General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

A REPORT ON

THE ST ScI OPTICAL DISK WORKSTATION

NASA Contract NASW3783

6 November 1985

by:

Space Telescope Science Institute
3700 San Martin Drive
Baltimore, Maryland 21218

The STScI Optical Disk Workstation.

1. Introduction.

The STScI optical disk project was designed to explore the options, opportunities and problems presented by the newly emerging optical disk technology, and particularly to see if optical disks would be a viable, and inexpensive, means of storing the large amount of data which are found in astronomical digital images. Rather than perform this effort in a vacuum or on the heavily loaded Institute VAX super-minicomputers, a separate workstation was purchased on which the development could be done. The workstation can also serve as an astronomical image processing computer, incorporating the optical disks into the solution of standard image processing tasks.

The results of our study indicate that small workstations can be powerful tools for image processing, and that astronomical image processing may be more conveniently and cost-effectively performed on small and inexpensive micro-computers than on the main-frame and super-mini computers which have been the standard in the past. The optical disks do provide unique capabilities in data storage, but their use is still somewhat risky given the flux in that technology.

2. Configuration.

The system consists of an OPTIMEM 1000 optical disk reader/writer, a Sun 2/120 microcomputer including a SKY board for floating-point calculations, an International Imaging Systems (IIS) 6500 image display device with 1280*x*1024*x*12 bit image memory and two graphics planes. The OPTIMEM was attached to the Sun using a Small Computer System Interface (SCSI) controller. The Sun was connected to other Institute VAX computers through an Ethernet LAN running the IP/TCP protocols. The terminal ports on the Sun were connected into the Institute complex via a MICOM Micro 600 terminal concentrator.

The Sun was run under Unix 4.2 BSD to which a SCSI driver was attached to drive the laser disk. The basic driver for the optical disk on the SCSI was obtained from the SUN software consulting group and modified at the Institute to our requirements. The IIS came with a set of low level routines for invoking the IIS imaging functions. Software supporting the IP/TCP protocols on VMS was acquired from the Wollongong Group [**3**]. The Sun came with these in place. No other commercial software was acquired for the system, and all higher level interfaces to the optical disk and the IIS were developed locally.

3. Installation and Integration.

The integration of the complete system was delayed several times due to hardware difficulties. Integration of the IIS and the Sun was performed by IIS with the assistance of STScI staff. This was delayed initially by the unavailability of the Model 6500 system, and later by difficulties with the interface board for the Multibus. Minor delays in installation were caused by logistic problems involving the STScI facility moves.

More serious problems arose out of hardware difficulties with the IIS and OPTIMEM. The CPU board originally shipped with the IIS did not support all of the functions of that device. In particular all cursor functions were unavailable. The upgraded board which was eventually sent would hang the system intermittently when the cursor was being interrogated. A third board with full capabilities, but so far without any serious difficulties, has been sent only within the last month. The graphics board for the IIS was also replaced to correct irregularities in the graphics overlays.

The SCSI interface, which was used to communicate with a cartridge tape and terminal lines connected to the MICOM as well as the optical disk, had to be replaced as several of its serial ports broke. This did not occasion any delays but limited the number of simultaneous users of the Sun.

Integration of the OPTIMEM followed the acquisition of the Sun/IIS and the SCSI driver. The disk was brought up soon after the driver was available, however within a few days the disk could no longer be read. Our calls to OPTIMEM determined that the problem was due to degradation of a detector within the disk and that this problem required shipping the unit back to the manufacturer. The unit was returned to the Institute at the beginning of September and its usage since then will be described below. OPTIMEM's plans to switch from disk media manufactured by Thompson-Alcatel to 3-M, will require that the unit once again be returned for an upgrade to make the disk compatible with both media. This is currently scheduled for late November 1985.

The Sun itself, and the Ethernet and Micom links have functioned without serious problems. However the difficulties with the IIS interface board, mentioned above, has required running the Sun with wait states enabled causing a general performance degradation. This is due to the lack of proper handshake lines on the IIS board. The software acquired to support the IP/TCP protocol on VAX/VMS appeared to have difficulty transferring large numbers of files across the net, so that the effective bandwidth between the VMS systems and the Sun could be reduced to 5 per cent of what was achievable between two UNIX systems.

