JET PROPULSION LABORATORY

INTEROFFICE MEMORANDUM

314.8–631

17 December 1986

TO: Chuck Acton/Steve Synnott

FROM: J. D. Callahan JDC

SUBJECT: *Mip* Update

The last major release of *Mip*, Multimission Interactive Picture Planning Program, was the 5/13/86 version (ref. 3). Since that time, the major development effort has consisted of modifications to support Space Telescope. A major demonstration was given to Space Telescope personnel from Goddard Space Flight Center and the Space Telescope Science Institute on July 16, 1986. The demonstration went quite well and there was considerable interest in *Mip*. Many of the improvements for Space Telescope will benefit all subsequent versions of the program. For example, landmarks on highly ellipsoidal bodies are no longer distorted from their true positions, as in earlier versions of *Mip*.

Since that time, a batch processor for Space Telescope, called *Percy*, has also been written by Bill Taber and Ian Underwood. It was delivered to ST the middle of October, while we were still waiting for our Microvax II workstation (with Peritek graphics board) to arrive at JPL. At the same time, development of *Mip* for ST has continued, but at a slower pace. In addition, low-level support of other *Mip* users at JPL has continued. And, as you know, I have also been working on some non-*Mip*-related activities.

Since the Microvax II workstation is now complete, *Mip* has been finished (version 12/4/86). It is fully compatible with *Percy* and has the capability to read and write the required SOGS files. The program is still highly portable, versatile, and interactive (typically generating 5 frames/sec) and may be transferred to almost any computer and graphics system within a day. This memo contains the user's guide, specifically for ST. The program will also handle deep-space missions as before, but the new Naif ephemeris file system is not fully developed yet for this to occur.

Exactly what role the program will play in future support of ST is unclear. This will depend on the experiences and desires of Space Telescope users and management, and on those involved at JPL, but the program is available for use, and to provide ideas for the future.

Distribution:

K. Baines	J. Berthias	J. Boyer	P. Breckheimer
P. Chodas	D. Coons	E. Danielson	R. Diehl
J. Dunne	T. Duxbury	C. Fuechsel	J. Hester
R. Holzman	J. Issacs	W. James	G. Johnston
F. Jordan	K. Klassen	L. Lane	J. Longuski
T. Martin	J. McDanell	W. McLaughlin	R. Mitchell
M. Morrison	G. Null	A. Ocampo	W. O'Neil
B. Paczkowski	J. Reimer	D. Rodgers	J. Roeder
J. Sepikas	D. Skillman	W. Smythe	J. Trauger
B. Waggoner	M. Wang	H. Weaver	R. Weidner
P. Weissman	S. Winterhalter	C. Yeates	

Optical Systems Analysis Group

SSOPS
(Solar System Observation Planning System)

MIP
(Multimission Interactive Planner)

A User's Guide

December 4, 1986

John D. Callahan
JPL

Navigation Ancillary Information Facility

NAIF

NASA

I. Introduction

Mip is an interactive graphics planning tool for Space Telescope, and deep-space missions. It simulates the positions and motions of stars and solar system bodies, and can typically generate several frames per second. The program is approximately 8,000 lines of FORTRAN 77 code, and has been developed at JPL over the last 5 years by John D. Callahan. During the last year, the major development effort has consisted of modifications to support Space Telescope. *Mip* runs off the same standard Naif files as the batch program *Percy*, which is also an ST planning tool. Unlike *Percy*, however, *Mip* runs off a special star file designed to increase speed of computation. *Mip* can run independently of any batch program, plan Space Telescope observations, and write the required SOGS critical window and moving target ephemeris files. In addition, *Mip* may also read the SOGS files it generates, or that of any other program. In other words *Mip* can be used to check and "fine tune" the results of other Space Telescope planning programs.

Mip runs on a Microvax II workstation with a Peritek graphics board. The Peritek board was chosen to maximize vector draw-speed and minimize cost. Although *Mip* runs under this configuration, the program, along with the graphics software, is still highly transportable. It has been configured and run on many different systems.

For background on the program see references 1 through 4.

II. Control Files

Below is an explanation of the control file, which is input to *Mip* at time of execution. The control file contains all the information and files *Mip* needs in order to run. An example control file is given below, followed by an explanation of each item. The control file is for Jupiter (it's name is *jup.ctr*).

Control File (jup.ctr):

```
'stf',5,.01,20.,.02,'star90.cat','brtstr.cat',
'PERCY$DATA:PERCY.GEF','PERCY$DATA:percy.sto',
'milk.dat','lm.dat'/
```

Explanation:

'stf' : Spacecraft – possible values are 'stf' and 'st'. If 'stf' is used then a SOGS-ST ephemeris file is used for the spacecraft (in this case PERCY$DATA:percy.sto). If 'st' is used then one may input orbital elements for the spacecraft via a small ASCII file – an example is given below (the file is sto.dat):

```
********** ST orbit **********
6878.              *** a (km)
.01                *** e
84,08,21,00,00,00  *** t (ut)
2.                 *** cap omega (deg)
9.                 *** omega (deg)
25.                *** i (deg)
```

The start time of the spacecraft file is always used as the start time of *Mip* at execution.

5 : Central body number: 1-9, planets; 10-14 asteroids; 15+ comets (see milk.dat below)

.01 : Field of view of the narrow angle camera in degrees. If this number is below .010314 then a set of about 10 ST instruments is drawn. Otherwise, the value input is used to draw a square box.

20. : 3σ pointing error on the narrow angle camera (degrees). A box will only be drawn if the value above is greater than .010314. This option has its origin in deep space missions, such as Voyager.

.02 : Field of view of the wide angle camera, if present (degrees). A box will be drawn only if the first field value above is greater than .010314.

'star90.cat' : The primary star catalog (SAO), used for fields below approximately 3°. The file is direct access and contains 2592 records each covering $5° \times 5°$ of sky. R.A. and DEC of the stars are stored as 4-byte integers, providing 9 places of precision. The '90' in the name indicates that the stars are at epoch 1990 (B1950 coordinates). The file can be remade at any date with program *sfm* (star file make), which reads and converts a much more extensive and non-optimized star catalog (strful.cat). The SAO catalog contains approximately 250,000 stars down to 10th magnitude.

