A Subset of GPAK, a Graphics Package for the 2250

A group of FORTRAN callable routines for graphics applications utilizing the 2250, known as GPAK, are available on the Model 75. This memo is an attempt to describe a subset of this package as an introduction to the rather formidable GPAK Program Manual which details the more than 150 routines. This manual is available for reference from the SLAC Program Librarian, for those interested. The routines described here allow the user to generate pictures in core, display them from the 2250 buffer and handle attentions for user interaction. It is hoped that for some applications, this subset will suffice.

The user may divide the CRT face into a number of areas. He must create tables for these areas defining their boundaries on the screen and correlate them with core areas where the images to be displayed are generated. He must also obtain 2250 buffer space to be associated with each area. Although the user needn't concern himself with the exact format of these tables, he must know their content to make use of the GPAK routines. Therefore, the tables will be described first and then the routines themselves.

For each of a possible 32 (0-31) screen areas, there is an Image Generation Table (IGTBL) and an XYTBL relating a screen area to a 2250 buffer area and to a Graphic Data Output Area (GDOA) in core.

**IGTBL**

<table>
<thead>
<tr>
<th>Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>lower left (x-coord) (y-coord) of screen area</td>
</tr>
<tr>
<td>2</td>
<td>upper right (x-coord) (y-coord) in raster units (4096, 4096)</td>
</tr>
<tr>
<td>3</td>
<td>initial 2250 buffer loc. - end 2250 buffer loc.</td>
</tr>
<tr>
<td>4</td>
<td>buffer transfer loc. - next available buffer loc.</td>
</tr>
<tr>
<td>5</td>
<td>light pen status - light pen routine entry point</td>
</tr>
</tbody>
</table>
XYTBL

Word
1 Address of Graphic Data Output Area
2 current order mode - char. mode - screen/grid options - scale option - data type - automatic display indicator
3 current beam position \((X, Y)\) \{ in raster units
4 last end point \((X)\) \}
5 last end point \((Y)\)
6 lower left \((X-COORD)\) \} of screen area in user
7 upper right \((X-COORD)\) \} coordinate values
8 lower left \((Y-COORD)\)
9 upper right \((Y-COORD)\)

The current order mode is saved by GPAK to determine whether a graphic order to change mode is really necessary.

Routines are provided to set all conditions and the following are default.

char. mode: basic size, protected mode
data type: fixed point
scale option: none
screen/grid option: display all data within user defined area only
automatic display indicator: off

One other table, the Attention Table (ATTBL), must be built to reflect the user's use of the attention interrupts.

ATTBL

<table>
<thead>
<tr>
<th>STATUS</th>
<th>ADDRESS OF ROUTINE FOR HANDLING THE INTERRUPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ON/OFF Asynchronous Error</td>
</tr>
<tr>
<td>2</td>
<td>ON/OFF Light pen</td>
</tr>
<tr>
<td>3</td>
<td>ON/OFF End-order-sequence</td>
</tr>
<tr>
<td>4</td>
<td>ON/OFF End key</td>
</tr>
<tr>
<td>5</td>
<td>ON/OFF Function keybd (N/A)</td>
</tr>
<tr>
<td>6</td>
<td>ON/OFF Cancel key</td>
</tr>
</tbody>
</table>
If no interrupts are desired, this table must be initialized to the system routines for ignoring interrupts. One routine, MSKGEN, is useful for generating a mask to set the status. The routines, briefly described, are:

**Initialization and Modification**

<table>
<thead>
<tr>
<th>*</th>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INIGPK</td>
<td>Open the 2250 DCB</td>
</tr>
<tr>
<td>2</td>
<td>STATTN</td>
<td>Set addresses of interrupt routines in the ATTBL</td>
</tr>
<tr>
<td>4</td>
<td>MDATTN</td>
<td>Modify ATTBL</td>
</tr>
<tr>
<td>5</td>
<td>MKATTN</td>
<td>Set status codes in ATTBL</td>
</tr>
<tr>
<td>6</td>
<td>IGINIT</td>
<td>Initialize IGTBL for all areas</td>
</tr>
<tr>
<td>7</td>
<td>MKIGT</td>
<td>Mask light pen status in IGTBL</td>
</tr>
<tr>
<td>8</td>
<td>STAREA</td>
<td>Get/release a 2250 buffer area</td>
</tr>
<tr>
<td>10</td>
<td>BUFSET</td>
<td>Get/release all 2250 buffer areas</td>
</tr>
<tr>
<td>11</td>
<td>STGDOA</td>
<td>Assign GDOA for image generation area</td>
</tr>
<tr>
<td>13</td>
<td>STIGT</td>
<td>Set IGTBL (define screen area)</td>
</tr>
<tr>
<td>15</td>
<td>SCALE</td>
<td>Set scale option</td>
</tr>
<tr>
<td>16</td>
<td>CSIZE</td>
<td>Set char. size and mode</td>
</tr>
<tr>
<td>17</td>
<td>SGOPIT</td>
<td>Set screen/grid option</td>
</tr>
<tr>
<td>18</td>
<td>STAUTD</td>
<td>Set automatic display indicator</td>
</tr>
<tr>
<td>19</td>
<td>TRANS</td>
<td>Set screen area limits in user coordinates</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(if not called, screen limits are used)</td>
</tr>
<tr>
<td>20</td>
<td>STDATAA</td>
<td>Set data type (fixed, floating point) referring to user's data</td>
</tr>
</tbody>
</table>

**Image Generation** (graphic orders and data to GDOA)

<table>
<thead>
<tr>
<th></th>
<th>Routine</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>STPOS</td>
<td>Position beam</td>
</tr>
<tr>
<td>23</td>
<td>POINTD</td>
<td>Point</td>
</tr>
<tr>
<td>26</td>
<td>LINED</td>
<td>Line</td>
</tr>
<tr>
<td>29</td>
<td>ARCD</td>
<td>Arc</td>
</tr>
<tr>
<td>32</td>
<td>CIRCLE</td>
<td>Circle</td>
</tr>
<tr>
<td>34</td>
<td>MSGBLD</td>
<td>Position beam and character string</td>
</tr>
</tbody>
</table>

If the automatic display indicator has been set, the buffer will also be started.

* Numbers refer to attached write-ups.
### Display

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>DISPLY</td>
</tr>
<tr>
<td>38</td>
<td>KEYIN</td>
</tr>
<tr>
<td>43</td>
<td>UPMENU</td>
</tr>
</tbody>
</table>

- **DISPLY**: Write GDOA to 2250 buffer and start buffer
- **KEYIN**: Enable a message from the keyboard; read it upon receipt of END KEY interrupt
- **UPMENU**: Display a list of light buttons, wait for lightpen interrupt and return number of item hit

### Attention Handling

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>GWAIT</td>
</tr>
<tr>
<td>49</td>
<td>TATTN</td>
</tr>
</tbody>
</table>

- **GWAIT**: Poll for attentions; if attention pending, execute appropriate routine from the ATTLB, else wait
- **TATTN**: Poll for attentions; if attention pending, execute appropriate routine, else return

### Data Handling

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>GETCHR</td>
</tr>
<tr>
<td>52</td>
<td>FIXBCD</td>
</tr>
<tr>
<td>53</td>
<td>FLPBCD</td>
</tr>
<tr>
<td>55</td>
<td>BCDFLP</td>
</tr>
<tr>
<td>57</td>
<td>MSKGEN</td>
</tr>
<tr>
<td>58</td>
<td>RCODE</td>
</tr>
<tr>
<td>59</td>
<td>GDOADP</td>
</tr>
<tr>
<td>60</td>
<td>DMPTBL</td>
</tr>
<tr>
<td>61</td>
<td>RUCONV</td>
</tr>
</tbody>
</table>

- **GETCHR**: Extract a character from a string (e.g. from a keyed-in message)
- **FIXBCD**: Integer to EBCDIC conversion
- **FLPBCD**: Floating point to EBCDIC conversion
- **BCDFLP**: EBCDIC to floating point conversion
- **MSKGEN**: Generate a mask (used for setting bits in tables)
- **RCODE**: Retrieve error code set by some of the GPAK routines
- **GDOADP**: Mnemonic dump of 2250 buffer or a GDOA to the printer
- **DMPTBL**: Hex dump of a GPAK table to the printer
- **RUCONV**: Convert raster units to user units or vice versa

### Plotting

<table>
<thead>
<tr>
<th>Code</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>STGLIM</td>
</tr>
<tr>
<td>65</td>
<td>STULIM</td>
</tr>
<tr>
<td>67</td>
<td>STGRID</td>
</tr>
<tr>
<td>69</td>
<td>STPLOT</td>
</tr>
<tr>
<td>71</td>
<td>STLABL</td>
</tr>
<tr>
<td>75</td>
<td>GRIDD</td>
</tr>
<tr>
<td>77</td>
<td>PLOTD</td>
</tr>
<tr>
<td>79</td>
<td>LABLD</td>
</tr>
</tbody>
</table>

- **STGLIM**: Set size of grid or plot in raster units (4096, 4096)
- **STULIM**: Set size of grid or plot in user units (floating point)
- **STGRID**: Set type of grid and number of divisions
- **STPLOT**: Set type of plot and screen/grid options
- **STLABL**: Set label parameters
- **GRIDD**: Generate orders and data for a grid in GDOA
- **PLOTD**: Generate orders and data for a graph in GDOA
- **LABLD**: Generate orders and data for labeling axes of a grid in GDOA
Data points supplied to the plotting routines must be in floating point mode. Calls to STGLIM and STGRID must precede a call to GRIDD. In fact, a call to STGLIM must precede calls to any of the generating routines. The default conditions for the various options are:

- Screen area - lower left (raster units) 500, 500
  - upper right (raster units) 3500, 3500
  - lower left (user units) 0.0, 0.0
  - upper right (user units) 10.0, 10.0
- Type of grid - linear in X and Y
- No. of divisions in grid in X and Y - 5,5
- Type of plot - vector and point
- Screen/grid option - eliminate all points and vectors off the screen
- No. of labels on an axis - 2
- No. of characters in a floating point number in the labels - 7
- No. of decimal places in a floating point number in the labels - 1
- Distance between axis and label - None
- Length of tick marks - 0

A typical sequence of calls to GPAK routines might be:

1. CALL INIGPK
   To open the 2250 DCB.
2. CALL STATTN (0,0,0,0,0,0)
   To initialize the ATTBL to ignore attentions.
3. CALL IGINIT
   To initialize all IGTBL's.
4. CALL WAITR (this is not a part of GPAK, but is in FORTLIB)
   To ready lower partition.
5. CALL MKATTN
   To turn on attentions desired in ATTBL.
6. CALL STIGT
   To define a screen area in an IGTBL
7. CALL MKIGT
   To turn on light pen status in IGTBL, if desired.
8. CALL STAREA  
   To establish a corresponding 2250 buffer area.

9. CALL STGDOA  
   To establish a corresponding Graphic Data Output Area (GDOA)

10. CALL any of the image generation or plotting routines  
    To generate orders and data in the GDOA.

11. CALL DISPLY  
    To start 2250 buffer and, optionally, erase the GDOA.

12. CALL GWAIT  
    To wait for attention.

13. CALL STAREA  
    To erase a buffer area.

Operating Information

The user should run in the 2250 partition (currently 100K bytes) but compilation should be done in the batch partition as GPAK is incompatible with FORTRAN G. The current version of GPAK employs Express Graphics so only 8K bytes of the 2250 buffer are available.

The following DD statements are required:

```
//IKED.SYSLIB       DD                  Col. 72
//                     DD DSNAME=SYS1.LINKLIB,DISP=OLD
//                     DD DSNAME=SYS1.GPAKLIB,DISP=OLD
//                     UNIT=2314,VOLUME=SER=SCFBO2
//GO.GPAKDD          DD UNIT=OD1

(if GDOADP is called)
//GO.BUFDUMP         DD SYSOUT=A

(if DMPTBL is called)
//GO.FT03FO01        DD SYSOUT=A,DCB=(,RECFM=FA, BLKSIZE=132)
```

The attached exercises are nonsensical but demonstrate the use of some of these routines. The detailed routine write-ups must be consulted for actual use. References to the function keyboard in the write-ups should be ignored.
COMPILER OPTIONS - NAME= MAIN,OPT=00,LINENE=50,SOURCE,EBDIC

C GPAK EXERCISE

C DISPLAY 3 AREAS AND ERASE EACH ONE UPON
C A LIGHTPEN HIT IN THAT AREA.
C TERMINATE WHEN ALL HAVE BEEN ERASED.
C DA1, DA2, DA3-LIGHTPEN INTERRUPT ROUTINES ERASE THE
C ASSOCIATED DISPLAY AREAS

ISN CCC2 DIMENSION KLR(2),LL(2),CA2(18),CA3(270),CHARS(20)
ISN CCC3 DIMENSION BCDITM(4)
ISN CCC4 EXTERNAL CA1,CA2,CA3,IGTBL

C INITIALIZE GPAK

ISN CCC5 CALL INITGPK
ISN CCC6 CALL STATTM(C,C,C,C,O,C)
ISN CCC7 CALL IGINIT
ISN CCC8 CALL WAITR

C INITIALIZE ATTBLE STATLS FOR LIGHTPEN

ISN CCC9 CALL MSKGEN(1,1,MASK,3)
ISN CCC10 CALL MKATTM(MASK)

C INITIALIZE IGTBL AREAS

C AREA 1 CONTAINS A KEYED IN MESSAGE

ISN CCC11 LL(1)=C
ISN CCC12 LL(2)=3CCG
ISN CCC13 KLR(1)=4C95
ISN CCC14 KLR(2)=4C95
ISN CCC15 CALL STIGT(1,LL,KLR,LCC,194,200,DA1)

C AREA 2 CONTAINS A LINE

ISN CCC16 LL(1)=C
ISN CCC17 LL(2)=C
ISN CCC18 KLR(1)=2CCG
ISN CCC19 KLR(2)=3CCG
ISN CCC20 CALL STIGT(2,LL,KLR,200,299,300,DA2)
ISN CCC21 LL(1)=2CCG
ISN CCC22 LL(2)=C
ISN CCC23 KLR(1)=4C95
ISN CCC24 KLR(2)=3CCG

C AREA 3 CONTAINS A GRID AND CIRCLE

ISN CCC25 CALL STIGT(3,LL,KLR,300,999,4092,DA3)

C INITIALIZE IGTBL STATLS CODES
C MAKES THE 3 AREAS LIGHT PEN ACTIVE

ISN CCC26 CALL MSKGEN(3,1,2,3,MASK,3)
ISN CCC27 CALL MKIGT(MASK,2)
INITIALIZE BUFFER AREAS AND TRANSFER POINTS WITHIN THE BUFFER. PUTS TRANSFERS IN BUFFER AT 4092 AND 8188—SEE WRITE-UP.