4. Optical disk development.

The optical disk system has provided novel challenges in its implementation. Prior to any software development, we attempted to define requirements for a general optical disk format. The use of write-once media is still very limited, and there are no commonly accepted standards. After communications with a number of groups, both in Europe and the U.S., we arrived at a set of minimum requirements for optical disk formats when they are to be used for archival storage. On the basis of these requirements we developed a complete optical disk format (see appendix). While the format was largely tailored to our

future archiving requirements, it has been shown to the National Bureau of Standards, and may form part of the basis for future NBS discussions of optical-disk standard data-structure formats.

Once we had decided on the optical disk format, it was possible to proceed with the development of the software to implement it. Currently the software has been implemented at a relatively low level. A set of routines which emulate the standard UNIX I/O calls have been developed and tested. These calls provide for the creation of a complete file system on the optical disk including directories and files. Redundancy in organizational information helps to ensure disk integrity in case of disk errors. Transparent use of magnetic disk files to shadow the directory of the optical disk allows efficient access to any file on the laser disk.

Since the optical disk I/O procedures closely follow the pattern of the UNIX I/O interface, the integration of the optical disk access into the UNIX kernel should not be difficult. However, this will await the resolution of certain difficulties with the disk.

The performance of the optical disk has not been entirely satisfactory. An apparent synchronization problem in either the driver or the hardware currently requires the user to attempt to read a sector before each write. Reading an unwritten sector is an error and the processing of this error slows the actual data transfer rate to little more than 10 K bits/s. While acceptable in a test environment, this must be corrected before more extensive applications can be attempted. The cause and solution to this problem is under discussion with OPTIMEM. When the disk is written to without the initial read, the data rate jumps to as much as 80 K bytes/s, depending on the block size used, but the error rate increases from effectively no errors (fewer than one error in 5000 blocks) to as high as an error every twenty blocks.

Notwithstanding the problem discussed above, the optical disk has been used for storing astronomical images. Writing a reasonably sized image to the optical disk takes several hours, due to the slow transfer rate, but the image may be read back within two minutes, typically a factor of 2-5 slower than magnetic disk. We expect that this time may be reduced by better blocking of the data. Software has been written which will read data directly from the optical disk to the IIS. We plan shortly to transfer the GSSS scan of an entire Schmidt plate (about 400 MB) to the optical disk; this was an early application of the system discussed in the original proposal. The software developed for the IIS would then allow any portion of that plate to be displayed on the IIS. However this effort has been deferred until the resolution of the optical disk synchronization problems, since at the current rates it would take about 4 days to write the image.

Further integration of the optical disk will be done at both the system and user levels. Inclusion of the fundamental I/O calls in the UNIX kernel will allow general access to the optical disk by all existing software. High level routines are being written to take advantage of the ability to store very large files.

5. Workstation Applications.

A number of software packages have been developed for, or adapted to the Sun workstation. By far the most important effort has been the porting of the entire IRAF (Image Reduction and Analysis Facility) system to the Sun. IRAF is a powerful, flexible system

specifically designed for analysis of astronomical images and spectra [1,2]. It includes its own command language, sophisticated means for parameter passing and retention, i/o libraries and an extensive suite of programs for analyzing astronomical data. IRAF will be the standard command language used for all astronomical data reduction at the STScI.

IRAF was specifically designed to be a portable system, but the initial effort to bring it up on the Sun uncovered a serious hardware dependency in the then current implementation. This dependency was eliminated, and the IRAF system was successfully ported to the Sun. The only difficulty encountered was with software purchased for the VAX to support the IP/TCP protocols on Ethernet. The Ethernet link is the primary external I/O link to the Sun, but the VAX/VMS software limited its bandwidth to less than 5 per cent of what we found to be achievable between two UNIX systems.

The version of IRAF which was brought up on the Sun initially did not support the IIS as an image display or graphics device. This was rectified completely with regard to the graphics capability and partially with regard to imaging. Currently the initial copying of an image to the IIS from disk is not completely integrated with IRAF. A special task must be invoked, but this can be done from within IRAF. Once the image is displayed on the IIS the standard IRAF functions may be used. All of the display functions are available including image zoom and pan, look-up table manipulation, and cursor read out.

