ESD-TR-70-124, Vol. I

MTR-1768, Vol. I

GRASP: A PL/I COMPATIBLE GRAPHICS SUBROUTINE PACKAGE FOR THE IBM 2260 DISPLAY STATION (LOCAL ATTACHMENT)

VOLUME I - INTRODUCTION AND USER'S MANUAL

R. H. Bullen, Jr.

MAY 1970

Prepared for

DIRECTORATE OF SYSTEMS DESIGN AND DEVELOPMENT
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 512A
Prepared by
THE MITRE CORPORATION
Bedford, Massachusetts
Contract F19(628)-68-C-0365

ESD-TR- 70-124, Vol. I

MTR-1768, Vol. I

GRASP: A PL/I COMPATIBLE GRAPHICS SUBROUTINE PACKAGE FOR THE IBM 2260 DISPLAY STATION (LOCAL ATTACHMENT)

VOLUME I - INTRODUCTION AND USER'S MANUAL

R. H. Bullen, Jr.

MAY 1970

Prepared for

DIRECTORATE OF SYSTEMS DESIGN AND DEVELOPMENT
ELECTRONIC SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts

Project 512A

Prepared by

THE MITRE CORPORATION
Bedford, Massachusetts

Contract F19(628)-68-C-0365

FOREWORD

This report was prepared by The MITRE Corporation, Bedford, Massachusetts under Contract No. F19(628)-68-C-0365, MITRE Project 512A. Volume I gives an overview of the IBM 2260 Display Station and an introduction to the PL/I compatible routines which provide programming support for the IBM 2260 in local attachment. Volume II provides detailed program specifications.

REVIEW AND APPROVAL

This technical report has been reviewed and is approved.

CHARLES E. MCKUSICK, Lt Colonel, USAF
Chief, Systems Analysis Division
Directorate of Systems Design and Development

ABSTRACT

GRASP is a set of PL/I compatible subroutines which provide programming support for the IBM 2260 Display Station in local attachment; i.e., the attachment of a 2260 directly to a System/360 CPU channel via the IBM 2848 Display Control. The subroutines are coded in OS/360 Assembler Language and are reentrant. They permit the PL/I programmer to manipulate the 2260 as an I/O device in the same manner available to the Assembler Language programmer using the Graphics Access Method under OS/360 (with restrictions as noted in the Introduction to Volume I of this document). All errors, except those which normally result in OS/360 abnormal ends (ABENDs), are returned to the user via subroutine parameters. GRASP is designed to operate under the MFT configuration of OS/360.

Volume I of this document gives an overview of the 2260 and an introduction to the GRASP routines. Volume II gives detailed program specifications.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

The IBM 2260 Display Station is a small character-oriented display which may be attached to an IBM System/360 computer via transmission lines to an IBM 2701 Data Adapter Unit (remote attachment) or directly to a CPU channel via the IBM 2848 Display Control Unit (local attachment). Programming support for the former mode of attachment is provided to Assembler Language programmers under OS/360 via the Basic or Queued Telecommunications Access Methods (BTAM and QTAM). Programming support for the latter mode, also available only to Assembler Language programmers, is provided by the Graphics Access Method (GAM). At the time of this writing IBM does not support either mode of attachment in any of its available higher level languages, although QTAM support has been announced for PL/I Version 5, to be released in early 1970.

For some time, two 2260's in local attachment (and therefore supported by GAM) were available at MITRE, although until early 1969 there was much interest but little use. With the advent of IBM's Conversational Programming System (CPS) and interest in the 2260's by the MITRE Management Information System (MIS) and Military Airlift Command (MAC) projects, use of the 2260's increased. A need was immediately felt for an interface to GAM for the PL/I Language, since both the MIS and MAC projects were using that language.

A preliminary investigation of available software was undertaken by the MIS project and revealed that there was no interface package available at the time which reflected, on the one hand, the inherent simplicity of the GAM support and the 2260 itself and which, on the other hand, did not inhibit the higher level language programmer from taking full advantage of the capabilities available to the Assembler Language programmer using GAM. In addition, since the PL/I Language already provided certain basic system programming capabilities, foremost among these being the capability of producing reentrant programs, it was felt that the interface package should be reentrant. This would guarantee in addition that the package could be transferred to a computing system using MVT, without degrading that system's capabilities and without reprogramming. A suitable interface package not being found, it was decided to build such a package at MITRE; that package is GRASP.

The version of GRASP documented in this paper does not provide the full capability of GAM programming support. It supports:

(1) I/O operations addressing a 2260;

(2) the keyboard and line addressing features of the 2848; and

(3) attention handling in a background-type user program via the "basic" method (see reference 2).

GRASP does not support:

(1) user written asynchronous attention processing programs; and

(2) the "express" method of attention handling (see reference 2).