CALL BUFSIZE(MASK)

INITIALIZE IMAGE GENERATION OUTPUT AREAS.
CALL STGDOCA(2,DA2,18)
CALL STGDOCA(3,CA3,27D)

INITIALIZE MESSAGE AREA TO BE ALL BLANK CHARACTERS WHICH ARE DISPLAYED. A BLANK BIT PATTERN=01000000.

CALL MSGGEN(4,1,5,17,25,CHARS(1),3)
DO 1 I=2,20
CHARS(I)=CHARS(1)

GENERATE A LINE ORDER AND PLACE IT IN AREA 2 GDOA.
CALL LINED(500,600,1500,2000,2,IBFLCC,0,0,IRC)

GENERATE AREA 3.
CALL STGRID(0,6,6)
CALL STGLIP(LL(1),LL(2),KU(1),KU(2))
CALL GRIDD(I,BFLGC)
CALL CIRCLE(3CC,1000,500,64,3,IABLECC)
CALL RCDE(I)

IF(IRC.EQ.0)GTO TC 4

ERROR IN CIRCLE CALL—DISPLAY ERRCR VALUE
CALL FBCD(IRC,BCDITM)
CALL MSGBLD(1,20CC,1000,BCDITM,16,IBFLCC)

DISPLAY ALL SCREEN AREAS
CALL DISPLAY(2,0)
CALL DISPLAY(3,0)

OPERATOR KEYS IN A MESSAGE WHEN HE SEES THE CURSOR
IRDCNT=-2
CALL KEYIN(1,CHARS,74,10,IRDCNT,ERRCR)

INTERCEPT ON END KEY REMOVES THE CURSOR
NEXT TWO CALLS FOR DEBUGGING
CALL CMPTBL(1GTR,60)
CALL GOCAGP(3)

WAIT FOR LIGHT PEN DETECTION
KEY=C
CALL GWAIT
KEY=KEY+1
IF(KEY=3)10,2C,2C
STGP
ENC
LEVEL 2 FEB 67

COMPILER OPTILNS - NAME= MAIN,OPT=00,LINENCT=50,SOURCE,EBCDIC,

C
PROCESS A LIGHT PEN IN AREA1

SUBROUTINE DA1

C
SKIP OVER BUFFER AREA1 DISPLAY OORDERS

CALL SAREA(1,1)

ISN 0CC3
ISN 0CC4
ISN 0CC5
RETURN
END

LEVEL 2 FEB 67

COMPILER OPTILNS - NAME= MAIN,OPT=00,LINENCT=50,SOURCE,EBCDIC,

C
PROCESS A LIGHT PEN IN AREA2

SUBROUTINE DA2

C
SKIP OVER BUFFER AREA2 DISPLAY OORDERS

CALL SAREA(2,1)

ISN 0CC3
ISN 0CC4
ISN 0CC5
RETURN
END

LEVEL 2 FEB 67

COMPILER OPTILNS - NAME= MAIN,OPT=00,LINENCT=50,SOURCE,EBCDIC,

C
PROCESS A LIGHT PEN IN AREA3

SUBROUTINE DA3

C
SKIP OVER BUFFER AREA3 DISPLAY OORDERS

CALL SAREA(3,1)

ISN 0CC3
ISN 0CC4
ISN 0CC5
RETURN
END
The date was keyed in.
LEVEL 2 FEB 67
OS/360 FORTRAN H

COMPILER OPTIONS - NAME= MAIN, OPT=00, LINCNT=50, SOURCE, EBCDIC,

C
C PLOT A FUNCTION-TERMINATE UPON LIGHTPEN HIT
C
ISN 0002
DIMENSION Y(100), X(100)
ISN 0003
EXTERNAL AREA1, LTPNRT
C AREA1 DOES INITIALIZATION AND DEFINES SCREEN AREA
ISN 0004
CALL AREA1
C
ISN 0005
SET GRID LIMITS IN RASTER UNITS
CALL STGLIM(700, 500, 3200, 3000)
ISN 0006
SET GRID LIMITS IN USER UNITS (FLGATING POINT)
CALL STULIM(0., 0., 500., 500.)
ISN 0007
SET THREE DIVISIONS IN GRID
CALL STGRID(0, 3, 3)
C
ISN 0008
SET PLOT TO BF POINT MODE
CALL STPLOT(0, 1)
C
ISN 0009
SET GRID LABEL PARAMETERS
CALL STLABL(4, 10, 4, 100, 50)
C
ISN 0010
GENERATE GRID
CALL GRIDD(1, IRFLOC)
C
ISN 0011
GENERATE GRID COORDINATE LABELS
CALL LABLD(1, 0, IBFLCC)
ISN 0012
CALL LABLD(1, 1, IBFLCC)
C FUNCTION STOLEN FROM ADAM BOYARSKI
ISN 0013
DO 10 I=1, 100
ISN 0014
X(I)=5*I
ISN 0015
10 Y(I)=500.*EXP(-FLOAT(I-1)/10.)+200.,*EXP(-FLOAT(C(I-60)**2/2.))
C
ISN 0016
PLOT FUNCTION
CALL PLOTD(1, X, Y, 100, IRFLOC)
C
ISN 0017
DISPLAY
ISN 0018
CALL GWAIT
ISN 0019
STOP
ISN 0020
END
SUBROUTINE AREAl

C DEFINE ENTIRE SCREEN TO BE AREAl
C ENABLE LIGHTPEN ACTION IN AREAl
C USER MUST SUPPLY LTPNRT TO PROCESS INTERRUPT

EXTERNAL LTPNRT
INTEGER LL(2)/0,0/,KUR(2)/4095,4095/,OAREA(1000)
CALL INIGPK
CALL STATTN(0,0,0,0,0)
CALL IGINIT
CALL WAITR
CALL STIGT(1,LL,KUR,10C,4088,4092,LTPNRT)
CALL MSKGEN(1,1,MASK,3)
CALL MKATTN(MASK)
CALL MKIGT(MASK,2)
CALL STGDA(1,OAREA,1000)
CALL STAREA(1,1)
RETURN
END

SUBROUTINE LTPNRT

C LIGHTPEN INTERRUPT ROUTINE STOPS DISPLAY
CALL STAREA(1,1)
RETURN
END
NAME: INIGPK - Initialize GPAK

FUNCTION: This routine opens the GPAK Data Control Block (DCB), and clears any residual attentions.

CALLING FORMAT:

Fortran: CALL INIGPK
Assembly: CALL INIGPK

DETAILED DESCRIPTION:

This routine opens the GPAK Data Control Block (DCB), and clears any residual attentions. GPAK tables have been assembled with all status codes set to OFF and routine addresses set to the default conditions.

COMMENTS:

In an application program, a call to INIGPK must be provided prior to execution of any other GPAK routine. If a second call is made to INIGPK, the function is repeated except that all pending attentions will be cleared. No change will be made to GPAK or the application program.

If INIGPK is not called and an Input/Output action is initiated, OS/360 will cause an ABEND dump with the system completion code indicating that a DCB was not opened. The name of the GPAK DCB is GPAKDD. A DD card with this name must be included in the control cards (see Part III).

* "System Completion Code = 400"
NAME: STATTN - Setup Attention Table

FUNCTION: Initialize the attention table with addresses of the routines to handle light pen detect, end order sequence, end key, function key, and cancel key. Asynchronous errors are handled directly by GPAK.

CALLING FORMAT:

Fortran: CALL STATTN (ae, lpd, eos, end, fk, cancel)
Assembly: CALL STATTN, (adae, adlpd, adeos, adend, adfk, adcncl)

DESCRIPTION OF PARAMETERS:

ae = name of asynchronous error routine or "0"
lpd = name of light pen detect routine or "0"
eos = name of end order sequence routine or "0"
end = name of end key routine or "0"
fk = name of function key routine or "0" (N/A)
cancel = name of cancel key routine or "0"
adae = name of word containing address of ae or "0"
adlpd = name of word containing address of lpd or "0"
adeos = name of word containing address of eos or "0"
adend = name of word containing address of end or "0"
adfk = name of word containing address of fk or "0"
adcncl = name of word containing address of cancel or "0"

DETAILED DESCRIPTION:

Unless the user wishes to supply his own ATTBL table and associated routines via a CALL to GIATBL, he must give a CALL to STATTN at least once in a program. The CALL to STATTN must be given prior to enabling any attentions.

The user's attention table is initialized by placing the addresses of the ae, lpd, eos, end, fk, and cancel routines into the six locations beginning at ATTBL. If the argument is zero, no change is made in the corresponding table word. Status codes are left intact when the addresses are stored.

STATTN passes the address of ATTBL to GPAK by a CALL to GIATBL.
COMMENTS:

At load time the GPAK attention table will have all status codes set OFF and the following attention routine addresses:

\[ \text{ae} = \text{XATTBL (See below)} \]
\[ \text{lpd} = \text{SSLP. This is a routine which scans the IGTBL looking for associated logical sections. (See Chapter 1.a.6)} \]
\[ \text{eos} = \text{XATTBL (See below)} \]
\[ \text{end} = \text{XATTBL (See below)} \]
\[ \text{fk} = \text{SSFK. This is a routine which uses the FKTBL to enter a user routine for a specific function key. (See Chapter 1.a.7.)} \]
\[ \text{cancel} = \text{XATTBL (see below)} \]

XATTBL is an internal routine which will act as if the associated attention was disabled.

Normally only one CALL to STATTN need be given in a program. If changes to particular attentions are subsequently required the user may give a CALL to MDATTN. Once one of the GPAK routine addresses has been replaced, a subsequent CALL to STATTN with an associated parameter of zero will not restore the original setting. To do this the user must issue a CALL to STATTN or MDATTN with the associated routine symbolic name as shown above. Since these routines are in GPAK, an EXTERN card must be included in the application program.
NAME: MDATTN - Modify an Attention Table Routine Address

FUNCTION: Modify a particular entry routine address in the attention table.

CALLING FORMAT:

Fortran: CALL MDATTN(index, rtnname)
Assembly: CALL MDATTN, (index, adrtn)

DESCRIPTION OF PARAMETERS:

index = Integer code of routine to be modified
  0 = asynchronous error
  1 = light pen detect
  2 = end of sequence
  3 = end key
  4 = function key
  5 = cancel key

rtnname = Name of associated routine
adrtn = Name of word containing address of associated routine.

ERROR CONDITIONS:

If the index value is greater than 5, or negative, the table is not altered, and the standard return code subroutine RCODE is set with a value of '1'.

DETAILED DESCRIPTION:

The user's attention table is modified by placing the address of rtnname (adrtn) into the appropriate entry of ATTBL. The status code is unchanged. See the description of STATTN for initial setting of ATTBL.
NAME: MKATTN - Mask Attention Table Status Codes

FUNCTION: Set attention table status codes.

CALLING FORMAT:

Fortran: CALL MKATTN (mask)
Assembly: CALL MKATTN, (mask)

DESCRIPTION OF PARAMETERS:

mask is a one word parameter specifying the new ON/OFF status for each entry in the attention table, ATTBL.

A bit of 1 turns the status code ON, and is said to 'enable' the attention. A bit of zero turns the status code OFF, and is said to 'disable' the attention.

DETAILED DESCRIPTION:

Each bit of input mask word (positions 0-5) is used to set the status code (byte zero) of its corresponding entry top to bottom in the attention table. The status codes control activation of the routines whose entry points are stored in the address portion of each table word.

COMMENTS:

The Fortran programmer can generate the desired mask word using the GPAK utility routine MSKGEN (2. f. 9).

ERROR CONDITION:

In assembly language programs a specification error in MKATTN will occur if the parameter (mask) is not on a full word boundary.
NAME: IGINIT - Initialize the Image Generation Table and the Buffer.

FUNCTION: Initialize all 32 areas of the IGTBL.

CALLING FORMAT:

Fortran: CALL IGINIT
Assembly: CALL IGINIT

DETAILED DESCRIPTION:

Each of the 32 table areas in the IGTBL are initialized, and the screen is erased. All status codes in the IGTBL are turned OFF disabling light pen detects. Initialization of a table area sets the sb=0. The reserved buffer areas are loaded and buffer execution started. This is accomplished by a CALL to BUFSET.

