# Chapter 11

# Plotting

## 11.1 Overview

Plotting in AIPS is usually a two-step process. First a task or a verb creates an AIPS "plot file", which consists of plot-device independent "commands" that tell a device how to draw the plot. This file is always an extension file associated with a cataloged file. The second step in obtaining a plot is to run a task to read the plot file and write it to a specific device, such as a TV, or a hardcopy plotter. This two-step method greatly reduces the number of plot programs that must be written and maintained. For instance, if a new graphics device is added to the system, then only one new program, that reads the plot file and writes to the new device, is needed. All the other plotting programs work with no modification. Another advantage is that a plot file may exist for an extended period of time, thus allowing plots to be written to different devices, and copies to be generated at later times, without duplicating the calculations needed in making the plot.

There are exceptions to the two-step process. For example, slices of map files can be plotted directly on the Tektronix 4012. This is done to simplify matters in interactive situations such as gaussian fitting of slices. The plotting package now also offers an "immediate mode" in which it writes directly to a TV display device and omits the plot file entirely. This is good for quick looks and parameter setting, but leads to duplication when tasks need to be rerun to produce the plot file for display on other devices.

AIPS contains some very powerful routines for plotting in a variety of coordinate systems in use in astronomy. The complexity of these routines is commensurate to their power. Fortunately, a set of plot program templates exist to provide a starting point. These routines are described in a later section in this chapter.

## 11.2 PLOT FILES

### 11.2.1 General Comments

Plot files are a generalized representation of a graphics display. They contain scaling information and commands for drawing lines, pixels, and characters, and a command for putting miscellaneous information in the image catalog. The image catalog is used by programs that must know details about an image currently displayed on the graphics device in order to allow user interaction with the device. For example, a program may want to read a cursor position and translate it to the coordinate system of the image displayed on the graphics device.

The records in plot files do not include a record length value. This means that it is inconvenient to invent new types of records (i.e., new opcodes) or to add new values on to the end of records of existing types because all of the programs must be changed. On the other hand, the rigid format definitions aided in debugging the code several years ago and continue to assure the integrity of I/O systems (AIPS device plotting programs refuse to proceed if they encounter an unknown opcode). So far, the increased flexibility supplied by length values seems not to have been absolutely required in AIPS.

The character drawing record includes neither a size value nor an angle value. This is because character plotting capabilities are device dependent. Orientations are either vertical or horizontal (and not backwards) and the position offsets for plotting character strings are specified in units of the device character size, permitting the device plotting program to position strings nicely no matter what size it chooses to use. It also follows that most plots produced by AIPS have only one size of character. One AIPS application program (PROFL) draws its own characters by using the line drawing commands in order to plot characters with arbitrary size, orientation, and even perspective.

### 11.2.2 Structure of a Plot File

The first physical record (256 words) in the plot file contains information about the task which created the file. It is not logically part of the "plot file", but is there to provide documentation of the file's origins. This record is ignored by the programs that actually do the plotting. The primary use of this information is by the the verb EXTLIST that lists all the plot files associated with a cataloged file. When new types of plots are added to AIPS, an experienced programmer should update the verb EXTLIST (found in subroutine AU8A) to list useful things about the plot. Otherwise the verb will print a line telling the user that he has a plot file of type UNKNOWN. A novice AIPS programmer should leave this code alone.

The contents of the first physical record are task-dependent and have the form:

```
FIELD   TYPE        DESCRIPTION
  1.    H(3)       Task name (6 character hollerith string)
  2.    I(6)       Date/time of file creation YYYY,MM,DD,HH,MM,SS
  3.    I          Number of (integer) words of task parameter data
  4.    R(*)       Task parameter block as transmitted from AIPS
                   (preferably with defaults replaced by the values
                    used).
```

The rest of the plot file contains a generalized representation of a graphics display. This representation is in the form of scaling information and commands for drawing lines, pixels, and characters and a command for putting miscellaneous information in the image catalog.

The lowest level plot file I/O routines read and write 256-word blocks. The applications programmer will be concerned with routines that read and write logical records. The logical records are of 6 types and vary in length. With the exception of the "draw pixels" record, logical records do not cross the block boundaries. Unused space at the end of a block consists of integer zeros. All values in the plot file are I variables or Hollerith characters. This aids in exporting plot files to other computers via tape. Unfortunately, this also limits the values that can be stored in the plot file, thus forcing us to use a scaling factor and offset for some plots to prevent integer overflow. The scaling factor and offset are not in the plot file. If we are not careful, this can cause some problems for interactive tasks that read positions from a graphics device and then try to convert them to the original coordinates. These interactive tasks must make do with information from the map header and data from the "miscellaneous information" record.

Plot files have names of the format PLdsssvv, where d is the format revision letter, sss is the catalog slot number of the associated map, and vv is the version number. Hexadecimal code is used for sss and vv, limiting the number of versions to 255.

### 11.2.3 Types of Plot File Logical Records

**Initialize plot record**

The first logical record in a plot file must be of this type. It is written by the basic subroutine GINIT; see the end of this chapter for the details of the calling sequence. GINIT also determines whether the plot goes to a file or directly to the TV.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equal to 1 for this record type).
```

```
  2.   I      User number.
  3.   I(3)   Date:  yyyy, mm, dd
  4.   I       ITYPE: type code 1 => misc.,  2 => CNTR,   3 => GREYS,
                4 => PROFL,  5 => SL2PL,  6 => PCNTR,  7 => IMEAN,
                8 => UVPLT,  9 => GNPLT, 10 => VBPLT, 11 => PFPLn,
               12 => GAPLT, 13 => PLCUB, 14 => IMVIM, 15 => TAPLT,
               16 => POSSM, 17 => SNPLT, 18 => KNTR,  19 => UVHGM,
               20 => ISPEC
```

**Initialize for line drawing record**

This record provides scaling information needed for a plot. The plot consists of a "plot window" in which all lines are drawn and a border (defined in terms of character size) in which labeling may be written. The second record in a plot file must be of this type. This record is written by the subroutine GINITL.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equal to 2 for this record type).
  2.   I      X Y ratio * 100.  The Ratio between units on the X
              axis and units on the Y axis (X / Y).  For example
              if the decrement between pixels in the X direction
              on a map is twice the decrement in the Y direction
              the X Y ratio can be set to 2 to provide proper
              scaling.  Some programs may ignore this field. For
              example TVPL when writing grey scale plots to the TV.
  3.   I      Scale factor (currently 16383 in most applications).
              This number is used in scaling graph positions before
              they are written to disk.  BLC values in field 4 are
              represented on disk by zero and TRC values are
              represented by integers equal to the scale factor).
  4.   I(4)   The bottom left hand corner X and Y values and the top
              right hand X and Y values respectively in the plot
              window (in pixels).
  5.   I(4)   1000 * the fractional part of a pixel allowed to occur
              outside the (integer) range of BLC and TRC (field 4
              above) in line drawing commands
  6.   I(4)   10 * the number of character positions outside the
              plot window on the left, bottom, right, and top
              respectively
  7.   I(5)   Location of the X Y plane on axes 3,4,5,6,7.  This
              field is valid only for plots associated with a map.
```

