File No. 1130-25
Form C27-6934-0

IBM Systems Reference Library

IBM 1130/2250 Graphic Subroutine Package

Preliminary Specifications

This publication contains information that enables a FORTRAN programmer to write programs for use with the IBM 2250 Display Unit Model 4 in association with the IBM 1130 Disk Monitor System, Version 2. It also contains supplementary information that describes how the same facilities may be used in a program written in Assembler language.

The Graphic Subroutine Package (GSP) consists of subroutines for displaying characters or graphic forms on the 2250 screen and for controlling communication between the program and the 2250 operator. The subroutines may be called from a program written in the 1130 Basic FORTRAN IV language or from a program written in 1130 Assembler language.

It is assumed that the FORTRAN user of this publication has had experience with the IBM 1130 Disk Monitor System and 1130 Basic FORTRAN IV language. It is assumed that the Assembler-language user of this publication is experienced in both the 1130 FORTRAN IV and Assembler languages.

PREFACE

This publication describes subroutines that can be called from a FORTRAN program to generate characters and graphic forms and to display them on the screen of an IBM 2250 Display Unit Model 4 attached to an IBM 1130 Computing System. The displays may consist of charts, circles, arcs, rectangles, or numerous other configurations.

This publication is divided into five major sections, a series of appendixes, and an index.

The first section introduces the reader to the Graphic Subroutine Package (GSP) and the 2250 model 4. It also discusses the format used in the detailed descriptions of each of the graphic subroutines.

The second section presents an overall view of how the GSP may be used to create, modify, and display an image. It also defines terminology used in the publication.

The third section provides detailed descriptions of all the graphic subroutines, except subroutines associated with communication between the 2250 operator and the program.

The fourth section describes subroutines associated with communication between the 2250 operator and the program.

The fifth section describes program errors and a subroutine that allows the programmer to check whether a subroutine was able to perform the desired operation.

Appendixes provide additional reference material, including a sample FORTRAN program, instructions for using GSP subroutines in an Assembler-language program, and the 2250 Assembler orders.

Before using this publication, the reader must be familiar with the publication IBM 1130/1800 Basic FORTRAN IV Language, Form C26-3715. He should also be familiar with the following publications:

IBM 1130 Disk Monitor System, Version 2; System Introduction, Form C26-3709

IBM 1130 Component Description: IBM 2250 Display Unit Model 4, Form A27-2723

In addition, the Assembler-language programmer should be familiar with the publication IBM 1130 Assembler Language, Form C26-5927.

First Edition (August 1967)

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or in Technical Newsletters.

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to IBM Corporation, Programming Publications, Department 637, Neighborhood Road, Kingston, New York 12401

© International Business Machines Corporation 1967

FIGURES

TABLES

The set of 1130/2250 subroutines available for use by the FORTRAN programmer is called the IBM 1130/2250 Graphic Subroutine Package (GSP). This package is not an extension of the FORTRAN IV language, but may be used in conjunction with it. The GSP allows a FORTRAN programmer to create displays on an IBM 2250 Display Unit Model 4 attached to an IBM 1130 Computing System having 16,384 words of core storage and one disk. These displays can be constructed of lines, points, and characters. The execution of each subroutine is requested by a CALL statement.

A program that uses the GSP includes calls to GSP subroutines in a sequence that causes displays to be produced and provides communication between the 2250 operator and the program. Such a program is described in detail in "Structure of a GSP Program."

Displays are produced on the basis of control information and data supplied by the programmer in the call to each GSP subroutine. This control information and data define what is to be displayed and where it is to be displayed. The input data can be provided in main storage arrays. As supplied by the programmer, this data is meaningful to the GSP, but not to the 2250. Therefore, the GSP converts this data to a format meaningful to the 2250.

THE 2250 DISPLAY UNIT

The 2250 model 4 is a cathode-ray tube display console with a light pen and optional features that enable data to be entered directly from the 2250 into the computer. Images are displayed on the cathode-ray tube under program control. The optional features are an alphameric keyboard and a programmed function keyboard.

The screen (12 in. x 12 in.) is defined by a matrix (1024 x 1024) of addressable point positions. The distance between any two adjacent points is a raster unit. Each point, or screen location, is specified by a pair of x- and y-coordinates in the range 0 to 1023. The origin begins at the lower-left corner of the screen (0,0) and extends horizontally to the right along the x-axis and vertically toward the top along the y-axis, so that the coordinates at the upper-right corner represent the maximum boundary of the screen (1023,1023).

A display is created when an electron beam in the 2250 moves over the screen as directed by graphic orders. The orders may designate that the beam is to be unblanked or blanked while it is being moved. Images are displayed only if the beam is moved in the unblanked mode. The images fade rapidly, however, and must be continually regenerated to make the display appear steady and stationary. Display regeneration is accomplished by repeating the execution of the orders thirty to forty times each second. The actual regeneration rate is a function of the amount of data displayed.

Alphameric characters are displayed by drawing them with a series of strokes. The IBM-supplied character generation subroutine produces a standard set of characters of one orientation (vertical), and upper or lower case (see Appendix F). The characters are capable of being produced in either of two sizes (basic or large). The 2250 model 4 provides the capability of subscripting and superscripting characters. The dimensions of characters produced and their spacing are listed in Appendix G.

For a more detailed discussion of the 2250 model 4 (including descriptions of the light pen, the alphameric keyboard, and the programmed function keyboard), refer to the publication IBM 1130 Component Description: IBM 2250 Display Unit Model 4, Form A27-2723.

MACHINE REQUIREMENTS

Programs using the GSP may be executed on any IBM 1130 Computing System having 16,384 words of core storage, one disk, and an attached 2250 model 4. Although the programmer is not required to use the total Graphic Subroutine Package, it is anticipated that an effective graphics application will require at least 16K words of 1130 core storage.

PROGRAMMING REQUIREMENTS

The GSP provides the 1130 programmer with a series of subroutines to aid him in programming graphics applications using the IBM 2250 Display Unit Model 4. The GSP is used only in association with the IBM 1130 Disk Monitor System Version 2.

An 1130 program using the GSP must use the *G2250 control card (see Appendix B), which causes the GSP support package

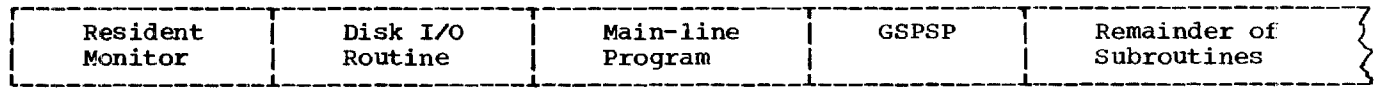


Figure 1. Required Core-Storage Layout for Programs Using the GSP

(GSPSP) to be loaded immediately following the main-line program (see Figure 1). Since the GSPSP must completely reside below core location 8192, the number of words occupied by the Resident Monitor, the main-line program, and the GSPSP must not exceed 8192. For more information about this requirement, as well as information about program links, see Appendix B.

LANGUAGE COMPATIBILITY

The facilities provided by the GSP can be used by programs written in the 1130 Basic FORTRAN IV language or in the 1130 Assembler language. The body of this publication is directed to the use of the GSP by programs written in the FORTRAN IV language. The use of the GSP by programs written in the Assembler language is described in Appendix D.

ERROR DETECTION

Input/output errors that occur while the program is communicating with the 2250 are handled automatically by standard IBM error-handling routines. These routines diagnose the errors and apply error-recovery procedures. If an error cannot be corrected, error information is made available to the program and/or the 2250 operator.

Invalid arguments in the CALL statement for a GSP subroutine result in the GSP making codes available to the program that indicate what type of error occurred. For further information see the section "Error Handling."

FORMAT OF SUBROUTINE DESCRIPTIONS

In this publication, the detailed description of each GSP subroutine is organized as follows:

1. Name -- The mnemonic entry name of the subroutine and a phrase explaining the meaning of that mnemonic.

2. Function -- A brief summary of what the subroutine accomplishes.

3. Format Description -- An illustration of the statement for calling the subroutine.

4. Argument Descriptions -- Detailed information about writing each argument.

5. Cautions -- Any special restrictions on the use of the subroutine.

6. Programming Notes -- Tutorial material describing the use of the subroutine and the results it accomplishes. Detailed information such as the use of paired subroutines is covered here.

7. Errors -- Logical errors that can be detected during the processing of a program are noted.

Items 1 through 4 are included, without headings, in all subroutine descriptions (unless a subroutine has no arguments). The remaining items are included when applicable, with appropriate headings.

This section provides an overall view of the use of the GSP to create and modify images and defines terminology used later in this publication.

GRAPHIC ELEMENTS

The programmer is concerned with creating, modifying, and displaying an image. He does this by defining graphic elements, the constituent parts of the image, in the form of data input to the GSP subroutines, which then convert the data to a form acceptable to the 2250. Element is a generic term meaning any part of an image. A specific type of element, one that serves a particular function, is denoted by the term entity preceded by another term identifying the type. An entity is either all the graphic data making up an image (an image entity) or is a discrete portion of that data. Following are the types of elements and their definitions; the symbols representing the types of elements will be of use in describing the structure of image entities.

{ } = Image entity, a collection of elements that can be displayed.

[ ] = Controlled entity, an element that is given controllable attributes of:

- visibility
- detectablilty

( ) = Uncontrolled entity, an element that is given fixed attributes of:

- visibility (always visible)
- non-detect

|S| = Subroutine entity, an element that may be displayed at different screen locations in the same display.

|T| = Tracking entity, an element used to collect the coordinates of points identified by the 2250 operator by means of the light pen.

< > = Singular entity, a generic term for an element that cannot contain other elements. Singular entities are specifically defined as follows:

<+> = Origin entity, an element that establishes the position of the element that follows it.

<•> = LPC (Line, Point, Character) entity, a collection of lines, points, or characters.

<↑> = Linkage entity, a request for linkage to a subroutine or tracking entity.

<M> = Message entity, an element used to collect characters entered through the alphameric keyboard.

NESTING

In creating an image entity, the programmer not only defines its elements, but also its structure. The GSP provides the facility of nesting (embedding) elements within other elements, subject to certain rules. The nested elements may be named, and depending on the name provided, the program can refer to one element or an entire set of nested elements. The nesting of elements is similar in concept to the nesting of FORTRAN DO statements.

A nested element must be completely within the nesting element; overlapping of elements is not permitted. The rules for nesting elements can be stated as follows:

- An image entity can contain any elements except image and tracking entities.
- A controlled entity can contain any elements except image, controlled, and tracking entities.
- An uncontrolled entity can contain any elements except image and tracking entities.
- A subroutine entity can contain any elements except image, controlled, and tracking entities.
- A tracking entity cannot contain any other elements.
- A singular entity cannot contain any other elements.

Following are two examples of image entity structures using the symbols defined above. The first image entity comprises an origin and an LPC entity. This simple structure might represent, say, positioning information (origin) and text (LPC) for displaying a message.

The second image entity comprises an uncontrolled and a controlled entity, each containing an origin and an LPC entity. The first origin and LPC entities might represent a message to the 2250 operator to point at a portion of the display with the light pen. The controlled entity would represent the portion of the display that can be detected by the light pen (detectability attribute). In addition, this portion of the display can be made to disappear without deleting the element from the program (visibility attribute).

```
{<+><•>}
{(<+><•>)[<+><•>]}
```

THE IMAGE ENTITY { }

An image entity is the only element that can be directly referenced for an actual display. All other types of elements which it is desired to display must therefore be nested in an image entity. (The subroutine and tracking entities are exceptions; see below.) Conversely, an image entity cannot be nested within any other element, including another image entity. The content of an image entity depends on the definition of its constituent elements: controlled, uncontrolled, subroutine, and singular entities.

THE CONTROLLED ENTITY [ ]

A controlled entity is an element with two controllable attributes -- visibility and detectability.

The visibility attribute gives the program the option of either displaying or not displaying a controlled entity without affecting its definition as part of an image entity. The attribute is initially set to display when the controlled entity is defined. It can be set to non-display by means of the Set Controlled Entity Attributes (SATRB) subroutine at any point in the program after the controlled entity has been completely defined. If the visibility attribute is to be set to non-display, the element immediately following the controlled entity must be absolutely positioned or it will be repositioned on the screen.

The detectability attribute is the property of selective identification, by means of the light pen, of the controlled entity. Two options are available, detect and non-detect.

The standard option established when the controlled entity is defined is detect. If detection is not desired, the attribute can be set to non-detect by means of the SATRB subroutine.

With detectability set to detect (and when light pen attentions are enabled; see "Attention Handling"), the closing of the light pen switch when a detect is made causes a light pen attention to be accepted for the detected controlled entity, and the attention data is made available upon request. The attention data will include the correlation value (identifyer; see below) of the controlled entity, the correlation value of the innermost named element detected, the character detected, or the x- and y-coordinates of the point or end point of the line detected. If the light pen is withdrawn before the switch is closed, no attention occurs.

THE UNCONTROLLED ENTITY ( )

The uncontrolled entity has fixed attributes of visibility (always visible) and non-detect. It can be a collection of controlled, uncontrolled, subroutine, or singular entities. Any controlled entities nested within an uncontrolled entity still possess their controllable attributes. Conversely, any uncontrolled entities nested within a controlled entity assume the visibility and detectability attributes of the controlled entity.

THE SUBROUTINE ENTITY |S|

A subroutine entity is similar in concept to a program subroutine. Its principal use is to display a graphic form in more than one area of the 2250 screen in the same display.

The only entry into a subroutine entity is by means of a linkage entity. It is not therefore necessary that a subroutine entity be defined within an image entity. This is one of the two exceptions to the rule that all elements which are to be displayed must be within an image entity (the other exception being the tracking entity). The linkage to a subroutine entity must be deleted or made inactive (performs no-operation) if the subroutine entity is to be deleted.

A subroutine entity should contain only incremental data. It should not contain any controlled entities or absolute origin entities.

The following image entity structure will result in the simultaneous displaying of three copies of the subroutine at three different places on the 2250 screen (assuming that all three origin entities are to different positions).

|S|{<+><↑><+><↑><+><↑>}

THE TRACKING ENTITY |T|

A tracking entity is like a singular entity in that it cannot have other elements nested within it. It is also a special form of subroutine entity, with the following features of a subroutine entity:

- The generated graphic data may be incremental and therefore may be relocated on the surface of the screen. This is dependent on the output graphic mode defined for the generation control area (GCA; see "Initializing a Generation Control Area").
- It may be requested by means of a linkage entity and therefore may appear at more than one screen location in the same display.
- It must follow the rule for deletion of a subroutine entity: linkage to the tracking entity should be deleted or made inactive before deletion of the tracking entity.

The tracking entity differs from the subroutine entity in that it may be in absolute or optimized output graphic mode (see "Initializing a Generation Control Area"). In these cases it will be displayed in a fixed location on the screen. It also differs from the subroutine entity in that it does not have to be requested by a linkage entity in order to be displayed.

THE SINGULAR ENTITY < >

The term "singular entity" is used as a general name for elements that by their nature are complete and cannot contain other elements within them. The term comprises the following: origin entity, LPC entity, linkage entity, and message entity.

The Origin Entity <+>

An origin entity is an element that establishes the position of the element that follows it. The positioning will be either absolute or incremental as determined by the output graphic mode defined in the appropriate generation control area (GCA; see "Initializing a Generation Control Area"). If the GCA output graphic mode is absolute, the element will be positioned to a specified point. If the output graphic mode is incremental, the positioning will be relative to the position of the previous element. If the output graphic mode is optimized, positioning will be either absolute or incremental, as determined internally by the GSP.

The LPC Entity <•>

An LPC (Line, Point, Character) entity is an element containing graphic data for the combination of lines, points, and characters that determine the appearance of the displayed image entity. It is therefore the principal building block of a typical graphic image entity.

The Linkage Entity <↑>

A linkage entity is a request for the inclusion of a subroutine entity or a tracking entity. A linkage entity can be established as either active or inactive (performs no operation); it can be named, if desired, and updated later (e.g., changed from inactive to active).

The Message Entity <M>

A message entity is an element that consists of alphameric data. The data can be generated by the program for display on the screen as a message to the 2250 Operator, or it can be entered from the alphameric keyboard as a message from the operator. A message entity has attributes of:

- Character size (basic or large)
- Length (number of characters)

CORRELATION VALUE

In defining the elements of an image entity to the GSP, the programmer may require the facility of referring to a particular element at some later point in the program. This facility is provided by the correlation value argument specified when the element is defined. At any point in the program after the element is named (i.e., assigned a correlation value), this name may be used as an argument to refer to that element.

Of the elements defined above, the following must be named:

- Image entity
- Controlled entity
- Uncontrolled entity
- Subroutine entity
- Tracking entity
- Message entity

The origin and linkage entities may be named at the programmer's option. An LPC entity cannot be named.

Correlation values should be unique within a GSP program.

STRUCTURE OF A GSP PROGRAM

Preparation of a program that uses the GSP requires the following basic steps:

1. Initialization and definition.
   - Initializing the GSP
   - Initializing an Image Construction Area
   - Initializing a Generation Control Area
2. Creating, modifying, and displaying the image entity.
3. Establishing communication between the GSP program and the 2250 operator.

INITIALIZATION AND DEFINITION

Before the programmer can begin to create image entities with the image generation subroutines, certain initializing steps must be performed: activating the GSP, defining environmental characteristics, defining and initializing data and control areas, establishing precision specifications, identifying I/O devices, etc. These initializing procedures are performed by the GSPIN, ICAIN, and GCAIN subroutines.

Initializing the Graphic Subroutine Package

In order to activate the GSP and to establish its environmental characteristics, the first GSP subroutine to be called must be GSPIN, the Graphic Subroutine Package Initialization subroutine. Based on the arguments provided by the programmer, GSPIN sets up the control structure needed by almost all other GSP subroutines:

1. Precision specifications, to define the format of integer and real input data.
2. Error return variables, to which the GSP returns indications of any errors that have been detected. There are two of these variables: one holds an error indication for the most recent call to a GSP subroutine, and the other a cumulative indication of errors for all previous calls. The error variables are set by most GSP subroutines.
3. Unit identification, to establish correspondence between the device address and its logical unit number. The programmer defines the device address and associates a logical unit number with it. Once this association has been established by means of the GSPIN subroutine, the programmer uses the logical unit number in calls to other GSP subroutines to refer to the device.

Initializing an Image Construction Area

Before the programmer can begin creating image entities by means of the image generation subroutines, he must provide and initialize an image construction area (ICA) that will contain the image entity or entities that he is to create. In this area the input data, converted to 2250 format, is stored for subsequent display.

The ICA is also a control area. It contains, in addition to the image entity, information needed to define the structure of the entity.

The ICA is initialized by a call to ICAIN, the Image Construction Area Initialization subroutine. More than one ICA can be used in a GSP program, but only one at a time can be active. To change ICAs, the programmer must issue another call to ICAIN.

The ICAIN subroutine does the following:

1. Assigns to the ICA an identifying correlation value.
2. Establishes the limits of the ICA, thereby specifying its length.
3. Makes the ICA referred to the active ICA.
4. Provides an option for redefining an ICA (re-activating an ICA without changing its contents).

Initializing a Generation Control Area

Another area that must be provided and initialized before the programmer can begin creating image entities is the generation control area (GCA). The GCA contains information needed by image generation subroutines to properly generate elements of the image entity. This information is instrumental in performing the following functions:

- Mapping (scaling and scissoring) of input data to raster units representing the entire 2250 screen or a smaller area of the screen (grid).
- Conversion of graphic data (i.e., x-

and y-coordinates in raster units and character data in 2250 format) to integer, real, or character arrays in program format which can be used by non-graphic FORTRAN statements.

- Selection of input data from arrays by means of an index factor, rather than sequentially.
- Specification of successive x- and y-coordinates by means of an increment value instead of requiring an input data array to contain them.
- Specification of real or integer, and absolute or incremental input data mode; and absolute, incremental, or optimized graphic output mode (see definitions of absolute, incremental, and optimized below).

More than one GCA may be defined for a program using the GSP. For example, different grid areas of the 2250 screen may be designated, different scaling factors (i.e., data range to grid area) may be specified, etc. However, each GCA must be fully defined before it is referred to by the subroutines concerned.

The GCA is a real array with 21 elements if the program uses standard precision or with 14 elements if the program uses non-standard EXTENDED PRECISION.

Since the data within the GCA is intended to be used only by GSP subroutines, it must not be referred to in any way other than as an argument in GSP subroutine calls.

A GCA can be fully initialized with standard values (see below) by means of the GCAIN subroutine. Modification of the area with other values is effected by means of six supplementary subroutines, each of which alters only a particular portion of the GCA. The GCA can therefore be defined by GCAIN alone, by a combination of GCAIN and supplementary subroutines, or by all six supplementary subroutines without GCAIN.

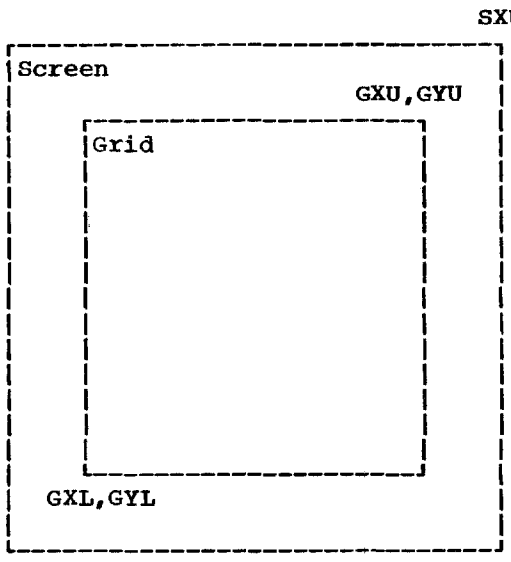
The GCA definition subroutines are:

GCAIN - Generation Control Area Initialization
SSCAL - Set Scaling Information (Screen, Grid, Data)
SSCIS - Set Scissoring Option
SINDX - Set Index Values
SINCR - Set Increment Values
SDATM - Set Input Data Mode
SGRAM - Set Output Graphic Mode

There may be, at some point in a program, several GCAs that have been defined, and depending on the options desired, the appropriate GCA can be used in a call to an image generation subroutine. This is unlike the ICA where only one is active at a time. Any and all GCAs are active.

THE SCREEN LIMITS: The 2250 screen contains a square matrix of 1024 x 1024 addressable points, and the device treats the screen coordinates as if they were all in the first quadrant, i.e., from (X=0, Y=0) to (X=1023, Y=1023). The programmer need not be concerned with these device coordinates. He may assign values in any units to the lower-left corner and the upper-right corner (see Figure 2).

THE GRID LIMITS: The programmer can also define a grid, a rectangular portion of the screen within which the elements are to be placed. The same units must be used for these as for the screen limits. He defines the lower-left corner and the upper-right corner of the grid. The grid size must be less than or equal to the screen size (see Figure 2).