COMMENTS:

This routine must be called before attempting to set up the IGTBL. It will reset any previous sb or status code entries, either from a prior user or due to a non-cleared core.

No cells in the IGTBL except the sb and status codes are affected. If memory was not cleared the other cells will contain undetermined values.
NAME: MKATTN - Mask Attention Table Status Codes

FUNCTION: Set attention table status codes.

CALLING FORMAT:

Fortran: CALL MKATTN (mask)
Assembly: CALL MKATTN, (mask)

DESCRIPTION OF PARAMETERS:

mask is a one word parameter specifying the new ON/OFF status for each entry in the attention table, ATTBL.

A bit of 1 turns the status code ON, and is said to 'enable' the attention. A bit of zero turns the status code OFF, and is said to 'disable' the attention.

DETAILED DESCRIPTION:

Each bit of input mask word (positions 0-5) is used to set the status code (byte zero) of its corresponding entry top to bottom in the attention table. The status codes control activation of the routines whose entry points are stored in the address portion of each table word.

COMMENTS:

The Fortran programmer can generate the desired mask word using the GPAK utility routine MSKGEN (2. f. 9).

ERROR CONDITION:

In assembly language programs a specification error in MKATTN will occur if the parameter (mask) is not on a full word boundary.
NAME: IGINIT - Initialize the Image Generation Table and the Buffer.

FUNCTION: Initialize all 32 areas of the IGTBL.

CALLING FORMAT:

Fortran:   CALL IGINIT
Assembly:  CALL IGINIT

DETAILED DESCRIPTION:

Each of the 32 table areas in the IGTBL are initialized, and the screen is erased. All status codes in the IGTBL are turned OFF disabling light pen detects. Initialization of a table area sets the sb=0. The reserved buffer areas are loaded and buffer execution started. This is accomplished by a CALL to BUFSET.

COMMENTS:

This routine must be called before attempting to set up the IGTBL. It will reset any previous sb or status code entries, either from a prior user or due to a non-cleared core.

No cells in the IGTBL except the sb and status codes are affected. If memory was not cleared the other cells will contain undetermined values.
NAME: MKIGT - Mask the IGTBL Status Codes

FUNCTION: Set the Image Generation Table's status codes that control transfers to the various attention routines on light pen detections.

CALLING FORMAT:

Fortran: CALL MKIGT (mask, iflag)
Assembly: CALL MKIGT, (mask, iflag)

DESCRIPTION OF PARAMETERS:

mask = integer word with a binary bit corresponding to each of the 5 word table areas in the IGTBL.
iflag = 0: 'OR' the mask bit into the table status code.
= 1: 'AND' the mask bit into the table status code.
= 2: place the mask bit value in the table status code.

DETAILED DESCRIPTION:

This routine interprets the mask word and inserts into the Image Generation table status codes the mask bit corresponding to the appropriate table areas. The left most bit in the mask word corresponds to table area 0, etc. All 32 bits are interpreted.

Mask bit = 1 sets the status code to ON and allows control to go to the attention routine named in IGTBL upon a light pen detect in the corresponding screen area.
= 0 sets the status code to OFF for iflag = 1 or 2. Light pen detects in the corresponding screen area are effectively ignored. An iflag of "0" and a mask bit of "0" leave the status code set as it was before the CALL to MKIGT.

COMMENTS:

The Fortran user should use MSKGEN (2.f. 9) to setup the mask word.
NAME:             STAREA - Setup a Buffer Area

FUNCTION:         Modify the 2250 buffer for one area only and start buffer

                    execution.

CALLING FORMAT:

Fortran:         CALL STAREA (iarea, iflag)
Assembly:       CALL STAREA, (iarea, iflag)

DESCRIPTION OF PARAMETERS:

iarea = area index of the specified buffer area.
iflag = Integer code for transfer order placement.
   (0): Reset the tp (transfer point) of the existing
        transfer order to the current value of the
        IGTBL.
   (1): Place a transfer order to the tp at the sb
        (start of buffer area).

DETAILED DESCRIPTION:

The indicated buffer area transfer order is reset accord­
        ing to the input flag value. An iflag of '0' resets the
        transfer point of the existing transfer order, (GTRU),
        which is located at the point in the buffer where the next
        orders for that iarea are to be loaded (i.e., the current
        buffer load point). If iflag is '1', effective erasing takes
        place by inserting the transfer order to the tp at the
        beginning of the buffer area (sb). Finally, the buffer is
        restarted.

ERROR CONDITIONS:

RCODE = 0; Normal return.
RCODE = 1; iarea>31 or negative.
RCODE = 20; sb = 0.

COMMENTS:

If sb=0, or tp ≤ eb (end of buffer area), no
reset occurs, and control is returned to the calling
program.
A buffer area can be skipped over by calling STIGT for the preceding area, altering its tp in the IGTBL to a point beyond the area to be skipped and then calling STAREA specifying the preceding area and an iflag=0.
NAME: BUFSET - Setup all 2250 Buffer Areas

FUNCTION: Set up all the 2250 Mod I buffer areas using current IGTBL values as indicated by the mask parameter; initialize all GPAK system areas in buffer; start the buffer execution.

CALLING FORMAT:

Fortran: CALL BUFSET (mask)
Assembly: CALL BUFSET, (mask)

DESCRIPTION OF PARAMETERS:

mask = a one word parameter with a binary bit corresponding to each of the table areas in the IGTBL.

DETAILED DESCRIPTION:

The first 100 bytes of the buffer are initialized; this area contains the GPAK light pen tracking orders. Each IGTBL area is then examined as follows:

If \(sb = 0\), or \(tp < eb\), no orders are stored; otherwise,

if the mask bit=0, the tp of the existing transfer order (at blp) is reset to the current tp value of the IGTBL.

If the corresponding input mask bit=1, a transfer to the tp (transfer point) is stored at the sb (start of buffer area).

Finally, a transfer order to byte 002 (4 bytes) is stored at location 4092. If the user has an 8K buffer the same order is also stored at location 8188. This allows a program set up for a 4K 2250 to be run on an 8K 2250.

COMMENTS:

8K buffer users should use BUFSET cautiously, since the writing of the GTRU at 4092 could conceivably overwrite the user's orders. The use of STAREA in place of BUFSET avoids this difficulty.
NAME: STGDOA - Setup a Graphic Data Output Area

FUNCTION: Assign a memory area as a graphic data output area (GDOA).

CALLING FORMAT:

Fortran: CALL STGDOA (iarea, array, length)
Assembly: CALL STGDOA, (iarea, array, length)

DESCRIPTION OF PARAMETERS:

d = area index (0-31).
a = start address of GDOA.
l = length (words) of GDOA.

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions;
0 = Normal return.
1 = area index value is greater than 31 or negative.
2 = length is less than 8 words.

DETAILED DESCRIPTION:

GPAK order generating routines (see chapters 2. c. and 2. d.) place orders sequentially into a CPU area (GDOA) specified by the STGDOA CALL. The same GDOA area may be used for two or more buffer areas. The first seven words of the GDOA are used by GPAK as a control table area and is initialized by this call. Orders will be written out of this GDOA:

1. When a CALL to DISPLAY is given; or
2. When enough orders have accumulated to cause an overflow past the length specified in the third parameter.
3. When a GDOA is being shared by two IGTBL areas and contains orders for one IGTBL area when orders are to be generated for the other IGTBL area.
DETAILED DESCRIPTION: (continued)

In either case, 4 bytes will be added to the orders placed in the buffer. These 4 bytes represent a graphic transfer order (GTRU) with a buffer address found in the TP field of the associated IGTBL area table. This causes the 2250 to jump to the next area in the buffer after the orders have been executed. In addition, tables are modified when orders are placed into the buffer as follows:

1. The next order generated for this GDOA will be placed in the eighth word of the buffer (i.e., word after the control section).
2. When the next overflow or CALL to DISPLAY occurs, the orders in the GDOA will be placed in the 2250 buffer starting at the GTRU previously written into that buffer area.

The GDOA can be effectively initialized to the beginning of the buffer area (sb) via a CALL to STAREA with an iflag of "1".

EXTERNAL ROUTINES OR TABLES USED:

XYTBL

COMMENTS:

The size of the GDOA will depend on the user's application. It need not be the same size as the buffer area. Since GPAK uses 7 words at the beginning of the GDOA for a control section and 2 words at the end (for a GTRU) the GDOA must be 9+n where n is the number of words into which orders are to be placed. Care should be taken so that n is at least as large as the largest single element to be generated. When using the plotting routines or ARCD/CIRCLE routines the GDOA \( \geq \frac{133}{532} \) words (\( \frac{32}{8} \) bytes).

If STGDOA is not called, each area in the IGTBL is preset with a common 500-byte GDOA. However, this default GDOA may not be used if the user is calling ARCD, CIRCLE, or the plotting routines GRIDD, PLOTD, and LABELD.
NAME: STIGT - Setup the Image Generation Table for One Area

FUNCTION: Set up a user defined screen display area and allocate the corresponding 2250 buffer area.

CALLING FORMAT:

Fortran: CALL STIGT (iarea, ll, kur, ksb, keb, ktp, rtnnam)
Assembly: CALL STIGT, (iarea, ll, kur, ksb, keb, ktp, adrtm)

DESCRIPTION OF PARAMETERS:

iarea = area index (0-31) assigned by the user to the screen area and the corresponding table area within the IGTBL. This code number must be used whenever referencing this table area of the IGTBL.

ll = name of a 2 word integer array containing the x and y raster unit values of the lower left corner of the rectangular screen area.

kur = name of a 2 word integer array containing the x and y raster unit values of the upper right corner of the rectangular screen area.

ksb = start address in the 2250 buffer where the screen area's orders will begin. sb must be >= 100. (See Comments.)

= 0 if this IGTBL section is to be closed. (A closed buffer area is defined as currently inactive.)

keb = end address of the 2250 buffer area.

ktp = transfer point; the buffer address transferred to (tp>eb) after orders in this buffer area have been executed.

rtnnam = name of attention handling routine where control will be passed upon a light pen detect in this screen area.

adrtm = name of word containing the address of the attention handling routine for a light pen detect in this screen area.

COMMENTS:

This routine interprets the area index and places each parameter into its respective position in the five word table area associated with each screen area. A negative parameter for ll, ur, sb, eb, or tp, or a zero parameter
in place of rtnam will cause no alteration of that entry in the IGTBL. If sb ≠ 0, that is, to reposition an active buffer area, the buffer (blp, automatically set by GPAK) load point is set to this new sb. The next set of orders generated for this area will be placed starting at sb.

The IGTBL is assembled with STIGT. See the writeup of IGINIT (Chapter 2.a.1.) for description of table initialization. The status codes in the IGTBL are set by the MKIGT subroutine and not altered by STIGT.

The system transfers control to buffer location 100 (first available user area), so that the user must call STIGT with sb ≥ 100. The user must have an active IGTBL area with sb = 100 at all times. The user must have an active tp = the last 4 bytes of the buffer where GPAK stores a recycling transfer.

STIGT does not directly affect the buffer. Therefore, a call to STAREA or BUFSET must be used after a CALL to STIGT to implement the changes in the IG table caused by STIGT.
NAME: SCALE - Set Scaling Option

FUNCTION: Set XYTBL table with indicated scaling option.

CALLING FORMAT:

CALL SCALE (iarea, iflag, irc)

DESCRIPTION OF PARAMETERS:

iarea  Area index (0-31).
iflag  (0): No scaling.  
       (1): Scaling.
irc    Return code parameter for error conditions.

ERROR CONDITIONS:

The return code parameter (irc) is set with a value for the following conditions:
irc = 0; Normal return.
irc = 1; Area index greater than 31, or less than zero.
irc = 6; Iflag value not in the range 0-1.

DETAILED DESCRIPTION:

The scaling indicator for "iarea" in the XYTBL table is set according to the input flag.

EXTERNAL ROUTINES OR TABLES USED:

XYTBL table.

COMMENTS:

If SCALE is not called, the default condition is no scaling.
NAME: CSIZE - Set Character Size

FUNCTION: Set XYTBL table with indicated character size and mode.

CALLING FORMAT:

CALL CSIZE (iarea, iflag, irc)

DESCRIPTION OF PARAMETERS:

iarea = Area index (0-31).
iflag = (0): Basic size, protected mode.
        (1): Large size, protected mode.
        (2): Basic size, unprotected mode.
        (3): Large size, unprotected mode.
irc = Return code parameter for error conditions.

ERROR CONDITIONS:

The return code parameter (irc) is set with a value for the following conditions:
irc = 0: Normal return.
irc = 1: Area index greater than 31, or less than zero.
irc = 6: iflag value not in the range 0-3.

DETAILED DESCRIPTION:

The character size indicator for "iarea" in the XYTBL table is set according to the input flag.

EXTERNAL ROUTINES OR TABLES USED:

XYTBL table

COMMENTS:

MSGBLD, which writes characters into the designated "iarea" will use the specified character size, until changed by another call to CSIZE.

If no character size is specified, the default condition is (0), basic size/ protected mode.
NAME: SGOPT - Set Screen/Grid Option

FUNCTION: Set XYTBL table with indicated screen/grid option.

CALLING FORMAT:

CALL SGOPT (iarea, iflag, irc)

DESCRIPTION OF PARAMETERS:

iarea = Area index (0-31).
iflag = (1): Display all data within area only.
        (2): Display all data within screen.
        (3): Display all data within area, but discontinue
             if outside screen.
        (4): Discontinue if any data is outside area.
        (5): Discontinue if any data is outside screen.
irc = Return code parameter for error conditions.

