PEPSE® On The PVNGS SUN Network

Ken Porter
Frank Todd

Arizona Public Service Company

PEPSE ON THE PVNGS SUN NETWORK

Ken Porter
Senior Performance Engineer
Palo Verde Nuclear Generating Station
Arizona Public Service

Frank Todd
Performance Engineer
Palo Verde Nuclear Generating Station
Arizona Public Service

ABSTRACT

The goal of Arizona Public Service Company is to be one of the top five electric utility providers by 1995. A key component in the fulfilling of this goal is maintaining the highest Capability Factor for the three 1340 MWe (gross) Units at Palo Verde Nuclear Generating Station (PVNGS). In order to fulfill this goal, timely technical support must be provided to the Operations Department.

In 1989, PVNGS Engineering Management decided to purchase a Unix based networked computer system. As a result, data organization and availability have been greatly improved and various analysis tools are available to the Performance Engineer. One of the central analysis tools used by the Performance Engineer is the PEPSE performance code. The goal of more timely and efficient analysis was hampered by having to transfer data from a Unix system to a MS-DOS computer to perform the required analysis. Therefore, it was decided to move (port) the PEPSE code to the Sun Unix Network.

The purpose of this paper is to expose other performance engineers to a new technology and aid those who want to use PEPSE on a Unix based system.This paper describes the advantages of using the Sun network for thermal performance analysis and the methods used to port the PEPSE code from a personal computer MS-DOS operating system to the Unix operating system on the Sun network.

Acknowledgments

The following individuals also deserve to be mentioned for their effort to this project. Don Hildebrant of United Energy Services made most of the changes to the source code and did most of the compiling and testing. Without the dedicated effort of Don this project would not have been possible. Jerome Lamer of Arizona Public Service wrote the C routines required for getting the PEPSE output to a window (screen). The support of Ron Griebenow, Greg Rice and Gene Minner of Haliburton-NUS was also very helpful and greatly appreciated. And finally, the editorial and technical work of Jodi Colville of Arizona Public Service was crucial to the completion this paper.

Introduction

In 1991 the PEPSE code was ported from a MS-DOS platform to a Unix based platform. While most of the changes were relatively simple and tedious some required a lot of thought and effort. One of the purposes of this paper is to provide lessons learned for other utilities who may be considering porting PEPSE to a Unix environment. It is hoped that these lessons will enable other utilities to spend less time and money in the conversion process.

Once PEPSE was ported to a Unix environment, the advantages of using PEPSE in a multi-tasking and LAN environment became evident. For example, PEPSE input decks are able to be created without moving the data between a personal computer and a workstation. Also, PEPSE input and output decks can now be accessed by multiple users at different workstations. And finally, there has been a significant reduction in the run time of the Palo Verde model.

The Man-Machine Interface and the PEPSE Interactive Processor (PIP) feature were not ported to Unix because of the extra time and effort required to convert these features to the Unix environment.

Description and History of Palo Verde

Palo Verde is a triple-unit standardized nuclear energy facility located 55 miles west of Phoenix, and generates electricity for approximately 4,000,000 people in four states; Arizona, New Mexico, Texas and California. Each of the project's three reactors have the capacity to generate 1,270 net megawatts of electricity, making it the second largest nuclear generating station in the world. Unit 1 began commercial service in March 1986. Unit 2 began commercial service in September 1986, and Unit 3 began commercial operation in January 1988.

Each unit consists of a Combustion Engineering System 80 Nuclear Steam Supply System, a General Electric N2 Turbine-Generator, four moisture separator reheaters with two stages of reheat, two General Electric feedwater pump turbines coupled to two Byron-Jackson feedwater pumps, a multipressure condenser, seven stages of feedwater heating, and three Marley forced draft, round, concrete cooling towers.

Palo Verde is unique because its the only nuclear energy facility in the world that uses treated sewage effluent for cooling water. Palo Verde buys wastewater from local cities in the Phoenix metropolitan area and recycles it at the PVNGS Water Reclamation Facility. Treated sewage effluent is pumped from the 80 acre storage reservoir to the cooling towers to replace water that has been lost due to evaporation. Reuse of wastewater is both environmentally sound and a substantial economic benefit for nearby communities.

Description of the Thermal Performance Group

The Thermal Performance group at Palo Verde is located on-site and is part of the Reactor Engineering group. Two engineers and two technicians work full time on performance issues including supplying the data needed for the NRC Monthly Operating Report and the INPO Performance Indicators.

There is no data link between the Plant Monitoring System (PMS) computer and the Sun Engineering LAN. Data is collected daily from the plant computer in each of the unit control rooms and put on magnetic tape. The data is transferred to a 5.25 floppy disk, brought down to the office, transferred to a 3.5 disk and then uploaded to the LAN. Other sources of data are hard copy of hourly logs produced by the PMS computer and operator logs.

ASME testing or other testing is not performed due to a lack of resources. However, the plant is well instrumented and most of the instrumentation is in good condition. Therefore, all performance monitoring is done with data from permanent plant equipment.

Description of the Sun Local Area Network

The purpose of the PVNGS Sun Local Area Network is to provide distributed resources and enough power to the users' enabling them to perform complicated analysis and compute intensive functions. The concept of a network of various workstations which can share resources and provide various levels of computing ability was developed to meet the needs of the PVNGS Engineering organization.

The workstations are served by several "servers". A server is a computer (usually more powerful than a work station) which keeps track of all system files and application programs. Many types of computers have been connected to the LAN. We have installed Sun SPARC +1 stations, IBM PS2s, generic IBM clones, Compac computers, Sun 386i's, diskless Sun stations called SLC's, and even some Apple computers. The system provides for versatility.

Generally, a workstation is a computer that has enough disk space, memory, and central processor to perform all the required operating system functions. Sometimes applications may be tightly bound to a particular workstation, however, most of the application programs reside on the servers and the workstations just use them. Workstations are also used to host a printer or some other peripheral device such as a scanner. Although not normally configured as such, workstations can operate on a stand-alone basis. Workstations are also used as servers in certain situations. The workstations have at least a 104 megabyte hard drive and a minimum of 8 megabytes of random access memory.

PVNGS uses the open system concept. Workstations are not bound to a single user or even to a user being at the workstation. Most workstations can be accessed freely through the network meaning any person with an account can access their files from any workstation. This homogenous system contributes to a system that is very flexible.

The servers used on the PVNGS LAN are Sun 4/490 computers although some are SparcServers (usually a SPARC 2). The server provides disk space for applications, user directories, and common data areas. Mass storage devices such as tape drives and CD ROM devices are attached to the servers. Networking devices such as gateways to the mainframe computer, modems and fiber optics are also attached to the servers. All the data files that define the Network configuration are located on the servers, therefore, the server provides the computing resources to run the processes that control the network. One disadvantage is that if a server fails or is taken out of service part of the network may stop working.

There are many peculiar terms that are used to define how the network is controlled: TCP/IP; Network Databases, NIS; NFS; Automounting and Daemons. Generally speaking all these terms refer to the way in which a user on the system is able to access and utilize all the resources of the system. What is intended here is a general overview of how the system works:

TCP/IP - Transmission Control Protocol and the Internet Protocol. Each node (or host) on the network has a unique Internet address defined by the Ethernet controller chips on the central processing unit chip or the Ethernet controller board. Therefore the TCP/IP defines the location of every device.

Network Databases- Network Databases define the network to the computer. They are simply flat ASCII files (a readable list of information).

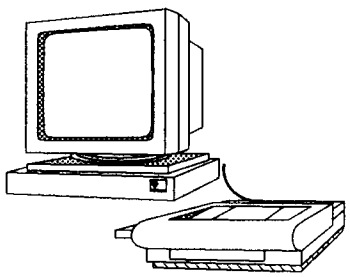
NIS - Name Information Service (or Yellow Pages)- This is a central method of keeping track of all the Network Databases on all the nodes without having to update all the nodes for every change to the configuration.

NFS - Network File System- This is the Sun interpretation and implementation of the distributed file system concept. Different computer systems use one form of NFS or another. This is how you can have many different file systems on various resources yet be able to have access to all those file systems and resources.

Automounting- The process of automounting allows the user to log on to any workstation without having to be aware of where their data and applications reside. When a host (workstation) is idle it has no NFS directories mounted. When a user logs in, the home directory is required, so it is automounted. As the user accesses applications and other common data areas, those resources are automounted. When a resource has not been used for a certain amount of time, it is unmounted.

Unix Networks do require a large amount of overhead to maintain. As can be seen from Figure 1, the Arizona Public Service Network is fairly large with links into many types of computer systems. The company has made a commitment to the concept of distributed information and application systems. The advantages of using Local Area Networks can be realized without such a complicated and encompassing implementation. Local Area Networks can be implemented much simpler in smaller applications.