SXL,SYL = x- and y-coordinates of lower-left corner of the screen
SXU,SYU = x- and y-coordinates of upper-right corner of the screen
GXL,GYL = x- and y-coordinates of lower-left corner of the grid
GXU,GYU = x- and y-coordinates of upper-right corner of the grid

The following are requirements for establishing screen and grid limits:

SXU ≥ GXU > GXL ≥ SXL
SYU ≥ GYU > GYL ≥ SYL

Figure 2. Screen and Grid Limits

THE DATA LIMITS (SCALING): The programmer specifies the range of his data, in programmer-defined units, that is to be mapped (scaled) into the grid. Any data which maps to a point outside the grid may be scissored (see Figure 3).

SCISSORING OPTION: It is possible for user data to map to a point outside the grid. The programmer is provided two choices: either nothing outside the grid will be displayed (i.e., lines are scissored at the grid limits) or everything outside the grid is to be displayed (i.e., lines are scissored at the screen limits). Figure 4 illustrates the two options.

INDEX VALUES: The programmer may provide values to specify that every "nth" element of an input data array, used by image generation subroutines, is to be used, where n≥1.

INCREMENT VALUES: These values provide the capability of generating successive x- and y-coordinates from some known starting point. This eliminates the need for one or more input data arrays for an image generation subroutine.

INPUT DATA MODE: This value specifies the type of data that the x- and y-arrays

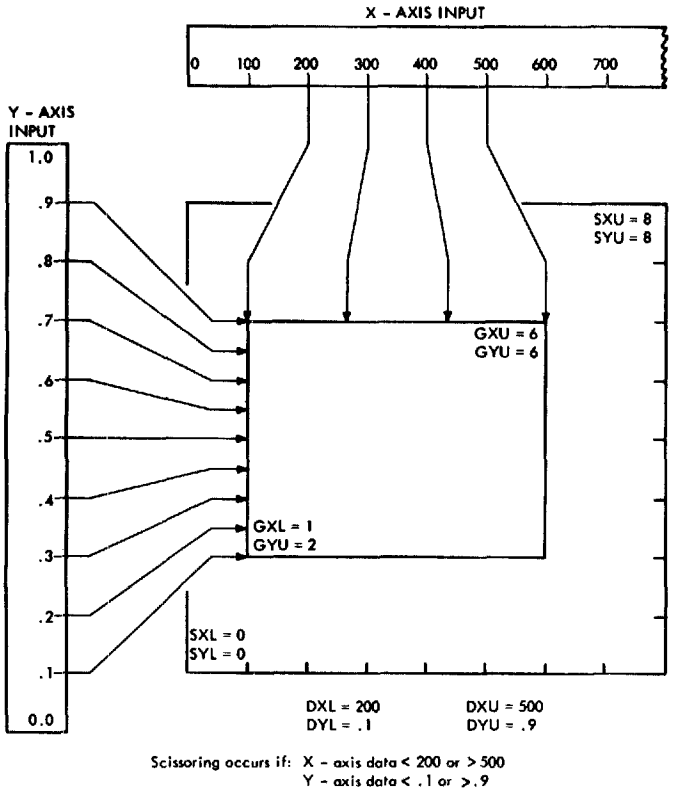
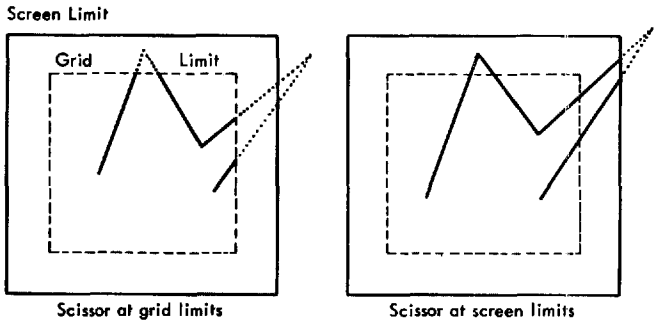


Figure 3. Mapping and Scaling



NOTES:

Dotted lines indicate points/vectors that would not be generated.

Solid lines indicate points/vectors that would be generated.

Figure 4. Scissoring Option

contain. The types are: real absolute, real incremental, integer absolute, integer incremental.

Absolute data are the actual coordinates (in programmer units) where an element is to be displayed on the screen. Incremental data are coordinate values that are displacements from the coordinate values that immediately precede them.

OUTPUT GRAPHIC MODE: Output data produced by the image generation subroutines (i.e. the elements in 2250 format) may be in absolute, incremental, or optimized form. Optimized data is input data that has been transformed into that 2250 format which requires the least amount of core storage. This form usually consists of a combination of absolute and incremental data.

STANDARD VALUES: The standard values established by a call to GCAIN are:

screen limits: 0,0 to 1023,1023 (modify by SSCAL)

grid limits: 0,0 to 1023,1023 (modify by SSCAL)

data limits: 0,0 to 1023,1023 (modify by SSCAL)

scissoring option: scissor to grid (modify by SSCIS)

index values: all = 1 (modify by SINDX)

increment values: all = 0 (modify by SINCR)

input data mode: real absolute (modify by SDATM)

output graphic mode: optimized (modify by SGRAM)

CREATING, MODIFYING, AND DISPLAYING AN IMAGE ENTITY

Once the communication paths to the GSP have been established (i.e., GSPIN has been called, and an ICA and at least one GCA have been defined), the programmer can begin defining an image entity. An image entity has three major characteristics:

1. It consists of one or more elements.
2. It has a structure.
3. It can be displayed.

The structure of an image entity is defined by a series of calls to the image management subroutines, while the elements of an image entity are defined by a series of calls to the image generation subroutines. In general, an element is defined by a call to an image management subroutine to define its beginning, one or more calls to the image generation subroutines to define its content (points, characters, lines), and a call to an image management subroutine to define its end. For some elements, however, only one subroutine is used to completely define the element. This is true of the message, linkage, and origin entities.

The Image Management Subroutines

The image management subroutines define and control the structure of an image entity. The following is a list of all the image management subroutines and their functions:

Mnemonic	Name & Function
GSPIN	Graphic Subroutine Package Initialization - specifies control information required by other GSP subroutines.
ICAIN	Image Construction Area Initialization - initializes an area for use by the GSP to contain the image entity and its control information.
BELMT	Begin Element - defines the beginning of a controlled, uncontrolled, subroutine, or image entity.
EELMT	End Element - defines the end of an element whose beginning was specified by BELMT, UELMT, or XELMT.
UELMT	Update Element - identifies an element whose content is to be completely modified by subsequent calls to image generation subroutines.
XELMT	Extend Element - identifies an element whose content is to be extended by subsequent calls to image generation subroutines.
DELMT	Delete Element - identifies an element that is to be completely deleted from the image entity.
SATRB	Set Controlled Entity Attributes - resets one or more of the attributes of a controlled entity.
EXEC	Execute Display - performs the processing required to display an image entity.
TMDSP	Terminate Display - performs the processing required to stop the display of an image entity.
GSPTM	GSP Termination - provides an optional dump of core storage and terminates use of the GSP.

The Image Generation Subroutines

The image generation subroutines define the content of an element by converting the input data to 2250 format. The following is a list of all the image generation subroutines and their functions:

Mnemonic	Name & Function
GCAIN	Generation Control Area Initialization - initializes a GCA with standard values.
SSCAL	Set Scaling Information - sets or resets the scaling information in the GCA.
SSCIS	Set Scissoring Option - sets or resets the scissoring to grid boundaries or screen boundaries.
SINDX	Set Index Values - sets or resets the input array indexing values in the GCA.
SINCR	Set Increment Values - sets or resets the increment values in the GCA.
SDATM	Set Input Data Mode - sets or resets the input data type to absolute or incremental, real or integer.
SGRAM	Set Output Graphic Mode - sets or resets the output data type to absolute, incremental, or optimized.
MVPOS	Move Element to a Position - establishes the starting coordinates for the next element.
IDPOS	Indicate Element Position - indicates the starting coordinates of the next element for proper scissoring after use of a linkage entity or when modifying an existing element.
PLINE	Plot Lines - converts input data to 2250 format for plotting lines.
PPNT	Plot Points - converts input data to 2250 format for plotting points.

PTEXT	Plot Text - converts input data to 2250 format for plotting characters.
PSGMT	Plot Line Segments - converts input data to 2250 format for plotting line segments.
LKSUB	Linkage to a Subroutine - establishes linkage to a subroutine or tracking entity.
PGRID	Plot Grid Outline - generates 2250 format data to plot the rectangular outline of a grid.
PCOPY	Plot Copy of an Element - generates a copy of an element at another place in an image or subroutine entity.

Using the Subroutines

For the purpose of illustration, assume that it is desired to display a circle using a series of points. The following might be the series of GSP subroutines called:

Subroutine Called	Reason
GSPIN	GSP Initialization.
ICAIN	ICA Initialization.
GCAIN	GCA Initialization.
BELMT	Define the beginning of an image entity.
PPNT	Generate the series of points in 2250 format for the circle.
EELMT	Define the end of the image entity.
EXEC	Display the image entity on the 2250.

The above series of calls defines a single element, the image entity, whose content is a circle. Symbolically, the image entity can be represented as follows:

{<•>}

If it were desired to display a circle using a series of lines, the following might be the GSP subroutines called:

Subroutine Called	Reason
GSPIN, ICAIN, GCAIN	Initialization.
BELMT	Define the beginning of an image entity.
MVPOS	Position the beam to the first point on the circle from which the lines will begin.
PLINE	Generate the series of lines in 2250 format for the circle.
EELMT, EXEC	

Symbolically, the image entity is represented as:

{<+><•>}

Note: If points were used to construct the circle, no origin entity would be needed.

Now suppose the programmer wishes to change the display from that of a circle to that of a square. The following series of subroutines might be used:

Subroutine Called	Reason
TMDSP	Stop the display for updating.
UELMT	Identify the element to be changed.
MVPOS	Define the starting position of the square.
PLINE	Generate the four sides of the square in 2250 format.
EELMT	Define the end of the new contents of the element.
EXEC	Display the image entity, now a square.

Note that it is necessary to stop the display (TMDSP) before updating can be accomplished.

It is now desired to extend the defined element, the square, to include another square. The following series of subroutines might be called:

Subroutine Called	Reason
TMDSP	Stop the display for updating.
XELMT	Define the element to which data is being added.
MVPOS	Define the starting position of the second square.
PLINE	Generate the lines for the second square.
EELMT, EXEC	

The display now consists of two squares, and the image entity can be symbolically represented as follows:

{<+><•><+><•>}

The following series of calls will define the same displayable image entity, but with the two squares as an uncontrolled entity.

Subroutine Called	Reason
GSPIN, ICAIN, GCAIN	Initialization.
BELMT	Define the beginning of the image entity.

BELMT	Define the beginning of two squares as an uncontrolled entity.
MVPOS	Position to a corner of the first square.
PLINE	Generate the first square.
MVPOS	Position to a corner of the second square.
PLINE	Generate the second square.
EELMT	Define the end of the two squares.
EELMT	Define the end of the image entity.
EXEC	Display the image entity.

Now the image entity is represented as:

{(<+><•><+><•>)}

indicating that an uncontrolled entity has been defined within the image entity.

The following adds another uncontrolled entity, a triangle, to the image entity.

Subroutine Called	Reason
TMDSP	Stop display for updating.
XELMT	Indicate that the image entity is being extended.
BELMT	Define the beginning of a new uncontrolled entity.
MVPOS	Define the starting position of the triangle.
PLINE	Generate the sides of the triangle.
EELMT	Referring to the image entity defines the end of both the triangle element and the update of the extended image entity.
EXEC	Display the new image entity consisting of the two squares and the triangle.

The structure of the above image entity is represented as:

{(<+><•><+><•>)(<+><•>)}

If it is wished to delete the uncontrolled entity containing the two squares from the image entity, the following series of calls can be used:

Subroutine Called	Reason
TMDSP	Stop the display for updating.
DELMT	Using the name of the element containing the two squares causes it to be removed from the image entity.
EXEC	The display will now contain only the triangle, and the structure of the image entity is: {(<+><•>)}

In order to label the triangle with the word TRIANGLE, the following procedure might be used:

Subroutine Called	Reason
TMDSP	Stop the display for updating.
XELMT	Identify the triangle uncontrolled entity for extended update.
MVPOS	Position to the point at which the first character is to appear.
PTEXT	Generate the characters in 2250 format for the word TRIANGLE.
EELMT	Define the end of this element.
EXEC	Display the triangle, which is now labeled with the word TRIANGLE.

This final image entity can be represented thusly:

{(<+><•><+><•>)}

COMMUNICATION BETWEEN A GSP PROGRAM AND THE 2250 OPERATOR

The IBM 2250 Display Unit is a powerful tool for two reasons:

1. It can quickly display a large amount of data.
2. The 2250 operator can communicate with and guide the running program by using the light pen or keyboards.

Up to this point, this section has been concerned mostly with the first item. The following paragraphs deal with four major areas in operator/program communication:

1. Attention handling
2. Using the alphameric keyboard
3. Using the light pen
4. Using the programmed function keyboard.

Attention Handling

An attention is the name of the signal generated when the 2250 operator depresses a programmed function key or an alphameric keyboard key, or points at a displayed image entity with the light pen and closes the light pen switch. The keyboards and the light pen are therefore attention sources. The attention causes the central

processing unit to interrupt and enter a GSP routine whose function is to gather all the available information about the attention. This attention information is made available to the program upon request.

The attention-handling subroutines of the GSP are used to establish which attentions are to be processed and which are to be ignored, and to obtain attention information. The following are the attention-handling subroutines and their functions:

Mnemonic	Name & Function
SATNS	Set Attention Status - enables the sources from which the program will process attentions.
RQATN	Request Attention Information - requests that available attention information be formatted and provided to the program.
ROCOR	Return Outer Correlation Value - requests the correlation value of an element within which another element is nested.

There are four attention sources available to the GSP programmer:

1. Light pen
2. END key
3. Alphameric keyboard (except END and CANCEL keys)
4. Programmed function keyboard

The attention information returned is different for the different sources. The details appear in the discussion of the RQATN subroutine.

The programmer determines which attention sources are meaningful to his program and enables those. All others become disabled. He may at any point in the program change enabled sources.

When the program reaches a point that requires the 2250 operator to specify or have specified some information by means of an attention, the program issues a call to RQATN. RQATN formats any available attention information and places it into a programmer-defined array. The program can then determine if an attention has occurred and, if so, whether it is the correct one. If no attention has occurred, the program might be able to continue processing that is unrelated to the attention. If an attention is required before further processing can take place, the program could issue a PAUSE statement followed by a GO TO statement to the RQATN call.

If an attention has occurred and it is the correct one, the program continues processing. If the attention is incorrect, the program could simply ignore it and re-execute RQATN or notify the 2250 operator that his attention is inappropriate.

At some point in the program it may be desirable to perform some unique function which requires the 2250 operator to respond with a series of attentions. However, the present attention status is to be re-established following performance of this function. This may be accomplished by saving the argument used in the last call to SATNS. Another call to SATNS with a different argument establishes the new attention-handling environment (i.e., enables a different set of sources). The unique function can then be performed. Another call to SATNS with the saved argument re-establishes the attention-handling environment as it existed just prior to performing the unique function.

In the previous examples of circles, squares, and triangles, a series of attentions could have been used to signal the program to change the displays. The following might have been the series of calls issued:

Subroutine Called	Reason
GSPIN, ICAIN, GCAIN	Initialization.
BELMT	Define the beginning of the image entity.
BELMT, MVPOS, PLINE, EELMT	Define a circle as a controlled entity in order to be light-pen detectable and define the end of the image entity.
SATNS	Enable light pen attentions.
EXEC	Display the image entity, which can be represented as: {[<+><•>]}
RQATN	Wait for the light pen attention on the circle, which indicates that the next image entity is to be displayed.
TMDSP, UELMT, MVPOS, PLINE, EELMT, EXEC	Change the display to the square. The image entity is still represented as: {[<+><•>]}
RQATN	Wait for the second light pen attention.
TMDSP, XELMT, MVPOS, PLINE, EELMT, EXEC	Change the display to that of two squares. The image entity is now represented as: {[<+><•><+><•>]}
SATNS	Enable the programmed function keyboard, disabling the light pen.
RQATN	Wait for a programmed function keyboard attention.

TMDSP, XELMT, MVPOS, PLINE, MVPOS, PTEXT, EELMT	Add the triangle and text controlled entity to the image entity.
SATNS	Enable the light pen, disabling the programmed function keyboard.
EXEC	Display the squares and triangle. The image entity is now represented as: {[<+><•><+><•>][<+><•>]}
RQATN	Wait for the next light pen attention.
DELMT	Delete the element on which the attention occurred.
EXEC	Display the remaining element. Depending on which element was deleted, the image entity can be represented as either {[<+><•><+><•>]} or {[<+><•>]}
RQATN	Wait for the next light pen attention.
TMDSP	Terminate the display.
GSPTM	Terminate program execution.

Using the Alphameric Keyboard

The alphameric keyboard provides the means of communicating characters (alphabetic, numeric, and special) to the program. The characters could be codes or text, depending on their use by the program.

If the program is enabled for alphameric keyboard attentions and is not in message-collection mode (i.e., a call to ICURS is not in effect), the alphameric keyboard attentions are returned one at a time in the same fashion as programmed function keyboard attentions. The alphameric keyboard could be used as a programmed function keyboard is used. In message-collection mode, however, depressing a key on the alphameric keyboard causes a character to be placed into a message entity. In this mode, the program does not receive any alphameric keyboard attentions until a call to RCURS is issued, which terminates the message-collection mode.

The following is a list of the alphameric keyboard and message-collection subroutines and their functions:

Mnemonic	Name & Function
DFMSG	Define Message Entity - defines an element into which alphameric keyboard characters will be placed.
MSGIN	Message Entity Initialization - initializes a message entity with text.
ICURS	Insert Cursor - places a cursor into a message entity and establishes the message-collection mode.
RCURS	Remove Cursor - deletes the cursor and terminates the message-collection mode.
TLMSG	Translate Message Data - converts alphameric data in a message entity to EBCDIC format for manipulation by the FORTRAN program.

The programmer defines one or more message entities in his program and may initialize them with text, such as instructions to the 2250 operator. By issuing a call to ICURS, the program enters message-collection mode. The alphameric keyborad attentions are no longer passed to the program, if the alphameric keyboard was enabled, but are sent to the GSP message-collection routine. When the 2250 operator depresses a key, the character is placed into the message entity identified by the call to ICURS, and the cursor advances one position. The character appears on the screen. The 2250 operator can key in as many characters as the message entity can hold. Once the message entity is filled, all other characters keyed in replace the last character until the JUMP key is depressed. When the GSP message-collection routine receives a JUMP signal, the cursor is moved to the next available message entity, and the 2250 operator can continue keying in more characters.

The message entities are considered to be a closed circle; that is, if the JUMP function is executed often enough or if there is only one message entity and the JUMP key is depressed, the cursor returns to the message entity identified by the initial call to ICURS. The 2250 operator signals that he is finished by depressing the END key or by any other suitable attention. When the program recognizes the end-of-message attention (by means of RQATN), it then calls RCURS to terminate the message-collection mode, and TLMSG to convert the characters from 2250 format to EBCDIC for further manipulation.

Using the Light Pen

The light pen is used to communicate at least two things to the program:

1. Identification of a particular controlled entity.

2. The x- and y-coordinates of points or lines being defined.

The SATNS subroutine enables or disables light pen attentions. If enabled, certain information is collected and, upon request, made available on each light pen attention. In particular, the correlation value of the

element pointed at with the light pen is returned. The program can then manipulate or modify the element according to the application.

The GSP provides the following subroutines that facilitate communication by means of the light pen.

Mnemonic	Name & Function
LOCPN	Locate the Position of the Light Pen - used to identify the x- and y-coordinates of the point being defined by the light pen.
LOCND	Locate the Position of the Light Pen on No Detect - used to find the x- and y-coordinates of the light pen if no attention occurred on a controlled entity during the display of the image entity.
LCPOS	Locate a Position with the Tracking Symbol - used to identify the x- and y-coordinates of a particular location on the screen.
TRACK	Track the Position of the Light Pen - identifies a series of x- and y-coordinates and creates a tracking entity.
CTLTK	Control Light Pen Tracking - changes the initial light pen tracking status or the status set by a prior call to CTLTK.
DISTE	Disconnect Tracking Entity - disconnects the tracking entity from the temporary linkage in an image entity.
CVTTD	Convert Tracking Data - converts x- and y-coordinates from 2250 format to integer or real FORTRAN format.

When a programmer calls a light pen subroutine (except CVTTD and DISTE), the GSP services light pen attentions, even if the program is enabled for light pen attentions, until the 2250 operator or the program signals that the function is completed. (This is not true for every light pen attention; see the description of TRACK.) The completion is sometimes signalled by a light pen attention (for LOCPN and LOCND) and sometimes by a call to CTLTK (for LCPOS and TRACK). When the function is signalled complete, the enable/disable status of the light pen is restored to what it was just prior to the call.

LOCPN and LOCND cause a scanning pattern to be displayed until a light pen attention occurs. When the attention occurs, the scanning pattern disappears, and the data about the light pen attention may be accessed by calling RQATN.

LCPOS and TRACK display a special symbol called a tracking symbol. For LCPOS, the 2250 operator places the light pen on the tracking symbol, closes the switch, and as he moves the light pen across the screen, the tracking symbol moves. When the tracking symbol is at the point that the 2250 operator wishes the define to the program, he signals the program that he has defined the point. The program then calls CTLTK to terminate LCPOS, the tracking symbol disappears, and the defined x- and y-coordinates are returned to the program.

TRACK uses the tracking symbol in much the same way as LCPOS; but here a tracking entity is being defined as the operator identifies x- and y-coordinates with the light pen. A call to TRACK causes the tracking symbol to appear on the screen. The 2250 operator places the light pen on the tracking symbol, closes the switch, and begins defining x- and y-coordinates.

As the tracking symbol is moved by the 2250 operator, the points being defined are placed into a tracking entity, and the defined points or lines appear on the screen. The 2250 operator signals the program when he wishes to change from curve tracking (continuous sketching) to linear tracking (defining points or straight lines; i.e., rubber-banding) and vice versa. The 2250 operator must also signal the program that he has defined all the desired points and lines. The program then calls CTLTK to terminate light pen tracking by removing the tracking symbol from the screen. The program next calls CVTTD to convert the x- and y-coordinates from 2250 format to a FORTRAN data format for further manipulation. DISTE may be used to disconnect the defined tracking entity without deleting it. It may still be treated as any other defined tracking entity, but if no linkage to it has been created, the tracking entity disappears from the screen after the call to DISTE.

