Chapter 10

Using the TV Display

10.1 Overview

The most useful implementations of the AIPS system include one or more computer peripheral devices capable of displaying images with multiple levels of grey and/or color. We refer to such devices as TV displays since most are implemented via large binary memories and standard television monitors. The main program AIPS and some tasks (e.g., TVFLG) use the TV display as an interactive input, as well as display, device. Other tasks (e.g., UVMAP, MX, APCLN) use the TV display simply to show the stages of the data processing. All use of the TV is optional and the AIPS system will run without such a device. The number of TV displays in the local system is parameterized (under control of the stand alone program SETPAR) and all programs are told which TV display (if any) is assigned to the current user.

10.1.1 Why Use (or not use) the TV Display

There are numerous reasons to use the TV display in writing AIPS routines. Grey scale images provide a realistic view of image data allowing the eye to integrate over noisy regions and to separate closely spaced features. Contour images require much more elaborate software to generate and they make unreasonably definitive assertions about the intensity levels. The TV may be used to display intermediate results which are never stored on disk. And the TV may be used to interact with the user in a very wide variety of ways. Current interactive usages include modification of the black and white transfer function, modification of pseudo coloring, selection of features of interest, selection of subimage corners, dynamic, multi-image displays, and communication to the task of simple information. The last is used primarily to tell iterative tasks that they may stop at the current iteration.

Despite these desirable features, an AIPS programmer should not put the TV in a task unless it is truly useful. A TV option requires some, potentially considerable extra coding effort and, during execution, some significant extra real and CPU time. Many TV devices also require a high rate of I/O in order to load an image and, especially, to interact with the user. If an algorithm is based on the TV display, then it will not be available at those AIPS sites which do not have one. Although TV displays can function as graphics devices, many of them are very slow in that mode. Finally, tasks which use the TV will interfere with the interactive AIPS user's other uses of the display by replacing current images in the TV memory or modifying the zoom, scroll, transfer functions, et al.

10.1.2 The AIPS Model of a TV Display Device

As AIPS was being designed, it was realized that there was already a wide variety of TV display devices on the market and that the market would not hold still. The NRAO initially purchased two International Imaging Systems (IIS) Model 70/E displays. However, that company changed rapidly to Model 70/F and now no longer sells the Model 70. Instead, it now markets, among other things, a Model 75 and a less expensive IVAS model. The NRAO acquired an IVAS in 1986. Support for work stations with imaging capability was added early in 1988 after SUN loaned a SUN 3/110 to the NRAO. Full support for a generic

X-windows workstation was completed in 1992. Our initial choices undoubtedly color our image of what a TV display device does and how it does it. Nonetheless, we have attempted to design the code to be very general and to account for the range of options available on individual models of display and for the range of different manufacturers.

We regard the TV display as being a computer peripheral device which accepts the basic I/O operations of open, close, initialize, read, and write. Special Z routines are provided in AIPS since we do not assume that these I/O operations are identical, for all TVs and host operating systems, to those for disks, tapes, or Fortran devices. We assume that the TV display may be subdivided logically into a variety of sub-units which control the various functions of the display. Special libraries of subroutines, subdivided by model of TV display, are provided for communicating to these sub-units. These subroutines are called "Y routines" because all of them have names beginning with the letter Y. The NRAO has, at this time, developed the Y routines for IIS Models 70/E, 70/F, and IVAS and for SUN workstations using the SUNView software system. In addition, we store, distribute, and attempt to maintain sets of Y routines developed by other institutions for other models of displays. At the moment, we have Y routines for DeAnza, developed by Walter Jaffe at the STScI, for IIS Model 75, developed by IIS, and for Comtal Vision 1/20, developed by Andy Lubenow at the University of Illinois. We expect (hope) to receive more sets of Y routines for distribution in the near future. We have developed and support a set of Y routines and the other software needed to operate a TV device "remotely". These so-called "virtual-television" routines connect the computer to the remote TV device operating on some cooperating AIPS computer, presumably located near the user.

AIPS also uses, at both the Y and non-Y programming levels, a TV device parameter common. This common is initialized by a Y routine (YTVCIN) and is maintained via a disk file and a stand alone program (SETTVP). The common contains both fundamental parameters (i.e., number of memories, display size, maximum intensity, maximum zoom, etc.) and parameters describing the current state of the TV (i.e., which planes are on, current zoom and scroll, etc.).

In order to provide the full functionality of the basic AIPS verbs and tasks, a TV display device needs to contain the following sub-units. Note, these subunits are logical devices. They may be implemented as control registers in the device or in numerous other fashions. It is only necessary that the Y routines impose on the device a control that forces it to this general structure.

1. IMAGE MEMORIES: These are one or more memories each n bits deep which hold the grey-scale images to be displayed. All n bits of the image contribute to the display. The memory is assumed to have a fixed number of pixels on each axis and to be addressable at the individual pixel level. The addresses are assumed to be one-relative and to begin at the lower left of the display. The number of bits, the dimensions of each axis, and the number of memories are parameters inside AIPS. It is also assumed that each memory may be turned on and off in each of the three colors individually, although the capability is used solely to display images having separate R, G, and B planes. Beginning with the 15APR90 release, we assume that the image memory is actually viewed through a rectangular "window" whose size and coordinates with respect to the image meory may change with time.

2. GRAPHICS MEMORIES: These are one or more memories each 1 bit deep used to display graphical information, such as axis labels or line drawings on top of the grey-scale images. It is assumed that the overlay planes have the same number of pixels on each axis as the image memories and that each overlay plane may be enabled or disabled individually. It is nice to be able to assign unique colors to each of the overlay planes. AIPS will want to use four overlay planes, but all standard programs will work more or less normally with only one. The number of graphics memories is a parameter.

3. CURSOR AND BUTTONS: The cursor is some form of marker which may be enabled or disabled and which is under the positional control of some mechanical device (e.g., trackball, joy stick, thumb wheels, mouse). The position of the cursor on the TV screen may be read at any time it is enabled. The "buttons" are some device to signal conditions to the programs, such as "this is the desired position" or "time to quit". AIPS assumes that there are four such buttons returning to the program a value between 0 and 15. Simultaneity of more than one button is never used, however.

4. LOOK UP TABLES: These are tables of numbers which convert the stored n-bit image intensities to the desired display intensities. AIPS assumes that there is one n-bit in, m-bit out look up table ("LUT") for each color of each image memory. AIPS also assumes that there is a second set of three

look-up tables, called the output function memory ("OFM"), which converts the sums of all enabled memories to the final displayed intensities. In practise, AIPS uses the individual channel LUTs for black and white enhancement (most of the time) and the OFM for pseudo-color enhancement. There are algorithms, such as TVHUEINT, which utilize the full capability of the two sets of look-up tables. Most other routines work well with a single LUT per image memory. Arrays inside AIPS are likely to be dimensioned for 11-bit image planes and a 10-bit OFM. (These assumptions will be increased as the need arises.)

5. SCROLL: It is assumed that each image memory may be displayed on the TV screen shifted along both axes by varying amounts. AIPS assumes that each memory may be scrolled independently and that the graphics memories may be scrolled together independent of the image memories. The minimum increments of scroll along each axis are parameters. Note that AIPS does not make heavy use of scroll except for the TVROAM display and, of course, TVSCROLL. TVROAM does not require the image memories to scroll independently.

6. SPLIT SCREEN: It is assumed that the screen may be divided into quadrants and different image channels enabled in each quadrant. There is a control parameter specifying the degree to which the local TV display has this capability. AIPS currently uses split screen primarily in the TVROAM display, but also uses it during image enhancement in the channel blink routines.

7. ZOOM: AIPS assumes that the display of an image may be blown up about any pixel by automatic pixel replication by simple integer factors (or by integer powers of two) without affect on the images stored in the image memories. The highest factor (or power of two) available is a parameter. Zoom is important to the TVMOVIE algorithm and is used in many of the image enhancement routines.

The most important TV operations of AIPS could be implemented on a TV device having one image memory, one LUT with three OFMs (or three LUTs), and a cursor with buttons. Additional image memories, graphics memories, an OFM, scroll, split screen, and zoom are needed primarily for less important aspects of the basic operations and for some interesting, but esoteric operations.

There are several other sub-units in the IIS Model 70 which are supported by the Y routines in that sub-library. They include an input function memory (translates input to the TV from the host and from the ALU), a histogram generator, a feedback arithmetic logic unit, shift and min/max registers, and the like. Although there are no standard routines in AIPS which use these units, there are two nonstandard tasks for histogram equalization which make some use of them. The special Y routines used by these two tasks will be described below, but they should not (yet) be required for other kinds of TV devices — if they are even possible on them. All functions which are not possible on the current TV device should be represented by "stubbed" versions of the appropriate Y routines and these stubs should return an error condition.

10.2 Fundamentals of the Coding

10.2.1 The Parameter Commons and their Maintenance

All application routines must open the TV device via a call to TVOPEN and close it via a call to TVCLOS. TVOPEN opens a disk file called ID10000n with exclusive use requested, where n is the number of the assigned TV device. From the first record of this file, it reads a 256-word record containing parameters which describe the structure and current status of the assigned TV device. The parameters are stored in a common called /TVCHAR/ which is obtained by including DTVC.INC. TVCLOS puts back to the disk the time variable portions of this common and then closes the file. In this way, several users/programs may share the TV in sequence and all will know the current status information. The disk file may be initialized and the individual parameters set by using the stand-alone program SETTVP. TVOPEN automatically reads the current window coordinates from the device after it is opened and puts them in this common. The parameters are important to the correct functioning of the local TV device and must be set and maintained carefully.

The fixed portion of /TVCHAR/, namely that portion not written by TVCLOS, includes the parameters:

```
NGRAY       The number of n-bit image memories.
NGRAPH      The number of 1-bit graphics overlay memories.
NIMAGE      The number of images which may be stored
            simultaneously in a grey-scale image plane (affects
            the image catalog mostly).
MAXXTV(2)   The number of pixels in the X and Y directions.
MAXINT      The highest grey-scale intensity = 2 ** n - 1.
LUTOUT      Peak intensity out of LUTs.
OFMINP      Peak intensity into OFMs.
OFMOUT      Peak intensity out of OFMs.
SCXINC      The minimum increment in scroll in the X direction.
SCYINC      The minimum increment in scroll in the Y direction.
MXZOOM      If > 0, the highest power of two for zooming.
            If < 0, highest linear zoom factor = 1 - MXZOOM
            in simple integer steps.
CSIZTV(2)   The size of characters in pixels in the X, Y
            directions.
TYPSPL      Type of split screen: 0 none, 1 vertical division
            only, 2 horizontal division only, 3 either, 4 both.
TVALUS      Number of TV arithmetic logic units.
TVXMOD      Mode for loading TV in X direction: 0 none, 1 ok in
            AIPS order (to right), 2 ok in reverse direction.
TVYMOD      Mode for loading TV in Y direction: 0 none, 1 ok in
            AIPS order (to top), 2 ok in reverse direction.
ISUNUM      Number of image storage units.
TVGAMA      (real) "Gamma correction" power for the OFM to
            correct for the non-linearity of the screens
TVDUMS(9)   Spare room

The time variable portion of the /TVCHAR/ common is:
TVZOOM(3)   Current zoom: power of two, X, Y zoom center
TVSCRX(16)  Current X scroll for 15 image planes and graphics.
TVSCRY(16)  Current Y scroll for 15 image planes and graphics.
TVLIMG(4)   Bit pattern for which images are on by quadrant:
            quadrants are numbered CCW from top right and the
            lsb is for grey plane one and NGRAY+NGRAPH bits are
            used.
TVSPLT(2)   Current split screen position in X, Y.
TVSPLM      10 * (number planes in X) + (number planes in Y) in
            Roam mode.
TVSPLC      Roam mode: digits imply which channels in which
            order.
TYPMOV(16)  Movie loop code: 4 * (magnification factor - 1) +
            64 * (number frames remaining). Add 2 if this is the
            first plane of the movie.  Add 1 if the frames are
            in "display" rather than movie order.
WINDTV(4)   BLC x,y, TRC x,y of the current window visible from
            the TV memory.  (Not affected by zoom, scroll!)
TVDUM2(4)   Spare room
YBUFF(160)  Machine-dependent parameters.
```

There is a second TV include which controls I/O, but is little used elsewhere. It is obtained by including and contains:

```
TVLUN       LUN of open TV device.
```

```
TVIND      Position of TV device in FTAB for I/O.
TVLUN2     LUN of open TV parameter disk.
TVIND2     Position of parameter disk in FTAB.
TVBFNO     Not used (map style I/O no longer supported).
TVMAP      Not used.
```

10.2.2 The I/O Routines

Four basic I/O operations for TV devices are supported: open, close, I/O reset ("master clear"), and data transfer (read/write). The actual Z subroutines which perform these operations are both TV device and host operating system specific. The subroutines are stored in the subdirectory appropriate for the host operating system with names reflecting the TV device type. To insure that the correct Z routines are link edited, a layer of Y routines is interposed between these Z routines and all other non-Y AIPS routines. No non-Y subroutine or program should call these Z routines. These Z subroutines have names of the form ZMMMOO, where MMM is the TV model (i.e., M70 for IIS Models 70 and 75, DEA for DeAnza) and OO is the type of operation (OP for open, CL for close, MC for I/O reset, and XF for data transfer).