**Initialize for grey scale record**

This record, if needed, must follow the "init for line drawing" record. This record allows proper interpretation of pixels for raster type display devices. Programs that write to line drawing type devices (e.g., the Tektronix 4012) ignore this record. Subroutine GINITG is used to write this record into the plot file.

```
FIELD  TYPE     DESCRIPTION
  1.    I       Opcode (equals 3 for this record type).
  2.    I       Lowest allowed pixel intensity.
  3.    I       Highest allowed pixel intensity.
  4.    I       Number of pixels on the X axis.
  5.    I       Number of pixels on the Y axis.
```

### Position record

This record, written by subroutine GPOS, tells a device where to start drawing a line, row/column of pixels, or character string.

```
FIELD  TYPE     DESCRIPTION
  1.    I       Opcode (equals 4 for this record type).
  2.    I       scaled x position, i.e., a value of 0 represents the
                BLC values defined in the "init for line drawing"
                record, and a value equal to the scale factor
                represents the TRC value.
  3.    I       Scaled Y position.
```

### Draw vector record

Subroutine GVEC writes this record to tell a device to draw a line from the current position to the final position specified by this record.

```
FIELD  TYPE     DESCRIPTION
  1.    I       Opcode (equals 5 for this record type).
  2.    I       Scaled final X position.
  3.    I       Scaled final Y position.
```

### Write character string record

This record tells a device to write a character string starting at the current position. Subroutine GCHAR writes this logical record.

```
FIELD  TYPE     DESCRIPTION
  1.   I        Opcode (equals 6 for this record type).
  2.   I        Number of characters.
  3.   I        Angle code: 0 = write characters horizontally.
                            1 = write characters vertically.
  4.   I        X offset from current position in characters * 100
  5.   I        Y offset from current position in characters * 100
                (net position refers to lower left corner of 1st char)
  6.   I(n)     Hollerith characters (n = INT((field2 + 3) / 4))
```

### Write pixels record

This record tells a raster type device to write an n-tuple of pixel values starting at the current position. Programs that write to line drawing type devices ignore records of this type. This type of record may span disk-record boundaries and is written by subroutine GRAYPX.

```
FIELD  TYPE     DESCRIPTION
  1.   I        Opcode (equals 7 for this record type).
  2.   I        Number of pixel values.
  3.   I        Angle code: 0 = write pixels horizontally.
                            1 = write pixels vertically (up).
  4.   I        X offset in characters * 100.
  5.   I        Y offset in characters * 100.
  6.   I(n)     n (equal to field 2) pixel values.
```

**Write misc. info to image catalog record**

This record tells the programs that write to interactive devices (TKPL, TVPL) to put up to 20 words of miscellaneous information in the image catalog starting at word IITRA + 2. This information is interpreted by routines such as AU9A (TKPOS, TKVAL, etc.) and is used, at the moment, only for slices by task SL2PL. Routines that write to non-interactive graphics devices (PRTPL) ignore this record. Subroutine GMCAT handles this record.

```
FIELD  TYPE      DESCRIPTION
  1.   I         Opcode (equals 8 for this record type).
  2.   I         Number of words of information.
  3.   I(n)      Miscellaneous info (n=value of field 2).
```

**End of plot record**

This record marks the end of a plot file and is written by subroutine GFINIS.

```
FIELD  TYPE     DESCRIPTION
  1.   I      Opcode (equals 32767 for this record type).
```

### 11.2.4 Other Plotting Customs

Over the years, certain plotting options have come to be expected by AIPS users. Among these are (1) recording the plot in the main file's history extension file, (2) offering to plot "star" positions on appropriate plots (i.e., those with the image's X-Y coordinates for axes), (3) offering to draw tick marks all the way across the plot, and (4) offering a variety of axis labeling options. The subroutines mentioned in the discussion below are shown with their precursor remarks at the end of this chapter.

1. HIPLOT adds a line to the main file's history extension file giving the time, date, task name, and plot file version number.
2. STARPL plots "star" positions on plots normally, but not necessarily, with axes which are a celestial-coordinate pair. The positions are entered as an extension file by the user with task STARS. The user then specifies which star extension file with INVERS and whether the positions are plotted and, if so, how large to plot them with adverb STFACTOR.
3. LABINI is an initialize routine which sets up the basic position common and initializes a variety of other standard plotting parameters. Calls CHNTIC. The user specifies the axis labeling type with adverb LTYPE (see the precursor remarks for supported values). See the chapter on Catalogs for details of the position common.
4. COMLAB is a plot labeling routine designed for contour maps. It is a good example of the standard conventions for labeling.
5. CLAB1 labels the X and Y axes and calls CTICS.
6. CLAB2 labels the X and Y axes and calls CTICS. It allows the plotting of the X axis at a specified Y value and vice versa.
7. CTICS draws tick marks on the plot, either as short ticks or as tracks following the coordinate across the full plot area. The user specifies this option with the AIPS adverb DOCIRCLE.
8. CHNTIC counts characters used for labeling Y-axis tick marks. This number is needed for the call to GINITL.
9. CONDRW draws a contour display of an image.

## 11.3 Plot Paraform Tasks

### 11.3.1 Introduction

Three paraform tasks (PFPL1, PFPL2 and PFPL3 - sometimes called PFPLn below) are available in AIPS for developing plot tasks that read a map and create a plot file to be associated with the map. These tasks use the standard AIPS defaults for adverb values such as INNAME, BLC, TRC, XYRATIO, PIXRANGE, etc. The programs are heavily commented and modular.

The three tasks correspond to the three types of plots that can be found in AIPS. The first type is a plot of an X Y plane of the map or a subimage of the map. In this case, the X and Y axes of the plot are the same as the X and Y axes of the map. Examples of this type are produced by tasks CNTR and GREYS. A second type of plot is when the X axis of the plot is a slice of the X and Y axes of the map and the Y axis of the plot is some other value such as intensity. Task SL2PL will create a plot of this type from a slice of a map. The third type of plot is when the axes of the plot have no real relation to the map axes. An example of this type of plot is the histogram produced by task IMEAN.

The structures of all three paraform tasks are very similar. The major differences are in subroutine PLINIn (the subroutine that initializes the commons used in labeling the plot), PLABLn (this routine does the actual labeling), and in the example plots in subroutine PLTORn. The adverbs received from AIPS also differ slightly. The tasks will be discussed individually in a following section, but first we will describe the general structure of all three programs. The tasks perform the following steps:

1. Open an image file corresponding to the users inputs from AIPS.
2. Create an extension file of type PL (plot) to be associated with the image file. The header of the image file will be updated to include this new extension file.
3. Write the plot file records to draw the borders and labels of the plot. The programmer can customize this section of the program by changing data statements and assignment statements in the main program.
4. Write the rest of the plot file records to the plot file. This is done by subroutine PLTORn. The programmer will have to modify the code in PLTORn for his needs.
5. Do the necessary clean up functions, write end of plot records, close all files, etc.

### 11.3.2 Getting Started

The first step is choosing a new name and making copies, using the new name, of the source code file and the help file. One should copy files APGNOT:PFPLn.FOR, and HLPFIL:PFPLn.HLP ("n" stands for 1, 2 or 3) to a user directory and work with the program there.

When a task is renamed, some source code must be changed. The first line of the program after the LOCAL INCLUDE

```
      PROGRAM PFPLn
```

and the data statement

```
      DATA PRGNAM /'PFPLn '/
```

should be changed to use the new name. The name in the HELP file should also be changed. See Appendix A in volume 1 for details of compiling, linking and debugging the task.

