Department of Applied Analysis
and Computer Science
Technical Report CSTR 1009
March 1971

THE PLOTTER PREVIEW SYSTEM

by
Doron J. Cohen
R. Beach & L.B. Douglas

TABLE OF CONTENTS

PLOTTER PREVIEW SYSTEM DESCRIPTION

The PLOTTER PREVIEW System provides faster turnaround for plotter users and also provides a flexible means for a user to view his plotted output without waiting for a hard copy. The heart of this system is the 2250 Graphical Display Unit. It is the user-computer interfact which accepts the users commands and also is the display output device. This system will be more beneficial to those who can get most information from just viewing their plot, rather than the permanment copy. If the user of this system requires a hard copy to be supplied, the plot may be marked for plotting to be done by our maintenance runs.

The system uses a disk based file to store the plotted information. When a user submits his job, he creates an entry for his job as well as for each plot within that job. The job name and job ID number printed on his computer output will be the identification required to select the plot from the file. Having selected the plot the user may view or mark for plotting in any order. The user can specify a "window" through which he will view his plot. This window can be manipulated for close-up or composite viewing. The user is guided by the display panels as to the information required and a little experimentation provides the necessary experience.

The following sections describe how to submit a job to the PLOTTER PREVIEW System and how to run the program. The operating instructions for the 2250 Display unit and error messages displayed are also given.

ADDING PLOTS TO THE FILE

In order to use the Plotter Preview System, the user must include 2 additional DD cards in his JCL (WATFIV users most also include a CALL to a special subroutine).

Once the job has run, the plots will have been stored in a special disk file, and the user may sit at the 2250 and view his plots at any scale. The additional requirements are detailed below.

NOTE: CLASS=P is not required since no plotter tape will be used.

JCL FOR WATFIV

```
//name       JOB   'acct. no., options'
//           EXEC  WATFIV
//GO.USERLIB       DD    DSN=GRAPHICS.Watlib,DISP=SHR
//GO.DISK          DD    DSN=GRAPHICS.IMAGEN,DISP=SHR
//GO.SYSIN         DD    *

           WATFIV program-including CALL statements
           to any of the plotting subroutines.

/*
```

When using WATFIV you must include the following subroutine call -

CALL G 2250X

It is important that this call appear before any calls to the other plotting software. This will ensure that the correct software is accessed by your program.

JCL FOR FORTRANG

```
//name       JOB   'acctno, options'
//           EXEC  FORTGCLG
//FORT.SYSIN       DD    *

           a FORTRAN source program, including CALL statements to
           plotting software.

/*
//LKED.USERLIB     DD    DSN=GRAPHICS.LOADLIB,DISP=SHR
//GO.DISK          DD    DSN=GRAPHICS.IMAGEN,DISP=SHR
```

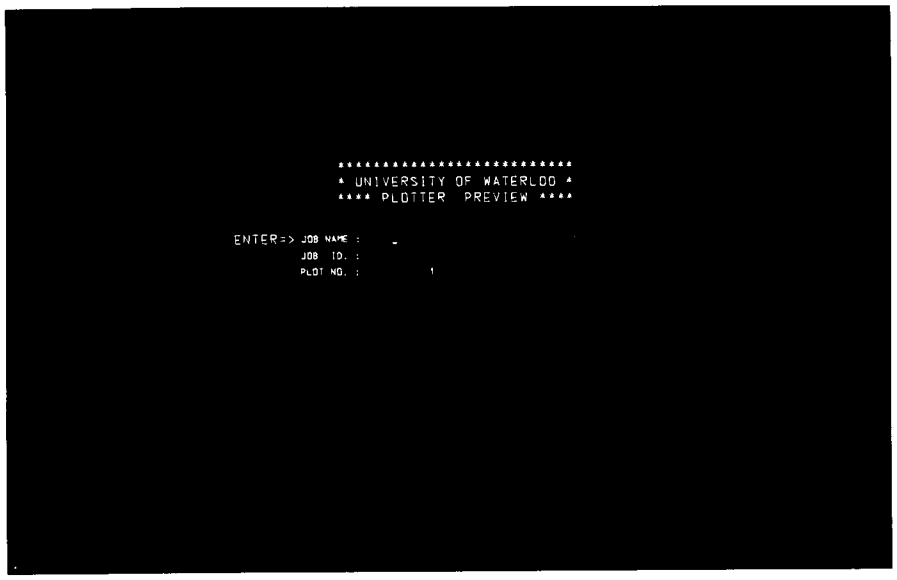
```
//GO.SYSIN      DD    *
          data for program, if any.
/*
```

To run the same program using FORTRAN H, change the EXEC card to:

```
//    EXEC FORTHCLG
```

The output from a program using the JCL described above will include a message informing the user of the JOBNAME and the JOBID number. These will be printed after the completion of the first plot in the job. The JOBNAME and JOBID will be required to fetch the plots from the disk file for viewing on the 2250 screen.

Photo: Initial Display on Screen



VIEWING PLOTS ON THE 2250 SCREEN

After a user has run his job and picked up his output the plots may be viewed on the 2250 screen. At the time of publication, a user may view his plots between the hours of 12 noon and 2 pm. An operator in the main I/O room (M&C 1078) will admit a user to the 2250 room.

The Plotter Preview program should be running when the user gets there; if not the operator will start it. While the program is running, the user may view his own plots-marking each to be plotted or scratched. 2250 screen acts as a "window" on the plots. By moving this window and changing the X and Y scales, a user may view his plots several times in different sizes or scales in order to get a complete idea of necessary changes.

NOTE: In some cases, an entire plot may not fit on the screen. This is due to the hardware limitation of the 8K buffer on the 2250. This is not a serious disadvantage to the user, since the remaining portion may be viewed by depressing a function key (to be described later).

OPERATION OF THE 2250

ALPHANUMERIC KEYBOARD

The typewriter type keyboard at the bottom of the screen may be used to enter numeric or character data to the program. Special functions of some of its keys are listed below.

CURSOR - this is a small underscore which appears on the screen to indicate the position at which the next character will be entered when it is typed in.

SPACE BAR - the space bar advances the cursor and erases the character now at the cursor.

ADVANCE - positions the cursor forward, without erasing the character now there.

JUMP - positions the cursor at the beginning of the next output area. This key moves the cursor in a cycle, so that it jumps from the last area to the first.

CONTINUE - when this key is held down and another depressed, that character will be repeated as long as both keys are held down.

END KEY - used to signal completion of input typing. It is activated by simultaneously pressing the "ALTN CODING" and "5" keys. Up until this signal is given, the input areas may be modified at will by using the cursor - positioning keys and the keyboard.

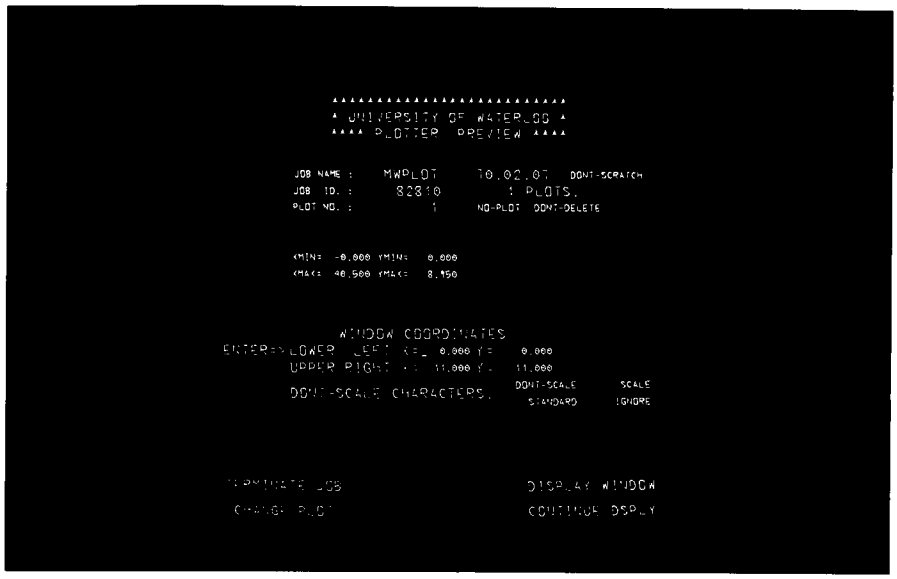
LIGHT PEN

This is a photo - sensitive device used to signal a choice of items on the screen or to indicate an action by pointing the pen at the appropriate text on the screen. The pen is aimed by holding it lightly on the screen and focusing the two aiming lights on the character string. Then the pen is triggered by pressing it against the screen. thus activating a pressure switch in the tip. The pen must be pointed at a lighted section of the screen in order to register an interrupt.

PROGRAM FUNCTION KEYBOARD

This is the 32-key box at the left of the screen. Only those keys which are lit are active. Explanation of their functions as applicable to this application are given in the next section.

Photo: Next Display on Screen



PLOTTER PREVIEW PROGRAM OPERATION

This section describes all the options available to the plotter preview user, how they are activated and what action each will initiate.

ENTER JOB NAME AND JOB ID:

The first image displayed on the screen will ask the user to enter the JOB NAME and JOBID from his output. This is done by typing from the alphanumeric keyboard.

Each user-file may contain more than 1 plot; consequently, the next information to supply the program is the plot number. If not specified, PLOT NO. 1 is assumed. If PLOT NO. 0 is typed, then a list of the plots contained in the file for this JOBNAME and JOB ID will be displayed.

SELECT THE JOB STATUS:

Any user job may have one of two states. These are SCRATCH and DON'T-SCRATCH. If the current job status is SCRATCH, the plots will be removed from the file during the next cleanup run. If the job is labelled DONT-SCRATCH, it will not be removed during the next cleanup run. Please note that no job may remain on the file for more than one week in the DONT-SCRATCH state. Jobs will automatically be scratched after this time period unless they contain requests to be plotted (but this has not yet been done).

To change the status from one to the other, simply "light pen detect" the current status and it will be changed to the other one.

SELECT PLOT NUMBER:

A particular plot from the file may be requested by typing the appropriate plot number or by detecting a plot in a list of plots, if this list has been requested as described above.

SELECT PLOT STATUS:

The plot status defines whether the plot is to be plotted, deleted or left untouched during the next maintenance run. The plot status may be altered at any time while the plot information is being displayed.

NO-PLOT -this plot will not be plotted during the next maintenance run.

PLOT -this plot will be plotted during the next maintenance run.

DELETE -remove this plot from the file.

DONT-DELETE -do not remove this plot from the file.

The current plot status may be changed to its opposite by a light pen detect. For example - if current status is PLOT, a light pen detect on this word causes a change in the status to NO-PLOT, and vice versa.

SET PLOT WINDOW SIZE:

The 2250 screen is used as a window onto your plot. The size of this window may be varied so that the entire plot or only a small portion of it will be shown on the screen. The window size may be changed at any time by typing from the alphanumeric keyboard into the designated areas on the screen.

The maximum and minimum x,y values for window co-ordinates are given on the general display. The default values for the window co-ordinates are (0.0,0.0) and (11.0,11.0). Remember that after you have altered the window size, you must press the "ALTN CODING" and "5" keys simultaneously to signal the program to accept these new values. When modifying the window size, it is important to keep the X and Y scales in the same proportion as the original. Otherwise, misleading and distorted plots will appear on the screen.

QUESTION A POINT ON THE SCREEN:

At any time during the display of a graph, the user may detect a line or point on the screen to determine its co-ordinate values. This can be particularly useful when used to determine new co-ordinates for the window size. These values are given in forms of inches on the user plot.

SET CHARACTER STYLE:

The generation of characters requires a lot of core in the 2250's 8K buffer, thus there are three optional character styles (plus no characters) to allow the user to view more of his plot rather than use the space for characters.

DONT-SCALE -draw actual characters as they would appear on the finished Calcomp plot.

STANDARD -use the 2250 standard small size hardware characters. This is restricted to small size horizontal, characters. Remember that these are not the same characters that will appear on the plot.

SCALE -use the 2250 hardware characters but use a size that is dependent on the window size. This will only apply to horizontal characters; vertical will be ignored.

IGNORE -ignore any character strings. The advantage of this is that the user will be able to get more of his plot on the screen at one time.

These different character styles are selected by causing a light pen detect on the required one in the table.

INITIATE SOME ACTION:

After a user has selected a plot and a character style and a window size, either explicitly or by default, then he may "digit pen detect" the string DISPLAY WINDOW. This interrupt will cause the selected plot to be displayed as required. On the screen, in the lower left and upper corners, the window co-ordinates are displayed. The window itself fills the centre portion of the screen. In the upper right corner are the JOBNAME, JOBID, plot number and image number.

While the image is being displayed the function keyboard becomes active. The functions of the lighted keys are explained below.

SELECT A DIFFERENT PLOT:

When the user has finished with a particular plot he may select another one that was generated in the same job by light-penning CHANGE PLOT. This will cause a return to the general display including a list of plots.