Note that the four Z routines may have TV-device specific call sequences and that not all devices require four Z routines. The current implementations are

```
Z...OP  :
    ZARGOP (LUN, IND, IERR)        ZARGO2 (FCB, PNAME, IERR)
    ZDEAOP (LUN, IND, IERR)        ZDEAO2 (FCB, PNAME, IERR)
    ZIVSOP (LUN, IND, IERR)
    ZM70OP (LUN, IND, IERR)        ZM70O2 (FCB, PNAME, IERR)
    ZSSSOP (LUN, IND, IERR)        ZSSSO2 (FCB, PNAME, IERR)
    ZV20OP (LUN, IND, IERR)        ZV20O2 (FCB, PNAME, IERR)
    ZVTVOP (LUN, IND, IERR)        ZVTVO2 (FCB, PNAME, IERR)
    ZVTVRO (LUN, IND, IERR)        ZVTVO3 (FCB, PNAME, IERR)
```

Performs the needed channel assignment and opens a non-map entry in the FTAB. The DeAnza version also calls ZDEAXF ('DAT ',...) to initialize the I/O. The Z...OP routines are generic routines which do machine-dependent things by calls to other Z routines including the Z...O2 versions listed. ZSSSOP exists only in APLSUN:.

```
Z...CL  :
    ZARGCL (LUN, IND, IERR)        ZARGC2 (FCB, IERR)
    ZDEACL (LUN, IND, IERR)        ZDEAC2 (FCB, IERR)
    ZM70CL (LUN, IND, IERR)        ZM70C2 (FCB, IERR)
    ZSSSCL (LUN, IND, IERR)        ZSSSC2 (FCB, IERR)
    ZV20CL (LUN, IND, IERR)        ZV20C2 (FCB, IERR)
    ZVTVCL (LUN, IND, IERR)        ZVTVC2 (FCB, IERR)
    ZVTVRL (LUN, IND, IERR)        ZVTVC3 (FCB, IERR)
```

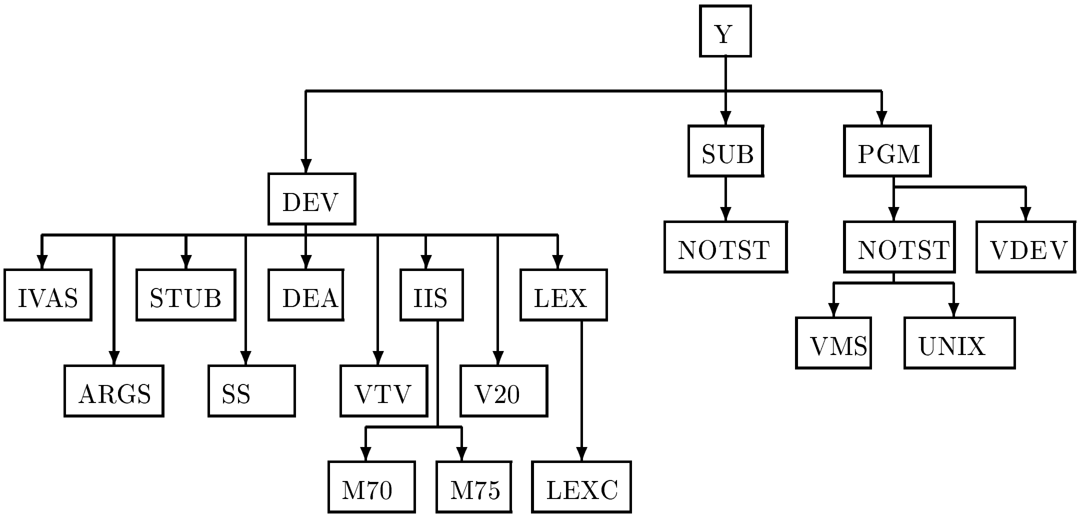
Performs a close (deassign) and clears the FTAB entry. The DeAnza version calls ZDEAXF ('DET ',...) to perform a deallocation before closing. The M70 version flushes the IO buffer before the close.

```
Z...MC  :
    ZM70MC (FCB)                   ZM70M2 (FCB, IERR)
       Performs a "rewind" QIO operation causing the IIS to
       reset its I/O status.
    ZARGMC (FCB)
       Invokes ZARGS
    ZDEAMC
    ZSSSMC
    ZV20MC
       Null subroutines.
```

```
Z...XF  :
    ZARGXF (OP, NBYTES, HEADER, BUFFER, IERR)
       translates IIS Model 70 like commands for ARGS and then
       calls a selected portion of ZARGS
    ZDEAXF (OPER, BUFFER, NBYTE, PA, PB, WAIT, IERR)
    ZDEAX2 (FCB, IOP, OPER, BUFFER, NBYTE, PA, PB, WAIT, IERR)
       "Calls to ZDEAXF map one to one to calls to IP8 routines
       in the DeAnza IP8500 level 0 software package."  Does
       requested I/O operation using opcode definitions
       contained in IP8IOF.MAR (supplied by DeAnza, not NRAO).
    ZM70XF (OPER, NBYTES, HEADER, BUFFER, IERR)
    ZM70X2 (OPER, FCB, BUFF, NBYTES, IERR)
       writes an eight-word command HEADER to the IIS after
       preparing the checksum word of the header.  Then reads
       from or writes to the IIS NBYTES of BUFFER.  Actual
       writing to the IIS is deferred until a large buffer is
       filled or a read or close occurs.  Issues a master
       clear on error.
    ZSSSXF (OP, DAT, NWORDS, BUFFER, NSWORD, IERR)
    ZSSSX2 (FCB, OP, DAT, NWORDS, BUFFER, NSWORD, ISTAT)
       reads/writes from the stand-alone SSS program which
       creates and operates the workstation ``TV'' device.
    ZV20XF (OPER, NBYTES, COMST, COMDB, BUFFER, IERR)
    ZV20X2 (OPER, FCB, NBYTES, COMST, COMDB, BUFFER, IERR)
       performs the actual QIOW operation to transmit data
       to/from the Vision 1/20.  See ZB.DOC provided with the
       Comtal device driver ZBDRIVER.
    ZVTVXF (BUFSW, BUFSR, HBUF, IERR)
    ZVTVX2 (FCB, BUFSW, BUFSR, BUFFER, IERR)
       reads/writes from/to server (real TV computer) from the
       client (virtual TV).  Thus, is used by AIPS et al.
    ZVTVRX (FIND, BUFSW, HBUF, IERR)
    ZVTVX3 (FCB, BUFSW, HBUF, IERR)
       reads/writes from/to client (i.e., the virtual TV) from the
       server (actual TV).  Thus, is used by program TVMON.
```

10.2.3 The Y Routines

The directory structure of AIPS was designed with the need to support multiple TV devices in mind. See the Appendix to Volume I of Going AIPS for details. The highest level of the Y-routine device-dependent directory tree, called Y/DEV with logical name YGEN, contains a "generic" version of the Y subroutine library. In some cases, there really can be a generic version of a routine, i.e., vectors can be drawn on the TV screen by a sequence of calls to the image writing routine. In many cases, however, the Y routines in YGEN are "stubs" which issue an error message and return error code 2. Below this is a set of directories for the various supported devices. The diagram below summarizes the full Y directory tree:



Y Directories

The STUB area is used to hold stubbed versions of those Y routines which have device-independent implementations. On those systems with no TV display, link edits of AIPS and other "TV" programs will use this area. Each of the end nodes of this tree has a corresponding link library. If a routine is present in the end node, then the compiled version of that routine will be in the corresponding library. If the routine is not present, then the compiled version of the routine from the next higher node having that routine will be placed in the library. Thus, one needs to write for the IVAS, for example, only those Y routines for which the DEV version will not do. In particular, since the IVAS does not have a feed back arithmetic logic unit (for example), we can use the generic (stubbed) version of YALUCT and have no version of YALUCT in the IVAS directory.

In the chart below, the names of all routines in the Y directories are given along with indications of which directories have versions of them. The codes are "G" for generic, "S" for stubbed, "D" for device dependent, and "O" for operating system dependent. The routines are broken into four "levels" depending on how they may be used. Levels 0 and 1 may be called from non-Y routines and level 3 may not. Level 2 was originally designed to be called from non-Y code, but experience has shown that the device capabilities are available only on IIS Models 70 and 75. and, hence, they are not recommended.

Level 0: required, but a generic is available

Routine	DEV	IIS	M70	M75	IVAS	DEA	STUB	VTV	SS
YCHRW	G				D		S	D	
YCINIT	G							D	
YCNECT	G				D		S	D	
YCOVER	G							D	
YCREAD	G							D	
YCUCOR	G		D				S	D	
YCURSE	G		D				S	D	
YCWRIT	G							D	
YFILL	G				D			D	
YFIND	G							D	
YLOCAT	G							D	
YLOWON	G								
YSLECT	G						S	D	
YTCOMP	G								
YTVCLS	G							D	

YTVOPN	G						S	D	
YWINDO	G							D	D

Level 1: required, device-dependent is needed

Routine	DEV	IIS	M70	M75	IVAS	DEA	STUB	VTV	SS
YCRCTL	S		D	D	D	D		D	D
YGRAFX	S	D			D	D		D	D
YGRAPH	S		D	D	D	D		D	D
YIMGIO	S	D			D	D		D	D
YINIT	S		D	D	D	D		D	D
YLUT	S	D			D	D		D	D
YOFM	S		D	D	D	D		D	D
YSCROL	S		D	D	D	D		D	D
YSPLIT	S	D			D	D		D	D
YTVCIN	S		D	D	D	D			D
YTVCL2	S	D			D	D		D	D
YTVMC	S	D			D	D		D	D
YTVOP2	S	D			D	D		D	D
YZERO	S	D			D	D		D	D
YZOOMC	S		D	D	D	D		D	D

Level 2: IIS dependent routines - not available to other TVs

Routine	DEV	IIS	M70	M75	IVAS	DEA	STUB	VTV	SS
YALUCT	S		D	D				D	
YCONST	S		D	D				D	
YFDBCK	S		D	D				D	
YIFM	S		D	D				D	
YMNMAX	S		D	D				D	
YRHIST	S		D	D				D	
YSHIFT	S		D	D				D	

Level 3: routines only called by Y routines - not available to non-Y

Routine	DEV	IIS	M70	M75	IVAS	DEA0	STUB	VTV	SS
BYTE2I									
I2BYTE									
YBUTON					D				
YCHACT									
YCMND									
YCMSET									
YDOERR					D				
YGGRAM	S		D	D		D			
YGRAFE	S		D	D					
YGYHDR	S		D	D					
YISDRM	S		D						
YISDSC	S		D						
YISJMP	S		D						
YISLOD	G								
YISMPM	S		D						

YLINTV						
YMAGIC	S			D		
YMKCUR	S					D
YMKHDR	S		D	D		
YSTCUR	S		D	D		
YVRTR					D	

The following sections provide a brief overview of the current Y routines. The precursor comments of most of the Y routines are reproduced near the end of this chapter.

Level 0

1. YCHRW writes characters into an image or graphics plane. The DEV version is TV independent and uses a 7 x 9 pixel area per character. The background intensity is set to 1 for multi-bit channels and 0 for graphics. Uses YIMGIO.
2. YCINIT initializes the TV image catalog. It is a Y routine to allow for remote TV devices.
3. YCNECT writes a line segment in an image or graphics plane at a specified intensity. It is a Y routine to allow for TV devices with hardware vector generators. Uses YIMGIO.
4. YCOVER checks the TV image catalog to see if there are overlapping images visible. It is a Y routine to allow for remote TV devices.
5. YCREAD uses the TV image catalog to determine which image is associated with a particular pixel. It is a Y routine to allow for remote TV devices.
6. YCUCOR converts cursor positions and obtains the corresponding image header. It is a specialized version of YCURSE to avoid any TV I/O and to do the image catalog work.
7. YCURSE enables/disables cursor and cursor blink and reads cursor position and buttons value. The main complications come from corrections for zoom and scroll. The generic version uses YCRCTL and assumes that zoom/scroll is done by specifying the pixel to be visible in the upper left corner.
8. YCWRIT updates the TV image catalog to add an image to the display list. It is a Y routine to allow for remote TV devices.
9. YFILL fills a rectangular region of the display with an image of constant intensity. Uses YIMGIO and is a Y routine to allow for devices with hardware polygon fill.
10. YFIND uses the TV image catalog to determine if only one image is visible and, if so, to return the image header. It is a Y routine to allow for remote TV devices.
11. YLOCAT uses the TV image catalog to convert a list of image pixel positions to a list of TV positions. It is a Y routine to allow for remote TV devices.
12. YLOWON selects the lowest numbered channel from a bit mask. It is a Y routine for no good reason.
13. YSLECT enables/disables grey and graphics channels setting the proper values into TVLIMG. Uses YSPLIT for image planes and YGRAPH for graphics planes.
14. YTCOMP performs logical tests on parameter values to see if they have changed. It is a Y routine only because it is now used only to minimize I/O to the DeAnza control registers.
15. YTVCLS updates and closes the TV parameter disk file and closes the TV device via YTVCL2.
16. YTVOPN opens and reads the TV parameter disk file and opens the TV device via YTVOP2.
17. YWINDO reads current viewing rectangle on the image memory, can force it on workstations.

Level 1

1. YCRCTL reads/writes the cursor/trackball control register including position, enable/disable on each axis, blink control.

2. YGRAFX reads or writes the basic colors to be used for the graphics planes and the cursor.

3. YGRAPH enables/disables graphics overlay planes by altering the graphics color look up tables. A non-essential nicety is the use of complimentary colors when two or more graphics planes are enabled at the same pixel.

4. YIMGIO reads/writes a line of image data from/to a grey-scale or graphics plane. It will perform buffer swaps if needed to get the desired angle and bit-level corrections when graphics planes are read. This is the most heavily used Y routine, in part because of the generic versions of YCHRW and YCNECT.

5. YINIT initializes all subunits of the TV, clears the TV memories, resets the image catalog, and resets status parameters in common.

6. YLUT reads/writes the full channel-level lookup table for one or more image channels and colors.

7. YOFM reads/writes the full OFM lookup table for one or more colors.

8. YSCROL writes the scroll control registers for one or more channels.

9. YSPLIT reads/writes the split screen control registers. This is the actual control of the split screen center and of which channel(s) are enabled/disabled in each quadrant.

10. YTVCIN provides initial values for the TV characteristics commons. Used now only in SETTVP.

11. YTVCL2 closes the TV device. Actually it is usually just an interface to the appropriate Z...CL subroutine.

12. YTVMC resets the TV I/O status. Actually it is usually just an interface to the appropriate Z...MC subroutine.

13. YTVOP2 opens the TV device. Actually it is usually just an interface to the appropriate Z...OP subroutine.

14. YZERO clears a full grey or graphics memory by the fastest possible method.

15. YZOOMC writes the zoom control registers giving magnification and zoom center.

Level 2

1. YALUCT reads/writes the IIS arithmetic logic unit control registers. No actual function is performed until a feedback operation is done via YFDBCK. This routine is very IIS specific and we doubt that its functions can be implemented on other TVs.

2. YCONST reads/writes the constant "biases" which are added to the sums of the individual enabled channels before the signals are sent to the OFM.

3. YFDBCK causes a feedback operation to occur. The ALU does its thing with one or more channels and returns an 8 or 16 bit result to one or two channels. A magic bit causes the function to be a simple zeroing of a channel.