ERROR CONDITIONS:

irc = 0; Normal return.
irc = 1; Area index greater than 31, or less than zero.
irc = 6; iflag value not in the range 1-5.

DETAILED DESCRIPTION:

The screen/grid indicator for "iarea" in the IGTBL table is set according to the input iflag. The order generating routine will check each point position as it is calculated against the iflag option. For the discontinue options (iflag = 3, 4, or 5), if the point is outside the area or screen, order generation is terminated for that graphic element. For iflag = 1 or 2, the display is trimmed to the area or screen, respectively.

EXTERNAL ROUTINES OR TABLES USED:

XYTBL table.

COMMENTS:

If SGOPT is not called the default condition is (1).
NAME: STAUTD - Set GDOA for Automatic Display

FUNCTION: Sets the automatic display indicator in the XYTBL.

CALLING FORMAT:

CALL STAUTD(iarea, iflag, irc)

Where:

iarea = area index (0-31)
iflag = 0, set indicator off in XYTBL for this iarea
       1, set indicator on in XYTBL for this iarea
irc = location of return code parameter for error conditions.

DETAILED DESCRIPTION:

The automatic display indicator is checked by the image order generating routines and if the indicator is on, the orders are automatically transmitted to the 2250 buffer and the buffer started running. If the indicator is off the orders remain in the GDOA until either DISPLAY is called or the GDOA overflows, at which time all orders in the GDOA are transmitted to the 2250 buffer and the buffer is started running.

ERROR CONDITIONS:

irc = 0  Normal return.
irc = 1  iarea ≠ (0-31)
irc = 6  iflag ≠ (0, 1)
NAME: TRANS - Set Translation Scaling Factors

FUNCTION: Set IGTBL with scaling factors.

CALLING FORMAT:

CALL TRANS (iarea, ul, vl, ur, vr, irc)

DESCRIPTION OF PARAMETERS:

iarea = Area index (0-31).
(ul, vl) = Lower-left coordinates of the user's area,
in user values.
(ur, vr) = Upper-right coordinates of the user's area,
in user values.
irc = Return code parameter for error conditions.

ERROR CONDITIONS:

The return code parameter (irc) is set with a value for
the following conditions:

irc = 0; Normal return.
irc = 1; Area index greater than 31, or less than zero.

DETAILED DESCRIPTION:

The scaling factors for "iarea" in the XYTBL table are set
with input values (ul, vl) and (ur, vr). Data format must
conform with the type established previously as either real
(floating-point) or integer (fixed-point) by a call to STDATA.

EXTERNAL ROUTINES OR TABLES USED:

XYTBL table.

COMMENTS:

If TRANS is not called, the default condition is no scaling.
User coordinates will then be interpreted as screen coor-
dinates. **TRANS must be proceeded by a call to STDATA if
default conditions are not used.**

If ul > ur, or vl > vr, the point pairs are adjusted such that
the smaller point will be the lower-left coordinates of
the user's area.
NAME: STDATA - Set Data Mode

FUNCTION: Set IGTBL table with data type indicator.

CALLING FORMAT:

CALL STDATA (iarea, iflag, irc)

DESCRIPTION OF PARAMETERS:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iarea</td>
<td>Area index.</td>
</tr>
</tbody>
</table>
| iflag     | Data mode flag.  
|           | (1) Fixed-point, both U and V absolute.  
|           | (2) Floating-point, both U and V absolute.  
| irc       | Location of return code for error conditions.  

ERROR CONDITIONS:

The return code parameter (irc) is set with a value for the following conditions:

<table>
<thead>
<tr>
<th>irc</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal return.</td>
</tr>
<tr>
<td>1</td>
<td>Area index greater than 31, or less than zero.</td>
</tr>
<tr>
<td>6</td>
<td>iflag value not in the range 1-8.</td>
</tr>
</tbody>
</table>

DETAILED DESCRIPTION:

The data type indicator for "iarea" in the XYTBBL table is set to handle fixed or floating point data for absolute of U and V.

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table.

COMMENTS:

If STDATA is not called, the default condition is (1), fixed-point data mode with both U and V absolute values. STDATA must be called prior to calling TRANS if default is not used.
NAME: STPOS - Generate orders to position the beam.

FUNCTION: Generate the buffer orders to turn off the beam, and move it to the indicated position, and place the orders in the assigned GDOA.

CALLING FORMAT:

CALL STPOS (u, v, iarea, ibfloc, iflag)

DESCRIPTION OF PARAMETERS:

u, v = Coordinates of point, in user's units.

iarea = Area index (0-3l).

ibfloc = User variable where the relative buffer address of the new blp will be returned.

iflag = (0): Pure data (see below).

(1): Mixed data (see below).

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions:

0 = Normal return.

1 = Area index greater than 3l, or less than zero.

3 = User's GDOA is less than the number of bytes generated by STPOS. Orders not stored.

4 = Point outside the image area. Point not stored.

8 = Buffer full, point not stored.

20 = Start buffer (sb) = 0. Data not stored.

The return code is also placed in general register :5.

DETAILED DESCRIPTION:

The user's coordinates (u, v) are converted to raster units, and tested against image area limit values as stored in the IGTBL table. If the point falls outside the specified image area, a return code is set, and no orders are generated. If the point falls inside the specified image area, beam position values in the XY limit table are updated. The '(X,Y)LAST' values in the XYTBL table are always updated with the given input parameters, regardless of whether the point falls inside or outside the image area.
If iflag = 1, a 2-byte point mode command (GEPM) is generated. When iflag = 0, the current mode is examined. Pure coordinates (Bix, iy) are generated if the previous order was in vector or point mode, otherwise a vector mode command (GEVM) is generated. If the buffer load point (blp) address equals the start buffer (sb) address, then a vector mode command is produced regardless of the iflag value.

Before any new orders are generated, a check is made to determine if there is space available. If the new order string will overflow the end buffer value (eb), a return code is set, and no orders are issued.

Normal program completion consists of adding a transfer (GTRU tp) to the end of the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibflc contains the new buffer load point (blp) value minus the start buffer value (sb).

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, XYTBL table, GSVPLT subroutine, CKIG subroutine, DISPLY subroutine.

COMMENTS:

If the generated data will overflow the user's graphic output area, DISPLY is called to write out the GDOA contents to the buffer.

If two or more IGTBL areas share a common GDOA, the DISPLY subroutine is called to write out current GDOA contents each time the area code is switched. The GDOA is then reset and the generated orders placed in the start of the GDOA.

The user's coordinate values \((u, v)\) must be in the same data mode (fixed or floating-point) as specified by a previous call to STDATA. If STDATA is not called, the user's data is interpreted as fixed-point.
NAME: POINTD - Generate the orders for a Point.

FUNCTION: Generate the buffer orders for a point, and place in the assigned GDOA.

CALLING FORMAT:

CALL POINTD(u, v, iarea, ibfloc, iflag)

DESCRIPTION OF PARAMETERS:

u, v = Coordinates of point, in user's units.
iarea = Area index (0-31).
ibfloc = User variable where the relative buffer address of the new blp will be returned.
iflag = (0): Pure point data (see below).
          (1): Mixed point data (see below).

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions:
0 = Normal return.
1 = Area index greater than 31, or less than zero.
3 = User's GDOA is less than the number of bytes generated by POINTD. Orders not stored.
4 = Point outside the image area. Point not stored.
8 = Buffer full, point not stored.
20 = Start buffer (sb) = 0. Data not stored.
The return code is also placed in general register 15.

DETAILED DESCRIPTION:

The user's coordinates (u, v) are converted to raster units, and tested against image area limit values as stored in the IGTBL table. If the point falls outside the specified image area, a return code is set, and no orders are generated. If the point falls inside the specified image area, the beam position values in the XY limit table are updated. The 'XY LAST' values in the XYTBL table are always updated with the given input coordinates, regardless of whether the point falls inside or outside the image area.
DETAILED DESCRIPTION: (continued)

The input parameter, iflag, allows inclusion or exclusion of a GEPM mode command. A flag of "1" generates the 2-byte GEPM mode command. If iflag = 0, the current mode indicator is examined. Pure coordinates (ix, iy) are generated if the previous order was in point mode, otherwise mixed data (GEPM, ix, iy) is generated. If the buffer load point (BLP) address equals the start buffer (sb) address, then mixed data is produced regardless of the iflag value.

Before any new orders are generated, a check is made to determine if there is space available. If the new order string will overflow the end buffer value (eb), a return code is set, and no orders are issued.

Normal program completion consists of adding a transfer (GTRU tp) to the end of the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibfoloc contains the new buffer load point (blp) value minus the start buffer value (sb).

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, XYTBL table, GSVPLT subroutine, CKIG subroutine, DISPLY subroutine.

COMMENTS:

If the generated data will overflow the user's graphic output area, DISPLY is called to write out the GDOA contents to the buffer.

If two or more IGTBL areas share a common GDOA, the DISPLY subroutine is called to write out current GDOA contents each time the area code is switched. The GDOA is then reset and the generated orders placed at the start of the GDOA.
COMMENTS: (continued)

The user's coordinate values \((u, v)\) must be in the same data mode (fixed or floating-point) as specified by a previous call to STDATA. If STDATA is not called, the user's data is interpreted as fixed-point.

Under certain screen/grid options, the point is not rejected if outside the area border. See SGOPT (C.2.a.7) for a description of the various options.
NAME: LINED - Generate the orders for a Line.

FUNCTION: Generate the buffer orders for a line and place in the assigned GDOA.

CALLING FORMAT:

*Fortran:
CALL LINED(ul, vl, u2, v2, iarea, ibfloc, iflag, iptflag, irc)

*Assembly:
CALL LINED,(ul, vl, u2, v2, iarea, ibfloc, iflag, iptflag, irc)

*(see Comments)

DESCRIPTION OF PARAMETERS:

ul, vl = Coordinates of first point in user's units.

u2, v2 = Coordinates of second point in user's units.

iarea = Area index (0-31).

ibfloc = User variable where the relative buffer address of the blp will be returned.

iflag = Flag indicating pure or mixed line data.
(0): pure line data (see comments)
(1): mixed line data (see comments)

iptflag = Point replacement flag.
(0): No replacement.
(1): Replace first point with last (u, v) value.
(2): Replace second point with last (u, v) value.

ERROR CONDITIONS:

The return code parameter irc is set with a value for the following conditions:

irc = 0 Normal return.
irc = 1 Area index greater than 31, or less than zero.
irc = 3 User's GDOA is less than number of bytes generated by LINED. Orders not stored.
irc = 4 Line outside image area. Data not stored.
irc = 8 Buffer will overflow. Data not stored.
irc = 91 Floating point error.
irc = Invalid image area limits. xl>xr, or yl>yr
irc = Invalid user area limits. ul>ur, or vl>vr.
irc = 20 Start buffer (sb) = 0. Data not stored.

General register 15 also contains the return code value.
DETAILED DESCRIPTION:

The point replacement flag (iptflag) is tested for a value of 1 or 2. If "1", the first point pair is replaced by \((X, Y)\) LAST from the XYTBL, and if "2", the second point is replaced by \((X, Y)\) LAST. Each of the image generation routines update \((X, Y)\) LAST with the coordinates of the line end points, except LINED, which places the second point coordinates into \((X, Y)\) LAST prior to any trimming operations.

Coordinates of the line end points are tested against the image area limit values as stored in the appropriate IGTBL table area. If the line segment lies entirely outside the image area, a return code is set, and no orders are generated. Whenever any portion of the line segment crosses the area border, the excess is trimmed off.

The current beam position for each area is also retained by GPAK. If the start point of the line specified in LINED equals the current beam position, only 4 bytes are generated for the end point.

Before any new orders are generated, a check is made to determine if there is space available. If the new order string will overflow the end buffer value (eb), a return code is set, and no orders are generated.

Normal program completion consists of adding a GTRU(tp) to the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibloc then contains the new buffer load point (blp) minus the start buffer (sb).

EXTERNAL ROUTINES AND TABLES USED:

IGTBL table, GSVPLT subroutine, DISPLY subroutine, RUCONV subroutine.
COMMENTS:

If the iflag = 0, LINED examines the current mode. If the previous order was in vector mode, the 2 bytes to initiate vector mode are not generated. If the previous order was not in vector mode, a 2-byte vector mode code is generated.

If the iflag = 1, a vector mode code (2 bytes) is generated regardless of the mode of the previous order.

If the buffer load point (blp) address equals the start buffer (sb), then a vector mode order is generated, regardless of the value of iflag.

The iptflag replacement option should not be used if LINED is the first call for that area.

If the generated data will overflow the user's graphic data output area, DISPLAY is called to write out the GDOA contents to the buffer.

If two or more IGTBL areas share a common GDOA, the DISPLAY subroutine is called to write out current GDOA contents each time the area code is switched. The GDOA is then reset and the generated orders placed at the start of the GDOA.

Input values U1, V1, U2, and V2 must be in the same data mode as specified by a previous call to STDATA. If STDATA is not called, GPAK assumes that data for this area is in fixed-point mode.

* The Version I LINED calling sequence of seven parameters will operate correctly with the new program, provided that the input values are in raster units and no scaling is used. Also, assembly language users must utilize the "VL" notation as follows:
CALL LINED, (ix1, iy1, ix2, iy2, iarea, ibfloc, iflag), VL

Under certain screen/grid options, the image is not trimmed at the area border. See SGOPT (C.2.a.7) for a description of the various options.
NAME: ARCD - Generate orders for an Arc.