TERMINATE THE JOB:

Upon completion of his examination, a user may terminate use of the program by detecting the string TERMINATE JOB. This will terminate processing of his job and return to the initial display. He may then resume with another job or have if he is finished.

THE FUNCTION KEYBOARD:

The function keyboard contains 32 buttons, numbered 0 - 31, when a button is lit it is active. If a button is active, you may cause an interrupt to the program initiating a specific action. During the display of a plot, 11 keys are lit; their functions are described below.

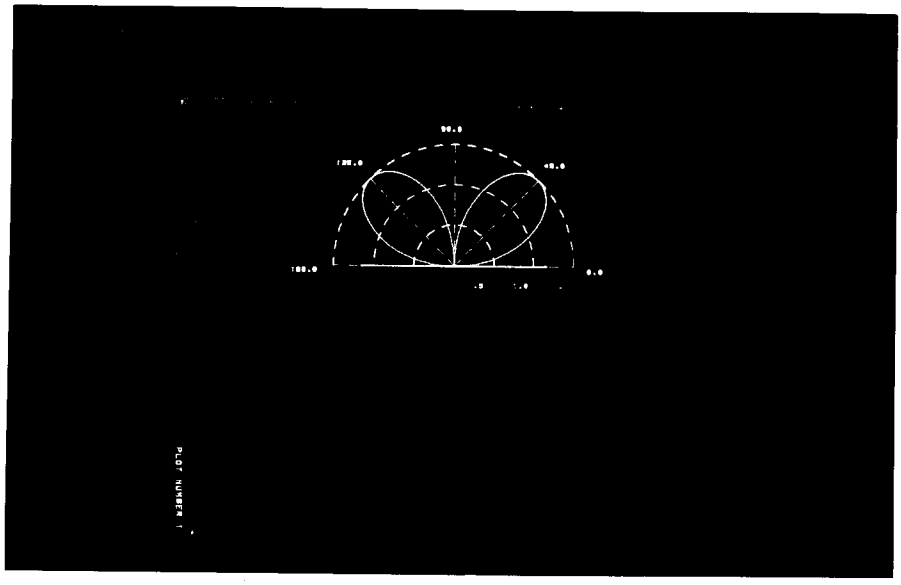
PFK 31 -return to the general display. Here the user may change plot status, get a new plot, modify the window co-ordinates, or anything else mentioned above.

PFK 29 -advance to the next window adjacent to the right of the current display.

The following keys indicate possible motions of the window. They are in the form of a cross, the outer keys implying a 90% shift and the inner keys implying a 45% shift. Remember that the direction of the shift is the direction of the window movement. Thus if you want to move the image down, you should move the window up. The light in the center of the cross will only be lit if there is more information to be generated for this display, that is, if the buffer was not able to hold all the information requested for this display.

PFK 3	-	shift window up 90% of window size.
PFK 7	-	shift window up 45% of window size.
PFK 11	-	shift window left 90% of window size.
PFK 12	-	shift window left 45% of window size.
PFK 13	-	display remaining information for this display.
PFK 14	-	shift window right 45% of window size.
PFK 15	-	shift window right 90% of window size.
PFK 19	-	shift window down 45% of window size.
PFK 25	-	shift window down 90% of window size.

Photo: User's Plot Displayed on Screen



PLOTTER PREVIEW DISPLAY MESSAGES

"* CONVERSION ERROR"

The character field following the =& sign was in error. Correct the field and resubmit.

"* INVALID JOB NAME OR JOB ID. TRY AGAIN."

The JOB NAME or JOB ID fields were not found in the file directory. Therefore check the spelling, and retype, or perhaps the job has been deleted already.

"* INVALID PLOT NO."

The PLOT NO was either non-positive or out of range. If the job name and job id fields were valid, then there should be a list of all the plots in this job. Select one of those.

"* WINDOW TOO SMALL."

The minimum window size has been exceeded. Please choose a larger size, and retype. The present minimum is set at .01 inches.

PLOTTER PREVIEW DISK FILE

Blocksize is 7200 bytes and each block consists of eight 900 byte records. Blocking and deblocking are done by READR. The records are addressed by the relative record number within the file.

DIRECTORY

The first complete block which is 7200 bytes in length is the directory for the file. It contains the number of job entries in the directory and also contains pointers to the job headers for every job entered in the file. All jobs are identified by a jobname and ID. The directory also contains a pointer to the last job header in the file; this is used to find the next available record when inserting a new job into the file.

The directory is considered as record 0, therefore the first job header is record 8.

JOB HEADER

The job header contains the jobname and ID plus pointers to each plot within the job. It also contains the number of records in the job and the number in each plot. The date on which the job was created is also included.

Each plot entry contains the maximum and minimum X and Y values for the plot plus the address of the first record and the plot status.

DATA RECORD

The data record contains the maximum and minimum X and Y values for the record plus a few other words which are not in use at this time. It also contains 216 fullword data entries.

DATA ENTRIES

Each data entry consists of a number of full words. There is a code word at the beginning of each group of entries which specifies the type of entries which can be one of the following: a chain of lines, a character string, or a string of special characters. The format is also indicated here.

CODE WORD

Byte 1 contains X'2A'

Byte 2 contains one of the following:

X'00' indicating dummy entries not used

X'10' indicating a chain of lines

X'20' indicating a character string

X'3x' indicating a string of special characters

If x = 0 then the pen is up moving to the character, if x = 1 then the pen is down.

The contents of bytes 3 and 4 are dependent on the contents of byte 2.

If byte 2 contains X'00'

byte 3 is not used and

byte 4 contains the number of words to skip

LINE MODE

If byte 2 contains X'10'

byte 3 contains the number of co-ordinate pairs and
byte 4 contains the number of words used.

Immediately following the code word are pairs of X and Y co-ordinates, each co-ordinate a fullword floating point. Drawing is done pen up to the first co-ordinate pair and pen down to all other pairs.

CHARACTER STRINGS

If byte 2 contains X'20'

byte 3 contains the number of characters in the string

byte 4 contains the number of words used.

FORMAT

The words following the code word are as follows:

X co-ordinate;Y co-ordinate;angle and height in one word; special word used by IMAGINE; character string for as many words as necessary.

The angle and height are combined in one word as follows:

angle * 10 in halfword binary in first halfword

height * 100 in halfword binary in second halfword.

SPECIAL CHARACTERS

If byte 2 contains either X'30' or X'31'

byte 3 contains the number of co-ordinate pairs and

byte 4 contains the number of words used.

FORMAT

The words following the code word are as follows: Angle and height in one word; number of the character, in binary;co-ordinate pairs. The angle and height are the same as for the character string.

Appendix 1A

FIELD	DISPL. DEC(HEX)	LENGTH (BYTES)	DATA TYPE*	CONST.**
DIRECTORY				
Record Identification	0(0)	1	C	C'F'
Not used	1(1)	3	-	
No. of Jobs	4(4)	4	B	
449 Job entries as below	8(8)			
Address of last Job Header	7192(1C18)	2	B	
No. of Tracks in File	7194(1C1A)	2	B	
Not used	7196(1C1C)	4	-	
JOB ENTRIES IN DIRECTORY				
Job Name	0(0)	8	C	
Job Identification	8(8)	3	P	
Job Status	11(B)	1	BITS	
Address of Job Header	12(C)	2	B	
Not used	14(E)	2	-	
DATA RECORD				
Identification	0(0)	1	C	C'D'
Flags	1(1)	1	BITS	
Link to next DATA record	2(2)	2	B	
Max. X for Record	4(4)	4	F	
Min. X "	8(8)	4	F	
Avg. X for Record	12(C)	4	F	
X Std. Dev. for Record	16(10)	4	F	
Max. Y for Record	20(14)	4	F	
Min. Y "	24(18)	4	F	
Avg. Y for Record	28(1C)	4	F	
Y Std. Dev. for Record	32(20)	4	F	
216 Full word data entries	36(24)			

* Codes used for Data Type are:
- B Binary number
- F Floating point full word
- C Character data
- P Packed decimal
- BITS Individual bits are used as flags

** If occupied indicates field always contains indicated value.

Appendix 1A

FIELD	DISPL. DEC(HEX)	LENGTH (BYTES)	DATA TYPE*	CONST.**
JOB HEADER				
Identification	0(0)	1	C	C'J'
Flags	1(1)	1	BITS	
Link to continued JHDR	2(2)	2	B	
No. of Plots	4(4)	2	B	
No. of Records in Job	6(6)	2	B	
16 Plot entries as below	8(8)			
Job Name	840(348)	8	C	
Job Identification	848(350)	3	P	
Date Job Created	851(353)	6	C	
Not used	857(359)	44	-	
PLOT ENTRIES IN JOB HEADER				
No. of Records in Plot	0(0)	2	B	
Address of first Record	2(2)	2	B	
Plot Status	4(4)	1	BITS	
Not used	5(5)	3	-	
Max. X value in Plot	8(8)	4	F	
Min. "	12(C)	4	F	
Avg. X in Plot	16(10)	4	F	
Stand. Dev. X in Plot	20(14)	4	F	
Max. Y in Plot	24(18)	4	F	
Min. Y "	28(1C)	4	F	
Avg. Y in Plot	32(20)	4	F	
Y Stand. Dev. in Plot	36(24)	4	F	
Spares for Doron	40(28)	12	-	

* Codes used for Data Type are:

- B Binary number
- F Floating point full word
- C Character data
- P Packed decimal
- BITS Individual bits are used as flags

** If occupied indicates field always contains indicated value.

GENERAL DESCRIPTION

This is a series of programs which replace the CALCOMP PLOT routines. Instead of creating a plottape as do the CALCOMP routines, these programs create a disk file which enables the plots to be viewed on the 2250 DISPLAY UNIT. There are two versions of these routines, one for IBM FORTRAN and one for WATFIV. The routines are written in 360 Assembler language and with the accompanying subprogrammes take up approxiamately 6K in both versions. The source is written in the form of a MACRO definition and all that is necessary to generate the programs is to expand the macro PLOTS with an operand of either IBM or WATFIV. The main coding is the same in both versions and the main differences are in the sections for getting the arguments out of the parameter lists and for printing error messages. There are also a few differences in the ENTRY names contained in the programs, the WATFIV version of the CALCOMP routines contains a few extra entry points which we must also replace.

WATFIV VERSION

The ENTRY points in this version of the routines are as follows:

G2250X
PLOTS
PLOT
OFFSET
FACTOR
WHERE
ZIP
$ACCNT
$TIME
SYMBOL
$YMBOL
PTHIRTY

G2250X is a dummy ENTRY which is called merely to get our routines from the library. ie. The first call in the program must be a call to G2250X in order to ensure that the user gets our routines instead of the other PLOT routines which are in WATLIB. PLOTS,PLOT,OFFSET,FACTOR and WHERE have the same functions as the corresponding CALCOMP routines. $ACCNT and $TIME replace the programs by those names which are in the WATFIV version of the CALCOMP programs but they pass back only dummy

parameters, $ACCNT passes back a string of blanks and $TIME passes back a fullword binary zero. ZIP is a dummy routine which is merely a BR 14. SYMBOL and $YMBOL are alternate names for the same entry point and have the same function as in the CALCOMP programs.

IBM VERSION

The ENTRY points in this version are as follows:

G2250X
PLOTS
PLOT
OFFSET
FACTOR
WHERE
ZIP
SYMBOL
$YMBOL
PTHIRTY

All of these routines have the same function as in the CALCOMP programs with the exception of ZIP which is a BR 14.

GENERAL

The following routines are called by both the IBM and the WATFIV versions of this program. READR: an assembler program to read and write our disk file. OCCURS: an assembler program which scans through a character string searching for certain special characters. SIN,COS: the standard FORTRAN functions.

WATFIV VERSION

Since this program runs under the control of WATFIV it must let WATFIV do all its unit record I/O for it. It does not read anything from the card reader so the only thing we have to worry about is printing error messages. This is accomplished through the use of the SPRIN macro which passes the message to WATFIV to be printed. In order to allow our messages to be printed we must initialize the WATFIV I/O routine (WATIO) every time we re-enter our routines from the WATFIV mainline.

Because we open the DCB for the disk file and issue an ENQ macro while executing these routines we must ensure that WATFIV will not kill us before we have a chance to close the DCB and DEQ the file since if the task attempts to terminate normally while something is still ENQed it will abend. The procedure we have used to ensure this will be explained in the description of PLOTS. Whenever we are forced to terminate a routine in the middle of execution in such a way that would ensure that the user will not continue to execute we branch to XRETRACE in WATFIV which terminates the job and provides a traceback.

G2250X

This provides a unique entry point name that the user can call to ensure that he will get our routines instead of the other routines in WATLIB with the same names. This feature is provided mostly for use under monitor where the user would not be able to specify a user library.

PLOTS

This is the initial routine of the series and must always be called before any of the others (except G2250X). If any other routine is called before PLOTS an error message will be issued and execution terminated. The first thing in all of the other routines is a branch to a routine to issue an error message. When PLOTS is executed it zaps all these branches to noops so that the messages will not be issued. At the same time it noops a branch in itself so that if called again it will issue a message and terminate. The program then goes and gets the jobname out of the TIOT and issues a TIME macro to get the time and date. The time is used along with the jobname to identify the job and the creation date is put in the file to indicate when to scratch the job from the file.