'brtstr.cat' : A star catalog of bright stars only, made from the main star catalog above with program *bsfm* (bright star file make).

'PERCY$DATA:PERCY.GEF' : All ephemerides except the ST spacecraft. See *Percy* documentation.

'PERCY$DATA:percy.sto' : Spacecraft ephemeris. See *Percy* documentation. This file could also be an ASCII file of orbital elements as described above (sto.dat).

'milk.dat' : A text data file containing constants for the solar system. There are 9 or more records for the 9 planets and any additional asteroids (records 10-14) and comets (records 15+). Each planet can have up to 15 rings and 20 moons; comets have an additional sub-record. Also, for comets, a nucleus is allowed by specifying one moon. The coma and tail of a comet roughly follow the behavior of Halley's Comet. A record consists of (units are deg, sec, km unless noted):

> *R.A. of pole, DEC of pole, large, intermediate, and small axes of planet ellipsoid (a, b, c), period of rotation of planet, number of rings, ring radii in units of a, number of moons, a, b, c of moons, periods of moons/*
> *radius of end of tail at perihelion, distance of tail at perihelion, UTC calendar date of perihelion/ [comet only]*
>
> The periods of the moons (shown above) are only used to orient the prime meridian if there is no IAU standard to correctly orient the pole and prime meridian (see theory section).

'lm.dat' : A text data file containing landmarks. A typical record looks like:
501,123,45.1,'w',123.1,20.1/
where:

> *501 : Body (Io in this case)*
> *123 : Number name of landmark (up to 999)*
> *45.1 : Latitude in degrees*
> *'w' : Signifies that longitude is to be measured plus west*
> *123.1 : Longitude in degrees*
> *20.1 : Radius of feature in km*

III. Menus

When *Mip* is executed, it goes into a beginner mode of operation. The user may immediately go to advanced mode by hitting key '0', or if he is just learning he may remain in beginner mode. In beginner mode, *Mip* is controlled through a series of menus. There is one main menu and 9 sub-menus. For each menu a certain part of the keyboard becomes operational and controls the program. The current menu is always displayed on the user's terminal, and the user may go to advanced mode at any time and then return to beginner mode. In advanced mode, the entire keyboard becomes operational, and, for the most part, help is only given when requested by the user. In both beginner and advanced modes, simple instructions are often given when a key is hit. The various *Mip* beginner menus are given and explained on the following pages.

Much of the time it is only necessary to hit a particular key to control *Mip* – no carriage return is required. However, when inputting some types of data (a calendar date for example), it is necessary to enter a carriage return also. As the user becomes more familiar with the program, this will become easy, but at first he should note the methods by which data is being requested and accepted by *Mip*. As mentioned above, the program will often give short instructions in an attempt to help the user, and it will also usually not "bomb" when a mistake is made. However, no program can be completely "idiot-proof", and the user should be patient when learning.

III.0 Main Menu

To execute *Mip*, simply give the VMS commands $ set term/notype_ahead, and then $ run mip. At JPL, the program resides on the Naif "Chico Vax" and the "MOSS 2 Microvax II" in user$disk:[jdc.mip]. The old 5/13/86 version of *Mip* resides in user$disk:[jdc.mip.mas]. Available control files are usually listed in *ctr.mip*.

Important Note: *The keyboard must be in lower case for the program to operate properly.*

When *Mip* is executed, it will ask for a control file, pause, draw a frame on the display device, and then print the main menu given below at the user's terminal. The menu is fairly self-explanatory. It is not necessary to enter a carriage return after hitting one of the keys below: the corresponding menu will come up immediately, or *Mip* will go into advanced mode (key '0'). If a key other than those listed below is hit, the main menu will be redisplayed.

On the following pages are given the various sub-menus.

```
  Mip functions are summarized below. To display a menu for the subject area
listed, hit the key given. The keyboard should be in lower case; however, if
a key is given in upper case it must be typed that way. The multi command key,
%, (allows you to stack commands) is not available in beginner mode.
  It is best to start with keys 9 or 4, when learning.
  Initial default pointing is at the central body.

  1 - Time, trace
  2 - Pointing
  3 - Observer
  4 - Field of view
  5 - Sensitivity, output
  6 - Observation files
  7 - Scene complexity
  8 - Return, stop program
  9 - Overview, display, keyboard summary
  0 - Go to advanced usage mode
```

III.1 Time

The time menu is shown below. Note that most of the keys will run time. That is, the corresponding operation will be performed, time will be advanced, and a new scene will be displayed on the device. The increment keys (i,o,p,[,k,l,;,') will accelerate or decelerate time when held down.

Hitting the ',' key once will show the current time increment. If this key is hit again before any other key, then the user may input any time increment, or carriage return (to retain the current one).

```
  The time and trace keys are listed below. To perform the
function, simply hit the key.

  u - Zero the time increment
  j - Run mip with current time increment
  i - Increase the time increment by 1 second, and run
  o - Increase the time increment by 10 seconds, and run
  p - Increase the time increment by 100 seconds, and run
  [ - Increase the time increment by 1000 seconds, and run
  k - Decrease the time increment by 1 second, and run
  l - Decrease the time increment by 10 seconds, and run
  ; - Decrease the time increment by 100 seconds, and run
  ' - Decrease the time increment by 1000 seconds, and run
  . - Input a calendar date UT
  , - See time increment, and optionally input any time increment, & run
  m - Change the sign of the current time increment, and run
  s - Toggles erase or no erase of previous scenes (makes a trace), & run
  ) - Return to main menu
```

III.2 Pointing

The pointing menu is shown below. This menu is more challenging to learn than the previous two, so patiently experiment with all the keys when learning. It is important to note that *Mip* will automatically adjust the pointing when stars are present and the field of view is exactly two times the first field value in the control file (see control file section). When *Mip* is executed it will go into this mode, or it can be set with key 'x' of the field of view menu, which will be given on one of the following pages. This option owes its origin to deep-space optical navigation. For optical navigation, it is important to capture stars with the target body. Therefore if stars are present, and the field is set as described above, pointing will be adjusted in an attempt to capture the stars with the target body. Also, the probabilities of capturing 1, 2 and 3 stars with the target body will be displayed at upper right.

```
  The pointing keys are listed below. To perform the function, simply
hit the key. WARNING: if the field of view is set at exactly 2X
the camera field (default, or set with key x) then stars will affect
the pointing. You may want to change the field of view first (return
to main menu and then go to the field of view menu).

  t - Toggles through the central body and any natural satellites
  = - Hit this key and then the number of the satellite you wish to point at.
      To point at the central body hit any other key
  / - Input an exact right ascension and declination
  d - Offset the pointing left
  f - Offset the pointing right
  r - Offset the pointing up
  e - Offset the pointing down
  h - Zero pointing offsets
  + - Fix or unfix the absoulute right ascension and declination pointing
  " - Toggle whether right ascension and declination (RA DC), or clock and cone
      (CK CN) for Galileo, or azimuth and elevation (AZ EL) for Voyager, or sun
      & moon for ST are displayed. For Galileo, the cone angle is for the Earth
  ) - Return to main menu
```