The thermal performance area on the Local Area Network consists of a set of subdirectories that contain all the specific application programs and data files for the Thermal Performance group. A separate menu system was developed so that most of the application programs are available by way of a menu pick with a mouse. Unix provides intrinsic file security so that anyone connected to the Local Area Network can access the data without having the ability to alter the original files. All the application programs are also available for anyone on the LAN to use.



The Workstation

- ***Contains an individual copy of the Operating System***
- ***Provides memory and a processor for local execution of all processes***
- ***Provides 'swap space' locally***
- ***May host a printer, and provide a 'spooling area' for that printer***
- ***Some applications are tightly bound to a particular workstation. Those workstations house the configuration files that are necessary for those applications to run. An example of such an application is DOS Windows.***

Sun WAN Diagram

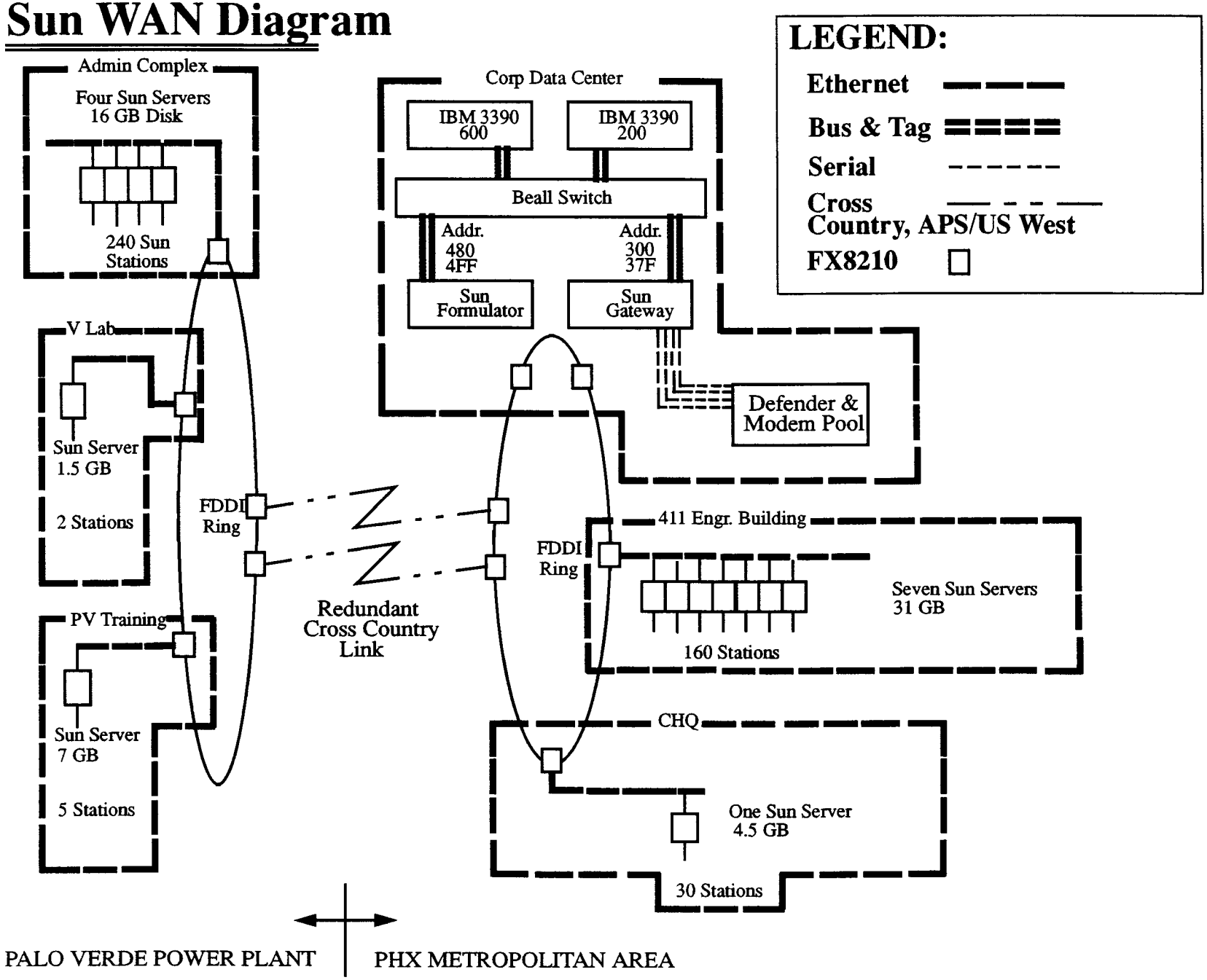
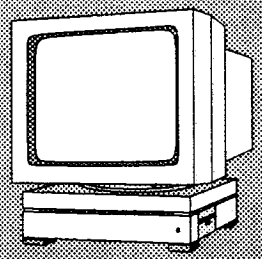


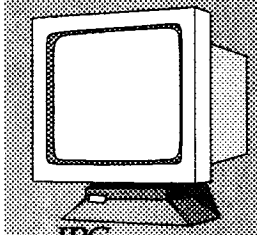
Figure 1

PV_LAN Setup for the New Administration Complex Buildings at Palo Verde



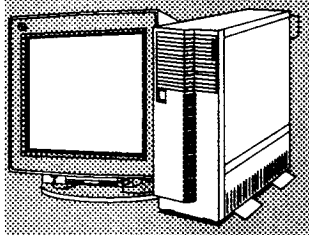
Sparc1+

There are 182 SparcStations connected to the LAN.



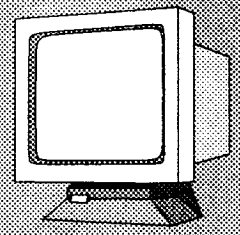
IPC

Four (4) IPC's are connected to the LAN.



SUN386i

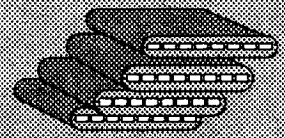
There are 36 SUN386i workstations connected to the LAN.

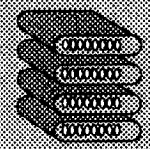


SLC

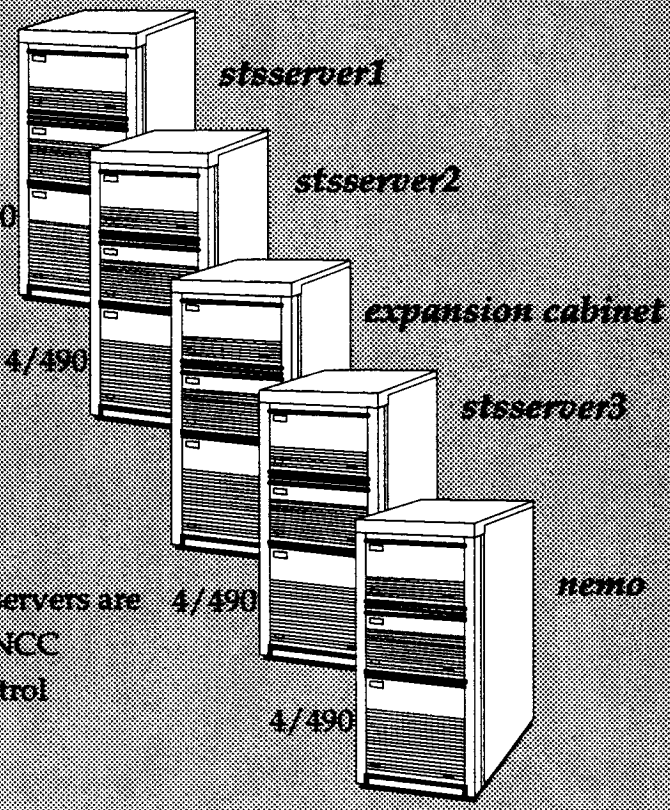
There are 4 diskless SLC's connected to the LAN in the Technical Publishing group.

4 19.2 kBPS data links are connections to the IBM mainframe, are connected to Server 1, 2 and nemo allows 120 concurrent 3270 terminal emulation sessions.





4 Asynchronous modems are used for dial-out connection



The 4/490 servers are located in the NCC "Network Control Center".

Figure 2

Use of Local Area Networks in Performance Analysis

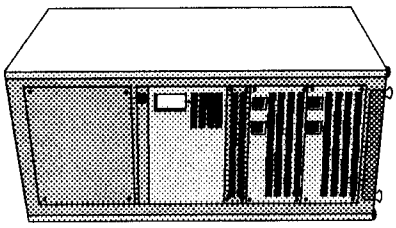
One of the major motivations for porting the PEPSE code to the PVNGS Sun Local Area Network was the ability to create an integrated analysis system. PVNGS has an abundant amount of installed plant instrumentation so a large quantity of data is available to the thermal performance engineer. If this data is not organized or if the tools are not available to analyze the data, then the data is useless to the thermal performance engineer. The PVNGS Thermal Performance group is utilizing the many organizational advantages of the Unix operating system on a Local Area Network and applying the various analysis tools to form an integrated method of monitoring and analyzing thermal performance of the three units at the plant.

