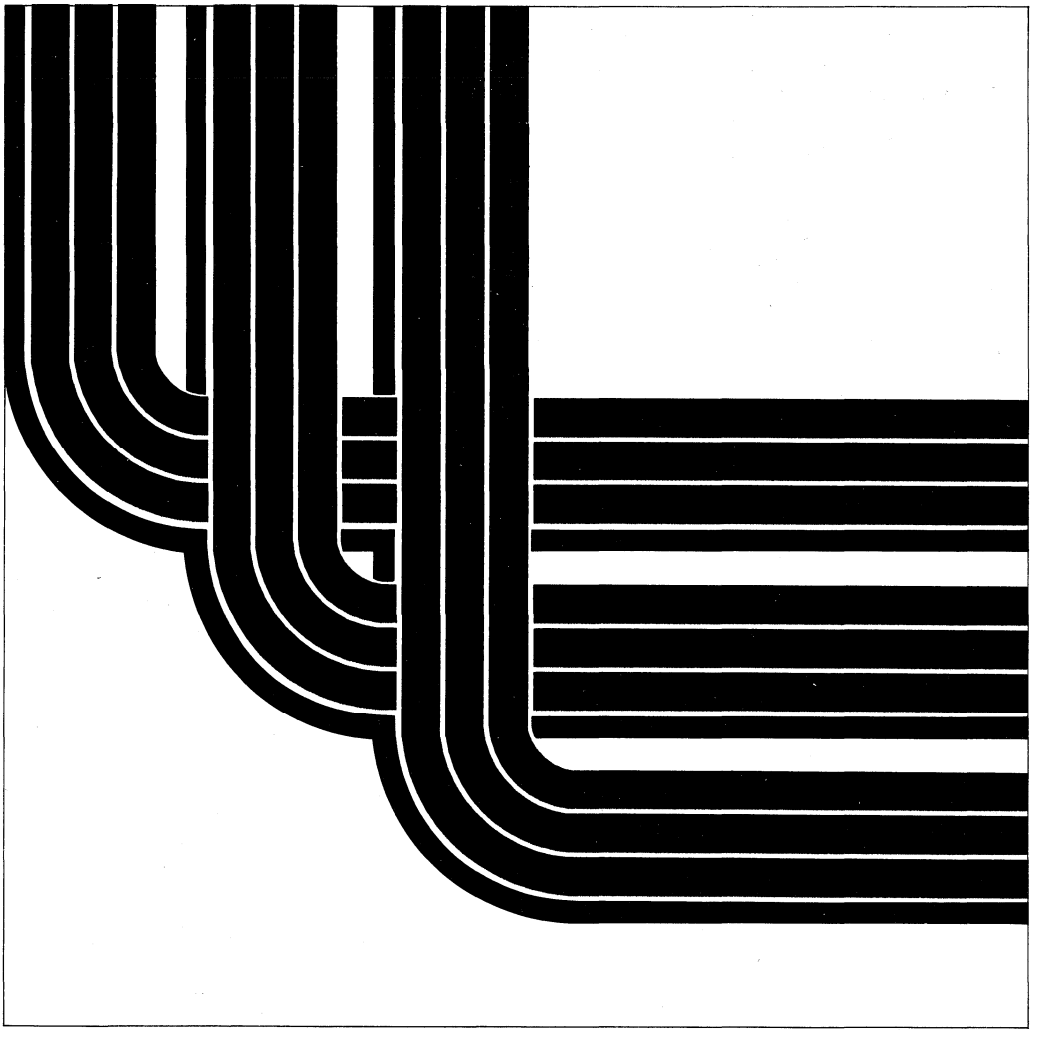
Application System/400

SC41-0537-00

Programming:
GDDM Programming Reference

Version 2



Application Development

Take Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page xi.

First Edition (May 1991)

This edition applies to the licensed program IBM Operating System/400 (Program 5738-SS1), Version 2 Release 1 Modification 0, and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the proper edition for the level of the product.

Order publications through your IBM representative or the IBM branch serving your locality. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, you may address your comments to:

Attn Department 245
IBM Corporation
3605 Highway 52 N
Rochester, MN 55901-7899

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you or restricting your use of it.

Contents

Figures

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

AD/Cycle	Application System/400
AS/400	COBOL/400
IBM	Operating System/400
OS/400	Personal System/2
RPG/400	SAA
Systems Application Architecture	System/370
400	

This publication could contain technical inaccuracies or typographical errors.

This manual may refer to products that are announced but are not yet available.

Information that has changed since Version 1 Release 3 Modification 0 is indicated by a vertical bar (|) to the left of the change.

This manual contains small programs which are furnished by IBM as simple examples to provide an illustration. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. All programs contained herein are provided to you "AS IS". THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE EXPRESSLY DISCLAIMED.

Programming Interface

The *GDDM Programming Reference* is intended to help the customer create graphic applications. The reference contains general-use programming interfaces, which allow the customer to write programs that use the services of the graphic data display manager (GDDM).

About This Manual

This manual is a reference manual for programmers to be used with the *Programming: GDDM Programming Guide*, SC41-0536. These manuals describe the application program interface (API) with the graphics capabilities of the Operating System/400 (OS/400) program. Included are detailed descriptions of all the graphics routines available in graphical data display manager (GDDM) and information about high-level language interfaces with GDDM.

This manual does not describe how to use the IBM AS/400 Business Graphics Utility (BGU), nor does it describe an interface with BGU. For information on BGU, refer to the manual *Business Graphics Utility User's Guide and Reference*, SC09-1408. You may need to refer to other IBM manuals for more specific information about a particular topic. The *Publications Guide*, GC41-9678, provides information on all the manuals in the AS/400 library.

For a list of publications related to this manual, see the "Bibliography."

Who Should Use This Manual

This manual is to be used by AS/400 application programmers responsible for creating graphics applications. Programmers who have used GDDM or GDDM-presentation graphics feature (GDDM-PGF) on the IBM System/370 computer can use this manual to familiarize themselves with differences between the System/370 and the AS/400 systems.

To use this manual, you should understand the concepts of the IBM Operating System/400 licensed program and the AS/400 system.

You should know how to write, debug, compile, and run programs in one of the following programming languages supported by the AS/400 system: BASIC, COBOL/400, Pascal, PL/I, or RPG/400 language.

Throughout this manual, the term *personal computer* applies to any IBM personal computer that uses work station function (WSF), work station emulation (WSE), or 5250 emulation.

Chapter 1. Introduction

Overview of OS/400 Graphics

Operating System/400* (OS/400*) Graphics includes both Graphical Data Display Manager (GDDM) and Presentation Graphics routines. GDDM is a means of displaying, printing, or plotting pictures. Presentation Graphics routines are a means of displaying, printing, or plotting business charts.

You can write and run OS/400 Graphics application programs using any model of the AS/400* system (it must have the OS/400 program installed).

Although you can write and compile the programs on any work station that has been described to the system, only the following devices can be used to display graphics:

- IBM* personal computer with work station function (WSF)
- IBM personal computer with work station emulation (WSE)
- 5292 Model 2
- IBM personal computer with 5250 emulation

In this manual the term "graphics work station" means one of these devices.

These plotters can be attached to graphics work stations:

- IBM 6180 Plotter
- IBM 6182 Plotter
- IBM 6184 Plotter
- IBM 6185 Plotter
- IBM 6186-1 Plotter
- IBM 6186-2 Plotter
- IBM 7371 Plotter
- IBM 7372 Plotter

Graphics can be printed on these SNA character string (SCS) devices:

- IBM 4214 Printer
- IBM 4234-2 Printer
- IBM 5224 Printer
- IBM 5225 Printer

You can also use any intelligent printer data stream (IPDS) device that supports graphics, including:

- IBM 3812 Printer
- IBM 3816 Printer
- IBM 4028 Printer
- IBM 4224 Printer

It is also possible to send a graphics data format (GDF) file (the internal data GDDM interprets to draw the picture) to other systems. The device receiving the graphics data must have the software necessary to interpret the data.

To use OS/400 Graphics, you code CALL statements in application programs written in a high-level language. The parameters you pass with the call determine what

OS/400 Graphics does. You must specify parameters of the correct data type to be used with graphics. For specific information, see "High-Level Languages."

The data you display graphically can come from any source normally available to your programs:

- Database files on your system
- Input from interactive display stations
- Communications lines connecting your system with other systems
- Device files (such as diskette files or tape files)
- Data coded directly in your programs

System Requirements for Using GDDM and Presentation Graphics Routines

- GDDM and Presentation Graphics routines are part of OS/400 Graphics. The IBM-supplied library QGDDM must be installed on your system for you to use OS/400 Graphics in Pascal or PL/I and for you to use graphics symbol sets in any language. To install library QGDDM, refer to the *Licensed Programs and New Release Installation Guide*.
- You cannot call GDDM or Presentation Graphics routines directly from the work station (for example, from the command entry display). You must code graphics calls in a high-level language (BASIC, the COBOL/400* language, Pascal, PL/I, or the RPG/400* language), then run the program.
- When you are using Pascal, PL/I, or graphics symbol sets, library QGDDM must be in your library list or in the library list named in the job description of a batch job used to compile or run graphics programs.

High-Level Languages

On the AS/400 system you use GDDM and Presentation Graphics routines by calling (CALL statement) graphics routines from your programs. The parameters that you pass with the calls *must* be of specific data types.

OS/400 Graphics uses the following data types:

- Four-byte binary integers
- Floating-point numbers
- Character strings
- Arrays of the above data types

If the data types of the parameters passed by your program do not match those required by OS/400 Graphics, undesired results, possibly a function check, can occur.

The following sections describe how to code the calls and data types for these high-level languages:

- IBM AS/400 BASIC
- IBM SAA* AD/Cycle* COBOL/400 programming language
- IBM AS/400 Pascal
- IBM AS/400 PL/I
- IBM SAA AD/Cycle RPG/400 programming language

Also, you can use DDS (data description specifications) to describe physical files whose fields are used as variables in your programs. A section is included to describe how you can define fields that are valid for the (nonarray) parameters on calls to graphics routines.

BASIC

Internally, IBM AS/400 BASIC uses a form of array that is incompatible with the form required by OS/400 Graphics. Therefore, you must use CALL GDDM to start any OS/400 Graphics routine that requires array parameters. Another advantage of using CALL GDDM is the additional error handling provided to help you diagnose data type errors.

AS/400 BASIC does not support character strings longer than 255 characters, thus limiting the use of OS/400 Graphics routines that can accept longer strings (such as GSIMG to draw graphics images).

Calling Graphics Routines in BASIC

In this manual, the syntax for GSCHAR, which is representative of the other OS/400 Graphics routines, is as follows:

```
GSCHAR(x,y,length,string)
```

In BASIC, the CALL statement for an OS/400 Graphics routine would look like:

```
CALL GDDM('GSCHAR',X,Y,LENGTH,STRING$)
```

where the name of the graphics routine is passed as a parameter in the same way as all parameters shown in parentheses in this book. The parameters passed to GDDM must be of the appropriate data type, as described below.

Data Types in BASIC

In BASIC, the name of the OS/400 Graphics routine is passed as a parameter on the CALL statement. It is easy to specify it as a character constant. For the other parameters passed to GDDM, you must use specific data types. For example, in

```
CALL  GDDM('GSCHAR',X,Y,LENGTH,STRING$)
```

the parameters must have the following data types:

- X and Y must be floating-point numbers
- LENGTH must be a 4-byte binary integer
- STRING must be a character string

The following describes how to specify these data types in BASIC:

- *Four-byte binary integers*: You can specify either a variable name or a numeric literal with no decimal value specified (such as 1).

  When a parameter is returned to your program by GDDM, you must specify a variable, not a literal, to receive the value returned by GDDM. This most frequently occurs on the query routines.

  When specifying an integer variable, the INTEGER statement is used, as in the following example:

  ```
  INTEGER LENGTH
  ```

  where the variable LENGTH becomes a 4-byte binary integer.

- *Floating-point numbers*: You can specify either a variable name or a numeric constant with a decimal value specified (such as 1.0 or 4.5). A numeric constant like 1E0 is treated as a floating-point number. Numeric constants like 1 are treated as *integers* and would cause an error if passed when a floating-point number is required.

  When a parameter is returned to your program by GDDM, you must specify a variable, not a constant, to receive the value returned by GDDM. This most frequently occurs on the query routines.

When specifying a variable name on the CALL statement, specify the DECIMAL statement earlier in the program. In BASIC, numeric variables default to DECIMAL unless you specify an INTEGER statement for them. Specifying DECIMAL ensures that the correct data type is passed to GDDM. The DECIMAL statement is used in this manual for this reason. The following example shows the DECIMAL statement:

```
DECIMAL  X
```

- *Character strings*: You can use a variable name ending in $ or a character constant as a parameter on the CALL. One exception is when a parameter is returned to your program by GDDM. In this case, you must specify a variable, not a constant, to receive the value returned by GDDM. This most frequently occurs on the query routines.

  When specifying a variable name, for a variable length of more than 18, you must specify a DIM statement, as follows:

  ```
  DIM  STRING$*20
  ```

  where the variable STRING$ is 20 bytes long.

- *Arrays of 4-byte binary integers*: Specify the INTEGER statement as in the following example:

  ```
  OPTION BASE 1
  INTEGER  ARRAY1
  DIM  ARRAY1(4)
  ```

  where the array named ARRAY1 has four elements that are 4-byte binary integers.

- *Arrays of floating-point numbers*: Specify the DECIMAL statement as in the following example:

  ```
  OPTION BASE 1
  DECIMAL  ARRAY2
  DIM  ARRAY2(4)
  ```

  where the array named ARRAY2 is an array with four elements that are floating-point numbers.

- *Arrays of character strings*: Use an array name ending in $. You must specify a DIM statement, as follows:

  ```
  OPTION BASE 1
  DIM  ARRAY$(4)*20
  ```

  where the array ARRAY$ has four elements, each 20 bytes long.

  Note: When a character array is used as a parameter that is returned by GDDM, the exact length must be specified in a DIM statement. For examples, see the descriptions of DSQDEV and GSQSS.

The COBOL/400 Language

The COBOL/400 language does not support floating-point data. Therefore, you must use CALL GDDM to invoke any graphics routine that requires floating-point parameters. Another advantage of using CALL GDDM is the additional error handling provided to help you diagnose data type errors.

Calling Graphics Routines in the COBOL/400 Language

In this manual, the syntax for GSCHAR, which is representative of the other OS/400 Graphics routines, is as follows:

```
GSCHAR(x,y,length,string)
```

In the IBM COBOL/400 language, the CALL statement is as follows:

```
CALL "GDDM" USING GSCHAR,X,Y,LEN,STR.
```

where GSCHAR, X, Y, LEN, and STR are program variables defined as shown in "Data Types in the COBOL/400 Language."

Data Types in the COBOL/400 Language

In the COBOL/400 language, the name of the OS/400 Graphics routine is passed as a parameter on the CALL statement. The name of the OS/400 Graphics routine must be specified in the DATA DIVISION as a data item 8 bytes long. For example:

```
77 GSCHAR PIC X(8) VALUE "GSCHAR".
```

For the other parameters passed to GDDM, you must use specific data types. For example, in

```
CALL "GDDM" USING GSCHAR,X,Y,LEN,STR.
```

the parameters must have the following data types:

- X and Y must be floating-point numbers
- LENGTH must be a 4-byte binary integer
- STR must be a character string

The following describes how to specify these data types in the COBOL/400 language:

- *Four-byte binary integers*: In the COBOL/400 language, for program-described variables, use PICTURE S9(n) COMPUTATIONAL-4, where n must be 5 through 9. For example:

  ```
  03 LEN PIC S9(5) COMP-4.
  ```

- *Floating-point numbers*: In the COBOL/400 language, floating-point numbers cannot be defined. Therefore, you should specify them as packed or zoned decimal variables. For packed variables, use PICTURE S9(n)V9(m) COMPUTATIONAL-3 where n plus m is less than or equal to 18. For zoned, use PICTURE S9(n)V9(m) DISPLAY where n plus m is less than or equal to 18.

  For packed:

  ```
  03 X PIC S9(5)V9 COMP-3.
  03 Y PIC S9(5)V9 COMP-3.
  ```

  For zoned:

  ```
  03 X PIC S9(5)V9 DISPLAY.
  03 Y PIC S9(5)V9 DISPLAY.
  ```

  Use zoned decimal data if your program is to prompt the work station user for data to be displayed (this avoids the necessity of a conversion). Use packed decimal data to reduce the storage requirements of your program.

- *Character strings*: In the COBOL/400 language, for program-described variables, use PICTURE X(n) DISPLAY where n is the length of the largest character string to be moved into the variable. In the COBOL/400 language, character variables default to DISPLAY. This manual shows DISPLAY to ensure that the correct data type is passed to GDDM. For example:

```
03  STR  PIC X(10)  DISPLAY.
```

- *Arrays of 4-byte binary integers*: Define a one-dimensional table using the OCCURS clause in the Working-Storage Section as follows:

```
01  X-VALUES.
    03  X-VAL  OCCURS  n  TIMES  PIC  S9(5)  COMP-4.
```

  where X-VAL is the name of the parameter you pass to GDDM, n is the number of elements in the array, and X-VAL is the name by which you refer to elements of the array.

  Note: If you pass a 01 level name to OS/400 Graphics, results are unpredictable. You must pass the data name for which the OCCURS clause is specified.

- *Arrays of floating-point numbers*: Define a one-dimensional table using the OCCURS clause in the Working-Storage Section as shown in the following examples:

```
01  X-VALUES.
    03  X-VAL  OCCURS  n  TIMES  PIC  S9(5)V9  COMP-3.
```

  or

```
    03  X-VAL  OCCURS  n  TIMES  PIC  S9(5)V9  DISPLAY.
```

  where X-VAL is the name of the parameter you pass to GDDM, n is the number of elements in the array, and X-VAL is the name by which you refer to elements of the array. For the PICTURE clause, follow the same rules as for floating-point numbers.

  Note: If you pass a 01 level name to OS/400 Graphics, results are unpredictable. You must pass the data name for which the OCCURS clause is specified.

- *Arrays of character strings*: Define a one-dimensional table using the OCCURS clause in the Working-Storage Section as follows:

```
01  CHARACTER-ARRAY.
    03  STR  OCCURS  n  TIMES  PIC  X(m)  DISPLAY.
```

  where STR is the name of the parameter you pass to GDDM, n is the number of elements in the array, STR is the name by which you refer to elements of the array, and m is the length of the strings in the array.

  Note: If you pass a 01 level name to OS/400 Graphics, results are unpredictable. You must pass the data name for which the OCCURS clause is specified.

PL/I

Calling Graphics Routines in PL/I

To call OS/400 Graphics routines in IBM AS/400 PL/I programs, you cannot use CALL GDDM; you must call OS/400 Graphics routines directly.

In this manual, the syntax for GSCHAR, which is representative of the other OS/400 Graphics routines, is as follows:

```
GSCHAR(x,y,length,string)
```

To call OS/400 Graphics routines in PL/I programs, use a CALL like the following:

```
CALL  GSCHAR(X,Y,LEN,STR);
```

where GSCHAR is an entry point declared elsewhere in the program and X, Y, LEN, and STR are program variables defined as shown in "Data Types in PL/I."

Data Types in PL/I

In PL/I, you need two sets of declarations:

- ENTRY declarations for each graphics call used in the program
- Data type declarations for each parameter you pass as a variable name on graphics calls in the program

ENTRY Declarations: To specify ENTRY declarations for graphics calls in a PL/I program, you can specify a declaration for each graphics call, as follows:

```
DECLARE  GSCHAR
         EXTERNAL
         ENTRY  (FLOAT  DECIMAL (6),
                 FLOAT  DECIMAL (6),
                 FIXED BINARY (31),
                 CHAR  (*))
         OPTIONS(ASM);
```

where the data types, in the ENTRY declaration above, should match in number and type the parameters that will be passed with the call. In the ENTRY declaration, use FLOAT DECIMAL (6) for floating-point numbers; use FIXED BINARY (31) for 4-byte binary integers; use CHAR (*) for character strings. Also, OPTIONS(ASM) is required if, and only if, arrays or character strings are used in the call.

Instead of declaring an ENTRY declaration for each graphics call you intend to use, you can include one or more of the sets of PL/I ENTRY declarations that IBM supplies (in file QATTPL1 in library QUSRTOOL). These declarations have parameter descriptions that match exactly the parameter specifications in the descriptions of the CALL statements. To include these IBM-supplied declarations in your PL/I program, specify INCFILE(QUSRTOOL/QATTPL1) on the Create PL/I Program (CRTPLIPGM) command and do the following in your source program:

- If you are only using GDDM routines and not Presentation Graphics routines, specify:

  ```
  %INCLUDE  SYSLIB  (ADMUPLNB);
  ```

- If you are using both GDDM routines and Presentation Graphics routines, specify:

  ```
  %INCLUDE  SYSLIB  (ADMUPLNO);
  ```

As an alternative, you can use the following in any combination:

- For GDDM routines beginning with the letter A, specify:

```
%INCLUDE  SYSLIB  (ADMUPINA);
```

- For routines beginning with the letter C (the Presentation Graphics routines), specify:

```
%INCLUDE  SYSLIB  (ADMUPINC);
```

- For GDDM routines beginning with the letter D, specify:

```
%INCLUDE  SYSLIB  (ADMUPIND);
```

- For GDDM routines beginning with the letter F, specify:

```
%INCLUDE  SYSLIB  (ADMUPINF);
```

- For GDDM routines beginning with the letter G, specify:

```
%INCLUDE  SYSLIB  (ADMUPING);
```

It is convenient to specify these include statements at the end of your PL/I program so that line numbering in the compiler listing more closely matches the line numbering in your source program.

Data Type Declarations: When you specify numeric or character constants on calls to OS/400 Graphics routines, you need not use data type declarations.

For program variables used on calls to OS/400 Graphics, you need data type declarations as follows:

- *Four-byte binary integers.* In PL/I, for program-described variables, use a declaration like the following:

```
DECLARE  LEN  BINARY  FIXED(31);
```

- *Floating-point numbers.* Data type declarations for floating-point numbers should match the ENTRY declarations you use. If they do not match, PL/I converts them to the correct format. If they do match, no conversion is required and better performance is obtained.

  When using IBM-supplied ENTRY declarations, your declaration should look like the following:

```
DECLARE  X  FLOAT  DECIMAL(6);
```

  When specifying your own ENTRY declarations, use a declaration like one of the following:

  - For floating-point decimal variables:

```
DECLARE  X  FLOAT  DECIMAL(n);
```

    where X is a variable name and n is 1 through 6 (should be the same as in your ENTRY declaration).

  - For floating-point binary variables:

```
DECLARE  X  FLOAT  BINARY(n);
```

    where X is a variable name and n is 1 through 24 (same as in your ENTRY declaration).

- *Character strings*: In PL/I, for program-described variables, use a declaration like the following:

  ```
  DECLARE  STR  CHARACTER(n);
  ```

  where STR is the name of the variable and n is the number of bytes in the variable.

  Note: You must not specify the VARYING attribute.

- *Arrays of 4-byte binary integers*: Declare an array as follows (for an array of four 4-byte binary integers):

  ```
  DECLARE  ARRAY1(4)  STATIC  FIXED  BINARY(31)  INIT(1,2,3,4);
  ```

  where ARRAY1 is the parameter you pass with the call, 4 is the number of elements in the array, and INIT specifies the initial values of the array (optional). OS/400 PL/I requires STATIC when you use INIT.

- *Arrays of floating-point numbers*: Declare an array as follows:

  ```
  DCL  ARRAY1(4)  STATIC  FLOAT  DEC(n)  INIT(1.5, 2.2, 3.6, 4.0);
  ```

  where ARRAY1 is the parameter you pass with the call, 4 is the number of elements in the array, n is from 1 through 6, and INIT specifies the initial values of the array (optional). PL/I requires STATIC when you use INIT.

- *Arrays of character strings*: Declare an array as follows:

  ```
  DCL  ARRAY1(4)  STATIC  CHAR(n)  INIT('A','B','C','D');
  ```

  where ARRAY1 is the parameter you pass with the call, 4 is the number of elements in the array, n is the length of the elements of the array (in this instance 1 would be enough), and INIT specifies the initial values of the array (optional). PL/I requires STATIC when you use INIT.

  Note: You must not specify the VARYING attribute.

Pascal

Calling Graphics Routines in Pascal

When calling OS/400 Graphics routines in IBM AS/400 Pascal programs, you cannot use CALL GDDM; you must call OS/400 Graphics routines directly.

In this manual, the syntax for GSCHAR, which is representative of other OS/400 Graphics routines is as follows:

```
GSCHAR(x,y,length,string)
```

To call OS/400 Graphics routines in Pascal, use a call like the following:

```
GSCHAR(X,Y,LEN,STR);
```

where GSCHAR is an entry point declared elsewhere in the program and X, Y, LEN, and STR are program variables defined as shown in "Data Types in Pascal."

Data Types in Pascal

In Pascal, you need three sets of declarations:

- PROCEDURE declarations for each graphics call used in the program.
- TYPE declarations for each variable type you use which is not a predefined Pascal type.
- VAR declarations for each parameter you pass as a variable name on graphics calls in the program.

Procedure Declarations: To specify PROCEDURE declarations for graphics calls in a Pascal program, you can specify a declaration for each graphics call as follows:

```
PROCEDURE GSCHAR(VAR X,Y : SHORTREAL;
                 VAR LEN : INTEGER;
                 VAR STR : CHARARR__132);
                 NONPASCAL;
```

The data types in the PROCEDURE declaration must match in number and type the parameters that will be passed with the call. In the PROCEDURE declaration, use SHORTREAL for floating point numbers; use INTEGER for 4-byte binary integers; use CHARARR__n (in this case 132) for character strings. Also, the reserved word NONPASCAL is required on all OS/400 Graphics routine declarations.

Instead of declaring a PROCEDURE declaration for each graphics call you intend to use, you can include one or more of the sets of Pascal PROCEDURE declarations that IBM supplies (in file QATTPAS in library QUSRTOOL). These declarations have parameter descriptions that match exactly the parameter descriptions of the CALL statements. To include these IBM-supplied declarations in your Pascal program, do the following:

- If you are using only GDDM routines and not Presentation Graphics routines, specify :

  ```
  %INCLUDE QATTPAS(ADMUSLNB);
  ```

- If you are using both GDDM routines and Presentation Graphics routines, specify:

  ```
  %INCLUDE QATTPAS(ADMUSLNO);
  ```

As an alternative, you can use the following in any combination:

- For GDDM routines beginning with the letter A, specify:

  ```
  %INCLUDE QATTPAS(ADMUSINA);
  ```

- For routines beginning with the letter C (the Presentation Graphics routines), specify:

  ```
  %INCLUDE QATTPAS(ADMUSINC);
  ```

- For GDDM routines beginning with the letter D, specify:

  ```
  %INCLUDE QATTPAS(ADMUSIND);
  ```

- For GDDM routines beginning with the letter F specify:

  ```
  %INCLUDE QATTPAS(ADMUSINF);
  ```

- For GDDM routines beginning with the letter G, specify:

  ```
  %INCLUDE QATTPAS(ADMUSING);
  ```

In Pascal programs, it is required to specify that these include statements be specified immediately after the VAR declarations, and before any FUNCTION declarations, or at the beginning of the program.

TYPE Definitions: To specify TYPE definitions for graphics calls containing user-defined types in a Pascal program, you can specify a definition for each user-defined type required. For the GSCHAR PROCEDURE declaration, on page 1-10, the following would be a required TYPE definition:

```
TYPE
  CHARARR_132 = PACKED ARRAY[1..132] OF CHAR;
```

Instead of defining a TYPE definition for each user-defined type required for graphics calls you intend to use, you can include the TYPE definitions that IBM supplies (in file QATTPAS in library QUSRTOOL). This include statement has TYPE definitions that match exactly those user-defined types used in the IBM-supplied PROCEDURE declarations. To use these IBM-supplied TYPE definitions in your Pascal program, do the following.

```
TYPE
  %INCLUDE QATTPAS(ADMUSTNO);
```

In Pascal programs, it is required that this include statement be specified within the TYPE definitions of your program.

VAR Declarations: Pascal requires that all non-Pascal procedures pass parameters only by reference. Because of this requirement, all OS/400 Graphics routine parameters must be type VAR in Pascal programs. Numeric or character constants on calls to OS/400 Graphics routines are not allowed.

For program variables used on calls to OS/400 Graphics, you need VAR declarations as follows:

- *Four-byte binary integers*: In Pascal, for program-described variables, use a declaration like the following:

  ```
  VAR
    LEN : INTEGER;
  ```

- *Floating-point numbers*: Data type declarations for floating point variables must match the PROCEDURE declarations you use. If they do not match, Pascal will generate a compiler error and the Create Pascal Program (CRTPASPGM) command will fail. In Pascal, for program-described variables, use a declaration like the following:

  ```
  VAR
    X : SHORTREAL;
  ```

- *Character strings*: Data type declarations for character strings must match the PROCEDURE declarations you use. If they do not match, Pascal will generate a compiler error and the CRTPASPGM (Create Pascal Program) command will fail. In Pascal, for program-described variables, use a declaration like one of the following:

  ```
  VAR
    STR : ARRAY[1..n] OF CHAR;
    STR : PACKED ARRAY[1..n] OF CHAR;
    STR : STRING(n);
    STR : CHARARR_n;
    STR : ALFA;
    STR : ALPHA;
  ```

  where STR is the name of the variable, and n is the number of bytes in the variable. As noted above, the PROCEDURE declaration must match the variable declaration exactly. In the IBM-supplied PROCEDURE declarations, character strings are defined as either CHARARR_n or ALFA. ALFA is a predefined type used for 8-character strings, and is equivalent to PACKED ARRAY [1..8] OF CHAR. CHARARR_n is used for varying length strings, where n is the length of the character string (typically 132), and CHARARR_n is a user-defined type declared in the IBM-supplied TYPE definitions (in member ADMUSTNO, file QATTPAS, library QUSRTOOL) as in the following:

  ```
  TYPE
    CHARARR_n = PACKED ARRAY[1..n] OF CHAR;
  ```

- *Arrays of 4-byte integers*: Declare an array as one of the following (for an array of five 4-byte binary integers):

```
VAR
  ARRAY1 : ARRAY[1..5] OF INTEGER;
  ARRAY1 : INTARR_5;
```

  where ARRAY1 is the name of the variable, 1..5 is the range of the array, and INTEGER is the type of the element in the array. In the IBM-supplied PROCEDURE declarations, integer arrays are defined as INTARR_n, where n is the length of the array, and INTARR_n is a user-defined type declared in the IBM-supplied TYPE definitions (in member ADMUSTNO, file QATTPAS, library QUSRTOOL) as in the following:

```
TYPE
  INTARR_n = ARRAY[1..n] OF INTEGER;
```

- *Arrays of floating-point numbers*: Declare an array as one of the following:

```
VAR
  ARRAY1 : ARRAY[1..5] OF SHORTREAL;
  ARRAY1 : REALARR_5;
```

  where ARRAY1 is the name of the variable, 1..5 is the range of the array, and SHORTREAL is the type of element in the array. In the IBM-supplied PROCEDURE declarations, shortreal arrays are defined as REALARR_n, where n is the length of the array, and REALARR_n is a user-defined type declared in the IBM-supplied TYPE definitions (in member ADMUSTNO, file QATTPAS, library QUSRTOOL) as in the following:

```
TYPE
  REALARR_n = ARRAY[1..n] OF SHORTREAL;
```

- *Arrays of character strings*: Declare an array as in any of the following:

```
VAR
  ARRAY1 : ARRAY[1..5] OF PACKED ARRAY[1..8] OF CHAR;
  ARRAY1 : ARRAY[1..5] OF STRING(8);
  ARRAY1 : ARRAY[1..5] OF ALFA;
  ARRAY1 : ARRAY[1..5] OF ALPHA;
  ARRAY1 : ALFAARR_5;
  ARRAY1 : NAMEARR_5;
```

  where ARRAY1 is the name of the variable, 1..5 is the range of the array, and the type of element in the array is a string of characters. In the case of ALFA and ALPHA, these are Pascal predefined types. ALFA is equivalent to PACKED ARRAY[1..8] OF CHAR and ALPHA is equivalent to PACKED ARRAY[1..16] OF CHAR. In the IBM-supplied PROCEDURE declarations, character arrays are defined as ALFAARR_n, where n is the length of the array, and ALFAARR_n is a user-defined type declared in the IBM-supplied TYPE definitions (in member ADMUSTNO, file QATTPAS, library QUSRTOOL) as the following:

```
TYPE
  ALFAARR_n = ARRAY[1..n] OF ALFA;
```

  or as NAMEARR_n, where n is the length of the array, and NAMEARR_n is a user-defined type declared in the IBM-supplied TYPE definitions as in the following:

```
TYPE
  NAMEARR_n = ARRAY[1..n] OF CHARARR_m;
```

OS/400 Graphics routines often have parameters which are defined as having varying lengths. For example, the GSCHAR routine defines its last parameter, STRING, simply as a character string, with no length specified. Pascal, unlike some other languages, does not allow a variable to be of undefined length (with the exception of the predefined type STRING, which due to the way in which it is processed by Pascal, cannot be used for OS/400 Graphics routines).

This means that lengths must be specified for varying length parameters before they may be used in a Pascal program. The IBM-supplied INCLUDE of TYPE definitions (member ADMUSTNO in file QATTPAS in library QUSRTOOL) define several array and character string types of specific lengths. These type definitions are used in the IBM-supplied include statements of PROCEDURE declarations. These same TYPE definitions must also be used in the VAR declarations for the corresponding variables to be passed as parameters on graphics calls in Pascal programs.

If you want to use the GSCHAR routine in your Pascal program, here is what the IBM-supplied declarations look like for TYPE and PROCEDURE:

```
TYPE
  CHARARR_132 = PACKED ARRAY[1..132] OF CHAR;

PROCEDURE GSCHAR(VAR I,J : SHORTREAL; VAR K : INTEGER;
                 VAR L : CHARARR_132);      NONPASCAL;
```

The VAR declarations used in your program would have to look like the following:

```
VAR
    X,Y : SHORTREAL;           /* Correspond to I,J above */
    A : INTEGER;               /* Corresponds to K above  */
    CHARSTRING : CHARARR_132;  /* Corresponds to L above  */
```

And the actual call in your program would look like the following:

```
GSCHAR(X,Y,A,CHARSTRING);
```

In most cases, the maximum value allowed by GDDM for the varying length parameter is the length specified in the IBM-supplied TYPE definitions. However, where this is not efficient (for example, the maximum length allowed is 32,000 bytes), a length is selected which is long enough to handle a typical case, yet still be efficient.

If it becomes necessary to change any of the procedure or type declarations supplied by IBM, it is suggested that the user make copies of these include statements in a personal library, then modify them as necessary for particular applications. Users must then ensure consistency on the PROCEDURE, TYPE, and VAR declarations and place the personal library containing these declarations in front of QUSRTOOL in the *LIBL in order to use the include statements in Pascal programs.

Following are the PROCEDURE declarations from the IBM-supplied include statements which contain varying length parameters:

Figure 1-1 (Page 1 of 4). Procedure Declarations. Declarations for IBM-supplied include statements which contain varying length parameters.

```
PROCEDURE CHAATT(VAR I : INTEGER; VAR J : INTARR_12);
                 NONPASCAL;
```

```
PROCEDURE CHBAR(VAR I,J : INTEGER; VAR K : REALARR_20);
                NONPASCAL;
```

Figure 1-1 (Page 2 of 4). Procedure Declarations. Declarations for IBM-supplied include statements which contain varying length parameters.

```
PROCEDURE CHBATT(VAR I : INTEGER; VAR J : INTARR_3);
                 NONPASCAL;

PROCEDURE CHCOL(VAR I : INTEGER; VAR J : INTARR_32);
                NONPASCAL;

PROCEDURE CHDATT(VAR I : INTEGER; VAR J : INTARR_3);
                 NONPASCAL;

PROCEDURE CHGATT(VAR I : INTEGER; VAR J : INTARR_12);
                 NONPASCAL;

PROCEDURE CHHATT(VAR I : INTEGER; VAR J : INTARR_4);
                 NONPASCAL;

PROCEDURE CHHEAD(VAR I : INTEGER; VAR J : CHARARR_132);
                 NONPASCAL;

PROCEDURE CHHIST(VAR I,J : INTEGER; VAR K,L,M : REALARR_20);
                 NONPASCAL;

PROCEDURE CHKATT(VAR I : INTEGER; VAR J : INTARR_4);
                 NONPASCAL;

PROCEDURE CHKEY(VAR I,J : INTEGER; VAR K : CHARARR_132);
                NONPASCAL;

PROCEDURE CHLATT(VAR I : INTEGER; VAR J : INTARR_6);
                 NONPASCAL;

PROCEDURE CHLT(VAR I : INTEGER; VAR J : INTARR_32);
               NONPASCAL;

PROCEDURE CHLW(VAR I : INTEGER; VAR J : REALARR_32);
              NONPASCAL;

PROCEDURE CHMARK(VAR I : INTEGER; VAR J : INTARR_32);
                 NONPASCAL;

PROCEDURE CHNATT(VAR I : INTEGER; VAR J : INTARR_6);
                 NONPASCAL;

PROCEDURE CHNOTE(VAR I : CHARARR_2; VAR J : INTEGER;
                 VAR K : CHARARR_132);       NONPASCAL;

PROCEDURE CHPAR(VAR I : INTEGER; VAR J : INTARR_32);
                NONPASCAL;

PROCEDURE CHPCTL(VAR I : INTEGER; VAR J : REALARR_1);
                 NONPASCAL;

PROCEDURE CHPEXP(VAR I : INTEGER; VAR J : INTARR_32);
                 NONPASCAL;

PROCEDURE CHPIE(VAR I,J : INTEGER; VAR K : REALARR_20);
                NONPASCAL;

PROCEDURE CHPLOT(VAR I,J : INTEGER; VAR K,L : REALARR_20);
                 NONPASCAL;

PROCEDURE CHSET(VAR I : ALFA);
                NONPASCAL;

PROCEDURE CHSURF(VAR I,J : INTEGER; VAR K,L : REALARR_20);
                 NONPASCAL;

PROCEDURE CHTATT(VAR I : INTEGER; VAR J : INTARR_4);
                 NONPASCAL;

PROCEDURE CHVATT(VAR I : INTEGER; VAR J : INTARR_4);
                 NONPASCAL;
```

Figure 1-1 (Page 3 of 4). Procedure Declarations. Declarations for IBM-supplied include statements which contain varying length parameters.

```
PROCEDURE CHXLAB(VAR I,J : INTEGER; VAR K : CHARARR_132);
                NONPASCAL;

PROCEDURE CHYLAB(VAR I,J : INTEGER; VAR K : CHARARR_132);
                NONPASCAL;

PROCEDURE CHXLAR(VAR I : INTEGER; VAR J : INTARR_6);
                NONPASCAL;

PROCEDURE CHYLAT(VAR I : INTEGER; VAR J : INTARR_6);
                NONPASCAL;

PROCEDURE CHXSET(VAR I : ALFA);
                NONPASCAL;

PROCEDURE CHYSET(VAR I : ALFA);
                NONPASCAL;

PROCEDURE CHXTTL(VAR I : INTEGER; VAR J : CHARARR_132);
                NONPASCAL;

PROCEDURE CHYTTL(VAR I : INTEGER; VAR J : CHARARR_132);
                NONPASCAL;

PROCEDURE DSOPEN(VAR I,J : INTEGER; VAR K : ALFA;
                VAR L : INTEGER; VAR M : INTARR_9;
                VAR N : INTEGER; VAR O : NAMEARR_2);
                NONPASCAL;

PROCEDURE DSQDEV(VAR I : INTEGER; VAR J : ALFA;
                VAR K : INTEGER; VAR L : INTARR_9;
                VAR M : INTEGER; VAR N : NAMEARR_2;
                VAR O : INTEGER; VAR P : INTARR_15);
                NONPASCAL;

PROCEDURE FSEXIT(VAR I : ALFA; VAR J : INTEGER);
                NONPASCAL;

PROCEDURE FSDEV(VAR I : INTEGER; VAR J : INTARR_15);
               NONPASCAL;

PROCEDURE FSQERR(VAR I : INTEGER; VAR J : CHARARR_564);
                NONPASCAL;

PROCEDURE GSCHAP(VAR I : INTEGER; VAR J : CHARARR_132);
                NONPASCAL;

PROCEDURE GSCHAR(VAR I,J : SHORTREAL; VAR K : INTEGER;
                VAR L : CHARARR_132);
                NONPASCAL;

PROCEDURE GSCTD(VAR I,J,K : INTEGER; VAR L,M,N :
              REALARR_7)          NONPASCAL;

PROCEDURE GSGET(VAR I : INTEGER; VAR J : CHARARR_400;
                VAR K : INTEGER);            NONPASCAL;

PROCEDURE GSGETS(VAR I : INTEGER; VAR J : INRARR_1);
                NONPASCAL;

PROCEDURE GSIMG(VAR I,J,K,L : INTEGER; VAR M : CHARARR_2040);
               NONPASCAL;

PROCEDURE GSIMGS(VAR I,J,K,L : INTEGER; VAR M : CHARARR_2040;
                VAR N,O : SHORTREAL);                NONPASCAL;

PROCEDURE GSLSS(VAR I : INTEGER; VAR J : ALFA; VAR L : INTEGER);
               NONPASCAL;

PROCEDURE GSMRKS(VAR I : INTEGER; VAR J,K : REALARR_20);
                NONPASCAL;
```

Figure 1-1 (Page 4 of 4). Procedure Declarations. Declarations for IBM-supplied include statements which contain varying length parameters.

```
PROCEDURE GSPFLT(VAR I : INTEGER; VAR J,K : REALARR_20);
                 NONPASCAL;

PROCEDURE GSPLNE(VAR I : INTEGER; VAR J,K : REALARR_20);
                 NONPASCAL;

PROCEDURE GSPUT(VAR I,J : INTEGER; VAR K : CHARARR_400);
                NONPASCAL;

PROCEDURE GSQCTD(VAR I,J,K : INTEGER; VAR L,M,N : REALARR_8);
                 NONPASCAL;

PROCEDURE GSQSS(VAR I : INTEGER; VAR J : INTARR_20;
                VAR K : ALFAARR_20; VAR L : INTARR_20);
                NONPASCAL;

PROCEDURE GSQTB(VAR I : INTEGER; VAR J : CHARARR_132;
                VAR K : INTEGER; VAR L,M : REALARR_5);
                NONPASCAL;

PROCEDURE GSVECM(VAR I : INTEGER; VAR J : INTARR_20);
                 NONPASCAL;
```

Following are the IBM-supplied TYPE definitions:

Figure 1-2. IBM-Supplied TYPE Definitions

```
REALARR_1 = ARRAY[1..1] OF SHORTREAL;
REALARR_5 = ARRAY[1..5] OF SHORTREAL;
REALARR_7 = ARRAY[1..7] OF SHORTREAL;
REALARR_8 = ARRAY[1..8] OF SHORTREAL;
REALARR_20 = ARRAY[1..20] OF SHORTREAL;
REALARR_32 = ARRAY[1..32] OF SHORTREAL;
INTARR_1 = ARRAY[1..1] OF INTEGER;
INTARR_3 = ARRAY[1..3] OF INTEGER;
INTARR_4 = ARRAY[1..4] OF INTEGER;
INTARR_6 = ARRAY[1..6] OF INTEGER;
INTARR_9 = ARRAY[1..9] OF INTEGER;
INTARR_12 = ARRAY[1..12] OF INTEGER;
INTARR_15 = ARRAY[1..15] OF INTEGER;
INTARR_20 = ARRAY[1..20] OF INTEGER;
INTARR_32 = ARRAY[1..32] OF INTEGER;
CHARARR_2 = PACKED ARRAY[1..2] OF CHAR;
CHARARR_10 = PACKED ARRAY[1..10] OF CHAR;
CHARARR_132 = PACKED ARRAY[1..132] OF CHAR;
CHARARR_400 = PACKED ARRAY[1..400] OF CHAR;
CHARARR_564 = PACKED ARRAY[1..564] OF CHAR;
CHARARR_2040 = PACKED ARRAY[1..2040] OF CHAR;
NAMEARR_2 = ARRAY[1..2] OF CHARARR_10;
ALFAARR_20 = ARRAY[1..20] OF ALFA;
```

The RPG/400 Language

The IBM RPG/400 language does not support floating-point data. Therefore, you must use CALL GDDM to call any graphics routine that requires floating-point parameters. Another advantage of using CALL GDDM is the additional error handling provided to help you diagnose data type errors.

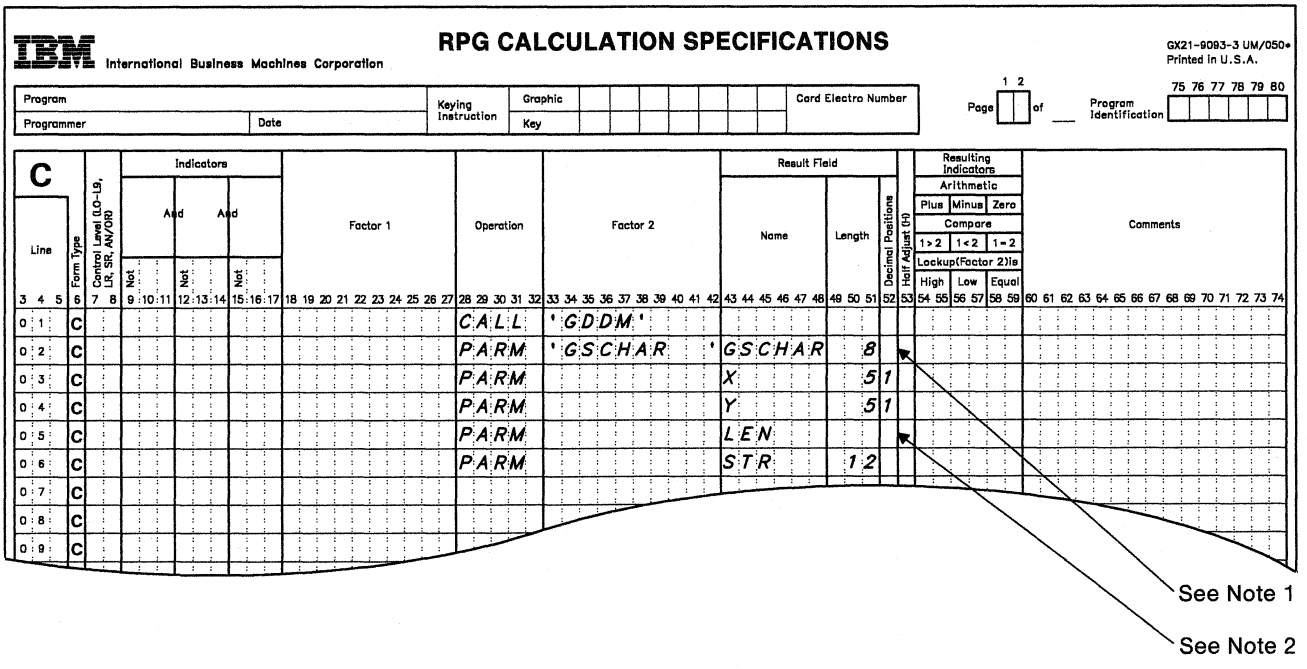
Calling Graphics Routines in the RPG/400 Language

In this manual, the syntax for GSCHAR, which is representative of the other OS/400 Graphics routines, is:

```
GSCHAR(x,y,length,string)
```

To call OS/400 Graphics routines in RPG/400 programs, use one of the following forms of the CALL operation code.

- *When specifying the parameters separately:*



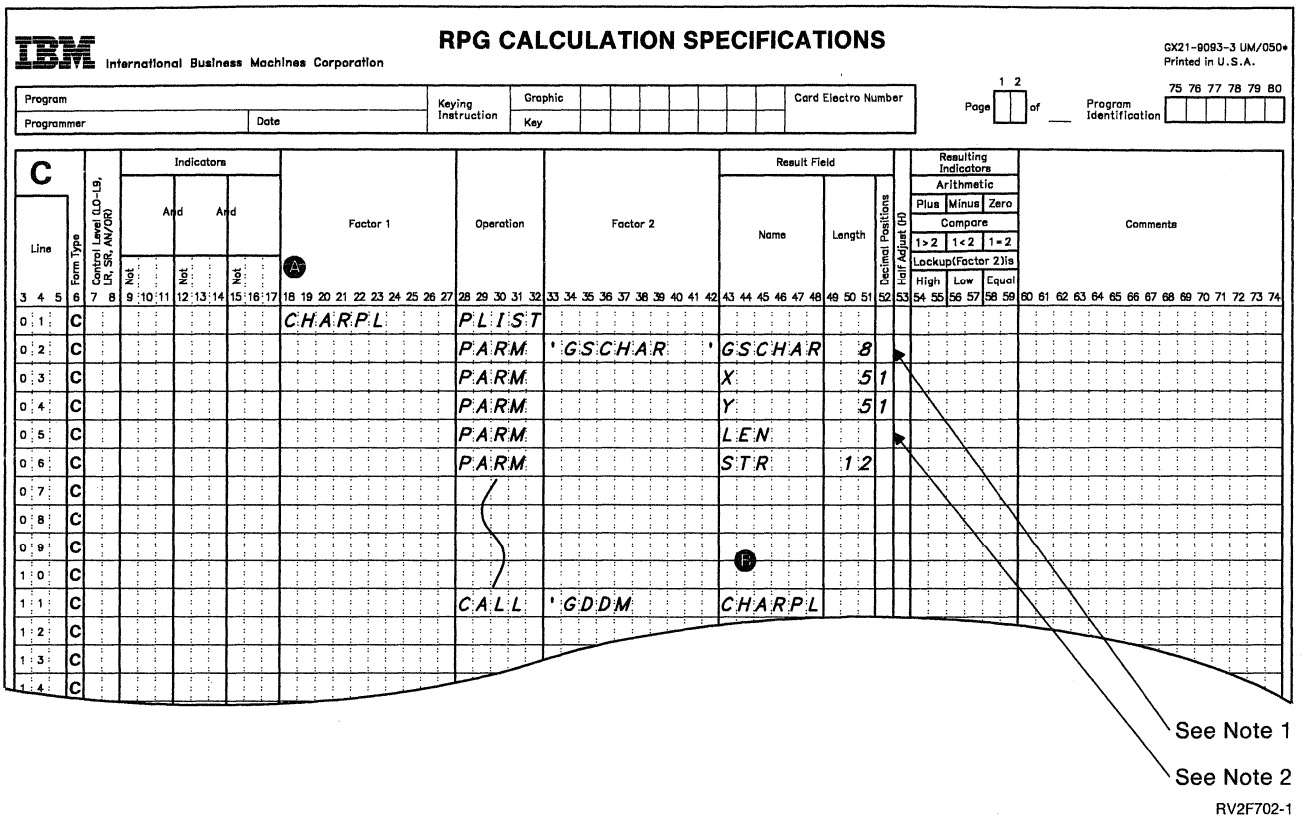
RV2F701-0

In the above example GSCHAR, X, Y, LEN, and STR are program variables defined as shown in "Data Types in the RPG/400 Language" on page 1-18.

Notes:

1. In the RPG/400 language, you cannot specify a character literal in positions 42 through 49 on a PARM statement. You must specify an 8-byte character field in positions 43 through 48. You can define the field on the PARM statement by specifying its length (8) in position 51 (leaving position 52 blank makes it a character field). Also, you can initialize the field on the PARM statement by specifying its value in positions 34 through 42, as shown.
2. LEN is a binary field and must be defined using a data structure input specification (see *four-byte binary integers* on page 1-19).

- *When specifying the parameters in a parameter list (PLIST):*



In the above example GSCHAR, X, Y, LEN, and STR are program variables defined as shown in "Data Types in the RPG/400 Language" and CHARPL is a parameter list defined at **A** and used on the call at **B**.

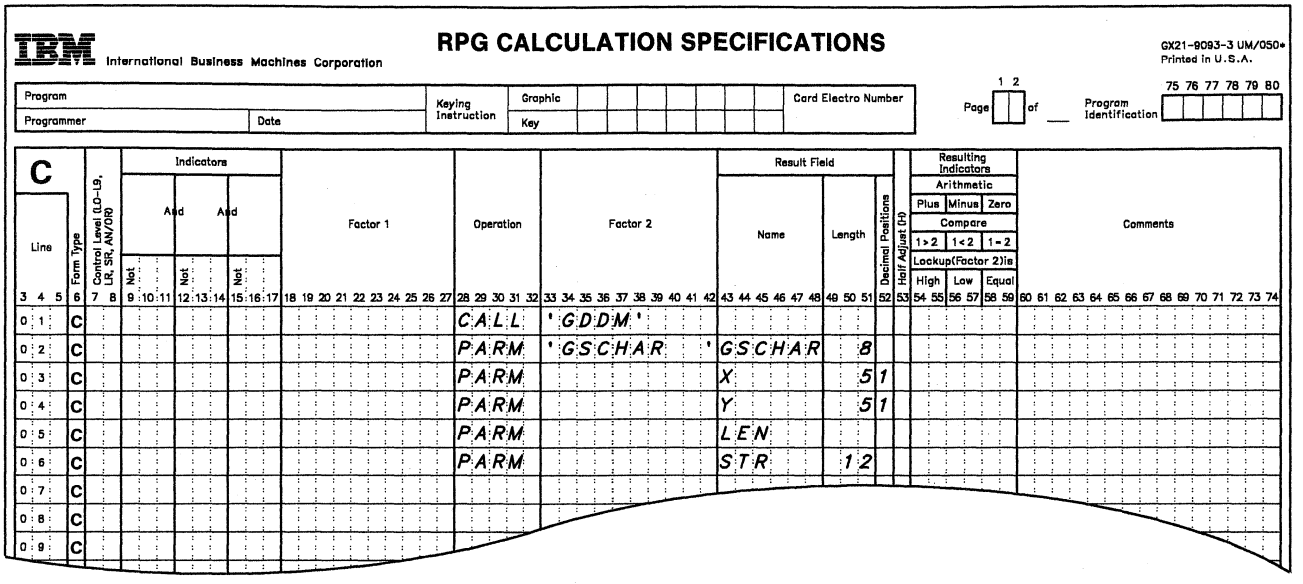
Notes:

1. In the RPG/400 language, you cannot specify a character literal in positions 42 through 49 on a PARM statement. You must specify an 8-byte character field in positions 43 through 48. You can define the field on the PARM statement by specifying its length (8) in position 51 (leaving position 52 blank makes it a character field). Also, you can initialize the field on the PARM statement by specifying its value in positions 34 through 42, as shown.
2. LEN is a binary field and must be defined using a data structure input specification (see *four-byte binary integers* on page 1-19).

Data Types in the RPG/400 Language

In the RPG/400 language, you cannot specify a numeric literal in positions 41 through 49 on a PARM statement; you must use variables (called subfields or fields in the RPG/400 language).

For the other parameters passed to GDDM, you must use specific data types.



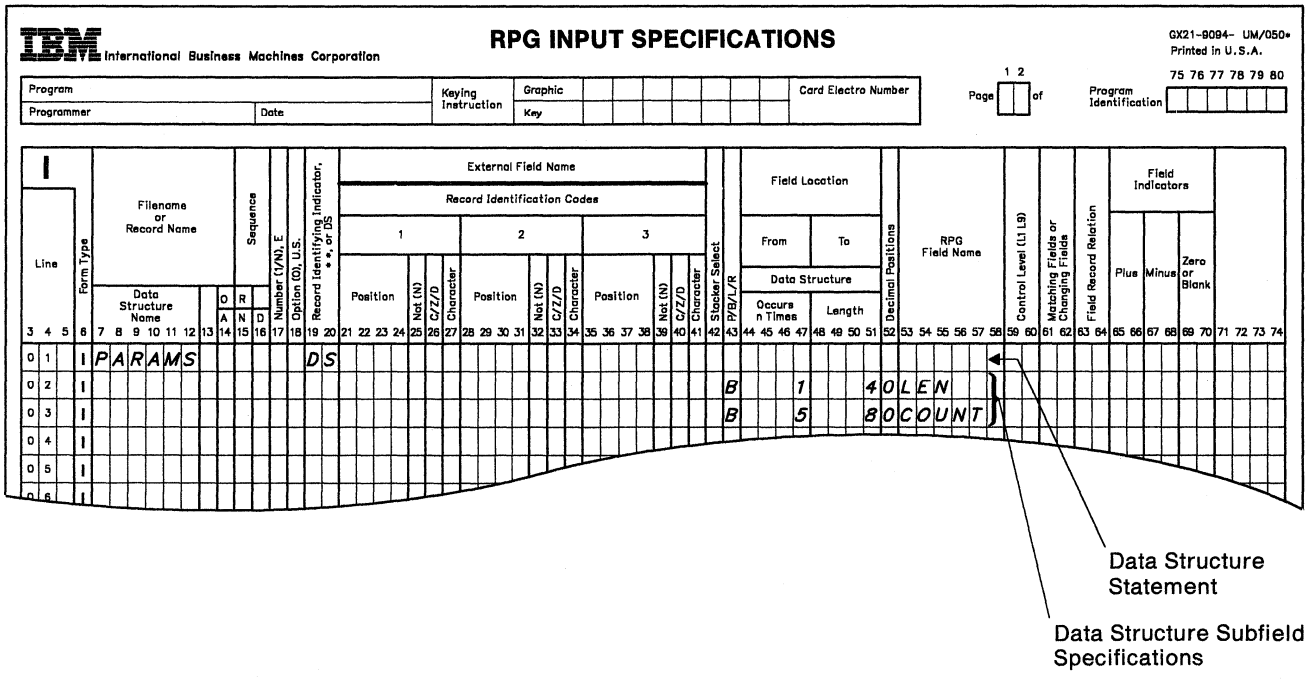
RV2F703-0

In this example the parameters must have the following data types:

- X and Y must be floating-point numbers.
- LEN must be a 4-byte binary integer.
- STR must be a character string.

The following describes how to specify these data types in the RPG/400 language:

- *Four-byte binary integers*: In the RPG/400 language, for program-described variables, specify a data structure and at least one subfield on input specifications as in the following example:



RV2F704-0

In the data structure statement:

- PARAMS is the name of the data structure (this is required by the RPG/400 language when more than one data structure is used in the program and is shown here for illustration).

In the data structure subfield specifications:

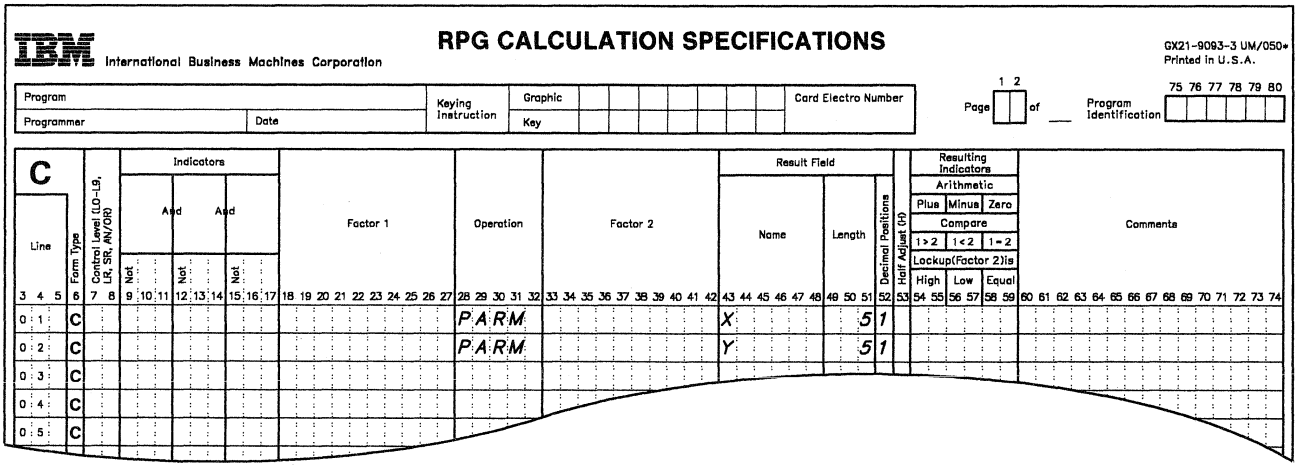
- B specifies that the field is binary
- 1 and 4, and 5 and 8 specify the from and to positions of the two subfields in this data structure
- 0 specifies the number of decimal places and makes these fields integers
- LEN and COUNT are the names of the fields.

- *Floating-point numbers*: In the RPG/400 language, floating-point numbers cannot be defined. Therefore, you should use packed decimal or zoned decimal fields. The data is converted to floating-point when you call a graphics routine using CALL GDDM. A numeric field that is part of an externally described file is a packed decimal field in RPG/400 programs.