### 11.3.3 Labeling the Plot

The labeling of the plot takes place in two subroutines called by subroutine PLTORn. PLINIn will set a number of variables in common that give the labeling routines and the plot drawing routines information about the corners of the plot, the types of the axes, the type of labeling, the size of the plot borders in characters, and other details.

Subroutine PLABLn uses the information provided by PLINIn to actually write the commands in the plot file to draw the labels, borders, and tic marks.

The programmer can customize the labeling somewhat without changing either PLINIn or PLABLn by setting values in an array PCODE, and changing data statements in the main program.

Optional text can be printed at the bottom of a plot by setting values INTEXT (number of lines of text), and ATEXT (an array containing the actual text lines). These values are currently set in DATA statements in the main program. The programmer can choose to set INTEXT to zero to suppress all of the lines. If the programmer wishes to use more than two lines, then the second dimension of arrays TEXT and ATEXT must be changed in all the routines in which they are declared.

See the section on the individual programs for details on setting PCODES.

### 11.3.4 Plotting

Plotting consists of reading the map, collecting the data, and then drawing lines or writing grey scale pixels. All of these steps are usually done in subroutine PLTORn. Reading a map is usually done with routine GETROW (see below). Setting a starting point of a line is usually done with routine PLPOS. Setting the end point of a line is done with PLVEC. Grey scale pixels are written with subroutine PLGRY.

### 11.3.5 Map I/O

This program uses the standard AIPS I/O package grouped into a few subroutines. This approach attempts to make life a little easier by hiding a few of the messy details, but not to eliminate the flexibility of the I/O by hiding it under a complex system. These routines use the "copy mode" approach to I/O in that data is read into a large buffer and then copied from the large I/O buffer to a smaller buffer when a row is needed. This is less efficient than using the bare AIPS I/O routines, but frees the programmer from having to deal with indexes into the large array.

There are four I/O routines in this program, MAKNAM (fills in an array with all the data items that go into specifying a map), INTMIO (initializes the I/O routines to read or write a cataloged map), REIMIO (initializes counters for reading a different subimage or making another pass through a map opened by INTMIO) and GETROW (reads a row of a map). MAKNAM and INTMIO are used in straight forward ways to open the map. The programmer can usually ignore these two routines unless a second map must be opened. If the program must make more than one pass through the data, REIMIO can be used to reset all of the counters. REIMIO assumes that the map is already opened in INTMIO and that a second pass is being made through the data. This routine can NOT be used to read different subimages from the same map at the same time. GETROW must be used (usually in subroutine PLTORn) to read data from the map, one row at a time.

The I/O routines in this program use a common named MAPHDR (include DCAT.INC). This common was chosen to interface with several of the plotting routines which expect this common to contain the map header. Besides the map header, this common contains an array, IMSTUF, which has several data items of interest. IMSTUF(9) is of particular interest since it contains the number of data values (pixels) in each row of the map. This number is usually the upper limit of a loop which operates on each element in the map row. A description of all the elements of IMSTUF are listed in the following table:

```
1.     AIPS I/O Logical unit number
2.     FTAB index
3.     unused
4.     unused
5.     Catalog slot of image.
6.     Size of I/O buffer in bytes.
7.     Disk volume number of image.
8.     Number of dimensions in image.
9.     Number of values read per row of image.
10-16. Number of values along all 7 axes
17-30. Window in BLC TRC pairs along all 7 axes.
31-36. Current position on last six axes.
```

```
      37     1 if read forward -1 if backward read on 2nd axis.
```

Minor modifications in the I/O routines could be made to produce routines for reading UV data, but this has not yet been done.

### 11.3.6 Cleaning Up

Some of the adverbs passed from AIPS may not be used for some types of plots. The programmer can make things easier for the AIPS user by removing them from the help file. The programmer must then remove them from the local include which can be found at the beginning of the file. The variable NPARMS is initialized in an assignment statement in the main program. This must be changed to correspond to the new number of words received from AIPS.

### 11.3.7 The Three Paraform Plot Tasks

**PFPL1**

This task should be used when developing a plotting task in which the X and Y axes of the plot are the same as the X and Y axes of the map.

Much of the labeling is controlled by values of array PCODE. The values for the elements of PCODE are summarized in the following table and are set in the main program between the calls to PFnINI and PLTORn.

```
If PCODES(1) equals

     1   then the plot axes consist of an unlabeled
         rectangular border.

     2   then draw a rectangular border plus
         the title and text at the bottom.

     3   then draw a rectangular border, labels,
         and border tick marks
         indicating absolute coordinates (r.a., decl., etc.).

     4   then draw a rectangular border, labels, and border tick marks
         indicating coordinates relative to the coordinates
         of the image reference pixel (units usually in
         arc seconds).

     5   draw border, labels, and border tick marks
         indicating coordinates relative to the center of
         the subimage plotted (units usually in arc seconds).

     6   draw border, labels, and border tick marks
         indicating image pixel numbers.

If PCODES(2) equals

   0       then label the X axis with the X axis value found in the
           map header.

   other   then label the X axis using variable AXUNIT which is set in
           a DATA statement in the main program.
```

```
If PCODES(3) equals

   0       then label the Y axis with the Y axis value found in the
           map header.

   other   then label the Y axis using variable AYUNIT which is set in
           a DATA statement in the main program.

If PCODES(4) equals

   0       then use the "standard" title consisting of map name,
           source name, and frequency.

   other   then  use  the  title  given  in  DATA  statement for
           variable ATITLE in the main program.

If PCODES(5) equals

   0       then no grey  scale  pixels are to be written for the
           plot.

   other   then grey scale pixels  with a range  given by PIXRNG
           (these values are usually passed from AIPS in adverb
           PIXRANGE) can be written to the plot.  This code value
           causes an "init for grey scale"  record  to be  written
           to the plot file.
```

Usually a task will let the AIPS user choose the value of PCODES(1) by setting adverb LTYPE, e.g., PCODES(1) is set to LTYPE after the task gets this adverb value from AIPS.

When using PLPOS and PLVEC, the positions for this type of plot are given in pixels.

The unmodified version of PFPL1 contains code in PLTOR1 to read the map, and draw a grey scale plot. The user should remove this example found between comment lines "** Plot specific code" and "** End plot specific code" and insert the code for his own application.

### PFPL2

This task should be used when developing a plotting task in which the X axis of the plot is a slice of some plane of the map, and the Y axis is some other value such as intensity. The PCODE usage is described below.

```
PCODES(1) equals

           The label type of the X axis.  The codes are the same
           as for PFPL1.

If PCODES(2) equals

   0       then label the X axis with the units determined by the
           "standard" slice labeling algorithm.

   other   then label the X axis using variable AXUNIT which is set in
           a DATA statement in the main program.
```

```
If PCODES(3) equals


   0       then label the Y axis with the units found in the
           map header for the map intensity.

   other   then label the Y axis using variable AYUNIT which is set in
           a DATA statement in the main program.

If PCODES(4) equals

   0       then use the "standard" title consisting of map name,
           source name, and frequency.

   other   then  use  the  title  given  in  DATA  statement for
           variable ATITLE in the main program.

If PCODES(5) equals

   0       then use the "standard" slice message at the bottom of
           the plot. This message will give the center of the slice.
           This message occurs above the message found in TEXT
           as described above.

   other   then do not print the "standard slice message"
```

The example program in PFPL2 will plot a slice of the X Y plane. The user should remove the example found between comment lines "** Plot specific code" and "** End plot specific code" and insert the code for his own application. This example uses no interpolation (it uses the value of the nearest pixel) and is NOT adequate for a production program. See the code in task SLICE for a good set of interpolation routines and a "rolling buffer" scheme.