4. YIFM reads/writes a portion of the input function memory. This lookup table can be used in writing data to the TV memory and in the feedback operation. AIPS does not do the former and only one non-standard task does the latter.

5. YMNMAX reads the min and max output from the sum of all enabled grey-scale planes for each color.

6. YRHIST reads a portion of the histogram of the output of the OFM for a selected color. The IIS can do this on the fly if properly equipped.

7. YSHIFT reads/writes the shift registers which shift the 13-bit output of the sum of all enabled channels before the data get to the OFM.

Level 3

1. BYTE2I converts VMS BYTE data to integer forcing the BYTE variable to be in the range 0-255.

2. I2BYTE converts integer data in range 0-255 to VMS BYTE data.

3. YBUTON reads the state of the button buffer of the TV. IVAS uses this for converting two different buttons into “button D”.

4. YCHACT activates a specified Comtal channel.

5. YCMND sends a command string to the Comtal via ZV20XF.

6. YCMSET sets up the Comtal driver for I/O.

7. YDEA.INC Include file giving parameter definitions to specify positions in YBUFF which correspond to the various registers in a DeAnza TV device.

8. YDOERR handles error conditions, reporting, and resetting for the IVAS.

9. YGGRAM reads/writes the lookup table used for graphics planes.

10. YGRAFE reads/writes the graphics control register which assigns a graphics plane as the “blotch” plane and another as the “status” plane. No use is made of this.

11. YGYHDR prepares a basic I/O control header for writing/reading image data to/from the IIS.

12. YISDRM reads/writes data memory of the NRAO Image Storage Unit.

13. YISDSC reads/writes micro-processor memory of the NRAO Image Storage Unit from/to the ISU disks.

14. YISJMP causes the micro-processor of the NRAO Image Storage Unit to jump to a specified address.

15. YISLOD reads/writes program memory of the NRAO Image Storage Unit from/to disk. Uses YISMP-M.

16. YISMPM reads/writes micro-processor memory of the NRAO Image Storage Unit.

17. YLINTV sends a line to/from Comtal image or graphics planes.

18. YMAGIC (Model 75 only) initializes graphics, zoom, and scroll subunits (called by YINIT only).

19. YMKCUR creates and loads the cursor pattern memory with a specified shape. Only the AIPS plus sign is implemented.

20. YMKHDR prepares a basic I/O control header for the IIS.

21. YSTCUR reads/writes the IIS cursor array. This 64 x 64 bit array provides a wide choice of patterns for the display “cursor”. AIPS uses only a simple plus sign with a blank pixel at the center.

22. YVRTR checks and switches as needed the transfer restrictions on the IVAS.

10.3 Current Applications

This section is devoted to a generally brief overview of the current application code. Primarily it will be used simply to point out which routines do what, with some comment on the methods. This should suffice as an introductory guide to the code for applications programmers wishing to include the TV display in their programs. In a couple of cases, some of the actual code will be reproduced in order to clarify the use of the various service routines. The precursor remarks for some of the most commonly used, non-Y service routines are reproduced at the end of this chapter.

10.3.1 Status Setting

By "status setting", we mean initializing the TV device, clearing memory channels, enabling and disabling portions of the display, and the like. Many of the applications which involve loading images to the TV display will zero the relevant memories (via YZERO) and clear the corresponding portions of the image catalog (via YCINIT) before carrying out their primary functions. However, the simplest examples of status setting are those performed by various AIPS verbs. The subroutine AU5 performs the verbs TVINIT (via YINIT), TVCLEAR (as follows), GRCLEAR (like TVCLEAR without the MOVIST call), TVON, TVOFF, GRON, GROFF (via calls to YSLECT), TV3COLOR (use YSLECT to turn off all channels, then YSLECT to turn on channels 1 through 3 in red, green, blue, respectively), and CURBLINK (via YCURSE). Other verbs in AU5 are described later.

The verb TVCLEAR is coded as follows. The channel number is picked up as an integer, the decimal code is converted to a bit pattern (via DECBIT), the movie status parameters are reset (via MOVIST), and then a loop over all selected grey planes is done to zero the memory (via YZERO) and clear the image catalog (via YCINIT).

```
C                                        Open TV device
      CALL TVOPEN (CATBLK, JERR)
      IF (JERR.NE.0) THEN
         POTERR = 101
         GO TO 980
         END IF
....
 200  ICHAN = ABS (TVCHAN) + EPS
C                                        convert to channel bit mask
      CALL DECBIT (NGRAY, ICHAN, ICHAN, ITEMP)
C                                        clear movie parameters
      CALL MOVIST ('OFFF', ICHAN, 0, 0, 0, IERR)
      DO 210 IP = 1,NGRAY
C                                        is plane requested
         ITEMP = 2 ** (IP-1)
         IF (IAND (ICHAN,ITEMP).NE.0) THEN

C                                        clear image catalog
            CALL YCINIT (IP, INBUF)
C                                        clear TV memory
            CALL YZERO (IP, JERR)
            IF (JERR.NE.0) GO TO 975
            TDEL = TDEL + 1.0
            END IF
 210     CONTINUE
      GO TO 900
....
C                                        normal TV close
 900  CALL TVCLOS (CATBLK, JERR)
      IF (TDEL.GT.0.0) THEN
```

```
         TDEL = TDEL + 1.0
         CALL ZDELAY (TDEL, IERR)
         END IF
      GO TO 999
```

10.3.2 Load Images, Label

Images are loaded to the TV by a wide variety of tasks (e.g., APCLN, TVPL, BLANK) and by several verbs (TVLOD, TVROAM, TVMOVIE). TVLOD will be illustrated in this subsection and the others mentioned in later subsections.

The full code from subroutine AU5A for TVLOD, except declarations, formats, error branches, and the like, is reproduced below. It begins by opening the TV control file and device (via TVOPEN). It moves the user adverbs to local variables to avoid changing their (global) values and opens the map file (via MAPOPN). It converts the user's PIXRANGE adverb using standard defaults (via RNGSET) and fills in some of the image catalog parameters in the header. It sets the window parameters (via TVWIND), selects a single grey scale memory plane (via DECBIT), and clears the movie parameters (via MOVIST). Finally, it finishes up the image catalog parameters, puts the header in the image catalog, and reads, scales, and loads the image to the TV memory (all via TVLOAD). Afterwards, it closes the map file (via MAPCLS) and the TV device and disk file (via TVCLOS).

```
      INCLUDE 'INCS:DHDR.INC'
C                                       open TV
      CALL TVOPEN (INBUF, IERR)
      IF (IERR.NE.0) GO TO 980
      IBSIZ = 2 * 4096
C                                       Map open junk: TVLOD, TVROAM
      IF (BRANCH.LE.2) THEN
C                                       adverbs -> local variables
C                                       Adverbs used:
C                                       TVCHAN = tv channel
C                                       INNAM  = File name
C                                       INCLS  = File class
C                                       INSEQ  = File sequence number
C                                       INDSK  = Disk number
C                                       USERID = User ID number
C                                       TVBLCO = TV bottom left corner
C                                       TVTRCO = TV top right corner
C                                       TVXINC = TV x pixel increment
C                                       TVYINX = TV y pixel increment
C                                       PXRANG = Range of pixel values
C                                       TVCORN = BLC on TV screen for
C                                                image
         ICHAN = IROUND (TVCHAN)
         IVOL = INDSK + EPS
         USID = ABS(USERID) + EPS
         SEQNO = INSEQ + EPS
         IF (USID.EQ.0) USID = NLUSER
         IF (USID.EQ.MAGIC) USID = 0
         CALL H2CHR (12, 1, INNAM, NAME)
         CALL H2CHR (6, 1, INCLS, CLASS)
         INC(1) = IROUND (TVXINC)
         INC(2) = IROUND (TVYINC)
         PTYPE = 'MA'
C                                       open map file
```

```
        CALL MAPOPN ('READ', IVOL, NAME, CLASS, SEQNO, PTYPE, USID,
    *      DLUN, DIND, CNO, CATBLK, INBUF, IERR)
        POTERR = 33
        IF (IERR.GT.1) GO TO 975
        CALL RCOPY (7, TVBLCO, LBLC)
        CALL RCOPY (7, TVTRCO, LTRC)
C                                      Image cat fill in some
        CALL RNGSET (PXRANG, CATR(KRDMX), CATR(KRDMN), CATR(IRRAN))
        CATBLK(IIVOL) = IVOL
        CATBLK(IICNO) = CNO
        CATH(IITRA) = FUNTYP
        ITVC(1) = TVCORN(1) + EPS
        ITVC(2) = TVCORN(2) + EPS
        POTERR = 49
        END IF
C                                      TVLOD
C                                      load one image plane
C                                      set windows
 100  TYPE = -1
      CALL DECBIT (NGRAY, ICHAN, ICHAN, I)
      CALL TVWIND (TYPE, INC, LBLC, LTRC, I, ITVC, IWIN, IERR)
      IF (IERR.NE.0) GO TO 970
C                                      convert channel number
      ICHAN = I
      CALL DECBIT (NGRAY, ICHAN, ICHAN, I)
C                                      clear movie parameters
C                                      load it
      CALL MOVIST ('OFFF', ICHAN, 0, 0, 0, IERR)
C                                      do the TV load
C                                      image catalog
      CALL TVLOAD (DLUN, DIND, I, INC, ITVC, IWIN, IBSIZ, RINBUF, IERR)
      IF (IERR.EQ.0) POTERR = 0
      GO TO 970
....
C                                      Close down ops
 970  CALL MAPCLS ('READ', IVOL, CNO, DLUN, DIND, CATBLK, F, INBUF,
    *   IERR)
C
 975  CALL TVCLOS (INBUF, IERR)
```

The verbs TVWEDGE, IMWEDGE, IMERASE, and WEDERASE load step wedge or pure zero images to the TV. They occur in subroutine AU5C. This routine calls TVFIND and possibly TVWHER to determine which image is desired. It then computes a buffer of appropriate values calling ISCALE (as TVLOAD does). AU5C then does a lot to set an appropriate image catalog header and writes that to the catalog via YCWRIT. Finally it loads the TV rows via calls to YIMGIO.

The image labeling verbs TVLABEL and TVWLABEL are implemented from subroutine AU5B. This routine calls TVFIND to determine which image is to be labeled and IAXIS1 to do the labeling. Subroutines IAXIS1 and ITICS are very similar to the standard axis labeling routines used to make plot files and to write directly to the TEK graphics device. Characters are written to a graphics memory with a black background by calls to IMANOT and lines are written to the graphics memory by calls to IMVECT. (See the precursor comments of these routines at the end of this chapter.) The verb TVANOT, which adds a user-provided string to the display, is also implemented in AU5B and uses TVWHER, IMANOT, and IMCHAR.

10.3.3 UVMAP

UVMAP uses the TV display for a fairly simple purpose — to show the pattern of sampled uv cells (after convolution of the data to the grid). In principle, the algorithm is simple: associate uv cells with TV pixels and display 0 on the TV when the uv cell is unsampled (0.00) and display MAXINT on the TV when the cell is sampled (not 0.0). Unfortunately, the uv grid may be larger than the TV display and the disk file contains the grid in transposed, quadrant-swapped order. The first problem is solved by decimation (examine only every n'th cell in X and m'th cell in Y). The quadrant swapping is solved by addressing the TV beginning in the middle and by starting in the middle of the buffer which is written to the TV. The transposition is solved by writing the rows of the file as columns on the TV. The subroutine in UVMAP which does this (UVDISP) uses the image writing mode parameters (TVYMOD and TVXMOD) to handle this correctly when possible, and to leave the display in transposed order when not (i.e., TVYMOD = 0).