  To specify a packed decimal field in your program, specify the field on calculation specifications as follows:

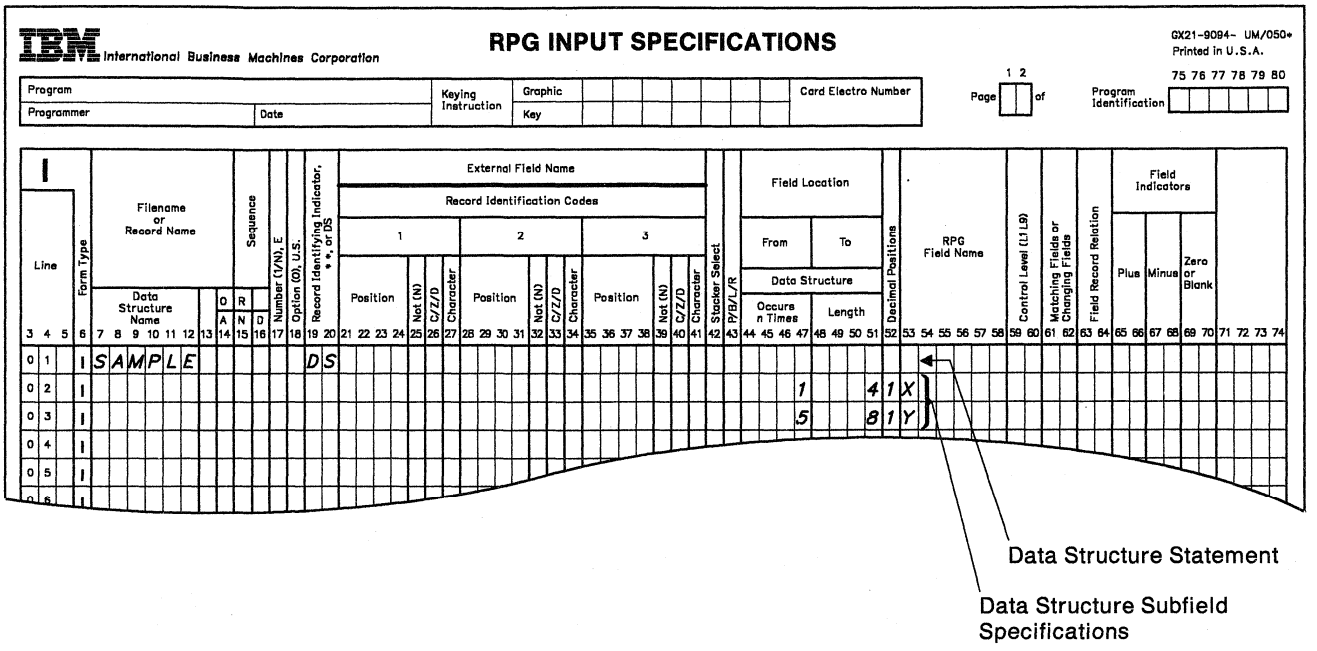
  - In positions 49 through 51: Specify 1 through 15, right justified (this is the length of the variable).
  - In position 52: Specify 0 through 9 (this is the number of decimal positions in the field and makes it a numeric field).

  The following is an example of packed decimal fields:



RV2F705-0

To specify a zoned decimal field in your program, specify the field as a subfield in a data structure as follows:



RV2F706-0

In the data structure statement:

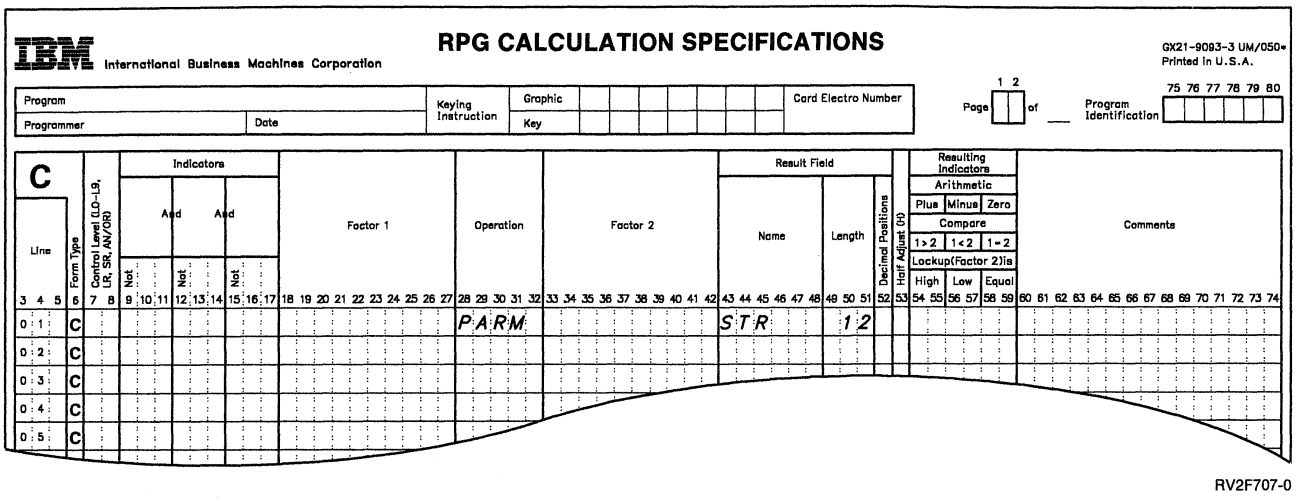
- SAMPLE is the name of the data structure (this is required by the RPG/400 language when more than one data structure is used in the program and is shown here for illustration).

In the data structure subfield specifications:

- Position 43 is blank for zoned decimal data
- 1 and 4, and 5 and 8 specify the from and to positions of the two subfields in this data structure
- 1 specifies the number of decimal places and makes these fields numeric fields
- X and Y are the names of the fields.

- *Character strings*: In the RPG/400 language, for program-described variables, use a character field, defined as follows on calculation specifications:
  - In positions 49 through 51: Specify the length of the variable, right justified.
  - In position 52: Leave blank (this makes the variable a character variable).

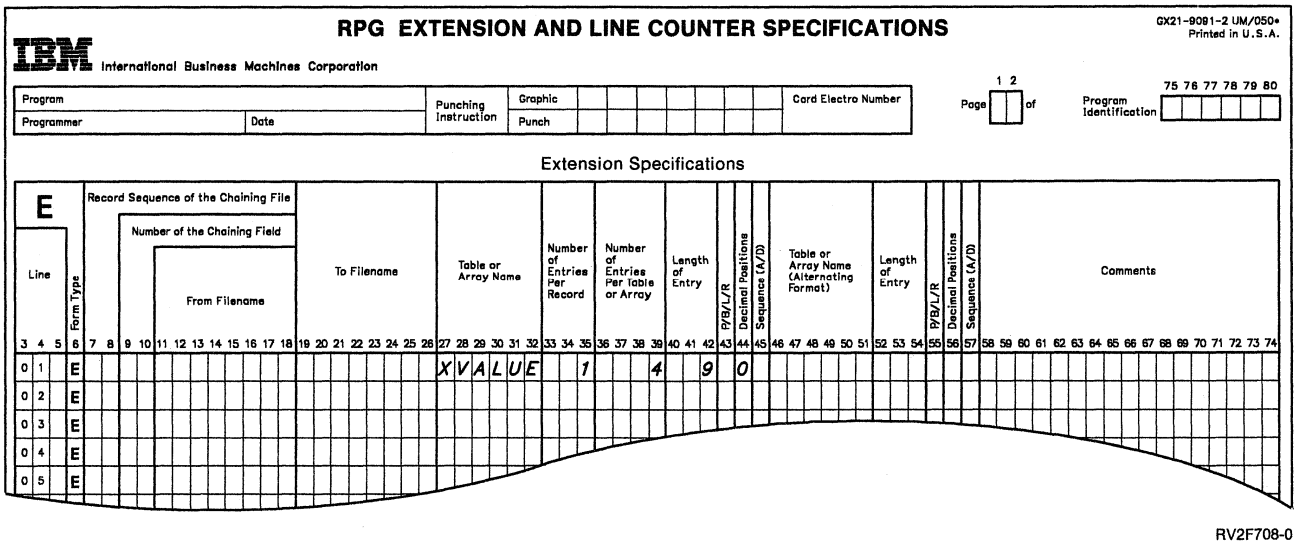
In the following example, a character field (STR), which is used with the PARM operation code, is defined to be 12 bytes long:



- *Arrays of 4-byte binary integers*: You must specify a binary array. To define a binary array, you must first define the array using an RPG extension specification, then redefine the array as a binary data structure using input specifications.

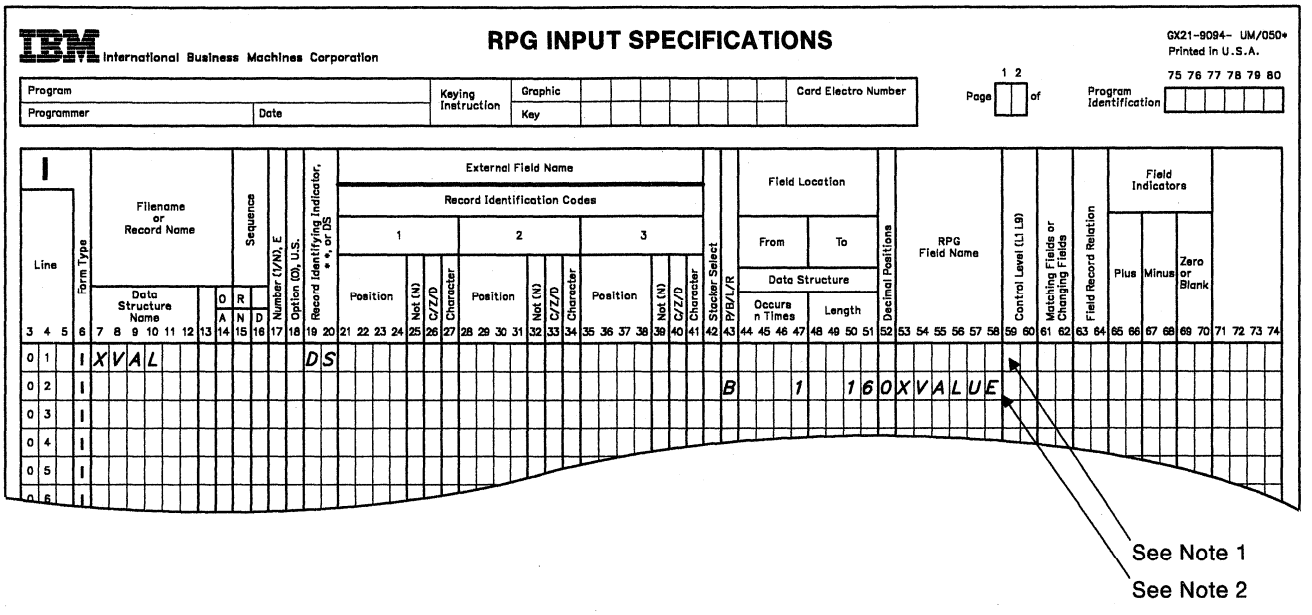
  You can specify the array as a compile-time array, an execution-time array, or a pre-execution time array. A compile-time array sets the array to the same initial values each time the program is run. With execution-time arrays, the program changes the values of array elements as the program runs. For example, the program might read data in from the display device to set the values of the array elements. This section describes compile-time arrays. For a description of execution-time arrays and pre-execution time arrays, refer to the manual *RPG/400* Reference*.

  The following shows the extension specification defining the array:



In this example, XVALUE is the name of the array (and the name of the parameter that is passed on the call to a graphics routine), 1 is the number of entries (array elements) on each array input record (these are specified in the source member after the last RPG specification), 4 is the number of elements in the array, 9 is the length of each element in the array (for 4-byte integers), and 0 specifies the number of decimal positions in each element (making it a numeric array).

The following shows the input specifications that redefine the array XVALUE as a binary data structure:

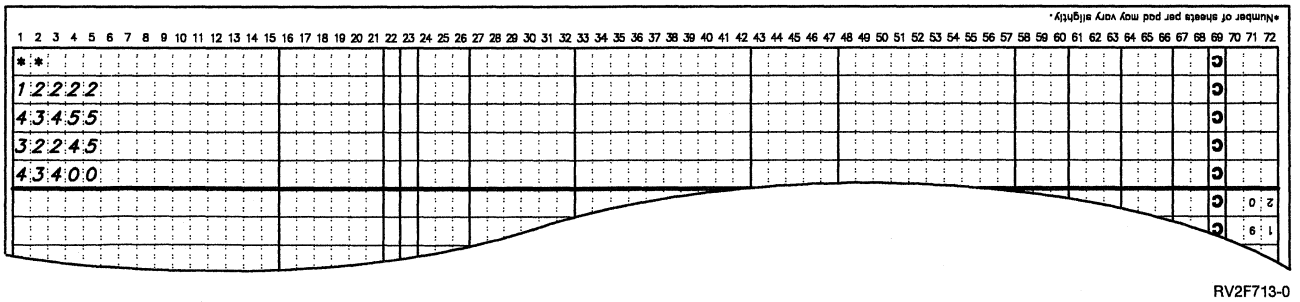


RV2F709-0

Notes:

1. This is the data structure statement. XVAL is a dummy value not used elsewhere in the program. DS identifies the statement as a data structure statement.
2. In this statement, B specifies binary data, 1 and 16 specify the beginning and ending points of the array (16 is four times the number of array entries), and XVALUE is the name of the array that this data structure is redefining.

A sample set of values could be:



RV2F713-0

- *Arrays of floating-point numbers:* In the RPG/400 language, specify either a packed decimal or zoned decimal array. The data is converted to floating-point when you call a graphics routine using CALL GDDM.

  You can specify the array as a compile-time array, an execution-time array, or a pre-execution time array. A compile-time array sets the array to the same initial values each time the program is run. With execution-time arrays, the program changes the values of array elements as the program runs. for example, the program might read data in from the display device to set the values of the array elements. This section describes compile-time arrays. For a description of execution-time arrays and pre-execution time arrays, refer to the manual *RPG/400* Reference*. For a sample execution-time array in a graphics program, refer to the *GDDM Programming Guide*.

  If you use zoned decimal fields, you must first define the array using an RPG extension specification, then redefine the array as a zoned decimal data structure using input specifications. When you use packed decimal fields, you need only define the array using an RPG extension specification. Therefore, packed decimal arrays are easier to specify than zoned decimal arrays.

  The following shows the extension specification defining a compile-time array:

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091-2 UM/050*
Printed in U.S.A.

IBM International Business Machines Corporation

Program | Punching Instruction | Graphic | Card Electro Number
Programmer | Date | Punch

Page 1 2 of | Program Identification 75 76 77 78 79 80

Extension Specifications

E

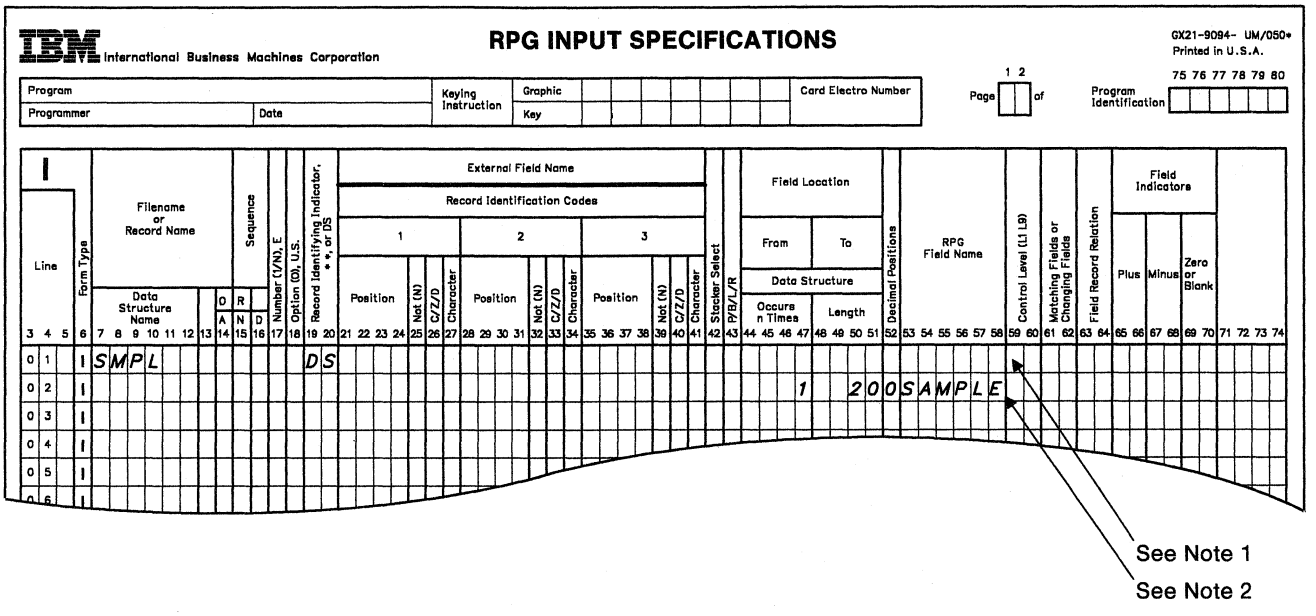
Line	Form Type	Record Sequence of the Chaining File / Number of the Chaining Field / From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
3 4 5	6	7 8 9 10 11 12 13 14 15 16 17 18	19 20 21 22 23 24 25 26	27 28 29 30 31 32	33 34 35	36 37 38 39	40 41 42	43	44	45	46 47 48 49 50 51	52 53 54	55	56	57	58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	E			SAMPLE	1	4	5		0							
0 2	E															
0 3	E															
0 4	E															
0 5	E															

RV2F711-0

In this example:

- SAMPLE is the name of the array.
- 1 is the number of entries (array elements) on each array input record (these are specified in the source member after the last RPG specification).
- 4 is the number of elements in the array.
- 5 is the length of each element in the array (1 through 15 for floating-point numbers).
- 0 specifies the number of decimal positions in each element.

The following shows input specifications that redefine the array SAMPLE as a zoned decimal data structure (not needed for packed decimal data):

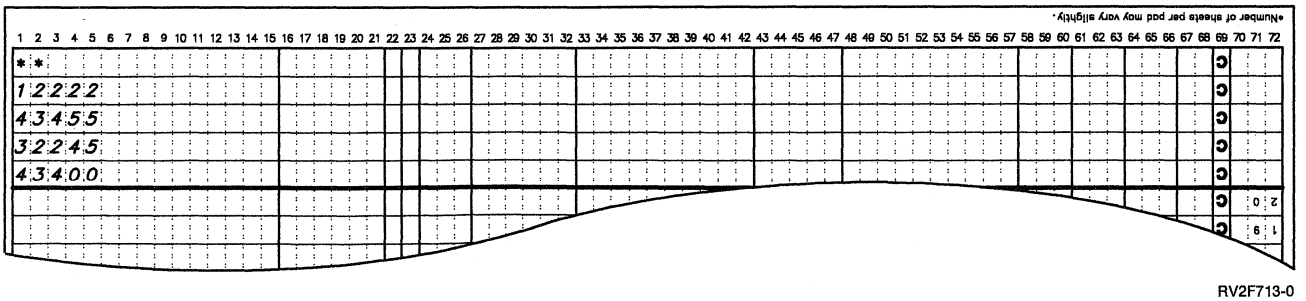


RV2F712-0

Notes:

1. This is the data structure statement. SMPL is a dummy value not used elsewhere in the program. DS identifies the statement as a data structure statement.
2. In this statement, 1 and 20 specify the beginning and ending points of the array (20 is the length of an array item times the number of array entries), and SAMPLE is the name of the array that this data structure is redefining.

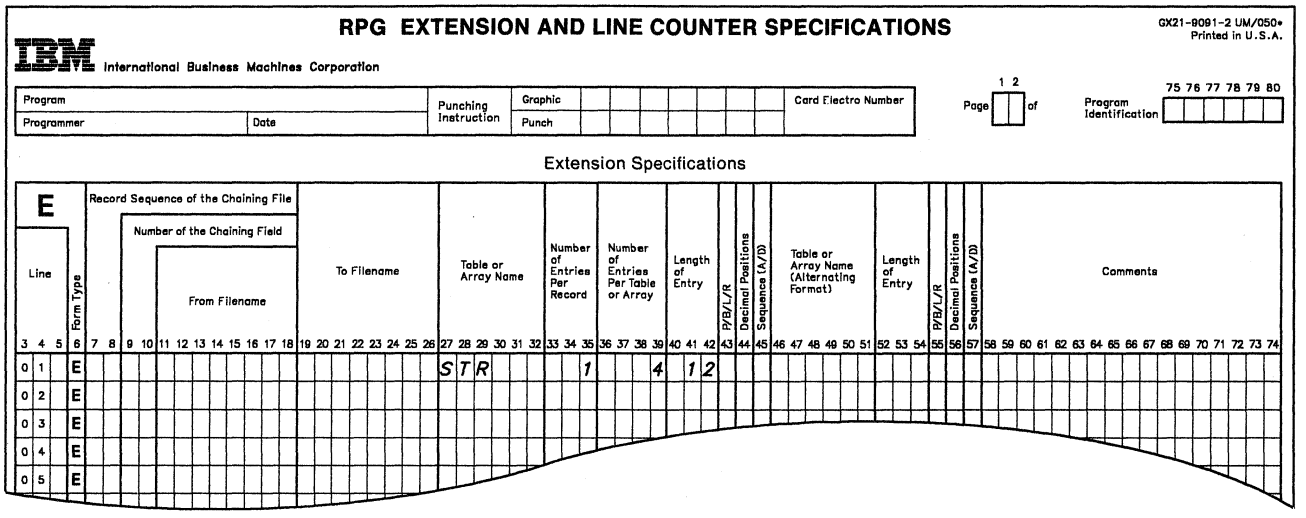
A sample set of values could be:



RV2F713-0

- *Arrays of character strings*: You can specify the array as a compile-time array, an execution-time array, or a pre-execution time array. A compile-time array sets the array to the same initial values each time the program is run. With execution-time arrays, the program changes the values of array elements as the program runs. For example, the program might read data in from the display device to set the values of the array elements. This section describes compile-time arrays. For a description of execution-time arrays and pre-execution time arrays, refer to the manual *RPG/400* Reference*. For a sample execution-time array in a graphics program, refer to the *GDDM Programming Guide*.

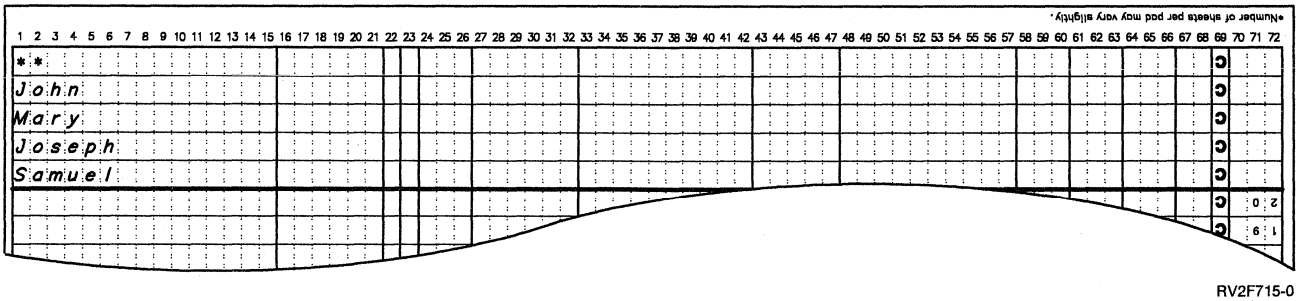
  The following shows the extension specification defining a compile-time array:



RV2F714-0

In this example STR is the name of the array, 1 is the number of entries (array elements) on each array input record (these are specified in the source member after the last RPG specification), 4 is the number of elements in the array, and 12 is the length of each element in the array.

A sample set of values could be:



RV2F715-0

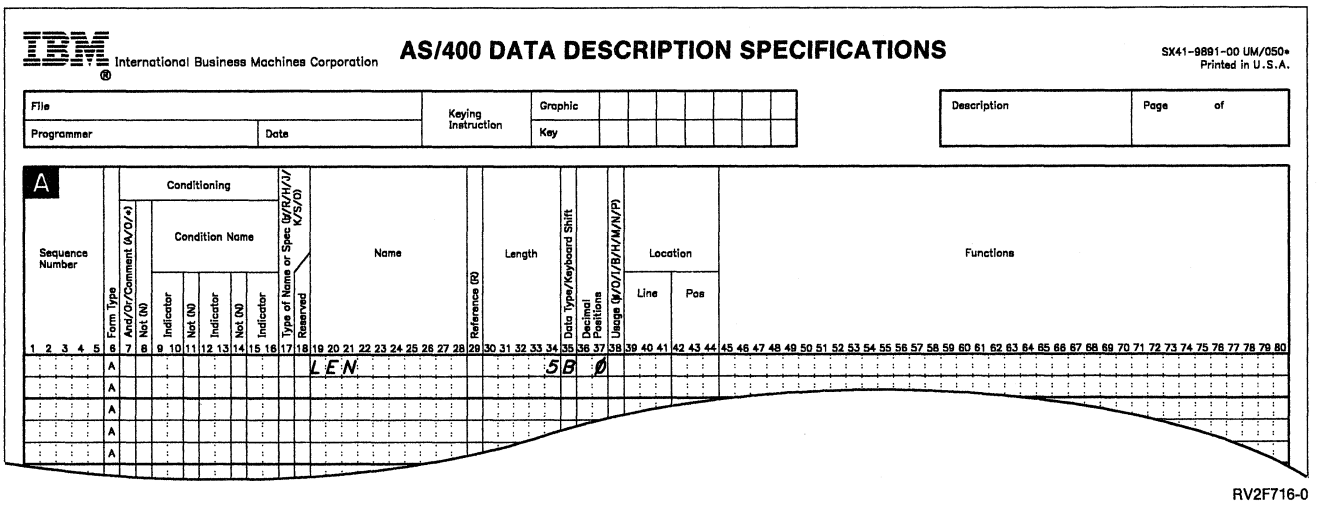
Data Description Specifications

Data Types in Data Description Specifications

To use externally described data in your programs, you define files on the AS/400 system using data description specifications (DDS), then name the files in your programs. The fields defined in the files become variables in your programs. For example, using DDS, you can define a display file by which your program reads in data keyed by the work station user. You can also define a field reference file to standardize parameters used in your programs (you cannot define arrays in DDS). According to the data type, specify:

- *Four-byte binary integers*: In DDS, for a field in a physical file:
  - In positions 19 through 28, specify the name of the parameter.
  - In positions 30 through 34, specify 5 through 9 (to give a 4-byte field).
  - In position 35, specify B (for binary).
  - In positions 36 through 37, specify 0 (defines an integer field).

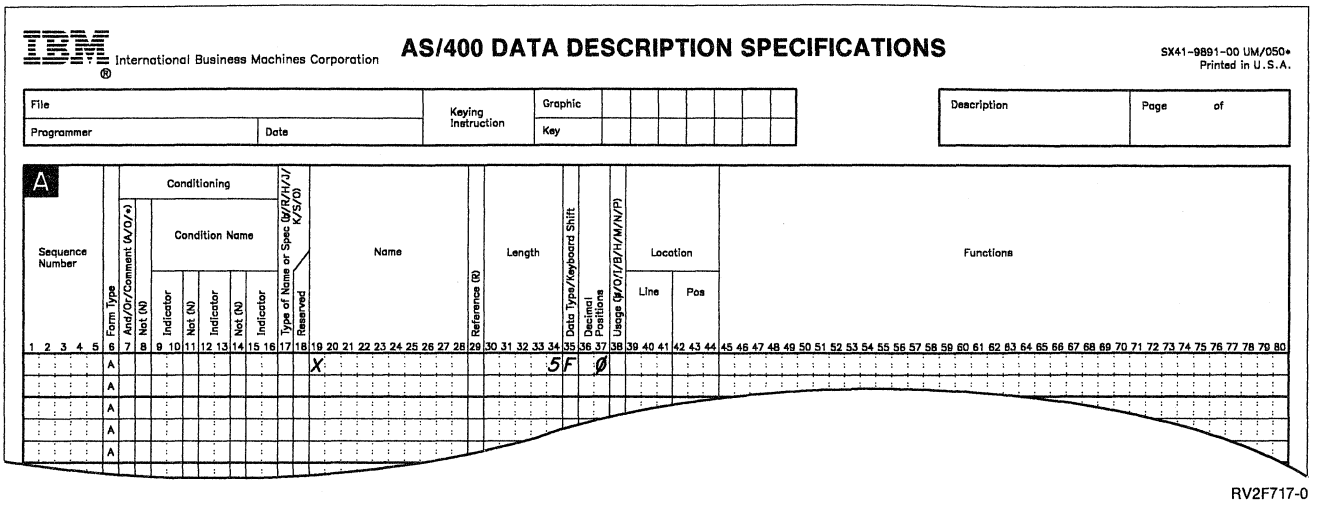
  Here is an example of a 4-byte binary field:



In this example LEN is the name of the field.

- *Floating-point numbers*: In DDS, for a field in a physical file:
  - In positions 19 through 28, specify the name of the parameter.
  - In positions 30 through 34, specify 5 through 9 (for single precision, which is the same as short precision).
  - In position 35, specify F (for floating-point).
  - In positions 36 through 37, specify 0 through 9.

Here is an example of a floating-point field:

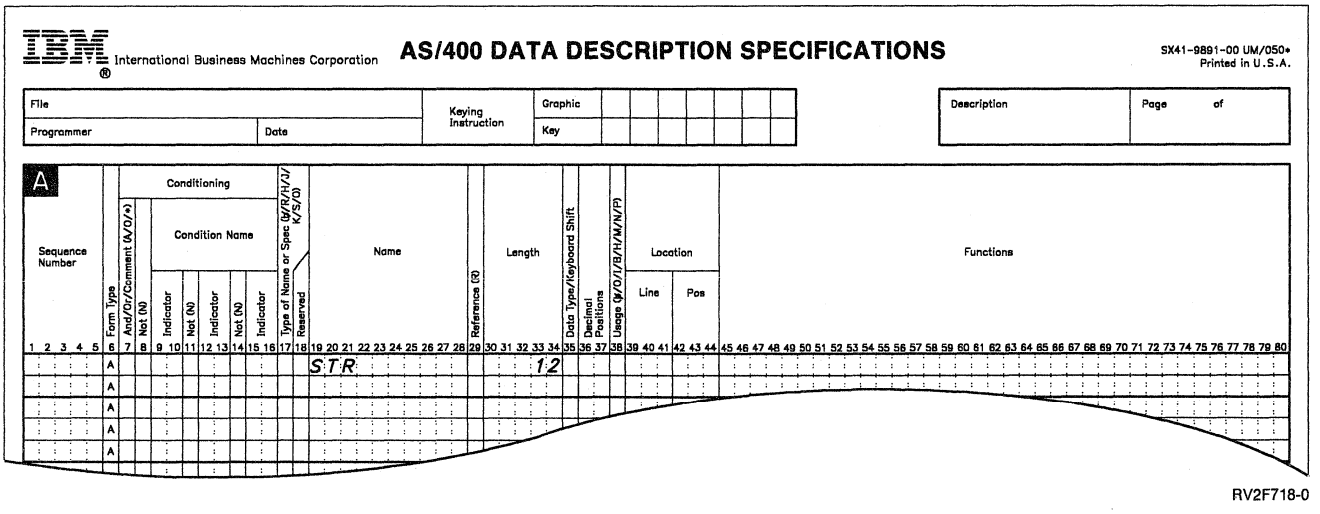


RV2F717-0

In this example X is the name of the field.

- *Character strings*: In DDS, specify the following for a field in a physical file:
  - In positions 19 through 28, specify the name of the parameter.
  - In positions 30 through 34, specify the length of the largest character string you will use for the parameter.
  - Leave position 35 blank (it defaults to character).
  - Leave positions 36 through 37 blank.

The following is an example of a character field:



RV2F718-0

In this example STR is the name of the field.

Chapter 2. GDDM Routines

GDDM Concepts

Figure 2-1 lists GDDM routines according to their function.

For the System/370* graphics functions, see Appendix A, "Conversion and Compatibility."

Figure 2-1. Graphics Routines Available in GDDM	
Drawing Pictures	
Setting color attributes • Selecting a color directly • Selecting a color indirectly • Color mixing	 GSCOL,GSQCOL GSCTD,GSCT,GSQCTD,GSQCT GSMIX
Current position	GSMOVE,GSQCP
Drawing lines • Line attributes • Straight lines • Curved lines	 GSLT,GSQLT,GSLW,GSQLW,GSFLW,GSQFLW GSLINE,GSPLNE,GSVECM GSARC,GSPFLT,GSELPS
Drawing filled areas	GSAREA,GSENDA,GSPAT,GSQPAT
Drawing graphics symbols • Controlling symbol sets • Selecting the current symbol set • Drawing graphics symbols • Setting attributes for graphics symbols – Character mode – Size – Angle – Direction – Shear	 GSLSS,GSRSS,GSQNSS,GSQSS GSCS,GSQCS GSCHAP,GSCHAR GSCM GSCB,GSQCB,GSQTB,GSQCEL GSCA,GSQCA GSCD,GSQCD GSCH,GSQCH
Drawing graphics images	GSIMG,GSIMGS
Drawing markers	GSMS,GSQMS,GSMSC,GSQMSC,GSMARK,GSMRKS
Controlling the Graphics Environment	
Program controls • Graphics program controls • Error handling controls • Display controls • Picture controls – Page – Graphics field – Picture space – Viewport – Window – Graphics segment – Graphics clipping	 FSINIT,FSRNIT,FSTERM FSEXIT,FSQERR ASREAD,FSFRCE,FSREST FSPCRT,FSPDEL,FSPCLR,FSPSEL,FSPQRY,FSQCP GSFLD,GSCLR GSPS,GSQPS GSVIEW,GSQVIE GSWIN,GSQWIN GSSEG,GSSCLS,GSSDEL GSCLP,GSQCLP
Device controls • Opening and closing devices • Using devices • Querying the device characteristics • Sounding the device alarm	 DSOPEN,DSCLS,DSQUID DSUSE,DSDROP,DSQUSE,GSQCUR DSQDEV,FSQDEV FSALRM
Graphics data format (GDF) file • Retrieving graphics data • Drawing data from a GDF	 GSGET,GSGETE,GSGETS GSPUT

Alphabetic List of GDDM Calls

In this section, the GDDM routines, with their syntax and explanations, are listed alphabetically.

The syntax is followed by a description of the routine, an explanation of the parameters, and a sample of the call in BASIC. For information on calling graphic routines in other languages, see "High-Level Languages" on page 1-2.

Syntax Rules

The syntax shown for each GDDM routine includes all the parameters required for calling it.

Each parameter shown is required; that is, if you use the routine, you must specify something for each parameter, even if your program does not use the parameter or if the parameter is ignored on the AS/400 system. (This happens when the function of the parameter is used on the System/370 computer but is not needed on the AS/400 system.)

The syntax shown for each GDDM routine is not a complete CALL statement in any language. Only the parts of the syntax that are specific to OS/400 Graphics are presented. This emphasizes the important aspects of the call. You must code a valid CALL statement in the language you are using. For example,

```
GSCHAR(x,y,length,string)
```

is representative of the syntax in this manual. However, you would code

```
CALL GDDM('GSCHAR',3.5,4.5,23,'Sample character string')
```

in a BASIC program and

```
CALL GSCHAR(3.5,4.5,LENGTH,'Sample character string');
```

in a PL/I program.

You must specify the correct data types in your program. If the data types do not match those required by OS/400 Graphics, undesirable results, possibly even a function check, can occur. For more details on coding graphics calls in various languages, see "High-Level Languages" on page 1-2.

The parameter names have been chosen to be meaningful. For example, in

```
GSCHAR(x,y,length,string)
```

x and y define a coordinate position in a Cartesian coordinate system (which traditionally shows the horizontal axis as the x axis and the vertical axis as the y axis), `length` gives the length of the string, and `string` is a character string to be shown.

ASREAD – Device Output/Input

```
ASREAD(function-key-type,F-key-number,count)
```

Performs all outstanding graphics output. When the current device is a display station, ASREAD waits for the user to press a function key in response to the display. When the response is received, the type of function key and (if the key is an F key) the F key number are returned. GDDM waits for a user response even if no graphics are sent.

When the device is a plotter or printer, graphics are sent to the device and program processing continues as if FSFRCE were specified. A warning error occurs because no input can be received from the device (this is also true for dummy devices).

Parameters

function-key-type (returned by GDDM) (4-byte binary integer variable)
It is one of the following:

0 Enter key.
1 F key.
5 Clear key.
6 Other: The user has pressed one of the following keys: Help, Home, Print, Roll Up, or Roll Down. The value for F key-number is undefined.
7 Output-only device: The primary device accepts only output, and does not return input; the values for F key-number and count are undefined. A warning error is returned to emphasize this situation.

F-key-number (returned by GDDM) (4-byte binary integer variable)
If the function-key-type parameter is 1, this is the number of the function key that was pressed. For example, if F12 was pressed, F-key-number is 12.

count (returned by GDDM) (4-byte binary integer variable)
It is always 0.

Coding Example

In this example, if the work station user presses the F1 key, FKEY is 1, FNUM is 1, and COUNT is 0. If the work station user presses the Help, Home, Print, Roll Up, or Roll Down key, FKEY is 6, FNUM is undefined, and COUNT is 0.

```
INTEGER  FKEY,FNUM,COUNT
CALL  GDDM('ASREAD',FKEY,FNUM,COUNT)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the following function-key-type values are not supported:

2 Light pen
3 Badge reader
4 Cp key.

DSCLS – Close Device

```
DSCLS(device-id,option)
```

Terminates the use of a device by GDDM. Any resources (such as symbol sets, page contents, or color tables) that have been defined for the device are released. Any device that is subsequently opened with the same device-id bears no relationship to the device now being closed.

Any usage currently in force for this device is discontinued.

Note that DSCLS differs from most GDDM routines, because it attempts to complete even if an error is returned.

On display devices, any graphics at the device, even if not displayed, are erased when this routine is called.

Parameters

device-id (4-byte binary integer)
The device identifier of the device to be closed.

option (4-byte binary integer)
Must be zero on the AS/400 system.

Coding Example

In the following example, the device with identifier 2 is closed and, if it is a display, any graphics at the device are erased:

```
CALL  GDDM('DSCLS',2,0)
```

Here 0 is a place holder for the option parameter (all parameters must be specified on calls to graphics routines). To give a device an identifier, use DSOPEN.

Differences between the System/370 Computer and the AS/400 System

On the System/370 computer, you can specify an option to keep the graphics at the device; on the AS/400 system, the graphics are always erased.

DSDROP – Line Device Usage

DSDROP(usage,device-id)

Causes a device to no longer operate with the specified usage. The device now enters a state in which the only operations that can be performed against it are DSCLS, DSRNIT, DSQDEV, and DSUSE. Its resources are not, however, released, and its usage (together with all such resources as page contents, symbol sets, and color tables that existed at the time of the DSDROP) can be subsequently restored by means of a DSUSE.

Parameters

usage (4-byte binary integer)
The device usage code, which must take the following value:

1 Current primary device.

device-id (4-byte binary integer)
The identifier of the device to be discontinued.

Coding Example

In the following example, the program first uses device 2 as the primary device, then drops the use of device 2 as primary device and starts using device 3 as the current primary device.

```
CALL GDDM('DSUSE',1,2)
   .
   .
   .
CALL GDDM('DSDROP',1,2)
CALL GDDM('DSUSE',1,3)
   .
   .
   .
```

To give a device an identifier, use DSOPEN.

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, secondary device support (usage 2) is not provided.

DSOPEN – Open Device

```
DSOPEN(device-id,family,device-token,procopt-count,procopt-list,
       name-count,name-list)
```

Opens a device to be used for graphics. DSOPEN gives the device a device identifier and describes the device to GDDM.

If you call DSOPEN, you must also call DSUSE before the device becomes the current device.

Parameter Summary

The use of the various parameters is summarized below to give a quick overview.

device-id (number used by other GDDM calls including DSUSE and DSCLS)

user-defined: Cannot be negative and should not be 0.

family

1 Graphics work station, plotter, or graphics-capable printer.

device-token (name from a list of device descriptions)

'* '	Actual device used is determined at execution time
'*PLOT '	Device is any available plotter
'5292M2 '	Device is a 5292 Model 2 or a personal computer configured as a 5292
'6180 '	Device is an IBM 6180 Plotter attached to a graphics work station
'6182 '	Device is an IBM 6182 Plotter attached to a graphics work station
'6184 '	Device is an IBM 6184 Plotter attached to a graphics work station
'6185 '	Device is an IBM 6185 Plotter attached to a graphics work station
'6186M1 '	Device is an IBM 6186-1 Plotter attached to a graphics work station
'6186M2 '	Device is an IBM 6186-2 Plotter attached to a graphics work station
'7371 '	Device is an IBM 7371 or 6180 Plotter attached to a graphics work station
'7372 '	Device is an IBM 7372 Plotter attached to a graphics work station
'L79A3 '	Device token used to obtain graphics data format (GDF) files. Only valid for dummy device support.
'522X '	Device is an IBM 5224 or 5225 Printer
'4214 '	Device is an IBM 4214 Printer
'IPDS '	Device is an IPDS Printer, such as an IBM 4224 Printer
'4234 '	Device is an IBM 4234 Model 2 Printer.

procopt-count (number of items in procopt-list)

0	Procopt-list is empty
2 through 11	Number of items in procopt-list

procopt-list (processing options)

10 Plotter form feed (2 items)
11 Plotter pen velocity (2 items)
12 Plotter pen width (2 items)
15 Plotter paper size (3 items)
16 Plotter paper orientation (2 items)

name-count (number of elements in name-list)

0 Name-list is empty
1 Name-list contains one name
2 Name-list contains one display device description name and a plotter address.

name-list (device description name or a special value)

First element:

'* '	*REQUESTOR device or default printer file QPGDDM
'name '	OS/400 device description name of graphics work station
' '	Dummy device
'printer-file '	Name of the user-defined printer file or the IBM-supplied printer file QPGDDM.

Second element:

' nn'	Two-character address of plotter, right justified in character string
'ADMPLOT '	Use lowest address of available plotters of the type specified in the device token.

For coding examples, see page 2-16.

Parameters

device-id (4-byte binary integer)
The device identifier. The device identifier must not be negative.

Avoid 0 because GDDM uses 0 as the default device identifier.

Note that GDDM issues an internal DSOPEN request to open the default primary device (device-id is 0). This occurs when you do not use DSOPEN.

family (4-byte binary integer)
The device family code, which must take the following value:

1 Graphics work station, IBM plotter, graphics-capable printer, or the dummy device support provided by device token L79A3.

device-token (8-byte character string)
Names a group of properties that apply to specific devices. You can specify the following values:

'* ' At execution time, GDDM determines the device properties from the value of the name-list parameter and the OS/400 device description (*DEVD) for the display device being used. For the actual device used, see Figure 2-2.

No procopts can be specified for use with this token (procopt-count must be 0), and the name-list is ignored.

'*PLOT ' When no address is specified in the second element of the name-list array, the device is the first available plotter (the plotter at the lowest address). If an address is specified, the device is whatever type of plotter that is configured at that address.

'5292M2 ' The device is an IBM 5292 Display Station Model 2, or an IBM Personal Computer configured as a 5292 Model 2

No procopts can be specified for use with this token (procopt-count must be 0), and the name-list is ignored.

'6180 ' The device is an IBM 6180 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'6182 ' The device is an IBM 6182 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'6184 ' The device is an IBM 6184 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'6185 ' The device is an IBM 6185 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'6186M1 ' The device is an IBM 6186-1 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'6186M2 ' The device is an IBM 6186-2 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'7371 ' The device is an IBM 7371 or 6180 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

'7372 ' The device is an IBM 7372 Plotter that is attached to a graphics work station. The plotter device must be specified on the device description of the graphics work station (AUXDEV parameter of the CRTDEVDSP (Create Device Description (Display)) or the CHGDEVD (Change Device Description) command).

No processing options can be specified for use with the following device tokens (that is, the processing option count must be 0):

'522X ' The device is an IBM 5224 or 5225 Printer.

'4214 ' The device is an IBM 4214 Printer.

'4234 ' The device is an IBM 4234 Model 2 Printer.

You can supply your own printer file via the name list parameter when using any of the device tokens.

'L79A3 ' This token *always* results in dummy device support, and is intended for use in the generation of GDF (see GSGET later in this chapter) at a level similar to that produced by the same token on the System/370 computer. The GDF produced when this token is used is more compact than that produced when other tokens are used.

For this token, the name-list is ignored because dummy device support is always assumed.

This token does *not* provide graphics support for IBM 3279 displays remotely attached to the AS/400 system.

The processing options for paper orientation can be specified for use with the following printer device token:

'IPDS ' The device is a printer that accepts an intelligent printer data stream (IPDS). This device token is used to access any supported IPDS printer, including the 3812, 3816, 4028, and 4224 printers.

GDDM left-justifies the supplied parameter, and converts it to uppercase, if necessary.

Figure 2-2 shows the type of device support that results from various combinations of the device-token and name-list parameters.

Figure 2-2 (Page 1 of 2). Devices Supported by DSOPEN Parameters

Job Type (Job), Name-List (NL),	**Device-Tokens**				
Device (Dev)	*	*PLOT	5292M2	Plotter	Printer/IPDS
Job = Interactive, NL = *REQUESTER, Dev = display+plotter	Display	Plotter	Display	Plotter	Printer
Job = Interactive, NL = *REQUESTER, Dev = display	Display	Plotter	Display	Dummy plotter	Printer
Job = Interactive, NL = *REQUESTER, Dev = nongraphics	Dummy display	Dummy plotter	Dummy display	Dummy plotter	Printer
Job = Interactive, NL = device-name, Dev = display+plotter	Display	Plotter	Display	Plotter	Error
Job = Interactive, NL = device-name, Dev = display	Display	Plotter	Display	Dummy plotter	Error

Figure 2-2 (Page 2 of 2). Devices Supported by DSOPEN Parameters

Job = Interactive, NL = device-name, Dev = nongraphics	Dummy display	Dummy plotter	Dummy display	Dummy plotter	Error
Job = Any, NL = ' ', Dev = N/A	Dummy display	Dummy plotter	Dummy display	Dummy plotter	Dummy printer
Job = Batch, NL = *REQUESTER, Dev = N/A	Dummy display	Dummy plotter	Dummy display	Dummy plotter	Dummy printer
Job = Batch, NL = device-name, Dev = display+plotter	Display	Plotter	Display	Plotter	Error
Job = Batch, NL = device-name, Dev = display	Display	Plotter	Display	Dummy plotter	Error
Job = Batch, NL = device-name, Dev = nongraphics	Dummy display	Dummy plotter	Dummy display	Dummy plotter	Error
Job = Interactive, NL = printer-file-name, Dev = Any	Error	Error	Error	Error	Printer
Job = Batch, NL = printer-file-name, Dev = N/A	Error	Error	Error	Error	Printer

Note: For printer or IPDS device tokens, device support depends upon whether the printer file (QPGDDM or user-defined) has been created/changed/overridden to SPOOL(*YES) or SPOOL(*NO). SPOOL(*YES) will result in a spooled graphics printer file, while SPOOL(*NO) will result in immediate output to the printer named in the printer file, if it is available. See the following table.

Figure 2-3 (Page 1 of 2). Printer Device Support

Name-List (NL), Overrides (Ovr)	**SNA Character String Printer Device-Token**	**Intelligent Printer Data Stream Printer Device-Token**
NL = *REQUESTOR, Ovr = None	Spooled default printer file QPGDDM	Spooled default printer file QPGDDM
NL = printer-file-name, Ovr = None	Spooled printer file 'printer-file-name'	Spooled printer file 'printer-file-name'
NL = *REQUESTOR, Ovr = FILE(printer-file-name)	Spooled printer file 'printer-file-name'	Spooled printer file 'printer-file-name'
NL = printer-file-name-1, Ovr = FILE(printer-file-name-2)	Spooled printer file 'printer-file-name-2'	Spooled printer file 'printer-file-name-2'
NL = *REQUESTOR, Ovr = FILE(printer-file-name), DEV(SCS-device-name)	Spooled printer file 'printer-file-name' compatible with SCS printer devices only	Spooled printer file 'printer-file-name' compatible with IPDS printer devices only
NL = *REQUESTOR, Ovr = FILE(printer-file-name), DEV(IPDS-device-name)	Spooled printer file 'printer-file-name' compatible with SCS printer devices only	Spooled printer file 'printer-file-name' compatible with IPDS printer devices only

Figure 2-3 (Page 2 of 2). Printer Device Support

Name-List (NL), Overrides (Ovr)	SNA Character String Printer Device-Token	Intelligent Printer Data Stream Printer Device-Token
NL = *REQUESTOR, Ovr = DEV(SCS-device-name), SPOOL(*NO)	Direct output to printer 'SCS-device-name'	Dummy device support
NL = *REQUESTOR, Ovr = DEV(IPDS-device-name), SPOOL(*NO)	Dummy device support	Direct output to printer 'IPDS-device-name'

procopt-count (4-byte binary integer)
Specifies the number of elements in procopt-list. You can specify 0 to indicate to GDDM that procopt-list is empty and is not to be inspected. (You must specify the procopt-list parameter on the call even if procopt-count is zero.)

If you are not using the plotter, specify 0. If you can take the plotter defaults, specify 0.

procopt-list (array of 4-byte binary integers)
Passes miscellaneous processing options to GDDM.

Each option is passed as an option group, consisting of an option code followed by one or more integers of option data. Option groups may be specified in any order. Only those option groups for which you specifically need to override GDDM's defaults need be specified. If the option group does not apply to the particular device, it must not be specified or an error occurs and the DSOPEN operation fails.

The parameter list takes the form:

Option Code	Option 1	Option Code	Option 2	Option 3

For example, for 8.5 inch by 11 inch transparencies:

11	30	15	1	2

In this case, procopt-count would be 5.

The supported option groups are:

Option Group	Option Code	Number of Integers
Plotter form feed	10	2
Plotter pen velocity	11	2
Plotter pen width	12	2
Plotter paper size	15	3
Paper orientation	16	2

Note: The procopt-count is the number of 4-byte integers in the option group, including the option code.

Valid values for each processing option group and plotter are:

Plotter	Processing Option Group					Notes
	10	11	12	15	16	
6180	NV	0–100	0–10	0–2	0–2	3
6182	0–2	0–100	0–10	NV	0–2	2
6184	NV	0–100	0–10	NV	0–2	
6185	NV	0–100	0–10	NV	0–2	
6186-1	NV	0–100	0–10	0–5	0–2	3
6186-2	0–2	0–100	0–10	0–5	0–2	1, 3
7371	NV	0–100	0–10	NV	NV	
7372	NV	0–100	0–10	0–2	0–2	3

NV - Processing option group is not valid for this plotter.

Notes:

1. Form feed option enables roll feed.
2. Form feed option enables 8.5 x 11 inch sheet feed.
3. Value shown for option group 15 is paper size code. The dimension type code value is 0–2.

Processing option group 10: Plotter form feed

This option group allows the form feed to be specified for the plotter. The form feed may be either enabled or disabled.

This option group has two elements:

- The option group code is 10.
- Form feed code:

  0 Default of enabled (same as 2)
  1 Disabled
  2 Enabled

Processing option group 11: Plotter pen velocity

This option group gives the relative pen speed to be used by the plotter. The default value should be suitable for fiber-tipped pens used on normal paper; when transparency pens are used, a slower pen speed, such as 30, might be necessary.

This option group has two elements, as follows:

- The option group code is 11.
- Pen velocity:

  0 Default pen velocity of 100 (the maximum)
  1 through 100: Specified pen velocity as percentage of the maximum velocity available on the plotter

Processing option group 12: Plotter pen width

This option group gives the width of the pens used in the plotter, and applies to *all* pens in the plotter. The pen width is important for fine detail, such as in area fill operations and drawing character strings with small-sized characters. If pens of

different widths are to be used in the same picture, the pen width should be set to the size of the pens that are to be used for the finest detail.

This option group has two elements:

- The option group code is 12.
- Pen width, in units of 0.1 millimeters:

0	Default pen width of 3 (0.3 mm)
1 through 10	Specified pen width (0.1 through 1.0 mm)

Typical pen widths for plotters are 3 (0.3 mm), 6 (0.6 mm), and 7 (0.7 mm).

Processing option group 15: Plotter paper size

This option group gives the size of the paper that has been loaded into the plotter. See Figure 2-4 for a summary of paper sizes.

The switches controlling paper size on the plotter must be set to indicate the same paper size specified in this option group, or undesired results may occur (such as aspect ratio distortion, mis-centering, or partial picture).

This option group has three elements:

- The option group code is 15.
- Paper size code:

0	Default of small (same as 1)
1	A or A4
2	B or A3
3	C or A2
4	D or A1
5	E or A0

- Dimension type code:

0	Default of ISO dimensions (same as 1)
1	ISO dimensions (A4, A3, A2, A1, A0)
2	ANSI dimensions (A, B, C, D, E)

Figure 2-4. Plotter Paper Sizes

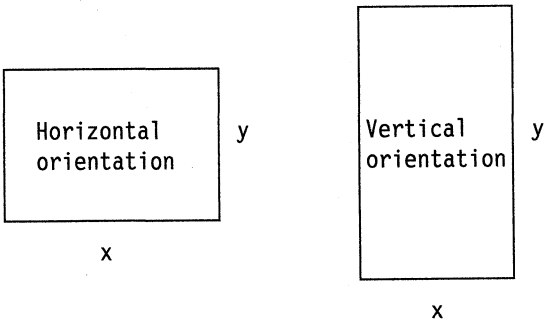
Paper Size Code	Dimension Type Code	Paper Size	Actual Paper Dimensions	Default Plotting Area
1	1 (ISO)	A4	210 x 297 mm	179.1 x 248.8 mm
	2 (ANSI)	A	8.5 x 11 inches	7.09 x 9.84 inches
2	1 (ISO)	A3	297 x 420 mm	248.8 x 378.1 mm
	2 (ANSI)	B	11 x 17 inches	9.84 x 14.96 inches
3	1 (ISO)	A2	420 x 594 mm	334 x 532 mm
	2 (ANSI)	C	17 x 22 inches	13.61 x 19.56 inches
4	1 (ISO)	A1	594 x 841 mm	508 x 779 mm
	2 (ANSI)	D	22 x 34 inches	18.61 x 31.56 inches
5	1 (ISO)	A0	841 x 1189 mm	779 x 1103 mm
	2 (ANSI)	E	34 x 44 inches	31.56 x 40.61 inches

Processing option group 16: Paper orientation

This option group allows the paper orientation to be specified for the plotter. The paper may be oriented either horizontally, with the long dimension of the paper along the x axis, or vertically, with the long dimension of the paper along the y axis.

Note that this option group *does not* have anything to do with the way that the paper is placed in the plotter; instead it specifies the paper's orientation relative to the x and y directions defined by GDDM.

When this option group is used, it is especially important to correctly handle the picture space definition, because the definition will change depending upon the paper orientation chosen. The picture space definition can be queried with GSQPS and set with GSPS.



This option group has two elements:

- The option group code is 16.
- Orientation:

  0 Default of horizontal orientation (1)
  1 Horizontal
  2 Vertical

name-count (4-byte binary integer)
Specifies the number of 10-byte strings in name-list. Specifying 0 indicates that name-list is empty and is not to be inspected. Even if name-list is empty, the name-count parameter must be specified in the parameter list of the call.

name-list (array of one or two 10-byte character string elements)
Specifies the device description name or a special value. For a plotter, use the name of the graphics work station to which the plotter is attached.

GDDM left-justifies the string and converts it to uppercase, if necessary.

Even if name-list is empty, its name must be specified in the parameter list of the call.

If the name-count parameter is 0, the device is the *REQUESTER device or default printer file.

If the name-count parameter is 1, the device is one of the following:

'* ' The device used is *REQUESTER, that is, the current interactive work station is to be used. The actual device used depends on the device-token parameter. See Figure 2-2 on page 2-10 for the actual device used. This is the same as specifying zero for the name-count parameter.

'name '	This is the device description name of a display station configured on your system, or, for printer or IPDS device tokens, the name of a printer file to be used to spool the graphics data. An error occurs if the device is not available when the application program is called and this DSOPEN call is executed. If the device token specified a printer, and no printer file is specified, the default printer file QPGDDM is used to format output, which is spooled to the appropriate output queue. If QPGDDM has been overridden to a specific printer (that is, SPOOL(*NO) has been specified), the output will be printed on the selected printer (if it is available). See Figure 2-3 on page 2-11 for more information.
' '	The device is a dummy device, (that is, no real device is associated with this GDDM device) or the default printer file QPGDDM. For dummy device support, GDDM generates the data streams required but does not send them to any real device, nor does it attempt to receive data from a device. This option can be used to check a GDDM application when a real device with the necessary features is unavailable.
'printer-file'	The name printer file is used. If QPGDDM is the named printer file, processing occurs as if this first element were left blank (described above). If a user-defined printer file is named in this element, it is used to format the output, and its attributes are used to determine whether the output is spooled to a particular output queue or sent directly to the printer.

If the name count is 2, the plotter device is selected, based on either of the following entries in the second array element:

' nn'	The plotter (of the type specified by the device token) at address nn is used. The address assigned to the plotter in the device description for the associated work station is represented in this element by two right-justified characters.
'ADMPLOT '	The plotter (of the type specified by the device token) at the lowest address of an available plotter is used.

Coding Examples

Example 1:

The following example opens the *REQUESTER device, that is, the actual device from which the program is called, and assigns it a device identifier of 2. If this device is a graphics work station, graphics are displayed; if this device is a nongraphics device, a diagnostic message is issued for successful execution.

```
OPTION BASE 1
INTEGER PROCOPT_LIST
DIM PROCOPT_LIST(1)
DIM NAME_LIST$(1)*10
CALL GDDM('DSOPEN',2,1,'*       ',0,PROCOPT_LIST(),0,NAME_LIST$())
CALL GDDM('DSUSE',1,2)
```

Example 2:

The following example opens the IBM 6180 Plotter, giving it a device identifier of 3. The name-list parameter must have the value of the graphics work station device description. In addition, this call passes processing option group 11 (so that the plotter pen runs slowly for transparencies) and group 12 (so that the lines drawn are wider). Each group requires two 4-byte binary integers; therefore there are four values in PROCOPT_LIST.

```
OPTION BASE 1
INTEGER PROCOPT_LIST
DIM PROCOPT_LIST(4)
MAT READ PROCOPT_LIST
DATA  11, 30, 12, 7
DIM NAME_LIST$(1)*10
LET NAME_LIST$(1) = 'GRFDSP1'
CALL GDDM('DSOPEN',3,1,'6180    ',4,PROCOPT_LIST(),1,NAME_LIST$())
CALL GDDM('DSUSE',1,3)
```

Example 3:

The following example opens the 5292 Model 2, giving it a device identifier of 2 and the name of its OS/400 device description ('GRFDSP1').

```
OPTION BASE 1
INTEGER PROCOPT_LIST
DIM PROCOPT_LIST(1)
DIM NAME_LIST$(1)*10
LET NAME_LIST$(1) = 'GRFDSP1'
CALL GDDM('DSOPEN',2,1,'5292M2  ',0,PROCOPT_LIST(),1,NAME_LIST$())
CALL GDDM('DSUSE',1,2)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, device families 2, 3, and 4 are not supported.

On the AS/400 system, different device tokens are supported.

On the AS/400 system, different procopt-list option group codes are supported for plotters.

On the AS/400 system, the name-list string is 10 bytes long (instead of 8 bytes) to accommodate AS/400 device names.

DSQDEV – Query Device Characteristics

```
DSQDEV(device-id,device-token,procopt-count,procopt-list,
       name-count,name-list,char-count,char-list)
```

Returns the characteristics of a device with the specified device identifier. A device token, processing options list, and name list are returned in DSOPEN format, including the values of any options that were assigned by default when the device was opened. A further list containing miscellaneous device characteristics is also returned.

In the count parameters, you must specify how many elements each list has. If there is insufficient space, GDDM returns only as many elements as it can; if there is extra space, GDDM does not change the excess elements.

Parameters

device-id (4-byte binary integer)
The identifier of the device whose characteristics you are querying.

device-token (returned by GDDM) (8-byte character variable)
For a list of the possible values, see the description of the device-token parameter under DSOPEN.

procopt-count (4-byte binary integer variable)
Specifies the number of elements in procopt-list. You can specify 0 to indicate that procopt-list is empty and is not to be returned.

procopt-list (returned by GDDM) (array of 4-byte binary integer variables)
The processing option groups that are applicable to the particular device's family. Each option group consists of a number of 4-byte binary integers, the first of which is a code identifying the particular option. Details of processing option groups, and the families to which they apply, can be found under DSOPEN.

name-count (4-byte binary integer)
Specifies the number of 10-byte elements in name-list. 0 can be specified to indicate that name-list is empty and is not to be returned.

name-list (returned by GDDM) (array of 10-byte character strings)
The device description name by which the device is known to the OS/400 program. On the AS/400 system, only zero through two elements are valid in the array. Again, see the description under DSOPEN for the meaning of these tokens for various device families.

char-count (4-byte binary integer)
The number of elements in char-list (must be 0 through 15). You can specify 0 to indicate that char-list is empty and is not to be returned.

char-list (returned by GDDM) (array of 4-byte binary integer variables)
Contains the following information:

1. Device family:

   1 Graphics work station, IBM plotters, graphics-capable printers, or the dummy device support provided by device token L79A3.

2. Device input/output capability:

   1 Dummy device
   2 Device supports input and output
   3 Device supports output only

3. Page depth:

   The maximum number of rows usable for a page created in an application or the page depth of the default page.

4. Page width:

   The maximum number of columns usable for a page created in an application or the page width of the default page.

5. Character box depth:

   The number of pixels in the vertical dimension of each character box in the default image symbol set associated with the device. For plotters and IPDS printers (which have no default image symbol set), this number represents the number of pixels per row.

6. Character-box width:

   The number of pixels in the horizontal dimension of each character box in the default image symbol set associated with the device. For plotters and IPDS printers (which have no default image symbol set), this number represents the number of pixels per column.

7. Vertical resolution:

   The number of pixels in the vertical dimension per meter.

8. Horizontal resolution:

   The number of pixels in the horizontal dimension per meter. For all printers except IPDS, this value varies with the characters-per-inch value held in the current printer file. At 15 cpi, the pixels per meter are more dense than they are at 10 cpi.

   For IPDS printers, the horizontal resolution is independent of the CPI value of the print file.

