



•

1.1

:

lan -Sairt

MICROCOPY RESOLUTION TEST CHART NATIONAL BUREAU OF STANDARDS-1963-A



CPESIM II: A COMPUTER SYSTEM IMULATION

FOR COMPUTER PERFORMANCE EVALUATION USE

THESIS

David L. Owen Captain, USAF

AFIT = CS/EE/83D-16

DEPARTMENT OF THE AIR FORCE AIR UNIVERSITY AIR FORCE INSTITUTE OF TECHNOLOGY

# Wright-Patterson Air Force Base, Ohio

DISTRICTION STATEMENT A

84 02 22 673

-ECTE

FEB 2 2 1984

D

DTIC FILE COPY

\_\_\_\_

6

1----

Accession For NTIS GRA&I  $\mathbf{X}$ DTIC TAB Unannounced Justification\_ By\_ Distribution/ Availability Codes Avail and/or Special Dist ) 



CPESIM II: A COMPUTER SYSTEM SIMULATION FOR COMPUTER PERFORMANCE EVALUATION USE

THESIS

David L. Owen Captain, USAF

AFIT/ JCS/EE/83D-16



spproved for public release; distribution unlimited

AFIT/GCS/EE/83D-16

CPESIM II: A COMPUTER SYSTEM SIMULATION FOR COMPUTER PERFORMANCE EVALUATION USE

#### THESIS

Presented to the Faculty of the School of Engineering of the Air Force nstitute of Technology Air University In Partial Fulfillment of the Requirements for the Degree of Master of Science

> David L. Owen, B.S. Captain, USA

> > December 1983

Approved for public release; distribution unlimited

## Preface

Professor Thomas Hartrum provided the original stimulus for this thesis. While teaching computer performance evaluation courses, he discovered a need for a vehicle to illustrate the problems and techniques he was discussing in class. In 1979, Capt Paul Lewis wrote a simulation to help solve some of these problems. As time past, limitations in the original simulation needed to be solve.

This thesis is the first step in converting the standalone simulation package into a total computer performanc evaluation simulation environment. This environment should include as a minimum a simulated computer, "user" interfaces, data base record keeping, and workload generators. My attempt was to redesign and code the simulated computer system, and create a "user" interface to the system From this basic environment follow on efforts can then enhance the environment in a modular fashion.

I would like to thank Professor Hartrum for lis guidance and valuable suggestions that he made through out the development of this thesis. Lt Col Clark proved to be not only a excellent committee member, but a source of knowledge on many obscure aspects of SLAM. Finally, I wish to thank my future wife Sue for mer understanding and support on those many nights when I had to work.

David L. Owen

# Contents

, ,

Ġ

																																		Page
Prefa	ce	•		•	•		•	•	,	•		•		•	•		•		•	•		•	•		•	•		•	•	•	•	•	•	ii
List	of	F	iį	gı	ır	e	s	•	,	•		•		•	•		•		•	•		•	•		•	•		•	•	•	•	•	•	v
Abstr	act	t		•	•		•	•	•	•				,	•		•		•	•		•			•	•		•	٠	•	•	•	•	vi
I.	Ľ٦	١t	r	od	lu	c	ti	ĹĊ	n			•	,	•	•		•		•	•		•	•		•	•		•	•	•	•	•	•	1
		P	ĩ	c k	٢g	r	οι	1 r	1 d			•		•		,	•		•	•		•			•	•		•	•	•			•	1
		F	Ξ	) i	<b>) 1</b>	e	m	I	)e	S	С	r	i p	) t	: 1	0	n		•	•		•			•	•		•	•	•	•		•	3
		À.	3	- 1	0	a	cł	n				•		,	•		•		•	•		•			•	•		•	•	•	•	•	•	4
		S			) e		•	•		•		•		•	•		•		•	•		•	•		•	•		•	•	•		•	•	5
		0	r	de	r		oſ	E	P	r	e	s	e r	1 t	a	t	i	01	n	•		•	•		•	•		•	•	٠	•	•	•	6
II.	Sy	7 S	t	ea	1	R	eç	Įu	ıi	r	e		e r	1t	: s	;	,		•	•		•	•		•	•		•	•	•	•	•	•	7
		Е	n	v 1	r	0	no	1 e	e n	t		•			•		•		•			J				•		•	•		•			7
		S	0	Et	w	a	re	2	R	e	a	u:	i 1	: e	2 00	ie	n	t	s						•			•					•	8
		S	u	DI	0	r	t	R	e	a	ú		r e	22	ıe	n	t	s										•			•			14
		Ċ	01	nf	1	1	c t	: 1	n	2	-	R	2 0	ιu	11	r	e		e r	۱t	s							•		•		•	•	15
		v	e	r i	f	1	ca	a t	: <b>í</b>	õ	n		a r	ıd	1	V	a	1:	ĺċ	ła	t	10		ı						•	,			15
		S	uı	<b>n</b> 1	a	r	y	•		•		•	•		•		•		•	•	-	•	•		•	•		•	•	•	•	•	•	16
111.	Si	L ma	u.	l a	ıt	i	01	נ	D	e	S	iį	3 T	1	•		•		•	•		•	•		•	•		•	•	•	•	•	•	17
		I	0 1	p1	. e	m	er	ı t	: a	t	i	01	n	L	a	n	8	u	ag	٤e	8					•			•	•	•			17
		С	01	D C	u	t	eı		A	r	c١	h:	Ĺt	: e	2 C	t	u	r	e`	΄.		•				•		•	•				•	18
		0	p	e i	a	t	iı	12	č	S	y :	s	tε	) U	1				•				•		•			•	•	•				24
		S	o 1	Et	:w	a	re	5 . 5	M	0	n	i	t c	) I	Ċ				•			•			•	•		•	•	•	•	•	•	28
		H	a	<b>r</b> d	W	a	re	2		0	n	i	t c	<b>,</b> .					•									•	•	•	•		•	29
		W	0	r k	:1	0	ac	ł	و	e	n	e	r a	ı t	: 1	. 0	n		•	•		•			•	•			•	•	•	•	•	30
		U	5 (	e r		I	n t	: e	r	f	a	c	e		•		•		•	•		•	•		•	•		•	٠	•	٠	•	•	30
IV.	Ve	2 C	1:	E 1	c	a	tj	L c	n		a	n	1	V	la	1	1	da	a t	:1	0	n	•		•	•		•	•	•	•	•	•	33
		v	e	r i	f	1	сŧ	a t	:1	0	n			,					•						•				•				•	33
			1	1a	1	n	I	? r	0	g	r	a	۵	F	1	0	W		•			•	•		•				•	•	4	•	•	33
			1	Di	. <b>s</b>	k	1	[/	0		a	n	ł	C	; h	a	n	n	e 1		I	m t	51	e	m e	e n	t	a t	:ic	n	•		•	34
				Гa	5	e	1	1	0		Iı	ן בכ	<b>5</b> 1	. e	2 00	ie	n	ta	a t	: <b>i</b>	0	ກ່			•	•			•	•	•		•	34
			5	5 o	ŕ	t	wa	ı r	:e		M	01	t r	t	: 0	r		Ιt	nŗ	1	e	шε	e n	t	a t	: <b>i</b>	0	n	•	•	•	•	•	34
			1	ła	r	đ	we	ı z	e:		M	0 1	ı i	t	: 0	r		Ir	n ŗ	51	e	шε	e n	t	a١	: i	01	n	•				•	34
		v	a 1	14	A	a	* 4	0																									_	34

iii

لتمتعمد

# Contents

•

Ö

							Page
V. Summar	y and Rec	commenda	ations		•••	•••	36
Summ Rec	ary nmendatio	ons	••••	• • • •	•••	•••	36 3 <sup>-</sup>
Bibliography		•••	• • •	• • • •	•••	•••	40
Appendix A:	CPESIM I	II ADT	Diagram	15	•••	•••	A-1
Appendix B:	CPESIM I	II RL'M	Network	k Drawin	gs.	• •	B-1
Appendix C:	CPESIM I	I Sturd	ctured I	English	•••	• •	C-1
Appendix D:	CPESIM I	II Code	• • •	• • • •	•••	••	<b>D</b> – 1
Appendix E:	CPESIM I	II Insti	uctor M	Manual	•••	• •	E – 1
oppendix F:	CPESIM I	II Stude	en' Manu	ual	•••	•••	F-1
Vita		• • • •			• • •	• •	V-1

ív

~

# List of Figures

Fig	gure			Page
1.	Performance Improvement	•	••	2
2.	Case Study Flow	•	•••	9
3.	Typical CPESIM II Configuration	•	••	12
4.	Job Flow	•	••	13
5.	CPESIM II Configuration Limits	•	•••	23
6.	CPTFIE IL Job Flow	•	••	2 5
7.	User Int face Structure	•	• •	31

Ċ)

#### Abscract

This report describes the development of an education tool which is used in conjunction with a c urse in computer performance evaluation. The tool is a system composed of background information and software which simulates a computer system. The student is assigned a position in a fictional computer site, and is given performance problems to solve. The instructor prepares scenarios, constructs a model of the site's worload, and describes the initial computer configuration. The student takes this information and uses it to decide if additional data is required from the hardware and software monitors.

The student forms an improvement hypothesis and makes changes to the simulated computer site. Changes to the simulated computer configuration are made by an interactive "user" interface. The configuration changes, along with the workload description and monitor requests are submitted to the simulation. The simulation uses these inputs to simulate a large mainframe computer with attached software and hardware monitors. Accounting and monitor logs are generated by the simulation. After examining the outputs, the student provides further changes until he is satisfied with the simulated computer sit 's performance.

vi

This document provides a copy of the current software used to imitate the computer, the software used to generate the "user" interface. and instructions to the instructor an! the student on executing the software on the ASD computer systems which support AFIT.

CPESIM II: A Computer System Simulation for Computer Performance Evaluation (se

#### I. Introduction

#### Background

57

Every computer system eventually comes to the point where more computing power is needed. In the past, the general solution has often been to buy more hardware without analyzing the performance of the present computer system. Recently, more and more effort has been placed on analyzing the computer environment as a whole to determine if additional resources are needed or if the computer can be "tuned" to perform more efficiently. In a study for the Air Force (Ref 6), Bell, Boehm and Watson suggested seven p ases fir a Computer Performance Evaluation (CPE) team to follow to successfully evaluate the computer system and its environment (Figure 1). In order for a student to effectively learn the CPE tools, he should gain experience in using these seven phases. The first four phases, understanding the system, analyzing operations, formulating performance improvement modifications, and analyzing probable cost-effectiveness of modifications, can be accomplished with information proviled by the instructor. The instructor can either use an actual scenario or can make



Fig. 1 Performance Improvement (Ref 6)

up a scenario to illustrate a particular performance problem. In order for the student to accomplish the last three phases, testing specific hypotheses, implementing appropriate combinations of modifications, and testing the effectiveness of the implemented modifications, the student must have access to CPI data from an actual or simulated computer system. CPE data comes from three general sources: accounting data, software monitor data, and hardware monitor data Many computer installations have only some of the data necessary, and even that may not be adequate. In addition, many installations cannot afford to have st ents experiment with their oper tional system. In order for the students to gain "hands on" experience in the last three phases, a computer simulation is needed. In 1979, Capt Paul Lewis developed a model (called CPESIM II) to aid students in performing CPE (Ref 7).

#### Problem Description

While Capt Lewis' model was a maj of step in aiding the CPE student with "hands on" experience, it had several shortcomings. The first major shortcoming is similar to many real life computer environments. The software monitor for CPESIM II did not provide the necessary information for the student to successfully analyze the problem. The second problem with CPESIM II was the language in which it was written. CPESIM II was written in SIMSCAIPT II (Ref 3),

which at the time was one of the better simulation languages. Since 1979, the expertise for SIMSCRIPT has not been available at AFIT and it is no longer leing taught. The final problem with CPESIM II was with the user interface. The amount of work required to run the simulation was burdensome to both the instructor and the stillent. Thus, an easy means of obtaining all the mecessary CPE data was still needed.

#### Approach

One possible approach to solv ng this problem was to modify the existing CPESIM II and to create a "friendly" user interface. There are several problems with this approach. First, since SIMSCRIPT is no longer taught at AFIT there is a question about how much longer it will be supported. Secondly, even if the defired enhancements were made, the possiblity of incorporating future improvements is small since the students do not understand the language. Finally, more powerful simulation languages have been developed since the original CPESIM II was written.

Since more powerful simulation langinges are now in existance, it seems reasonable that the one of these languages should be used to develop the new CPESIM II. The choosing of the proper simulation language to implement both the hardware and the software CPESIM II will be discussed in greater detail in later chapters. With this new model the

student will get a better environment to enhance his skills in the world of CPE. He will have an easier to understand and more realistic model of a real computer environment. In addition, the instructor can concentrate more on developing realistic scenarios and less time on changing the model to fit these scenarios.

#### Scope

As with any simulation, certain limitations have to be placed on CPESIM II. The number of possible architectures and operating systems i almost limitless. In order to get a model of reasonable size which ill take a reasonable amount of time to execute, a spec fic architecture and operating system was chosen. In partic lar, the architecture will consist of a classic von Neumann machine, which can vary in memory size, allocated and unallocated I/O devices, I/O channels, and number of CPUs. The operating system will be a static partion memory, multiprocessing operating system. In addition there will be a specific software and hardware monitor for the computer system. Interactive processing, distributed processing, and dynamic memory allocation will not be covered in this thesis.

The financial cost and manpower required to use the monitors, to buy more hardware, o to manipulate the computer environment is not addressed. This is a necessary

part of CPE, and should be supplied in the instructor scenario given to the students.

# Order of Presentation

Chapter II describes the system's requirements for CPESIM II. Chapter III describes the transformation of the requirements into the system design. Chapter IV describes the results and validation of CPESIM II, while Chapter V providers a final summary and recommendation for future enhancements. Appendix A provides a detailed representation of the requirements in SADT form. Appendices B and C provide the detailed design in SLAM network form and Structured English, respectfully. Appendix D lists the actual CPESIM II code. Appendix E consists of an instructor's manual for using the simulation, while Appendix F is the student's CPESIM II manual.

#### II. System Requirements

#### Environment

CPESIM II is intended to be used in graduate level computer performance evaluation classes and graduate level queueing theory classes. These classes concentrate on the application of moder: tools and techniques for evaluating a computer in its environment. Specific techniques are presented throughout the term and the student is given case studies to which these techniques are applied.

CPESIM II's high level requirement is to give the student a "realistic" system in which he can apply these techniques to the given case studies. In order to successfully integrate the case study with CPESIM II, the instructor needs to develop a scenario which will demonstrate the performance problems. The scenario will consist of describing the computer, its environment, and possibly its performance problem or desired goal. The description could be in prose form or in the form of accounting and monitor data of in existing system. The student will evaluate the given data and determine if he has enoug 1 data to formulate an improvement hypothesis. If more data is required, then he can run the CPESIM II model to get accounting data, software monitor data or hardware monitor data. Once sufficient data is available he will form a hypothesis that might solve the problem. The student will

then implement his hypothesis by changing the computer configuration of the CPESIM II model and run the model against the workload. The model shall provide "realistic" accounting data, and depending on the request, hardware and software monitor data. The student will analyze the data and determine if other changes are needed. If necessary this process can continue until the student is satisfied with his solution to the original problem. Figure 2 shows this case study flow in SADT (Ref 1) format.

#### Software equirements

. . The software implementing CPESIM II shall provide a "realistic" computer environment to the student. In order to provide this "realistic" environment, the software will be broken into four parts.

The first portion of software will produce a workload data file which may be used as input to the simulation. The instructor will input the different types of job streams in terms of empirical distributions. The program will take these distributions and calculate job streams. The job streams will consist of job parameters such as job name, cpu time required, number of disk I/Os, etc. Each job will then have its parameters written to the workload data file in a format that the computer simulation can use.



<u>.</u>

Sec.

The second portion of CPESIM II will be the user's interface to set up of the computer's architecture and operating system. In this portion of code the specific computer configuration must be specified. The software must be able to describe the hardware configuration in terms of number and speed of cous, I/O devices, and channels; size of main memory; and hardware monitor connection probes. In addition, the I/O matrix between the channels and the I/O devices has to be described. The user interface will also be able to specify the start and stop times for the software monitor and hardware monitor.

The third part of CPESIM II will be the post processing of the accounting data file, the software monitor data file and the hardware monitor data file. The processing will pr duce listings which will be similar in format to real accounting, software monitor and hardware monitor output files. This processing will not affect the original data file. This will allow the students to analyze the raw output using standard statistical software packages like Statistical Package for the Social Sciences (SPSS) (Ref 9).

The last software portion of CPESIM II will be the simulation of the computer and operating system. The architecture will vary slightly according to the the user's input parameters, but the basic machine will be the classic von Neumann computer (Ref 5). It will consist of one or

more cpus, main memory, I/O channels, allocated I/O devices (tapes, card readers and line printers), and unallocatable I/O devices (disks, drums) (Figure 3). The operating system is a basic multiprocessing, multiprogramming, partioned memory operating system (Ref 8). Looking at the hardware and the operating system as a combined unit, a typical job will perform six basic tasks as it flows through the simulation (Figure 4). At jub arrival the job will be spooled (1) onto disk. A ter spooling the job will be placed into the hold queue where it will wait for the job scheduler to allocate resources (2). After the resources are allocated the job will wait in the execute ueue. When a cpu becomes free the cpu will perform a cpu burst (3) on the job. After the cpu burst one of three things will happen. If the job is not finished the job will either go back into the execute queue or perform an I/O (4). If the job is completed it will be plased in the output queue until it can be spooled to the printer (5). The last process (6) is the writing of the data files and the freeing of the job's resources.

In addition to the hardware and the operating system the last software portion will contain both the hardware and the software monitors. The hardware monitor will be an event driven monitor which will record timing and counter data for the cpus, channels and I/O devices. The hardware monitor will not affect the (simulated) speed of the cpu.





.

· · · ·

The software monitor will keep track of the time that a job spends in each of the queues. Unlike the hardware monitor, the software monitor will slow down the relative speed of the simulated cpu. This slowdown represents the increased overhead which a real software monitor causes on a real system.

Appendix A contains the detailed requirements of the simulation in SADT form.

#### Support Requiremen.s

CPESIM II is only one portion of an educational tool in teaching CPE. There is sore to CPE than just applying tools in a "closed" system. With enough money and equipment almost anyone can improve the performance of a computer. To effectively learn all the aspects of CPE, the student needs to be presented with a scenario which represent a "total" computer environment. Scena-ios will aid the student in learning the complexities involved in the decision making process. The sce ario should include data on the present hardware and software configurations, possible limitations in configurations, and costs ass ciated with the different pieces of equipment. In addition to the measurable factors in the system, intangibles like the user's and management's needs and goals should be represented. With a sc mario the simulation will be more than a simple "plug and chug" exercise.

Support is also needed for the student to analyze the results of the simulation run. After the student has implemented his imployment hypothesis, he needs to analyze the affects on the system. In older to do this, he needs access to standard statistical packages like SPSS (Ref 9).

#### Conflicting Requirements

There are a few conflicts with the requirements for CPESIM II. Like any simulation, the conflicts involve flexibility and complexity of the simulation. An increase in flexiblity directly correlates to an increase in complexity. There are an infinite number of computer configurations which are available. In order to get a "reasonable" simulation, limitations must be made in the flexiblity of the system. This "reasonable" simulation must be flexible enough to effectively teach the tools of CPE while simple enough for the student to comprehend the simulation and for the developer to implement it.

## Verification and Validation

7

The verification of CPESIM II will test the accuracy of the model to execute the statements in the proper orcer. This will be done by running a workload of a few programs through the model. These programs will cause the execution of all the statements in the model. A visual inspection will be made to insure that the model executes the statements as expected. How does one validate a simulation that is not a model of anything in real life? That is the problem one has to address when trying to validate 'PESIM II. CPESIM II is a representation of a "computer" in the general sense, but not of any particular model. It should behave like a typical computer given a certain workload, but it should not produce exactly the same results. For that reason, it cannot be directly compared to any real life data from a computer system. Thus, the validation by comparing it to a real life machine will have to be intuitive in nature.

The new CPESIM II should act similar to the old version of CPESIM II, but once again the matching will not be exact. The new version has enhancements, particularly in the job scheduler algorithm and the I/O scheduler algorithm, which the original CPESIM II did not have. As a recall of these enhancements, the output data of the simulations should have some correlation for the same workload, but not an exact correlation.

#### Summary

This chapter has presented only the high level requirements for PESIM II and a general verification and validation procedure. If the reader is interested in the detailed requirements needed to design and test the simulation, he should now read Appendix A before going on to the next chapter which will cover the design of CPESIM II.

#### III. Simulation Design

This chapter discusses the thought process and decisions that were made in the design and development of the CPESIM II simulation software. In particular, this chapter will discuss the simulation language chosen, the design of the simulated computer architecture, operating s"stem, workload carameters, user interface, and post processing software.

## Implementation Languages

57

.

In order to decide on which language or languages would be best to use, certain criteria had to be established. First, the language must be presently supported by one of AFIT's computers, and should have a high likelihood of future support. Secondly, the language should be user friendly. The users should not have to know all the details of simulation but only about those portions which effect the performance of a computer. The third criteria that a language was gauged against was that of its power. The power of a language shall be defined as the ability of the language to handle the mundane bookkeeping associated with simulations, with little or no us r intervention. At the same time the language should give the programmer the ability to go to a lower level if desired.

The original CPESIM II was written in SIMSCRIPT II so Simscript was a natural benchmark against which to judge other languages. In 1979, AFIT supported SIMSCRIPT II and it was one of the more powerful languages. By 1983, more powerful languages had come into being. AFIT is no longer teaching SIMSCRIPT II and most of the expertise at AFIT has been lost. It was questionable whether or not it would be supported in the future. SLAM (Ref 10) on the other hand, is well supported by AFIT and is one of the most powerful languages of the day. The support of SLAM and its power make it a very desirable language, but SLAM is not user friendly. In order to over ome this drawback, a second language was introduced to make a user friendly interface. This interface will use FORTRAN V (refs 2, 3) to modify the configuration data file which initializes the simulation to a particular hardware configuration. This interface shall query t a user about the differ at options available and then generate a configuration data file. Thus the power of SLAM can be used to program the simulation and the user can still have a user friendly system.

#### Computer Architecture

After the simulation language was decided upon, the type of computer architecture to be imulated had to be established. There were two main types of architectures considered. The bus oriented architecture, which is

representative of many of the mini computers, has a central bus which links the cpu, the memory and the I/O devices together. The central-memory architecture uses main memory as the interface between the cpu and the I/O devices. The central-memory architecture was chosen for two reasons. First of all, it is representative of the majority of the large scale computers in which CPE would be used and secondly, it is the architecture which proved itself as a valuable teaching aid in the original CPESIM II.

There are many possible varieties of central-memory computers. The CPESIM II design is a trade off between flexiblity of the different varieties and a reasonable sized simulation. Most of the limits on the simulation were based on having a reasonable simu'ation (in terms of size and speed). Since SLAM was written in FORTRAN, the basic memory allocation for a SLAM program is the same as a FORTRAN program. This memory allocation requires that the size of arrays be specified at compile time. Unlike PASCAL and some of the newer languages, these arrays are initialized at run time and use a fixed amount of memory during the entire simulation. For this reason, dimensions of arrays were carefully chosen. The number of channels, allocatable I/O devices, unallocatable devices, and memory partitions had to be analyzed. The simulation can easily be expanded if desired by changing the limits on these arrays.