III.3 Observer

The observer menu is shown below. When *Mip* is executed, the initial scene is that seen from the spacecraft. The user then has the option, at any time, of observing from the center of the Earth (key '5'), or from an arbitrary point in space centered on the central body of the system being observed (key '6'). Read the menu below carefully and note that the field orientation for each case is explained in the field of view menu, which is the next menu to be given in this document.

One other important point should be mentioned before moving on. A light-time correction is, of course, made when viewing from the spacecraft or the center of the Earth. However, this correction *is also maintained* when viewing from any point in space – no matter what the distance to the body is. This was done so that the user can see the geometry of the system at the same time as that from the Earth, but at different perspectives.

```
The observer keys are listed below. To perform the function, simply
hit the key. For an explanation of how the field of view is oriented
for each case, return to the main menu and go to the field of view menu.

  g - Observe from the spacecraft (default)
  5 - Observe from the center of the Earth
  6 - Observe from an any point in space, centered on the central body.
      Once key 6 is hit the keys listed below can be used:
        9 - Observe from over the pole of the central body (default)
        0 - Observe from the equator of the central body
        7 - Increase the distance to the central body
        8 - Decrease the distance to the central body
        1 - Observe from a smaller central body longitude
        2 - Observe from a greater central body longitude
        3 - Observe from a higher central body latitude
        4 - Observe from a lower central body latitude
  ) - Return to main menu
```

III.4 Field of View

The field of view menu is given below. It is fairly self-explanatory. However, the following points should be made. Hitting key '!' once will show the current field of view size in degrees, at the user's terminal. The user may then continue, or he may hit key '!' again. If the latter is done, *Mip* will ask the user to input a new field size followed by a carriage return. Hitting carriage return alone will retain the current field size. When using keys '3', '4', and '2', a rotation angle will be displayed at the user's terminal. This angle is with respect to the current nominal field orientation – see key 'N' below.

Hitting key 'I' will display the names of the major ST instruments. The user may then center on any one of these instruments by hitting a corresponding number given with the instrument and then a carriage return. This will be very clear when the user actually trys it. For now the set of ST instruments displayed is "canned" in the program. However, in the future the user will be able to specify – by way of an ASCII data file – any set of instruments.

One last point should be made. The sensitivity of keys 'c', 'v', '3', and '4' can be adjusted with key 'n' of the next menu – the sensitivity and output menu.

```
  The field of view keys (field size and rotation) are listed below.
To perform the function, simply hit the key. Default field orientation
for deep-space spacecraft (Voyager, Galileo, etc.) is the actual one, and for
ST, it is sun down. From the Earth given a deep space mission it is somewhat
arbitrary. But from the Earth given ST it is essentially the same as ST.
For views from any point in space, the orientation is somewhat arbitrary.
  Read the explanation for keys x, and 3 carefully.

  c - Zoom the field in
  v - Zoom the field out
  ! - Print out the current field size (degrees), and optionally
      input any field size
  x - Toggle through a set of exact field sizes. For 2X the camera field
      (default when mip is executed), pointing is adjusted to improve star
      capture probabilities and these are displayed at the upper right
  3 - Rotate the field one way. This key does not work when viewing from
      any point in space (key 6)
  4 - Same as key 3 above, but rotate in the other direction
  2 - Zero the effects of keys 3 and 4 above
  N - Toggle whether N. Celestial is up rather than the default orientation
  I - For ST, change the instrument
  ) - Return to main menu
```

III.5 Sensitivity and Output

The sensitivity and output menu is given below. The sensitivity key, 'n', has 3 levels: .1, 1., and 5. The level 1. is the normal default level. When the level is changed, it is shown at the user's terminal. The '*' key makes a hard-copy plot of the current scene being displayed. However, depending on how *Mip* is configured, there may be another way to get hard-copy, and the '*' key may not work. The 3 remaining keys, 'A', 'Z', and 'W', show useful information at the user's terminal.

```
  The sensitivity and output keys are listed below. To
perform the function, simply hit the key.

  n - Toggles the sensitivity, throuh 3 levels, of keys c,v (zooming),
      2,3,4 (field rotation), e,r,d,f (pointing offsets), and, 1,2,3,4
      (latitude and longitude relative to central body for observing
      from space, key 6)
  * - Will make an imagen plot of the scene being viewed. Many plots can
      be made, and when you leave Mip, give the command 'implot' to
      get the plots
  A - Show R.A. and DEC of first 10 stars in picture
  Z - Show status of parameters set by keyboard
  W - Show status of ST supplementary parameters (SAA, TDRS, Earth's shadow)
  ) - Return to main menu
```

III.6 Observation Files

The menu which handles SOGS interface files is shown below. The user may read and write critical window and moving target ephemeris files. For the moving target ephemeris files, however, a conversion must be made from the ASCII form to a direct access form. This is necessary in order to handle the data quickly and efficiently. Key '^' will make the conversion. The converted file consists of either positions or Chebyshev polynomials, which are interpolated by *Mip*.

It should be noted that if one is interested in simply reading a SOGS ephemeris file, and if the file is simply tracking a planet or satellite, then it is better to use the Naif ephemeris which *Mip* is already reading. In other words simply point *Mip* at the target body, and the program will take care of everything else.

When learning *Mip*, it is best to spend a fair amount of time experimenting with the keys below – use files which have no real value.