9. Reserved: 0
10. Reserved: 0
11. Reserved: 0
12. Number of colors that can be drawn at one time
13. Reserved: 0
14. Reserved: 0
15. Reserved: 0

Coding Example

In the following example, the program queries device 2:

```
OPTION BASE 1
INTEGER PROCOPT_LIST,CHAR_LIST
DIM PROCOPT_LIST(9)
DIM NAME_LIST$(1)*10
DIM CHAR_LIST(15)
CALL GDDM('DSQDEV',2,DEVTOK$,9,PROCOPT_LIST(),1,
     NAME_LIST$(),12,CHAR_LIST())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the name-list character strings are 10 bytes long instead of 8 bytes.

New procopt-list option group codes are supported for plotters.

DSQUID – Query Unique Device-ID

DSQUID(unique-device-id)

Returns a unique unused device-id. This call can be used by a modular application program to obtain an identifier for a new device, without conflicting with a device that has already been opened by another part of the application. The device-id returned is the highest available unused number.

Parameters

unique-device-id (returned by GDDM) (4-byte binary integer variable)
A device identifier for which no device currently exists.

Coding Example

In the following example, GDDM returns an unused device identifier in variable DEVID:

```
INTEGER DEVID
CALL GDDM('DSQUID',DEVID)
```

DSQUSE – Query Device Usage

```
DSQUSE(usage,device-id)
```

Returns the identifier of the device currently operating with the specified usage.

Parameters

usage (4-byte binary integer)
Must be a valid usage code as described in the DSUSE routine description. GDDM returns (in the device-id parameter) the device identifier of the device currently having this usage. On the AS/400 system, the value returned is always 1.

device-id (returned by GDDM) (4-byte binary integer variable)
The device identifier of the device currently operating with the usage specified on the usage parameter. If no device is using the specified usage, a value of −1 is returned.

Coding Example

In the following example, GDDM returns the currently used device identifier in variable DEVID:

```
INTEGER DEVID
CALL GDDM('DSQUSE',1,DEVID)
```

DSRNIT – Reinitialize Device

```
DSRNIT(device-id,option)
```

Restores the status of a device to the status it had just after it was first explicitly opened. DSRNIT frees all resources (such as page contents, symbol sets, and color tables) that have been defined for the device. This is the same as doing a DSCLS followed by a DSOPEN.

Any usage currently in force for this device is discontinued.

Note that DSRNIT differs from most GDDM routines because it attempts to complete even if an error is returned.

On display devices, the graphics are erased when DSRNIT is called.

Parameters

device-id (4-byte binary integer)
The identifier of the device to be reinitialized.

option (4-byte binary integer)
Specifies an action to be carried out when the device is reinitialized. It must have the following value:

0 Erase screen.

Coding Example

In the following example, the application program reinitializes the device identified as 2:

```
.
.       (graphics processing that opens a device with identifier 2)
.
CALL GDDM('DSRNIT',2,0)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the graphics are always erased; you cannot specify an option to keep the graphics on the screen.

DSUSE – Specify Device Usage

DSUSE(usage,device-id)

Activates a device that has been opened, and specifies which usage the application can make of the device. The device activated becomes the one used by subsequent graphics calls. For example, the current primary device is the one within which a page would be created if an FSPCRT call were issued.

The following rules apply to DSUSE:

1. The device must already be open (DSOPEN routine).
2. There must be no other device currently operating with the specified usage; if there is, an error message is issued unless DSDROP is used to drop one of the devices.

Note that, when a routine is called that requires a device to be in use, but none is in use, GDDM issues an internal DSUSE against the default primary device.

Parameters

usage (4-byte binary integer)
The device usage code. The following value is valid:

1 Current primary device.

device-id (4-byte binary integer)
The device identifier of the device concerned.

Coding Example

In the following example, the program first uses device 2 as the primary device, then drops the use of device 2 as primary device and starts using device 3 as the current primary device.

```
CALL GDDM('DSUSE',1,2)
   .
   . (graphics processing)
   .
CALL GDDM('DSDROP',1,2)
CALL GDDM('DSUSE',1,3)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, secondary device support is not provided (usage is 2).

FSALRM – Sound Device Alarm

FSALRM

Sounds the work station alarm at the next call to ASREAD or FSFRCE.

FSALRM is ignored for devices that do not have alarms (such as plotters).

Parameters

None

Coding Example

In the following example, the program detects an unusual situation, such as an overstock of parts in inventory, then constructs an appropriate display (or constructs a warning to include in the display). The FSFRCE causes the alarm to be sounded, if the device has an alarm.

```
    .
    .  (program detects unusual situation)
    .
CALL GDDM('FSALRM')
    .
    .  (program constructs appropriate display, as required)
    .
CALL GDDM('FSFRCE')
```

FSEXIT – Specify an Error Exit and/or Error Severity

```
FSEXIT(user-error-program,severity)
```

Specifies an error program that you have created that receives control if, on a graphics call, a graphics error of at least the specified severity is encountered. (Graphics error messages have message identifiers beginning CPG.) This call can also be used to change the severity threshold at which messages are sent to the specified program.

All messages below the specified severity are sent to your job log. Messages at or above the specified severity are sent to the user exit program.

Note: You can use this call to have more messages sent to the low-level messages display at your display station (which you display by pressing F7 from the command entry display). This avoids having to display your job log or to set up a user error program to see messages. Specify *SAME or *NONE for the user-error-program parameter, and specify a value less than 40 (such as 20 or 30) for the severity parameter.

A return from the error program gives control to the statement after the one that invoked GDDM.

The error program is passed a single parameter, a 569-byte error record.

FSEXIT can be called at any time. Subsequent FSEXIT calls override the error program and severity specifications of the previous FSEXIT.

Parameters

user-error-program (8-byte character variable)

The name of the program to receive control. The program must be in the library list of the user that calls the graphics application program.

If you have not changed the error exit program parameter and an error whose severity is equal to or greater than the current severity level occurs, the OS/400 program sends escape message CPF8619 to your graphics application program, thus terminating your program with a function check.

The defaults are reset when you call FSTERM. A user error exit program is not given control as the result of errors arising from FSTERM.

The following special values can be specified for this parameter:

'*NONE '	No user error exit program is to be used; instead, the default error exit program is used.
'*SAME '	The error exit program is to remain unchanged, allowing only the severity parameter to be changed.

severity (4-byte binary integer)
Specifies the minimum severity that causes the user error program to receive control. You can specify any value from 0 through 99. Severity values used by OS/400 Graphics are:

00 No error
10 Warning
20 Error
30 Severe error
40 Unrecoverable error

When a severity of zero is specified, the error exit is invoked after *each* call to a GDDM routine, whether or not an error has occurred. In such cases, each error record will contain only information about the most recent GDDM CALL statement. If you specify a severity of 41 or above, the user error program is never invoked.

When graphics is initialized, the default severity is 40 (unrecoverable error).

Coding Example

In the following example, if an error with a severity greater than 40 occurs during input processing, the graphics program calls program INERR; if the error occurs during output processing, the program calls OUTERR.

```
CALL GDDM('FSEXIT','INERR     ',40)
   .
   .  (input processing)
   .
CALL GDDM('FSEXIT','OUTERR    ',40)
   .
   .  (output processing)
   .
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system a special value of *NONE is used instead of zero to reset the user-error-program parameter.

On the AS/400 system there is a new value, *SAME. This value changes the severity parameter while leaving the user-error-program parameter unchanged.

FSFRCE – Display Outstanding Graphics

FSFRCE

Performs all outstanding graphics output. Unlike ASREAD, FSFRCE does not wait for a response from the work station user when data is sent to a display device.

Use FSFRCE when you also want to use externally described record formats, which you define using DDS. You can use externally described record formats to display alphanumeric data with graphics data. You can also use externally described record formats to give the work station user control over the actions taken by your program after seeing the display. (The work station user does this by entering data into input-capable fields or by pressing function keys, both of which you allow through DDS.)

Use ASREAD when you just want to display graphics and wait; *any* response from the work station user (including any F key) is sufficient to continue operations.

Parameters

None

Coding Example

In the following example, the graphics program calls FSFRCE to display graphics, then does a WRITE operation to display an externally described record format. The externally described record format (defined in DDS, not shown) contains the ALWGRH keyword and allows the work station user to press the function keys specified in the record format.

```
    .
    .  (graphics processing)
    .
CALL GDDM('FSFRCE')
WRITE #1, EXTDESCR 'CUSFLDS', INDIC INDICATOR$:
READ #1, EXTDESCR 'CUSFLDS', INDIC INDICATOR$:
    .
    .  (further processing depending on the user response)
    .
```

FSINIT – Initialize Graphics

FSINIT

Initializes graphics processing.

This routine must be the first graphics routine to be executed.

The initialization process sets graphics variables to their default values.

Parameters

None

Coding Example

In the following example, the graphics program calls FSINIT to begin graphics processing.

```
      .
      .  (nongraphics processing)
      .
CALL GDDM('FSINIT')
      .
      .  (graphics processing)
      .
```

FSPCLR – Clear the Current Page

FSPCLR

Deletes all graphic fields within the current page. Also resets the current color table and the current symbol set to the IBM-supplied defaults.

Parameters

None

Coding Example

The following example shows how to code FSPCLR:

```
CALL GDDM('FSPCLR')
```

FSPCRT – Create a Page

```
FSPCRT(page-id,depth,width,type)
```

Creates a page of the specified size and type. The new page is empty.

Use FSPCRT to do the following:

- Draw more than one picture at the same time.
- Send the same picture to both a display and a plotter (you must put the same picture into two pages).
- Reset all attributes, including the graphics field, picture space, viewports, and window, to default values.

The following shows the position of this call in the graphics hierarchy:

Page	**FSPCRT**
Graphics field	GSFLD
Picture space	GSPS
Viewport	GSVIEW
Window	GSWIN
Segment	GSSEG

To make a picture smaller, you can also use GSFLD.

For displays, the page must fit on the screen (its depth and width must not be greater than the depth and width of the display screen).

For plotters, the depth and width parameters are ignored. The entire page is used (see processing option group 15, plotter paper size, under DSOPEN).

If you specify FSPCRT, it must follow the device control routines (such as DSOPEN, DSUSE, DSRNIT, and DSQUID).

By default, the new page is also the one that is currently selected. To select a page other than the one most recently created, use FSPSEL.

If GSVIEW, GSPS, and GSWIN have not been called, the default values are as follows:

- On an IBM PC or 5292 Model 2, the depth is 24 and the width is 80.
- On the IBM plotters, the depth is 24 and the width is 80.
- On a printer, the depth and width of the page are dependent on the forms width, overflow line, number, and cpi (characters per inch) value of the printer file. For more information, refer to the *GDDM Programming Guide*.

Parameters

page-id (4-byte binary integer)
The page identifier for the new page. It must be greater than zero. Zero is reserved for the page identifier of the default page, which is always available.

depth,width (4-byte binary integers)
The size of the page, in rows and columns, starting at (1,1) in the upper left corner. If either depth or width is zero, the appropriate depth or width according to the size of the display screen is used.

type (4-byte binary integer)
This parameter is ignored on the AS/400 system, but it must be 0 through 3 for compatibility with GDDM on other systems.

Coding Example

In the following example, the program creates a page with identifier 1; with depth and width of zero, the page is the size of the display screen or plotter page.

```
CALL GDDM('FSPCRT',1,0,0,0)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the type parameter is ignored.

FSPDEL – Delete a Page

```
FSPDEL(page-id)
```

Deletes a page. This also causes any graphic segments associated with the page to be deleted.

If this page was the current page, the default page becomes the new current page.

The default page (identifier zero) cannot be deleted.

Parameters

page-id (4-byte binary integer)
Identifies the page to be deleted.

Coding Example

In the following example, the program deletes the page with identifier 1:

```
CALL GDDM('FSPDEL',1)
```

FSPQRY – Query Specified Page

```
FSPQRY(page-id,depth,width,type)
```

Returns the depth, width, and type of the specified page. If you have not used FSPCRT, the depth, width, and type of the default page (created by GDDM) are returned.

Parameters

page-id (4-byte binary integer)
Identifies the page about which information is required.

depth,width (returned by GDDM) (4-byte binary integer variables)
The dimensions of the page. The depth parameter gives the number of rows; the width parameter gives the number of columns.

type (returned by GDDM) (4-byte binary integer variable)
The page type as defined for FSPCRT. On the AS/400 system, GDDM always returns a value of zero.

Coding Example

In the following example, the program requests the depth, width, and type of page 2:

```
INTEGER DEPTH,WIDTH,TYP
CALL GDDM('FSPQRY',2,DEPTH,WIDTH,TYP)
```

If the page is using the full display screen on a graphics work station, DEPTH is 24, WIDTH is 80, and TYP is undefined after the call completes.

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the type parameter is ignored, and GDDM returns a value of zero.

FSPSEL – Select a Page

```
FSPSEL(page-id)
```

Selects a page and makes it the current one. This has two effects:

- When the device is updated (through the use of ASREAD or FSFRCE), only the current page is displayed.
- Any subsequent graphics field creation or reference is associated with this page.

Only one page can be selected at any time, and the specified page remains selected until it is deleted, or until another page is explicitly selected, or until a new page is created.

Parameters

page-id (4-byte binary integer)
Identifies the page to be selected. A page-id of zero causes the default page to be selected.

Coding Example

In the following example, the program first selects page 2, then selects the default page:

```
CALL GDDM('FSPSEL',2)
.
.
.
CALL GDDM('FSPSEL',0)
.
.
.
```

FSQCPG — Query Current Page-ID

```
FSQCPG(page-id)
```

Returns the page-id of the current page.

Parameters

page-id (returned by GDDM) (4-byte binary integer variable)
The identifier of the current page.

Coding Example

In the following example, the program queries GDDM for the page-id of the current page:

```
INTEGER PAGEID
CALL GDDM('FSQCPG',PAGEID)
```

FSQDEV – Query Device Characteristics

```
FSQDEV(count,array)
```

Returns the characteristics of the current primary device.

Parameters

count (4-byte binary integer)

The number of elements in the array parameter. If you specify fewer elements than are required, information about the remaining attributes is not returned. If you specify more elements than are returned by GDDM, the excess elements in your array are not changed.

Note that this routine returns the characteristics of the current primary device. The DSQDEV call (described earlier) can be used instead, either to determine additional information about this device, or to determine information about a device that is not the current primary device.

array (returned by GDDM) (array of 4-byte binary integer variables)

Refer to the DSQDEV routine for a description of the values that can be returned.

Coding Example

In the following example, the program queries GDDM for the first two characteristics of the current primary device (family and input/output capability). With count equals 2, only two characteristics are returned. The program then tests the value of the second element in array DEVCHAR and performs the appropriate subroutine (DUMMY if the device is a dummy device; DISP if the device is input/output capable; PLOT if device is output only).

```
OPTION BASE 1
INTEGER DEVCHAR
DIM DEVCHAR(2)
CALL GDDM('FSQDEV',2,DEVCHAR())
IF DEVCHAR(2) = 1 THEN GOSUB DUMMY
IF DEVCHAR(2) = 2 THEN GOSUB DISP
IF DEVCHAR(2) = 3 THEN GOSUB PLOT
```

FSQERR – Query Last Error

```
FSQERR(length,error-record)
```

Returns information about the last graphics error, that is, the last graphics call that resulted in a message with a severity other than zero. Only messages with CPG in the message identifier are queried. If no error has occurred since initialization, or since the last call to FSQERR, a dummy result is returned (in the error record returned by GDDM, character fields are all blanks and numeric fields are all zeros).

Parameters

length (4-byte binary integer)
The length in bytes of storage provided to receive the error data. The number of bytes of information provided is that specified in the length parameter, or 564, whichever is smaller.

error-record (returned by GDDM) (character variable)
The data describing the error. For a description of the structure of the error record, refer to the *GDDM Programming Guide*.

Coding Example

In the following example, the program requests information about the most recent error:

```
OPTION BASE 1
DIM ERROR$*255
CALL GDDM('FSQERR',255,ERROR$)
```

In BASIC, the maximum length of a character string is 255. Therefore, you cannot retrieve all the data supplied by GDDM. However, most of the useful data is in the first 100 bytes returned. Therefore, this string length is large enough for most purposes.

Differences between the System/370 Computer and the AS/400 System

The format and length of the error record are different on the two systems.

FSQUPG – Query Unique Page-ID

```
FSQUPG(page-id)
```

Returns a unique unused page-id. This call can be used by a modular application program to obtain an identifier for a new page. The page-id returned is the highest available unused number that does not conflict with a page already created by another part of the application program.

Parameters

page-id (returned by GDDM) (4-byte binary integer variable)
An identifier for which no page currently exists.

Coding Example

In the following example, the program requests a unique page-id:

```
INTEGER PAGEID
CALL GDDM('FSQUPG',PAGEID)
```

FSREST – Retransmit Data

```
FSREST(code)
```

Causes all retained data (data included in segments defined with GSSEG) in the current page to be sent to the device again the next time your program calls FSFRCE or ASREAD.

Note: This also occurs when an operation against a page or segment is performed (such as create or delete) even if FSREST has not been called since the last call to FSFRCE or ASREAD. Use FSREST to make sure that all retained data in the current page is sent to the device on the next FSFRCE or ASREAD.

Parameters

code (4-byte binary integer)
Must be 0 (data only).

Coding Example

The following example shows how to use FSREST:

```
CALL GDDM('FSREST',0)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, the code parameter may not be set to values other than 0.

FSRNIT – Reinitialize Graphics

FSRNIT

Reinitializes GDDM. This function is equivalent to FSTERM followed by FSINIT, except that GDDM retains information about the current device, for example, the device currently selected for use by DSUSE.

FSRNIT does not reinitialize Presentation Graphics routines. If you are using Presentation Graphics routines, terminate them with CHTERM before using FSRNIT. (Issuing a call to FSRNIT before CHTERM produces unpredictable results.)

Use FSRNIT as an alternative to FSTERM and FSINIT, when you want to restart GDDM with no memory of any previously defined graphics resources such as symbol sets, pages, or color tables, but the same current device is to remain open.

FSRNIT causes an internal DSRNIT call to be issued against the current device (whether explicitly opened, or opened by default), and an internal DSCLS call to be issued against any other open devices.

After FSRNIT, the device that had been in use as the current device remains open (no DSOPEN is necessary) but is no longer in use (see DSRNIT). Before it can be used, it must again be made the current device using DSUSE (although, as always, an implicit DSUSE will be issued against the default primary device if necessary). Other devices will need to be reopened using DSOPEN before they can be reused.

Parameters

None

Coding Example

The following example shows how to use FSRNIT:

```
CALL GDDM('FSRNIT')
```

FSTERM – Terminate Graphics

FSTERM

Terminates graphics processing for the application program and frees all the resources (such as symbol sets, pages, and color tables) acquired.

To do the necessary housekeeping, this routine must be the last graphics routine executed in a program.

Parameters

None

Coding Example

The following example shows how to specify FSTERM:

```
CALL GDDM('FSTERM')
```

GSARC – Draw a Circular Arc

GSARC(xc,yc,angle)

Draws a circular arc about the specified center point, starting at the current position. You must specify a picture space with a width-to-height ratio of one to one for the arc to appear circular on the graphics work station. The angle parameter specifies in degrees how much of the arc to draw. When the angle is positive, the arc is drawn counterclockwise from the current position; when negative, the arc is drawn clockwise.

Note that the direction of the arc is determined in world coordinate space, assuming the x axis runs left to right and the y axis runs from bottom to top. The direction might appear to be reversed on the display surface if the lower window limit is larger than the upper limit on either axis.

The color, line width, and line type of the arc are given by the current values of these attributes.

After the arc is drawn the current position is set to the end of the arc.

Parameters

xc,yc (short floating-point numbers)
Specify the coordinates of the center as an absolute point in world coordinates.

angle (short floating-point number)
The angle used by the arc in degrees. When the angle is positive, the arc is drawn counterclockwise; when it is negative, the arc is drawn clockwise.

Coding Examples

Example 1:

The following example shows how to draw a complete circle (outline only) with the center at x is 50 and y is 50:

```
! The following sets the aspect ratio of the picture space to 1:1
CALL GDDM('GSPS', 1.0, 1.0)
! Move current position
CALL GDDM('GSMOVE', 20.0, 20.0)
! The following draws the circle
CALL GDDM('GSARC', 50.0, 50.0, 360.0)
```

Example 2:

The following example shows how to draw a complete circle (the outline, filled with a pattern) with the center at x is 50 and y is 50:

```
! Set the aspect ratio of the picture space
CALL GDDM('GSPS', 1.0, 1.0)
! Move current position
CALL GDDM('GSMOVE', 20.0, 20.0)
! Start an area that will be filled with a pattern
CALL GDDM('GSAREA',1)
! Draw the circle
CALL GDDM('GSARC', 50.0, 50.0, 360.0)
CALL GDDM('GSENDA')
```

The pattern appears because the program calls GSAREA and GSENDA, and the outline appears because the parameter on GSAREA has a value of one.

GSAREA – Start a Shaded Area

GSAREA(boundary-control)

Begins the construction of a shaded area. The construction is ended by GSENDA (see "GSENDA – End a Shaded Area" on page 2-74).

The only drawing routines that can be used between GSAREA and GSENDA are:

GSARC
GSELPS
GSPFLT
GSLINE
GSLT
GSLW
GSCOL
GSMIX
GSMOVE
GSPLNE
GSVECM

The area boundary consists of one or more *closed* figures, which you construct by calling GSARC, GSLINE, GSPLNE, GSELPS, GSPFLT, or an element of GSVECM having a control value of 1. You can intersperse calls to GSLT, GSLW, GSCOL, and GSMIX among the other calls to change line attributes while you construct the figure.

The starting point of each closed area is the current position when GSAREA is called, or as specified by a subsequent call to GSMOVE. GDDM closes the area if you specify a call to GSMOVE, to an element of GSVECM with a control value of zero, or to GSENDA. The end point of the figure is the current position resulting from the last line or arc drawn.

Each figure should be closed; that is, the starting point and end point should be identical. If this is not the case, GDDM closes the figure arbitrarily, by a straight line connecting the start and end points.

The figures formed in this way jointly define the area boundary. Any enclosed area is shaded (with the current pattern and color when the GSAREA call is made) if an odd number of boundary lines must be crossed to leave the area. If an even number of boundary lines must be crossed to leave an area, the area is not shaded.

If the boundary control parameter of the GSAREA call is zero, the actual boundary lines are *not* drawn, but the shading ends at the boundaries. If the boundary control parameter is 1, boundary lines and any lines added to close figures are drawn.

The current position is not changed by the GSAREA command, but can be changed by the moves and lines between GSAREA and GSENDA, including any used to close figures.

Area definitions cannot be nested.

Parameters

boundary-control (4-byte binary integer)
Specifies whether or not the boundary lines are to be drawn:

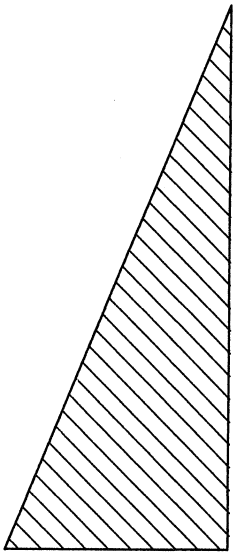
0 Do not draw boundary lines
1 Draw boundary lines

Coding Examples

Example 1:

The following example shows how to draw a shaded area with an outline:

```
CALL GDDM('GSAREA',1)
CALL GDDM('GSLINE',12.0,40.0)
CALL GDDM('GSLINE',12.0,0.0)
CALL GDDM('GSENDA')
```



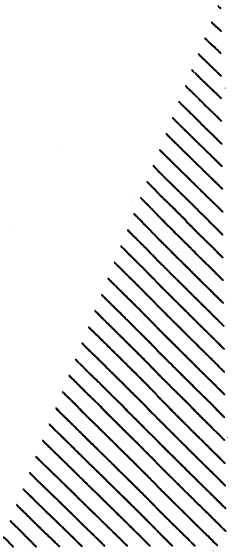
RV2F719-0

The bottom line in Example 1 is drawn implicitly by graphics when the GSENDA call is completed.

Example 2:

The following example shows how to draw a shaded area without an outline:

```
CALL GDDM('GSAREA',0)
CALL GDDM('GSLINE',12.0,40.0)
CALL GDDM('GSLINE',12.0,0.0)
CALL GDDM('GSENDA')
```



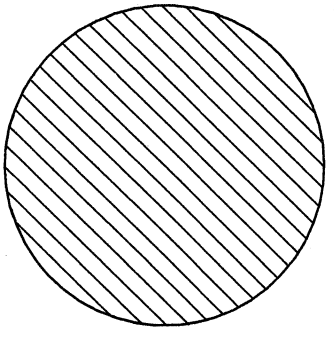
RV2F720-0

Example 3:

The following example shows how to draw a complete circle (the outline, filled with a pattern) with the center at x is 50 and y is 12:

```
! Move current position away from corner of picture space
CALL GDDM('GSMOVE',50.0,0.0)
! Start area fill
CALL GDDM('GSAREA',1)
! Draw the circle
CALL GDDM('GSARC', 50.0, 12.0, 360.0)
! End area fill
CALL GSENDA
```

The pattern appears because the program calls GSAREA and GSENDA, and the outline appears because 1 is specified as the boundary-control parameter. The pattern shown is pattern 11.



RV2F721-0

GSCA – Set Current Character Angle

GSCA(dx,dy)

Specifies the angle of the baseline along which characters in subsequent graphics text are written.

This is valid only in character-mode 2 or 3 (see "GSCM – Set Current Character Mode" on page 2-64). GSCA affects mode 2 and mode 3 characters differently. In mode 2, characters are not rotated to lie along the baseline; in mode 3, they are rotated to lie along the baseline. See Examples 1 and 2 later in this function description.

Parameters

dx,dy (short floating-point numbers)
Define the angle of the baseline with the horizontal.

The dx and dy parameters have the following effects:

- When dx equals 0 and dy equals 1 (default), the baseline is vertical.
- When dx is greater than 0 and dy is greater than 0 the baseline rises to the right. The baseline is between 0° and 90° to the baseline. For example, if dx and dy are equal, the angle is 45°.
- When dx is less than 0 and dy is greater than 0 the baseline rises to the left. The baseline is between 90° and 180° to the baseline.
- When dx is greater than 0 and dy is less than 0 the baseline declines to the right.
- When dx is less than 0 and dy is less than 0 the baseline declines to the left.

For a baseline at angle A, dx is cos(A) and dy is sin(A).

Coding Example

Example 1:

The following example shows how GSCA affects mode 2 characters:

```
! Set character mode to 2
CALL GDDM('GSCM',2)
.
.
.
! Set baseline angle to 45 degrees
CALL GDDM('GSCA',1.0,1.0)
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string appears after using GSCA:

RV2F722-0

Example 2:

The following example shows how GSCA affects mode 3 characters:

```
! Set character mode to 3
CALL GDDM('GSCM',3)
.
.
.
! Set baseline angle to 45 degrees
CALL GDDM('GSCA',1.0,1.0)
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string appears after using GSCA:

RV2F723-0

GSCB – Set Current Character Box Size

```
GSCB(width,height)
```

Sets the width and height of the character box for subsequent characters. The width parameter sets the spacing between characters along the line of writing (which is defined by GSCA). The height parameter sets the spacing between consecutive lines of characters. GSCB has the following effects:

- For mode 2 characters (see "GSCM – Set Current Character Mode" on page 2-64), GSCB has the following effects:
  - On the display, GSCB changes only the spacing of the characters, not the size of the characters.
  - On the plotter, the width and height parameters set the size (and the aspect ratio) of the characters themselves. Each character fills one character box.
- For the 4214, 4234-2, 5224, and 5225 printers, GSCB changes only the spacing of the characters, not the size of the characters.
- For IPDS printers, GSCB changes the spacing of the characters and also adjusts the character size within the character box in integer multiples of the standard size character.
- For mode 3 characters (see "GSCM – Set Current Character Mode" on page 2-64), the width and height parameters set the size (and the aspect ratio) of the characters themselves. Each character fills one character box. See Example 2 on page 2-52.

Parameters

width,height (short floating-point numbers)

The width and height of the character box in world coordinate units.

Both the width and height must be positive. The width determines the size and/or spacing of consecutive characters along the line of writing. The height determines the spacing of consecutive lines.

If GSVIEW, GSPS, and GSWIN have not been called, the default values are as follows:

Device	Default width	Default height
Graphics work station	1.250	4.166
6180/7371/7372 plotter	1.200	2.833
Printers	0.757	1.111

Coding Examples

Example 1:

The following example shows how GSCB affects mode 2 characters:

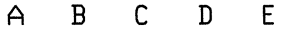
```
! Set character mode to 2
CALL GDDM('GSCM',2)
.
.
.
! Set width and height of character box to 10 world
  coordinate units each:
CALL GDDM('GSCB',10.0,10.0)
.
.
.
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string would appear before using GSCB:

RV2F724-0

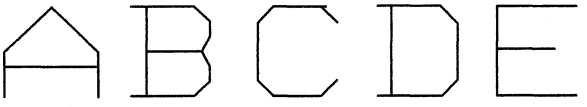
The following shows how the string would appear after using GSCB:

On the display:



RV2F725-0

On the plotter:



RV2F726-0

Example 2:

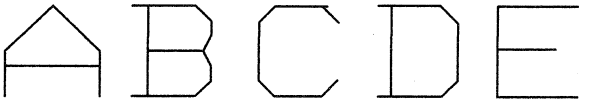
The following example shows how GSCB affects mode 3 characters:

```
! Set character mode to 3
CALL GDDM('GSCM',3)
.
.
! Set width and height of character box to world
  coordinate units each:
CALL GDDM('GSCB',10.0,10.0)
.
.
.
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string would appear before using GSCB:

RV2F727-0

The following shows how the string would appear after using GSCB:



RV2F728-0

GSCD – Set Current Character Direction

GSCD(character-direction-code)

Specifies the direction in which the characters in a string are to be drawn, relative to the baseline specified by GSCA.

This is valid only in character mode 2 or 3 (see "GSCM – Set Current Character Mode" on page 2-64).

Parameters

character-direction-code (4-byte binary integer)
Can have one of the following values:

0 The default; same as 1.
1 Character boxes (defined by GSCB) are arranged parallel to, and directed along, the baseline. New-line direction is 90° clockwise from the baseline. This is the normal convention for roman text.
2 Character boxes are arranged in columns directed 90° *clockwise* from the baseline. New-column direction is the reverse of the baseline direction. This option can be used for drawing roman text vertically (a y-axis title on a graph, for example).
3 Character boxes are arranged parallel to, but in the reverse of the baseline direction. New-line direction is 90° clockwise from the baseline.
4 Character boxes are arranged in columns directed 90° *counterclockwise* from the baseline. New-column direction is the reverse of the baseline direction.

Coding Example

The following example shows how to use GSCD:

```
! Set character mode to 3
CALL GDDM('GSCM',3)
.
.
.
! Set direction to 2 (90° clockwise from the baseline)
CALL GDDM('GSCD',2)
.
.
.
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string would appear before using GSCD:

ABCDE

The following shows how the string would appear after using GSCD:

A
B
C
D
E

GSCH – Set Current Character Shear

```
GSCH(dx,dy)
```

Shears mode 3 characters to an angle defined by the dx and dy parameters. This causes upright lines in the characters to be inclined to the angle, as if the characters were italic. Horizontal lines in the characters remain horizontal to the baseline. The dx and dy parameters define the angle of the upright character strokes with the baseline.

For mode-2 characters, character shear (GSCH) has no effect except when the new-line character (X'15') is used in a character string, in which case the starting point of the new line is along the line of shear (not necessarily perpendicular to the baseline). See Example 2 on page 2-57.

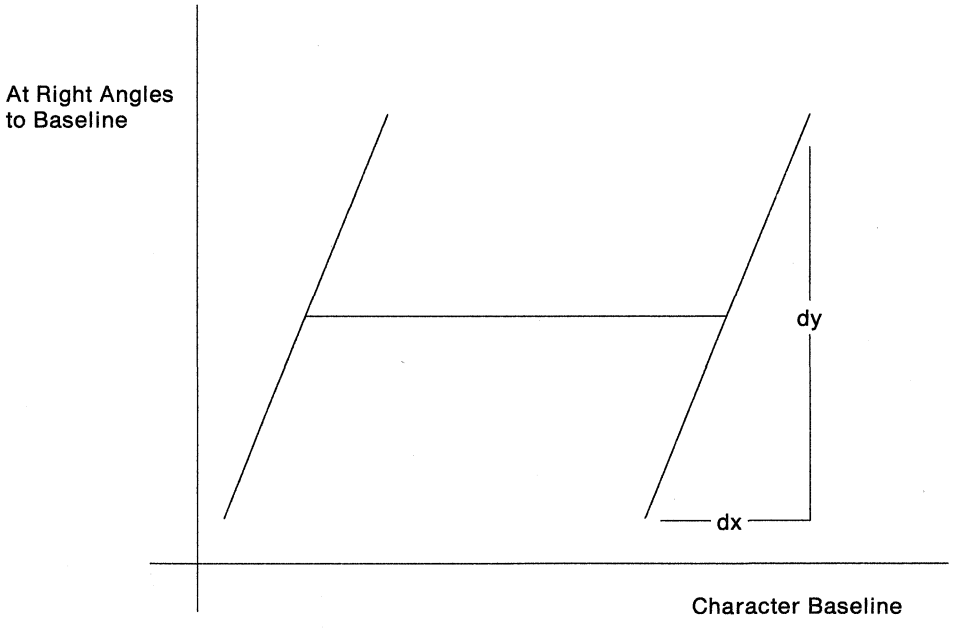
Parameters

dx,dy (short floating-point numbers)

Define the angle to which the characters are sheared, as follows:

- When dx equals 0 and dy equals 1 (default), normal upright characters result. Upright lines in the characters are 90° to the baseline.
- When dx is greater than 0 and dy is greater than 0 the characters slope forwards. Upright lines in the characters are between 0° and 90° to the baseline. For example, if dx and dy are equal, the angle is 45°.
- When dx is less than 0 and dy is greater than 0 the characters slope backwards. Upright lines in the characters are between 90° and 180° to the baseline.
- When dy is less than 0 the characters are inverted.

The following shows how the parameter values define the angle of shear.



RV2F731-0

Coding Examples

Example 1:

The following example shows how GSCH affects mode 2 characters:

```
! Set character mode to 2
CALL GDDM('GSCM',2)
.
.
.
! Set character shear slope forwards
CALL GDDM('GSCH',1.0,2.0)
.
.
.
! Set up character string 'ABCDEFGHIJ' with new-line character
!  after the E
DIM STRING$*11
NEWLINE$ = HEX$('15')
STRING$ = 'ABCDE'&NEWLINE$&'FGHIJ'
CALL GDDM('GSCHAR',50.0,50.0,11,STRING$)
```

The following shows how the string would appear before using GSCH:

ABCDE
FGHIJ

The following shows how the string would appear after using GSCH:

ABCDE
FGHIJ

Example 2:

The following example shows how GSCH affects mode-3 characters:

```
! Set character mode to 3
CALL GDDM('GSCM',3)
.
.
.
! Set character shear slope forwards
CALL GDDM('GSCH',1.0,2.0)
.
.
.
! Draw character string 'ABCDE'
CALL GDDM('GSCHAR',50.0,50.0,5,'ABCDE')
```

The following shows how the string would appear before using GSCH:

RV2F734-0

The following shows how the string would appear after using GSCH:

RV2F735-0

GSCHAP – Draw a Character String at Current Position

```
GSCHAP(length,string)
```

Draws a character string in the graphics field starting at the current position. Also allows a character string to be appended to a previous one. Each character in the string is drawn with its lower left corner at the current position, and the current position is then advanced to the starting position of the next character. After using GSCHAP, the current position is at the end of the string. For an overview of attribute settings that affect character strings, see "GSCM – Set Current Character Mode" on page 2-64.

Parameters

length (4-byte binary integer)
The number of characters in the string parameter.

string (character string)
The character string to be drawn. Control and undefined values are reserved and should not be used, except for X'15', which is interpreted as a new-line character. See Example 2 on page 2-59. Each new-line character in the string sets the current position one character box height (set by GSCB) *below* the start of the string. *Below* is strictly defined by a line whose direction is 90° clockwise from the character baseline (set by GSCA). For an easy way to control the direction a string reads, see "GSCD – Set Current Character Direction" on page 2-53.

Hex FF has a reserved meaning and should not be used.

In BASIC, you cannot directly specify how to set bits on and off but you can use the HEX$ built-in function. In RPG III, you could use the BITON and BITOFF operation codes; in PL/I, you could use BIT data; in the COBOL/400 language, there is no direct way of manipulating bit strings; you might want to call a program in another language, passing a character string to another program to initialize and return with the correct bit values.

Coding Examples

Example 1:

The following example shows how to use GSCHAP:

```
CALL GDDM('GSCHAP',23,'Sample Character String')
```

Example 2:

The following example shows how the new-line character affects character strings:

```
DIM STRING$*11
! Set up character string 'ABCDEFGHIJ' with new-line character
!  after the E
NEWLINE$ = HEX$('15')
STRING$ = 'ABCDE'&NEWLINE$&'FGHIJ'
CALL GDDM('GSCHAP',11,STRING$)
```

The following shows how the string would appear at the current position:

ABCDE
FGHIJ

GSCHAR – Draw a Character String at a Specified Point

```
GSCHAR(x,y,length,string)
```

Draws a character string starting at the specified point.

GSCHAR(x,y,length,string) is equivalent to:

```
GSMOVE(x,y)
GSCHAP(length,string)
```

Each character in the string is drawn with its lower left corner at the current position, and the current position is then advanced to the starting position of the next character. For an overview of attribute settings that affect character strings, see "GSCM – Set Current Character Mode" on page 2-64.

After the string is drawn, the new current position is the point at which the next character after the end of the string would be drawn. To find out the new current position, use GSQCP.

If clipping is enabled, any part of any character that is outside the window is removed. Also, for mode-2 characters, the entire character is clipped if any part of the character box is outside the window.

When clipping is disabled, and any part of the string is outside the window, the result is undefined.

Parameters

x,y (short floating-point numbers)
The starting position of the string in world coordinates.

length (4-byte binary integer)
The number of characters in the string supplied.

string (character string)
The character string to be drawn, as described for GSCHAP.

Coding Examples

Example 1:

The following example shows how to use GSCHAR:

```
CALL GDDM('GSCHAR',10.0,5.0,23,'Sample Character String')
```

Example 2:

The following example shows how the new-line character affects character strings:

```
DIM STRING$*11
! Set up character string 'ABCDEFGHIJ' with new-line character
!  after the E
NEWLINE$ = HEX$('15')
STRING$ = 'ABCDE' &NEWLINE$&'FGHIJ'
CALL GDDM('GSCHAR',50.0,50.0,11,STRING$)
```

The following shows how the string would appear:

ABCDE
FGHIJ

GSCLP – Enable and Disable Clipping

GSCLP(clipping-mode)

Enables and disables clipping to window boundaries. The setting applies only to the current page.

Clipping should be enabled when there is reference to world coordinates that lie outside the window. When clipping is enabled, parts of the picture that fall outside the window boundaries (defined by GSWIN) are removed. If a primitive, for example a line, falls completely outside the window, it is discarded. If a primitive falls partly within and partly outside the window, the primitive is clipped to keep only that part within the window. Lines and arcs are shortened so as to stop at the window boundary. Markers appear if their center points are within the window. For character strings, the result depends on the character mode; see "GSCM – Set Current Character Mode" on page 2-64.

When clipping is disabled, all parts of the drawing should lie within the window. If they do not, the results are undefined.

Parameters

clipping-mode (4-byte binary integer)
Controls the clipping state, as follows:

0 Clipping disabled (default)
1 Clipping enabled

Coding Example

The following example shows how to use GSCLP:

```
CALL GDDM('GSCLP',1)
```

GSCLR – Clear the Graphics Field

GSCLR

Clears the graphics field by deleting all of its graphics segments. The effect does not appear on the display until the next transmission takes place (that is, on the next call to ASREAD or FSFRCE).

Parameters

None

Coding Example

The following example shows how to use GSCLR:

```
CALL GDDM('GSCLR')
```

GSCM – Set Current Character Mode

```
GSCM(character-mode)
```

Controls the character mode to be used in drawing character strings in the graphics field.

You must use GSCM to set character mode 2 or 3 before using GSCA, GSCB, GSCD, or GSCH. You must also use GSCM to set character mode 2 or 3 for a GSCS call to take effect.

Parameters

character-mode (4-byte binary integer)
Interpreted as follows:

0 Default; use this code to return to using default hardware characters after using another character mode. Character box (GSCB), angle (GSCA), shear (GSCH), and direction (GSCD) are ignored. Characters are positioned horizontally in adjacent hardware cells.

1 Not valid on the AS/400 system.

2 Graphics image symbols (defined by matrices of dots). The character set must be the standard symbol set 0, or be a graphics image symbol set previously loaded by GSLSS. Character angle (GSCA), box (GSCB), direction (GSCD), and shear (GSCH) affect only the string, not individual characters in the string. Character angle (GSCA) rotates the baseline, but individual characters are not rotated to lie along the baseline. Character box (GSCB) changes the spacing between characters and between lines of characters.

Note: For IPDS printers, GSCB adjusts the character size within the character box in integer multiples of the standard size character.

Character direction (GSCD) changes the direction in which the string reads. Character shear (GSCH) has no effect except when the new-line character (hex 15) is used in a character string, in which case the starting point of the new line is along the line of shear (not necessarily perpendicular to the baseline; see "GSCH – Set Current Character Shear" on page 2-55). When clipping is enabled (see "GSCLP – Enable and Disable Clipping" on page 2-62) and any part of a character box is outside the window, no part of the character in the box is drawn.

3 Vector symbols (defined by a series of vectors). The character set must be the standard vector set 0, or be a vector symbol set previously loaded by GSLSS. Character angle (GSCA), box (GSCB), direction (GSCD), and shear (GSCH) affect both the string and individual characters in the string. Character angle (GSCA) rotates the baseline and rotates individual characters to lie along the baseline. Character box (GSCB) changes the spacing between characters and between lines of characters and changes the size of the characters. Character direction (GSCD) changes the direction in which the string reads. Character shear (GSCH) shears the characters at an angle. When clipping is enabled (see GSCLP), any portion of a character that is outside the window is truncated.

Coding Example

The following example shows how to use GSCM:

```
CALL GDDM('GSCM',3)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

GSCOL – Set Current Color

GSCOL(color-index)

Sets the current value of the color attribute. This attribute value applies to *all* subsequent graphics primitives until it is changed.

The color attribute value selected may not produce the indicated color if:

- On the display, a color table other than the default has been selected.
- On the plotter, the pens have been loaded in a manner not corresponding to the default definitions.

Parameters

color-index (4-byte binary integer)
The color attribute value. For the attribute value definitions, see "Color Numbers" on page C-1.

Coding Example

The following example shows how to use GSCOL:

```
CALL GDDM('GSCOL',8)
```

GSCS – Set Current Symbol Set

GSCS(symbol-set-id)

Sets the current value of the symbol set attribute. Characters in subsequent character strings are drawn from the set specified.

This is valid only in character mode 2 or 3; see "GSCM – Set Current Character Mode" on page 2-64.

Parameters

symbol-set-id (4-byte binary integer)
The symbol set identifier.

For graphics symbol text (character mode 2 or 3) the symbol-set-id designates the particular graphics symbol set (*GSS) to be used. The symbol set can be the default symbol set (symbol-set-id equals 0) or one loaded by GSLSS (can be 65 through 223).

Coding Example

The following example shows how to use GSCS:

```
! Set character mode to 3
CALL GDDM('GSCM',3)
.
.
.
! Load symbol set
CALL GDDM('GSLSS',2,'ADMUVDRP',65)
.
.
.
! Set symbol set attribute
CALL GDDM('GSCS',65)
```

GSCT – Select Color Table

GSCT(color-table-id)

Selects a color table for use on the primary device.

Other than the default color table, a color table must have been defined (using the GSCTD routine) before it can be selected.

When a color table is selected, it is associated with the current page. If a new page is subsequently selected, the color table for the new page is the default table. The page must be created before the color table is selected.

Parameters

color-table-id (4-byte binary integer)
The color table identifier.

Zero is reserved for the identifier of the default color table, which is always available. The allowable color-table-ids for definable color tables are 65 through 223.

Coding Example

The following example shows how to use GSCT:

```
CALL GDDM('GSCT',65)
```

GSCTD – Define Color Table

```
GSCTD(color-table-id,start-color,count,hue-array,lightness-array,
  saturation-array)
```

Defines a color table, including all entries contained in it.

Each color table has eight entries, corresponding to color attribute values 1 through 8. The definition of entry number 8, which represents the background color (black on displays), is always fixed and cannot be changed.

The color table does not contain an entry for color attribute value 0. If color 0 is selected (with GSCOL) when a user-defined color table is in use, the resulting color is that which is defined by color table entry number 4 (which defaults to green on displays). This is for compatibility with devices that do not support color tables.

Any entries not defined contain the default color for the corresponding color attribute value. Any definitions for entries beyond entry number 7 are ignored.

When a color table is defined, it is associated with the current primary device and it may be selected for use on a page with the GSCT routine. If the current primary device does not support color tables, the call is ignored. When the current primary device is closed (DSCLS or DSRNIT), all color table definitions for it are lost.

Parameters

color-table-id (4-byte binary integer)
The identifier by which your program will refer to this color table in later calls.

The allowable color-table-ids for definable color tables are 65 through 223.

start-color (4-byte binary integer)
The number of the entry at which the color table definitions should start.

The start-color must be in the range 1 through 7. When greater than 1, entries up to the start-color will contain default color definitions.

count (4-byte binary integer)
The number of color table entries being defined. This number should be equal to the number of entries in the hue, lightness, and saturation arrays.

hue-array (array of short floating-point numbers)
The hue values for each of the table entries being defined. Values can range from 0.00 through 1.00, with 0.00 and 1.00 giving blue, and intermediate values giving other hues.

lightness-array (array of short floating-point numbers)
The lightness values for each of the table entries being defined. Values range from 0.0 through 1.0, with 0.0 giving the least lightness (black), 1.0 indicating the most (white), and intermediate values giving varying degrees of lightness between the extremes.

saturation-array (array of short floating-point numbers)
The saturation values for each of the table entries being defined. Values range from 0.0 through 1.0, with 0.0 giving a completely unsaturated color (gray), 1.0 giving full saturation, and intermediate values giving varying degrees of partial saturation.

Figure 2-5 shows the default values for graphics work stations. For negative color index values or color index values greater than 8, refer to the table shown in the GSCOL description.

Figure 2-5. Default Color Definitions for Graphics Work Stations

Color Code	Display Default Color	H (Hue)	L (Lightness)	S (Saturation)
1	Blue	0.0	0.5	1.0
2	Red	0.33	0.5	1.0
3	Pink	0.16	0.5	1.0
4	Green	0.66	0.5	1.0
5	Turquoise	0.83	0.5	1.0
6	Yellow	0.5	0.5	1.0
7	Neutral (white)	0.0	1.0	0.0
8	Background (black)	0.0	0.0	0.0

Other color definitions that might be useful are:

Color	H (Hue)	L (Lightness)	S (Saturation)
Orange	0.4	0.5	1.0
Gray	0.0	0.5	0.0
Brown	0.35	0.2	0.75
Purple	0.16	0.2	1.0
Rose	0.22	0.38	0.7
Dark green	0.66	0.2	1.0
Light green	0.6	0.6	0.8
Dark blue	0.0	0.2	1.0
Light blue	0.9	0.7	1.0

Coding Example

The following example shows how to use GSCTD to redefine colors 1 through 3 as orange, gray, and brown:

```
OPTION BASE 1
DECIMAL HUE_ARR
DIM HUE_ARR(3)
MAT READ HUE_ARR
DATA 0.4, 0.0, 0.35
DECIMAL LT_ARR
DIM LT_ARR(3)
MAT READ LT_ARR
DATA 0.5,0.5,0.2
DECIMAL SAT_ARR
DIM SAT_ARR(3)
MAT READ SAT_ARR
DATA 1.0, 0.0, 0.75
CALL GDDM('GSCTD',65,1,3,HUE_ARR(),LT_ARR(),SAT_ARR())
.
.
.
CALL GDDM('GSCT',65)
```

Differences between the System/370 Computer and the AS/400 System

This routine is not supported on the System/370 computer.

GSELPS – Draw an Elliptic Arc

```
GSELPS(p,q,tilt-angle,xe,ye)
```

Draws a curve that starts at the current position and follows an elliptical curve until it reaches the end point. The ellipse has major and minor axis lengths given by p and q, and the major axis (p) is inclined to the x axis by the axis tilt angle supplied.

The arc is considered to be constructed as follows. An ellipse with the given major and minor axis lengths is constructed with the major axis lying along the x axis. The ellipse is then rotated by the axis tilt angle and moved so that current position and the end point both lie on the curve.

In general, there are two possible positions for the ellipse center. One is to the right and one is to the left of the line drawn from the starting point to the end point viewed in that direction. If p and q have the same sign, the center point to the left of the line is chosen, and the arc is drawn counterclockwise about it. When p and q have opposite signs, the arc is drawn clockwise about the center to the right of the line.

The arc drawn is never more than half an ellipse.

The color, line width, and line type of the arc are given by the current values of these attributes.

The current position is set to the end of the arc.

Parameters

p (short floating-point number)
The major axis length; cannot be zero.

q (short floating-point number)
The minor axis length; cannot be zero.

tilt-angle (short floating-point number)
The inclination of the major axis to the x axis in degrees (the major axis is that with length p, regardless of which axis is the longer). Positive tilt angles result in counterclockwise rotation of the ellipse, negative tilt angles result in clockwise rotation.

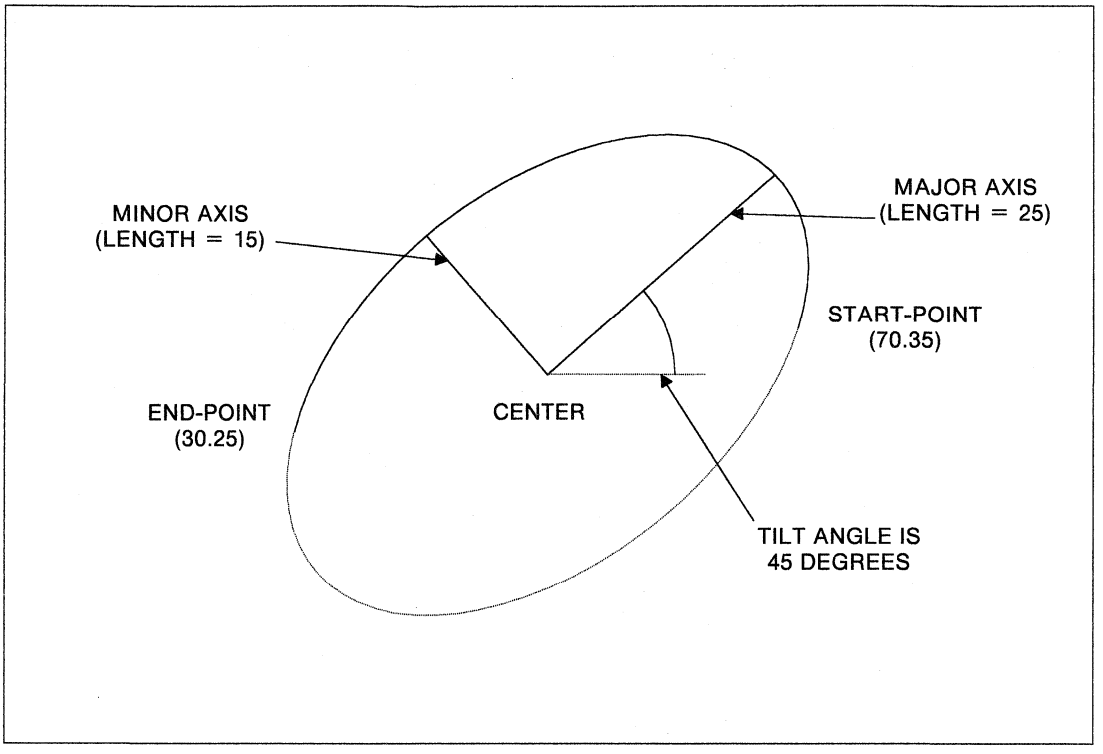
xe,ye (short floating-point numbers)
The coordinates of the end point of the arc.

Coding Example

The following example shows how to use GSELPS:

```
! The following sets the aspect ratio of the picture space to 1:1
CALL GDDM('GSPS',1.0,1.0)
!  The following draws the ellipse
CALL GDDM('GSELPS', 8.0, 4.5, 30.0, 20.0, 5.0)
```

This is the sample ellipse:



RV2F738-0

GSENDA – End a Shaded Area

GSENDA

Ends the construction of a shaded area. If necessary, a final line is drawn to close the area.

The current position is not changed, unless a final line must be drawn to close the figure. In that case, it is moved to the end point of the final line.

Parameters

None

Coding Example

The following example shows how to use GSENDA:

```
CALL GDDM('GSAREA',1)
CALL GDDM('GSLINE',12.0,40.0)
CALL GDDM('GSLINE',12.0,0.0)
CALL GDDM('GSENDA')
```

For an illustration of the area drawn, see Example 1 under "GSAREA – Start a Shaded Area" on page 2-45.

GSFLD – Define the Graphics Field

```
GSFLD(row,column,depth,width)
```

Explicitly defines the graphics field, which is the area of the current page where the picture is drawn. The graphics field is defined in row/column coordinates.

Only one graphics field is allowed in any one page.

Use GSFLD to:

- Make the picture in a page smaller.
- Create a field that can be cleared using GSCLR.
- Reserve an area on the screen for alphanumeric data (described using data description specifications).

The following shows the position of this call in the graphics hierarchy:

Page	FSPCRT
Graphics field	**GSFLD**
Picture space	GSPS
Viewport	GSVIEW
Window	GSWIN
Segment	GSSEG

If the graphics field is redefined (using GSFLD), the existing graphics contents of the page are lost, and all segments in the original field are deleted. A new graphics field is created in the newly defined area.

If no graphics field has been created when one is required to perform a requested function, one is created automatically. The default graphics field covers the entire screen or printed page.

If the primary device is an IBM plotter, the graphics field always covers the entire page regardless of the values specified for the row, column, width, and depth. However, those specified values must be valid for a screen on an graphics work station. (That is, row plus depth cannot exceed 24, and column plus width cannot exceed 80.)

If the device token is L79A3, the row plus depth cannot exceed 32, and column plus width cannot exceed 80.

For printers, the graphics field cannot exceed the row/column (overflow for rows, forms width for columns) specification for the printer file in use.

Parameters

row,column (4-byte binary integers)
The position on the page of the top left corner of the graphics field. If a row or column of zero is specified, the graphics field is deleted.

depth,width (4-byte binary integers)
The size of the field, in rows and columns. If either the depth or width is zero, the graphics field is deleted.

Coding Example

The following example shows how to use GSFLD:

```
CALL GDDM('GSFLD',1,1,24,40)
```

GSFLW – Fractional Line Width

GSFLW(linewidth-multiplier)

Sets the current line width as a floating-point value.

The internal representation of the line width can thus be set to fractional values. GDDM draws the widest line that does not exceed the requested line width. If there is no line width thin enough, the thinnest available line is drawn.

The line width attribute does not affect mode-3 characters that are drawn using GSCHAP or GSCHAR. These characters are always drawn using the default line width.

Parameters

linewidth-multiplier (short floating-point number)
The line width. Can be from 0 (thinnest) through 100 (thickest).

Line width 1.0 is the standard line width for the device. Line widths greater than one give proportionately wider lines. Line widths less than one give narrower lines. A line width of zero produces the thinnest line.

The standard line width in pixels for the current device (see following table) is multiplied by the line-width-multiplier, and the result rounded down to an integer value. This value defines, in pixels, the width of the lines subsequently drawn. If the result is zero, a width of one pixel is used. If the result is more than the maximum for the current device, the maximum is used.

Parameter values are interpreted on devices attached to the AS/400 system like this:

Device	Minimum (Value is 0)	Standard (Value is 1)	Maximum (Value is 100)
5292 Model 2	1	1	2
IBM PCs	1	1	1
IBM plotters	1	1	1
Printers	1	1	2

Coding Example

The following example shows how to use GSFLW:

```
CALL GDDM('GSFLW',1.5)
```

GSGET – Retrieve Graphics Data

```
GSGET(buffer-length,buffer,GDF-length)
```

Retrieves graphics data from the current page. When this routine is called, GDDM converts graphics data from the page into GDF format (see Appendix B, "GDF Order Descriptions") and returns it in the buffer parameter.

This call is valid only if a dummy device is being used (for more information on dummy devices, see Figure 2-2 on page 2-10).

The GDF data that results does not necessarily resemble the original commands used to generate the picture, as these have been processed to suit the primary device in use. For example, coordinates are converted to an internal coordinate system with some loss of precision. Complex primitives (such as curved fillets) might be simplified and approximated. Clipping might have caused changes to the data. The data is thus not the same as the original. However, you can use GDF data to store a description of a picture in a form such that you need not make individual calls to OS/400 Graphics functions again.

Color tables cannot be saved as graphics data.

The first order in the generated GDF data is a GDF comment order (see "Comment Order" on page B-17), which contains the following information to allow you to use the GDF data later (see GSPUT):

1. One 2-byte binary integer indicating the coordinate format used in the GDF data (see GSPUT). This always takes a value of 2, indicating that the coordinates are given as 2-byte binary integers.
2. One 2-byte binary integer indicating the data type of following fields. If the GDF data was generated on the AS/400 system, this field takes a value of 2, and the following fields are 2-byte binary integers. If the GDF data was generated other than on the AS/400 system, this field can be one of the following:

   1 Following fields are 1-byte binary integers (not supported on the AS/400 system).

   2 Following fields are 2-byte binary integers (the default on the AS/400 system).

   3 Following fields are 4-byte binary integers (not supported on the AS/400 system).

   4 Following fields are real (System/370 floating point; not supported on the AS/400 system).
3. Four 2-byte binary integer coordinates giving, in order, the lower and upper x bounds and the lower and upper y bounds of the coordinates. (Coordinates are only within these limits if clipping was enabled throughout the picture generation.)

Retrieval of graphic data must start with a call to GSGETS (described later in this chapter). This indicates the data that is needed. One or more calls to GSGET can then be used to fetch the data.

If the buffer is long enough to contain the GDF data requested, the data is returned, and the last parameter is set to indicate its length. If the buffer is not long enough, the buffer is filled, and more data can be obtained by another call to GSGET. All data has been extracted when a length of zero is returned (without error).

Each call to GSGET retrieves a number of complete GDF orders. Part of an order is never returned. (This means that buffers obtained from GSGET can be returned as input to GDDM through GSPUT.) If there is not enough room in the buffer to take a complete order, no data is returned, and an error occurs.

Graphics retrieval must be ended by GSGETE (see "GSGETE—End Retrieval of Graphics Data" on page 2-81).

Parameters

buffer-length (4-byte binary integer)
The length of the buffer. This should exceed 260 bytes to guarantee that errors do not occur.

Note: BASIC does not support character strings longer than 255 bytes. This could cause problems for graphics programs for which a large number of points are specified for GSPFLT or GSPLNE or for which long character strings are specified for GSCHAP or GSCHAR.

buffer (returned by GDDM) (character)
A program variable, of stated length, to receive the GDF data.

GDF-length (returned by GDDM) (4-byte binary integer)
A 4-byte binary integer variable that GDDM normally sets to the length of GDF data placed in the buffer. If it is zero (and no error is reported), all the GDF data requested has already been returned.

Coding Example

The following example shows how to use GSGET with its companion calls GSGETS and GSGETE:

```
OPTION BASE 1
! Create segment to get retained data
CALL GDDM('GSSEG',1)
.
. (draw graphics)
.
CALL GDDM('GSSCLS')
! Use GSGETS to start GDF retrieval
INTEGER IDARRAY
DIM IDARRAY(1)
LET IDARRAY(1) = 0
CALL GDDM('GSGETS',1,IDARRAY())
! Set up variables for GSGET
INTEGER GDFLEN
DIM BUF$*255          ! NOTE: Can't be >255 in BASIC
! Set up loop to read all graphics data in current segment
LOOP:
  GSGET(255,BUF$,GDFLEN)
    .
    . (processing to save contents of BUF$
    .  elsewhere in program or in a file)
    .
IF GDFLEN > 0 THEN GOTO LOOP ELSE GOTO ENDGET
! End retrieval of graphics data
ENDGET: CALL GDDM('GSGETE')
```

Note: Some GDF orders can require more than 255 bytes for the graphics-data parameter (in this example, BUF$). These include orders for graphics text (GSCHAP and GSCHAR with very long character strings), polyfillets, and polylines. If your graphics data includes such orders, you should use another language to write the program using GSPUT.

GSGETE – End Retrieval of Graphics Data

GSGETE

Ends the retrieval of graphics data (see "GSGET – Retrieve Graphics Data" on page 2-78 and "GSGETS – Start Retrieval of Graphics Data" on page 2-82).

This call is valid only when a dummy device is being used. For more information on dummy devices, see Figure 2-2 on page 2-10.

GSGETE can be called whether or not all the graphics data has been retrieved.

Parameters

None

Coding Example

The following example shows how to use GSGETE with its companion calls GSGET and GSGETS:

```
OPTION BASE 1
! Create segment to get retained data
CALL GDDM('GSSEG',1)
.
. (draw graphics)
.
CALL GDDM('GSSCLS')
! Use GSGETS to start GDF retrieval
INTEGER IDARRAY
DIM IDARRAY(1)
LET IDARRAY(1) = 0
CALL GDDM('GSGETS',1,IDARRAY())
! Set up variables for GSGET
INTEGER GDFLEN
DIM BUF$*255          ! NOTE: Can't be >255 in BASIC
! Set up loop to read all graphics data in current segment
LOOP:
  GSGET(255,BUF$,GDFLEN)
    .
    . (processing to save contents of BUF$
    .  elsewhere in program or in a file)
    .
IF GDFLEN > 0 THEN GOTO LOOP ELSE GOTO ENDGET
! End retrieval of graphics data
ENDGET: CALL GDDM('GSGETE')
```

Note: Some GDF orders can require more than 255 bytes for the graphics-data parameter (in this example, BUF$). These include orders for graphics text (GSCHAP and GSCHAR with very long character strings), polyfillets, and polylines. If your graphics data includes such orders, you should use another language to write the program using GSPUT.