The basic computer consists of a single cpu multiprocessing system. Because the cpu is treated as a server by the SLAM compiler, multiple cpu configurations are available, but the hardware monitor will treat all the cpus as one. There will still be one execute queue in the multiple cpu configuration. When a cpu is free, the cpu will get the first job in the execute queue and will perform a time burst on that job. There is no limit on the number of cpus that the simulated computer can handle.

A static partition memory management system is implemented in the simulated computer. There is a maximum of twenty partitions available to the user. Twenty partitions should be a sufficient number of partitions so that the student can tailor the size of the partitions to fit several different workload streams, the input spooler, output spooler and software monitor. There is no limit on the size of the memory partitions. The implementation of a dynamic memory management scheme was invest gated. Based upon experience with the original CPESIM II, dynamic memory management took excessive amounts of real cpu time to execute the simulation. For this reason, it was determined that the simulated computer would not have dynamic memory management.

20

The IOCs connect the I/O devices to memory. There is a limit of five IOCs connected to memory. The limit on the IOCs was based on past experience with CPESIM II. Performance problems such like having all the high-speed devices on one channel, or just a general backlog for an IOC can be modeled with just one or two IOCs. The IOCs can have any data transfer rate desired and be multiplexed to any or all I/O devices. This combination will allow the instructor to create a scenario for just about any I/O performance problem.

The I/O devices are broken into four types. The first type is the card reader. There is only one card reader. The card reader is only used by the input spooler, and can be connect to any or all channels. The card reader can be set to any desired speed (cards per minute). Presently the simulation assumes that the card contains 80 bytes of information. This could be changed by changing one variable in subroutine INTLC. The choice for only one card reader was to simplify the algorithm for the input spooler. Adding the capablity of having multiple card readers does not substantially add to teaching of CPE.

The second type of I/O device is the line printer. Like the card reader, there is only one line printer, and it can be connect to any IOC. The line printer prints a 132 character line at any desired speed (lines per minute).

The allocatable I/O devices (tapes) are the third type of I/O devices. Allocatable devices are assigned to a job prior to the job being placed in the execute queue and released back to the system when the job is placed in the output queue. The I/O requests on allocated devices are spread evenly between the devices allocated to the job. The allocatable I/O devices can be set to any desired speed and be connected to any IOC. There is a limit of ten allocatable devices in any one simulation.

The last type of I/O device is the unallocatable device. These devices are usually disks or drums. The unallocatable devices are assigned to a job when a job requests a unallocatable I/O and released back to the system when that particular I/O is finished. Unallocatable device I/Os are distributed evenly between all unallocatable devices in a round robin fashion. Like the other devices, unall cated I/O devices can be specified at any speed and connected to any or all channels. There is a limit of ten unallocatable devices at one time (Figure 5).



### Operating System

57

The operating system for CPESIM II is a multiprogrammed static partition operating system. The operating system executes five different types of jobs. The first type is a typical applications program. The other four types of jobs are actual parts of the operating system: the input spooler, the output spooler, the job scheduler and a software monitor.

\_\_\_\_

A typical application program gets into the system through the input spooler (Fig re 6). After the job is spooled it is placed into the hold queue to await resources. The job scheduler exami es the jobs in the hold queue, according to the job priorities and the time the job entered the hold queue, and allocates memory partitions and allocatable I/O devices to each job. After the job is allocated it proceeds to the execute queue. When the cpu is free it takes the highest ranking job in the execute queue and performs a cpu burst. Job ranking in the execute queue is done by job type. The job scheduler gets the highest priority, followed by the software monitor start-up job, output spooler, input spooler and user jobs, respectively. The cpu burst lasts until one of four things happen: an I/O was issued for an allocatable I/O device, an I/O was requested issued for an unallocatable device, a timeout occurred, or the job finishes. If an allocatable device I/O request occurred then the job gets placed in the smallest



• •

6

.

.

•.

Fig 6: CPESIM II Job Flow
channel queue which is connected to the requested I/Odevice. The I/O is performed, the channel is freed, and the job is placed back into the execute queue. If an unallocatable device I/O occurred the job gets placed in the proper device queue. Once the device has been acquired, the job is placed in the smallest channel queue which is connected to the device. The 1/0 is then performed. After completion of the I/O the channel and the deviced are freed and the job is placed back into the execute queue. If a timeout occurred then the job is placed back into the execute queue to wait for another time slice. If t e job is completed the job is placed in the output queue and waits for the output spooler to print the job. Also at that time, the memory partition used and the allocatable devices assigned to the job are released back to the operating system.

The input spooler takes the applications programs in the input queue (card reader) and spools them onto the disk. When a new job arrives at the input queue the operating system checks to see if the input spooler is already in memory. If the spooler is not in memory it is placed in the hold queue and must compete for resources and cpu time like any other job. When the input spooler acquires the cpu, it first schedules an I/O to read the cards into a system buffer. Presently, the system buffer is 1K, but it can be changed to any size by modifying subroutine INTLC. After

the buffer is loaded the spooler is placed back into the execute queue. The next time the spooler gets the cpu the buffer is spooled to the disk. This two step process continues until the entire job is spooled. At that time the operating system checks to see if there are any more jobs in the input queue. The input spooler continues until a 1 the jobs in the input queue are spooled. The input spooler is then released from the system and the memory partition is returned to the operating system.

The output spooler takes the job's output file which is stored on disk and prints it. Whenever a job arrives at the output queue, the operating system checks to see if the output spooler is loaded. If necessary the output spooler will be placed in the hold queue and will compete for resources. When the output spooler first executes it will perform an I/O which will load the 1K system buffer from the disk file. The spooler will be placed back into the execute queue so that it can print the buffer on the next time through. The process continues until the entire job has been printed. Like the input spooler, the output spooler continues to spool until its associated queue is empty.

67

The job scheduler allocates memory partitions and allocatable I/O devices to jobs in the hold queue. The job scheduler is loaded in the execute queue (if it is not already) every time a new job arrives in the hold queue or when resources are freed (after each applications job,

spooler or software monitor completion). The job scheduler is always core resident as part of the operating system and does not count against the sultiprogramming level. When the job scheduler obtains the cpu, it looks at every job in the hold queue to see if it can assign resources. Resources are assigned to 'obs according to a priority system. The job scheduler ranks the jobs according to the job's priority attribute. This attribute is p rt of the job's input parameters, which are read in from the workload data file. If there is a tie, the jobs are arranged according to the hold queue arrival time. Each job in the hold queue is analyzed according to this priority scheme. The job scheduler first checks to see if a memory partition is free which can handle the job. If a memory partition is free the schedul 'r thens checks to see if there are enough allocatable I/O devices free. Only if both conditions are met, the job is assigned resources and is taken from the hold queue and placed into the execute queue.

### Software Monitor

The software monitor is like any other job in the simulated system. It is entered into the system and must compete for computer resources. The monitor is loaded into the hold queue at the user specified starting time. It sits in the hold queue until a partition of at least 4K is available. It is then placed in the execute queue until it

can acquire the cpu. When the monitor acquires the cpu it starts the tracing of jobs through five user specified queles. The software monitor then releases the cpu back to the operating system. Once the trace has started the monitor will record data about every job that enters one of the five queues. The queue name, the time spent in the queue, where the job came from, and where the job is going, is recorde: every time a job leaves a queue. Like a real software monitor the monitoring of the queues and the recording of the data puts an extra burden on the system. This burden is represented by making the cpu run at 95% efficiency. When the scheduled stopping time of the monitor occurs, the memory partition is freed and the cpu goes back to running at full efficiency.

### Hardware Monitor

Unlike the software monitor, th hardware monitor does not use up the simulated computer resources. The hardware monitor has its own timers, counters, and data recording devices. It is an event driven monitor which can be connected to any I/O device, channel, or cpu. The hardware monitor has two timer probes and three counter probes. These probes along with the starting time, the stopping time, and the sample data rate are specified in the configuration file. One limitation in the hardware monitor is that the monitor treats mulitple cpus as one if it was

on cpu. The timer or counter connected to the cpus will be pulsed every time any cpu is activated. The output of the hardware monitor is written to a data file which is available for post processing.

### Workload Generation

The software necessary for the generation of the workload data came from the original CPESIM develoced by Capt Lewis (Ref 7). No changes to workload generator were made.

### User Interface

In order to easily change the configuration of the simulated computer, a user interface was written in FORTRAN V. This interface allowes the user to specify the number and types of cpus, allocatable I/O devices, unallocatable devices, and channels. In addition, the connections between the I/O devices and the channels, and the number and size of memory partitions are specified. If the user desires the software monitor and/or the hardware monitor, they will be specified in this interface as well.

The interface was broken down into three sections (Figure 7). The first part of the interface takes an input file called TYPES and reads in the different types of cpus, I/O devices and channels that the student may pick from. The second part queries the student about the simulated computer configuration desired. The program does this by



Fig 7. User Interface Structure

· · · ·

displaying a menu of options for each piece of equipment and allowing the user to make his choice. This section of code also queries the user about the monitors and their parameters. The third section of code takes the user's responses and writes a configuration file, called CONFIG, which can be read by the simulated computer.

6

.

### IV. Verification and Validation

This chapter will discuss the method in which CPESIM II was tested and the process that occurred in validating the model. The chapter is broken into two main sections. The first section will cover the testing that was performed to verify that the code was executing as expected. The second section will discuss the validation process that occurred in validating the new CPESIM II against the old CPTSIM II.

### Verification

The testing of the computer simulation followed the top down implementation of simulation progra. The coding of CPESIM II was broken into five parts and the testing if the program followed these five phases.

Main Program Flow. The first phase concentrated on the testing of the "main" job flow through the computer simulation. The "main" flow consisted of job arrival, input queue, hold queue, execute queue, and output queue. There was not any I/O implemented in this phase and the monitors were not operational. Reading job arrivals from the disk, figuring cpu bursts, writing output to disk files, and general job flow were the major test objectives demonstrated in this phase.

Disk I/O and Channel Implementation. The second phase took the "main" job flow and added unallocatable I/O devices (disks) and channels to the network. The main test objective of this phase was the proper selection of devices and channels for given I/O conditions.

Tape I/O Implementation. The third phase was the implementation of allocatable I/O devices (tapes) into the network. The allocation of the devices in the job scheiuler module was tested, as well as how the tape drives interacted with the channels and disk drives.

Software Monitor Implementation. The fourth phase was the implementation of the software monitor into the network. Key factors which were tasted are the increase cpu time required because of the software monitor and the writing of the job's queue times to the appropriate disk file. <u>Hardware Monitor Implementation</u>. The fifth phase implemented the hardware monitor network. Key factors which were tested in this phase were the counters and timers for the cpu, I/O devices and the channels.

### Validation

CPESIM II validation had to be general in nature. CPESIM II represents a computer in the "general" sense. It is not a model of any particular computer and can not be directly compared to one. The simulation should behave like anyother computer in that changes in the work...d and

configuration of the system should have an affect on the turnaround time. In additio, CPESIM II could not be directly compared to CPESIM for two reasons. First, the alogorithms used in CPESIM II are different the n the algorithms used in CPESIM. Secondly, because of a SLAM imitation on the Cyber, only about 50 jobs could be handled by the simulation. Fifty jobs was not sufficient to perform a detail validation of the syste .

5,

### V. Summary and Recommendations

### Summary

This thesis provides a simulation which aids in the teaching of CPE to graduate students. The simulation, along wich instructor provided scenarios, gives the stulent "hands on" experience in analyzing and solving performance problems in a computer center.

The simulation models a central memory architecture which has multiple processors, continuous memory, I/O channels, allocatable I/O devices, and unallocatable I/O devices. The operating system is a basic multiprocessing, multiprogramming, partioned memory operating system. The user is allowed to ...odify the configuration of the basic s stem by changing the input configuration file. The configuration file, along with a workload data file, is needed to execute the simulation. The output of each simulation run will consist of accounting data, and if requested, monitor data. If the monitors are used, the user has the choice of a software monitor and/or hardware monitor. The software monitor records the length of time a job spends in the queues. The software monitor requires computer resources like any other job. The hardware monitor is an event driven monitor which can track timer and counter data of the cpus, I/O devices and channels. Unlike the

software monitor, the hardware monitor does not compete for computer resources.

The simulation does not implement interactive processing, computer networking or dynamic memory allocation. These limitations were imposed upon the simulation because of the lack of time. These limitations can be resolved in follow-on efforts, but at the present time the simulation simulates the arrival of jobs by a card reader and the printing of jobs by a single line printer.

### Recommendations

CPESIM II was designed to be the nucleus of a tool to teach CPE. Several enhancements to both the simulation and its environment are are readily apparent. The following enhancements could be made to the simulation software:

1. Most of today's large computer systems are interactive in nature. The problems associated with an interactive system cannot be addressed with the present system, so the adding of remote terminals is a logical extension to the model.

2. Another area of improvement is in the world of networking or distributive processing. With the advent of minicomputers and microcomputers, more and more networks have appeared. It is likely that the students will be involved with networks and it would be beneficial if they could study the problems associated with networks.

3. During the development of the simulation, simplifications of a real computer system had to be made. One of the major simplifications that was made was that I/O requests were evenly distributed throughout the execution of a job. It is unlikely that a job's I/Os are evenly distributed. The clustering affect of I/Os could easily be implemented by changing the subroutine "USER".

In addition to the improvements that could be made in the simulation, the following are logical extensions to the front and back end of the simulation:

1. The generation of the workload data still uses the original workload generation program (ref 7). This program can only generate one job stream at a time and it is a problem to generate a multiple job stream workload data file. The ability to easily generate multiple job streams and changing the program from SIMSCRIPT II to SLAM is a logical extension.

2. The post processing of the accounting, software monitor an. hardware monitor data .iles also needs to be written.

3. The tracking of the changes that a student makes to the simulated environment is presently a manual task by the instructor. A database which would track the changes to the configuration would greatly benefit the instructor. In addition this data base could assess charges to the student for using different pieces of equipment and making changes

to the system. By adding this change the student would have incentive to carefully analyze the problem prior to making changes.

A final recommendation would be to do a careful time and space analysis. Typical of most simulations, CPESIM II takes up a large amount of computer time and space. A typical simulation run of 50 jobs with an average simulated cpu time of 150 seconds and 1000 I/Os takes about 400 Cyber seconds. Because of the time and space required, a careful analysis of the time and space required by different simulated configurations and workloads should be done when SLAM is most field to handle more than 50 jobs at one time.

Ć

### Bibliography

 ----- An Introduction to SADT, Structure Analysis and Design Technique, "9022-78R, Softech, Inc., 460 Totten Pond Road, Witham, MA 02154, November 1976.

57

- 2. ----. FORTRAN Extended Vers on 5 Reference Manual (Revision C), Control Data Corp, Sunnyvale, CA, 1930. (60481300).
- 3. ----. <u>SIMCRIPT II.5</u> <u>Reference Handout</u>, CACI, Inc., 12011, San Vicente Blvd., Los Angeles, CA 90049, March 1976.
- 4. Ageloff, Roy and Mojena, Richard. <u>Applied FORTRAN 77</u> <u>Featuring Sturctured Programming</u>. Wadsworth Pubilishing Company, Belmont, CA, 1981.
- 5. Baer, Jean-Loup. <u>Computer Systems Architecture</u>. Computer Science Press, Inc., 11 Taft Court, Rockville, Maryland 20850, 1980.
- 6. Bell, T. E., <u>et al.</u> <u>Computer Performance Analysis:</u> <u>Framework and Initial Phases for a Performance</u> <u>Improvement Effort</u>. A Report prepared for United States <u>Air Force Project Rand</u>, November, 1972. (R-549-1-PR.
- 7. Lewis, Paul C. <u>A Computer Performance Evaluation</u> <u>Education Tool</u>. <u>MS thesis</u>. <u>Wright-Patterson AFB</u>, Ohio: <u>Air Force Institute of Technology</u>, <u>Jecember</u>, 1979. (AD A080154).
- 8. Madnick, Stuart E. and John J. Dovovan. <u>Operating</u> Systems. McGraw-Hill Book Company, New York, NY, 1974.

- 0

- 9. Nie, Norman H., et al. SPSS Statistical Package for the Social Sciences, (Second Editon). New York, NY, McGraw-Hill Book Company, 1975.
- 10. Pritsker, A. Alan B. and Pegden, Claude Dennis. <u>Introduction to Simulation and SLAM</u>. Halsted Press, New York, NY, 1979.

أستعكمه

•

Appendix A CPESIM II SADT Diagrams

Ġ

.

Al CPESIM II

6

a particular computer configuration which will be excented using the workload file as input to the SLAM simulation. The raw output for the CPESIM II will generate a simulated workload as directed by the ictor's input. Either the instructor or the stadent will set up SLAM simulation will be used to generate accounting information, software monitor information and hardware monitor information. instructor's input.



· . ·

5

· . .

# Al.l Determine Workload

5

7

Also each of the job's computer resources (arrival time, cpu time required, etc.) must have a distribution associated with it. After all the jobs by the instructor. Then ior each job stream the start and stop times of that job stream and the arrival distribution will be entered. Also for all the job streams have been generated, the jobs will be merged workload. The start and stop time of the simulation will be entered Determine Workload will create a file of a simulated computer and sorted by arrival time.

1.0 2-V REV: :UXBER: Workloa.1 Parameters 5 Lec 1983 Merge And Dorth Work load Prometers CPESIM Runnie fers Determine Workload Job Simulation Run Time PROJECT: is Resource Reguirements 1.1.3 Job Streams L. ġ Computer Resource Job Streams Sistributi. Determine Distributions Determine Start/stop Time of Smulation 1.1.1 1.1. David L. Uwen TITLE: Stotistical Stream Shart itutistical A1.1 Streum Stop Stop Time Start Time AUTHOR: NUDE:

٠.

- ي

A-5

57

•

Al.3 Simulation Run

6

"spooler" job has Upon job After allocation has taken place the job the job is completed. If an I/O was requested then the I/O is performed and the job goes back into the execute queue. When the job hold queue. In order to spool a job into memory the "spooler" job ho to be executing, which means that the "spooler" job has to be placed ready for the tob scheduler. The job scheduler allocates memory and Execute Job takes a job from the queue and executes the job until a timeout occurs, an I/O is requested or After the job has been printed the job's resources are reljased and output spooler must execute in order to spool a 12h to the printer. arrival the job will be spooled onto the disk and be placed in the Once the job is in the hold queue the job is completes its execution it is sent to the output job spooler. The The simulation run will consist of six major processes. goes to the execution queue. devices to particular jobs. statistics are gathered. Into the hold queue.



.

.

٠.

`\_`

Ó

Al.3.1 Spool Job

•••

6

Upon job arrival the job will be placed into the input queue. The "spooler" job will be sent to the hold queue (Al.3.2). Once the "spooler" is executing it will take a job from the input queue and transfer it to disk.

1.0 REV: A-4 ч 0 1 + 01 NUMBER: 03 5 Dec 1983 Job on Disk Spooler Job Execution s/w Moniter Attributes 1.3.1.2 Transfer Job to Disk CPESIM Input Spoler PROJECT: Spool Job Input Queve Input Queue 1.1.6.2 TITLE: David L. Owen Put Job in A.1.3.1 Job Arrival A D'THOR: NODE:

.

5

-

Al.3.2 Schedule Job

67

Ø queue is searched to see if it can allocate memory or I/O devices to Upon arrival of a Job On Disk (user job), a spooler job, or an job in the hold queue. Once a job has been allocated resources it moves to the execute queue (see Al.3.3). output spooler job, the job will be placed into the hold queue. Whenever a new job gets into the queue or resources are freed, the



Ċ,

. . .

Al.3.3 Execute Job

5

(Al.3.5). If an I/O was requested the the job is sent to Perform I/O jobs are placed into the execute queue. The first job in the execute queue is taken out of the queue and a cpu burst is performed on it. conditions. If the job is completed then it is sent to Output Job After allocating resources, completing an I/O, or timing out, (Al.3.4) and if it timed out it is sent back to the execute queue. At the end of the cpu burst the job will be in one of three



# Al.3.4 Perform I/O

Ò

IOM queue until it gets a channel to its desired device. After acquiring the channel the job gets put into the device queue until it can get the device and perform its I/O. After completion of the I/O the job is sent back to the execute queue. When a job is blocked because of I/O the job gets put into the

A-14

.



• • •

۰.

.

A-15

5

.

Al.3.5 Output Job

G

-

characateristics (job statistics and job resources) are sent to Free After the "output After the job has completed its cpu time, it is ready to be The job is first placed into the output queue and the The job spooler" starts execution the job can then be printed. job is sent to the hold queue. Resources and Collect Job Statistics (Al.3.6). printed. The job "output spooler"



2. P.

.

2

57

Appendix B

# CPESIM II

# SLAM Network Drawings

5

.



B-2

67

-



.

67

.

B-3

÷. .


٠.

67

•.•

B**-4** 



67

.....

La na -

3-5



Q.

<u>]</u>



÷ľ.

-

Ϋ́,

۰.

••••

l v

••••

Ġ7



|

6

۳.



1

•••

•. • •.

5



• • •

67

.

.

8-10



.



18 - C. .

6

Appendix C

•

.

Ø

5

CPESIM II

# Structured English

### Initialization Routine

.

1.	Read CONFIG FILE
2. to l	Initialize the input, hold, output queues and channels nave 0 resources to start with.
3.	Initialize all 'XX' variables.
4.	Initialize memory partition sizes.
5.	Initialize tape speeds and set them to free status.
5.	Initialize disks to be a) available, b) their associated channel connections, c) and, define disk speeds.
7. for	Initialize disk count, disk start time, and disk time each disk to O (used in H/W monitor).
B. Eor	Initailize tape count, tape start time, and tape time each tape to O (used in H/W monitor).
9.	Open job stream, s/w monitor and h/w monitor files

10. Call ARIV at time 0.0

K

### Subroutine ARVL (Job Arrival - Event 1)

1. Read next job's atributes from workload file

2. Initialize cpu time used and i/o time used to 0.0

3. Initialize time of last I/O and time of last tape I/O to 0.0  $\,$ 

5. Initialize cpu time left to cpu time required.

5. Schedule job arrival into the network.

.

### Subroutine ISPO (Put Input Spooler in Hold Q - Event 2)

1. Set arrival time to tnow

2. Set memory size to 4k

3. Set Job Type to 'spooler'

4. Set cpu time used to total cpu time used by the input spooler

5. Initialize all other atributes to 0.0

6. Put spooler into holdq

Subroutine ENT1 (Put Job into Input Queue - Event 3)

1. Enter job into network.

67

1

2. Call ARVL (to find out when next job should arrive).

#### Subroutine CPU (CPU Burst - Event 4)

l. cpu-start-time = tnow
 if h/w-mon = 'on' then
 h/w-mon-count = h/w-mon-count + 1

- 2. If s/w-monitor not on then
   cpu-ratio = 1.0
   else
   cpu-ratio = .95
- 3. If JOB-TYPE = 'user' then CALL USER
- 4. If JOB-TYPE = 'input spooler' then CALL SPOOL
- 5. If JOB-TYPE = 'output spooler' then CALL OSPOL
- 6. If JOB-TYPE = 's/w monitor' then CALL SMON
- 7. If JOB-TYPE = 'job scheduler' then CALL JSCHD
- 8. If JOB-TYPE not equal to one of the above write error

.

#### Subroutine USER (User CPU Burst - call by cpu)

. . .