Files are maintained for each fuel cycle for each unit and can be easily accessed to help in plant analysis. Graphs can be quickly obtained for any of the performance parameters. Thermal performance data can be easily moved to other tools on the PVNGS Sun Local Area Network for further analysis. Plant data can be put into a PEPSE deck and analyzed in a manner of minutes. This allows for the rapid response to a particular problem that a unit may be experiencing. At this time, PVNGS does not have a computer link to the Plant Monitoring System and this is the weakest part of the program.

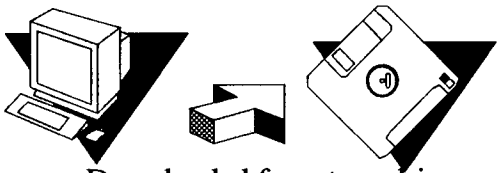
The key aspect of this versatility is the involvement of the thermal performance engineer in the entire process. The system consists of easily accessible tools that a thermal performance engineer can use. The system is not automated and the data is accessible. By using a spreadsheet to store and organize the data it can be easily manipulated for problems such as non-normal operating conditions, instruments out of scan, or instruments which are reading incorrectly. Normally the plant computer performs certain performance related calculations automatically in its balance of plant performance calculation package. Often there is a problem with a particular instrument or a calculation is not performed. In these cases, the calculations can be performed in the spreadsheet using the raw data values. Thus a trend may continue whereas in the past it may have been lost.

Another significant advantage to using Local Area Networks is the accessibility of the data by anybody with access to the LAN. System engineers have easy access to the data, but they can be restricted from altering the original data files. If a problem is detected with a feedwater heater the performance engineer can call up the system engineer and the two of them can look at the same data at the same time on different workstations.

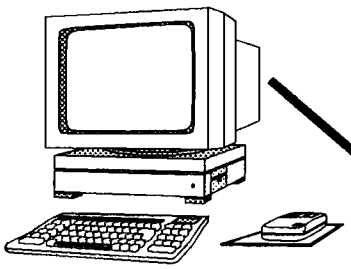
The PVNGS Sun Local Area Network provides access to many tools for the thermal performance engineer. LOTUS 1-2-3 for Sun is the preferred spreadsheet. Another spreadsheet package, WINGZ, is also available. The RS/1 statistical package is



Data Snapshot taken from Plant Computer



Downloaded from tape drive and transferred to floppy disk



Data is brought back to office - files are delogged and uploaded onto database.

Local Area Network enables data to be easily accessible to other personnel/departments.

Data is graphed and trended- 2 Sigma calculations are done. Engineering analysis is performed on all abnormal findings. Engineering calculations are also performed, as well as PEPSE runs when needed.

Questions/inquiries are sometimes made to other departments/system engineers, in an effort to find the cause for the abnormalities found.

Problem Identification and possible solutions/ suggestions are made.

Figure 3

available for statistical analysis when in-depth analysis is needed beyond what is available in LOTUS 1-2-3. FRAMEMAKER is used for word processing and desktop publishing. Other software tools are also available to perform flow calculations, heat balance calculations, pump calculations, condenser calculations, and steam table lookups. PEPSE can also be used in a much more interactive way. Two decks can be run at the same time and then compared to each other side by side on the computer screen. If needed, the entire deck can be put into a text editor or a section can be cut out and added to an already existing document. Stacked PEPSE decks can be run and the results graphed or analyzed very quickly.

Much of the data used in thermal performance comes from various sources; procedures, plant logs, PMS snapshot data, etc. On a Local Area Network the data analysis can use all these sources of data together. Comparisons can be made and calculations using parts of all the data sources can be performed. Historical data can be accessed very quickly to make comparisons.

The end result is that the thermal performance engineer can solve problems much more efficiently and quickly by having all the data and all the tools available at the same place. More time can be spent on thinking about a problem then on how to get at the data or tools to solve a problem. This is a method that has been developed over many years at PVNGS.

An area which has great potential for benefit is the integration of the application programs. The output of PEPSE runs can be automatically input into other programs such as data trending, statistical analysis, report generation or other engineering analysis software.

Performance Gains

A significant gain in run time was achieved by porting PEPSE to the Sun SPARC stations. Before discussing these gains, a description of the model will help put the run times in perspective (see Figures 4 - 9). The standard model at Palo Verde has 167 components and 268 streams.The model contains four HP turbine stage groups, three LP turbines with 6 stage groups per turbine, two HP heater trains with 3 heaters per train, three LP turbine trains with 4 heaters per train, four MS/Rs with one stage of moisture removal and two reheat stages. All of this adds up to 22 turbine stage groups, 6 HP heaters and 12 LP heaters, four 2nd stage reheaters, four 1st stage reheaters and 4 moisture removal chevrons. In addition to this the model also contains two feedwater pumps, two feedwater pump turbines, three condensate pumps, two extraction drain pumps, six combined intercept valves, four main stop valves and three condensers. A model of the steam generator blowdown system, including a flashtank and heat