In addition, although GRASP was written to support any valid 2848/2260 combination, it has only been tested with a 2848 Model 3 with two attached 2260 Model 1 Display Stations. The omission of the express attention handling capability is not serious since it does not appear to be as useful as the basic method, and if it were required, it could be provided with a minimum of effort. However, the omission of support for user written asynchronous attention processing programs is significant. Considerable work was done to determine if it would be possible to implement a PL/I programmer-defined condition which would be raised whenever an attention occurred. The PL/I programmer would then be able to use the full capabilities of PL/I for enabling and disabling condition handling specifications. But it was found that the manner in which asynchronous exit routines are activated by the Operating System is not compatible with the PL/I implementation of condition on-units. One method of interfacing the Operating System and PL/I was identified, but it required the addition of a special SVC routine to the system library. This was felt to be an unsatisfactory approach to the problem, and the attempt to implement this capability was abandoned.

This paper is divided into two volumes. The first gives a brief introduction to the 2260, and describes GRASP and its use. The second gives detailed specifications of the GRASP routines. The reader is assumed to be familiar with the PL/I Language.

SECTION II

THE 2260 ENVIRONMENT

GENERAL DESCRIPTION

This section presents a subset of information from selected documents in the Bibliography. Readers desiring more information should consult the appropriate document.

The 2260 is a small character-oriented display capable of displaying either 6 or 12 lines of either 40 or 80 characters each, depending on the 2260/2848 configuration (see Figure 1). The range of characters includes:

26 alphabetic characters;

10 numeric characters;

25 special symbols (including the space and new line symbols); and

3 control symbols (cursor, check and start symbols).

Graphics associated with the special and control symbols are shown in Figure 2. Bit patterns for all symbols displayable on a 2260 Display Station are shown in Reference 3. Operating instructions for the 2260 Display Station can be found in Reference 1.

SPECIAL FEATURES

Special features of 2848/2260 configuration of interest to the user of GRASP include the keyboard, line addressing, non-destructive cursor, data entry, and 1053 Printer options. Other special features are available and are described in Reference 3.

Keyboard

If the selected 2260 is not equipped with the keyboard feature, the parts of the GRASP package dealing with attention handling and input operations must not be used. Such a 2260 is only an output device.

2848 Model	2260 Model	Maximum No. of Devices	Lines per Display	Characters per Line	Characters per Display
1	2	24	6	40	240
2	2	16	12	40	480
3	1	8	12	80	960
21[1]	2	24	6	40	240
22[1]	2	16	12	40	480

Figure 1. 2260/2848 Configurations[2]

---

(1) Data entry configurations; see "Special Features".

(2) As of this writing, MITRE's configuration at the Bedford 360/50 computer center includes a 2848 Model 3 and four 2260 Model 1's with the keyboard, line addressing, and non-destructive cursor features.

$	#	◢	new line
@	¬	▶	start
<	|	▬	(EOM symbol; destructive cursor)
>	/	▮	non-destructive cursor
%	;	■	check
+	:		
*	,		
(	.		
)	?		
–	/		
-	(blank)		
=	&		

Figure 2. Special and Control Symbols

The CPU is interrupted via the ENTER key on the keyboard attached to a 2260, causing the control program to store information related to the attention in an attention queue. Via the attention handling capabilities of GRASP, the programmer may inspect the attention queue and service the device.

Line Addressing

This feature permits the programmer to specify, via a control character transmitted to the 2260 as the first byte of data to be written, that display is to begin at column 1 of a particular line of the screen. This feature also permits a very limited cursor positioning capability: if only the control character is transmitted, the cursor is positioned to column 1 of the indicated display line.

Non-Destructive Cursor

The standard destructive cursor is a symbol, indistinguishable from the EOM symbol, which indicates the display position to be occupied by the next character key depressed or character displayed. It occupies a display position and overwrites display positions as it advances. The non-destructive cursor performs the same function as the destructive cursor, but is a different graphic and does not occupy a display position; it is displayed to the left of and slightly below the next character position to be filled when data is transmitted or when a character key is depressed.

Data Entry

After the completion of a read operation from a 2260, the keyboard of the selected 2260 is mechanically unlocked and operator entry may resume. The data entry models of the 2848 Display Control permit the bypassing of this operation, thereby enabling program verification and response to operator input prior to the entry of another message. This is supported in GRASP by the "lock" (L) code in the mode arguments to the read, write, and erase routines. If the 2848 controlling the selected 2260 is not a data entry model, these codes must not be used. The data entry models of the 2848 have several other features, including a keypunch-type keyboard layout (rather than a typewriter layout), which are described in Reference 3.

Printer

A 2848/2260 configuration can be equipped with a slave 1053 printer. This is a low speed (14 characters per second) modified SELECTRIC printer and may either be activated by the operator via the PRINT key on the 2260 or be treated as an output device by the computer program. Programming considerations for the 1053 printer are given in References 2 and 3.

SECTION III

GRASP/2260 USER'S MANUAL

INTRODUCTION

GRASP is a set of subroutines and functions which provide the PL/I programmer programming support for the 2260 Display Station. The subroutines available can be broken down into the following categories:

- General I/O Support - this group includes subroutines which allocate and format graphics control blocks, and perform the open and close functions.
- Specific I/O Routines - this group includes routines which initiate read, write, and erase operations to a 2260, and which wait for the completion of I/O operations.
- Attention Handling - this group includes subroutines which allocate and build attention environment control blocks, and which support the inspection of the Operating System attention queue.
- Support Functions - this group includes a set of functions which, among other things, permit the PL/I programmer to access information in graphics control blocks.

