THE CURRENT STATE OF GRAPHICS AT THE VLA

Written by Jim Torson
Version 3
February 17, 1984

1. INTRODUCTION

This document is intended to be a description of the current state of VLA graphics software that I have been involved with. This will include various tidbits of information that may be of interest to someone in the future. (This document is contained in DISPLY::[201,223]TIDBIT.DOC. In order to output it to the laser printer, I first copied it into the DEC-10 and processed it with SCRIBE[13,476].)

2. DEC-10 GRAPHICS

2.1. OMNIGRAPH

OMNIGRAPH is a package of general purpose graphics routines that runs on the DEC-10. It is used by most of the DEC-10 programs that do graphics. (An exception to this is Jerry Hudson's DEC-10 contour program, which uses its own set of routines for doing Tektronix 4012 graphics.)

We first obtained the OMNIGRAPH package from the National Institutes of Health in 1975. We obtained an updated version in 1977. Another updated version was obtained in 1981. This is the version that we are currently using. OMNIGRAPH supports a variety of devices, but we are currently using it only for the Tektronix 4012 and Tektronix 4025 terminals. OMNIGRAPH can be called from SAIL, FORTRAN or BASIC, but we currently use it only with SAIL programs.

The sources for the pieces of this software that we are using are contained in [11,12]. The disk area [11,16,OMNI,SOURCE] contains what appears to be the unmodified sources that we received from NIH. I think the only change that we made was a small change to DISSAI.FAI that was done by Bob Duquet.

The tape from NIH contains several machine-readable documentation files, including a user manual and a description of implementation details. I don't think any of these documentation files are currently stored on the DEC-10 disks. The file DISMAN.DOC[12,12] is probably derived from the user manual that came with the 1975 version of OMNIGRAPH and should thus be ignored, although the more recent releases mostly contain new features rather than changes to the original system.

2.2. AXIS DRAWING AND DATA PLOTTING ROUTINES FOR USE WITH OMNIGRAPH

I implemented a set of routines for drawing labelled axes and for plotting data. These are contained in LCPLT.SAI[11,12]. Also,

LCUTIL.SAI[11,12] contains the DRESTOR routine which reads in a picture description that was saved in a disk file. GRFUTL.SAI[11,12] contains some utility routines that are used by the LCPLT.SAI routines and also by the character plotting routines.

2.3. CHARACTER GRAPHICS ROUTINES

I implemented a set of routines that do crude graphics in a character array for output to the line printer or a text terminal. CHPRIM.SAI[11,12] contains the set of character graphics routines which are similar to the basic OMNIGRAPH functions.

2.4. AXIS DRAWING AND DATA PLOTTING ROUTINES FOR USE WITH CHARACTER GRAPHICS ROUTINES

CHPLT.SAI[11,12] contains a set of routines which are similar to the routines in LCPLT. I seem to recall that at some point Dave Rosenbush broke CHPLT up into separate source files for each routine so that only the needed routines would be included in an executable program. I think he may have made some changes to the routines when he did this, and it may have led to some problems.

2.5. OTHER DEC-10 ROUTINES AND DOCUMENTS

LCPLT3.SAI[11,16] contains some routines for doing 3D plotting on the Tektronix terminal. These were never submitted to the standard system. [11,16] also contains some miscellaneous documents which might be of interest to someone. [21,16] contains some additional documents and an old version of the DEC-10 FITS program. I don't remember why we kept the old FITS program around. Perhaps it had something to do with the way scaling was done on tapes written long ago.

3. PDP-11 GRAPHICS

3.1. IMPS

The "standard" IMPS source files and task files are contained in [201,10]. This includes the sources for the general utility routines as well as the IMPS main programs. For many of these source files, the older version is the version that was used before the change from the PDP-11/40 CPU to the PDP-11/44 CPU. The [201,10] area also contains the IMPS version of the FITS program. The updating of this disk area (excluding FITS and the changes for the 11/44) was documented by the use of system change forms which are all filed together in a binder.

Currently, when the system is reloaded, the STARTUP command file executes [1,2]PREIMPS.CMD, which runs [201,10]STDUFD to initialize some

things in the global common area used by IMPS. STARTUP also executes [201,10]INSIMPS.CMD to install the IMPS tasks. Since there are a lot of IMPS tasks, this takes up a lot of pool space. The command file [201,10]REMIMPS.CMD will remove most of these and thus free up some pool space.

When you start up IMPS, it asks you to type in your user number. This is just a string which forms the filename of the catalog file which is to be used. Currently, the IMPS task only allows three characters to be typed for the user number. The current user number string gets stored in the global common area, where space for six characters is provided. Thus, it might be easy to change things to allow IMPS to handle four character user numbers. However, there might be a few cases where the user number string is passed from one task to another in the packet of data that gives the parameters to be used by a processing operation. Another problem might be that GETMAP also only allows you to type in a three character user number. (I'm not even sure if GETMAP still works these days.)

Most of the IMPS routines are compiled with a line buffer size that will allow reading in and processing a map line that contains 1024 floating point pixels. A map line that is 2048 pixels long can be handled if the map is stored in 16-bit format on the disk and conversion to floating point is not required (or conversion of no more than 1024 pixels to floating point is required). As long as you don't run out of PDP-11 address space, it is quite easy to expand the buffer size.

The area [201,207] contains some "special" versions of some of the IMPS tasks. This includes versions of the image loading programs that allow the user to specify a range of map values to be scaled into the 8-bit pixel values. It also includes versions of WRTDIC and WRTDC2 that have been compiled with bigger line buffer sizes so that a 2048 pixel floating point map can be output to the Dicomed. I seem to recall that there once were some other special versions, e.g., a version of the Versatec contour program that would draw thick contour lines and a special version of the optical depth (?) program. These seem to be gone from the disk. Perhaps they exist on a back-up tape somewhere.