We then ENQ on the name PLOTS.IMAGEN to get exclusive use of the file. If two jobs attempt to use the file at the same time they will wipe each other out so we must ensure that any job creating plots on the file has exclusive use of the file. In the regular WATFIV version of the program this is an uncoditional ENQ so that the program is just put in a WAIT state until the file becomes available. In the MONITOR version we cannot do this because we cannot have MONITOR put in WAIT for two or three minutes while someone else is using the file. Therefore in the MONITOR version we issue a conditional ENQ and if the file is already in use we notify the user of this fact and terminate.

The program now plays the fudgy game of ensuring that WATFIV cannot kill us without coming back to us so that we can close the

file and DEQ. To do this we first find the PICA and zap the exit address so that if there is a program interrupt control will be passed to us instead of the WATFIV error routine. We first save the exit address in the PICA so that we can restore it later and so that we can pass control to the WATFIV routine if we are sure that it is not going to bomb us. WATFIV has a routine called MAINREST which is used to clean up after every job in the batch. This routine is always referenced through an adcon at XAMNRST so we also go in and zap this adcon so that control will first be passed back to us so that we can close the DCB and DEQ. The routine to which control is passed on a program check will check the interrupt type to see if it is one of the interrupts which will cause WATFIV to kill the batch, if it is it will close the DCB, DEQ on PLOT.IMAGEN and restore the adcons in XAMNRST and the PICA, then pass control to the WATFIV error routine. If the interrupt is one which will merely kill the job and not the whole batch then we merely pass control to the WATFIV error handler since this will then attempt to go to MAINREST and we will get control anyway. When we get control from WATFIV through the adcon in XAMNRST we restore the two adcons, close the DCB, DEQ and pass control to MAINREST.

The parameters passed to PLOTS are the address of a buffer area and the length of that buffer in bytes. Since this buffer is used as the I/O buffer for the disk file it must be at least 7200 bytes in length and if it is not the user will get an error message and execution will be terminated. The address of the buffer may be passed as either the name of the array or the first

name in the array, the CALCOMP routines accept only the first element in the array.

PLOTS

This is the name of the routine that does the plotting of lines. The lines are chained together in the file with each chain consisting of one pair of co-ordinates to which no line is drawn, followed by a variable number of points to which continuous lines are drawn. If sequential calls are made to PLOT with the pen up, all but the last of these are ignored. If a line is drawn which goes over the boundary of the plotter then the intercept is calculated and the plotter is simulated. When the user first starts plotting, the plots are put in the second availble record in the file, leaving the first record for the job header, the program continues to put data in the file until the end of the first plot. At this time the job header is set up and a job entry is put in the directory. An entry is put in the job header for each plot after that plot is completed. In this way there are no plot entries for plots that are not complete, nor are there job entries or job headers for any jobs that do not have at least one complete plot. This also ensures that if the user bombs in the middle of the second or later plot he can still get at all those plots which were completed.

Every time it is necessary to update the directory the program will ENQ on the name PLOTS.DIRECTORY. The PLOTTER PREVIEW routines forthe 2250 will also do this to ensure that 2

programs do not attempt to update the directory at the same time. Other than that the 2250 routines can be used at the same time as these routines since the 2250 programs do not write in the file except to update the directory.

SYMBOL($YMBOL)

This is the routine which does the plotting of character strings or special characters. Character strings are put in the file one string at a time with all the attendant information. Special characters are chained together as long as they are identical in all respects, this is done because special characters are generally used as symbols for plotting in which case there are usually many of them which are identical except for position. The co-ordinates of the corners of a rectangle containing the character or string are computed and used to determine the max and min X and Y values for the record and plot. If the co-ordinates passed to SYMBOL are outside the limits of the plotter then the user will get an error message and will be terminated. This rather drastic measure is necessitated by the fact that we do not know what the character looks like and cannot duplicate the action taken by the plotter. If any of the corners of the rectangle are outside the plotter limits the user will get a warning and execution will continue.

$ACCNT, $TIME

These two routines are just dummy routines inserted because they may be called. They both pass back dummy arguments, $ACCNT passes back a string of 12 blanks and $TIME returns a fullword binary zero.

PTHIRTY

This routine changes the plotter size from the default 11 inches to thirty inches.

MISC.

OFFSET and FACTOR work identically to the CALCOMP routines of the same name. If WHERE is called immediately after a call to SYMBOL it will return, not the actual position of the pen, but the co-ordinates passed to SYMBOL. ZIP is merely a BR 14. STATIC is a CSECT based on R13 which contains the data areas for the program. A WATFIV type savearea is the first 25 words in STATIC.

MONITOR VERSION

This version of the program is the same as the WATFIV version except that it does a conditional ENQ on the disk file instead of an unconditional one. This is because we cannot put the MONITOR terminal in a WAIT state for two or three minutes while we wait for the file to become free. This version of the program may be generated by expanding the macro PLOTS with the operand MONITOR.

IBM FORTRAN VERSION

Because it is running as a subroutine to a FORTRAN program and is not running under WATFIV, this version uses QSAM to print out the error messages rather than the SPRIN macro as does the WATFIV version. All the ENQ's are unconditional so the program just waits until the file is free. If the buffer passed to PLOTS is not large enough the program will not terminate as does the WATFIV version but will issue a GETMAIN macro for the necessary 7200 bytes. This is an uncoditional GETMAIN so if the core is not available the program will ABEND. When the guy does a no-no that will terminate him we issue the error message and abend. Other than that all functions are the same as the WATFIV version.

SKRUNCH

This program is designed to cleanup the plotter preview disk file by deleting all jobs or plots which have been marked to be deleted. There are status bits in the directory which indicate the status of the job and in the job header which indicate the status of the individual plots within the job. The status bits in the directory indicate whether the job is to be scratched, whether there are any plots in the job which are to be deleted and whether there are any plots within the job which are to be plotted. The status bits in the job header indicate which, if any, plots are to be deleted and/or plotted.

The squishing of the file is done in place, the program scans through the file and as it finds jobs which are to be kept it moves them immediately after the last one it has done. A job will not be scratched if it has any plots to be plotted and a plot will not be deleted if it is marked to be plotted. The program also checks the creation date of the job and if it is older than a given number of days it is marked to be scratched the next time. The default value for this retention period is seven days but it may be passed to the program at execution time as PARM=LIFE=x where x is the time in days.

The diagram on the following page illustates the display which originally appears on the 2250 screen. The heavy printing is standard, i.e. it never changes but the information in the squares is the optional information, the input to be supplied by the user.

The commands which have circled numbers associated with them are the only commands which can be changed by LPD. The numbers are the correlation values and control is transferred to the action which is indicated by the LPD.

The four keys indicated in the upper left corner are the JOB KEY, PLOT KEY, CO-ORDINATE KEY and the WINDOW KEY. The jobname, plot number and other information are entered from the keyboard.

JEKY
PKEY
CKEY
WKEY

UNIVERSITY OF WATERLOO

** PLOTTER PREVIEW **

ENTER=> JOB NAME : [PJBNM] DD.MM.YY (8) SCRATCH DON'T-SCRATCH

JOB ID. : [PJBID] [PLOTS] PLOTS. (7)

ENTER=> PLOT NO. : [PLT#] (10) PLOT NO-PLOT (9) (12) DELETE DON'T-DELETE (11)

XMIN = [XMIN(PLT#)] YMIN = [YMIN(PLT#)]

XMAX = [XMAX(PLT#)] YMAX = [YMAX(PLT#)]

WINDOW COORDINATES

ENTER=> LOWER LEFT X= [LLX] Y= [LLY]

UPPER RIGHT X= [URX] Y= [URY]

(13) [CHAR(K)] CHARACTERS. (14) SCALE DON'T-SCALE (15)

(16) IGNORE STANDART (17)

(1) TERMINATE JOB (5) DISPLAY WINDOW

(2) CHANGE PLOT (6) CONTINUE DSPLY

SUBROUTINES

IMAGINE

IMAGINE is called from PLOTAID and upon exit control is returned to the calling program with the return code as the function value.

A parameter list is not required for the input but the following values are needed:

PLTIDNT	GSP, #, ATTL, GDS, C1, C2, C4.
PLTPARM	PLT#, IMG#, PLTP, #RECORD, MODE, TBL#, CSCL, LLX, LLY, URX, URY, PJBNM, PJBID. If the mode is negative then the REC# and the ITM# are also required.
MINIMAX	MINIMAX must be allocated and supplied with valid data.

The mode as given from PLTPARM will determine whether we are starting or continuing. If the mode is positive, then we are starting and the plot number, record number and the item number at 1 are required as input. If the mode is negative, then we are continuing and the record number and the item number from the previous call to IMAGINE are required as input. If not continuing in camera mode, then a call to IMGINIT is necessary.

In sequencing through the data records, a check is made for the required minimax table within the window.

Sequence through data item: the operation codes are as follows:

X'2A'

X / flags; where X is one of the following:
  0 indicating that the beam is on/off.
  1 indicating lines.
  2 indicating a character string.
  3 indicating centred symbols.
COUNT which counts the number of data lines or characters.
LENGTH which is the length of the item in 4-byte words.

<u>LINES</u> The initial (X,Y) position of the lines is moved to with the beam off. The next and following (X,Y) are the pointers to which the lines are drawn. Count = 1 for each pair of (X,Y) points. the format appears as shown:

2A	1/X	count	length
start X co-ordinate			
start Y co-ordinate			
line to (X,Y)			
:			
:			

where: count is the number of (X,Y) co-ordinate pairs.
length is the length of the item in 4-byte words.

CHARACTER STRINGS

The format is as shown:

2A	2/X	count	length
x co-ordinate			
y co-ordinate			
A*10		H*100	
Ibs	Icr	Isup	Isub
character string : :			

where: count is the number of characters in the string.

(X,Y) are the lower left co-ordinates of the first character.

the angle is multiplied by 10 and the height is multiplied by 100 in order that the two floating point numbers can be stored as one word.

the four bytes Ibs, Icr, Isup, Isub note the first occurence of each special character described as follows:

BS	backspace character	HEX(11)
CR	carriage return	HEX(19)
SUP	superscript	HEX(2E)
SUB	subscript	HEX(2F)

CENTRED SYMBOLS

The format for centred symbols is as follows:

2A	3/X	count	length
A*10		H*100	
zeroes		character	
X			
Y			

Here the angle and height are the same as in the character string. Count is the number of pairs of (X,Y) co-ordinates given for each symbol.

The general function of IMAGINE is to generate a display of the selected plot as viewed throught he window which is specified. This program can be used in one of the following four modes:

MODE=1 to generate a display for user viewing from the plot data records.

MODE=2 to generate a display for camera mode viewing from the beginning of the plot data records.

MODE=-1 to generate a display as mode 1 but continue from where we left off at the last call to IMAGINE.

MODE=-2 to generate a display as mode 2 but continue from where we left off at the last call to IMAGINE.

The required input to IMAGINE has already been discussed. The output from IMAGINE is the return code via the function value which is one of the following:

IMAGINE=-2	Data Read Error
IMAGINE=0	normal return
IMAGINE=1	buffer full, mode=1
IMAGINE=2	buffer full, mode=2

The following values may be altered:

PLTPARM

REC#	current record number of data record
ITM#	current index to data item
GSPRC	latest non-zero return code
GSPAI	latest additional information for call referred to by GSPRC
#SSCIS	cummulative total of items scissored
BYPASS	the number of data records bypassed
CORRVAL	current correlation value

The keyboard function lights will be turned off. The following messages may be output to SYSPRINT:

'**IMAGINE** READR ERROR. CODE=XXXX'. This message indicates an error return from the function READR and causes a -2 return code from IMAGINE.

'**IMAGINE** ILLEGAL OPR. REC#=XXXX,ITM#= XXXX,OPR=bit string'. This message indicates an error in the data file caused by an invalid OPR code (i.e. other than X'2A') or an unimplemented code. This code is ignored and processing will continue.

'**IMAGINE** ILLEGAL OWD. REC#=XXXX, ITM#=XXXX, OWD=bit string'. This message indicates an error in the data file caused by a large special character index in the occurs word, i.e. the position in the string is greater than the string length.

'**IMAGINE** BUFFERFULL. OLDCORRVAL=XXXX, CORRVAL =XXXX'. This message indicates that the 2250 buffer was filled while generating display elements for the item were referred to by CORRVAL. The mode is +2 or -2 , then the buffer will be reset to the OLDCORRVAL and an EOS OWD element added. The REC# and ITM# are decoded from the OLDCORRVAL and a negative return code is issued. Otherwise(i.e. if the mode is 0 or 1) the RESET is not performed and a negative return code is issued.

Upon exit, the Graphic Data Set will contain the following:

1. plot identification text
2. window co-ordinate values
3. LP detect text in OMIT status
4. graphic display data to fill the buffer
5. if mode is ±2, then an END-ORDER SEQUENCE element will be displayed on the screen.