Figure 3 lists the GRASP routines under the above headings and gives a brief definition of each.

This section describes the use and features of the GRASP routines. Control blocks of interest to the GRASP user are described as background information. The input and output routines are then described, followed by a discussion of attention processing. Several special programming considerations are presented and a final section describes a sample GRASP application.

General I/O Support

GOPEN - Define and open a graphics data control block.

GCLOSE - Close a graphics data control block.

Specific I/O Routines

GREAD - Initiate a read from a 2260.

GWRITE - Initiate a write to a 2260.

GERASE - Initiate an erase of a 2260.

GWAIT - Wait for the completion of an I/O operation.

Attention Handling

GAAP - Activate attention processing.

GDAP - Deactivate attention processing.

GAQ - Attention inquiry.

Support Functions

GNUNITS - Number of units in a unit group.

GUNIT - Unit in the unit group causing an attention.

GLINENO - Line number conversion.

GNCP - Number of channel programs value.

Figure 3. GRASP Routines

CONTROL BLOCKS

The GRASP user is required to be familiar with three OS/360 I/O control blocks: the Data Control Block, the Graphics Attention Control Block, and the Data Event Control Block. This section describes these control blocks and their use by GRASP routines.

The user is not expected to know the format of these control blocks. Each control block is automatically generated by an appropriate GRASP routine and its address is returned to the user as a PL/I POINTER variable. This variable is then passed as an argument to other GRASP routines which require access to the particular control block.

Data Control Block (DCB)

The DCB is the primary link between the external device and other I/O control blocks. It is generated by the open process by a call to the GOPEN routine, is specified in calls to GREAD, GWRITE, and GERASE to initiate I/O operations, and is referenced in the call to GCLOSE to perform the close process. It is also used as an argument to the GAAP routine (Activate Attention Processing) to initiate attention handling operations (see below).

Graphics Attention Control Block (GACB)

A GACB is generated for each DCB for which attentions are to be processed. The GACB is generated by the GAAP subroutine, is specified as an argument to the GAQ (Attention Inquiry) routine to inspect the attention queue, and is input to the GDAP (Deactivate Attention Processing) routine to terminate the processing of attentions for a specific DCB. Since attention processing is always DCB-specific, the pointer to the DCB must be specified as an argument to the GAAP routine. Thereafter, only the GACB need be specified since it contains an internal pointer to the DCB with which it is associated.

Data Event Control Block (DECB)

The DECB is exactly equivalent in function to a PL/I EVENT variable. A unique DECB is generated each time a call is made to GREAD, GWRITE, or GERASE to initiate an I/O operation, and its address is returned as a PL/I POINTER variable. This pointer may then be specified as an argument to the GWAIT routine to wait for completion of the associated I/O operation.

INPUT OPERATIONS

Three types of input operations are provided to the user of GRASP: buffer input, manual input, and short manual input. All three types are implemented by options of the GREAD routine.

Buffer Input

If 'B' is coded as the read-mode argument to the GREAD routine*, the entire contents of the buffer of the selected 2260 Display Station is transferred to memory. After transfer, the screen is erased. All characters, including special characters, are transferred.

Manual Input

If 'M' is coded as the read-mode argument to the GREAD routine, the characters on the screen between the START symbol (►) and the EOM symbol (▬) are transferred to memory. The START symbol is deleted from the screen and the cursor is placed immediately to the right of the position originally occupied by the START symbol.

The new line symbol (◢) has a special use with manual input read operations. When a new line symbol is encountered between the START and EOM symbols during the read operation, characters on the same line as and to the right of the new line symbol are skipped and data transfer begins again with the first character on the next line. The new line symbol is transferred as a data character.

The behavior of the manual input read operation on encountering a new line symbol can be used to advantage when more than one distinct piece of information is to be transferred in a single read operation. Consider an example. The program initially displays command information on the right of the screen and places the cursor at column 1 of the first line:

```
▌          JOB NAME
           PROJECT NO.
           DEPT.
```

---

* Detailed definitions of the arguments to GRASP routines are given in Volume II of this document.

The display operator enters the START symbol and the "job name" and depresses the new line key. This causes the cursor to move to column 1 of the second line:

```
▶ TEST◢          JOB NAME
▮                PROJECT NO.
                 DEPT.
```

The operator continues by typing the other required information and depresses the ENTER key when the message is complete:

```
▶ TEST◢          JOB NAME
  512A◢          PROJECT NO.
  D73■           DEPT.
  ▮
```

When the program issues a manual input read operation, the string transferred is:

'TEST ◢ 512A ◢ D73'.