The Sun/IIS system as a whole has run well. Throughput is typically at least as good as on the STScI VAX 8600's with a normal load, and during the busiest hours in the early afternoon, the Sun can be much faster. Since IRAF was originally developed on a UNIX system, certain features of IRAF, particularly the spawning of sub-processes, are done more efficiently on the Sun than on a VAX, regardless of the difference in the fundamental speeds of the machines.

The initial difficulties in bringing IRAF up on the Sun led to our developing an independent image display capability as a testbed. This package combines the extensive image display capabilities of the IIS with the graphics windows supported on the Sun terminal to provide a particularly powerful and convenient tool. Basically a window on the Sun terminal is used to graphically represent what is being done on the IIS. For example a Sun window may plot the current look-up tables being used on the IIS. Or if the file being displayed is too large to fit within the IIS, the Sun window may indicate the position of the region being displayed. The Sun supports multiple independent windows and several may be used to refer to different aspects of the current display. The unique capabilities of this system have led us to adapting it into a package within IRAF.

6. Summary and Recommendations.

The Sun is now available as a powerful imaging workstation. The latest version of IRAF is completely implemented on the system. Additional image manipulation software is available either independently or within IRAF packages. Processing image data on the Sun is as fast or faster than on an Institute Vax.

The laser disk is also working, but the writing of data files is very slow and only a few high level programs are available for transferring files to and from the laser disk. The major outstanding issue is the laser disk synchronization problem which may require return of the disk to the factory for repair. In any case the drive must be shipped back to OPTIMEM to

be made compatible with the media being produced by 3-M which are to be the standard media for this drive.

Another significant difficulty in using the Sun for scientific applications, is getting the data to and from the Sun. Although there is an Ethernet link with the Institute VAX's, the VMS software is slow and unreliable. Also a user must first be able to read the file onto a disk on the VAX before it can be transferred over the Ethernet. For large files, and for tapes written in the UNIX TAR format, this can be difficult. The most direct solution to this problem would be to purchase a standard tape drive for the Sun. In addition to addressing the data transfer problem, this would also permit backups of the sytem onto half-inch tape.

Recently a VAX 750 running UNIX was added to the STScI LAN. Since UNIX network protocols permit the use of a remote tape drive, it may be possible to use this VAX's tape drive for the Sun, but we have not yet had experience doing this.

The optical disk workstation in combination with the IIS image diplay device has already proven to be a capable system for astronomical image analysis. We have already had a number of enquiries about using the system for storing very large image files, or a series of smaller (e.g. 20 MB) files. With the laser disk, the workstation combines processing power with a remarkable capacity for data storage permitting projects which would be unfeasible even on much larger systems.

In order to promote easy data interchange within the astronomical community we expect to send out copies of our disk structure for review at several sites. A standard for the archive media that supports all standard astronomical formats, including FITS [**4**,**5**], is expected. Further investigation of the performance issues and of media portability across nominally compatible machines must also be carried out. Much of this effort will be carried out as part of a prototype optical-disk based data management project for the Hubble Space Telescope [**6**].

Acknowledgements.

The hardware for this project was acquired under NASA contract NASW 3783. Personnel support was funded by NASA contract NAS5-26555.

References

[**1**] Douglas Tody, *A Reference Manual for the IRAF System Interface.* National Optical Astronomy Observatories, May 1984.

[**2**] Peter Shames and Douglas Tody, *A User's Introduction to the IRAF Command Language.* National Optical Astronomy Observatories, May 1985.

[**3**] *IP/TCP for VMS-User's Guide, Release 2.1.* The Wollongong Group Inc., March 1985.

[**4**] D.C. Wells, E.W. Griesen, R.H. Harten, "FITS: A Flexible Image Transport System", *Astronomy and Astrophysics Supplement Series* **44**, 363, 1981.

[5] E.W. Griesen and R.H. Harten, "An Extension of FITS for Groups of Small Arrays of Data", *Astronomy and Astrophysics Supplement Series* **44**, 381, 1981.

[6] Thomas McGlynn, Guido Russo, Alan Richmond, Peter Shames and Francoise Ochsenbein, *Data Management Facility-Design Guide*, Space Telescope Science Institute, October 1985.

A. Appendix–An Optical Disk Format

A.1. Header block.