FUNCTION: Generate the buffer orders for an arc, and place them in the assigned GDOA.

CALLING FORMAT:

CALL ARCD(uc, vc, radius, itheta, igamma, ichord, iarea, ibfloc)

DESCRIPTION OF PARAMETERS:

uc, vc = Coordinates of center point, in user's units.
radius = Radius length, in user's units.
itheta = Start angle (integer degrees) ≥ 0.
igamma = Number of degrees of arc (integer ≥ 0, modulo 360).
ichord = Chord length in raster units. Must be ≥ 4, and less than \(2\pi R_{xy} \times (igamma/360)\), where \(R_{xy}\) is the radius value after scaling to raster units.
ialpha = Area index (0-31).
ibfloc = User variable where the relative buffer address of the new blp will be returned.

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions:

0 = Normal return.
1 = Area index greater than 31, or less than zero.
3 = User's GDOA is less than number of bytes generated by ARCD. Data not stored.
4 = Arc outside image area.
8 = Less than 10 bytes available in buffer area. Arc cannot be stored.
12 = Input data error. Radius or igamma equals zero, or ichord value is larger than arc length.
20 = Start buffer value (sb) = 0. Arc cannot be stored.
The return code is also placed in general register 15.
DETAILED DESCRIPTION:

The user values uc, vc, and radius are converted to raster units. Any chord segments falling outside the image area limits specified in the IGTBL are trimmed off. The beam position of the last vector generated is used to update beam values in the XY limit table. The arc end coordinates are calculated and stored in \((x, y)_{\text{last}}\) of the XYTBL table.

Normal program completion consists of adding a transfer, GTRU (tp), to the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibflc contains \((\text{blp} - \text{sb})\), the new buffer load point minus the start buffer address.

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, GVARC subroutine, CKIG subroutine, DISPLY subroutine, RUCONV subroutine.

COMMENTS:

If the generated data will overflow the user's graphic data output area, DISPLY is called to write out the GDOA contents to the buffer. The GDOA is then reset and the generated orders placed at the beginning of the GDOA.

If two or more IGTBL areas share a common GDOA the DISPLY subroutine is called to write out the current GDOA contents each time the area code is switched. The GDOA is then reset and the generated orders placed at the start of the GDOA.

A vector mode code is always generated at the beginning of each circle regardless of the preceding mode. It is recommended that the GDOA of the specified area be greater than 133 words (532 bytes). If the number of orders generated for a given arc take up < 125 words but more than the \((\text{GDOA}-7)\) words, a return code is set, and no circle is generated.
COMMENTS: (continued)

The user values uc, vc, and radius, must be in the same data mode (fixed or floating-point) as specified by a previous call to STDATA. If STDATA is not called, the user's data is interpreted as fixed-point.

Under certain screen/grid options, the image is not trimmed at the area border. See SGOPT (C. 2. a. 7) for a description of the various options.
NAME: CIRCLE - Generate the orders for a circle.

FUNCTION: Generates the buffer orders for a full circle, and places them in the assigned GDOA.

CALLING FORMAT:

CALL CIRCLE (uc, vc, radius, ichord, iarea, ibfloc)

DESCRIPTION OF PARAMETERS:

uc, vc = Coordinates of center point, in user's units.
radius = Radius length, in user's units.
ichord = Chord length in raster units. Must be \( \geq 4 \), and less than \( 2\pi R_{xy} \), where \( R_{xy} \) is the radius value after scaling to raster units.
iarea = Area index (0-31).
ibfloc = User variable where the relative buffer address of the new blp will be returned.

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions:
0 = Normal return.
1 = Area index greater than 31, or less than zero.
3 = User's GDOA is less than number of bytes generated by ARCD. Data not stored.
4 = Arc outside image area.
8 = Less than 10 bytes available in buffer area. Arc cannot be stored.
12 = Input data error. Radius or igamma equals zero, or ichord value is larger than arc length.
20 = Start buffer value (sb) = 0. Arc cannot be stored.
The return code is also placed in general register 15.

DEDALCE DESCRIPTION:

The user values uc, vc, and radius are converted to raster units. Any chord segments falling outside the image area limits specified in the IGTLBL are trimmed off. The beam position of the last vector generated is used to update beam values in the \( (x, y) \) limit table. The \( (x, y) \) last' values in the XYTBL table are updated with arc end point coordinates. \( x = uc + radius \), and \( y = vc \).
DETAILED DESCRIPTION: (continued)

Normal program completion consists of adding a transfer, GTRU (tp), to the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibflolc contains (blp-sb), the new buffer load point minus the start buffer address.

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, GVARC subroutine, CKIG subroutine, DISPLY subroutine, ARCD subroutine.

COMMENTS:

If the generated data will overflow the user's graphic data output area, DISPLY is called to write out the GDOA contents to the buffer. The GDOA is then reset and the generated orders placed at the beginning of the GDOA.

If two or more IGTBL areas share a common GDOA the DISPLY subroutine is called to write out the current GDOA contents each time the area code is switched. The GDOA is then reset and the generated orders placed at the start of the GDOA.

A vector mode code is always generated at the beginning of each circle regardless of the preceding mode. It is recommended that the GDOA of the specified area be greater than \(133\) words (\(538\) bytes). If the number of orders generated for a given arc take up \(125\) words but more than the (GDOA-7) words, a return code is set, and no circle is generated.

The user values \(uc, vc, \) and radius, must be in the same data mode (fixed or floating-point) as specified by a previous call to STDATA. If STDATA is not called, the user's data is interpreted as fixed-point.

Under certain screen/grid options, the image is not trimmed at the area border. See SGOPT (C.2.a.7) for a description of the various options.
NAME: MSGBLD - Generate Orders for a Message

FUNCTION: Generate the buffer orders to display a given string of characters and place the orders in the assigned GDOA.

CALLING FORMAT:

Fortran: CALL MSGBLD (iarea, uc, vc, chrloc, nochar, ibfloc)

Assembly: CALL MSGBLD, (iarea, uc, vc, chrloc, nochar, ibfloc)

DESCRIPTION OF PARAMETERS:

- **iarea** = area index
- **uc** = x-coordinate of beam position for center of first character in user units.
- **vc** = y-coordinate of beam position for center of first character in user units.
- **chrloc** = in main memory of the first character of the message to be generated.
- **nochar** = number of characters in message; if >74, the message is truncated.
- **ibfloc** = relative buffer address (returned in location ibfloc).

The routine takes the character string, starting at chrloc and writes nochar into the assigned GDOA. Normal program completion consists of adding a transfer order, GTRU(tp), to the order string. However, the tables are updated so that the next order generated for this area will overwrite the GTRU order. The relative buffer address value returned in the ibfloc parameter reflects the value associated with where the next order will be placed. Ibflor contains the (blp-sb), new buffer load point minus the start buffer address.

ERROR CONDITIONS:

The standard return code subroutine RCODE is set with a value for the following conditions:
ERROR CONDITIONS: (continued)

0 = Normal return.
1 = iarea not valid (#0-31)
3 = GDOA too small for character string
4 = Message is totally outside of the rectangle defined. Message is ignored.
5 = Message string will extend beyond the limits of the defined grid area. All characters beyond the limits will not be displayed.
8 = Message will extend beyond the eb of the specified area. Message is ignored.
20 = Specified buffer area has a sb=0. Message is ignored.

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, XYTBL table, RUCONV subroutine, CKIG subroutine, RCODE subroutine.

COMMENTS:

If the generated orders will overflow the user's GDOA, DISPLY is called to write out the GDOA's contents to the buffer before inserting the new orders. If the GDOA is assigned to more than one area and orders exist in the GDOA for another area, DISPLY is called to write out the GDOA's contents to the buffer. The GDOA is reset and the new orders placed in the GDOA.

The maximum amount of buffer space required for a message is equal to nochar plus eight bytes of mode and beam positioning data plus one null byte if the number of characters is odd.

Data type (raster units or user values) for uc and vc, current order mode, character mode, character size, and automatic display of message are all determined from information in the XYTBL.

If uc, vc equal the last beam position the six bytes of positioning orders are not generated. If the current order mode is the same character mode as requested and uc, vc equal the last beam position all eight bytes of positioning and mode setting orders are not generated.

Under certain screen/grid options, the image is not trimmed at the area border. See SGOPT (C.2.a.7) for a description of the various options.
NAME: DISPLY - Move Orders to the Buffer for Display

FUNCTION: Transfer a GDOA sequence of orders to a buffer area and start the display

CALLING FORMAT:

Fortran: CALL DISPLY (iarea, iflag)
Assembly: CALL DISPLY, (iarea, iflag)

DESCRIPTION OF PARAMETERS:

iarea = area index or -1
iflag=0; display orders and clear GDOA.
iflag=1; display orders but don't clear GDOA.

ERROR CONDITIONS:

RCODE = 0; Normal return.
RCODE = 1; The area index is greater than 31, or -1.
RCODE = 3; No orders in this GDOA. Display has been started.
RCODE = 6; iflag = 1 but BLP # SB

RCODE = 9; Orders in the associated GDOA are for another iarea code. Display is started.
RCODE = 20; Specified area has a SB=0. Display is started.

DETAILED DESCRIPTION:

If the iarea parameter is = -1, the display is started and control is returned to the CALLer. Otherwise the area index is used as an argument to find its corresponding graphic data output area (GDOA). The associated GDOA may be in one of three states:
1. There are no orders in the GDOA—the 2250 display is started.
2. There are orders in the GDOA associated with the specified iarea code—
   the orders are transferred from the GDOA to the buffer, the display is started and the iflag parameter is checked (see below).
DETAILED DESCRIPTION: (continued)

3. There are orders in the GDOA associated with another iarea code—
   the 2250 Display is started, but the orders in the GDOA are not transmitted to the 2250 buffer.

In case 2 above a GTRU (graphic transfer order) is placed immediately following the last order. This causes the 2250 to execute orders from the next buffer area.

The iflag is used when the GDOA contains orders associated with the specified iarea. As noted above, the orders are written into the buffer and the display is started. Then the iflag is interrogated. If iflag = 0, the GDOA and IGTBL are updated, so that the next order generated for this buffer area will be placed at the beginning of the GDOA (after the 7 word control section) and when placed in the buffer will overwrite the previous GTRU. If iflag = 1, the GDOA and IGTBL are not updated. Therefore, when new orders are generated they will be placed in the GDOA after the last order generated and when the orders are again sent to the buffer the same orders will go in the same place plus any additional orders added subsequently. If the GDOA overflows the orders are sent out automatically with iflag = 0. The error condition 6 indicates this situation but new orders added to the GDOA will be saved for user calls with iflag equal to 1.

EXTERNAL ROUTINES OR TABLES USED:

XYTBL, GWRITE macro
NAME: KEYIN - Enter alphanumerical keyboard data.

FUNCTION: To specify a message (maximum of 72 characters) which will be immediately displayed on the 2250 and be used to receive operator keyed data.

CALLING FORMAT:

Fortran:   CALL KEYIN(iarea, msgloc, nchars, icurpo,
                   irdcnt, irc [, iflag [, iread [, ix, iy]]] )

Assembly: CALL KEYIN, (iarea, msgloc, nchars, icurpo,
                    irdcnt, irc [, iflag [, iread [, ix, iy]]]), VL

DESCRIPTION OF PARAMETERS:

iarea = area index (0-31).
msgloc = location of the message to be displayed.
nchars = location of total number of characters to be displayed, and accepted from keyboard (fixed + variable), packed four per word.
icurpo=0: A string of blanks equal in length to the value of the "nchars" parameter will be displayed with the cursor under the first position.
\=-c: The number of characters specified by "nchars" and starting at "msgloc" will be displayed with the cursor under the cth position.
\=+c: The first (c-1) characters located at "msgloc" will be displayed followed by a number of blanks equal to the number specified by ("nchars" + 1) minus c. (e.g., in a 10 character message with curpos=5, the first 4 characters will be displayed followed by 6 blanks). The cursor will be in the cth position.
Note: a plus one and a zero for curpos are equivalent.
irdcnt = actual number of characters keyed in (return parameter).
   \neq -2 when no wait option is desired (specified as input in CALL KEYIN)
   = -2 when wait option is desired (specified as input in CALL KEYIN)
irc = location of return code parameter for error conditions.
iflag=0: only if alternate read area is specified.
=2: new ix and iy (for positioning start of message only.)
DESCRIPTION OF PARAMETERS: (continued)

iflag=4: combine iflag = 0 and 2.
iread = location of alternate read area.
ix = coordinate value for new x
iy = coordinate value for new y

If all parameters are specified, they must appear in the order shown in the calling format. If the iread, ix and iy are omitted then the iflag parameter must be omitted.

ERROR CONDITIONS:

Contents of irc = 0 normal return
    = 4 nchars greater than 72 - truncated to 72
    = 8 curpos greater than 72 - no setup and exit to caller.
    = 16 curpos outside range of nchars (i.e., curpos > nchars) - no setup and exit to caller.
    = 20 iarea less than 0
    = 24 iarea greater than 31
    = 28 SB = 0
    = 32 buffer area too small for number of characters requested.

Note: Error codes 20 to 32 result in no set-up and exit to caller.

DETAILED DESCRIPTION:

The user constructs a message which contains a constant part (i.e., not to be changed by the operator) and a variable part. KEYIN will cause the total message (both parts) to be displayed (See curpos). The constant part will be in a protected mode and the variable part will be in an unprotected mode. A cursor will be placed at the first position of the unprotected area.