10.3.4 APCLN, VTESS, MX, et al.

Iterative map analysis programs can make good use of the TV display. The user may, for example, request that APCLN display the residual map after each major cycle. APCLN does this, then turns on the cursor and waits up to 15 seconds for the user to push Button D to signify that sufficient iterations have been performed. Several tasks (currently MX, APGS, SDCLN, VTESS, UTESS, APVC) use code similar to that in APCLN for loading the image to the TV and requesting the user input. Given below is the TV subroutine from APCLN. Note that it uses the array processor to scale the data for YIMGIO. This is reasonable, but only for tasks which are already using the array processor for more important computations. The costs of opening and closing the AP device and performing the I/O to it make any improvement in computational speed marginal for computations such as these. Note also the scaling parameters used here. The lowest displayed intensity gets TV value 1.01 and the highest gets MAXINT+0.99 (after the 0.5 for rounding is added and before the integers are truncated by routine QVVFIX). This scaling is assumed (primarily by CURVALUE) for all linear transfer functions. TV value zero is reserved for “blanked” (indefinite) pixels and should always be given zero intensity on the display (by the LUTs and OFMs).

```
      SUBROUTINE DISPTV (TVPASS)
C-----------------------------------------------------------------------
C   DISPTV displays the current residual map on the TV, showing inner
C   portion only if that's all that will fit.
C   Inputs:
C      TVPASS  I     code: 0,1 => clear screen, else don't
C                          0,3 => don't question the user about
C                                 quitting
C   Output:
C      TVPASS  I     code: 32700 => user wants to quit cleaning
C-----------------------------------------------------------------------
      INTEGER   TVPASS
C
      CHARACTER PREFIX*5
      INTEGER   JROW(1), WIN(4), MY, FIND, BIND, IERR, ICH, CATII(256),
     *   IQ, IB, I, MX2, MX, IWIN(4), IY
      REAL      XN(4), XBUFF(1), TD, RPOS(2), XFLUX, TVLMAX, TVLMIN,
     *   ARG, CATIR(256)
      LOGICAL   MAP, EXCL, WAIT, LERR, F
      INCLUDE 'APCLN.INC'
      INCLUDE 'INCS:DFIL.INC'
      INCLUDE 'INCS:DTVC.INC'
      INCLUDE 'INCS:DMSG.INC'
      INCLUDE 'INCS:DHDR.INC'
      INCLUDE 'INCS:DTVD.INC'
      INCLUDE 'INCS:DCAT.INC'
```

```
      EQUIVALENCE (JROW, BUFF2),    (BUFF1, XBUFF)
      EQUIVALENCE (CATII, CATIR, BUFF1(513))
      DATA MAP, EXCL, WAIT, F / 3*.TRUE., .FALSE./
C-----------------------------------------------------------------------
      ICH = 1
      CALL TVOPEN (BUFF1, IERR)
      IF (IERR.NE.0) THEN
         WRITE (MSGTXT,1000) IERR
         CALL MSGWRT (6)
         GO TO 999
         END IF
      IF (TVPASS.LE.1) THEN
         CALL YZERO (ICH, IERR)
         IF (IERR.NE.0) THEN
            WRITE (MSGTXT,1010) IERR
            CALL MSGWRT (6)
            GO TO 998
            END IF
         CALL YCINIT (ICH, XBUFF)
         END IF
C                                       Set max/min for display
      IF (TVFMAX.LE.TVFMIN)
         TVFMAX = TVREMX
         TVFMIN = TVREMN
         END IF
      IF (TVREMX.GT.TVFMAX) TVFMAX = TVREMX
      IF (TVREMN.LT.TVFMIN) TVFMIN = TVREMN
      TVLMAX = TVFMAX - TVFMIN
      IF (0.1*TVLMAX.GT.TVREMX-TVREMN) THEN
         ARG = 0.1 * TVFMIN
         TVFMIN = MIN (ARG, TVREMN)
         ARG = TVFMIN + 0.1 * TVLMAX
         TVFMAX = MAX (ARG, TVREMX)
         TVLMAX = TVFMAX - TVFMIN
         END IF
      XN(1) = TVFMIN
      XN(2) = TVFMAX
      XN(3) = (MAXINT - 0.02) / TVLMAX
      XN(4) = 0.51 - TVFMIN * XN(3)
      CALL QPUT (XN, 0, 4, 2)
C                                       Write scaling factor
      XFLUX = TVLMAX
      CALL METSCA (XFLUX, PREFIX, LERR)
      TVLMIN = TVFMIN * XFLUX / TVLMAX
      TVLMAX = TVFMAX * XFLUX / TVLMAX
      WRITE (MSGTXT,1020) TVLMIN, TVLMAX, PREFIX
      CALL MSGWRT (1)
C                                       Set window to display
      WIN(1) = (WINM(3,1) + WINM(1,1)) / 2 - MAXXTV(1) / 2 + 1
      WIN(1) = MAX (1, WIN(1))
      WIN(2) = (WINM(4,1) + WINM(2,1)) / 2 - MAXXTV(2) / 2 + 1
      WIN(2) = MAX (1, WIN(2))
      WIN(3) = (WINM(3,1) + WINM(1,1)) / 2 + MAXXTV(1) / 2
      WIN(3) = MIN (NX, WIN(3))
```

```
      WIN(4) = (WINM(3,1) + WINM(1,1)) / 2 + MAXXTV(2) / 2
      WIN(4) = MIN (NY, WIN(4))
      DO 70 I = 1,2
         IWIN(I) = (MAXXTV(I) - WIN(I+2) + WIN(I) + 1)/2
         IF (IWIN(I).GE.1) GO TO 50
            IWIN(I) = 1
            WIN(I) = (WIN(I+2) + WIN(I) - MAXXTV(I) + 1)/2
            GO TO 60
 50      IWIN(I+2) = IWIN(I) + WIN(I+2) - WIN(I)
         IF (IWIN(I+2).LE.MAXXTV(I)) GO TO 70
 60      IWIN(I+2) = MAXXTV(I)
         WIN(I+2) = WIN(I) + IWIN(I+2) - IWIN(I)
 70   CONTINUE
C                                        Prepare to read map.
      CALL ZOPEN (LUNRES, FIND, RESVOL, RESFIL, MAP, EXCL, WAIT, IERR)
      CALL MINIT ('READ', LUNRES, FIND, NX, NY, WIN, XBUFF, BUFSZ1,
     *   BORES, IERR)
      IF (IERR.NE.0) GO TO 110
      MX = WIN(3) - WIN(1) + 1
      MY =  WIN(4) - WIN(2)+ 1
C                                        loop, passing map to TV.
      DO 100 I = 1,MY
         IY = I + IWIN(2) - 1
         CALL MDISK ('READ', LUNRES, FIND, XBUFF, BIND, IERR)
         IF (IERR.NE.0) GO TO 110
c                                        clip, sacle, and fix in Q routines
         CALL QPUT (XBUFF(BIND), 4, MX, 2)
         CALL QWD
         CALL QVCLIP (4, 1, 0, 1, 4, 1, MX)
         CALL QVSMSA (4, 1, 2, 3, 4, 1, MX)
         CALL QVFIX (4, 1, 4, 1, MX)
         CALL QWR
         CALL QGET (JROW, 4, MX, 1)
         CALL QWD
C                                        Send row to TV.
         MX2 = MX
         CALL YIMGIO ('WRIT', ICH, IWIN, IY, 0, MX2, JROW, IERR)
         IF (IERR.NE.0) GO TO 110
 100     CONTINUE
 110  CALL ZCLOSE (LUNRES, FIND, IERR)
C                                        Release the AP
      CALL QRLSE
C                                        Update image catalog
      CALL COPY (256, CATBLK, CATII)
      CATII(IIVOL) = 0
      CATII(IICNO) = 0
      CALL FILL (5, 1, CATII(IIDEP))
      CALL COPY (4, IWIN, CATII(IICOR))
      CALL COPY (4, WIN, CATII(IIWIN))
      CALL CHR2H (2, '  ', 1, CATII(IITRA))
      CATIR(IRRAN) = TVFMIN
      CATIR(IRRAN+1) = TVFMAX
      CATIR(KRDMN) = TVREMN
      CATIR(KRDMX) = TVREMX
```

```
      CALL YCWRIT (ICH, IWIN, CATII, XBUFF, IERR)
      IF (IERR.EQ.0) GO TO 120
         WRITE (MSGTXT,1110)
         CALL MSGWRT (6)
C                                       Ask user to quit?
 120  IF ((TVPASS.NE.1) .AND. (TVPASS.NE.2)) GO TO 998
         WRITE (MSGTXT,1120)
         CALL MSGWRT (1)
         WRITE (MSGTXT,1121)
         CALL MSGWRT (1)
         RPOS(1) = (WINDTV(1) + WINDTV(3)) / 2.0
         RPOS(2) = (WINDTV(2) + WINDTV(4)) / 2.0
         TD = 0.2
         CALL YCURSE ('ONNN', F, F, RPOS, IQ, IB, IERR)
         IF (IERR.NE.0) GO TO 998
         DO 130 I = 1,75
            CALL ZDELAY (TD, IERR)
            CALL YCURSE ('READ', F, F, RPOS, IQ, IB, IERR)
            IF (IB.GT.7) GO TO 140
            IF (IB.GT.0) GO TO 135
            IF (IERR.NE.0) GO TO 135
 130        CONTINUE
 135     WRITE (MSGTXT,1135)
         CALL MSGWRT (1)
         GO TO 150
C                                       Wants to quit
 140     TVPASS = 32700
         WRITE (MSGTXT,1140)
         CALL MSGWRT (3)
C                                       Off cursor
 150     CALL YCURSE ('OFFF', F, F, RPOS, IQ, IB, IERR)
 998  CALL TVCLOS (BUFF1, IERR)
C
 999  RETURN
C-----------------------------------------------------------------------
 1000 FORMAT ('CAN''T OPEN TV IER=',I6)
 1010 FORMAT ('IMCLEAR ERROR =',I6)
 1020 FORMAT ('TVDISP: Display range =',2F8.3,1X,A5,'Jy')
 1110 FORMAT ('CAN''T UPDATE IMAGE CATALOG  IER=',I6)
 1120 FORMAT ('Hit button D within 15 seconds to stop cleaning now')
 1121 FORMAT ('Hit buttons A, B, or C to continue sooner')
 1135 FORMAT ('Continuing')
 1140 FORMAT ('TV Button D hit: have done enough I guess')
      END
```

10.3.5 Plot Files (TVPL)

Plots in AIPS are usually produced as device-independent plot files (see the chapter on plotting). The task which interprets such files and writes on the TV display is called TVPL. It will scale line drawings to fill the TV screen or, at the user's option, plot them at the original pixel scaling (converted to TV pixels). Grey-scale plot files are always done at pixel scaling. The character and vector portions of the plot are written to one of the graphics planes (chosen by the user) via subroutines IMVECT and IMCHAR. Grey-scale records, if any, are written via YIMGIO to the user-specified grey-scale memory. TVPL also updates the image catalog as needed.

All plot tasks in AIPS now have the option to draw their plot directly on the TV display, forgoing the creation of the plot file. This has the advantage of simplicity and speed for the user at the cost of larger plot tasks and occasional repetitions needed to get the plot file for display on other devices.

10.3.6 Transfer Function Modification, Zooming

Subroutine AU6A carries out the verbs OFFTRAN, TVTRANSF, TVLUT, and TVMLUT which perform modifications on the black and white (or single color) LUTs of the specified grey-scale memories. OFFTRAN simply writes a linear, 0 through MAXINT array to the LUTs via YLUT. TVTRAN is implemented by the subroutine IENHNS which is also used by other verbs and tasks (e.g., TVFIDDLE, BLANK, TVMOVIE, TVBLINK). IENHNS allows a linear LUT with the cursor position controlling the slope and intercept and buttons allowing a switch in the sign of the slope and a continually updated plot of the LUT. TVLUT and TVMLUT allow the user to plot his own LUT function on a graphics plane with the cursor and the buttons. They both use the subroutine GRLUTS.

Subroutine AU6 implements the verbs OFFPSEUD, OFFZOOM, and OFFSCROL to clear the OFM, the zoom setting, and the scroll(s). It also implements interactive setting of the zoom factor and center (verb TVZOOM), of individual channel scrolls (TVSCROL), and of the pseudo-color OFM (TVPSEUDO). OFFPSEUD simply sends a linear OFM to all colors via YOFM; OFFZOOM sends a 0 zoom factor via YZOOMC; and OFFSCROL sends a 0 scroll via YSCROL. TVZOOM makes considerable use of YCURSE and YZOOMC, while TVPSEUDO uses YCURSE and alternately IMLCLR (RGB color triangle), IMPCLR (circle in hue), and IMCCLR (color contours). AU6 also implements a much more complicated enhancement algorithm in which one grey-scale channel is used to set the intensity and another to set the hue. This algorithm requires the TV to have an LUT for each color and channel and an OFM for the sum of the enabled channels for each color. A log function is put in the LUTs and an exponential in the OFM to carry out the required multiplication of the two signals. Subroutines HIENH and HILUT actually do most of the algorithm including interactive enhancements (via an algorithm similar to IENHNS) and switching of the roles of the two channels.

One of the most commonly used image enhancement routines is TVFIDL. It is called by the verb TVFIDDLE via subroutine AU6C and several tasks including BLANK and TVFLG. It is a deliberately limited interactive routine designed to provide easy-to-use enhancement in black and white (via IENHNS) or pseudo-color (via IMCCLR with a single type of color contour). A simple zoom procedure is also provided. During image enhancement, the cursor position controls slope and intercept and, during zoom, the cursor position controls zoom center. Button A (value 1) alternately selects color and black and white enhancement, button B/C increments/decrements the zoom and selects zoom mode. As in all interactive algorithms, button D (values ≥ 8) terminates the function.

The algorithm for TVSCROL is a good example to present in detail since the action required when the cursor moves is quite simple. The most important thing to notice below is the routine DLINTR. This routine tests the output of YCURSE to see if anything has changed. If not, it delays the program by some period of time which increases slowly as the time since the last change increases. Without this algorithm, the tight loop on reading the TV cursor is capable of jamming the CPU and I/O channels, especially when the user does not move the cursor. DLINTR also keeps the cursor from moving off the screen (wrapping around to the other side).

```
C                                         general parameters
      QUAD = -1
      RPOS(1) = 0.0
      RPOS(2) = 0.0
C                                        open TV device
      CALL TVOPEN (BUFFER, IERR)
C                                        get start time
      CALL ZTIME (ITW)
      IF (IERR.NE.0) THEN
         POTERR = 101
         GO TO 980
```

```
         END IF
......
C                                       TVSCROL
C                                       user instructions
 500  WRITE (MSGTXT,1500)
      CALL MSGWRT (1)
      WRITE (MSGTXT,1505)
      CALL MSGWRT (1)
C                                       find channel(s) to scroll
C                                       scroll graphics too ?
      IC = ABS(TVCHAN) + EPS
      CALL DECBIT (NGRAY, IC, IC, J)
      ITEMP = 2 ** NGRAY
      IF (ABS(GRCHAN).GT.EPS) IC = IOR (IC, ITEMP)
      IF (IC.EQ.0) THEN
         IC = MOD (TVLIMG(1), ITEMP)
         IF (IC.NE.TVLIMG(1)) IC = IOR (IC, ITEMP)
         END IF
      IX = 0
      IY = 0
      RPOS(1) = (WINDTV(1) + WINDTV(3)) / 2
      RPOS(2) = (WINDTV(2) + WINDTV(4)) / 2
C                                       turn on cursor
      CALL YCURSE ('ONNN', F, F, RPOS, QUAD, IBUT, IERR)
      IF (IERR.NE.0) GO TO 900
C                                       force scroll
 510  CALL YSCROL (IC, IX, IY, T, IERR)
      IF (IERR.NE.0) GO TO 900
         PPOS(1) = RPOS(1)
         PPOS(2) = RPOS(2)
C                                       read until cursor moves
 520     CALL YCURSE ('READ', F, F, RPOS, QUAD, IBUT, IERR)
         IF (IERR.NE.0) GO TO 900
C                                       test for change
         CALL DLINTR (RPOS, IBUT, F, QUAD, PPOS, ITW, DOIT)
         IF (.NOT.DOIT) GO TO 520
C                                       cursor moved, change scroll
            IX = RPOS(1) - (WINDTV(1) + WINDTV(3)) / 2
            IY = RPOS(2) - (WINDTV(2) + WINDTV(4)) / 2
C                                       any button => done
            IF (IBUT.EQ.0) GO TO 510
               POTERR = 0
               GO TO 900
......
C                                       close down
C                                       cursor off, TV closed
 900  IF (BRANCH.GE.4) CALL YCURSE ('OFFF', F, F, RPOS, QUAD, IBUT,
     *   JERR)
 910  CALL TVCLOS (BUFFER, JERR)
```