The operation of IMAGINE is explained as follows: if the mode is -2, then we immediately fetch the required data record and continue processing. CSW is used for this purpose and will have the following values: CSW=0; not continue mode, calculate new number. CSW=1; continue mode, use item number supplied.

Otherwise (when mode is other than -2) IMGINIT is called to produce the image identification. Then if mode=-1, we go and fetch the data record. If starting fresh, we reset the REC# and correlation values and save the current beam position.

When a new record is requested, the window co-ordinates are checked with those for the data record stored in MINIMAX. If any part of the data is within the window, the whole window is processed. If the record is bypassed then a tally is updated and the next record is checked.

When a new data item is requested, the item index is updated by the length of the previous item plus 1. Then the operation code is unpacked. If it is illegal an error message is issued and this item is ignored.

Otherwise the new correlation value is calculated and processing of the item is performed by seperate code.

The three operations are explained as follows:

OPR=1 LINE DATA

The first (X,Y) co-ordinate pair is used to POSITION the beam and all successive ones are used to draw connecting line segments. The last (X,Y) co-ordinate pair is saved as the current beam position.

OPR=2 CHARACTER STRINGS

The (X,Y) co-ordinates are shifted from the left corner of the character to the center for PLSTR. The angle and height of the character string are decoded and the orientation vector is computed.

If CSL=0, the characters are ignored; if CSL is negative, PTEXT is used to generate the characters; and if CSL is positive, it is used as a scaling factor to the height.

If there are no special characters, we call PLSTR to display the remaining character string and to get the next item.

Otherwise the first special character is identified by MINBYTE and the action is performed accordingly.

BACKSPACE reset X co-ordinate by width of character.

CARRIAGE RETURN	reset X co-ordinate to original X and move Y co-ordinate down one line.
SUPERSCRIPT	if already in superscript mode (STATE=1) then ignore this one. If in normal mode (STATE=0) then set height to .714 of normal height and set Y co-ordinate to the center of the resulting character size. If in superscript mode (STATE=-1) then reset the height and Y co-ordinate to the normal values.
SUBSCRIPT	set as for superscript but move the Y co-ordinate down instead of up.

OPR=3 CENTRED CHARACTERS

The height and angle of the character string are decoded and CSL is supplied only as a scaling factor if specified, otherwise the given height will be used. If the INCLUDE bit is on (bit 12 of the occurs word), then a line joining the previous point and the character will be drawn. Then for every (X,Y) co-ordinate pair given, the (X,Y) co-ordinates are transformed to the centre of the character for PLSTR and the connecting line is drawn if desired and then the line itself.

After every display element generation, a check is made for buffer overflow. A non-zero return code from GSP causes a call to

GSPERR to analyze the error and jump to the Bufferfull routine if there is such an error.

When a Bufferfull condition arises, the GDS will be reset back to the OLDCORRVAL and an END ORDER SEQUENCE element will be generated only when the mode is ±2. The record number and the item numbers are decoded from the old correlation value. The return code is set and display is executed and control is returned to the caller.

IMGINIT

The general function of IMGINIT is to initialize the Graphic Data Set for IMAGINE and to prepare the image identification on the screen.

The entry is IMGINIT and exit is to the calling program. The required input values to IMGINIT from the external data area PLTPARM are as follows:

IMG#	the current image number
LLX	the lower left X co-ordinate value
LLY	the lower left Y co-ordinate value
PJBID	the job ID for the plot
PJBNM	the job name for the plot
PLT#	the current plot number
URX	the upper right X co-ordinate value
URY	the upper right Y co-ordinate value

For the output, the Graphic Data Set is initialized to the co-ordinate system of the plot and contains the identification text for the plot.

The operation of IMGINIT is as follows: the Graphic Data Set data limits are set to (0,0) and (4095,4095) for the

identification text. Then the following text is created:

'jobname jobid plt# img#'	in upper left corner
'(LLX,LLY)'	in lower left corner
'(URX,URY)'	in upper right corner

The appropriate values are supplied then the GDS data limits are set to the plot window co-ordinates and the following text is placed in omit status in the lower right corner for LP detects;

'LP AT X=DDDD.DDD, Y=DDDD.DDD'

GSPERR

The function of GSPERR is to analyze a GSP return code. The normal action is to print a message, cause a PL/1 trace back and return. Scissoring errors are accumulated and a buffer overflow error causes a branch to the label passed as input.

Entry is GSPERR and exit is return on the invalid GSP return codes.

For return codes 1,2, and 4; return to the caller.

For return code 3 and not buffer overflow; return to caller.

For return code 3; normal return.

For return code 5; stop.

The input to GSPERR is the single parameter which is the LABEL of a statement to be jumped to in case of buffer overflow. GSPRC in PLTPARM must be set prior to calling GSPERR.

The output is one of the following:

a) if there is a scissoring error, #SSCIS in PLTPARM is incremented by one.
b) if there is not a buffer overflow, then a message is printed.

The operation of GSPERR is as follows: GSPRC is tested for validity and an immediate return is made if it is not valid. The additional information is obtained from GSP and is stored in GSPAI in PLTPARM. Then a jump is made to the error handler.

The error is one of the following:

GSPRC=1 scissoring; count the number of errors in #SSCIS.

GSPRC=2 scaling; print the message and signal TRACE.

GSPRC=3 storage overflow; on buffer overflow, go to the LABEL parameter, otherwise, print the message and signal TRACE.

GSPRC=4 parameter error; print message and signal TRACE.

GSPRC=5 I/O error; print message and stop.

The ON CONDITION (RETRACE) is a dummy on unit to find the calling statement near where the error occured. Note here that the statement number printed will not be the offending one but that it is the call to GSPERR where the error occurred.

GSPTRCE

The function of GSPTRCE is to snap the GSP return code for the previous GSP call.

Entry is GSPTRCE and exit is to the calling program. The input is the parameter list declared as follows:

```
DECLARE GSPTRCE ENTRY(CHAR(30)VAR, FIXED BIN(31), FIXED BIN(31));

ARG 1   character string to identify the GSP call;
ARG 2   snap level number
        if ARG 2=0, then ignore
        if ARG 2=1, then snap only if there is a non-zero
                    return code
        if ARG 2=3, then snap everytime.
ARG 3   GSP name variable for GSP calls.
```

The output from GSPTRCE is as follows: if the level and the return code are such, then the character string and the return code are printed as well as any non-zero additional information.

In the operation of GSPTRCE a check is made for a non-zero level; if it is zero then return. The return code is obtained from GSP and if it is zero and the level=1, return. Otherwise print the identification and return code value. Additional information is obtained from GSP and is printed if it is non-zero, then return is made.

GSP02

GSP02 is a stroke table for displaying special characters on the screen.

The characters can be displayed individually or in character strings. The lines forming the characters are drawn to the co-ordinates specified and are either blanked or unblanked. The last line drawn in each character is generally unblanked and is specified as the last line.

When the program is run, the stroke table identifiers appear on the screen and the user is requested to indicate the character or character string that he requires. The characters can be indicated either by numbers or by the characters themselves. For example, the letter A can be indicated by asking for character number 65 or by indicating A.

If n characters are to be requested by name, they must be preceded by -n and seperated by commas. For example, if the 3 characters A,B,C are requested, the command would be '-3,A,B,C'.

CANCEL

The function of CANCEL is to cancel the PL/1 STAE so that normal ABEND dumps can be produced without PL/1 interference.

To enter, CANCEL is called from PLOTAIDC (the debugging mainline). Exit is to the calling program.

There is no input to CANCEL and the user does not receive output but STAE is cancelled.

In the operation of CANCEL, a STAE 0 is issued to cancel the STAE issued by PL/1. Then no checking is performed and return is made immediately to the caller.

READR is a subprogram which reads and writes the disk file used in the PLOTTER PREVIEW SYSTEM. It is a function written in /360 Assembler language in PL1 compatible form (ie. the function value is returned on the end of the parameter list, rather than in R0 or F0). Other than this, all OS conventions are followed.

In addition to the arguments passed to it, READR requires the following external symbols;

BPNTR: a full word containing the address of the buffer area where READR will expect the block it is about to write and will place the block it reads. This buffer must be at least 7200 bytes.

JPNTR: a full word where READR will place the address of a Job Header record it has read.

DPNTR: same as JPNTR only for a Data record.

BUR: 3 bytes in which READR will place in packed decimal form, the number of the current block in the buffer. This is the relative block number starting from 0 at the beginning of the file.

LIM: 3 bytes in which READR will place, after the first call to READR(1,0) the total number of tracks in the file.

READR is set up for the file as outlined in Appendix 1A and any changes in the file will necessitate corresponding changes in READR.

READR is set up as a PL1 function with 2 (two) arguments. The two arguments are halfword binary and the return argument is a full word binary. Since it is set up as a PL1 function the return argument is passed back as the third element in the parameter list. The first argument is a code which indicates what type of record we expect to read. These codes may be found in Appendix 1B. The second argument is the relative number of the record we wish to read (starting at 0). The file directory is considered to be record zero and as it occupies the entire first track or 8 records we have decided that records 1-7 are invalid and cannot be read. A call to READR with a record number in the range 1-7 will result in nothing being read and control will be returned with a return code of -2. This return code is the return argument from READR. A return code of zero indicates that the read was performed correctly, a positive return code indicates that a record was read but its type did not agree with the code given by the first parameter, if the return code is negative no read operation was performed.

If the code passed as the first argument is negative and the block to be read is not the same as that currently in the buffer then the block in the buffer is written onto the disk before the next block is read. If the code is positive, only the read is performed.

The first time READR is called the file is opened and a branch around the OPEN is zapped so that the OPEN will not be executed again. If READR is called with WHAT negative (i.e. write) the first time through, control is returned with a return code of -5 and no read or write is performed.

When READR is entered the registers are stored and chained forward and backward. WHAT (the first argument) is then checked to see if it is negative, if it is a switch is set so that the next time we read a new block we will write out the current one. We then complement WHAT and check it to ensure that it is in the range 1-4, otherwise it is invalid and control is returned with a return code of -1. The record number is then divided by 8 to get REG, the block number and REC, the relative number of the record within the block. If REG = 0, indicating that we are going to read the directory, REC must also be 0 or we return with a return code of -2. If WHAT is not equal to 1 (read the directory), we now compare REG to LIM to ensure that the record we are about to read is within the limits of the file. If REG is greater than LIM control is returned with a -3 return code. REG is now compared to BUR to see if the block we are about to read matches that which is currently in the buffer, if they are equal no read or write is performed. If WHAT = 1 we do not check these because the first time we read the directory BUR and LIM are undefined. The current block is now written if required, the new block is read into the buffer and BUR is set equal to the new block.

We now branch, using WHAT, to a routine which will set any necessary pointers and check the first byte of the record which was read to see if it agrees with WHAT. For WHAT=1 this byte must be C'F', for WHAT=2 it is C'J' and for WHAT=3 C'D'. If WHAT=4 we do not care what we are reading and both JPNTR and DPNTR are set to point to the appropriate record. If this first byte does not agree control is returned with a return code equal to WHAT, otherwise the return code is 0.

READR is 768 bytes long and contains the DCB for the disk file. The DDNAME for this file is DISK.

TUBE DISPLAY ROUTINES

MONUSER

Interface with the 2250 tube is done withe the IBM provided package GSP(Graphic Subroutine Package) to function calls from the PL/1 programs.

The zero state,PLTINIT initializes the GSP with a single data set GDS. From this state, a jump is made to state=10 which is MONUSER.

In MONUSER, the GDS is cleaned and the lights are turned off. The user is asked to identify himself i.e. his JOB NAME and JOB ID by means of a display on the tube. The users' JOB NAME and ID are accepted and a search through the directory is performed for the given job. If the search fails, the user is asked again to identify himself. When the job is identified it reads the job header into core and then asks the user to identify the specific plots.

The specified plots are then taken; their size,XMIN and XMAX is flashed on the screen along with the current status of the plot,i.e. delete or plot.

The light pen can be used here for changes or the user can key in the desired window size and then light pen the proper structure. Additional actions can be signalled such as character handling methods of moving the window to the plot.

ACT is a vector of labels. When an interrupt is received, transfer goes to one of these labels. Each label specifies an action; accordingly instruction key words are displayed with correlation values greater than 10,000 and the value specified with the action should be taken so that the correlation value can be used in a GO TO statement to transfer to the required action. There is only one point where attention is being requested. A set of GSP keys are being used to reset portions of the GDS and to update the information on the screen.

The following is a list of actions being done. These actions can be reached by light pen detect, hitting the end key, pressing function keys or by programme control.

ACT(1)

ACT(1) is activated by the light pen and it will reset the panel to its' initial state of asking the user for a new JOB NAME and JOB ID.

ACT(2)

ACT(2) is initialized by a light pen request for a change of JOB# and will read the job header and display on the screen a list of all plots contained in the job and will prepare the system to accept a LPD on the plot as a choice of plot, or a

number entered from the keyboard as a plot number. This is done by setting KBACT to indicate action #22 in the event that the next attention received is from the keyboard. Each plot is being displayed with its' boundaries and status indicators.