**PFPL3**

This task should be used when developing a plotting task in which the X and Y axes have no relation to the map X and Y axes. The plot could be of a function, a histogram of some values, or a table.

The only PCODES values used are PCODES(4) and PCODES(5). If PCODES(4) is 0, then the program plots the "standard" title line. Otherwise, it uses whatever string is in variable ATITLE. If PCODES(5) is not zero, then this signals the existence of grey scale pixels. The program automatically uses whatever strings are in variables AXUNIT and AYUNIT to label the units for X and Y. Thus, the programmer will have to edit the DATA statements for these variables in the main program, or fill them in by some other means.

The example program in the unmodified version of PFPL3 will plot a simple histogram of map intensities. The subroutine PLTOR3 reads the map to determine the histogram values and the range of the Y axis (number of pixels). Then the standard initializing routine (PLINIn) and labeling routine (PLABLn) are called. Finally the histogram is plotted. The programmer must remove the two sections of example code found between two sets of comment lines "** Plot specific code" and "** End plot specific code" and insert the code for his own application.

## 11.4 Plotting to Devices

There are a wide variety of devices which may be used as plotters by AIPS code. The plot files may be interpreted to each of these devices by a device-dependent task. Currently, AIPS supports TV devices with

TVPL (IIS models 70, 75 and IVAS, DeAnza, Comtal Vision 1/20), Tektronix 4010 and 4012 and emulator devices with TKPL, Versatec and similar electrostatic printer/plotters with PRTPL, and QMS Lasergraphix printers with QMSPL. Many other devices could, in principle, be supported, but it is hard for the AIPS group at the NRAO in Charlottesville to create tasks, and to provide reliable support, for devices not physically present at the NRAO in Charlottesville. For this, we depend primarily on our user community. We will be glad to receive any plotter tasks which you may wish to submit and will do what we can to provide support. We are, of course, quite adept at shipping code to the world-wide AIPS user community.

### 11.4.1 Versatec

The Versatec is an electrostatic printer/plotter with 200 dots to the linear inch. It works by drawing a row of dots, advancing a 200th of an inch, and drawing the next row of dots. In other words, it is not a randomly addressable device like a TV or the QMS. Therefore, PRTPL must prepare a file containing the entire bit matrix to be plotted. Then the subroutine ZDOPRT reads the file a row at a time, prepends special device plot codes, and sends them to the device (or the device spooler). Unfortunately, these special codes are specific to the Versatec device driver in use and hence are different for other devices (e.g., Printronix) and can even be different for the same device with a different driver. The standard version of ZDOPRT provided with VMS AIPS is intended to work for spooled Versatecs under VMS with drivers of revision C or greater. The precursor comments for this routine are reproduced at the end of this chapter.

To adapt PRTPL to other, similar devices, you will need to do at least two things. The simpler is to correct the setting of YPRDMM at the beginning of subroutine PRTDRW. This sets the number of dots per millimeter in the Y direction and is, in general, not the same as the X dots per millimeter (as is assumed by PRTPL). You should also set the plotter size and X dots per millimeter using the stand-alone program SETPAR, since these are "global" AIPS parameters carried in the device characteristics common (include DDCH.INC. Second, you must produce a version of ZDOPRT suitable to your device. To assist in this, we provide, in the directory APLVMS:, four other versions of ZDOPRT, called ZDOPRn, which may be of assistance.

### 11.4.2 QMS Laser Printer

The QMS Lasergraphix is a printer/plotter based on a laser print device manufactured by Canon. (Numerous other OEMs are also selling devices based on this engine.) The resolution is 300 dots to the linear inch and there is a full-page, randomly addressable memory. The device is capable of accepting downloaded "fonts", vector commands, and other useful plot commands. AIPS task QMSPL translates AIPS plot files into ASCII print files containing the commands to the QMS. The Z routine ZQMSIO performs the necessary open, write, and close operations. It opens the file in a way which will cause it to be deleted if the program aborts. This is to prevent partial files from clogging the disk and/or being printed and leaving the QMS in a strange state. The close operation redirects the file to "print/delete", or, at the user's request, to "print/keep" in a file named by the user. The precursor comments for ZQMSIO are given at the end of this chapter.

To convert ZQMSIO for use on some other comparable device, two major operations must be performed. First, a ZxxxIO must be written to perform the operations of ZQMSIO. They are likely to be very similar. Second, the command grammar and text of the QMS must be translated to the new device in the QMSPL task (under a new xxxPL name, of course). Where possible, we recommend doing this second operation, rather than creating a wholly new xxxPL. QMSPL contains an excellent algorithm, based on one used by Starlink in Great Britain, for displaying grey scale pixels on dot matrix devices. There are also a number of other scaling algorithms for grey scale pixels in QMSPL, which have proven useful. It would be best to avoid reinventing these wheels. QMSPL represents grey scale pixels in two modes, quick and slow. If there is a significant number of dots per image pixel, then QMSPL can use the quick mode of precomputing a "font" for each of some number of grey levels and then translate the pixels into letters to be printed. A better, but much more expensive, display is produced by computing the dot pattern for each image pixel based on its exact value and an exact implementation of the Starlink algorithm.

In order to understand QMSPL in order to do the translation, it is necessary to have some feel for the QUIC language of QMS. The commands used by QMSPL are summarized below:

```
^PY^-           Go into QUIC command mode
```

```
^IOP             Use portrait mode
^IOL             Use landscape mode
^F^-             Use free format (ignore system-provided
                 carriage returns, line feeds, etc.
^DCnnnn          Make nnnn copies of current page
^ISYNTAX00010    Change measurements to dots, other options
                 remain at defaults
^ISTFX0          Text processing: enable automatic font
                 baselining, don't change auto-justify
^ITnnnn          Set x origin (left page margin) to nnnn dots
^IJnnnn          Set y origin (top page margin) to nnnn dots
^SM00204         Select font 204 in standard vertical,
                 used for labeling
^SM09001         Select "font" 9001, used for grey scales
^DF09001L0AIPSnnn  Define "font" 9001 in landscape (P for
                 portrait) called AIPS version 0 with nnn
                 dots in Y per character
,aannn^Myyyxxx000000 Define character aa (in hex), width nnn
                 dots, using bit map of actual size xxx
                 by yyy dots, with no border dots ---
                 followed by the definition given in 4-bit
                 hex characters padded to integer words
                 on each row.
^ILnnnn          Set character line spacing to nn.nn characters
                 per inch; nnnn = 0 => 6 lines per inch
^ICnnnn          Set character horizontal spacing to nn.nn
                 characters per inch; nnnn = 0 means use
                 proportional, not fixed spacing
^IGV             Turn on vector graphics mode
^IGE             Turn off vector graphics mode
^PWnn            Set pen width to nn dots (nn odd, <= 31)
^Uxxxx:yyyy      Move position to xxxx, yyyy in dots; yyyy is
                 measured down the page
^Dxxxx:yyyy      Draw vector to xxxx, yyyy in dots
^Pnnnn           Start binary transmission, nnnn dots per row;
                 followed by data given in 4-bit hex characters
                 padded to integer multiple of 4 characters
^G               Terminate current command, i.e., stop font
                 definition, binary transmission, etc.
^DF09001L^G      Delete "font" 9001 in landscape mode
^DF09001P^G      Delete "font" 9001 in portrait mode
^ISYNTAX00000    Restore default syntax, measurements in inches
^,               Print this page
^O               Off free format
^PN^-            Exit QUIC mode
```