1. Calculate intermediate variables. cpu time used = cpu time required - cpu time left i/o-time = time-last-i/o + time-between / cpu-ratio tape-time = time-last-tape + time-between-tape / cpu-ration 2. /\* check for job complete \*/ If (cpu-time-left /cpu-ratio time-slice) and ((cpu-time-left / cpu-ratio) (i/o-time)) and ((cpu-time-left / cpu-ratio) (tape-time)) then /\* schedule completion of job \*/ CALL SCHDL(7,tnow + (cpu-time-left / cpu-ratio), atrib) /\* collect cpu timer data for h/w monitor \*/ if h/w-mon = 'on' then cpu-timer = cpu-timer + cpu-time-left / cpu-ratio return end if 3. /\* check for disk io \*/ If ((i/o-time) (tape-time)) and ((i/o-time) (tnow + time-slice)) then /\* schedule time of disk request \*/ CALL SCHDL (8, i/o-time, atrib) /\* collect cpu timer data for h/w monitor \*/ if h/w-mon = on then cpu-timer = cpu-timer + tnow - i/o-time exit end if 4. /\* check for tape i/o \*/ If (tape-time (tnow + time-slice)) then /\* schedule time of the tape request \*/ CALL SCHDL (9, tape-time, atrib) /\* collect cpu timer data for h/w monitor \*/ if h/w-mon = 'on' then cpu-timer = cpu-timer + tnow - tape-time 5. If is none of the above it must be a time out

C-7

/\* schedule time-out \*/
CALL SCHDL (10,tnow + time-slice,atrib)

57

·...

/\* collect cpu timer data for h/w monitor \*/
if h/w-mon = 'on' then
 cpu-timer = cpu-timer + time-slice

Subroutine JSCHD (Job Scheduler - called by CPU)

\_\_\_\_\_

87

```
1.
   for I = 1, num-in-holdq do
      begin
        \mathbf{J}=\mathbf{1}
        While (J .LE. number-of-partitions) and
                (found = false) do
          begin
            if size-of-partion .GT. memory-needed and
                    (partion = free) then
              begin
                if (num-of-free-allocatable-devices .GE.
                    atrib(6) ) then
                  begin
                    acquire partition
                    memory size used = partion size
                    Call ATAPE (acquire number of tapes
                         required)
                    get job from holdq
                    insert job into execq
                    if s/w-monitor = 'on' then
                       call userf(23);
                    found = true
                 end if
             end if
        end while
      end for
2.
      /* collect cpu timer data for h/w monitor */
      if h/w-mon = on then
         cpu-timer = cpu-timer + job-sched-cpu-time
3.
      set job scheduler in execq flag to false
     /* schedule the completion of job scheduler */
      CALL SCHDL(18, job-sched-cpu-time, atrib)
4.
```

### Subroutine SPOOL (Input Spooler - called by CPU)

an se sur cata a

1. /\* collect cpu timer data for h/w monitor \*/
 if h/w-mon = `on` then
 cpu-timer = cpu-timer + spool-cpu-time

67

Ē

- 2. if spool-status = 0 then
   /\* if start of new job spooling \*/
   if num-in-inputq not = 0 do
   begin
   copy job atributes from inputq
   num-cards-left = cards
   if s/w-mon = `on` then call event(23)
   CALL SCHDL(11,tnow + spool-cpu-time,atrib)
   end if
   end if spool-status = 0
- 3. if spool-status = 1
   /\* write buffer to disk \*/
   begin
   /\* schedule completion of cpu portion then
   do i/o \*/
   CALL SCHDL(8,tnow + spool-cpu-time,atrib)
   end if
  4. if spool-status = 2 then
   /\* read cards onto disk \*/
  - begin CALL SCHDL(11,tnow + spool-cpu-time,atrib) end if

Subroutine OSPOL (Output Spooler - called by CPU)

/\* collect cpu timer data for h/w monitor \*/ 1. if h/w-mon = 'on' then cpu-timer = cpu-timer + spool-cpu-time 2. if ospool-status = 0 then /\* start of new job \*/ begin get job from outq if s/w-mon = 'on' then call userf(23) /\* do disk i/o \*/ CALL SCHDL(8, tnow + spool-cpu-time, atrib) end if end if ospool-status = 03. if ospool-status = 1 /\* write buffer to printer \*/ begin CALL SCHDL(14, tnow + spool-cpu-time, atrib) end if if ospool-status = 2 then 4. /\* load buffer from disk \*/ begin /\* do disk i/o \*/ CALL SCHDL(8, tnow + spool-cpu-time, atrib) end if

C-11

### Subroutine SMON (S/W Monitor - called by CPU)

1. s/w-mon = on

6

 2. free cpu /\* Note that the s/w monitor doesn't take
cpu time \*/

Subroutine LJSCH (Load Job Scheduler into EXECQ - Event 5)

Initialize job scheduler atributes

 a. Arrival time = tnow
 b. Job Name = 888 (dummy name)
 c. Job Type = job scheduler
 d. all others = 0.0

 Put job scheduler into EXECQ

·..

3. Set job scheduler in EXECQ flag to true

C-13

Subroutine LOSPL (Load Out Spooler into HOLD Q - Event 6)

1. Set ospooleron flag to on

6

2. Initialize output spooler atributes Arrival time = tnow a. Job Name = 998 (dummy name) b. Memory size = 4.0 kс. d. Job Type = output spooler CPU Time Used = Total CPU Time Used e. by Output Spooler f. All other atributes = 0.03. Put output spooler into hold queue

### Subroutine 7 Job Complete

- If s/w mon = 'on' then call event(23)
   cpu-time-used = cpu-time-used + (tnow cpu-start-time)
   move job into outq
   Free cpu
- 5. Free memory partion
- 6. Free tapes
- 7. Put job-scheduler into execq (if it is already there)

57

Subroutine DISKS (Disk I/O - Event 8)

- 2. free cpu

- 3. Look up next-disk
  Disk used = next-disk
  If h/w-mon = `on` then
  disk-count(X) = disk-count(X) + 1
  disk-start(X) = tnow
- 4. Update next-disk
- 5. Await Disk "X" (in SLAM Disk Network)

Subroutine 9 Tape I/O

1.	<pre>cpu-time-used = cpu-time-used + (tnow - cpu-start-time) cpu-time-left = cpu-time-left - (tnow - cpu-start-time)</pre>							
	<pre>* cpu-ratio start-io = tnow</pre>							
2.	free cpu							
3.	Look up next-tape If h/w-mon = 'on' then tape-count(X) = tape-count(X) + 1 tape-start(X) = tnow							
4.	Update next-tape							
5.	Look in matrix for channels for tape "X"							
6.	Find smallest channel q and put job in channel q							
7.	Await data-rate portion of channel							
8.	<pre>If h/w-mon = `on` then   tape-timer(X) = tape-timer(X) + (tnow +        tape-speed(X)) - tape-start(X)   channel-time(q) = channel-timer(q) + tape-speed(X)</pre>							

9. CALL SCHDL(46,tnow + tape-speed(X),atrib)

7

H

AD-A138 111 UNCLASSIFI	CPESIN II: PERFORMANCE WRIGHT-PATT D DEC 83 AFIT	R <b>2/4</b> NL			
9					



MICROCOPY RESOLUTION TEST CHART NATIONAL BUREAU OF STANDARDS 1963-4

J

.

### Subroutine TIMEO (Time Out - Event 10)

cpu-time-used = cpu-time-used + (tnow - cpu-start-time)
 cpu-time-left = cpu-time-left - (tnow - cpu-start-time)
 \* cpu-ratio
 Free cpu

4. Place job back into execq

C-18

The State St

#### Suroutine RCARD (Read Cards into Buffer - Event 11)

- 2. If num-cards-left buff-size then
   num-cards-read = num-cards-left
   else
   num-cards-read = buff-size

3. num-cards-left = num-cards-left - num-cards-read

4. Free cpu

5. /\* Schedule completion of loading buffer \*/ CALL SCHDL(14, card-speed, atrib)

ید مدین است. آرکار در از دوری ۲۰۰۰ (۲۰۰۱ میکند. ا

#### Subroutine DCOMP (Disk Transfer Completed - Event 12)

- Free channel of portion job was using
   Free disk
- 3. i/o-time-used = i/o-time-used + (tnow - start-io) if JOB-TYPE = 'user' 4. time-last-i/o = tnowPut job back into execq end if 5. If JOB-TYPE = 'spooler' then begin /\* check for completion of spooling \*/ if num-cards-left .LE. 0 then begin spool-status = 0insert job and its atributes into holdq if num-in-inputq not = 0 then begin put spooler back into execq end else begin
  - else begin spooleron = 0 free memory partion of spooler put job-scheduler in execq end if inputq not = 0 end else begin spool-status = 2 /\* ready to read more cards \*/ put spooler back into execq end if

end if

.

1

6. If JOB-TYPE = "outspooler" then begin ospool-status = 1 /\* ready to print buffer \*/ put ospooler back into execq end if

C-20

### Subroutine TCOMP (Tape I/O Completion - Event 13)

1. Time-last-tape-i/o = tnow

έ.

- 2. Free channel of portion job was using
- 3. i/o-time-used = i/o-time-used + (tnow start-io)
- 4. Insert job into execq

## Subroutine FULLB (Finish Loading Buff w/ Cards - Event 14)

- 1. /\* Ready to read buffer to disk \*/
  spool-status = 1
- 2. Insert spooler back into execq

### Subroutine CHANL (Find Channel - Event 15)

1. Look in matrix for channels for disk "X"

Į.

- 2. Find smallest channel q and put job in channel q If h/w-mon = 'on' then channel-count(q) = channel-count(q) + 1
- 3. Await data-rate portion of channel (in SLAM Channel Network)

C-23

### Subroutine DTRAN (Disk Transfer - Event 16)

1. if h/w-mon = 'on' then
 disk-timer(X) = disk-timer(X) + tnow +
 disk-speed(X) - disk-start(X)

channel-timer(q) = disk-speed(X)

2. /\* Schedule completion of disk transfer \*/ CALL SCHDL(12,tnow + disk-speed(X),atrib)

#### Subroutine FPRNT (Finished Print Buff - EVENT 17)

```
If num-lines-left = 0 then
1.
      begin
        ospool-status = 0
        release job from outq
        if s/wmon = on then event(23)
        if num-in-outq = 0 then
           begin
             free memory-partion of outspooler
             ospool = 0
             ospooleron = 0
             put job scheduler into execq
          else
           begin
             ospool-status = 0
             put outspooler back into execq
         end if
      end
     else
      begin
        ospool-status = 2
        put outspooler back into execq
    end if
```



. . . . . . .

<u>es la Marina de Carlo de Social de Carlo de Social de Carlo de Ca</u>

100.00
## Subroutine JSCOM (Job Scheduler Completion - Event 18)

1. Free Cpu

1

#### Subroutine ACTLG (Accounting Data - Event 19)

 Write arrival-time, cpu-time-used, memory-used, i/o-time-used, priority, allocated-devices, and tnow to accounting data file.
 Note: alot more data is available for accounting data, if desired.

#### Subroutine HWMON (H/W Monitor Collection - Event 20)

1. Write tnow, cpu-counter, cpu-timer, all disk, tapes, and channels counters and timers to the h/w monitor raw data file.

2. Reset counters and timers to zero

------

## Subroutine LSMON (Turn S/W Monitor On - Event 21)

- 1. Open s/w monitor raw data file
- 2. Insert s/w monitor into hold queue



# Subroutine ESMON (Turn S/W Monitor Off - Event 22)

- 1. Close s/w monitor raw data file
- 2. Free memory partition

ş

3. Put Job scheduler into execq

C-30

## Subroutine SMON (Write S/W Monitor Data - Event 23)

1. Write qname, (time-entered - tnow), input-from, and output-to to S/W Monitor file.

يحصب بالانبيلاسين يهرانا الالاليا تهم

ś

#### Subroutine PBUFF (Print Buffer - Event 24)

Note: cpu time for spooling is charged to job but is not counted against the job's cpu time required to execute the job.

- 2. Free cpu
- 3. if num-lines-left buf-size-lines then
   num-lines-printed = num-lines-left
   else
   num-lines-print = buf-size-lines
- 4. num-lines-left = num-lines-left num-lines-printed
- 5. /\* schedule completion of printing \*/ CALL SCHDL(17,tnow + num-lines-printed \* printer-speed, atrib)

Appendix D CPESIM II

ŧ

ŧ

Code



```
**********************************
C****
       DATE: 5 DEC 1983
       VERSION: 1.0
       TITLE: CPESIM II Discrete File
       FILENAME: SIMF
       AUTHOR: David L. Owen
       SOFTWARE SYSTEM: CPESIM II
       USE: Used in combination with SIMS to run CPESIM
       CONTENTS: EVENTS 1-24, INTLC, EVENT, ATAPE,
           JSCHD, OSPOL, SPOOL, SSMON, USER, USERF
       FUNCTION: Performs the discrete portion of CPESIM*
        C*
       PROGRAM MAIN(INPUT, OUTPUT, TAPE5=INPUT, TAPE6=OUTPUT, TAPE7, TAPE11,
     1TAPE12, TAPE13, TAPE14, TAPE15)
       DIMENSION NSET(20000)
       COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP
     1, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100)
     1,SSL(100), TNEXT, TNOW, XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
     1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM,
     1 PCNT, PSTRT, PFIN, PSUM
     COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10)
      COMMON/UCOM4/CARD(6), PRNTR(6)
      COMMON/UCOM5/IQ(5)
       COMMON QSET(20000)
       EQUIVALENCE (NSET(1), QSET(1))
      NNSET = 20000
      NCRDR = 5
       NPRNT = 6
       NTAPE = 7
       CALL SLAM
       STOP
       END
```

С

С C

С С

С С

С

С

С

С

С

С

С

С

```
С
С
        DATE: 5 DEC 1983
С
        VERSION: 1.0
С
С
        NAME: INTLC
С
        EVENT NUMBER: N/A
С
        FUNCTION: Initialize simulation
С
        FILES READ: CONFIG
С
        FILES WRITTEN: none.
       EVENTS CALLED: 21, 20, 1
С
С
        CALLING EVENTS
                        ne.
С
        SUBROUTINES CALLED: ALTER
С
        CALLING SUBROUTINES: none.
С
С
        AUTHOR: David L. Owen
С
        HISTORY: n/a.
С
C*:
   **********************************
       SUBROUTINE INTLC
       COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP
     1, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100)
     1,SSL(100), TNEXT, TNOW, XX(100)
      COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
      COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
     1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM,
     1 PCNT, PSTRT, PFIN, PSUM
      COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10),TFIN(10)
      COMMON/UCOM4/CARD(6), PRNTR(6)
      COMMON/UCOM5/10(5)
      COMMON/UCOM6/IFROBE(5)
      DIMENSION DSPEED(10), DCOST(10), TSPEED(10), TCOST(10),
     1
         CSPEED(5), CCOST(5),
     1
         PART2(20), IDISK(10,5),
           ▶ E(10,5), IR EAD(5), IPR INT(5), ISW MQ(5), IHW CNT(5)
     1
      CHARACTER DNAME(10)*20, TNAME(10)*20, CNAME(5)*20,
     1
        RNAME*20, PNAME*20, UNAME*20, SANSWR*1, HANSWR*1, CRATE*20
     REAL HSEC
      PARAMETER (FREE = 0.0, BUSY = 1.0, OFF = 0.0, ON = 1.0)
C*** OPEN CONFIGURATION FILE
     OPEN(UNIT=15, FILE="TAPE15", STATUS="OLD", ERR=990, IOSTAT=IOS,
     1
           ACCESS='SEQUENTIAL', FORM='FORMATTED')
```

\*\*\* C\*\*\* READ DATA FROM CONFIGUARATION FILE C\*\*\* NUMBER OF CPUS READ (UNIT=15, FMT=100, IOSTAT=IOS, ERR=999)NCPUS \*\*\*\* TYPE OF CPU READ (UNIT=15, FMT=101, IOSTAT=IOS, ERR=999) UNAME, USPEED, UCOST 1 C\*\*\* TIME SLICE READ (UNIT=15, FMT=104, IOSTAT=IOS, ERR=999)TSLICE C\*\*\* READ NUMBER OF MEMORY PARTIONS READ (UNIT=15, FMT=100, IOSTAT=IOS, ERR=999)NPARTS C\*\*\* **READ SIZE OF MEMORY PARTIONS** DC 40 I=1,NPARTS READ (UNIT=15, FMT=104, IOSTAT=IOS, ERR=999)PART2(I) 40 CONTINUE C\*\*\* **READ NUMBER OF CHANNELS** READ (UNIT=15, FMT=100, IOSTAT=IOS, ERR=999)NCHAN C\*\*\* READ CHANNEL'S NAME, SPEED AND COST DO 41 I=1, NCHAN READ (UNIT=15, FMT=101, IOSTAT=IOS, ERR=999) CNAME(I), CSPEED(I), CCOST(I) 1 CONTINUE 41 C\*\*\* READ NUMBER OF DISKS READ (UNIT=15, FMT=100, IOSTAT=IOS, ERR=999)NDISK C\*\*\* READ DISK'S NAME, SPEED, COST AND CHANNEL CONNECTIONS DO 42 I=1.NDISK READ (UNIT=15, FMT=105, IOSTAT=IOS, ERR=999) 1 DNAME(I), DSPEED(I), DCOST(I), (IDISK(I,K),K=1,5) 42 CONTINUE C\*\*\* READ NUMBER OF TAPES READ (UNIT=15, FMT=100, IOSTAT=IOS, ERR=999)NTAPE C\*\*\* READ TAPE'S NAME, SPEED, COST AND CHANNEL CONNECTIONS DO 43 I=1.NTAPE**READ (UNIT=15, FMT=105, IOSTAT=IOS, ERR=999)** TNAME(I), TSPEED(I), TCOST(I), (ITAPE(I,K), K=1,5)1 43 CONTINUE C\*\*\* READ CARD READER'S NAME, SPEED, COST AND CHANNEL CONNECTIONS READ (UNIT=15, FMT=105, IOSTAT=IOS, ERR=999) RNAME, RSPEED, RCOST, (IREAD(K), K=1, 5) 1 \*\*\* READ LINE PRINTER'S NAME, SPEED, COST AND CHANNEL CONNECTIONS READ (UNIT=15, FMT=105, IOSTAT=IOS, ERR=999) PNAME, PSPEED, PCOST, (IPR INT(K), K=1, 5) 1 \*\*\* READ IF SOFTWARE MONITOR IS DESIRED READ (UNIT=15, FMT=106, IOSTAT=IOS, ERR=999)SANSWR READ START TIME, STOP TIME, AND QUEUES IF DESIRED C\*\*\* IF (SANSWR .EQ. 'Y') THEN READ (UNIT=15, FMT=107, IOSTAT=IOS, ERR=999) 1 ISDAY, ISHOUR, ISMIN, SSEC READ (UNIT=15, FMT=107, IOSTAT=IOS, ERR=999)

```
1
               JSDAY, JSHOUR, JSMIN, SSSEC
        READ (UNIT=15, FMT=109, IOSTAT=IOS, ERR=999)(ISWMQ(I), I=1,5)
     END IF
C*** READ IF HARDWARE MONITOR IS DESIRED
     READ (UNIT=15, FMT=106, IOSTAT=IOS, ERR=999) HANSWR
C*** READ START TIME, STOP TIME, TIMERS AND COUNTERS IF DESIRED
     IF (HANSWR .EQ. 'Y') THEN
        READ (UNIT=15, FMT=107, IOSTAT=IOS, ERR=999)
    1
               IHDAY, IHHOUR, IHMIN, HSEC
        READ (UNIT=15, FMT=107, IOSTAT=IOS, ERR=999)
               JHDAY, JHHOUR, JHMIN, SHSEC
    1
        READ (UNIT=15, FMT=104, IOSTAT=IOS, ERR=999)RATEHW
        READ (UNIT=15, FMT=109, IOSTAT=IOS, ERR=999)
               IHTIM1, IHTIM2, (IHW CNT(I), I=1,3)
    1
     END IF
100
     FORMAT(12)
     FORMAT(A20, 1X, F10.4, 1X, F8.2)
101
103
     FORMAT(15)
104
     FORMAT(T22, F10.4)
105
     FORMAT(A20,1X,F10.4,1X,F8.2,1X,5(1X,I1))
106
     FORMAT(A1)
107
     FORMAT(T22,3(14,1X),F8.4)
109
     FORMAT(5(12,1X))
     PRINT *. CONFIGURATION FILE READ SUCCESSFULLY
***
C*** PRINT DATA FROM CONFIGUARATION FILE
C*** NUMBER OF CPUS
     PRINT *, NCPUS
C***
    TYPE OF CPU
     PRINT *, UNAME, USPEED, UCOST
C###
    TIME SLICE
     PRINT *, TIME SLICE = ', TSLICE
C*** PRINT NUMBER OF MEMORY PARTIONS
     PRINT *.NPARTS
C***
     PRINT SIZE OF MEMORY PARTIONS
     DO 240 I=1, NPARTS
        PRINT *, I, PART2(I)
240
      CONTINUE
C***
     PRINT NUMBER OF CHANNELS
     PRINT *, NCHAN
C***
     PRINT CHANNEL'S NAME, SPEED AND COST
     DO 241 I=1, NCHAN
        PRINT *, CNAME(I), CSPEED(I), CCOST(I)
```

```
241
      CONTINUE
C***
     PRINT NUMBER OF DISKS
      PRINT *, NDISK
C***
     PRINT DISK'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
     DO 242 I=1,NDISK
      PRINT *, DNAME(I), DSPEED(I), DCOST(I),
           (IDISK(I,K),K=1,5)
     1
242
      CONTINUE
C###
     PRINT NUMBER OF TAPES
     PRINT *, NTAPE
C*** PRINT TAPE'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
     DO 243 I=1,NTAPE
         PRINT *, TNAME(I), TSPEED(I),
     1 TCOST(I), (ITAPE(I, K), K=1, 5)
243
      CONTINUE
C***
     PRINT CARD READER'S NAME, SPEED, COST AND CHANNEL CONNECT
     PRINT *, RNAME, RSPEED, RCOST,
           (IREAD(K), K=1, 5)
     1
C***
    PRINT LINE PRINTER'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
     PRINT *, PNAME, PSPEED, PCOST,
     1
           (IPRINT(K), K=1, 5)
C*** PRINT IF SOFTWARE MONITOR IS DESIRED
      PRINT *. SANSWR
C***
     PRINT START TIME, STOP TIME, AND QUEUES IF DESIRED
      IF (SANSWR .EQ. 'Y') THEN
         PRINT *, ISDAY, ISHOUR, ISMIN, SSEC
         PRINT *, JSDAY, JSHOUR, JSMIN, SSSEC
         PRINT *,(ISWMQ(I),I=1,5)
     END IF
C*** PRINT IF HARDWARE MONITOR IS DESIRED
     PRINT *, HANSWR
     PRINT START TIME, STOP TIME, TIMERS AND COUNTERS IF DESIRED IF (HANSWR .EQ. 'Y') THEN
C***
         PRINT *, IHDAY, IHHOUR, IHMIN, HSEC
         PRINT *, JHDAY, JHHOUR, JHMIN, SHSEC
        PRINT *, RATEHW
         PRINT *, IHTIM1, IHTIM2, (IHVCNT(I), I=1, 3)
      END IF
C*** INITIALIZE SIMULATION VARIABLES
                                                 ***
C*** MAKE THE INPUT QUEUE HAVE O RESOURCES AT START OF PROGRAM
      CALL ALTER(1,-1)
```