GSGETS – Start Retrieval of Graphics Data

```
GSGETS(count,array)
```

Starts the retrieval of graphics data from the current page.

This call is valid only if a dummy device is being used. For more information on dummy devices, see Figure 2-2 on page 2-10.

You can retrieve the graphics data for an individual named segment or the data for all segments in the page. You cannot retrieve temporary data (graphics primitives that are not in a segment); therefore all parts of a picture that are to be retrieved must be produced by primitives that occur between calls to GSSEG and GSSCLS.

GSGETS specifies the data required and begins the retrieval. Data is obtained by one or more calls to GSGET (described earlier in this chapter), and the retrieval is ended by GSGETE.

Retrieval of graphics data cannot be started when there is an open segment.

Parameters

count (4-byte binary integer)
The number of elements in the following array. This must be zero or one.

array (array of 4-byte binary integers)
The array can have at most one element. This is the identifier of the segment required. If the identifier is zero or the array has no elements, all segments are retrieved.

Coding Example

The following example shows how to use GSGETS with its companion calls GSGET and GSGETE:

```
OPTION BASE 1
! Create segment to get retained data
CALL GDDM('GSSEG',1)
.
. (draw graphics)
.
CALL GDDM('GSSCLS')
! Use GSGETS to start GDF retrieval
INTEGER IDARRAY
DIM IDARRAY(1)
LET IDARRAY(1) = 0
CALL GDDM('GSGETS',1,IDARRAY())
! Set up variables for GSGET
INTEGER GDFLEN
DIM BUF$*255          ! NOTE: Can't be >255 in BASIC
! Set up loop to read all graphics data in current segment
LOOP:
  GSGET(255,BUF$,GDFLEN)
    .
    . (processing to save contents of BUF$
    .  elsewhere in program or in a file)
    .
IF GDFLEN > 0 THEN GOTO LOOP ELSE GOTO ENDGET
```

```
! End retrieval of graphics data
ENDGET: CALL GDDM('GSGETE')
```

Note: Some GDF orders can require more than 255 bytes for the graphics-data parameter (in this example, BUF$). These include orders for graphics text (GSCHAP and GSCHAR with very long character strings), polyfillets, and polylines. If your graphics data includes such orders, you should use another language to write the program using GSPUT.

GSIMG – Draw a Graphics Image

```
GSIMG(type,width,depth,length,graphics-image-data)
```

Draws a graphics image at the current position. All images are drawn within a rectangle on the device and consist of an array of PELs.

The width and depth parameters determine the dimensions of the rectangle in PELs. The data-graphics image parameter determines which of the PELs are visible. A bit set to one in the data-graphics image parameter sets the associated PEL on. A bit set to zero leaves the associated PEL unchanged.

The top left corner of the graphics image is placed at the current position, and the data supplied is drawn row by row starting at the top. Each row is drawn from left to right. The number of PELs in each row must be a multiple of 8. You can pad the row with zeros when the graphics image width specified is not a multiple of 8. If, for example, the graphics image width specified is 12, each row of data must be padded out to a length of 16 so that the data occupies 2 bytes exactly. (See Example 2 later in this function description.) If this is not done, the graphics image is distorted.

The length parameter must take into account the padding of each row of data. The length must be given in bytes (1 byte for each 8 bits in the graphics image); an error occurs if the length is incorrect.

The color of the graphics image is determined by the current value of the color attribute. The current position remains unchanged after the graphics image has been drawn.

Parameters

type (4-byte binary integer)
The type of the graphics image to be drawn. Must be set to 0.

width (4-byte binary integer)
The width of the graphics image in PELs (not including padding with zeros, if needed to fill out full bytes). The width parameter must be smaller than 2040 PELs.

depth (4-byte binary integer)
The depth of the graphics image in PELs.

length (4-byte binary integer)
The length in bytes of the graphics image data. The length must be the product of the width and the depth, divided by eight, or

(width rounded up to a multiple of 8) x (depth) / 8 = length.

graphics-image-data (character string)
The graphics image data to be displayed. The PELs must be given row by row, starting at the top and running from left to right within each row.

In BASIC, you cannot directly specify how to set bits on and off but you can use the HEX$ built-in function. In RPG III, you could use the BITON and BITOFF operation codes; in PL/I, you could use BIT data; in Pascal, you could use hexadecimal string constants; in the COBOL/400 language, there is no direct way of manipulating bit strings. You might want to call a program in another

language, passing a character string for the other program to initialize and return with the correct bit values.

For further examples using GSIMG in other languages, see the *GDDM Programming Guide*.

In BASIC, in the HEX$ built-in function, each number in the apostrophes is equivalent to a 4-bit string, as follows:

Hex digit	Bit string
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

For example, the first number (8) is equivalent to 1000. The second number (1) is equivalent to 0001. Together they fill out 1 byte (8 bits) for the top row of the graphics image.

Coding Examples

Example 1:

The following example shows how to use GSIMG:

```
OPTION BASE 1
! Move current position to center of screen
CALL GDDM('GSMOVE',50.0,50.0)
! Set up graphics image
DIM CROSS$*7
LET CROSS$ = HEX$('81422418244281')
CALL GDDM('GSIMG',0,8,7,7,CROSS$)
```

This example gives the following output:

Hex values	Bit values	Sample output
81	10000001	
42	01000010	
24	00100100	
18	00011000	X
24	00100100	
42	01000010	
81	10000001	

RV2F739-0

Example 2:

The following example shows how to use GSIMG when the graphics image must be padded on the right with zeros:

```
OPTION BASE 1
! Move current position to center of screen
CALL GDDM('GSMOVE',50.0,50.0)
! Set up graphics image
DIM LGCROSS$*22
LET LGCROSS$ = HEX$('80104020204010800900060009001080204040208010')
CALL GDDM('GSIMG',0,12,11,22,LGCROSS$)
```

This example gives the following output:

Hex values	Bit values	Sample output
8010	1000 0000 0001 0000	
4020	0100 0000 0010 0000	
2040	0010 0000 0100 0000	
1080	0001 0000 1000 0000	
0900	0000 1001 0000 0000	
0600	0000 0110 0000 0000	X
0900	0000 1001 0000 0000	
1080	0001 0000 1000 0000	
2040	0010 0000 0100 0000	
4020	0100 0000 0010 0000	
8010	1000 0000 0001 0000	

RV2F740-0

GSIMGS – Draw Scaled Graphics Image

```
GSIMGS(type,width,depth,length,graphics-image-data,x-size,y-size)
```

Draws a graphics image in the same way as the GSIMG call but scales the size of the graphics image as well.

The first five parameters are interpreted as the parameters of the GSIMG call. The x- and y-size parameters are floating-point numbers determining the size of the graphics image window, the image being scaled independently in the x- and y-directions to fit within the graphics window.

Scaling up is performed only by factors of 1, 2, 3, and so forth (integers only). Each bit in the graphics image is mapped onto a small rectangular area on the screen. If the window is smaller than the provided graphics image in either dimension, a scale factor of one is used in that dimension.

Parameters

type (4-byte binary integer)
The type of the graphics image to be drawn. Must be set to 0.

width (4-byte binary integer)
The width of the graphics image in pixels. It must be smaller than 2040.

depth (4-byte binary integer)
The depth of the graphics image in pixels.

length (4-byte binary integer)
The length in bytes of the graphics image data.

graphics-image-data (character string)
The graphics image data to be displayed. The pixels must be given row by row, starting at the top and running from left to right within each row.

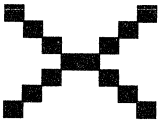
x-size (short floating-point number)
The size of the graphics image window in the x direction.

y-size (short floating-point number)
The size of the graphics image window in the y direction.

Coding Example

The following example shows how to use GSIMGS:

```
OPTION BASE 1
! Move current position to center of screen
CALL GDDM('GSMOVE',50.0,50.0)
! Set up graphics image
DIM CROSS$*8
LET CROSS$ = HEX$('81422418244281')
CALL GDDM('GSIMG',0,8,7,7,CROSS$,10.0,10.0)
```



RV2F741-0

GSLINE – Draw a Straight Line

GSLINE(x,y)

Draws a straight line from the current position to the specified end point.

The color, line width, and line type of the line are given by the current values of these attributes. The current position is set to the end point of the line.

When the specified end point lies outside the window boundaries, the results (including current position) are undefined. Otherwise, if clipping is enabled, only the section of the line within the current window is visible.

Parameters

x,y (short floating-point numbers)
The end point of the line in world coordinates.

Coding Example

The following example shows how to use GSLINE:

```
CALL GDDM('GSLINE',3.5,6.0)
```

GSLSS – Load a Graphics Symbol Set from Auxiliary Storage

```
GSLSS(type,symbol-set-name,symbol-set-id)
```

Loads a set of symbol definitions from auxiliary storage. The symbol set is a graphics symbol set (*GSS) from a *GSS object. The library containing the graphics symbol set (usually QGDDM) must be in the library list of the user who calls the graphics application program. The symbol set is loaded for use by the current device, and must be loaded once for each device on which it is to be used.

For more information on graphics symbol sets, refer to the *GDDM Programming Guide*.

The definitions are retained by GDDM for graphics use.

Parameters

type (4-byte binary integer)

The type and usage of this symbol set follow:

1 Graphics image symbol set (used for character mode 2; see GSCM earlier in this chapter)
2 Vector symbol set (used for character mode 3; see GSCM earlier in this chapter)
3,5 Not supported on the AS/400 system
4 Marker symbol set (ISS or VSS)

symbol-set-name (8-byte character string)

The name (left-justified) of the symbol set (*GSS) to be read from auxiliary storage.

If a symbol set with the same name is loaded more than once, without first being released (see GSRSS), it is undefined whether GDDM uses the copy that it already has, or takes a new copy.

Note: Although the AS/400 system allows object names to be 10 bytes long, *this routine only accepts a maximum name length of 8 bytes.*

The following are the names of the symbol sets supplied with OS/400 Graphics:

- For graphics image symbol sets (character-mode 2 only):
  - ADMMISSG (default for GDDM on the IPDS printers)
  - ADMMISSI (default for GDDM on the 5224/5225 printers and on the 4234 Model 3 Printer @ 10 cpi)
  - ADMMISSP (default for GDDM on the 4234 Model 3 Printer @ 15 cpi)

Note: For IPDS printers, the character set specified on the print file will be used for graphics image symbols. For information on specifying a print file, see Figure 2-2 on page 2-10. For more information, see the CHRID parameters on the CRTPRTF command in the *CL Reference* manual.

- For vector symbol sets (character-mode 3 only):
  - ADMMVSS (default for graphics work stations and IBM plotters)
  - ADMDVSS (American English Standard characters)
  - ADMDVSSE (U.K. English Standard characters)
  - ADMDVSSF (French Standard characters)
  - ADMDVSSG (German Standard characters)
  - ADMDVSSI (Italian Standard characters)
  - ADMDVSSK (Katakana Standard characters)
  - ADMDVSSS (Spanish Standard characters)
  - ADMUVCIP (Complex Italic Principal)
  - ADMUWCIP (Complex Italic Principal; proportionally spaced)
  - ADMUVCRP (Complex Roman Principal)
  - ADMUWCRP (Complex Roman Principal; proportionally spaced)
  - ADMUVCSP (Complex Script Principal)
  - ADMUWCSP (Complex Script Principal; proportionally spaced)
  - ADMUVDRP (Duplex Roman Principal)
  - ADMUWDRP (Duplex Roman Principal; proportionally spaced)
  - ADMUVGEP (Gothic English Principal)
  - ADMUWGEP (Gothic English Principal; proportionally spaced)
  - ADMUVGGP (Gothic German Principal)
  - ADMUWGGP (Gothic German Principal; proportionally spaced)
  - ADMUVGIP (Gothic Italian Principal)
  - ADMUWGIP (Gothic Italian Principal; proportionally spaced)
  - ADMUVSRP (Simplex Roman Principal)
  - ADMUWSRP (Simplex Roman Principal; proportionally spaced)
  - ADMUVTIP (Triplex Italic Principal)
  - ADMUWTIP (Triplex Italic Principal; proportionally spaced)
  - ADMUVTRP (Triplex Roman Principal)
  - ADMUWTRP (Triplex Roman Principal; proportionally spaced)
  - ADMVMSS (Multinational Standard Simple)
  - ADMWMSS (Multinational Standard Simple; proportionally spaced)
  - ADMVMSB (Multinational Standard Bold)
  - ADMWMSB (Multinational Standard Bold; proportionally spaced)
  - ADMVMOB (Multinational Open Block)
  - ADMWMOB (Multinational Open Block; proportionally spaced)
  - ADMVMFB (Multinational Filled Block)
  - ADMWMFB (Multinational Filled Block; proportionally spaced)
  - ADMVMRP (Multinational Roman Principal)
  - ADMWMRP (Multinational Roman Principal; proportionally spaced)
- For marker symbol sets:
  - Any graphics image symbol set or vector symbol set may be used as a marker symbol set.

symbol-set-id (4-byte binary integer)
The identifier by which your program refers to this symbol set in later statements.

The allowable symbol-set-ids are:

0	marker symbol sets
65 through 223	other symbol sets

If a symbol-set-id is the same as one issued previously for the same type, the new definitions replace the previous ones.

Coding Example

In the following example, two symbol sets are loaded. Then the program queries for as many as three symbol sets. The values returned by GDDM are shown after the example:

```
OPTION BASE 1
CALL GDDM('GSLSS',2,'ADMUVDRP',65)
CALL GDDM('GSLSS',2,'ADMUVSRP',66)
  .
  . (graphics processing)
  .
INTEGER TYP,SSIDS
DIM TYP(3),SSNAMES$(3)*8,SSIDS(3)
CALL GDDM('GSQSS',3,TYP(),SSNAMES$(),SSIDS())
```

The values of elements in the arrays are as follows:

- TYP(1) is 65; TYP(2) is 66; TYP(3) is 0.
- SSNAME$(1) is 'ADMUVDRP';
  SSNAME$(2) is 'ADMUVSRP';
  SSNAME$(3) is ' '.
- SSIDS(1) is 65; SSIDS(2) is 66; SSIDS(3) is 0.

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, symbol set types 3 (shading pattern set) and 5 (4250 composed page printer font) are not supported.

GSLT – Set Current Line Type

```
GSLT(line-type)
```

Sets the current value of the line type attribute. Subsequent primitives using lines (that is, those constructed by GSLINE, GSARC, GSPLNE, GSELPS, GSPFLT, or GSVECM) have the specified line type until this is reset.

The line-type attribute does not affect mode-3 characters that are drawn using GSCHAP or GSCHAR. These characters are always drawn using the default line type.

Parameters

line-type (4-byte binary integer)
For valid values of line-type, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Example

The following example shows how to code GSLT for dotted lines:

```
CALL  GDDM('GSLT',1)
```

GSLW – Set Current Line Width

GSLW(line-width)

Sets the current value of the line width attribute. Subsequent primitives that use lines have the specified width.

The line-width attribute does not affect mode-3 characters that are drawn using GSCHAP or GSCHAR. These characters are always drawn using the default line width.

Parameters

line-width (4-byte binary integer)
It must be one of the following values:

Device	Minimum (value is 0)	Standard (value is 1)	Maximum (value is 100)
5292 Model 2	1	1	2
IBM PCs	1	1	1
IBM plotters	1	1	1
Printers	1	1	2

Note: The line width on a plotter is controlled by the width of the selected pen rather than by GSLW. To set the width of pens used in the plotter, see processing option group 12 under DSOPEN earlier in this chapter.

Coding Example

The following example shows how to code GSLW for double-width lines:

```
CALL GDDM('GSLW',2)
```

GSMARK – Draw a Marker Symbol

```
GSMARK(x,y)
```

Draws a single marker at a specified position.

A marker is a symbol used to indicate a position. It is like a character, but is positioned by its center.

The marker color is set by GSCOL, and the marker is set by GSMS.

The current position is set to (x,y).

Parameters

x,y (short floating-point numbers)
The position of the marker in world coordinates.

Coding Example

The following example shows how to code GSMARK:

```
CALL GDDM('GSMARK',50.0,50.0)
```

GSMIX – Set Current Color-Mixing Mode

GSMIX(color-mixing-mode)

Controls how the color of a primitive is combined with any underlying color when lines or shaded areas cross.

Not all mix modes are supported on all devices. A device-dependent alternative mode may be used instead. Figure 2-6 on page 2-97 shows how the different mix modes are interpreted on each of the graphics devices. Regardless of how the mode is interpreted by a particular device, any generated GDF will contain the correct mix mode order.

Parameters

color-mixing-mode (4-byte binary integer)
One of the following:

0 Default, same as 2.

1 Mix mode (OR mode).

Display points common to two primitives (for example, a line crossing a color-shaded area) take the color resulting from the mixture of the two colors. Mixing is accomplished by ORing the color index values together. The binary value of the current color index is logically ORed with the binary value of the color index for the existing pixel to produce the new color index value. For mixing results, see Figure 2-7 on page 2-97.

On a plotter, a modified mix mode is always used. In this mode, the color index values are not ORed together, but the resulting colors may appear to have been mixed together on the paper.

2 Overpaint mode.

Display points common to two primitives take the color of the current primitive. Any underlying color is not visible. On a plotter, overpaint is not supported. The device always operates in a modified mix mode.

3 Underpaint mode.

Display points common to two primitives retain the color specified for the first one drawn, unless that color was the background color (color 8). For example, if an area is shaded blue and then a yellow line is drawn across it, the line is visible only where it lies outside the blue area. However, if the shaded area is specified as color 8, the line is yellow throughout its length.

Underpaint mode is not supported on the plotters or printers.

For device token L79A3, mix mode 3 produces an underpaint GDF order.

4 Exclusive-OR mode.

Display points common to two primitives take the color resulting from the logical exclusive OR of the color index values. The binary value of the current color index is exclusively ORed with the binary value of the color index for the existing pixel to produce the new color index value.

This option allows part of a picture to be deleted without having to delete the whole picture and redrawing the part that is not to be deleted.

On a plotter, exclusive-OR mode is not supported. The device always operates in a modified mix mode. On the printers, exclusive-OR mode is not supported.

Figure 2-6. Color-Mixing Results on Different Devices

Mix Mode Number	GDDM Mix Mode Definition	Display Mix Mode	Plotter Mix Mode	SCS Printers	IPDS Printers
0	Default, same as mode 2	Overpaint	Mix (modified)	Overpaint	Overpaint
1	Mix (OR)	Mix (OR)	Mix (modified)	Mix (OR)	Overpaint
2	Overpaint	Overpaint	Mix (modified)	Overpaint	Overpaint
3	Underpaint	Overpaint	Mix (modified)	Overpaint	Overpaint
4	Exclusive OR	Exclusive OR	Mix (modified)	Overpaint	Overpaint

The color that results from mixing any other two colors together in mix mode 1 is given in the following table and is independent of the order in which the colors are mixed.

Figure 2-7. Results of Mixing Colors

		Original Color						
		1	**2**	**3**	**4**	**5**	**6**	**7**
	1	1	3	3	5	5	7	7
	2	3	2	3	6	7	6	7
Mixed	**3**	3	3	3	7	7	7	7
With	**4**	5	6	7	4	5	6	7
	5	5	7	7	5	5	7	7
	6	7	6	7	6	7	6	7
	7	7	7	7	7	7	7	7

For example, if color 2 is mixed with color 4, the result is color 6; color 3 mixed with color 5 produces color 7.

Coding Example

The following example shows how to code GSMIX:

```
CALL GDDM('GSMIX',2)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, mix mode 3 (underpaint) is not supported.

On the AS/400 system, mix mode 4 (exclusive OR) is supported.

GSMOVE – Move Without Drawing

```
GSMOVE(x,y)
```

Moves the current position to the specified point.

If the specified position lies outside the window, the current position is undefined unless clipping has been enabled (see GSCLP).

Parameters

x,y (short floating-point numbers)
A point in world coordinates to which the current position is to be moved. The new value of the current position is (x,y).

Coding Example

The following example shows how to code GSMOVE:

```
CALL GDDM('GSMOVE',30.0,44.0)
```

GSMRKS – Draw a Series of Marker Symbols

```
GSMRKS(count,xarray,yarray)
```

Draws a series of markers at specified points. Marker color is set by GSCOL. The current position is set to the position of the last marker in the series. The marker symbol is set by GSMS.

Parameters

count (4-byte binary integer)
The number of markers to be drawn. This is also the number of elements in xarray and yarray.

xarray (array of short floating-point numbers)
The x coordinates of the marker positions (in world coordinates).

yarray (array of short floating-point numbers)
The y coordinates of the marker positions (in world coordinates).

Coding Example

The following example shows how to code GSMRKS to draw four markers:

```
OPTION BASE 1
DECIMAL XARRAY,YARRAY
DIM XARRAY(4),YARRAY(4)
MAT READ XARRAY
DATA 0,2,4,6
MAT READ YARRAY
DATA 5,8,11,13
CALL GDDM('GSMRKS',4,XARRAY(),YARRAY())
```

GSMS – Set Current Marker Symbol

GSMS(marker-symbol)

Selects the current marker symbol to be used by calls to GSMARK and GSMRKS.

Parameters

marker-symbol (4-byte binary integer)
The marker symbol. The default is 0 (a cross; same as marker symbol 1).

For valid values of markers, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Example

The following example shows how to code GSMS to set subsequent markers to shaded squares:

```
CALL GDDM('GSMS',8)
```

GSMSC – Set Marker Scale

GSMSC(scale)

Sets the size for subsequent marker symbols. The scale parameter determines the scale of the marker symbol with respect to the default marker box. If the marker scale is not specified before to drawing the first vector marker, a scale of 1 is assumed. The default marker box is a square whose sides are equal to the width of the default character box.

Parameters

scale (short floating-point number)
The scaling of the marker box with respect to the default marker box.

The scale must be greater than 0.

Coding Example

The following example shows how to code GSMSC to get markers twice as large as the default:

```
CALL GDDM('GSMSC',2.0)
```

GSPAT – Set Current Shading Pattern

GSPAT(pattern-number)

Sets the current value of the pattern attribute. Subsequent shading (area fill) uses the specified pattern until this is reset.

Parameters

pattern-number (4-byte binary integer)
The number of the shading pattern to be used.

Valid values are:

0	Default (solid)
1 through 16	IBM-supplied patterns as shown in Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Example

The following example shows how to code GSPAT:

```
CALL GDDM('GSPAT',3)
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, shading patterns 65 through 239 (user-defined patterns on the System/370 computer) are not supported.

GSPFLT – Draw a Curved Fillet

```
GSPFLT(count,xarray,yarray)
```

Draws a curve starting at current position and defined by the xarray and yarray parameters.

As shown in the coding examples later in this function description, if two points are supplied, an imaginary line is drawn from the current position to the first point, and a second line is drawn from the first point to the second. A curve is then constructed, starting at the current position and in the direction of the first line. The curve is drawn such that it reaches the last point at a tangent to the second line, having followed a path somewhat like a curve of pursuit.

The curve has the appearance of a fillet.

If you specify more than two points, an imaginary series of lines is constructed through them (like GSPLNE). All the lines except the first and last are then divided in two at their mid-points. A series of curved fillets are then drawn each starting at the end point of the last.

The curves have color, line width, and line type given by the current values of these attributes. The current position is set to the last point.

Parameters

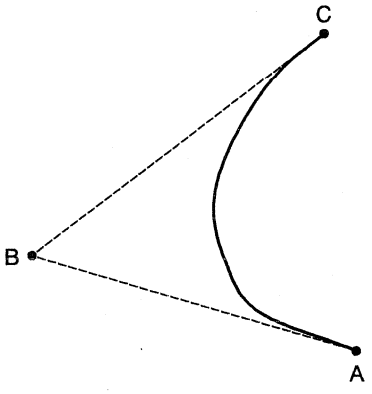
count (4-byte binary integer)
The number of points defined by xarray and yarray.

xarray,yarray (array of short floating-point numbers)
Two arrays whose elements specify the points defining the curve.

Coding Examples

Example 1: This example uses GSPFLT to draw the line shown following the example:

```
OPTION BASE 1
DECIMAL XARRAY,YARRAY
DIM XARRAY(2),YARRAY(2)
MAT READ XARRAY
DATA 0,20
MAT READ YARRAY
DATA 20,40
  .
  . (graphics processing)
  .
! Set current position
CALL GDDM('GSMOVE',22.0,10.0)
CALL GDDM('GSPFLT',2,XARRAY(),YARRAY())
```



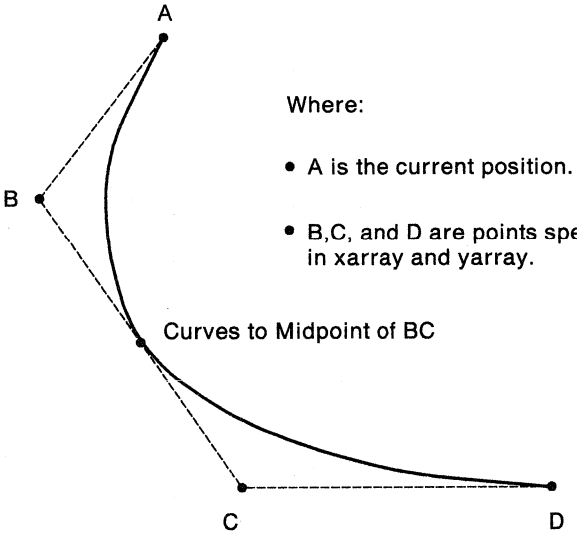
Where:

- A is current position.
- B and C are points specified in xarray and yarray.

RV2F742-0

Example 2:

```
OPTION BASE 1
DECIMAL XARRAY,YARRAY
DIM XARRAY(3),YARRAY(3)
MAT READ XARRAY
DATA 0,10,20
MAT READ YARRAY
DATA 10,0,0
  .
  . (graphics processing)
  .
! Set current position
CALL GDDM('GSMOVE',7.0,28.0)
CALL GDDM('GSPFLT',3,XARRAY(),YARRAY())
```



Where:

- A is the current position.
- B,C, and D are points specified in xarray and yarray.

RV2F743-1

GSPLNE – Draw a Series of Lines

```
GSPLNE(count,xarray,yarray)
```

Draws a series of straight lines (sometimes called a polyline) starting at the current position and passing through the points defined by the xarray and yarray parameters.

The color, line width, and line type of the lines are given by the current values of these attributes. The current position is set to the end point of the last line in the series.

Parameters

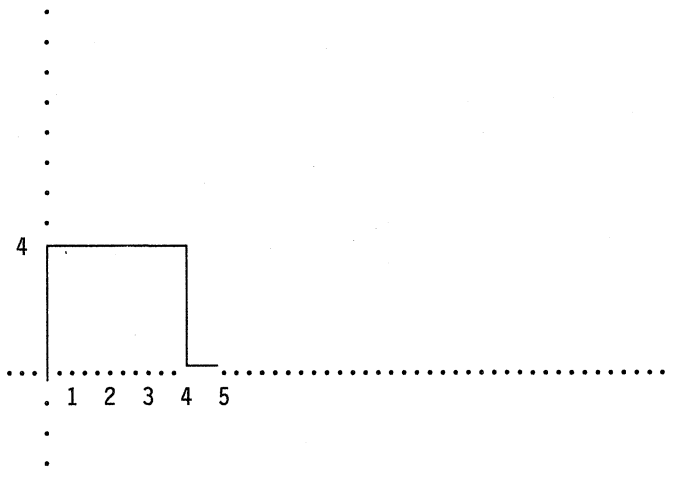
count (4-byte binary integer)
The number of points to be passed through.

xarray,yarray (arrays of short floating-point numbers)
The end points of the lines. After drawing each line in the series, the current position is set to the end point of that line. The nth line is constructed by drawing a line from the current position to the point whose x coordinate is the nth element of xarray and whose y coordinate is the nth element of yarray.

Coding Example

In this example, the program uses GSPLNE to draw a series of lines:

```
OPTION BASE 1
DECIMAL XARRAY,YARRAY
DIM XARRAY(5),YARRAY(5)
MAT READ XARRAY
DATA 0,0,4,4,5
MAT READ YARRAY
DATA 0,4,4,0,0
CALL GDDM('GSPLNE',5,XARRAY(),YARRAY())
```



GSPS – Define the Picture Space

```
GSPS(width,height)
```

Explicitly defines the picture space to be used in the current graphics field. The picture space defines the aspect ratio (the ratio of width to height) of the picture unless viewports are also defined with a call to GSVIEW.

Use GSPS to set up a desirable aspect ratio (the ratio of width to height). You should specify a one-to-one aspect ratio for pictures in which the vertical and horizontal scales should be the same, such as circles, and for any drawing which should be done to scale, such as a floor plan.

The following shows the position of this call in the graphics hierarchy:

Page	FSPCRT
Graphics	GSFLD
Picture space	**GSPS**
Viewport	GSVIEW
Window	GSWIN
Segment	GSSEG

The width and height specified are numbers between zero and one. These numbers define the aspect ratio of the picture to be drawn.

The center of the picture space is mapped onto the center of the graphics field. The projection ratios are then adjusted so that the entire picture space is drawn in the graphics field, as large as possible while maintaining the aspect ratio specified. Either the left and right edges, or the top and bottom edges, of the picture space and graphics field coincide. All edges coincide only if the aspect ratios of picture space and graphics field are identical. If no graphics field is defined (using GSFLD) or defaulted before GSPS is issued, a default graphics field is set.

A default picture space is used if no GSPS call is issued before:

- The first viewport is defined or queried (GSVIEW or GSQVIE).
- The first segment is opened (GSSEG).
- The picture space is queried (GSQPS).
- The first primitive or attribute is drawn or set (such as GSLINE or GSCOL).
- The cursor is queried in graphics coordinates (GSQCUR).

The default equals the actual aspect ratio (in physical units such as millimeters) of the graphics field. Note that this usually alters the aspect ratio from that drawn on the window, if viewports are not explicitly specified. The *default* picture space, therefore, covers the graphics field.

When GSVIEW, GSPS, and GSWIN have not been called, the default values are:

- On a graphics work station, the default width is 1 and the default height is 0.529663. For an IBM PC, the default height is device dependent.
- On plotters, the default width is 1 and the default height is 0.72.
- On printers, the default width for the QPGDDM printer file (as shipped) is 1 and the default height is 0.757.

At other times, the aspect ratio of the picture space can be obtained by a call to GSQPS.

Once specified or taken by default, the picture space cannot be changed for that graphics field.

Parameters

width,height (short floating-point numbers)
The width and height of the picture space. One of the numbers, width or height, must be 1; the other must be greater than 0 but not greater than 1.

Coding Example

In the following example, the program uses GSPS to define the picture space for a one-to-one aspect ratio. This causes vertical and horizontal distances specified in world coordinates to have the same scale. This is particularly important when drawing a picture to scale, such as a floor plan.

```
CALL GDDM('GSPS',1.0,1.0)
```

GSPUT – Draw Data from a Graphics Data Format File

```
GSPUT(control,length,graphic-data)
```

Draws the graphics data from a graphics data format (GDF) file.

Any series of graphic primitives (such as lines, arcs, areas, and character strings) together with their attributes can be coded as a string of data bytes called a Graphics Data Format (GDF) file. This consists of a series of *orders* (see Appendix B, "GDF Order Descriptions"). Each order defines a graphics primitive or an attribute setting that applies to following primitives.

When graphics data is stored by an application program in a GDF file, it can be supplied to GDDM by a single GSPUT call, rather than by a large number of calls to the individual primitives.

The interpretation of a GDF order string is equivalent to the execution of CALL statements corresponding to the orders in the string. Current position depends on the last order executed, and the current values of the attributes (color, line type, and so on) depend on the attribute setting orders that have occurred.

Note that errors can occur during interpretation of the individual orders as if corresponding calls had been made directly.

Since the GDF orders do not contain any window information, a call to GSWIN must be made before the GSPUT is done. If the default window of (0,100,0,100) is used, only part of the original picture appears because the GDF assumes a window that can be as large as (0,32767,0,32767). Appropriate GSWIN parameters can be taken from the first comment order in the GDF (see the description given for GSGET). Similarly, the aspect ratio of the original picture can be restored by a call to GSPS using the difference in the x bounds for the width and the difference in the y bounds for the height.

Parameters

control (4-byte binary integer)
The coordinate type used in the GDF data. On the AS/400 system, it must be:

2 2-byte binary integers

length (4-byte binary integer)
The length of the string of GDF orders to be used in drawing (see the graphics-data parameter). The string of orders is assumed to end when an apparent order code of X'FF' is encountered, or when the complete string has been interpreted, whichever is the sooner.

graphic-data (character)
A character variable, whose length is specified by the length parameter, and whose value your program supplies, that is used by GDDM instead of separate calls to OS/400 Graphics routines. You can use GDF data that has been generated by GSGET for this parameter. The detailed rules for formatting the GDF string are given in Appendix B, "GDF Order Descriptions."

Coding Example

The following example shows how to use GSPUT:

```
OPTION BASE 1
DIM BUF$*255    ! 255 is maximum allowed by BASIC
   .
   . (processing to read first GDF order, which is a comment)
   .
! Loop that includes the following:
   CALL GDDM('GSPUT',2,255,BUF$)
   .
   . (processing to read rest of GDF orders)
   .
! After all GDF orders have been read, send graphics to device
!  if desired
CALL GDDM('FSFRCE')
```

Note: Some GDF orders can require more than 255 bytes for the graphic-data parameter (in this example, BUF$). These include orders for graphics text (GSCHAP and GSCHAR with very long character strings), polyfillets, and polylines. If your graphics data includes such orders, you should use a language other than BASIC to write the program using GSPUT.

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, control values other than 2 are not allowed; therefore all graphics data files must contain 2-byte binary integer coordinates.

GSQCA – Query the Current Character Angle

```
GSQCA(dx,dy)
```

Returns the current slope of the character baseline.

Parameters

dx,dy (returned by GDDM) (short floating-point variables)
The current slope of the character baseline.

Coding Example

The following example shows how to use GSQCA:

```
DECIMAL DX,DY
CALL GDDM('GSQCA',DX,DY)
```

GSQCB – Query the Current Character Box Size

GSQCB(width,height)

Returns the current width and height of the character box.

Parameters

width,height (returned by GDDM) (short floating-point variables)
The current dimensions of the character box.

Coding Example

The following example shows how to use GSQCB:

```
DECIMAL WIDTH,HEIGHT
CALL GDDM('GSQCB',WIDTH,HEIGHT)
```

GSQCD – Query the Current Character Direction

GSQCD(character-direction-code)

Returns the code for the current character direction as set by GSCD.

Parameters

character-direction-code (returned by GDDM) (4-byte binary integer variable)
The current character direction code, from 0 through 4. For an explanation of the codes, see "GSCD – Set Current Character Direction" on page 2-53.

Coding Example

The following example shows how to use GSQCD:

```
INTEGER DIRECTION
CALL GDDM('GSQCD',DIRECTION)
```

GSQCEL – Query the Current Hardware Cell Size

```
GSQCEL(width,height)
```

Returns the hardware cell size in current window units. This call can be used to determine the area bounded by the cell in which the cursor lies.

For plotters and IPDS printers, GSQCEL returns a pseudo-cell size. This size is used as the default character box size as returned in GSQCB.

Parameter

width,height (returned by GDDM) (short floating-point variables)
The dimensions of the character cell.

Coding Example

The following example shows how to use GSQCEL:

```
DECIMAL WIDTH,HEIGHT
CALL GDDM('GSQCEL',WIDTH,HEIGHT)
```

GSQCH – Query the Current Character Shear

```
GSQCH(dx,dy)
```

Returns the current angle at which characters are sheared (this is set by GSCH).

Parameters

dx,dy (returned by GDDM) (short floating-point number variables)
Values that define the angle at which characters are sheared.

Coding Example

The following example shows how to use GSQCH:

```
DECIMAL DX,DY
CALL GDDM('GSQCH',DX,DY)
```

GSQCLP – Query the Clipping Mode

GSQCLP(clipping-mode)

Returns the current clipping mode.

Parameters

state (returned by GDDM) (4-byte binary integer variable)
The current clipping mode for the page. The values returned are:

0 Clipping is disabled
1 Clipping is enabled

Coding Example

The following example shows how to use GSQCLP:

```
INTEGER CLPMODE
CALL GDDM('GSQCLP',CLPMODE)
```

GSQCM – Query the Current Character Mode

```
GSQCM(character-mode)
```

Returns the current value of the character mode attribute (which is set by GSCM).

Parameters

character-mode (returned by GDDM) (4-byte binary integer variable)
The current character mode.

Coding Example

The following example shows how to use GSQCM:

```
INTEGER CHARMODE
CALL GDDM('GSQCM',CHARMODE)
```

GSQCOL – Query the Current Color

GSQCOL(color-index)

Returns the current value of the color attribute (which is set by GSCOL).

Parameters

color-index (returned by GDDM) (4-byte binary integer variable)
The current color attribute.

Coding Example

The following example shows how to use GSQCOL:

```
INTEGER COLIND
CALL GDDM('GSQCOL',COLIND)
```

GSQCP – Query the Current Position

```
GSQCP(x,y)
```

Returns the current position.

Parameters

x,y (returned by GDDM) (short floating-point variables)
The x and y world coordinates of the current position.

Coding Example

The following example shows how to use GSQCP:

```
DECIMAL X,Y
CALL GDDM('GSQCP',X,Y)
```

GSQCS – Query the Current Symbol Set Identifier

GSQCS(symbol-set-id)

Returns the current symbol set identifier.

Parameters

symbol-set-id (returned by GDDM) (4-byte binary integer variables)
The current symbol set identifier.

Coding Example

The following example shows how to use GSQCS:

```
INTEGER SSID
CALL GDDM('GSQCS',SSID)
```

GSQCT – Query the Current Color Table

```
GSQCT(color-table-id)
```

Returns the current color table identifier.

Parameters

color-table-id (returned by GDDM) (4-byte binary integer)
Receives the current color table identifier. If no color table has been selected explicitly by the GSCT function, a value of zero is returned.

Coding Example

The following example shows how to use GSQCT:

```
INTEGER COLTABID
CALL GDDM('GSQCT',COLTABID)
```

Differences between the System/370 Computer and the AS/400 System

This function is not supported on the System/370 computer.

GSQCTD – Query the Color Table Definition

```
GSQCTD(color-table-id,start-color,count,hue-array,lightness-array,
  saturation-array)
```

Returns the values of the entries in a color table.

Parameters

color-table-id (4-byte binary integer)
The identification of the color table.

start-color (4-byte binary integer)
The number of the first entry in the table to be returned. If the whole table is to be returned, this should be set to 1.

count (4-byte binary integer)
The number of color table entries to be returned. If the whole table is to be returned, this should be set to the size of the table (8).

hue-array (returned by GDDM) (array of short floating-point numbers)
The hue values for each of the table entries being returned. Values range from 0.0 through 1.0, with 0.0 and 1.0 indicating blue, and intermediate values indicating other hues.

lightness-array (returned by GDDM) (array of short floating-point numbers)
The lightness values for each of the table entries being returned. Values range from 0.0 through 1.0, with 0.0 indicating the least lightness (black), 1.0 indicating the most (white), and intermediate values indicating degrees between.

saturation-array (returned by GDDM) (array of short floating-point numbers)
The saturation values for each of the table entries being returned. Values range from 0.0 through 1.0, with 0.0 indicating a completely unsaturated color (gray), 1.0 indicating full saturation, and intermediate values indicating degrees between.

Coding Example

The following example shows how to use GSQCTD to query the characteristics of the whole color table:

```
OPTION BASE 1
DECIMAL HUE_ARR,LT_ARR,SAT_ARR
DIM HUE_ARR(8),LT_ARR(8),SAT_ARR(8)
CALL GDDM('GSQCTD',65,1,8,HUE_ARR(),LT_ARR(),SAT_ARR())
```

Differences between the System/370 Computer and the AS/400 System

This function is not supported on the System/370 computer.

GSQCUR – Query the Cursor Position

```
GSQCUR(inwin,x,y)
```

Returns the cursor position in current window units. The position returned is that of the cursor as last set by ASREAD. The x and y are in world coordinates.

Parameters

inwin (returned by GDDM) (4-byte binary integer variable)
One of the following:

0 The cursor is outside the window
1 The cursor is within the window

x,y (returned by GDDM) (short floating-point variables)
The x and y coordinates of the center of the character cell containing the cursor.

Coding Example

The following example shows how to use GSQCUR:

```
INTEGER INWIN
DECIMAL X,Y
CALL GDDM('GSQCUR',INWIN,X,Y)
```

GSQFLW – Query the Current Fractional Line Width

GSQFLW(linewidth-multiplier)

Returns the current line width as a floating-point value.

Parameters

linewidth-multiplier (returned by GDDM) (short floating-point variable)
The current value of the fractional line width.

Coding Example

The following example shows how to use GSQFLW:

```
DECIMAL LWMULT
CALL GDDM('GSQFLW',LWMULT)
```

GSQLT – Query the Current Line Type

```
GSQLT(line-type)
```

Returns the current value of the line type.

Parameters

line-type (returned by GDDM) (4-byte binary integers)
The current line type, from 0 through 8. For an explanation of the line type, see "GSLT – Set Current Line Type" on page 2-93.

Coding Example

The following example shows how to use GSQLT:

```
INTEGER LINTYP
CALL GDDM('GSQLT',LINTYP)
```

GSQLW – Query the Current Line Width

```
GSQLW(line-width)
```

Returns the current value of the line width. If the current line width is fractional, the integral part is returned. To obtain the fractional part, use GSQFLW.

Parameters

line-width (returned by GDDM) (4-byte binary integer variable)
The integer part of the value of the current line width, from 0 through 2. For an explanation of the line width, see "GSLW – Set Current Line Width" on page 2-94.

Coding Example

The following example shows how to use GSQLW:

```
INTEGER LINWID
CALL GDDM('GSQLW',LINWID)
```

GSQMAX – Query the Number of Segments

```
GSQMAX(number-of-segments,max-segment)
```

Returns the number of segment identifiers and the maximum segment identifier on the current page.

Parameters

number-of-segments (returned by GDDM) (4-byte binary integer variable)
The number of segments currently defined.

max-segment (returned by GDDM) (4-byte binary integer variable)
The maximum segment identifier currently defined.

Coding Example

The following example shows how to use GSQMAX:

```
INTEGER NUMSEG,MAXSEGID
CALL GDDM('GSQMAX',NUMSEG,MAXSEGID)
```

GSQMIX – Query the Current Color-Mixing Mode

GSQMIX(color-mixing-mode)

Returns the current value of the color-mixing mode.

Parameters

color-mixing-mode (returned by GDDM) (4-byte binary integer variable)
The current value of the color-mixing mode. Possible values are:

0 Default. This is the same as a value of 2.
1 Mix mode. Where two primitives cross, the color displayed is a mixture of the two colors. For the results of mixing, see Figure 2-7 on page 2-97.
2 Overpaint mode. The color of the current primitive takes precedence.
3 Underpaint mode. The color of the first primitive drawn takes precedence.
4 Exclusive-OR mode. Where two primitives cross, the resulting color is a result of the logical exclusive OR of the color index values.

Coding Example

The following example shows how to use GSQMIX:

```
INTEGER MIXMOD
CALL GDDM('GSQMIX',MIXMOD)
```

GSQMS – Query the Current Marker Symbol

```
GSQMS(marker-type)
```

Returns the current marker symbol number.

Parameters

marker-type (returned by GDDM) (4-byte binary integer variable)
The current marker symbol number for the current graphics segment. The number can be from 0 through 10 or from 65 through 254. For an explanation of the marker types, see "GSMS – Set Current Marker Symbol" on page 2-100.

Coding Example

The following example shows how to use GSQMS:

```
INTEGER MRKTYP
CALL GDDM('GSQMS',MRKTYP)
```

GSQMSC – Query the Current Marker Scale

```
GSQMSC(scale)
```

Returns the current marker scale to be used when drawing marker symbols defined as vector symbols.

Parameters

scale (returned by GDDM) (short floating-point variable)
The scale of marker symbols with respect to the default marker box.

Coding Example

The following example shows how to use GSQMSC:

```
DECIMAL SCAL
CALL GDDM('GSQMSC',SCAL)
```

GSQNSS – Query the Number of Loaded Symbol Sets

```
GSQNSS(number-of-symbol-sets)
```

Returns the number of loaded symbol sets. This call allows you to specify an accurate count parameter value on a subsequent GSQSS request.

Parameters

number-of-symbol-sets (returned by GDDM) (4-byte binary integer variable)
The number of symbol sets loaded by GSLSS.

Coding Example

The following example shows how to use GSQNSS:

```
INTEGER NUMSS
CALL GDDM('GSQNSS',NUMSS)
```

GSQPAT – Query the Current Shading Pattern

```
GSQPAT(pattern-number)
```

Returns the current shading pattern number.

Parameters

pattern-number (returned by GDDM) (4-byte binary integer variable)
The current shading pattern number, from 0 through 16. For an explanation of the pattern numbers, see "GSPAT – Set Current Shading Pattern" on page 2-102.

Coding Example

The following example shows how to use GSQPAT:

```
INTEGER PATNUM
CALL GDDM('GSQPAT',PATNUM)
```

GSQPS – Query the Picture Space Definition

```
GSQPS(width,height)
```

Returns the picture space definition. This call can be used to adapt a picture layout to suit the shape of the graphics field.

Parameters

width,height (returned by GDDM) (short floating-point variables)
The width and height of the picture space. One of the values returned is 1; the other is less than or equal to 1.

Coding Example

The following example shows how to use GSQPS to set the viewport to be the same as the graphics field:

```
DECIMAL WIDTH,HEIGHT
CALL GDDM('GSQPS',WIDTH,HEIGHT)
CALL GDDM('GSVIEW',0.0,WIDTH,0.0,HEIGHT)
```

GSQSS – Query Loaded Symbol Sets

```
GSQSS(count,types,symbol-set-names,symbol-set-ids)
```

Returns information about symbol sets loaded by GSLSS.

Each of the parameters (except the count parameter) is an array. The count parameter specifies the number of symbol sets to be queried. The array parameters return information about the symbol sets that were loaded first after graphics was last initialized. If there are fewer arrays loaded than specified in the count parameter, the remaining elements in the symbol-set-ids array are not changed from their former value.

Parameters

count (4-byte binary integer variable)
The number of loaded symbol sets.

types (returned by GDDM) (array of 4-byte binary integer variables)
The types of symbol definitions, which are:

1 Image symbol set
2 Vector symbol set
4 Marker symbol set

symbol-set-names (returned by GDDM) (array of 8-byte character variables)
The *GSS object names from which the symbol sets were loaded.

symbol-set-ids (returned by GDDM) (array of 4-byte binary integer variables)
The identifiers associated with each symbol set.

Coding Example

The following example shows how to query as many as four current graphics symbol sets:

```
OPTION BASE 1
INTEGER COUNT,TYPS,SSIDS
DIM TYPS(4)
DIM SSIDS(4)
DIM SSNAMES$(4)*8
CALL GDDM('GSQSS',COUNT,TYPS(),SSNAMES$(),SSIDS())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, excess array elements are not set to zero or blank. Also, symbol set types 3 (shading pattern set) and 5 (4250 composed page printer font) are not supported.

GSQTB – Query the Text Box

```
GSQTB(length,string,count,xarray,yarray)
```

Returns the x and y components of the relative coordinates of the four corners of the text box. The text box is the parallelogram that encloses the character string. Also returned are the relative coordinates of the concatenation point; that is, the position where the next character would be. All coordinates are relative to the starting point, as defined in the GSCD call. The box is evaluated using the current character mode, box, angle, shear, direction, and set.

Parameters

length (4-byte binary integer)
The number of characters in the string.

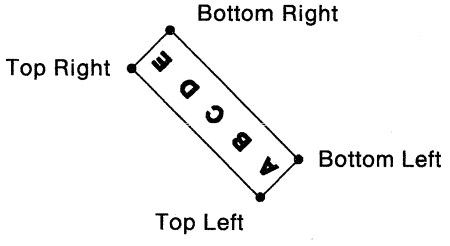
string (character string)
The character string to be processed.

count (4-byte binary integer)
The number of elements in xarray and yarray. Any values after the fifth are ignored.

xarray,yarray (returned by GDDM) (short floating-point variables)
The relative coordinates of the text box as follows:

Element 1: Top left corner
Element 2: Bottom left corner
Element 3: Top right corner
Element 4: Bottom right corner
Element 5: Concatenation point

The terms *top left* and so on are literally true when the baseline is parallel to the x axis and running left to right, and there is no character shear. If the character string is rotated or sheared, the term *top left* applies to the corner of the box that appears top left when no rotation or shear is applied.



RV2F745-0

Coding Example

The following example shows how GSQTB is used:

```
OPTION BASE 1
DECIMAL XARRAY,YARRAY
DIM XARRAY(5),YARRAY(5)
CALL GDDM('GSQTB',5,'ABCDE',5,XARRAY(),YARRAY())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, any excess array elements are not set to zero.

GSQVIE – Query the Current Viewport Definition

```
GSQVIE(left-boundary,right-boundary,lower-boundary,upper-boundary)
```

Returns the current viewport definition.

Parameters

left-boundary,right-boundary (returned by GDDM) (short floating-point variables)
The position within the picture space of the left and right boundaries of the viewport.

lower-boundary,upper-boundary (returned by GDDM) (short floating-point variables)
The position within the picture space of the lower and upper boundaries of the viewport.

Coding Example

The following example shows how to use GSQVIE:

```
DECIMAL X1,X2,Y1,Y2
CALL GDDM('GSQVIE',X1,X2,Y1,Y2)
```

GSQWIN – Query the Current Window Definition

```
GSQWIN(left-boundary,right-boundary,lower-boundary,upper-boundary)
```

Returns the current window definition.

Parameters

left-boundary,right-boundary (returned by GDDM) (short floating-point variables)
The left and right boundaries of the window.

lower-boundary,upper-boundary (returned by GDDM) (short floating-point variables)
The lower and upper boundaries of the window.

Coding Example

The following example shows how to use GSQWIN:

```
DECIMAL LEFTB,RIGHTB,LOWB,UPPB
CALL GDDM('GSQWIN',LEFTB,RIGHTB,LOWB,UPPB)
```

GSRSS — Releasing a Graphics Symbol Set

```
GSRSS(type,symbol-set-id)
```

Releases the symbol set specified. The type must correspond exactly to that specified when the symbol set was loaded. A symbol set can be released, freeing the storage occupied, when the application program no longer needs it. A set should not be released while a graphics picture using it still exists.

Parameters

type (4-byte binary integer)
The type of the symbol set to be released, being either:

1 ISS (image symbol set)
2 VSS (vector symbol set)
4 Marker symbol set (ISS or VSS)

symbol-set-id (4-byte binary integer)
The identifier of the symbol set to be released (see "GSLSS — Load a Graphics Symbol Set from Auxiliary Storage" on page 2-90).

Coding Example

In the following example, the program releases the symbol set for Duplex Roman Principal.

```
CALL GDDM('GSRSS',2,65)
```

Differences between the System/370 Computer and the AS/400 System

The System/370 computer also supports symbol set types 3 (shading pattern set).

GSSCLS – Close the Current Segment

GSSCLS

Causes the current segment to be closed, so that no more primitives can be added to it. After this operation, there is no current segment within the page. Closing a segment does not delete the segment or affect the graphics primitives that are displayed.

Parameters

None.

Coding Example

The following example shows how to use GSSCLS:

```
CALL GDDM('GSSCLS')
```

GSSDEL – Delete a Segment

```
GSSDEL(segment-id)
```

Causes the specified segment, together with any graphics primitives defined within it, to be deleted.

When a visible segment is deleted, all other segments are redisplayed; however, the effect appears at the device only at the next ASREAD or FSFRCE.

If the segment is current when it is deleted, there is no current segment after this operation.

Parameters

segment-id (4-byte binary integer)
The identifier of the segment to be deleted.

Coding Example

The following example shows how to use GSSDEL:

```
CALL GDDM('GSSDEL',4)
```

GSSEG – Create a Segment

GSSEG(segment-id)

Creates a segment with the specified identification number. A segment is a group of graphics primitives that share the current viewport and window.

Use GSSEG for the following:

- To group graphics primitives together.
- To reset all attributes to defaults.
- To set the current position at the world-coordinate origin.
- When your program will store a picture (using GSGET) in a graphics data file. This is called retained data. For a description of retained and temporary data, refer to the *GDDM Programming Guide*.

The segment created becomes the current segment, to which subsequent primitives are added. The window and viewport cannot be changed while a segment is being constructed.

The following shows the position of this call in the graphics hierarchy:

Page	FSPCRT
Graphics field	GSFLD
Picture space	GSPS
Viewport	GSVIEW
Window	GSWIN
Segment	**GSSEG**

Parameters

segment-id (4-byte binary integer)
The identifier of the segment (can be zero).

If this parameter is zero, the segment is unnamed and cannot be explicitly deleted. If the segment-id is nonzero (and positive), it must be unique within the current page. Segment identifier 32,767 is reserved and cannot be used.

Coding Example

The following example shows how to use GSSEG:

```
CALL GDDM('GSSEG',4)
```

GSVECM – Vectors

GSVECM(count,vector)

Performs a series of operations, which either draw a line or move the current position to a specified end point. The color, line width, and line type of lines drawn are given by the current values of these attributes.

The current position is set to the last end point.

If clipping is disabled and any specified point lies outside the window boundaries, that line, subsequent lines, and current position are undefined. If clipping is enabled, only those sections of the lines within the current window are visible.

Parameters

count (4-byte binary integer)
The total number of lines or moves to be performed.

vector (array of 4-byte binary integers)
A series of operations in the order (control1,x1,y1, control2,x2,y2, ... controli,xi,yi,... controln,xn,yn), where for each operation the control value determines whether a line is drawn or a move is made. Control values can be:

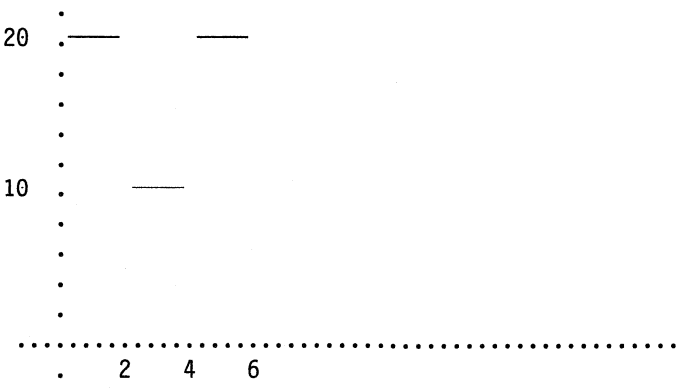
0 Move to the point (xi,yi)
1 Draw a line to the point (xi,yi)

Coding Example

In the following example, the program uses GSVECM to draw a broken line:

```
OPTION BASE 1
INTEGER VECTOR
DIM VECTOR(18)
MAT READ VECTOR
DATA 0,0,20, 1,2,20, 0,2,10, 1,4,10, 0,4,20, 1,6,20
CALL GDDM('GSVECM',6,VECTOR())
```

The following shows how the broken line is drawn:



GSVIEW – Define a Viewport

```
GSVIEW(left-boundary,right-boundary,lower-boundary,upper-boundary)
```

Explicitly specifies the current viewport boundaries in picture-space units. If not specified by GSPS earlier in the program, the width and height of the picture space depend on the device. You can use GSQPS to find out what they are.

The following shows the position of this call in the graphics hierarchy:

Page	FSPCRT
Graphics field	GSFLD
Picture space	GSPS
Viewport	**GSVIEW**
Window	GSWIN
Segment	GSSEG

If you do not use GSVIEW, the viewport takes up the whole of the picture space.

A viewport is a subregion of the total picture space. Viewports can be used to position the parts of a composite picture. The viewport boundaries are parallel to those of the picture space, and must be entirely within the picture space.

When the picture space of a graphics field is defined or defaulted, the current viewport is set to cover the complete picture space. *By default, therefore, the viewport covers the picture space.*

If GSVIEW, GSPS, and GSWIN have not been called, the default values are as follows:

- On a graphics work station, the left boundary is 0, the right boundary is 1, the lower boundary is 0, and the upper boundary is 0.529663.
- On plotters, the left boundary is 0, the right boundary is 1, the lower boundary is 0, and the upper boundary is 0.72.
- On printers, the default width for the QPGDDM printer file (as shipped) is 1 and the default height is 0.757.

Parameters

left-boundary,right-boundary (short floating-point numbers)
The left and right viewport boundaries in picture-space units. The left boundary must not be larger than the right boundary.

lower-boundary,upper-boundary (short floating-point numbers)
The lower and upper viewport boundaries in picture-space units. The lower boundary must not be larger than the upper boundary.

Coding Example

The following example shows how to use GSVIEW:

```
! Set picture space
CALL GDDM('GSPS',1.0,0.4)
! Set viewport
CALL GDDM('GSVIEW', 0.0, 1.0, 0.0, 0.5)
```

GSWIN – Define a Window

GSWIN(left-boundary,right-boundary,lower-boundary,upper-boundary)

Explicitly specifies the world coordinates to be used in drawing subsequent graphics primitives in the viewport. Before the first GSWIN is issued, the default values of the world coordinates are 0 through 100 in the x direction and 0 through 100 in the y direction.

The following shows the position of this call in the graphics hierarchy:

Page	FSPCRT
Graphics field	GSFLD
Picture space	GSPS
Viewport	GSVIEW
Window	**GSWIN**
Segment	GSSEG

The world coordinates specified with this call are used to position all graphics primitives on the current page until a new window is specified. All graphics primitives in a segment must have the same window.

Parameters

left-boundary,right-boundary (short floating-point numbers)
The left and right boundaries of the window. The left boundary is mapped to the left edge of the viewport, and the right boundary is mapped to the right. If clipping is in effect, no part of a primitive (line, arc, character, or area) is visible to the left of the left boundary, or to the right of the right boundary.

lower-boundary,upper-boundary (short floating-point numbers)
The lower and upper boundaries of the window. The lower boundary is mapped to the bottom of the viewport, and the upper boundary is mapped to the top. If clipping is in effect, no part of a primitive (line, arc, character, or area) is visible below the lower boundary or above the upper boundary.

Coding Example

The following example shows how to use GSWIN:

```
CALL GDDM('GSWIN', 0.0, 200.0, 0.0, 200.0)
```

Summary of Data Types for GDDM Routines

Figure 2-8 is a table in which you can look up the parameter types for the GDDM routines. For example, for ASREAD, there are three parameters and they are all 4-byte binary integers that must be variables so that GDDM can return values to your program.

Figure 2-8 (Page 1 of 2). Summary of Data Types for the GDDM Routines

Name	Parm 1	Parm 2	Parm 3	Parm 4	Parm 5	Parm 6	Parm 7	Parm 8
ASREAD	in var	in var	in var					
DSCLS	in	in						
DSDROP	in	in						
DSOPEN	in	in	ch	in	in arr	in	ch arr	
DSQDEV	in	ch var	in	in arr	in	ch arr	in	in arr var
DSQUID	in var							
DSQUSE	in	in var						
DSRNIT	in	in						
DSUSE	in	in						
FSALRM								
FSEXIT	ch var	in						
FSFRCE								
FSINIT								
FSPCLR								
FSPCRT	in	in	in	in				
FSPDEL	in							
FSPQRY	in	in var	in var	in var				
FSPSEL	in							
FSQCPG	in var							
FSQDEV	in	in arr var						
FSQERR	in	ch var						
FSQUPG	in var							
FSREST	in							
FSRNIT								
FSTERM								
GSARC	fp	fp	fp					
GSAREA	in							
GSCA	fp	fp						
GSCB	fp	fp						
GSCD	in							
GSCH	fp	fp						
GSCHAP	in	ch						
GSCHAR	fp	fp	in	ch				
GSCLP	in							
GSCLR								
GSCM	in							
GSCOL	in							
GSCS	in							
GSCT	in							
GSCTD	in	in	in	fp arr	fp arr	fp arr		
GSELPS	fp	fp	fp	fp	fp			
GSENDA								
GSFLD	in	in	in	in				
GSFLW	fp							
GSGET	in	ch var	in var					
GSGETE								
GSGETS	in	in arr						
GSIMG	in	in	in	in	ch			
GSIMGS	in	in	in	in	ch	fp	fp	

Figure 2-8 (Page 2 of 2). Summary of Data Types for the GDDM Routines

Name	Parm 1	Parm 2	Parm 3	Parm 4	Parm 5	Parm 6	Parm 7	Parm 8
GSLINE	fp	fp						
GSLSS	in	ch	in					
GSLT	in							
GSLW	in							
GSMARK	fp	fp						
GSMIX	in							
GSMOVE	fp	fp						
GSMRKS	in	fp arr	fp arr					
GSMS	in							
GSMSC	fp							
GSPAT	in							
GSPFLT	in	fp arr	fp arr					
GSPLNE	in	fp arr	fp arr					
GSPS	fp	fp						
GSPUT	in	in	ch					
GSQCA	fp var	fp var						
GSQCB	fp var	fp var						
GSQCD	In var							
GSQCEL	fp var	fp var						
GSQCH	fp var	fp var						
GSQCLP	in var							
GSQCM	in var							
GSQCOL	in var							
GSQCP	fp var	fp var						
GSQCS	in							
GSQCT	in var							
GSQCTD	in	in	in	fp arr var	fp arr var	fp arr var		
GSQCUR	in var	fp var	fp var					
GSQFLW	fp var							
GSQLT	in var							
GSQLW	in var							
GSQMAX	in var	in var						
GSQMIX	in var							
GSQMS	in var							
GSQMSC	fp var							
GSQNSS	in var							
GSQPAT	in var							
GSQPS	fp var	fp var						
GSQSS	in	in arr var	ch arr var	in arr var				
GSQTB	in	ch	in	fp arr var	fp arr var			
GSQVIE	fp var	fp var	fp var	fp var				
GSQWIN	fp var	fp var	fp var	fp var				
GSRSS	in	in						
GSSCLS								
GSSDEL	in							
GSSEG	in							
GSVECM	in	in arr						
GSVIEW	fp var	fp var	fp var	fp var				
GSWIN	fp var	fp var	fp var	fp var				

Note:

in = 4-byte binary integer
fp = Short floating-point number
ch = Character string
arr = Array
var = Variable (whose value is returned by GDDM)

Chapter 3. Presentation Graphics Routines

Concepts of Presentation Graphics Routines

Presentation Graphics routines can be used to draw the following types of chart:

- Line graphs
- Surface charts
- Bar charts
- Histograms
- Scatter plots
- Pie charts
- Venn diagrams

The following is a list of some of the more important GDDM routines that affect Presentation Graphics routines (for a description of them, see Chapter 2).