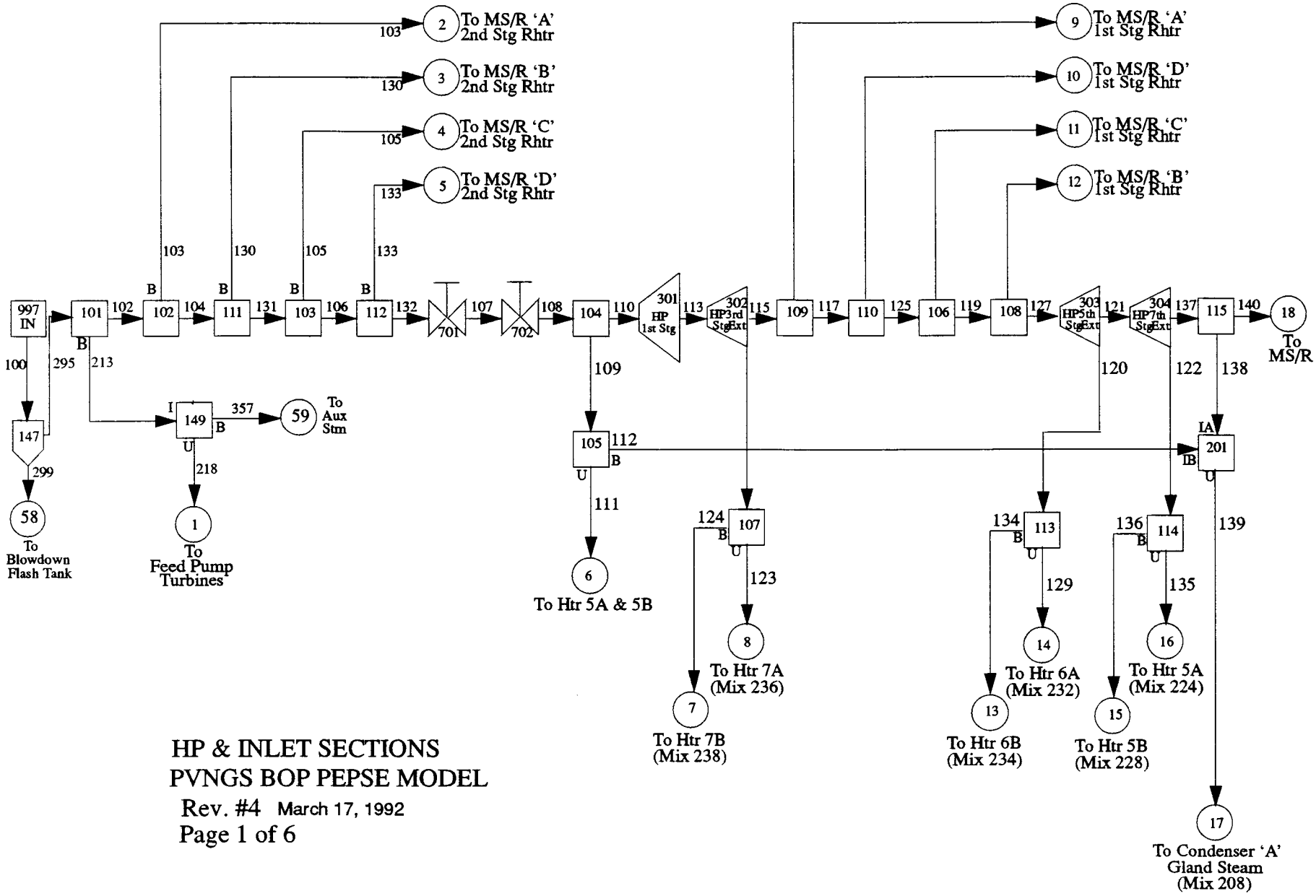


Figure 4

HP & INLET SECTIONS
PVNGS BOP PEPSE MODEL
Rev. #4 March 17, 1992
Page 1 of 6

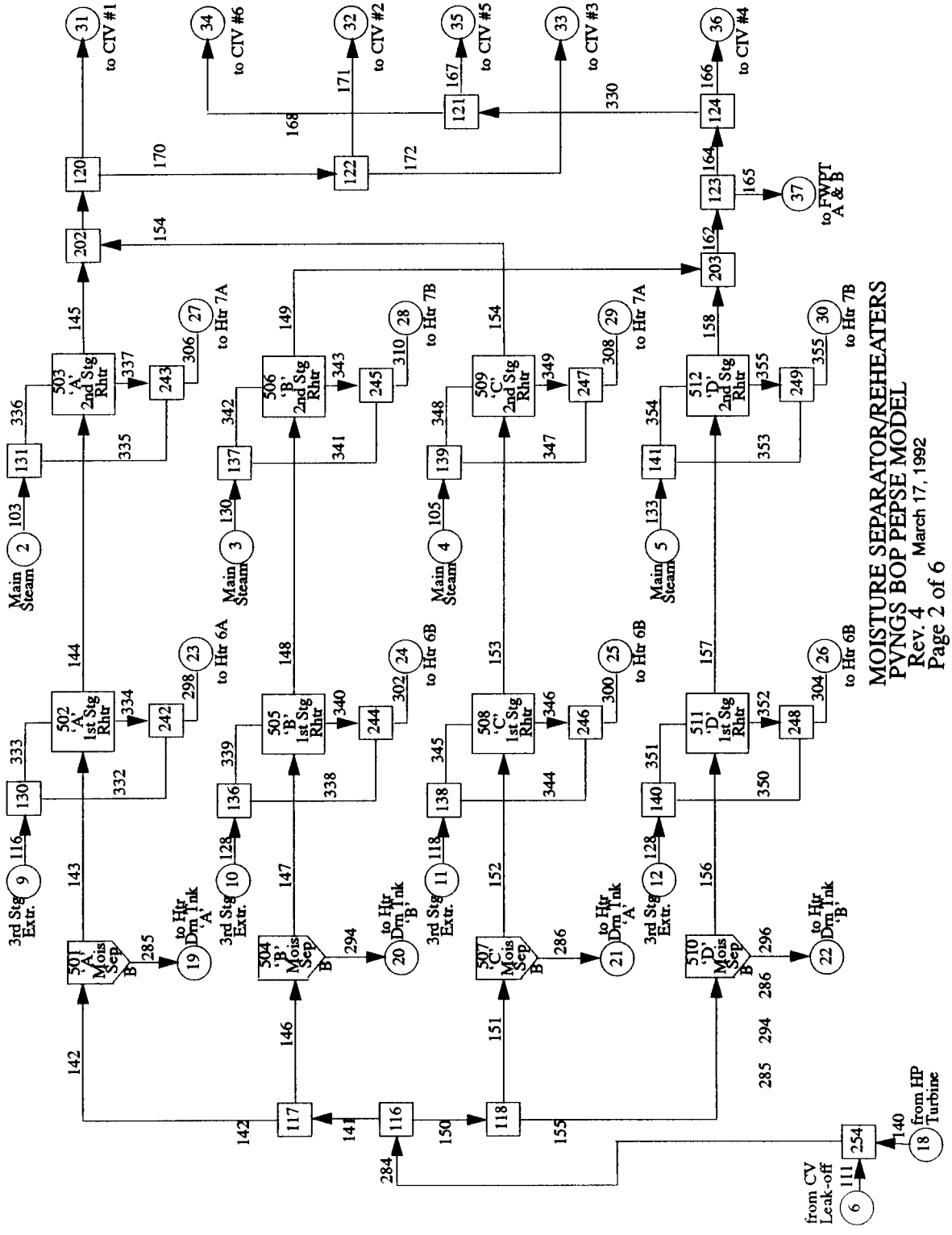

MOISTURE SEPARATOR/REHEATERS
PVNGS BOP PEPSE MODEL
Rev. 4 March 17, 1992
Page 2 of 6
Main Steam
3rd Stg Extr.
to Htr 6A
to Htr 7A
to Htr 6B
to Htr 7B
to Htr Drn Tnk 'A'
to Htr Drn Tnk 'B'
501 'A' Mois Sep
502 'A' 1st Stg Rhtr
503 'A' 2nd Stg Rhtr
504 'B' Mois Sep
505 'B' 1st Stg Rhtr
506 'B' 2nd Stg Rhtr
507 'C' Mois Sep
508 'C' 1st Stg Rhtr
509 'C' 2nd Stg Rhtr
510 'D' Mois Sep
511 'D' 1st Stg Rhtr
512 'D' 2nd Stg Rhtr
to CIV #1
to CIV #6
to CIV #2
to CIV #5
to CIV #3
to CIV #4
to FWPT A & B
from HP Turbine
from CV Leak-off

Figure 5

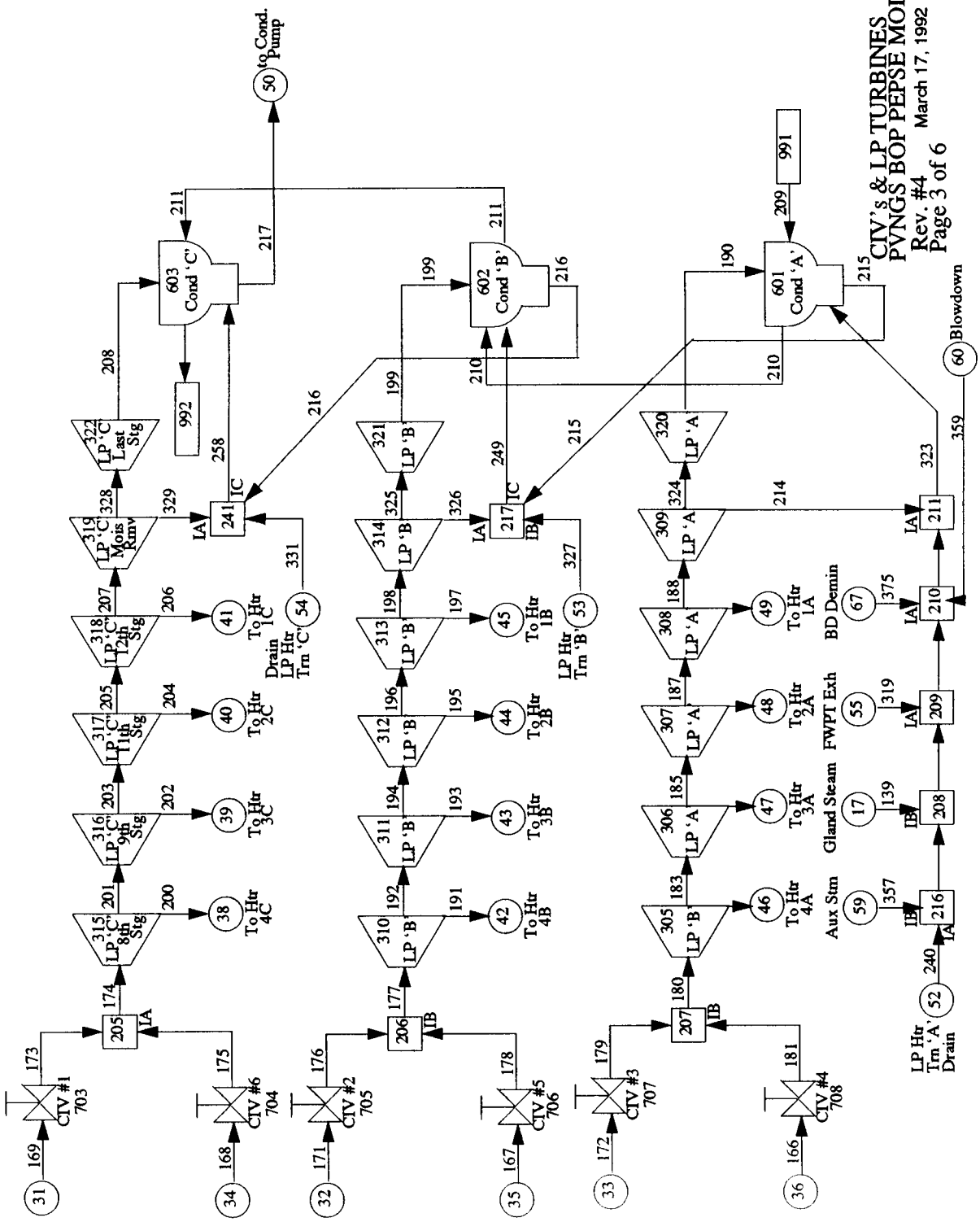

CIV's & LP TURBINES
PVNGS BOP PEPSE MODELS
Rev. #4 March 17, 1992
Page 3 of 6
to Cond. Pump
Blowdown
Cond 'C'
Cond 'B'
Cond 'A'
LP 'C' Last Stg
LP 'C' Mois Rmv
LP 'C' 12th Stg
LP 'C' 11th Stg
LP 'C' 9th Stg
LP 'C' 8th Stg
LP 'B'
LP 'A'
CIV #1 703
CIV #6 704
CIV #2 705
CIV #5 706
CIV #3 707
CIV #4 708
To Htr 1C
To Htr 2C
To Htr 3C
To Htr 4C
To Htr 1B
To Htr 2B
To Htr 3B
To Htr 4B
To Htr 1A
To Htr 2A
To Htr 3A
To Htr 4A
Drain LP Htr Trn 'C'
LP Htr Trn 'B'
LP Htr Trn 'A' Drain
Aux Stm
Gland Steam
FWPT Exh
BD Demin