10.3.7 Object location, window setting

Subroutine AU5 performs the verbs TVPOS, IMXY, IMPOS (see below), and TVNAME (via TVFIND) as well as a variety of status setting verbs. IMPOS is implemented as follows. It calls TVWHER to find the cursor position indicated by the user. Then it checks all enabled memories via YCREAD to see if there is an image displayed at that pixel position. Finally, it calls MP2SKY to set up the coordinate commons and get the primary positions and goes through some other messy stuff to display the results to the user.

```
      CALL TVOPEN (CATBLK, JERR)
      IF (JERR.NE.0) THEN
         POTERR = 101
         GO TO 980
         END IF
......
C                                       IMPOS
C                                       read cursor to get position
 600  CALL TVWHER (IPL, RPOS, IBUT, JERR)
      IF (JERR.NE.0) GO TO 975
C                                       Set output adverb = button #
      TVBUTT = IBUT
......
C                                       image pix -> map pixel pos
 625  IX = RPOS(1) + EPS
      IY = RPOS(2) + EPS
C                                       Find lowest plane with x,y
      IN2 = NGRAY + NGRAPH
      DO 630 IP = 1,IN2
C                                       skip off channels
         ITEMP = 2 ** (IP-1)
         IF (IAND (TVLIMG(IQUAD), ITEMP).EQ.0) GO TO 630
C                                       get image cat block
         CALL YCREAD (IP, IX, IY, CATBLK, IERR)
C                                       loop if x,y not in image
         IF (IERR.EQ.1) GO TO 630
         IF (IERR.EQ.0)  GO TO 650
         GO TO 975
 630     CONTINUE
C                                       x,y not in on image
      WRITE (MSGTXT,1630) IX, IY
      CALL MSGWRT (6)
      GO TO 900
C                                       image -> map positions
 650  CALL IMA2MP (RPOS, RPOS)
      WRITE (MSGTXT,1650) RPOS
      CALL MSGWRT (5)
......
C                                       map -> sky positions
 660  CONTINUE
         CALL MP2SKY (RPOS, SKYPOS, IERR)
         IF (IERR.NE.0) THEN
            WRITE (MSGTXT,1660) IERR
            CALL MSGWRT (6)
            GO TO 900
            END IF
C                                       3rd axis pairs w 1st or 2nd
         IF ((AXTYP.EQ.2) .OR. (AXTYP.EQ.3)) CALL AXSTRN (CTYP(3),
     *      SKYPOS(3), KLOCA, NCHLAB(1), SAXLAB(1))
C                                       Primary axes
```

```
C                                       Tell user results via MSGWRT.
         WRITE (MSGTXT,1661)
         ICH = 8
         DO 665 I = 1,2
            ITEMP1 = I-1
            CALL AXSTRN (CTYP(1,I), SKYPOS(I), ITEMP1, ILEN, RSTR)
            MSGTXT(ICH:ICH+ILEN-1) = RSTR(1:ILEN)
            ICH = ICH + ILEN + 2
 665        CONTINUE
         CALL MSGWRT (5)
C                                       Secondary axes values
         IF ((NCHLAB(1).LE.0) .AND. (NCHLAB(2).LE.0)) GO TO 900
            ICH = 8
            MSGTXT(ICH:) = ' '
            DO 670 I = 1,2
               IF (NCHLAB(I).LE.0) GO TO 670
                  MSGTXT(ICH:ICH+NCHLAB(I)-1) = SAXLAB(I)(1:NCHLAB(I))
                  ICH = ICH + NCHLAB(I) + 2
 670           CONTINUE
            CALL MSGWRT (5)
            GO TO 900
......
C                                       normal TV close
 900  CALL TVCLOS (CATBLK, JERR)
```

The interactive window setting verbs TVWIN, TVBOX, TVSLICE, and REBOX are initiated from subroutine AU5C and performed primarily by subroutine GRBOXS. This routine is another instance of interactivity via YCURSE and line drawing via IMVECT. It uses YCUCOR at the end to obtain the image catalog header and thence, to correct the cursor positions to map pixel locations. CURVALUE is an interactive verb which displays on a TV graphics channel the position and image value of the pixel currently under the TV cursor. It is implemented by subroutine AU6B. The image values are read from the original map files on disk, if possible, using MAPOPN, MINIT, and MDISK. However, the intensities of step wedges and temporary images (i.e., intermediate residual maps displayed by APCLN) are read from the TV memory via YIMGIO. The routine makes extensive use of IMCHAR and, although too long to reproduce here, is an interesting example of AIPS image plus TV coding.

10.3.8 Blotch Setting, Use

A "blotch" is a region within an image over which some action is to be performed. Pixels outside the blotch are ignored or have some alternative action performed on them. At present, AIPS has three functions which generate and use blotches: the verb TVSTAT which returns image statistics within the blotch area, the task BLANK which blanks out all pixels outside the blotch, and the task BLSUM which sums an image and every plane of a second image over the blotch areas. In all three, the user uses the TV cursor to set the vertices of one or more polygonal areas and the routines draw lines on a graphics plane between the vertices. When the user is done, the routines fill in the blotch areas on the TV graphics and then read and act on the map file. Subroutine AU6D implements TVSTAT for whatever image is visible on the TV, obtaining the polygons through subroutine GRPOLY. AU6D itself does the data reading, determination of whether a pixel is inside or outside the blotch, and the computation and display of the image statistics. Task BLANK uses internal subroutines BLNKTV and BLKTVF to display the image (via TVLOAD), allow transfer modification (via TVFIDL), to obtain the polygons (BLKTVF), and to use them to blank the output image (BLNKTV). BLSUM is similar to BLANK in the creation and handling of the blotch regions. The subroutine BLTFIL does the filling of the polygons on the TV graphics screen for TVSTAT, BLANK and BLSUM.

10.3.9 Roam

Roam is a mode of display which requires multiple grey-scale memories and the capability to do split screen and scroll. Adjacent portions of the image are loaded into separate image memories. Then the screen is split horizontally and/or vertically and the appropriate memories are enabled in each quadrant, each with the same scroll. This allows the user to view a screen-size portion of a rather larger image. By shifting the scroll and split point interactively, the user may select which portion is viewed. Roam is implemented in AIPS from the subroutine AU5A. This routine loads the image to the TV memories in a manner similar to TVLOD (above). However, it uses TVWIND to determine a much more complicated window and must itself play with windows further before calling TVLOAD. The interactive portion of the Roam is carried out by AU5A calling subroutine TVROAM. That routine can handle images of up to 1 x 4, 4 x 1, or 2 x 2 planes and uses YCURSE for interactive input, YSCROL to set the scroll (identical for all planes), and YSPLIT to set the split point and enable the appropriate channels. A zoom option is also available.

10.3.10 Movie, Blink

The verbs TVMOVIE and TVCUBE use a very interesting algorithm implemented via subroutines AU5D and TVMOVI. A movie is a method of displaying a 3-dimensional image as a time sequence of 2-dimensional planes. Each grey-scale TV memory is subdivided into an n x n matrix of images of consecutive planes of the cube (where the allowed n are the available zoom factors). During the display phase, the zoom factor is set to n, so that only one plane is visible at a time. The zoom center is moved from frame to frame at a user controlled rate to simulate a movie. Subroutine AU5D determines which zoom factor and windows to use, zeros the grey-scale memories, loads the planes to the TV (via TVLOAD), transfers the LUT of the first TV memory to the other TV memories, draws border lines around each plane (via IMVECT), annotates each plane with the 3rd coordinate axis value, and puts a small pointer in the image as well. TVMOVI executes an interactive algorithm in which the cursor controls the frame rate and the buttons allow a single frame at a time mode and interactive enhancement of the LUTs (via IENHNS) or the OFM (via IMCCLR). The verb REMOVIE is also done by AU5D and TVMOVI using the stored parameters which describe how the movie was loaded to the TV memories (parameter TYPMOV in the /DTVC.INC/ common). TVCUBE and TVMOVIE differ in the placement of the sub-images on the TV screen. The former orders the images from left to right, top to bottom, an order which is good for viewing all together, without zoom. The latter adopts a funny order designed to minimize changes between frames in movie mode, making a smoother switch between frames.

The subroutines AU6A and TVBLNK implement the verbs TVBLINK and TVMBLINK. Blinking is simply enabling one grey-scale memory for a while, then disabling it and enabling another for a second period of time, then disabling the second channel and re-enabling the first, and so on. These two verbs allow manual as well as timed switching between the two planes and transfer function modification via the subroutine IENHNS (see above). Image comparison may also be done by comparing images from different image memories in halves or quarters of the screen. The verb TVSPLIT implements an interactive algorithm in subroutine AU6A.

10.3.11 Tasks

There are three tasks which use a wide range of television functions in order to edit visibility data under interactive control of the user. All of these implement a menu written to a TV graphics screen to allow the user to select among many separate functions. All of the standard TVFIDDLE and TVTRANS enhancements are present. For window setting, an especially friendly version of GRBOXS is coded into the tasks. These tasks are:

1. TVFLG grids the data in baseline-time order displaying the visibility amplitude, phase, or amplitude rms as grey levels, with baseline on the horizontal axis and time in a semi-regular grid on the vertical axis. The user may select which IF, Stokes, or channel to display. The cursor is used to select the data to be flagged, as well as to make selections of functions to perform and enhancements.

2. SPFLG is similar to TVFLG, but grids the data with spectral channel on the X axis instead of baseline number. The user may select which baseline is displayed.

3. IBLED is also an interactive editor for baseline-oriented data. It uses the TV mostly as a graphics device to plot the data from a user-selected baseline. Time is on the horizontal axis, and amplitude is on the vertical axis. The cursor is used to select the data to be flagged and to do the selection of operations.

10.3.12 Non-Standard Tasks

There are a number of tasks in AIPS which are seriously non-standard in their coding and in their use of various devices. Among these are several which use the TV display. We will list them here briefly. Programmers should not use these tasks as models of how to code in AIPS and should not assume that they can even be made to run on non-VMS, non-IIS systems.

1. IMLHS uses up to 3 maps to create a false color image on the TV. It uses the first map to modulate the brightness of the image, the 2nd to modulate the hue and the 3rd to modulate the saturation. If any of the images are omitted the corresponding parameter is set to a constant. (Note: verb TVHUEINT is standard and does a similar function with two images.)

2. TVHLD loads up to 13-bit image to two TV memories and performs an interactive histogram equalization of the display. Can feed the result back to a 3rd TV memory. This task uses YRHIST, YALUCT, YFDBCK, YIFM, and the dual-channel mode of the IIS and will be hard to implement on TV display devices other than the IIS.

3. TVHXF does an interactive histogram equalization of the image which is currently displayed. This task uses YRHIST which is currently IIS specific. However, a TV-independent (but SLOW) YRHIST can be coded if someone wishes to do the work.

10.4 Includes

10.4.1 DTVC.INC

```
C                                                           Include DTVC.
      INTEGER   NGRAY, NGRAPH, NIMAGE, MAXXTV(2), MAXINT, LUTOUT,
     *   OFMINP, OFMOUT, SCXINC, SCYINC, MXZOOM, CSIZTV(2), TYPSPL,
     *   TVALUS, TVXMOD, TVYMOD, ISUNUM,
     *   TVDUMS(9),
     *   TVZOOM(3), TVSCRX(16), TVSCRY(16), TVLIMG(4), TVSPLT(2),
     *   TVSPLM, TVSPLC, TYPMOV(16), WINDTV(4), TVDUM2(4), YBUFF(160)
      REAL      TVGAMA
      COMMON /TVCHAR/ NGRAY, NGRAPH, NIMAGE, MAXXTV, MAXINT, LUTOUT,
     *   OFMINP, OFMOUT, SCXINC, SCYINC, MXZOOM, CSIZTV, TYPSPL,
     *   TVALUS, TVXMOD, TVYMOD, ISUNUM, TVGAMA,             TVDUMS,
     *   TVZOOM, TVSCRX, TVSCRY, TVLIMG, TVSPLT, TVSPLM, TVSPLC,
     *   TYPMOV, WINDTV, TVDUM2, YBUFF
C                                                           End DTVC.
```

10.4.2 DTVD.INC

```
C                                                           Include DTVD.
      INTEGER   TVLUN, TVIND, TVLUN2, TVIND2, TVBFNO
      LOGICAL   TVMAP
      COMMON /TVDEV/ TVMAP, TVLUN, TVIND, TVLUN2, TVIND2, TVBFNO
C                                                           End DTVD.
```

10.4.3 YDEA.INC

```
C                                                Begin YTPARM.INC
C               Parameter definitions specifying positions
C               in YBUFF (c.f. DTVC.INC) which correspond
C               to various registers in a DeAnza TV device.
      INTEGER   INTREG, IRBYTE, MEMREG, MRBYTE, VOCREG, VCBYTE,
     *          FCR,    FCBYTE, CURREG, CRBYTE, CHANOW, CHBYTE,
     *          GRPHCL, GRBYTE
      PARAMETER (INTREG = 1,          IRBYTE = 48,
     *           MEMREG = INTREG+24,  MRBYTE = 8,
     *           VOCREG = MEMREG+64 , VCBYTE = 8,
     *           FCR    = VOCREG+4 ,  FCBYTE = 16,
     *           CURREG = FCR   +8 ,  CRBYTE = 14,
     *           CHANOW = CURREG+7 ,  CHBYTE = 24,
     *           GRPHCL = CHANOW+12,  GRBYTE = 16)
C                                        Positions in INTERFACE REGISTERS
C                                        Relative to CONTROL REGIST (#10)
      INTEGER   CONTRL, RES, FG, BG, XR, XT, XMIN, XMAX,
     *          XAMIN, XAMAX, DX, XTEMP, YR, YT,   YMIN,
     *          YMAX, YAMIN, YAMAX, DY, YTEMP, CMR0, CMR0A
      PARAMETER (CONTRL=INTREG,   RES=INTREG+1,    FG=INTREG+2,
     *           BG=INTREG+3  ,   XR= INTREG+6,    XT=INTREG+7,
     *           XMIN=INTREG+8,   XMAX=INTREG+9,   XAMIN=INTREG+10,
     *           XAMAX=INTREG+11, DX=INTREG+12,    XTEMP=INTREG+13,
     *           YR=INTREG+14,    YT=INTREG+15,    YMIN=INTREG+16,
     *           YMAX=INTREG+17,  YAMIN=INTREG+18, YAMAX=INTREG+19,
     *           DY=INTREG+20,    YTEMP=INTREG+21, CMR0=INTREG+22,
     *           CMR0A=INTREG+23)
C                                        Positions in MEMORY REGISTERS
      INTEGER   XSCZ, YSCZ, BITPL, MEMLUT
      PARAMETER (XSCZ=MEMREG,    YSCZ = MEMREG+1,
     *           BITPL=MEMREG+2, MEMLUT=MEMREG+3)
C                                        Positions in VOC REGISTERS
      INTEGER   XVSP, YVSP, VOCCON, VOCLUT
      PARAMETER (XVSP=VOCREG,     YVSP=VOCREG+1,
     *           VOCCON=VOCREG+2, VOCLUT=VOCREG+3)
C                                        Positions in Cursor registers
      INTEGER   CURX1, CURY1, CURX2, CURY2, CURCON, CURLUT, CURBLI
      PARAMETER (CURX1=CURREG,   CURY1=CURREG+1,   CURX2=CURREG+2,
     *           CURY2=CURREG+3, CURCON=CURREG+4,  CURLUT=CURREG+5,
     *           CURBLI=CURREG+6)
```