C\*\*\* MAKE THE HOLD QUEUE HAVE O RESOURCES AT START OF PROGRAM CALL ALTER(2,-1)

C***	CONFIGURE MODEL TO HAVE THE RIGHT NUMBER OF CPUS NCPUS = NCPUS - 1 CALL ALTER(3,NCPUS)
C***	MAKE THE OUTPUT QUEUE HAVE O RESOURCES AT START OF PROGRAM CALL ALTER(4,-1)
C***	INITIALIZE SPOOLERON FLAG
C***	INITIALIZE S/W MONITOR FALG XX(2) = OFF
C***	INITIALIZE NUMBER OF MEMORY PARTIONS XX(3) = NPARTS
C***	INITIALIZE PARTITION SIZES AND STATUS DO 5 I=1,XX(3)
_	PART(I,1) = PART2(I) PART(I,2) = FREE
5	CONTINUE $TEMP = XX(3) + 1$
	DO 6 I=TEMP, 20
	PART(I,1) = 0. $PART(I,2) = BUSY$
6	CONTINUE
C***	INITIALIZE THE CPU TO RUN AT 100% CAPACITY
C***	NOTE: WHEN S/W MON IS RUNNING IT GETS SET TO 95% BY SSMON
~***	XX(5) = 1.0
C***	$\frac{1}{2} \frac{1}{2} \frac{1}$
C***	INITIALIZE TIME SLICE (IN SECONDS)
-	XX(7) = TSLICE
C***	INITIALIZE NUMBER OF TAPE DRIVES
	XX(9) = NTAPE
C***	INITIALIZE NUMBER OF AVAILABLE TAPE DRIVES
***	XX(10) = XX(9) TNTAT TZE TADEC CDEED AND CTATUC
C	$\frac{1}{1} \frac{1}{1} \frac{1}$
	TAPE(I, I) = TSPEED(I)
	TAPE(I,2) = FREE
_	TAPE(1,3) = 0.
7	CONTINUE
CHHH	INITIALIZE TAPES CHANNEL CONNECTIONS AND SPEED
	DO 11 1-1,88(7) DO 60 K=1 5
	TAP(I,K) = ITAPE(I,K)
60	CONTINUE
	TAP(T 6) = TSPERD(T)

	TAP(1,7) = 0.
11	CONTINUE
C***	INITIALIZE NEXT TAPE
	XX(24) = 0.
C***	INITIALIZE JOB SCHEDULER IN EXECO FLAG
•	XX(11) = OFF
***	INITIAL THE OSPOOLEDON FLAC
•	YY(12) = OFF
C***	INITIALIZE CON TIME DECUTOED FOR TOR SCHEDITER
C	$\frac{1}{2} \frac{1}{2} \frac{1}{2} = \frac{1}{2} $
***	TNITTALIZE COULTIME FOR TNDIT COMPLET TO EVECUTE
C	WY(14) - 001
C***	TNITIALIZE INDUT COONED STATUS TO A NEW IOR
C	YY(15) = 0.0
****	$\Delta A(1) = 0.0$
C	WY(16) = 0.0
***	INITIALIZE BLOCK SIZE (BVTES)
0	$\frac{1}{1}$
	AA(17) = 1024.
***	TNITTALITE CADD DEADED CDEED (CADDS /MINITE)
<u> </u>	INITIALIZE CARD READER SPEED (CARDS/MINUTE)
***	AA(10) - KOPLED Initialize Carr Decorr (Dytec)
<u> </u>	$\frac{1}{1} \frac{1}{1} \frac{1}$
****	AA(17) = 00.
<u> </u>	$\frac{1}{2} \frac{1}{2} \frac{1}$
C+++	$AA(20) = MINI((AA(17) / AA(17)) = \cdot J)$
6	INITIALIZE CARD READER SPEED AND CHANNELS
	$\frac{1}{2} \frac{1}{2} \frac{1}$
<b>4</b> 1.	CARD(I) = IREAD(I)
04	CONTINUE
C+++	TNITIAL TOP NEWT DICK
6	$\frac{1}{2} \frac{1}{2} \frac{1}$
~***	AA(21) = 1
6	INITALIZE NUMBER OF DISK DRIVES
~***	AA(22) = NUIDA INITIALIZE DICKC CONVECTIONS AND CREEDS
6	INITIALIZE DISKS CONNECTIONS AND SPEEDS
	$DU = 0$ (1 $F_{-1}$ )
	$\frac{1}{2} \frac{1}{2} \frac{1}$
<i>~</i>	DISK(1,K) = IDISK(1,K)
01	
•	DISK(1,0) = DSPEED(1)
8	CONTINUE
C===	INITAILIZE NUMBER OF CHANNELS
	XX(23) = NCHAN
CRAR	SET CHANNEL DATA RATE FLOW
	DO 65 I=1, NCHAN
	ITEMP = CSPEED(I)
	ITEMP2 = 30 + I
	CALL ALTER(ITEMP2,ITEMP)

```
65
      CONTINUE
C***
     INITIALIZE PRINTER SPEED (LINES PER MINUTE)
      XX(25) = PSPEED
C***
     INITLALIZE LINE RECORD SIZE (BYTES)
      XX(26) = 132.
C***
     INITIALIZE LINES PER BUFFER
      XX(27) = NINT(XX(17) / XX(26) - .5)
C***
     INITIALIZE LINE PRINTER CHANNEL CONNECTIONS
      DO 62 I=1,5
         PRNTR(I) = IPRINT(I)
62
      CONTINUE
C*** INITIALIZE H/W MONITOR FLAG
      XX(30) = OFF
C*** INITIALIZE CPU COUNTER FOR H/W MONITOR
      XX(31) = 0.
     INITIALIZE CPU SUM FOR H/W MONITOR
C***
      XX(32) = 0.
C***
     INITIALIZE H/W MONITOR START/STOP TIME AND INTERVAL
      IF (HANSWR .NE. 'Y') THEN
C*
         INTIALIZE TO INFINITE AND NO PROBE CONNECTIONS
         XX(33) = 999999999.
         XX(34) = 9999999999.
         XX(35) = 999999999.
         IPROBE(1) = 0
         IPROBE(2) = 0
         IPROBE(3) = 0
         IPROBE(4) = 0
         IPROBE(5) = 0
       ELSE
         XX(33) = (IHDAY * 86400.) + (IHHOUR * 3600.) +
     1
                  (IHMIN * 60.) + HSEC
         XX(34) = (JHDAY * 86400.) + (JHHOUR * 3600.) +
     1
                  (JHMIN * 60.) + SHSEC
         XX(35) = RATEHW
         IPROBE(1) = IHTIM1
         IPROBE(2) = IHTIM2
         IPROBE(3) = IHVCNT(1)
         IPROBE(4) = IHWCNT(2)
         IPROBE(5) = IHWCNT(3)
     END IF
C***
     INITIALIZE CPU START TIME FOR H/W MONITOR
      XX(36) = -1.0
C***
     INTIALIZE CPU FIN TIME FOR H/W MONITOR
      XX(37) = 0.
C***
     INITIALIZE TOTAL CPU TIME USED BY INPUT SPOOLER
      XX(40) = 0.
```

```
***
      INITIALIZE TOTAL CPU TIME USED BY OUTPUT SPOOLER
      XX(41) = 0.
      INITIALIZE TOTAL CPU TIME USED BY JOB SCHEDULER
****
      XX(42) = 0.
C*** INITIALIZE S/W MONITOR START/STOP TIME AND QUEUES
      IF (SANSWR .NE. 'Y') THEN
         XX(43) = 999999999.
         XX(44) = 999999999.
         IQ(1) = 0
         IQ(2) = 0
         IQ(3) = 0
         IQ(4) = 0
         IQ(5) = 0
        ELSE
         XX(43) = (ISDAY * 86400.) + (ISHOUR * 3600.) +
     1
                  (ISMIN * 60.) + SSEC
         XX(44) = (JSDAY * 86400.) + (JSHOUR * 3600.) +
                  (JSMIN * 60.) + SSSEC
     1
         DO 98 I=1,5
           IQ(I) = ISWMQ(I)
98
         CONTINUE
      END IF
C***
     OPEN JOB STREAM FILE
      OPEN(UNIT=11, FILE= TAPE11, STATUS= OLD,
           ACCESS='SEQUENTIAL', FORM='FORMATTED')
     1
C*** OPEN ACTIVITY LOG FILE
      OPEN(UNIT=12, FILE='TAPE12', STATUS='NEW',
     1
           ACCESS='SEQUENTIAL', FORM='FORMATTED')
      IF (SANSWR .EQ. 'Y') THEN
C***
         OPEN S/W MONITOR DATA FILE
         OPEN(UNIT=14, FILE='TAPE14', STATUS='NEW'
              ACCESS='SEQUENTIAL', FORM='FORMATTED')
     1
C***
         SCHEDULE START OF S/W MONITOR
         CALL SCHDL(21,XX(43),ATRIB)
      END IF
      IF (HANSWR .EQ. 'Y') THEN
C***
         OPEN H/W MONITOR DATA FILE
         OPEN(UNIT=13, FILE= TAPE13, STATUS= NEW,
              ACCESS='SEQUENTIAL', FORM='FORMATTED')
     1
C***
         SCHEDULE START OF H/W MONITOR
         CALL SCHDL(20, XX(33), ATRIB)
C***
```

SCHEDULE STOP OF H/W MONITOR CALL SCHDL(20,XX(34),ATRIB)

END IF

DO 860 I=1,44 PRINT \*, XX(',I,') = ',XX(I)

- 860 CONTINUE
- C\*\*\* SCHEDULE THE READING OF THE FIRST JOB'S ATRIBUTES CALL SCHDL(1,0.,ATRIB)

RETURN

- 990 PRINT \*, ERROR IN OPENING CONFIG FILE IOSTAT = ', IOS RETURN
- 999 PRINT \*, ERROR IN READING CONFIG FILE IOSTAT = ', IOS RETURN END

\*\*\*\*\* C#1 С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: EVENT С EVENT NUMBER: N/A FUNCTION: CALLS DIFFERENT EVENTS ACCORDING TO THE С С EVENT NUMBER PASSED. С FILES READ: NONE. С FILES WRITTEN: NONE С EVENTS CALLED: 1 - 24С CALLING EVENTS: ALL С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: ALL С AUTHOR: DAVID L. OWEN С HISTORY: N/A С С SUBROUTINE EVENT(I) COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP 1, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) GO TO (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17, 1 18,19,20,21,22,23,24),1 C\*\*\* ARRIVAL OF JOB AT CARD READER CALL ARVL 1 RETURN C\*\*\* PUT INPUT SPOOLER INTO HOLD QUEUE 2 CALL ISPO RETURN C\*\*\* PUT JOB INTO INPUT QUEUE 3 CALL ENT1 RETURN C\*\*\* PERFORM CPU BURST 4 CALL CPU RETURN C\*\*\* LOAD JOB SCHEDULER INTO EXECUTE QUEUE CALL LJSCH 5 RETURN C\*\*\* LOAD OUTPUT SPOOLE INTO SYSTEM

6	CALL LOSPL RETURN
C*** 7	USER JOB COMPLETE CALL JCOMP RETURN
C*** 8	PERFORM DISK I/O CALL DISKS RETURN
C*** 9	PERFORM TAPE I/O CALL TAPES RETURN
C*** 10	USER JOB TIME-OUT CALL TIMEO RETURN
C*** 11	READ CARDS INTO BUFFER CALL RCARD RETURN
C*** 12	DISK TRANFER COMPLETED CALL DCOMP RETURN
C*** 13	TAPE I/O COMPLETION CALL TCOMP RETURN
C*** 14	FULL BUFFER CALL FULLB RETURN
C*** 15	FIND CHANNEL CALL CHANL RETURN
C*** 16	DISK/TAPE TRANSFER CALL DTRAN RETURN
C*** 17	FINISHED PRINTING BUFFER CALL FPRNT RETURN
C*** 18	COMPLETION OF JOB SCHEDULER CALL JSCOM RETURN

C*** 19	WRITE ACCOUNTING DATA TO DISK FILE CALL ACTLG RETURN
C*** 20	H/W MONITOR COLLECTION CALL HWMON RETURN
C*** 21	LOAD S/W MONITOR INTO HOLD QUEUE CALL LSMON RETURN
C*** 22	END OF S/W MONITOR TRACE CALL ESMON RETURN
C*** 23	WRITE S/W MONITOR DATA TO DISK CALL SMON RETURN
C*** 24	PRINT BUFFER CALL PBUFF RETURN END

С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: ARVL С C. **EVENT NUMBER: 1** FUNCTION: READS JOBS ATTRIBUTES AND SCHEDULES THE\* С С THE ARRIVAL TIME OF THE JOB С FILES READ: WORKDES. С FILES WRITTEN: NONE. С **EVENTS CALLED: 3** С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С AUTHOR: DAVID L. OWEN С С HISTORY: N/A. С SUBROUTINE ARVL COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP 1, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM READ ATRIBUTES FROM WORKLOAD FILE С 50 READ (UNIT=11, FMT=1, END=999, ERR=999)(ATRIB(1), I=1, 10) FORMAT(12X, F10.4, 1X, F10.0, 1X, F3.0, 1X, F3.0, 1X, F2.0, 1X, F2.0, 1 1 1X, F4.0, 1X, F5.0, 1X, F4.0, 1X, F4.0C\*\*\* IF THE JOB REQUIRES O CPU TIME THEN REJECT JOB C\*\*\* AND READ NEXT JOB'S ATRIBUTES IF (ATRIB(3) .LE. 0.0) THEN PRINT \*, JOB REJECTED: REQUESTED O CPU TIME PRINT \*,(ATRIB(I),I=I,21) GOTO 50 END IF CONVERT HOURS TO SECONDS FOR ARRIVAL TIME C\*\*\* ATRIB(1) = ATRIB(1) \* 3600.C\*\*\* CONVERT CPU TIME TO THE RELATIVE SPEED OF THE CPU

D-15

ATRIB(3) = ATRIB(3) \* XX(6)

C****	** CALCULATE OTHER ATRIBUTES
	DO 5 I=11,24 ATRIR(12) = 0.0
5	CONTINUE
C***	I/O TIME USED = O
C***	CPU TIME LEFT = CPU TIME REQUIRED ATRIB(18) = ATRIB(3)
C***	SET ATRIB(21) TO ARRIVED WHEN S/W MON NOT ON ATRIB(21) = 99.
C****	SCHEDULE JOB ARRIVAL AT PROPER TIME CALL SCHDL(3,ATRIB(1) ~ TNOW,ATRIB)
999	RETURN

END

```
С
С
       DATE: 5 DEC 1983
С
       VERSION: 1.0
C
С
       NAME: ISPO
С
       EVENT NUMBER: 2
С
       FUNCTION: INITIALIZES INPUT SPOOLER AND PUTS
С
               SPOOLER INTO HOLD QUEUE
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
С
       EVENTS CALLED: NONE.
       CALLING EVENTS: NONE.
С
С
       SUBROUTINES CALLED: NONE.
С
       CALLING SUBROUTINES: NONE.
С
С
       AUTHOR: DAVID L. OWEN
С
       HISTORY: N/A.
С
SUBROUTINE ISPO
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
    1, MSTOP, NCLNR, NCRDR, NPR NT, NNRUN, NNSET, NTAPE, SS(100)
    1,SSL(100), TNEXT, TNOW, XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
    1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM,
    1 PCNT, PSTRT, PFIN, PSUM
     REAL AA(24)
     PARAMETER (SPLER=2.)
C*** SET ATRIBUTES FOR INPUT SPOOLER
C**
     ARRIVAL TIME = TNOW
     AA(1) = TNOW
C**
     JOB NAME = 999 (DUMMY NUMBER FOR INPUT SPOOLER)
     AA(2) = 999.
C**
     MEMORY SIZE NEEDED IS 4K
     AA(4) = 4.0
C*** SET JOB TYPE TO INPUT SPOOLER
     AA(11) = SPLER
     CPU TIME USED = TOTAL CPU TIME USED BY INPUT SPOOLER
С
     AA(12) = XX(40)
С
C***
     INITIALIZE OTHER ATRIBUTES TO O
     AA(3) = 0
```

- AA(5) = 0 AA(6) = 0 AA(7) = 0 AA(8) = 0 AA(8) = 0 AA(9) = 0 AA(10) = 0  $D0 \ 11 \ I = 13,24$  AA(I) = 0CONTINUE
- 11 CONTINUE
- C\*\*\* INTITIALIZE ATRIB(21) TO ARRIVED WHEN S/W MON NOT ON AA(21) = 99.

7

C\*\*\* PUT INPUT SPOOLER INTO HOLD QUEUE CALL ENTER(2,AA)

> RETURN END

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: ENTI С EVENT NUMBER: 3 С FUNCTION: ENTERS JOB INTO INPUT QUEUE AND CALLS С EVENT 1 TO SCHEDULE NEXT JOB С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: 1 С CALLING EVENTS: 1 С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE ENT1 COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM C\*\*\* PUT JOB INTO INPUT QUEUE PRINT \*, 'NEW USER JOB IN SYSTEM" C\*\*\*\*\*\* PRINT ATRIBUTES FOR DEBUG PURPOSES PRINT \*, (ATRIB(I), I=1, 20) С CALL ENTER(1,ATRIB) C\*\*\* SCHEDULE THE NEXT JOB ARRIVAL BY CALLING ARVL CALL SCHDL(1,0) RETURN END

C\* \*\*\*\*\*\*\*\*\* С DATE: 5 DEC 1983 С VERSION: 1.0 С С С NAME: CPU **EVENT NUMBER:** 4 С FUNCTION: DETERMINES WHAT TYPE OF JOB IS GETTING С THE CPU TIME SLICE AND CALLS APPROPRIATE С ROUTINE TO HANDLE CPU EXECUTION С С FILES READ: NONE. С FILES WRITTEN: NONE. EVENTS CALLED: NONE. С С CALLING EVENTS: NONE. SUBROUTINES CALLED: USER, SPOOL, OSPOL, SSMON, С С AND JSCHD CALLING SUBROUTINES: NONE. С С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE CPU COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1, SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (USR=1.0, SPL=2.0, OSPL=3.0, SW MNT=4.0, JSC=5.0) C\*\*\* CPU-START-TIME = TNOW XX(4) = TNOWC\*\*\* IF H/W MONITOR = ON, THEN H/W-MON-CPU-CNT = H/W-MON-CPU-CNT + 1 IF (XX(30) .EQ. 1) XX(31) = XX(31) + 1C\*\*\* CHECK TO SEE IF JOB TYPE = USER IF (ATRIB(11) .EQ. USR) CALL USER C\*\*\* CHECK TO SEE IF JOB TYPE = INPUT SPOOLER IF (ATRIB(11) .EQ. SPL) CALL SPOOL C\*\*\* CHECK TO SEE IF JOB TYPE = OUTPUT SPOOLER IF (ATRIB(11) .EQ. OSPL) CALL OSPOL C\*\*\* CHECK TO SEE IF JOB TYPE = S/W MONITOR

```
С
С
       DATE: 5 DEC 1983
С
       VERSION: 1.0
С
С
       NAME: LJSCH
С
       EVENT NUMBER: 5
С
       FUNCTION: SETS ATTRIBUTES FOR JOB SCHEDULTER AND
С
            LOADS JOB INTO HOLD QUEUE
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
С
       EVENTS CALLED: NONE.
С
       CALLING EVENTS: 7, 12, 17, 22
С
       SUBROUTINES CALLED: NONE.
С
       CALLING SUBROUTINES: NONE.
C
С
       AUTHOR: DAVID L. OWEN
С
       HISTORY: N/A.
С
SUBROUTINE LJSCH
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
    1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100)
    1,SSL(100),TNEXT,TNOW,XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
    1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM,
    1 PCNT, PSTRT, PFIN, PSUM
     REAL AAA(24)
C***
     PUT JOB SCHEDULER INTO EXECUTE QUEUE
C**
     FIRST SET UP ATRIBUTES FOR JOB SCHEDULER
     AAA(1) = TNOW
     AAA(2) = 888
     AAA(11) = 5.
     AAA(12) = XX(42)
     DO 20 III=3,10
20
        AAA(III) = 0.
     DO 21 III=13,24
21
       AAA(III) = 0.
C***
     INITIALIZE ATRIB(21) TO ARRIVED WHILE S/W MON NOT ON
     AAA(21) = 99.
C**
     NOW PUT JOB SCHEDULER IN EXECQ
     CALL ENTER (18, AAA)
```

C\*\*\* SET JSCHED IN EXECQ FLAG TO TRUE XX(11) = 1.0

> R ETUR N E ND

1



C\*\* \* С DATE: 5 DEC 1983 С С VERSION: 1.0 С С NAME: LOSPL С **EVENT NUMBER: 6** FUNCTION: SETS ATRIBUTES TO OUPUT SPOOLER AND С С LOADS JOB INTO HOLD QUEUE С FILES READ: NONE. С FILES WRITTEN: NONE. EVENTS CALLED: NONE. С С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С С SUBROUTINE LOSPL COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1, SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM REAL AA(24) PARAMETER (OSPL=3.0,ON=1.0) С C\*\*\* SET OSPOOLERON FLAG TO ON XX(12) = ONC\*\*\* SET ATRIBUTES FOR INPUT SPOOLER С C\*\* ARRIVAL TIME = TNOW AA(1) = TNOWC\*\* JOB NAME = 998 (DUMMY NUMBER FOR OUTPUT SPOOLER) AA(2) = 998.C\*\* MEMORY SIZE NEEDED IS 4K AA(4) = 4.0С C\*\*\* SET JOB TYPE TO INPUT SPOOLER AA(11) = OSPLС C\*\*\* SET CPU TIME USED = TOTAL CPU TIME USED BY OUTPUT SPOOLER AA(12) = XX(41)

С	
C***	INITIALIZE OTHER ATRIBUTES TO O
	AA(3) = 0
	AA(5) = 0
	AA(6) = 0
	AA(7) = 0
	AA(8) = 0
	AA(9) = 0
	AA(10) = 0
	DO 11 I=13,24
	AA(I) = 0
11	CONTINUE
C***	INITIALIZE ATRIB(21) TO ARRIVED WHILE S/W MON NOT ON
	AA(21) = 99.
C***	DITT OUT SPOOLED INTO HOLD OUFUE
C	CALL ENTED (7 AA)
	CALL ENTER(', AA)
	RETURN
	END

C\*\* С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: JCOMP С EVENT NUMBER: 7 FUNCTION: FREES RESOURCES AFTER JOB COMPLETE С С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: 5 С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: USER С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE JCOMP COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (FRE=0.0, BUSY=1.0) C\*\*\* CPU-TIME-USED = CPU-TIME USED + (TNOW - CPU-START-TIME) ATRIB(12) = ATRIB(12) + (TNOW - XX(4))C\*\*\* MOVE JOB TO OUTPUT QUEUE CALL ENTER (10,ATRIB) C\*\*\* FREE CPU, I.E. RELEASE A JOB FROM THE EXECUTE QUEUE CALL FREE(3,1)C\*\*\* FIND MEMORY PARTION USED AND FREE IT DO 5 III=1,XX(3) IF (PART(III,1) .EQ. ATRIB(14)) THEN PART(III,2) = FREGOTO 10 END IF 5 CONTINUE C\*\*\* IF THIS STATEMENT IS REACHED A PARTION WAS NOT FOUND