## 11.5 Includes

### 11.5.1 DLOC.INC

used by all position routines; set by SETLOC.

```
C                                                     Include DLOC.
```

```
C                                      Position labeling common
      DOUBLE PRECISION RPVAL(4), COND2R, AXDENU, GEOMD1, GEOMD2, GEOMD3,
     *   GEOMD4
      CHARACTER CTYP(4)*20, CPREF(2)*5, SAXLAB(2)*20
      REAL      RPLOC(4), AXINC(4), ROT, REPOCH
      INTEGER   ZDEPTH(5), ZAXIS, AXTYP, CORTYP, LABTYP, SGNROT,
     *   AXFUNC(7), KLOCL, KLOCM, KLOCF, KLOCS, KLOCA, KLOCB,
     *   NCHLAB(2)
      COMMON /LOCATC/ CTYP, CPREF, SAXLAB
      COMMON /LOCATI/ RPVAL, COND2R, AXDENU, GEOMD1, GEOMD2, GEOMD3,
     *   GEOMD4, RPLOC, AXINC, ROT, ZDEPTH,
     *   ZAXIS, AXTYP, CORTYP, LABTYP, SGNROT, AXFUNC, KLOCL, KLOCM,
     *   KLOCF, KLOCS, KLOCA, KLOCB, NCHLAB, REPOCH
C                                                         End DLOC.
```

### 11.5.2 DGPH.INC

used by the basic plot subroutines GINIT, GINITL, GPOS, etc.

```
C                                                         Include DGPH.
C                                      Plotting common
      INTEGER   GPHSIZ, GPHLUN, GPHIND, GPHPOS, GPHRRN, GPHVOL,
     *   GPHTLO, GPHTHI
      REAL      GPHX1, GPHX2, GPHY1, GPHY2, SCALEF
      CHARACTER GPHNAM*48, TVSUBR*6
C
      LOGICAL   GPHDOT, GPHPAG
      INTEGER   GPHCRN(2), GPHTVC, GPHTVG, GPHIX0, GPHIX1, GPHIX2,
     *   GPHIXL, GPHIY0, GPHIY1, GPHIY2, GPHIYL, GPHTVW(4), GPHCHL,
     *   GPHCHB, GPHCHR, GPHCHT, GPHNGR, GPHCAT(256), GPHSCR(2048),
     *   GPHERO, GPHERG, GPHERC, GPHERV, GPHPLX(2), GPHPLY(2)
      REAL      GPHSCX, GPHSCY, GPHRX1, GPHRX2, GPHRY1, GPHRY2
C
      COMMON /GPHCOM/ GPHX1, GPHX2, GPHY1, GPHY2, SCALEF, GPHSCX,
     *   GPHSCY, GPHRX1, GPHRX2, GPHRY1, GPHRY2, GPHDOT, GPHPAG,
     *   GPHSIZ, GPHLUN, GPHIND, GPHPOS, GPHRRN, GPHVOL, GPHTLO,
     *   GPHTHI, GPHCRN, GPHTVC, GPHTVG, GPHIX0, GPHIX1, GPHIX2,
     *   GPHIXL, GPHIY0, GPHIY1, GPHIY2, GPHIYL, GPHTVW, GPHCHL,
     *   GPHCHB, GPHCHR, GPHCHT, GPHNGR, GPHCAT, GPHSCR, GPHERO,
     *   GPHERG, GPHERC, GPHERV, GPHPLX, GPHPLY
      COMMON /GPHCHR/ GPHNAM, TVSUBR
C                                                         End DGPH.
```

### 11.5.3 DPLT.INC

used by PFPL1, PFPL2, PFPL3.

```
C                                                         Include DPLT.
C                                      Include for plotting
      DOUBLE PRECISION CATD(128)
      REAL      PBLC(2), PTRC(2), XY, XSCAL, XOFF, YSCAL, YOFF, GFAC,
     *   GOFF, RANGE(2), XLAST, YLAST, CATR(256)
      INTEGER   PLTBLK(256), IOFFPL, IVER, IBLKSZ, IMSTUF(37),
```

```
     *   IOBLK(8192), CATBLK(256), GRCHN, TVCHN, TVCORN(4)
      LOGICAL   DOTV
      HOLLERITH CATH(256)
      COMMON /PLTCOM/ IOBLK, PBLC, PTRC, XY, XSCAL, XOFF, YSCAL,
     *   YOFF, GOFF, GFAC, RANGE, XLAST, YLAST, PLTBLK, IOFFPL, IVER,
     *   IBLKSZ, IMSTUF, GRCHN, TVCHN, TVCORN, DOTV
      COMMON /MAPHDR/ CATBLK
      EQUIVALENCE (CATBLK, CATH, CATR, CATD)
C                                                           End DPLT.
```

## 11.6 Routines

### 11.6.1 CHNTIC

Counts the number of characters to the left of a plot used for labeling the vertical axis.

```
CHNTIC (BLC, TRC, CHMAX)
Inputs:
   BLC      R(2)   X AND Y pixels to form bottom left hand
                   corner of the graph.
   TRC      R(2)   X and Y pixels to form the top right hand
                   corner of the graph.
Output:
   CHMAX    I      Max number of characters.
```

### 11.6.2 CLAB1

Controls some axis drawing and labeling functions: labels each axis with RA/DEC or the type. call CTICS to draw tics and tick labels

```
CLAB1 (BLC, TRC, CH, ILTYPE, XYR, DOGRID, IBUFF, IERR)
Inputs:
   BLC      R(2)    X, Y pixels to form bottom left hand corner
   TRC      R(2)    X, Y pixels to form the top right hand corner
   CH       R(4)    left, bot, right, top : total character offsets
   ILTYPE   I       label type: 1 none, 2 no ticks, 3 RA/DEC
                       4 center relative
   XYR      R       The ratio of the distance between X axis pixels
                    and the distance between Y axis pixels on plot
   DOGRID   L       T => full coord grid, else ticks
In/out:
   IBUFF    I(256)  the updated graphics output buffer.
   IERR     I       error indicator: 0 = No error.
```

### 11.6.3 CLAB2

Controls some axis drawing and labeling functions: labels each axis with RA/DEC or the type. Call CTICS to draw tics and tick labels. Differs from CLAB1 in that the axes are drawn as subplots (fewer ticks and only one axis line in each direction, arguments AYV and AXV are used).

```
CLAB2 (BLC, TRC, CH, ILTYPE, XYR, AYV, AXV, IBUF, IERR)
Inputs:
   BLC      R(2)    X, Y pixels to form bottom left hand corner
   TRC      R(2)    X, Y pixels to form the top right hand corner
```

```
   CH       R(4)    left, bot, right, top : total character offset
   ILTYPE   I       label type: 1 none, 2 no ticks, 3 RA/DEC
                       4 center relative
   XYR      R       Ratio of the distance between X axis pixels and
                    the distance between Y axis pixels on the plot
   AYV      D       Draw the x axis at this Y value
   AXV      D       Draw the y axis at this x value
In/out:
   IBUF     I(256)  the updated graphics output buffer.
   IERR     I       error indicator: 0 = No error.
```