ACT(3) and ACT(4)

ACT(3) and ACT(4) are spare light pen actions.

ACT(5)

ACT(5) is activated by a LPD on the message display window. It will calculate the size of the window and reset the image number in PLTPARM to zero. If the window specified is too small then an error message is given or else procede to ACT(6).

ACT(6)

ACT(6) is activated by a LPD on continue display message. It checks to make sure that there is a valid PLT# in PLTPARM, the image number is incremented, mode is set to 1 and return is made to the mainline with a code of 0 or 1 depending on whether or not the MINIMAX table is already set,i.e., if it is the first time that the plot is being displayed, MINIMAX must be created.

ACT(7)-ACT(12)

ACT(7) to ACT(12) are activated by LPD on the status indicators; the status byte can be changed in either the job or the plot but it is necessary to record these changes by updating the directory in disk or the job header accordingly. If the status byte 5 equals 0, go to ACT(8) where if the light is on scratch the plot or job else if the status byte 5 equals 1, go to ACT(7) where if the light is off, don't scratch the plot or job.

ACT(13)-ACT(17)

ACT(13) to ACT(17) are activated by a LPD on the character options; set CSCL in PLTPARM according to the desired action. When this is finished, action goes to DISPWIN where the display of character options on the screen are updated.

ACT(18)

ACT(18) is activated when entering window co-ordinates from the keyboard. Text is created in converted numbers of X and Y boundaries and then action is transferred to update the display and the boundaries or else to display,which is ACT(19).

ACT(19)

In ACT(19), the necessary GSP function calls are made to display the window co-ordinates on the screen and at the same time, the KBACT is set to 18.

ACT(20)

ACT(20) is activated when the JOB NAME and JOB ID are entered from the keyboard. KBACT is initialized to 20 every time we start to consider a new job. Text would be accepted from the keyboard and then a search would be performed for the specified job and ID in the directory. If the specified job is not found, action is transferred to ACT(21) where the user will be asked to enter his request again.

ACT(21)

ACT(21) sets the KBACT to 20; if a match is found it will enter the JOB NAME and ID in PLTPARM and then go to NEWPLOT where the user will be asked to identify a specific plot.

ACT(22)

ACT(22) is activated when the user enters a plot number from the keyboard. Text is converted to a number and then action goes to NEWPLOT where the plot number is checked; if it is valid, the parameter is entered into PLTPARM; if it is not valid, a message will be displayed and the user will be asked to make another attempt to identify the plot.

ACT(23)

ACT(23) is a direct continuation of ACT(2) where the screen is prepared for accepting a keyboard message identifying a plot. This action would be taken in the event that the user has specified an invalid plot where set KBACT to 22.

ACT(24)

ACT(24) is the initialization step; it is the first action being done. It examines PLTPARM to determine whether or not the job name was specified and if the job number is valid. It also checks to see if it is in the middle of a job or if it is a new users' job. If it is continuing then a valid job number will exist and it will display the job name and ID, the status and the four operation messages. If it is not continuing, then the job number will not be valid and action will be transferred to ACT(1).

ACT(25)

ACT(25) follows directly from ACT(24) and is taken when a valid plot number is found in PLTPARM. If the plot number is valid, all information associated with the plot will be displayed on the screen. This is followed directly by DISPWIN where the default window will be displayed and is set for accepting the window co-ordinates from the user. If the plot number specified by the user is not valid, action is transferred to ACT(2).

Most actions terminate at statement ATE where an error message would be displayed if necessary. The cursor would be inserted if a keyboard action was inspected and the GDS was executed,i.e. displayed on the screen.

Following statement ATE the program goes into a wait state; waiting for an attention from the user. When such attention occurs, appropriate actions are taken as follows:

a) if function keyboard -1 is pressed, a forced return with a return code of -1 is taken.

b) if KBACT is pressed, the action specified by KBACT is taken.

c) if LPD, actions 1-18 are being taken according to the correlation value specified by LPD.

d) if LPD on a plot number, the plot number is set accordingly.

All other attentions from the user are ignored as only the ones specified above receive responses from the program.

PLOTAID

PLOTAID is the mainline routine from which all other functions are called. It is the only mainline program in this system; all others are function calls.

Each function called returns an integer value ranging between -3 and +4 as a return code. The most normal return code is 0; if the code is less than 0 then this indicates an error; if the code is greater than 0; it indicates the different actions to be taken.

The error condition is set so that when an error occurs, the errcode is saved and snap is made back to the start again. If the errcode= errstart indicating that the same error occurred again, action is terminated, otherwise start is made again.

To initialize the GSP and the file, PLTINIT is called and it initializes the values in PLTIDNT.

The transition table is set up as a matrix of numbers and all action consists of swapping these numbers. The first index is the program being called and the second index is the return code which the called program returns; for example, after IMAGINE is called an entry in the table would be T(5,0) indicating that IMAGINE returned a return code of 0. Action is then transferred to PLTEND (state=6) because the entry a T(5,0) is 6. Diagrams of the transition table are found on the following two pages to illustrate the transfer of action from one state to another.

By pressing the buttons, action is transferred to the states specified by the numbered buttons.

If the zero button is pressed twice, the action will stop completely. STATE=-1 indicates that the kill key was activated. An attempt at RESTART is made if the state is less than 1 by pressing the 31 button.

TRANSITION TABLE

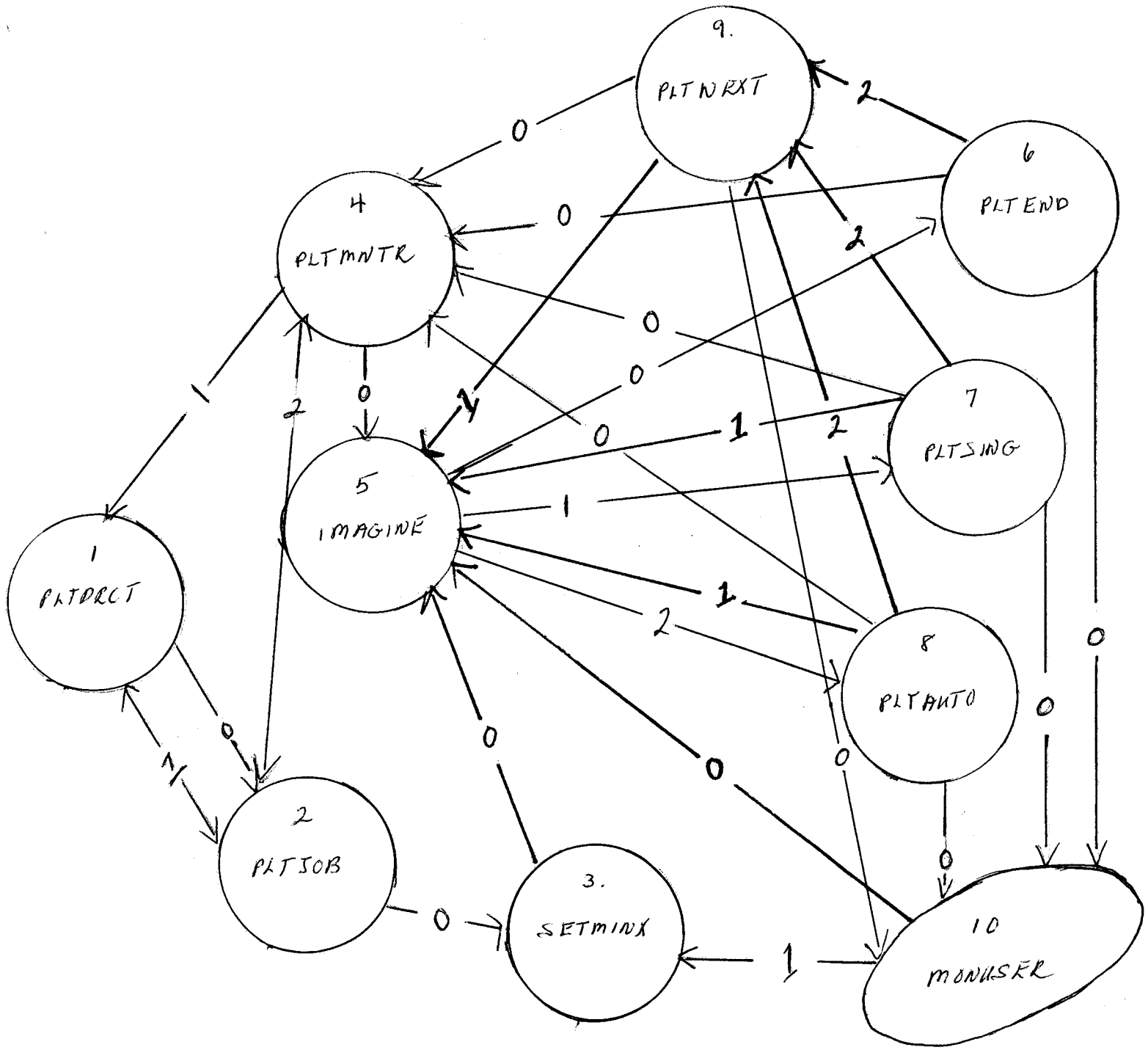
	-3	-2	-1	0	1	2	3	4	
-3	-3	-2	-1	0					
-2	-3	-2	-1	0					
-1	-3	-2	-1	0					
0	-3	-2	-1	0					
1	-3	-2	-1	2	2				PLTPRCT
2	-3	-2	-1	3	1	4			PLTJOB
3	-3	-2	-1	5					SETMINX
4	-3	-2	-1	5	1	2			PLTMNTR
5	-3	-2	-1	6	7	8			IMAGINE
6	-3	-2	-1	10		9			PLTEND
7	-3	-2	-1	10	5	9			PLTSING
8	-3	-2	-1	10	5	9			PLTAUTO
9	-3	-2	-1	10	5				PLTNEXT
10	-3	-2	-1	5	3				MONUSER
11									SPARE
12									SPARE

RETURN CODES

ROUTINES

-3 ≡ } READR return codes passed on - READ ERROR
-2 ≡ }
-1 ≡ KEY "0" was activated i.e. "KILL"
0 OKAY

* 3 and 4 are spare unused return codes
* 11 and 12 are spare unused routines


9.
PLTNRXT
6
PLTEND
4
PLTMNTR
7
PLTSING
5
IMAGINE
1
PLTDRCT
8
PLTAUTO
2
PLTJOB
3.
SETMINX
10
MONUSER

PLTINIT

PLTINIT initializes the GSP (Graphic Subroutine Package) with a single data set GDS. It sets the values in the external data area PLTIDNT.

The file access is initiated by reading the directory.

PLTEND

PLTEND is a program designed to handle the end of an image displayed on the screen. When the image has ended display, PLTEND returns one of three return codes as function values.

If a code of -1 is returned, an error message is given. If a code of 0 is returned, action is transferred back to the initialization state and if a code of 2 is returned, action is transferred to ACT(2) where a new plot is requested from the user.

PLTSING

PLTSING is a program to process the end of the buffer in a single shot. PLTSING has the same coding as PLTEND with the exception of an additional code=13 which calls for the rest of the plot to be displayed on the screen.

PLTAUTO

PLTAUTO is a program to process the automatic sequence images. The return codes from PLTAUTO are identical to those from PLTEND.

PLTNEXT

The function of PLTNEXT is to set up the display for the next image. It is common to PLTSING, PLTEND, and PLTAUTO and can run while an image is being displayed on the screen.

In setting up for the next image, the image number is incremented, the frame number is set to 1, and the window co-ordinates are set according to the mode.

PLTDRCT, PLTJOB and PLTMNTR

PLTDRCT, PLTJOB and PLTMNTR are three programs which do the equivalent functions which are done by MONUSER.

These programs have greater facility for handling operations but are not available to the user. The system can be brought into any one of these three states but only on special request.