The program [201,201]WHO will tell you the user number that was last specified when the IMPS task was run.

The program [201,10]CRCOP will allow you to copy a catalog record from one user's catalog file into another's. If you then use PIP to make a copy of the corresponding map file, you will effectively copy a map from one user's area into another's.

The program [201,10]CHKCAT will check to make sure all of the maps listed in a catalog file actually have existing data files. It will try to read in the last map line stored in the map file to make sure that the file is at least as big as it should be.

If you set the UIC to [201,220] and type @SPACE, then you will get a report of how much disk space is being used by each user for storing IMPS maps. This list is put into a disk file which can then be printed out on the Versatec. (If you try to spool this file to the DEC-10 line printer, part of the file will not get printed out.)

The program [201,10]MASDEL will do a massive delete of all the maps listed in a given user's catalog. (It will not delete the catalog file, but SPACE will delete any empty catalog files that it finds.) [201,201]CATDEL is similar to MASDEL except that it will delete only maps that are in a given category.

3.2. DEMO

The command file [201,10]INSIMPS also installs and runs the DEMO program. This program is contained in [201,201]. The purpose of DEMO is to automatically display a sequence of sample images on the Comtal image display when nobody is running IMPS. DEMO finds the images to display by looking in the map area for user number 8. Since user numbers are octal numbers, this does not correspond to any real user. Global event flag number 34 is used for communication between IMPS and DEMO. When DEMO starts up, it clears the flag. The IMPS task sets the flag when it starts up. DEMO checks the state of the flag each time a line is loaded into the Comtal, and it just exits when it sees that IMPS has set the flag. When the user exits from IMPS, it restarts DEMO. DEMO is installed at the lowest system priority so that it will not interfere with other things in the system.

3.3. PIPELINE DISPLAY SYSTEM

This section contains some general comments on the Pipeline Display System (PDS). The next several sections contain comments on the various individual packages of graphics routines which are used for handling the various devices. These are followed by a section that describes the overall software organization of how these pieces are used together in the PDS.

The file DISPLY::[201,223]GP.DOC contains a description of new functions that were planned for the Pipeline Display System and the priorities that were assigned to them.

The PDS is initialized by the execution of the command file [300,20]EXP.CMD. This sets the default UIC to [300,20], installs the global common area used by some of the tasks, installs the individual tasks, runs the three special display handling tasks (described below), and runs the TVINIT task. Unlike IMPS, the PDS appears to the user as a collection of separate tasks that are run from the console terminal.

TVINIT TASK. This purpose of this task is to initialize the state

of the IIS image display device by setting the various IIS control registers and table memories to a standard state. Information about the current state of the IIS is contained in a block of data defined by the file TVSTAT.DCL. This block of data is stored in the file [300,21]TVSTAT.TVS between task activations. Routines are available to read and write this file (TSREAD and TSWRIT). The contents of the IIS image memory planes is kept track of by using an image catalog file which is similar to that used in the AIPS system. The image catalog file is [300,21]IMCAT.IMG. The records in this file which describe an image loaded into the IIS are the same as the records in the map catalog file plus a few additional items. The items in the map catalog record are defined in CATREC.DCL. The additional items used in the image catalog record are defined in ICREC.DCL. Several routines are available for dealing with the image catalog file, i.e., ICINIT, ICOPEN, ICREAD, ICREAL, ICREAN, ICREA1, and ICWRIT. A very simple task called ICCREA is used to initially create the image catalog file. This should only have to be run if something terrible happens to the image catalog file. Ordinarily, the user will not need to explicitly run the TVINIT program. However, it is OK for him to run it if he wants to initialize the IIS.

TVLOD TASK. The user runs this task to load a map image into one of the IIS image planes. This task is also intended to draw a wedge with the image, draw map identification labels, and draw coordinate tic marks and labels. However, these things have not been fully implemented due to the fact that the change to the new map catalog record format has not yet been made. For coordinate tic drawing, all that is implemented is to draw a box around the image and to title the axes as either Right Ascension/Declination or Time/Baseline. For the map identification labels, the program always labels the peak intensity as "JY." This will have to be improved when the new catalog record format is implemented. The title on the wedge is always "INTENSITY - JY." This also will have to be improved. The program was initially set up to use the standard command CHANNEL to allow the user to select a spectral line channel plane out of a 3D data cube. However, this command has been disabled since the current mapping system does not produce cubes and users were expecting this command to select different spectral line channel maps which are stored as separate 2D images.

TVDIS TASK. The user runs this task to modify the way the image is displayed on the IIS. If any mode other than intensity/hue is selected, then the "Insert Wedge" function will just draw an unlabelled wedge across the bottom of the screen. This is thus a rather useless function now that labelled wedge drawing is implemented in TVLOD. However, if the user enables intensity/hue mode and picks the "Insert Wedge" function, then this will draw a two dimensional intensity/hue wedge (without labels). The last function in the top level menu, "Label Transfer Function Plot," was intended to put nice labelled tic marks on the transfer function plots, but this has not been implemented. (When the program starts up, it will label the X axis of the transfer function plots with the min and max map values. It will also put the zero value grid line in the right location.) Most of the things displayed on the

VT-11 by TVDIS are contained in a VT-11 display file which is saved in a disk file and gets read in when TVDIS starts up. Changes to the VT-11 display are done by turning VT-11 "subpictures" on and off. TVDIS expects to find the display file in [300,21]TVDIS.DIS. This file is generated by running the GENDIS program, which of course needs to be run only when a change needs to be made to the display file. Note that there was not enough room in the display file for everything that was needed. Thus, TVDIS generates some things at run-time and then deletes them when it is done with them. An example of this is the messages used with drawing the intensity/hue wedge.