D-26

PRINT \*, ERROR - UNABLE TO FIND MEMORY PARTION TO FREE

```
PRINT *, JOB ATRIBUTES ARE'
      PRINT *,(ATRIB(I),I=1,20)
      PRINT *, MEMORY PARTIONS ARE
      DO 6 J=1, XX(3)
       PRINT *, PART(J,1), PART(J,2)
6
      CONTINUE
      RETURN
C*** FREE TAPES
10
      NTAPE = 0
      DO 15 III=1,XX(9)
        IF (TAPE(III,3) .EQ. ATRIB(2)) THEN
           NTAPE = NTAPE + 1
           TAPE(III, 2) = FRE
C***
           INCREMENT NUMBER OF AVAILABLE TAPE DRIVES
           XX(10) = XX(10) + 1
        END IF
15
      CONTINUE
C*** CHECK TO SEE THAT YOU FREE THE RIGHT NUMBER OF TAPES
      IF (NTAPE .NE. NINT(ATRIB(6))) THEN
         PRINT *, 'ERROR - FREED ', NTAPE, ' TAPES'
         PRINT *, ~
                          SHOULD BE ',ATRIB(6), ' TAPES'
      END IF
C*** LOAD JOB SCHEDULER INTO EXECQ IF IT ISN'T ALREADY
      IF (XX(11) .NE. 1.) THEN
        CALL LJSCH
        XX(11) = 1.0
      END IF
```

RETURN END

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: DISKS С EVENT NUMBER: 8 С FUNCTION: FREES CPU, FIGURES OUT WHICH DISK TO С GO TO AND PUTS JOB INTO DISK QUEUE С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: SPOOL, OSPOOL, USER С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE DISKS COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON = 1.0) C\*\*\* CALCULATE CPU-TIME-USED AND CPU-TIME-LEFT ATRIB(12) = ATRIB(12) + (TNOW - XX(4))ATRIB(18) = ATRIB(18) - ((TNOW - XX(4)) \* XX(5))C\*\*\* START-I/O = TNOW XX(8) = TNOWC\*\*\* FREE CPU CALL FREE(3,1)  $C^{***}$  NEXT DISK = XX(21) NDISK = NINT(XX(21))C\*\*\* SET JOB ATRIB(19) TO THE DISK USED ATRIB(19) = NDISK + 20C\*\*\* UPDATE NEXT DISK IF (XX(21) .GE. XX(22)) THEN

XX(21) = 1 ELSE XX(21) = XX(21) + 1 END IF C\*\*\* AWAIT DISK CALL ENTER (23 ATRIB)

CALL ENTER(23,ATRIB) RETURN

END

D-29

...
```
С
С
       DATE: 5 DEC 1983
С
       VERSION: 1.0
С
С
       NAME: TAPES
С
       EVENT NUMBER: 9
       FUNCTION: FREES CPU, FIGURES OUT WHICH DISK TO
С
С
               USE, AND PLACES JOB IN SMALLEST CHAN Q
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
       EVENTS CALLED: NONE.
С
С
       CALLING EVENTS: NONE.
С
       SUBROUTINES CALLED: NONE.
С
       CALLING SUBROUTINES: USER
С
С
       AUTHOR: DAVID L. OWEN
С
       HISTORY: N/A.
С
SUBROUTINE TAPES
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
     1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100)
     1,SSL(100),TNEXT,TNOW,XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
     1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM,
     1 PCNT, PSTRT, PFIN, PSUM
     COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10)
     INTEGER CHL, CHNL
     PARAMETER (ON = 1.0)
C***
     CALCULATE CPU-TIME-USED AND CPU-TIME-LEFT
     ATRIB(12) = ATRIB(12) + (TNOW - XX(4))
     ATRIB(18) = ATRIB(18) - ((TNOW - XX(4)) * XX(5))
C^{***} START-I/O = TNOW
     XX(8) = TNOW
C*** FREE CPU BY RELEASING A JOB FROM THE EXECUTE QUEUE
     CALL FREE(3,1)
C***
     FIND NEXT TAPE DRIVE ASSIGNED TO THAT JOB
C**
     MAKE SURE YOU DON'T GO INTO INFINITE LOOP
     DO 5 I=1, XX(9)
```

```
IF (XX(24) .GE. XX(9)) THEN
          XX(24) = 1
         ELSE
          XX(24) = XX(24) + 1.0
        END IF
        NTAPE = NINT(XX(24))
        IF ( TAPE(NTAPE, 3) .EQ. ATRIB(2) ) GO TO 10
5
      CONTINUE
     IF THIS STATEMENT WAS REACHED THEN THEIR ARE NO TAPES ASSIGNED
C***
C***
     TO THE JOB
      PRINT *, 'ERROR: TAPES NOT ASSIGNED TO THIS JOB'
      PRINT *,(ATRIB(K),K=1,21)
      RETURN
C*** SET JOB ATRIB(19) TO THE TAPE USED
10
      ATRIB(19) = NTAPE + 10
C*** FIND FIRST CHANNEL TAPE IS CONNECTED TO
      DO 1 I=1, XX(23)
         IF (TAP(NTAPE, I) .EQ. 1.0) THEN
            CHNL = I
            CSIZE = NNQ(30 + CHNL)
            ATRIB(20) = CHNL
            GO TO 2
         END IF
      CONTINUE
1
C***
      IF THIS STATEMENT IS REACHED THEN THE TAPE IS NOT CONNECTED
C***
      TO ANY CHANNEL
      PRINT *, "ERROR: TAPE NOT CONNECTED TO ANY CHANNEL"
      PRINT *, TAPE = ', NTAPE
      PRINT *, TAPE CHANNELS ARE'
      PRINT *,(TAP(NTAPE, I), I=1, XX(23))
      RETURN
C*** FIND THE SMALLEST CHANNEL QUEUE WHICH THE TAPE CAN USE
2
      ICHNL = CHNL
      DO 15 CHL=ICHNL,XX(23)
         IF (TAP(NTAPE, CHL) .EQ. ON) THEN
            IF (NNQ(30+CHL) .LT. CSIZE) THEN
               CHNL = CHL
               CSIZE = NNQ(30+CHL)
               ATRIB(20) = CHNL
            END IF
         END IF
```

## 15 CONTINUE

## C\*\*\* AWAIT CHANNEL CALL ENTER(25,ATRIB) RETURN

END

i i i

. . . .

THE REAL PROPERTY OF



С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: TIMEO С EVENT NUMBER: 10 С FUNCTION: FREES CPU AND PLACES JOB BACK INTO С EXECUTE QUEUE С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: USER С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С \*\*\*\*\*\*\*\* \*\*\*\*\* SUBROUTINE TIMEO COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1, SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM C\*\*\* CALCULATE CPU-TIME-USED AND CPU-TIME-LEFT ATRIB(12) = ATRIB(12) + (TNOW - XX(4))ATRIB(18) = ATRIB(18) - ((TNOW - XX(4)) \* XX(5))C\*\*\* FREE CPU BY RELEASING A JOB FROM THE EXECUTE QUEUE CALL FREE(3,1)C\*\*\* PLACE JOB BACK INTO EXECUTE QUEUE CALL ENTER(4,ATRIB) RETURN END

С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: RCARD С С **EVENT NUMBER: 11** С FUNCTION: FREES CPU AND READS CARDS INTO BUFFER С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 14** С CALLING EVENTS: 16 SUBROUTINES CALLED: NONE. С С CALLING SUBROUTINES: NONE. С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С С SUBROUTINE RCARD COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON = 1.0) C\*\*\* UPDATE CPU TIME USED ATRIB(12) = ATRIB(12) + (TNOW - XX(4)) \* XX(5)C\*\*\* UPDATE NUMBER OF CARDS LEFT TO READ IF (ATRIB(7) .LE. XX(20)) THEN ATRIB(7) = 0.ELSE ATRIB(7) = ATRIB(7) - XX(20)END IF C\*\*\* FREE CPU CALL FREE(3,1)C\*\*\* IF H/W MON IS ON THEN UPDATE READER AND CHANNEL DATA IF  $(XX(30) \cdot EQ \cdot ON)$  THEN RCNT = RCNT + 1IF (RSTRT .NE. -1.0) THEN IF (RFIN .LT. TNOW) THEN RSUM = RSUM + RFIN - RSTRT

```
RSTRT = TNOW
           END IF
          ELSE
           RSTRT = TNOW
         END IF
         RFIN2 = ((XX(20) / XX(18)) * 60) + TNOW
         IF (RFIN2 .GE. RFIN) THEN
            RFIN = RFIN2
         END IF
         NCHL = NINT(ATRIB(20))
         IF (CSTRT(NCHL) .NE. -1.0) THEN
           IF (CFIN(NCHL) .LT. TNOW) THEN
             CSUM(NCHL) = CSUM(NCHL) + CFIN(NCHL) - CSTRT(NCHL)
             CSTRT(NCHL) = TNOW
            ELSE
             CCNT(NCHL) = CCNT(NCHL) + 1
           END IF
          ELSE
           CSTRT(NCHL) = TNOW
           CCNT(NCHL) = CCNT(NCHL) + 1
         END IF
         IF (RFIN2 .GT. CFIN(NCHL)) THEN
            CFIN(NCHL) = RFIN2
         END IF
     END IF
C***
     SCHEDULE COMPLETION OF LOADING BUFFER
      CALL SCHDL (14,(XX(20) / XX(18)) * 60,ATRIB)
```

RETURN END



С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: DCOMP С **EVENT NUMBER: 12** С FUNCTION: FREES CHANNEL AND DISK AFTER DISK С TRANSFER. С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 5** С CALLING EVENTS: 16 С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: SPOOL С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE DCOMP COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON = 1.0, USR = 1.0, SPL=2.0, OSPL=3.0, FRE=0.0) C\*\*\* FREE CHANNEL OF THE PORTION TEAT THE JOB WAS USING NCHL = 30 + ATRIB(20)ITEMP = NINT(USERF(1))CALL FREE(NCHL, ITEMP) C\*\*\* FREE DISK NDSK = ATRIB(19)CALL FREE(NDSK,1) C\*\*\* UPDATE 1/0 TIME USED ATRIB(13) = ATRIB(13) + (TNOW - XX(8))C\*\*\* IF JOB TYPE = USER IF (ATRIB(11) .EQ. USR) THEN C\* UPDATE TIME OF LAST I/O (CPU-TIM-REQ - CPU-TIM-LEFT) ATRIB(16) = ATRIB(3) - ATRIB(18)C\* PUT JOB BACK INTO EXECUTE QUEUE

CALL ENTER(3,ATRIB) RETURN END IF C\*\*\* IF JOB TYPE IS INPUT SPOOLER IF (ATRIB(11) .EQ. SPL) THEN C\*\* CHECK FOR COMPLETION OF SPOOLING IF (ATRIB(7) .EQ. 0.) THEN C\* SPOOL STATUS = 0 XX(15) = 0.0C\*\*\*\*\*\*\* NEED TO PUT CPU TIME FOR SPOOLER IN HERE C\*\* FREE JOB FROM INPUT QUEUE CALL FREE(1,1) IF (NNQ(1) .NE. 0) THEN C\*\* PUT SPOOLER BACK INTO EXECUTE QUEUE CALL ENTER(3,ATRIB) ELSE C\*\* SPOOLERON = 0XX(1) = 0C\*\* FIND MEMORY PARTION USED AND FREE IT DO 5 III=1,XX(3)IF (PART(III,1) .EQ. ATRIB(14)) THEN PART(III,2) = FREGOTO 10 END IF 5 CONTINUE C\*\*\* IF THIS STMT IS REACHED A PARTION WAS NOT FOUND PRINT \*, ERROR - UNABLE TO FIND MEMORY PARTION TO FREE PRINT \*, JOB ATRIBUTES ARE PRINT \*, (ATRIB(I), I=1, 18) PRINT \*, MEMORY PARTIONS ARE DO 6 J=1, XX(3)PRINT \*, PART(J,1), PART(J,2) 6 CONTINUE RETURN 10 CONTINUE C\*\*\* PUT JOB SCHEDULER INTO EXECQ (IF NOT ALREADY THERE) IF (XX(11) .NE. ON) THEN XX(11) = ONCALL SCHDL(5,0.,ATRIB) END IF END IF ELSE C\*\* SPOOL STATUS = 2 (READY TO READ MORE CARDS) XX(15) = 2.0C\* PUT SPOOLER BACK INTO EXEC QUEUE CALL ENTER (3, ATRIB)

	END IF RETURN END IF
C***	IF JOB TYPE IS OUTPUT SPOOLER
	IF (ATRIB(11) .EO. OSPL) THEN
C**	OSPOOL STATUS = READY TO PRINT BUFFER XX(16) = 1.0
C**	PUT OUTPUT SPOOLER BACK INTO EXECQ CALL ENTER(3,ATRIB) RETURN
	END IF
C***	IF JOB TYPE IS NOT USER OR SPOOLER OR OUTPUT SPOOLER THEN
C	PRINT * "FRROR ILIFCAL TOR TYPE IN DOOMP"
	PRINT *, (ATRIB(I), I=1,21)
	RETURN

END

K

С С DATE: 5 DEC 1983 С VERSION: 1.0 С NAME: TCOMP С С EVENT NUMBER: 13 FUNCTION: FREES CHANNEL AFTER TAPE IO COMPLETION С С FILES READ: NONE. С FILES WRITTEN: NONE. EVENTS CALLED: NONE. С С CALLING EVENTS: 16 С SUBROUTINES CALLED: USERF С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С \*\*\*\*\*\*\*  $C^{*}$ SUBROUTINE TCOMP COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10) C\* UPDATE TIME OF LAST TAPE I/O (CPU-TIM-REQ - CPU-TIM-LEFT) ATRIB(17) = ATRIB(3) - ATRIB(18)FREE CHANNEL OF THE PORTION THAT THE JOB WAS USING C\*\*\* NCHL = 30 + ATRIB(20)ITEMP = NINT(USERF(1))CALL FREE(NCHL, ITEMP) C\*\*\* UPDATE I/O TIME USED ATRIB(13) = ATRIB(13) + (TNOW - XX(8)) $C^{\star}$ PUT JOB BACK INTO EXECUTE QUEUE CALL ENTER (3, ATRIB) RETURN

END

```
С
       DATE: 5 DEC 1983
С
С
       VERSION: 1.0
С
С
       NAME: CHANL
С
       EVENT NUMBER: 15
       FUNCTION: FINDS SMALLET CHANNEL QUEUE THAT IS
С
С
              CONNECT TO DISK, CARD READER, OR PRINTER
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
       EVENTS CALLED: NONE.
С
С
       CALLING EVENTS: 11
С
       SUBROUTINES CALLED: NONE.
C
       CALLING SUBROUTINES: OSPOL, SPOOL
С
C
       AUTHOR: DAVID L. OWEN
       HISTORY: N/A.
С
С
SUBROUTINE CHANL
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
    1, MS TOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100)
    1,SSL(100), TNEXT, TNOW, XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
     1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM,
    1 PCNT, PSTRT, PFIN, PSUM
     COMMON/UCOM4/CARD(6), PRNTR(6)
     INTEGER CHL, CHNL
     PARAMETER (ON = 1.0, USR=1.0, SPLR=40.0, OSPLR=41.0)
C*** IF IT IS A DISK NEEDING THE CHANNEL
     IF ((ATRIB(19) .GT. 20) .AND. (ATRIB(19) .LE. 30)) THEN
     NDISK = NINT(ATRIB(19) - 20)
C*** FIND FIRST CHANNEL DISK IS CONNECTED TO
     DO 1 I=1,XX(23)
        IF (DISK(NDISK, I) .EQ. 1.0) THEN
           CHNL = I
           CSIZE = NNQ(30 + CHNL)
           ATRIB(20) = CHNL
           GO TO 2
        END IF
     CONTINUE
1
C*** IF THIS STATEMENT IS REACHED THEN THE DISK IS NOT CONNECTED
```

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: FULLB С EVENT NUMBER: 14 С FUNCTION: FREES CHANNEL AFTER BUFFER HAS BEEN С LOADED BY CARD READER С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: 11 С SUBROUTINES CALLED: USERF С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE FULLB COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM COMMON/UCOM4/CARD(6), PRNTR(6) PARAMETER (ON=1.0, OFF=0.0, NEW=0.0, BFULL=1.0, BEMPT=2.0) C\*\*\* SET SPOOL-STATUS TO BUFFER FULL XX(15) = BFULLC\*\*\* FREE CHANNEL OF THE PORTION THAT THE CARD READER WAS USING NCHL = 30 + ATRIB(20)ITEMP = USERF(1)CALL FREE (NCHL, ITEMP) C\*\*\* PUT SPOOLER BACK INTO EXECUTE QUEUE CALL ENTER(3,ATRIB) RETURN END

```
C*** TO ANY CHANNEL
      PRINT *, ERROR: DISK NOT CONNECTED TO ANY CHANNEL"
      PRINT *, DISK = ', NDISK
PRINT *, DISK CHANNELS ARE'
      PRINT *,(DISK(NDISK,I),I=1,XX(23))
      RETURN
C*** FIND THE SMALLEST CHANNEL QUEUE WHICH THE DISK CAN USE
      ICHNL = CHNL
2
      DO 10 CHL=ICHNL, XX(23)
         IF (DISK(NDISK, CHL) .EQ. ON) THEN
            IF (NNQ(30+CHL) .LT. CSIZE) THEN
               CHNL = CHL
               CSIZE = NNQ(30+CHL)
               ATRIB(20) = CHNL
            END IF
         END IF
      CONTINUE
10
C*** AWAIT CHANNEL
      CALL ENTER(25,ATRIB)
      RETURN
      END IF
C*** IF IT IS THE CARD READER THAT NEEDS THE CHANNEL
      IF (ATRIB(19) .EQ. SPLR) THEN
      ATRIB(19) = SPLR
C***
     FIND FIRST CHANNEL CARD READER IS CONNECTED TO
      DO 4 I=1, XX(23)
         IF (CARD(I) .EQ. 1.0) THEN
            CHNL = I
            CSIZE = NNQ(30 + CHNL)
            ATRIB(20) = CHNL
            GO TO 5
         END IF
4
      CONTINUE
C*** IF THIS STATEMENT IS REACHED THEN THE CARD READER IS NOT
C***
      CONNECTED TO ANY CHANNEL
      PRINT *, 'ERROR: CARD READER NOT CONNECTED TO ANY CHANNEL'
      PRINT *, CARD READER CHANNELS ARE
      PRINT *,(CARD(I), I=1, XX(23))
      RETURN
C*** FIND THE SMALLEST CHANNEL QUEUE WHICH THE CARD READER CAN USE
5
      ICHNL = CHNL
      DO 6 CHL=ICHNL,XX(23)
         IF (CARD(CHL) .EQ. ON) THEN
            IF (NNQ(30+CHL) .LT. CSIZE) THEN
```

```
DFIN2 = DISK(NDSK, 6) + TNOW
         IF (DFIN2 .GE. DFIN(NDSK)) THEN
            DFIN(NDSK) = DFIN2
         END IF
         NCHL = NINT(ATRIB(20))
         IF (CSTRT(NCHL) .NE. -1.0) THEN
           IF (CFIN(NCHL) .LT. TNOW) THEN
             CSUM(NCHL) = CSUM(NCHL) + CFIN(NCHL) - CSTRT(NCHL)
             CSTRT(NCHL) = TNOW
            ELSE
             CCNT(NCHL) = CCNT(NCHL) + 1
           END IF
          ELSE
           CSTRT(NCHL) = TNOW
           CCNT(NCHL) = CCNT(NCHL) + 1
         END IF
         IF (DFIN2 .GT. CFIN(NCHL)) THEN
            CFIN(NCHL) = DFIN2
         END IF
      END IF
     SCHEDULE COMPLETION OF DISK TRANSFER
C***
      CALL SCHDL(12, DISK(NDSK, 6), ATRIB)
      RETURN
      END IF
C*** IF IT IS A TAPE TRANSFER
      IF ((ATRIB(19) .LE. 20.0) .AND. (ATRIB(19) .GT. 10.)) THEN
      NTAPE = NINT(ATRIB(19) - 10)
C*** IF H/W MON IS ON THEN UPDATE TAPE AND CHANNEL DATA
      IF (XX(30) .EQ. ON) THEN
         TCNT(NTAPE) = TCNT(NTAPE) + 1
         IF (TSTRT(NTAPE) .NE. -1.0) THEN
           IF (TFIN(NTAPE) .LT. TNOW) THEN
             TSUM(NTAPE) = TSUM(NTAPE) + TFIN(NTAPE) - TSTRT(NTAPE)
             TSTRT(NTAPE) = TNOW
           END IF
          ELSE
           TSTRT(NTAPE) = TNOW
         END IF
         TFIN2 = TAP(NTAPE, 6) + TNOW
         IF (TFIN2 .GE. TFIN(NTAPE)) THEN
            TFIN(NTAPE) = TFIN2
```

```
D-46
```

```
END IF
         NCHL = NINT(ATRIB(20))
         IF (CSTRT(NCHL) .NE. -1.0) THEN
IF (CFIN(NCHL) .LT. TNOW) THEN
             CSUM(NCHL) = CSUM(NCHL) + CFIN(NCHL) - CSTRT(NCHL)
              CSTRT(NCHL) = TNOW
            ELSE
              CCNT(NCHL) = CCNT(NCHL) + 1
           END IF
          ELSE
           CSTRT(NCHL) = TNOW
           CCNT(NCHL) = CCNT(NCHL) + 1
         END IF
         IF (TFIN2 .GT. CFIN(NCHL)) THEN
            CFIN(NCHL) = TFIN2
         END IF
      END IF
C***
      SCHEDULE COMPLETION OF TAPE TRANSFER
      CALL SCHDL(13, TAP(NTAPE, 6), ATRIB)
      RETURN
      END IF
C*** IF JOB NEEDS TO READ CARDS THEN CALL RCARD
      IF (ATRIB(19) .EQ. SPLR) THEN
         CALL SCHDL(11,0.0,ATRIB)
         RETURN
      END IF
C***
      IF JOB NEEDS TO PRINT CALL PBUFF
      IF (ATRIB(19) .EQ. OSPLR) THEN
         CALL SCHDL(24,0.0,ATRIB)
         RETURN
      END IF
```

END

END IF

END

-

- í

t + + + -



```
CHNL = CHL
               CSIZE = NNQ(30+CHL)
               ATRIB(20) = CHNL
            END IF
         END IF
6
      CONTINUE
C*** AWAIT CHANNEL
      CALL ENTER (25, ATRIB)
      RETURN
      END IF
C*** IF IT IS THE LINE PRINTER THAT NEEDS THE CHANNEL
      IF (ATRIB(19) .EQ. OSPLR) THEN
      ATRIB(19) = OSPLR
C*** FIND FIRST CHANNEL LINE PRINTER IS CONNECTED TO
      DO 20 I=1, XX(23)
         IF (PRNTR(I) .EQ. 1.0) THEN
            CHNL = I
            CSIZE = NNQ(30 + CHNL)
            ATRIB(20) = CHNL
            GO TO 25
         END IF
20
      CONTINUE
C*** IF THIS STATEMENT IS REACHED THEN THE LINE PRINTER IS NOT
C*** CONNECTED TO ANY CHANNEL
      PRINT *, FRROR: LINE PRINTER NOT CONNECTED TO ANY CHANNEL
      PRINT *, LINE PRINTER CHANNELS ARE
      PRINT *,(PRNTR(I),I=1,XX(23))
      RETURN
C*** FIND THE SMALLEST CHANNEL QUEUE WHICH THE LINE PRINTER CAN USE
25
      ICHNL = CHNL
      DO 30 CHL=ICHNL, XX(23)
         IF (PRNTR(CHL) .EQ. ON) THEN
            IF (NNQ(30+CHL) .LT. CSIZE) THEN
               CHNL = CHL
               CSIZE = NNQ(30+CHL)
               ATRIB(20) = CHNL
            END IF
         END IF
30
      CONTINUE
C*** AWAIT CHANNEL
      CALL ENTER(25,ATRIB)
      RETURN
```