APPENDIX

```
                                        /********* BSTROKE *********/
                                        /* CHARACTER STROKE FORMAT */
DCL  1     BSTROKE,
           2  LBIT BIT(1),              /* BEAM CONTROL BIT        */
           2  XBIT BIT(7),              /* X GRID COORD (0-127)    */
           2  CBIT BIT(1),              /* CONTINUE CHARACTER BIT  */
           2  YBIT BIT(7),              /* X GRID COORD (0-127)    */
      HALFWORD FIXED BIN(15) DEF BSTROKE;


                                        /********** DASET *********/
                                        /**************************/
DCL  1 DASET EXT CTL,                   /* DATA SET STRUCTURE     */
       2(#DES,                          /* NO.OF DESCRIPTORS      */
         #PNT,                          /* NO.OF POINTS.          */
         #GRP,                          /* NO.OF GROUPS.          */
         LDES,                          /* LAST DESCRIPTOR USED.  */
         LPNT)FIXED BIN(31),            /* LAST POINT USED.       */
       2 DSNM CHAR(8),                  /* DATA SET NAME.         */
       2 DADES(M#DES),                  /* DATA DESCRIPTORS.      */
         3 DANM CHAR(8),                /* GROUP/MEMBER NAME.     */
         3(PNTR,                        /* POINT 0 OF A GROUP  OR */
                                        /* OFSET OF MEMBER        */
           CUNT                         /* NO.OF MEMBERS       OR */
               )FIXED BIN(31),          /* NO.OF VECTORS.         */
         3(RMIN,                        /* RANGE OF POINTS IN A   */
           RMAX)FLOAT,                  /* MEMBER.                */
       2 POINTS(M#PNT)FLOAT ;           /* THE POINTS.            */
                                        /**************************/

/******* CORRELATION VALUES CONVENTIONS USED IN DEFDA  *******/
/*       R  CRR                                              */
/*       0000001II      INSTRUCTION       II    = INS        */
/*       100000000      DATA SET NAME                        */
/*       2000GGG00      GROUP NAME        GGG   = GRPP       */
/*       3MMMGGG00      MEMBER NAME       MMM   = MEMP       */
/*       4MMMPPPPP      POINT ID          PPPPP = PNTP       */
/*       5000VVVVV      VECTOR ID.        VVVVV = VECP       */
/*************************************************************/
```

```
DCL  1 DATAIN EXT,
       2 FILNM CHAR(8),
       2 INSTAT FIXED BIN(31),
       2 CARDIN,
         3 CARD CHAR(72),
         3 CID CHAR(8),
       2 IP FIXED BIN(31) ;
```

```
/******** DATAIN *********/
/*************************/
/* INPUT DATA STRUCTURE.  */
/* INPUT FILE DDNAME.     */
/* INPUT STATUS.          */
/* LAST INPUT CARD IMAGE  */
/* INPUT CARD TEXT.       */
/* CARD ID 73-80          */
/* POINTER TO THE LAST    */
/* CHARACTER BEING READ.  */
/*************************/
```

```
%DCL
    (#BF,
     #WF,
     #BC,
     #CG,
     #JOBS,
     #PLTS,
     #PNTS,
     I#REG,
     S#REG,
     #BP1,
     #BP2,
     #BP3) FIXED

%#BF=7200
%#CG=8
%#WF=#BF/4
%#BC=#BF/#CG
%#JOBS=(#BF-12)/16
%#PLTS=(#BC-25)/52
%#PNTS=(#BC-36)/4
%#BP1=#BF-12-#JOBS*16
%#BP2=#BC-25-#PLTS*52
%#BP3=#BC-36-#PNTS*4
%I#REG=200
%S#REG=100
```

```
/******DISKPARM*************/
/* NO OF BYTES IN BUFFER   */
/* NO OF WORDS IN BUFFER   */
/* NO OF BYTES IN RECORD   */
/* NO OF RECS IN REG       */
/* NO OF JOBS IN DRCT      */
/* NO OF PLOTS IN JHDR     */
/* NO OF POINTS IN DATA    */
/* INITIAL REGS IN FILE    */
/* SEC ALLOC IN REGS       */
/* NO OF BYTES IN PAD1     */
/* NO OF BYTES IN PAD2     */
/* NO OF BYTES IN PAD3     */
/*                         */;
/* SET BUFFER SIZE         */;
/* SET # OF RECS IN BUFFER */;
/* CALCULATE NO OF WORDS   */;
/* CALCULATE ALL OTHER     */;
/* COMPILE TIME VARIABLES  */;
/*                         */;
/*                         */;
/*                         */;
/*                         */;
/*                         */;
/*                         */;
/*                         */
/***************************/;
```

```
                                        /********* DISKFILE ********/
DCL  BUFFER (#WF) FIXED BIN(31)          /* THE FILE BUFFER ON WHICH*/
        BASED(BPNTR),                   /* THE RECORDS ARE BASED.  */
     BUR FIXED BIN(15) EXT,              /*CURRENT REGION IN BUFFER */
                                        /*                         */
     BPNTR POINTER EXT,                 /*     BUFFER POINTER      */
     JPNTR POINTER EXT,                 /*     JOB HEADER POINTER  */
     DPNTR POINTER EXT,                 /*     DATA RECORD POINTER */
    (IBPNTR  DEF BPNTR,                 /* FOR ARITHMETIC          */
     IJPNTR  DEF JPNTR,                 /* FOR ARITHMETIC          */
     IDPNTR  DEF DPNTR)                 /* FOR ARITHMETIC          */
          FIXED BIN(31),                /*                         */
     LIM  FIXED BIN(15) EXT,             /* NO OF RECS IN FILE      */
                                        /***************************/
                                        /***************************/
     1    DRCT BASED (BPNTR),            /* DIRECTORY RECORD        */
          2    DID CHAR(1),              /* RECORD IDENT            */
          2    SP1 CHAR(3),              /* PAD                     */
          2    JOBS FIXED BIN(31),       /* NUMBER OF JOB ENTRIES   */
          2    ENTRY(#JOBS),             /* JOB ENTRIES             */
               3 JOBNM CHAR(8),          /* JOB NAME                */
               3 JOBID FIXED(5),         /*  JOB IDENTIFICATION     */
               3 JOBSTAT CHAR(1),        /* JOB STATUS              */
               3 JADDR FIXED BIN(15),/* ADDR OF JOB HEADER          */
               3 SPARE1 CHAR(2),         /* PADDING TO ALIGN        */
          2    NEXTADDR FIXED BIN(15),/*POINTER TO LAST JHDR       */
          2    HIGHLIM FIXED BIN(15),/*NO OF REGIONS IN FILE       */
          2    PAD1 CHAR(#BP1),          /* PAD TO FILL FILE        */
                                        /***************************/
/*****************************************************************/
/*  THE FOLLOWING ARE THE CODES USED FOR STATUS CODES.           */
/*                                                               */
/*     BITS      ,    JOBSTAT      ,    PSTAT                    */
/*---------------------------------------------------------------*/
/*     0-3       ,   1 FOR NOW     ,  1 FOR NOW                  */
/*      4        , SOME PLOTS TO   , PLOT ON CALCOMP             */
/*               , BE PLOTTED      ,                             */
/*      5        , DELETE THE JOB  , DELETE THE PLOT             */
/*      6        , SOME PLOTS TO   ,      ---                    */
/*               , BE DELETED      ,                             */
/*      7        , WAS EXAMINED    , WAS EXAMINED                */
/*---------------------------------------------------------------*/
/*  THE STATUS IS INDICATED BY A 1 BIT.                          */
/*****************************************************************/
```

```
/******* DISKFILE CONT'D *******/

                                /*************************/
1     JHDR BASED (JPNTR),       /* JOB HEADER RECORD     */
      2    JID CHAR(1),         /* RECORD IDENT          */
      2    SP2 CHAR(3),         /* PAD                   */
      2    (PLOTS,              /* NUMBER OF PLOTS       */
            RECORDS)            /* NO.OF RECORDS IN JOB. */
            FIXED BIN(15),      /*                       */
      2    PLENTRY(#PLTS),      /* PLOT ENTRIES          */
           3(PRECS,             /* NO.OF RECORDS IN PLOT. */
             PADDR)             /* FIRST RECORD IN PLOT. */
             FIXED BIN(15),     /*                       */
           3 PSTAT CHAR(1),     /* PLOT STATUS           */
           3 SPARE3 CHAR( 3),   /* SPARE ROOM FOR LATER  */
           3 (XMAX,             /* MAX X IN PLOT         */
              XMIN,             /* MIN X IN PLOT         */
              XAVR,             /* AVERAGE X IN PLOT.    */
              XSDV,             /* ST.DEV. X IN PLOT.    */
              YMAX,             /* MAX Y IN PLOT         */
              YMIN,             /* MIN Y IN PLOT         */
              YAVR,             /* AVERAGE Y IN PLOT.    */
              YSDV) FLOAT BIN,  /* ST.DEV. Y IN PLOT.    */
           3 XSCLP FLOAT BIN,   /* X SCALING FACTOR - PLOT */
           3 YSCLP FLOAT BIN,   /* Y SCALING FACTOR - PLOT */
           3 CSCLP FLOAT BIN,   /* CHARACTER FACTOR - PLOT */
      2    JBNAM CHAR(8),       /* JOB NAME              */
      2    JBID FIXED(5),       /* JOB IDENTIFICATION    */
      2    JOBDAT CHAR(6),      /* JOB DATE.             */
      2    PAD2 CHAR(#BP2),     /* FILL OUT RECORD       */
                                /*************************/

                                /*************************/
1     DATA BASED(DPNTR),        /* DATA RECORD           */
      2    DATID CHAR(1),       /* RECORD IDENT          */
      2    SP3 CHAR(3),         /* PAD                   */
      2    (DXMAX,              /* MAX X FOR RECORD      */
            DXMIN,              /* MIN X FOR RECORD      */
            DXAVR,              /* AVERAGE X FOR RECORD. */
            DXSDV,              /* ST.DEV. X FOR RECORD. */
            DYMAX,              /* MAX Y FOR RECORD      */
            DYMIN,              /* MIN Y FOR RECORD      */
            DYAVR,              /* AVERAGE Y FOR RECORD. */
            DYSDV)FLOAT BIN,    /* ST.DEV. Y FOR RECORD. */
      2    POINT(#PNTS)         /* DATA POINTS           */
           FIXED BIN(31),       /* EVERYTHING GOES IN THIS */
                                /* THIS ARRAY            */
      2    PAD3 CHAR(#BP3);     /* FILL OUT RECORD       */
                                /*************************/
```

```
/******** DISKFILE CONT'D ********/

/*****************************************************/
/*   THE FORM OF POINT IS AS FOLLOWS.                */
/*     X'2A'  OPR  CNT  LEN   EACH OF THESE IS ONE BYTE */
/*  THE CODES FOR THE FIRST HALF BYTE OF OPR ARE;    */
/*        1      POINT DATA FOLLOWS                  */
/*        2      CHARACTER DATA FOLLOWS              */
/*        3      SPECIAL CHARACTER FOLLOWS           */
/*  IF BIT 4 IS 0 WITH CODE 3 THEN MOVE TO THE CHARACTER */
/*  WITH THE BEAM BLANKED. IF IT IS 1 THEN TURN THE BEAM */
/*  ON TO MOVE TO THE CHARACTER.                     */
/*****************************************************/


DCL  GETABLE                          /****** GETABLE ************/
        ENTRY(CHAR(100) VAR);         /* TAKE PARAMETER LIST      */
                                      /* AND READ NO. OF TABLES,  */
                                      /* AND TABLE NO.S AND GET   */
                                      /* TABLES IN CORE VIA GSP   */
                                      /***************************/


                                      /******* GSPDATA **********/
DCL  1  GSPDATA  EXTERNAL,           /* GSPDATA STRUCTURE       */
         2 (IGSP,                    /* GSP NAME                */
            #,                       /* NULL VARIABLE           */
            I2250,                   /* DEVICE NAME             */
            ILEVEL,                  /* ATTENTION LEVEL         */
            GDS1,                    /* GRAPHIC DATA SET 1      */
            GDS2)                    /* GRAPHIC DATA SET 2      */
             FIXED BIN(31);          /***************************/
```

```
                                        /******** GSPERR ***********/
DCL  GSPERR ENTRY                       /* GSP ERROR HANDLING       */
                                        /*                ROUTINE   */
        (LABEL);                        /* RETURN LABEL WHEN ERROR */
                                        /* CAUSED BY BUFFER OVERFLOW*/
                                        /* REQUIRES GSPRC SET TO A */
                                        /* NON-ZERO GSP RETURN CODE*/
                                        /* AND SETS GSPAI TO THE   */
                                        /* ADDITIONAL INFO BY GSP  */
                                        /***************************/
/*****************************************************************/
/* RICK. PASSING A "LABEL" AS A PARAMETER IS DANGEROUS.          */
/* WE PLAN TO REDUCE THE CORE REQUIREMENTS BY OVERLAY STRUCTURE*/
/* JUMPS BETWEEN LOAD MODULES WITHOUT PROPER RETURN  IS A SURE*/
/* WAY TO FELO DE SE. I SUGGEST INSTEAD OF "CALL GSPERR(LABEL)"*/
/* WRITE "GO TO LABEL( GSPERR )" JUMPING ACCORDING TO THE      */
/* RETURN CODE.                          D.J.C                 */
/*****************************************************************/


    /******************* GSPTRACE ***************************/
    /******  GSP STATUS FUNCTIONS AND TRACE ROUTINE  *********/
DCL (ITRCF,                         /* TEST RETURN CODE        */
     ITSTF,                         /* TEST GDS CHARACTERISTICS*/
     ITBPF)                         /* TEST INTEGER BEAM POS   */
          ENTRY  (FIXED BIN(31),FIXED BIN(31))
         RETURNS (FIXED BIN(31)),
     RTBPF                          /* TEST REAL BEAM POSITION */
          ENTRY  (FIXED BIN(31),FIXED BIN(31))
         RETURNS (FLOAT BIN),
     GSPTRCE                        /* TRACE GSP CALLS         */
          ENTRY (CHAR(30) VAR,       /* THIS CALL IDENTIFICATION*/
                FIXED BIN(31),       /* TRACE LEVEL MODE        */
                                    /* =0  =] NO TRACE         */
                                    /* =1  =] TRACE ERRORS ONLY*/
                                    /* =2  =] TRACE EVERY CALL */
                FIXED BIN(31));      /* GSP NAME IDENTIFICATION */

/* E.G.  CALL GSPTRCE('INGDS',1,IGSP);  */
```

```
L  GSPTRCE   ENTRY                          /******** GSP TRACE ********/
       (CHAR(30) VARYING,                   /* GSP TRACE ROUTINE        */
        FIXED BIN(31),                      /* CALL IDENTIFICATION      */
                                            /* TRACE LEVEL MODE         */
                                            /* =0 =] NO TRACE           */
                                            /* =1 =] TRACE ERRORS ONLY */
                                            /* =2 =] TRACE EVERY CALL   */
        FIXED BIN(31));                     /* GSP NAME IDENTIFICATION */
                                            /**** ********************/
  /*   I.E.   CALL GSPTRCE('INGDS',1,GSP);                              */
```

```
                                            /********IDNTS**************/
   DCL  1 IDNTS STATIC EXT,                 /** GSP IDENTIFIERS ********/
          2( GSP,                           /* GSP NUMBER               */
             #  ,                            /* THE NULL VALUE           */
             ID2250,                        /* 2250 ID NUMBER TO GSP    */
             XIS,                           /* AXIS GSP IDENT.          */
             PLT,                           /* PLOT AREA GSP IDENT.     */
             MSG,                           /* MESSAGE AREA GSP IDENT.  */
             MSG1,                          /* MSG1 EQUIV. WITH MSG     */
             MSG2,                          /* MSG2 EQUIV. WITH MSG     */
             SP1,                           /*                          */
             SP2,                           /*                          */
             SP3,                           /*     SPARES.              */
             SP4,                           /*                          */
             ATTL,                          /* ATTENTION LEVEL IDNT.    */
             COKEY,                         /* COORDINATES KEY          */
             SK1,                           /*                          */
             SK2,                           /*      SPARES.             */
             SK3,                           /*                          */
             SK4,                           /*                          */
         #XXXV,                             /* CONSTANT -35             */
            VI,                             /* CONSTANT 6 (SIX)         */