Figure 6

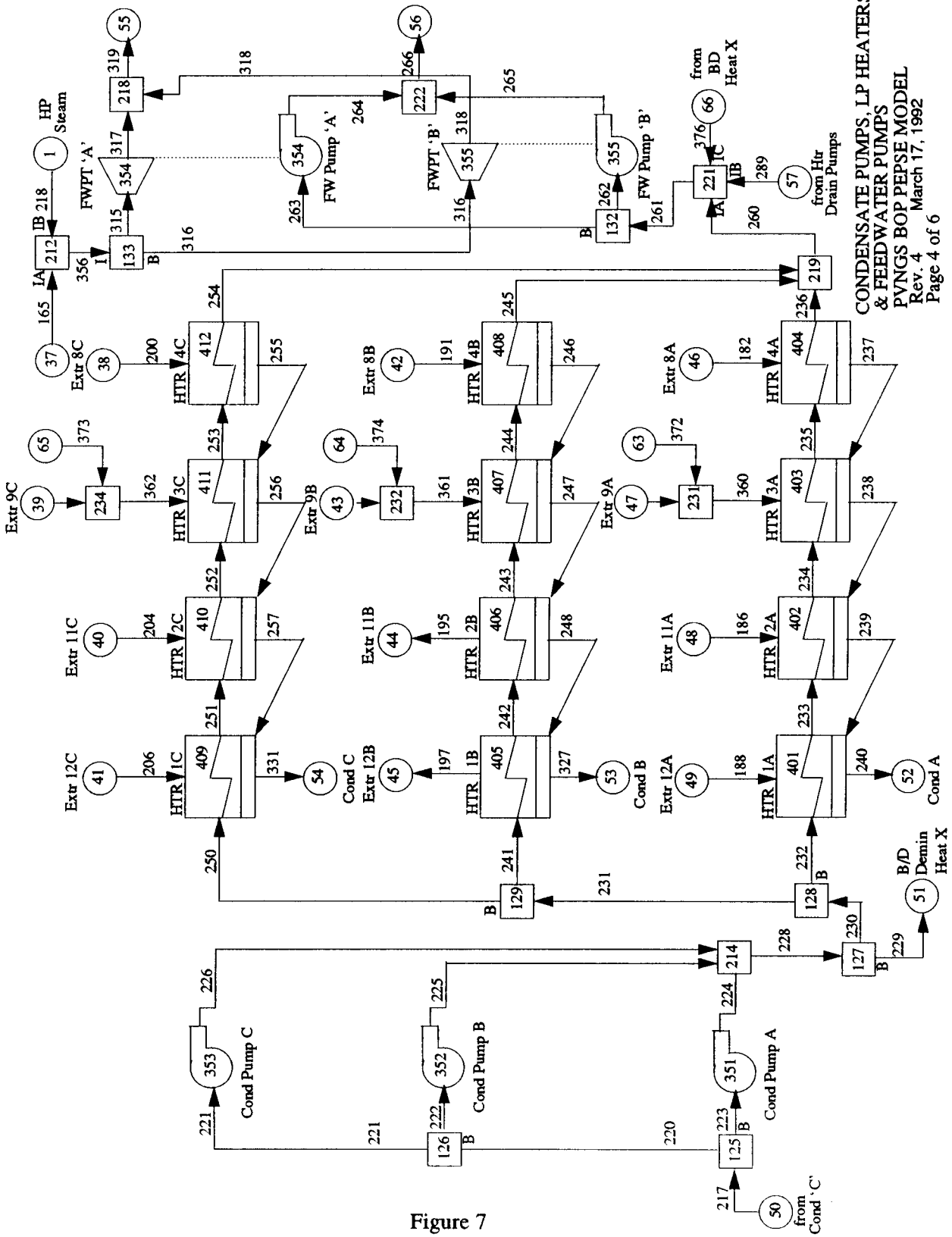

CONDENSATE PUMPS, LP HEATERS
& FEEDWATER PUMPS
PVNGS BOP PEPSE MODEL
Rev. 4
March 17, 1992
Page 4 of 6
HP Steam
FWPT 'A'
FWPT 'B'
FW Pump 'A'
FW Pump 'B'
from BD Heat X
from Htr Drain Pumps
Extr 9C
Extr 8C
Extr 8B
Extr 8A
Extr 11C
Extr 11B
Extr 11A
Extr 12C
Extr 12B
Extr 12A
Extr 9B
Extr 9A
Cond C
Cond B
Cond A
B/D Demin Heat X
Cond Pump C
Cond Pump B
Cond Pump A
from Cond 'C'