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: FPRNT (FINISHED PRINTING) С **EVENT NUMBER: 17** С FUNCTION: FREES CHANNEL AFTER PRINTING BUFFER. IF THERE IS NO MORE PRINTING TO BE DONE С C THEN THE OUPUT SPOOLER IS RELEASED FROM С THE SYSTEM , THE PARTITION IS FREED AND THE JSCHD IS LOADED. С С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 5** С CALLING EVENTS: 24 С SUBROUTINES CALLED: USERF С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE FPRNT COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON = 1.0, FRE=0.0) C\*\*\* FREE CHANNEL OF THE PORTION THAN THE LINE PRINTER WAS USING NCHL = 30 + ATRIB(20)ITEMP = USERF(1)CALL FREE(NCHL, ITEMP) C\*\* CHECK FOR COMPLETION OF OUTPUT SPOOLING IF (ATRIB(8) .EQ. 0.) THEN C\* OSPOOL STATUS = 0XX(16) = 0.0C\*\*\*\*\*\*\* NEED TO PUT CPU TIME FOR SPOOLER IN HERE FREE JOB FROM OUTPUT QUEUE C\*\* CALL FREE(4,1) IF (NNQ(4) .NE. 0) THEN PUT OUTPUT SPOOLER BACK INTO EXECUTE QUEUE C\*\* CALL ENTER(3,ATRIB)

	ELSE
C**	OSPOOLERON = O
	XX(12) = 0.
C**	FIND MEMORY PARTION USED AND FREE IT
	DO 5 III=1,XX(3)
	IF (PART(III,1) .EQ. ATRIB(14)) THEN
	PART(III,2) = FRE
	GOTO 10
	END IF
5	CONTINUE
C***	IF THIS STMT IS REACHED A PARTION WAS NOT FOUND
	PRINT *, ERROR - UNABLE TO FIND MEMORY PARTION TO FREE
	PRINT *, JOB ATRIBUTES ARE
	PRINT *, (ATRIB(I), I=1, 18)
	PRINT *, MEMORY PARTIONS ARE
	DO 6 J=1,XX(3)
	<pre>PRINT *, PART(J,1), PART(J,2)</pre>
6	CONTINUE
	RETURN
10	CONTINUE
C***	PUT JOB SCHEDULER INTO EXECQ (IF NOT ALREADY THERE)
	IF (XX(11) .NE. ON) THEN
	XX(11) = ON
	CALL SCHDL(5,0.,ATRIB)
	END IF
	END IF
	ELSE
C**	SPOOL STATUS = 2 (READY TO READ MORE CARDS)
-	XX(10) = 2.0
C#	PUT SPOOLER BACK INTO EXEC QUEUE
	CALL ENTER(J,ATRIB)
	KETUKN

END

The second second

- A-stemators n

\* C\*\* С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: JSCOM (JOB SCHED COMPLETION) С EVENT NUMBER: 18 С FUNCTION: FREES CPU. SINCE JOB SCHEDULER IS С ALSO CORE RESIDENT THERE IS NO MEMORY С PARTITION TO FREE. С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: JSCHD С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE JSCOM COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100)

- C\*\*\* FREE CPU BY RELEASING JOB FROM EXECUTE QUEUE CALL FREE(3,1)
- C\*\*\* UPDATE CPU TIME USED BY JOB SCHEDULER XX(42) = XX(42) + (TNOW - XX(4))

RETURN END

D~50

```
С
С
       DATE: 5 DEC 1983
С
       VERSION: 1.0
С
С
       NAME: DTRAN
С
       EVENT NUMBER: 16
       FUNCTION: PERFORMS A DISK OR TAPE TRANSFER IF
С
С
               NECESSARY. IF A CARD IS TO BE READ OR
               A LINE TO PRINTED THEN THE APPROPRIATE
C
C
               SUBROUTINE IS CALLED
C
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
С
       EVENTS CALLED: 11, 12, 13, 24
С
       CALLING EVENTS: NONE.
С
       SUBROUTINES CALLED: NONE.
С
       CALLING SUBROUTINES: NONE.
С
С
       AUTHOR: DAVID L. OWEN
С
       HISTORY: N/A.
С
SUBROUTINE DTRAN
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
    1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100)
    1,SSL(100), TNEXT, TNOW, XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
    1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM,
    1 PCNT, PSTRT, PFIN, PSUM
     COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10)
     COMMON/UCOM4/CARD(6), PRNTR(6)
     PARAMETER (ON = 1.0, SPLR=40.0, OSPLR=41.0)
C*** IF IT IS A DISK TRANSFER
     IF ((ATRIB(19) .GE. 21.) .AND. (ATRIB(19) .LE. 30.)) THEN
     NDSK = NINT(ATRIB(19) - 20)
C*** IF H/W MON IS ON THEN UPDATE DISK AND CHANNEL DATA
     IF (XX(30) .EQ. ON) THEN
        DCNT(NDSK) = DCNT(NDSK) + 1
        IF (DSTRT(NDSK) .NE. -1.0) THEN
          IF (DFIN(NDSK) .LT. TNOW) THEN
            DSUM(NDSK) = DSUM(NDSK) + DFIN(NDSK) - DSTRT(NDSK)
            DSTRT(NDSK) = TNOW
          END IF
         ELSE
          DSTRT(NDSK) = TNOW
        END IF
```

\*\*\*\*\*\*\*\*\*\*\*\*\*\* C#: С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: ACTLG (ACTIVITY LOG) С EVENT NUMBER: 19 С FUNCTION: WRITE ACOUNTING DATA TO THE ACCOUNTING С LOG С FILES READ: NONE. С FILES WRITTEN: ACTLOG С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С C \*\*\*\*\*\*\*\* SUBROUTINE ACTLG COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP 1, NCLNR, NCRDR, NPR NT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM C\*\*\* SET DEPARTURE TIME TO TNOW ATRIB(15) = TNOWС WRITE ARRIVAL TIME, JOB NAME, CPU TIME REQUIRED, MEMORY, PRIORITY, С ALLOCATABLE DEVIECES NEEDED, CARDS, LINES, DISK BLOCKS, ALLOCATABLE DEVICE BLOCKS, JOB TYPE, CPU TIME USED, I/O TIME USED, MEMORY SIZE С USED, AND DEPARTURE TIME TO THE ACOUNTING DATA FILE С PRINT \*,(ATRIB(I),I=1,15) WRITE (UNIT=12, FMT=2, ERR=990)(ATRIB(I), I=1, 15) 2 FORMAT(1X, F15.4, 1X, F10.0, 1X, F5.0, 1X, F5.0, 1X, F5.0, 1X, F5.0, 1X, F6.0, 1X, F6.0, 1X, F6.0, 1X, F6.0, 1X, F2.0, 1X, F10.3, 1X, 1 1 F10.3,1X,F6.1,1X,F15.4) RETURN 990 PRINT \*, ERROR IN WRITING TO ACCOUNTING DATA FILE RETURN

END

Construction of the second

1

E.

\*\*\*\*\*\*\*\*\*\*\*\* C\*\*\*\* С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: HW MON (HARDWARE MONITOR) С EVENT NUMBER: 20 FUNCTION: WRITE HARDWARE MONITOR DATA. RESECTS С COUNTERS AND TIMERS. SCHEDULES NEXT TIME С С TO RECORD H/W MONITOR DATA С FILES READ: NONE. С FILES WRITTEN: HRDMON С **EVENTS CALLED: 20** С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE HWMON COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA, MSTOP 1, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10) COMMON/UCOM6/IPROBE(5) DIMENSION TIMER(2), ICNTER(3) PARAMETER(OFF=0.0, ON=1.0, ST=-1.0) C\*\*\* CHECK TO SEE IF IT TIME TO START H/W MONITOR IF (XX(33) .EQ. TNOW) THEN PRINT \*, START OF H/W MON' XX(30) = ONC\*\* INITIALIZE START TIMES, STOP TIMES, COUNTS AND SUMS RSTRT = STPSTRT = ST XX(36) = STXX(37) = 0.0RSUM = 0.0PSUM = 0.0XX(31) = 0.0RCNT = 0.0PCNT = 0.0XX(37) = 0.0

RFIN = 0.0PFIN = 0.0DO 60 I=1,10 DSTRT(I) = STTSTRT(I) = STDSUM(I) = 0.0TSUM(I) = 0.0DFIN(I) = 0.0TFIN(I) = 0.0DCNT(I) = 0.0TCNT(I) = 0.060 CONTINUE DO 61 I=1,5 CSTRT(I) = STCSUM(I) = 0.0CCNT(I) = 0.0CFIN(I) = 0.061 CONTINUE CALL SCHDL(20,XX(35),ATRIB) RETURN END IF C\*\*\* IF H/W MON IS ON THEN IF (XX(30) .EQ. ON) THEN C\*\*\* CALCULATE CPU TIME IF (XX(36) .NE. -1.0) THEN IF (XX(37) .GT. TNOW) THEN XX(32) = XX(32) + TNOW - XX(36)XX(36) = TNOWELSE XX(32) = XX(32) + XX(37) - XX(36)XX(36) = -1.0END IF END IF C\*\*\* CALCULATE CARD READER TIME IF (RSTRT .NE. -1.0) THEN IF (RFIN .GT. TNOW) THEN RSUM = RSUM + TNOW - RSTRT RSTRT = TNOW ELSE RSUM = RSUM + RFIN - RSTRT RSTRT = -1.0END IF END IF C\*\*\* CALCULATE LINE PRINTER TIME IF (PSTRT .NE. -1.0) THEN IF (PFIN .GT. TNOW) THEN

```
PSUM = PSUM + TNOW - PSTRT
           PSTRT = TNOW
         ELSE
           PSUM = PSUM + PFIN - PSTRT
           PSTRT = -1.0
         END IF
      END IF
C*** CALCULATE DISK TIME
      DO 50 I=1,10
      IF (DSTRT(I) .NE. -1.0) THEN
         IF (DFIN(I) .GT. TNOW) THEN
           DSUM(I) = DSUM(I) + TNOW - DSTRT(I)
           DSTRT(I) = TNOW
         ELSE
           DSUM(I) = DSUM(I) + DFIN(I) - DSTRT(I)
           DSTRT(I) = -1.0
         END IF
      END IF
50
      CONTINUE
C*** CALCULATE TAPE TIME
      DO 51 I=1,10
      IF (TSTRT(I) .NE. -1.0) THEN
         IF (TFIN(I) .GT. TNOW) THEN
           TSUM(I) = TSUM(I) + TNOW - TSTRT(I)
           TSTRT(I) = TNOW
         ELSE
           TSUM(I) = TSUM(I) + TFIN(I) - TSTRT(I)
           TSTRT(I) = -1.0
         END IF
      END IF
51
      CONTINUE
C***
      CALCULATE CHANNEL TIME
      DO 52 I=1,5
      IF (CSTRT(I) .NE. -1.0) THEN
         IF (CFIN(I) .GT. TNOW) THEN
           CSUM(I) = CSUM(I) + TNOW - CSTRT(I)
           CSTRT(I) = TNOW
         ELSE
           CSUM(I) = CSUM(I) + CFIN(I) - CSTRT(I)
           CSTRT(I) = -1.0
         END IF
      END IF
52
      CONTINUE
C***
     GET TIMER VALUES OF TIMER PROBES
      DO 700 I=1,2
        IF (IPROBE(I) .EQ. 1) THEN
           TIMER(I) = XX(31)
           GOTO 700
```

```
D-55
```

```
END IF
        IF (IPROBE(I) .EQ. 2) THEN
           TIMER(I) = RSUM
           GOTO 700
        END IF
        IF
           ((IPROBE(I) .GE. 11) .AND. (IPROBE(I) .LE. 20)) THEN
           TIMER(I) = TSUM(IPROBE(I)-10)
           GOTO 700
        END IF
        IF ((IPROBE(I) .GE. 21) .AND. (IPROBE(I) .LE. 30)) THEN
           TIMER(I) = DSUM(IPROBE(I)-20)
           GOTO 700
        END IF
        IF ((IPROBE(I) .GE. 31) .AND. (IPROBE(I) .LE. 35)) THEN
           TIMER(I) = CSUM(IPROBE(I)-30)
           GOTO 700
        END IF
C***
        IF NONE OF THE ABOVE THEN PROBE NOT CONNECTED
        TIMER(I) = 0.0
700
      CONTINUE
C***
      GET COUNTER VALUES OF COUNTER PROBES
      DO 710 I=3,5
        K = I - 2
        IF (IPROBE(I) .EQ. 1) THEN
           ICNTER(K) = NINT(XX(32))
           GOTO 710
        END IF
        IF
           (IPROBE(I) .EQ. 2) THEN
           ICNTER(K) = NINT(RCNT)
           GOTO 710
        END IF
           ((IPROBE(I) .GE. 11) .AND. (IPROBE(I) .LE. 20)) THEN
        IF
           ICNTER(K) = NINT(TCNT(IPROBE(I)-10))
           GOTO 710
        END IF
        IF ((IPROBE(I) .GE. 21) .AND. (IPROBE(I) .LE. 30)) THEN
           ICNTER(K) = NINT(DCNT(IPROBE(I)-20))
           GOTO 710
        END IF
        IF ((IPROBE(I) .GE. 31) .AND. (IPROBE(I) .LE. 35)) THEN
           ICNTER(K) = NINT(CCNT(IPROBE(I)-30))
           GOTO 710
        END IF
C***
        IF NONE OF THE ABOVE THEN PROBE NOT CONNECTED
        ICNTER(K) = 0.0
710
      CONTINUE
```

```
D-56
```

```
C*** WRITER H/W MONITOR DATA TO DISK
      WRITE(UNIT=13, FMT=3, IOSTAT=IOS, ERR=990) [NOW, TIMER(1), TIMER(2),
     1
            (ICNTER(I), I=1, 3)
3
      FORMAT(1X, F15.4, 1X, F8.3, 1X, F8.3, 1X, I5, 1X, I5, 1X, I5)
C*** RESET CPU COUNT, AND CPU SUM
      IF (XX(37) .LE. TNOW) THEN
        XX(31) = 0.
      ELSE
         XX(31) = 1.0
      END IF
      XX(32) = 0.
C*** RESET READER COUNT, AND READER SUM
      IF (RFIN .LE. TNOW) THEN
        RCNT = 0.
      ELSE
         RCNT = 1.0
      END IF
      RSUM = 0.
C*** RESET PRINTER COUNT, AND PRINTER SUM
      IF (PFIN .LE. TNOW) THEN
        PCNT = 0.
      ELSE
            PCNT = 1.0
      END IF
      PSUM = 0.
C*** RESET TAPE COUNT, AND TAPE SUM
      DO 12 I=1,10
         IF (TFIN(I) .LE. TNOW) THEN
           TCNT(I) = 0.
         ELSE
            TCNT(I) = 1.0
         END IF
         TSUM(I) = 0.
12
      CONTINUE
C*** RESET DISK COUNT, AND DISK SUM
      DO 9 I=1,10
         IF (DFIN(1) .LE. TNOW) THEN
            DCNT(I) = 0.
          ELSE
            DCNT(I) = 1.0
         END IF
         DSUM(I) = 0.
9
      CONTINUE
C*** RESET CHANNEL COUNT AND CHANNEL TIME
```

```
DO 10 I=1,5
         IF (CFIN(I) .LE. TNOW) THEN
           CCNT(I) = 0.
          ELSE
           CCNT(I) = 1.0
         END IF
         CSUM(I) = 0.
10
       CONTINUE
C*** CHECK TO SEE IF IT TIME TO TURN OFF THE H/W MON
      IF (XX(34) .LE. TNOW) THEN
PRINT *, TURNING OFF H/W MON NOW *
          XX(30) = OFF
          RETURN
        ELSE
C***
          SCEDULE NEXT RECORDING OF H/W MONITOR DATA
          CALL SCHDL(20,XX(35),ATRIB)
          RETURN
      END IF
C*** END OF IF H/W MON IS ON
      END IF
      RETURN
      PRINT *, ERROR IN WRITING TO H/W MON'
PRINT *, IOSTAT = ', IOS
990
```

END

RETURN

С DATE: 5 DEC 1983 С С VERSION: 1.0 С С NAME: LSMON (LOAD S/W MONITOR) EVENT NUMBER: 21 С FUNCTION: LOADS S/W MONITOR INTO HOLD QUEUE С С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: INTLC С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С C\* SUBROUTINE LSMON COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), FIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM REAL AA(24) PARAMETER (SWMN=4.0, ON=1.0) C\*\*\* SET ATRIBUTES FOR S/W MONITOR C\*\* ARRIVAL TIME = TNOW AA(1) = TNOWC\*\* JOB NAME = 997 (DUMMY NUMBER FOR S/W MONITOR) AA(2) = 997.C\*\* MEMORY SIZE NEEDED IS 4K AA(4) = 4.0C\*\*\* SET JOB TYPE TO INPUT SPOOLER AA(11) = SWMNC\*\*\* INITIALIZE OTHER ATRIBUTES TO O AA(3) = 0AA(5) = 0 AA(6) = 0AA(7) = 0AA(8) = 0AA(9) = 0AA(10) = 0DO 11 I=12,24 AA(I) = 0

## 11 CONTINUE

C\*\*\* INITIALIZE ATRIB(21) TO ARRIVED WHILE S/W MON NOT ON AA(21) = 99.

C\*\*\* PUT S/W MON INTO HOLD QUEUE CALL ENTER(21,AA)

> RETURN END

С C DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: ESMON (END OF S/W MONITOR TRACE) С EVENT NUMBER: 22 С FUNCTION: TERMINATES S/W MONITOR TRACE AND FREES \* С MFMORY PARTITION. IF NECESSARY IT WILL Ç LOAD JOB SCHEDULER. С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 5** С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: SSMON С AUTHOR: DAVID L. OWEN С С HISTORY: N/A. С SUBROUTINE ESMON COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (FRE=0.0, BUSY=1.0, OFF=0.0) PRINT \*, 'END OF SOFTWARE MONITOR TRACE' C\*\*\* TURN OFF S/MON ON FLAG XX(2) = OFFC\*\*\* SET CPU RATIO BACK TO 100% XX(5) = 1.0C\*\*\* FIND MEMORY PARTION USED AND FREE IT DO 5 III=1,XX(3)IF (PART(III,1) .EQ. ATRIB(14)) THEN PART(III,2) = FREGOTO 10 END IF CONTINUE 5 C\*\*\* IF THIS STATEMENT IS REACHED A PARTION WAS NOT FOUND PRINT \*, 'ERROR - UNABLE TO FIND MEMORY PARTION TO FREE' PRINT \*, JOB ATRIBUTES ARE PRINT \*,(ATRIB(I),I=1,20)

and the second se

```
PRINT *, 'MEMORY PARTIONS ARE'

DO 6 J=1,XX(3)

PRINT *, PART(J,1), PART(J,2)

6 CONTINUE

RETURN

C*** LOAD JOB SCHEDULER INTO EXECQ IF IT ISN'T ALREADY

10 IF (XX(11) .NE. 1.) THEN

CALL LJSCH

XX(11) = 1.0

END IF

RETURN

END
```

C\* C С DATE: 5 DEC 1983 С VERSION: 1.0 C С NAME: SMON (S/W MONITOR) С **EVENT NUMBER: 23** С FUNCTION: RECORDS S/W MONITOR DATA С FILES READ: NONE. С FILES WRITTEN: SFTMON С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С AUTHOR: DAVID L. OWEN С С HISTORY: N/A. C C\* SUBROUTINE SMON COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM COMMON/UCOM5/IQ(5) CHARACTER NAMES(42)\*15 , INPUT QUEUE DATA NAMES/'JOB ARRIVAL 1'HOLD QUEUE , EXEC QUEUE , OUPUT QUEUE , ARVL OSPOL , ARVL S/WMON , NOT USED , TAPE 3 QUEUE , TAPE 6 QUEUE , TAPE 9 QUEUE , DISK 2 QUEUE , DISK 5 QUEUE , DISK 8 QUEUE , CHAN 1 QUEUE 1'ARVL SPOOL , ARVL USPOL , JOB FINISHED , TAPE 2 QUEUE , TAPE 5 QUEUE , TAPE 8 QUEUE , DISK 1 QUEUE , DISK 4 OUEUE 1 ARVL JSCHED 1 TAPE 1 QUEUE 1 TAPE 4 QUEUE 1 TAPE 7 QUEUE 1 TAPE 10 QUEUE , DISK 1 QUEUE , DISK 4 QUEUE , DISK 7 QUEUE , DISK 10 QUEUE , CHAN 3 QUEUE , CPU , NOT USED 1'DISK 3 QUEUE 1°DISK 6 QUEUE , CHAN 1 QUEUE 1'DISK 9 QUEUE CHAN 4 QUEUE 1 CHAN 2 QUEUE , CHAN 4 QUEUE , GENERAL CHAN , SPOOL QUEUE 1'CHAN 5 QUEUE 1'NOT USED -1 1'OSPOOL QUEUE C\*\*\* CHECK TO SEE IF IT IS ONE OF THE 5 QUEUES BEING MONITORED KQ = NINT(ATRIB(21))IF ((KQ .EQ. IQ(1)) .OR. (KQ .EQ. IQ(2)) .OR. (KQ .EQ. IQ(3)) .OR. (KQ .EQ. IQ(4)) .OR. (KQ .EQ. IQ(5))) THEN 1

C\*\*\* CALCULATE TIME IN QUEUE

## TIME = TNOW - ATRIB(22)

RETURN END

C\*\*\* TABLE LOOK UP FOR QUEUE NAMES C\*\*\* CREATE INTEGER INDEXES TO LOOK UP NAMES C\*\*\* NOTE: NEED TO OFFSET BY ONE BECAUSE ARRAY STARTS AT 1 NOT 0 IQUEUE = NINT(ATRIB(21) + 1)IFROM = NINT(ATRIB(??) +1)ITO = NINT(ATRIB(24, + 1) PRINT \*, WRITING S/W MON DATA NOW PRINT \*, JOB= ',ATRIB(2), Q= ',NAMES(IQUEUE), TIME = ', TIME, FROM= ', NAMES(IFROM), TO= ', NAMES(ITO) 1 WRITE (UNIT=14, FMT=1, IOSTAT=IOS, ERR=990)NAMES(IQUEUE), TIME, 1 NAMES(IFROM), NAMES(ITO) 1 FORMAT(1X,A15,1X,F8.3,1X,A15,1X,A15) BETURN END IF C\* IN FOR DEBUG ONLY IF (ATRIB(21) .EQ. 99.) THEN PRINT \*, JOB ENTERED QUEUE BEFORE S/W MON WAS TURNED ON' ELSE PRINT \*, SWMON NOT MONITORING THIS QUEUE PRINT \*, Q IS `, NAMES(KQ+1) END IF RETURN PRINT \*, ERROR IN WRITING TO S/W MON DATA FILE" PRINT \*, IOSTAT = ', IOS 990