TVVAL AND CURVAL TASKS. These tasks allow the user to position a cursor on the IIS and dynamically display the pixel coordinates (or antenna pair) and map intensity of the pixel under the cursor. When the user freezes the display by pushing down on the data tablet pen, the sky coordinates (or time) are also displayed. TVVAL gets the pixel value by reading it back from the IIS image memory. This thus allows pointing to pixels in the wedge. CURVAL gets the pixel value by reading it from the map stored on the disk. This thus gives a full precision map value. Both programs only allow the user to point to the last image that was loaded into the enabled IIS image plane. Perhaps it would be useful to allow the user to select the desired image if more than one has been loaded into the image plane.

CLNMAP TASK. This task allows the user to submit a clean request to GRIDER. When the user types the GO command, CLNMAP (running in DISPLY) communicates with the CLNDIS task that runs in GRIDER. The BOX command allows the user to interactively specify clean boxes on the IIS screen. This command is implemented in the source file BOX.FTN. The INTBOX and GETICR routines in this file should probably be put into separate files so that they can be used by other tasks that would want to allow box specifications, e.g., the map statistics task. CLNMAP allows specification of the PACK names for the input maps and output map. However, the programs in GRIDER do not yet have this feature implemented, so these pack names are not put into the MIX record and are thus not sent to GRIDER.

The files for ICCREA, GENDIS, TVINIT, TVLOD, TVDIS, TVVAL, and CURVAL are kept in [210,50] and [220,50] in DISPLY. The files for CLNMAP and CLNDIS are stored in [210,51] and [220,51].

3.4. CONCURRENT TASKS IN THE PIPELINE DISPLAY SYSTEM

Since the Pipeline Display System consists of a collection of separate tasks, the user may want to have more than one task running at the same time. This can be done as long as certain restrictions are observed. (It may be easier to just tell the user that he cannot run more than one task at a time than it is to make sure he understands the restrictions.) Running two tasks that output to the IIS does not present a problem. (This is explained further below.) For example, the user can

start loading an image plane in the TVLOD task and then run TVDIS to modify the transfer function, etc. The restrictions involve the image catalog file which describes what maps are currently loaded in the IIS image planes. When the user gives the GO command to TVLOD, the new entry is made in the image catalog file and then the image is loaded. When the TVDIS task starts up, it looks in the image catalog file to get the map min and max for labelling the transfer function plots and for positioning the zero value grid line. If the user loads a new image while TVDIS is running, then TVDIS will not know about the new image and will not update the transfer function plots. In a similar manner, TVVAL and CURVAL just get the map scaling out of the image catalog file when they start up and will not know it if TVLOD loads a new image. CLNDIS looks in the image catalog file each time the interactive box specification is used. On the other hand, trying to talk to both TVLOD and CLNDIS on the same terminal at the same time would be a bit confusing. Of course it makes no sense to try to run things like TVDIS and TVVAL at the same time since you can't do more than one thing at a time with the data tablet.

3.5. VT-11 SOFTWARE

The software package supplied by DEC is used for doing the basic display functions on the VT-11. The PDS uses the latest version of these routines, which is contained in the object module library [210,10] or [220,10]GLIBVT.OLB. The sources for these routines are stored in [200,200].

The IMPS tasks are built with the previous version of the VT-11 routines, which have been moved into [200,201]. (The IMPS task build command files point to [200,200]GLIBVT.OLB since the previous version used to be stored in [200,200].)

The file [200,200]GLIBIN.OLB contains the latest version of the VT-11 routines that have been compiled to accept integer rather than floating point arguments. These are not used by any of the current applications software.

The VT-11 manual says that RSX uses the top 32 words of APR 7 and this area should not be used by the VT-11 display file. Our display file partition extends into this area, but I have been careful not to store anything there. This seems to work OK.

It may be of interest to note that VT-11 characters are 14 units wide by 25 units high (in terms of VT-11 screen coordinates). This allows 73 characters across the screen and 40 characters down the screen.

I have implemented a few additional routines for dealing with the VT-11. The sources for these routines are contained in [210,53] or [220,53] and their object modules are in the object module library [210,53] or [220,53]INTLIB.OLB. These routines include:

ABSMOV - put a move to an absolute screen location into the

VT-11 display file. This is done without drawing a visible dot and without changing the intensity.

FMPLOT - update an IIS function memory plot which is being displayed on the VT-11.

FNDSUB - find a given subpicture in the VT-11 display file.

MAKCUR - add a subpicture which contains a cursor to the display file.

3.6. VERSATEC SOFTWARE

The latest version of the Versaplot software supplied by Versatec is contained in [7,7] and [7,11]. The object module library is [7,11]PEPLIB.OLB. The IMPS contouring program is currently built to use this latest version. (The previous version has been moved into [107,7] and [107,11]. I seem to recall that this had a bug which resulted in the RASM task aborting sometimes when it was outputing the plot.) The Pipeline Display System currently does not use the Versatec.

The following is a summary of the changes that were made to the distributed version of Versaplot. PLOTS.FTN was changed to specify our plotter model number. Also, VPMCOM common was added to the PLOTS routine, the VPARAM routine was added, and IWORD (in PLOTS) was changed to indicate a new size of buffer in the vector-to-raster phase. IOPKG.MAC was changed so that the output device is SY0: rather than VP0:. VTOR.MAC was set to use the MUL instruction. Also, the change shown on p. F-13 of the Versatec manual was applied to allow a buffer larger than 32K bytes. Debug messages were removed from OFFSET, SYMBOL, TONE, THREAD, ADDVEC, RASM and RASL. The buffer size was set to 19000 words in RASM, RASL, and RASS. A request of the IMPS task was added at the end of RASM. The following command files in [7,7] were modified: INSTAL.CMD, INSTP2.CMD, PCOMP1.F77, PCOMP2.F77, PSORT.F77, RASM.CMD, RASS.CMD, RASL.CMD, THREAD.CMD, SORT.CMD. Note that the original versions of files that were modified are still stored on the disk as version 1, so it is easy to find out exactly what changes were made.