### 11.6.4 COMLAB

Is an axis drawing and labelling routine for use with the common labeling for contour plots and pol vector plots. It calls GINITL and puts subsidiary labels (source, frequency, Stokes, image name), (peak flux), (contour levels) in file.

```
COMLAB (BLC, TRC, LTYPE, IVER, YGAP, CH, XMULT, XLEVS, XYR, IBUFF,
 *   IERR)
Inputs:
   BLC      R(7)   bottom left hand corner of the map.
   TRC      R(7)   top right hand corner of the map.
   LTYPE    I      label type: 1 none, 2 no ticks, 3 Ra/dec
                   4 center relative, 5 subimg center-rel,
                   6 pixels, 7 as 3 no top labels
                   < 0 => no date/time, else as positive
   IVER     I      plot file version number
   XMULT    R      the multiplier for the LEVS to find the
                   actual contour levels.
   XLEVS    R(30)  the contour levels (when used with XMULT)
In/out:
   IBUFF    I(256) the updated graphics output buffer.
   YGAP     R      On input: # lines at bottom to skip before
                   peak flux line in addition to standard
                   On output: includes standard
Output:
   IERR     I      error indicator: 0 = No error.
```

### 11.6.5 CONDRW

Will write commands to a plot file for the execution of a contour plot.

```
CONDRW (IMLUN, IMFIND, IGLUN, IGFIND, XMULT, BLC, TRC, LEVS, IGBLK,
  *   IERR)
Inputs:
   IMLUN   I       logical unit number for the map file.
   IMFIND  I       FTAB index for open map file.
   IGLUN   I       logical unit number of the graph file.
   IGFIND  I       FTAB index for initialized graph file.
   IGBLK   I(256)  I/O block for graph file.
   XMULT   R       Contour interval (image units)
   BLC     R(7)    Bottom left corner
   TRC     R(7)    Top right corner
   LEVS    R(30)   Selected contour intervals in increasing order
```

```
                   (any decrease terminates the list)
Common:
   CATBLK  I(256)  map header.
   CNTRBU  R(8192) buffers
Output:
   IERR    I   error code. 0 = ok.
                            9 => QUIT op received from TELL
                           10 => ABOR op received from TELL
```

### 11.6.6 CTICS

Writes tick marks and tick labels to a plot file. If CPREF(IAX) and CTYP(IAX) are all blank, no tick labels are done.

```
CTICS (LAXIS, BLC, TRC, XYRATO, YX, DOACRS, IBUFF, IERR)
Inputs:
   LAXIS   I        1 => horizontal,  2 => vertical full plots
                    3 => horiz subplot 4 => vertical subplot
   BLC     R(2)     X and Y pixels to form bottom left hand
                    corner of the graph.
   TRC     R(2)     X and Y pixels to form the top right hand
                    corner of the graph.
   XYRATO  R        X to Y scaling factor
   YX      D        LAXIS=3: plot x axis at y = YX; 4: plot y axis
                    at x = YX - out range => BLC(1,2) value
   DOACRS  L        Do coordinate grid (T) or just ticks (F)
In/out:
   IBUFF   I(256)   buffer being used for output to
                    the graphics file.
Outputs:
   IERR    I        error code: 0 => ok
                                1 => bad IAXIS
                                2 => graph drawing error
                                3 => tic algorithm fails
```

### 11.6.7 GCHAR

Will write a 'draw string' command record to a graph file. The output record description is:

```
I     opcode, 6 in this program.
I     number of characters (NCHAR).
I     angle (IANGL).
I     X offset in characters * 100  (DX * 100).
I     Y offset in characters * 100  (DY * 100).
I(*)  characters (STR).

GCHAR (NCHAR, IANGL, DX, DY, STR, BUFF, IERR)
Inputs:
   NCHAR  I       number of characters in STR
   IANGL  I       angle to print STR. 0=horizontal, 1=vertical.
   DX     R       x offset in characters from current position for
                  the bottom left corner of first character printed.
```

```
     DY     R       y offset in characters from current position for
                    the bottom left corner of first character printed.
     STR    C*(*)   string to be printed.
     BUFF   I(256)  buffer to use for I/O.
  Output:
     IERR   I       error code:  0 => ok
                                 1 => disk problems
                                 2 => string too big
  Common:
     GPHPOS  incremented by 5 + (NCHAR+3)/4.
     GPHRRN  incremented by 1 if a write to disk is needed.
```

### 11.6.8 GETROW

This routine will read a row of an image file that has been opened with and initialized with INTMIO. The routine will copy the row from the I/O buffer to the user buffer.

```
  GETROW (IMSTUF, IOBLK, ROW, EOF, IERR)
 Inputs:
    IMSTUF    I(37)   IO pointers, LUNs, counters and such. They are
                      set in INTMIO.
 In/out:
    IOBLK     R(*)   IO buffer.
 Outputs:
    ROW       R(*)   Output row of image.
    EOF       L      TRUE means last row specified in INTMIO by the
                     BLC, TRC arguments has been read.
    IERR      I      Error code, 0=ok, others from MDISK.
```

### 11.6.9 GFINIS

Places an "end of plot" command in the buffer, writes the last buffer to disk, compresses the plot file if needed, and closes the plot file.
***** NOTE: any catalog operations for the plot file must be performed by the calling program. *******
The plot common is reinitialized in part. The command record has the form:

```
        I     opcode (32767)

  GFINIS (BUFF, IERR)
  In/out:
     BUFF  I(256)     plot file work buffer
  Output:
     IERR  I          error code: 0 => ok
                                  1 => disk error
                                  2 => data error
                                  3 => compress error
                                  4 => close error
```

### 11.6.10 GINIT

Initializes the graphics common, creates and opens the graphics file, and places the "initialize plot" command in the file.
******* NOTE: cataloging of the graphics file must be handled by the calling program. *********