ASREAD (either ASREAD or FSFRCE required to get graphics output)
FSFRCE (either ASREAD or FSFRCE required to get graphics output)
FSINIT (required before using graphics calls)
DSCLS
DSDROP
DSOPEN (DSOPEN and DSUSE both required to send output to a plotter)
DSUSE (DSOPEN and DSUSE both required to send output to a plotter)
FSEXIT
FSPCRT
FSRNIT
FSTERM (required after using graphics calls)
GSFLD
GSLSS
GSPS

You can also use any of the GDDM query routines related to the above (such as DSQUSE and GSQPS).

Figure 3-1 on page 3-2 shows the GDDM and Presentation Graphics routines to use for specific tasks you might have when using Presentation Graphics:

Notes:

1. At least one call from this group is required.
2. Both of these calls are required.
3. At least one of these calls is required.

Figure 3-1 (Page 1 of 3). Routines You Can Use in Presentation Graphics

Chart Types	
Line charts	CHPLOT
Scatter plots	CHPLOT
Surface charts	CHSURF
Bar charts	CHBAR (*see note 1*)
Histograms	CHHIST
Pie charts	CHPIE
Venn diagrams	CHVENN
Drawing Charts with Presentation Graphics Routines	
Control operations	
• Graphics program	FSINIT,FSRNIT,FSTERM (*see note 2*)
• Error handling	FSEXIT
• Display controls	ASREAD,FSFRCE,FSREST (*see note 3*)
• Symbol sets	GSLSS,GSCS
• Picture controls	FSPCRT,GSPS,GSFLD
• Device open and close	DSOPEN,DSCLS,DSQUID,DSRNIT
• Using devices	DSUSE,DSDROP,DSQUSE
• Sounding device alarm	FSALRM
Chart Definition	
Chart layout	
• Size	CHAREA
• Margins	CHHMAR,CHVMAR
• Frame	CHSET(**NCBOX**|CBOX|CBACK),CHBATT
• Basic character box size	CHCGRD
Chart features	
• Headings	
– Text	CHHEAD
– Heading or no heading	CHSET(**HEADING**|NOHEADING)
– Text attributes	CHHATT
– Position	CHSET(**HTOP**|HBOTTOM), CHSET(**HCENTER**|HLEFT|HRIGHT)
• Chart legends	
– Drawing or not	CHSET(**LEGEND**|NOLEGEND)
– For pie charts: legend or spider tags	CHSET(**PIEKEY**|SPIDER)
– Attributes	CHKATT
– Legend position within margin	CHKEYP
– Final position (offset)	CHKOFF
– Maximum dimension	CHKMAX
– Order of items in legend	CHSET(**KNORMAL**|KREVERSED)
– Blank legend area or not	CHSET(**NBKEY**|BKEY)
– Box or not	CHSET(**NKBOX**|KBOX)
– Labels for legend keys	CHKEY
• Chart notes	
– Attributes	CHNATT
– Note position and text	CHNOTE
– Final position (offset)	CHNOFF
– Blank note area or not	CHSET(**NBNOTE**|BNOTE)
– Box or not	CHSET(**NONBOX**|NBOX)

Figure 3-1 (Page 2 of 3). Routines You Can Use in Presentation Graphics

Chart features (continued)	
• Axes	
– Axis or no axis	CHSET(**AXIS**|NOAXIS)
– Duplicate axis	CHSET(**XNODUP**|XDUP|**YNODUP**|YDUP)
– Line attributes	CHAATT
– Position	CHXSET(**LOWAXIS**,MIDDLE,HIGHAXIS,INTERCEPT), CHXINT,CHYINT, CHSET(**YVERTICAL**|XVERTICAL)
– Type of scale	CHXSET(**LINEAR**|LOGARITHMIC), CHYSET(**LINEAR**|LOGARITHMIC)
– Range	
Explicit range	CHXRNG,CHYRNG,
Auto-ranging with zero	CHXSET(**FORCEZERO**|NOFORCEZERO), CHYSET(**FORCEZERO**|NOFORCEZERO)
Tick mark intervals	CHXTIC,CHYTIC
Types of ticks	CHXSET(**NTICK**|PTICK|XTICK|PLAIN), CHYSET(**NTICK**|PTICK|XTICK|PLAIN)
– Axis title	
Text attributes	CHTATT
Position	CHXSET(**ATCENTER**|ATEND|ATABOVE), CHYSET(**ATCENTER**|ATEND|ATABOVE)
Text	CHXTTL,CHYTTL
– Axis labels	
Text attributes	CHLATT
Position	CHXSET(**LABADJACENT**|LABMIDDLE|NOLAB), CHYSET(**LABADJACENT**|LABMIDDLE|NOLAB)
Axis label attributes	CHXLAT,CHYLAT
Blank label area or not	CHSET(**NBLABEL**|BLABEL)
Type of label	CHXSET(**NUMERIC**|DATE|ALPHANUMERIC), CHYSET(**NUMERIC**|DATE|ALPHANUMERIC)
Scale factor	CHXSCL,CHYSCL
Punctuate 4-digit integers or not	CHSET(**NPGFS**|PGFS)
Month labels	CHXMTH,CHYMTH CHXSET(**NOSKIP**|SKIP) CHYSET(**NOSKIP**|SKIP)
Day labels	CHXDAY,CHYDAY
Abbreviate month and day labels	CHSET(**ABREV**|FULL|LETTER)
User-defined labels	CHXLAB,CHYLAB
– When drawn	
When defined, after data plotted, suppressed	CHSET(**IDRAW**|DRAW|NDRAW),
Draw at specific time	CHDRAX
– Other reference lines	
- Grid lines	
Attributes	CHGATT
Drawing or not	CHXSET(**NOGRID**|GRID), CHYSET(**NOGRID**|GRID)
- Axis translation lines and datum lines	
Attributes	CHDATT
Drawing or not	CHXDTM,CHYDTM

Figure 3-1 (Page 3 of 3). Routines You Can Use in Presentation Graphics

Drawing the Chart Using Component Attributes	
Line charts	CHCOL,CHLT,CHMARK,CHSET(NOMARKERS), CHSET(CURVE),CHFINE,CHPLOT, CHSET(VALUES|**NOVALUES**), CHSET(BVALUES|**NBVALUES**)
Scatter plots	CHCOL,CHMARK,CHSET(**LINES**|NOLINES),CHPLOT
Surface charts	CHCOL,CHSET(CURVE),CHFINE,CHPAT, CHSET(**FILL**|INFILL|NOFILL), CHSET(**ABSOLUTE**|RELATIVE),CHSURF, CHSET(VALUES|**NOVALUES**), CHSET(BVALUES|**NBVALUES**)
Bar charts	CHCOL,CHPAT,CHSET(NOFILL), CHSET(CVALUES|VALUES|**NOVALUES**), CHSET(BVALUES|**NBVALUES**), CHVCHR,CHVATT,CHGAP,CHBAR
Multiple – bar charts	Add to *Bar charts:* CHSET(MBAR),CHNUM
Composite – bar charts	Add to *Bar charts:* CHSET(CBAR),CHSET(RELATIVE)
Floating – bar charts	Add to *Bar charts:* CHSET(FBAR)
Pie charts	CHCOL,CHPAT,CHSET(NOFILL),CHVCHR,CHVATT, CHSET(VALUES|**NOVALUES**), CHSET(BVALUES|**NBVALUES**), CHSET(SPISECTOR),CHSET(ABPIE),CHPIE, CHPCTL,CHPEXP
Multiple – pie chart	Add to *Pie charts:* CHNUM,CHPIER,CHSET(PROPIE),CHXLAB
Histograms	CHCOL,CHSET(NOFILL),CHPAT,CHSET(RELATIVE), CHSET(NORISERS),CHHIST.
Venn diagrams	CHCOL,CHSET(NOFILL),CHPAT,CHVENN
More Chart Control Routines	
Reset processing state	CHSTRT
Reinitialize Presentation Graphics routines	CHRNIT
Terminate Presentation Graphics routines	CHTERM

Drawing charts involves two states:

- State 1: Tailoring the default layout to suit your specific needs (also called *chart definition*).
- State 2: Passing the data and drawing the chart (also called *chart construction*). The calls CHBAR, CHHIST, CHPIE, CHPLOT, CHSURF, and CHVENN put your program in state 2.

Calls in state 1 merely set up information to be used when the chart is constructed in state 2; therefore, they can be made in any order. Calls in state 2 cause the

construction of parts of the graph and are therefore irreversible (unless you start again from scratch by using CHSTRT). See Figure 3-2.

Figure 3-2 (Page 1 of 3). Presentation Graphics Routines Valid in State 1 and State 2

CALL	Description	State 1	State 2
CHAATT	Axis line attributes	Y	
CHAREA	Chart area	Y	
CHBAR	Bar charts	Y	Y
CHBATT	Framing box attributes	Y	
CHCGRD	Character space and size	Y	
CHCOL	Color table	Y	
CHDATT	Datum line attributes	Y	Y
CHDRAX	Specific control of axis drawing		Y
CHFINE	Curve fitting smoothness	Y	Y
CHGAP	Spacing between bars	Y	Y
CHGATT	Grid line attributes	Y	
CHGGAP	Spacing between bar groups	Y	Y
CHHATT	Heading text attributes	Y	
CHHEAD	Heading text	Y	
CHHIST	Histograms	Y	Y
CHHMAR	Chart margins	Y	
CHKATT	Legend text attributes	Y	
CHKEY	Legend key labels	Y	
CHKEYP	Legend base position	Y	
CHKMAX	Maximum legend width/height	Y	
CHKOFF	Legend offsets	Y	
CHLATT	Axis label text attributes	Y	
CHLT	Component line type table	Y	
CHMARK	Component marker table	Y	
CHNATT	Text attributes for note	Y	Y
CHNOFF	Offsets for notes	Y	Y
CHNOTE	Construct a chart note		Y
CHNUM	Set number of components	Y	Y
CHPAT	Component shading pattern table	Y	
CHPCTL	Control pie chart slices	Y	
CHPEXP	Exploded slices in pie charts	Y	
CHPIE	Pie charts	Y	Y
CHPIER	Size specification	Y	
CHPLOT	Line graphs and scatter plots	Y	Y
CHRNIT	Reinitialize Presentation Graphics routines	Y	
CHSET	Specify chart options (see following options)		

Figure 3-2 (Page 2 of 3). Presentation Graphics Routines Valid in State 1 and State 2

CALL	Description	State 1	State 2
ABRE	Date label abbreviations	Y	
ABSO	Absolute or relative data	Y	
BKEY	Legend area blanked	Y	
BLAB	Label area blanked	Y	Y
BNOT	Note area blanked	Y	Y
BVAL	Blanking behind bar values	Y	Y
CBOX	Chart area boxed and shaded	Y	
CURV	Curve fitting	Y	Y
FILL	Shading method	Y	Y
HCEN	Heading justification	Y	
HEAD	Heading or no heading	Y	
HTOP	Heading position	Y	
IDRA	Time of drawing axes	Y	Y
KNOR	Order of elements in legend	Y	
KBOX	Box around legend	Y	
LEGE	Legend or no legend	Y	Y
LINE	Lines in line graphs	Y	Y
MARK	Markers on line graphs	Y	Y
MBAR	Type of bar chart	Y	Y
NBOX	Box around notes	Y	Y
PERP	Type of data for pie chart	Y	Y
PGFS	Punctuation of values	Y	Y
PIEK	Pie chart labeling	Y	
PROP	Relative sizes if pie charts	Y	
RISE	Risers in histograms	Y	Y
SPIS	Pie chart labeling	Y	Y
VALU	Values in pie and bar charts	Y	Y
VONTOP	Bar values in bar charts	Y	Y
XDUP	Secondary x axis	Y	Y
YDUP	Secondary y axis	Y	Y
YVER	Axis orientation	Y	
CHSTRT	Reset to state 1		Y
CHSURF	Surface charts	Y	Y
CHTATT	Axis title text attributes	Y	
CHTERM	Terminate Presentation Graphics routines	Y	Y
CHVATT	Values text attributes (bar & pie)	Y	Y
CHVCHR	Number of bar chart characters	Y	Y
CHVENN	Venn diagrams	Y	Y
CHVMAR	Chart margins	Y	
CHXDAY	Day labels	Y	
CHYDAY	Day labels	Y	

Figure 3-2 (Page 3 of 3). Presentation Graphics Routines Valid in State 1 and State 2

CALL	Description	State 1	State 2
CHXDTM	Datum lines	Y	Y
CHYDTM	Datum lines	Y	Y
CHXINT	Intercept	Y	
CHYINT	Intercept	Y	
CHXLAB	Label text	Y	
CHYLAB	Label text	Y	
CHXLAT	Label attributes	Y	
CHYLAT	Label attributes	Y	
CHXMTH	Month labels	Y	
CHYMTH	Month labels	Y	
CHXRNG	Explicit ranges	Y	
CHYRNG	Explicit ranges	Y	
CHXSCL	Scale factor	Y	
CHYSCL	Scale factor	Y	
CHXSEL	Axis selection	Y	Y
CHYSEL	Axis selection	Y	Y
CHXSET	Specify axis option (all options)	Y	
CHYSET	Specify axis option (all options)	Y	
CHXTIC	Scale mark interval	Y	
CHYTIC	Scale mark interval	Y	
CHXTTL	Axis title test	Y	
CHYTTL	Axis title text	Y	

Missing Data Values

Each data group must contain the same number of x and y values. If one or more values is missing from the data, supply a dummy value. This value should be 1E20 (10^{20}) for the RPG/400 language, and 1E35 (10^{35}) for all other supported languages. This type of dummy value is called a missing value.

This applies to all chart types except Venn diagrams.

Alphabetic List of Presentation Graphics Routines

In this section, the Presentation Graphics routines, with their syntax and explanations, are listed alphabetically.

The syntax is followed by a description of the routine, an explanation of the parameters, and a sample of the call in BASIC. For information on calling graphics routines in other languages, see "High-Level Languages" on page 1-2.

The description also tells you in which state the call is valid. State 1 is the state of a program using Presentation Graphics routines before the first plot has been made. State 2 is the state of the program after the first plot has been made (which constructs the axes).

All CHY... calls are listed under the equivalent CHX call. For example, CHYSCL is described with CHXSCL.

Syntax Rules

The syntax shown for each Presentation Graphics routine includes all the parameters for the call.

Each parameter shown is required; that is, if you use the graphics routine, you must specify something for each parameter, even if your program does not use the parameter or if the parameter is ignored on the AS/400 system. (This happens when the parameter is used on the System/370 computer but is not needed on the AS/400 system.)

The syntax shown for each Presentation Graphics routine is not a complete CALL statement in any language. Only the parts of the syntax that are specific to OS/400 Graphics are presented. This emphasizes the important aspects of the call. You must code a valid CALL statement in the language you are using. For example,

```
CHHEAD(length,text)
```

is representative of the syntax in this manual. However, you would code

```
CALL GDDM('CHHEAD',23,'Sample character string')
```

in a BASIC program, and

```
CALL CHHEAD(LENGTH,'Sample character string');
```

in a PL/I program. For more details on coding graphics calls in various languages, see "High-Level Languages" on page 1-2.

You must specify the correct data types in your program. If the data types do not match those required by OS/400 Graphics, a function check can occur.

The parameter names have been chosen to be as meaningful as possible. For example, in

```
CHHEAD(length,text)
```

length is the length of the text and text is a chart heading.

CHAATT – Axis Line Attributes

```
CHAATT(count,array)
```

Overrides or resets the default attributes to be used for each axis line and its major and minor scale marks.

The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Line type	Solid
Line width	Normal

Valid only in state 1.

Parameters

count (4-byte binary integer)

The number of elements in the array parameter. If count is greater than 12, only 12 attributes are changed (the remainder of the elements in array have no effect). If count is less than 12, the rest of the attributes keep their previous setting.

When count is zero, all defaults are reinstated.

array (array of 4-byte binary integers)

The attributes for the axes. The first set of three attributes is for the primary x axis, the second set is for the primary y axis, the third set is for the secondary x axis, and the fourth set is for the secondary y axis.

1. Color (except for −1, valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Line type (except for −1, valid values have the same effect as the GDDM routine GSLT).

   For valid values of line-type, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

3. Line width (except for −1, valid values have the same effect as the GDDM routine GSLW). Valid values are:

−1	Leave unchanged
0	Default (normal)
1	Normal
2	Double

Coding Example

The following example shows how to use CHAATT:

```
OPTION BASE 1
INTEGER ATT
DIM ATT(6)
MAT READ ATT
DATA -1,0,2, -1,0,2
CALL GDDM('CHAATT',6,ATT())
```

In this example, color of both primary axes is left unchanged, their line types are the default (solid), and they are made double width. The secondary axes are left unchanged (because array ATT has only six elements).

CHAREA – Chart Area

```
CHAREA(x1,x2,y1,y2)
```

Overrides the default area to be used for the chart.

By default, the whole of the screen (or printed page) is used to display the chart. If this is not your intention, you can specify a graphics field (using the GDDM call GSFLD) and optionally a picture space within the graphics field (using the GDDM call GSPS).

Whether the graphics field and picture space have been explicitly defined or not, the chart area is specified in terms of the picture space. A call to the GDDM routine GSQPS must be used to obtain the defined or defaulted picture space dimensions. The parameters to CHAREA are then given as fractions of these values.

CHSET(CBOX) or CHSET(CBACK) can be used to draw a framing box around the chart area or to shade in the background.

Valid only in state 1.

Parameters

x1,x2 (short floating-point numbers)
The left and right boundaries of the chart area in picture-space units.

y1,y2 (short floating-point numbers)
The lower and upper boundaries of the chart area in picture-space units.

Coding Example

The following example shows how to use CHAREA:

```
CALL GDDM('CHAREA', 0.0, 1.0, 0.0, 0.5)
```

CHBAR – Plot a Bar Chart

```
CHBAR(data-groups,count,y-values)
```

Plots one or more bar chart data groups on the currently selected axes. The type of bar chart constructed is as specified by the current chart-type setting of CHSET (MBAR|CBAR|FBAR). The possible choices are: multiple bar chart, composite bar chart, or floating bar chart. The shading is determined by the CHSET shading option.

The bars or bar groups are centered about the x axis values 1, 2, 3, ... If autoranging applies, the x axis range is constructed to run from 0.5 through (count + 0.5).

CHSET(CVALUES|VALUES) can be specified to show the data value at the end of each bar (multiple bar chart), or to annotate the last data group plotted (composite and floating bar charts). If there is no room for the data values, a GDDM message may result.

Valid in state 1 or state 2. If coded in state 1, the program enters state 2.

Parameters

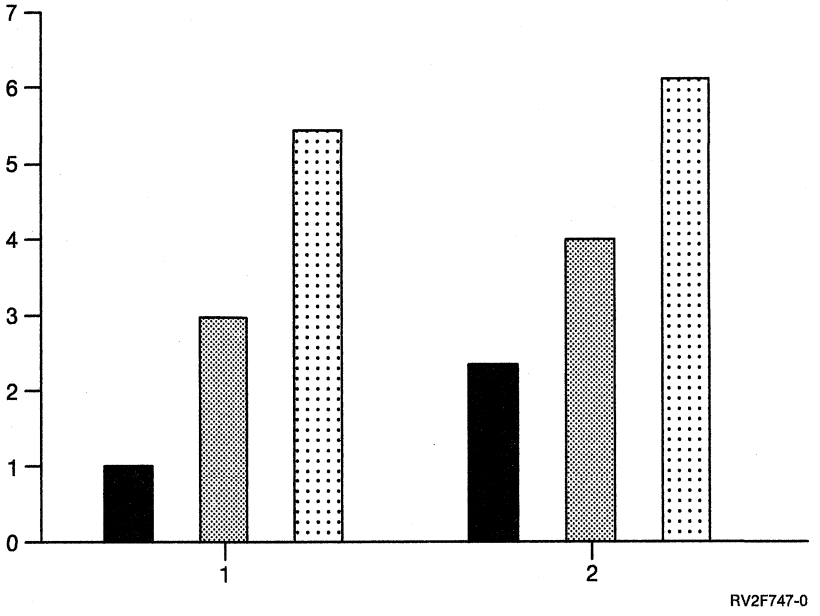
data-groups (4-byte binary integer)
The number of data groups (types of bar) to be created. The bar for each data group has new attributes as indicated by the appropriate chart attribute tables.

count (4-byte binary integer)
The number of points on the x axis at which bars are placed (that is, the number of bars in each data group). All data groups must have the same number of bars.

y-values (array of short floating-point numbers)
The dependent data governing the lengths of the bars. For multiple data groups, the values must be stored sequentially by data group; that is, all values for the first data group, followed by all values for the second data group, and so forth, for as many data groups as specified. No array of independent values is necessary.

Coding Example

This bar chart has three data groups (differently shaded bar types) each with two bars (two positions on the x axis labeled 1 and 2):



The following program displays this bar chart:

```
OPTION BASE 1
! Set type of chart to multiple-bar
CALL GDDM('CHSET','MBAR')
DECIMAL YARRAY
DIM YARRAY(6)
MAT READ YARRAY
DATA 1.0, 2.2, 3.1, 4.0, 5.4, 6.1
CALL GDDM('CHBAR',3,2,YARRAY())
```

The MBAR option of CHSET was in use for this example.

CHBATT – Set Framing Box Attributes

```
CHBATT(count,array)
```

Overrides or resets the default attributes for the chart area background or the chart area frame box, if these are used. The chart area background is produced only when CHSET(CBACK) is used. A chart area framing box is produced only when CHSET(CBOX) is used. By default neither a background nor a box is produced.

The defaults are:

Color	7 is neutral
Line type	0 is solid
Line width	0 is normal

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. If more than three elements are supplied, the excess elements are ignored. If fewer than three are supplied, the remainder are unchanged from their previous setting.

When the count parameter is set to zero, the defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array are the following attributes:

1. Color (valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

   When the device does not support one or more of these colors, a device-dependent alternative is chosen.

2. Line type (valid values have the same effect as the GDDM routine GSLT).

   For valid values of line-type, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

3. Line width (valid values have the same effect as the GDDM routine GSLW). Valid values are:

0	Default (normal)
1	Normal
2	Double

Coding Example

The following example shows how to use CHBATT to set up a framing box with a thick, solid, blue line:

```
OPTION BASE 1
CALL GDDM('CHSET','CBOX')
INTEGER ATT
DIM ATT(3)
MAT READ ATT
DATA 1,0,2
CALL GDDM('CHBATT',3,ATT())
```

CHCGRD – Basic Character Spacing/Size

```
CHCGRD(n,m)
```

Specifies the basic character size of all text constructed by Presentation Graphics routines (for example, axis labels, keys, chart headings, chart notes). The basic character size is multiplied by the character multiplier specified in the text attributes associated with each type of text, in order to obtain the actual text character size.

The default basic character size is the size of the hardware characters of the primary device current at the time of transition from state 1 to state 2, reduced if the chart area does not fill the picture space (see below).

The basic character size is the fundamental measurement in chart layout. Use of CHCGRD can alter the basic proportions of the chart. It is intended to allow charts to be drawn in the same proportions on different types of device.

CHCGRD should be used with care. Normally, the defaults should be used. CHCGRD governs:

- Horizontal and vertical size of mode 3 (vector) characters; that is, the basic size upon which any multipliers that you use in your program work.
- The size of the horizontal and vertical margins, and hence the size of the plotting area.

The basic character size is specified in terms of a character grid. The parameters specify the number of characters (n) to be accommodated across the width, and (m) the number of character rows to be accommodated within the height of the chart area.

The default values correspond to the hardware character size of the primary device current at the time of transition from state 1 to state 2 (for example, when n is 80 and m is 24). If a CHAREA call is issued, this default value is reduced in proportion to the ratio of the chart area dimensions to the picture space dimensions (horizontal and vertical).

The physical width and height of the chart margins are affected when n or m, or both, differ from the default values. For example, if the default value of n is 80, but n is set to 60, the 10-character default width of the vertical margins is physically 4/3 as wide as the default value, and the physical width of the plotting area is correspondingly reduced.

When initially constructing a chart on a display, to get chart legends and notes constructed at the same scale, from one device to another, use a character grid unit based on a 24-by-80 screen size.

However, if you are initially constructing your chart on a printer (did not use display or plotter until chart was done), to get chart legends and notes constructed at the same scale, from one device to another, use a character grid unit based upon the printer page size. For example, if you are initially constructing your chart for the IBM 5225 Printer using 10 cpi, 9 lpi, forms length of 99 and forms width of 132, then specify 99-by-132 for character grid units.

Valid only in state 1.

Parameters

n (4-byte binary integer)
The number of character columns across the width of the chart, which determines the basic character width and the size of the vertical margins. If zero is specified, the default basic character width is used (see above).

m (4-byte binary integer)
The number of character rows in the height of the chart, which determines the basic character height and the size of the horizontal margins. If zero is specified, the default basic character height is used (see above).

Coding Example

The following example shows how to use CHCGRD:

```
INTEGER N,M
LET N = 60      !  Reduce width of the plotting area
LET M = 0       !  Leave height of the plotting area the default
CALL GDDM('CHCGRD',N,M)
```

CHCOL – Component Color Table

CHCOL(count,colors)

Overrides the default table of color attribute values to be used for data (such as lines on line graphs). Colors are taken from this table in turn to construct each data group, pie sector, or Venn diagram circle.

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of attribute values to be taken from the colors parameter (which is an array) and used as the color attribute table. If count is zero, the default table is used. The maximum number of values that you can specify is 32.

colors (array of 4-byte binary integers)
The color codes to be used consecutively for data groups.

For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Example

If you wanted the first two data groups (lines on a line graph for example) to be green and the third red, you would enter:

```
OPTION BASE 1
INTEGER COLARRAY
DIM COLARRAY(3)
MAT READ COLARRAY
DATA 4,4,2
CALL GDDM('CHCOL',3,COLARRAY())
```

In this example, 3 is the number of attribute values, 4 means green, and 2 means red.

Note that the table is used cyclically and in this case the fourth data group (if any) would be green.

CHDATT – Datum Line Attributes

```
CHDATT(count,array)
```

Overrides or resets the default attributes used in drawing translated axis lines (when used in state 1) or datum lines (when used in state 2).

The defaults are:

Color	0 is green on displays, black on printers, highest pen number on plotters
Line type	0 is solid
Line width	0 is normal

Valid in state 1 or state 2.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter (which is an array). If more than three elements are supplied, the excess elements are ignored. If fewer than three are supplied, the remainder are unchanged from their previous setting. If the count parameter is set to zero, all defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array are the following attributes:

1. Color (valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Line type (valid values have the same effect as the GDDM routine GSLT).

   For valid values of line type, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

3. Line width (valid values have the same effect as the GDDM routine GSLW). Valid values are:

0	Default (normal)
1	Normal
2	Double

Coding Example

The following example shows how to use CHDATT to change datum lines to a dotted yellow line with normal thickness:

```
OPTION BASE 1
INTEGER ATT
DIM ATT(3)
MAT READ ATT
DATA 6,1,1
CALL GDDM('CHDATT',3,ATT())
```

CHDRAX – Specific Control of Axis Drawing

CHDRAX

Specifies that the axis lines, scale marks, labels, and associated grid and/or datum reference lines are to be constructed immediately, or suppressed. Does not apply to axis titles. Note that CHSET (NDRAW) should be used with CHDRAX.

Valid only when in state 2, after the currently selected axes have been defined by a previous call to a plotting routine.

Parameters

None

Coding Example

The following example shows how to use CHDRAX:

```
CALL GDDM('CHSET','NDRAW')
CALL GDDM('CHDRAX')
```

CHFINE – Curve Fitting Smoothness

CHFINE(smoothness)

Overrides or resets the default smoothness of the curves used to connect consecutive data points in each data group of a line graph or surface chart.

Note: A parameter value higher than 10 might not add any noticeable degree of smoothness to the curve and will increase the time needed by the system to draw the chart.

CHFINE has no effect unless CHSET(CURVE) is explicitly specified, and CHPLOT or CHSURF is issued. It also has no effect on a CHPLOT call for which CHSET(NOLINES) is specified (that is, a scatter plot).

The *smooth curve* joining consecutive pairs of data points is constructed by interpolating a number of straight lines between each pair of data points. As the number of such interpolated lines increases, the resulting curve appears smoother. On higher resolution devices, a larger number of interpolated lines are required to give a smooth appearance. CHFINE can be used to control the apparent smoothness by specifying the number of interpolated lines. The default number of interpolated lines is 10.

If a nonsolid line type is being used on the display, a smooth curve line may appear different from a straight line, in terms of the line-type pattern.

Valid in state 1 or state 2.

Parameters

smoothness (4-byte binary integer)

The number of straight lines to be constructed between each consecutive pair of data points to form the appearance of a smooth curve. Must be 0 through 1000.

The default value, which applies if no CHFINE call is issued or if CHFINE(0) is issued, is 10.

Coding Example

The following example shows how to use CHFINE:

```
CALL GDDM('CHSET','CURVE')
CALL GDDM('CHFINE',20)
```

CHGAP – Spacing between Bars

CHGAP(gap-to-bar-ratio)

Overrides the default ratio of the gap between adjacent bars to the bar width. For a figure showing the use of CHGAP and CHGGAP, see "CHGGAP – Spacing between Bar Groups" on page 3-25.

The default ratio is half the bar width (0.5).

Valid in state 1 or state 2.

Parameters

gap-to-bar-ratio (short floating-point number)

The ratio of the gap between bars to the bar width. Gap-to-bar-ratio must be between plus and minus one. If zero is specified, the bars have common lines that might appear in unexpected colors because of color mixing.

If gap-to-bar-ratio is a negative number and CHGAP is used before CHBAR, the bars in the chart are positioned so that they partially overlap; that is, a bar is partially hidden behind the previous one. Values close to −1 are not recommended.

The total length of the axis must accommodate all bars and gaps. As this parameter is increased, the actual bar width decreases. It is therefore possible to produce bars that appear as single lines.

Coding Example

The following example shows how to use CHGAP:

```
CALL GDDM('CHGAP',0.7)
```

CHGATT – Grid Line Attributes

```
CHGATT(count,array)
```

Overrides or resets the default attributes used in drawing grid lines.

The defaults are:

Color	0 is green on displays, black on printers, highest pen number on plotters
Line type	0 is solid
Line width	0 is normal

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. If more than 12 elements are specified, the excess elements are ignored. If less than 12 are specified, the remainder are unchanged from their previous setting.

When count is zero, all defaults are reinstated.

array (array of 4-byte binary integers)
The attributes for the grid lines.

The first set of three attributes applies to the primary x axis grid lines, the second set applies to the primary y axis grid lines, and the third and fourth sets apply to the secondary x and y axis grid lines. The three elements of each set are:

1. Color (except for −1, valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Line type (except for −1, valid values have the same effect as the GDDM routine GSLT).

   For valid values of line-type, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

3. Line width (except for −1, valid values have the same effect as the GDDM routine GSLW). Valid values are:

−1	Leave unchanged
0	Default (normal)
1	Normal
2	Double

Coding Example

The following example shows how to use CHGATT to set up the primary and secondary grid lines as green, dotted, normal width lines:

```
OPTION BASE 1
INTEGER ATT
DIM ATT(6)
MAT READ ATT
DATA 4,1,1, 4,1,1
CALL GDDM('CHGATT',6,ATT())
```

CHGGAP – Spacing between Bar Groups

CHGGAP(group-gap-to-bar-ratio)

For *multiple* bar charts with more than one data group, CHGGAP overrides the default ratio of the gap between bar groups to the width of a single bar.

The default ratio is twice the bar width (2.0).

Valid in state 1 or state 2.

Parameters

group-gap-to-bar-ratio (short floating-point number)
The ratio of the gap between bar groups to the width of a single bar. Must be positive or zero.

If this parameter is zero, data groups are drawn with no gap between them.

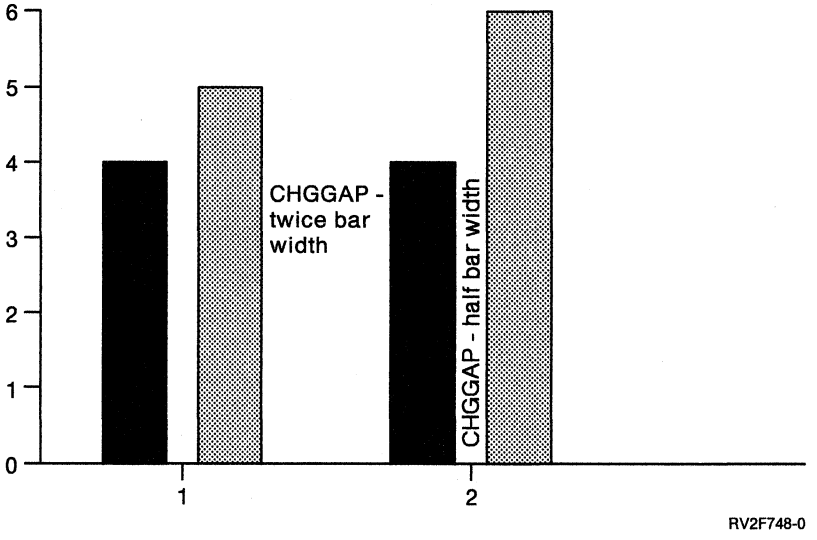
The total length of the axis must accommodate all bars and gaps. Therefore, as this parameter is increased, the actual bar width decreases. It is thus possible to produce bars that appear as single lines.

Coding Example

The following example shows how to use CHGGAP:

```
CALL GDDM('CHGGAP',1.8)
```

In the following chart, the default gap (2.0) is shown:



CHHATT – Heading Text Attributes

```
CHHATT(count,array)
```

Overrides or resets the default attributes to be used in the chart heading.

The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (default characters for the device)
Character size multiplier	100 results in the character size of device unless overridden by CHCGRD

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. If more than four elements are specified, the excess elements are ignored. If fewer than four are specified, the remainder are unchanged from their previous setting.

If count is zero, all four defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array set the following attributes:

1. Color (valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Character mode.

   Valid values are:

   0 System default (mode 3)
   2 Mode 2
   3 Mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier can be:

   0 Default character set
   n A graphics symbol set that has been previously loaded using the GDDM routine GSLSS

4. Character size multiplier.

   For mode 2 or mode 3, the character size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. For example, if character size multiplier is 200 and mode-3 characters are being used, they are drawn twice as large as the basic size, which is that of the hardware characters unless overridden by CHCGRD. The character size multiplier must have a value greater than zero.

Coding Example

The following example shows how to use CHHATT:

```
OPTION BASE 1
INTEGER ARRAY
DIM ARRAY(4)
MAT READ ARRAY
DATA 7,3,65,150
CALL GDDM('CHHATT',4,ARRAY())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character-mode 1 is not supported.

CHHEAD – Heading Text

```
CHHEAD(length,text)
```

Specifies the chart heading text. CHSET options are available to control its occurrence, position, and justification. Its attributes are controlled by CHHATT.

Valid only in state 1.

Parameters

length (4-byte binary integer)
The length in bytes of the EBCDIC character string supplied. The heading length cannot exceed 132 characters. Specifying the length as zero suppresses the heading.

text (character string)
The chart heading text. The number of characters in the text parameter must be the value of the length parameter.

The character string can be formatted as a number of separate physical lines of heading on the chart by the use of the semicolon character (;) as a control character, indicating *new line*. The character string is scanned from left to right, and whenever a semicolon control character is found a new line of heading is started. The control character itself is not displayed. The lines of heading are constructed in turn, each line immediately under the preceding line. Each line is subject to separate justification.

When a semicolon character is required in the heading, two consecutive semicolons should be supplied. In this case no new-line action is taken, and the two consecutive semicolons cause a single semicolon character to appear in the heading.

Coding Examples

Example 1:

In the following example, the heading of a chart is specified to be 'Sample heading':

```
CALL GDDM('CHHEAD',14,'Sample heading')
```

Length (14) is the length of the character string 'Sample heading'.

Example 2:

In the following example, a two-line heading is specified:

```
CALL GDDM('CHHEAD',35,'Line 1 of Heading;Line 2 of Heading')
```

It would appear as follows:

```
Line 1 of Heading
Line 2 of Heading
```

CHHIST – Plot a Histogram

```
CHHIST(data-groups,count,range1,range2,y-values)
```

Plots one or more histograms on the currently selected axes.

The histogram is drawn as a set of bars perpendicular to the x axis. Each bar extends over a range defined by a start value and an end value. The ends of the bars can be connected to the x axis (or datum reference line) by *risers* parallel to the y axis (see the CHSET option RISERS). The shading method is determined by the CHSET shading option. A histogram with more than one data group is plotted in a manner similar to a *composite* bar chart.

The color of the bars is controlled by the color table, which can be specified by a call to CHCOL. The shading patterns are controlled by the shading pattern table, which can be specified by a call to CHPAT.

Valid in state 1 or state 2. If coded in state 1, the program enters state 2.

Parameters

data-groups (4-byte binary integer)
The number of sets of histogram bars to be drawn on the current axes.

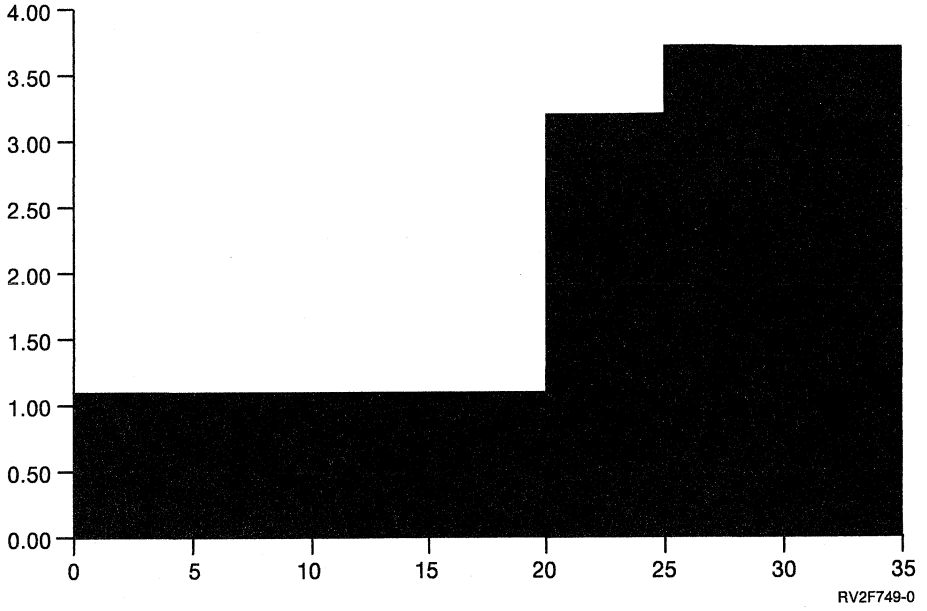
count (4-byte binary integer)
The number of histogram bars for each data group.

range1 (array of short floating-point numbers)
Each element in this array gives the start of a range.

range2 (array of short floating-point numbers)
Each element in this array gives the end of the range whose start is given by the corresponding element of range1. In general, the successive ranges are not all equal. If range2 contains the same values as range1 (or is the same array) each bar is only as wide as a line.

y-values (array of short floating-point numbers)
The dependent values (y-values) used in drawing the histogram. For multiple data groups, the values must be stored sequentially by data group; that is, all values for the first data group, followed by all values for the second data group, and so forth, for as many data groups as specified.

Coding Example



The following program displays this histogram:

```
OPTION BASE 1
DECIMAL RANGE1,RANGE2,YVALUES
DIM RANGE1(3), RANGE2(3), YVALUES(3)
MAT READ RANGE1
DATA 0,20,25
MAT READ RANGE2
DATA 20,25,35
MAT READ YVALUES
DATA 1.1,3.2,3.7
CALL GDDM('CHHIST',1,3,RANGE1(),RANGE2(),YVALUES())
```

CHHMAR – Bottom and Top Margins

CHHMAR(bottom-margin,top-margin)

Overrides the default margin sizes of the chart, and hence changes the size of the plotting area also. The margin sizes are specified in terms of character grid rows and character grid columns, where the character grid is that of the device, unless explicitly overridden by a call to CHCGRD.

By default, the horizontal margins are 5 character rows above and below the plotting area, and the vertical margins are 10 character columns on each side of the plotting area.

For the left and right margins, see CHVMAR.

CHHMAR and CHVMAR can enlarge the margins to prevent the legend encroaching on the chart plotting area, or reduce the margins to increase the size of the chart plotting area.

Valid only in state 1.

Parameters

bottom-margin,top-margin (4-byte binary integers)
The number of character rows at the bottom and top respectively of the plotting area.

Coding Examples

In the following example, CHHMAR sets the bottom margin at seven lines above the bottom of the plotting area and sets the top margin at five lines below the top of the plotting area (which is the default).

```
CALL GDDM('CHHMAR',7,5)
```

CHKATT – Legend Text Attributes

```
CHKATT(count,array)
```

Overrides or resets the default attributes to be used for legend key labels, spider labels on pie charts, and Venn diagram labels. For a description of how CHKATT applies to pie chart spider labels, see CHSET(SPISECTOR).

The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (default characters for the device)
Character size multiplier	100 (text size is defined by CHCGRD; by default the size of hardware characters)

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. If more than four elements are specified, the excess elements are ignored. If fewer than four are specified, the remainder are unchanged from their previous setting.

If count is zero, all four defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array set the following attributes:

1. Color (valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

   If the device does not support one or more of these colors, a device-dependent alternative is chosen.

2. Character mode.

   Valid values are:

   0 System default (mode 3)
   2 Mode 2
   3 Mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier can be:

   0 Default character set
   n A graphics symbol set that has been previously loaded using GSLSS

4. Character size multiplier.

   For mode 2 or mode 3, the character size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. For example, if character size multiplier is 200 and mode 3 characters are being used, they are drawn twice as large as the basic size set by CHCGRD. The character size multiplier must have a value greater than zero.

Coding Examples

In the following example, CHKATT sets the character attributes of labels as blue, mode 3 characters from symbol set 65 as loaded by GSLSS, drawn one and a half times as large as the basic character size specified by CHCGRD.

```
OPTION BASE 1
INTEGER ATTARR
DIM ATTAR(4)
MAT READ ATTARR
DATA 1,3,65,125
CALL GDDM('CHKATT',4,ATTARR())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

CHKEY – Legend Key Labels

```
CHKEY(count,length,labels)
```

Specifies legend key labels for data groups (on a line graph, surface chart, bar chart, histogram) or for pie chart sectors. Also has other interpretations for pie charts and Venn diagrams, as follows:

- For pie charts, the labels specified by CHKEY appear as sector labels around each pie if CHSET(SPIDER) is specified; they appear instead as key labels in a legend only if CHSET(PIEKEY) is specified.
- For Venn diagrams, the labels specified by CHKEY appear as labels connected to the Venn circles and their overlap. Only the first three labels are used. No legend can be specified for a Venn diagram.

In all cases (legend key labels, pie sector labels, and Venn diagram labels), the labels are constructed with the attributes specified by CHKATT, or with the CHKATT default attribute values.

For keys in a legend, on each plotting routine call, a key is created for each data group or pie sector plotted using the supplied labels in sequence. Thus, if more labels are provided than are needed for the number of data groups or sectors plotted, the remaining labels are ignored. If fewer labels are provided than are needed, the number of keys constructed is equal to the number of labels supplied. In all cases, legend base position justification (see CHKEYP) is based on the assumption that all labels provided are generated. Thus, if all labels are not used, the legend might not be where you expect it.

If no CHKEY call is issued, then no legend is constructed. Even if a CHKEY call is issued, a legend is constructed only if CHSET(LEGEND) applies (and, in the case of pie charts, only if CHSET(PIEKEY) applies also).

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of labels supplied in the labels parameter. Count is 0 can be useful for deleting labels defined by a previous CHKEY call.

length (4-byte binary integer)
The number of characters in each label supplied in the labels parameter. All labels supplied have the same length. The effect of varying length labels can be produced by padding each label with null (X' 00') characters. All such trailing nulls are ignored when the legend is constructed.

If a label consists entirely of null characters, the associated legend key label or Venn diagram label is not constructed. For pie sector labels, the sector label and associated spider line connecting it to the pie sector are not constructed (unless CHSET(VALUES) is specified, when the value and associated connecting line are constructed).

labels (character string)
A list of labels each as long as the value of the length parameter. The total length, L, of the text string that is seen is given, in bytes, by:

```
L = count*length
```

The maximum value of L permitted is approximately 32,000.

Semicolons (;) and underscores (_) in each label are interpreted specially when the labels appear in legends. A semicolon causes a line break: a new line in the legend is started, left-justified. An underscore causes the key symbol to be drawn at that point (taking up 3 key label character positions).

Specifying ;; causes a semicolon to appear in the label, without a line break. Similarly, specifying __ (two underscores) causes an underscore to appear in the label, without substituting a key symbol.

If a legend key label contains *no* underscore control characters, a key symbol followed by a blank character (the whole occupying 4 character positions) is constructed at the start of the first line of the key label. Any further lines of the key label are indented 4 character positions to align with the text of the first line. If a legend key label contains one or more underscore control characters, key symbols (each occupying three character positions) are constructed where underscore control characters appear. No indenting of lines occurs.

In sector labels on pie charts, single semicolon characters are interpreted as line break control characters as described above, and single underscore characters are interpreted as blanks. In Venn diagram labels, both semicolon and underscore single characters are interpreted as blanks. In both pie sector labels and Venn diagram labels, ;; appears as ; and __ (two underscores) appear as _ when drawn.

The following character codes also have special interpretations:

- X'15' is interpreted as a line break control character.
- X'1A' is interpreted as a key symbol substitution control character.

In BASIC, you cannot directly specify how to set bits on and off but you can use the HEX$ built-in function. In RPG III, you could use the BITON and BITOFF operation codes; in PL/I, you could use BIT data; in COBOL, there is no direct way of manipulating bit strings; you might want to call a program in another language, passing the other program a character string to be initialized and returned with the correct bit values. For an example of coding X'15' in a BASIC program, see the GDDM call GSCHAP (where X'15' also acts as a line break control character).

Coding Examples

The following example shows how to specify CHKEY:

```
CALL GDDM('CHKEY',2,19,'This Label is Long;Short Label      ')
```

CHKEYP – Legend Base Position

```
CHKEYP(orientation,margin,justification)
```

Overrides the default base position of the legend. The legend offsets (0 by default, see CHKOFF) are applied to the base position to obtain the legend final position. The legend base position also affects many aspects of the legend format. The default base position, which applies when CHKEYP is not called, specifies a vertical legend centered in the right margin.

Valid only in state 1.

Parameters

orientation (character string)

A character string of length 1 containing V or H.

V specifies a vertical legend (base position in the left or right margin, adjacent to the left or right chart area boundary).

H specifies a horizontal legend (base position in the top or bottom margin, adjacent to the top or bottom chart area boundary).

margin (character string)

A character string of length 1 containing R, L, T, or B.

R and L specify a base position in the right and left margins respectively and are valid only for vertical legends.

T and B specify a base position in the top and bottom margins respectively and are valid only for horizontal legends.

justification (character string)

A character string of length 1 containing C, R, L, T, or B.

C specifies a base position centered in the margin and is valid for both horizontal and vertical legends.

R and L specify right and left justification, and are valid only for horizontal legends. For example, L specifies that the legend is justified to the left of the top or bottom margin.

T and B specify top and bottom justification, and are valid only for vertical legends. For example, T specifies that the legend is justified to the top of the left or right margin.

Coding Examples

The following example shows how to specify CHKEYP:

```
CALL GDDM('CHKEYP','H','B','C')
```

CHKMAX – Maximum Legend Width/Height

```
CHKMAX(hmax,vmax)
```

Specifies the maximum width of a horizontal legend and the maximum height of a vertical legend.

Both parameters are 4-byte binary integers. Their values must not be negative. The units are the character width and height, which default to the hardware character size unless the defaults are explicitly overridden by CHCGRD.

Zero specifies that the legend height or width is limited by the height or width of the chart area.

Valid only in state 1.

Parameters

hmax (4-byte binary integer)

The maximum width of a horizontal legend. As many key entries are constructed in each row of the legend as is possible, given the maximum width specified (except that the first key entry in each row is always constructed, even if it is wider than the maximum width specified).

When the maximum width would be exceeded by a new entry, a new line is started.

A zero value specifies that the full chart area width is to be used.

If the value specified results in a maximum legend width greater than the chart area width, the chart area width is used as the maximum.

Each line of a key entry begins and ends with a blank character (for spacing). Therefore the width of a key entry is:

2 plus the number of character positions in longest line of key entry

The maximum legend width specified applies to the actual width including these 2 extra spacing characters for each key entry.

vmax (4-byte binary integer)

The maximum height of a vertical legend. As many key entries are constructed in each column of the legend as is possible given the maximum height specified (except that the first key entry in each column is always constructed, even if it is taller than the maximum height specified). When the maximum height would be exceeded by a new entry, a new column is started.

A zero value specifies that the full chart area height is to be used.

If the value specified results in a maximum legend height greater than the chart area height, the chart area height is used as the maximum.

The top line of text in a key entry begins a half-character height below the top of the space occupied by the key entry. Similarly there is a half-character height between the bottom line of a key entry and the bottom of the space occupied by the key entry. Therefore the height of a key entry is:

1 plus the number of lines in key entry

The maximum legend height specified applies to the actual height including this extra spacing character for each key entry.

Coding Examples

The following example shows how to specify CHKMAX:

```
! When the label is as follows:
CALL GDDM('CHKEY',2,18,'This Label is Long;Short Label      ')
! You can specify CHKMAX as follows so that one label appears
! on a line:
CALL GDDM('CHKMAX',18,4)
```

CHKOFF – Legend Offsets

CHKOFF(horizontal-offset,vertical-offset)

Specifies the horizontal and vertical offsets, interpreted in character grid units, which are used to obtain the final legend position from the legend base position. Character grid units are the character size of the device unless explicitly overridden by a call to CHCGRD.

If no CHKOFF call is issued, no offsets are applied.

When the legend is constructed, it is formatted in the legend base position, then translated across the chart area by the specified horizontal and vertical offsets. If, as a result, any part of the legend lies outside the chart area, the legend is further translated horizontally and/or vertically as necessary until it lies just within the nearest chart area boundaries.

Valid only in state 1.

Parameters

horizontal-offset (short floating-point number)
The horizontal offset, in character grid units (which default to the character size of the device unless explicitly overridden by a call to CHCGRD) to be applied to the legend base position. A positive value specifies translation to the right, and a negative value translation to the left.

vertical-offset (short floating-point number)
The vertical offset, in character grid units (which default to the character size of the device unless explicitly overridden by a call to CHCGRD) to be applied to the legend base position. A positive value specifies upward translation, and a negative value downward translation.

Coding Examples

The following example shows how to specify CHKOFF:

```
! When the label position is in the right margin, for example:
CALL GDDM('CHKEYP','V','R','T')
! You can choose to offset the legend closer to the bottom:
CALL GDDM('CHKOFF',0,2)
```

CHLATT – Axis Label Text Attributes

CHLATT(count,array)

Overrides or reestablishes the default character-appearance attributes used in displaying *all* axis labels, and pie titles (on pie charts) on all the axes. You can use the CHXLAT, and CHYLAT calls to change the axis label attributes for the currently selected x, or y axis.

By default, the attributes are as follows:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (system default)
Character-size multiplier	100 (results in the text size as defined by CHCGRD (by default, the size of the hardware characters)
Character height/width multiplier	100 (results in the text size as defined by CHCGRD (by default, the size of the hardware characters) unless overridden by CHXLAT)
Label rotation	0

Valid only in state 1.

Parameters

count (4-byte binary integer)
Specifies the number of elements in **array**. If more than six elements are specified, the excess elements are ignored. If fewer than six, the remainder are unchanged from their previous setting. If zero is specified, all six defaults are reinstated.

array (an array of 4-byte binary integers)
An array of, at most, six 4-byte binary integers:

1. Color.

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Character mode.

   Valid values are:

   0 system default (mode 3)
   2 mode 2
   3 mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier denotes:

   0 the default character set
   n a graphics symbol set that has been previously loaded into GDDM by the application program using the GDDM call GSLSS

4. Character-size multiplier.

   For modes 2 and 3, the character-size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. For example, if character-size multiplier is 200 and mode 3 characters are being used, they are drawn twice as large as the basic size set by CHCGRD. The character-size multiplier must have a value greater than zero.

5. Character height/width multiplier.

   Specifies the height of the character box relative to its width. For example, 100 indicates that the height is multiplied by the same amount as the width. 200 causes the height to be multiplied by twice the amount of the width.

6. Axis-label rotation.

   When axis label rotation is requested, each label is written at the specified angle to the horizontal. Label rotation is indicated by specifying a nonzero value in this element. The value must be between −9000 and +9000. This number is divided by 100 to give a value between −90 and +90, which is the angle, in degrees, to the horizontal at which the labels will be drawn. The default is zero. When positive, the angle is in a counterclockwise direction. When negative, it is in a clockwise direction.

Coding Examples

In the following example, CHLATT sets the character attributes of axis labels as blue, mode 3 characters from symbol set 65 as loaded by GSLSS, drawn one and a half times as large as the basic character size specified by CHCGRD.

```
OPTION BASE 1
INTEGER ATTARR
DIM ATTAR(4)
MAT READ ATTAR
DATA 1,3,65,150
CALL GDDM('CHLATT',4,ATTARR())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

CHLT – Component Line Type Table

CHLT(count,line-types)

Overrides or resets the default table of line type attributes. Line type attributes are taken in sequence from the table and used for each data group, pie sector, or Venn diagram circle.

The default table has only one entry:

Index	Line type
1	1 (solid)

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of attribute values to be taken from the line-types parameter and used as the line type attribute table. If count is zero, the default table is used. A maximum of 32 values can be specified.

line-types (array of 4-byte binary integers)
The line types to be used consecutively for data groups.

For valid values of line types, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Examples

In the following example, CHLT sets up a table of three line types: dot, short-dash, and dash-dot.

```
OPTION BASE 1
INTEGER LINTYPS
DIM LINTYPS(3)
MAT READ LINTYPS
DATA 1,2,3
CALL GDDM('CHLT',3,LINTYPS())
```

CHLW – Component Line Width Table

CHLW(count,line-widths)

Overrides or resets the default table of line width attributes. Line width attributes are taken in turn from the table and used for each data group, pie sector, or Venn diagram circle.

The default table has only one entry:

Index	Line width
1	1.0

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of attribute values to be taken from the line-widths parameter (which is an array) and used as the line width attribute table. If count is zero, the default table is used. A maximum of 32 values can be specified.

line-widths (array of short floating-point numbers)
The line widths to be used consecutively for data groups. Valid values are 0 through 100.

The specified value is a multiplication factor to be applied to the standard line width, shown below. The standard width is multiplied by the specified factor, and then rounded down to an integral number of pixels. If the result is less than one pixel, a width of one pixel is used. If the result is more than the maximum shown below, the maximum is used.

Device	Minimum (Value is 0)	Standard (Value is 1)	Maximum (Value is 100)
5292 Model 2	1	1	2
IBM PCs	1	1	1
IBM plotters	1	1	1
Printers	1	1	2

Coding Examples

In the following example, CHLW sets up a table of three line widths: the standard width, one and a half times the standard width, and twice the standard width.

```
OPTION BASE 1
DECIMAL LINWIDS
DIM LINWIDS(3)
MAT READ LINWIDS
DATA 1.0,1.5,2.0
CALL GDDM('CHLW',3,LINWIDS())
```

CHMARK – Component Marker Table

```
CHMARK(count,markers)
```

Overrides or resets the default table of marker values. Markers from this table are used in sequence to construct each data group.

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of attribute values to be taken from the markers parameter (which is an array) and used as the marker attribute table. If count is zero, the default table is used. A maximum of 32 values can be specified.

markers (array of 4-byte binary integers)
The marker values to be used consecutively for data groups. The default is 0 (a cross; same as marker symbol 1). For valid values of markers, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Examples

In the following example, CHMARK sets up a table of three markers to be used consecutively for data groups: cross, diamond, and square.

```
OPTION BASE 1
INTEGER MARKERS
DIM MARKERS(3)
MAT READ MARKERS
DATA 1,3,4
CALL GDDM('CHMARK',3,MARKERS())
```

CHNATT – Specify Attributes for Notes

CHNATT(count,array)

Overrides or reestablishes the default character attributes used for notes.

The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (system default)
Character-size multiplier	100 (text size as defined by CHCGRD (by default, the size of the hardware characters))
Character height/width multiplier	100 (text size as defined by CHCGRD (by default, the size of the hardware characters))
Note rotation	0

Valid in state 1 or state 2.

Parameters

count (4-byte binary integer)
A 4-byte binary integer specifying the number of elements in **array**. If more than six elements are specified, the excess elements are ignored. If fewer than six, the remainder are unchanged from their previous setting. If zero is specified, all six defaults are reinstated.

array (an array of 4-byte binary integers)
An array of, at most, six 4-byte binary integers:

1. Color.

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Character mode.

   Valid values are:

   0 system default (mode 3)
   2 mode 2
   3 mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier denotes:

   0 the default character set
   n a graphics symbol set that has been previously loaded into GDDM by the application program using the GDDM call GSLSS

4. Character-size multiplier.

   For modes 2 and 3, the character-size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. (CHCGRD defaults are the size of characters for the device.) For example, if character-size multiplier = 200 and mode 3 characters are being used, they are drawn twice as large as the hardware characters unless CHCGRD has been used to override the default. The character-size multiplier must have a value greater than zero.

5. Character height/width multiplier.

   Specifies the height of the character box relative to its width. For example, 100 indicates that the height is multiplied by the same amount as the width. 200 causes the height to be multiplied by twice the amount of the width.

6. Note rotation.

   Note rotation is indicated by specifying a nonzero value in this element. When note rotation is requested, each note is written at the specified angle to the horizontal. The value must be between −36000 and +36000. This number is divided by 100 to give a value between −360 and +360, which is the angle, in degrees, to the horizontal at which the notes are drawn. The default is zero. When positive, the angle is in a counterclockwise direction. When negative, it is in a clockwise direction.

   The rotation occurs when the note is drawn (that is, when CHNOTE is called). The rotation is performed so that the part of the imaginary rectangle surrounding the note (as specified in CHNOTE) remains at the offset specified in CHNOFF.

   When mode-2 characters are used, the area that is blanked (if blanking is specified) or the surrounding box (if boxing is specified) may not surround the note but might cut through some of the characters. This is because of the inherent characteristics of mode-2 text.

   Note rotation is not supported for class 2 position codes (see CHNOTE).

Coding Examples

In the following example, CHNATT sets the character attributes of notes as blue, mode 3 characters from symbol set 65 as loaded by GSLSS, drawn one and a half times as large as the basic character size specified by CHCGRD.

```
OPTION BASE 1
INTEGER ATTARR
DIM ATTAR(4)
MAT READ ATTAR
DATA 1,3,65,150
CALL GDDM('CHNATT',4,ATTARR())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

CHNOFF – Specify Offsets for CHNOTE

```
CHNOFF(horizontal-offset,vertical-offset)
```

Specifies the offsets to be applied to the base position of the note to obtain the note position (see CHNOTE). Applies to all subsequent calls to CHNOTE (until superseded by another CHNOFF call). If CHNOFF is not called, no offset is applied to the note.

Valid in state 1 or state 2.

Parameters

horizontal-offset (signed, short floating-point number)
The horizontal offset to be applied to the note. Depending on the CHNOTE position code, the offset can be interpreted in two ways:

- As an offset in character grid units. A positive number specifies movement to the right, and a negative number movement to the left.

  The size of character grid units defaults to the character size of the device, but can be altered by a CHCGRD call.

- As an offset in scale units of the currently selected horizontal axis. The (positive or negative) number specifies movement from the base position (the axis origin (0)) to the axis position given by the number. (The offset must be positive for a logarithmic axis.)

The default value is zero.

vertical-offset (signed, short floating-point number)
The vertical offset to be applied to the note base position. Depending on the CHNOTE position code, the offset can be interpreted in two ways:

- As an offset in character grid units. A positive number specifies upward movement, and a negative number downward movement. The size of character grid units defaults to the character size of the device, but can be altered by a CHCGRD call.
- As an offset in scale units of the currently selected vertical axis. The (positive or negative) number specifies movement from the base position (the axis origin (0)) to the axis position given by the number. (The offset must be positive for a logarithmic axis.)

The default value is zero.

Coding Examples

In the following example, CHNOFF offsets notes by 2 character-grid positions to the right:

```
CALL GDDM('CHNOFF',2.0,0.0)
```

CHNOTE – Construct a Character String at a Designated Position

```
CHNOTE(position-code,length,string)
```

This call is valid only in state 2. It constructs a character string (a *note*) at a designated location on the chart.

The position of the note is adjusted by the offsets specified by the most recent CHNOFF call. The note text attributes are those specified by the most recent CHNATT call. If no CHNOFF or CHNATT call has been issued, defaults apply for offsets and text attributes, as described under CHNOFF and CHNATT.

The final position of the note on the chart is determined from the position code in the following way. The position code specifies a base position within the chart area. Horizontal and vertical note offsets are then added algebraically to the base position to obtain the note position. Finally, depending on the position code, the note can be relocated relative to the note position to obtain its final position, where it is constructed.