KEYIN will start the string of characters to be displayed at either the position specified by the "ix" and "iy" parameters of the CALL or the "xll" and "yll" values in the IG Table specified by the "iarea" parameter. The center of the first character is assumed to be the x and y values in either case.
DETAILED DESCRIPTION: (continued)

This routine extracts the prior ENDKEY routine address from the ATTBL and inserts the entry point to "READAN" with the mask bits enabled.

When KEYIN is called with parameter irdcnt = -2 (wait option) a call GWAIT is issued automatically and upon returning to the user at the statement following the CALL KEYIN, the parameter irdcnt will contain the actual number of characters keyed in. If irdcnt is specified as anything other than -2, no CALL GWAIT is issued and control returns to the user's statement following the CALL KEYIN with irdcnt unchanged. In this case, the user must issue his own GWAIT to field the end key or other attentions.

When the END key is hit, the characters keyed in by the operator will be read into the area specified by the "iread" parameter or into the nth position of the original message (n = icurpo parameter) if the "iread" parameter is not specified. When control is returned to the caller, the parameter "irdcnt" will have the number of characters keyed in. The data keyed in is stored four char/word left justified.

Upon receipt of the end key attention the data extracted from ATTBL is restored to its original status prior to the CALL KEYIN. When the NO WAIT option is specified the user must not alter the END KEY parameter in ATTBL or the buffer parameters in IGTBL (KEYIN area only) until the attention has been received and processing completed.

KEYIN does not update the BLP. Therefore, any subsequent calls to image generation routines for that same area will overwrite the KEYIN orders and data in the buffer.

The message with the keyed in data will remain on the screen until the user resets the area specified by the iarea parameter. However, the cursor will be removed as soon as the END key is processed.
COMMENTS:

This routine has a self-contained GDOA which cannot be accessed by the user. The routine also handles the DISPLY function so the user should not CALL DISPLY for the orders. However, the user must allocate the buffer (2250) area in order to display (key in) data. The buffer may be allocated using the following algorithm:

When:

1. icurpo = 0
   Buffer length is an even number \( \geq (nchars + 14) \)
   where: \( nchars = \) total number of characters in string.

2. icurpo ≠ 0
   Buffer length is an even number \( \geq (nchars + 18) \)
   where: \( nchars = \) total number of characters in string.

Note: Buffer length is specified by the isb and ieb parameters in the CALL STIGT to setup the screen area.

As noted above, KEYIN automatically modifies the END key word in the ATTLBL without altering any other attentions. If the user has enabled any other attentions and also selects the WAIT option (see irdcnt parameter), a return can be made to the KEYIN CALL to GWAIT but the operator keyed data will not have been placed into CPU memory nor will the irdcnt parameter be updated. If this occurs, it automatically places KEYIN in the non-WAIT mode. It is the user's responsibility to be aware of this and perform either of the two following functions:

1. If the keyed in data is still desired, the user can issue his own CALL to GWAIT. If the END key attention occurs, the return to the user GWAIT will find the data properly stored in the CPU.

2. If the keyed data is no longer desired, then the user must issue the following:
   CALL READAN
   No parameters are necessary. This will clear the cursor and reset the KEYIN routine.
COMMENTS: (continued)

If the user chooses the no-WAIT option or is forced into this option as above, it is possible that he can give a second CALL to KEYIN without having cleared the first. This may result in two cursors being displayed causing unpredictable keyboard action. The user should never allow this to happen. The cursor will be automatically removed when the END key is processed. The user may cancel the CALL to KEYIN by issuing a "CALL READAN" as above and ignoring the values stored in irdcnt and the message input area.

The size of the alternate read area, if selected, should be \( \left( \text{nchars} + 3 - \text{curpos} \right) \mod 25. \)
NAME: UPMENU - Display a menu and identify item selected.

FUNCTION: Immediately displays a menu string and returns the number of the item detected with light pen to user.

CALLING FORMAT:

Fortran: CALL UPMENU (iarea, menloc, nitems, nchars, itemno, irc)
Assembly: CALL UPMENU, (iarea, menloc, nitems, nchars, itemno, irc)

DESCRIPTION OF PARAMETERS:

iarea = area index (0-31)
menloc = location of first byte in menu string.
nitems = number of distinct items separated by slashes.
nchars = total number of char in string (incl. slashes) packed 4 per word.
itemno = location where integer value denoting item detected will be stored (return parameter).

irc  = return code parameter for error conditions.

ERROR CONDITIONS:

The return code irc is set as follows:
0 = Normal return no error.
4 = nchars greater than 160 - truncated to 160.
8 = Space between grid limits not enough to accept largest item length.
12 = iarea less than 0
16 = iarea greater than 31
20 = SB equal 0
24 = buffer area too small for number of characters in menu string.

Note: Error codes 8 to 24 result in no set-up and exit to caller.
DETAILED DESCRIPTION:

A menu item is a string of characters terminated by a slash. A menu is composed of one or more such strings tied together in a single alphanumerical message. The characters up to and including the first slash represent menu item #1. The characters between the first slash and the second slash (including the second slash) represent menu item #2, etc. UPMENU fits these items within the confines of the screen area associated with the first parameter, iarea. When an enabled detect occurs on a character in this string (including the slash), the associated menu item number will be returned to the user.

UPMENU will start the string of characters to be displayed at X(LL)+14 and Y(UR)-20 as specified in parameters when CALL STIGT was issued to define screen area. The routine will scissor the character string such that no item name is continued to the next line (i.e., the last character on any line will be a slash). Further, the maximum number of characters possible, per line, will be displayed as per the following equation:

Let \( N = \text{maximum number of characters/line} \)

then: \[ N = \frac{X(\text{UR}) - X(LL)}{56} + 1 \]

The maximum number of lines within a users area is determined by the following equation:

Let \( L = \text{maximum number of lines/menu area} \)

then: \[ L = \frac{Y(\text{UR}) - Y(LL)}{160} \]

This routine uses basic character size only and double spaces between lines.

This routine extracts the data from the IGTBL specified by the "iarea" parameter and inserts the entry point to "MENHAN" with the mask bits enabled. When the light pen detect occurs and processing is completed the extracted data is restored to its original status prior to calling UPMENU.
COMMENTS: (continued)

2. when X(LL)≠0
   Buffer Length = nchars + n + cnt + 14
   where:
     nchars = same as above
     n = number of bytes to position beam for next line
       (9 if nchars/line is odd; 10 if nchars/line is even)
     cnt = same as above.

Note: The buffer length is obtained by specifying in the
CALL to STIGT the SB and EB parameters (i.e.,
Buffer Length = EB-SB)

If the WAIT option is selected the user must enable the
Light Pen Detect entries in both the ATTBL and the asso-
ciated area in the IGTBL before CALLing UPMENU.
However, if any other attentions are enabled, a return can be
made to UPMENU's internal GWAIT before a detect has
been made on the menu. In this case the user must take
the same option as if he had selected the no-WAIT option.
It is the user's responsibility to test for this condition.
The itemno field could be used for this purpose.

If in the above condition or in a requested no-WAIT option,
the user gives another CALL to UPMENU before a detect
on the previous menu the results could be invalid.
1. A menu specifying the same area.
   The old menu would be replaced by the new menu
   and detects would be processed properly. However,
   the old menu work area would be still in CPU memory.
2. A menu specifying another buffer area.
   The old menu would remain on the display since
   it is in another physical part of the buffer. If the
   same screen area is used, the two menus will over-
   lap. The old menu can be effectively erased from
   the buffer by a CALL to STAREA. However, the old
   menu work area would still be in CPU memory. If
   a detect is made on the new menu it will be processed
   properly. If a detect is made on the old menu, a
   value will be returned to the itemno of the routine
   which established the new menu. This value could
   be any number up to the number of items in the new
   menu.
DETAILED DESCRIPTION: (continued)

If the user calls UPMENU with itemno=-2, a CALL GWAIT is issued automatically and when the detect occurs, the detected item number is stored into the parameter "itemno" and control is returned to the user at the statement following the CALL UPMENU. When this routine is called with itemno≠-2, no CALL GWAIT is issued and control is returned as with a -2 value. However, upon receipt and analysis of the light detect on the menu area, with itemno ≠-2, control is returned to the user following the CALL GWAIT which is used to allow attentions to be processed. It is recommended, in the no-wait mode, that the user set "itemno" to a minus value (-2) initially and reset after use such that an indicator is available to know if the detect has occurred.

UPMENU does not update the BLP (Buffer Load Point). Therefore, any subsequent calls to image generation routines for that same area will overwrite the UPMENU orders and data in the buffer.

UPMENU has a built-in GDOA and the user should not use STGDOA and DISPLAY for displaying the menu.

COMMENTS:

The user must allocate space to place the characters of the menu string in the 2250 buffer. The buffer space may be obtained by the following algorithm:

1. when X(LL)=0
   Buffer Length = nchars + n + cnt + 14
   where:
   nchars = total number of characters in string.
   n = number of new line characters required
   (2 per line)
   cnt = 1, if nchars is odd;
        = 2, if nchars is even.
NAME: GWAIT - Wait for Attention

FUNCTION: GWAIT puts the program (task) into an effective wait state until an enabled attention occurs. When the attention occurs, control is passed to the user's attention processing routine.

CALLING FORMAT:

Fortran: CALL GWAIT
Assembly: CALL GWAIT

DETAILED DESCRIPTION:

When a program can do no further processing until an attention is received from the console operator, it must issue a CALL to GWAIT. If an enabled attention occurs or was pending, control is passed to the user's attention handling routine. To do this GWAIT examines the associated entry in the ATTBL. If the status code corresponding to the attention is OFF, the attention is rejected and the program continues in the effective WAIT state. If the status code is ON, a call is executed to the corresponding attention routine. When the attention processing routine issues a return to GWAIT, control is then returned immediately following the CALL to GWAIT.

If more than one attention was enabled at the time a CALL is made to GWAIT, the return to the point immediately following the GWAIT CALL signifies that one of these attentions has occurred. No information is available at that point indicating which attention actually occurred. If this information is required, the user must set some indicator in his attention processing routine which can be tested by the GWAIT calling routine.

EXTERNAL ROUTINES OR TABLES USED:

ATTBL and the Express macros: ANALYZ, GCNTRL, and GREADR.
COMMENTS:

For those users of GWAIT that have chosen to do their own attention handling by supplying their own attention routines via STATTN, or table addresses via GIATBL, the following information is pertinent.

GWAIT will issue CALLs to the individual attention handling routines with parameters pointing to the associated information as follows:

1. Asynchronous Error
   This attention will be handled by GPAK regardless of the entry in the user's or GPAK's ATTBL.

2. Light Pen Detect
   Call light pen detect routine (ARRAY) where ARRAY(l) is the buffer location counter contents at time of interrupt and ARRAY(2) is integer value of detected IG area (0-31).

3. End of Order Sequence
   Call end of order sequence routine (bufadr) where bufadr is the buffer location of the EOS order.

4. End Key
   CALL end key routine
   no parameters

5. Function Keys
   CALL function key routine (fkindx, ovrlay)
   where fkindx is a full word integer (0-31) corresponding to the function key causing the interrupt, ovrlay is a full word integer (0-255) corresponding to the keyboard overlay code.

6. Cancel Key
   CALL cancel key routine
   no parameters
NAME: TATTN - Test for Attention

FUNCTION: The routine TATTN checks for the presence of any pending attention. If an enabled attention is pending, the user's attention routine is called as in GWAIT.

If there are no pending attentions, or if the pending attentions were disabled, then control returns to the calling program.

CALLING FORMAT:

Fortran: CALL TATTN
Assembly: CALL TATTN

EXTERNAL ROUTINES:

The Express support macro, ANALZE, is used by this routine.

COMMENTS:

All attentions will be processed before control returns to the caller. The following procedure can be used to insure that all attentions have been cleared:
1. Disable all attentions.
2. CALL TATTN.
On the return all attentions will have been cleared. Desired attentions can then be enabled.
NAME: GETCHR - Get a character.

FUNCTION: To provide the FORTRAN programmer the ability to extract a character from a string and place the character in a word.

CALLING FORMAT:

Fortran:   CALL GETCHR (char, n, istr)
Assembly: CALL GETCHR, (char, n, istr)

DESCRIPTION OF PARAMETERS:

char    = location of word into which character is placed.
n       = index \geq 0 which describes the position of the character in istr to be placed in char.
istr    = character string

ERROR CONDITIONS:

If n < 0, char is set to zero.

DETAILED DESCRIPTION:

The nth character (the first character, n=0) of istr is placed in char right-justified with the three leading bytes in char equal to zero.

EXAMPLE:

CALL GETCHR(word, 8, istr)

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>A</td>
</tr>
</tbody>
</table>

\[ \text{word} \]

\[ \text{word} \]
EXTERNAL ROUTINES OR TABLES USED:

None

COMMENTS:

This routine permits the Fortran program to test for specific characters in an input stream, e.g. from the 2250 alphanumeric keyboard.
NAME: FIXBCD - Convert Integer to EBCDIC

FUNCTION: To convert integer values to a string of BCD characters.

CALLING FORMAT:

Fortran: CALL FIXBCD (ifix, out)
Assembly: CALL FIXBCD, (ifix, out)

DESCRIPTION OF PARAMETERS:

ifix = The integer to be converted.
out = Location into which the BCD equivalent is put. Out must be four words in length.