WARNING: Get PNAME right! If the create fails because the file already exists, GINIT will destroy PNAME and create the desired plot file. GINIT writes 1 or more records to disk containing the task name and parameters. It starts on the next physical record with a command record of the form:

```
I       opcode (1)
I       user number
I(3)    date:  19yy, mm, dd
I       ITYPE: type code 1 => misc.,  2 => CNTR,   3 => GREYS,
            4 => PROFL,  5 => SL2PL,  6 => PCNTR,  7 => IMEAN,
            8 => UVPLT,  9 => GNPLT, 10 => VBPLT, 11 => PFPLn,
           12 => GAPLT, 13 => PLCUB, 14 => IMVIM, 15 => TAPLT,
           16 => POSSM, 17 => SNPLT, 18 => KNTR,  19 => UVHGM,
           20 => ISPEC

GINIT (IVOL, ICNO, PNAME, IGSIZE, ITYPE, NPARM, PARMS, DOTV, TVGREY,
   TVGRPH, TVCORN, CATBLK, BUFF, LUN, FIND, IERR)
Inputs:
   IVOL    I         disk volume for file creation
   ICNO    I         catalog number
   PNAME   C*48      physical file name for created file
   IGSIZE  I         0 => small 1 => medium 2 => large file
   ITYPE   I         plot type code (if < 0, then ITYPE is taken to
                     be #rows (# cols if writing vertically) and is
                     used to adjust SCALEF to integer*abs(itype)
                     The output ITYPE is set to 1.
   NPARM   I         #  values in input parameter list.
   PARMS   R(NPARM)  task input parameter list (with defaults
                     filled in)
   DOTV    L         Selects use of TV (True) or plot file (False)
   TVGREY  I         Grey channel to use (0 => 1)
   TVGRPH  I         TV Graphics channel to use (0 => 1)
   TVCORN  I(2)      X,Y TV pixel to put plot origin (0 => center)
   CATBLK  I(256)    Image header
In/out:
   BUFF    I(256)    plot commands are placed in this buffer and
                     written to disk as needed.  The calling program
                     must not alter this area between calls to GINIT
                     and GFINIS
Outputs:
   LUN     I         logical unit number of plot file
   FIND    I         location in FTAB for file
   IERR    I         error code: 0 => ok
                        1 => another plot file open
                        2 => input data error
                        3 => create error (no file)
                        4 => open error (file gone)
```

### 11.6.11 GINITG

Will write an "init for gray-scale" command record to the graphics file. The output record has the form:

```
I     opcode (3)
I     lowest pixel value
```

```
      I     highest pixel value
      I     number of pixels on x axis
      I     number of pixels on y axis

GINITG (IGLO, IGHI, BUFF, IERR)
Inputs:
   IGLO  I         lowest allowed pixel value
   IGHI  I         highest allowed pixel value
In/Out:
   BUFF  I(256)    graphics buffer
Output:
   IERR  I         error code: 0 => ok
                               1 => disk error
                               2 => input error
```

### 11.6.12 GINITL

Will write an 'init for line drawing' command record to a graph file. common variables will also be initialized. The output record description is:

```
      I       opcode, 2 in this program.
      I       XYRATO * 100
      I       scale factor used in calculating I   X and Y in
              GPOS and GVEC. Equal to SCALEF.
      I(4)    BLC,TRC integer part (BLC rounded up, TRC down)
      I(4)    floating remainder * 1000 from BLC, TRC
      I(4)    10 * (CHOUT) number of characters
              outside pix area on left, bot, right, top, resp.
      I(5)    array location on axes 3,4,5,6,7

GINITL (BLC, TRC, XYRATO, CHOUT, DEPTH, BUFF, IERR)
Inputs:
   BLC     R(2)    number of first pixel in row & col
   TRC     R(2)    last pixel pos in row/col (usually 1 more
                   than the number of pixels)
   XYRATO  R       ratio between x and y axis plotting
   CHOUT   R(4)    number of characters outside pix on left,
                   bottom, right, top, resp.
   DEPTH   I(5)    array location on axes 3 thru 7
In/Out:
   BUFF    I(256)  buffer to use for I/O.
Output:
   IERR    I       error code. 0 = ok.
                               1 => disk problems
Common:
   GPHPOS  incremented by 20.
   GPHRRN  incremented by 1 if a write to disk is needed.
   GPHX1   set to X1 = BLC(1) (rounded up to integer)
   GPHX2   set to X2 = TRC(1) (rounded down to integer)
   GPHY1   set to Y1 = BLC(2) (rounded up to integer)
   GPHY2   set to Y2 = TRC(2) (rounded down to integer)
```

### 11.6.13 GMCAT

This routine will write a 'copy misc info into the image catalog' command record to a graph file. The output record description is:

```
         I        opcode, 8 for this record type.
         I        number of words.
         I(INO)   miscellaneous information.

   GMCAT (INO, IBLK, IPBLK, IERR)
   Inputs:
      INO     I       number of words of misc info.
      IBLK    I(INO)  array containing misc info.
      IPBLK   I       plot file I/O block.
   Output:
      IERR    I       error code 0=ok, 1=no. of words over 20.
   Common:
      GPHPOS  I       incremented by 2 + INO.
      GPHRRN  I       incremented by 1 if write to disk necessary.
```

### 11.6.14 GPOS

Will write a 'position vector' command record to a graph file. The output record description is:

```
         I     opcode, 4 in this program.
         I     scaled X position.
         I     scaled Y position.

   GPOS (X, Y, BUFF, IERR)
   Inputs:
      X       R       x position
      Y       R       y position.
      BUFF    I(256)  buffer to use for I/O.
   Output:
      IERR    I       error code. 0 = ok.
                                  1 = disk problems.
   Common:
      GPHPOS  incremented by 3.
      GPHRRN  incremented by 1 if a write to disk is needed.
```

### 11.6.15 GRAYPX

Places a "write string of gray values" command in the graphics file. The gray values are clipped using range given in the call to GINITG and placed in graphics file.
NOTE: this command may extend over more than 1 physical record in the file! The command form is:

```
         I          opcode (7)
         I          NPIX: number of values
         I          IANGL; 0 horizontal, 1 vertical
         I(NPIX)    clipped gray values

   GRAYPX (NPIX, IANGL, IVALS, BUFF, IERR)
   Inputs:
      NPIX   I          number of pixel values
```

```
   IANGL  I          direction code: 0 horizontal, 1 vertical
   IVALS  I(NPIX)    pixel values
In/out:
   BUFF   I(256)     graphics working buffer
Output:
   IERR   I          error code:  0 => ok
                                  1 => disk error
                                  2 => input data error
```

### 11.6.16 GVEC

Will write a 'write vector' command record to a graph file. The output record description is:

```
      I     opcode, 5 in this program.
      I     scaled X position.
      I     scaled Y position.

GVEC (X, Y, BUFF, IERR)
Inputs:
   X      R       x position
   Y      R       y position.
   BUFF   I(256)  buffer to use for I/O.
Output:
   IERR   I       error code. 0 = ok.
                              1 = disk problems.
Common:
   GPHPOS  incremented by 3.
   GPHRRN  incremented by 1 if a write to disk is needed.
```

### 11.6.17 HIPLOT

Places one record in a file's HI extension file reporting that a plot extension was created by the present program.

```
HIPLOT (IVOL, ICNO, IVER, IBUF, IERR)
Inputs:
   IVOL   I         Disk volume
   ICNO   I         Catalog number
   IVER   I         Plot file version number created
Output:
   IBUF   I(256)    Scratch
   IERR   I         Error code: 0 => ok - don't quit on error
                            1 => no HI file
                            2,3,4 => open, add, close error
Uses LUN=29 and inits the HI common
```