Mip reads and writes normal J2000 moving target ephemeris files. However, all internal calculations are, for now, done in B1950 coordinates. A conversion is made only when a file is just about to be written (B1950 → J2000), or just after being read (J2000 → B1950). UTC is used in the critical window file, and ET (no light-time correction) is used in the moving target ephemeris file.

```
  SOGS interface files are handled below. To perform the
function, simply hit the key. When writing a file, there are no editing
functions; however, one can always switch files and start over. SOGS
moving target ephemeris files must be converted to Mip format to read
-- use key ^ or program 'sogmip'.

  ? - Input names of critical window and moving target ephemeris
      files to read and write.
  > - Display current file names
    - Toggle through start-stop times of critical window, see '-' below
  - - Go to start time of critical window # input
  X - Set Mip to use the moving target ephemeris file rather than the
      default pointing
  _ - Go back to default pointing rather than reading the SOGS moving target
      ephemeris file
  E - Input moving target ephemeris file start time and time step
  R - Input moving target ephemeris file stop time and time step,
      and write file
  C - Input critical window start time
  V - Input critical window stop time, and write interval
    - Convert ephemeris file from SOGS to Mip format
  ) - Return to main menu
```

III.7 Scene Complexity

The scene complexity menu is shown below. When running *Mip* it is often desirable to speed things up, especially if the user is fairly familiar with the particular scenes being displayed. This can be done with the keys below. For instance, if the user does not want to see terminators or grids on bodies, then he may remove them with key 'a'. This will make all aspects of the program run faster. For keys '&' and 'q' a carriage return is required after the requested input.

```
  The scene complexity (or how much to draw) keys are listed below.
To perform the function, simply hit the key. Stars may affect pointing if
the field of view is exactly 2X the inner camera field -- see the field of
view menu.

  a - Toggles grids on bodies on and off
  z - Toggles landmarks on bodies on and off
  w - Toggles through 3 levels of labeling -- none,
      around boarder, around boarder and on objects
  & - Input a resolution factor to draw at
  q - Input a star magnitude limit
  Q - Remove all stars
  ) - Return to main menu
```

III.8 End Operations

For ST, *Mip* should be terminated with key 'y'. When this key is hit, it will tell the user to input an 's', carriage return to stop the program. If the user hits the 'y' key by accident, or if he changes his mind, then he has only to input any other key, carriage return in order not to exit. If the user accidently hits key 'b' there is also a safety. If for some reason the user hits 'Ctrl y' in order to exit, no damage is done.

```
  Return, and stop program keys are listed below.
To perform the function, simply hit the key.

  y - Stop program and exit to operating system
      after a 's', caraige return is input
  b - Go back to program from which Mip was called, if
      Mip is being used as a subroutine. There are some
      automated picture planning options which you may
      not wish to use
  ) - Return to main menu
```

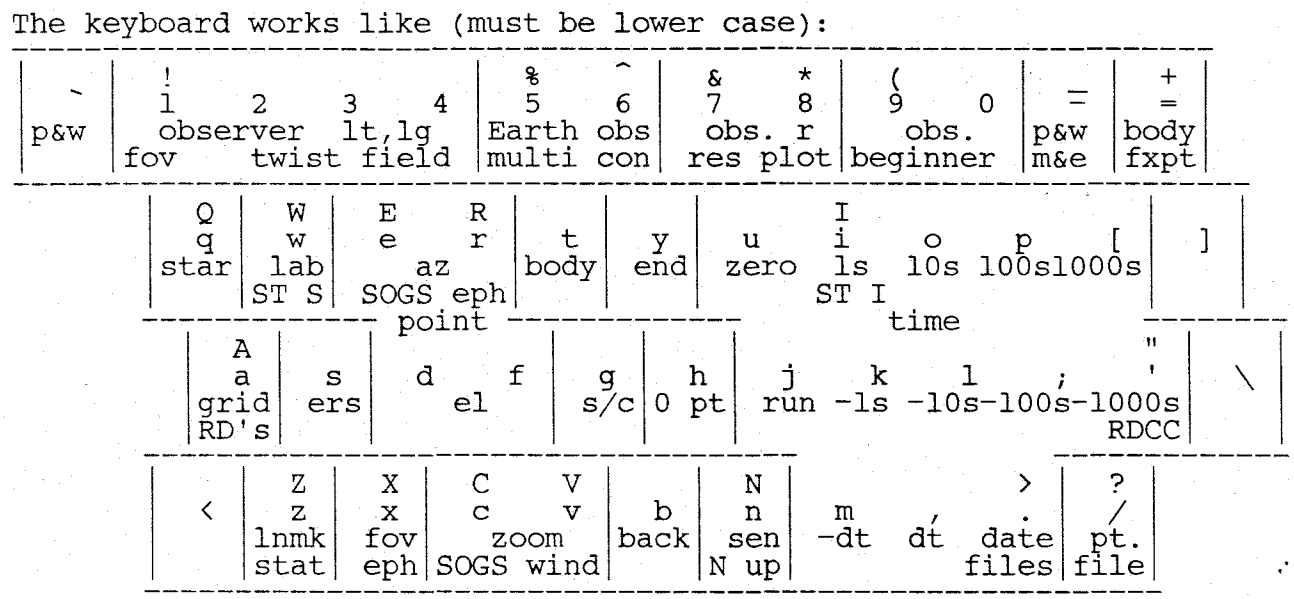
III.9 Overview

The overview menu given below is useful as a further source of help to the user. Also, in advanced mode, the 4 keys below, 'O', 'D', 'K', and 'H', can be used at any time to receive help, and this is the only source of help in advanced mode. When using key 'H', remember that you must type a capital 'E' to return to normal operation. The result of hitting key 'K', the keyboard summary, is shown on the next page.