We recently received the October 1983 issue of the Versatec User's Newsletter. I noticed that it says the current revision of the RSX-11M version of Versaplot is 7.0H. I think this may be a more recent revision than the one we have. The latest operating manual is 5716-12, but we have manual number 5716-11. It probably would be cheap to obtain an update. Perhaps they have fixed some bugs. I suspect that our version has a bug which sometimes causes the end of the plot to get truncated.

3.7. IIS SOFTWARE

The software supplied by IIS is divided into four categories: device interrupt handler, diagnostic interpreter, interface and utility routines, and primitives package. Al Braun is taking care of the device interrupt handler and Arnold Rots is taking care of the diagnostic

interpreter, so I won't say any more about them. The interface and utility routine package includes the basic routines to control the various subunits in the IIS plus some higher level routines such as character drawing. These also include various lower level routines which get called by the routines of interest to the application programmer. The primitives package contains higher level routines which do things which are more application-oriented. For the most part, the existing software uses the routines in the interface and utility package. Only a few routines in the primitives package have been used.

For some unknown reason, IIS sent us an obsolete version of their software when they sent us the hardware. The obsolete version of the utility routines is currently stored in [314,30]. The obsolete version of the primitives is currently stored in [314,32]. These files should be ignored, and maybe they should just be deleted.

The updated version of the utilities package and the primitives package that we received from IIS is stored in [201,217]. The subset of routines that has actually been used is stored in [201,224]. (A few of these have been modified.) The object module library that contains these routines is [210,10] or [220,10]IILIB.OLB.

The documentation for the IIS software is inadequate. For one thing, there is no description of the buffering scheme. Some of the key routines such as INFCB and WTEXEC are not described. Sometimes the routines that are described have incomplete or incorrect descriptions. I have made a few annotations in the manual, but the applications programmer will probably have to look in the listings of the code.

An important feature of the current version of the IIS routines is the buffering of output to the device. The last parameter in the INFCB routine specifies the number of buffers to use. 0 means no buffering, i.e., each 8-word header and each data block are sent out to the IIS with a separate DMA transfer. 1 means single buffering. As calls to output to the IIS occur, the headers and data blocks are accumulated in a buffer (which is contained in the FCB parameter supplied to all the routines). When the buffer gets full, it is output to the IIS with a single DMA transfer. Or, the application program can call WTEXEC to send out whatever is in the buffer and wait for the completion of the I/O. A value of 2 in the last parameter to INFCB indicates double buffering is desired. This might not work properly since it was necessary for me to make some internal changes to some of the routines. I recommend that a buffering parameter of 1 be used.

The following is a summary of changes that have been made to the distributed version of the IIS software.

M70OP.UTI - changed to correspond to our hardware configuration.

M70ERR.DSP - changed so that the "Model 70 time-out" message will be output on a new line.

DGREY.PRI - removed call to SETUP so that all it does is just draw the greyscale wedge.

M70UTILS.UTI - changed DVECT routine so that it expects the input data array to be a byte array rather than an I*2 array. (This now works the way it is described in the manual.)

DISPIO.FMD - changed GTM70 routine so that it doesn't try to allocate the device. Changed M70IO routine so that it always does a WTQIO instead of a QIO. Added calls to P and V routines in the MC routine. (The P and V operations and most of the other changes were made in order to allow more than one task to do I/O to the IIS at the same time. This is explained further below.)

DEXECW.DSP - changed to make sure it does a wait. (NOTE: this routine does not contain the P and V calls and thus should probably not be used. The applications tasks should call WTEXEC.)

DEXEC.DSP - added P and V calls.

WTEXEC.DSP - added P and V calls.

M70XF.DSP - changed calls to M70BF so that it tells it whether a header or a block of data is being added to the buffer.

M70BF.DSP - added a parameter to the calling sequence which indicates whether header or data is being buffered. Added P and V calls and made other changes so that things would work properly when multiple tasks try to access the IIS.

I did some tests with different buffer sizes to see how fast an image plane could be loaded. The maximum speed is less than the transfer rate that is in the Model 70 specs. Someone at IIS told me that this is due to a transfer rate limitation in the PDP-11 interface. Also, very large buffer sizes gave a Model 70 time-out error. I don't know what causes that.

One of the things that the IIS is used for is to allow the user to interactively specify a box location on the screen. This is done by using the data tablet to move a cursor on the IIS screen which in turn specifies the location of one of the corners of the box which is drawn on the screen. This is implemented by a loop which gets the latest cursor position and then erases the old box and draws the new box. Erasing the old box was initially done by a call to BCHAN, which in turn calls FDBCK. This resulted in the box jumping around on the screen when zoom was enabled. Since this seems to be a "feature" of the hardware rather than a malfunction, I changed the program so that the old box is erased by drawing it with "0" bits rather than "1" bits. This seems to work OK.

It may be of interest to know that the software character generator in the IIS routines seems to make characters that are contained within a box that is seven pixels wide and eleven pixels high. This seems to apply to the letters and numbers. Things like spaces and punctuation marks are not as wide. Although the background character box is eleven pixels high, the actual characters are smaller. Thus, a line spacing of ten pixels will work OK.