Using the Programmed Function Keyboard

The programmed function keyboard (PFKB) provides 32 pushbutton keys with 32 indicator lights and eight code-sensing contacts that sense notches punched in a plastic overlay sheet. The overlay code is a binary configuration giving values from 0 to 255, thereby allowing each key to issue 256 unique signals, or a total of 8192 signals for all 32 keys. The meaning of each signal is defined by the program.

The programmed function keyboard is either enabled or disabled. If enabled, any key depressed causes an attention, and the attention data is made available by calling RQATN. In addition to the key

number, the overlay number is also returned with the PFKB attention.

A subroutine is provided to enable the program to selectively light the programmed function keys. This subroutine, SPFKL, provides the programmer the means of indicating to the 2250 operator which of the PFKB keys are meaningful at any particular point in the program. SPFKL sets each of the 32 key lights independently, either on or off, as specified by the program.

This section describes in detail each GSP subroutine (except for subroutines for communicating with the 2250 operator). It begins with a discussion of those arguments used in many of the argument lists for the subroutines. Attention related, light pen, and keyboard subroutines are described in the section "Communicating With the 2250 Operator."

ARGUMENTS USED BY MANY OF THE SUBROUTINES

In order to avoid repeated descriptions of arguments that are common to several GSP subroutines, a general description of such arguments is given at this point. When an argument has an extension to the primary definition given here, the extended meaning is included in the description of the specific subroutine.

corrval
: represents any correlation value, used as an identifier of an element. It is used by the defining subroutine and other GSP subroutines as a common means of reference to the element identified by it. It may be used by the programmer to associate the element with the data used to generate the element.

  The "corrval" argument is a positive integer constant, integer variable, or integer arithmetic expression in the range 1 to 32767. The value should be unique for each element defined; otherwise, the reference will be taken to mean the first element, in the active ICA, having that correlation value.

gca
: represents any generation control area (GCA), which contains information needed by image generation subroutines to properly generate elements of the image entity. This information is initialized or defined by the following GSP subroutines:

  GCAIN - GCA Initialization (standard values)
  SSCAL - Set Scaling Information
  SSCIS - Set Scissoring Option
  SINDX - Set Index Values
  SINCR - Set Increment Values
  SDATM - Set Input Data Mode
  SGRAM - Set Output Graphic Mode

  The GCA is a real array with either 21 elements (standard precision) or 14 elements (extended precision). If the precision of the FORTRAN program is changed, the DIMENSION statement should be changed accordingly. (However, if the GCA dimension is 21, it will be valid for either precision.)

count
: is a positive integer constant, integer variable, or integer arithmetic expression specifying the number of elements to be accessed from input data arrays for generating lines, line segments, points, or characters. This count includes those lines, line segments, and points calculated but not displayed because of scissoring. The value of "count" must be equal to or less than the number of elements in the input array.

device
: is a positive integer constant, integer variable, or integer arithmetic expression, with values from 1 to 4, specifying the logical unit number assigned in the call to the GSPIN subroutine to a 2250 display unit.

textcode
: is a positive integer constant, integer variable, or integer arithmetic expression that defines the format of the alphameric data designated by the "text" argument in the PTEXT, MSGIN, and TLMSG subroutines. The "textcode" argument has the following values and meanings:

  1 = "text" is a real variable or array with either four characters (standard precision) or six characters (extended precision) in each element. The data corresponds to the FORTRAN A-type format.

  2 = "text" is an integer variable or array with two characters in each element. The data corresponds to the FORTRAN A-type format.

  3 = "text"
  is an integer variable or integer array; the data in each element consists of a positive integer value representing a single character, enabling the programmer to use character data in terms of their decimal equiv-

alents. The data corresponds to the FORTRAN I-type format (EBCDIC codes and their decimal equivalents are given in Appendix F).

Characters can be displayed aligned, subscripted, or superscripted, as defined by the integer value in each element of the "text" array. The normal (aligned) character range is 0-255. Any decimal value in this range plus 256 causes the corresponding character to be subscripted; any value plus 512 causes the character to be superscripted.

Examples:

129 + 0 = 129 aligned a

129 + 256 = 385 subscripted a

129 + 512 = 691 superscripted a

4 = "text" is an integer variable or integer array with two characters in each element. The data corresponds to the FORTRAN A-type format; the first character in each element is a control character for aligned, subscripted, or superscripted characters as follows:

Normal	Subscript	Superscript
0	1	2
or <	(	+

Examples:

0A or <A = aligned A

1A or (A = subscripted A

2A or +A = superscripted A

IMAGE MANAGEMENT SUBROUTINES

Image management subroutines are image "housekeeping" subroutines that are used to establish the environmental characteristics of a GSP program and to define and control the structure of an image or subroutine entity. They perform functions such as defining and identifying areas, establishing operating modes, etc. They do not normally generate display data. They can be considered as being analogous to FORTRAN Specification statements. The image management subroutines are listed below, and a detailed description of each, in the listed sequence, follows:

Category	Mnemonic	Name
Definition	GSPIN	Graphic Subroutine Package Initialization
	ICAIN	Image Construction Area Initialization
Identification	BELMT	Begin Element
	EELMT	End Element
Image Control	UELMT	Update Element
	XELMT	Extend Element
	DELMT	Delete Element
	SATRB	Set Controlled Entity Attributes
Image Display	EXEC	Execute Display
	TMDSP	Terminate Display
Termination	GSPTM	GSP Termination

GSPIN--Graphic Subroutine Package Initialization

The GSPIN subroutine specifies control information required by other GSP subroutines: the precision used by the program, variables for GSP error handling, and association of the device address with the logical unit number.

```
General Form
CALL  GSPIN(integer,real,return,
            cumulative,unit1,unit2,unit3,
            unit4)
```

integer
: is an integer constant or integer variable with the following values and meanings:

  0 = standard precision
  1 = "ONE WORD INTEGERS" FORTRAN control record used

real
: is an integer constant or integer variable with the following values and meanings:

  0 = standard precision
  1 = "EXTENDED PRECISION" FORTRAN control record used

return
: is an integer variable specifying where the called GSP subroutines will return an error code. (See "Error Handling.")

cumulative
: is an integer variable specifying where the called GSP subroutines will accumulate error indicators. (See "Error Handling.")

unit1,unit2,unit3,unit4
: are integer constants or integer variables defining the device address associated with the logical unit num-

ber (unit1...unit4). These logical unit numbers are arguments in many GSP subroutines. If an argument has a value of zero, the corresponding logical unit number will not have a 2250 device associated with it. Only one logical unit number can be associated with a device address. The device address must be the integer value 25. All other logical unit numbers must have the value 0.

CAUTION: A call to GSPIN must be issued before any other GSP subroutines are called. The call is issued at least once per program or link; additional calls to GSPIN within a program or link should be preceded by calls to GSPTM (GSP termination). An error will be indicated by the GSP if GSPIN is not the first GSP subroutine called.

GSPIN disables all attention sources. No attentions can be received till after the Set Attention Status (SATNS) subroutine is called enabling one or more attention sources (see "Attention Handling").

PROGRAMMING NOTES:

1. The 1130 FORTRAN precision rules are described in the 1130 Disk Monitor publication, and are summarized below. The numbers specify number of words.

	Integer		Real	
	Constant	Variable	Constant	Variable
Standard Precision	1	2	2	2
ONE WORD INTEGERS	1	1		
EXTENDED PRECISION	1	1		
	3*	3*	3	3

* if ONE WORD INTEGERS is specified, use 1 word; if not specified, use 3 words.

2. If a program is recompiled with a different set of precision control records, the arguments of the GSPIN call must be altered to agree with the new precision options.

3. 2250 display units will have device addresses assigned when they are installed. The "unit1...unit4" arguments permit the programmer to relate a logical number (1,2,3,4) to the actual device address, and in other GSP calls to refer to logical device numbers. By changing the arguments for GSPIN, the logical device may be reassigned and thus will not require extensive changes for other GSP calls.

ERRORS:

1. The "integer" or "real" argument is not 0 or 1.

2. A "unit" argument is not 0 or 25.

ICAIN--Image Construction Area Initialization

The ICAIN subroutine defines and initializes or redefines an image construction area (ICA) where image entities and associated control information are to be generated. The ICA so defined is the active ICA.

General Form
CALL ICAIN(corrval,ica(start),ica(end), option)

corrval
is defined in "Arguments Used by Many of the Subroutines." This ICA correlation value functions as an ICA identifier and is included in light pen attention data (see the discussion of the RQATN subroutine).

ica(start)
is a subscripted or non-subscripted integer variable specifying the first element of the array where image management and image generation subroutines will construct image entities and their control information. The subscript, if used, is defined by "(start)".

ica(end)
is a subscripted integer variable specifying the last element of the array named by "ica(start)". The subscript is defined by "(end)".

option
is an integer constant, integer variable, or integer arithmetic expression with the following values and meanings:

0 = define and initialize an ICA
1 = redefine an ICA

(See the programming notes for discussion and meanings of define, initialize, and redefine.)

CAUTIONS:

1. A redefine "option" code should not be used unless the ICA has been previously defined with a zero "option" code.

2. When redefining an ICA, the

"ica(start)" and "ica(end)" arguments must be the same as they were when the ICA was last defined.

3. If a defined ICA is referred to by statements other than calls to GSP subroutines, the results will be unpredictable.

PROGRAMMING NOTES:

1. Multiple ICAs may be defined, although this may not be necessary for most programs since more than one image entity may be constructed in a single ICA. The ICA last defined by ICAIN is the active ICA and is used by image management and image generation subroutines until another ICA is defined as the active ICA.

2. Redefining an ICA does not change its contents, but merely establishes it as the active ICA.

3. If a previously defined ICA is used as an argument in a call to ICAIN, with "option"=0, the ICA is set in an initial status, with no elements established; therefore, a previously defined ICA may be reused (redefined and reinitialized).

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The "ica(end)" argument is equal to or less than the "ica(start)" argument.

3. The "option" argument is not 0 or 1.

BELMT--Begin Element

The BELMT subroutine specifies the beginning of an element and establishes an identification of the element.

General Form
CALL BELMT(corrval,elementcode)

corrval
is defined in "Arguments Used by Many of the Subroutines."

elementcode
is an integer constant, integer variable, or integer arithmetic expression defining the type of element which is to be begun:

1 = uncontrolled entity
2 = controlled entity
3 = subroutine entity
4 = image entity

CAUTIONS:

1. Although elements may be nested, there are certain restrictions which must be observed. These are provided in "Nesting."

2. Nesting of subroutine entities within elements which may be deleted (see DELMT) should be done with caution, since the deletion of an element also deletes all embedded elements. A linkage to a subroutine entity thus deleted causes unpredictable results.

3. A subroutine entity must be defined before any linkages are made to it.

PROGRAMMING NOTES:

1. A light pen attention on a controlled entity ("elementcode" = 2) makes the correlation value available along with other attention data (see RQATN.), thus identifying the element detected by the light pen.

2. Message entities, linkage entities, tracking entities, and origin entities are defined, named, and generated by means of other GSP subroutines and therefore do not require BELMT and EELMT calls.

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The "elementcode" argument is not in the range 1 to 4.

EELMT--End Element

The EELMT subroutine defines the end of one or more elements.

General Form
CALL EELMT(corrval)

corrval
is defined in "Arguments Used by Many of the Subroutines." It specifies the outermost element to be ended.

PROGRAMMING NOTE: The EELMT subroutine defines the end (close) of all elements within and including the element identified by the "corrval" argument that have been previously defined and have not been previously closed. Figure 5 illustrates the concept.

```
10      CALL BELMT(1,n)      Begin Element #1


20      CALL BELMT(2,n)      Begin Element #2


30      CALL EELMT(2)        End Element #2
40      CALL BELMT(3,n)      Begin Element #3

50      CALL BELMT(4,n)      Begin Element #4


60      CALL BELMT(5,n)      Begin Element #5


70      CALL BELMT(6,n)      Begin Element #6


80      CALL EELMT(5)        End Element #5 & 6
90      CALL BELMT(7,n)      Begin Element #7


100     CALL EELMT(1)        End Elements
                             #1,3,4, & 7
```

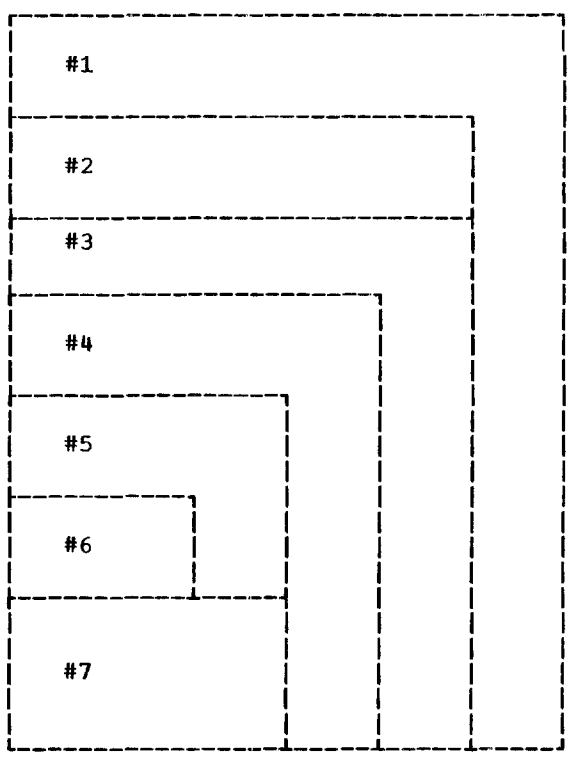


Figure 5. Beginning and Ending Elements

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The correlation value is not currently defined.

UELMT--Update Element

The UELMT subroutine specifies the beginning of an element update, starting at the beginning of the element. UELMT includes the option of changing the element type.

General Form
CALL UELMT(corrval,elementcode)

corrval
is defined in "Arguments Used by Many of the Subroutines."

elementcode
is an integer constant, integer variable, or integer arithmetic expression identifying the type of element which is to be generated:

1 = uncontrolled entity
2 = controlled entity
3 = subroutine entity
4 = image entity

CAUTIONS:

1. UELMT is in effect a combination of DELMT and BELMT. Therefore, if the element which is to be updated contains other embedded elements, these are also deleted, and their control information is lost. To avoid this, the cautions mentioned in the description of DELMT should be applied to UELMT as well.

2. A UELMT call may not be followed by XELMT or other UELMT calls until the updated element is ended.

3. If a subroutine entity is changed to another element type, linkages to that subroutine entity should first be deleted or made inactive.

PROGRAMMING NOTES:

1. UELMT, when applied to an embedded element, updates the element without disturbing its structural relation to the element(s) within which it is embedded.

2. After a call to UELMT is issued and before updating is halted by a call to

EELMT, other inner elements may be defined.

3. If it is expected that an element is to be inserted between other elements, the contingency may be provided for by defining a named, inactive, linkage entity. This inactive element can be updated when required.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value is not currently defined.
3. The call to UELMT was issued while the GSP was already in update mode.
4. The "elementcode" argument is not in the range 1 to 4.

XELMT--Extend Element

The XELMT subroutine designates the beginning of an element update, starting at the end of the element. The programmer provides an estimate of the number of lines, points, or characters that are to be added to the element.

General Form
CALL XELMT(corrval,lpcvalue)

corrval
is described in "Arguments Used by Many of the Subroutines."

lpcvalue
is a positive integer constant, integer variable, or integer arithmetic expression and should be an estimate of the number of lines, points, or characters which are to be added to the element.

CAUTIONS:

1. An XELMT call may not be followed by UELMT or other XELMT calls until the updated element has been ended.
2. XELMT may only be used to extend elements defined by BELMT or UELMT.

PROGRAMMING NOTE: The purpose of the "lpcvalue" argument is to improve the performance of the XELMT function. A value equal to or greater than the actual number of generated lines, points, or characters provides maximum efficiency. If the value is lower, efficiency is lost. If no reasonable estimate can be made, a value of zero can be used.

ERRORS:

1. The correlation value is not in the range of 1 to 32767.
2. The correlation value is not currently defined.
3. The correlation value is not for a controlled, uncontrolled, subroutine, or image entity.
4. The XELMT call was issued while the GSP was already in update mode.

DELMT--Delete Element

The DELMT subroutine deletes one or more previously defined elements. If the deleted element contains embedded elements, the embedded elements are also deleted.

General Form
CALL DELMT(corrval)

corrval
is defined in "Arguments Used by Many of the Subroutines."

CAUTIONS:

1. Since the deletion of an element also causes the deletion of embedded elements, care must be taken to prevent the inadvertent deletion of subroutine entities that are still referred to by linkage entities. To avoid the possibility of undesired deletion, such subroutine entities should not be defined as embedded elements.
2. The element following a deleted element may be repositioned on the screen unless it is absolutely positioned.

PROGRAMMING NOTE: When an element is deleted, its correlation value is no longer defined as currently valid, and the element may therefore not be referred to.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value was never defined, or the element has been previously deleted.

SATRB--Set Controlled Entity Attributes

The SATRB subroutine is used to change the visibility and detectability attributes of a completed controlled entity. When a controlled entity is first defined, the GSP assigns standard attributes to the entity. The SATRB subroutine can be used to alter the attributes from the standard attributes, or it can be used to change the attributes from those set by a previous call to the SATRB subroutine.

General Form
CALL SATRB(corrval,displaycode, detectcode)

corrval
is defined in "Arguments Used by Many of the Subroutines." It must identify a completed controlled entity.

displaycode
is an integer constant or integer variable specifying whether or not the controlled entity is to be displayed on the screen, as follows:

- -1 = the controlled entity is not to be displayed
- 0 = the visibility attribute is not to be changed
- +1 = the controlled entity is to be displayed
  Note: This is the standard visibility attribute established by the BELMT and UELMT subroutines.

detectcode
is an integer constant or integer variable specifying the detectability attribute for the controlled entity, as follows:

- -1 = no light pen attentions provided
- 0 = the detectability attribute is not to be changed
- +1 = light pen attention provided upon light pen detect with the light pen switch closed
  Note: This is the standard detectability attribute established by the BELMT and UELMT subroutines.

PROGRAMMING NOTES: The following paragraphs contain further information concerning the visibility and detectabity attributes:

displaycode = -1: When a "displaycode" of minus one is specified for a controlled entity, neither the controlled entity nor any embedded elements are displayed on the screen. However, the controlled entity remains available (i.e., it is not deleted) and is displayed if the programmer issues another call to the SATRB subroutine specifying that the controlled entity is to be displayed. When a controlled entity is not being displayed, the positions of elements following it are affected if they are not absolutely positioned.

displaycode = 0: This permits changing of the detectability attribute without affecting the existing visibility attribute.

displaycode = +1: The controlled entity and all embedded elements are displayed on the screen.

detectcode = -1: No light pen detect occurs within the specified controlled entity or any element nested within it.

detectcode = 0: This permits changing of the visibility attribute without affecting the existing detectability attribute.

detectcode = +1: A light pen attention is provided, if enabled, when a light pen detect with switch closure ccurs in the specified controlled entity or any element nested within it. The light pen attention data includes: (1) the light pen attention code; (2) one of the following:

a. the x- and y-coordinates for the point on which the detect occurred,

b. the x- and y-coordinates for the end point of a line on which the detect occurred,

c. the actual character on which the detect occurred;

(3) the correlation values of the ICA, image entity, and controlled entity; (4) the correlation value of the innermost named element within the controlled entity, in which the detect occurred; and (5) the correlation value of the lowest level subroutine entity or the innermost named element in the lowest level subroutine entity in which the light pen detect occurred. Items (4) and (5) above will be zero if not applicable.

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The correlation value is not for a controlled entity.

3. The definition of the controlled entity was never completed with a call to the EELMT subroutine.

4. An invalid "detectcode" argument was provided.

5. An invalid "displaycode" argument was provided.

EXEC--Execute Display

The EXEC subroutine causes an image entity to be displayed.

General Form
CALL EXEC(device,corrval,zero)

device and corrval
are defined in "Arguments Used by Many of the Subroutines." The "corrval" argument must identify a completed image entity residing in the active ICA.

zero
is an integer constant or integer variable and must have a value of zero.

PROGRAMMING NOTE: While an image entity is being displayed, other image entities may be generated in other ICAs. Image generation or updating of an image entity in the ICA containing the image entity being displayed must be preceded by a call to TMDSP to terminate the display.

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The correlation value is not defined in the active ICA as an image entity.

3. The "device" argument is invalid.

TMDSP--Terminate Display

The TMDSP subroutine terminates the display of an image entity.

General Form
CALL TMDSP(device)

device
is defined in "Arguments Used by Many of the Subroutines."

PROGRAMMING NOTE: Keyboard attentions may be accepted from the 2250 display unit after the display has been terminated by a call to TMDSP.

ERROR:

1. The "device" argument is invalid.

2. The display has already been terminated.

GSPTM--GSP Termination

The GSPTM subroutine resets the 2250, turns off the programmed function keyboard indicators, provides a dump of a specified length, and terminates the use of the GSP. The GSPTM subroutine should be called when the graphic processing portion of the program has been completed.

General Form
CALL GSPTM(dump,frmt,lolim,uplim)

dump
is an integer constant or integer variable specifying whether a dump is required:

0 = no dump
1 = dump

frmt
is an integer constant or integer variable and must have a value of zero to specify hexadecimal format.

lolim
is an integer constant or integer variable specifying the lower limit of the core dump.

uplim
is an integer constant or integer variable specifying the upper limit of the core dump.

PROGRAMMING NOTE: To reinitialize the GSP, the GSPIN subroutine must be called.

ERRORS:

1. The "dump" code is not 0 or 1.

2. The "frmt" argument does not have a value of zero.

3. The dump limits are negative.

IMAGE GENERATION SUBROUTINES

Image generation subroutines are concerned with the creation of graphic elements that will eventually be displayed. They define the content of an element by converting the input data to 2250 format. They do not actually cause a display; this is done by means of the EXEC subroutine.

The image generation subroutines can be considered as being analogous to Arithmetic statements, which process data with no actual output produced until the issuance of I/O statements. The image generation subroutines are listed below, and a detailed description of each, in the listed sequence, follows:

Category	Mnemonic	Name
GCA Definition	GCAIN	Generation Control Area Initialization
	SSCAL	Set Scaling Information
	SSCIS	Set Scissoring Option
	SINDX	Set Index Values
	SINCR	Set Increment Values
	SDATM	Set Input Data Mode
	SGRAM	Set Output Graphic Mode
Positioning	MVPOS	Move Element to a Position
	IDPOS	Indicate Element Position
LPC Generation	PLINE	Plot Lines
	PPNT	Plot Points
	PSGMT	Plot Line Segments
	PTEXT	Plot Text
Linkage	LKSUB	Linkage to a Subroutine
Miscellaneous	PGRID	Plot Grid Outline
	PCOPY	Plot Copy

GCAIN--Generation Control Area Initialization

The GCAIN subroutine initializes a generation control area (GCA) with standard values.