Figure 7

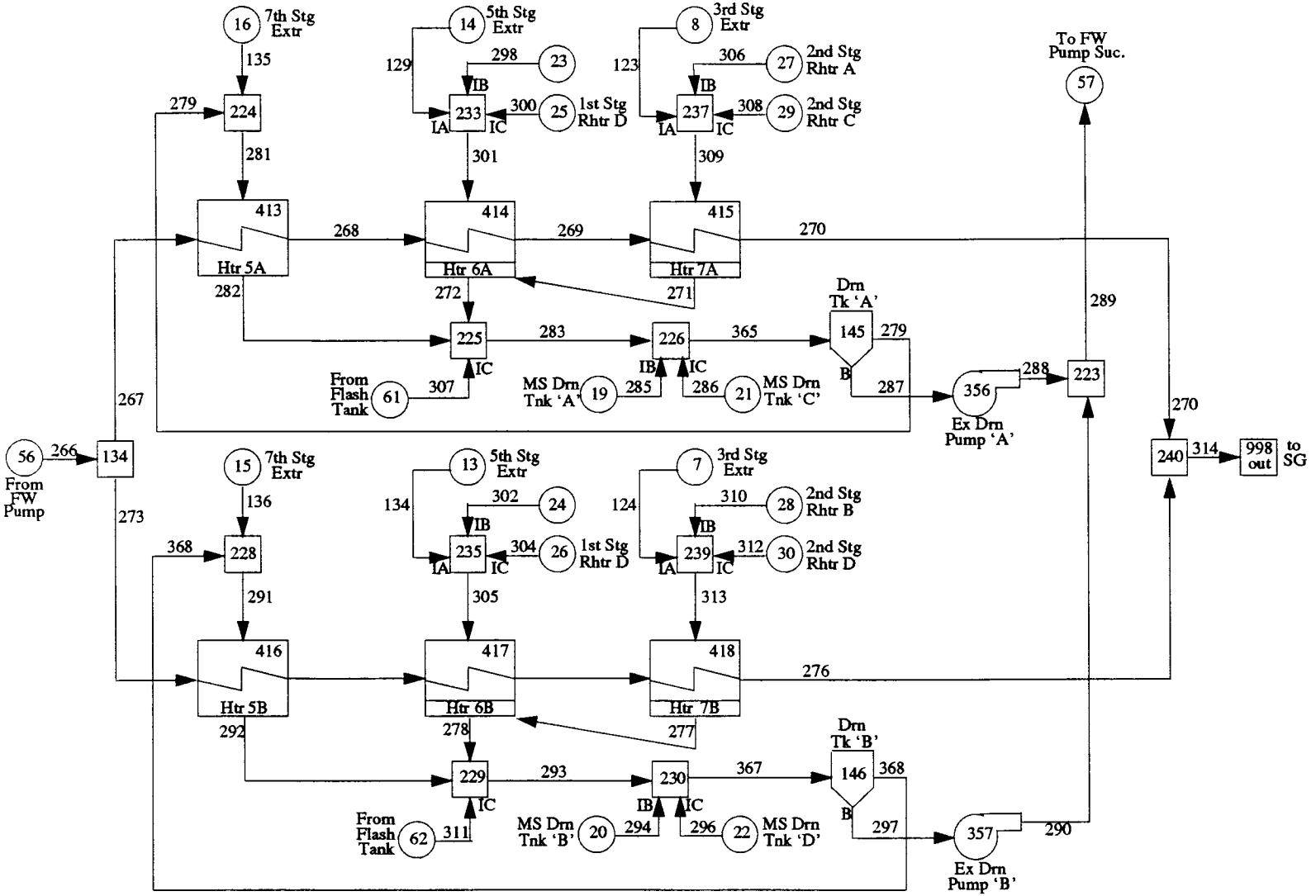


Figure 8

HP HEATERS & EXTRACTION DRAIN
PUMPS & TANKS
PVNGS BOP PEPSE MODEL
Rev. #4 March 17, 1992
Page 5 of 6

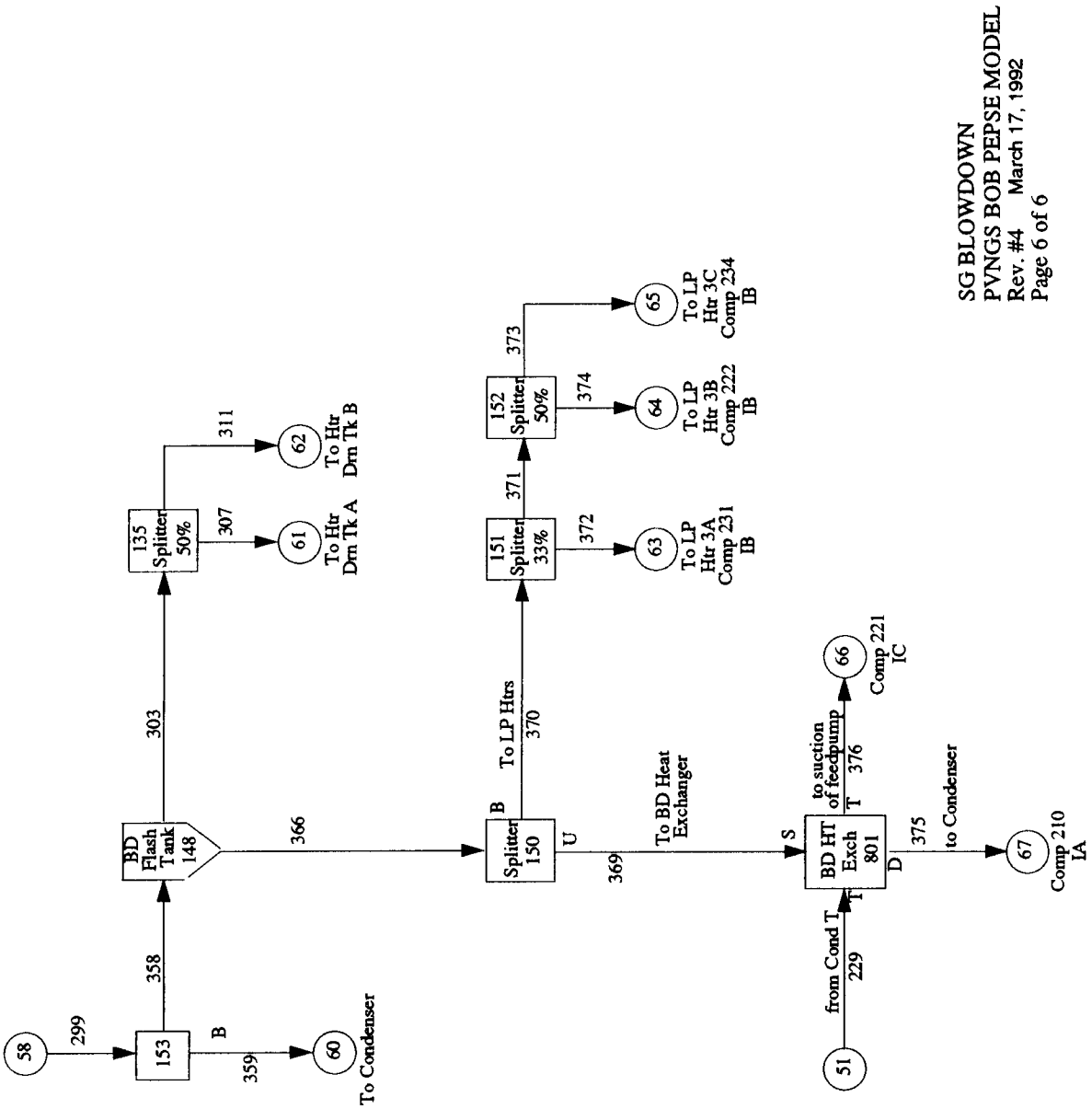

SG BLOWDOWN
PVNGS BOB PEPSE MODEL
Rev. #4 March 17, 1992
Page 6 of 6
58
299
153
B
359
60
To Condenser
358
BD Flash Tank 148
303
135 Splitter 50%
307
61
To Htr Drn Tk A
311
62
To Htr Drn Tk B
366
Splitter 150
B
U
To LP Htrs
370
151 Splitter 33%
372
63
To LP Htr 3A Comp 231 IB
371
152 Splitter 50%
374
64
To LP Htr 3B Comp 222 IB
373
65
To LP Htr 3C Comp 234 IB
369
To BD Heat Exchanger
S
BD HT Exch 801
T
D
to suction of feedpump
376
66
Comp 221 IC
375
to Condenser
67
Comp 210 IA
51
from Cond T
229
T

Figure 9

exchanger, was added to the Palo Verde model in 1991.

This model running in performance mode would take 22.9 seconds per iteration for a total run time of 9 minutes and 25 seconds on a 386 type of computer running with a clock speed of 16 megahertz. The same model runs five times faster on the SPARC station 1+ at 4.3 seconds per iteration for a total run time of one minute and 48 seconds. This doesn't include the efficiencies of having the data available on the LAN for anyone who wants to see the results. It also assumes that during the time that PEPSE is running on the SPARC station no other functions are being performed.

If two pepse models run at the same time on the same workstation the run time doubles. Multiple PEPSE runs can be accomplished by running one model on one workstation, another on a second workstation and a third on another workstation. Windows can be opened to the other workstations so the operator can run all three from one workstation.

Running models with this kind of speed tends to decrease the desirability of the interrupt features to insert new values. When a model runs in just over one and one-half minutes with four seconds per iteration the need to break in the programs to fix iteration problems is diminished.

Table 1 is a comparison of the run times between the different computers running the same input deck.

Advantages / Disadvantages

Advantages

1. Integrated System - With PEPSE on the LAN, data for PEPSE decks can be directly transferred into a deck. Previously, PEPSE decks were put together on the LAN, converted to a DOS ASCII format, transferred to a diskette, and transferred onto a personal computer. The PEPSE output file would then be read to a diskette and manually transferred to the engineering LAN for reference and storage.
2. Performance gain - A significant reduction in the amount of time it takes to run a PEPSE deck resulted from porting PEPSE to Unix. The section entitled Performance Gains discusses these gains in detail.
3. Versatility gain- PEPSE can be used in conjunction with other application programs on the Local Area Network.

Disadvantages

1. The Man-Machine Interface (MMI) was not ported to Unix. The performance engineers at Palo Verde have been using PEPSE since 1985 and learned to use

Table 1: PEPSE RUN TIME COMPARISON

Iteration #	386 -16 MHz Computer Elapsed Time	Sun SPARC +1 1 job executing Elapsed Time	Sun SPARC +1 2 jobs executing Elapsed Time	Sun 490 Server 1 job executing Elapsed Time
Read	48 sec	10 sec	13 sec	6 sec
1	1 min 15 sec	17 sec	23 sec	8 sec
2	1 min 39 sec	21 sec	32 sec	11 sec
3	2 min 0 sec	25 sec	40 sec	13 sec
4	2 min 22 sec	29 sec	48 sec	15 sec
5	2 min 45 sec	34 sec	57 sec	17 sec
6	3 min 7 sec	39 sec	1 min 5 sec	19 sec
7	3 min 30 sec	42 sec	1 min 14 sec	21 sec
8	3 min 53 sec	46 sec	1 min 22 sec	23 sec
9	4 min 15 sec	50 sec	1 min 30 sec	25 sec
10	4 min 38sec	55 sec	1 min 39 sec	28 sec
11	5 min 1 sec	59 sec	1 min 48 sec	30 sec
12	5 min 24 sec	1 min 3 sec	1 min 56 sec	32 sec
13	5 min 47 sec	1 min 7 sec	2 min 5 sec	34 sec
14	6 min 9 sec	1 min 11 sec	2 min 14 sec	36 sec
15	6 min 32 sec	1min 16 sec	2 min 22 sec	38 sec
16	6 min 55 sec	1 min 20 sec	2 min 31 sec	40 sec
17	7 min 18 sec	1 min 24 sec	2 min 39 sec	42 sec
18	7 min 41 sec	1 min 28 sec	2 min 48 sec	44 sec
19	8 min 3 sec	1 min 33 sec	2 min 56 sec	46 sec
20	8 min 26 sec	1 min 37 sec	3 min 5 sec	48 sec
21	8 min 49 sec	1 min 41 sec	3 min 13 sec	50 sec
22	9 min 12 sec	1 min 45 sec	3 min 17 sec	52 sec
Write	9 min 25 sec	1 min 48 sec	3 min 20 sec	53 sec

PEPSE before MMI was created. Consequently, they do not use MMI. Because of this fact and the additional amount of time and money required to port MMI, a decision was made not to port it to Unix. At Palo Verde this is not viewed as a disadvantage. However, if a person has learned to use the MMI and works with it regularly then the lack of MMI is a disadvantage.