3.8. DATA TABLET SOFTWARE

A "temporary" kludge was implemented in IMPS for using the data tablet since no device driver for the tablet was available at that time. The kludge was never changed. An IMPS task that uses the tablet maps the I/O page into its address space. When it wants to get a set of coordinates from the tablet, it calls a routine which triggers the tablet and waits for the coordinates to be produced. It does the wait by executing a tight loop which just checks the appropriate bit in the tablet status word in the I/O page. In order for this to work properly, the data tablet must be set to "remote" mode with the buttons on the front of the controller box. This will also work if the tablet is set to "not switch stream" mode since the controller will then generate its own trigger at the rate set by the slide switch.

The Pipeline Display System makes use of the data tablet device driver which was written by Al Braun. In order for this to work properly, the tablet MUST be in "not switch stream" mode. (Thus, if it is always left in this mode, both the PDS and IMPS will work.) When the tablet generates an interrupt, the driver will cause an Asynchronous System Trap (AST) to occur. (In the original version of the driver, an AST was generated only if something had changed since the last interrupt. However, it was necessary to change this so that each interrupt generates an AST.) Note that the setting of the stream rate slide switch is important. If a slow rate is selected, then the user will get a sluggish response from the tablet. However, the rate should not be set too high. As the rate of interrupts is increased, more of the available CPU cycles will be used by the AST processing. More CPU used by the AST processing of course means less CPU available for the application program that wants to do something with the data from the tablet. Also, if the AST's start coming faster than they can get processed, then the pool space will quickly get gobbled up since the AST queue uses pool space.

There is a program called TIME in [201,223] which is useful for setting the slide switch. If this program is installed and run at the lowest priority then it will report a rough estimate of what percent of the total available CPU cycles is actually being used by the other tasks in the system. When the system is idle, TIME will usually report that only one percent or less of the avialable CPU cycles is being used. However, sometimes it reports that about 17% is being used. I don't know what is causing this. Perhaps it is something related to the network.

Two sets of low level routines have been implemented for using the data tablet. The sources are stored in [210,53] and [220,53]. They are contained in the object module library [210,53] or [220,53]INTLIB.OLB. With the routines in T1RTNS.MAC, the AST routine accumulates information about what has been happening with the tablet. Routines which can be called by the application program are available for obtaining this information, e.g., the tablet coordinates and status flags at the latest AST and information about what has happened with the status flags since the last time the application program asked for information. The

routines in T2RTNS.MAC allow the run-time specification of a Fortran-coded routine that is to be activated when an AST is received. This makes it easy for the AST routine to do application-specific things such as moving a cursor in response to relative movements of the pen.

From time to time the data tablet hardware develops a problem that is not fatal but is a bit annoying. When the tablet is operating properly, the user can lift the pen off the tablet surface and then put it back down near the surface without the cursor on the screen jumping. When the problem develops, the cursor will jump back and forth a bit when the pen is moved up and down. This usually happens when the pen is in the center or right-center area of the tablet surface. If you open up the tablet surface you will see that it consists of a set of horizontal wires and a set of vertical wires separated by a sheet of plastic. The problem seems to be caused by some of the wires getting stuck to the plastic, perhaps resulting from users pushing down too hard on the pen. To fix it, all you have to do is move the plastic sheets around a little to unstick the wires. Of course in doing this you should be very careful not to break any of the wires.

3.9. KEYBOARD SOFTWARE

The file KBRTNS.MAC in [210,53] and [220,53] contains a set of routines for handling input from the keyboard that sits in front of the VT-11. These are contained in the object module library [210,53] or [220,53]INTLIB.OLB. These are the routines that are used in the Pipeline Display System. (IMPS uses an earlier version of these contained in [201,10]KBPRIM.MAC.) When a key is typed on the keyboard, an AST is generated and the AST routine adds the character to a buffer. The application program can call routines to do things like get the next character out of the buffer or wait for a key to be typed.

The keyboard contains a beeper which makes a loud obnoxious noise. Sending a control-G out on the terminal line that is connected to the keyboard will not ring this beeper. I have a vague recollection that you can ring it by doing something like trying to write in the location of the I/O page that contains the VT-11 read-only status register.

There is a power switch on the back edge of the keyboard case. Occasionally a user will push the keyboard against the VT-11 and turn off the keyboard power, which of course causes it to quit working.

3.10. ZETA PEN PLOTTER SOFTWARE

The pen plotter routines supplied by Zeta are contained in [210,35]. ZETASUBS.OLB is the object module library. Arnold Rots has handled the installation of the Zeta routines, so I won't say any more about them.

3.11. DI-3000 SOFTWARE

The DI-3000 software is contained in [201,221] and [201,222]. I installed this early in 1983. This included the basic DI-3000 Level A routines and the Versatec device driver. I modified some of the command files, applied a few patches recommended by Precision Visuals, and made some changes to the Versatec driver. A little later I installed the contour package when we received it, but it did not include the mesh surface package. Some time after that we received the Zeta pen plotter device driver, but I did not install it. Arnold Rots recently installed the Zeta driver. We recently received a tape that contains an updated version of the contour package which includes the mesh surface package. This is the "standard" version of the contour package, i.e., it does not contain anything that is specific to the PDP-11 system. This package has not yet been installed. Currently, the only application that uses DI-3000 is Arnold's text processing program. Arnold is now taking care of DI-3000, so I won't say any more about it.

3.12. INTERACTION ROUTINES

The interactive programs in the Pipeline Display System use a set of interaction routines which were implemented to follow the general input model that has been proposed for GKS. This is described in the article "The Detailed Semantics of Graphics Input Devices," by Rosenthal, et. al., in COMPUTER GRAPHICS Vol. 16, No. 3 (July 1982), p. 33. The interaction routines provide the application program with a set of logical input devices which make it easy for the application program to get a variety of input values from the user without getting bogged down with the details of the I/O to the various physical devices. The implementation of the interaction routines takes care of the details of using the physical devices that are present in order to obtain the desired logical input values. This involves accepting input from the data tablet and keyboard and providing appropriate feedback and acknowledgement on the VT-11 and IIS screens.