Transfer of input data to the string variable specified in the call to GREAD occurs asynchronously to the execution of the user's program. In addition, the current length of this string variable, which has the VARYING attribute, is not set until a GWAIT is issued for the read operation. For these reasons, it is advisable not to make reference to the string variable until after the operation is complete; i.e., after a call has been made to GWAIT.

Short Manual Input

At the completion of a manual input operation, the START symbol is deleted from the screen and the cursor is placed to the left of the position originally occupied by the START symbol. Short manual input, identical to manual input in all other respects, does not delete the START symbol from the screen and therefore results in a somewhat faster read operation.

OUTPUT OPERATIONS

Three types of output operations are provided by GRASP: screen erase, buffer output, and line addressed output. The screen erase operation is provided by the GERASE routine. The remaining two write operations are performed by the GWRITE routine.

Screen Erase

Two methods are available for erasing the screen of a 2260. The first method, used when a write operation will not immediately follow the erase, is provided by the GERASE routine. The second method is the pre-write erase option of the GWRITE routine. In this case, the screen is erased immediately prior to the write operation specified by the call to GWRITE.

Buffer Output

By coding 'B' or 'EB' (if a pre-write erase is desired) as the write-mode argument to the GWRITE routine, the programmer may specify that the specified string argument is to replace the contents of the buffer of the selected 2260 Display Station. If this mode is used, the length of the string to be displayed must be exactly equal to the size of the buffer for the selected 2260 Display Station (either 240, 480 or 960 characters, depending upon the 2848 model in use).

The first character of the string to be displayed appears in column 1 of the first display line, and subsequent characters are displayed adjacent and to the right with spillover to subsequent display lines.

Line Addressed Output

When a 2848/2260 combination is equipped with the line addressing feature, the programmer may specify that a message is to be displayed on a particular line of the selected 2260 Display Station. This type of write operation is specified by coding 'A' or 'EA' (if a pre-write erase is desired) as the write-mode argument of the GWRITE routine.

In this mode of operation, the first byte of data is interpreted as a control character, indicating the display line to which the string will be written. This character must be included in the length of the string, but is not displayed. Display of the string begins at column 1 of the addressed line and spills over to subsequent lines (with wrap around from the last line to the first line) if the string is longer than one display line. Characters on the screen beyond the last character of the displayed string are not affected by a line addressed write operation.

A special function, called GLINENO, is provided as a part of GRASP to aid the user of line addressed output. The user specifies the desired line number as a FIXED BIN (31) integer, and codes the following expression as the string argument to the GWRITE routine:

```
GLINENO(line_number)||string
```

where "string" is the string to be displayed. The returned value of the GLINENO function has the attribute CHAR(1). The dummy argument generated by PL/I for the string expression above automatically accounts for the addition of the control character to the string to be displayed.

ATTENTION HANDLING

Attention handling support is provided in GRASP by the GAAP, GAQ and GDAP routines. This section describes these routines and their use.

Initialization

To initiate attention processing the user calls the GAAP routine specifying the DCB for which attentions are to be accepted. GAAP generates a GACB, notifies the control program that attentions for that DCB/GACB combination are to be honored, and returns the GACB address to the user. Any attentions occurring after the call to GAAP are queued by the control program until the user calls GAQ to inspect the queue.

Attention Inquiry

Whenever the user wishes to inspect the attention queue, he calls the GAQ routine, passing as an argument the GACB associated with the DCB for which attentions are desired. He also specifies, via an argument, the mode of the query:

(1) Wait (W) or Relinquish (R) mode*;

(2) Conditional (C) mode; and

(3) Clear (X) mode.

---

* Wait and Relinquish modes are equivalent in this implementation of GRASP. They are included for compatibility with possible future extensions of GRASP.

In W-mode or R-mode, the user is placed in a wait state until the desired attention occurs. In C-mode, the user requests that a BIT(1) variable specified as an argument be set to '1'B or 'Ø'B to indicate whether or not the desired attention is present. If X-mode is specified, the attention queue is cleared; any attentions on the queue are lost.

The user selects a desired attention by specifying as an argument the unit number of the 2260 Display Station for which attentions are expected. This number, meaningless for X-mode, may be an integer greater than or equal to 1 to select an attention from a particular 2260 or may be zero to indicate that attentions are to be accepted from any 2260 defined by the DCB associated with the specified GACB.

On return from an R-mode or W-mode call to GAQ, the desired attention is present and a GREAD may be issued to the unit causing the attention. Similarly, on return from a C-mode call, if the BIT(1) variable has been set to '1'B the attention is present and a read may be issued.

If for C-mode, R-mode, or W-mode, a non-zero unit number was specified, the user knows which unit caused the attention, and a read may be issued to that particular unit by specifying the same unit number in the call to GREAD. If, however, zero was specified as the unit number to indicate that attentions are to be accepted from any device, the user needs to know which of the devices defined by the GACB/DCB combination caused the attention. For this purpose, a GRASP routine, called GUNIT, is provided. On return from GAQ, and in the presence of an attention, the function returns the unit number of the unit causing that attention.