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: PBUFF (PRINT BUFFER) С EVENT NUMBER: 24 С FUNCTION: PRINTS BUFFER TO LINE PRINTER С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 17** С CALLING EVENTS: 16 С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE PBUFF COMMON/SCOM1/ ATRIB(100), DD(100), DCL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON= 1.0) C\*\*\* UPDATE CPU TIME USED ATRIB(12) = ATRIB(12) + (TNOW - XX(4)) \* XX(5)C\*\*\* UPDATE NUMBER OF LINES LEFT TO PRINT IF (ATRIB(8) .LE. XX(27)) THEN ATRIB(8) = 0.ELSE ATRIB(8) = ATRIB(8) - XX(27)END IF C\*\*\* FREE CPU CALL FREE(3,1) C\*\*\* IF H/W MON IS ON THEN UPDATE PRINTER AND CHANNEL DATA IF (XX(30) .EQ. ON) THEN PCNT = PCNT + 1IF (PSTRT .NE. -1.0) THEN IF (PFIN .LT. TNOW) THEN PSUM = PSUM + PFIN - PSTRT PSTRT = TNOW END IF ELSE
```
PSTRT = TNOW
         END IF
         PFIN2 = ((XX(27) / XX(25)) + 60) + TNOW
          IF (PFIN2 .GE. PFIN) THEN
             PFIN = PFIN2
         END IF
         NCHL = NINT(ATRIB(20))
         IF (CSTRT(NCHL) .NE. -1.0) THEN
IF (CFIN(NCHL) .LT. TNOW) THEN
              CSUM(NCHL) = CSUM(NCHL) + CFIN(NCHL) - CSTRT(NCHL)
              CSTRT(NCHL) = TNOW
             ELSE
              CCNT(NCHL) = CCNT(NCHL) + 1
           END IF
          ELSE
           CSTRT(NCHL) = TNOW
           CCNT(NCHL) = CCNT(NCHL) + 1
         END IF
         IF (PFIN2 .GT. CFIN(NCHL)) THEN
             CFIN(NCHL) = PFIN2
         END IF
      END IF
C***
      SCHEDULE COMPLETION OF PRINTING BUFFER
      CALL SCHDL (17, (XX(27) / XX(25)) * 60, ATRIB)
```

RETURN END

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: ATAPE (ASSIGN TAPES) С EVENT NUMBER: N/A С FUNCTION: ASSIGNS TAPE DRIVES TO JOB С FILES READ: NONE. С FILES WRITTEN: NONE. С EVENTS CALLED: NONE. С CALLING EVENTS: NONE. С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: JSCHD С С AUTHOR: DAVID L. OWEN HISTORY: N/A. С С SUBROUTINE ATAPE COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (FREE = 0.0, BUSY = 1.0) C\*\*\* ACQUIRE NUMBER OF TAPES REQUESTED DO 10 III=1,A(6) C\*\* FIND FREE TAPE DO 5 JJJ=1,XX(9)IF (TAPE(JJJ,2) .EQ. FREE) THEN TAPE(JJJ, 2) = BUSYTAPE(JJJ,3) = A(2)C\* DECREMENT NUMBER OF AVAILABLE TAPE DRIVES XX(10) = XX(10) - 1**GOTO 10** END IF 5 CONTINUE IF THIS STATEMENT IS REACHED THEN TAPES WERE NOT ASSIGNED С PRINT \*, 'UNABLE TO ASSIGN TAPES PROPERLY' CONTINUE 10 RETURN END

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: JSCHD (JOB SCHEDULER) С EVENT NUMBER: N/A С FUNCTION: ASSIGNS RESOUCES TO JOBS IN THE HOLD С QUEUE AND THEN SENDS JOB TO EXECUTER QUEUE\* С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 18** С CALLING EVENTS: 4 С SUBROUTINES CALLED: ATAPE С CALLING SUBROUTINES: NONE. AUTHOR: DAVID L. OWEN HISTORY: N/A. С SUBROUTINE JSCHD COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT,RSTRT,RFIN,RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (ON = 1.0, FREE=0.0, BUSY=1.0) C\*\*\* PRINT JOB SCHEDULER STARTED CXXXXXPRINT \*, JOB SCHEDULER IS NOW EXECUTING C\*\*\* PRINT MEMORY PARTIONS CXXXXXPRINT \*, MEMORY PARTIONS ARE CXXXXXPRINT \*, (PART(I,1), PART(I,2), I=1, XX(3)) CXXXXXPRINT \*, TAPES ARE CXXXXXPRINT \*, (TAPE(I,1), TAPE(I,2), TAPE(I,3), I=1, XX(9)) С C\*\*\* FOR I = 1, NUM-IN-HOLD QUEUE DO III = 115 IF (NNQ(2) .EQ. 0) GOTO 10 IF (III .GT. NNQ(2)) GOTO 10 C\*\*\* COPY THE ATRIBUTES OF THE JOB CALL COPY(III,2,A) C\*\*\* PRINT WHAT JOB LOOKING AT FOR DEBUG PURPOSES CXXXXXPRINT \*, TRYING TO FIND RESOURCES FOR: CXXXXXPRINT \*, (A(K), K=1, 20)

```
C**
        WHILE JJJ .LE. NUM-OF-PARTIONS
        DO 5 JJJ=1,XX(3)
C*
          CHECK TO SEE IF PARTION IS LARGE ENOUGH AND FREE
           IF ((PART(JJJ,1) .GE. A(4)) .AND.
              (PART(JJJ,2) .EQ. FREE)) THEN
     1
С
CXXXXXPRINT *, 'PARTION ', JJJ, ' IS GOOD'
С
C*
              CHECK TO SEE IF ENOUGH TAPE DRIVES ARE AVAILABLE
             IF (XX(10) .GE. A(6)) THEN
CXXXXXPRINT *, "ENOUGH TAPES, ACQUIRE RESOURCES"
                GET JOB FROM HOLD QUEUE
C*
                 CALL RMOVE(III,2,A)
С
                ACQUIRE MEMORY PARTITION
C***
                PRINT WHAT PARTION USED FOR DEBUG
CXXXXXPRINT *, 'PARTION ', JJJ, 'USED'
                PART(JJJ, 2) = BUSY
С
                MEMORY-SIZED-USED = PARTION SIZE
                A(14) = PART(JJJ,1)
С
                ACOUIRE TAPES
                 CALL ATAPE
C***
                PUT JOB INTO EXEC QUEUE
                CALL FILEM(5,A)
C*** PRINT MEMORY PARTIONS
CXXXXXPRINT *, 'MEMORY PARTIONS ARE'
CXXXXXPRINT *, (PART(I,1), PART(I,2), I=1, XX(3))
CXXXXXPRINT *, TAPES ARE
CXXXXXPRINT *, (TAPE(I,1), TAPE(I,2), TAPE(I,3), I=1, XX(9))
C*
                 FOUND RESOURCES SO GO TO NEXT JOB IN HOLD QUEUE
                GOTO 15
             END IF
          END IF
С
5
        CONTINUE
C*** THIS STATEMENT REACHED ONLY IF JOB NOT ALLOCATED
C***
        IF JOB NOT ALLOCATABLE THEN WRITE SO
CXXXXXPRINT *, JOB NOT ALLOCATED'
C***
        INCREMENT POINTER IN HOLDO
        III = III + 1
C***
        GO BACK AND CHECK NEXT JOB
        GOTO 15
10
      CONTINUE
C*** IF H/W MON = ON, THEN COLLECT TIMER DATA
      IF (XX(30) .EQ. ON) THEN
         IF (XX(36) .NE. -1.0) THEN
           IF (XX(37) .LT. TNOW) THEN
```

```
D-69
```

```
XX(32) = XX(32) + XX(37) - XX(36)
             XX(36) \approx TNOW
          END IF
          ELSE
           XX(36) = TNOW
         END IF
         UFIN2 = XX(13) + TNOW
         IF (UFIN2 .GE. XX(37)) THEN
            XX(37) = UFIN2
         END IF
      END IF
С
C*** SET JSCHED IN EXECQ FLAG TO FALSE
      XX(11) = 0.
С
C*** SCHEDULE COMPLETION OF JOB SCHEDULER
      CALL SCHDL(18,XX(13),ATRIB)
С
      RETURN
      END
```

C\*: С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: OSPOL (OUTPUT SPOOLER) С EVENT NUMBER: N/A С FUNCTION: SPOOLS JOB ON DISK TO THE PRINTER С FILES READ: NONE. С FILES WRITTEN: NONE. EVENTS CALLED: 8, 15 С С CALLING EVENTS: 4 С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С SUBROUTINE OSPOL COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM PARAMETER (OSPLR=41.0, ON=1.0, OFF=0.0, NEW=0.0, BFULL=1.0, BEMPT=2.0) C\*\*\* IF H/W MON IS ON THEN COLLECT CPU TIMER DATA IF (XX(30) .EQ. ON) THEN IF (XX(36) .NE. -1.0) THEN IF (XX(37) .LT. TNOW) THEN XX(32) = XX(32) + XX(37) - XX(36)XX(36) = TNOWEND IF ELSE XX(36) = TNOWEND IF UFIN2 = XX(14) + TNOWIF (UFIN2 .GE. XX(37)) THEN XX(37) = UFIN2END IF END IF C\*\*\* IF START OF NEW JOB SPOOLING IF (XX(16) .EQ. NEW) THEN IF (NNQ(4) .NE. 0) THEN

CALL COPY(1,4,A)

C\*\*\* SET NUM-LINES IN INPUT SPOOLER ATRIB = JOB'S NUM-CARDS ATRIB(8) = A(8)C\*\*\* SCHEDULE CPU COMPLETION, THEN LOAD BUFFER FROM DISK CALL SCHDL(8,XX(14),ATRIB) RETURN END IF END IF C\*\*\* IF BUFFER IS FULL IF (XX(16) .EQ. BFULL) THEN C\*\*\* SET ATRIB(19) TO LINE PRINTER ATRIB(19) = OSPLRC\*\*\* ACQUIRE CHANNEL FOR CARD READER PRIOR TO PRINTING CALL SCHDL(15,XX(14),ATRIB) RETURN END IF C\*\*\* IF BUFFER IS EMPTY IF (XX(16) .EQ. BEMPT) THEN C\*\*\* SCHEDULE CPU COMPLETION THEN LOAD BUFFER FROM DISK CALL SCHDL(8,XX(14),ATRIB) RETURN END IF RETURN END

C С DATE: 5 DEC 1983 С VERSION: 1.0 С NAME: SPOOL (INPUT SPOOLER) С С EVENT NUMBER: N/A С FUNCTION: SPOOLS JOB ON TO DISK FROM THE READER С FILES READ: NONE. С FILES WRITTEN: NONE. EVENTS CALLED: 8, 15 С CALLING EVENTS: 4 С С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С AUTHOR: DAVID L. OWEN С С HISTORY: N/A. С SUBROUTINE SPOOL COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPR NT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100),TNEXT,TNOW,XX(100) COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3) COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10), 1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM, 1 PCNT, PSTRT, PFIN, PSUM COMMON/UCOM4/CARD(6), PRNTR(6) PARAME TER (SPLR=40.0, ON=1.0, OFF=0.0, NEW=0.0, BFULL=1.0, BEMPT=2.0) C\*\*\* COLLECT CPU TIMER DATA FOR H/W MONITOR IF (XX(30) . EQ. ON) THEN IF (XX(36) .NE. -1.0) THEN IF (XX(37) .LT. TNOW) THEN XX(32) = XX(32) + XX(37) - XX(36)XX(36) = TNOWEND IF ELSE XX(36) = TNOWEND IF UFIN2 = XX(14) + TNOWIF (UFIN2 .GE. XX(37)) THEN XX(37) = UFIN2END IF END IF C\*\*\* IF START OF NEW JOB SPOOLING IF (XX(15) .EQ. NEW) THEN

C <b>*</b> **	IF (NNQ(1) .NE. 0) THEN ACQUIRE CARD READER ??????????	
	CALL COPY(1,1,A) PRINT *, SPOOLING NEW JOB PRINT *, (A(I), I=1,21)	
C***	SET NUM-CARDS IN INPUT SPOOLER ATRIB = JOB'S NUM-CARDS ATRIB(7) = A(7)	
C***	SET ATRIB(19) TO CARD READER ATRIB(19) = SPLR	
C***	SCHEDULE CPU TIME OF CPU COMPLETION, THEN ACQUIRE CHANNEL CALL SCHDL(15,XX(14),ATRIB) RETURN END IF END IF	
C <b>*</b> **	IF BUFFER IS FULL IF (XX(15) .EQ. BFULL) THEN SCHEDULE CPU COPMLETION THEN DO I/O CALL SCHDL(8,XX(14),ATRIB) RETURN END IF	
C*** C*** C***	IF BUFFER IS EMPTY IF (XX(15) .EQ. BEMPT) THEN SET ATRIB(19) TO CARD READER ATRIB(19) = SPLR SCHEDULE CPU COMPLETION THEN GET CHANNEL CALL SCHDL(15,XX(14),ATRIB) RETURN END IF	
	R E TUR N E ND	

С С DATE: 5 DEC 1983 С VERSION: 1.0 С С NAME: SSMON (START S/W MONITOR TRACE) С EVENT NUMBER: N/A С FUNCTION: FREES CPU AND STARTS S/W MONITOR TRACE \* С FILES READ: NONE. С FILES WRITTEN: NONE. С **EVENTS CALLED: 22** С CALLING EVENTS: 4 С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С AUTHOR: DAVID L. OWEN С С HISTORY: N/A. С SUBROUTINE SSMON COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA 1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100) 1,SSL(100), TNEXT, TNOW, XX(100) PARAMETER (ON=1.0) PRINT \*, \*\*\* START OF SOFTWARE MONITOR TRACE C\*\*\* TURN S/W MON ON FLAG TO ON XX(2) = ONC\*\*\* SET THE CPU RATIO TO 95% OF ITS PRESENT SPEED XX(5) = .95C\*\*\* FREE CPU CALL FREE(3,1) C\*\*\* SCHEDULE END OF S/W MONITOR TRACE CALL SCHDL(22,XX(44)-TNOW,ATRIB)

RETURN END

```
С
С
        DATE: 5 DEC 1983
С
        VERSION: 1.0
С
С
        NAME: USER (USER CPU BURST)
С
        EVENT NUMBER: N/A
С
        FUNCTION: PERFORMS A TIME BURST ON A USER JOB.
С
                CALCULATES ENDING TIME OF BURST.
С
                DETERMINES WHY BURST ENDED AND CALLS
С
                APPROPRIATE ROUTINE.
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
       EVENTS CALLED: 7, 8, 9, 10
С
С
        CALLING EVENTS: 4
С
        SUBROUTINES CALLED: NONE.
С
        CALLING SUBROUTINES: NONE.
С
С
       AUTHOR: DAVID L. OWEN
        HISTORY: N/A.
C
C
                   *******************
Cź
      SUBROUTINE USER
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
     1, MSTOP, NCLNR, NCRDR, NPR NT, NNR UN, NNSET, NTAPE, SS(100)
     1,SSL(100), TNEXT, TNOW, XX(100)
      COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
      COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
     1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM,
     1 PCNT, PSTRT, PFIN, PSUM
      PARAMETER (ON = 1.0)
     REAL IOTIM
C***
      CALCULATE INTERMEDIATE VARIABLES I/O-TIME, TAPE-TIME, CPU-TIM-LEFT
C***
      JOB-COMPLETE-TIME, AND TIME-OUT-TIME
C**
      IOTIM = TIME OF NEXT DISK I/O
C**
      TTIME = TIME OF NEXT TAPE I/O
C**
      CTIME - TIME WHEN JOB WOULD BE COMPLETED IF NO I/OS OR TIMEOUTS
C**
      TIMEO = TIME WHEN TIME SLICE WOULD BE OVER
C**
      CTU = CPU TIME LEFT
C***
      CPU-TIM-USED = CPU-TIME-REQUIRED - CPU-TIME-LEFT
C***
        THIS IS DIFFERENT FROM ATRIB(12) BECAUSE ATRIB(12) INCLUDES
C***
        SPOOLER CPU TIME AND I/O CPU TIME
      CTU = ATRIB(3) - ATRIB(18)
C***
     CHECK TO MAKE SURE YOU DON'T DIVIDE BY O FIRST
      IF (ATRIB(9) .NE. 0) THEN
C***
        I/O-TIME= TIME-LAST-IO + TIME-BETWEEN - (ROUND OFF ERROR)
```

```
IOTIM = ATRIB(16) + (ATRIB(3) / ATRIB(9)) - .0000000001
     ELSE
       IOTIM = 9999999.
     END IF
     IF (ATRIB(10) .NE. 0) THEN
C***
       TAPE-TIME = TIME-LAST-TAPE + TIME-BETWEEN-TAPE - (ROUND OFF ERR)
       TTIME = ATRIB(17) + (ATRIB(3) / ATRIB(10)) - .0000000001
     ELSE
       TTIME = 9999999.
     END IF
C*** JOB COMPLETE TIME = TNOW + CPU-TIME-LEFT
     CTIME = CTU + (ATRIB(18) / XX(5))
C*** TIME-OUT-TIME = TNOW + TIME-SLICE
     TIMEO = CTU + XX(7)
C*** CHECK FOR DISK I/O
C**
      IF I/O TIME LESS THAN TAPE, JOB COMPLETE AND TIME OUT THEN
     IF ((IOTIM .LE. TTIME) .AND. (IOTIM .LT. CTIME) .AND.
    1
          (IOTIM .LE. TIMEO)) THEN
C**
         SCHEDULE TIME OF DISK REQUEST
         CALL SCHDL (8, (IOTIM - CTU), ATRIB)
C**
         IF H/W MON = ON, THEN COLLECT TIMER DATA
         IF (XX(30) . EQ. ON) THEN
            IF (XX(36) .NE. -1.0) THEN
             IF (XX(37) .LT. TNOW) THEN
               XX(32) = XX(32) + XX(37) - XX(36)
               XX(36) = TNOW
             END IF
            ELSE
             XX(36) = TNOW
            END IF
           UFIN2 = (IOTIM - CTU) + TNOW
            IF (UFIN2 .GE. XX(37)) THEN
              XX(37) = UFIN2
            END IF
         END IF
         RETURN
     END IF
C*** CHECK FOR TAPE I/O
     IF ((TTIME .LE. CTIME) .AND. (TTIME .LE. TIMEO)) THEN
```

```
C*×
        SCHEDULE TIME OF TAPE REQUEST
        CALL SCHDL (9, (TTIME - CTU), ATRIB)
C**
        IF H/W MON = ON THEN COLLECT TIMER DATA
        IF (XX(30) .EQ. ON) THEN
           IF (XX(36) .NE. -1.0) THEN
             IF (XX(37) .LT. TNOW) THEN
               XX(32) = XX(32) + XX(37) - XX(36)
               XX(36) = TNOW
             END IF
            ELSE
             XX(36) = TNOW
           END IF
           UFIN2 = TTIME - CTU + TNOW
           IF (UFIN2 .GE. XX(37)) THEN
              XX(37) = UFIN2
           END IF
        END IF
        RETURN
     END IF
C*** CHECK FOR JOB COMPLETE
     IF (CTIME LE. TIMEO) THEN
     PRINT *, 'USER JOB WILL BE COMPLETED WITH THIS BURST'
C*** SCHEDULE COMPLETION OF JOB
     CALL SCHDL(7, (ATRIB(18) / XX(5)), ATRIB)
C**
       IF H/W MONITOR = ON, THEN COLLECT H/W MONITOR DATA
C**
        IF H/W MON = ON THEN COLLECT TIMER DATA
        IF (XX(30) .EQ. ON) THEN
           IF (XX(36) .NE. -1.0) THEN
             IF (XX(37) .LT. TNOW) THEN
               XX(32) = XX(32) + XX(37) - XX(36)
               XX(36) = TNOW
             END IF
            ELSE
             XX(36) = TNOW
           END IF
           UFIN2 = (ATRIB(18) / XX(5)) + TNOW
           IF (UFIN2 .GE. XX(37)) THEN
              XX(37) = UFIN2
           END IF
        END IF
      RETURN
```

```
D-78
```

END IF

```
C*** TIME OUT
C*** IF IT IS NONE OF THE ABOVE CONDITIONS IT MUST BE A TIME-OUT
C**
     SCHEDULE TIME-OUT
     CALL SCHDL (10,XX(7),ATRIB)
C*** IF H/W MON IS ON THEN COLLECT CPU TIMER DATA
     IF (XX(30) .EQ. ON) THEN
        IF (XX(36) .NE. -1.0) THEN
         IF (XX(37) .LT. TNOW) THEN
           XX(32) = XX(32) + XX(37) - XX(36)
           XX(36) = TNOW
         END IF
        ELSE
         XX(36) = TNOW
        END IF
        UFIN2 = XX(7) + TNOW
        IF (UFIN2 .GE. XX(37)) THEN
          XX(37) = UFIN2
        END IF
     END IF
     RETURN
```

END

```
С
С
       DATE: 5 DEC 1983
                                                       *
С
       VERSION: 1.0
С
С
       NAME: USERF (USER FUNCTION)
С
       EVENT NUMBER: N/A
С
       FUNCTION: CALCULATES THE PORTION OF THE CHANNEL
С
               THAT THE PARTICULAR I/O DEVICES IS USING.
С
       FILES READ: NONE.
С
       FILES WRITTEN: NONE.
С
       EVENTS CALLED: NONE.
С
       CALLING EVENTS: 13, 14, 17
С
       SUBROUTINES CALLED: NONE.
С
       CALLING SUBROUTINES: NONE.
С
С
       AUTHOR: DAVID L. OWEN
С
       HISTORY: N/A.
С
FUNCTION USERF(1)
      COMMON/SCOM1/ ATRIB(100), DD(100), DDL(100), DTNOW, II, MFA
     1, MSTOP, NCLNR, NCRDR, NPRNT, NNRUN, NNSET, NTAPE, SS(100)
     1,SSL(100), TNEXT, TNOW, XX(100)
     COMMON/UCOM1/A(100), PART(20,2), TAPE(20,3)
     COMMON/UCOM2/DISK(10,7), DCNT(10), DSTRT(10), DSUM(10), DFIN(10),
    1 CCNT(5), CSUM(5), CFIN(5), CSTRT(5), RCNT, RSTRT, RFIN, RSUM,
    1 PCNT, PSTRT, PFIN, PSUM
     COMMON/UCOM3/TAP(10,7), TCNT(10), TSTRT(10), TSUM(10), TFIN(10)
     COMMON/UCOM4/CARD(6), PRNTR(6)
     PARAMETER (SPLR = 40.0, OSPLR = 41.0)
C*** IF IT IS A LIFE USING THE CHANNEL
     IF ((ATRIB(19) .GE. 11.) .AND. (ATRIB(19) .LE. 20.)) THEN
        J = ATRIB(19) - 10
C***
       CALCULATE THE PORTION OF THE CHANNEL THE TAPE IS USING
       IF ((J .GE. 1) .AND. (J .LE. XX(9))) THEN
          USERF = NINT(1 / TAP(J,6))
       ELSE
         PRINT *, ERROR : ILLEGAL TAPE
         PRINT *, (ATRIB(K), K=1, 20)
       END IF
       RETURN
     END IF
C***
     IF IS A DISK USING THE CHANNEL
     IF ((ATRIB(19) .GT. 20.) .AND. (ATRIB(19) .LE. 30.)) THEN
       J = ATRIB(19) - 20
```