             V,                             /* CONSTANT 5 (FIVE)        */
            IV,                             /* CONSTANT 4 (FOUR)        */
           III,                             /* CONSTANT 3 (THREE)       */
            II,                             /* CONSTANT 2 (TWO)         */
             O,                             /* CONSTANT 1 (ONE)         */
             Z    )FIXED BIN(31);           /* CONSTANT 0 (ZERO)        */
                                            /***************************/
```

```
DCL  IMAGINE  ENTRY                    /******** IMAGINE **********/
                                       /* IMAGE GENERATION ROUTINE*/
                                       /* PRODUCES A SCREEN IMAGE */
                                       /*FROM DATA RECORDS POINTED*/
                                       /* TO BY PLTP AND EXEC. IT.*/
     RETURNS (FIXED BIN(31));           /* FUNCTION VALUE IS RETURN*/
                                       /*  CODE:                  */
                                       /* =-2 =] DATA READ ERROR. */
                                       /* = 0 =] NORMAL RETURN    */
                                       /* = 1 =]  BUFFER FULL IN  */
                                       /*  SINGLE SHOT MODE.      */
                                       /* = 2 =] BUFFER FULL IN   */
                                       /*  SEQUENCE MODE. -STEOS- */
                                       /***************************/

                                       /********** IMGINIT ********/
DCL IMGINIT ENTRY;                     /* IMAGINE INITIALIZER     */
                                       /* DISPLAY PLOT IDENTITY   */
                                       /* SET GDS FOR IMAGINE     */
                                       /***************************/

DCL ITRCF                              /******** ITRCF ************/
    ENTRY                              /* TEST GSP RETURN CODE    */
         (FIXED BIN(31),               /* GSP: GSP NAME VARIABLE  */
         FIXED BIN(31))                /* CODE: RETURN CODE LEVEL */
                                       /* =9 FOR ADDITIONAL INFO  */
    RETURNS(FIXED BIN(31));             /* GSP RETURN CODE         */
                                       /***************************/
```

```
                                        /******** MINBYTE **********/
DCL  MINBYTE                            /* FIND THE INDEX OF THE   */
     ENTRY                              /* MINIMUM BYTE OF THE WORD*/
         (FIXED BIN(31),                /* GIVEN HERE              */
           )                            /* AND GIVE THE BYTE VALUES*/
                                        /* IN THE 4 ELT ARRAY HERE */
     RETURNS (FIXED BIN(31));            /* RETURN INDEX HERE       */
                                        /***************************/
```

```
                                        /******** MINIMAX *********/
L  1 MINIMAX(0:PLTPARM.#REC) EXTERNAL CONTROLLED,
                                        /* MINIMUM AND MAXIMUM     */
                                        /*  COORD VALUES IN BOTH   */
                                        /*  THE PLOT AND EACH DATA */
                                        /*  RECORD.                */
     2    (RXMIN,                       /* REAL X MINIMUM          */
           RXMAX,                       /* REAL X MAXIMUM          */
           RYMIN,                       /* REAL Y MINIMUM          */
           RYMAX)                       /* REAL Y MAXIMUM          */
                FLOAT BIN;               /*                         */
                                        /***************************/
```

```
                                        /******** MONUSER **********/
DCL   MONUSER ENTRY                     /* USER ORIENTED MONITOR   */
     /* JOB,PLOT AND IMAGE CONTROL ALL IN ONE SIMPLE STEP.          */
     RETURNS(FIXED BIN(31)) ;       /* RETURN CODES :               */
     /* -3 = FAILED TO READ OR WRITE THE DIRECTORY.                 */
     /* -2 = FAILED TO READ OR WRITE THE JOB HEADER.                */
     /* -1 = THE  KILL  BUTTON WAS ACTIVATED.                       */
     /*  0 = OK, ALL SET TO CALL IMAGINE.                           */
     /*  1 = CHANGE OF PLOT CALL SETMINX BEFORE IMAGINE.            */
     /* MODE,MOMO,LLX,LLY,URX,URYIAND THE SCALES ARE SET.           */
     /*************************************************************/
```

```
DCL  MOVE EXT ENTRY                 /*********MOVE*************/
     (,                             /* SUBROUTINE TO MOVE      */
      ,                             /* FIRST PARAMETER IS      */
      FIXED BIN(31),                /* ADDRESS TO MOVE TO      */
      FIXED BIN(31),                /* SECOND IS FROM ADDRESS  */
      FIXED BIN(31));               /* TO OFFSET               */
                                    /* FROM OFFSET             */
                                    /* NUMBER OF BYTES         */
                                    /***************************/

                                    /******** OCCURS ***********/
DCL  OCCURS  ENTRY                  /* FUNCTION TO SCAN A CHAR */
    (     ,                         /* STRING FOR SPECIAL CHARS*/
          ,                         /* ADDRESS OF NEW STRING   */
     FIXED BIN(31),                 /* ADDRESS OF OLD STRING   */
                                    /* INPUT  - LENGTH OF OLD  */
                                    /* OUTPUT - LENGTH OF NEW  */
                                    /* **NOTE** WHEN LENGTH ± 0*/
                                    /* AS FOR CENTERED CHARS   */
                                    /* THEN #CHARS SET TO 1    */
                                    /* AND #NEW LINES SET TO 0 */
                                    /* LENGTH IS NOT CHANGED   */
                                    /* AND RESULT IS GARBAGE   */
     FIXED BIN(31),                 /* OUTPUT - NO. OF NEW LINES*/
     FIXED BIN(31),                 /* OUTPUT - MAX# OF CHARS/ */
                                    /*                    LINE */
                                    /**** OPTIONAL PARAMETERS **/
     FIXED BIN(31),                 /* NEW STRING OFFSET       */
     FIXED BIN(31))                 /* OLD STRING OFFSET       */
                                    /* IGNORE ERROR MESSAGE    */
                                    /* IF THESE PARMS NOT CODED*/
     RETURNS (FIXED BIN(31));        /* FUNCTION VALUE IS A FULL*/
                                    /* WORD WHICH CONTAINS THE */
/*BYTE1,BYTE2,BYTE3,BYTE4*/          /* INDEX OF THE FIRST OCCUR*/
/*BACK , CAR , SUP , SUB */          /* -ENCE OF EACH OF THE 4  */
/*SPACE, RET ,     ,     */          /* SPECIAL CHARS           */
                                    /***************************/
```

```
DCL PACK EXT ENTRY                    /******* PACK *************/
     (FIXED BIN(31),                  /* PACKING ROUTINE          */
      FIXED BIN(31),                  /* OPR: THE OPERATION CODE.*/
      FIXED BIN(31))                  /* CNT: NO.OF ITEMS.        */
     RETURNS(FIXED BIN(31)) ;         /* LEN: NO.OF WORDS.        */
                                      /* RETURNS THE PACKED WORD.*/
                                      /*************************/
```

```
                                      /******** PARAM ************/
 DCL  1 PARAM STATIC EXT,             /** GRAPHIC PARAMETERS    ***/
        2( XMIN,                      /*  RANGE OF THE X AXIS     */
           XMAX,                      /*                          */
           YMIN,                      /*  RANGE OF THE Y AXIS     */
           YMAX,                      /*                          */
           XSP1,                      /*      SPARES              */
           XSP2,                      /*                          */
           XSP3,                      /*                          */
           XSP4,                      /*      SPARES              */
           XSP5,                      /*                          */
           XSP6 )FLOAT,               /*                          */
        2( XPRC,                      /*  FORMAT PRECISION        */
           YPRC,                      /*                          */
           XEXP,                      /*  EXPONENTS (10)          */
           YEXP,                      /*                          */
           XINC,                      /*  NO.OF INCREMENTS        */
           YINC,                      /*                          */
           PS1,                       /*                          */
           PS2,                       /*                          */
           PS3,                       /*                          */
           PS4,                       /*      SPARES.             */
           PS5,                       /*                          */
           PS6 )FIXED BIN(31) ;       /*                          */
```

```
                                        /******** PLTAUTO **********/
DCL   PLTAUTO ENTRY                     /* PROC TO MONITOR THE      */
     /* DISPLAY AFTER A COMPLETION CODE OF 2 FROM IMAGINE.          */
     RETURNS(FIXED BIN(31)) ;           /* RETURN CODES :           */
     /* -2 =                                                        */
     /* -1 = THE  KILL  BUTTON WAS ACTIVATED.                       */
     /*  0 = OK, ALL FINISHED RETURN TO PLTMNTR.                    */
     /*  1 = RETURN TO IMAGINE FOR ANOTHER FRAME.                   */
     /*  2 = CALL PLTNEXT FOR NEXT IMAGE.                           */
     /*************************************************************/


                                        /******** PLTDRCT **********/
DCL   PLTDRCT ENTRY                     /* PROC TO DISPLAY THE      */
     /* DIRECTORY OF THE DISK FILE AND PROVIDE THE USER WITH        */
     /* FACILITIES TO PICK A SPECIFIC JOB FOR FURTHER PROCESS-*/
     /* ING OR MODIFYING THE JOB NAME OR JOB STATUS.                */
     RETURNS(FIXED BIN(31)) ;           /* RETURN CODES :           */
     /* -3 = FAILED TO READ THE DIRECTORY.                          */
     /* -2 = FAILED TO WRITE THE DIRECTORY.                         */
     /* -1 = THE "KILL" BUTTON WAS ACTIVATED.                       */
     /*  0 = OK, JOB# WAS CHANGED.                                  */
     /*  1 = OK, JOB# WAS NOT CHANGED.                              */
     /* IN ADDTION, THE PLTPARM JOB#,JOBP,PJBNM, AND PJBID          */
     /*              ARE SET BY THIS PROCEDURE.                     */
     /*************************************************************/


                                        /******** PLTEND **********/
DCL   PLTEND ENTRY                     /* PROC TO MONITOR THE      */
     /* DISPLAY AFTER A COMPLETION CODE OF 0 FROM IMAGINE.          */
     RETURNS(FIXED BIN(31)) ;           /* RETURN CODES :           */
     /* -2 =                                                        */
     /* -1 = THE  KILL  BUTTON WAS ACTIVATED.                       */
     /*  0 = OK, ALL FINISHED RETURN TO PLTMNTR.                    */
     /*  1 =                                                        */
     /*  2 = CALL PLTNEXT, FOR NEXT IMAGE.                          */
     /*************************************************************/
```

```
                                        /******** PLTJOB  **********/
DCL   PLTJOB  ENTRY                     /* PROC TO DISPLAY THE      */
      /* HEADER OF A JOB               AND PROVIDE THE USER WITH  */
      /* FACILITIES TO PICK A SPECIFIC PLOT FOR FURTHER           */
      /* PROCESSING OR MODIFY THE STATUS OF THE PLOT.             */
      RETURNS(FIXED BIN(31)) ;        /* RETURN CODES :           */
      /* -3 = FAILED TO READ OR WRITE THEYDIRECTORY.              */
      /* -2 = FAILED TO READ OR WRITE THERJOB HEADER.             */
      /* -1 = THE "KILL" BUTTON WAS ACTIVATED.                    */
      /*  0 = OK, PLT# WAS CHANGED.                               */
      /*  1 = OK, PLT# WAS NOT CHANGED.                           */
      /* IN ADDTION, THE PLTPARM PLT#,PLTP,AND #REC               */
      /*              ARE SET BY THIS PROCEDURE.                  */
      /*************************************************************/
```

```
                                        /******** PLTNEXT **********/
DCL   PLTNEXT ENTRY                     /* PROC TO SET UP THE MODE */
      /* AND BOUNDERIES FOR THE NEXT IMAGE IN A SEQUENCE.         */
      RETURNS(FIXED BIN(31)) ;        /* RETURN CODES :           */
      /* -2 = READR ERROR ON JOBHEADER.                           */
      /* -1 =                                                     */
      /*  0 = OK, ALL FINISHED RETURN TO PLTMNTR.                 */
      /*  1 = RETURN TO IMAGINE FOR ANOTHER IMAGE.                */
      /*  2 =                                                     */
      /*************************************************************/
```

```
                                        /********** RTBPF **********/
DCL  RTBPF  ENTRY                       /* REAL BEAM POSITION      */
        (FIXED BIN(31),                 /* GDS IDENTIFICATION      */
         FIXED BIN(31))                 /* COORD TYPE REQUESTED    */
     RETURNS (FLOAT BIN);               /* COORD VALUE             */
                                        /***************************/
```

```
                                        /******** PLTPARM **********/
DCL  1    PLTPARM EXTERNAL,             /* PLOT PARAMETERS          */
          2   (JOB#,                    /* CURRENT JOB NUMBER       */
               PLT#,                    /* CURRENT PLOT NUMBER      */
               REC#,                    /* CURRENT RECORD NUMBER    */
               ITM#,                    /* CURRENT ITEM NUMBER      */
               IMG#,                    /* IMAGE NO.IN A PLOT.      */
               JOBP,                    /* JOB HEADER RECORD NO.    */
               PLTP,                    /* PLOT DATA RECORD NO.     */
               #REC,                    /* NO.OF RECORDS IN PLOT    */
               GSPRC,                   /* GSP RETURN CODE (ITRCF)  */
               GSPAI,                   /* ADDITIONAL INFO (ITRCF)  */
               #SSCIS,                  /* # OF SSCISORING ERRORS   */
               STATE,                   /* STATE OF THE SYSTEM  :   */
     /*  0 = PLTINIT                        1 = PLTDRCT             */
     /*  2 = PLTJOB                         3 = SETMINX             */
     /*  4 = PLTMNTR                        5 = IMAGINE             */
     /*  6 = PLTEND                         7 = PLTSING             */
     /*  8 = PLTAUTO                        9 = PLTNEXT             */
     /* 10 = MONUSER                       11 =                     */
               MODE,                    /* IMAGINE MODE :           */
                                        /* IN GENERAL:              */
                                        /*    POSITIVE =] RESTART   */
                                        /*    NEGATIVE =] CONTINUE  */
                                        /* ABS VALUE:               */
                                        /*    =1 =] SINGLE SHOT     */
                                        /*    =2 =] MULTIPLE SHOTS  */
               FRM#,                    /* FRAME NO.IN AN IMAGE.    */
               MOMO ,                   /* WINDOW ADVANCE MODE :    */
                                        /* 0 = NORMAL SCANN         */
                                        /*+-1 = LEFT OR RIGHT HALF.*/
                                        /*+-2== LEFT OR RIGHT FULL.*/
                                        /*+-3 = UP OR DOWN HALF    */
                                        /*+-4 = UP OR DOWN FULL    */
               #JOB  ,                  /*NO.OF JOBS IN FILE.      */
               #BYPASS) FIXED BIN(15),  /* NO. OF RECORDS BYPASSED */
          2   (SPARE5,                  /*                          */
               TBL#,                    /* STROKE TABLE NUMBER      */
               CORRVAL,                 /* ITEM CORRE$ATION VALUE   */
                                        /* = REC# *10**4 + ITM#     */
```