Termination

To terminate the processing of attentions, the GRASP user calls the GDAP routine. This program notifies the control program that queueing of attention information for the DCB associated with the specified GACB should cease; storage occupied by the GACB is freed. Any attentions occurring after the call to GDAP are lost.

SPECIAL CONSIDERATIONS

The Number of Channel Programs (NCP) Parameter

It is possible, when servicing more than one 2260 using a single Data Control Block (DCB) (and therefore a single Data Definition (DD) statement), to overlap I/O operations on these devices. The NCP parameter of the DCB may be used for this purpose. The function of NCP is to specify to OS/360 the maximum number of channel programs; i.e., I/O operations, which may be outstanding for a given DCB at any time. Outstanding I/O operations are ones which have been initiated (in GRASP by a call to GREAD, GWRITE, or GERASE) for the same DCB without an intervening wait operation (performed in GRASP by calling GWAIT).

The value of NCP is an integer between 1 and some maximum value specified during system generation of the OS/360 version in control, but is never greater than 99. The value of NCP for a particular DCB may be specified by the programmer in his call to GOPEN or may be specified on the DD statement by coding DCB=GNCP=value. If the NCP value is specified in the call to GOPEN, this value overrides any value specified on the DD statement.

In a GRASP application where more than one 2260 is defined by a single DCB, the programmer may set the NCP value for the DCB to a value high enough to permit simultaneous I/O operations to be in effect for all of the 2260's defined by the DCB. By calling the GNUNITS routine prior to opening the DCB, the program determines the number of devices defined on the DD statement for the DCB. The programmer may then set the NCP argument to GOPEN to this value. This allows the GRASP program to exercise a certain measure of control over the external specification of the NCP value, while at the same time not restricting the program to a particular, perhaps too small, value.

The following rules must be followed in making use of the NCP capability for overlapping I/O operations:

(1) Every overlapped I/O operation must have its own DECB. This is accomplished by using unique identifiers for the first arguments ("decbptr") in overlapped calls to the I/O initiation routines (GREAD, GERASE, and GWRITE).

(2) I/O operations must be waited on (by calling GWAIT) in the order in which they were initiated. For example,

```
CALL GREAD (DECB1,...);
CALL GREAD (DECB2,...);
CALL GWAIT (DECB2,...);
CALL GWAIT (DECB1,...);
```

is not correct; the waits must be performed in the opposite order.

Writing the START Symbol

The START and EOM symbols are used to identify the characters in the buffer to be transmitted to memory on a manual input read operation. Both symbols may be entered on the screen by the display operator. It is, however, desirable under certain circumstances for the user program to display the START symbol as part of a command. The program might display:

"ENTER NAME ▶ "

where "▶" represents the START symbol. The cursor would appear immediately to the right of the START symbol. The operator of the display would then follow by typing his name and depressing the ENTER key. The displaying of the START symbol by the program can result in a significant increase in "throughput", especially when operator responses are short.

Since the START symbol corresponds to the graphic "¢", which is generally not available on print trains, a CHAR(1) string variable, GCENT, the value of which is this symbol, is available to the user of the GRASP standard include set (see "Sample GRASP Application"). If the user wishes to include the START symbol in his displayed string, he simply concatenates the variable GCENT to the string he wants displayed. In the above example, the string argument to the call to GWRITE which resulted in the above message being displayed would have been coded:

```
'ENTER NAME'||GCENT
```

Using Line Addressed Output

Care must be taken when using line addressed output to "clean up" areas of the screen affected by previous operator messages. Consider the example in section a of Figure 4. The call to GWRITE causes the message

ENTER NAME ▶

to be displayed on line 1 of the display. The cursor is shown to the immediate right of the START symbol to indicate that the first character key to be depressed by the display operator will cause that character to appear adjacent to and to the right of the START symbol.

The display operator then types in his name and depresses the ENTER key causing the EOM symbol to appear. The first line of the display would now contain:

ENTER NAME ▶ R. H. BULLEN ■

When the GREAD subroutine is called, the characters between the START and EOM symbols are transferred to memory and stored in the program variable "NAME" and the display line is automatically modified to contain:

ENTER NAME R. H. BULLEN ■

A branch is now taken to the statement labelled "LOOP", causing the command to be rewritten, and the display line to appear as:

ENTER NAME ▶ R. H. BULLEN ■

The previous operator input remains on the screen since the command is output using line addressing.

One way of solving this problem is shown in section b of Figure 4. By adding an extra call to GWRITE, the first line of the display is blanked out prior to writing the command. It is necessary to use two separate calls to ensure that the cursor will appear immediately to the right of the START symbol, when the command is displayed.

```
a) LOOP:   CALL GWRITE (...,'A',...,GLINENO(1)||'ENTER NAME'||GCENT,...);
                  :
           CALL GREAD (...,'M',...,NAME,...);
                  :
           GO TO LOOP:

b) LOOP2:  CALL GWRITE (...,GLINENO(1)||(80)' ',...);
                  :
           CALL GWRITE (...,GLINENO(1)||'ENTER NAME'||GCENT,...);
                  :
           CALL GREAD (...,NAME,...);
                  :
           GO TO LOOP2;
```

Figure 4. A Line-Addressed Output Example

SAMPLE GRASP APPLICATION

Figure 5 shows the PL/I source language for a sample GRASP application. In general, the program initializes a display and waits for attentions. When an attention occurs, a read is issued and the string received is printed. When the operator at the display enters "STOP", the program erases the display and terminates.