2. The PEPSE Interactive Processor (PIP) was not adapted to the Unix system. PIP has the ability to interrupt a PEPSE run and write the current results to an output file or insert new values into PEPSE variables. The conversion of these features to Unix requires that new subroutines be written in C to handle the interruption and data input from the keyboard. This was beyond our technical capability at the time and was not done. However, this disadvantage is minimal due to the increased processing speed. The faster a model runs and converges the less need and desire there is to interrupt it. If there is a need to use the PIP features, they can be worked around by doing some pre-job planning. For example, if a person wants to see the output at a particular iterate then the maximum number of iterations can be changed to that iterate value before the job is run so it will stop at the desired iteration.
3. The porting of PEPSE can be expensive if not controlled. One of the purposes of this paper is to present lessons learned on porting PEPSE from a DOS platform to a Unix platform in order to help other utilities reduce the cost of the conversion.

Description of Changes to the Source Code

Summary of Changes

Four types of changes were made to the PEPSE source code. The first type were those that converted the code from a DOS format to a Unix format. The second type of changes were those where a Fortran function in the PEPSE source code had to be replaced with a similar function from Sun Fortran. The third type of changes were those that wrote or read data directly into or from a memory location. And the fourth type of changes were those that modified the way PEPSE wrote information to the computer screen.

1. DOS to Unix Conversion - The carriage return character in DOS is read as a control M in Unix. Each PEPSE source code file had to have all of its carriage return characters converted to the Unix format by running a program called DOS2UNIX.

   Path names are written slightly differently in Unix. MS-DOS uses a backward slash between sub-directory names. Unix uses a forward slash for the same purpose. Consequently, all PEPSE source code files that contained path names

had to have all the back slashes (\) converted to forward slashes (/). Also, the path names had to be changed to match the directory structure of the PVNGS Sun LAN.

The default read and write file numbers #5 and #6 used with the READ and WRITE commands were already in use by the Sun operating system. These default file numbers were changed from #5 to #55 and #6 to #66 in all source files that used READ and WRITE commands. In the Sun operating system file #5 is normally reserved for the standard input (keyboard) and file #6 for the standard output (screen).

2. Fortran Function Changes - Several functions used in the PEPSE source code did not exist in the Sun version of Fortran and had to be replaced with similar functions.
3. Memory Address Locator Routines - Unix handles its memory addresses differently than DOS due to its multitasking capabilities. Consequently, several sub-routines in PEPSE that read from or wrote directly to a memory address had to be changed.
4. Screen Writing Routines - The screen writing routines used in DOS had to be modified to work with the graphic user interface used on the SPARC workstations. Also, the routines that checked the keyboard buffer to see if the user wanted to interrupt the program and use the PEPSE Interactive Processor (PIP) had to be rewritten in C in order to be compatible with the workstations. However, this task was not fully completed and the current version of PEPSE running on the PVNGS Sun LAN does not have the capability to use the PIP feature.

Code Organization and Size

Version 56 of the PEPSE code consists of one Fortran program, 391 Fortran sub-routines and 91 Fortran "INCLUDE" routines. Refer to the PEPSE Manual, Volume II, under the Code Organization tab for more details on the organization of PEPSE. In addition to the Fortran files, five C object files are used for detecting keyboard input and screen writing. Locating and making changes in all these files as well as compiling the files was a complicated task. Unix has some very powerful commands that can be used to simplified this task.

1. grep - allows you to search one or more files for particular character patterns.
2. ls - lists the files in the present working directory (pwd)
3. more - prints a particular file to the screen
4. wc - counts the number of words or lines in a file

5. vi - a basic line editor used in Unix which is useful for making simple changes
6. sed - a stream editor that is good for search and replace routines
7. make - recompiles only files that have been changed and need recompiling

Changes to the PEPSE Source Code

PEPPC.FOR changes

The file **PEPPC.FOR** is the actual program file. The rest of the source files are either subroutines or INCLUDE routines. Extensive changes were made to this file and it was renamed **sunpep.f**. The changes made to PEPPC.FOR are representative of the kinds of changes required by the rest of the source code.

1. The default file numbers had to be changed to accommodate the fact that the normal Fortran default files, #5 and #6 were already used by the Sun Unix operating system. The default files were changed to #55 and #66. The following is an example of these changes. The comment lines are the original source code.

```
IF (.NOT. RESTRT)

C   *   OPEN(5, FILE=IN_TEMP, STATUS = 'OLD',
C   *   CARRIAGECONTROL = 'FORTRAN', ERR = 1400)
    IF (.NOT. RESTRT)
   *
OPEN(55,FILE=IN_TEMP,FORM='PRINTER',STATUS='OLD',ERR=1400)
```

2. The screen control methods were changed to use C program application of the curses routines verses the Fortran screen control methods. Five programs were written to accomplish this task.
   - clrwin -- clear window
   - initwn -- initialize curses window (set up to read input without a "CR"
   - readch -- read a character
   - wrstr -- write a single character or string
   - winend -- end curses window

The following is a program listing of the C routines.

clrwin.c:

```
#include <curses.h>
clrwin()
{
clear();
}
```

initwn.c

```
#include <curses.h>
initwn()
{
initscr();
cbreak();
noecho();
nonl();
}
```

winend.c

```
#include <curses.h>
winend()
{
endwin();
}
```

wrchar.c

```
#include <curses.h>
wrchar(line, col, ch, length)
int  *line;
int  *col;
char *ch;
int  *length;*
{

mvaddch(*line, *col, ch);
move(LINES-1, 0);
refresh();
}
```

wrstr.c

```
#include <curses.h>
wrstr(line, col, str, length)
int  *line;
int  *col;
char *str;
int  *length;
{
if (*length == 1)
 {
 mvaddch(*line, *col, str[0]);
 }
else
```

```
 {
 mvaddstr(*line, *col, str);
 }
move(LINES-1, 0);
refresh();
}
```

In general, the following changes must be made to the source code to allow the use of these routines:

```
CHARACTER STRING*8 - Variable for strings to write to curses window
CHARACTER  readch*1 - Used to read a single character
CHARACTER NCHAR*1 - Used in with readch to read a single char

EXTERNAL initwn !$pragma C( initwn ) - Defines the C routines
EXTERNAL winend !$pragma C( winend )
EXTERNAL clrwin !$pragma C( clrwin )
EXTERNAL wrstr  !$pragma C( wrstr  )
```

The file **readch** is missing from the EXTERNAL list. It is defined by the two variables above. This is because of the convoluted way you pass a string/character back to a Fortran program. The following is some sample code.

```
CALL initwn() - Initialize the curses window

CALL clrwin() -  Clear the window

LENGTH = 0

CALL wrstr( LINE, COL, INPUT, LENGTH) - Writes a string to the curses
window where LINE is the line number and COL is the column and LENGTH
is set to 0. LENGTH is set to 1 if you wish to write a single character.
```

NCHAR = readch('*',1) read a single character from the curses window, no carriage return required; readch returns a -1 if no character present.

```
LENGTH = 1
CALL wrstr(LINE, COL, NCHAR, LENGTH)  -  Writes a single character
CALL winend() - Closes curses window
```

The following are the changes made to incorporate the C routines in the Fortran. The comment line is the original code.

```
C   INTEGER*2 RPMODE, RTSTAT, ICODE, NEXEQ, EXELIM
    EXTERNAL initwn !$pragma C( initwn )
    EXTERNAL clrwin !$pragma C( clrwin )
    EXTERNAL winend !$pragma C( winend )
```

3. The Unix operating system uses the forward slash (/) rather than the back slash (\) to define a path. Therefore, all references to files and their paths had to be put in the proper format. Unix is also case sensitive and this must be taken into account when referencing files or making calls. The following two statements are interpreted differently by Unix.

```
INCLUDE '/home/perfdata/perfprgm/pepse/inc/parms.for'
INCLUDE '/home/perfdata/perfprgm/pepse/inc/PARMS.FOR'
```

Due to the case sensitivity of Unix, the file parms.for is considered to be a different file from PARMS.FOR.