10.5 Y-Routine Precursor Remarks

10.5.1 Level 0

YCHRW

writes characters into image planes of the TV. The format is 5 x 7 or a multiple thereof with one or more blanks all around. The net is set to match CSIZTV if possible. We recommend CSIZTV = 7, 9 for TVs of size around 512 square and CSIZTV = 14, 22 or so for TVs of size around 1024 square. This version will work on all TVs which allow horizontal writing to the right. It is a Y routine to allow for hardware character generators on the TV.

```
YCHRW (CHAN, X, Y, STRING, SCRTCH, IERR)
Inputs:
   CHAN    I       channel select  (1 to NGRAY + NGRAPH)
   X       I       X position lower left corner first char.
   Y       I       Y position lower left corner first char.
   STRING  C*(*)   character string - length passed from Fortran
Output:
   SCRTCH  I(>)    scratch buffer (dim = 14*count+8 < MAXXTV(1))
   IERR    I       error code of Z...XF:0 - ok
                                         2 - input error
```

YCINIT

Initialize image catalog for plane IPLANE - TK now done with TKCATL

```
YCINIT (IPLANE, BUFF)
Input:
   IPLANE   I        Image plane to initialize
Output:
   BUFF     I(256)   Working buffer
```

YCNECT

Writes a line segment on the TV. This version will work on all TVs. It is called a Y routine to allow the use of hardware vector generators on those TVs equiped with them.

```
YCNECT (X1, Y1, X2, Y2, IC, BUFFER, IERR)
Inputs:
   X1      I      start X position
   Y1      I      start Y position
   X2      I      end X position
   Y2      I      end Y position
   IC      I      Channel (1 to NGRAY+NGRAPH)
   BUFFER  I(*)   Buffer contains desired intensity (size > max
                  horizontal or vertical line, e.g., 1280)
Output:
   IERR    I      error code : 0 => ok
```

YCOVER

Checks to see if there are partially replaced images in any of the TV planes currently visible by quadrant

```
YCOVER (OVER, BUF, IERR)
Outputs:
   OVER   L(4)     T => there are in quadr. I
   BUF    I(512)   scratch
   IERR   I        Error code: 0 => ok, other catlg IO error
```

YCREAD

Read image catalog block into CATBLK - TV only (TK in TKCATL)

```
YCREAD (IPLANE, IX, IY, CATBLK ,IERR)
Inputs:
   IPLANE  I       plane containing image whose block is wanted
   IX      I       X pixel coordinate of a point within image
   IY      I       Y pixel coordinate of point within image
Outputs:
   CATBLK  I(256)  Image catalog block
   IERR    I       error codes: 0 => ok
                      1 => IX, IY lies outside image
                      2 => Catalog i/o errors
                      3 => refers to TK device
```

YCUCOR

Takes a cursor position (corrected for zoom, but not scroll) corrects it for scroll, determines the quadrant of the TV, and gets the corresponding image header in common /MAPHDR/ and returns the image coordinates. NOTE WELL: RPOS ON INPUT MUST BE CORRECTED FOR ZOOM AND NOT SCROLL. To get this from a raw TV position call YCURSE with OPCODE 'FXIT' (or, of course, 'READ') and quadrant set to -1. *************

```
YCUCOR (RPOS, QUAD, CORN, IERR)
Inputs:
   RPOS    R(2)    X,Y screen pos before  zoom & scroll
Output:
   QUAD    I       TV quadrant to use for scrolls
                   Out: if in=-1, no scroll, else find quadrant
                        (needs real TV pos)
   CORN    R(7)    Image coordinates (pixels)
   IERR    I       error code of Z...XF : 0 - ok
                                          2 - input error
```

YCURSE

Reads cursor positions and controls the blink and visibility of the TV cursor.

```
YCURSE (OP, WAIT, CORR, RPOS, QUAD, BUTTON, IERR)
Inputs:
   OP      C*4     'READ' read cursor position
                   'ONNN' place cursor at RPOS & leave on
                   'OFFF' turn cursor off
                   'BLNK' reverse sense of cursor blink
                   'FXIT' fix RPOS for zoom scroll, no IO
   WAIT    L       wait for event; then return RPOS & BUTTON
                   (done on all OPs)
   CORR    L       T => correct RPOS for zoom & scroll
In/Out:
   RPOS    R(2)    X,Y screen pos before  zoom & scroll
   QUAD    I       TV quadrant to use for scrolls
                      In: if <1 >4, no scroll
```

```
                       Out: if in=-1, no scroll, else find
                            quadrant (needs real TV pos)
   Output:
      BUTTON  I       event # (0 none, 1-7 low buttons,
                      8-15 the "quit" button)
      IERR    I       error code of Z...XF : 0 - ok
                                             2 - input error
```

YCWRIT

Write image catalog block in CATBLK into image catalog

```
   YCWRIT (IPLANE, IMAWIN, CATBLK, BUFF, IERR)
   Inputs:
      IPLANE   I        image plane involved
      IMAWIN   I(4)     Corners of image on screen
      CATBLK   I(256)   Image catalog block
  Outputs:
      BUFF     I(256)   working buffer
      IERR     I        error code: 0 => ok
                           1 => no room in catalog
                           2 => IO problems
```

YFILL

Will write a constant in a given rectangle in a given graphics or image plane. It will use fast methods if full screen requested with IVAL 0.

```
   YFILL (CHAN, IX0, IY0, IXT, IYT, IVAL, IBLK, IERR)
   Inputs:
      CHAN     I   Channel (1 to NGRAY+NGRAPH)
      IX0      I   lower left X pixel (1 relative) of rectangle
      IY0      I   lower left Y pixel of rectangle.
      IXT      I   top right X pixel of rectangle.
      IYT      I   top right Y pixel of rectangle.
      IVAL     I   desired value: for graphics = 0 or 1
                                  for grey scale = 0 - MAXINT
   In/out:
      IBLK     I(IXT-IX0+1)   work buffer.
   Output:
      IERR     I   error code of Z...XF: 0 ok, 2 input error
```

YFIND

Determines which of the visible TV images the user wishes to select. The TV must already be open.

```
   YFIND (MAXPL, TYPE, IPL, UNIQUE, CATBLK, SCRTCH, IERR)
   Inputs:
      MAXPL   I        Highest plane number allowed (i.e. do graphics
                       count?)
      TYPE    C*2      2-char image type to restrict search
```

```
Output:
   IPL     I        Plane number found
   UNIQUE  L        T => only one image visible now
                        (all types except zeroed ones ('ZZ'))
   CATBLK  I(256)   Image catalog block found
   SCRTCH  I(256)   Scratch buffer
   IERR    I        Error code: 0 => ok
                       1 => no image
                       2 => IO error in image catalog
                       3 => TV error
                      10 => > 1 image of requested type
```

YLOCAT

Locates a set of image pixel positions on the TV for a specified image using only those grey planes that are turned on.

```
YLOCAT (NP, XP, YP, NAME, CLASS, SEQ, DISK, PTYP, IX,
  *   IY, IQ, SCRTCH, IERR)
Inputs:
   NP      I         Number of pixel positions
   XP      R(NP)     X image pixel positions
   YP      R(NP)     Y image pixel positions
   NAME    C*12      image name (name) '    ' => any
   CLASS   C*6       image name (class) '    ' => any
   SEQ     I         Image name (sequence #)  0 => any
   DISK    I         Image file disk 0 => any
   PTYP    C*2       Image type  '  ' => any
Output:
   IX      I(NP)     TV x positions
   IY      I(NP)     TV y positions
   IQ      I(NP)     TV channels  0 => none this position
   IERR    I         Error code: 0 -> ok
                        2 -> input error
                        3 -> IO error
                       11 -> some positions bad
                       12 -> no positions found
Uses common /MAPHDR/ results unpredictable except on IERR = 0
then = image catlg header of last position found.
```

YLOWON

Returns a zero-relative least bit on number up to NGRAY. If MASK=0, it returns 0. If MASK $> 2 * (NGRAY - 1)$ it returns NGRAY. Since some TVs can't have more than one channel on at a time in each color, determine the lowest channel that is flagged ON in MASK, and return it (zero relative) in ICHAN. If MASK is clear, set ICHAN to be a 0.

```
YLOWON (MASK, ICHAN)
Inputs:
   MASK    I   IIS type channel mask
Output:
   ICHAN   I   # of lowest bit set in MASK
```

YSLECT

Enables and disables gray and graphics planes

```
YSLECT (OP, CHAN, COLOR, BUFFER, IERR)
Inputs:
   OP       C*4      'ONNN' or 'OFFF'
   CHAN     I        channel number ( 1 to NGRAY+NGRAPH)
   COLOR    I        0 - all,   1,2,3 = R,G,B, resp.
Output:
   BUFFER   I(256)   scratch buffer (for graphics only)
   IERR     I        error code of Z...XF: 0 - ok, 2 - input error
YSLECT sets TVLIMG in the TV device parms common /TVDEV/
```

YTCOMP

Check whether a value in a soft register has to be changed. If so, change it and set a flag indicating that the hard register must be updated too.

```
YTCOMP (OLD, NEW, UPDATE)
Inputs:
   OLD      I   Current value in soft register
   NEW      I   New value
Output:
   OLD      I   Put new value here too if necessary
   UPDATE   L   Set true if update is needed, else leave alone
```

YTVCLS

Closes the TV device and the TV status disk file, updating the information on the disk. Actual device call done by YTVCL2.

```
YTVCLS (BUF, IERR)
Outputs:
   BUF   I(256)     Scratch buffer
   IERR  I          Error code : 0 => ok
                       else as returned by ZFIO
                       11 => close disk error
                       12 => close device error
```

YWINDO

Reads the current viewport into the TV memory. It is hoped that someday we will also offer writing to force the size.

```
YWINDO (OPER, WIND, IERR)
Inputs:
   OPER   C*4    'READ', 'WRIT'
In/out:
   WIND   I(4)   BLC x,y, TRC x,y of window in TV pixels
                 In: desired window ('WRIT' only)
                 Out: actual window given
```

```
Output:
   IERR    I     error code of Z...XF: 0 -> ok, 2 -> input error
                 -1 => on WRIT, device not windowing type
Generic version - returns full memory size always and IERR = -1
on WRIT.
```

YTVOPN

Opens the TV including TV lock/parameter file and reads the parameters, placing them in commons.

```
YTVOPN (BUF, IERR)
Outputs:
   BUF   I(256)   Scratch buffer
   IERR  I        Error return from YTVOP2
                     = 10 TV unavailable to this version
```

10.5.2 Level 1

YCRCTL

Reads/writes the cursor/trackball control register of TV

```
YCRCTL (OP, ON, X, Y, LINKX, LINKY, RBLINK, BUTTON,
  *   VRTRTC, IERR)
Inputs:
   OP      C*4   'READ' from TV or 'WRIT' to TV
   VRTRTC  L     T => do on vertical retrace only
In/out:
   ON      L     T => cursor visible, F => off
   X       I     X position cursor center (1-512, 1 => LHS)
   Y       I     Y position cursor center (1-512, 1 => bot)
   LINKX   L     T => trackball moves cursor in X
   LINKY   L     T => trackball moves cursor in Y
   RBLINK  I     rate of cursor blink: 0-3 no-fast blink
Output:
   BUTTON  I     button value (0 - 15)
   IERR    I     error code of Z...XF : 0 => ok
                                        2 => input error
```

YGRAFX

Is used to modify the color of the graphics or cursor.

```
YGRAFX (OP, GRFX, RED, GREEN, BLUE, IERR)
Inputs:
   OP       C*4   READ or WRIT graphics color assignment.
   GRFX     I     Cursor (0), or graphics plane number.
In/Out:
   RED      R     Red,   0.0 - 1.0
   GREEN    R     Green, 0.0 - 1.0
   BLUE     R     Blue,  0.0 - 1.0
Output:
```

```
    IERR     I     Error status, 0 means success.
```

YGRAPH

Is used to turn graphics overlay planes on and off by altering the graphics color look up table. The color pattern is:

```
CHAN = 1    insert   yellow             drawing plots
       2    insert   green+.05 red      axis labels
       3    insert   blue + 0.6 green   blotch
                     + red
       4    insert   black              label backgrounds
       5-7  add      nothing            null channels
       8    insert   purple             cursor
by default.  It may be altered by YGRAFX.

YGRAPH (OP, CHAN, SCRTCH, IERR)
Inputs:
   OP      C*4       'ONNN' or 'OFFF',
                     'INIT' sets default colors in common
   CHAN    I         channel number (1 - 8)
Output:
   SCRTCH  I(256)    scratch buffer
   IERR    I         error code of Z...XF: 0 => ok
                                           2 => input error
```

YIMGIO

Reads/writes a line of image data to the TV screen. For graphics overlay planes, the data are solely 0's and 1's in the least significant bit of IMAGE after a READ. For 'WRIT', all bits of each word should be equal (i.e. all 1's or all 0's for graphics). NOTE***** on 'WRIT', the buffer may be altered by this routine for some IANGLs.

```
YIMGIO (OP, CHAN, X, Y, IANGL, NPIX, IMAGE, IERR)
Inputs:
   OP      C*4       'READ' from TV or 'WRIT' to TV
   CHAN    I         channel number (1 to NGRAY+NGRAPH)
   X       I         start pixel position
   Y       I         end pixel position
   IANGL   I         = 0 => horizontal (to right)
                     = 1 => vertical (up the screen)
                     = 2 => horizontal (to left)
                     = 3 => vertical (down the screen)
   NPIX    I         number of pixels
In/Out:
   IMAGE   I(NPIX)   data (only no header)
Output:
   IERR    I         error code of Z...XF - 0 => ok
                                             2 => input err
```