The routines do not strictly adhere to the GKS specification because the proposed Fortran binding was not available when I implemented the routines. Also, I only implemented the subset of the GKS functions that were needed. For the ones that were implemented, only the input modes that were needed were implemented. And, I implemented some additional functions that were useful in the PDS. Some of these are composites of the basic logical device types, e.g., a composite of the valuator and choice devices. Some of the additional devices are new types, e.g., routines for inputing box locations on the IIS.

The first character of the routine name indicates the type of function and the rest of the name indicates the logical input device to which the function applies. The following are the first letter codes and functions:

I - initialize the logical device.

R - request a value from the logical device.
M - set the mode of operation. This will select either request mode or sample mode for logical devices that support sample mode.
S - sample the value of the logical device.
A - The Fortran part of the AST routine that implements the logical device. These routines should never be called directly by the application program.

The following are the logical input devices that are implemented and the types of data that they supply to the application program:

LOC1 - locator no. 1 - VT-11 screen location.
LOC2 - locator no. 2 - IIS screen location.
CHO1 - choice no. 1 - item chosen from a VT-11 menu display.
CHO2 - choice no. 2 - choice indicated by typing a keyboard key or by pushing down on data tablet pen.
VC1 - valuator-choice no. 1 - pair of values plus choice indicated by keyboard key or pen push.
LC1 - locator-choice no. 1 - IIS screen location plus choice indicated by keyboard key or pen push.
LIN1 - line no. 1 - pair of IIS screen coordinates which define a line on the IIS screen.
BOX1 - box no. 1 - two pairs of IIS image plane coordinates which define a box on the IIS image plane. User positions two of the box corners.
BOX2 - box no. 2 - two pairs of IIS image plane coordinates which define a box on the IIS image plane. User positions the center of a fixed size box.

Note that some of the logical devices that deal with the IIS were initially implemented to return IIS screen locations but have now been changed to supply IIS image plane coordinates. These include BOX1 and BOX2. These thus work properly when the IIS zoom is enabled. However, some of the logical devices have not been changed to deal with image plane coordinates. The BOX1 and BOX2 implementations make use of the LC1 logical device, so it is useful for LC1 to return screen coordinates. On the other hand, the LIN1 device should probably be changed so that it returns image plane coordinates rather than screen coordinates. (LIN1 is currently not used by any part of the Pipeline Display System.)

Note that for the LOC1 logical device, the Fortran-coded AST routine does the movement of the VT-11 cursor in response to movements of the data tablet. Similarly, in LC1, the Fortran-coded AST routine moves the IIS cursor. The RBOX1 routine makes use of the LC1 logical device for positioning the IIS cursor. It does this by calling MLC1 to put the LC1 device into sampled mode and then going into a loop where it calls SLC1 to sample the current IIS cursor location and then redraws the box on the IIS screen based on the new corner location. There are thus effectively two concurrent processes running in the system. The higher priority process is the AST routine which does the quick and easy task of moving the cursor. The lower priority process is the box drawing, which runs at

the normal task priority. The time to redraw the box can be longer than the time between successive interrupts from the tablet. However, this organization gives the user good response in moving the IIS cursor. The redrawing of the box will just sometimes lag behind the movement of the cursor.

The sources for the interaction routines are stored in [210,53] and [220,53]. They are contained in the object module library [210,53] and [220,53]INTLIB.OLB.

3.13. GENERAL UTILITY ROUTINES

The sources for the general utility routines that are used in IMPS are stored in [201,10] along with the IMPS main programs. The routines are contained in the object module library [201,10]IMPLIB.OLB.

The sources for the general utility routines that are used in the Pipeline Display System are stored in [210,47] and [220,47]. The routines are contained in the object module library [210,47] or [220,47]DISLIB.OLB. Some of these routines are adaptations of some of the IMPS utility routines. However, there may be some routines in the IMPS area which are of general usefullness but which are not in the PDS area.

3.14. GRAPHICS SOFTWARE ORGANIZATION IN THE PIPELINE DISPLAY SYSTEM

This section describes the overall organization of how the Pipeline Display System makes use of the graphics routines described above. The applications tasks in the PDS do not directly handle the graphics devices. Instead, there are three special tasks that do the actual I/O to the VT-11, IIS, data tablet and keyboard. When an applications task makes a call on a graphics function, the subroutine that is linked into the task image doesn't actually do the function. Instead, it calls upon one of the special tasks to perform the graphics function for it. The main advantage of this is that it reduces the memory size and thus the need for overlays in the applications tasks since the linkage routines require less memory than the code which executes the graphics functions. Also, a task which uses the VT-11 does not need to have the VT-11 display file in its address space. A secondary advantage of this organization is that it allows more than one applications task to be accessing a graphics device at the same time. (As a practical matter, this is only desirable for some uses of the IIS.)

The three special tasks that handle the graphics devices are called VT, II, and TK. VT processes requests for VT-11 functions, II processes requests for IIS functions, and TK (Tablet-Keyboard) processes requests for the interaction routines that implement the logical input devices.

LINKAGE CONVENTIONS. Conceptually, the VT, II and TK tasks are initially waiting to receive a request for their services from an applications task. An applications task makes a request by sending to the appropriate task a packet of information which specifies the desired operation and any needed parameters. The applications task then waits for the special task to send back a packet which indicates completion of the operation and which contains parameter values being passed back to the caller. One way to implement this would be to use the operating system functions which pass data packets between different tasks. However, in order to get faster run time execution (I hope), the global common area defined in GBLCOM.DCL is used for communications between the tasks and global event flags are used for synchonization. Space is allocated in GBLCOM for a set of request packets for each of the three special graphics tasks.