General Form
CALL GCAIN(gca)

gca
is defined in "Arguments Used by Many of the Subroutines."

PROGRAMMING NOTES:

1. GCAIN enables the programmer to initialize a GCA with a set of standard values and options which may correspond to some or all of the values and options he would specify using the six supplementary GCA definition subroutines: SSCAL, SSCIS, SINDX, SINCR, SDATM, and SGRAM. In many cases, GCAIN in combination with one or two supplementary GCA definition subroutines is sufficient to completely define a GCA. If all six supplementary GCA definition subroutines are used to define all values and options, GCAIN may be omitted.

2. A GCA may be altered at any time by one or more of the GCA definition subroutines.

3. A GCA must be fully defined before it is used as an argument for GSP subroutines other than the GCA definition subroutines.

4. The following set of standard values and options are used to initialize the specified GCA:

   a. Scaling (see SSCAL for definitions of mnemonics)
      SXL, SYL, GXL, GYL,
      DXL, DYL = 0
      SXU, SYU, GXU, GYU,
      DXU, DYU = 1023

      The grid limits are equated to the screen limits, and both screen and grid are defined by the diagonal of a rectangle whose lower-left corner is established by x- and y-coordinates of 0,0 and whose upper-right corner is established by x- and y-coordinates of 1023,1023. The x and y upper and lower data limits, specifying that input data is to be scaled to the grid area, are initialized with a range from 0 to 1023. Therefore, the x and y scaling factors are both 1 to 1. Note that the 2250 coordinate system, in raster units, has the same range, 0 to 1023.

   b. Scissoring (see SSCIS for definition)
      scissoring option = 2
      The scissoring option is set for scissoring to occur at the grid boundaries.

   c. Indexing (see SINDX for definitions)
      XSIND, YSIND, XEIND, YEIND = 1
      All array index values are set to 1.

   d. Incrementation (see SINCR for definitions)
      XSINC, YSINC, XEINC, YEINC = 0.
      All increment values are set to zero.

   e. Input data mode (see SDATM for definitions)
      XIPMD, YIPMD = 1
      The x and y input data mode is set to real absolute.

   f. Output graphic mode (see SGRAM for definitions)
      output mode = 1

The optimized graphic output mode is set.

SSCAL--Set Scaling Information

The SSCAL subroutine specifies scaling information for a GCA.

General Form
CALL SSCAL(gca,scalearray)

gca
is defined in "Arguments Used by Many of the Subroutines."

scalearray
is a real array with 12 elements:
element 1 = SXL - screen lower-left x-coordinate
2 = SYL - screen lower-left y-coordinate
3 = SXU - screen upper-right x-coordinate
4 = SYU - screen upper-right y-coordinate
5 = GXL - grid lower-left x-coordinate
6 = GYL - grid lower-left y-coordinate
7 = GXU - grid upper-right x-coordinate
8 = GYU - grid upper-right y-coordinate
9 = DXL - data lower-left x-coordinate
10 = DYL - data lower-left y-coordinate
11 = DXU - data upper-right x-coordinate
12 = DYU - data upper-right y-coordinate

where:

SXL, SYL
are arbitrary real values (screen units) representing the x- and y-coordinates that correspond to the lower-left corner of the screen. These coordinate values must be less than the coordinate values used to represent the upper-right corner of the screen.

SXU, SYU
are arbitrary real values (screen units) representing the x- and y-coordinates that correspond to the upper-right corner of the screen. These coordinate values must be greater than the coordinate values used to represent the lower-left corner of the screen.

GXL, GYL
are arbitrary real values representing the x- and y-coordinates that correspond to the lower-left corner of the grid and must be in the same units used to represent the screen. These coordinate values must be less than the coordinate values used to represent the upper-right corner of the grid.

GXU, GYU
are arbitrary real values representing the x- and y-coordinates of the upper-right corner of the grid and must be in the same units used to represent the screen. These coordinate values must be greater than the coordinate values used to represent the lower-left corner of the grid.

DXL, DYL
are real values (data units) representing the x- and y-coordinates of lower limits of the input data to be mapped into the grid. These values may be greater or less than the values specified for DXU and DYU.

DXU, DYU
are real values (data units) representing the x- and y-coordinate values that correspond to the upper limits of the input data to be mapped into the grid.

PROGRAMMING NOTES:

1. The screen is the total usable surface of the 2250 cathode-ray tube; the grid is a rectangular area equal to or smaller than the screen; the data limits represent the minimum and maximum data values which are to be mapped into the grid area.
2. Images projected beyond the grid or screen boundaries are cut off or "scissored" as specified by the scissoring option in the GCA (see SSCIS and GCAIN).
3. The scaling data defined by SSCAL is effective for elements generated by any calls that refer to the specified GCA until a new call to SSCAL alters the scaling data.
4. The scale array may be altered after a call to SSCAL without affecting the GCA.
5. See Figure 3 in the preceding section for an example of scaling data arguments used by the SSCAL subroutine.

ERRORS:

1. The lower-left screen or grid coordi-

nate values are greater than the corresponding upper-right values.

2. The grid lies outside the screen boundaries.

SSCIS--Set Scissoring Option

The SSCIS subroutine specifies the scissoring option for a GCA (see Figure 4 in the preceding section).

General Form
CALL SSCIS(gca,scisoption)

gca
is defined in "Arguments Used by Many of the Subroutines."

scisoption
is an integer constant, integer variable, or integer arithmetic expression with the following values and meanings:

1 = scissoring occurs at screen boundaries
2 = scissoring occurs at grid boundaries

PROGRAMMING NOTES:

1. The subroutine entity makes it possible to display a particular element at various locations on the screen, depending on the positioning prior to the linkage to the subroutine entity. The generation of data for subroutine entities is therefore in incremental, rather than absolute, graphic units. Scaling is based on the GCA scaling data which establishes scaling factors from the screen, grid, and data definitions (see SSCAL). Within a subroutine entity, however, the grid no longer applies to a particular rectangular area on the screen. Scissoring is therefore suppressed for the generation of elements within a subroutine entity.

2. The scissoring option defined by a call to SSCIS is effective for elements generated by calls using the specified GCA until a new call to SSCIS changes the scissoring option for that GCA.

ERROR: The "scisoption" argument does not have a value of 1 or 2.

SINDX--Set Index Values

The SINDX subroutine specifies indexing information used in accessing data from input arrays.

General Form
CALL SINDX(gca,indexarray)

gca
is defined in "Arguments Used by Many of the Subroutines."

indexarray
is an integer array with four elements:

element 1 = XSIND, X start index: used in PLINE, PPNT, and PSGMT subroutines to index the X input array for successive input data.
2 = YSIND, Y start index: performs for the y-coordinate the same function as XSIND.
3 = XEIND, X end index: used in the PSGMT subroutine, which requires 2 arrays for the X start and end values of a line segment.
4 = YEIND, Y end index: performs for the y-coordinate the same function as XEIND.

The value assigned to each element of the index array must be a positive integer.

PROGRAMMING NOTES:

1. Indexing provides a means of making a small selection or sampling from a large array.

2. Input data may be combined into one array. For example, the input data for the x- and y-coordinates used by PLINE may be alternated in a single input array, and identified by subscripting (i.e. arrayname(1) for x data and arrayname(2) for y data). Then x and y start index values of +2 would refer to the desired data in alternating sequence.

3. The first input data obtained from, or placed in, an array is the array element named as the argument. The index applied to this array element causes the second and succeeding elements to be obtained.

4. The index data defined by any call to SINDX is used for elements generated by subroutines using the specified GCA

until a new call to SINDX alters the indexing information for the GCA.

5. The index array may be altered after a call to SINDX without affecting the GCA.

ERRORS: Elements in the index array contain zeros or negative values.

SINCR--Set Increment Values

The SINCR subroutine specifies the incrementation of current x- and y-coordinates by a designated amount for each point, line, and line segment.

General Form
CALL SINCR(gca,incrementarray)

gca
is defined in "Arguments Used by Many of the Subroutines."

incrementarray
is a real array with four elements, each specified in data units:

element 1 = XSINC, X start increment: the value by which the x-coordinate is to be incremented.
2 = YSINC, Y start increment: the value by which the y-coordinate is to be incremented.
3 = XEINC, X end increment: the value by which the x-coordinate that defines the end of a line segment is to be incremented. It is used only with the PSGMT subroutine.
4 = YEINC, Y end increment: the value by which the y-coordinate that defines the end of a line segment is to be incremented. It is used only with the PSGMT subroutine.

PROGRAMMING NOTES:

1. If the value of an element in the increment array is non-zero (e.g. XSINC = 2), a new coordinate value is computed by increasing the last established coordinate by the specified constant increment (here, 2).

2. If the value of an element in the increment array is zero, the derivation of a new coordinate value depends on the x or y input data mode which has been established in the GCA. For a further description of this condition, see SDATM.

3. It is possible for x-coordinates to be derived by a constant increment and y-coordinates to be derived by input data, and vice versa. See Figure 6 for the relationship of a constant increment specified by means of a call to SINCR and x or y incremental data specified by means of a call to SDATM.

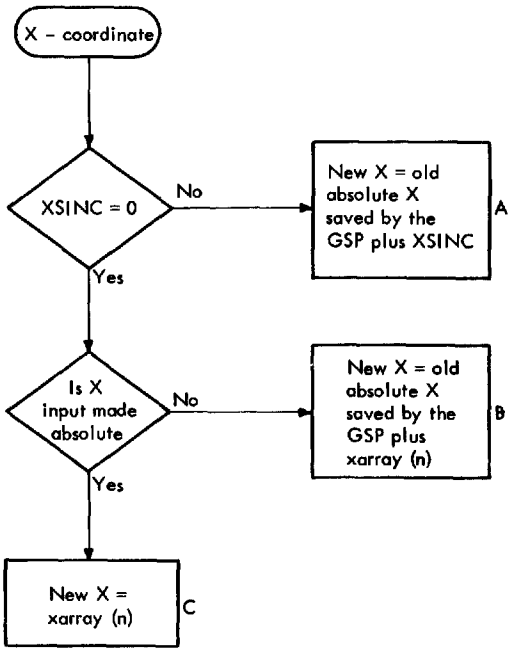
4. Increments may be positive or negative.

5. The increment data defined by any call to SINCR is used for elements generated by subroutines using the specified GCA until a new call to SINCR alters the incrementation values for that GCA.

6. After a call to SINCR, the increment array may be altered without affecting the GCA.

SDATM--Set Input Data Mode

The SDATM subroutine specifies the input data mode for a GCA.



NOTE: The increment specified by a call to SINCR has priority and furnishes a fixed increment regardless of the input data mode or contents of the data array (see A above).

Figure 6. Incrementation by SINCR and SDATM

General Form
CALL SDATM(gca,xipmd,yipmd)

gca
is defined in "Arguments Used in Many of the Subroutines."

xipmd,yipmd
are integer constants, integer variables, or integer arithmetic expressions defining the mode in which input data is available for the designation of x- and y-coordinates:

1 = real absolute
2 = real incremental
3 = integer absolute
4 = integer incremental
5 = integer absolute in 2250 raster units

CAUTION: If the GCA x or y input data mode and the mode of the input data do not agree, the results will be unpredictable.

PROGRAMMING NOTES:

1. The input data modes defined by any call to SDATM are used for elements generated by subroutines using the specified GCA until a new call to SDATM changes the mode for that GCA.

2. No scaling is done if the input data mode specifies 2250 raster units.

3. See Figure 6 for the relationship of an increment specified by means of a call to SINCR and x or y incremental input data specified by means of a call to SDATM.

ERRORS: The "xipmd" or "yipmd" argument is not a positive integer in the range 1 to 5.

SGRAM--Set Output Graphic Mode

The SGRAM subroutine specifies the output generation mode for a GCA.

General Form
CALL SGRAM(gca,opmd)

gca
is defined in "Arguments Used by Many of the Subroutines."

opmd
is an integer constant, integer variable, or integer arithmetic expression with the following values and meanings:

1 = optimized output graphic mode
2 = absolute output graphic mode
3 = incremental output graphic mode

PROGRAMMING NOTES:

1. The GSP image generation subroutines generate incremental output data for subroutine entities regardless of the output mode specified for the GCA.

2. The output mode specified by any call to SGRAM is used for elements generated by calls using the specified GCA until a new call to SGRAM changes the mode for that GCA.

ERRORS: The "opmd" argument is not a positive integer in the range 1 to 3.

MVPOS--Move Element to a Position

The MVPOS subroutine creates an origin entity, in either absolute or incremental mode, which positions the next element. It provides the option of naming the origin entity or of updating a previously named origin entity.

General Form
CALL MVPOS(gca,xcoor,ycoor,corrval)

gca and corrval
are defined in "Arguments Used by Many of the Subroutines." The "corrval" argument may have a value of zero (see programming notes).

xcoor,ycoor
area constants, variables, or arithmetic expressions representing the x- and y-coordinates where the element is to be positioned. These values must be in the input data mode defined in the GCA.

CAUTION: A previously defined non-zero correlation value must have been defined as an origin entity (see programming notes).

PROGRAMMING NOTES:

1. If the "corrval" argument is zero, MVPOS generates an unnamed origin entity.

2. If the "corrval" argument is non-zero, and has not been previously defined, MVPOS defines a named origin entity. Thus defined, it may be updated by subsequent calls to MVPOS.

3. If the "corrval" argument is non-zero, and has been previously defined as identifying an origin entity, MVPOS

updates the origin entity using the specified location.

4. The output generation mode used is as defined in the GCA, except that an output mode of optimize results in absolute positioning, and origin generation within a subroutine entity is always in incremental form.

5. If MVPOS is called following a call to a subroutine entity, input and output should, ideally, be in absolute mode in order to reestablish absolute positioning and to permit scissoring of subsequently generated incremental data.

6. If absolute repositioning does not follow a call to a subroutine entity, the GSP assumes a closed subroutine entity where positioning after execution of the subroutine is the same as it was just prior to calling the subroutine entity.

7. The positioning may be indicated to the GSP without causing generation of graphic data by issuing a call to IDPOS. With an absolute origin reestablished, subsequent generation may be incremental and still be capable of scissoring.

ERRORS:

1. An attempt is made to update an element previously defined as other than an origin entity.

2. The correlation value is not in the range 0 to 32767.

3. The correlation value is not currently defined.

IDPOS--Indicate Element Position

The IDPOS subroutine indicates the starting point from which subsequent x- and y-coordinates are to be computed. IDPOS produces no graphic data, but provides the GSP with a starting point that is necessary for applying scissoring as defined in the GCA.

General Form
CALL IDPOS(gca,xcoor,ycoor)

gca
is defined in "Arguments Used by Many of the Subroutines."

xcoor,ycoor
are constants, variables, or arithmetic expressions representing the x- and y-coordinates where the next element is to be positioned. The "xcoor" and "ycoor" arguments must be in the input data mode defined in the GCA.

PROGRAMMING NOTE: The IDPOS will normally be used following generation of a linkage entity and during element updates.

PLINE--Plot Lines

The PLINE subroutine generates graphic data to produce lines. Scaling, indexing, incrementation, and scissoring are performed in accordance with the control data in the GCA.

General Form
CALL PLINE(gca,xcoor,ycoor,count)

gca and count
are defined in "Arguments Used by Many of the Subroutines."

xcoor,ycoor
are constants, variables, arrays, or arithmetic expressions specifying the x- and y-coordinate input data. The "xcoor" and "ycoor" arguments must be in the input data mode defined in the GCA.

CAUTION: If the value of the "count" argument exceeds the number of elements in "xcoor" and "ycoor", the results will be unpredictable.

PROGRAMMING NOTE: The PLINE subroutine assumes that the element has been positioned on the screen and that the first input data, either from the x or y input data arrays or computed by increment values, represents the end position of the first line to be generated.

ERROR: The "count" argument is negative or zero.

PPNT--Plot Points

The PPNT subroutine generates graphic data to produce points. Scaling, indexing, incrementation, and scissoring are performed in accordance with the control data in the GCA.

General Form
CALL PPNT(gca,xcoor,ycoor,count)

gca and count
are defined in "Arguments Used by Many of the Subroutines."

xcoor,ycoor
are constants, variables, arrays, or arithmetic expressions specifying the x- and y-coordinate input data. "xcoor" and "ycoor" must be in the input data mode defined in the GCA.

CAUTION: If the value of the "count" argument exceeds the number of elements in "xcoor" and "ycoor", the results will be unpredictable.

PROGRAMMING NOTE: The PPNT subroutine assumes that the element has been positioned on the screen and that the first input data, either from the x and y input data arrays or computed by increment values, represents the position of the first point to be generated.

ERROR: The "count" argument is negative or zero.

PSGMT--Plot Line Segments

The PSGMT subroutine generates graphic data to produce one or more line segments. Scaling, indexing, incrementation, and scissoring are performed in accordance with the control data in the GCA.

General Form
CALL PSGMT(gca,xscoor,yscoor,xecoor, yecoor,count)

gca and count
are defined in "Arguments Used by Many of the Subroutines."

xscoor,yscoor
are constants, variables, arrays, or arithmetic expressions specifying the x and y starting coordinates for each line segment to be produced. The "xscoor" and "yscoor" arguments must be in the input data mode defined in the GCA.

xecoor,yecoor
are constants, variables, arrays, or arithmetic expressions specifying the x and y end coordinates for each line segment to be produced. The "xecoor" and "yecoor" arguments must be in the input data mode defined in the GCA.

CAUTION: If the value of the "count" argument exceeds the number of elements in the input data, the results will be unpredictable.

ERROR: The "count" argument is negative or zero.

PTEXT--Plot Text

The PTEXT subroutine generates graphic data to produce characters (see Appendix F).

General Form
CALL PTEXT(gca,text,count,size,textcode)

gca, count, and textcode
are defined in "Arguments Used by Many of the Subroutines."

text
is a variable or array designating the alphameric characters to be displayed. The character data designated by "text" must agree with the type (real or integer) and format (A or I) specified in "textcode."

size
is an integer constant, integer variable, or integer arithmetic expression with the following values and meanings:

1 = basic character size
2 = large character size

CAUTION: If the "count" argument exceeds the number of elements in "text", the results will be unpredictable.

PROGRAMMING NOTES:

1. The PTEXT subroutine assumes that the beam has been positioned at the center point of the first character to be displayed.

2. If the "textcode" argument has a value of 3 and the "count" argument has a value of 1, a single character can be generated from alphameric keyboard data or light pen attention data returned by a call to RQATN.

3. Scaling, incrementation, and scissoring options are not applicable to PTEXT.

4. Characters are either absolutely or incrementally positioned as determined by the output graphic mode for the GCA. In absolute mode, a new line is begun automatically when the x-coordinate for a character is beyond the right edge of the screen. If the new line function should cause the y-coordinate for a character to be below the bottom of the screen, the

character is positioned in the upper-left corner of the screen.

In incremental mode, the new line function is not performed. When the x-coordinate for a character is beyond the right edge of the screen, that and following characters are blank. If there are more than 74 basic size or 49 large size characters beyond the right edge of the screen, the excess appear on the left side of the screen and on the same line.

If the GCA specifies optimized output graphic mode, characters are positioned absolutely.

ERRORS:

1. The "count" argument is negative or zero.

2. The "size" argument is not 1 or 2.

3. The "textcode" argument is not in the range 1 to 4.

LKSUB--Linkage to a Subroutine

The LKSUB subroutine creates either an active or inactive linkage to a subroutine or tracking entity. It provides the option to name the linkage entity or update a previously named linkage entity.

General Form
CALL LKSUB(subroutine,corrval,switch)

subroutine
is a correlation value identifying a previously named subroutine or tracking entity. This argument may have a value of zero (see programming notes).

corrval
is defined in "Arguments Used by Many of the Subroutines." It identifies the linkage entity that is being created or updated. This argument may have a value of zero (see programming notes).

switch
is an integer constant, integer variable, or integer arithmetic expression with the following values and meanings:

1 = set as active linkage entity
2 = set as inactive linkage entity

PROGRAMMING NOTES:

1. Creating a linkage entity: LKSUB creates an unnamed linkage entity if the "corrval" argument has a value of zero. A linkage entity so created cannot be updated. LKSUB creates a named linkage entity if the "corrval" argument has a non-zero value that does not identify a previously defined element. A linkage entity so created can be referred to by its correlation value and modified (see below). The named linkage entity may be either active ("switch" = 1) or inactive ("switch" = 2). If the "subroutine" argument has a value of zero, the linkage entity is made inactive regardless of the value specified for "switch".

2. Modifying a linkage entity: LKSUB updates a linkage entity if the "corrval" argument identifies a previously created linkage entity. An inactive linkage entity can be made active and vice versa; the "subroutine" argument can be respecified; or both the "subroutine" and "switch" arguments can be respecified. In changing an inactive linkage entity to active, the "subroutine" argument may have a value of zero if the correlation value of a subroutine or tracking entity was specified for that argument when the linkage entity was created.

ERRORS:

1. The linkage is being made to an element not defined as a subroutine entity or tracking entity.

2. The correlation value is not in the range 0 to 32767.

3. The "subroutine" argument is not in the range 0 to 32767.

4. The correlation value of the subroutine or tracking entity is not currently defined.

5. The "switch" argument is not 1 or 2.

EXAMPLES: The first example below creates a named, inactive linkage entity. The second example makes that linkage entity active. Since the element being linked to is identified in the first statement, the "subroutine" argument in the second statement may have a value of zero.

```
CALL  LKSUB(5,10,2)
CALL  LKSUB(0,10,1)
```

PGRID--Plot Grid Outline

The PGRID subroutine generates the four

lines which form the grid as defined in the GCA.

General Form
CALL PGRID(gca)

gca
is defined in "Arguments Used by Many of the Subroutines."

PROGRAMMING NOTES:

1. Any rectangle (grid outline) may be generated by altering the GCA by means of a call to SSCAL.

2. The data defined by SSCAL effectively defines the "area of interest" of the input data. PGRID provides a means of putting a visible frame around this area of interest. This may be even more evident if scissoring on screen boundaries is specified. In this case, all scaled input data which falls within the screen area will be displayed, and the grid frame will outline the area of interest.

PCOPY---Plot Copy

The PCOPY subroutine copies a previously defined, named element and includes it as all or part of the current element which is being generated or updated.

General Form
CALL PCOPY(corrval)

corrval
is defined in "Arguments Used by Many of the Subroutines." It identifies the element which is to be copied.

CAUTIONS:

1. Named elements embedded within the element being copied are also copied, but in their new location will no longer have correlation values associated with them.

2. Any positioning data (see MVPOS) within the element being copied becomes a part of the new element which is being generated or updated. If this positioning data is in absolute mode, the copied element will be displayed at the same location on the 2250 screen as the original element.