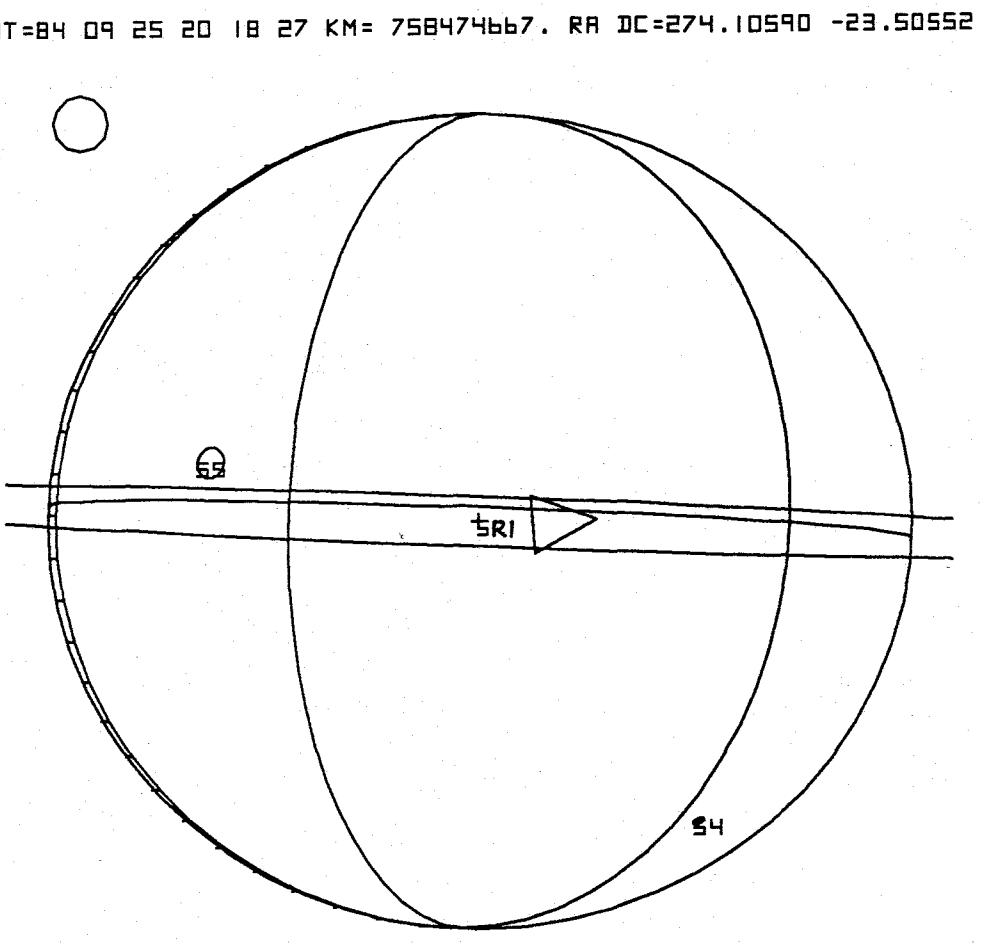
```
  The overview, display, and keyboard summary keys are listed
below. To perform the function, simply hit the key.

  O - Will print and overview of program Mip
  D - Will print a summary of the display being shown
  K - Will print a summary of the keyboard operation
  H - Brief information on any key. To exit from this
      you must type capital E (don't forget!)
  ) - Return to main menu
```

III.9.a Keyboard

A summary of the keyboard is shown below. This summary can be obtained by hitting key 'K' in beginner mode while in menu 9, or by hitting key 'K' at any time in advanced mode. The keyboard summary comes up automatically when switching from beginner to advanced mode.



IV. Video Output



Above is an example of the video output one can expect when running *Mip*. Note the UT calendar date (UT=), the distance to the observed body in kilometers (KM=), the right ascension and declination pointing angles (RA DC=), the phase angle in degrees (PHASE=), the body-relative latitude and longitude in degrees (LT LG=), the sun-Earth-ST angle in degrees (SUN-EARTH-ST=), and the angle from the ST pointing direction to the lit/dark limb of the Earth (LIT/DARK LIMB=).

By hitting key '"' the user can replace the right ascension and declination angles (RA DC=) with the angle from the ST pointing direction to the Moon (MOON=) and the angle from the ST pointing direction to the sun (SUN=).

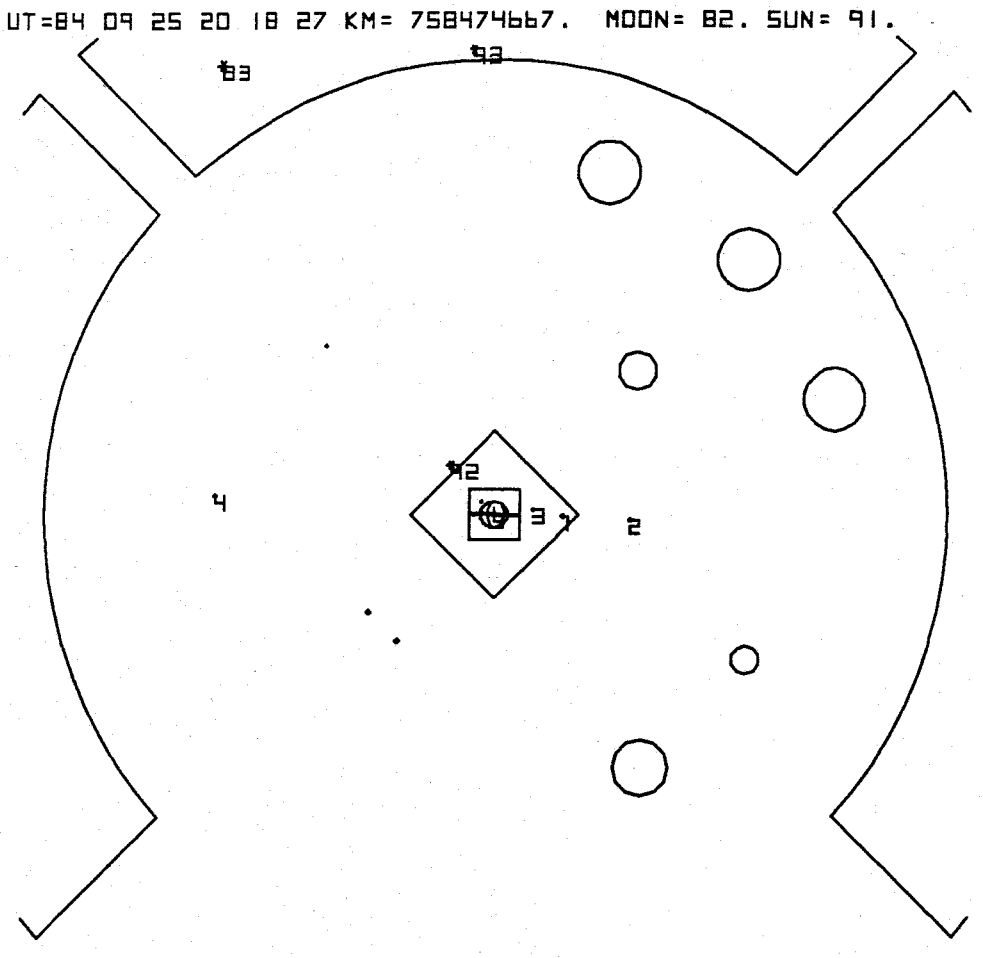
If the distance to the observed object becomes greater than 2147483645 kilometers, then the distance is given in megameters (MM=), not kilometers. This is due to the precision limitation of 4 byte integers, and the way *Mip* prepares data for presentation.