ERROR CONDITIONS:

Specification error in FIXBCD if out does not start on a full word boundary.

DETAILED DESCRIPTION:

The integer ifix is converted to BCD characters and put into the array out. The characters are right justified and the remainder of out is filled with blanks and the minus sign if necessary. The FORTRAN format I16 is used.

+ + +
4 8 16

16 characters

b b b b b b - x x x x x x x x x

o o o o o
u u u u u

minus sign precedes BCD if ifix is negative, otherwise the BCD is preceded by blanks.
NAME: FLPBCD - Convert Floating Point to EBCDIC

FUNCTION: To convert single precision floating point values to EBCDIC characters.

CALLING FORMAT:

Fortran: CALL FLPBCD (flno, iw, id, out)
Assembly: CALL FLPBCD, (flno, iw, id, out)

DESCRIPTION OF PARAMETERS:

flno = Single precision floating-point number to be converted.

iw, id = FORMAT specification (described below)

out = Array into which the characters are to be placed.

ERROR CONDITIONS:

1. If (iw-6) < id or id ≤ 0, then the first word of out (out(1)) is set to 0.

DETAILED DESCRIPTION:

The floating-point no. in flno is converted to the E notation in BCD characters in the FORTRAN FORMAT EIW.ID as shown below. The array out must be large enough to contain IW characters.

IW characters

<table>
<thead>
<tr>
<th>ID characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>S.X-</td>
</tr>
</tbody>
</table>

bbbbb b = blanks
S = blank or minus
XXXX = fraction
YY = decimal exponent
EXTERNAL ROUTINES OR TABLES USED:

SYS1. FORTLIB
ZERO
FORMIT
FIXBCD } INCLUDED WITH FLPBCD

COMMENTS:

FLPBCD uses routines from the FORTRAN Library. Therefore, the user should read Chapter D.7 "Using GPAK Plotting Routines" regarding assembly level use of FORTRAN library routines.
NAME: BCDFLP - Convert EBCDIC to Floating Point.

FUNCTION: Converts fixed or floating point character representations to floating point values.

CALLING FORMAT:

Fortran: CALL BCDFLP (flno, istr, n, iterm)
Assembly: CALL BCDFLP, (flno, istr, n, iterm)

DESCRIPTION OF PARAMETERS:

flno = word into which the value is placed.
istr = an input character string.
n = the total no. of characters that BCDFLP should scan in looking for the BCD characters.
iterm = the decimal equivalent of a character which will terminate the scan before n characters have been scanned.

ERROR CONDITIONS:

If a character is detected during the scan which is not b+-0123456789.E or the value in iterm, then iterm is set to zero and the number is computed from the characters encountered to that point.

DETAILED DESCRIPTION:

The BCD string in istr is assumed to represent a fixed or floating-point value. It is scanned and converted to a floating-point number in flno. The scan continues until either n characters are scanned or until the character iterm is found. An example follows:

Assume istr contains the following characters: 1234,56
and the following FORTRAN program is executed:
DETAILED DESCRIPTION: (continued)

ITERM = 107  (decimal equivalent of a comma)
CALL BCDFLP (FLNO, ISTR, 7, ITERM)
IF (ITERM) 100, 100, 200

100  (code for non-valid character in ISTR character)
STOP

200  (FLNO now equals the floating number = 1234.0)
I = FLNO  (user wanted FIXED No. = 1234)

The input format can be any of the following forms:

SXX  Integer
SXX.XX  Real without exponent
SXXESYY  Integer with exponent
SX. XESY  Real with exponent

S = sign of fraction, integer or exponent
XX = fraction or integer
Y or YY = one or two digit exponent
\(|YY| < 10^{75}\)

EXTERNAL ROUTINES OR TABLES USED:

GETCHR
SYSI. FORTLIB

COMMENTS:

BCDFLP uses routines from the FORTRAN library. Therefore the user should read Chapter D.7. "Using GPAK Plotting Routines" regarding assembly level use of FORTRAN library routines.

The sign character may be "+", "-", or " ". If the sign character is a numeric value, "+" is returned for a plus.


NAME: MSKGEN - Generate a Mask.

FUNCTION: Develop a mask word by setting selected bits in a word to zero or one.

CALLING FORMAT:

Fortran: CALL MSKGEN(n, ib1, ib2, ..., ibn, mask, iflag)

Assembly: CALL MSKGEN, (n, ib1, ib2, ..., ibn, mask, iflag), Y.

DESCRIPTION OF PARAMETERS:

n = Number of unique bits to be set in MASK word (1-32)
ib = Fortran integers specifying the bits to be set in mask (0-31)
mask = Location used to generate MASK. Must be a full word.
iflag = A Fortran integer that specifies how the bits of the mask are to be generated.
   =0; specified bits in MASK are to be zero,
    all other bits in MASK are unaffected.
   =1; specified bits in MASK are to be one,
    all other bits in MASK are unaffected.
   =2; specified bits in MASK are to be zero,
    all other bits in MASK are set to one.
   =3; specified bits in MASK are to be one,
    all other bits in MASK are set to zero.

COMMENTS:

This routine is primarily designed for FORTRAN users of GPAK. The MASK is generated from left to right in a full word. Allowable values for ib's range from zero to 31 and must be in ascending order in the call parameter list. See programming techniques (section D.5) for other uses of MSKGEN.

ERROR CONDITION:

A specification error could occur if the parameter (mask) does not start on a full word boundary.
NAME: RCODE - Retrieve Error Code

FUNCTION: Retrieve current error code value.

CALLING FORMAT:

CALL RCODE (j)

DESCRIPTION OF PARAMETERS:

j = Integer variable for error code.

DETAILED DESCRIPTION:

This subroutine extracts the current error code value and places it in the word indicated by j.

COMMENTS:

Not all GPAK routines require use of RCODE. Those routines which do not use RCODE either have specified an error return parameter in the CALLing sequence or have no error conditions. Those routines which require use of RCODE will store the error code in a cell which is contained in RCODE. On a CALL to RCODE the value in the cell is returned to the user in j.

Each routine using RCODE will store a zero in the cell if there is no error. Therefore, if the user does not interrogate RCODE immediately on a return from the CALL, the error code may be lost.
NAME: GDOADP - GDOA dump in mnemonic format

FUNCTION: Dump on-line the contents of a specified GDOA.

CALLING FORMAT:

Assembly: CALL GDOADP, (iarea), VL
Fortran: CALL GDOADP (iarea)

Where:

iarea is a full word integer (0 ≤ iarea ≤ 31) indicating a particular IG area GDOA to be dumped. If iarea is not within the above range, the entire 2250 buffer will be dumped.

ERROR CONDITIONS:

Other than normal OS/360 error conditions, this routine provides no error messages.

DETAILED DESCRIPTION:

See GBDUMP

COMMENTS:

The following DD card must be supplied in the JCL whenever GDOADP is used:

    // BUFDUMP DD SYSOUT=A
NAME: DMPTBL - Print Contents of Table

FUNCTION: Print out in hexadecimal the contents of an array in storage.

CALLING FORMAT:

FORTRAN:
(a) EXTERNAL NAME NAME
CALL DMPTBL(NAME, iwords)

where: NAME = Symbolic name of library table.
      iwords = Number of words to print, starting at location NAME.

ASSEMBLY:
(b) CALL DMPTBL(adnam, iwords)

where: adnam = Constant containing address of an array.
      iwords = Number of words to print, starting at address stored in adnam.

DETAILED DESCRIPTION:

The indicated array contents are printed out in hexadecimal, using the Fortran subroutine, PDUMP.

COMMENTS:

The following DD card must be supplied in the job control deck.

// FTO3FOO1 DD SYSOUT = A, DCB = (, RECFM = FA, BLKSIZE = 132)
NAME: RUCONV - Raster Unit Conversion

FUNCTION: Convert raster units to user units (problem coordinates) or user units to raster units.

CALLING FORMAT:

CALL RUCONV(iarea, input, iflag, output, irc)

DESCRIPTION OF PARAMETERS:

iarea = area index (0-31)
input = two word array of X, Y values to be converted
iflag = 0 convert from user values to raster units
        = 1 convert from raster units to user values
output = two word array of converted values of X, Y
irc = Return location for error conditions.

ERROR CONDITIONS:

irc = 0 Normal return
irc = 1 iarea not within range of 0 to 31, no output.
irc = 4 input values are outside defined rectangular area, converted values are in output.
irc = 6 iflag not equal to 0 or 1, no output
irc = 9 if iflag = 0 the results of subtracting the lower limit from the upper limits in the IGTBL has resulted in a neg. number, no output
        if iflag = 1 the results of subtracting the lower limit from the upper limit in the XYTBL has resulted in overflow or a neg. number, no output
irc = 16 if iflag = 0 the results of subtracting the lower limit from the upper limit in the XYTBL has resulted in overflow or a neg. number, no output
        if iflag = 1 the results of subtracting the lower limit from the upper limit in the IGTBL has resulted in a neg. number, no output.

DETAILED DESCRIPTION:

The input values are converted from one set of units to a different set depending on the setting of the iflag, scale/no scale option, and data type option. If no scale option and data type is integer the input is transferred to output and IGTBL limits tested when iflag = 0. If no scale option and data type is real the input is converted to integer before transferring to output. IGTBL limits are then tested when iflag = 0. If scale option is selected the input is transformed from one set of units to a second set depending on the value of iflag. If iflag = 0 and data type is integer the transformation equations are used to transform integer user values
into raster units. If iflag = 1 the transformation equations are used to transform raster units to integer user values. If data type is real, single precision floating point transformation equations are used to go from user to raster or raster to user, depending on iflag.

EXTERNAL ROUTINES OR TABLES USED:

FLOTE subroutine, IGTBL table, XYTBL table.

COMMENTS:

Raster units are always signed fix point integers. Negative values are permitted for raster points outside of the screen. No limit test is performed on output when going from raster units to user values if the no scale option is selected.

Transformation equations:
user to raster
\[
X = \frac{(U - U_1)(X_2 - X_1) + X_1}{U_2 - U_1}, \quad Y = \frac{(V - V_1)(Y_2 - Y_1) + Y_1}{V_2 - V_1}
\]
raster to user
\[
U = \frac{(X - X_1)(U_2 - U_1) + U_1}{X_2 - X_1}, \quad V = \frac{(Y - Y_1)(V_2 - V_1) + V_1}{Y_2 - Y_1}
\]
NAME:  STGLIM - Set Size of Grid or Plot.

FUNCTION:  Define raster unit coordinates for lower-left and upper-right corners of the rectangular grid or plot.

CALLING FORMAT:

Fortran:  CALL STGLIM(ixl, iy1, ix2, iy2)
Assembly:  CALL STGLIM, (ixl, iy1, ix2, iy2)

DESCRIPTION OF PARAMETERS:

ixl, iy1 = coordinates for the lower-left corner of a plot or grid in raster units.
ix2, iy2 = coordinates for the upper-right corner of a plot or grid in raster units.

ERROR CONDITIONS:

RCODE  = 0; for normal return
RCODE  = 1; ix2 \leq ix1 or iy2 \leq iy1
RCODE  = 2; any value > 4095
If the above occurs, no new values are stored.

DETAILED DESCRIPTION:

The fixed point values ixl, iy1, ix2, iy2 are placed in the GXYLIM table. These values are then used by GRIDD, PLOTD and LABLD for placing a grid on the screen and drawing plots and labels. If any of the arguments are negative, the corresponding value is unchanged.

EXTERNAL ROUTINES OR TABLES USED:

ERR (word contained in RCODE)

COMMENTS:

Read Chapter D.7 on "Using the GPAK Plotting Routines".

This routine must be called prior to GRIDD, PLOTD, LABLD.
The default conditions for STGLIM are:

\[
\begin{align*}
ix1 &= 500 \\
iy1 &= 500 \\
ix2 &= 3500 \\
iy2 &= 3500 \\
\end{align*}
\]

The value established by the last CALL to STGLIM will be used by all subsequent CALLs to GRIDD, PLOTD and LABLD regardless of their area value parameters. The values established by STGLIM are independent of the values in the IGTBL for those areas. Plots, grids and labels are not scissored if they are outside of the established IGTBL lower left and upper right values for the area specified in xxxJD routines.
NAME: STULIM - Set User's Limits for grid or plot.

FUNCTION: Defines the user's values for the lower-left and upper-right corners of the grid or plot and for labeling.

CALLING FORMAT:

Fortran: CALL STULIM(ul, vl, u2, v2)
Assembly: CALL STULIM,(ul, vl, u2, v2)

DESCRIPTION OF PARAMETERS:

ul, vl = User's values corresponding to the lower left corner of a plot or grid.
u2, v2 = User's values corresponding to the upper right corner of a plot or grid.

ERROR CONDITIONS:

RCODE = 0 for normal return.
RCODE = 1; u2 ≤ ul or v2 ≤ vl. no new values are stored.

DETAILED DESCRIPTION:

The floating-point values ul, vl, u2, v2 are placed in the GXYLIM table. These values are then in force for use by PLOTD and LABLD for scaling the user's data for plotting and labelling a grid.

EXTERNAL ROUTINES OR TABLES USED:

GXYLIM (a table contained in STGLIM)
ERR (a word contained in RCODE)

COMMENTS:

Read Chapter D.7 on "Using GPAK Plotting Routines".