/****** PLTPARM CONT'D ******/

```
     SPARE6) FIXED BIN(31),    /*                          */
2   ( DX ,                     /* X AND Y INCREMENTS FOR   */
      DY ,                     /* AUTO SEQ. OF A PLOT.     */
     CSCL ,                    /* CHARACTER SCALE          */
                               /* -1.= BASIC 2250  HAR.S   */
                               /*  IGNORE NONEHORIZONTALS  */
                               /*  0.= IGNORE ALL CHAR.S   */
                               /*  1.= NO SCALING.         */
                               /* 10.= SCALE ACCORDINGLY.  */
     XSCL,                     /* SCALING FACTOR FOR X     */
     YSCL,                     /* SCALING FACTOR FOR Y     */
     LLX,                      /* LOWER LEFT IMAGE COORD.  */
     LLY,                      /* LOWER LEFT IMAGE COORD.  */
     URX,                      /* UPPER RIGHT IMAGE COORD.*/
     URY)FLOAT BIN ,           /* UPPER RIGHT IMAGE COORD.*/
2    PJBNM CHAR(8),            /* CURRENT JOB NAME.        */
2    PJBID FIXED(5),           /* CURRENT JOB ID.          */
2   (CJS,                      /* CURRENT JOB STATUS,      */
     CPS)CHAR(1),              /* CURRENT PLOT STATUS.     */
2    SPARE CHAR(3);            /* FOR LATER USE            */
                               /*************************/
```

```
                                        /******* PLTIDNT ***********/
L  1    PLTIDNT   EXTERNAL,             /*                          */
        2   (GSP,                       /* GSP NAME                 */
             #,                         /* GSP NULL VARIABLE        */
             ID2250,                    /* GSP DEVICE NAME          */
             ATTL,                      /* GSP ATTENTION LEVEL      */
             GDS ,                      /* THE ONLY ONE GDS.        */
             IDSP,                      /* NOT USED                 */
             C0,                        /* VARIOUS                  */
             C1,                        /*        FULL              */
             C2,                        /*            WORD          */
             C3,                        /*                CONSTANTS*/
             C4,                        /*                          */
             C5,                        /*                          */
             C35)                       /*                          */
                  FIXED BIN(31);        /***************************/


                                        /******** PLTINIT **********/
 DCL  PLTINIT  ENTRY                    /* INTIALIZE PLOT ROUTINES */
          (FIXED BIN(31),               /* ID: GSP DEVICE NUMBER   */
           FIXED BIN(31));              /* MAXIOB: GRAPHIC OUTPUT  */
                                        /*    AREA LENGTH          */
                                        /***************************/


                                        /******** PLTMNTR **********/
 DCL   PLTMNTR ENTRY                    /* PROC TO MONITOR THE     */
      /* DISPLAY GENERATED BY IMAGINE.                              */
      RETURNS(FIXED BIN(31)) ;          /* RETURN CODES :          */
      /* -2 = FAILED TO READ OR WRITE THE JOB HEADER.               */
      /* -1 = THE "KILL" BUTTON WAS ACTIVATED.                      */
      /*  0 = OK, ALL SET TO CALL IMAGINE.                          */
      /*  1 = CHANGE OF JOB, GO TO CALL PLTDRCT.                    */
      /*  2 = CHANGE OF PLO, GO TO CALL PLTJOB.                     */
      /* MODE,MOMO,LLX,LLY,URX,URYIAND THE SCALES ARE SET.          */
      /*************************************************************/
```

```
                                        /******** PLTSING **********/
DCL   PLTSING ENTRY                     /* PROC TO MONITOR THE      */
     /* DISPLAY AFTER A COMPLETION CODE OF 1 FROM IMAGINE.          */
     RETURNS(FIXED BIN(31)) ;           /* RETURN CODES :           */
     /* -2 =                                                        */
     /* -1 = THE  KILL  BUTTON WAS ACTIVATED.                       */
     /*  0 = OK, ALL FINISHED RETURN TO PLTMNTR.                    */
     /*  1 = RETURN TO IMAGINE FOR ANOTHER FRAME.                   */
     /*  2 = CALL PLTNEXT FOR NEXT IMAGE.                           */
     /*************************************************************/
```

```
                                        /******** READR ************/
DCL  READR EXT ENTRY                    /* PROC. TO READ RECORDS   */
        (FIXED BIN(15),                 /* WHAT: A CODE INDICATING */
                                        /*  THE TYPE OF REC. TO BE */
                                        /*  READ: 1- DIRECTORY REC,*/
                                        /*        2- JOBHDR REC,   */
                                        /*        3- DATA   REC.   */
                                        /*        4- BLANK  REC.   */
                                        /* IF WHAT±0 THE SAME       */
                                        /* ACTION TAKES PLACE ONLY */
                                        /* TLL PREVIOUS RECORD IN  */
                                        /* THE BUFFER IS WRITTEN   */
                                        /* BEFORE THE NEXT IS READ */
         FIXED BIN(15))                 /* RECNO: ABS.REC.ADDRESS  */
                                        /* IF WHAT=1 IT MUST BE 0. */
        RETURNS(FIXED BIN(31));         /* RETURNS COMPLETION CODE */
                                        /* -3 - RECNO OUT OF RANGE,*/
                                        /* -2 - INVALID RECNO,     */
                                        /* -1 - INVALID WHAT,      */
                                        /*  0 - OK                 */
                                        /* 1,2,AND 3 ID ERROR.     */
                                        /***************************/
```

```
/*************************************************************/
/*   N.B.   ****  THE FIRST CALL TO READR IN A PROGRAM MUST   */
/*          BE FOR THE DIRECTORY RECORD AND WHAT MUST BE ] 0  */
/*          IE.   THE FIRST CALL MUST BE READR(1,0)  *****    */
/*          IF THIS IS NOT DONE READR WILL NOT BE INITIALIZED */
/*          PROPERLY AND WILL BOMB WITH A DATA EXCEPTION.     */
/*************************************************************/
```

```
                                        /******** TBLDATA **********/
DCL  1  TBLDATA  EXTERNAL,              /* STROKE TABLE DATA        */
          2  N  FIXED BIN(31),          /* NUMBER OF STROKE TABLES */
          2  TBL#(20) FIXED BIN(31), /* STROKE TABLE NUMBERS  */
          2  TADDR(20) POINTER;         /* TABLE ADDRESS IN CORE    */
                                        /***************************/
```

```
                                        /******* UNPK **************/
DCL UNPK EXT ENTRY                      /* UNPACKING ROUTINE.       */
     (FIXED BIN(31),                    /* PNT:WORD TO BE UNPACKED.*/
      FIXED BIN(31),                    /* CNT: NO.OF ITEMS.        */
      FIXED BIN(31))                    /* LEN: NO.OF WORDS.        */
     RETURNS(FIXED BIN(31)) ;           /* RETURNS OPR FIELD FOUND.*/
                                        /* IN THE PACKED WORD, THE */
                                        /* CNT AND LEN FIELDS ARE  */
                                        /* RETURNED THROUGH PARAM.S*/
                                        /* IF THE FIRST BYTE OF PNT*/
                                        /* NE TO X'2A' RETURNS -1 .*/
                                        /***************************/
```

```
                                    /********SETMINX ***********/
DCL  SETMINX  ENTRY                 /* INITIALIZE MINIMAX TABLE*/
                                    /*INPUT:                   */
                                    /*  JOBP - RECNO OF JHDR   */
                                    /*  PLT# - PLOT NUMBER     */
                                    /*OUTPUT:                  */
                                    /*  #REC - # RECS IN PLOT  */
                                    /*  PLTP - RECNO OF PLOT   */
                                    /*  MINIMAX - COORD LIMITS */
                                    /*  OF WHOLE PLOT (I=0) AND*/
                                    /*  OF EACH RECORD(I]0).   */
        RETURNS(FIXED BIN(31));     /*RETURN CODES:            */
                                    /* = 0 =] NORMAL RETURN    */
                                    /* =-2 =] JOB HEADER NOT RD*/
                                    /* =-3 =] DATA READ ERROR  */
                                    /*OPERATION:               */
                                    /* FREES MINIMAX STORAGE,  */
                                    /* READS JHDR AND SETS VARS*/
                                    /* ALLOCATES MINIMAX AND   */
                                    /* INITIALIZES ENTRIES FROM*/
                                    /* EACH DATA RECORD.       */
                                    /***************************/


      /************* STABLE *************/

DCL  1  STABLE BASED(GSPNTR),
          2  NUMBER FIXED BIN(31),
          2  NEXT POINTER,
          2  CADDR(0:255) FIXED BIN(31),
     1  STROKES BASED(STPNTR),
          2  DATA(100) FIXED BIN(15),
    (IGPNTR DEF GSPNTR, ISPNTR DEF STPNTR) FIXED BIN(31);
```