If the view is from the center of the Earth, or from any point in space, rather than

the ST spacecraft, then the sun-Earth-ST and lit/dark limb angles are not displayed.

As far as the actual objects being displayed: North celestial has been made the up direction (key 'N') and the scene is for Jupiter with the Galilean moons. Only Jupiter can be seen, with a few landmarks on its surface and its one ring. The sub-solar point is a triangle and the terminator has short lines connected to it on the sun side. Note that hidden line removal is done for the surface but not for the ring. Landmarks and rings are always drawn in yellow, while labelling and instruments are always green. The sub-solar point and terminator are white. The bodies are different colors, but what particular color a body is depends on how many bodies are present and the order the body is named. If any stars were present in the above scene, they would be color coded according to spectral class: O-B, dark blue; A, light blue; F, white; G, yellow; K, pink; M, red; unknown, green.

Shown below is the same scene but for a much larger field of view. Note the ST instruments, the Galilean moons, and the stars.



V. Theory

A brief discussion of the mathematical models used in *Mip* is given on the following pages. Every attempt to increase speed of computation has been made, both in the mathematical modeling and in the coding itself.

Since *Mip* is a highly interactive graphics program, most of the detail of how the program actually works should be obtained by looking at the FORTRAN 77 code. This code is well commented. However, it has been developed over 5 years on 4 different machines: a PDP 11/55 with Picture System 2 graphics, a Chromatics 7900 Color Graphics Computer, a VAX 11/780 with Grinnell and Ramtek frame buffers, and an IBM/AT microcomputer with EGA graphics. The program runs on several computers at JPL. Results, rather than coding style, have been emphazied. In the future, the code may be rewritten to improve the style.

In the following discussion, the expression $[\theta]_i$ is used to denote a 3×3 rotation transformation matrix about the ith axis by the angle θ. For example,

$$[\theta]_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}. \tag{1}$$

Also it is understood that constructions like $\hat{\mathbf{a}} \times \hat{\mathbf{w}}$ are always converted to unit vectors when used in transformation matrices:

$$M = \begin{bmatrix} \hat{\mathbf{a}} \\ \hat{\mathbf{a}} \times \hat{\mathbf{w}} \\ \hat{\mathbf{c}} \end{bmatrix}.$$

V.1 Geometry

Stellar aberration is not done in *Mip*. A light-time correction is made from the observer (ST spacecraft or center of Earth) to the central body of the system (usually a planet). No differential correction is made to the natural satellites of the system. This usually results in worst-case errors on the order of 10's of kilometers. At Callisto, the worst-case error is $\sim$ 50 kilometers. The very outer moons of Jupiter have the worst error – on the order of 150 kilometers. For now, since *Mip* is primarily a review and planning program, these errors are considered acceptable.

For Space Telescope, the transformation from inertial coordinates (B1950) to instrument coordinates follows the following formulation. The sun is in a plane perpendicular to the instrument plane. TIC, the transformation from inertial to celestial, is constructed as follows:

$$TIC = \begin{bmatrix} \hat{\mathbf{t}} \times \hat{\mathbf{s}} \\ \hat{\mathbf{s}} \times (\hat{\mathbf{t}} \times \hat{\mathbf{s}}) \\ \hat{\mathbf{s}} \end{bmatrix} \tag{2}$$

where $\hat{\mathbf{t}}$ is the pointing direction and $\hat{\mathbf{s}}$ is the sun direction.

The pointing direction is then transformed from inertial to celestial space. Call this vector $\hat{\mathbf{q}}$,

$$\hat{\mathbf{q}} = TIC\hat{\mathbf{t}}. \tag{3}$$

Now get azimuth and elevation angles, az, and el from $\hat{\mathbf{q}}$:

$$az = \tan^{-1}(q_2, q_1) \quad [\textbf{correct quadrant}], \tag{4}$$

$$el = \cos^{-1}(q_3). \tag{5}$$

Then the final transformation to instrument space, call $TITV$ (for transformation from inertial to TV), is construted like

$$TITV = [90^\circ]_3[el]_2[az]_3TIC. \tag{6}$$

Bodies are rotated to their correct latitude and longitude using the IAU standard (ref. 5), but terms below 1° have been droped. Rotations are done in the standard way, such as

$$TBFI = [-\alpha - 90^\circ]_3[-90^\circ + \delta]_1[-w]_3 \tag{7}$$

where $TBFI$ is a 3×3 transformation matrix from body-fixed to inertial, α is the R.A. of the body pole, δ is the DEC of the body pole, and w is the rotation of the prime meridian.

Landmarks are represented as circles on the body at their latitude and longitude. For large landmarks, the rim of the feature will still be on the body. The points which describe the landmarks on a body (scaled to unit radius) are computed once and then stored within the program. When the landmarks are to be displayed on the screen, they are transformed

and scaled along with the grid lines (also computed and stored once on a unit body), to get final picture coordinates.

In order to get the unit vectors representing a landmark in body-fixed, prime meridian coordiantes the following straight-forward method is used. First construct a transformation matrix *to* body-fixed coordinates from a coordinate system where the z-axis is normal to the body at the latitude and longitude of the surface feature. Call this matirx TL:

$$TL = [90^\circ - lt]_2[lg]_3 \tag{8}$$

where lt and lg are the latitude and longitude of the surface feature.