YINIT

Initializes the TV subunits: doing everything

```
YINIT (SCRTCH, IERR)
Output:
   SCRTCH  I(1024)    scratch buffer
   IERR    I          error code of Z...XF - 0 => ok
                                             2 => input error
```

YLUT

Reads/writes full channel look up tables to TV.

```
YLUT (OP, CHANNL, COLOR, VRTRTC, LUT, IERR)
Inputs:
   OP      C*4    'READ' from TV, 'WRIT' to TV
   CHANNL  I      channel select bit mask
   COLOR   I      color select bit mask (RGB <-> 421)
   VRTRTC  L      T => do it only during vertical retrace
In/Out:
   LUT     I(*)   look up table (dimension = MAXINT+1, values to
                  LUTOUT are used)
Out:
   IERR    I      error code of Z...XF : 0 => ok, 2 => input error
```

YOFM

Reads/writes full OFM look up tables to TV.

```
YOFM (OP, COLOR, VRTRTC, OFM, IERR)
Inputs:
   OP      C*4    'READ' from TV, 'WRIT' to TV
   COLOR   I      color select bit mask (RGB <-> 421)
   VRTRTC  L      T => do it only during vertical retrace
In/Out:
   OFM     I(*)   look up table (dimension = OFMINP+1, values to
                  OFMOUT used)
Output:
   IERR    I      error code of Z...XF : 0 => ok, 2 => input error
```

YSCROL

Writes the scroll registers on the TV.

```
YSCROL (CHANNL, SCROLX, SCROLY, VRTRTC, IERR)
Inputs:
   CHANNL   I   bit map channel select: bits 1-NGRAY gray channels,
                bit NGRAY+1 => all graphics
   VRTRTC   L   T => do it on vertical retrace only
In/Out:
   SCROLX   I   amount of X scroll (>0 to right)
```

```
   SCROLY   I   amount of Y scroll (>0 upwards)
Output:
   IERR     I   error from Z...XF : 0 => ok, 2 => input error
YSCROL updates the scroll variables in /TVDEV/ common
```

YSPLIT

Reads/writes the look up table/ split screen control registers of the TV - turns channels on/off by quadrant

```
YSPLIT (OP, XSPLT , YSPLT , RCHANS, GCHANS, BCHANS,
  *   VRTRTC, IERR)
Inputs:
   OP       C*4    'READ' from TV, 'WRIT' to TV
   VRTRTC   L      T => do on vertical retrace only
In/Out:
   XSPLT    I      X position of split (1-512, 1 => LHS)
   YSPLT    I      Y position of split (1-512, 1 => bot)
   RCHANS   I(4)   chan select bit mask 4 quadrants : red
   GCHANS   I(4)   chan select bit mask 4 quadrants : green
   BCHANS   I(4)   chan select bit mask 4 quadrants : blue
Output:
   IERR     I      error code of Z...XF: 0 => ok, 2 => input error
Quadrants are numbered CCW from top right!!!!
```

YTVCIN

Initializes the common which describes the characteristics of the interactive display devices and the common which has the current status parameters of the TV.
NOTE: These are default values only. They are reset to the current true values by a call to TVOPEN.
NOTE: YTVCIN resets the common values of TVZOOM and TVscroll, but does not call the TV routines to force these to be true. A separate call to YINIT or YZOOMC and YSCROL is needed.

```
   YTVCIN
(no arguments)
```

YTVCL2

Closes TV DEVICE associated with LUN removing any EXCLusive use state and clears up the FTAB.

```
YTVCL2 (LUN, IND, IERR)
Inputs:
   LUN    I   logical unit number
   IND    I   pointer into FTAB
Output:
   IERR   I   error code: 0 -> no error
                 1 -> Deaccess or Deassign error
                 2 -> file already closed in FTAB
                 3 -> both errors
                 4 -> erroneous LUN
```

YTVMC

Issues a "master clear" to the TV. This resets the TV I/O system (if necessary) to expect a command record next. YTVMC gets all needed parameters from the TV device common. The TV must already be open.

```
   YTVMC
(no arguments)
```

YTVOP2

Performs a system "OPEN" on the TV device. It is a Y routine in order to call the appropriate Z routine only.

```
YTVOP2 (LUN, IND, IERR)
Inputs:
   LUN    I   Logical unit number to use
Output:
   IND    I   Pointer to FTAB entry for open device
   IERR   I   Error code: 0 => ok
                 1 = LUN already in use
                 2 = file not found
                 3 = volume not found
                 4 = excl requested but not available
                 5 = no room for lun
                 6 = other open errors
```

YZERO

Fills an TV TV memory plane with zeros the fast way.

```
YZERO (CHAN, IERR)
Inputs:
   CHAN   I     channel # ( 1 - NGRAY+NGRAPH), 0 => all
Outputs:
   IERR   I     error code of Z...XF: 0 - ok, 2 - input error
```

YZOOMC

Writes (ONLY!!!!) the zoom control registers of the TV

```
YZOOMC (MAG, XZOOM, YZOOM, VRTRTC, IERR)
Inputs:
   MAG     I     0-3 for magnification 1,2,4,8 times, resp.
   XZOOM   I     X center of expansion (1-512, 1 => LHS)
   YZOOM   I     Y center of expansion (1-512, 1 => bot)
   VRTRTC  L     Do on vertical retrace only?
Output:
   IERR    I     error code of Z...XF: 0 -> ok, 2 -> input error
YZOOMC updates the /TVDEV/ common TVZOOM parameter
```

10.5.3 Level 2 (Used as level 1 in non-standard tasks)

YALUCT

Reads / writes the TV arithmetic logic unit control registers. The actual feedback-ALU computation is performed only upon a call to YFDBCK.

```
YALUCT (OP, ARMODE, BFUNC, NFUNC, CONSTS, OUTSEL,
  *   EXTOFM, ESHIFT, SHIFT, CARYIN, CARRY, EQUAL, IERR)
Inputs:
   OP      C*4     'READ' from TV or 'WRIT' to TV
In/Out:
   ARMODE  L       T => arithmetic mode F => logic mode
   BFUNC   I       function number (1-16) in blotch
   NFUNC   I       function number (1-16) outside blotch
   CONSTS  I(8)    constant array (may select as ALU output)
   OUTSEL  I(8)    lookup table selects output based on carry
                   (lsb), equal, ROI (msb) input. values -
                   0 - 7 : constants 1 - 8
                   8     : accumulator channel pair
                   9     : selected OFM
                   10    : ALU
                   11    : external
   EXTOFM  L       T => extend sign of OFM on input to ALU
   ESHIFT  L       T => extend sign of ALU output if SHIFT
   SHIFT   L       T => right shift ALU output
   CARYIN  L       T => add one to arithmetic results
Output:
   CARRY   L       T => carry condition occurred in frame
   EQUAL   L       T => equal condition occurred in frame
   IERR    I       error code of Z...XF : 0 - ok
                                          2 - input error
```

YCONST

Reads/writes the constants which are added to the 3 sum channels on the TV.

```
YCONST (OP, RCONST, GCONST, BCONST, VRTRTC, IERR)
Inputs:
   OP      C*4   'READ' from TV, 'WRIT' to TV
In/out:
   RCONST  I     constant for red channel
   GCONST  I     constant for green channel
   BCONST  I     constant for blue channel
   VRTRTC  L     Vertical retrace - do only on ?
Output:
   IERR    I     error code of Z...XF : 0 => ok
```

YFDBCK

Sends a feedback command to the TV

```
YFDBCK (COLOR, CHANNL, BITPL, PIXOFF, BYPIFM, EXTERN,
  *   ZERO, ACCUM, ADDWRT, IERR)
```

```
Inputs:
   COLOR   I    bit map of color to be fedback (RGB = 4,2,1)
   CHANNL  I    bit map of channels to receive feedback
   BITPL   I    bit map of bit planes to receive feedback
   PIXOFF  I    offset fedback image to left by 0 - 1 pixels
   BYPIFM  L    F => image goes thru IFM lookup before store
   EXTERN  L    T => image from external input (iedigitizer)
   ZERO    L    T => feed back all zeros
   ACCUM   L    T => use 16-bit accumulator mode then CHANNL must
                     give even-odd pair lsbyte goes to even (lower)
                     # channel
   ADDWRT  L    T => additive write  F => replace old data
Outputs:
   IERR    I    error code of Z...XF: 0 -> ok
                                      2 -> input error
```

YIFM

Reads/writes a section of TV input function memory. This look up table takes 13 bits in and gives 8 bits out.

```
YIFM (OP, START, COUNT, PACK, VRTRTC, IFM, IERR)
Inputs:
   OP      C*4    'READ' from TV or 'WRIT' to TV
   START   I      start address of IFM (1 - 8192)
   COUNT   I      # elements of IFM to transfer (1-8192)
   PACK    L      T => 2 values/word, F => 1 value/word
   VRTRTC  L      T => do it only on vertical retrace
In/Out:
   IFM     I(*)   function values (0-255)
Output:
   IERR    I      error code of Z...XF: 0 - ok
                                        2 - input error
```

YMNMAX

Reads the min and max values put out by the 3 summers (before application of constants, shifts and OFM) from the TV

```
YMNMAX (RMIN, RMAX, GMIN, GMAX, BMIN, BMAX, VRTRTC,
  *   IERR)
Inputs:
   VRTRTC  L    do it on vertical retrace only
Output:
   RMIN    I    red minimum
   RMAX    I    red maximum
   GMIN    I    green minimum
   GMAX    I    green maximum
   BMIN    I    blue minimum
   BMAX    I    blue maximum
   IERR    I    error code of Z...XF : 0 => ok, 2 => input error
```

YRHIST

Reads the histogram of the output of a selected OFM of the TV. Intensity values 0 through OFMOUT may be read.

```
YRHIST (MODE, COLOR, INITI, NINT, HISTOG, IERR)
Inputs:
   MODE    I          selects area to histogram: 0 blotch,
                      1 not blotch, 2 all, 3 external blotch
   COLOR   I          bit map of single color (RGB - 4,2,1)
   INITI   I          first intensity to histo (1 - 1024)
   NINT    I          # values to get
Output:
   HISTOG  I(NINT)    histogram
   IERR    I          error code of Z...XF : 0 => ok, 2 => input err
```

YSHIFT

Reads/writes the TV shift registers - which shift the 13-bit output of the adders before entry into the 10-bit OFM

```
YSHIFT (OP, SHIFTR, SHIFTG, SHIFTB, VRTRTC, IERR)
Inputs:
   OP      C*4   'READ' from TV or 'WRIT' to TV
   VRTRTC  L     T => do on vertical retrace only
In/Out:
   SHIFTR  I     # bits to shift (right) red channel
   SHIFTG  I     # bits to shift green channel
   SHIFTB  I     # bits to shift blue channel
Output:
   IERR    I     error code of Z...XF : 0 - ok, 2 - input error
```

10.5.4 Level 3 (selected ones of some general interest)

YBUTON

Reads the state of the Button Buffer of the TV display.

```
YBUTON (LBUT, BUTTON, IERR)
Input:
   LBUT     I   Button value returned from an initial status read
                (Send -1 if none done).
Output:
   BUTTON   I   Button value: 0, 1, 2, 4, or 8
   IERR     I   Error: 0 ok, 1, bad IO
```

YGGRAM

Reads/writes the TV graphics color assignment RAM. The data are packed in this look up table for colors as:

```
   bit 15 (msb)   :   1 - graphics replace 0 - add to image
   bits 10-14     :   red value for this graphics value
   bits  5- 9     :   green
   bits  0- 4     :   blue

YGGRAM (OP, VRTRTC, RGBBUF, IERR)
Inputs:
   OP      C*4       'READ' from TV or 'WRIT' to TV
   VRTRTC  L         T => do it only on vertical retrace
In/Out:
   RGBBUF  I(256)    data array
Out:
   IERR    I         error code of Z...XF : 0 => ok
```

YGRAFE

Reads/writes the TV graphics control register. This version does not support all the TV options: it forces the black and white status monitor on and refuses to allow the disable video, graphics, and cursor options.

```
YGRAFE (OP, BLOTCH, STATUS, VRTRTC, IERR)
Inputs:
   OP      C*4   'READ' from TV or 'WRIT' to TV
   VRTRTC  L     T => do it only during vertical retrace
In/Out:
   BLOTCH  I     graphics plane used as blotch (1 - 7)
   STATUS  I     graphics plane used as status (1 - 7)
Output:
   IERR    I     error code of Z...XF : 0 => ok
                                        2 => input error
```

YSTCUR

Reads/writes the TV cursor array which has the pattern exhibited when the cursor is visible. NOTE: if more than one row is read/written at a time, then the Y value decreases!!!

```
YSTCUR (OP, X, Y, NPOINT, PACK, VRTRTC, BUFFER, IERR)
Inputs:
   OP      C*4    'READ' from TV or 'WRIT' to TV
   X       I      initial X position (1-64, 1 => LHS)
   Y       I      initial Y position (1-64, 1 => bot)
   NPOINT  I      # pixel values in BUFFER
   PACK    L      T => 2 values/word, F => 1 value/word
   VRTRTC  L      T => do it on vertical retrace only
In/Out:
   BUFFER  I(*)   data array (lsb's used only)
Output:
   IERR    I      error code of Z...XF : 0 => ok, 2 => input error
```

10.6 Selected Applications Subroutines

10.6.1 Basic TV I/O Operations

DLINTR

Is called by interactive routines to delay the task when nothing is happening (i.e. the user is thinking or out to lunch.) It also prevents cursor wrap around.

```
DLINTR (RP, IEV, DOCOR, QUAD, PP, IT, DOIT)
Inputs:
   IEV    I      not = 0  => event has occurred
   DOCOR  L      Scroll correction parameter for YCURSE
   QUAD   I      quadrant parameter for YCURSE
In/out:
   RP     R(2)   cursor position read (fixed on wraps)
   PP     R(2)   previous cursor position
   IT     I(3)   time of last action
Output:
   DOIT   L      T => something has happened.
```