PROGRAMMING NOTES:

1. The PCOPY subroutine permits duplication of previously generated elements without the necessity either of providing the input data or of duplicating the image generation calls used to produce the previous generation.

2. To be meaningful, the element being copied should consist only of incremental data, so that it may be relocated on the 2250 screen.

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The correlation value is not currently defined.

3. The element defined by the correlation value is incomplete.

The GSP provides subroutines that permit communication between the 2250 operator and the program. The program communicates with the operator by placing a message on the 2250 screen. It also provides the means by which the operator can respond to the message by making the light pen, programmed function keyboard, or alphameric keyboard available for use. The operator communicates with the program by entering information from the alphameric keyboard, by depressing a programmed function key, or by pointing the light pen at a displayed image entity. Any such response causes an attention.

An attention is an interruption that causes the program to change its course at an unpredictable point. When an attention occurs, processing of the user's program is interrupted and control is passed to the GSP to process the attention. This processing results in information about the attention being made available to the user's program upon request.

How the attention information is to be handled must be specified by the programmer. The information may be ignored or may be held for later processing upon request. The processing is done by routines written by the programmer.

There are four attention sources available to the GSP user:

- The programmed function keyboard
- The END key on the alphameric keyboard
- The alphameric keyboard (except END or CANCEL keys)
- The light pen

Note: The GSP redisplays the image entity after a keyboard attention. The GSP restarts the 2250 (initiates regeneration) after a light pen attention, but does not redisplay the image entity. The program must therefore issue a call to EXEC following light pen attentions to redisplay the image entity.

Enabling and Disabling Attention Sources

The programmer permits the saving of attention information from a particular source by enabling that source. He causes the attention information to be ignored by disabling the source. The GSP provides one subroutine that both enables and disables attention sources.

Desired sources must be enabled by a call to the Set Attention Status (SATNS) subroutine. Attention information from disabled sources is ignored. Attention sources may be repeatedly enabled or disabled.

The enabling of an attention source causes all attentions from that source to be accepted by the GSP. By calls to the Set Controlled Entity Attributes (SATRB) subroutine, the programmer can designate controlled entities for which he does not wish light pen attentions to be accepted. Light pen attentions do not occur on elements so designated, thus preventing unwanted light pen attentions from being processed.

Saving Attention Information

When an attention occurs from an enabled source, the program is interrupted and attention information is saved. The attention information contains the identification of the source that caused the attention. Additional information about that attention (e.g., the character detected by the light pen) is also provided.

Once the attention information is saved, control is returned to the program at the point where that program was interrupted. The program is not notified that an attention has occurred until attention information is requested by a call to the Request Attention Information (RQATN) subroutine.

Using the CANCEL Key

The alphameric keyboard CANCEL key is reserved for direct communication between the 2250 operator and the GSP. The key is automatically enabled as soon as the 2250 is identified by means of a call to GSPIN. It is to be used when the 2250 operator has recognized a condition in the program which warrants interruption.

When the CANCEL key is depressed, all activity on the 2250 is immediately suspended, and the alphameric keyboard END key is enabled. (The CANCEL key remains enabled.) All other attention sources are temporarily disabled. The 2250 operator must now make one of the following choices:

1. Depress the END key, which causes the program to terminate and a core dump to be produced.

2. Again depress the CANCEL key, which restores the program as it was before the preceding CANCEL key attention occurred and causes it to continue execution at the point of the initial interruption.

ATTENTION-HANDLING SUBROUTINES

This section describes the attention-handling subroutines available for communication between the program and the 2250 operator. These subroutines are as follows:

Set Attention Status (SATNS)
Request Attention Information (RQATN)
Return Outer Correlation Value (ROCOR)

SATNS--Set Attention Status

The SATNS subroutine designates attention sources to be processed by the program (enabled sources) and disables all other sources, including any sources enabled by a prior call to SATNS. Any unprocessed attention information is removed.

Once attention sources are enabled, attention information from the enabled sources is accepted, while attention information from sources not enabled is ignored. Once attention information has been accepted, it can be requested at any time by a call to the RQATN subroutine.

The SATNS subroutine may be called as often as desired to enable and disable attention sources. Prior to the first call to SATNS, all attention sources are disabled.

General Form
CALL SATNS(device,attnsource)

device
is defined in "Arguments Used by Many of the Subroutines."

attnsource
is an integer constant, integer variable, or integer arithmetic expression representing the code of an attention source, or attention sources, to be enabled. The sources and their codes are as follows:

```
 0 = all sources disabled
 2 = light pen
 4 = END key
 8 = alphameric keyboard key
16 = programmed function keyboard key
30 = all sources enabled
```

More than one source at a time can be designated by adding the individual codes and using the sum as the "attnsource" argument (e.g., 6 for light pen and END key).

PROGRAMMING NOTES: The CANCEL key on the alphameric keyboard is reserved for a system function and is not under the control of the SATNS subroutine. (See "Using the CANCEL Key.")

ERRORS:

1. The "attnsource" code is invalid.
2. The "device" argument is invalid.

RQATN--Request Attention Information

The RQATN subroutine enables the programmer to obtain attention information at any point in his program. By calling upon this subroutine, the programmer can determine if an attention has occurred and can identify its source.

General Form
CALL RQATN(device,arrayname)

device
is defined in "Arguments Used by Many of the Subroutines."

arrayname
is a 20-element integer array into which the attention information is to be placed. The attention information is inserted into the array as shown in Table 1. An array element contains meaningful information only for those attention sources designated as applicable to the element. For example, elements 5, 6, and 7 do not contain meaningful information if a keyboard is the attention source.

PROGRAMMING NOTE: A call to RQATN should normally be followed by an IF statement to determine whether or not an attention has occurred (a value of zero in the first element of the array indicates that an attention did not occur).

Example: The following sample coding illustrates a procedure that can be followed when it is necessary to wait for an attention before proceeding with program execution.

```
10  CALL RQATN(1,IARAY)
    IF (IARAY(1)) 80,20,30
20  PAUSE
    GO TO 10
30  (process attention)
.
.
.
80  (error - this condition should not
    occur)
```

ERROR: The "device" argument is invalid.

Table 1. Format of the Array for RQATN

Element	Contents
Array(1)	Zero if no attention occurred; or one of the following attention source codes: Code — Source 2 — light pen 4 — END key 8 — alphameric key 16 — programmed function key
Array(2)	A decimal number in the range 1 to 4 corresponding to the logical unit number.
Array(3)	For light pen, the decimal equivalent of the character detected, if applicable (see the description of "textcode" = 3 in "Arguments Used by Many of the Subroutines"); or minus one if the detect was not on a character. For programmed function key, a decimal number in the range 0 to 31 corresponding to the depressed key. For alphameric key, the decimal equivalent of the character entered from the keyboard (see Appendix F); or one of the following values: 256 for the JUMP key 512 for the BACKSPACE key 1024 for the ADVANCE key
Array(4)	For light pen, the x-coordinate beam position in 2250 raster units. For programmed function key, a decimal number in the range 0 to 255 corresponding to the overlay.
Array(5)	For light pen, the y-coordinate beam position, in 2250 raster units.
Array(6)	For light pen, the correlation value of the ICA.
Array(7)	For light pen, the correlation value of the image entity.
Array(8)	For light pen, the correlation value of the controlled entity.
Array(9)	For light pen, the correlation value of the innermost named element within the controlled entity in which the light pen detect occurred; or zero if not applicable.
Array(10)	For light pen, the correlation value of the lowest level subroutine entity or the innermost element within the lowest level subroutine entity in which the light pen detect occurred; or zero of not applicable.
Array(11) . . . Array(20)	Reserved. Reserved.

ROCOR--Return Outer Correlation Value

The ROCOR subroutine enables the programmer to obtain the correlation value and the identity of an element in which another element is nested.

General Form
CALL ROCOR(corrval,outer,elementcode)

corrval
is defined in "Arguments Used by Many of the Subroutines."

outer
is an integer variable indicating where the correlation value of the outer element (the one in which the element identified by the "corrval" argument is nested) is to be returned.

elementcode
is an integer variable indicating where a code identifying the outer element is to be returned. The values that can be returned are as follows:

0 = correlation value of an image entity, tracking entity, or subroutine entity was provided
1 = uncontrolled entity
2 = controlled entity
3 = subroutine entity
4 = image entity

CAUTION: Only the active image construction area (that is, the one last referred to in a call to the ICAIN subroutine) is searched. The "corrval" argument must therefore identify an element within the active ICA.

PROGRAMMING NOTE: The RQATN subroutine returns the correlation value of an element on which a light pen detect occurs (see Table 1). This correlation value can be used as the "corrval" argument for the ROCOR subroutine to identify the next outer element.

ERROR: The "corrval" argument is not defined in the active ICA.

EXAMPLE: Given is the following image entity:

```
678 9
{[()()]}
```

If the "corrval" argument identifies element 8 or 9 the ROCOR subroutine returns the correlation value of element 7 for the "outer" argument and a value of 2 (controlled entity) for the "elementcode" argument.

ENTERING DATA WITH THE ALPHAMERIC KEYBOARD

The following paragraphs describe the subroutines that allow text data to be entered from the alphameric keyboard. These subroutines are as follows:

Define Message Entity (DFMSG)
Message Entity Initialization (MSGIN)
Insert Cursor (ICURS)
Remove Cursor (RCURS)
Translate Message Data (TLMSG)

DFMSG--Define Message Entity

The DFMSG subroutine is used to create a message entity or to redefine a message entity that was previously defined. A message entity is an element in which alphameric data entered from the alphameric keyboard or generated by the program can be placed. The message entity is identified by the correlation value supplied in the call to the DFMSG subroutine.

If the message entity was previously defined, a call to this subroutine can be used to change the number of characters associated with the entity or to change the size at which those characters are to be displayed on the screen. The DFMSG subroutine can also be used to fill the entity with null characters or blank characters.

General Form
CALL DFMSG(corrval,count,size,initval)

corrval and count
are defined in "Arguments Used by Many of the Subroutines." The "count" argument indicates the number of characters in the message entity. A value of zero specifies no change.

size
is an integer constant, integer variable, or integer arithmetic expression specifying the size in which characters are to be displayed on the screen, as follows:

0 = no change
1 = basic size characters
2 = large size characters

initval
is an integer constant, integer variable, or integer arithmetic expression specifying whether the message entity is to be initialized with null characters or blank characters, as follows:

0 = no change

1 = null characters (which do not cause the beam to be repositioned)
2 = blank characters (which do cause the beam to repositioned)

CAUTION: If null characters are used in a message entity and the entity is followed by incremental graphic data, the replacement of a null character with a blank or alphameric character will cause the graphic data following the message entity to be displayed at a different point on the screen. To avoid this, the programmer should either: (1) initialize the message entity with blank characters, or (2) ensure that the following element is absolutely positioned.

PROGRAMMING NOTES: Displayable characters are placed in a message entity in one of two ways. Either a call to the MSGIN (Message Entity Initialization) subroutine causes alphameric data from the program to be placed in the message entity, or a call to the ICURS (Insert Cursor) subroutine permits characters entered at the alphameric keyboard to be placed in the message entity.

See the programming notes for PTEXT for information about the effect of the output graphic mode on displaying characters.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value does not identify a message entity.
3. The "count" argument is negative.
4. The "size" argument is not 0, 1, or 2.
5. The "initval" argument is not 0, 1, or 2.
6. The "count", "size", or "initval" argument is 0, but the message entity was not previously defined.

MSGIN--Message Entity Initialization

The MSGIN subroutine is used to place alphameric data into a previously defined message entity.

General Form
CALL MSGIN(corrval,charpos,text,count, textcode)

corrval, count, and textcode
are defined in "Arguments Used by Many of the Subroutines." The correlation value identifies the message entity into which the characters are to be placed. The "count" argument specifies the number of characters to be placed in the message entity.

charpos
is an integer constant, integer variable, or integer arithmetic expression indicating the character position within the message entity at which the first character is to be placed. Its value must be positive. A value of 1 indicates the first character position; 2 indicates the second character position, etc. By using a variable and by increasing its value by 1 each time the variable is used, the programmer can place one character at a time into the message entity.

text
is a variable or array designating the alphameric characters to be placed in the message entity. The character data designated by "text" must agree with the type (real or integer) and format (A or I) specified in "textcode".

CAUTION: The total number of character positions in a message entity is established by the "count" argument provided when the entity is defined by a call to the DFMSG (Define Message Entity) subroutine. The "count" and "charpos" arguments of the MSGIN subroutine determine the maximum number of character positions that are filled in the message entity. If the number of characters in the variable or array exceeds the available character positions in the message entity, the rightmost characters in the variable or array are omitted. If the number of characters in the variable or array is not sufficient to fill the available character positions, the remaining characters in the message entity are unchanged.

PROGRAMMING NOTE: The message entity specified in a call to the MSGIN subroutine can later be specified in a call to the ICURS (Insert Cursor) subroutine.

See the programming notes for PTEXT for information about the effect of the output graphic mode on displaying characters.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value does not identify a message entity.

3. The "count" argument is not a positive non-zero integer.

4. The value of "count" plus "charpos" specified in this call exceeds the value of the "count" argument used in the last call to the DFMSG subroutine for this message entity.

5. The "textcode" argument is not in the range 1 to 4.

ICURS--Insert Cursor

The ICURS subroutine causes a cursor to be placed at a specified position in a message entity and to be displayed. Once the cursor has been set, characters entered at the alphameric keyboard are placed in the message entity and displayed on the screen.

General Form
CALL ICURS(device,corrval,charpos)

device and corrval
are defined in "Arguments Used by Many of the Subroutines." The "corrval" argument identifies the message entity into which the cursor and alphameric data are to be placed.

charpos
is an integer constant, integer variable, or integer arithmetic expression specifying the character position within the message entity at which the cursor is to be set. Its value must be positive. A value of 1 indicates the first character position; 2 specifies the second character position, etc.

CAUTION: Before calling this subroutine, the image entity containing the message entity (identified by the "corrval" argument) must have been displayed by means of a call to EXEC.

PROGRAMMING NOTES:

1. After the cursor appears on the screen, a character entered from the alphameric keyboard is placed at the location of the cursor, and the cursor is moved to the next position in the message entity. A space produces a blank character. If a character is entered in the last position of the message entity, the cursor is not moved, and any new characters are entered at the cursor.

2. The cursor can be backspaced using the BACKSPACE key, but not beyond the first position of the message entity. Backspacing does not destroy characters.

3. The same character can be entered several times in succession by holding down the CONTINUE key and depressing the desired character key. Note that the multiple entries are accepted until the CONTINUE key is released or until the message entity is filled.

4. Depression of the JUMP key causes the cursor to be moved to the first position of the next message entity in the image entity, if any, or to the first position of the same message entity if there is only one. This enables the 2250 operator to "tabulate" or "skip to next line", depending on where the message entities are positioned on the screen.

5. When a call to the ICURS subroutine is issued for a 2250, all alphameric keyboard attentions are processed by the GSP even though those attentions may have been enabled for program processing.

6. The cursor produced by the ICURS subroutine remains visible until it is removed from the screen by a call to the RCURS (Remove Cursor) subroutine.

7. If a second call to the ICURS subroutine is issued before a call is made to the RCURS subroutine, the arguments in the second ICURS call become effective immediately.

ERRORS:

1. The correlation value does not identify a message entity.

2. The image entity containing the message entity is not being displayed by means of a call to EXEC.

3. The "device" argument is invalid.

4. The "charpos" argument is greater than the size of the message entity, or is zero or negative.

RCURS--Remove Cursor

The RCURS subroutine removes the cursor from the 2250 screen and terminates message collection from the device.

General Form
CALL RCURS(device)

device
is defined in "Arguments Used by Many of the Subroutines."

PROGRAMMING NOTE: This subroutine is called when the programmer no longer wants to accept alphameric keyboard input from the device. The cursor is removed from the screen, no further alphameric input is accepted from the device, and the alphameric keyboard reverts to its previous attention status.

ERROR: The "device" argument is invalid.

TLMSG--Translate Message Data

The TLMSG subroutine is used to convert data associated with a message entity from 2250 format to EBCDIC format and to place the translated data into a specified variable or array. When in 2250 format, alphameric data is suitable only for display.

General Form
CALL TLMSG(corrval,text,elcount, textcode)

corrval and textcode
are defined in "Arguments Used by Many of the Subroutines." The correlation value identifies the message entity that is to be translated. The message entity must have been defined previously by a call to the DFMSG subroutine.

text
is a variable or array specifying where the translated data is to be placed. The character data designated by "text" must agree with the type (real or integer) and format (A or I) specified in "textcode".

elcount
is an integer constant, integer variable, or integer arithmetic expression defining the number of elements in the text array.

CAUTIONS:

1. If the message entity contains fewer characters than the capacity of the variable or array, the character positions in the unused portion of the variable or array are filled with blanks.

2. If the number of characters associated with the message entity exceeds the capacity of the variable or array, excess characters at the end of the message entity are not translated.

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The correlation value does not identify a previously defined message entity.

3. Message entity characters were not translated because the message entity exceeds the capacity of the variable or array.

4. The "textcode" argument is not in the range 1 to 4.

5. The "elcount" argument is zero or negative.

ENTERING DATA WITH THE LIGHT PEN

The following paragraphs describe the subroutines that facilitate communication between the GSP program and the 2250 operator through use of the light pen. The subroutines involve locating a position on the screen at which the light pen is pointed and using the light pen to move a tracking symbol from one screen location to another. The subroutines are as follows:

Locate Position of Light Pen (LOCPN)
Locate Position of Light Pen on No Detect (LOCND)
Locate a Position with the Tracking Symbol (LCPOS)
Track Position of Light Pen (TRACK)
Control Light Pen Tracking (CTLTK)
Disconnect Tracking Entity (DISTE)
Convert Tracking Data (CVTTD)

LOCPN--Locate Position of Light Pen

The LOCPN subroutine displays a scanning pattern for locating the position of the light pen. Its use allows the program to identify the position of the light pen when the light pen is pointing at a blank area of the screen. Closing the light pen switch causes a light pen attention to occur. The original display is then restored. The program can access the attention information by calling the Request Attention Information (RQATN) subroutine. The coordinates returned by RQATN are in 2250 raster units.

General Form
CALL LOCPN(device,corrval)

device and corrval
are defined in "Arguments Used by Many of the Subroutines." The "corrval"

argument must identify an image entity that is being displayed.

PROGRAMMING NOTES: During the scan for the light pen, the original display is not on the screen. To locate a position that bears some relationship to the original display, therefore, the 2250 operator should point the light pen at the appropriate position on the screen and then, by an alphameric or programmed function keyboard attention, signal the program to initiate the scan.

After the scanning is begun, the LOCPN subroutine returns control to the calling program. Thus the program can continue processing while the 2250 operator is selecting a position for the light pen. When a light pen attention occurs, it is serviced by the GSP, the programmer's enable/disable status for light pen attentions is restored, and a light pen attention is available to the program. The correlation values supplied as part of the attention information are zero (see RQATN).

The x- and y-coordinates returned by the RQATN subroutine can be used as arguments for other subroutine calls, for example, MVPOS or TRACK.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value does not identify an image entity that is being displayed.
3. The "device" argument is invalid.

LOCND--Locate Position of Light Pen on No Detect

The LOCND subroutine displays a scanning pattern for locating the position of the light pen, but only if there was no light pen detect on the image entity that was being displayed and the light pen switch is closed. Except for these conditions, it functions in the same manner and provides the same services as the LOCPN subroutine.

General Form
CALL LOCND(device,corrval)

device and corrval
are defined in "Arguments Used by Many of the Subroutines." The "corrval" argument must identify an image entity that is being displayed.

PROGRAMMING NOTES: Until the light pen switch is closed the display is not affected, and scanning is not initiated. Otherwise, the programming notes for the LOCPN subroutine also apply to the LOCND subroutine.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The correlation value does not identify an image entity that is being displayed.
3. The "device" argument is invalid.

LCPOS--Locate a Position with the Tracking Symbol

The LCPOS subroutine displays a tracking symbol which the 2250 operator uses to define the absolute x- and y-coordinates of a position on the screen. Using the light pen with the switch closed, the 2250 operator moves the tracking symbol till its center point is in the desired position. He then signals the program to call CTLTK (Control Light Pen Tracking) to indicate that the desired position has been found. The coordinates of the position are returned in integer variables specified in the call to CTLTK.

General Form
CALL LCPOS(device,xstart,ystart, gca)

device and gca
are defined in "Arguments Used by Many of the Subroutines."

xstart,ystart
are constants or variables indicating the initial x- and y-coordinates for the center of the tracking symbol. These values must be real or integer type as currently defined for input data mode in the generation control area, and must be absolute coordinates.

CAUTION: If the coordinates for the initial position of the center point of the tracking symbol are off the screen, the tracking symbol is positioned on the nearest boundary of the screen.

PROGRAMMING NOTES: After the tracking symbol is displayed, the LCPOS subroutine returns control to the calling program. Thus the program can continue processing while the 2250 operator is positioning the tracking symbol. See the programming notes

for CTLTK for information about 2250 operator communication with the program.

The coordinates returned by the CTLTK subroutine can be used as arguments for other subroutine calls, for example, MVPOS and TRACK.

ERROR: The "device" argument is invalid.

TRACK--Track Position of Light Pen

The TRACK subroutine enables the programmer to define and create a tracking entity. The TRACK subroutine displays a tracking symbol which it uses to follow pen movement as the 2250 operator sketches. The sketching is displayed immediately. The programmer specifies the tracking mode, whether lines or points are to be displayed, and the smoothness of the curves displayed by means of a call to CTLTK.

General Form
CALL TRACK(device,corrval,xstart,ystart, gca)

device, corrval, and gca
: are defined in "Arguments Used by Many of the Subroutines." The "corrval" argument identifies the tracking entity being defined by this call to TRACK.

xstart,ystart
: are constants or variables indicating the starting position for the center of the tracking symbol. These values must be absolute and must agree with the input data mode (integer or real) defined in the GCA.

CAUTIONS: If the generated data fills the image construction area while the 2250 operator is sketching, the letter F (for Full) is superimposed on the tracking symbol, and the tracking symbol no longer follows the movement of the light pen.

TRACK must not be specified for a device on which an image entity is not currently being displayed. Note, however, that an image entity need not contain visible graphic data.

PROGRAMMING NOTES: After TRACK has placed the tracking symbol on the 2250 screen, it returns control to the calling program, thus enabling the calling program to continue other processing. TRACK intercepts light pen attentions on the tracking symbol, but other light pen attentions are made available to the calling program if they are enabled.

Sketching by means of TRACK is essentially image generation, and though processing may continue after a call to TRACK (see above), this processing must not include image generation in the ICA that contains the image entity being displayed. Image generation in the same ICA may be accomplished by (1) calling TMDSP, (2) calling image management and image generation subroutines for updating or image generation, and (3) reestablishing the display and tracking by a call to the EXEC subroutine. Note that, once displayed, the tracking symbol remains a part of the image entity till a call to CTLTK specifying the end function is issued.