Several points are worth noting about the program:

(1) In the compile step, a DD statement defining the include library AAINC is required, if the GRASP standard include set is used. The include set is shown in Figure 6 and contains declarations for the GRASP routines and the definition of the special variable GCENT, the value of which appears as the START symbol when displayed on a 2260.

(2) A % INCLUDE statement is required to cause the GRASP program declarations to be inserted in the program. The statements which follow the % INCLUDE cause selected declarations to be inserted in the program as source code. The specific declarations are selected by coding the name of the desired subroutine followed by a # sign; for example, GOPEN# is coded to cause the declaration of the GOPEN routine to be inserted in the program. Each subroutine declaration selection must be followed by a semicolon. Figure 7 shows the listing of the PL/I compilation of the sample program, indicating the inclusion of the selected declarations. In addition to the individual declarations, a special symbol GALL# is provided which, when included in the user's program, results in the inclusion of declarations for all GRASP routines.

(3) The third argument to the GOPEN routine is coded as 'IOE' to indicate that the GREAD, GWRITE, and GERASE routines will be used. The NCP value is set to 1.

SAMPLE GRASP APPLICATION

```
01/02/70 16:02:53
000000000111111111122222222223333333333444444444455555555556666666666777777777 8
123456789012345678901234567890123456789012345678901234567890123456789012345678 0
//DISPLAY  JOB ('512A',D73,10,D082),'BULLEN RH',CLASS=O
//        EXEC  PL1LFCLG,PARM.PL1L='M'
//PL1L.DCLS DD DSN=AAINC,DISP=SHR,UNIT=PACK,
//             VOL=(PRIVATE,RETAIN,SER=DP5010)
//PL1L.SYSIN DD *
 DISPLAY : PROC OPTIONS(MAIN);
          DCL    (DCBPTR,GACBPTR,DECBPTR) POINTER,
                 COND BIT(1),
                 MIMSG CHAR(30) VARYING,
                 R FIXED BIN(31);
 % INCLUDE DCLS(GRASPINC);
         GOPEN#;
         GAAP#;
         GWRITE#;
         GWAIT#;
         GLINENO#;
         GAQ#;
         GREAD#;
         GERASE#;
         GDAP#;
         GCLOSE#;
         GCENT#;
          CALL GOPEN   (DCBPTR,'DISP','IOE','BASIC',1,0,R);
          CALL GAAP    (GACBPTR,DCBPTR,R);
 LOOP :   CALL GWRITE  (DECBPTR,'EA',DCBPTR,
                         GLINENO(1)||'READY '||GCENT,1,R);
          CALL GWAIT   (DECBPTR,R);
          CALL GAQ     (GACBPTR,'W',COND,1,R);
          CALL GREAD   (DECBPTR,'M',DCBPTR,MIMSG,1,R);
          CALL GWAIT   (DECBPTR,R);
          PUT  EDIT    (MIMSG) (SKIP,A);
          IF MIMSG¬='STOP' THEN GO TO LOOP;
          CALL GERASE (DECBPTR,'',DCBPTR,1,R);
          CALL GWAIT   (DECBPTR,R);
          CALL GDAP    (GACBPTR,R);
          CALL GCLOSE (DCBPTR);
          END;
/*
//LKED.SYSLIB DD DISP=SHR
//        DD     DSN=AALIB,DISP=SHR,UNIT=PACK,
//               VOL=(PRIVATE,RETAIN,SER=DP5010)
//GO.DISP DD    UNIT=DISPLAY
```

```
000000000111111111122222222223333333333444444444455555555556666666666777777777 8
123456789012345678901234567890123456789012345678901234567890123456789012345678 0
```