The command file [300,20]EXP.CMD initializes the system by running the three tasks, each of which suspends itself by waiting on the global event flag that is to be set when the task has work to do. (If the communications gets hung up, you can restart things by aborting any applications tasks that are running and then executing [300,20]RESTART.CMD.) The code for a graphics routine that is linked into an application task will obtain an available request packet and fill it with a request code and parameter values. It will then set the global event flag associated with the desired graphics task and wait for the global event flag which will be set when the graphics operation is complete. As a practical matter, only a single request packet is used for the VT task and a single is used for the TK task. As many as three are available for use with the II task.

[210,47] and [220,47] contain a collection of routines which have the same names as the various graphics routines but which just provide the linkage to the special tasks which actually perform the functions. These linkage routines are included in DISLIB.OLB. Thus, the task builder command file for an applications task should specify a search of DISLIB, but should NOT specify a search of GLIBVT, IILIB, or INTLIB.

Note that the set of linkage routines that has been implemented does not include all of the available graphics routines. Thus, it may be necessary to add some linkage routines and to make additions to the VT, II, or TK tasks. Also, note that use of the linkage routines puts some restrictions on what the applications task may do. These restrictions are described below. Some of these restrictions result from the fact that fixed arrays in GBLCOM are used for passing some parameters between the applications tasks and the graphics tasks. This increases efficiency by eliminating copying of large data arrays. Also, it reduces the size of the request packets. In a few cases, a linkage routine is provided which results in several graphics routines being executed in the graphics task. This is done to increase efficiency by reducing the communications between tasks.

VT TASK. The services provided by this task include a subset of

DEC's VT-11 routines plus the additional routines ABSMOV, FMPLOT, FNDSUB, and MAKCUR. Also included in the linkage routines is DSET, which sets a word in the VT-11 display file. This is used for setting the position of subpictures. Note that some of DEC's VT-11 routines allow a variable number of parameters. I have always used them with a fixed set of parameters, so the linkage routines do not handle variable numbers of parameters. The VT-11 request packet contains room for 40 words of parameter data. Thus, a call to TEXT should not contain a string longer than 79 characters. (The 80'th character would be needed for the terminating null byte.) The FMPLOT function obtains the new function memory from the FM array in GBLCOM.

II TASK. The services provided by this task include a subset of the IIS supplied routines plus some special versions of some of these routines plus a couple functions which execute several IIS routines. The special versions include GIMAGE (write 1's to a graphics plane) and RIMAGE (read pixel data from an image memory into the RDBUF array in GBLCOM). The composite functions include ZBOX (erase a box in a graphics plane) and BBOX (draw a box in a graphics plane). IMAGE and IFM get their data from LINBUF in GBLCOM. LUT, POFM, PLUT, and GRRAM get data from BUF in GBLCOM. BCHAR will use only the first 76 characters that are supplied. For routines that require a scratch buffer, the buffer is of course supplied by the II task and does not need to be supplied by the application task. The II task calls INFCB to initialize its FCB when it starts up. Thus, the application program does not have to call INFCB. If it does, it is just a null operation in the II task.

Note that the current implementation of the system will support two (or maybe three?) concurrent applications tasks that are making requests for services from the II task. Thus, for example, the user can run TVLOD and give the GO command to start the loading of an image memory plane. Then, while the loading is in progress, he can run TVDIS to change the display mode, set the color encoding, modify the transfer function, etc.

TK TASK. The services provided by the TK task include most of the logical input device routines described above. It also includes some of the keyboard functions in KBRTNS, some of the tablet functions in T1RTNS, and some functions for dealing with the local event flags associated with the keyboard and tablet.

All input from the data tablet and keyboard is handled by the TK task in the logical input device routines. For the most part, all handling of the VT-11 is done by the VT task. Thus TK calls upon VT to provide the feed-back on the VT-11 that is needed for some of the logical input devices. The only exception to this is the positioning of the VT-11 cursor that is done in the AST routine for the LOC1 logical device (routine named ALOC1). In this case, the TK task also includes the VT-11 display file in its address space, and ALOC1 directly sets the locations which contain the VT-11 coordinates of the cursor.

For the most part, all handling of the IIS is done by the II task.

Thus, TK calls upon II to provide the feed-back on the IIS screen that is needed for some of the logical input devices. The only exception to this is the positioning of the IIS cursor that is done in the LC1 logical input device by the RLC1, MLC1 and ALC1 routines. In these routines, the TK task itself sends cursor positioning commands out to the IIS. This is done by the calls to CURSOR followed by the calls to DEXEC to flush the buffer out to the IIS. In order for this to work properly, the FCB (defined in CLC1.DCL) gets initialized by the INFCB call in ILC1. Note that the task builder command file for TK is set up so that the "real" versions of INFCB, CURSOR and DEXEC (and lower level routines which they call) get included from IILIB.OLB. The linkage versions of all the other user-callable IIS routines get included from DISLIB.OLB. Thus, when RLC1 and MLC1 in TK call CRCTL and WTEXEC, they execute the linkage routines and the actual output is done by the II task.

The DEXEC and WTEXEC routines basically perform the same functions. (I'm not sure, but I think DEXEC might not wait for completion of the I/O. The wait would not occur until the next call that tried to put something into the buffer.) In the TK task, DEXEC flushes out the buffer that is contained in the same task. WTEXEC flushes out the buffer that is contained in the II task. Any applications task should only use the WTEXEC routine.

The above considerations would also be true for the LOC2 logical device if it were to be added to TK. Some of the DEXEC calls in RLOC2 and MLOC2 would need to get changed to WTEXEC calls.