Now, given a set of points, $\hat{\mathbf{c}}_i$, which describe an appropriately scaled circle in the x-y plane, and the z coordinate is to the unit surface [$z = \cos(\tan^{-1}(r/a))$, where a is the radius of the body, and r is the radius of the surface feature], then the body-fixed coordinates of the points, $\hat{\mathbf{f}}_i$, are given by

$$\hat{\mathbf{f}}_i = TL\hat{\mathbf{c}}_i. \tag{9}$$

Bodies are represented as tri-axial ellipsoids by the following method. Consider a set of unit vectors, in the body-fixed frame, $\hat{\mathbf{v}}_i$, which contains grid lines and landmarks, but assumes that the body is round. The body can be given its tri-axial shape by transforming the vectors $\hat{\mathbf{v}}_i$ with a 3×3 transformation matrix, TS:

$$TS = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \tag{10}$$

where a, b, and c are the prime, intermediate, and small axis of the tri-axial ellipsoid respectively. These values are in the graphics system coordinates – not kilometers. Grid lines are distorted from their true positions, but these lines are only drawn to show the shape of the body. Unless a body is very ellipsoidal the distortion is not appreciable. Landmarks retain their true positions even after the scaling, because a correction is made to the latitude and longitude before the points are stored. The correction follows the formulation (lt_c and lg_c are the corrected latitued and longitude, and are easily solved for)

$$m \begin{bmatrix} \cos(lg)\cos(lt) \\ \sin(lg)\cos(lt) \\ \sin(lt) \end{bmatrix} = TS \begin{bmatrix} \cos(lg_c)\cos(lt_c) \\ \sin(lg_c)\cos(lt_c) \\ \sin(lt_c) \end{bmatrix} \tag{11}$$

with m being the distance from the origin to the ellipsoid:

$$m = \left[\left(\frac{\cos(lg)\cos(lt)}{a} \right)^2 + \left(\frac{\sin(lg)\cos(lt)}{b} \right)^2 + \left(\frac{\sin(lt)}{c} \right)^2 \right]^{-1/2}. \tag{12}$$

In order to get the tri-axial limb in TV coordinates, first transform a circle in TV coordinates to body-fixed coordinates. Call the transformation matirx, $TBFTV^T = (TITV$

$TBFI)^T$. Now that the circle is represented in the body's coordinates, it may be scaled by the axes of the tri-axial ellipsoid. That is, apply the matrix TS described above. Now transform back to the TV system, and the original circle will be properly scaled to give the true limb. If we call the set of points representing a circle in TV space, $\hat{\mathbf{c}}_i$, and the true limb in TV space, $\mathbf{l}_i$, then the following equation holds:

$$\mathbf{l}_i = TBFTV\ TS\ TBFTV^T \hat{\mathbf{c}}_i. \tag{13}$$

To get the terminator of a tri-axiel body in TV coordinates, do the following. Knowing the sun direction, construct a transformation matrix from a sun system, where the z axis is in the direction of the sun, to inertial space. Now transform an x-y plane circle in sun space to inertial space to body-fixed space; scale by the tri-axial body as above, and then transform back to TV space. If we call the circle in sun space $\hat{\mathbf{c}}_i$, the transformation matrix from sun to inertial space TSI, the transformation from inertial to body-fixed space, $TBFI^T$, and the terminator in TV space, $\mathbf{t}_i$, then

$$\mathbf{t}_i = TBFTV\ TS\ TBFI^T\ TSI\hat{\mathbf{c}}_i. \tag{14}$$

For Space Telescope, the angle from the pointing direction to the Earth's lit or dark limb is calculated as follows. First construct a coordinate system transformation from inertial space to a "limb" system, TIL:

$$TIL = \begin{bmatrix} \hat{\mathbf{x}} \\ (\hat{\mathbf{x}} \times \hat{\mathbf{t}}) \times \hat{\mathbf{x}} \\ \hat{\mathbf{x}} \times \hat{\mathbf{t}} \end{bmatrix} \tag{15}$$

where $\hat{\mathbf{x}}$ is the unit vector from the Earth's center to the ST spacecraft, and $\hat{\mathbf{t}}$ is the unit pointing direction of the ST instruments.

The angle from the pointing direction to the Earth's center, θ, is easily calculated:

$$\theta = \cos^{-1}(-\hat{\mathbf{x}} \cdot \hat{\mathbf{t}}). \tag{16}$$

Now consider the limb point of interest. It is in the plane containing the $\hat{\mathbf{x}}$ and $\hat{\mathbf{t}}$ vectors. Also it is on a line containing the ST spacecraft and tangent to the Earth's surface. The Earth is considered to be round with a radius of 6378 km. Therefore, the limb point to Earth's center to ST angle, ϕ, is easily calculated:

$$\phi = \cos^{-1}(r_e/\|\mathbf{x}\|) \tag{17}$$

where r_e is the radius of the Earth (6378 km), and $\|\mathbf{x}\|$ is the distance to the ST spacecraft from the center of the Earth.

The angle from the pointing direction to the Earth's limb, λ, is now easily calculated from the angles θ and ϕ:

$$\lambda = \theta - 90^\circ + \phi. \tag{18}$$

To determine whether this limb point is lit or dark, transform the unit direction vector from the Earth's center to the sun (inertial coordinates) to the limb coordinate system constructed above. We have

$$\hat{\odot}_l = TIL\hat{\odot}_i \tag{19}$$

where $\hat{\odot}_i$ is the sun direction from the Earth's center, in inertial coordinates, and $\hat{\odot}_l$ is the sun direction in the limb coordinate system.

Next construct the direction vector to the limb intersection point in the limb coordinate system. Call this vector $\hat{\mathrm{l}}$:

$$\hat{\mathrm{l}} = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \\ 0 \end{bmatrix}. \tag{20}$$

The angle from the sun direction to the limb point, call ρ, now follows:

$$\rho = \cos^{-1}(\hat{\odot}_l \cdot \hat{\mathrm{l}}). \tag{21}$$

If this angle is less than 90° the limb point is lit, otherwise, it is dark.

V.2 Ephemerides

Mip uses the same standard Naif ephemeris files as the batch program *Percy*. See *Percy* documentation.

V.3 Stars

Star positions (R.A. and DEC) are stored as 4-byte integers, providing 9 places of precision. For small fields of view ($< 3°$), star positions are simply an offset from the center of the picture, with R.A. scaled by the cosine of the declination (of the center of the field). A rotation is then applied to get the correct field orientation. For larger fields of view, a set of more complicated equations is used to get the star positions, and then the rotation is applied.