This routine must be called prior to calling GPDATA, GPLABL.
The default conditions for STULIM are:

\[
\begin{align*}
    u_1 &= 0.0 \\
    v_1 &= 0.0 \\
    u_2 &= 10.0 \\
    v_2 &= 10.0
\end{align*}
\]

The value established by the last CALL to STULIM will be used by all subsequent CALLs to PLOTD and LABLD regardless of their area value parameters.
NAME: STGRID - Set type of grid and Number of Divisions

FUNCTION: To set the type of grid and the number of divisions or log-cycles.

CALLING FORMAT:

Fortran: CALL STGRID (itype, nx, ny)
Assembly: CALL STGRID,(itype, nx, ny)

DESCRIPTION OF PARAMETERS:

itype = -n, value left unchanged
= 0, linear in x and y
= 1, log in x, linear in y
= 2, linear in x, log in y
= 3, log in x and y

nx, ny = -n, value left unchanged
= no. of divisions or log cycles in the x and y directions respectively.

ERROR CONDITIONS:

RCODE = 0; for normal return
RCODE = 1; itype > 3. itype set to 0.
RCODE = 2; nx or ny = 0. nx, ny set to 1. itype may also be in error.

DETAILED DESCRIPTION:

The type code and number of divisions of x and y are stored in a table for later use by the GRIDD routine. Negative arguments leave the parameters unchanged from previous calls.

EXTERNAL ROUTINES OR TABLES USED:

GPARTAB (this table in GRIDD),
ERR (word contained in RCODE)

COMMENTS:

This routine must be called prior to calling GRIDD.
The default conditions for STGRID are:

\[
\begin{align*}
\text{itype} & = 0 \\
\text{nx} & = 5 \\
\text{ny} & = 5
\end{align*}
\]

If the itype specifies a log grid, the user must convert his values to log values before calling PLOTD.

The value established by the last CALL to STGRID will be used by all subsequent CALL's to GRIDD regardless of the area value in GRIDD.
NAME: STPLOT - Set type of plot and screen grid option.

FUNCTION: To set the type of plot and the off-screen or off-grid condition.

CALLING FORMAT:

Assembly: CALL STPLOT, (itype, isgopt)
Fortran: CALL STPLOT (itype, isgopt)

DESCRIPTION OF PARAMETERS:

itype = -n, parameter left as is
= 0, for point plot
= 1, for vector plot
= 2, for vector and point plot

isgopt = -n, parameter left as is
= 0, for eliminating all points and parts of vectors off the grid.
= 1, for eliminating all points and vectors off the screen.

ERROR CONDITIONS:

RCODE = 0 for normal return
RCODE = 1; itype > 2. itype set to 0
RCODE = 2; isgopt > 1. isgopt set to 0. itype may also be in error.

DETAILED DESCRIPTION:

The type code and off-screen/grid option are stored in a table for later use by the PLOTD plotting routine. A negative value for either parameter will leave that parameter unchanged.

Default values:

itype = 2
isgopt = 1

EXTERNAL ROUTINES OR TABLES USED:

PLTAB (table contained in PLOTD)
ERR (word contained in RCODE)

COMMENTS:

This routine should be called prior to calling PLOTD.
The value established by the last CALL to STPLOT will be used by all subsequent CALLs to PLOTD regardless of the area value in PLOTD.
NAME: STLABL - Set label Parameters

FUNCTION: Sets parameters affecting the labeling of grids for either axis.

CALLING FORMAT:

Fortran: CALL STLABL (n, iw, id, lgap, ltick)
Assembly: CALL STLABL, (n, iw, id, lgap, ltick)

DESCRIPTION OF PARAMETERS:

n = number of labels on an axis
iw = number of total characters in a floating point number (see detailed description)
id = number of decimal places in a floating point number (see detailed description)
lgap = distance in raster units between the axis and the center line of the number label (for the x axis) or the center of the last character in the number label (for the y axis). See detailed description.
ltick = length of tick mark in raster units.
        = 0; no tick mark is placed on the grid.

Negative values for any of the above parameters do not alter the tables for those parameters.

ERROR CONDITIONS:

RCODE = 0; normal return
RCODE = 1; n=0 or 1. n is set to 2
RCODE = 2; iw<7. iw set to 7 and id set to 1
RCODE = 2; iw alright but (id + 6) > iw. id set to iw-6

DETAILED DESCRIPTION:

STLABL sets the parameters for a later CALL to LABLD. The values established by STLABL apply to both the horizontal and the vertical axis. One call to LABLD must be made for each axis to be labeled. If different values are required for each axis, then a CALL to STLABL must be given between CALL's to LABLD to alter the values.
Labels are placed on the axis in FORTRAN floating point exponential form as follows:

\[
\text{bb--bs. xx-----xEyy}
\]

where

- \( \text{bb--b} \) = padding with blanks if required
- \( s \) = a blank or minus sign
- \( . \) = decimal point
- \( xx----x \) = decimal fraction as specified by id parameter
- \( E \) = \( E \); notation for exponential
- \( yy \) = decimal exponent

The format is specified to STLABL by the id and iw parameters. The id parameter specifies the number of decimal characters in the fraction (\( xx----x \)). The iw parameter specifies the total number of characters in the label. Padding is added to the left of the label if \((\text{id} + 6) < \text{iw}\). For the x-axis, labels are centered at the lower left x-value and at the lower right x-value. N-2 labels are evenly spaced between. Since all labels for the x-axis are printed on one line, care must be taken to properly space the labels so they won't overlap. For the y-axis labels are centered on the lower y-value and on the upper y-value. N-2 labels are evenly spaced between. See figure I.

![Figure I](image-url)

The distance of the labels from the axis and the length of the tick marks associated with each label is determined by the lgap and ltick parameters respectively. The relationship to the axis is shown in figure II.
For x-axis

\[ \downarrow \text{x-axis} \]

\[ \uparrow \text{ltick} \]

\[ \ldots \text{xxxE} - \text{yy} \]

\[ \uparrow \text{lgap} \]

For y-axis

\[ \downarrow \text{y-axis} \]

\[ \text{ltick} \rightarrow \]

\[ \ldots \text{xxxEy} \]

\[ \text{lgap} \rightarrow \]

Figure 2.

Note that tick marks are drawn independently of the grid lines created by a CALL to GRIDD.

COMMENTS:

Before the user makes the first CALL to STLABL, the GPAK plotting tables contain preassembled values as follows:
\begin{align*}
n &= 2 \\
iw &= 7 \\
id &= 1 \\
lgap &= 0 \\
ltick &= 0
\end{align*}

The values established by the last CALL to STLABL will be used by all subsequent CALL's to LABLD regardless of their area value parameters.
NAME: GRIDD - Generate orders for a grid.

FUNCTION: To create the orders for a rectangular grid according to the specifications set by the most recent calls to STGLIM and STGRID.

CALLING FORMAT:

Fortran: CALL GRIDD (iarea, ibfloc)
Assembly: CALL GRIDD, (iarea, ibfloc)

DESCRIPTION OF PARAMETERS:

iarea = area index corresponding to the logical buffer area into which the orders for the grid are to be placed. iarea must be from 0 to 31. A previous call to STIGT for this area must have been made.

ibfloc = returned address of the new BLP (Buffer Load Point) after the grid has been generated. This value can then be used to compare for light pen detects.

ERROR CONDITIONS:

Error return code (RCODE) is set as follows:

1 = iarea not 0 through 31, immediate return.
3 = User's GDOA is less than 132 words and will not hold orders generated by one call to GRIDD. Orders not stored.
8 = User's Buffer Area would be filled by this CALL to GRIDD. Orders not completely stored.
20 = SB for iarea = 0. Orders not stored.

DETAILED DESCRIPTION:

This routine, when called, computes the user's increments $U_i$ and $V_i$ from the values of $U_1$, $V_1$, $U_2$, $V_2$ and $N_x$, $N_y$. These values are then placed in a table for use by the GCGGRID routine. The GCGGRID routine is then called. The generated orders are then placed into the appropriate user's GDOA.
EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, CKIG routine, GKOACB GPAK OACB, PORGDOA table, GCGRID Cartesian Grid POR (See Express SRL), GXYLIM in STGLIM x-y limit table, ERR in RCODE routine, NX/NY table in STGRID.

COMMENTS:

1. STGLIM and STGRID should be called prior to calling GRIDD.
2. The GDOA in the associated IGTBL should be at least 132 words to avoid the possibility of an error return code of 3.
3. Read Chapter D.7 on "Using the GPAK Plotting Routines".
NAME: PLOTD - Generate orders for plotting.

FUNCTION: To create the orders to plot the user's data according to the specifications set by the latest calls to STPLOT, STGLIM, and STULIM

CALLING FORMAT:

Fortran: CALL PLOTD (iarea, utab, vtab, n, ibfloc)
Assembly: CALL PLOTD (iarea, utab, vtab, n, ibfloc)

DESCRIPTION OF PARAMETERS:

iarea = Area index corresponding to the logical buffer area into which the orders for the plot are to be placed. iarea must be from 0 to 31. A previous call to STIGT for this area must have been made.

utab = ARRAY which contains the user floating-point values for the X-axis.

vtab = ARRAY which contains the user floating-point values for the Y-axis.

n = no. of U-V pairs to be plotted.

ibfloc = returned address of the new BLP (Buffer Load Point) after the plot has been generated. This value can then be used to compare for light-pen detects.

ERROR CONDITIONS:

RCODE is set as follows:

1 = iarea not 0 to 31, immediate return
2 = n \leq 0
3 = User's GDOA is less than 132 words and will not hold orders generated by one CALL to PLOTD. Orders not stored.
8 = User's buffer area would be filled by this CALL to PLOTD. Orders not completely stored.
20 = SB = 0 in IGTBL. Orders not stored.
EXTERNAL ROUTINES OR TABLES USED:

PORGDOA table
N
IW
ID
LTICK
LGAP
IGTBL table
GKOACB table
GCPRTNT Alphanumeric print POR-(See Express SRL)
CKIG Routine
ERR Return Code in RCODE
FLPBCD Floating-point to BCD subroutine.
SYS1.FORTLIB Fortran Library
GXYLIM Table in STGLIM

COMMENTS:

1. The values supplied to the GPKA plotting routines via the CALL to STULIM are used to create the labels. Labels are created such that the lower limit label is equal to $U_1$ or $V_1$ and the upper limit label is equal to $U_2$ or $V_2$. The intermediate labels are computed and generated on linear basis. For this reason it is not possible to properly label a logarithmic axis.

2. Labels are not scissored. Therefore, the user must provide enough space for the labels as follows:
   a. For vertical labels:
      \[ X_1 \geq \text{lgap} + (iw-1)\times56 \]
   b. For horizontal labels:
      \[ Y_1 \geq \text{lgap} \]
      \[ X_1 \geq (iw-1)\times28 \]
      \[ X_2 \leq 4095 - (iw-1)\times28 \]

3. Read Chapter D.7 on "Using the GPKA Plotting Routines".
NAME: LABLD - Generate orders for labels.

FUNCTION: Creates the orders to display labels for the X or Y axis of a grid using basic character size.

CALLING FORMAT:

Fortran: CALL LABLD (iarea, iaxis, ibfloc)
Assembly: CALL LABLD, (iarea, iaxis, ibfloc)

DESCRIPTION OF PARAMETERS:

iarea = Area index with value 0 through 31 which indicates the user's buffer area into which the orders are to be placed.

iaxis = 0 for X-axis to be labeled
= 1 for Y-axis to be labeled

ibfloc = Returned buffer address of the new buffer load point (BLP) after the orders are created. This value can be ignored, or saved for use in comparing against light-pen detects.

ERROR CONDITIONS:

Return Codes in Routine RCODE are set as follows:

1 = iarea not 0 through 31, no orders stored.
2 = Axis code not 0 or 1, no order stored.
3 = User's GDOA is less than 132 words and will not hold orders generated by one CALL to LABLD. Orders not stored.
8 = Overflow EB, orders not completely stored.
20 = SB=0 in IGTBL, orders not stored.

DETAILED DESCRIPTION:

If LABLD is to be used in an assembly program, the user must review Section D.7 "Using GPAK Plotting Routines". This routine creates the orders to display numeric labels on either the X or Y axis. One axis is labeled per call to LABLD. STLABL should be called between calls to LABLD to change the specifications for the other axis labels. Calls should be made to the routines STGLIM and STULIM before calling LABLD.
DETAILED DESCRIPTION:

This routine scales the user's data contained in UTAB and VTAB according to the values previously specified for $X_1$, $Y_1$, $X_2$, $Y_2$, and $U_1$, $V_1$, $U_2$, $V_2$, by calls to STGLIM and STULIM. This data is then used to create the necessary orders in the user's GDOA.

EXTERNAL ROUTINES OR TABLES USED:

IGTBL table, CKIG routine, GKOACB GPAK OACB, PORGDOA table, GSVPLT Plotting POR - (See EXPRESS SRL), GXYLIM table in STGLIM, ERR Return Code in RCODE.

COMMENTS:

1. STPLOT, STULIM, and STGLIM should be called prior to calling PLOTD.
2. The GDOA in the associated IGTBL should be at least 132 words to avoid the possibility of an error return code of 3.
3. To continue plotting a given curve with subsequent CALL's to PLOTD, the user must use the last value plotted in the previous CALL as the first value to be plotted in this CALL. In addition the user must not make any changes to the Plotting tables or to the associated IGTBL area.
4. To create a logarithmic plot the user must translate his data to logarithmic values before calling PLOTD and supply STULIM with the users limits as logarithmic values.
5. Read Chapter D.7 on "Using the GPAK Plotting Routines".