If the ICA becomes full during tracking (indicated by the letter F in the tracking symbol), the 2250 operator should inform the program of this condition. He may do this in the same manner as he communicates tracking options to the program, as discussed under CTLTK, "Communicating Tracking Options to the Program." The program, upon being notified of this condition, should call CTLTK to remove the tracking symbol and end light pen tracking.

During the creation of a tracking entity, the tracking entity is linked to by means of graphic data that is part of the image entity being displayed. Another tracking entity should not be defined with TRACK until the end function for the first tracking entity has been indicated by means of CTLTK. If this rule is not followed, the last line or point may be lost.

If only one tracking entity is to be defined for an image entity, it is not necessary to update a linkage entity to call the tracking entity. However, if another tracking entity is defined for the image entity being displayed, the first tracking entity is "disconnected" from the image entity and will not be visible on the screen until it has been called by means of LKSUB. To avoid the GSP automatic "disconnect," the program may call DISTE (Disconnect Tracking Entity).

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The "device" argument is invalid.

CTLTK--Control Light Pen Tracking

The CTLTK subroutine, in conjunction with the TRACK subroutine, permits the 2250 operator to sketch with the light pen. CTLTK allows the program to select one of the three modes of tracking performed by TRACK. These tracking modes are described in the programming notes below. CTLTK also

enables the program to choose whether lines or points are to be used for sketching; to control the smoothness of curved lines sketched by the 2250 operator; to signal the end of light pen tracking or to fix a point and continue tracking, either in the same or a different tracking mode; and to obtain the x- and y-coordinates of the tracking symbol.

CTLTK, in conjunction with the LCPOS subroutine, permits the 2250 operator to locate a position with the tracking symbol. Unlike TRACK, LCPOS does not generate any tracking data. LCPOS and TRACK are therefore mutually exclusive. If a tracking entity is being created by means of TRACK, a call to LCPOS terminates generation of the tracking entity. Similarly, a call to TRACK following a call to LCPOS causes a tracking entity to be created from subsequent light pen attentions on the tracking symbol.

General Form
CALL CTLTK(device,trackmode,distance, plopt,xyarray)

device
is defined in "Arguments Used by Many of the Subroutines." It must be the same as the "device" argument used in a call to TRACK or LCPOS.

trackmode
is an integer constant, integer variable, or integer arithmetic expression specifying the tracking mode or end function, as follows:

0 = do not change the tracking mode
1 = position tracking mode
2 = curve tracking mode
3 = linear tracking mode
4 = end function; remove the tracking symbol and terminate light pen tracking (TRACK) or locating a position (LCPOS)

Note: The coordinates of the center of the tracking symbol are stored (see "xyarray") each time CTLTK is called. In addition, when tracking is being performed by means of TRACK, these coordinates are used to fix a point each time CTLTK is called, regardless of the "trackmode" option chosen. See the programming notes below for details about tracking modes.

distance
is an integer constant, integer variable, or integer arithmetic expression in the range 1 to 63 indicating the distance in raster units (measured along the x- or y-axis) by which the light pen must be moved before a new point or line is generated in curve tracking mode. A value of zero indicates that the prior "distance" value is not to be changed.

plopt
is an integer constant, integer variable, or integer arithmetic expression indicating whether points or lines are to be generated, as follows:

0 = do not change last setting
1 = generate lines
2 = generate points

xyarray
is a 2-element integer array where the CTLTK subroutine is to place the current x- and y-coordinates of the center point of the tracking symbol. The first element contains the x-coordinate, and the second element contains the y-coordinate, both in 2250 raster units.

PROGRAMMING NOTES: The following paragraphs discuss the tracking modes and communication of tracking options to the program by the 2250 operator.

Tracking Modes: Three tracking modes are available to the program: curve tracking, linear tracking, and position tracking. All tracking is done with the light pen switch closed. If the switch is open, the tracking symbol does not follow the movement of the pen.

Curve Tracking is used for continuous sketching of curved images. The images are displayed on the screen as they are sketched by the 2250 operator. The smoothness of the curves can be controlled by the program by means of the "distance" argument. When the 2250 operator is finished sketching, he signals the program to call CTLTK, specifying the end function ("trackmode" = 4). The tracking symbol is removed. Specifying "trackmode" = 0, 1, 2, or 3 fixes a point at the current position of the tracking symbol, even if it has been moved a distance less than the value of the "distance" argument since the last point was fixed, and allows tracking to continue in the mode specified by "trackmode".

Linear tracking is used to define points or straight lines. In this mode, the 2250 operator moves the tracking symbol with the light pen, and the line or point is not fixed till the program issues a call to CTLTK, specifying either the end function, if tracking is to be terminated, or "trackmode" = 0, 1, 2, or 3 if tracking is to be continued. If the end function is specified, the tracking symbol is removed.

The end point of a line can be made to move up or down, right or left, till it is fixed (rubber-banding).

Position tracking is used to move the tracking symbol to a new position and re-origin using that position. When the tracking symbol is in the desired position, the 2250 operator signals the program to call CTLTK, specifying the end function, if tracking is to be terminated, or "trackmode" = 0, 1, 2, or 3 if tracking is to be continued. If the end function is specified, the tracking symbol is removed.

Note: The GSP is initialized as follows:

"trackmode" = 1 (position tracking)
"distance" = 30
"plopt" = 1 (lines)

These initial values may be changed any time after GSP initialization (GSPIN) by means of CTLTK.

Communicating Tracking Options to the Program: The program using the light pen tracking subroutines LCPOS, TRACK, and CTLTK may choose its own techniques for the 2250 operator to communicate the light pen tracking options represented by the "trackmode", "distance", and "plopt" arguments. The programmed function keyboard, alphameric keyboard, or light pen may be used to communicate these options. If the light pen is used, the image entity being displayed should contain controlled entities for the various options, since controlled entities can have the attribute of light pen detection. The correlation value of the controlled entity returned in the attention information would inform the program of the option chosen. The programmer may also consider the inclusion of attentions to signal the "distance" option either by fixed amounts or by amounts indicated by means of the attentions.

ERRORS:

1. The "device" argument is invalid.
2. The "trackmode" argument is not in the range 0 to 4.
3. The "distance" argument is not in the range 0 to 63.
4. The "plopt" argument is not 0, 1, or 2.
5. The "trackmode" argument is 2, and the current "distance" value is 0.

DISTE--Disconnect Tracking Entity

The DISTE subroutine causes the specified tracking entity to be disconnected from the image entity to which it had been temporarily linked while the tracking entity was being created by light pen sketching (see TRACK).

General Form
CALL DISTE(corrval)

corrval
is defined in "Arguments Used by Many of the Subroutines." In addition, "corrval" must identify a previously defined (via TRACK) tracking entity that is in the active ICA.

PROGRAMMING NOTE: When a tracking entity is disconnected from the temporary linkage in an image entity, it is no longer displayed unless it has been linked to as a subroutine by means of LKSUB. Disconnecting a tracking entity does not delete it, and it may still be referred to in other GSP subroutine calls, e.g., LKSUB, PCOPY, DELMT.

ERRORS: The correlation value is not in the range 1 to 32767 or does not identify a tracking entity.

CVTTD--Convert Tracking Data

The CVTTD subroutine is used to convert the x- and y-coordinate data associated with a tracking entity to program coordinates. In addition, a call to CVTTD may be used to convert the x- and y-coordinate data in origin entities, attention data arrays, or xyarrays used by CTLTK to program coordinates.

General Form
CALL CVTTD(gca,{corrval|atnarray| xyarray},xarray,yarray, elcount,audit,tereturn, source)

gca and corrval
are defined in "Arguments used by Many of the Subroutines." In addition, "corrval" identifies either a tracking entity or an origin entity.

atnarray
is an integer array that has been used as the "arrayname" argument in a call to RQATN.

xyarray
is an integer array that has been used as the "xyarray" argument in a call to CTLTK.

xarray,yarray
are variables or arrays specifying where CVTTD is to place the x- and y-coordinate data. They must agree in type (real or integer) with the input data mode specified in the GCA; however, the converted data placed in "xarray" and "yarray" corresponds to the mode (absolute, incremental) in the tracking entity, origin entity, xyarray, or attention data array. Optimized tracking entities are converted to absolute program units.

elcount
is an integer constant, integer variable, or integer arithmetic expression specifying the number of elements in each of the arguments "xarray " and "yarray".

audit
is an integer variable specifying where the CVTTD subroutine is to place a positive value indicating the number of x- and y-coordinates converted. This value will not exceed the "elcount" argument value.

tereturn
is an integer variable that provides CVTTD the means for continuing the conversion of graphic data to program data. This variable must be set with a zero value on the first of a series of calls to CVTTD. If CVTTD returns a value of zero for this variable, the convert function was completed. If a non-zero value is returned, there is more graphic data to convert. In this case the "tereturn" argument must not be altered for subsequent calls to CVTTD.

source
is an integer variable that indicates whether the second argument specifies a correlation value, an attention array, or an xyarray:

1 = correlation value
2 = attention array
3 = xyarray

In addition, CVTTD sets this variable with a return code indicating the type of data returned in the "xarray" and "yarray" arguments:

1 = origin data absolute
2 = origin data incremental
3 = line data absolute
4 = line data incremental
5 = point data absolute
6 = point data incremental

CAUTIONS: The "corrval" argument, if used, must identify a previously defined tracking entity or origin entity.

The "tereturn" argument must have an initial value of zero when converting tracking entity data to program data and must not be altered in a series of calls to convert all of a tracking entity.

PROGRAMMING NOTES: If the second argument is an attention array or an xyarray, the "tereturn" argument is returned as zero, and no further conversion is required.

Since the programmer will not know how many points, lines, or origins may be sketched by the 2250 operator by means of light pen tracking, the "audit", "tereturn", and "source" arguments provide the facility to convert all of the tracking entity to program coordinate units and maintain a count of the number of these units.

The CTLTK subroutine provides the facility for switching the tracking mode while the tracking entity is being created. However, CVTTD converts only one type of graphic data during a single call. The "audit" argument specifies the number of coordinates converted. The "tereturn" argument specifies if there is more data to be converted. The "source" argument specifies the type of data placed in "xarray" and "yarray."

It is the programmer's option as to the disposition of data returned in "xarray" and "yarray". This data may be moved or saved as disk records and the same "xarray" and "yarray" may be used for subsequent calls; or new "xarray" and "yarray" arguments may be used in subsequent calls.

The x- and y-coordinates may be placed in a single array by furnishing subscripted array arguments and indicating indexing in the GCA; however, CVTTD assumes that the "elcount" argument includes the indexing factor, as in the following example:

```
xarray = array(1)
yarray = array(2)
x index = 2
y index = 2
elcount = 10
```

Unless the tracking entity data changes (from points to lines, etc.), 10 x-coordinates and 10 y-coordinates will be placed in alternate elements of the array from array(1) to array(20).

ERRORS:

1. The correlation value is not in the range 1 to 32767.

2. The "tereturn" argument is invalid.

3. The "source" argument is not 1, 2, or 3.

4. The "elcount" argument is invalid.

ENTERING DATA WITH THE PROGRAMMED FUNCTION KEYBOARD

The GSP includes one subroutine, Set Programmed Function Keyboard Lights (SPFKL), that allows the program to indicate to the 2250 operator which programmed function keys are available for use.

SPFKL--Set Programmed Function Keyboard Lights

The SPFKL subroutine turns on programmed function keyboard lights. The lights to be turned on are controlled by values in an array supplied by the programmer. All lights not specified are turned off.

General Form
CALL SPFKL(device,arrayname,elcount)

device
: is defined in "Arguments Used by Many of the Subroutines."

arrayname
: is an integer array that contains values to indicate which lights at the programmed function keyboard are to be turned on.

elcount
: is an integer constant, integer variable, or integer arithmetic expression indicating the number of elements in the array. The "elcount" argument may also be used to turn on or off all 32 lights. The values that can be assigned to "elcount" and their meanings are as follows:

-1	= All lights are turned on.
0	= All lights are turned off.
1 to 31	= The number of elements in the array. Each element contains a decimal number (0 to 31) indicating a light to be turned on.
32	= The array contains 32 elements. The first element corresponds to the first light, the second element to the second light, etc. Each element contains a 0 or 1, indicating off (0) or on (1).

ERRORS:

1. The value of the "elcount" argument is outside the range -1 to 32.

2. The "elcount" argument is 32, but array elements do not contain 0 or 1.

3. The "elcount" argument is 1 to 31, but array elements contain values outside the range 0 to 31.

4. The "device" argument is invalid.

All of the GSP subroutines (image management, image generation, attention handling, etc.) are accessed by means of the FORTRAN CALL statement. The FORTRAN compiler detects and identifies CALL statements which do not follow 1130 FORTRAN rules for CALL statement arguments.

During compilation and execution, there are no checks for an invalid number of arguments in the CALL statements. The result of this type of error is unpredictable.

During compilation there are no checks for the validity of the type of arguments (array, variable, constant, etc.). During execution, the GSP subroutines assume that the arguments provided are valid in type. If they are not the result is unpredictable.

The GSPIN (GSP Initialization) subroutine defines two error variables which will be set by GSP subroutines. The "return" variable is reset by each GSP subroutine (except GSPIN) with an integer value indicating the error detected by the GSP subroutine. The "cumulative" variable is not reset by each GSP subroutine, but contains the accumulated error indicators set by each GSP subroutine.

The FORTRAN programmer may use these variables as he deems necessary. Both error variables may be tested for a zero value (no errors) by means of the FORTRAN IF statement. The "return" variable might also be tested by the computed GO TO statement. The FORTRAN programmer cannot test the individual error indicators in the "cumulative" error variable but may call the IERRS (Interpret Errors) subroutine to have the accumulated errors printed. The error code values set in the "return" variable or printed by the IERRS subroutine are the same.

The programmer may employ FORTRAN statements to test the error variables in the initial debugging of a program using GSP. He might also retain error variable tests in a final version of the program, especially if errors might result from input from the 2250 operator.

IERRS--Interpret Errors

The IERRS subroutine produces a print-out, on the system print-out device, of the error codes that correspond to accumulated errors resulting from GSP subroutine calls.

General Form
CALL IERRS

No arguments are required.

PROGRAMMING NOTES: If no errors were detected, the output of this call is:

```
GSP ERRORS = 0
```

If any errors were detected, the output of this call is:

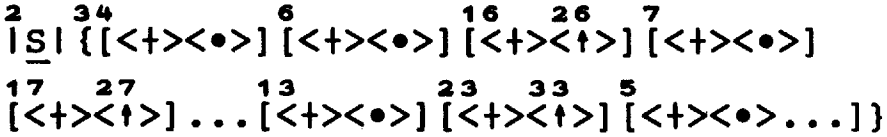
```
GSP ERRORS = N1
    .
    .
    .
GSP ERRORS = Nn
```

where N_1 is an integer value corresponding to the lowest error code identifying a detected error, and N_n is the highest. A message appears for each type of error that occurs, but only once for each type. Thus, for example, if a program contained several invalid "device" arguments, only one message for that particular kind of error would appear. The error codes in the messages are the same as those returned in the "return" variable defined by GSPIN.

The IERRS call is a debugging aid and does not provide the capability for testing and correcting errors dynamically.

The sample program in Figure 8 illustrates the use of GSP subroutines in a FORTRAN program. It assumes that the 2250 model 4 is equipped with the programmed function keyboard, in addition to the light pen. Comments in the coding explain the purpose of the program and the functions of its parts. Displays produced by the sample program are shown in Figure 9.

Figure 7 represents the graphic elements of the same program in terms of the symbols described in the second section of this publication.



Correlation Value	Element	Contents	Comments
2	Subroutine entity	Increment data to form circle with 72 lines	
3	Image entity	Total display except for circle subroutine	Circles displayed by means of linkages to circle subroutine
4	Controlled entity	Origin entity - large X	
6 - 13	Controlled entity	Origin entity - small X	
16 - 23	Controlled entity	Origin entity - named linkage entity	Linkage entity initialized as inactive
26 - 33	Linkage entity	Linkage to circle subroutine	
5	Controlled entity	Eight origin entities for labels - eight labels	Initialized as non - display

Figure 7. Graphic Elements in the Sample Program

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 1 OF 7

PROGRAMMER | DATE | CARD ELECTRO NUMBER*

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | IDENTIFICATION SEQUENCE

```
C     THIS SAMPLE PROGRAM DRAWS A CIRCLE OF 8 BASIC SIZE X'S WITH A
C     LARGE SIZE X AT ITS CENTER (STATEMENT NUMBERS 20 TO 50). DEPENDING
C     ON THE ATTENTION CREATED, THE PROGRAM WILL DO THE FOLLOWING:
C           1. LIGHT PEN DETECT ON CENTER X LABELS THE OUTER X'S FROM NUM1
C              TO NUM8 (STATEMENT 240).
C           2. LIGHT PEN DETECT ON OUTER X DISPLAYS A CIRCLE WHOSE CENTER
C              IS THE DETECTED X (STATEMENT 260).
C           3. LIGHT PEN DETECT ON CIRCLE AROUND OUTER X MAKES THAT CIRCLE
C              INVISIBLE (STATEMENT 280).
C           4. LIGHT PEN DETECT ON ANY LABEL MAKES ALL LABELS INVISIBLE
C              (STATEMENT 220).
C           5. PROGRAMMED FUNCTION KEY 1 ENDS RUN (STATEMENT 150).
C           6. PROGRAMMED FUNCTION KEY 2 MAKES ALL CIRCLES INVISIBLE
C              (STATEMENT 160).
C           7. PROGRAMMED FUNCTION KEY 3 MAKES ALL CIRCLES VISIBLE
C              (STATEMENT 190).
C                                    BEGIN PROGRAM
C                 DIMENSION STATEMENT FOR VARIOUS ARRAYS
      DIMENSION IICA(400),CIRX(8),CIRY(8),SMALX(72),SMALY(72),RLABL(32),
     XGCA(21),IATN(20),IPFK(3)
C                 INITIALIZE GSP-PRECISION, ERROR VARIABLES, AND DEVICE
      CALL GSPIN(0,0,IRET,ICUM,25,0,0,0)
C                 INITIALIZE IMAGE CONSTRUCTION AREA, CORRVAL=1.
      CALL ICAIN(1,ICA(1),ICA(400),0)
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 2 OF 7

PROGRAMMER | DATE | CARD ELECTRO NUMBER*

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | IDENTIFICATION SEQUENCE

```
C                 INITIALIZE GCA FOR INCREMENTAL INPUT DATA
      CALL GCAIN(GCA)
      CALL SDATM(GCA,2,2)
C                 COMPUTE COORDINATES FOR CIRCLE SUBROUTINE
      C=3.141596/180.0
      R=8.0
      THETA=5.0
      DO 10 I=1,72
      RADIN=THETA*FLOAT(I)*C
      SMALX(I)=R*COS(RADIN)
   10 SMALY(I)=R*SIN(RADIN)
C                 GENERATE CIRCLE SUBROUTINE, CORRVAL=2
      CALL BELMT(2,3)
      CALL PLINE(GCA,SMALX,SMALY,72)
      CALL EELMT(2)
C                 READ X AND LABEL CHARACTERS FROM CARD READER
   20 FORMAT(A4)
      READ(2,20)X
   30 FORMAT(8A4)
      READ(2,30)(RLABL(INCR),INCR=1,8)
C                 ESTABLISH COORDINATE FOR LARGE X
      CX=512.0
      CY=512.0
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

Figure 8. Sample Program (Sheet 2 of 7)

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 3 OF 7 | CARD ELECTRO NUMBER*

PROGRAMMER | DATE

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | IDENTIFICATION SEQUENCE

```
C                     INITIALIZE GCA FOR ABSOLUTE INPUT DATA
      CALL SDATM(GCA,1,1)
C                     BEGIN IMAGE ENTITY, CORRVAL=3
      CALL BELMT(3,4)
C                     GENERATE LARGE X, CORRVAL=4
      CALL BELMT(4,2)
      CALL MVPOS(GCA,CX,CY,Ø)
      CALL PTEXT(GCA,X,1,2,1)
      CALL EELMT(4)
C                     COMPUTE COORDINATES OF SMALL X'S
      R=3ØØ.Ø
      THETA=45.Ø
      DO 4Ø I=1,8
      RADIN=THETA*FLOAT(I)*C
      CIRX(I)=CX+R*COS(RADIN)
   4Ø CIRY(I)=CY+R*SIN(RADIN)
C                     GENERATE SMALL X'S, CORRVAL=6-13
      DO 5Ø I=1,8
      CALL BELMT(5+I,2)
      CALL MVPOS(GCA,CIRX(I),CIRY(I),Ø)
      CALL PTEXT(GCA,X,1,1,1)
      CALL EELMT(5+I)
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 4 OF 7 | CARD ELECTRO NUMBER*

PROGRAMMER | DATE

STATEMENT NUMBER | CONT | FORTRAN STATEMENT | IDENTIFICATION SEQUENCE

```
C                     GENERATE ORIGINS AND LINKAGES TO CIRCLE SUBROUTINE
C                     EACH IN A CONTROLLED ENTITY, CORRVAL=16-23
      CALL BELMT(15+I,2)
      CALL MVPOS(GCA,CIRX(I),CIRY(I)-8Ø.Ø,Ø)
      CALL LKSUB(2,25+I,2)
   5Ø CALL EELMT(15+I)
C                     GENERATE LABELS, CORRVAL=5
      CALL BELMT(5,2)
      DO 6Ø I=1,8
      CALL MVPOS(GCA,CIRX(I)+1ØØ.Ø,CIRY(I),Ø)
   6Ø CALL PTEXT(GCA,RLABL(I),4,1,1)
C                     END LABEL CONTROLLED ENTITY AND IMAGE ENTITY
      CALL EELMT(3)
C                     SET LABELS NON-DISPLAY
      CALL SATRB(5,-1,1)
C                     ENABLE LIGHT PEN AND PROGRAMMED FUNCTION KEYBOARD
C                     ATTENTIONS
      CALL SATNS(1,18)
C                     SET PROGRAMMED FUNCTION KEYBOARD LIGHTS
      IPFK(1)=1
      IPFK(2)=2
      IPFK(3)=3
      CALL SPFKL(1,IPFK,3)
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

Figure 8. Sample Program (Sheet 4 of 7)

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 5 OF 7 | CARD ELECTRO NUMBER*