### 11.6.18 INTMIO

This routine will open a map file, set values in common for use with close down routine DIE and set up two arrays containing all the values and counters needed by reading and writing routines compatible with this one.

```
INTMIO (ILUN, ACCESS, NAME, BLC, TRC, IBSIZE, CATBLK, IMSTUF, IERR)
Inputs:
   ILUN    I      Logical unit number to use for the map file.
   ACCESS  C*4    'READ' or 'WRITE' status to mark catalog.
                  also 'HDWR' - mark write, open non-exclusive
   IBSIZE  I      Size of IO buffer in INTEGER values.
In/Out: (defaults filled in)
   NAME    C*36   "Name string" in the tradition of WaWa IO.
   BLC     R(7)   Bottom left corner of map.
   TRC     R(7)   Top right corner of map
Outputs:
   COMMON /CFILES/  Values updated so that subroutine DIE will
                    close this file.
   CATBLK  I(256)   Map header.
   IMSTUF  I(37)    IO pointers and stuff that are needed by other
                    IO routines compatible with this one. They are:
                   1. LUN
                   2. FTAB index
                   3.
                   4.
                   5. Catalog slot of image.
                   6. Size of IO buffer in bytes of all things.
                   7. Volume number of image.
                   8. Number of dimensions in image.
                   9. Number of values read per row of image.
                   10-16. Number of values along all 7 axis
                   17-30. Window in BLC TRC pairs along all 7 axis.
                   31-36. Current position on last six axis.
                   37 1 if read fwd -1 is bckwrd read on 2nd axis.
   IERR    I      Error code. 0=ok.
```

### 11.6.19 LABINI

Performs initializations for labeling: calls SETLOC, converts units (w METSCA) to get reasonable scaling, and, for LTYPEC = 4 (center-relative labeling), converts units ("lies") and prepares text giving true center position.

```
LABINI (BLC, TRC, IDEP, CH, ILTYPE, SLICE, YGAP, TEXT, NTEXT)
Inputs:
   BLC    R(2)     Bottom left corner
   TRC    R(2)     Top right corner
   IDEP   I(5)     Depth on axes 3 - 7
   ILTYPE I        Labeling type: 1 none, 2 no ticks, 3 RA/dec,
                   4 center-relative, 5 subim center-rel, 6 pixels,
                   7 like 3, but no top line
   SLICE  L        T => for slice display: rotation will be the
                   slice PA - any map rotation; F => for maps: the
                   rotation is any map rot.
In/out:
   CH     R(4)     # chars outside plot to left, bot, right, top:
                   on in # extra for plot, on out totals
                   These must have input, reasonable values, LABINI
                   just adds a 0.5 border and room for the labeling.
```

```
                    IF CH(1) < -10., do not count axis label
                    characters with CHNTIC and CH(1) has little
                    meaning on output.
Output:
   YGAP   R         Char row position for 1st line of text at bottom
   TEXT   C(2)*80   Text to put at YGAP, used only if LTYPE=4
   NTEXT  I         Number of lines used in TEXT: 0, 1, 2
```

### 11.6.20 PLEND

Do some plotting cleanup functions. Write "end of plot" record, close plot file, check for vectors that were off the plot. Then terminates the task with a call to DIE.

```
PLEND (ISTAT, IDEBUG)
Inputs:
  ISTAT  I          0=successful completion, other=dies unnaturally.
  IDEBUG I          > 0 => don't delete PL file despite errors
```

### 11.6.21 PLGRY

This routine will put draw grey scale commands in the plot file.

```
PLGRY (IANGLE, NVAL, VALUES, IERR)
Inputs:
   IANGLE  I      Angle code. 0 = horizontal, 1 = vertical.
   NVAL    I      The number of grey scale pixel values.
   VALUES  R(*)   Grey scale values.
Output:
   IERR    I      Error code. 0=ok.
```

### 11.6.22 PLMAKE

This routine will create and open a plot file, put it in the map header and write the first record into the plot file. PLMAKE assumes that the map/uv file has been marked write and will change it to a read flag. It would be nice if the defaults had been filled into RPARM before this routine is called.

```
PLMAKE (NP, RPARM, IGTYPE, IERR)
Inputs:
   NP     I      Number of words in parameter list
   RPARM  R(NP)  AIPS parameters
   IGTYPE I      Plot file type: 1 misc., 2 CNTR, 3 GREYS, 4 PROFL,
                 5 SL2PL, 6 PCNTR, 7 IMEAN (hist), 8 UVPLT,
                 9 GNPLT, 10 VBPLT, 11 PFPL1, PFPL2, PFPL3.
                 Use 1 unless your inputs match those of these
                 tasks - or take a new number, but
                 AIPSUB:AU8A will need to know about it too.
Output:
   IERR   I      Error code. two digit, first digit indicates
                 subroutine: 1: MAPOPN, 2: MADDEX, 3: ZPHFIL,
                 4: GINIT, second digit indicates error code of
                 that subroutine.
```

### 11.6.23 PLPOS

This routine will put a 'position vector' command in a plot file.

```
PLPOS (X, Y, IERR)
Inputs:
   X       R    X value.
   Y       R    Y value.
   COMMON /PLTCOM/ (DPLT.INC)
Output:
   IERR    I    Error code.  0 means OK.
```

### 11.6.24 PLVEC

This routine will put a 'draw vector' command in a plot file. Vectors that are too big are interpolated.

```
PLVEC (X, Y, IERR)
Inputs:
   X       R    X value.
   Y       R    Y value.
Common:
   /PLTCOM/ (DPLT.INC)
Output:
   IERR    I    Error code.  0 means OK.
```

### 11.6.25 REIMIO

This routine will reinitialize the counters in IMSTUF for reading another subimage of a map opened and set up with INTMIO. All IMSTUF values that can be found in the header are re-initialized even if they are not changed by the standard routines.

```
REIMIO (BLC, TRC, IBSIZE, CATBLK, IMSTUF, IERR)
Inputs:
   BLC      R(7)    Bottom left corner of map.
   TRC      R(7)    Top right corner of map
   IBSIZE   I       Size of IO buffer in words.
   CATBLK   I(256)  Map header.
   IMSTUF   I(*)    (1) LUN
                    (2) FTAB index
                    (5) Catalog slot of image.
                    (6) Size of IO buffer in bytes of all things.
                    (7) Volume number of image.
Outputs:
   IMSTUF   I(*)    (8) Number of dimensions in image.
                    (9) Number of values read per row of image.
                    (10-16)  Number of values along all 7 axes
                    (17-30)  Window in BLC TRC pairs along all 7
                             axes.
                    (31-36)  Current position on last six axis.
                    (37) 1 if read forward, -1 if backward read on
                         2nd axis.
   IERR     I       Error code. 0=ok.
```

### 11.6.26 STARPL

Plots star positions in a plot file as given by an ST extension file of version VERS. The ST file contains the center position (RA-DEC, GLON-GLAT, ELON-ELAT) of each star and the "uncertainties" in those star positions. The plotted plus signs are scaled by these uncertainties and then further scaled by multiplying by FACTOR.

```
STARPL (FACTOR, IVOL, CNO, VERS, BLC, TRC, PLBUF, IERR)
Inputs:
   FACTOR  R         Star scaling factor: <= 0 => no plot.
   VERS    I         Desired ST file version number: 0 => high
   IVOL    I         File disk number
   CNO     I         File catalog number
   BLC     R(2)      Plot lower left corner (pixels)
   TRC     R(2)      Plot upper right corner (pixels)
In/Out:
   PLBUF   I(256)    Plot IO buffer
Output:
   IERR    I         Error code: 0 => okay
                             -1 => there was no ST file
                             +1 => logical error in ST file
                             +2 => IO error in ST file
                             +3 => IO error in plotting
Common:
   /MAPHDR/ CATBLK input   Image header having the ST file
```

THIS PAGE DELIBERATELY BLANK