Figure 5.

```
                    GRASP_STANDARD_INCLUDE_SET_:_AAINC(GRASPINC)
01/13/70  08:47:15
00000000011111111112222222222333333333344444444455555555556666666666777777777788
12345678901234567890123456789012345678901234567890123456789012345678901234567890
%       DCL    (GAAP#,GAQ#,GCLOSE#,GDAP#,GERASE#,GLINENO#,GNCP#,                00000000
                GNUNITS#,GOPEN#,GREAD#,GUNIT#,GWAIT#,GWRITE#) CHAR;             00000010
%       DCL    GCENT# CHAR;                                                     00000020
%       DCL    GALL# CHAR;                                                      00000030
%       GAAP#     = 'DCL GAAP ENTRY (POINTER,POINTER,                           00000040
                                    FIXED BIN(31))';                            00000050
%       GAQ#      = 'DCL GAQ ENTRY (POINTER,CHAR(1),BIT(1),                     00000060
                                   FIXED BIN(31),FIXED BIN(31))';               00000070
%       GCLOSE#   = 'DCL GCLOSE ENTRY (POINTER)';                               00000080
%       GDAP#     = 'DCL GDAP ENTRY (POINTER,FIXED BIN(31))';                   00000090
%       GERASE#   = 'DCL GERASE ENTRY (POINTER,CHAR(1)VAR,POINTER,              00000100
                                     FIXED BIN(31),FIXED BIN(31))';             00000110
%       GLINENO#  = 'DCL GLINENO ENTRY (FIXED BIN(31))                          00000120
                                   RETURNS (CHAR(1))';                          00000130
%       GNCP#     = 'DCL GNCP ENTRY (POINTER) RETURNS (FIXED BIN(31))';         00000140
%       GNUNITS#  = 'DCL GNUNITS ENTRY(CHAR(8))                                00000150
                                   RETURNS(FIXED BIN(31))';                     00000160
%       GOPEN#    = 'DCL GOPEN ENTRY (POINTER,CHAR(8),CHAR(3)VAR,               00000170
                                     CHAR(5),FIXED BIN(31),,                    00000180
                                     FIXED BIN(31))';                           00000190
%       GREAD#    = 'DCL GREAD ENTRY (POINTER,CHAR(3)VAR,POINTER,               00000200
                                     CHAR(*)VAR,FIXED BIN(31),                  00000210
                                     FIXED BIN(31))';                           00000220
%       GUNIT#    = 'DCL GUNIT ENTRY(POINTER) RETURNS(FIXED BIN(31))';          00000230
%       GWAIT#    = 'DCL GWAIT ENTRY (POINTER,FIXED BIN(31))';                  00000240
%       GWRITE#   = 'DCL GWRITE ENTRY (POINTER,CHAR(3)VAR,POINTER,              00000250
                                      CHAR(*)VAR,FIXED BIN(31),                 00000260
                                      FIXED BIN(31))';                          00000270
%       GCENT#    = 'DCL GCENT CHAR(1) STATIC INIT(' ')';                       00000280
%       GALL#     = 'GAAP#;GAQ#;GCLOSE#;GDAP#;GERASE#;GLINENO#;                 00000290
                     GNCP#;GNUNITS#;GOPEN#;GREAD#;GUNIT#;GWAIT#;                00000300
                     GWRITE#;GCENT#';                                           00000310
--------------------------------------------------------------------------------
00000000011111111112222222222333333333344444444455555555556666666666777777777788
12345678901234567890123456789012345678901234567890123456789012345678901234567890
```

Figure 6.

```
DISPLAY : PROC OPTIONS(MAIN);

          SOURCE LISTING.

STMT LEVEL NEST
   1                DISPLAY : PROC OPTIONS(MAIN);                                    1
   2     1                  DCL   (DCBPTR,GACBPTR,DECBPTR) POINTER,                  2
                                COND BIT(1),                                        3
                                MIMSG CHAR(30) VARYING,                             4
                                R FIXED BIN(31);                                    5
   3     1                  DCL GOPEN ENTRY (POINTER,CHAR(8),CHAR(3)VAR,             7 1
                                 CHAR(5),FIXED BIN(31),,                            7 1
                                 FIXED BIN(31)) ;                                   7 1
   4     1                  DCL GAAP ENTRY (POINTER,POINTER,                         8 1
                                 FIXED BIN(31)) ;                                   8 1
   5     1                  DCL GWRITE ENTRY (POINTER,CHAR(3)VAR,POINTER,            9 1
                                  CHAR(*)VAR,FIXED BIN(31),                         9 1
                                  FIXED BIN(31)) ;                                  9 1
   6     1                  DCL GWAIT ENTRY (POINTER,FIXED BIN(31)) ;               10 1
   7     1                  DCL GLINENO ENTRY (FIXED BIN(31))                       11 1
                                 RETURNS (CHAR(1)) ;                                11 1
   8     1                  DCL GAC ENTRY (POINTER,CHAR(1),BIT(1),                  12 1
                                 FIXED BIN(31),FIXED BIN(31)) ;                     12 1
   9     1                  DCL GREAD ENTRY (POINTER,CHAR(3)VAR,POINTER,            13 1
                                 CHAR(*)VAR,FIXED BIN(31),                          13 1
                                 FIXED BIN(31)) ;                                   13 1
  10     1                  DCL GERASE ENTRY (POINTER,CHAR(1)VAR,POINTER,           14 1
                                  FIXED BIN(31),FIXED BIN(31)) ;                    14 1
  11     1                  DCL GDAP ENTRY (POINTER,FIXED BIN(31)) ;                15 1
  12     1                  DCL GCLOSE ENTRY (POINTER) ;                            16 1
  13     1                  DCL GCENT CHAR(1) STATIC INIT(' ') ;                    17 1
  14     1                  CALL GOPEN  (DCBPTR,'DISP','IDE','BASIC',1,0,R);        18
  15     1                  CALL GAAP   (GACBPTR,DCBPTR,R);                         19
  16     1          LOOP :  CALL GWRITE (DECBPTR,'EA',DCBPTR,                       20
                                 GLINENO(1)||'READY '||GCENT,1,R);                  21
  17     1                  CALL GWAIT  (DECBPTR,R);                                22
  18     1                  CALL GAC    (GACBPTR,'W',COND,1,R);                     23
  19     1                  CALL GREAD  (DECBPTR,'R',DCBPTR,MIMSG,1,R);             24
  20     1                  CALL GWAIT  (DECBPTR,R);                                25
  21     1                  PUT  EDIT   (MIMSG) (SKIP,A);                           26
  22     1                  IF MIMSG¬='STOP' THEN GO TO LOOP;                       27
  24     1                  CALL GERASE (DECBPTR,'',DCBPTR,1,R);                    28
  25     1                  CALL GWAIT  (DECBPTR,R);                                29
  26     1                  CALL GDAP   (GACBPTR,R);                                30
  27     1                  CALL GCLOSE (DCBPTR);                                   31
  28     1                  END;                                                    32
```