The header block normally begins at sector 500000, and 100 sectors are reserved for the header block. The header contains 2 sectors of information with the additional 98 sectors being reserved for handling errors and for future expansion. The format of the two sectors are:

Sector 0:

Byte	Format	Value	Explanation
0-3	A4	"HDR1"	Key for first header block.
4-35	A32	Disk name	Unique label for optical disk.
36-51	A16	Disk format	Format disk was written in.
52-63	A12	Disk owner/creator	E.g. a site ID.
64-67	A4	Year	
68	A1	" "	
69-70	A2	Month (1-12)	Date disk opened.
71	A1	" "	
72-73	A2	Day	
74-76	A3	" "	
77-78	A2	Hour (00-23)	
79	A1	":"	
80-81	A2	Minute (0-59)	Time disk opened.
82	A1	":"	
83-84	A2	Second (0-59)	
85-89	A5	" "	
90-93	A12	Directory sector_0.	First sector in initial directory allocation.
94-97	A12	Directory size.	Number of sectors in initial directory allocation.
94-127	A30	Blank	
128-511		Unused (reserved)	
512-1023	A512	Comments	(optional)

Sector 1:

0-3	A4	"HDR2"	Key for second header block.
4-63		As in first sector	These must match with sector 0.

64-84			As above but for date/time disk closed.
85-127		Blank	
128-511			Unused (reserved)
512-1023	A512	Comments	(optional)

The first sector of a header must be written when the disk is first initialized, while the second is written when the disk is "full", i.e. when no more files are to be written to it.

A disk may be in three states with regard to the header:

1. No header has been written. This is signalled by errors in attempting to read sectors 500000-500010, any program aside from the disk initiation utility should fail.
2. Only the first sector has been written. Here an open call should find HDR1 in sectors 500000-500010 followed by at least 4 blank sectors. I.e. the HDR1 sector should be successfully read and the following four sectors should return an error. The disk may be opened for either reading or writing.
3. Both sectors have been written. Here a HDR1 sector should be immediately followed by a HDR2 sector which in turn should be followed by 4 blank sectors. The disk may be opened read-only.

If an error occurs in writing the first header sector, the disk initialization program will try subsequent sectors until it reaches sector 500010 at which point it will fail.

If an error occurs in writing the second header sector, the disk finalization program will attempt to re-write the first header sector after the sector which caused the error and after successfully rewriting the first header sector will retry the second sector.

A.2. Directory.

The directory consists of an initial allocation of contiguous blocks which will normally follow the header block at sector 500100 and additional allocations which may be needed if the disk contains many small files. Note that the first and last block of the initial directory allocation should be allocated as a file header and trailer in the same fashion as later directory allocations (see below). Each sector in the directory points to a file except that the 5 sectors in each allocation are reserved as a pointer to the next allocation. The format of each directory block is:

Byte	Format	Value	Explanation
0-3	A4	"DIRE"	Directory key.
4-35	A32	File name	Including version, etc.
36-51	A16	Directory format	
52-63	A12	File owner/creator	
64-84	A21	Date/time file created.	
85	A1	" "	
86-89	A4	File type	Fixed, fixed blocked, variable.
90-97	A8		Reserved for internal descriptor
98-103	A6	Number of extents	
104-115	A12	Record length	For F and FB records.

116-127	A12	Bytes in the last sector	FB and V records
128-255	A128	Comments	
256-511	Unused		
512-515	A12	Sector_0 of first extent.	
516-519	A12	Sector_n of first extent.	
520-523	A12	Sector_0 of second extent.	
524-1023		...	

The end of the directory is signalled by 5 consecutive blank or error sectors with the last 5 sectors of the current allocation also being blank or error.

A pointer to a new directory allocation has exactly the same format as a normal directory entry except that the name of the file is *##Directory_allocation_?* where *?* should be replaced by the number of the directory allocation. The additional allocation only has one extent and unlike other files it will not be written into until after the directory record itself has been written. However the file header and trailer for the additional allocation should be written immediately. Thus the additional allocation appears exactly as another file except that it must be contiguous.

A.3. Data.

Note: Procedures for reading and writing are given assuming that there are no ancillary magnetic disk files used. Normally such files will be used to facilitate efficient access to the optical disk, but the structure of the optical disk should not be dependent on their existence.