Let's consider the non-rotated star positions. For small fields of view the simple equations are

$$\Delta\delta = \delta_s - \delta_c, \tag{22}$$

$$\Delta\alpha = (\alpha_s - \alpha_c)\cos(\delta_c) \tag{23}$$

where $\Delta\delta$ is the declination offset from the center of field, δ_s is the declination of the star, δ_c is the declination of center of the field, $\Delta\alpha$ is the right ascension offset from the center of field, α_s is the right ascension of the star, and α_c is right ascension of center of field.

For large fields the following equations are used. If $\delta_s > 0$, then

$$\Delta y = 90° - \delta_c - (90° - \delta_s)\cos(\alpha_s - \alpha_c). \tag{24}$$

Otherwise if $\delta_s < 0$, then

$$\Delta y = -90° - \delta_c - (-90° - \delta_s)\cos(\alpha_s - \alpha_c), \tag{25}$$

and

$$\Delta x = \sin(\alpha_s - \alpha_c)\cos(\delta_s) \tag{26}$$

where Δy is the corrected declination offset from the center of field, and Δx is the corrected right ascension offset from the center of field.

In order to get the correct camera orientation for the stars, start with $TITV$ – the transformation from inertial space to TV camera space (right ascension and declination are polar coordinates in inertial space). In inertial space, there is a unit vector which points in the direction of the camera. This vector is the z-axis in the TV coordinate system. Therefore, it's coordinates in inertial space are (let the componets of $TITV$ be denoted by t): (t_{31}, t_{32}, t_{33}). Now the projection of this vector onto the x-y inertial plane is: $(t_{31}, t_{32}, 0)$. Transforming this vector into TV space by multiplying it by $TITV$ and taking only the x-y projection in TV space, yields a vector normal to the direction of sight (that is, normal to the z-axis in TV space) and along the direction of increasing or decreasing declination (in other words pointing north or south). Call this vector

$$\mathbf{p} = (t_{11}t_{31} + t_{12}t_{32}, t_{21}t_{31} + t_{22}t_{32}, 0) = (x_\delta, y_\delta, 0). \tag{27}$$

One may obtain the angle of rotation, θ, from the x-axis in TV coordinates to the **p** vector as follows. If $\delta > 0$, then

$$\theta = \tan^{-1}(-y_\delta, -x_\delta) \ \textbf{[correct quadrant]}. \tag{28}$$

Otherwise if $\delta < 0$, then

$$\theta = \tan^{-1}(y_\delta, x_\delta) \ \textbf{[correct quadrant]}. \tag{29}$$

It is a simple matter now to form the transformation matrix from R.A. and DEC coordinates to TV line-of-sight coordinates. Call this matrix $TRDTV$, then

$$TRDTV = [90^\circ - \theta]_3. \tag{30}$$

V.4 Graphics

Within the FORTRAN 77 code of *Mip*, all rotations, clipping, and alphanumeric generation is done. All that is required of the graphics software is to (1) clear the screen, (2) change the color, and (3) connect the 2-D points with straight lines. This makes *Mip* extremely portable, and the program has run on many different machines with different graphics packages. The program is written in such a way that all the graphics routines necessary are written into one small file. Moving *Mip* between very different computers requires only editing this file, and compiling and linking *Mip* on the new machine.

All the rotations are explained in this memo. Once final TV coordinates are calculated (z-axis is the line-of-sight), the x and y coordinates are scaled by the distance of the observer and then plotted on the 2-D screen. This is a simple orthographic projection.

The resolution to draw bodies is varied by the program automatically depending on the size of the bodies and a user input resolution value. The maximum possible resolution is 1°. In other words a maximum resolution circle would be drawn with 360 points. The resolution is determined by the equation

$$\theta = 2\cos^{-1}(1 - D/r) \tag{31}$$

where θ is the resolution angle in degrees, r is the radius of the body in question (screen units), and D is a user input resolution value. *Mip* allows 5 levels of reslolution. A value of $D = .7$ will give very satisfactory results – segments rarely being visible. If the user wishes the program to run a little faster, he may make D larger. The program picks an integer value of θ from the set (divisors of 360): 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 18, 20, 30, 36, 45, 60, 90. The equation above is based on the concept that the maximum difference between a true arc and the actual straight line connecting the points is no greater than the value D.

Once the 2-D points have been calculated, a clipping routine is applied to get the actual points to be plotted. Of course the program is smart enough not to apply this proceedure to bodies which are obviously way outside of the field of view. Some machines will do some clipping themselves (Ramtek), but others will not do any (Grinnell). Therefore, to make *Mip* universally portable it was necessary to write a clipping routine into the FORTRAN 77 code.

Hidden line removal is done correctly for bodies as follows. Given a tri-axial body with axes a, b, and c, a vector perpendicular to the surface at (x, y, z) is $\mathbf{p} = (x/a^2, y/b^2, z/c^2)$. If we call the line-of-sight vector $\hat{\mathbf{t}}$, then $\mathbf{p} \cdot \hat{\mathbf{t}} < 0$ for the point to be visible. All longitude and latitude (equator only) lines on bodies are composed of one or two semi-circle arcs respectively. *Mip* checks the end-point of an arc to see if it's visible. If it's not, the program checks the other end, and connects points along the arc until they are no longer visible.

Alphanumeric generation is done by simply storing a set of points which describe the letters A-Z, and the digits 0-9. When an alphanumeric character is desired, its coordinates are generated in the proper place by adding the stored coordinates of the character to the location desired.

References:

1. "Galileo Interactive Picture Planning Program (GIP)," J. D. Callahan, IOM 314.8-391, 25 October 1982
2. "Voyager and Galileo Interactive Picture Planning Programs on the Chromatics (vip and gip)," J. D. Callahan, IOM 314.8-500, 10 November 1984
3. "Release of *Mip* Version 5/13/86," J. D. Callahan, IOM 314.8-603, 20 June 1986
4. "VAX ONP Mathematical Models," W. M. Owen, Jr., EOM 314-394, 11 August 1986
5. "Report Of The IAU Working Group On Cartographic Coordinates And Rotational Elements Of The Planets And Satellites, 1982," M. E. Davies, et al, Celestial Mechanics 29(1983)309-321, 1983