PROGRAMMER | DATE

```
C                                 START DISPLAY
     7Ø  CALL EXEC(1,3,Ø)
C                                 REQUEST ATTENTIONS - WAIT IF NO ATTENTIONS
     8Ø  CALL RQATN(1,IATN)
         IF (IATN(1))1ØØ,1ØØ,11Ø
   1ØØ  PAUSE
         GO TO 8Ø
C                                 ATTENTION AVAILABLE - SEE IF LIGHT PEN OR PFKB
C                                 IF LIGHT PEN GO TO 21Ø ELSE 12Ø
   11Ø  IF (IATN(1)-2)12Ø,21Ø,12Ø
C                                 IF PFKB GO TO 13Ø ELSE REPEAT TEST
   12Ø  IF (IATN(1)-16)8Ø,13Ø,8Ø
C                                 IF PFKB 1 OR Ø GO TO 14Ø
C                                 IF PFKB 2 GO TO 16Ø
C                                 IF PFKB 3 OR MORE GO TO 18Ø
   13Ø  IF (IATN(3)-2)14Ø,16Ø,18Ø
C                                 IF PFKB 1 GO TO 15Ø ELSE REPEAT TEST
   14Ø  IF (IATN(3)-1)8Ø,15Ø,8Ø
C                                 TERMINATE DISPLAY
   15Ø  CALL TMDSP(1)
         CALL GSPTM
C                                 MAKE ALL CIRCLES INVISIBLE
   16Ø  DO 17Ø I=1,8
   17Ø  CALL LKSUB(2,25+I,2)
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

Figure 8. Sample Program (Sheet 5 of 7)

IBM FORTRAN Coding Form

PROGRAM FORTRAN GSP SAMPLE PROGRAM | PUNCHING INSTRUCTIONS | GRAPHIC | PUNCH | PAGE 6 OF 7 | CARD ELECTRO NUMBER*

PROGRAMMER | DATE

```
         GO TO 8Ø
C                                 MAKE ALL CIRCLES VISIBLE IF PFKB 3 ELSE REPEAT TEST
   18Ø  IF (IATN(3)-3)8Ø,19Ø,8Ø
   19Ø  DO 2ØØ I=1,8
   2ØØ  CALL LKSUB(2,25+I,1)
         GO TO 8Ø
C                                 IF LIGHT PEN DETECT ON LABEL MAKE LABELS INVISIBLE
   21Ø  IF (IATN(7)-5)23Ø,22Ø,23Ø
   22Ø  CALL SATRB(5,-1,1)
         GO TO 7Ø
C                                 IF CORRVAL LESS THAN 4 REPEAT TEST
C                                 IF CORRVAL = 4 GO TO 24Ø
C                                 IF CORRVAL GREATER THAN 4 GO TO 25Ø
   23Ø  IF (IATN(7)-4)7Ø,24Ø,25Ø
C                                 LP DETECT ON CENTER X, MAKE LABELS VISIBLE
   24Ø  CALL SATRB(5,1,1)
         GO TO 7Ø
C                                 IF CORRVAL LESS THAN 14 GO TO 26Ø ELSE 27Ø
   25Ø  IF (IATN(7)-14)26Ø,27Ø,27Ø
C                                 LP DETECT ON OUTER X, MAKE ITS CIRCLE VISIBLE
   26Ø  CALL LKSUB(2,IATN(7)+2Ø,1)
         GO TO 7Ø
```

*A standard card form, IBM electro 888157, is available for punching statements from this form

Figure 8. Sample Program (Sheet 6 of 7)

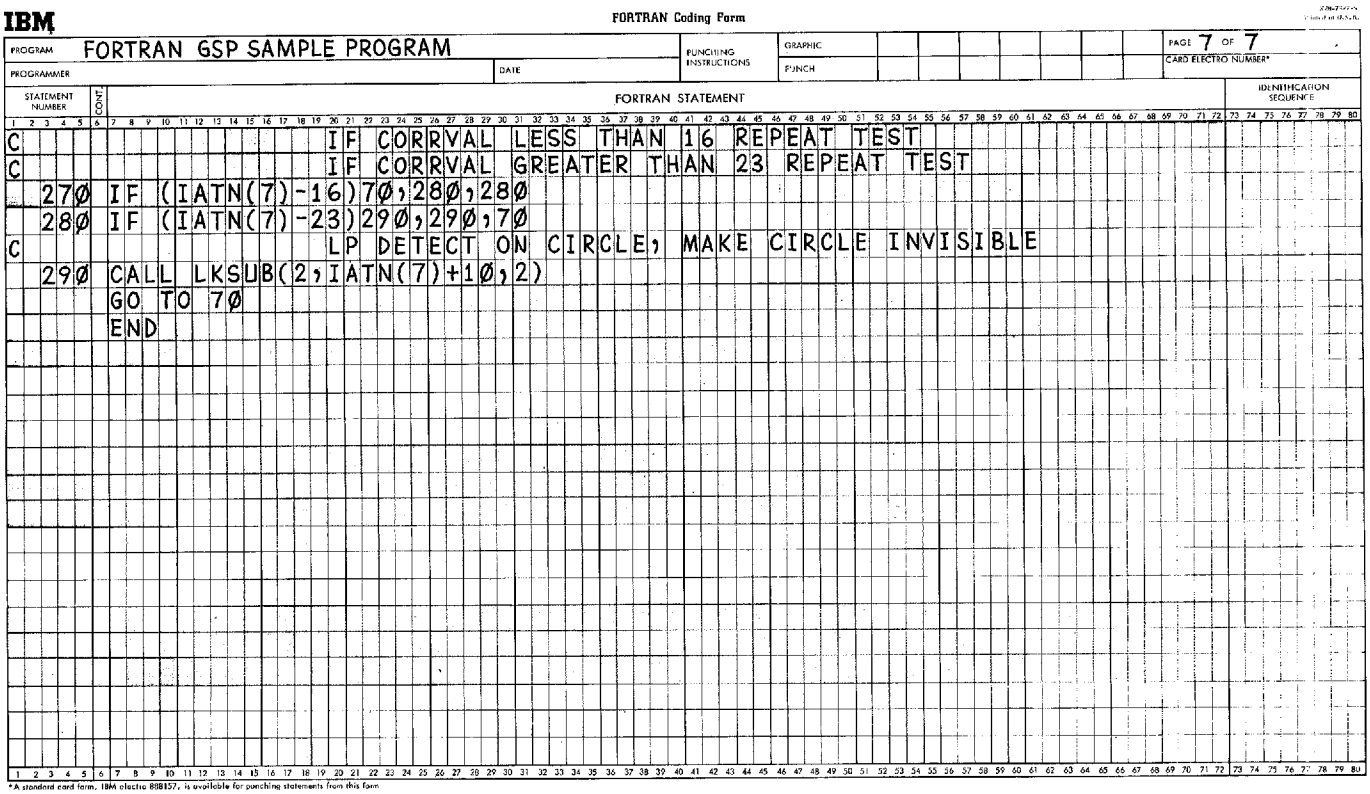
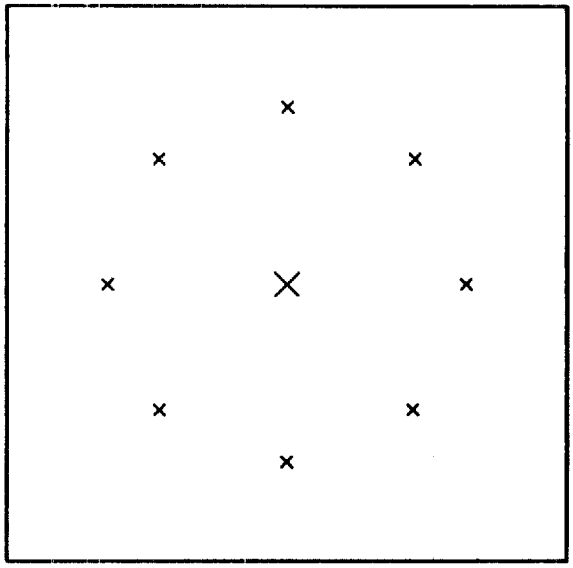
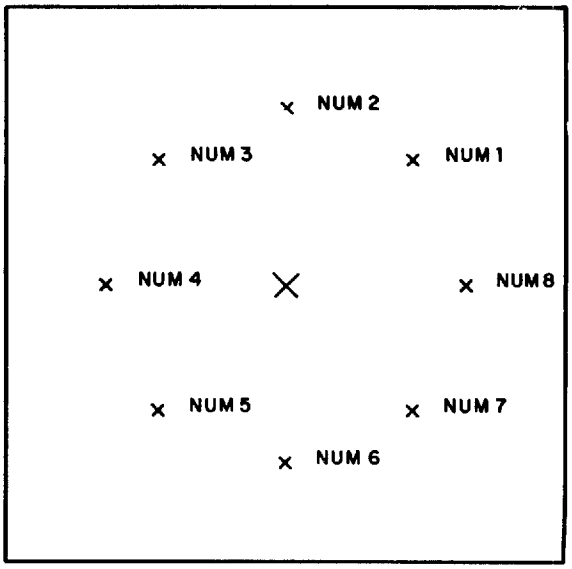


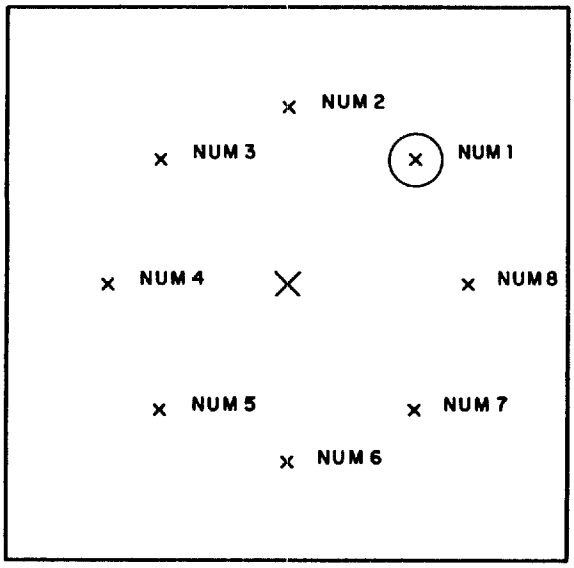
Figure 8. Sample Program (Sheet 7 of 7)



A. Display Produced After Call to EXEC



B. Display Produced After Light Pen Attention on Center X.



C. Display Produced After Light Pen Attention on X Labeled NUM1.

Figure 9. Displays Produced by Sample Program

*G2250 Control Card

The *G2250 is a Supervisor Control Record, one of which must be entered for each main-line program of a job that uses the GSP. The format for this control record is as follows:

```
cc 1
   *G2250MLNME
```

where MLNME is the name of a main-line program using GSP.

Note: If the main-line program is executed from working storage, the main-line name may be omitted.

The *G2250 control card follows the same rules as *LOCAL, *NOCAL, and *FILES control cards (see 1130 Disk Monitor publication).

XEQ Card

The following is a modification to the format rules for the XEQ card required for program execution using the GSP:

> Columns 16 and 17 of the XEQ record must contain the count of the *G2250, *LOCAL, *NOCAL, and *FILES records. The count is decimal and right justified.

GSP Subroutines as LOCALs

Any of the GSP subroutines called by the program can be defined as LOCALs. GSP internal subroutines must never be defined as LOCALs.

Core Storage Layout Requirement

The requirement described in the first section of this publication, that GSPSP must completely reside below core location 8192, can be met as follows:

1. Compile and store the main graphic processing program as a subroutine.
2. Compile and execute a main-line program that contains nothing more than a call to the main graphic processing program.

```
*MAIN-LINE PROGRAM
          CALL  SUBX
          STOP
          END
*MAIN GRAPHIC PROCESSING PROGRAM
SUBX      CALL GSPIN
          .
          .
          .
          CALL GSPTM
          STOP
          END
```

Program Links

When a program contains one or more links and it is desirable to transmit an image entity in an ICA residing in COMMON from one link to another for display, GSPSP must reside in the same core locations for all links. This can be accomplished by ensuring that the links use the same disk I/O subroutine and that their main-line programs are the same size. The above requirement, that GSPSP reside completely below 8192, must also be met. Both requirements can be met as follows:

1. Compile and store the main graphic processing programs as subroutines.
2. Compile and execute the main-line programs that contain nothing more than calls to the appropriate main graphic processing programs.

The skeleton coding in Figure 10 illustrates this approach in a program containing three links.

Note: FORTRAN DEFINE FILE statements, if used, must appear in the main-line program. Program links that must have main-line programs of equal size must therefore contain duplicate DEFINE FILE statements if these statements are needed.

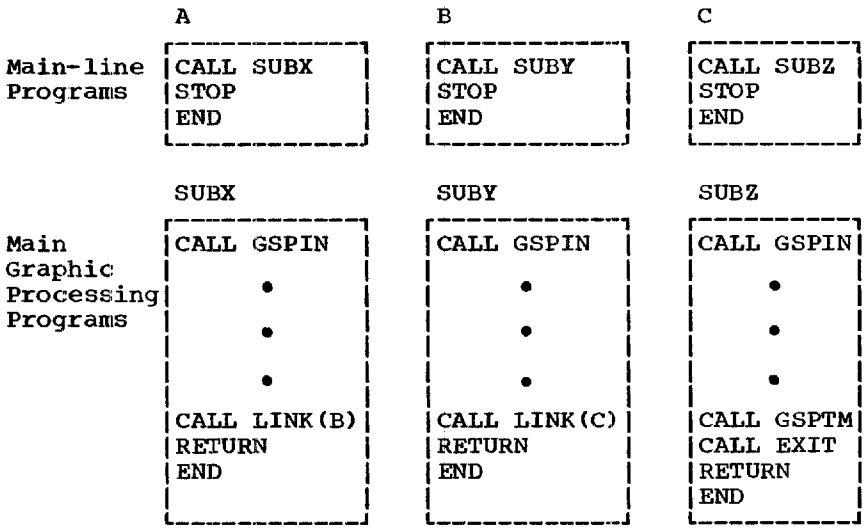


Figure 10. Program Links

This appendix provides information about the 1130/2250 assembler orders: their names, mnemonics, and the corresponding 2250 model 4 orders; coding formats; and notes pertaining to using the orders. The information about each order is presented as follows:

Assembler Order (2250 Order)

Coding format and Assembler mnemonic

Notes: Operand limitations, restrictions, significance of format and tag fields, etc.

Functional descriptions of the 2250 model 4 orders are contained in the publication IBM 1130 Component Description: IBM 2250 Display Unit Model 4, Form A27-2723. This appendix is to be used in conjunction with that publication.

Table 2 lists the codes used to identify errors encountered during assembly of the orders, the causes of the errors, and the actions taken by the Assembler.

Set Graphic Mode Vector (Set Graphic Mode)

21	27
[label]	SGMV

Notes: The graphic mode vector must be established before generating lines. Vector mode is set automatically if no graphic mode has been previously set (see the 2250 model 4 Component Description publication).

Set Graphic Mode Point (Set Graphic Mode)

21	27
[label]	SGMP

Notes: The graphic mode point must be established before generating points.

Set Character Mode Basic (Set Character Mode)

21	27
[label]	SCMB

Notes: A character mode must be established before executing a character stroke or entering a character stroke subroutine.

Set Character Mode Large (Set Character Mode)

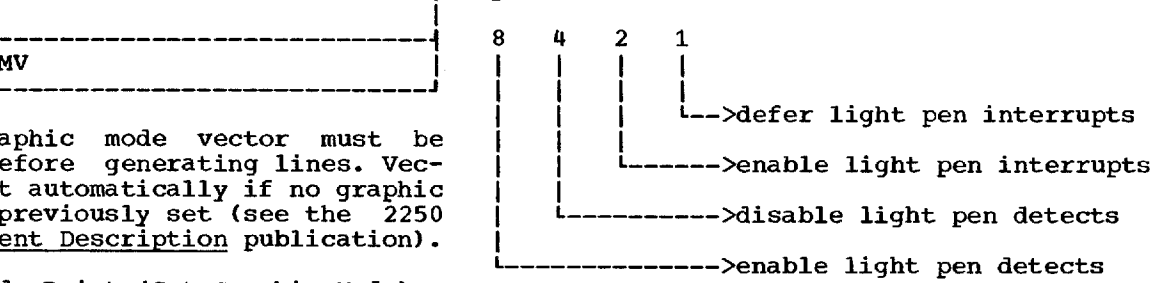
21	27
[label]	SCML

Notes: The notes for SCMB also apply for SCML.

Set Pen Mode (Set Pen Mode)

21	27	35
[label]	SPM	/hex digit or equivalent

Notes: The operand may be any hex digit or any valid absolute Assembler expression in the range 0 to F. The bit pattern of a hex digit and the effect of a 1 in a bit position are as follows:



For example, /9 (1001) defers interrupts and enables detects. Hex values 0, 3, C, and F result in no-operation.

Start Regeneration Timer (Start Timer)

21	27
[label]	STMR

Notes: The STMR order should be the first order in an order program. Its use is required for accepting keyboard attentions and setting the status of the light pen switch.

Store Revert Register (Store Revert Register)

21	27
[label]	SRVT

Notes: The SRVT order is used for return linkage when multiple levels of subroutines are used. A graphic branch indirect through the second word of the SRVT order returns control to the calling program. The SRVT order should appear in the subroutine preceding any graphic branch orders within the subroutine.

Revert (Revert)

21	27
[label]	RVT

Notes: If an order subroutine does not contain any graphic branch orders, the RVT order can be used to return control to the main order program at the order following the branch to the subroutine. If the subroutine is two or more levels from the main order program, the RVT order does not pass control to the main order program.

Graphic No-operation (Set Pen Mode)

21	27
[label]	GNOP

Notes: GNOP is assembled as an SPM order with an operand of hexadecimal 00. It can be used to reserve a single word in an order stream for later modification.

Move* Beam Incremental (Incremental XY)

21	27	35
[label]	MBI	X,Y

Notes: The x- and y-coordinates may be any valid absolute Assembler expressions, but must be in the range +63 to -64.

*The word "move" here and in following orders relates to blanked beam movement.

Draw* Beam Incremental (Incremental XY)

21	27	35
[label]	DBI	X,Y

Notes: In vector mode, beam movement is unblanked; in point mode, only the end point is unblanked. The notes for MBI also apply.

*The word "draw" here and in following orders relates to unblanked beam movement.

Move Beam Absolute (Absolute XY)

21	27	35
[label]	MBA	X,Y

Notes: The x- and y-coordinates may be any valid absolute Assembler expressions, but must be in the range 0 to 1023.

Draw Beam Absolute (Absolute XY)

21	27	35
[label]	DBA	X,Y

Notes: In vector mode, beam movement is unblanked; in point mode, only the end point is unblanked. The notes for MBA also apply.

Move Beam Absolute X (Absolute X/Y)

21	27	35
[label]	MBAX	X

Notes: The operand may be any valid absolute Assembler expression, but must be in the range 0 to 1023.

Move Beam Absolute Y (Absolute X/Y)

21	27	35
[label]	MBAY	Y

Notes: The notes for MBAX also apply here.

Draw Beam Absolute X (Absolute X/Y)

21	27	35
[label]	DBAX	X

Notes: In vector mode, beam movement is unblanked; in point mode, only the end point is unblanked. The notes for MBAX also apply.

Draw Beam Absolute Y (Absolute X/Y)

21	27	35
[label]	DBAY	Y

Notes: In vector mode, beam movement is unblanked; in point mode, only the end point is unblanked. The notes for MBAX also apply.

Move Beam Stroke (Character Stroke Word)

21	27	32	35
[label]	MBS	[R]	X,Y

Notes: The x- and y-coordinates may be any valid absolute Assembler expressions, but X must be in the range 0 to 6, and Y must be in the range 0 to 7. The x- and y-coordinates occupy a half-word. For consecutive orders, the coordinates for two orders are placed in one word. The revert function is executed if an R is placed in column 32. Character stroke orders must be executed out of line by means of a graphic branch following a set character mode order.

Draw Beam Stroke (Character Stroke Word)

21	27	32	33	35
[label]	DBS	[R]	[D]	X,Y

Notes: A D in column 33 indicates that less than normal (decreased) intensity is desired (recommended for character strokes less than 3 character units long; see the 2250 model 4 Component Description publication for details about character units). Programmed intensity provides a means of generating characters that have nearly uniform intensity for all the strokes of the character regardless of the stroke lengths. The programmer should experiment with this facility to achieve desired results. The notes for MBS also apply.

Control Stroke (Character Stroke Word)

21	27	32	35
[label]	CS	[R]	1,[data]
[label]	CS		2,[data]
[label]	CS	R	2,[data]
[label]	CS	[R]	4,[data]
[label]	CS	R	7,[data]

Notes: The first operand, which may be any valid absolute Assembler expression, has the following meanings:

1 = subscripting - the character grid is offset downward 3 vertical character units.
2 = no operation - the order performs no operation if R is not specified.
2 = null function - the order performs a null function if R is specified.
4 = superscripting - the character grid is offset upward 3 vertical character units.
7 = new line - the beam is positioned at the next line (R must be specified).

The "data" operand may be any data the programmer wants, but must not exceed 7 bits. Data exceeding the limit is truncated to the 7 low-order bits.

If revert (R) is not specified for the superscript control order, execution continues with the second word after the superscript control order. Placing a subscript control after the superscript control order gives a character stroke subroutine the capability of being executed in superscript, subscript, or normal mode if different entry points to the subroutine are defined.

Graphic Short Branch (Short Branch)

21	27	35
[label]	GSB	address

Notes: The address may be either symbolic or an absolute Assembler expression, but must have a value less than 8192. Use of the symbolic operand is restricted to referring to graphic orders that are within the same Assembler-language program.

Graphic Branch (Long Branch/Interrupt)

21	27	32	33	35
[label]	GB	[I]	[N]	address

Notes: An I in column 32 specifies an indirect branch. An N in column 33 specifies a two-word no-operation. The notes for GSB also apply, except that the address is not restricted to a value of 8191.

Graphic Branch Conditional (Long Branch/Interrupt)

21	27	32	33	35
[label]	GBC	[I]	[N]	address, condition

Notes: The condition for the branch may be one of the following:

D = branch if light pen detect
S = branch if light pen switch closed
DS or SD = branch if light pen detect and switch closed

The notes for GB also apply.

Graphic Branch External (Long Branch/Interrupt)

21	27	33	35
[label]	GBE	[N]	name

Notes: Name is the name of an external order program (subroutine). An N in column 33 specifies a two-word no operation.

Graphic Branch Conditional External (Long Branch/Interrupt)

21	27	33	35
[label]	GBCE	[N]	name, condition

Notes: The conditions for the branch are the same as those described for GBC. The notes for GBE also apply.

Graphic Interrupt (Long Branch/Interrupt)

21	27	33	35
[label]	GI	[N]	[data]

Notes: An N in column 33 specifies a two-word no-operation. Data may be a symbolic address, number, or expression. The range of numerical data or an expression, when resolved, must be +32767 to -32768. The data word may be used for any purpose.

Graphic Interrupt Conditional (Long Branch/Interrupt)

21	27	33	35
[label]	GIC	[N]	[data], condition

Notes: The conditions for the interrupt are the same as for GBC. The notes for GI also apply.

Table 2. Assembler Error Codes for 2250 Orders

Error Code	Cause	Action Taken by Assembler
W	x- or y-coordinate, or both, not within the specified range; or invalid operand.	Operand set to zero.
X	Character other than R or I in column 32; or character other than D or N in column 33.	Field set to zero.
Y	Unnecessary operand specified; or unnecessary Tag or Format field entry.	Operand ignored; Tag or Format field ignored.
Z	Invalid condition in a conditional branch or interrupt order.	Condition bits in first word set to zero.

The Assembler language programmer can use the GSP in a number of ways:

1. He may use all the GSP subroutines as defined for the FORTRAN programmer.
2. He may use only the image generation subroutines.
3. He may use only the subroutines for attention handling (except ROCOR).
4. He may use both image generation and attention handling (except ROCOR) subroutines.

In order to use all the GSP subroutines he must be thoroughly familiar with the body of this publication.

In order to use only the image generation subroutines, the programmer must use BXGEN and EXGEN as defined below under "Generating Orders Outside of an ICA."

The programmer is required to use the attention handling subroutines to allow the 2250 operator to communicate with the program, unless no operator communication is required or unless the programmer has replaced the IBM-supplied 2250 ISS with his own.

If any part of all of the GSP is being used, the first GSP subroutine called must be GSPIN. GSPIN starts the 2250 (the device is regenerating but nothing is displayed), therby activating the keyboards for attentions. In particular, the CANCEL key is activated, which the 2250 operator can use to terminate a program. If the Assembler language programmer wishes to have complete control of the program, he may call DSPYN (see below) to perform a Reset Display. This action stops regeneration and deactivates the keyboards. Since the CANCEL key is not active, the 2250 operator can not terminate the program until the device is restarted.

Since the format of the Assembler CALL statement is different from that of the FORTRAN CALL statement, the programmer must be aware of the proper calling sequence.

CALLING A GSP SUBROUTINE

In Assembler language, the calling sequence to a GSP subroutine is as follows:

```
[label] CALL  GSP subroutine name
[label] DC    address of first parameter
[label] DC    address of second parameter
          .
          .
          .
[label] DC    address of last parameter
```

For example, to call GSPIN the Assembler language programmer might code the following:

```
      CALL GSPIN
      DC   ZERO  address of integer argument
      DC   ZERO  address of real argument
      DC   RETRN address of return error field
      DC   CUMUL address of cumulative error field
      DC   UNIT1 address of logical unit 1
      DC   ZERO  address of zero field indicating no device 2
      DC   ZERO  address of zero field indicating no device 3
      DC   ZERO  address of zero field indicating no device 4
      .
      .
      .
ZERO  DC   0
RETRN DC   0
CUMUL DC   0
UNIT1 DC   25
```

The list of DCs following a call to any GSP subroutine must contain nothing but addresses, and there must be as many DCs as there are arguments in the FORTRAN argument list.

ARRAY ARGUMENTS FOR GSP SUBROUTINES

Input arrays to the GSP must be stored in column order in descending storage addresses, with the value of the first of the array's subscripts increasing most rapidly and the value of the last increasing least rapidly. In other words, arrays must be stored with element (1,1,1) in a higher core location than element (2,3,4). In scanning the array from element (1,1,1), the left indices are advanced more rapidly than those on the right. (See IBM 1130/1800 Basic FORTRAN IV Language, Form C26-3715, "Arrangement of Arrays in Storage".)

ADDITIONAL ASSEMBLER-LANGUAGE FACILITIES

Three subroutines, BXGEN, EXGEN, and IELMT, allow the Assembler-language pro-

grammer to use the GSP image generation subroutines without the necessity of using the image management subroutines, that is, to generate orders outside of an ICA. Before any call to an image generation subroutine, the programmer calls BXGEN, Begin External Generation. He can now use any of the image generation subroutines to create a graphic order program. The generated orders are placed in an area that the Assembler-language programmer designates, not in an ICA. If the programmer is using both external generation facilities and image management, i.e., an ICA, he can re-establish an ICA as the generation output area by calling EXGEN, End External Generation. IELMT, Include Element, allows the Assembler-language programmer to define to the GSP any subroutine entities that have been generated outside of an ICA.

The EXEC, SATNS, and RQATN subroutines have additional facilities for use by the Assembler-language programmer as detailed below. Finally, a 2250 ISS called DSPYN has been defined to provide the Assembler-language programmer with all the necessary 2250 I/O operations.

BXGEN, EXGEN, IELMT, EXEC, SATNS, RQATN, and DSPYN are defined below for use by the Assembler language programmer.

BXGEN--Begin External Generation

The BXGEN subroutine enables the Assembler-language programmer to use image generation subroutines to generate graphic data outside of an image construction area. It initializes the GSP in the external generation mode until a call to EXGEN has been serviced.

```
CALL    BXGEN
DC      start
DC      length
DC      gen-count
DC      non-gen-count
```

start
: is the symbolic or absolute address where the first graphic order generated is to be stored.

length
: is the address of a one-word field containing an integer value that defines the length of the area in which the image generation subroutines can store generated orders.

gen-count
: is the address of a one-word field where the total number of words generated is to be stored. This field is set to zero by BXGEN and is updated by each image generation subroutine.

non-gen-count
: is the address of a one-word field that is set to zero by BXGEN. At the completion of each image generation subroutine call it is set to zero if the generation was completed. If the generation was incomplete, it contains the amount of the "count" argument not used by the image generation subroutines (which have a "count" argument) when the area specified by the "start" and "length" arguments has been filled. Those image generation subroutines which do not have the "count" argument (PGRID, PCOPY, LKSUB, and MVPOS) store the number of words they were unable to generate.

CAUTIONS:

1. Only image generation subroutines can be called between a call to BXGEN and a call to EXGEN. The generation produced by MVPOS and LKSUB is unnamed.

2. It is the Assembler language program's responsibility to check "non-gen-count" for a non-zero value (see programming notes). No error return code is furnished if "non-gen-count" is non-zero.

3. A second call to BXGEN before a call to EXGEN is invalid.

PROGRAMMING NOTES: The Assembler-language programmer may use the external generation facility of GSP to generate, in line, graphic data for displaying if he provides those graphic orders necessary to form a complete graphic program (i.e., an STMR order at the beginning of the graphic program and a GB order to the STMR order to insure regeneration).

The Assembler-language program must take corrective action if the "non-gen-count" argument becomes non-zero. Two possible approaches are as follows:

1. The program can maintain a pointer to the last order generated by each image generation subroutine. (The "gen-count" argument provides the information needed to maintain the pointer.) Upon an occurrence of the non-zero condition, the program can (a) call EXGEN; (b) call BXGEN specifying a new generation area; (c) tie the two areas together by means of a graphic branch order; (d) reissue the last image generation call and continue. This sequence is illustrated in Figure 11.

2. Following a call to BXGEN, a call to PLINE (for example) contains arguments XARRAY, YARRAY, and COUNT, where COUNT

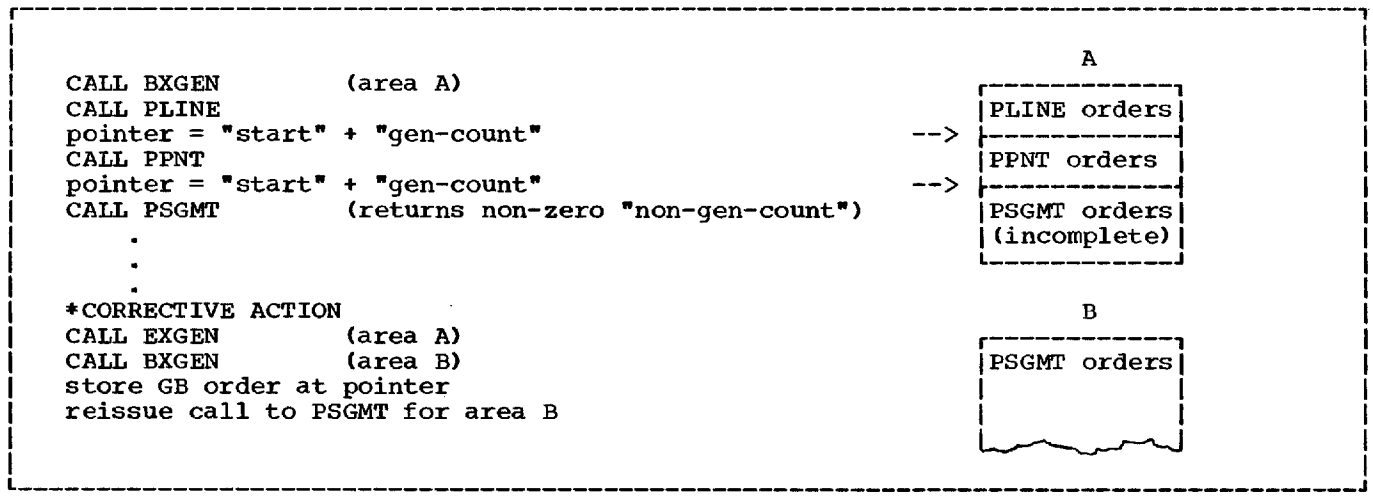


Figure 11. Overflow of External Generation Area

is equal to 100. Upon return from PLINE, "non-gen-count" has a value of 40. This value indicates that the last 40 elements of XARRAY and YARRAY could not be generated because the area specified by "start" and "length" became filled. The program can (a) save the generated orders in another area or on disk; (b) call EXGEN; (c) call BXGEN, either for the same or a new area; (d) repeat the call to PLINE after adjusting the XARRAY and YARRAY arguments to point to the sixty-first elements in the arrays, and the COUNT argument to 40.

The first approach is recommended should corrective action be needed following a call to PGRID, PCOPY, LKSUB, or MVPOS. Either approach can be used following those subroutines that have a "count" argument.

It is suggested that the external generation area be at least two words longer than what is specified by the "length" argument. Having the extra words ensures that the program can insert a branch order if the external generation area becomes full.

With careful use of BXGEN and EXGEN, the Assembler-language programmer can mix graphic data produced by the image generation subroutines and pre-assembled graphic orders moved to the same area.

ERRORS:

1. The value of the "length" argument is zero or negative.

2. A call to BXGEN was issued when the GSP was already in the external generation mode.

EXGEN--End External Generation

The EXGEN subroutine enables the Assembler-language programmer to specify the end of the external generation mode.

```
CALL    EXGEN
```

ERROR: A call to EXGEN was issued when the GSP was not in the external generation mode.

IELMT--Include Element

The IELMT subroutine identifies to the GSP any subroutine entities outside an ICA that are to be included in the active ICA.

```
CALL    IELMT
DC      corrval
DC      startaddr
DC      endaddr
```

corrval
is the address of a one-word field containing a correlation value, as defined in "Arguments Used by Many of the Subroutines."

startaddr
is the address of the first order of the included element.

endaddr
is the address of the last order of the included element.

CAUTIONS: The included element is not moved from its defined area. IELMT establishes control information pointing to its location. The included element is assumed to be a subroutine entity and may be referred to only by means of a call to LKSUB.

PROGRAMMING NOTES: Included elements may be

deleted by means of DELMT. Only the control information established by IELMT is deleted; the subroutine entity itself is not altered.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The the address for the first order is equal to or greater than the address for the last order.
3. No ICA has been defined.

EXEC--Execute Display

The EXEC subroutine causes an image to be displayed on the 2250 screen.

```
CALL    EXEC
DC      device
DC      corrval or stadd
DC      idopt
```

device and corrval
are addresses of one-word fields containing the logical unit number and correlation value, as defined in "Arguments Used by Many of the Subroutines." When "corrval" is used, it must have been previously defined as an image entity (order program).

stadd
is the address of a graphic order in an order program. It may be used instead of correlation value in Assembler-language programs, and is written as an integer constant, integer variable, or integer arithmetic expression.

idopt
is the address of a one-word field that specifies whether the second argument is a correlation value or a starting address. It is written as an integer constant or integer variable with the following meanings:

0 = the second argument is a correlation value
1 = the second argument is a starting address

CAUTION: The correlation value, when used, must have been previously defined as a completed image entity. The image entity must reside in the active ICA when the "idopt" argument equals zero.

PROGRAMMING NOTE: While an image entity is being displayed, other image entities may be generated in other ICAs. Image generation or updating of an image entity in the ICA containing the image entity being displayed must be preceded by a call to TMDSP to terminate the display.

ERRORS:

1. The correlation value is not in the range 1 to 32767.
2. The "idopt" argument is zero, but the correlation value is not defined in the active ICA as an image entity.
3. The "device" argument is invalid.

SATNS--Set Attention Status

The "attnsource" codes for this subroutine differ for the Assembler-language programmer as follows:

Code	Meaning
1	Order controlled attentions are enabled.
30	All sources except order controlled attentions are enabled.
31	All sources are enabled.

All other codes remain as described for the FORTRAN programmer.

When code 1 is specified (or code 31), an attention occurs when a Graphic Interrupt or Graphic Conditional Interrupt order is encountered. (If conditional, the condition or conditions for the interrupt must be met.) The interrupt orders are described in Appendic C.

RQATN--Request Attention Information

For order controlled attentions, RQATN places attention source code 1 in Array(1); a decimal number in the range 1 to 4 corresponding to the logical unit number in Array(2); the address of the order causing the attention in Array(11); and the address of the next sequential order in Array(12). The other elements of the array are filled with zeros.

Following an order controlled attention, the GSP restarts the display but does not redisplay the image entity (the 2250 is regenerating, but the screen is blank).

For light pen attentions, in addition to the data explained in Table 1, RQATN places the address of the order detected by the light pen in Array(11) and the address of the next sequential order in Array(12).

DSPYN--2250 I/O Routine

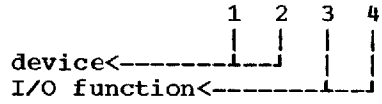
This subroutine provides the Assembler-language programmer with various I/O functions for the 2250, in addition to those provided by other GSP subroutines.

```
LIBF  DSPYN
DC    Control parameter
DC    I/O area
DC    Error routine
```

The calling sequence parameters are described in the following paragraphs.

Control Parameter

This parameter consists of four hexadecimal digits as shown below:



Device is the actual device in use. A hexadecimal 19 is required.

The I/O function digits specify the operation to be performed. The functions, their associated hexadecimal values, and the required parameters are listed and described below.

Function	Hex Code	Required Parameters*
Start Regeneration	50	Control,I/O area, Error
Set Programmed Function Lights	58	Control,I/O area
Read Status	60	Control,I/O area
No-Operation	40	Control
Reset Display	48	Control,Error
Sense Device	70	Control,Error

*Any parameter not required for a particular function must be omitted.

Start Regeneration: Starts regeneration of the display at the core address specified by the I/O area parameter. If the device is busy, regeneration is stopped and restarted at the address specified. Regeneration is stopped by any interrupt.

Should regeneration fail to start, a second attempt is made. Should this fail, DSPYN branches to the specified error routine, or to a standard error routine, described under "Error Parameter," if zero is specified for the error parameter.

The calling sequence for this function is:

```
LIBF   DSPYN
DC     /1950
DC     AREA
DC     ERROR
```

Set Programmed Function Lights: Resets the display (see below) and lights the programmed function lights specified in a two-word I/O area. The display is not restarted.

The I/O area consists of eight hexadecimal digits which represent a bit pattern to indicate the keys to be lighted. All other keys are turned off. The keys are specified by bits 0-31 as follows:

Keys 0-3	Keys 4-7	Keys 8-11	Keys 12-15	Keys 16-19	Keys 20-23	Keys 24-27	Keys 28-31

The calling sequence for this function is:

```
LIBF   DSPYN
DC     /1958
DC     AREA
```

Read Status: Causes the status of the 2250 at the time of the last attention to be transferred from the area where it was stored when the attention was recognized to a six-word area specified as the I/O area. When an attention occurs, subsequent attentions are ignored until the program executes the Read Status function. If no attention has occurred since the last Read Status function was executed, the first word of the six-word area will contain zero.

For the status format see the Component Description publication for the 2250 model 4. Since Read Status does not initiate a read, no error conditions are applicable.

The calling sequence for this function is:

```
LIBF   DSPYN
DC     /1960
DC     AREA
```

No-Operation: Provides no-operation; no input/output command is issued. The calling sequence for this function is:

```
LIBF   DSPYN
DC     /1940
```

Reset Display: Resets the address register, device status word, keyboards, all data registers, and stops regeneration.

Should Reset Display fail to effect a reset, a second attempt is made. Should this fail, DSPYN branches to the specified error routine, or to a standard error routine, described under "Error Parameter," if zero is specified for the error parameter.

The calling sequence for this function is:

```
LIBF    DSPYN
DC      /1948
DC      ERROR
```

Sense Device: Reads the device status word and places it in the accumulator.

Should the operation be unsuccessful, a second attempt is made. Should this fail, DSPYN branches to the specified error routine, or to a standard error routine, described under "Error Parameter," if zero is specified for the error parameter.

The calling sequence for this function is:

```
LIBF    DSPYN
DC      /1970
DC      ERROR
```

I/O Area Parameter

The I/O area parameter applies to the Start Regeneration, Set Programmed Function Lights, and Read Status functions. Information about this parameter is contained in the above descriptions of these three functions.

Error Parameter

The error parameter is the address of the user's routine that is to receive control in the event of an error, or is zero. When control is passed to the user's error routine, the accumulator contains a code identifying the error. If the accumulator contains zero when control is returned from the user's error routine, the function is retried. If the error code is still in the accumulator when control is returned from the user's error routine, or if zero is specified for the error parameter, the following action is taken.

A Reset Display followed by a final attempt is made to effect the desired operation. Should this fail the following information is typed out on the system print-out device:

1. Last status information
2. Function being performed
3. The error
4. Last DSW sensed
5. Address of last instruction executed

The routine then waits for either a D (for terminate with dump), R (for retry), or T (for terminate without dump) to be entered on the 1130 console keyboard. Upon sensing a D, the 2250 is reset, the PFKB lights are turned off, and the program exits. If an R is sensed, the operation is retried.

An Assembler-language programmer can generate graphic elements for use by FORTRAN programs in one of three ways:

1. By using a common ICA.
2. By using BXGEN, EXGEN, and IELMT.
3. By using preassembled order subroutines and IELMT.

If the FORTRAN program calls the Assembler-language program as a subroutine or vice versa, one of the programs will call GSPIN and define an ICA. If the GCA is passed as an argument, or if the called program sets up its own GCA, both programs can now use any of the GSP subroutines, and the elements generated will be placed in the ICA for subsequent display.

On the other hand, a FORTRAN program can call an Assembler-language program passing as an argument the ICA. The Assembler-language program can then use the external generation mode to create one or more subroutine entities and include them in the ICA (see Appendix D). The correlation values for the subroutines can either be provided by the FORTRAN program as arguments or passed back to the FORTRAN program by the Assembler-language program. Care must be exercised to ensure that these correlation values are unique within that ICA.

APPENDIX F: STANDARD 1130/2250 CHARACTER SET

The following are the characters and their decimal equivalents that can be generated for display on the 2250 by using the PTEXT or MSGIN subroutne, or can be entered from the alphameric keyboard.

Character Graphic	Decimal Equivalent	Character Graphic	Decimal Equivalent	Character Graphic	Decimal Equivalent
A	193	*d	132	6	246
B	194	*e	133	7	247
C	195	*f	134	8	248
D	196	*g	135	9	249
E	197	*h	136	*¢	74
F	198	*i	137	.	75
G	199	*j	145	<	76
H	200	*k	146	(	77
I	201	*l	147	+	78
J	209	*m	148	*|	79
K	210	*n	149	&	80
L	211	*o	150	*!	90
M	212	*p	151	$	91
N	213	*q	152	*	92
O	214	*r	153	)	93
P	215	*s	162	*;	94
Q	216	*t	163	*¬	95
R	217	*u	164	- (minus)	96
S	226	*v	165	/	97
T	227	*w	166	,	107
U	228	*x	167	%	108
V	229	*y	168	*_ (under-	109
W	230	*z	169	score)	
X	231	0	240	*>	110
Y	232	1	241	*?	111
Z	233	2	242	*:	122
*a	129	3	243	#	123
*b	130	4	244	@	124
*c	131	5	245	'	125
				=	126
				*"	127
				(space)	64

*These characters are not in the 1130 FORTRAN Source Program Character set.

Table 3 lists the dimensions and spacing of the characters produced by the 2250 character stroke subroutines. Character spacing is center to center. One raster unit (r.u.) equals approximately .012 inches. All inch dimensions are approximate.

Table 3. Character Dimensions and Spacing

Characteristics	Character Size	
	Basic	Large
Characters per line (maximum)	74	49
Lines per display (maximum)	52	35
Number of characters on display (maximum)	3,848	1,715
Spacing between characters (r.u.)	14	21
Spacing between characters (inches)	.16	.25
Spacing between lines of characters (r.u.)	20	30
Spacing between lines of characters (inches)	.24	.36
Character size (r.u. - vertical x horizontal)	14 x 10	21 x 15
Character size (inches - vertical x horizontal)	.16 x .12	.24 x .18
Subscript offset (r.u. - grid movement down)	6	9
Subscript offset (inches - grid movement down)	.07	.11
Superscript offset (r.u. - grid movement up)	6	9
Superscript offset (inches - grid movement up)	.07	.11

When more than one reference is given, the first page number indicates the major reference.

READER'S COMMENTS

IBM 1130/2250 Graphic Subroutine Package; Preliminary Specifications

C27-6934-0

Your comments will help us produce better publications for your use. Please check or fill in the items below, adding explanations and other comments in the space provided. All comments and suggestions become the property of IBM.

Which of the following terms best describes your job?

☐ Programmer	☐ Systems Analyst	☐ Customer Engineer
☐ Manager	☐ Engineer	☐ Systems Engineer
☐ Operator	☐ Mathematician	☐ Sales Representative
☐ Instructor	☐ Student/Trainee	☐ Other (explain) _____

Does your installation subscribe to the SRL Revision Service? ☐ Yes ☐ No

How did you use this publication?

- ☐ As an introduction
- ☐ As a reference manual
- ☐ As a text (student)
- ☐ As a text (instructor)
- ☐ For another purpose (explain) _____

Did you find the material easy to read and understand? ☐ Yes ☐ No (explain below)

Did you find the material organized for convenient use? ☐ Yes ☐ No (explain below)

Specific criticisms (explain below)

Clarifications on pages _____
Additions on pages _____
Deletions on pages _____
Errors on pages _____

Explanations and other comments:

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 116
KINGSTON, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
NEIGHBORHOOD ROAD
KINGSTON, N. Y. 12401

ATTN: PROGRAMMING PUBLICATIONS
DEPARTMENT 637

FOLD FOLD

Printed in U.S.A. C27-6934-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

Printed in U.S.A. C27-6934-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]