An attempt to write a new file to the optical disk must specify the size, format, internal format identifier, and name of the file. The name must be unique on the disk. For F and FB records the record size must be specified also. Files are allocated from the center of the disk outward, alternating from the inner half to the outer half, using the half with the most available storage. If a file can be accommodated only by using both halves, it will be written in two extents. The procedure for writing a file (not using ancillary magnetic disk) is as follows:

1. Read through the directory of the disk noting files closest to the edges of the disk.
2. Check that no additional data has been written on the disk (i.e. that a program didn't crash with a file half written) beyond the two limits. (**NOTE**: This won't always work for the outer edge–sectors 0 to 500000– because the sectors which might have been written upon could be a separated from the main portion of the written disk by many sectors. This requires use of ancillary magnetic disk files for proper handling.)
3. Determine which edge to use. If only one edge can accommodate the file choose it, if either choose that with the most space, if both edges are needed prepare to write the file in two extents, and if insufficient space on disk, crash. For this calculation add at least 4 sectors and 1% of the file size to accommodate errors.
4. Write the file skipping the first sector. When an error occurs attempt rewrites on subsequent sectors up for up to 10 consecutive sectors. Keep info on where errors occurred in memory for use in filling in extent info in file header, file trailer and directory.

5. Write file header and file trailer. These have formats identical to the directory format but have the keys "FHDR" and "FTLR" respectively. If both of these fail write a directory entry for a file with the name *##ERROR_FILE_name*, where name is replaced by as much of the original file name as will fit, then abend.

6. Write the directory entry for the file. If there is an error in writing a directory block retry on subsequent blocks until there are 10 consecutive errors at which point abend.

Three formats are supported for data, fixed unblocked, fixed blocked and variable-length blocked data. Fixed-length unblocked records are written with each record beginning a new sector, possibly ignoring the ends of sectors. Fixed-length blocked records are written as a continuous byte stream, with a subsequent record beginning on the next byte after the previous record. Records may span blocks (or even extents). Each variable length record is preceded by a four-byte prefix giving the size of the record, including the four bytes of the prefix. As with FB data the next record begins on the byte after the last byte of the previous record so that V data is automatically blocked. Note that the only blocking length available at this level is the sector size of the optical disks, 1024 bytes. If the user wishes to block records to some other blocksize then other routines must block the data and each of these blocks must be sent to the optical disk as a record.

Note that the size of the file given when the file is opened is the maximum number of sectors that the file will use. It need not be accurate but it is an error to attempt to write beyond this limit and an overlarge specification may cause space to be wasted on the disk.

When a file is to be read from the optical disk the user need only specify the file name. The procedure is as follows:

1. The directory is searched sequentially until the appropriate entry is reached. If 5 consecutive directory blocks return an error, and there is no additional allocation for the directory, the end of the directory has been reached and the file is not on the disk.

2. The next four sectors after an apparently correct entry are checked to see if any of them refer to the same file, if so the first entry is erroneous and step two is repeated with the new entry.

3. Data in the directory and header are compared to ensure that all directory information is correct. If the directory does not point to a header block, then abend. If the information in header and directory disagrees, then the trailer record is used as a deciding vote. If it disagrees with both then abend.

4. Data is read using directory information to skip bad blocks. If an error is found reading a block which is not labelled as bad then retry 5 times, then return an error to the user but do not necessarily abend.

5. When an attempt is made to read past the last record return EOF.

A.4. Efficiency and Directory Structure Considerations.

The directory structure which has just been described is a simple non-sorted flat file which is written as a linked list. This is non-optimum for speed or convenience of access, but is a direct consequence of the write-once media being used. During normal processing

of a data disk it is expected that the directory sectors will be scanned when the disk is just mounted and a sorted random access directory structure created on magnetic disk (or in memory) to speed processing.

The file naming structure is equally simple in structure, but can be used to create a logically hierarchical view of the data on a disk. The fully qualified name of any file on disk is *disk!filename;version*. Using the site name and the user name as part of the fully qualified name yields: *site!disk!user!filename;version*. This is sufficient to construct a nearly globally meaningful reference. Implementations that wish to preserve the simplicity of the flat file approach can use the raw naming convention. For those applications where a more structured view of the data is required, the fully qualified name may be used to unambiguously refer to any data set stored on disk.