Position codes fall into two mutually exclusive classes. Class 1 codes support multiple-line notes, rectangular boxing around the note, note rotation, and relocation relative to the note position. Class 2 supports none of these, and is provided mainly for compatibility with early releases of System/370 GDDM-PGF.

The CHSET call option BNOT can be used with position codes of both classes to blank the rectangular area occupied by the characters of the note, before note construction.

Some examples of the use of CHNOTE are:

- To annotate individual plotted points, lines, or areas on a chart.
- To provide a subheading in a different font from the main heading.
- To provide a footnote on a chart.
- To extend the basic axis annotation functions (labels and titles) where required. For example, multiple-line axis titles or labels can be constructed as notes.

Parameters

position-code (character string)

A 2-character code indicating the manner in which the final position of the note is determined in terms of a base position, note position, and relocation relative to the note position. As stated above, position codes are divided into two classes:

- Class 1 position codes

  This class contains the following 36 position codes:

  ```
  C1, C2, C3, C4, C5, C6, C7, C8, C9
  H1, H2, H3, H4, H5, H6, H7, H8, H9
  V1, V2, V3, V4, V5, V6, V7, V8, V9
  Z1, Z2, Z3, Z4, Z5, Z6, Z7, Z8, Z9
  ```

  The first character of each code is interpreted as follows:

  C The base position is the bottom left corner of the chart area. Both horizontal and vertical CHNOFF note offsets are interpreted in character grid units from the base position to obtain the note position.

  H The horizontal CHNOFF note offset is interpreted in units of the scale of the currently selected horizontal axis from the base position given by the origin (0) on that axis (even when the origin is outside the range of that axis). The vertical CHNOFF note offset is interpreted in character grid units from the base position given by the bottom of the chart area.

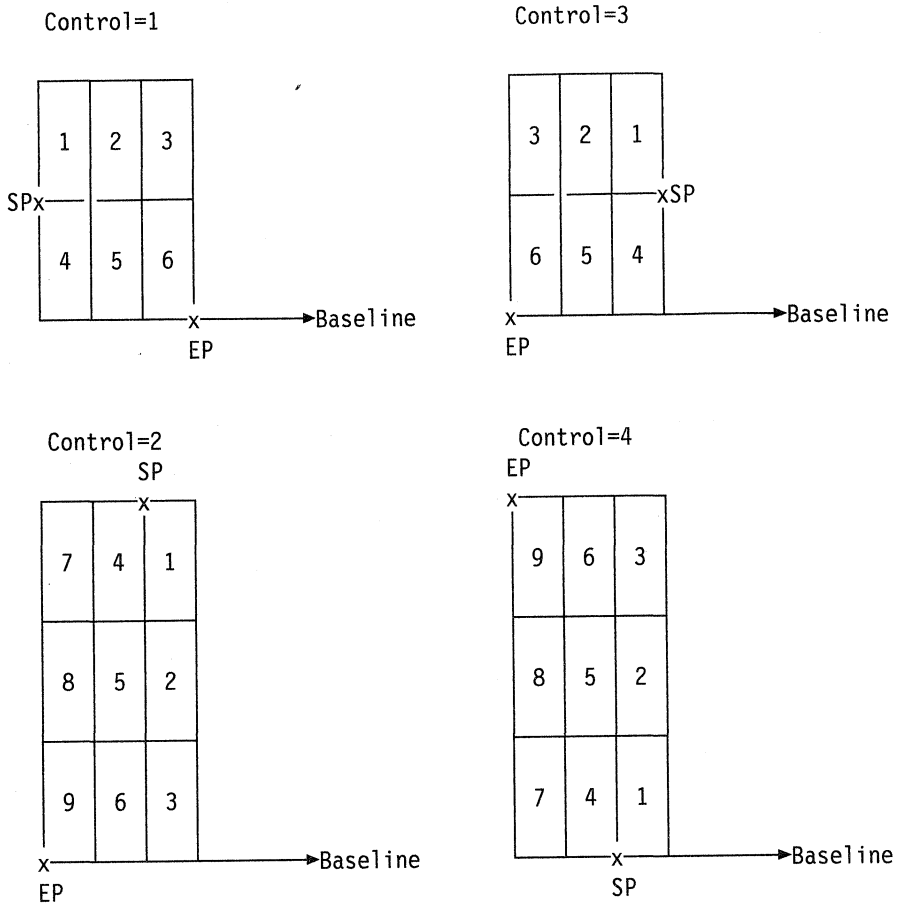
  V The vertical CHNOFF note offset is interpreted in units of the scale of the currently selected vertical axis from the base position given by the origin (0) on that axis (even when the origin is outside the range of that axis). The horizontal CHNOFF note offset is interpreted in character grid units from the base position given by the left of the chart area.

  Z The horizontal CHNOFF note offset is interpreted as for H above, and the vertical CHNOFF note offset is interpreted as for V above.

  The second character of each class 1 position code specifies the relocation that takes place from the note position to obtain the final position of the note. It is a number in the range 1 through 9, which might be thought of as a compass key defining the point of the note that is to be fixed at the note position. The numbers correspond to the corners and mid-points of an imaginary rectangle just surrounding all the characters of the note (the *note rectangle*), as follows:

  ```
  1   2   3
  4   5   6
  7   8   9
  ```

  For example, number 1 relocates the note to place its top left corner at the note position, number 7 places the bottom left corner of the note at the note position, number 6 places the mid-point of the right side of the note at the note position, and so on.



SP = start point
EP = end point

Position codes beginning with H, V, or Z are only valid for notes on charts with axes (that is, for all chart types except pie charts and Venn diagrams). Further, for these position codes, any axis used for interpretation of note offsets must be defined when the CHNOTE call is issued. Finally, for these position codes, if an axis offset determines a note position outside the range of the corresponding axis, the note is not constructed (no diagnostic is issued in this case).

For position codes beginning with H, V, or Z, and with the second compass key character being any value except 5, the final position of the note is adjusted by a slight amount in the direction of each axis used for interpretation of note offsets. This should provide acceptable positioning in typical cases where the note is used to annotate a particular point on a chart (such as a marker), using as axis offsets the x or y values of the marker.

For all class 1 position codes, semicolon (;) characters within the note text are interpreted as line break control characters. This facility allows multiple-line notes to be constructed easily. For example the following note text:

```
This;note; is a;multiple-line;note.
```

is constructed so as to appear on the chart as:

```
This
note
 is a
multiple-line
note.
```

When a semicolon is needed in the text on the chart, two contiguous semicolon characters must be supplied in the note text. These do not cause line breaks, and each such contiguous pair found during a start-to-end scan of the note text is replaced by a single semicolon character in the constructed note.

For all class 1 position codes, when the CHSET call option NBOX is specified, the note is surrounded by a rectangular framing box, constructed of lines with default line type (solid), default line thickness (normal thickness), and line color the same as the color of the note text (specified by CHNATT).

For all class 1 position codes, any trailing nulls in the note text string are ignored, and do not contribute to the constructed note or its position.

If any part of the note rectangle (including any framing box) lies outside the chart area, the entire note rectangle is moved horizontally and/or vertically until it lies within the chart area. An error diagnostic is produced only if the note rectangle, after this translation, is too wide or too high to be contained entirely within the chart area.

- Class 2 position codes

  This class contains nine position codes. Each code is associated with a base position as follows:

  TL The top left point of the string is at the top left point of the chart area.

  TC The top center point of the string is at the top center point of the chart area.

  TR The top right point of the string is at the top right point of the chart area.

  BL The bottom left point of the string is at the bottom left point of the chart area.

  BC The bottom center point of the string is at the bottom center point of the chart area.

  BR The bottom right point of the string is at the bottom right point of the chart area.

  DL The bottom left point of the string is at the left end of a horizontal datum line (specified using CHYDTM), or at the bottom end of a vertical datum line (specified using CHXDTM).

  DC The bottom center point of the string is at the mid-point of a horizontal datum line (specified using CHYDTM), or the left center point of the string is at the mid-point of a vertical datum line (specified using CHXDTM).

  DR The bottom right point of the string is at the right end of a horizontal datum line (specified using CHYDTM), or the top left point of the string is at the top of a vertical datum line (specified using CHXDTM).

Position codes beginning with D refer to datum lines or translated axis lines currently selected when CHNOTE is called. The particular line to be annotated is determined as follows:

- If calls have been made to CHXDTM or CHYDTM in state 2 before CHNOTE is called, the most recent of such calls is used.
- If there have been no calls to CHXDTM or CHYDTM in state 2 preceding CHNOTE, the most recent state 1 call to CHYDTM (if any) is used. If CHYDTM has not been called, an error message is issued. Any call to CHXDTM in state 1 cannot be implicitly referred to in this manner.

For all class 2 position codes, the note position is obtained from the base position by applying CHNOFF note offsets, interpreted in character grid units.

For class 2 position codes, semicolons are never interpreted as line break control characters, and a note framing box is never constructed.

If the note position is not related to a datum line or translated axis line, and the position after adjusting for offsets lies outside the chart area, the note is drawn justified to the chart area boundary. No error or warning message is issued unless the note is too large to fit into the chart area.

length (4-byte binary integer)
The number of characters in the string parameter.

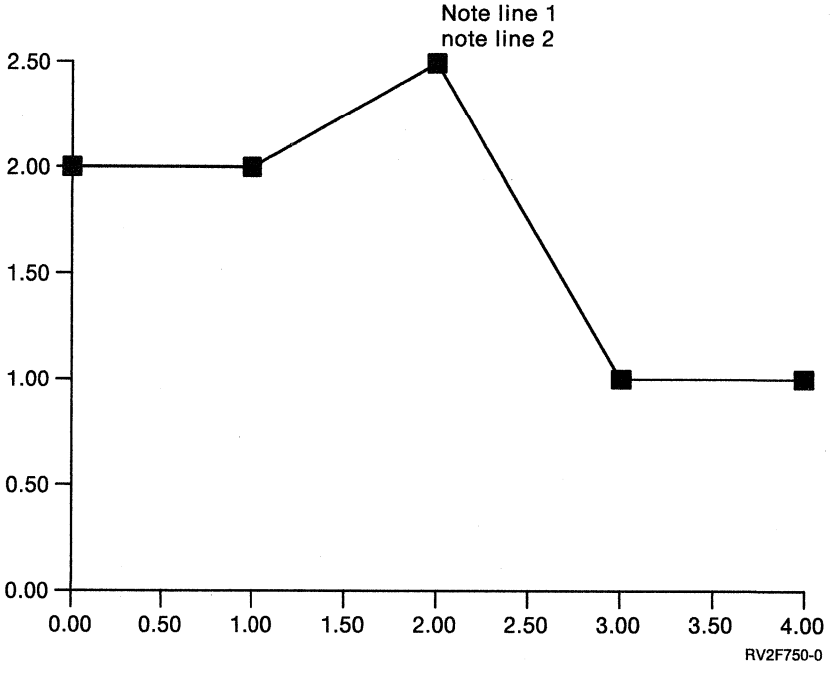
string (character string)
The character string to be constructed on the chart. (Depending on the position code, semicolon characters in the string might be interpreted as line break control characters.)

Coding Examples

Specifying CHNOTE as follows causes the note to appear as shown in the sample shown below.

```
DECIMAL XVALUES,YVALUES
DIM XVALUES(5)
MAT READ XVALUES
DATA 0.0, 1.0, 2.0, 3.0, 4.0
DIM YVALUES(5)
MAT READ YVALUES
DATA 2.0, 2.0, 2.5, 1.0, 1.0
CALL GDDM('CHPLOT',1,5,XVALUES(),YVALUES())
CALL GDDM('CHNOFF',2.0,2.5)                    ! Third point on graph
CALL GDDM('CHNOTE','Z7',23,'Note line 1;note line 2')
```

The note in this example appears as:



CHNUM – Set Number of Components

CHNUM(n)

Sets the total number of pie charts or multiple bar charts to be constructed by subsequent calls to CHPIE and CHBAR. Controls spacing and size of the pies (and bars) so that the complete chart fits within the plotting area. Should be used for multiple pie charts, and multiple bar charts, whenever they are to be constructed by more than one plotting call.

Valid only in state 1.

Parameters

n (4-byte binary integer)
The total number of charts to be drawn. If n is set to zero, or if no call is made, the number of pies is determined from the first subsequent call to CHPIE.

Coding Examples

In the following example, three charts are to be drawn:

```
CALL GDDM('CHNUM',3)
```

CHPAT – Component Shading Pattern Table

CHPAT(count,patterns)

Overrides or resets the default table of shading pattern values to be used. Patterns from the table are used in turn to construct each data group, pie sector, or Venn diagram circle.

If CHPAT is not specified, the first eight of the 16 patterns shown in the default pattern table are used in drawing shaded components. If CHPAT is specified, all 16 patterns are used in drawing shaded components.

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of attribute values to be taken from the patterns array and used as the shading pattern attribute table. If count is zero, the default table is used. A maximum of 32 values can be specified.

patterns (array of 4-byte binary integers)
The pattern codes to be used consecutively for data groups.

For valid pattern values, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

Coding Examples

In the following example, three patterns are used to draw charts.

```
OPTION BASE 1
INTEGER PATARR
DIM PATARR(3)
MAT READ PATARR
DATA 2,8,3
CALL GDDM('CHPAT',3,PATARR())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, user-defined patterns are not supported, (types 65 through 239).

CHPCTL – Control Pie Chart Slices

```
CHPCTL(count,list)
```

Controls how the slices of a pie chart are displayed. The call operates on exploded slices (those that have been moved out from the rest of the pie, see CHPEXP) and non-exploded slices.

Valid only in state 1.

Parameters

count (4-byte binary integer)
Specifies the number of items in the **list** array. The value must be in the range 0 through 1. If 0 is specified, default values are used for the option.

list (array of short floating-point numbers)
Controls the option for the slices of a pie chart. If the list does not contain a value, the option is not changed. The option is:

Explode Defines the amount, in terms of a multiple of the horizontal radius, by which exploded slices are moved out. For example, when a value of 0.5 is specified, the tip of each exploded slice is moved out to halfway between the center of the pie and its circumference. The horizontal radius of the pie is reduced as necessary to ensure that the entire pie remains within the plotting area. Values greater than 1 make the slices of the pie small and difficult to see. The default is 0.2.

Coding Examples

The following example shows how to use CHPCTL to move the tip of a slice to halfway between the center and circumference of a pie.

```
OPTION BASE 1
DECIMAL LARR
DIM LARR(1)
MAT READ LARR
DATA 0.5
CALL GDDM('CHPCTL',1,LARR())
```

Differences between System/38 and the AS/400 System

This call is not supported on System/38.

CHPEXP – Exploded Slices in Pie Charts

```
CHPEXP(count,list)
```

Specifies which slices of a pie chart are to be exploded (moved out from the other slices of the pie). The CHPCTL call (see page 3-56) controls the amount by which the slice is to be moved.

Valid only in state 1.

Parameters

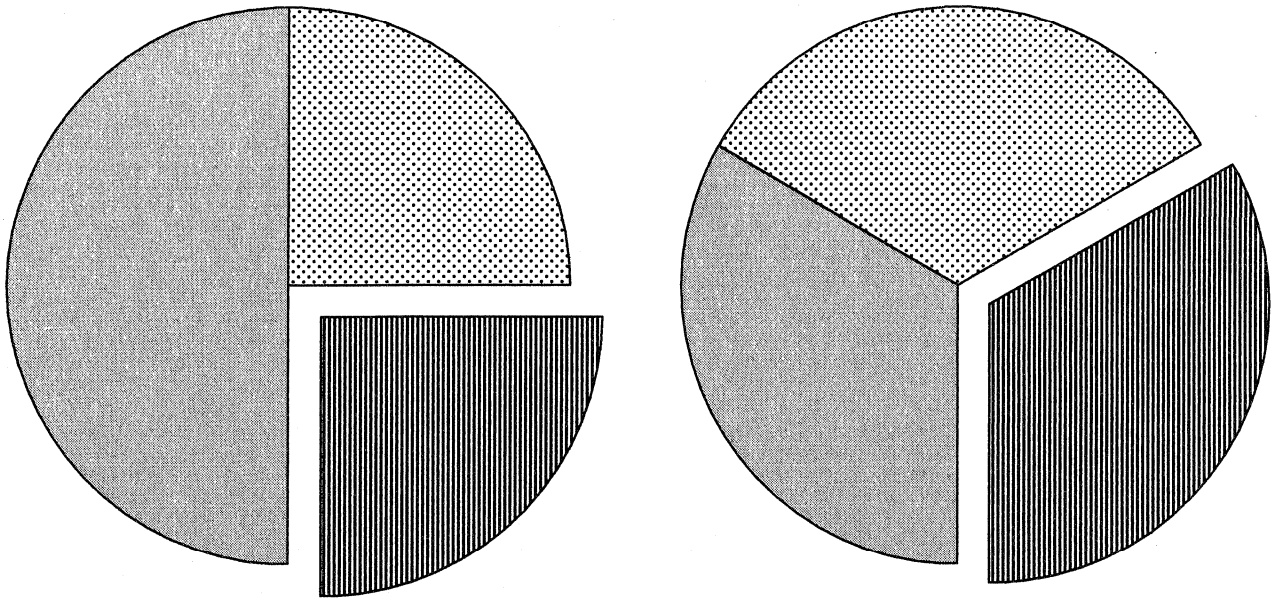
count (4-byte binary integer)
Specifies the number of items contained in the **list** array. The value must be in the range 0 through 32. If 0 is specified, a default explosion array is restored.

list (an array of 4-byte binary integers)
Elements in this array correspond to slices of the pie. If there is more than one pie, the elements correspond to the equivalent slice of each pie. An element that contains 1 indicates that the corresponding slice of the pie is moved. An element that contains 0 indicates that the corresponding slice of the pie is not moved. Only 0 and 1 can be specified in this parameter.

If the length of the list does not match the number of slices in the pie, the list is truncated (if it is too long) or resumes from the start (if it is too short).

The default **list** array contains all zero values, which causes none of the slices to be moved out.

Coding Example



RV2F751-0

The second sector in each of the pies is exploded using CHPEXP.

```
OPTION BASE 1
INTEGER LARR
DIM LARR(3)
MAT READ LARR
DATA 0,1,0                    ! Move second slice
CALL GDDM('CHPEXP',3,LARR())
```

Differences between System/38 and the AS/400 System

This call is not supported on System/38.

CHPIE – Plotting Pie Charts

```
CHPIE(data-groups,count,y values)
```

Draws one or more pie charts. When more than one pie is drawn, horizontal and vertical alignment of the pies is controlled by the CHSET YVERTICAL and XVERTICAL options. YVERTICAL specifies pies each beside the other (default) and XVERTICAL specifies pies one above the other.

The positioning and size of pies for a multiple pie chart is based on the number of pies specified on the first call to CHPIE, or on the value of CHNUM, if CHNUM was used. The CHNUM value takes precedence.

Unless CHSET(NOFILL) has been specified, each sector of a pie is colored and shaded according to the current settings of the attribute tables and the CHSET shading option. Corresponding sectors in each pie have the same attributes.

All axis definition and datum line specifications are ignored, except that any labels specified by CHXLAB are used as titles for the pies.

Valid in state 1 or state 2. If entered in state 1, the program changes to state 2.

Parameters

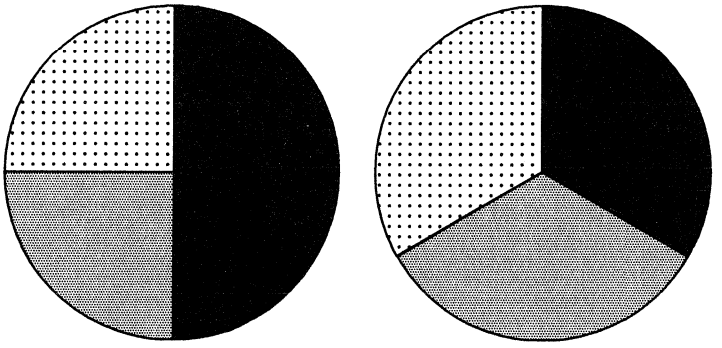
data-groups (4-byte binary integer)
The number of pie charts to be drawn.

count (4-byte binary integer)
The number of sectors in each pie. In a chart with more than one data group, all pies must have the same number of sectors.

y values (array of short floating-point numbers)
The sizes of the pie sectors in each pie. Must be positive or zero. The data can be given in percentages (the default) or in absolute values. Percentage values are subject to rounding errors and may not be precise. CHSET(PERPIE|ABPIE) can be set to indicate which data type is being used.

For absolute pie charts, the total for each data group is assumed to correspond to 360°, and a complete pie is always drawn. The total for each data group in this case must be positive. For percentage pie charts, the total for each data group must not exceed 100. If the total is less than 100, an incomplete circle is drawn. Sectors are drawn in a clockwise direction, with the first sector beginning at the 12 o'clock position.

Coding Examples



RV2F752-0

The following program displays this pie chart. Two pie charts are drawn with three sectors in each pie:

```
OPTION BASE 1
DIM YVALUES(6)                  ! 6 must be the product of 2 and 3
MAT READ YVALUES
DATA 30,40,30,  50,25,25        ! Each 3-sector group adds to 100
CALL GDDM('CHPIE',2,3,YVALUES())
```

CHPIER – Pie Reduction

CHPIER(reduction)

Reduces the size of each pie in a pie chart. This can be useful if the automatic sizing and spacing prevents the display of a multiple pie chart or produces a congested chart.

Valid only in state 1.

Parameters

reduction (4-byte binary integer)
The percentage reduction of the area of each pie constructed. The value must be positive: 99 is the largest reduction.

Coding Examples

In the following example, the size of pie charts to be drawn is reduced by 90%:

```
CALL GDDM('CHPIER',90)
```

CHPLOT – Line Graphs and Scatter Plots

```
CHPLOT(data-groups,count,x values,y values)
```

Constructs one or more data groups consisting of line graphs or scatter plots on the currently selected axes. By default, markers are drawn at each data point, and solid straight lines connect them. The CHSET options (LINES|NOLINES) and (MARKERS|NOMARKERS) can be used to suppress the markers, or to produce a scatter plot by suppressing the lines.

CHSET(VALUES) can be specified to show the data value at the end of each data point. CHVATT can be specified to control the attributes of the values being generated.

The color of the lines and markers, and the line types, line widths, and markers used are determined by the current attribute table settings. The default settings can be modified by CHCOL, CHLT, CHLW, and CHMARK.

Valid in state 1 or state 2. If entered in state 1, the program changes to state 2.

Parameters

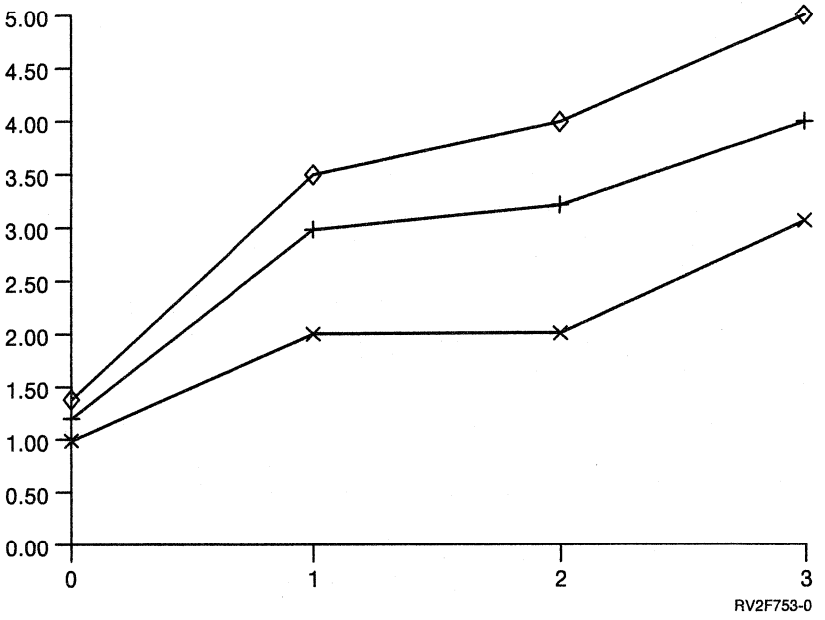
data-groups (4-byte binary integer)
The number of data groups to be drawn, that is, the number of individual graph lines or groups of scatter plot data. Each data group has a new set of attributes.

count (4-byte binary integer)
The number of data points in each graph line or scatter plot group.

x values (array of short floating-point numbers)
The values of the independent variable (x values). Each data group uses the same set of values for the independent variable. For example, if count is 5, five values must be supplied.

y values (array of short floating-point numbers)
The dependent variable (y values). One value must be supplied for each point to be plotted. For multiple data groups, the values must be stored sequentially by data group; that is, all values for the first data group, followed by all values for the second data group, and so forth, for as many data groups as are specified. Each data group must contain as many values as are specified in the count parameter. The y values specified for each data group are associated in sequence with the x values supplied in the x values parameter to give the (x,y) coordinate pairs to be plotted.

Coding Examples



In the following example, a line graph with three lines of four points each is drawn:

```
OPTION BASE 1
DECIMAL XVALUES,YVALUES
DIM XVALUES(4)
MAT READ XVALUES
DATA 0,1,2,3
DIM YVALUES(12)   !  Number of elements is the product of 3 and 4
MAT READ YVALUES
DATA 1,2,2,3.1, 1.2,3,3.2,4, 1.4,3.5,4,5
CALL GDDM('CHPLOT',3,4,XVALUES(),YVALUES())
```

CHRNIT – Reinitialize Presentation Graphics Routines

CHRNIT

Resets all chart definition options and parameters to their default values. If issued in state 2, it also returns the application program to state 1. (See CHSTRT to return to state 1 without reinitialization.)

CHRNIT can be used, for example, to reinitialize the Presentation Graphics routines before using CHAREA to construct a second chart in another area of the screen. See also CHTERM.

CHRNIT does not clear charts from the page. Any further charts would be added to the existing ones. Charts can be cleared by the GSCLR call, described in "GSCLR – Clear the Graphics Field" on page 2-63.

Valid in state 1 or state 2. If coded in state 2, the application program reenters state 1.

Parameters

None

Coding Examples

The following example shows how to specify CHRNIT:

```
CALL GDDM('CHRNIT')
```

CHSET – Specify Chart Options

CHSET(option)

Overrides defaults on all chart attributes that are not related to axes. In general, any attribute that does not need a value and does not refer to an axis is an option of CHSET.

Only one option may be set by each CHSET call. For example, to request a left-justified heading in the bottom margin, two calls are needed:

```
CHSET ('HBOT')
CHSET ('HLEFT')
```

Unless otherwise stated, CHSET calls are valid only in state 1.

Parameters

option (4-byte character string)

The option string must be at least four characters long, and must be enclosed in single quotes. If more than four characters are specified, they are ignored. However, they may be used to increase the readability of the program.

The options are as follows, in alphabetic order by function; the defaults are shown in uppercase **BOLD**.

Function	Option
Axis, autoranging	**YVERTICAL** XVERTICAL
Axis, time of drawing	**IDRAW** DRAW NDRAW
Bar chart, type	**MBAR** CBAR FBAR
Bar values, placement	VINSIDE **VONTOP**
Chart area, boxed or shaded	CBOX CBACK **NCBOX**
Curve fitting or not	CURVE **NOCURVE**
Data, absolute or relative	RELATIVE **ABSOLUTE**
Date label abbreviations	**ABREV** FULL LETTER
Heading justification	**HCENTER** HLEFT HRIGHT
Heading or not	**HEADING** NOHEADING
Heading position	**HTOP** HBOTTOM

Function	Option
Histogram risers	**RISERS** NORISERS
Label areas blanked	BLABEL **NBLABEL**
Legend areas blanked	NBKEY **BKEY**
Legend, box around	KBOX **NKBOX**
Legend or not	**LEGEND** NOLEGEND
Legend, order of elements	**KNORMAL** KREVERSED
Line graphs, lines	**LINES** NOLINES
Line graphs, markers	**MARKERS** NOMARKERS
Marker scaling	BNOTE **NBNOTE**
Notes, box around	NBOX **NONBOX**
Number punctuation	PGFS **NPGFS**
Pie charts, labeling appearance	**SPISLICE** SPILABEL
Pie charts, labeling type	**PIEKEY** SPIDER
Pie charts, relative size	PROPIE **NOPROPIE**
Pie data, type of	**PERPIE** ABPIE
Shading, mountain range	MOUNTAIN **NOMOUNTAIN**
Shading type	**FILL** INFILL NOFILL
Value labels	CVALUES VALUES **NOVALUES**
Value labels, blanking behind	BVALUES **NBVALUES**
x axis, secondary specification	XDUP **XNOD**
y axis, secondary specification	YDUP **YNOD**

CHSET Options

The groups of CHSET options are listed in alphabetic order by option.

- **CHSET 'ABREV' | 'FULL' | 'LETTER'**

  Specifies the form of axis labels supplied by CHXMTH, CHYMTH, CHXDAY, and CHYDAY. It applies to all charts except pie charts and Venn diagrams.

  ABREV specifies three-character abbreviations.

  FULL specifies complete words.

  LETTER specifies first letters only.

  The following tables list the month/day sequence numbers and variations.

Number	ABREV	FULL	LETTER
Month			
1	JAN	JANUARY	J
2	FEB	FEBRUARY	F
3	MAR	MARCH	M
4	APR	APRIL	A
5	MAY	MAY	M
6	JUN	JUNE	J
7	JUL	JULY	J
8	AUG	AUGUST	A
9	SEP	SEPTEMBER	S
10	OCT	OCTOBER	O
11	NOV	NOVEMBER	N
12	DEC	DECEMBER	D
Day			
1	MON	MONDAY	M
2	TUE	TUESDAY	T
3	WED	WEDNESDAY	W
4	THU	THURSDAY	T
5	FRI	FRIDAY	F
6	SAT	SATURDAY	S
7	SUN	SUNDAY	S

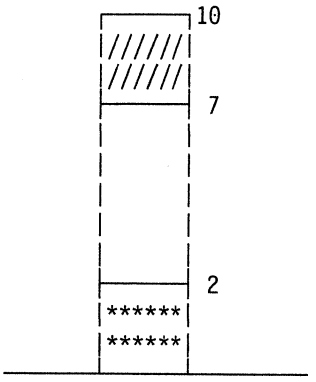
- **CHSET 'ABSOLUTE' | 'RELATIVE'**

  ABSOLUTE specifies that absolute data values are supplied.

  RELATIVE specifies that the dependent data values, for all data groups except the first, are relative to the previous component. Relative data is the natural and recommended form of data for all chart types that involve stacking of components. The depth of each bar (or layer in a surface chart) then gives the absolute data value. The following example illustrates this point:

Component	Value Supplied	Value Plotted
1	2	2
2	5	7
3	3	10

The resultant bar chart is:



- **CHSET** 'BKEY' | **'NBKEY'**

  BKEY specifies that the rectangular area to be occupied by the legend is blanked before the legend is constructed.

  NBKEY specifies that the legend area is not blanked.

- **CHSET** 'BLABEL' | **'NBLABEL'**

  Controls blanking of the areas to be occupied by axis labels, before the labels are constructed. It can be used with CHSET('CBACK'), CHSET('DRAW'), or CHDRAX, to ensure that the axis labels are not obscured by background shading or some aspect of the plot.

  Valid in state 1 or state 2.

  NBLABEL does not blank label areas.

  BLABEL blanks the label areas before the labels are written.

- **CHSET** 'BNOTE' | **'NBNOTE'**

  Controls the blanking of areas to be occupied by notes, before the notes are written (see CHNOTE).

  BNOTE blanks note areas before the notes are written.

  NBNOTE does not blank note areas.

  Valid in state 1 or state 2.

- **CHSET** 'BVALUES' | **'NBVALUES'**

  Controls blanking of the areas to be occupied by data values on bar, line, or surface charts if CHSET('VALUES') is set.

  Valid in state 1 or state 2.

  NBVALUES does not blank the value areas.

  BVALUES blanks the value areas before the values are written.

- **CHSET** 'CBOX' | **'NCBOX'** | 'CBACK'

  Permits the drawing of a framing box about the perimeter of the chart area or the shading of the background. Defaults for the attributes of the lines defining the box can be overridden by CHBATT.

  CBOX specifies that the box is to be drawn.

  NCBOX suppresses the framing box.

CBACK specifies that the entire chart area is to be shaded with a solid pattern as a background, in addition to drawing the framing box. The background color will be the same as the lines defining the box, as set by CHBATT.

CBACK should *not* be used for Venn diagrams, for monochrome devices, or when the chart is to be printed. Furthermore, remember that the data streams generated by this option may be very long, because many more cell definitions are generated with multicolor contents. PS stores may be quickly exhausted with other than simple charts. Note that to cancel a CBACK request, CBOX or NCBOX must be specified.

- **CHSET** 'CURVE' | **'NOCURVE'**

  Controls the manner in which the supplied data points in a line graph, surface chart, or polar chart are connected. Each consecutive pair of points may be connected by a straight line, or by a smooth curve. For a line graph or polar chart, it is effective only if CHSET('LINES') is also specified (or defaulted).

  If a smooth curve is specified, the smoothness of the curve may be controlled by the CHFINE call. Note that the smooth curve constructed by use of this option is not intended to provide an interpretation of the supplied data values. Rather, the option is provided purely for its cosmetic effect on the resulting graph.

  Valid in state 1 or state 2.

  CURVE specifies that consecutive data points within each data group of a line graph, polar chart, or surface chart are to be connected by a smooth curve constructed using the current data-group attributes.

  NOCURVE specifies that consecutive data points within each data group of a line graph, polar chart, or surface chart are to be connected by straight lines constructed using the current data-group attributes.

- **CHSET** 'CVALUES' | 'VALUES' | **'NOVALUES'**

  Specifies how values are to be displayed for bar charts (or, in the case of the 'VALUES' option, for bar, pie, line, and surface charts).

  Valid in state 1 or state 2.

  CVALUES specifies that controlled bar-chart values are to be provided. Related calls are CHSET('VINSIDE' | 'VONTOP') and CHVATT with the bar-value rotation option. Values and bars might overlap on composite (stacked) or floating bar charts. See "CHVCHR — Number of Characters in Value Labels" on page 3-85 for a further discussion. Text attributes for the value representations are set by CHVATT. The area to be occupied by each bar value may be blanked, before the value is constructed, with the BVALUES option.

  VALUES controls the appearance of dependent data values at the ends of bars on bar charts, at the data points of line and surface charts, or the percentage values on pie charts.

  For bar charts, VALUES constructs character representations of the dependent data values at the end of each bar, on each CHBAR or CHBARX call. Overlap of values and bars may occur on composite (stacked) or floating bar charts. See the description of CHVCHR for further discussion. Text attributes for the value representations are set by CHVATT. The area to be occupied by each bar value may be blanked, before the value is constructed, with the BVALUES option.

  For line and surface charts, VALUES constructs character representations of the dependent data values above or below each data point on each CHPLOT or CHSURF call. Overlap of values may occur. See "CHVCHR — Number of Characters in Value Labels" on page 3-85 for further discussion. Text attributes

for the value representations are set by CHVATT. The area to be occupied by each value may be blanked before the value is constructed, with the BVALUES option.

Following are the rules for the number of decimal places that are generated on bar, line, and surface chart values when displaying the chart:

- All values always have the same number of decimal places displayed, even if they did not have the same number when entered.
- If at least one value entered has at least one decimal place, a minimum of two decimal places is displayed for all values. For example, if the values 12, 18, and 15.1 were input, the displayed values would have two decimal places (12.00, 18.00, and 15.10).
- If all values entered are equal, two decimal places are always displayed. If more than two decimal places were entered, the value is rounded off to two decimal places. For example, if the values 38.748, 38.748, and 38.748 were input, the displayed values would be 38.75, 38.75, and 38.75.
- The following table describes the general algorithm used to determine the number of decimal places generated:

Smallest Difference Between Values	Decimal Places Displayed
1 > *diff* > = .1	2
.1 > *diff* > = .01	3
.01 > *diff* > = .001	4
.001 > *diff* > = .0001	5
.0001 > *diff* > = .00001	6
.00001 > *diff* > = .000001	7
⋮	⋮

Note: *diff* can either be between two input values or between an input value and its truncated integer value. For example, if the values 34.004, 33.054, and 32.844 were specified as input, six differences would be considered:

1. 34.004 – 33.054 = 0.950 (Difference between value 1 and value 2)
2. 34.044 – 32.844 = 1.160 (Difference between value 1 and value 3)
3. 33.054 – 32.844 = 0.210 (Difference between value 2 and value 3)
4. 34.004 – 34.0 = 0.004 (Difference between value 1 and truncated value 1)
5. 33.054 – 33.0 = 0.054 (Difference between value 2 and truncated value 2)
6. 32.844 – 32.0 = 0.844 (Difference between value 3 and truncated value 3)

The smallest difference (which is used to determine the number of decimal places displayed) would be 0.004, which is the difference between 34.004 and 34, where 34 is the truncated value of 34.004. This would produce the following values with four decimal places: 34.0040, 33.0540, and 32.8440.

For pie charts, VALUES has two possible meanings:

- If the option CHSET('PIEKEY') (the default) is in effect, integer percentage data values are constructed as spider labels around the pie, connected to their corresponding slices by lines. Each consists of the character

representation of the integer closest to the percentage slice value, followed by a % sign.

– If CHSET('SPIDER') is in effect, spider labels are constructed, each consisting of a percentage representation as above, followed by a label text string as specified by CHKEY.

In both cases, the data values are always in percentage form, regardless of the pie chart data type. See CHSET('SPISLICE') for a description of text attributes.

NOVALUES indicates that no values are given on bar charts, pie charts, line charts, or surface charts.

- **CHSET 'FILL'** | 'INFILL' | 'NOFILL'

Specifies the method of shading. Applies to all charts except line graphs and scatter plots. Note that shading, when specified, always shades the *interior* of a defined boundary. If the defined boundary crosses itself, the effects of shading may be unexpected.

Valid in state 1 or state 2.

Shading patterns are taken from the pattern-attribute table. The default table may be overridden by a call to CHPAT.

– Surface charts, bar charts, and histograms

FILL specifies that all data groups are shaded with patterns taken from the pattern-attribute table. The first data group specified in a plotting routine call is shaded to the x axis (or y datum reference line if specified). Subsequent data groups in the same call are shaded to the previous data group.

INFILL is equivalent to FILL for multiple bar charts. It suppresses the shading of the first data group for surface charts, histograms, and composite (stacked) bar charts.

NOFILL suppresses all shading for all charts except surface charts, for which it is equivalent to INFILL.

– Pie charts

FILL specifies that all pie slices are shaded with patterns taken from the pattern-attribute table. Corresponding slices in different pies are shaded with the same pattern.

INFILL is equivalent to FILL.

NOFILL suppresses shading for all pie slices.

– Venn diagrams

FILL specifies that the two population circles are shaded with patterns from the pattern attribute table.

INFILL is equivalent to FILL.

NOFILL suppresses shading of the two population circles.

- **CHSET 'HCENTER'** | 'HLEFT' | 'HRIGHT'

Controls the justification of the chart heading.

HCENTER specifies that each line of the heading is centered between the left and right edges of the chart area.

HLEFT specifies that each line of the heading is left-justified to the edge of the chart area.

HRIGHT specifies that each line of the heading is right-justified to the edge of the chart area.

- **CHSET 'HEADING'** | 'NOHEADING'

  This option controls whether or not the chart heading is displayed.

  HEADING specifies that the heading is displayed.

  NOHEADING specifies that the heading is suppressed.

- **CHSET 'HTOP'** | 'HBOTTOM'

  Controls the location of the chart heading.

  HTOP specifies that the heading is placed in the top margin.

  HBOTTOM specifies that the heading is placed in the bottom margin.

- **CHSET 'IDRAW'** | 'DRAW' | 'NDRAW'

  This option governs the time of construction of the axes, together with their scale marks, labels, and associated grid/datum lines.

  Valid in state 1 or state 2.

  IDRAW specifies that each axis is constructed automatically at the time of its *definition*, that is, when the first plotting routine call is issued with the axis selected. In this case each axis is constructed once and once only, *before* the data supplied on the plotting routine call is plotted. Data thus overwrites the axis where they cross.

  DRAW specifies that each axis is constructed automatically on *each* plotting routine call for which the axis is selected, *after* the data is plotted. In this case each axis is constructed as many times as there are plotting routine calls for which the axis is selected. The axis thus overwrites the data. DRAW may result in unnecessary processing if several plotting routines calls are made. In such a situation, data can be overwritten using the CHDRAX call and possibly the NDRAW option.

  NDRAW suppresses the automatic construction of axes on plotting routine calls. Axis drawing can be explicitly requested by means of a CHDRAX call.

  In all cases, axis construction may be forced at any time in state 2 by execution of CHDRAX.

- **CHSET** 'KBOX' | **'NKBOX'**

  Controls the construction of a box around each key in the legend. NKBOX specifies that no key box is to be constructed.

  KBOX specifies that each key in the legend is surrounded by a box whose attributes are the same as those for the primary x axis line (see CHAATT).

- **CHSET 'KNORMAL'** | 'KREVERSED

  Controls the order of items in a legend (K stands for Key).

  KNORMAL specifies that the first key is on the left or bottom.

  In a horizontal legend, each row of key entries is constructed from left to right (using the keys in the order in which they are defined by a CHKEY call). Similarly, in a vertical legend, each column of key entries is constructed from bottom to top.

  KREVERSED specifies that the first key is on the right or top.

  In a horizontal legend, each row of key entries is constructed from right to left (using the keys in the order in which they are defined by a CHKEY call).

Similarly, in a vertical legend, each column of key entries is constructed from top to bottom.

- **CHSET 'LEGEND'** | 'NOLEGEND'

  Specifies whether a legend is to be constructed. Legends are not produced for Venn diagrams irrespective of the setting of this option. See the PIEKEY option for information about construction of legends for pie charts.

  LEGEND specifies that a legend is to be constructed after any subsequent plotting calls (for example, CHPLOT, CHBAR, and CHBARX).

  Valid in state 1 or state 2.

  NOLEGEND specifies that a legend is not to be constructed.

  Valid in state 1 only.

- **CHSET 'LINES'** | 'NOLINES'

  Controls whether CHPLOT and CHPOLR produce a line graph (LINES) or a scatter plot (NOLINES).

  Valid in state 1 or state 2.

  LINES specifies that consecutive points on each data group are joined by a line, or curve (see CHSET('CURVE')).

  NOLINES specifies that only the markers are drawn, giving the effect of a scatter plot. If the CHSET('NOMARKERS') option is specified, nothing is drawn for CHPLOT.

- **CHSET 'MARKERS'** | 'NOMARKERS'

  Indicates whether markers are shown.

  Valid in state 1 or state 2.

  MARKERS specifies that markers on line graphs, scatter plots, or polar charts appear at the data points specified on a CHPLOT or CHPOLR call. Markers are obtained from the marker-attribute table. The default marker table can be overridden by a CHMARK call.

  NOMARKERS suppresses the construction of markers.

- **CHSET 'MBAR'** | 'CBAR' | 'FBAR'

  Specifies the type of bar chart to be constructed.

  Valid in state 1 or state 2.

  MBAR specifies a multiple bar chart (groups of bars, side-by-side).

  CBAR specifies a composite (stacked) bar chart (colinear bars).

  FBAR specifies a floating bar chart (like a stacked bar chart but with first data group bars omitted).

- **CHSET** 'MOUNTAIN' | **'NOMOUNTAIN'**

  MOUNTAIN specifies that for surface charts the successive data groups appear behind the previous data groups so that the shading does not overlap. This effect is called mountain-range shading. Chart details (for example, grid lines and data lines) that are behind a mountain range are not shown. The first component is shaded if the CHSET option 'FILL' has been specified; otherwise, shading is suppressed. Also, this option is not valid for hardcopy devices (plotters and printers).

NOMOUNTAIN suppresses mountain-range shading and restores the standard method of shading.

- **CHSET 'NONBOX' | 'NBOX'**

  Controls the construction of a framing box round all subsequent chart notes, until the option is changed. Refer to the CHNOTE call description for the parameters that affect framing-box construction.

  Valid in state 1 or state 2.

  NONBOX suppresses construction of a framing box round subsequent chart notes.

  NBOX specifies construction of a framing box round subsequent chart notes.

- **CHSET 'PERPIE' | 'ABPIE'**

  Specifies the data type for pie charts.

  Valid in state 1 or state 2.

  PERPIE specifies that the data supplied in any subsequent calls to CHPIE is a percentage. Each slice subtends an angle of V*360/100 degrees, where V is the data value for the slice.

  If the total for all slices is less than 100, the complete circle is not drawn. If it is greater than 100, an error message is issued.

  ABPIE specifies that the data is absolute. All values of a pie are added together and the slice angle determined as a proportion of the total. Each slice subtends an angle of V*360/T degrees, where V is the data value and T is the total of all data values for the pie.

- **CHSET 'PGFS' | 'NPGFS'**

  Specifies the method used to punctuate numbers of 1000 and over. The convention is used for the display of numeric data values in fixed-point format. Applies to all numeric axis labels, to bar values on bar charts, and to table chart values. These options are used in CHXSET and CHYSET calls to specify punctuation for numeric labels on the x or y axis.

  Valid in state 1 or state 2.

  PGFS suppresses all punctuation except the decimal point. This minimizes the space required for numeric values, for example:

  ```
  1234567.999
  ```

  NPGFS selects the default national convention that may be specified when GDDM is installed. Widely used conventions are:

  - The period decimal convention. Example: 1,234,567.999
  - The comma decimal convention. Example: 1.234.567,999
  - The French convention. Example: 1 234 567,999

- **CHSET 'PIEKEY' | 'SPIDER'**

  Specifies the configuration of pie chart labels.

  PIEKEY specifies that a legend will be constructed in the chart margin.

  SPIDER suppresses the generation of a legend, and specifies the construction of spider labels consisting of the labels specified by CHKEY written around the circumference of each pie. The labels are connected to their corresponding slices by lines. See CHSET('SPISLICE') for further discussion.

Note: Any labels supplied by CHXLAB are written as pie titles adjacent to the pies, independently of labels supplied by CHKEY.

- **CHSET** 'PROPIE' | **'NOPROPIE'**

  Applies only to multiple pie charts using absolute (not percentage) data, generated by a single call to CHPIE.

  NOPROPIE specifies that all pies generated are the same size.

  PROPIE specifies that the area of each pie will be proportional to the sum of its slice values. The pies generated from a single call to CHPIE will be in proportion. The largest pie in each such group of pies has the same size as if NOPROPIE had been specified, and the remaining pies are reduced in proportion.

- **CHSET 'RISERS'** | 'NORISERS'

  Controls whether steps of a histogram are delimited by riser lines.

  Valid in state 1 or state 2.

  RISERS specifies that lines perpendicular to the x axis are drawn at the ends of each histogram range. NORISERS suppresses the generation of intermediate histogram risers. The two outermost risers remain, joining the ends of the step to the x axis or datum reference line.

- **CHSET 'SPISLICE'** | 'SPILABEL'

  Specifies the appearance of pie-chart labels. It determines whether the pie labels and spider lines are the same colors as the slice to which they are attached, or are controlled by CHKATT or CHVATT. SPISLICE and SPILABEL are effective only if slice values, or slice labels, or both are specified. Slice values are specified by CHSET('VALUES'), and slice labels are specified by CHKEY and CHSET('SPIDER').

  Valid in state 1 or state 2.

  SPISLICE draws the spider line and associated text (slice value, or label, or both) in the color of the corresponding slice. The character mode, symbol-set identifier, and character multiplier for the text are determined by CHKATT if CHSET('SPIDER') is in effect, by CHVATT otherwise.

  SPILABEL draws the spider lines in the default color. The associated text attributes are those specified by CHKATT if CHSET('SPIDER') is in effect, by CHVATT otherwise.

- **CHSET** 'VINSIDE' | **'VONTOP'**

  Controls the placement of values for bar charts. The values can appear either on top of or inside the bars. A CHSET('CVALUES') call is required if you specify one of these options.

  VINSIDE specifies that bar values are to be centered within each bar.

  VONTOP specifies that bar values are to be placed on top of the bars.

  Valid in state 1 or state 2.

- **CHSET** 'XDUP' | **'XNODUP'**
- **CHSET** 'YDUP' | **'YNODUP'**

  These options specify that a duplicate of the primary axis is used as a secondary axis. By default this will give axes at both sides or ends of the graph.

  XDUP specifies that the primary x axis is duplicated.

YDUP specifies that the primary y axis is duplicated.

XNODUP specifies that the x axis is not duplicated.

YNODUP specifies that the y axis is not duplicated.

For pie charts and Venn diagrams, a duplicate axis specification is ignored.

When a duplicate axis is specified, the secondary axis is used to support it. For this reason, certain items are copied into the secondary axis fields on entry to state 2 if the duplicate axis option is set. The following items are copied: axis range, scale mark positions, scale type (linear or logarithmic), scaling factor, and axis attributes (line type, width, and color). These supersede any corresponding items that may have been explicitly specified for the secondary axis. Also, the secondary axis grid lines option is suppressed to avoid redrawing grid lines.

- **CHSET 'YVERTICAL'** | 'XVERTICAL'

  Specifies the orientation of *all* axes; also controls the orientation of multiple pie charts, and Venn diagrams.

  YVERTICAL specifies that the y axis should be vertical and the x axis horizontal. YVERTICAL is the default and creates horizontal line graphs, and bar charts and histograms with vertical bars. Venn diagrams are arranged horizontally and pie charts in a horizontal line if there are more than one.

  XVERTICAL specifies that the x axis is to be vertical and the y axis horizontal. XVERTICAL creates vertical line graphs, and bar charts and histograms with horizontal bars. Venn diagrams are vertically aligned. Pie charts are arranged above one another if there are more than one.

Differences between System/38 and the AS/400 System

The following options are not supported on System/38:

- CHSET (CVALUES|VALUES|NOVALUES)
- CHSET (VINSIDE|VONTOP)

CHSTRT – Reset the Processing State to State 1

CHSTRT

If issued in state 1, CHSTRT has the same effect as CHRNIT. It resets all chart definition options and parameters to their default values.

If issued in state 2 (as is usually the case), CHSTRT resets the processing state to state 1, and resets the chart definition options and parameters to their values just before the most recent transition from state 1 to state 2. CHSTRT also causes the primary axes to become selected. Note that because very few options can be set in state 2, CHSTRT leaves most settings unaltered.

CHSTRT can be used to plot a series of charts using the same (or similar) formats, without the overhead of specifying again the common format for each chart.

Parameters

None

Coding Example

The following example shows how to use CHSTRT:

```
CALL GDDM('CHSTRT')
```

CHSURF – Surface Charts

```
CHSURF(data-groups,count,x values,y values)
```

This call produces one surface chart on the currently selected axes.

A surface chart is essentially a shaded line graph without marker symbols. As in a line graph, each data group is drawn as a series of lines connecting the data points.

CHSET(VALUES) can be specified to show the data value at the end of each data point. CHVATT can be specified to control the attributes of the values being generated.

Data group colors, shading patterns, line types, and line widths are determined by the current attribute tables. The default settings can be modified by CHCOL, CHPAT, CHLT, and CHLW. Surface charts are, in general, drawn with relative data and are often improved by curved lines. CHSET(RELATIVE) and CHSET(CURVE) control these aspects.

Type of shading can be controlled by the CHSET(MOUNTAIN|NOMOUNTAIN) call.

Shading of the first data group can be suppressed by calling CHSET(INFILL).

Valid in state 1 or state 2. If coded in state 1, the program changes to state 2.

Parameters

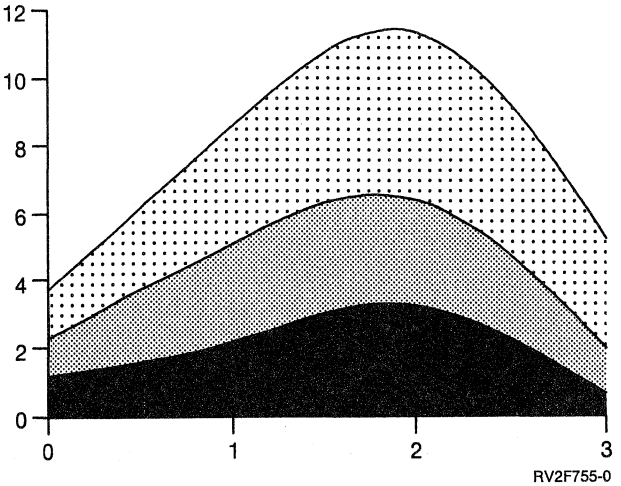
data-groups (4-byte binary integer)
The number of data groups to be drawn, that is, the number of individual graph lines. Each data group has a new set of attributes.

count (4-byte binary integer)
The number of data points in each data group.

x values (array of short floating-point numbers)
The x values for the chart. Each data group uses the same set of x values. For example, if count is 5, five values must be supplied.

y values (array of short floating-point numbers)
The y values for the chart. One value must be supplied for each point to be plotted. For multiple data groups, the values must be stored sequentially by data group; that is, all values for the first data group, all values for the second data group, and so forth, for as many data groups as specified. Each data group must contain count values. The y values specified for each data group are associated in sequence with the x values supplied in the x values parameter to give the (x,y) coordinate pairs to be plotted.

Coding Example



The following shows how to draw this surface chart (with three lines of four points each):

```
OPTION BASE 1
CALL GDDM('CHSET','RELATIVE')  !  Draw with relative data
CALL GDDM('CHSET','CURVE')     !  Use curved lines
DECIMAL XVALUES,YVALUES
DIM XVALUES(4)
MAT READ XVALUES
DATA 0,1,2,3
DIM YVALUES(12)
MAT READ YVALUES
DATA 1,2,3.1,0.5, 1.2,3,3.2,1.4, 1.4,3.5,5,3.3
CALL GDDM('CHSURF',3,4,XVALUES(),YVALUES())
```

CHTATT – Axis Title Text Attributes

```
CHTATT(count,array)
```

Overrides or resets the default attributes to be used for all axis titles.

The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (default characters for the device)
Character size multiplier	100 results in the character size of device unless overridden by CHCGRD

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. If more than four elements are specified, the excess elements are ignored. If fewer than four are specified, the remainder are unchanged from their previous setting. If count is zero, all four defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array are the following attributes:

1. Color (valid values have the same effect as the GDDM routine GSCOL).

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Character mode.

   Valid values are:

   0 System default (mode 3)
   2 Mode 2
   3 Mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier denotes:

   0 Default character set
   n A graphics symbol set that has been previously loaded using the GDDM call GSLSS

4. Character size multiplier.

   For mode 2 or mode 3, the character size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. (The default character size is the size of hardware characters for the device on which the chart is displayed.) For example, if character size multiplier is 200 and mode-3 characters are being used, they are drawn twice as large as the basic. The character size multiplier must have a value greater than zero.

Coding Example

The following example shows how to use CHTATT:

```
OPTION BASE 1
INTEGER ARRAY
DIM ARRAY(4)
MAT READ ARRAY
DATA 7,3,65,150
CALL GDDM('CHTATT',4,ARRAY())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

CHTERM – Terminate Presentation Graphics Routines

CHTERM

Terminates the Presentation Graphics routines. The main action taken is to free any storage obtained on previous calls to the Presentation Graphics routines. Use CHTERM when your application program has finished using the Presentation Graphics routines.

Further, it is recommended that CHTERM be issued as soon as the final form of the chart has been constructed, and before ASREAD or FSFRCE is called to send data to the device. This practice provides the most efficient use of storage. The next call to the Presentation Graphics routines (after return from ASREAD or FSFRCE) is treated as if it were the first call. CHTERM thus provides the same function as CHRNIT, except that it also releases all storage used by the Presentation Graphics routines.

Neither CHTERM nor CHRNIT clears charts from the page. Any further charts would be added to the existing ones. Charts can be cleared by the GSCLR call.

Valid in state 1 or state 2.

Parameters

None

Coding Example

The following example shows how to use CHTERM:

```
CALL GDDM('CHTERM')
```

CHVATT – Attributes of Values Text in Bar and Pie Charts

```
CHVATT(count,array)
```

Overrides or resets the default attributes to be used for data values on bar, line, and surface charts, or for percentage values on pie charts. These values are only shown if CHSET(VALUES) has been specified.

See CHSET(SPISECTOR) for a description of the applicability of CHVATT to pie chart spider labels.

By default, the attributes are as follows:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (default characters for the device)
Character size multiplier	100 results in the character size of device unless overridden by CHCGRD
Character height/width multiplier	100 results in the character size of device unless overridden by CHCGRD
Rotation	0

Valid in state 1 only.

Parameters

count (4-byte binary integer)
The number of elements in the array parameter. When more than six elements are specified, the excess elements are ignored. When fewer than six are specified, the remainder are unchanged from their previous setting. If count is zero, all six defaults are reinstated.

array (array of 4-byte binary integers)
The elements of the array are the following attributes:

1. For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."
2. Character mode.

   Valid values are:

   0 System default (mode 3)
   2 Mode 2
   3 Mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier denotes:

   0 Default character set
   n A graphics symbol set that has been previously loaded using the GDDM call GSLSS

4. Character size multiplier.

   For mode 2 or mode 3, the character size multiplier is divided by 100, and the horizontal and vertical character sizes are multiplied by this factor before the characters are drawn. The default character size is the size of the hardware character for the device. For example, if character size multiplier is 200 and mode 3 characters are being used, they are drawn twice as large as normal. The character size multiplier must have a value greater than zero.

5. Character height/width multiplier.

   Specifies the height of the character box relative to its width. For example, 100 indicates that the height is multiplied by the same amount as the width; 200 causes the height to be multiplied by twice the amount of the width.

6. Rotation.

   Value labels may be rotated away from the horizontal. This element must have a value from −9000 through +9000. This number is divided by 100 to give a value from −90 through +90 (the angle in degrees) to the horizontal at which value labels are drawn. The default is zero. When positive, the angle is in a counterclockwise direction. When negative, it is in a clockwise direction.

Coding Example

The following example shows how to use CHVATT:

```
OPTION BASE 1
INTEGER ARRAY
DIM ARRAY(4)
MAT READ ARRAY
DATA 7,3,65,150
CALL GDDM('CHVATT',4,ARRAY())
```

Differences between the System/370 Computer and the AS/400 System

On the AS/400 system, character mode 1 is not supported.

CHVCHR – Number of Characters in Value Labels

```
CHVCHR(number)
```

Overrides the default maximum number of characters to be used for displaying the value on each bar of a bar chart, or on each data point of a line or surface chart if CHSET(VALUES) is specified. Value text attributes are controlled by CHVATT.

The default maximum is 9.

Valid in either state 1 or state 2.

Parameters

number (4-byte binary integer)

The maximum number of value characters that are displayed at the end of each bar of a bar chart, or at each data point of a line or surface chart. This includes the minus sign, decimal point character, and separator characters for 10**3 and 10**6 positions, if present. If a value cannot be displayed in this number of characters, asterisks replace it.

The default value is 9, and the maximum is 15. Since the 15-character maximum includes the sign, decimal point, and possibly a signed exponent, the maximum number of digits displayed is eight.

For bar charts, if the number of characters specified does not fit across the bar, and character-mode 3 is specified for the value text, the character box width or height is adjusted to force a fit. Note that this can produce very small mode-3 characters.

For line and surface charts, no character size adjustments are made. This is the responsibility of the programmer.

In general, the Presentation Graphics routines involved use the minimum number of characters needed to distinguish between different values. The format used is similar to those of the matching y axis labels.

Coding Example

The following example shows how to use CHVCHR:

```
CALL GDDM('CHSET','VALUES')
CALL GDDM('CHVCHR',2)
```

CHVENN – Venn Diagram

```
CHVENN(population1,population2,overlap)
```

Draws a Venn diagram.

Two circles are drawn within the plotting area, proportional in area to the first two population data values supplied, and positioned in such a way that the area of their overlap is proportional to the third *overlap* parameter supplied.

The centers of the circles can be aligned horizontally or vertically with the first circle to the left or at the top respectively. Alignment specifications are controlled by the CHSET option YVERTICAL (default horizontal alignment) or XVERTICAL (vertical alignment).

Three labels corresponding to the three populations can be specified by CHKEY; their attributes are controlled by CHKATT. The labels are constructed in the chart margins. If fewer than three labels are specified, only those specified appear. If more than three are specified, only the first three are used.

When the circles are arranged horizontally, the label for the first area is to the left, the label for the second area is to the right, and the label for the area of overlap is below the overlapping area. The labels are joined to the areas by straight lines. If no overlap exists, the third label does not appear.

When the circles are arranged vertically, the first label is above, the second label is below, and the third label is to the left of the circles.

Each circle is shaded and colored according to the current settings of the attribute tables and the CHSET shading option setting. The intersection merges the two shading patterns and mixes the two color settings. Because color mixing is used for Venn diagrams (and only Venn diagrams), CHSET(CBACK) should not be specified for a Venn diagram. Color and shading attribute tables can be altered by CHCOL and CHPAT.

Only one Venn diagram can be produced at a time. Subsequent invocations of this call in state 2 produce an error message.

Valid only in state 1.

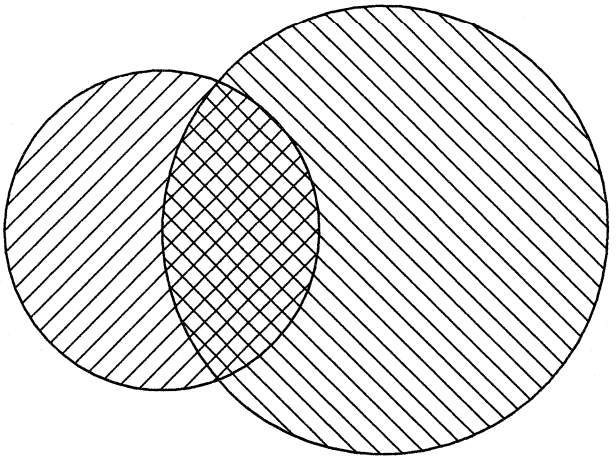
Parameters

population1 (short floating-point number)
The size of the set represented by the area of the first circle. Must not be negative.

population2 (short floating-point number)
The size of the set represented by the area of the second circle. Must not be negative.

overlap (short floating-point number)
The size of the intersection of the two population sets, represented by the area of overlap of the two circles. Must be less than or equal to the smaller of population1 or population2. Must not be negative.