Figure 7.

(4) At the statement labelled LOOP, the writemode argument to the GWRITE routine is coded as 'EA' to indicate that a pre-write screen erase is to occur, followed by a line-addressed output operation. The GLINENO routine is used to supply the proper line code for line 1 of the display. The variable GCENT is used to supply the character code for the START symbol. The line will appear on the display as

READY ▶
▮

with the cursor immediately to the right of the START symbol.

(5) The wait mode of the GAQ routine is used to wait for the appearance of an attention at the device.

(6) The manual input mode of the GREAD routine is used to cause the characters entered by the operator to be transferred to the string variable MIMSG. When the call is made to GWAIT, the current length of the varying string MIMSG is set to the number of characters read from the device. The unit number specified in the call to GREAD is 1 (as it is in the calls to GWRITE and GAQ) because only one display is being serviced.

(7) When the program is complete, a call is made to GERASE to erase the screen, GDAP is called to terminate the processing of attentions by the control program, and the graphics DCB is closed by a call to GCLOSE.

(8) In the linkedit step of the job, a DD statement defining the library containing the GRASP routines must be concatenated to the SYSLIB DD statement in the PL1LFCLG catalogued procedure.

(9) In the go step of the job, a DD statement is included defining the 2260 device. As coded in the example, an available 2260 will be allocated to the program. If more than one 2260 is desired, the following DD statement may be used:

```
//GO.DISP  DD  UNIT=(2260-1,n)
```

where "n" is a decimal integer indicating the number of 2260's desired.

LIST OF REFERENCES

1. IBM Systems Reference Library, Operator Manual, 2260 Display Station, 2848 Display Control, 1053 Printer, Form C20-1688.

2. IBM Systems Reference Library, System/360 Operating System, Graphic Programming Services for 2260 Display Station (Local Attachment), Form C27-6912.

3. IBM Systems Reference Library, System/360 Component Description, 2260 Display Station, 2848 Display Control, Form A27-2700.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* The MITRE Corporation Bedford, Massachusetts	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

GRASP: A PL/I COMPATIBLE GRAPHICS SUBROUTINE PACKAGE FOR THE IBM 2260 DISPLAY STATION (LOCAL ATTACHMENT)
VOLUME I - INTRODUCTION AND USER'S MANUAL

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
N/A

5. AUTHOR(S) *(First name, middle initial, last name)*

Richard H. Bullen, Jr.

6. REPORT DATE MAY 1970	7a. TOTAL NO. OF PAGES 31	7b. NO. OF REFS 3
8a. CONTRACT OR GRANT NO. F19(628)-68-C-0365 b. PROJECT NO. 512A c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) ESD-TR-70-124, Vol. I 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* MTR-1768, Vol. I	

10. DISTRIBUTION STATEMENT This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES N/A	12. SPONSORING MILITARY ACTIVITY Directorate of Systems Design and Development, Electronic Systems Division, AF Systems Command, L. G. Hanscom Field, Bedford, Massachusetts

13. ABSTRACT

GRASP is a set of PL/I compatible subroutines which provide programming support for the IBM 2260 Display Station in local attachment; i.e., the attachment of a 2260 directly to a System/360 CPU channel via the IBM 2848 Display Control. The subroutines are coded in OS/360 Assembler Language and are reentrant. They permit the PL/I programmer to manipulate the 2260 as an I/O device in the same manner available to the Assembler Language programmer using the Graphics Access Method under OS/360 (with restrictions as noted in the Introduction to Volume I of this document). All errors, except those which normally result in OS/360 abnormal ends (ABENDS), are returned to the user via subroutine parameters. GRASP is designed to operate under the MFT configuration of OS/360.

Volume I of this document gives an overview of the 2260 and an introduction to the GRASP routines. Volume II gives detailed program specifications.

DD FORM 1 NOV 65 1473

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
COMPUTER DISPLAYS						
PROGRAMMING IBM 2260						
PL/I COMPATIBLE GRAPHICS SUBROUTINES						
PL/I GRAPHICS ACCESS METHOD						