4. The function **GETCL** in the source code did not exist in Sun Fortran. In order for characters from the PEPSE command line to be used in **sunpep.f**, the function calls for passing the command line input were altered as follows. The comment line is the original source code and the next line is the change made to the source code.

```
C    CALL GETCL(CMD_LINE)
     CALL CMDL(L,CMD_LINE)
```

5. The file **nblank.f** was to be written to replace the function **NBLANK** function used in **PEPPC.FOR**. The function **NBLANK** is an intrinsic function of the Fortran used by PEPSE and it is not available in Sun Fortran.The following is a listing of **nblank.f**.

```
      SUBROUTINE NBLANK(STR,I)
C     The function returns the length of the string.
C
C       STR = String
C       I   = Length of STR
C
      CHARACTER*128 STR
C
      DO 100 J=1,128,1
        K = J + 1
        IF(STR(J:K).EQ.'  ')GOTO 200
  100 CONTINUE
      I = 128
      GOTO 300
  200 CONTINUE
      I = J - 1
  300 CONTINUE
      RETURN
      END
```

6. All references to CHAR(0) had to be replaced with ' '. The following are some examples of these changes where the comment line is the original source code.

```
C       IF (IN_TEMP(L:L) .NE. CHAR(0)) THEN
        IF (IN_TEMP(L:L) .NE. ' ') THEN
```

```
C       OUPATH  = OUPATH(1:K)//IN_NAME(1:L)//'.OUT'//CHAR(0)
        OUPATH  = OUPATH(1:K)//IN_NAME(1:L)//'.out'
```

```
C         IF (IN_TEMP(I:I) .NE. '.'  .AND.
C  *          IN_TEMP(I:I) .NE. CHAR(0)) THEN
          IF (IN_TEMP(I:I) .NE. '.'  .AND.
     *        IN_TEMP(I:I) .NE. ' ') THEN
```

7. All calls to the MS-DOS operating system command "del" were changed to the Unix equivalent "rm". The following is an example of this change where the comment line is the original code.

```
C       IF (EXST) CALL SYSTEM('DEL '//OUPATH(:IEXT)//'BCK')
        IF (EXST) CALL SYSTEM('rm '//OUPATH(:IEXT)//'bck')
```

8. All calls to MS-DOS operating system command "RENAME" were changed to the Unix equivalent "mv". The following is an example of this change where the comment line is the original source code.

```
C       CALL SYSTEM('RENAME '//OUPATH(:OUTLEN)//' '//
C  *              OUPATH(II:IEXT)//'BCK')
        CALL SYSTEM('mv '//OUPATH(:OUTLEN)//' '//
     *              OUPATH(II:IEXT)//'bck')
```

9. Changes to the printer control were made to remove the carriage control and add FORM + PRINTER to change |h| top of form character to CNTR L:

```
C  *   OPEN(5, FILE=IN_TEMP, STATUS = 'OLD',
C  *   CARRIAGECONTROL = 'FORTRAN', ERR = 1400)
     IF (.NOT. RESTRT)
     *
OPEN(55,FILE=IN_TEMP,FORM='PRINTER',STATUS='OLD',ERR=1400)
```

10. The function **TIMER** does not exist in Sun Fortran and was changed to a similar Sun Fortran routine called **ITIME**. The comment line is the original source code.

```
C     CALL TIMER(ITICKS)
      CALL ITIME(JTM(1),JTM(2),JTM(3))
      ITICKS=(JTM(1)*3600)+(JTM(2)*60)+JTM(3)
```

Changes to PEPSE.FOR

1. The changes made to the **PEPSE.FOR** file consisted of changing the system commands from DOS to Unix, changing the INCLUDE references to the Unix format and some dummy variables to prevent compiler error messages.

Changes to SCRUPD.FOR

Subroutine **SCRUPD.FOR** is responsible for writing to the screen and therefore had to have extensive changes.

1. The INCLUDE statements were changed to reflect the UNIX directory structure.
2. EXTERNAL statements were added to allow use of the clrwin and wrstr screen programs. The following is an example of these changes:

```
C       PVNGS CHANGE AREA
        EXTERNAL clrwin !$pragma C( clrwin )
        EXTERNAL wrstr  !$pragma C( wrstr  )
```

3. The call for the date and time are different for Unix. The comment lines are the original source code.

```
C     CBLANK(1:) = ' '
C     CALL DATE (CDATE)
C     CALL TIME (CTIME)
      CALL FDATE (XDATE)
      CDATE = XDATE(5:10)//XDATE(20:24)
      CTIME = XDATE(12:19)
C-----------------------------------------------
```

4. Subroutine **break.f** is used to break into the program with keyboard input. This feature has not been implemented on the PVNGS Sun LAN. Some of the programming has been accomplished, however more work is required. This will be accomplished with calls to C routines.
5. Subroutine **cbrk_chk.f** is used in conjunction with **break.f** and is not implemented on the PVNGS Sun LAN.
6. Subroutine **kebord.f** is used to read the keyboard buffer and determine if an interrupt has been issued. As previously stated, this function is not implemented on the PVNGS Sun LAN. Some of the programming has been accomplished.

7. Subroutine **lrs.** is the same subroutine as **RS.FOR**. The name was changed for consistency with an integer function name (starting with letters i - n).

Memory Addressing

Memory addressing was the most perplexing problem in porting the PEPSE code to the Unix platform. The problem centered around the way memory is addressed and array elements are handled. One of the basic differences between the Unix operating system and the Disk Operating System is the manner in which memory is handled by the machine. On a machine which uses DOS the memory locations are fixed. When a program is running it sequesters a particular area of memory for itself and no other process can use it. Therefore, when programmers decide to use a particular address they can be assured that it will always be there.This happens because DOS will only allow one process to run at a time.

In the Unix operating system the memory is relative. Unix constantly swaps memory around keeping various processes going at the same time. Therefore, what may be a memory address in one iteration may not be the same address in the next iteration. On a PC the array elements were defined and stored in absolute memory locations using an assembly language routine and then retrieved from those locations. In Unix, the programming had to take into account the relative nature of the memory and come up with a reference point from which to relate the array element. Since REAL * 8 variables are used in order to find a particular element of the array, the memory offset was divided by eight and that number was used to find the element of the array and retrieve the data.

The following subroutines were written by NUS and modified at Palo Verde to perform memory addressing in Unix environment.

getref.f

```
      DOUBLE PRECISION FUNCTION GETREF(ILOC)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON / AAAAA / AAADUM
       DIMENSION REF(1)
      EQUIVALENCE (REF(1), AAADUM)
      GETREF=REF(ILOC)
      RETURN
      END
```

storef.f

```
      SUBROUTINE STOREF(ILOC,VALUE)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON / AAAAA / AAADUM
      DIMENSION REF(1)
```

```
      EQUIVALENCE (REF(1), AAADUM)
      RETURN
      END
```

locf.f

```
      INTEGER*4 FUNCTION LOCF (XDUM)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON / AAAAA / AAADUM
      DIMENSION REF(1)
      EQUIVALENCE (REF(1), AAADUM)
      LOCF=((LOC(XDUM)-LOC(REF(1)))/8)+1
      RETURN
      END
```

aaaaa.for

```
      COMMON / AAAAA / AAADUM
      DIMENSION REF(1)
      EQUIVALENCE (REF(1), AAADUM)
C
C     END OF MODULE
```

Compiling

After all the changes are made to the source code the last step in the porting process is compiling. The basic compilation process consists of compiling each of the different files into an object file and then linking the object files together into one executable file. Compiling and linking the modified source code was a time consuming chore due to the large number of files involved. We compiled all the files at least a two dozen times before we had all the bugs worked out and were satisfied with the results. The type of Fortran used will determine the exact method used to compile, but Sun Fortran had a helpful command called MAKE.

The MAKE command recompiles only those files that have been changed so the recompile process is much faster. Compilation times were on the order of 2 to 3 hours to compile all the files. But, with the MAKE command a new executable file could be made in a matter of minutes depending on the number of files modified. In order to use MAKE, a "makefile" containing various compiling options and the names of all the object files in the library had to be developed. This file can then be used each time any file or files are modified and need to be recompiled.

Conclusion

Palo Verde Nuclear Generating Station Thermal Performance has seen an increase in productivity and the ability to identify and analyze plant thermal performance problems as a result of implementing the Thermal Performance Monitoring Program on the PVNGS Sun Local Area Network. The use of PEPSE has become much faster and more accessible. No abnormalities in the execution of the model have been detected as a result of porting the PEPSE code to the Sun Unix platform.

Other utilities with Local Area Networks should consider the use of these networks to assist the thermal performance/results engineers in better performing their tasks. If these networks happen to be in an Unix environment the utilities could benefit in porting PEPSE to the Unix environment.