Coding Example



RV2F756-0

The following shows how to draw this Venn diagram:

```
CALL  GDDM('CHVENN',  1000.0,  2100.0,  400.0)
```

CHVMAR – Left and Right Margins

CHVMAR(left-margin,right-margin)

Overrides the default left and right chart margins, and hence the size of the plotting area also. The margin control sizes are specified in terms of character grid rows and character grid columns, where the character grid is defined by CHCGRD, or is taken by default (the default is the hardware character size of the device on which the chart is displayed or printed). By default, the horizontal margins are 5 character-rows above and below the plotting area, and the vertical margins are 10 character-columns on each side of the plotting area.

For top and bottom margins, use CHHMAR.

CHHMAR and CHVMAR can be used to enlarge the margins to give more room for the legend, or to reduce them to give more room to the chart.

Valid only in state 1.

Parameters

left-margin,right-margin (4-byte binary integers)
The number of character columns at the left and right sides respectively of the plotting area.

Coding Example

In the following example, CHVMAR sets the left margin and right margins so that the plotting area is 15 columns in from the left and 15 columns in from the right:

```
CALL GDDM('CHVMAR',15,15)
```

CHXDAY, CHYDAY – Day Labels

```
CHXDAY(day)
CHYDAY(day)
```

Specify English language, uppercase day labels for the currently selected axes. The specified day is associated with the first major scale mark position. Each successive day is assigned to the next major mark. See the comments under CHXMTH about label abbreviations.

These calls implicitly set the corresponding label type to DATE.

Valid only in state 1.

Parameters

day (4-byte binary integer)
The day to be associated with the first major scale mark. Valid values are 1 through 7. Monday is day 1.

Coding Example

In the following example, CHXDAY sets the first label (on the x axis) to Monday:

```
CALL GDDM('CHXDAY',1)
```

CHXDTM, CHYDTM – Specify Translated Axis Lines or Datum Lines

```
CHXDTM(x-value)
CHYDTM(y-value)
```

Specify a translated axis line or datum line to be drawn relative to the currently selected x axis or y axis. The x or y value must lie within the axis range.

CHXDTM draws a line parallel to the y axis with the x-value parameter specifying the point of intersection with the x axis.

CHYDTM draws a line parallel to the x axis with the y-value parameter specifying the point of intersection with the y axis.

The color, line type, and line width are controlled by the current datum line attribute values, which can be specified by CHDATT.

Translated axis lines: In state 1, the call specifies that a translated axis is to be constructed when the axis is constructed. *If CHYDTM is specified in state 1, the line drawn is the reference line to be used in place of the axis for all shaded charts. The line also acts as a baseline for bars of bar charts or risers of histograms.*

If more than one call is made in state 1, the most recent is used.

Datum lines: In state 2, the call results in an immediate construction of a datum line. More than one datum line can be added in state 2. Such lines have no effect on the shading of charts.

Parameters

x-value (short floating-point number)
The x-axis value through which the line is to pass.

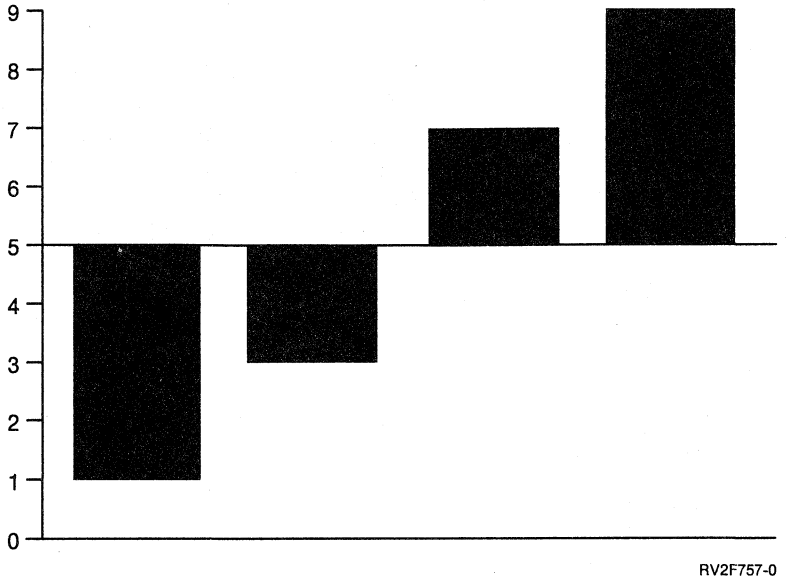
y-value (short floating-point number)
The y-axis value through which the line is to pass.

Coding Example

The following example shows how to specify CHYDTM:

```
CALL GDDM('CHYDTM',5.0)
```

When used in state 1, CHYDTM has an effect like the following:



Note that a line drawn with CHYDTM is parallel to the x axis.

CHXINT, CHYINT – Specify Intercept

```
CHXINT(x)
CHYINT(y)
```

Specify interception points for the currently selected axes. (Primary axes are selected by default.) CHXINT positions the y axis by giving the position on the x axis at which the y axis should cross it, and CHYINT positions the x axis by giving the position on the y axis where the x axis should cross. CHXINT is only brought into effect if CHYSET(INTERCEPT) is also specified. Similarly, CHYINT is only applied if CHXSET(INTERCEPT) is also specified. Ignored for pie charts and Venn diagrams.

If the intercept value is outside the range when the axis is to be constructed, the range is extended to include it. Note that intercept values for logarithmic axes must be positive and nonzero.

If CHXSET or CHYSET INTERCEPT is specified and CHXINT or CHYINT not used, a default intercept of 0 is assumed.

Valid only in state 1.

Parameters

x (short floating-point number)
The x-axis intercept.

y (short floating-point number)
The y-axis intercept.

Coding Example

The following example shows how to specify CHXINT:

```
CALL GDDM('CHYINT',5.0)
```

CHXLAB, CHYLAB – Label Text

```
CHXLAB(count,length,text)
CHYLAB(count,length,text)
```

Allow the specification of alphanumeric labels as a sequence of character strings. The labels are placed, in sequence, at major scale marks on the corresponding currently selected axis, starting with the scale mark with the algebraically lowest value. If the number of labels specified is less than the number of major scale marks on the axis, the sequence of labels is reused as necessary until each scale mark is associated with a label.

CHXLAB can also be used for assigning a title to each pie of a pie chart.

These commands implicitly cause calls to CHXSET(ALPHANUMERIC) and CHYSET(ALPHANUMERIC), and cause a logarithmic axis type specification to be ignored.

CHSET(BLABEL) can be used to clear the areas occupied by labels, before they are written.

The label text attributes can be set by CHLATT.

Valid only in state 1.

Parameters

count (4-byte binary integer)
The number of labels supplied. A specification of zero suppresses the labels.

length (4-byte binary integer)
The number of characters associated with each label. All labels must be the same length. Each label is centered on the corresponding axis position (scale mark position, or point midway between scale mark positions). Centering occurs after any trailing null (X'00') characters have been deleted. Hence the effect of varying-length labels can be obtained by using nulls as padding characters to extend each label to the standard length.

text (character string)
A character string that contains all of the labels, in order of use. The total length of the string must not exceed 32,000 characters.

Coding Example

The following example shows how to specify CHXLAB:

```
CALL GDDM('CHXLAB',12,3,'JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVDEC')
```

12 is the number of labels and 3 is the number of characters in each label.

CHXLAT, CHYLAT – Label Attributes

```
CHXLAT(count,array)
CHYLAT(count,array)
```

Establish the default character-appearance attributes to be used in displaying labels for the currently selected axis.

By *default*, the attributes are as follows: The defaults are:

Color	0 (green on displays, black on printers, highest pen number on plotters)
Character mode	3
Symbol-set identifier	0 (system default)
Character-size multiplier	100 (results in the text size as defined by CHCGRD (by default, the size of the hardware characters))
Character height/width multiplier	100 (results in the text size as defined by CHCGRD (by default, the size of the hardware characters))
Note rotation	0

Valid only in state 1.

Parameters

count (4-byte binary integer)
Specifies the number of elements in *array*. If more than six elements are specified, the excess elements are ignored. If fewer than six, the remainder are unchanged from their previous setting. If zero is specified, all six defaults are reinstated.

array (an array of 4-byte binary integers)
An array of, at most, six 4-byte binary integers:

1. Color.

   For valid values of color, see Appendix C, "Colors, Line-Types, Markers, and Shading Patterns."

2. Character mode.

   Valid values are:

   0 system default (mode 3)
   2 mode 2
   3 mode 3

3. Symbol-set identifier.

   For mode 2 or mode 3, the identifier denotes:

   0 the default character set
   n a graphics symbol set that has been previously loaded into GDDM by the application program using the GDDM call GSLSS

4. Character-size multiplier.

   This specification is ignored for character mode 1. For modes 2 and 3, the character-size multiplier is divided by 100, and the horizontal and vertical spacings or sizes specified by CHCGRD (or its defaults) are multiplied by this factor before the characters are drawn. For example, if character-size multiplier = 200 and mode-3 characters are being used, they are drawn twice as large as the basic size set by CHCGRD. The character-size multiplier must have a value greater than zero.

5. Character height/width multiplier.

   Specifies the height of the character box relative to its width. For example, 100 indicates that the height is multiplied by the same amount as the width. 200 causes the height to be multiplied by twice the amount of the width.

6. Axis-label rotation.

   When axis-label rotation is requested, each label is written at the specified angle to the horizontal. Label rotation is indicated by specifying a non-zero value in this element. The value must be from −9000 through +9000. This number is divided by 100 to give a value from −90 through +90, which is the angle, in degrees, to the horizontal at which the labels will be drawn. The default is zero. When positive, the angle is in a counterclockwise direction. When negative, it is in a clockwise direction.

Coding Example

The following example shows how to use CHXLAT:

```
OPTION BASE 1
INTEGER ATTAR
DIM ATTAR(5)
MAT READ ATTAR
DATA 1,3,0,150,100
CALL GDDM('CHPXLAT',5,ATTAR())
```

Differences between System/38 and the AS/400 System

This call is not supported on System/38.

CHXMTH, CHYMTH – Month Labels

```
CHXMTH(month)
CHYMTH(month)
```

Specify English language, uppercase month labels for the currently selected x axis or y axis. The specified month is associated with the first major scale mark position. If CHXSET(NOSKIP), CHYSET(NOSKIP), or default month label skipping is in effect, each successive month is assigned to the next major mark. If CHXSET(SKIP) or CHYSET(SKIP) is specified, then months are assigned to major marks using the same skip factor that was specified on the CHXTIC or CHYTIC call. The labels normally are three-letter abbreviations, but a CHSET option is available to change this (see CHSET(ABREV)).

These calls implicitly set the corresponding label type to DATE.

Valid only in state 1.

Parameters

month (4-byte binary integer)
The month to be associated with the first scale mark. Valid values are 1 through 12. January is month 1.

Coding Example

The following example shows how to specify CHXMTH:

```
CALL GDDM('CHXMTH',1)
```

1 is the number of the month to start with (January).

CHXRNG, CHYRNG – Explicit Ranges

```
CHXRNG(lower-limit,upper-limit)
CHYRNG(lower-limit,upper-limit)
```

Allow explicit specification of the ranges of the currently selected axes. When the range is not specified for a particular axis, or is specified as (n,n) (with lower and upper limit values identical), autoranging applies to the axis. In general, the second parameter is greater than the first, but you can change the direction of the axis by reversing the order of the parameters. Note that logarithmic axis ranges must have positive, nonzero values.

Valid only in state 1.

Parameters

lower-limit,upper-limit (short floating-point numbers)
The data values at the starting point and end point, respectively, of the x axis. For a horizontal x axis, the starting point data value is at the left, and the end point data value at the right. For a vertical x axis, the starting point data value is at the top of the axis, and the end point data value at the bottom of the axis.

lower-limit,upper-limit (short floating-point number)
The data values at the starting point and end point, respectively, of the y axis. For a vertical y axis, the starting point data value is at the bottom, and the end point data value at the top. For a horizontal y axis, the starting point data value is at the left, and the end point data value at the right.

Coding Example

The following example shows how to specify CHXRNG so that the x axis extends from a lower limit of 1 to an upper limit of 150:

```
CALL GDDM('CHXRNG',1.0,150.0)
```

CHXSCL, CHYSCL – Scale Factor

```
CHXSCL(scale-factor)
CHYSCL(scale-factor)
```

Specify a scale factor for numeric labels on the currently selected x axis or y axis. These calls specify that, when labels are generated from numeric values, the values should be multiplied by the scale factor before the labels are generated. This enables superfluous zeros to be suppressed. Scale factors are ignored for alphanumeric and date labeling.

If a scaling factor is specified, it can be useful to include a notation in the axis title describing its use. For example, when the scaling factor is 0.001 and the data is in dollars, the axis title might be *Thousands of Dollars*.

Valid only in state 1.

Parameters

scale-factor (short floating-point number)
A scaling factor for each label to be generated. The value associated with each major scale mark is first multiplied by scale-factor. For example, if the value is 7000 and the factor is 0.001, the resulting label is 7. Scale-factor must have a value greater than zero.

Coding Example

The following example shows how to specify CHXSCL:

```
CALL GDDM('CHXSCL',0.001)
```

CHXSEL, CHYSEL – Axis Selection

```
CHXSEL(axis-number)
CHYSEL(axis-number)
```

Override the default axis selection to select the primary or secondary axis. By default the primary axis is selected.

The selected pair of axes are those against which data is plotted by the plotting routines CHPLOT, CHSURF, CHHIST, and CHBAR.

In addition, CHXSET or CHYSET options, and calls to CHXRNG, CHYRNG, CHXTTL, CHYTTL, CHXTIC, CHYTIC, CHXINT, CHYINT, CHXSCL, CHYSCL, CHXLAB, CHYLAB, CHXMTH, CHYMTH, CHXDAY, and CHYDAY apply to the selected axes.

Valid in state 1 or state 2.

Parameters

axis-number (4-byte binary integer)
The axis number to be selected. The integer must be 1 for the primary axis or 2 for the secondary axis. The default is 1.

Coding Example

The following example shows how to specify CHXSEL:

```
CALL GDDM('CHXSEL',2)
```

CHXSET, CHYSET – Specify Axis Option

```
CHXSET(option)
CHYSET(option)
```

Control the drawing of currently selected axis lines when the chart is constructed.

Note: Only one option can be used with one call. For example, if you want your x axis to be logarithmic and to have no tick marks you need to enter:

```
CALL GDDM ('CHXSET','LOGA')
CALL GDDM ('CHXSET','PLAIN')
```

Valid only in state 1.

Parameters

Option (character)

The string passed must be at least 4 characters long. Any characters over four are ignored, but might enhance the readability of the program.

Function	Option
Axis, drawn or not	**AXIS** NOAXIS
Axis, intercept	**LOWAXIS** MIDDLE HIGHAXIS INTERCEPT
Axis, title position	**ATCENTER** ATEND ATABOVE
Grid, drawn or not	GRID **NOGRID**
Label position	**LABADJACENT** LABMIDDLE NOLAB
Label, skip months	SKIPMONTH **NOSKIPMONTH**
Label, type of	**NUMERIC** ALPHANUMERIC DATE
Scale, linear or logarithmic	**LINEAR** LOGARITHMIC
Tick mark, style	**NTICK** PTICK XTICK PLAIN
Zero forced into autorange	**FORCEZERO** NOFORCEZERO

Options

The groups of CHYSET options are listed below in alphabetic order:

- **'ATCENTER'** | 'ATEND' | 'ATABOVE'

  Control the positioning or justification of the titles, or both, for the currently selected axis.

  ATCENTER specifies that the axis title is centered along the axis.

  ATEND specifies that the axis title is right-justified for a horizontal axis and justified to the top for a vertical axis.

  ATABOVE provides for placing the title above the vertical axis.

  For a horizontal axis, the ATABOVE specification is equivalent to ATEND.

  When both primary and secondary vertical axes are employed, and the titles are positioned above them, overlap may occur, depending on the distance between the axes and the title lengths. ATABOVE may cause the titles to overlap horizontal axis labels and scale marks, if the horizontal axis is at the top of the chart.

  ATABOVE may also cause overlap of the vertical axis labels and scale marks in some situations. The axis title is initially positioned with its baseline above the vertical axis. However, if the height of the axis title has been increased using the CHTATT call such that the title would extend above the top of the chart area, the baseline is moved down to a point where the axis title fits on the chart, thus possibly causing overlap with the vertical axis labels and scale marks.

- **'AXIS'** | 'NOAXIS'

  Control whether the currently selected axis lines are drawn when the chart is constructed.

  AXIS specifies that the axis line is to be drawn.

  NOAXIS specifies that the axis line should not be drawn. The axis title is not suppressed, nor are the scale marks and labels, unless they have been suppressed by other options.

- **'FORCEZERO'** | 'NOFORCEZERO'

  If autoranging applies, these options specify whether the range is to be extended, if necessary, to include zero.

  FORCEZERO specifies that the axis range is to be extended to include zero, if the range of the data does not. This option is ignored for logarithmic axes and for axes whose range is explicitly specified by a CHYRNG call.

  NOFORCEZERO suppresses the forced inclusion of zero in the range.

- 'GRID' | **'NOGRID'**

  Specify whether grid lines are to be constructed.

  CHYSET controls grid lines perpendicular to the y axis. Grid lines are constructed at the same time as their associated axes. Grid lines are positioned on major tick marks, whose positions are controlled by a CHYTIC call. The attributes of the grid lines are controlled by CHGATT.

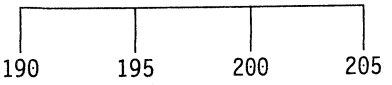
  GRID specifies that grid lines are to be constructed when the axis is constructed.

  NOGRID specifies that grid lines are not required.

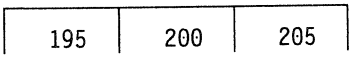
- **'LABADJACENT'** | 'LABMIDDLE' | 'NOLAB'

  Specify label position for the currently selected y axis. These calls control the positioning of labels with respect to the major scale marks.

  LABADJACENT centers the label on the major scale-mark position, thus:



  LABMIDDLE places the label between the major scale mark it is associated with and the *preceding* one. (The label that would normally be adjacent to the first mark on the axis is omitted.) The label is centered on the point midway between the two scale marks as shown below. This option is ignored for logarithmic axes.



  NOLAB specifies that no labels are to be constructed for the corresponding axis.

- **'LINEAR'** | 'LOGARITHMIC'

  Specify whether the axis scale is linear or logarithmic (base 10).

  A logarithmic specification is ignored unless the axis label type is numeric. In particular, a logarithmic specification is ignored if alphanumeric labels are specified by means of CHYLAB.

- **'LOWAXIS'** | 'MIDDLE' | 'HIGHAXIS' | 'INTERCEPT'

  LOWAXIS positions the selected axis at the bottom or left-hand side of the plot area.

  MIDDLE positions the selected axis in the middle of the plot area.

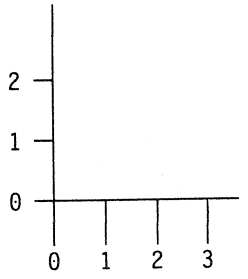
  HIGHAXIS positions the selected axis at the top or right-hand side of the plot area.

  INTERCEPT indicates that the position of the selected axis will be controlled by the CHYINT call referring to the other selected axis. CHYSET is used with CHXINT, and CHXSET with CHYINT. If the appropriate CHYINT (or CHXINT) call is not made, an intercept value of 0 is assumed.

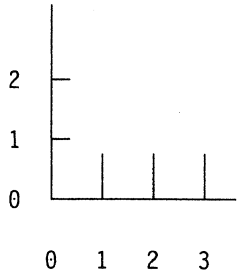
- **'NTICK'** | 'PTICK' | 'XTICK' | 'PLAIN'

  Specify whether tick marks are to be drawn on the axis and, if so, the style of drawing. The options apply to both major and minor scale marks, on the currently selected y axis.

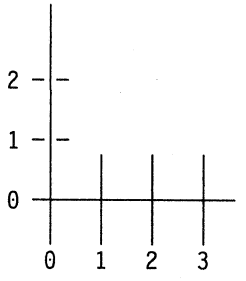
  NTICK draws scale marks from the y axis in the negative direction of the x axis, as shown below. The positive direction of an axis is that defined by movement from the end of the axis with the algebraically lower value to the end with the algebraically higher value. The negative direction is the opposite.



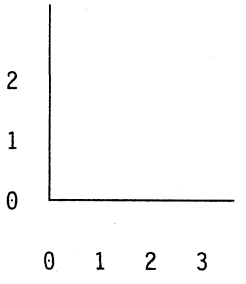
PTICK reverses the scale-mark direction from that described for NTICK:



XTICK draws scale marks perpendicular to the axis and crossing it, projecting equally on each side:



PLAIN indicates that the axis is to be constructed without scale marks:



Note: 'NTICK' is the default for primary axes, and 'PTICK' for secondary axes.

- **'NUMERIC'** | 'ALPHANUMERIC' | 'DATE'

  Specify the labeling method used for the currently selected axis.

  NUMERIC specifies that a label is to be generated automatically for each major scale mark. The label is an EBCDIC representation of the numeric value along the axis defining the scale mark. The representation may be in fixed- or floating-point format. If in fixed-point format, you can control the format using CHSET('NPGFS'), and CHSET('PGFS') calls. You can control scaling by using CHYSCL.

  ALPHANUMERIC specifies that each interval on the axis represents an item described by a label provided by a CHYLAB call. ALPHANUMERIC is implicitly set by a CHYLAB call.

  DATE implies that each interval on the axis is associated with a month of the year or a day of the week as specified by calls to CHYMTH and CHYDAY. DATE is implicitly set by these calls.

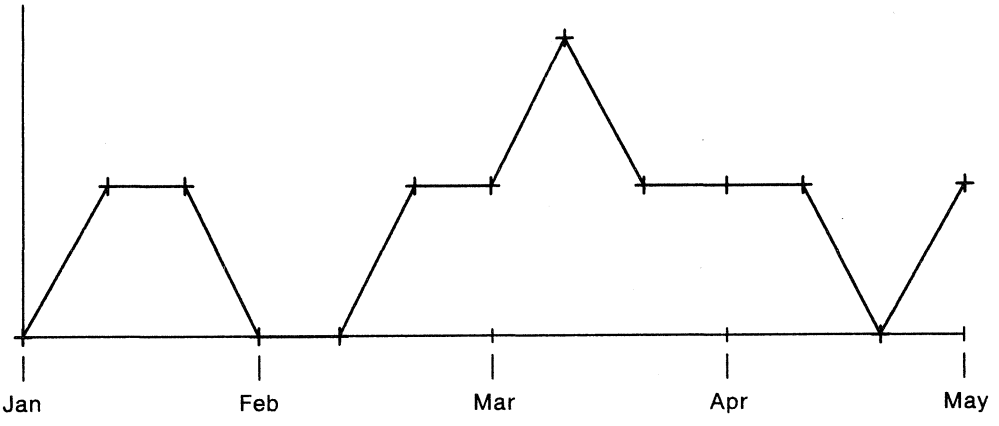
If ALPHANUMERIC or DATE is specified, any logarithmic axis-type specification is ignored; the axis is forced to be linear.

Note that all forms of labels correspond one-for-one with the major scale marks. The spacing of such marks along an axis is controlled by CHYTIC.

- 'SKIPMONTH' | '**NOSKIPMONTH**'

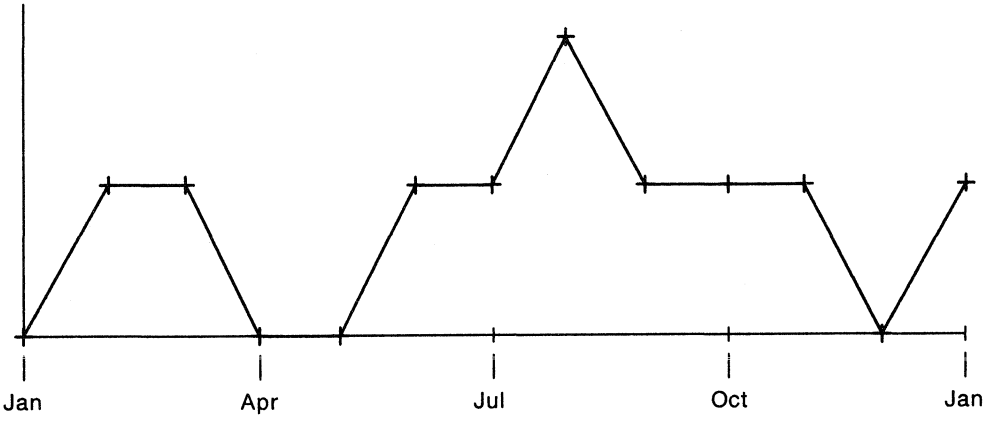
  Specify whether or not month labels are to be skipped.

  NOSKIPMONTH (default) indicates that each successive month label is assigned to the next major tick mark. If CHXTIC(3,0) and CHXMTH(1) were specified with NOSKIPMONTH, the following might be output:



RV2F761-0

SKIPMONTH indicates that month labels correspond to the major tick interval specified on the CHXTIC or CHYTIC call. If CHXTIC(3,0) and CHYMTH(1) were specified with SKIPMONTH, the following might be output:



RV2F762-0

CHXTIC, CHYTIC – Scale Mark Interval

```
CHXTIC(major,minor)
CHYTIC(major,minor)
```

Override the default positions of major and minor scale marks (also known as *tick marks*) on the currently selected x axes and y axes. CHXTIC is ignored for bar charts.

By default, the interval on a linear axis is 1, 2, or 5 multiplied by some power of 10 so as to separate the major marks by approximately 9-character cell widths. Minor marks are never generated for logarithmic axes.

The default for logarithmic axes place a major mark at each power of 10 over the range of the axis; for example: 1, 10, 100, 1000, and so forth. Minor specifications are ignored, because minor marks cannot be generated for logarithmic axes.

The above default actions are taken if no CHXTIC or CHYTIC call is issued.

Valid only in state 1.

Parameters

major

A short floating-point number that specifies the interval between major scale marks. If zero is specified, the default action is taken. Linear and logarithmic axes are treated differently.

Linear axes

Linear axes can have both minor and major scale marks. The major scale marks are placed at the interval specified by the major parameter.

Logarithmic axes

Logarithmic axes, if they have any major scale marks, always have them at powers of 10. CHXTIC and CHYTIC are used to define extra tick marks between the powers of 10.

Only the high-order digit of the parameter has any effect. The high-order digit specifies at which multiples of power of 10 such extra marks should be drawn.

Assume a logarithmic axis with a range of 1 to 100 (remember, there is no 0 on a logarithmic axis). Tick marks would be placed at the points shown in the following table:

CHXTIC or CHYTIC Value	Major Scale Mark Points Assuming an Axis Range of 1 to 100
0	1,10,100
1	1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100
2	1,2,4,6,8,10,20,40,60,80,100
3	1,3,6,9,10,30,60,90,100
4	1,4,8,10,40,80,100
5	1,5,10,50,100
6	1,6,10,60,100
7	1,7,10,70,100
8	1,8,10,80,100
9	1,9,10,90,100
10	1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100

Method of determining logarithmic scale marks: If marks are required on a logarithmic axis, each power of 10 receives a major mark, and additional intermediate major marks are placed at a frequency corresponding to the high-order digit (base 10) of the major interval specification. This correspondence is as follows. If the digit is n, then in the interval between 10**i and 10**(i+1), major marks are placed at each point defined by the following equation:

```
P=j*n*10** i
```

for j = 1,2,3,...9, when (j*n)<10

Each power-of-10 scale mark is labeled in full. Intermediate scale marks are labeled with a digit in the range 2 through 9. This is the multiplication factor that must be applied to the immediately lower power-of-10 label to derive the value of the scale mark.

minor (short floating-point number; must be an integer)
The number of minor marks between major marks. If zero is specified, no minor marks are drawn. Minor marks can be specified even though major marks are automatically generated (when the major parameter is 0). The maximum number of minor marks that can be specified is 2048.

Minor marks are drawn half the length of major marks. Minor marks are never drawn for logarithmic axes, so the parameter is ignored.

Coding Example

The following example shows how to specify CHXTIC for major ticks every 40 intervals and minor ticks every 4 intervals:

```
CALL GDDM('CHXTIC',40,4)
```

CHXTTL, CHYTTL – Axis Title Specification

```
CHXTTL(length,text)
CHYTTL(length,text)
```

Specify titles for the currently selected axes.

To change the appearance of the characters in the titles, see CHTATT.

Valid only in state 1.

Parameters

length (4-byte binary integer)
The number of characters in the title. A length of zero suppresses the title. The length must not exceed 32,000.

text (character string)
Specifies the character string to be used as a title.

Coding Example

The following example shows how to specify CHXTTL and CHYTTL:

```
CALL GDDM('CHXTTL',12,'X axis Title')
CALL GDDM('CHYTTL',12,'Y axis Title')
```

CHY... — y Axis Calls

All CHY... calls are listed under the equivalent CHX... . For example, CHYSCL is described with CHXSCL.

CHY.....	(see equivalent CHX.... call)

Summary of Data Types for Presentation Graphics Routines

Figure 3-3 is a table in which you can look up the parameter data types for the Presentation Graphics routines. For instance, for CHAATT, there are two parameters: the first is a 4-byte binary integer and the second is an array of 4-byte binary integers.

Figure 3-3 (Page 1 of 2). Summary of Data Types for Presentation Graphics Routines

Name	Parm 1	Parm 2	Parm 3	Parm 4	Parm 5
CHAATT	in	in arr			
CHAREA	fp	fp	fp	fp	
CHBAR	in	in	fp arr		
CHBATT	in	in arr			
CHCGRD	in	in			
CHCOL	in	in arr			
CHDATT	in	in arr			
CHDRAX					
CHFINE	in				
CHGAP	fp				
CHGATT	in	in arr			
CHGGAP	fp				
CHHATT	in	in arr			
CHHEAD	in	char			
CHHIST	in	in	fp arr	fp arr	fp arr
CHHMAR	in	in			
CHKATT	in	in arr			
CHKEY	in	in	ch		
CHKEYP	ch	ch	ch		
CHKMAX	in	in			
CHKOFF	fp	fp			
CHLATT	in	in arr			
CHLT	in	in arr			
CHLW	in	fp arr			
CHMARK	in	in arr			
CHNATT	in	in arr			
CHNOFF	fp	fp			
CHNOTE	ch	in	ch		
CHNUM	in				
CHPAT	in	in arr			
CHPCTL	in	fp arr			
CHPEXP	in	fp arr			
CHPIE	in	in	fp arr		
CHPIER	in				

Figure 3-3 (Page 2 of 2). Summary of Data Types for Presentation Graphics Routines

CHPLOT	in	in	fp arr	fp arr	
CHRNIT					
CHSET	ch				
CHSTRT					
CHSURF	in	in	fp arr	fp arr	
CHTATT	in	in arr			
CHTERM					
CHVATT	in	in arr			
CHVCHR	in				
CHVENN	fp	fp	fp		
CHVMAR	in	in			
CHXDAY	in				
CHYDAY	in				
CHXDTM	fp				
CHYDTM	fp				
CHXINT	fp				
CHYINT	fp				
CHXLAB	in	in	ch		
CHYLAB	in	in	ch		
CHXLAT	in	in arr			
CHYLAT	in	in arr			
CHXMTH	in				
CHYMTH	in				
CHXRNG	fp	fp	fp	fp	
CHYRNG	fp	fp	fp	fp	
CHXSCL	fp				
CHYSCL	fp				
CHXSEL	in				
CHYSEL	in				
CHXSET	ch				
CHYSET	ch				
CHXTIC	fp	fp			
CHYTIC	fp	fp			
CHXTTL	in	ch			
CHYTTL	in	ch			

in = 4-byte binary integer
fp = Short floating point number
ch = Character string
arr = Array
var = Variable (whose value is returned by GDDM)

Appendix A. Conversion and Compatibility

Compatibility with the System/370 Computer

GDDM on the AS/400 system is a compatible subset and adaptation of Version 1, Release 3 of the System/370 GDDM licensed program.

The Presentation Graphics routines are a compatible subset of GDDM-PGF (GDDM Presentation Graphics Facility), which is a System/370 GDDM licensed program.

The Application Programming Interface on the AS/400 system is equivalent to the nonreentrant interface on the System/370 computer.

Summary of GDDM Functions

This section (Figure A-1 through Figure A-24) presents a listing, by function call, of the differences between the System/370 computer and the AS/400 system.

In this section, the right column shows changes from the System/370 licensed program:

D: Deleted in the AS/400 version

M: Modified for the AS/400 version

R: Restricted in the AS/400 version

A: Added to the AS/400 version

Control Functions

Figure A-1. Control Functions

Name	Description	AS/400 Support
DSCMF	Control mode function	D
DSQCMF	Query control mode function	D
ESEUDS	Encoded user default specification	D
ESLIB	Identify library	D
ESPCB	Identify PCB (IMS/VS only)	D
ESSUDS	Source user default specification	D
FSEXIT	Specify error exit	M
FSINIT	Initialize graphics	
FSQERR	Query last error	M
FSRNIT	Reinitialize	
FSTERM	End graphics	
FSTRCE	Internal trace control	D
SPINIT	Initialize with SPIB	D

Device Functions

Figure A-2. Device Functions

Name	Description	AS/400 Support
DSCLS	Close device	R
DSDROP	Discontinue device usage	R
DSOPEN	Open device	M
DSQDEV	Query device characteristics	
DSQUID	Query unique device-id	
DSQUSE	Query device usage	
DSRNIT	Reinitialize device	R
DSUSE	Specify device usage	R
FSCLS	Close alternative device	D
FSOPEN	Open alternative device	D
FSQDEV	Query device characteristics	
FSQURY	Query device characteristics	D
GSENAB	Enable or disable a logical input device	D

Input/Output Functions

Figure A-3. Input/Output Functions

Name	Description	AS/400 Support
ASREAD	Device output/input	
FSALRM	Sound device alarm	
FSCHEK	Check picture complexity before output	D
FSFRCE	Update the display	
FSQUPDM	Query update mode	D
FSREST	Retransmit data	R
FSSAVE	Save current picture contents	D
FSSHOR	Extended FSSHOW	D
FSSHOW	Display a saved picture	D
FSUPDM	Set update mode	D
GSREAD	Await graphics input	D
MSREAD	Present mapped data	D

Symbol Set Functions

Figure A-4. Symbol Set Functions

Name	Description	AS/400 Support
ASCSS	Character symbol sets	D
ASFPSS	Define primary symbol set for field	D
GSCPG	Set current code page	D
GSDSS	Load graphics symbol set from program	D
GSLSS	Load graphics symbol set	R
GSQCPG	Query current/associated code page	D
GSQNSS	Query how many loaded symbol sets	
GSQSS	Query number of symbol sets loaded	R
GSQSSD	Query symbol set data	D
GSRSS	Release a graphics symbol set	R
PSDSS	Load alpha symbol set from program	D
PSLSS	Load alpha symbol set from storage	D
PSLSSC	Load alpha symbol set from storage if needed	D
PSQSS	Query device store status	D
PSRSS	Release a symbol set from PS store	D
PSRSV	Reserve or free a PS store	D
SSQF	Query symbol set on auxiliary storage	D
SSREAD	Read a symbol set from auxiliary storage	D
SSWRT	Write symbol set	D

Page Handling Functions

Figure A-5. Page Handling Functions

Name	Description	AS/400 Support
FSPCLR	Clear the current page	
FSPCRT	Create a page	
FSPDEL	Delete a page	
FSPQRY	Query specified page	
FSPSEL	Select a page	
FSPWIN	Set page window	D
FSQCPG	Query current page id	
FSQUPG	Query unique page id	
FSQWIN	Query page window	D
MSPCRT	Create a page for mapping	D

Partition Functions

Figure A-6. Partition Functions

Name	Description	AS/400 Support
PTNCRT	Create a partition	D
PTNDEL	Delete a partition	D
PTNMOD	Modify the current partition	D
PTNSEL	Select a partition	D
PTSCRT	Create a partition set	D
PTSDEL	Delete a partition set	D
PTSSEL	Select a partition set	D

Partition Query Functions

Figure A-7. Partition Query Functions

Name	Description	AS/400 Support
PTNQRY	Query the current partition	D
PTSQRY	Query partition set attributes	D
PTSQUN	Query unique partition set id	D

Mapping Functions

Figure A-8. Mapping Functions

Name	Description	AS/400 Support
MSCPOS	Set cursor position	D
MSDFLD	Create a mapped field	D
MSGET	Get data from mapped field	D
MSPCRT	Create a page for mapping	D
MSPUT	Put data into a mapped field	D

Mapping Query Functions

Figure A-9. Mapping Query Functions

Name	Description	AS/400 Support
MSPQRY	Query current page	D
MSQADS	Query application data structure definition	D
MSQFIT	Query map fit	D
MSQFLD	Query mapped field	D
MSQGRP	Query map group characteristics	D
MSQMAP	Query map characteristics	D
MSQMOD	Query modified fields	D
MSQPOS	Query cursor position	D

Alphanumeric Field Functions

Figure A-10. Alphanumeric Field Functions

Name	Description	AS/400 Support
ASCCOL	Character colors in field	D
ASCGET	Get field contents	D
ASCHLT	Character highlights	D
ASCPUT	Specify field contents	D
ASCSS	Character symbol sets	D
ASDFLD	Define a single field	D
ASDFLT	Set default field attributes	D
ASDFMT	Define multiple fields	D
ASDTRN	Define I/O translation tables	D
ASFBDY	Field outline attributes	D
ASFCLR	Clear fields	D
ASFCOL	Define field color	D
ASFCUR	Position the cursor	D
ASFEND	Define field end attribute	D
ASFHLT	Define field highlighting	D
ASFIN	Define input null-to-blank conversion	D
ASFINT	Define field intensity	D
ASFMOD	Change field status	D
ASFOUT	Define output blank-to-null conversion	D
ASFPSS	Define primary symbol set for a field	D
ASFSEN	Define field shift controls	D
ASFTRA	Define field transparency attribute	D
ASFTRN	Assign translation table set to a field	D
ASFTYP	Define field type	D
ASGGET	Get data from dual character field	D
ASGPUT	Put data in dual character field	D
ASMODE	Define the operator reply mode	D
ASRATT	Define field attributes	D
ASRFMT	Redefine fields	D
ASTYPE	Override alphanumeric character-code assignments	D

Query Alphanumeric Fields

Figure A-11. Query Alphanumeric Fields

Name	Description	AS/400 Support
ASQCOL	Query character colors	D
ASQCUR	Query cursor position	D
ASQFLD	Query field attributes	D
ASQHLT	Query character highlights	D
ASQMAX	Query the number of fields	D
ASQMOD	Query modified fields	D
ASQNMF	Query number of numeric fields	D
ASQSS	Query character symbol set	D

Graphics Field Control

Figure A-12. Graphics Field Control

Name	Description	AS/400 Support
GSCLP	Enable and disable clipping	
GSCLR	Clear the graphics field	
GSFLD	Define the graphics field	
GSPS	Define the picture space	
GSSCLS	Close the current segment	
GSSDEL	Delete a segment	
GSSEG	Create a segment	
GSUWIN	Define a uniform window	D
GSVIEW	Define a viewport	
GSWIN	Define a window	

Graphics Field Character Handling

Figure A-13. Graphics Field Character Handling

Name	Description	AS/400 Support
GSCA	Set current character angle	
GSCB	Set character box size	
GSCD	Set current character direction	
GSCH	Set character shear	
GSCM	Set current character mode	R

Graphics Color, Symbol Set, Line, Marker

Figure A-14. Graphics Color, Symbol Set, Line, Marker

Name	Description	AS/400 Support
GSCOL	Set current color	
GSCS	Set current symbol set identifier	
GSFLW	Set fractional line width	
GSLT	Set current line type	
GSLW	Set current line width	
GSMB	Set current marker box size	D
GSMIX	Set current color mixing mode	R
GSBMIX	Set background mix	D
GSMS	Set current marker symbol	
GSMSC	Set marker scale	
GSPAT	Set current shading pattern	R

Graphics Drawing Functions

Figure A-15. Graphics Drawing Functions

Name	Description	AS/400 Support
GSARC	Draw circular arc	
GSAREA	Start shaded area	
GSCHAP	Draw character string at current position	
GSCHAR	Draw character string at specified point	
GSCP	Set current position	D
GSDEFS	Start drawing defaults definition	D
GSDEFE	End drawing defaults definition	D
GSELPS	Draw an elliptical arc	
GSENDA	End a shaded area	
GSIMG	Draw a graphics image	
GSIMGS	Draw scaled graphics image	
GSLINE	Draw a straight line	
GSMARK	Draw a marker symbol	
GSMOVE	Move without drawing	
GSMRKS	Draw a series of marker symbols	
GSPFLT	Draw a curved fillet	
GSPLNE	Draw a series of lines	
GSVECM	Draw vectors	

Retrieve and Display Graphics Data

Figure A-16. Retrieve and Display Graphics Data

Name	Description	AS/400 Support
GSGET	Retrieve graphics data	
GSGETE	End retrieval of graphics data	
GSGETS	Start retrieval of graphics data	
GSLOAD	Load segments	D
GSPUT	Display retrieved graphics data	
GSSAVE	Segment save	D

Graphics Query Functions

Figure A-17. Graphics Query Functions

Name	Description	AS/400 Support
GSQBMIX	Query background mix	D
GSQCA	Query character angle	
GSQCB	Query character box size	
GSQCD	Query character direction	
GSQCEL	Query hardware cell size	
GSQCH	Query character shear	
GSQCLP	Query the clipping state	
GSQCM	Query the current character mode	
GSQCOL	Query the current color	
GSQCP	Query the current position	
GSQCS	Query the current symbol set	
GSQCUR	Query the cursor position	
GSQFLD	Query graphics field	D
GSQFLW	Query fractional line width	
GSQLT	Query the current line type	
GSQLW	Query the current line width	
GSQMAX	Query the number of segments	
GSQMB	Query marker box size	D
GSQMIX	Query the current color mixing mode	
GSQMS	Query the current marker symbol	
GSQMSC	Query marker scale	
GSQPAT	Query the current shading pattern	
GSQPS	Query the picture space definition	
GSQTB	Query the text box	
GSQVIE	Query the current viewport definition	
GSQWIN	Query the current window definition	

Interactive Graphics Functions

Figure A-18. Interactive Graphics Functions

Name	Description	AS/400 Support
GSCORR	Correlate segment/tag pair	D
GSENAB	Enable/disable logical input device	D
GSFLSH	Clear graphics input queue	D
GSIDVF	Clear data value, float	D
GSIDVI	Clear data value, integer	D
GSILOC	Initialize locator	D
GSIPIK	Initialize pick	D
GSISTK	Initialize stroke	D
GSISTR	Initialize string	D
GSSATI	Set initial segment attributes	D
GSSATS	Modify segment attributes	D
GSSPOS	Set segment position	D
GSTAG	Set current primitive tag	D

Interactive Graphics Query Functions

Figure A-19. Interactive Graphics Query Functions

Name	Description	AS/400 Support
GSQATI	Query initial segment attributes	D
GSQCHO	Query choice device data	D
GSQLID	Query logical input device	D
GSQLOC	Query graphics locator data	D
GSQPIK	Query pick data	D
GSQSIM	Query simultaneous queue entry	D
GSQSTK	Query stroke data	D
GSQSTR	Query string data	D
GSQTAG	Query current tag	D

Copy Functions

Figure A-20. Copy Functions

Name	Description	AS/400 Support
FSCLS	Close alternative device	D
FSCOPY	Send page to alternative device	D
FSLOG	Send character string to alternative device	D
FSLOGC	Send character string with carriage control to alternative device	D
FSOPEN	Open alternative device	D
GSCOPY	Send graphics to alternative device	D
GSARCC	Aspect ratio control on copy	D

Color Table Functions

Figure A-21. Color Table Functions

Name	Description	AS/400 Support
GSCTD	Define color table	A
GSQCTD	Query color table definition	A
GSCT	Select color table	A
GSQCT	Query color table	A
GSDCT	Define a color table	D
GSQUCT	Query currently used color table	D
GSUCT	Use a color table	D

Graphics Segment Functions

Figure A-22. Graphics Segment Functions

Name	Description	AS/400 Support
GSQATS	Query segment attributes	D
GSQPOS	Query segment position	D
GSSAGA	Set all geometric attributes	D
GSSATI	Set initial segment attributes	D
GSSATS	Modify segment attributes	D
GSSCLS	Close the current segment	
GSSCPY	Segment copy	D
GSSDEL	Delete a segment	
GSSEG	Create a segment	
GSSINC	Segment include	
GSSORG	Set segment origin	D
GSSPOS	Set segment position	D
GSSPRI	Segment priority	D
GSSTFM	Set segment priority	D

Graphics Segment Query Functions

Figure A-23. Graphics Segment Query Functions

Name	Description	AS/400 Support
GSQAGA	Query all geometric attributes	D
GSQATI	Query initial segment attributes	D
GSQATS	Query segment attributes	D
GSQMAX	Query the number of segments	D
GSQORG	Query segment origin	D
GSQPOS	Query segment position	D
GSQPRI	Query segment priority	D
GSQTFM	Query segment transform	D

Environment

Figure A-24. Environment

Name	Description	AS/400 Support
FSQSYS	Query systems environment	D

Summary of Presentation Graphics Routines

This section (Figure A-25 through Figure A-34) lists the functions provided by the Presentation Graphics routines, which include all API functions from the System/370 licensed program except those which invoke the GDDM-PGF Interactive Chart Utility. Presentation Graphics routines are all prefixed with CH.

In this section, the right-hand column shows changes from the System/370 licensed program:

D: Deleted in the AS/400 system version

M: Modified for the AS/400 system version

R: Restricted in the AS/400 system version

The modifications are primarily restrictions in the Presentation Graphics routines caused by restrictions in GDDM. For example, on all of the text attributes functions, character types 1 and 2 are not supported because OS/400 Graphics does not support them.

Control Functions

Figure A-25. Control Functions

Name	Description	AS/400 Support
CHSTRT	Reset the processing state	
CHRNIT	Reinitialize Presentation Graphics routines	
CHTERM	Terminate Presentation Graphics routines	

Chart Definition Functions

Chart Layout Specifications

Figure A-26. Chart Layout Specifications

Name	Description	AS/400 Support
CHAREA	Chart area	
CHQARE	Query chart area	D
CHBATT	Set framing box attributes	
CHCGRD	Basic character spacing/size	
CHCNST	Constant outlines around chart data	D
CHHMAR	Horizontal chart margins	
CHVMAR	Vertical chart margins	

Legend Specifications

Figure A-27. Legend Specifications

Name	Description	AS/400 Support
CHKEYP	Legend position	
CHKEY	Legend key labels	
CHKATT	Legend text attributes	
CHXLAT	x-axis label attributes	
CHYLAT	y-axis label attributes	
CHKMAX	Maximum legend width/height	
CHKOFF	Legend offsets	

Note Specifications

Figure A-28. Note Specifications

Name	Description	AS/400 Support
CHNOFF	Offsets for chart note	
CHNOTE	Construct a chart note	
CHCONV	Convert coordinates	D
CHNATT	Note text attributes	R
CHQPOS	Query positional information	D
CHSSEG	Set segment identifier range	D

Chart Heading

Figure A-29. Chart Heading

Name	Description	AS/400 Support
CHHEAD	Heading text	
CHHATT	Heading text attributes	R

Axis Specifications

Figure A-30. Axis Specifications

Name	Description	AS/400 Support
CHXSET	x-axis characteristics	R
CHYSET	y-axis characteristics	R
CHZSET	z-axis option	D
CHXINT	x-axis intercept	
CHYINT	y-axis intercept	
CHXRNG	x-axis explicit range	
CHYRNG	y-axis explicit range	
CHQRNG	Query axis range	D
CHZRNG	z-axis explicit range	D
CHXTIC	x-axis scale mark interval	
CHYTIC	y-axis scale mark interval	
CHZTIC	z-axis scale mark interval	D
CHAATT	Axis line attributes	
CHTATT	Axis title text attributes	R
CHXTAT	x-axis title attributes	D
CHYTAT	y-axis title attributes	D
CHXLAT	x-axis label attributes	
CHYLAT	y-axis label attributes	
CHZLAT	z-axis label attributes	D
CHXTTL	x-axis text specification	
CHYTTL	y-axis text specification	
CHXSCL	x-axis scale factor	
CHYSCL	y-axis scale factor	
CHLATT	Axis label text attributes	R
CHXLAB	x-axis label text	
CHYLAB	y-axis label text	
CHXDLB	x-axis data labels	D
CHZDLB	z-axis data labels	D
CHXMTH	x-axis month labels	
CHYMTH	y-axis month labels	
CHXDAY	x-axis day labels	
CHYDAY	y-axis day labels	

Grid and Datum Lines

Figure A-31. Grid and Datum Lines		
Name	**Description**	**AS/400 Support**
CHGATT	Grid line attributes	
CHXDTM	x datum line	
CHYDTM	y datum line	
CHDATT	Datum line attributes	

Component Attribute Tables

Figure A-32. Component Attribute Tables		
Name	**Description**	**AS/400 Support**
CHCOL	Component color table	
CHLT	Component line type table	
CHMARK	Component marker table	M
CHMKSC	Set marker scale values	D
CHPAT	Component shading pattern table	M
CHLC	Component line color table	D

Chart Appearance Options

Figure A-33. Chart Appearance Options		
Name	**Description**	**AS/400 Support**
CHSET	Set chart appearance option	R

Chart Construction Routines

Chart Routines

Figure A-34. Chart Routines

Name	Description	AS/400 Support
CHFINE	Curve fitting smoothness	
CHDRAX	Specific control of axis drawing	
CHPLOT	Line graphs and scatter plots	
CHSURF	Surface charts	
CHHIST	Histograms	
CHNUM	Set number of chart components	
CHBAR	Bar charts	
CHBARX	Bar charts with numeric x-axis	D
CHGAP	Spacing between bars	
CHGGAP	Spacing between bar groups	
CHZGAP	z-axis bar gap ratio	D
CHVCHR	Number of bar value characters	
CHVATT	Value text attributes	
CHTHRS	Bar value threshold limit	D
CHVDIG	Decimal digits in bar values	D
CHPIE	Pie charts	
CHPIER	Size specification	
CHPCTL	Control pie chart slices	
CHPEXP	Exploded pie chart slices	
CHVENN	Venn diagram	
CHPOLR	Polar charts	D
CHTOWR	Tower charts	D
CHTPRJ	Set projection for tower charts	D
CHDTAB	Plot table chart	D
CHDCTL	Data table control values	D

RCP Codes

This section provides a list of request control parameter (RCP) codes supported by the AS/400 system. RCP codes can be used instead of the function name when calling OS/400 Graphics functions from BASIC, RPG III, and COBOL/400 programming languages. (RCP codes are required only for compatibility with IBM BASIC.) RCP codes, like function names, are passed as parameters on a CALL GDDM statement, but they must be defined as 4-byte binary integers. For example, for GSCHAR in BASIC, the following are equivalent:

```
CALL GDDM('GSCHAR', 50.0, 50.0, 23, 'Sample character string')
```

and

```
CALL GDDM(202114304, 50.0, 50.0, 23, 'Sample character string')
```

Also, RCP codes (as well as the function name) are set by OS/400 Graphics in error records passed to user error exits, or produced in response to FSQERR calls.

RCP Codes for GDDM

The RCP codes for GDDM are:

Name	RCP code	Description
FSEXIT	196608	Specify an error exit and/or error severity
FSQERR	262144	Query last error
FSTERM	201326592	Terminate graphics
FSINIT	201326593	Initialize graphics
FSRNIT	201326594	Reinitialize graphics
DSOPEN	201327104	Open device
DSCLS	201327105	Close device
DSUSE	201327106	Specify device usage
DSDROP	201327107	Discontinue device usage
DSQDEV	201327110	Query device characteristics
DSQUID	201327108	Query unique device-id
DSQUSE	201327109	Query device usage
DSRNIT	201327111	Reinitialize device
FSPCRT	201588736	Create a page
FSPSEL	201588737	Select a page
FSPDEL	201588738	Delete a page
FSPCLR	201588739	Clear the current page
FSPQRY	201588740	Query specified page
FSQCPG	201588741	Query current page-id
GSQNSS	201588994	Query the number of loaded symbol sets
GSQSS	201588995	Query loaded symbol sets
GSLSS	201589504	Load a graphics symbol set
GSRSS	201589761	Release a graphics symbol set
FSQDEV	201590016	Query device characteristics
FSQUPG	201591040	Query unique page-id
FSALRM	201850880	Sound device alarm
FSREST	201853952	Retransmit data
GSFLD	202113024	Define the graphics field
GSPS	202113025	Define the picture space
GSWIN	202113026	Define a window
GSVIEW	202113027	Define a viewport
GSQPS	202113028	Query the picture space definition
GSQVIE	202113029	Query the current viewport definition
GSQWIN	202113030	Query the current window definition
GSQMAX	202113280	Query the number of segments
GSQCUR	202113281	Query the cursor position

Name	RCP code	Description
GSQCEL	202113538	Query hardware cell size
GSCLP	202113539	Enable and disable clipping
GSQCLP	202113540	Query the clipping mode
GSSEG	202113792	Create a segment
GSSCLS	202113793	Close the current segment
GSSDEL	202113794	Delete a segment
GSCLR	202113795	Clear the graphics field
GSMOVE	202114048	Move without drawing
GSLINE	202114049	Draw a straight line
GSPLNE	202114050	Draw a series of lines
GSMARK	202114054	Draw a marker symbol
GSMRKS	202114055	Draw a series of marker symbols
GSAREA	202114056	Start a shaded area
GSENDA	202114057	End a shaded area
GSVECM	202114058	Vectors
GSCHAR	202114304	Draw a character string at a specified point
GSCHAP	202114305	Draw a character string at the current position
GSQTB	202114306	Query the text box
GSARC	202114560	Draw a circular arc
GSELPS	202114561	Draw an elliptic arc
GSPFLT	202114562	Draw a curved fillet
GSQCP	202114816	Query the current position
GSCOL	202114817	Set current color
GSMIX	202114818	Set current color mixing mode
GSLT	202114819	Set current line type
GSLW	202114820	Set current line width
GSCM	202114821	Set current character mode
GSCS	202114822	Set current symbol set
GSCB	202114823	Set current character box size
GSCA	202114824	Set current character angle
GSCD	202114825	Set current character direction
GSPAT	202114826	Set current shading pattern
GSMS	202114827	Set current marker symbol
GSCH	202114828	Set current character shear
GSFLW	202114830	Set the fractional line width
GSQFLW	202114831	Query the current floating point line width
GSQCOL	202114833	Query the current color
GSQMIX	202114834	Query the current color mixing mode
GSQLT	202114835	Query the current line type
GSQLW	202114836	Query the current line width
GSQCM	202114837	Query the current character mode
GSQCS	202114838	Query the current symbol set
GSQCB	202114839	Query the current character box
GSQCA	202114840	Query the current character angle
GSQCD	202114841	Query the current character direction
GSQPAT	202114842	Query the current shading pattern
GSQMS	202114843	Query the current marker symbol
GSQCH	202114844	Query the current character shear
GSMSC	202114845	Set marker scale
GSQMSC	202114846	Query marker scale
GSIMG	202115584	Draw graphics image
GSIMGS	202115588	Draw scaled graphics image
ASREAD	202375168	Device output/input
FSFRCE	202375169	Force output
GSCTD	202117377	Define color table
GSQCTD	202117378	Query color table definition
GSCT	202117379	Select color table
GSQCT	202117380	Query color table

RCP Codes for Presentation Graphics Routines

The RCP codes for the Presentation Graphics routines are:

Name	RCP code	Description
CHTERM	268435712	Terminate Presentation Graphics routines
CHRNIT	268501248	Reinitialize Presentation Graphics routines
CHSET	268566785	Specify chart options
CHKEY	268567041	Legend key labels
CHHEAD	268567042	Heading text
CHMARK	268567297	Component marker table
CHLT	268567298	Component line type table
CHCOL	268567299	Component color table
CHPAT	268567300	Component shading pattern table
CHLW	268567301	Component line width table
CHPEXP	268567302	Control pie chart slices
CHXSET	268567553	Specify x-axis options
CHYSET	268567554	Specify y-axis options
CHXTTL	268567809	x-axis title specification
CHYTTL	268567810	y-axis title specification
CHXLAB	268567811	x-axis label text
CHYLAB	268567812	y-axis label text
CHXSCL	268568071	Set x-axis scale factor
CHYSCL	268568072	Set y-axis scale factor
CHXMTH	268568073	Set x-axis month labels
CHYMTH	268568074	Set y-axis month labels
CHXDAY	268568075	Set x-axis day labels
CHYDAY	268568076	Set y-axis day labels
CHNUM	268568079	Set number of components
CHGAP	268568080	Set spacing between bars
CHGGAP	268568081	Set spacing between bar groups
CHHMAR	268568082	Set horizontal margins
CHVMAR	268568083	Set vertical margins
CHPIER	268568084	Set pie reduction
CHCGRD	268568085	Basic character spacing/size
CHVCHR	268568086	Number of characters in bar values
CHFINE	268568090	Curve fitting smoothness
CHAATT	268568321	Set axis attributes
CHGATT	268568322	Set grid line attributes
CHKEYP	268568577	Set legend base position
CHHATT	268568833	Set heading text attributes
CHTATT	268568834	Set axis title text attributes
CHLATT	268568835	Set axis label text attributes
CHKATT	268568837	Set legend text attributes
CHVATT	268568838	Set values attributes
CHXLAT	268568839	x-axis label attributes
CHYLAT	268568840	y-axis attributes
CHAREA	268569090	Set chart area
CHXRNG	269092353	Set x-axis explicit range
CHYRNG	269092354	Set y-axis explicit range
CHXINT	269092355	Set x-axis intercept
CHYINT	269092356	Set y-axis intercept
CHXTIC	269092357	Set x-axis tick mark interval
CHYTIC	269092358	Set y-axis tick mark interval
CHKOFF	269092376	Set legend offsets
CHKMAX	269092377	Set maximum legend width/height
CHNOFF	269157911	Specify offsets for CHNOTE
CHNATT	269158660	Set note attributes
CHDATT	269158913	Set datum line attributes
CHBATT	269158915	Set framing box attributes
CHBAR	269289985	Plot a bar chart
CHHIST	269289986	Plot a histogram
CHPLOT	269289987	Plot a line graph or scatter plot
CHSURF	269289988	Plot a surface chart
CHVENN	269289989	Plot a Venn diagram
CHPIE	269289990	Plot a pie chart
CHXDTM	269354509	Set translated axis line or datum line
CHYDTM	269354510	Set translated axis line or datum line
CHXSEL	269420545	Select x-axis

Name	RCP code	Description
CHYSEL	269420546	Select y-axis
CHSTRT	269549824	Reset processing state to state 1
CHNOTE	269680896	Annotate chart
CHDRAX	269746432	Specific control of axis drawing
CHPCTL	269811969	Control pie chart slices

Symbol Sets

Symbol sets produced on the System/370 computer by the licensed programs *GDDM Base Image Symbol Editor* (mode-2 image characters) and *GDDM-PGF Vector Symbol Editor* (mode-3 vector characters) are fully compatible with the AS/400 system, as are symbol sets supplied with System/370 GDDM.

To use System/370 symbol sets with OS/400 Graphics, you must transport the source data from the System/370 computer to the AS/400 system, and then execute the CRTGSS (Create graphics symbol set) CL command to convert the symbol set source data into an AS/400 *GSS object type.

For more information on creating graphics symbol sets, refer to the *GDDM Programming Guide*.

Appendix B. GDF Order Descriptions

Graphics data format (GDF) orders are like graphics CALL statements because they can be used to define pictures. Normally, they are generated from calls to GDDM routines, and are used internally by GDDM. One or more GDF orders may be *saved* using the GSGET call, and included among other GDF orders using the GSPUT call.

The information in this appendix will help you to interpret GDF orders created by OS/400 Graphics or created by GDDM on the System/370 computer.

Coordinates and Aspect Ratio in Comment Order

When GDDM creates GDF orders, it prefaces them with a comment order that gives the window coordinate system, (and therefore the aspect ratio) on which it is to be built. For accuracy, the window coordinates are set to the largest possible values. Thus the largest of the window coordinate values will be in the region of 32,000. When a picture is produced in GDF, then read back into GDDM using default (0 through 100) or similar window coordinates, the result is often that all plotted points are outside the window.

To reshow a GDF picture, the window coordinates must be reset to the values in the comment order. The GDF picture can be reshown at any size by varying the size of the viewport in which it is presented. To preserve the aspect ratio of the picture, a GSPS call is required that is based on the coordinate values in the GDF comment.

The comment order containing coordinate information has the format shown in Figure B-1.

Figure B-1. GDF Initial Comment Order Format

Field Length	Field Type	Content	Meaning
1	Character	X'01'	Comment order code
1	Binary integer	LEN	Length of following data
2	Binary integer	TYP	Data type of following fields as follows: • 1: Following fields are 1-byte (not supported on the AS/400 system) • 2: Following fields are 2-byte (the default on the AS/400 system) • 3: Following fields are 4-byte (not supported on the AS/400 system) • 4: Following fields are real (System/370 floating point; not supported on the AS/400 system)
2	Binary integer	XLO	Lowest x value
2	Binary integer	XHI	Highest x value
2	Binary integer	YLO	Lowest y value
2	Binary integer	YHI	Highest y value

To recreate the aspect ratio and window of the original picture, the 2-byte binary integers in the initial comment order must first be converted to floating point numbers and then can be used as follows:

```
CALL GSWIN(XLOFLOAT,XHIFLOAT,YLOFLOAT,YHIFLOAT)

WIDTH  = XHIFLOAT - XLOFLOAT
HEIGHT = YHIFLOAT - YLOFLOAT
IF (WIDTH > HEIGHT) THEN
  CALL GSPS(1.0,HEIGHT/WIDTH)
ELSE
  CALL GSPS(WIDTH/HEIGHT,1.0)
```

Not all comment orders found in a GDF file contain the coordinate system information. For the format of comment orders used for other purposes, see the description of the comment order on page B-17.