The II task can of course process only one request for IIS I/O at a time. Thus, if II is busy processing a request from the TVLOD task to output to the IIS and a request for IIS I/O is made by the TVDIS task, then the request from TVDIS is queued. The buffer that gets sent out to the IIS in a single DMA transfer may then contain output that was requested by more than one concurrent applications task, but this works fine. Since all of the actual I/O is being done by the II task, there is no problem with multiple tasks trying to access the IIS at the same time. However, there are situations when two tasks may try to access the IIS at the same time. When an applications task calls RBOX1, the request gets processed by the TK task. The "real" RBOX1 code running in TK sets the LC1 logical device to sample mode. This means that ALC1 will send a cursor positioning command out to the IIS each time a tablet AST is received. Meanwhile, the main-line code in RBOX1 will sample the current LC1 value and make calls which request that the II task erase the old box on the screen and draw the new box. There is thus a possibility that both II and the AST routine in TK will attempt to output to the IIS at the same time. In order to prevent this, it was necessary to make some modifications to the low level routines provided by IIS. At those points where an actual I/O transfer is to be done, a call to P(33) was added. The parameter value "33" specifies a global event flag which is used as a semaphore to insure that only one task at a time is permitted to access the IIS. When the I/O is complete, a V(33) is called to release the IIS. Some additional changes were necessary in the M70BF and M70XF routines.

These insure that the 8-word IIS header and all of its associated data block will be output before the V(33) is called to release control of the IIS.

Note that the GENDIS task has not yet been changed to make use of the separate task for handling the VT-11. If this is done, either the SMLCUR routine or the routines it calls will have to be added to VT.

3.15. TASK PRIORITIES

In order the make sure that the user gets good response when he runs the Pipeline Display System, the various tasks are set up to run at priorities that are higher than the default priority. The TK task runs at the highest priority (101) since it must be able to process the AST's from the tablet and keyboard. Next, VT and II run at priority 100. The user-runnable applications tasks (TVLOD, TVDIS, etc.) run at priority 70.

3.16. MISCELLANEOUS

The file DISPLY::[201,223]EDTINI.EDT might be of interest to someone. It sets up EDT so that some of the most commonly used keypad functions can be done by typing control characters. This is nice since you don't have to take your hands off the normal keyboard so often. The following are the new commands:

ctrl-U move cursor up one line
ctrl-N move cursor down one line
ctrl-J move cursor left one word
ctrl-K move cursor right one word
ctrl-F move cursor right one character
ctrl-D move cursor left one character
ctrl-R delete one character
ctrl-E delete one word

4. APPENDIX

4.1. SUMMARY OF GLOBAL EVENT FLAG USEAGE

33 - semaphore for gaining exclusive use of the IIS
34 - set by IMPS when DEMO should exit
35 - set when VT task has work to do
36 - set when VT has processed the request
37 - set when TK task has work to do
38 - set when TK has processed the request
40 - set when II has processed first request packet
41 - set when II has processed second request packet
42 - set when II has processed third request packet
43 - set when II task has work to do

4.2. SUMMARY OF DISK AREAS FOR GRAPHICS FILES

DEC-10 Disk Areas:

[11,12]	Standard system area - includes Omnigraph, Tektronix 4012/4025 plotting routines (LCPLT, LCUTIL), character plotting routines (CHPRIM, CHPLT, etc.), GRFUTL
[11,16]	Jim Torson's work area - includes Tektronix 3D plotting routines (LCPLT3), miscellaneous document files, unmodified Omnigraph sources (in [11,16,OMNI,SOURCE])
[21,16]	Jim Torson's area for submitting files to the standard system - includes old Fits program and miscellaneous document files
[13,16]	Jim Torson's scratch area - currently empty

PDP-11/44 (DISPLY::) Disk Areas:

[1,2]	Command file for IMPS (PREIMPS.CMD)
[7,7] [7,11]	Versatec software - most recent version, (sources and PEPLIB.OLB)
[107,7] [107,11]	Versatec software - old version - should be ignored or deleted
[200,200]	VT-11 software - most recent version, (sources and GLIBVT.OLB)
[200,201]	VT-11 software - old version - used by IMPS, but should be ignored
[201,10]	Standard system area for IMPS - includes utility routines (sources and IMPLIB.OLB), IMPS task files, CRCOP, CHKCAT, MASDEL
[201,201]	Jim Torson's work area for IMPS - includes WHO, DEMO, CATDEL
[201,205]	Arnold Rots' work area
[201,207]	Special versions of IMPS tasks - includes image loading and Dicomed writing
[201,213]	"Games" - includes a simple paint program and color table animation program for the Comtal
[201,217]	Most recent software received from IIS - includes interface and utilities package plus primitives package
[201,220]	UIC that should be used for running IMPS - disk area includes SPACE.CMD
[201,221]	DI-3000 files
[201,222]	DI-3000 files
[201,223]	Jim Torson's work area for Pipeline Display System - includes TIME
[201,224]	IIS software that is being used - includes subset of files in [201,217] (some of which have been modified)

[201,225]	Jim Torson's work area for Pipeline Display System - used for VT, II and TK tasks and logical input device routines
[210,10] [220,10]	Object module libraries used by Pipeline Display System - includes IILIB.OLB, GLIBVT.OLB
[210,47] [220,47]	Utility routines for Pipeline Display System - sources and DISLIB.OLB
[210,50] [220,50]	Pipeline Display System user callable tasks - includes TVINIT, TVLOD, TVDIS, TVVAL, CURVAL, also includes GENDIS, ICCREA
[210,51] [220,51]	CLNMAP, CLNDIS
[210,53] [220,53]	Logical input device routines (ILOC1, RLOC1, etc.), data tablet routines (T1RTNS.MAC, T2RTNS.MAC), keyboard routines (KBRTNS.MAC), object module library that contains all these (INTLIB.OLB), special tasks for handling display devices (VT, II, and TK)