IMANOT

Is used to annotate an image by writing the string into the lettering plane (usually graphics plane 2) and, if possible writing a block of ones NEDGE pixels wider than the string into graphics plane 4 to force a black background: $NEDGE = 2 * MAXXTV/512$

```
IMANOT (OP, X, Y, IANGL, CENTER, STRING, SCRTCH, IERR)
Inputs:
   OP      C*4     'ONNN' enables the 2 graphics planes
                   'OFFF' disables the 2 planes
                   'INIT' zeros and enables the 2 planes
                   'WRIT' writes strings to the planes
   X       I       X position of string
   Y       I       Y position of string
   IANGL   I       0 - horizontal, 3 - vertical (DOWN)
   CENTER  I       0 - XY are lower left first character
                   1 - XY are center of string
                   2 - XY are top right of last character
   STRING  C*(*)   character string: length from LEN (STRING)
Output:
   SCRTCH  I(*)    scratch buffer (> 1280)
   IERR    I       error code of ZM70XF : 0 - ok
                                          2 - input error
```

IMVECT

Writes a connected sequence of line segments on a TV channel calling YCNECT

```
IMVECT (OP, CHAN, COUNT, XDATA, YDATA, SCRTCH, IERR)
Inputs:
   OP      C*4     'ONNN' line of ones (max intensity)
                   'OFFF' line of zeros (min intensity)
   CHAN    I       channel number (1 to NGRAY+NGRAPH)
   COUNT   I       number of X,Y pairs ( > 1)
   XDATA   I(COUNT)      X coordinates X1,X2,...
```

```
   YDATA   I(COUNT)      Y coordinates Y1,Y2,...
Output:
   SCRTCH  I(*)    scratch buffer (size MAXXTV)
   IERR    I       error code of ZM70XF - 0 => ok;  2 => input error
```

TVCLOS

Closes the TV device and the TV status disk file, updating the information on the disk. Does this all by call to YTVCLS.

```
TVCLOS (BUF, IERR)
Outputs:
   BUF   I(256)     Scratch buffer
   IERR  I          Error code : 0 => ok,   2 => not open in parms,
                    else as returned by YTVCLS
```

TVFIND

Determines which of the visible TV images the user wishes to select. If there is more than one visible image, it requires the user to point at it with the cursor. The TV must already be open. This routine is available only from program AIPS.

```
TVFIND (MAXPL, TYPE, IPL, UNIQUE, CATBLK, SCRTCH,
  *   IERR)
Inputs:
   MAXPL   I        Highest plane number allowed (i.e. do graphics
                    planes count?)
   TYPE    C*2      2-char image type to restrict search
Output:
   IPL     I        Plane number found
   UNIQUE  L        T => only one image visible now
                    (all types except zeroed ones ('ZZ'))
   CATBLK  I(256)   Image catalog block found
   SCRTCH  I(256)   Scratch buffer
   IERR    I        Error code: 0 => ok
                       1 => no image
                       2 => IO error in image catalog
                       3 => TV error
```

TVOPEN

Open the TV, passing pointers through common/DTVC.INC/. Almost all except error checking done by YTVOPN these days.

```
TVOPEN (BUF, IERR)
Outputs:
   BUF   I(256)     Scratch buffer
   IERR  I          Error return from YTVOPN
                    = 10 TV unavailable to this version
```

TVWHER

Is the routine to use if you want the user to point at something on the TV screen. It turns on the cursor, waits for a button push, determines which quadrant the event occurred in, and returns the quadrant number and the TV coordinates of the event corrected for the scroll of the lowest numbered plane which is "on" in that quadrant and for zoom.

```
TVWHER (QUAD, RPOS, IBUT, IERR)
Outputs:
   QUAD  I       Quadrant number
   RPOS  R(2)    TV position corrected for scroll (& zoom)
   IBUT  I       Value of button(s) pushed
   IERR  I       Error code of ZM70XF
```

10.6.2 TV I/O Utilities

BLTFIL

Fills in a series of closed polygons on a TV "blotch" plane

```
BLTFIL (NP, NV, XV, YV, GRCHAN, BUF, IERR)
Inputs:
   NP      I        # of polygons
   NV      I(NP)    # of vertices in each polygon
   XV      I(*)     vertex X-positions list
   YV      I(*)     vertex Y-positions list
   GRCHAN  I        graphics channel number
Output:
   BUF     I(*)     scratch buffer (> 1280)
   IERR    I        ZM70XF error code: 0 ok
```

GRPOLY

Uses a graphics plane to let the user develop a set of closed polygons as a "blotch" region. This routine is available only from program AIPS.

```
GRPOLY (IG, NPY, NV, XV, YV, SCRTCH, IERR)
Inputs:
   IG      I          graphics plane to use
Output:
   NPY     I          Number of polygons set
   NV      I(50)      Number of vertices in each polygon
   XV      I(500)     X-position of vertices in image
   YV      I(500)     Y-position of vertices in image
   SCRTCH  I(*)       Scratch buffer: > 1 line length (> 1280)
   IERR    I          Error code
Common:
   /MAPHDR/ CATBLK image catalog block of blotched image
                   (actually used at lower level in YCUCOR)
```

IENHNS

Performs an interactive linear enhancement of TV LUTs. X cursor => intercept, Y cursor => slope, high button => quit

```
IENHNS (ICHAN, ICOLOR, ITYPE, RPOS, BUFFER, IERR)
Inputs:
   ICHAN   I          channel select bit mask
   ICOLOR  I          color select bit mask
In/Out:
   ITYPE   I          on in: 1 => do plot, A, B switch plot
                                 C switch sign of slope
                             2 => no plot, A, B return
                                 C switch sign of slope
                             3 => no plot, return any button
                      on out - button value
   RPOS    R(2)       Cursor position: initial -> final
Output:
   BUFFER  I(>3072)   Scratch buffer
   IERR    I          Error code of ZM70XF: 0 => ok
```

ILNCLR

Computes a piecewise linear OFM and writes it to the TV. If NEND(NPOINT) is <= OFMINP/2 or /3 or /4, the table is repeated an approprite number of times.

```
ILNCLR (COLOR, NPOINT, NEND, SLOPE, OFFSET, GAMMA,
  *   BUFFER, IERR)
Inputs:
   COLOR   I            color bit mask: RGB = 421
   NPOINT  I            # of segments
   NEND    I            end points of segments
   SLOPE   R(NPOINT)    slopes of segments
   OFFSET  R(NPOINT)    offsets of segments
   GAMMA   R            power applied to colors (1 /gamma)
Output:
   BUFFER  I(1024)      scratch buffer
   IERR    I            error code of ZM70XF : 0 - ok
Form is C = (i-1)*SLOPE + OFFSET with 0 <= C <= 1.0.
Used to be called YLNCLR.
```

IMCCLR

Writes a color contour OFM using a standard table and sequence of colors.

```
IMCCLR (ITYPE, NLEVS, NSTART, NCONT, GAMMA, BUFFER,
  *   IERR)
Inputs:
   ITYPE   I          Which table: 2 Dutch 10c, 1 Dutch 9c,
                      3 IMPS 8c, 4 IMPS 64c
   NLEVS   I          number of levels (256 or 1024 usually)
   NSTART  I          intensity level of first contour
   NCONT   I          intensity range to contour
   GAMMA   R          gamma power for color correction
Output:
   BUFFER  I(1024)    scratch buffer
   IERR    I          error code of ZM70XF: 0 => ok
```

IMCHAR

Causes characters to appear on the TV by calling YCHRW.

```
IMCHAR (CHAN, X, Y, IANGL, CENTER, STRING, SCRTCH,
  *   IERR)
Inputs:
   CHAN    I       channel number (1 - NGRAY+NGRAPH)
   X       I       X position of string
   Y       I       Y position of string
   IANGL   I       0 - horizontal (to right), 3 - vertical (down)
                   ONLY ones supported.
   CENTER  I       0 - XY are lower left of first character
                   1 - XY are center of string
                   2 - XY are upper right of last character
   STRING  C*(*)   character string to go to TV - length from LEN
Output:
   SCRTCH  I(*)    scratch buffer (TV size > 1280)
   IERR    I       error code of ZM70XF: 0 - ok;   2 - input error
```

IMLCLR

Creates a continuous coloring from blue thru green to red.

```
IMLCLR (NLEVS, ICOLR, NBRK, GAMMA, BUFFER, IERR)
Inputs:
   NLEVS   I      # of intensities (usually 256 or 1024)
   ICOLR   I      initial color R,G,B = 1,2,3
   NBRK    I      break point between blue & red
   GAMMA   R      gamma correction power
Output:
   BUFFER  I(*)   scratch buffer
   IERR    I      error code of ZM70XF: 0 - ok
```

IMPCLR

Uses an STC algorithm to produce color contouring along a helix in the lightness-hue-saturation space.

```
IMPCLR (NLEVS, STEPS, LOOPS, LITE, SATUR, HUE, GAMMA,
  *   BUFFER, IERR)
Inputs:
   NLEVS   I          number of intensities (256 or 1024)
   STEPS   I          number of output colors (1 - 1024)
   LOOPS   I          number of loops of helix
   LITE    R(2)       min,max lightness (0.-100.)
   SATUR   R(2)       min,max saturation (0.-100.)
   HUE     R          start hue in degrees (0-360.)
   GAMMA   R          gamma correction power (2.7 or 1.8 ok?)
Output:
   BUFFER  I(1024)    scratch buffer
   IERR    I          error code of ZM70XF
```

TVFIDL

Does an interactive run with button A selecting alternately TVTRANSF and TVPSEUDO (color contour type 2 only), button B incrementing the zoom and C decrementing the zoom.

```
TVFIDL (LCHAN, NLEVS, INBUF, IERR)
Inputs:
   LCHAN    I      Selected gray-scale channels: bit mask
   NLEVS    I      Number of gray levels (usually LUTOUT+1)
Output:
   INBUF    I      Scratch buffer >3072
   IERR     I      Error code: 0 -> ok;  else set by ZM70XF
```

TVLOAD

Subroutine to load a map from an already opened map file to one TV memory plane. TVLOAD puts TV and map windows in the image header and writes it in the image catalog. It assumes that the other parts of the image header are already filled in (and uses them) and that the windows are all computed.

```
TVLOAD (LUN, IND, IPL, PXINC, IMAWIN, WIN, BUFSZ, BUFF, IERR)
Inputs:
   LUN      I      Logical unit # of map file
   IND      I      FTAB pointer for map file
   IPL      I      Channel to load
   PXINC    I(2)   Increment in x,y between included pixels
   IMAWIN   I(4)   TV corners: BLC x,y TRC x,y
   WIN      I(4)   Map window: ""
   BUFSZ    I      Buffer size in bytes
Outputs
   BUFF     R(*)   Buffer
   IERR     I      Error code: 0 => ok
                               1 => input errors
                               2 => MINIT errors
                               3 => MDISK errors
Commons: /DCAT.INC/ CATBLK  image header
```

10.6.3 Non-I/O Utilities

DECBIT

translates a decimal based channel number into a binary channel number, e.g., 1453 => $2^0 + 2^3 + 2^4 + 2^0$. A maximum of nine channels are addressable (6 at a time).

```
DECBIT (NMAX, ICHAN, IPL, LOW)
Inputs:
   NMAX       I       Maximum allowed channel number
   ICHAN      I       Input channel decimal number
Outputs:
   IPL        I       Binary channel # pattern
   LOW        I       Lowest of specified channels
```

ISCALE

Rescale a line of map data in preparation for output device

```
ISCALE (TYP, OMAX, RANGE, NPIX, NINC, RBUF, OUTBUF)
Inputs:
   TYP      C*2    Type of scaling function '  ' = linear
   OMAX     I      Maximum value accepted by output device
   RANGE    R(2)   Map range to transfer; RANGE(1) and (2)
                   should be transfered to output 0 and OMAX
                   RANGE(1) < RANGE(2) forced inside routine.
   NPIX     I      Pixels/line
   NINC     I      Increment between included pixels
   RBUF     R(*)   Input buffer
Outputs:
   OUTBUF   I(*)   output buffer with scaled data
```

MOVIST

Sets and resets the movie status parameters in the TV common.

```
MOVIST (OP, ICHAN, NFR, NFRPCH, MAG, IERR)
Inputs:
   OP      C*4   'ONNN' when turning on a movie
                 'OFFF' when clearing channel(s)
   ICHAN   I     Bit pattern of channels involved (OFFF)
                 Actual first channel number (1-NGRAY, ONNN)
   NFR     I     Number of frames in movie total (ONNN)
   NFRPCH  I     Number of frames per TV channel (ONNN)
                 < 0 => in display rather than movie order
   MAG     I     Magnification number (0 - 3, ONNN)
Output:
   IERR    I     Error = 2 => bad input, else ok
The code: bit 1 (lsb) 1 (on) = display order, 0 => movie order
              2       1 => frame starts movie
              3-6     magnification step
              7-15    # frames
```

RNGSET

Calculates range parameters for displaying a map using the IRANGE adverb supplied by POPS plus scaling information derived from the map header.

```
RNGSET (IR, MMAX, MMIN, RANG)
Inputs:
   IR     R(2)   Range values specified by user
   MMAX   R      Map maximum value from header
   MMIN   R      Map minimum value from header
Outputs:
   RANG   R(2)   Output range values calculated using defaults
```

TVWIND

Sets windows for normal and split screen TV loads.

```
TVWIND (TYPE, PXINC, BLC, TRC, ICHAN, ITVC, IWIN,
  *   IERR)
In/out:
```

```
   TYPE   I       In: <0 -> 1 plane, other -> split method
                  Out: 0 -> 1 plane, other = 10 * (#planes in X) +
                                           (# planes in Y)
   PXINC  I(2)    X, Y increments
   BLC    R(7)    User requested bot left corner
   TRC    R(7)    User requested top right corner
   ICHAN  I       User requested TV chan (decimal form)
   ITVC   I(4)    IN: first 2 user req. TVCORN
                  Out: full "pseudo-TV" corners
Output:
   IWIN   I(4)    Window into map
   IERR   I       error code: 0 -> ok, else fatal
Common:
   /DCAT.INC/ CATBLK image header used extensively, the depth array is
                                set here
```

THIS PAGE DELIBERATELY BLANK