List of GDF Orders

Figure B-2 shows the GDF orders in alphabetical order as they are described in this appendix. Figure B-3 shows the GDF orders in the order of their code values and should be of use for those who are interpreting the orders.

Figure B-2. Alphabetical Summary of GDF Orders

Order Code	Hex Value	Page Number
Arc	C6 or 86	B-10
Arc parameters	22	B-10
Area end	60	B-11
Area	68	B-11
Character	C3 0r 83	B-14
Character mode	39	B-13
Character set	38	B-14
Character box	33	B-12
Character angle	34	B-12
Character shear	35	B-15
Character direction	3A	B-13
Color	0A	B-15
Color mix	0C	B-16
Color set extended	26	B-16
Comment	01	B-17
Fillet	C5 or 85	B-17
Graphics image begin	D1 or 91	B-18
Graphics image data	92	B-19
Graphics image end	93	B-19
Line	C1 or 81	B-20
Line relative	E1 or A1	B-21
Line type	18	B-21
Line width	19	B-22
Marker	C2 or 82	B-22
Marker type	29	B-23
Pattern	28	B-23
Segment attribute	72	B-24
Segment start	70	B-24
Segment close	71	B-25
Set tag	43	B-25

Figure B-3. Summary of GDF Orders in Order of Code Values

Hex Value	Order Code	Page Number
01	Comment	B-17
0A	Color	B-15
0C	Color mix	B-16
18	Line type	B-21
19	Line width	B-22
22	Arc parameters	B-10
26	Color set extended	B-16
28	Pattern	B-23
29	Marker type	B-22
33	Character box	B-12
34	Character angle	B-12
35	Character shear	B-15
38	Character set	B-14
39	Character mode	B-13
3A	Character direction	B-13
43	Set tag	B-25
60	Area end	B-11
68	Area	B-11
70	Segment start	B-24
71	Segment close	B-25
72	Segment attribute	B-24
81	Line at current position	B-20
82	Marker at current position	B-22
83	Character at current position	B-14
85	Fillet at current position	B-17
86	Arc at current position	B-10
91	Graphics image begin	B-18
92	Graphics image data	B-19
93	Graphics image end	B-19
A1	Line relative, current position	B-21
C1	Line	B-20
C2	Marker	B-22
C3	Character	B-14
C5	Fillet	B-17
C6	Arc	B-10
D1	Graphics image begin	B-18
E1	Line relative	B-21

The following table shows which GDF orders are generated with the corresponding device:

Figure B-4. GDF Orders Generated by Device

Order	Graphics Work Station	L79A3 Token	Plotter	SCS Printer	IPDS Printer
01 – Comment order	X	X	X	X	X
0C – Color mix	X	X	X	X	X
18 – Line type	X	X	X	X	X
19 – Line width	X	X	X	X	X
22 – Arc parameters	_	X	_	_	_
26 – Color, set extended	X	X	X	X	X
28 – Pattern	X	X	X	X	X
33 – Character box	_	X	_	_	X
34 – Character angle	_	X	_	_	X
35 – Character shear	_	X	_	_	X
38 – Character set	_	X	_	_	X
39 – Character mode	_	X	_	_	X
3A – Character direction	_	X	_	_	X
43 – Set tag	X	X	X	X	X
60 – Area end	_	X	_	X	X
68 – Area	_	X	_	X	X
70 – Segment start	X	X	X	X	X
71 – Segment end	X	X	X	X	X
72 – Segment attribute	X	X	X	X	_
81 – Line at current position	X	X	X	X	X
83 – Character at current position	_	X	_	_	X
86 – Arc at current position	_	X	_	_	_
91 – Graphics image begin	X	X	_	X	X
92 – Graphics data	X	X	_	X	X
93 – Graphics image end	X	X	_	X	X
C1 – Line	X	X	X	X	X

The following table shows the GDF orders that are accepted by AS/400 GDDM but not generated. The table also shows whether AS/400 GDDM performs processing when the order is accepted or whether the order is simply ignored.

These orders are not described in this appendix. For information on these orders, refer to the documentation for the product that was used to generate the GDF stream that contains the orders.

Figure B-5 (Page 1 of 2). GDF Orders Accepted but Not Generated

Order	Processed	Ignored
02 – Process specific control	_	X
03 – Push and set character box	_	X
04 – Segment characteristics	_	X
08 – Pattern set	_	X
09 – Push and set pattern	_	X
0A – Color	X	_
0D – Background mix	X	_
11 – Fractional line width	_	X
21 – Current position	X	_
23 – Push and set pick (tag) identifier	_	X
24 – Model transform	_	X
29 – Marker type	X	_
37 – Marker cell	_	X
3B – Marker precision	_	X
3C – Marker set	_	X
3E – Segment end prolog	_	X
3F – Pop attribute	_	X
41 – Marker scale	_	X
4A – Push and set color	_	X
4C – Push and set mix	_	X
51 – Push and set fractional line width	_	X
53 – Segment position	_	X
58 – Push and set line type	_	X
59 – Push and set line width	_	X
62 – Push and set arc parameters	_	X
64 – Push and set model transform	_	X
66 – Push and set extended color	_	X
69 – Push and set marker type	_	X
73 – Segment attribute modify	_	X
74 – Push and set character angle	_	X
75 – Push and set character shear	_	X
78 – Push and set character set	_	X

Figure B-5 (Page 2 of 2). GDF Orders Accepted but Not Generated

Order	Processed	Ignored
79 – Push and set character mode	_	X
7A – Push and set character direction	_	X
82 – Marker at current position	X	_
85 – Fillet at current position	X	_
87 – Full arc at current position	X	_
A1 – Relative line at current position	X	_
C2 – Marker	X	_
C3 – Character	X	_
C5 – Fillet	X	_
C6 – Arc	X	_
C7 – Full arc	X	_
D1 – Graphics image	X	_
E1 – Relative line	X	_

Orders Supported by the AS/400 System

All currently defined GDF orders are accepted by the AS/400 system, but not all are supported. In particular, those orders dealing with segment attributes are not supported.

Orders that are not supported by the AS/400 system or orders that contain parameter values that are not supported are ignored.

General Structure

A GDF file consists of a sequence of orders.

Each order is identified by a 1-byte order code and contains one or more bytes of operand data.

Order Formats

The order is represented in one of two formats depending on the length of the operand data.

The first format applies to orders with up to 255 bytes of operand data. The second applies only to orders that have a single byte of operand data.

Normal Format

In the normal format, there is a 1-byte order code and a 1-byte length field, followed by *length* bytes of operand data:

order code	length (can be zero)	operand data

Short Format

In the short format, the length field is not present and there is a single byte of data:

order code	operand data

In all short format orders, the first hexadecimal digit must be less than 8, and the second must be 8 or greater.

Padding

Orders can be followed by padding bytes (X'00') to align the next order on a convenient boundary.

Coordinate Data

Many of the orders contain coordinate data or coordinate-related data. On the AS/400 system, these coordinates are always represented as 2-byte binary integers; that is, they are normal 15-bit numbers with sign. If negative, they are in twos-complement notation.

The type and length of coordinates must be specified on the GSPUT call, and for the AS/400 system must always be specified as 2-byte binary integers.

Primitives

The following graphics primitives can be represented:

- Line (relative or absolute)
- Marker
- Character string
- Curved fillet
- Arc (circular, elliptical, or full)
- Graphics image

The orders correspond closely with many of the API functions.

Current Position

The GDF order formats given below contain all relevant coordinates. However, for brevity the start position of the graphic primitive can be omitted. If it is omitted, current position is used in its place.

Current position is set by each of the orders. It is set to the end point of a line or arc and, except in the case of character strings, the rule is the same as for the corresponding API function.

The presence or absence of the initial coordinate is indicated by bit 1 of the order code. If it is 0, the first coordinate in the order is omitted, and the current position is assumed in its place.

Coordinate Lengths

In the following primitives, coordinates are assumed to be pairs of 2-byte binary integers. This is the only type accepted by the AS/400 system. Coordinate and coordinate-related fields are marked by *.

Attributes

As with the equivalent CALL statements, attribute setting orders change the current values of the attributes. An attribute setting applies to all subsequent primitives (to which it is relevant) until a new setting is made.

Attribute setting orders appearing in a GDF string argument to GSPUT affect the current attribute settings after the call. The effects are not purely local to primitives within the string, but may affect subsequent primitives.

The details of the effects of the order are not given in full. For more explanation, see the corresponding CALL statement description.

Format of Examples

The examples are given in hexadecimal with 2-byte binary integer coordinates. Blanks in the hexadecimal strings are to aid readability. They have no other significance.

Orders

The following section describes GDF orders that are generated or accepted and processed by the AS/400 system. Orders that are accepted *and* ignored are not described here.

Arc Order

Format:

Field Length	Content	Meaning
1	X'C6' or X'86'	Arc order code
1	LEN	Length of following data
2*	X0	x coordinate of start of arc
2*	Y0	y coordinate of start of arc
2*	X1	x coordinate of intermediate point
2*	Y1	y coordinate of intermediate point
2*	X2	x coordinate of end of arc
2*	Y2	y coordinate of end of arc

This order constructs an arc starting at (X0,Y0) and passing through (X1,Y1) and ending at point (X2,Y2).

The intermediate point (X1,Y1) should, for greatest accuracy, lie midway along the arc. (If it coincides with either end point, the arc becomes undefined.) The initial point and the final point must not coincide.

The arc may be part of a circle or part of the ellipse defined by the previous *set arc parameters* order.

The initial coordinate pair (X0,Y0) may be omitted. The current position is then used as the starting point of the arc and the order code becomes X'86'.

The current position is set to point (X2,Y2).

Set Arc Parameters Order

Format:

Field Length	Content	Meaning
1	X'22'	Set Arc Parameters order code
1	LEN	Length of following data
2*	P	x coordinate of major axis end
2*	Q	y coordinate of minor axis end
2*	R	x coordinate of major axis end
2*	S	y coordinate of minor axis end

The order determines the shape of subsequent arcs. The full parameters give a transformation that maps the unit circle to the required ellipse:

```
x' = Px + Ry
y' = Sx + Qy
```

A circle results if P=Q and R=S=0.

If P equals a and Q equals b, an ellipse results. The axis parallel to the x-axis has a length proportional to 'a'; the axis parallel to the y-axis has a length proportional to 'b'.

If R and S are nonzero, the ellipse is tilted. In general, for an ellipse with major and minor axes proportional to 'a' and 'b', tilted at angle 'theta' to the x-axis:

```
P = a*cos(theta)
Q = b*cos(theta)
R = -b*sin(theta)
S = a*sin(theta)
```

Area End Order

The Area End order has the same meaning as the Area order with the area-end bit set. GDDM accepts both the Area order and the Area End order, but generates only the Area End order.

Format:

Field Length	Content	Meaning
1	X'60'	Area end order code
1	LEN	Length of following data
n	X'00'	Reserved (must be all X'00')

Area Order

The Area order corresponds approximately to the GSAREA and GSENDA functions. See "GSAREA — Start a Shaded Area" on page 2-45 and "GSENDA — End a Shaded Area" on page 2-74 for more details.

Format:

Field Length	Content	Meaning
1	X'68'	Area order code
1	FLAGS	Bit 0: • On if this marks the start of an area • Off if this order marks the end of an area (not generated on the AS/400 system) Bit 1: • On if the boundary lines are to be drawn

Examples:

```
8 80
C1 14 0000 0000  0005 0000  0005 0005  0000 0005  0000 0000
60 00
```

Draws a rectangular area 5 units square. Boundary lines are not drawn.

```
68 C0
C1 14 0000 0000  0005 0000  0005 0005  0000 0005  0000 0000
60 00
```

Draws the same area, but includes boundary lines.

Character Angle Order

The Character Angle order corresponds to the GSCA function. It controls the angle of subsequent character strings.

Format:

Field Length	Content	Meaning
1	X'34'	Character angle order code
1	LEN	Length of following data
2*	AX	x coordinate of a point that defines the angle of the text
2*	AY	y coordinate of the point

AX and AY specify a relative vector that defines the angle of the baseline of the string. If the coordinate (X,Y) is on the baseline, (X + AX, Y + AY) is also on the baseline. (See "GSCA – Set Current Character Angle" on page 2-48.)

Character Box Order

The Character Box order corresponds to the GSCB function. It controls the size of subsequent characters. Refer to GSCB for more details.

Format:

Field Length	Content	Meaning
1	X'33'	Character box order code
1	LEN	Length of following data
2*	CHARWIDTH	Integer width of character box
2*	CHARHEIGHT	Integer height of character box
2*	FRACTWIDTH	Fractional portion of character box width, specified as multiples of 1/65536 for 2-byte format.
2*	FRACTDEPTH	Fractional portion of character box depth, specified as multiples of 1/65536 for 2-byte format.

If either the fractional width or depth is to be specified, both must be specified. The integer and fractional widths (depths) together form a character width (depth) that defines the character box required.

The order specifies the size of characters in following character strings. (See "GSCB – Set Current Character Box Size" on page 2-50.)

Character Direction Order

The Character Direction order corresponds to the GSCD function. See "GSCD – Set Current Character Direction" on page 2-53 for further details.

Format:

Field Length	Content	Meaning
1	X'3A'	Character direction order code.
1	DIRECTION	Value for character direction. These are interpreted as follows: X'00': Default X'01': Left to right X'02': Top to bottom X'03': Right to left X'04': Bottom to top

The Character Direction order gives the placement of each character relative to the previous one, either along or perpendicular to the baseline.

Text Attributes

Character Mode Order

The Character Mode order corresponds to the GSCM function. Refer to "GSCM – Set Current Character Mode" on page 2-64 for more details.

Format:

Field Length	Content	Meaning
1	X'39'	Character mode order code.
1	MODE	Value for character mode attribute. These are interpreted as follows: X'00': Default X'01': String precision (not valid on the AS/400 system; will leave the character mode unchanged) X'02': Character precision (image) X'03': Stroke precision (vector) Other: Undefined

Character Order

The Character order corresponds to the GSCHAR and GSCHAP functions.

Format:

Field Length	Content	Meaning
1	X'C3' or X'83'	Character order code
1	LEN	Length of following data
2*	X0	x coordinate at which character string is to be placed
2*	Y0	y coordinate of character string
V	STRING	EBCDIC character code of each character in the string; all characters above and including X'40' are valid

The character string is placed at the indicated coordinate. The attributes of the string (for example, mode, size, angle) are taken from the current values.

If the character string has an odd length, the length field in the order will contain an odd number. The order must be padded with padding characters to an even number of bytes.

The position (X0,Y0) may be omitted, in which case the order code becomes X'83' and the string is placed at the current position.

Current position is not changed. (This is different from GSCHAR.)

Examples:

```
C3 08 0002 0003 C1C2C3C4
```

Draws the character string 'ABCD' at coordinate (2,3).

```
83 03 C5C6C7
```

Draws the character string 'EFG' at the current position.

Character Set Order

The Character Set order corresponds to the GSCS function.

Format:

Field Length	Content	Meaning
1	X'38'	Character set order code.
1	LCID	Local identifier for the character set: X'00': Default X'41' through X'DF': User-defined set Other: Undefined

Character Shear Order

The Character Shear order corresponds to the GSCH function. It controls the shear of subsequent characters.

Format:

Field Length	Content	Meaning
1	X'35'	Character shear order code.
1	LEN	Length of following data.
2*	HX	HX and HY specify a relative vector that defines the angle at which characters are to be sheared.
2*	HY	y increment; see above.

HX and HY specify a vector that defines the angle of the upright strokes of a character relative to the baseline. If the lower-left corner of a character is placed at (0,0) and the character baseline lies along the x-axis, the line from (0,0) to (HX,HY) gives the direction of upright strokes. (See "GSCH – Set Current Character Shear" on page 2-55.)

Color Order

The Color order corresponds approximately to the GSCOL function. Refer to "GSCOL – Set Current Color" on page 2-66 for more details.

Format:

Field Length	Content	Meaning
1	X'0A'	Color order code
1	COLOR	Value for color attribute The value is an index to a color table that has the following default values: X'00': Default X'01': Blue X'02': Red X'03': Pink X'04': Green X'05': Turquoise X'06': Yellow X'07': Neutral (white on displays, black on printers and plotters) X'08': Background (black on displays, white on printers and plotters) Other: Undefined

All subsequent primitives have the color given until this is reset.

Color Set Extended Order

The existing GSCOL call and the 1-byte Color order are mapped to the 2-byte Color Set Extended order as follows (the existing Color order is not generated).

Format:

Field Length	Content	Meaning
1	X'26'	Color set extended order code
2	COLOR	Value for color attribute The value is an index to a color table that has the following values: X'0000': Green X'FF00': Green X'FF01': Blue X'FF02': Red X'FF03': Pink X'FF04': Green X'FF05': Turquoise X'FF06': Yellow X'FF07': Neutral X'FF08': Background (black on displays, white on printers and plotters) X'0009' through X'7FFF' The same color definitions as X'FF01' through X'FF08' are repeated.

All subsequent primitives have the color given until this is reset.

Color Mix Order

The Color Mix order corresponds to the GSMIX function. Refer to "GSMIX—Set Current Color-Mixing Mode" on page 2-96 for more details.

Format:

Field Length	Content	Meaning
1	X'0C'	Mix order code.
1	MODE	Value for mix mode attribute. Mix mode controls how an inserted primitive affects the existing picture. In all modes, generated 0 bits leave the underlying features untouched; new 1 bits become whatever the current color is if that bit was previously of background color. The effect of a new 1 bit over an existing 1 bit depends upon the particular mode: either the old color (underpaint), or the new color (overpaint), or a mixture (mix or exclusive OR) will result. X'00': Default X'01': Mix (OR) X'02': Overpaint X'03': Underpaint X'04': Exclusive (OR) Other: Undefined

Comment Order

Format:

Field Length	Content	Meaning
1	X'01'	Comment order code.
1	LEN	Length of following data.
V	ANYDATA	The data is ignored.

The Comment order can be used to embed user data in a GDF file. The contents are ignored by GDDM.

The first order output by GSGET is a Comment order whose data field gives the coordinate range used in the file and the coordinate length. Comment orders are occasionally generated by GDDM to pad GDF strings to a known length.

Fillet Order

The Fillet order corresponds approximately to the GSPFLT function.

Format:

Field Length	Content	Meaning
1	X'C5' or X'85'	Fillet order code
1	LEN	Length of following data
2*	X0	x coordinate of line start
2*	Y0	y coordinate of line start
2*	X1	x coordinate of first line end
2*	Y1	y coordinate of first line end
2*	X2	x coordinate of second line end
2*	Y2	y coordinate of second line end

The order shown generates a single fillet. Further coordinate pairs may be added to form a polyfillet. The points are joined in order by imaginary straight lines. A curve is then fitted to the lines as follows. The curve is tangential to the first line at its starting point and to the last line at its end point. If there are intermediate lines, the curve is tangential to these lines at their center points.

In the special case when only two points are supplied, a straight line results.

The initial coordinate pair (X0,Y0) may be omitted. Current position is then used as the starting point of the arc, and the order code becomes X'85'.

Current position is set to the last point specified.

Examples:

```
C5 08 0002 0003  0004 0006
```

Draws a line from coordinate (2,3) to coordinate (4,6).

```
C5 0C 0000 0000  0004 0000  0004 0004
```

Draws a curve, beginning at coordinate (0,0) and tangential to the line from (0,0) to (4,0). Initially the curve is horizontal. The curve then takes an approximately circular path to meet the line from (4,0) to (4,4) at (4,4).

Graphics Image Order — Begin

The Begin Graphics Image order, together with the Graphics Image Data and End Graphics Image orders, corresponds approximately to the GSIMG and GSIMGS functions. Refer to "GSIMG — Draw a Graphics Image" on page 2-84 and "GSIMGS — Draw Scaled Graphics Image" on page 2-87 for more details.

Format:

Field Length	Content	Meaning
1	X'D1' or X'91'	Begin Graphics Image order code Begin Graphics Image (at current position) order code
1	LEN	Length of following data
2*	X0	The x position at which the graphics image is to be placed
2*	Y0	The y position at which the graphics image is to be placed
2	FORMAT	The format of the graphics image data; this field must have the value 0
2	WIDTH	The width of the graphics image in pixels
2	DEPTH	The depth of the graphics image in pixels
2*	IMAGEWIDTH	The desired width of the graphics image in coordinate units
2*	IMAGEDEPTH	The desired depth of the graphics image in coordinate units

A graphics image consists of a rectangular array of pixels.

It is represented by a sequence of orders. The first is a Begin Graphics Image order and the last is an End Graphics Image order (see "Graphics Image Order — End" on page B-19). Between these delimiters, a number of Graphics Image Data orders may occur, giving the array of pixels in the graphics image.

The size of the pixel array and its representation are given by the Begin Graphics Image order. The fields IMAGEWIDTH and IMAGEDEPTH are optional and may be either both specified, or both omitted. When specified, the graphics image will be scaled to fill the area identified by the fields, using the rules defined in the GSIMGS call. When omitted, each pixel is represented by one bit in the pixel array.

The position of the graphics image (top left corner for format 0) is optional.

Graphics Image Order – Data

Format:

Field Length	Content	Meaning
1	X'92'	Graphics Image Data order code
1	LEN	Length of following data
V	PIXELDATA	The pixels of the graphics image

For graphics image FORMAT 0, each Graphics Image Data order contains the pixels for one row of the pixel array. Thus for a graphics image with a DEPTH of N there will be N Graphics Image Data orders between the Begin and End Graphics Image orders. Each Graphics Image Data order contains data for WIDTH pixels. Each pixel is represented by a single bit. If the bit is one, the pixel is *on*; if the bit is zero, the pixel is *off*.

Example:

```
91 06 0000 0009 0004
92 02 FF80
92 02 8080
92 02 8080
92 02 FF80
93 02 0000
```

Draws a graphics image whose size is nine display points wide by four deep at current position. The graphics image consists of a small square of *on* pixels (which will appear in the current color) surrounding an *off* center.

Graphics Image Order – End

Format:

Field Length	Content	Meaning
1	X'93'	End Graphics Image order code
1	LEN	Length of following data
2	RESERVED	

This order ends the construction of a graphics image.

Line Order

The Line order corresponds approximately to the GSLINE and GSPLNE functions.

Format:

Field Length	Content	Meaning
1	X'C1' or X'81'	Line order code
1	LEN	Length of following data
2*	X0	x coordinate of line start
2*	Y0	y coordinate of line start
2*	X1	x coordinate of first line end
2*	Y1	y coordinate of first line end

A line is drawn from the first coordinate given (X0,Y0) to the second (X1,Y1).

The order shown is for a single line, but in general any number of coordinates can be present. Consecutive coordinates in the order will be joined by straight lines. The data length must be an even multiple of the coordinate length.

The initial coordinate pair (X0,Y0) may be omitted. Current position is then used as the starting point of the first line, and the order code becomes X'81'.

Current position is set to the last point specified.

Note that a line order with only an initial position is permitted. This serves only to move current position.

Examples

```
C1 08 0002 0003  0004 0006
```

Draws a line from coordinate (2,3) to coordinate (4,6).

```
C1 0C 0002 0003  0004 0006  0009 0009
```

Draws a line from coordinate (2,3) to coordinate (4,6) and a line from (4,6) to (9,9).

```
C1 04 0002 0003
```

Draws no lines. However, current position is changed to the last coordinate (2,3).

```
81 04 0004 0006
```

Draws a line from current position to point (4,6). Thus, the pair of orders:

```
C1 04 0002 0003
81 04 0004 0006
```

has the same effect as the first:

```
C1 08 0002 0003  0004 0006
```

Line Relative Order

The Line Relative order defines one or more straight lines.

Format:

Field Length	Content	Meaning
1	X'E1' or X'A1'	Line Relative order code Line Relative (at current position) order code
1	LEN	Length of following data
2*	X0	x coordinate of line start
2*	Y0	y coordinate of line start
1	X1	x coordinate of first line end
1	Y1	y coordinate of first line end

The data length must be an even multiple of the coordinate length.

X1 and Y1 are relative to the start position (if X'E1') or current position (if X'A1'). They are 1-byte, signed values that are added/subtracted from the 1-byte x and y positions to determine the line end.

Line Type Order

The Line Type order corresponds to the GSLT function.

Format:

Field Length	Content	Meaning
1	X'18'	Line Type order code.
1	LINETYPE	Value for line type attribute. The value is an index into a notional line-type table as follows: X'00': Default X'01': Dotted line X'02': Short dashed line X'03': Dash-dot line X'04': Double dotted line X'05': Long dashed line X'06': Dash-double-dot line X'07': Solid line X'08': Invisible line Other: Undefined

Line Width Order

The Line Width order corresponds to the GSLW function.

Format:

Field Length	Content	Meaning
1	X'19'	Line Width order code.
1	LINEWIDTH	Value for line width attribute. The value is an index into a notional table of line-widths as follows: X'00': Default X'01': Normal line X'02': Thick line Other: Undefined

Marker Order

The Marker order corresponds approximately to the GSMRKS function.

Format:

Field Length	Content	Meaning
1	X'C2' or X'82'	Marker order code
1	LEN	Length of following data
2*	X0	x coordinate of marker
2*	Y0	y coordinate of marker

The order is shown for a single marker. Further coordinate pairs may be added. The current marker is placed at each point specified.

The first (or only) coordinate pair may be omitted. The order code then becomes X'82', and a marker is placed at current position in addition to any points specified.

Current position is set to the last coordinate specified. If no coordinate has been specified, current position is unchanged.

Examples:

```
C2 04 0002 0003
```

Draws the current marker at coordinate (2,3).

```
C2 0C 0002 0003  0004 0006  0009 0009
```

Draws markers at (2,3) (4,6) and (9,9).

```
82 00
```

Draws the current marker at current position.

Marker Type Order

The Marker Type order corresponds to the GSMS function.

Format:

Field Length	Content	Meaning
1	X'29'	Marker Type order code.
1	N	Marker number. The attribute determines which symbol is displayed by the marker primitive. The interpretation of the values is as follows: X'00': Default X'01': Cross X'02': Plus X'03': Diamond X'04': Square X'05': 6-point star X'06': 8-point star X'07': Filled diamond X'08': Filled square X'09': Dot X'10': Small circle X'41'–X'EF': User defined Other: Undefined

Pattern Order

The Pattern order corresponds to the GSPAT function.

Format:

Field Length	Content	Meaning
1	X'28'	Pattern order code.
1	PATTERN	Value for pattern attribute. This attribute determines which pattern is to be used to shade the interior of subsequent areas. The interpretation of the values is as follows: X'00': Default X'01' through X'08': Density 1 to density 8 (decreasing) X'09': Vertical lines X'0A': Horizontal lines X'0B': Diagonal lines 1 (bottom-left to top-right) X'0C': Diagonal lines 2 (bottom-left to top-right) X'0D': Diagonal lines 1 (top-left to bottom-right) X'0E': Diagonal lines 2 (top-left to bottom-right) X'0F': No shading X'10': Solid shading Other: Undefined

Segment Attribute Order

The Segment Attribute order sets the attributes that will be assigned to subsequently generated segments.

Format:

Field Length	Content	Meaning
1	X'72'	Segment Attribute order code.
1	LEN	Length of following data.
1	ATTRIBUTE	The attributes that can be set for subsequent segments are: X'01': Detectability X'02': Visibility X'03': Highlighting X'04': Transformability X'05': Stored/nonstored state
1	VALUE	The values that can be assigned to the specified attribute are: X'00': Not detectable, invisible, not highlighted, or stored X'01': Detectable, visible, highlighted, nontransformable, or nonstored X'02': Transformable

Segment Start Order

Format:

Field Length	Content	Meaning
1	X'70'	Segment Start order code
1	LEN	Length of following data
4	SEGMENT ID	The identifier for to the following segment, or 0 if unnamed
2	FLAGS	Bit 0 0 = visible 1 = invisible 1 Reserved, must be 1 2 0 = nondetectable 1 = detectable 3 Reserved, must be 1 4 0 = no highlighting 1 = highlighting 5 Reserved, must be 1 6-10 Reserved, must be 0 11 0 = no prolog 1 = prolog 12 0 = nontransformable 1 = transformable 13-15 Reserved, must be 0
2	LEN	Length of segment
4	X'0000'	Reserved, must be 0

GDDM returns the length of a fixed-point GDF segment in the Segment Start order retrieved using GSGET. The length of the segment is ignored on GSPUT; segments must be closed by an explicit Segment Close order.

The order corresponds to the GSSEG function. Refer to "GSSEG – Create a Segment" on page 2-141 for more details.

Segment Close Order

The Segment Close order corresponds to the GSSCLS function.

Format:

Field Length	Content	Meaning
1	X'71'	Segment Close order code
1	LEN	0, no data

Set Tag Order

The Set Tag order is used to set the current value of the tag. All output primitives that follow this order and precede the next Set Tag order are given the tag value specified.

This order is ignored on the AS/400 system.

Format:

Field Length	Content	Meaning
1	X'43'	Set Tag order code
1	4	Length of following data
4	TAG	Tag value

Appendix C. Colors, Line-Types, Markers, and Shading Patterns

The colors, line types, markers, and shading patterns that are given to distinguish one data group from another are taken from tables whose values are used in rotation. Each data group can have a different color, line-type, marker, and shading pattern. Tables with default values are supplied. You can control the attributes of the data by resetting the values in the tables.

Color Numbers

Valid color numbers and their meanings are:

Color Attribute Value	System/370 GDDM Color Definition	AS/400 Display Default Color	618x Plotter Pen Number	7372 Plotter Pen Number	7371 Plotter Pen Number	5224/5, 3812, 3816, 4028, 4214, 4234-2 Printer	4224 Printer
-2	White	White	No pen	No pen	No pen	Background	Background
-1	Black	Black	7	6	1	Black	Black
0	Default	Green	8	6	2	Black	Black
1	Blue	Blue	1	1	1	Black	Blue
2	Red	Red	2	2	2	Black	Red
3	Magenta	Pink	3	3	1	Black	Magenta
4	Green	Green	4	4	2	Black	Green
5	Turquoise	Turquoise	5	5	1	Black	Cyan
6	Yellow	Yellow	6	6	2	Black	Yellow
7	Neutral	White	7	6	1	Black	Black
8	Background	Background	No pen	No pen	No pen	Background	Background
9	Dark blue	Blue	1	1	1	Black	Blue
10	Orange	Red	2	2	2	Black	Red
11	Purple	Pink	3	3	1	Black	Magenta
12	Dark green	Green	4	4	2	Black	Green
13	Dark turquoise	Turquoise	5	5	1	Black	Cyan
14	Mustard	Yellow	6	6	2	Black	Yellow
15	Gray	White	7	6	1	Black	Black
16	Brown	Green	8	6	2	Black	Brown
17 — 32,767	Beginning with 17, the color definitions for values 9 through 16 are repeated up to 32,767.						

Line-Type Numbers

Valid line-type numbers and their meanings are:

Line type attrib. value	Graphics work stations		SCS Printers and Plotters		IPDS Printers	
	Description	Appearance	Description	Appearance	Description	Appearance
0	Solid (default)	————	Solid (default)	————	Solid (default)	————
1	Dotted	.....	Dotted	.....	Dotted	▪ ▪ ▪ ▪ ▪ ▪ ▪
2	Short-dashed	— — — —	Short-dashed	— — — —	Short-dashed	— — — —
3	Dash-dot	—.—.—.—	Dash-dot	—.—.—	Dash-dot	—.—.—
4	Double-dot	.. ..	Long-dash short-dash	— — — —	Double-dot	.. .. ..
5	Long-dashed	— — — —	Long-dashed	— — —	Long-dashed	— — — —
6	Dash-dot-dot	—..—..—	Long-dash short-dash-dash	— — —	Dash-dot-dot	—..—..—
7	Solid	————	Solid	————	Solid	————
8	Invisible		Invisible		Invisible	

RV2F758-0

Note: Line styles on an IBM personal computer must be configured to match these defaults.

Marker Numbers

Valid marker numbers and their meanings are:

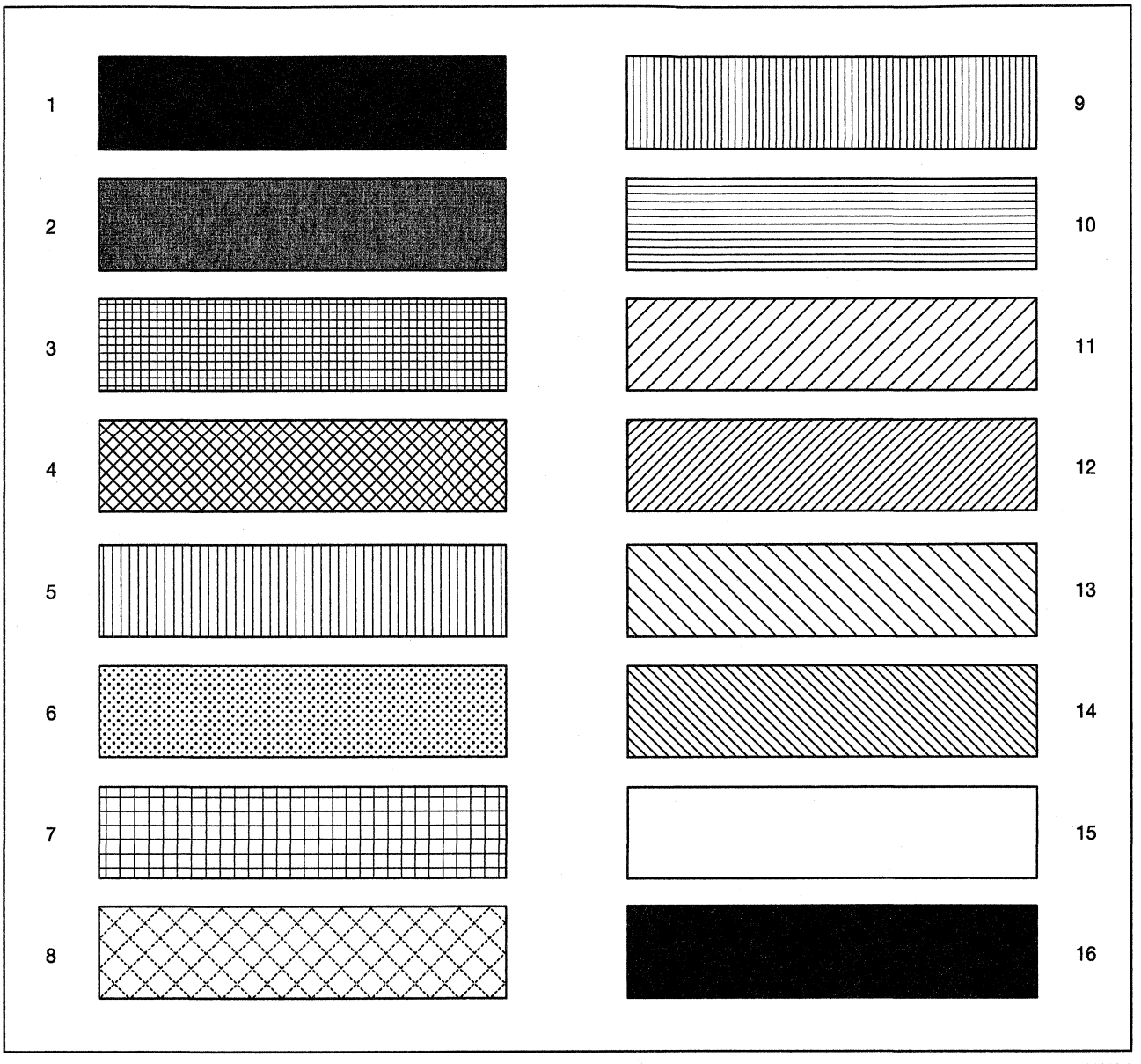
1 × 6 ✳
2 + 7 ◆
3 ◇ 8 ■
4 □ 9 ●
5 ✳ 10 ○

RSLF401-0

On the AS/400 system, values 65–254 are also supported. Symbol sets defining markers can be loaded by an application program before the chart is constructed. (See "GSLSS – Load a Graphics Symbol Set from Auxiliary Storage" on page 2-90.)

Shading Pattern Numbers

Valid pattern numbers and their meanings are:



RV2F760-0

For IPDS printers, and plotters, the area fill patterns are device dependent.

Glossary

This glossary includes terms and definitions from the *ISO Vocabulary—Information Processing* and the *ISO Vocabulary—Office Machines*, developed by the International Organization for Standardization, Technical Committee 97, Subcommittee 1. Definitions of published segments of the vocabularies are identified by the symbol (I) after the definition; definitions from draft international standards, draft proposals, and working papers in development by the ISO/TC97/SC1 vocabulary subcommittee are identified by the symbol (T) after the definition, indicating final agreement has not yet been reached among participating members.

adapter card. The electrical circuits on a logic card that connect one device to another or to a computer.

alphabetic character. (1) Any one of the letters A through Z or a through z or one of the characters #, $, or @. (2) In COBOL, a character that is one of the 26 uppercase letters of the alphabet, or a space. (3) In BASIC, a character that is one of the 26 uppercase or 26 lowercase letters of the alphabet.

alphanumeric character. In COBOL, any character in the character set of the computer.

American National Standard Code for Information Interchange (ASCII). The code developed by the American National Standards Institute for information exchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters, plus one parity bit.

API. See *application program interface (API)*.

application. A particular business task, such as inventory control or accounts receivable.

application program. A program used to perform a particular data processing task, such as inventory control or payroll.

application program interface (API). A functional interface supplied by the operating system or a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

area fill. The filling in of an enclosed area with a pattern.

argument. In a C language function call, an expression that represents a value that the calling function passes to the function specified in the call.

array. (1) In BASIC, a named set of data items, all of which are of the same type, arranged in a pattern (for example, rows and columns). An array can be implicitly declared through use or explicitly declared in a DIM statement. Contrast with *scalar*. (2) In PL/I, a collection of one or more elements with identical characteristics, grouped into one or more dimensions. (3) In RPG, a series of elements with like characteristics. An array can be searched for a uniquely identified element; or elements in an array can be accessed by their position relative to other elements. Contrast with *table*.

array element. In RPG, BASIC, and PL/I, one of the data items in an array.

ASCII. See *American National Standard Code for Information Interchange (ASCII)*.

aspect ratio. In PC Support/400 and AS/400 Business Graphics Utility, a ratio of one dimension to another. For example, the ratio of the width of the data to its height as it appears on the display.

attribute. A characteristic or trait of one or more items.

autoranging. The use of system defaults to determine the intervals on a chart so that the maximum and minimum data values can be represented on the graphics display station or plotter.

axis. In AS/400 Business Graphics Utility and GDDM, one of the intersecting horizontal or vertical scales where data values are plotted on a chart.

axis grid lines. In AS/400 Business Graphics Utility and GDDM, straight lines extending perpendicular to either axis at each major tick.

BGU. See *IBM AS/400 Business Graphics Utility Version 2 (BGU)*.

binary. A numbering system with a base of two (0 and 1).

binary floating-point value. In PL/I, an approximation of a real number in the form of a significand, which can be considered as a binary fraction, and an exponent, which can be considered as an integer exponent to the base of 2. Contrast with *decimal floating-point value*.

binary format. Representation of a decimal value in which each field must be 2 or 4 bytes long. The sign (+ or −) is in the far left bit of the field, and the number value is in the remaining bits of the field. Positive numbers have a 0 in the sign bit and are in true form. Negative numbers have a 1 in the sign bit and are in twos complement form.

bit string. A series of bits consisting of the values 0 and 1.

business graphics. See *graphics*.

Business Graphics Utility (BGU). See *IBM AS/400 Business Graphics Utility Version 2 (BGU)*.

called program. A program that is the object of a CALL statement combined at run time with the calling program to produce a run unit.

CGA. Color graphics adapter.

character. Any letter, number, or other symbol in the data character set that is part of the organization, control, or representation of data.

character constant. The actual character value (a symbol, quantity, or constant) in a source program that is itself data, instead of reference to a field that contains the data. Contrast with *numeric constant*.

character grid. In AS/400 Business Graphics Utility, an invisible network of uniformly spaced horizontal and vertical lines covering the chart area. Used by the AS/400 Business Graphics Utility to determine the physical dimensions of the chart and the placement of the data on it.

character grid unit. In AS/400 Business Graphics Utility, the distance between two adjacent horizontal or vertical lines on a character grid.

character set. A group of characters used for a specific reason; for example, the set of characters the display station can display, the set of characters a printer can print, or a particular set of graphic characters in a code page; for example, the 256 EBCDIC characters.

character string. A sequence of consecutive characters that are used as a value.

character variable. Character data whose value is assigned or changed while the program is running.

chart format. In AS/400 Business Graphics Utility, an object containing chart characteristics, such as the chart type, chart heading, legend position, and so on. The chart format does not include the data values to be plotted. The system-recognized identifier for the object type is *CHTFMT.

chart layout. In AS/400 Business Graphics Utility, the arrangement of the various parts in the chart area and surrounding margins.

clipping. In GDDM, the process of cutting off the image at the border of the display but allowing the coordinates of the lines to extend beyond.

closure line. In GDDM, a line added by the system to enclose an area being filled with a pattern, in instances when the routines that precede the GSENDA routine fail to form an enclosed area.

color table. A compilation of eight entries, each defining a color to be used in AS/400 graphics, from which individual colors are selected. Many color tables can be defined, but only one can be current.

column. In COBOL, a character position within a print line. The columns are numbered from one, by one, starting at the farthest-left character position of the print line and extending to the farthest-right position of the print line.

comment. Source program information that is not translated by the compiler. A comment consists of the characters (on one or more lines) beginning with /* and ending with */.

compile-time array. In RPG, an array that is compiled with the source program and becomes a permanent part of the program. Contrast with *run-time array* and *prerun-time array*.

compile-time table. In RPG, a table that is built into the source program and that becomes a permanent part of the compiled program. Contrast with *prerun-time table* and *run-time table*.

component. In Pascal, the name of each separate value, where a variable has multiple values in a structured type.

constant. (1) Data that has an unchanging, predefined value to be used in processing. (2) In Pascal, a value that is either a literal or an identifier that has been associated with a value in a CONST declaration. (3) In RPG, data that has an unchanging, predefined value to be used in processing. A constant does not change during the running of a program, but the contents of a field or variable can. See also *literal*.

current device. The current output device for the application program, usually a display screen.

current mode. In GDDM, the characteristics of the controlling session. For example, when a color is defined, everything the program draws uses that color until the color is changed.

current position. In GDDM, the position, in user coordinates, that becomes the starting point for the next graphics routine, if that routine does not explicitly specify a starting point.

cursor. A movable symbol, often a blinking or solid block of light, that tells the display station user where to type, or identifies a choice to select.

data description specifications (DDS). A description of the user's database or device files that is entered into

the system in a fixed form. The description is then used to create files.

data file utility (DFU). The part of the AS/400 Application Development Tools licensed program that is used to enter, maintain, and display records in a database file.

data group. (1) In AS/400 Business Graphics Utility, a collection of values that identify the comparisons in a chart. For example, the relative size of the slices in a pie chart or the relative height of the bars in a bar chart. See also *paired data*. (2) In GDDM, a collection of data values displayed, for example, as a pie chart or as the plotted points on a line of a line chart. More than one data group may be displayed on a chart.

data value. In AS/400 Business Graphics Utility, a single numeric data item entered as a value for a horizontal line or a vertical line. Contrast with *data group*.

database. All the data files stored in the system.

datum line. A straight reference line drawn from either axis that helps the user see the exact data values on the chart.

DDS. See *data description specifications (DDS)*.

debug mode. An environment in which programs can be tested.

decimal floating-point constant. In PL/I, a value consisting of a significand that contains a decimal fixed-point constant, and an exponent that contains the letter E followed by an optionally signed integer constant not exceeding three digits.

decimal floating-point value. In PL/I, an approximation of a real number, in the form of a significand, which can be considered as a decimal fraction, and an exponent, which can be considered as an integer exponent to the base of 10. Contrast with *binary floating-point value*.

default program. A user-specified program that is assumed when no other program is specifically named on a debug command, or a user-defined program for handling error messages.

device token. In GDDM, an 8-byte code, required to set the devices to a predefined set of hardware characteristics.

device type. The generic name for a group of devices. For example, 5219 for IBM 5219 Printers. Contrast with *device class*.

DFU. See *data file utility (DFU)*.

diagnostic message. A message that contains information about errors or possible errors. This message is generally followed by an escape message.

digit. Any of the numerals from 0 through 9.

dimension specification. In BASIC, the specification of the size of an array and the arrangement of its elements. Up to seven dimensions can be specified.

direction. The orientation of a string of mode-2 or mode-3 graphics symbols. Direction can dictate that the string reads left to right, right to left, top to bottom, or bottom to top.

display file. (1) A device file to support a display station. (2) In BASIC, any file that has the keyword DISPLAY specified in the OPEN statement for the file.

display screen. The part of the display device, which is similar to a television (TV) picture tube, used to display information entered or received at a display station.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or the information received from the system.

dummy device. In GDDM, an imaginary output device for which the program does all the normal processing but for which no actual output is received.

EBCDIC. See *extended binary-coded decimal interchange code (EBCDIC)*.

EGA. Enhanced graphics adapter.

element. (1) In a list of parameter values, one value. (2) In BASIC, FORTRAN, and RPG, the smallest addressable unit of an array or table.

extended binary-coded decimal interchange code (EBCDIC). A coded character set consisting of 8-bit coded characters.

externally described data. Data contained in a file for which the fields and the records are described outside of the program (such as with files created by DDS, IDDU, or the SAA Structured Query Language/400 licensed program) that processes the file. Contrast with *program-described data*.

fillet. In GDDM, a curve that is tangent to the end points of two connected lines. See also *polyfillet*.

floating-point format. (1) In binary floating-point representation, the storage format that represents a binary floating-point value. See also *long format* and *short format*. (2) In PL/I, a number shown as an optional sign, a decimal number with a decimal point, followed by the letter E, followed by an optional sign and a 1- to 3-digit integer. For example, 3.0E-2, is 3 times 10 to the -2 power or 0.03. (3) In BASIC, a number shown as an optional sign followed by an

integer or fixed-point constant, followed by the letter E, followed by an integer constant with up to 3 significant digits.

function key. A keyboard key that allows the user to select keyboard functions or programmer functions.

GDDM. See *graphical data display manager (GDDM)*.

GDF file. See *graphics data format (GDF) file*.

graphical data display manager (GDDM). A function of the operating system that processes both text and graphics for output on a display, printer, or plotter. Contrast with *presentation graphics routines (PGR)*.

graphics. (1) Pictures and illustrations. (2) Pertaining to charts, tables, and their creation.

graphics data format (GDF) file. A picture definition in a coded order format used internally by GDDM and, optionally, providing the user with a lower-level programming interface than the GDDM application programming interface.

graphics field. In GDDM, that part of the display or the paper that is used for pictures and graphics text.

graphics hierarchy. An ordered division of parts of the graphics program, of which the device is the highest level and parts of the picture are the lowest.

graphics image. Pictorial information that is specified in terms of the dots of which it is made up.

graphics primitive. In GDDM, a single item of graphics information, such as a line or a string of graphics text.

graphics segment. In GDDM, a group of graphics primitives (lines, arcs, and text) that are operated as a common set. The graphics primitives inside a graphics segment share characteristics, such as visibility and angle of rotation, but keep their individual characteristics, such as color and line width.

graphics symbol set. In GDDM, an object that can contain either lines or images. The system-recognized identifier for the object type is *GSS. See also *vector symbol set (VSS)* and *image symbol set (ISS)*.

graphics text. In GDDM, text displayed by an application program using a graphics symbol set.

graphics window. In GDDM, the view of the graphics picture that is defined by the range of the world coordinates specified by the user.

grid lines. Uniformly spaced horizontal and vertical reference lines on a chart. See also *axis grid lines*.

hardware character. In GDDM, an alphanumeric character provided by the display station, usually from a display file. See also *mode-2 character* and *mode-3 character*.

hexadecimal. Pertaining to a numbering system with a base of 16.

high-level language (HLL). A programming language, such as RPG, BASIC, PL/I, Pascal, COBOL, and C used to write computer programs.

HLL. See *high-level language (HLL)*.

hue. The gradual variations of colors such as blue, green, red, yellow, and so on.

IBM AS/400 Business Graphics Utility Version 2 (BGU). The IBM licensed program that can be used to design, plot, display, and print business charts.

image. An electronic representation of an original document recorded by a scanning device.

image symbol set (ISS). In GDDM, a graphics symbol set in which each character is treated as a small image and is described by a rectangular array of display points. Characters in an image symbol set are always drawn in a fixed size. Contrast with *vector symbol set*; see also *graphics symbol set*.

imaginary line. In GDDM, a construction line used to build a fillet. The beginning and ending points of imaginary lines are defined, but the lines themselves do not appear as part of the picture.

include statement. For OS/400 application programming interfaces, a statement that causes the compiler to replace the include statement with the contents of the specified header or file.

integer. (1) A positive or negative whole number. (2) In COBOL, a numeric constant or a numeric data item that does not include any digit position to the right of the assumed decimal point.

intelligent printer data stream (IPDS). (1) An all-points-addressable data stream that allows users to position text, images, and graphics at any defined point on a printed page. (2) In GDDM, a structured-field data stream for managing and controlling printer processes, allowing both data and controls to be sent to the printer.

IPDS. See *intelligent printer data stream (IPDS)*.

legend. In AS/400 Business Graphics Utility and GDDM, an explanatory list of the symbols, lines, and shaded areas on a chart.

library list. A list that indicates which libraries are to be searched and the order in which they are to be searched. The system-recognized identifier is *LIBL.

lightness. The characteristic that allows colors to be put in order from light to dark.

literal. In RPG, a character string whose value is defined by the characters themselves. For example, the numeric constant 7 has the value 7, and the character constant 'CHARACTERS' has the value CHARACTERS. See also *constant*, *character constant*, and *numeric constant*.

load. To move data or programs into storage.

logical file. A description of how data is to be presented to or received from a program. This type of database file contains no data, but it defines record formats for one or more physical files. Contrast with *physical file*.

long format. In binary floating-point storage formats, the 64-bit representation of a binary floating-point number, not-a-number, or infinity. Contrast with *short format*.

major tick. In AS/400 Business Graphics Utility, a mark on an axis that denotes character grid units on a chart. See also *minor tick*.

minor tick. In AS/400 Business Graphics Utility, one of the marks located between major ticks on an axis of a chart. See also *major tick*.

missing values. Values that do not appear on a chart but that are needed so that each data group has the same number of values.

mixing. The result of the intersection of two or more colors.

mode-2 character. In GDDM, a graphics character (symbol), characterized by an unchanging size, constructed from picture elements. Contrast with *mode-3 character*; see also *hardware character*.

mode-3 character. In GDDM, a graphics character (symbol), characterized by a variable size and shape, constructed from lines and curves. Contrast with *mode-2 character*; see also *hardware character*.

nonpaired data. In AS/400 Business Graphics Utility and GDDM, data that is specified such that each X-value has a set of Y-values associated with it. Contrast with *paired data*.

numeric constant. The actual numeric value to be used in processing, instead of the name of a field containing the data. A numeric constant can contain any of the numeric digits 0 through 9, a sign (plus or minus), and a decimal point. Contrast with *character constant*.

offset. In GDDM, the number of character grid units from a reference point.

overpaint. The default result of the intersection of two or more colors, in which the color of the first graphics primitive to appear is given the color of the graphics primitive that intersects it, at the point of intersection.

page. In GDDM, the picture or chart. All specified graphics are added to the current page. An output statement always sends the current page to the device.

paired data. In AS/400 Business Graphics Utility and GDDM, data that is specified so that every X value has only one Y value associated with it. See also *data group*. Contrast with *nonpaired data*.

parameter. A value supplied to a command or program that is used either as input or to control the actions of the command or program.

parameter list. A list of values that provide a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

pel. See *picture element*.

physical file. A description of how data is to be presented to or received from a program and how data is actually stored in the database. A physical file contains one record format and one or more members. Contrast with *logical file*.

picture element. In computer graphics, the smallest element of a display area, such as a dot, that can be independently assigned color and intensity.

picture space. In GDDM, the area of the page which contains the graphics.

pixel. A picture element that represents the finest degree of resolution.

plot. In GDDM, to represent graphically on paper using a plotting device.

polyfillet. In GDDM, a curve based on a sequence of lines. A polyfillet is line that is tangent to the end points of the first and last lines and tangent to the midpoints of all other lines. See also *fillet*.

polygon. In GDDM, a sequence of adjoining straight lines that enclose an area.

polyline. In GDDM, a sequence of adjoining lines.

prerun-time array. In RPG, an array that is loaded at the same time as the program, before the program actually begins to run. Contrast with *compile-time array* and *run-time array*.

prerun-time table. In RPG, a table that is loaded at the same time as the source program, before the program

actually begins to run. Contrast with *compile-time table* and *run-time table*.

presentation graphics routines (PGR). A group of routines within the operating system that allows business charts to be defined and displayed procedurally through function routines. Contrast with *graphical data display manager (GDDM)*.

program-described data. Data contained in a file for which the fields in the records are described in the program that processes the file. Contrast with *externally described data*.

radius. The length of a line that extends from the center of a circle or ellipse to a point on the curve of the circle or ellipse. Plural is radii.

reference line. In AS/400 Business Graphics Utility, a straight line parallel to either the vertical or horizontal axis relative to which data values are plotted on a chart. Sometimes called a translated axis line.

retained data. Graphics data that is associated with a graphics segment. Retained data is kept by the system and is capable of being displayed again within the same graphics environment. Contrast with *temporary data*.

run-time array. In RPG, an array that is loaded or created by input or calculation specifications after the program starts to run. Contrast with *compile-time array* and *prerun-time array*.

run-time table. In RPG, a table that is loaded or created by input or calculation specifications after the program starts to run. Contrast with *compile-time table* and *prerun-time table*.

saturation. The amounts of color and gray in a hue that affect its vividness; that is, a hue with high saturation contains more color and less gray than a hue with low saturation. See also *hue* and *lightness*.

scalar. Pertaining to a single data item. Contrast with *array*.

scale. (1) In GDDM, the number of and progression of ticks along a vertical or horizontal axis. (2) In GDDM, to enlarge an image or marker.

shear. In GDDM, the forward or backward slant of a mode-3 graphics symbol or string of such symbols relative to a line perpendicular to the baseline of the symbol.

short format. In binary floating-point storage formats, the 32-bit representation of a binary floating-point number, not-a-number, or infinity. Contrast with *long format*.

smoothness of curve. In GDDM, the connection of the plotted points in a data group by a continuous curve. On System/370 GDDM, called curve fitting.

source statement. A statement written in symbols of a programming language. For example, AS/400 BASIC, AS/400 PL/I, SAA COBOL/400, SAA RPG/400, and DDS statements are source statements.

sweep. In AFP support, the movement around an arc from the center point of the arc.

table. (1) In COBOL, a set of logically consecutive data items that are defined in the Data Division with the OCCURS clause. (2) In RPG, a series of elements with like characteristics. A table can be searched for a uniquely identified element, but elements in a table cannot be accessed by their position relative to other elements. Contrast with *array*.

tangent. In GDDM, the single point at which a straight line meets a curve or surface.

temporary data. Graphics data that is not associated with a graphics segment. Temporary data is lost once it is sent to the display. Contrast with *retained data*.

text box. The imaginary rectangle that encloses a string of mode-2 or mode-3 graphics symbols.

tick. In AS/400 Business Graphics Utility, a reference point on either the vertical or horizontal axis of some chart types that represents the location of specified data values. See also *major tick* and *minor tick*.

translated axis line. A straight reference line parallel to either axis relative to which data values are plotted on a chart. Sometimes called a translated line; on the System/370 computer called a reference line.

variable. A name used to represent data whose value can be changed while the program is running by referring to the name of the variable.

vector. In GDDM, a directed line segment; a straight line between two points.

vector line. A series of lines constructed by one GDDM routine.

vector symbol set (VSS). In GDDM, a set of characters each of which is treated as a small picture and is described by a sequence of lines and arcs. Characters in a vector symbol set can be drawn to scale, rotated, and positioned precisely. Contrast with *image symbol set (ISS)*; see also *graphics symbol set*.

viewport. In GDDM, a rectangular area within the picture space that defines where the output of the current page appears on the work station.

work station. A device used to transmit information to or receive information from a computer; for example, a display station or printer.

work station emulation. Allows IBM Personal System/2* models 50, 60, and 80 to be attached to System/36, System/38, and AS/400 computers. Configured as a 5292 Model 2, workstation emulation is treated by the host as a graphics display with extended capabilities. Abbreviated WSE.

work station function (WSF). The part of the AS/400 PC Support/400 licensed program for DOS users that allows a personal computer attached to an AS/400 system to emulate up to five display stations, and a PC printer to emulate a system printer.

world coordinates. In GDDM, the user-defined set of coordinates that define the graphics window, and that serve as the horizontal and vertical range for all graphics primitives within the graphics window.

WSF. See *work station function (WSF)*.

x axis. See *axis*.

y axis. See *axis*.

Bibliography

The following AS/400 manuals contain information you may need. The manuals are listed with their full title and base order number. When these manuals are referred to in this manual, the short title is used.

- *Business Graphics Utility User's Guide and Reference*, SC09-1408, provides the application programmer, programmer, system administrator, or business or technical professional with information about using the AS/400 Business Graphics Utility (BGU) to create various types of charts. It is divided into two sections: the first section includes several exercises that familiarize the user with the functions of BGU, and the second section contains reference material.

  Short Title: *BGU User's Guide and Reference*

- *Data Description Specifications Reference*, SC41-9620, provides the application programmer with detailed descriptions of the entries and keywords needed to describe database files (both logical and physical) and certain device files (for displays, printers, and ICF) external to the user's programs.

  Short Title: *DDS Reference*

- *Device Configuration Guide*, SC41-8106, provides the system operator or system administrator with information on how to do an initial configuration and how to change that configuration. This manual also contains conceptual information about device configuration.

  Short Title: *Device Configuration Guide*

- *Languages: BASIC User's Guide and Reference*, SC09-1157, provides the application programmer with the information needed to write, test, and maintain AS/400 BASIC programs in both the AS/400 environment and the System/38 environment.

  Short Title: *BASIC User's Guide and Reference*

- *Languages: Pascal User's Guide*, SC09-1209, provides the application programmer with information about how to use the AS/400 Pascal compiler. The manual explains how to enter, compile, run, and debug AS/400 Pascal programs. It also describes how to use input and output (I/O) function and storage.

  Short Title: *Pascal User's Guide*

- *Languages: PL/I User's Guide and Reference*, SC09-1156, provides the application programmer with information about using AS/400 PL/I in the System/38 environment. Differences between the System/38 environment and the AS/400 environment are identified as well as the enhancements available in the AS/400 environment.

  Short Title: *PL/I User's Guide and Reference*

- *Languages: Systems Application Architecture* AD/Cycle* COBOL/400* User's Guide*, SC09-1383, provides the application programmer with the information needed to design, write, test, and maintain COBOL/400 programs on the AS/400 system.

  COBOL/400 supports the ANSI 85 intermediate-level COBOL.

  Short Title: *COBOL/400* User's Guide*

- *Languages: Systems Application Architecture* AD/Cycle* RPG/400* User's Guide*, SC09-1348, provides the application programmer with the information needed to write, test, and maintain RPG/400 programs on the AS/400 system. The manual provides information on data organizations, data formats, file processing, multiple file processing, automatic report function, RPG command statements, testing and debugging functions, application design techniques, problem analysis, and compiler service information. The differences between the System/38 RPG III, System/38 compatible RPG, and RPG/400 are identified.

  Short Title: *RPG/400* User's Guide*

- *Licensed Programs and New Release Installation Guide*, SC41-9878, provides the system operator or system administrator with step-by-step procedures for initial installation, installing licensed programs, program temporary fixes (PTFS), and secondary languages from IBM.

  Short Title: *Licensed Programs and New Release Installation Guide*

- *Programming: Control Language Programmer's Guide*, SC41-8077, provides the application programmer or programmer with a wide-ranging discussion of AS/400 programming topics. Included in the guide are discussions about objects and libraries, control language (CL) programming, messages and message handling, how to define and create user-defined commands and menus, and application testing.

  Short Title: *CL Programmer's Guide*

- *Programming: Control Language Reference*, SC41-0030, provides the application programmer with a description of the AS/400 control language (CL) and its commands. Each command description includes a syntax diagram, parameters, default values, keywords, and an example. The information should be used to refer to the control language commands to request functions of the Operating System/400 (5738-SS1) licensed program and of the various languages and utilities.

Short Title: *CL Reference*

- *Programming: GDDM Programming Guide*, SC41-0536, provides the application programmer with information about using OS/400 graphical data display manager (GDDM) to write graphics application programs.

  This manual includes many example programs and information to help managers and planners understand how the product fits into data processing systems.

  Short Title: *GDDM Programming Guide*

Index

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Numerics

Special Characters

Reader's Comments

Application System/400
Programming:
GDDM Programming Reference
Version 2

Publication No. SC41-0537-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

IBM®

Cut
Alo

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

C
A

Reader's Comments

Application System/400
Programming:
GDDM Programming Reference
Version 2

Publication No. SC41-0537-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Phone No.

IBM®

Cu
Alc

Fold and Tape | **Please do not staple** | Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN DEPT 245
IBM CORPORATION
3605 HWY 52 N
ROCHESTER MN 55901-7899

Fold and Tape | **Please do not staple** | Fold and Tape

Cu
Al

Program Number: 5738-SS1

Printed in U.S.A.

SC41-0537-00