D-80

and a second second

```
C***
        CALCULATE THE PORTION OF THE CHANNEL THE DISK IS USING
        IF ((J .GE. 1) .AND. (J .LE. XX(22))) THEN
           ITEMP = NINT(1 / DISK(J,6))
           USERF = ITEMP
        ELSE
          PRINT *, 'ERROR : ILLEGAL DISK'
          PRINT *,(ATRIB(K),K=1,20)
        END IF
        RETURN
      END IF
C*** IF IS THE CARD READER USING THE CHANNEL
      IF (ATRIB(19) .EQ. SPLR) THEN
C***
        CALCULATE THE PORTION OF THE CHANNEL THE DISK IS USING
           ITEMP = NINT(1 / ((IIX(20)/XX(18)) * 60))
           USERF = ITEMP
        RETURN
      END IF
C*** IF IS THE LINE PRINTER USING THE CHANNEL
      IF (ATRIB(19) .EQ. OSPLR) THEN
C***
        CALCULATE THE PORTION OF THE CHANNEL THE DISK IS USING
           ITEMP = NINT(1 / ((XX(27)/XX(25)) * 60))
           USERF = ITEMP
        RETURN
      END IF
C*** IF IT IS NONE OF THE ABOVE THEN THERE IS AN ERROR
```

PRINT \*, ERROR: TRYING TO ACCESS CHANNEL WITH ILLEGAL DEVICE PRINT \*, (ATRIB(K), K=1,20) RETURN

END





• 1

MICROCOPY RESOLUTION TEST CHART NATIONAL BUREAU OF STANDARDS-1963-A

```
******************************
        DATE: 5 DEC 1983
        VERSION: 1.0
        NAME: SIMS
        FUNCTION: PERFORMS NETWORK PORTION OF CPESIM II
        FILES READ: NONE.
        FILES WRITTEN: NONE.
        NETWORKS: MAIN JOB FLOW, OUTPUT SPOOLER, DISK,
                   AND CHANNEL NETWORKS
        AUTHOR: DAVID L. OWEN
        HISTORY: N/A.
    ****************
GEN, DAVID L. OWEN, ABC COMPUTER SIMULATION, 12/05/83;
LIMITS, 40, 24, 300;
PRIORITY/3, HVF(11)/2, LVF(5);
NETWORK;
      RESOURCE/INQ(1),1;
                             INPUT QUEUE
                             HOLD QUEUE
      RESOURCE/HOLDQ(1),2;
      RESOURCE/EXECQ(1),3;
                             EXECUTE QUEUE
      RESOURCE/OUTQ(1),4;
                             OUTPUT QUEUE
      RESOURCE/DUM1(1),5;
                             DUMMY 1 QUEUE
                             DUMMY 2 QUEUE
      RESOURCE/DUM2(1),6;
      RESOURCE/DUM3(1),7;
                             DUMMY 3 QUEUE
      RESOURCE/DUM4(1),8;
                             DUMMY 4 QUEUE
                             DUMMY 5 QUEUE
      RESOURCE/DUM5(1),9;
      RESOURCE/DUM6(1),10;
                             DUMMY 6 QUEUE
      RESOURCE/TPQ1(1),11;
                             TAPE 1 QUEUE
      RESOURCE/TPQ2(1), 12;
                             TAPE 2 QUEUE
                             TAPE 3 QUEUE
      RESOURCE/TPQ3(1),13;
                             TAPE 4 QUEUE
      RESOURCE/TPQ4(1), 14;
                             TAPE 5 QUEUE
      RESOURCE/TPQ5(1),15;
                             TAPE 6 QUEUE
      RESOURCE/TPQ6(1),16;
      RESOURCE/TPQ7(1),17;
                             TAPE 7 QUEUE
      RESOURCE/TPQ8(1),18;
                             TAPE 8 QUEUE
                             TAPE 9 QUEUE
      RESOURCE/TPQ9(1),19;
      RESOURCE/TPQ10(1),20;
                             TAPE 10 QUEUE
      RESOURCE/DSQ1(1),21;
                             DISK 1 QUEUE
      RESOURCE/DSQ2(1),22;
                             DISK 2 QUEUE
                             DISK 3 QUEUE
      RESOURCE/DSQ3(1),23;
                             DISK 4 QUEUE
      RESOURCE/DSQ4(1), 24;
                             DISK 5 QUEUE
      RESOURCE/DSQ5(1),25;
                             DISK 6 QUEUE
      RESOURCE/DSQ6(1),26;
      RESOURCE/DSQ7(1),27;
                             DISK 7 QUEUE
      RESOURCE/DSQ8(1),28;
                             DISK 8 QUEUE
      RESOURCE/DSQ9(1), 29:
                             DISK 9 QUEUE
                             DISK 10 QUEUE
      RESOURCE/DSQ10(1), 30;
                              CHANNEL 1 QUEUE
      RESOURCE/CNLQ1(0),31;
      RESOURCE/CNLQ2(0),32;
                              CHANNEL 2 QUEUE
```

RESOURCE/CNLQ3(0),33; CHANNEL 3 QUEUE **RESOURCE/CNLQ4(0),34;** CHANNEL 4 QUEUE RESOURCE/CNLQ5(0),35; CHANNEL 5 CIEUE **ATTRIBUTES** ATRIB(1) ARRIVAL TIME ATRIB(2) JOB NAME ATRIB(3) CFU TIME REQUIRED ATRIB(4) MEMOR Y ATRIB(5) PRIORITY ATRIB(6) ALLOCATABLE DEVICES NEEDED ATRIB(7) CARDS LINES ATRIB(8) ATRIB(9) DISK BLOCKS ATRIB(10) ALLOCATABLE DEVICE BLOCKS ATRIB(11) JOB-TYPE 1. USER JOB 2. INPUT SPOOLER 3. OUTPUT SPOOLER 4. S/W MONITOR 5. JOB SCHEDULER ATRIB(12) CPU TIME USED ATRIB(13) I/O TIME USED ATRIB(14) MEMEORY SIZE USED ATRIB(15) DEPARTURE TIME ATRIB(16) TIME OF LAST I/O ATRIB(17)TIME OF LAST TAPE I/O CPU TIME LEFT ATRIB(18) ATRIB(19)ASSIGNED DISK ATRIB(20) ASSIGNED CHANNEL ATRIB(21) QUEUE NAME (USED WITH S/W MONITOR) ATRIB(22) TIME (TIME JOB ENTERED QUEUE) INPUT-FROM (WHERE JOB CAME FROM PRIOR TO QUEUE) ATRIB(23) ATRIB(24) OUTPUT-TO (WHERE JOB WILL GO AFTER QUEUE) XX VARIABLES XX(1) = SPOOLERON (INPUT SPOOLER IN SYSTEM) XX(2) = SWMON FLAGXX(3) = NUMBER OF MEMORY PARTITIONSXX(4) = CPU START TIMEXX(5) = CPU RATIO FOR S/W MON (1.0 = S/W MON OFF, .95 = S/W MON ON)XX(6) = RELATIVE CPU SPEEDXX(7) = TIME SLICEXX(8) = START I/O

57

XX(9) = NUMBER OF TAPE DRIVES; XX(10) = NUMBER OF AVAILABLE TAPE DRIVES; XX(11) = JOB SCHEDULER IN EXECQ (0 = TRUE, 1 = FALSE) ; XX(12) = OSPOOLERON (OUTPUT SOOLER IN SYSTEM)XX(13) = CPU TIME REQUIRED FOR JOB SCHEDULER TO EXECUTE ; XX(14) = CPU TIME FOR INPUT SPOOLER TO EXECUTEXX(15) = INPUT SPOOLER STATUS (0 -START OF JOB, : 1 - WRITE BUFFER TO DISK, 2 - READ CARDS INTO BUFFER) XX(16) = OUTPUT SPOOLER STATUS (0 -START OF JOB, ; 1 - PRINT BUFFER, 2 - LOAD BUFFER FROM DISK) : XX(17) = BLOCK SIZE (BYTES)XX(18) = CARD READER SPEED (CARDS PER MINUTE) XX(19) = CARD RECORD SIZE (BYTES) XX(20) = CARD PER BUFFER (CALCULATED FROM XX(17) AND XX(19)); XX(21) = NEXT DISK; XX(22) = NUMBER OF DISK DRIVESXX(23) = NUMBER OF CHANNELS: XX(24) = NEXT TAPE: XX(25) = PRINTER SPEED (LINES PER MINUTE) ; XX(26) = LINE RECORD SIZE (BYTES) : XX(27) = LINES PER BUFFER (CALCULATED FROM XX(17) AND XX(26)XX(30) = H/W MONITOR (0 = OFF, 1 = ON) XX(31) = CPU COUNT FOR H/W MONXX(32) = CPU TIMER FOR H/W MONXX(33) = H/W MONITOR START TIME XX(34) = H/W MONITOR STOP TIME XX(35) = HOW OFTEN H/W MONITOR RECORDS DATA: XX(36) = CPU START TIME FOR H/W MON XX(37) = CPU FINISH TIME FOR H/W MONXX(40) = TOTAL CPU TIME USED BY THE INPUT SPOOLER: XX(41) = TOTAL CPU TIME USED BY THE OUTPUT SPOOLERXX(42) = TOTAL CPU TIME USED BY THE JOB SCHEDULERXX(43) = S/W MONITOR START TIME XX(44) = S/W MONITOR STOP TIME BEGIN NETWORK : ENTER.1: JOB ARRIVAL, ASSIGN ATTR BUTES GOON,2; ACT/1,0.,,GON2; PUT JOB IN INPUT QUEUE ACT/2,0,XX(1).NE.1,SPO; SPOOLER IDLE, PUT IN HOLD QUEUE FLAG INDICATING SPOOLER LOADED SPO ASSIGN, XX(1)=1;ACT: EVENT,2; LOAD SPOOLER IN HOLD QUEUE TERM:

いいいと、同時になるためので、「」

 $\overline{\mathbf{S}}$ 

```
;
 GON2 GOON,1;
     ACT/3,0,XX(2).NE.1,QINP;
                                 LOAD JOB IMEDIATELY INTO INPUT QUEUE
                                  STORE DATA FOR S/W MONITOR
     ACT/4, 0, XX(2).EQ.1;
     ASSIGN,ATRIB(21)=1.0;
     ASSIGN, ATRIB(22)=TNOW;
     ASSIGN,ATRIB(23)=0.0;
     ASSIGN, ATRIB(24)=2.0;
     ACT;
QINP AWAIT(1), INQ/1, 1;
                                 WAIT IN INPUT QUEUE
                                  TRANSFER JOB TO HOLD OUEUE
;
     ACT/5,0,XX(2).NE.1,JT;
     ACT/6, 0, XX(2).EQ.1;
                                  RECORD DATA ON S/W MON FILE
     EVENT, 23;
     ASSIGN,ATRIB(11)=1;
                                 SET JOB TYPE TO USER JOB
 JT
     ASSIGN,ATRIB(23)=1.0;
                                 SET 'INPUT FROM' TO INPUT QUEUE
     ACT/7,0,,GON3;
     ENTER,2;
                                  INPUT SPOOLER
     ASSIGN,ATRIB(23)=5.0;
                                  SET 'INPUT FROM' TO INPUT SPOOLER
     ACT/8,0.0,,GON3;
                                  OUTPUT SPOOLER
     ENTER,7;
                                  SET 'INPUT FROM' TO OUTPUT SPOOLER
     ASSIGN,ATRIB(23)=6.0;
     ACT/9,0,,GON3;
                                 S/W MONITOR
     ENTER, 21;
     ASSIGN,ATRIB(23)=7.0;
                                SET 'INPUT FROM' TO S/W MONITOR
GON3 GOON,1;
;
     ACT/10,0,XX(2).NE.1,GON4;
                                  IF S/W MON NOT ON, GO TO GON4
     ACT/11,0,XX(2).EQ.1;
                                    IF S/W MON ON, STORE DATA FIRST
     ASSIGN, ATRIB(21)=2.0;
     ASSIGN, ATRIB(22)=TNOW;
     ASSIGN, ATRIB(24)=3.0;
     ACT;
;
 GON4 GOON;
     ACT/22,0.,,QHLD;
     ACT/23,0.,XX(11).NE.1,ET5; IF JSCHED NOT LOADED IN EXECQ
ET5
     EVENT.5:
                                          THEN LOAD JOB SCHED
     TERM:
;
 QHLD AWAIT(2), HOLDQ/1,1; WAIT IN HOLD QUEUE
;
      TERM;
;
;
     QUEUE(5);
                             DUMMY QUEUE SO THAT CAN RECORD
     ACT/78,0.0;
                             S/W MONITOR DATA BEFORE GOING TO
     GOON,1;
                             EXECUTE QUEUE
     ASSIGN, ATRIB(23)=2.0;
                                  SET 'INPUT FROM' TO HOLD QUEUE
                               IF S/W MON NOT ON THEN GO STAIGHT
;
```

ACT/12,0,XX(2).NE.1,GON5; TO THE GOON NODE ; IF S/W MON ON THEN ACT/13,0,XX(2).EQ.1; WRITE DATA TO DISK AND GO TO GOON NODE EVENT,23: ACT/14,0,,GON5; ENTER.4; **RETURN OF JOB AFTER TIMEOUT** ASSIGN,ATRIB(23)=3.0; SET 'INPUT FROM' TO EXEC QUEUE ACT/15,0,,GON5; RETURN OF JOB AFTER DISK I/O ENTER.3; ASSIGN, ATRIB(23)=ATRIB(20)+30; SET 'INPUT FROM' TO THE CHANNEL ACT/16.0., GON5: ENTER, 18; INSERTION OF JOB SCHEDULER SET 'INPUT FROM' TO JOB SCHEDULER ASSIGN, ATRIB(23)=8.0; ACT/17,0, .GON5; RETURN OF JOB AFTER TAPE I/O ENTER, 5; ASSIGN, ATRIB(23)=ATRIB(20)+30; SET 'INPUT FROM' TO INPUT QUEUE GON5 GOON.1: ; ACT/18,0,XX(2).NE.1,QEXC; IF S/W MON NOT ON THEN PUT JOB INTO QEXEC ; IF S/W MON ON THEN ACT/19,0,XX(2).EQ.1; ASSIGN.ATRIB(21)=3.0: ASSIGN, ATRIB(22)=TNOW; ASSIGN.ATRIB(24)=36.0: ACT: QEXC AWAIT(3), EXECQ/1,1; WAIT IN EXECUTE QUEUE FOR CPU IF S/W MON NOT ON THEN GO TO ET4 ACT/20,0,XX(2).NE.1,ET4; ACT/21,0,XX(2).EQ.1; IF S/W MON ON THEN WRITE S/W MON DATA TO DISK BEFORE EVENT, 23; ACT; GOING TO ET4 ET4 PERFORM CPU BURST EVENT,4; TERM; \*\*\*\*\*\*\* \*\*\* OUTPUT SPOOLER NETWORK \*\*\* \*\*\* \* ENTER, 10; ENTER USER JOB AFTER COMPLETION ACT/24; GON6 GOON; ACT/25,0,,GON7; PUT JOB INTO OUTPUT OUEUE ACT/26,0,XX(12).NE.1; OUTPUT SPOLER NOT IN SYSTEM SO LOAD SPOOLER EVENT,6; TERM; GON7 GOON,1; ACT/27,0.,XX(2).NE.1,QOUT; S/W MON NOT ON S/W MON ON SO RECORD DATA IN ATRIB ACT/28,0.,XX(2).EQ.1; ASSIGN,ATRIB(21)=4.0;

D-86

Ċ)

```
ASSIGN, ATRIB(22)=TNCW;
     ASSIGN, ATRIB(23)=36.0;
     ASSIGN, ATRIB(24)=9.0;
     ACT/29;
QOUT AWAIT(4), OUTQ/1;
     ACT, 0, XX(2).NE.1, ET19;
                               IF S/W MON NOT ON THEN GO TO ET19
     ACT.0,XX(2).EQ.1;
                               IF S/W MON ON THEN
                                  WRITE S/W MON DATA TO DISK BEFORE
     EVENT,23;
                                  GOING TO ET19
     ACT/30:
ET19 EVENT, 19;
;
     TERM;
****
:***
***
        DISK NETWORKS
***
  ENTER, 23, 1;
     ACT/80,0.,XX(2).NE.1,GON8;
                                     IF S/W MON NOT ON GO TO GON8
     ACT/81,0.,XX(2).EQ.1;
                                 IF S/W MON ON THEN RECORD ATRIBUTES
     ASSIGN, ATRIB(21)=ATRIB(19); DISK WAITING FOR
     ASSIGN, ATRIB(22)=TNOW;
     ASSIGN, ATRIB(23)=36.0;
                                  CPU
     ASSIGN, ATRIB(24)=37.0;
                                  GENERAL CHANNEL
                                  DECIDE WHICH DISK TO GO TO
GON8 GOON.1:
     ACT(1)/31,0.0,ATRIB(19).EQ.21,DK1;
     ACT(1)/32,0.0,ATRIB(19).EQ.22,DK2;
     ACT(1)/33,0.0,ATRIB(19).EQ.23,DK3;
     ACT(1)/34,0.0,ATRIB(19).EQ.24,DK4;
     ACT(1)/35,0.0,ATRIB(19).EQ.25,DK5;
     ACT(1)/36,0.0,ATRIB(19).EQ.26,DK6;
     ACT(1)/37,0.0,ATRIB(19).EQ.27,DK7;
     ACT(1)/38,0.0,ATRIB(19).EQ.28,DK8;
     ACT(1)/39,0.0,ATRIB(19).EQ.29,DK9;
     ACT(1)/40,0.0,ATRIB(19).EQ.30,DK10;
     ACT(1)/41.0.0. ERR1:
DK1 AWAIT(21), DSQ1/1;
     ACT, 0, XX(2).NE.1, D1;
                              IF S/W MON NOT ON THEN GO TO D1
     ACT, 0, XX(2).EQ.1;
                              IF S/W MON ON THEN
     EVENT,23;
                                  WRITE S/W MON DATA TO DISK BEFORE
     ACT/42;
                                  GOING TO D1
D1
     EVENT, 15;
     ACT/43;
     TERM;
DK2 AWAIT(22), DSQ2/1;
                              IF S/W MON NOT ON THEN GO TO D2
     ACT, 0, XX(2).NE.1, D2;
                              IF S/W MON ON THEN
     ACT, 0, XX(2).EQ.1;
```

D-87

D2	EVENT,23; ACT/44; EVENT,15; ACT/45; TERM;	WRITE S/W MON DATA TO DISK BEFORE GOING TO D2
, DK3 D3	AWAIT(23),DSQ3/1; ACT,0,XX(2).NE.1,D3; ACT,0,XX(2).EQ.1; EVENT,23; ACT/46; EVENT,15; ACT/47;	IF S/W MON NOT ON THEN GO TO D3 IF S/W MON ON THEN WRITE S/W MON DATA TO DISK
:	TERM;	
DK4	AWAIT(24),DSQ4/1; ACT,0,XX(2).NE.1,D4; ACT,0,XX(2).EQ.1;	IF S/W MON NOT ON THEN GO TO D4 IF S/W MON ON THEN
D4	EVENT,23; ACT/48; EVENT,15; ACT/49; TERM;	WRITE S/W MON DATA TO DISK
; DK5	AWAIT(25),DSQ5/1;	
	ACT,0,XX(2).NE.1,D5; ACT,0,XX(2).EQ.1;	IF S/W MON NOT ON THEN GO TO D5 IF S/W MON ON THEN
D5	ACT/49; EVENT,15; ACT/50; TERM;	WRITE S/W MON DATA TO DISK
; DK6	AWAIT(26).DS06/1:	
	ACT, 0, XX(2).NE.1, D6; ACT, 0, XX(2).EQ.1;	IF S/W MON NOT ON THEN GO TO D6 IF S/W MON ON THEN
D6	EVENT,23; ACT/51; EVENT,15;	WRITE S/W MON DATA TO DISK
	ACT/52; TERM;	
;	•	
DK7	AWAIT(27),DSQ7/1; ACT,0,XX(2).NE.1,D7; ACT,0,XX(2).EQ.1;	IF S/W MON NOT ON THEN GO TO D7 IF S/W MON ON THEN UBLIE S/W MON DATA TO DISK
D7	ACT/53; EVENT,15; ACT/54; TERM;	WRITE 5/W MUN DATA TU DISK
; DK8	AWAIT(28),DSQ8/1; ACT,0,XX(2).NE.1,D8;	IF S/W MON NOT ON THEN GO TO D8

D-88

 $\overline{\cdot}$ 

Ċ

```
ACT, 0, XX(2).EQ.1; IF S/W MON ON THEN
      EVENT, 23;
                                   WRITE S/W MON DATA TO DISK
      ACT/55:
D8
      EVENT, 15:
      ACT/56;
      TERM;
;
DK9 AWAIT(29), DSQ9/1;
     ACT, 0, XX(2).NE.1, D9;
ACT, 0, XX(2).EQ.1;
IF S/W MON NOT ON THEN
IF S/W MON ON THEN
                                    WRITE S/W MON DATA TO DISK
      EVENT,23;
      ACT/57;
D9
      EVENT, 15;
      ACT/58;
      TERM;
DK10 AWAIT(30), DSQ10/1;
     ACT, 0, XX(2).NE.1, D10; IF S/W MON NOT ON THEN GO TO D10
ACT, 0, XX(2).EQ.1; IF S/W MON ON THEN
      EVENT,23;
                                    WRITE S/W MON DATA TO DISK
     ACT/60;
D10 EVENT.15:
      ACT/61;
      TERM:
ERR1 TERM;
                         ERROR ROUTINE TO GO HERE
           *******
***
. ***
         CHANNEL NETWORKS
 ***
 ************************************
;
      ENTER, 25, 1;
      ACT/82,0.,XX(2).NE.1,GON9; IF S/W MON NOT ON GO TO GON9
      ACT/83,0.,XX(2).EQ.1;
                                 IF S/W MON ON THEN LOAD ATRIBUTES
      ASSIGN, ATRIB(21)=ATRIB(20)+30; CHANEL WAITING FOR
     ASSIGN, ATRIB(22)=TNOW;
      ASSIGN, ATRIB(23)=ATRIB(19); I/O DEVICE
      ASSIGN, ATRIB(24)=3.0;
                                   EXEC OUEUE
                        DECIDE WHICH CHANNEL TO GO TO
GON9 GOON,1;
      ACT(1)/62,0.0,ATRIB(20).EQ.1,CH1;
      ACT(1)/63,0.0,ATRIB(20).EQ.2,CH2;
      ACT(1)/64,0.0,ATRIB(20).EQ.3,CH3;
      ACT(1)/65,0.0,ATRIB(20).EQ.4,CH4;
      ACT(1)/66,0.0,ATRIB(20).EQ.5,CH5;
      ACT(1)/67,0.0,,ERR2;
CH1 AWAIT(31), CNLQ1/USERF(1);
     ACT, 0, XX(2). NE. 1, C1; IF S/W MON NOT ON THEN GO TO C1
                              IF S/W MON ON THEN
      ACT, 0, XX(2).EQ.1;
      EVENT,23;
                                   WRITE S/W MON DATA TO DISK
```

 $\cdot$ 

67

```
ACT/68;
      EVENT, 16;
 C1
      ACT/69;
      TERM;
;
CH2 AWAIT(32), CNLQ2/USERF(1);
      ACT, 0, XX(2).NE.1, C2; IF S/W MON NOT ON THEN GO TO C2
      ACT,0,XX(2).EQ.1;
                              IF S/W MON ON THEN
      EVENT,23;
                                   WRITE S/W MON DATA TO DISK
      ACT/70;
 C2
      EVENT,16;
      ACT/71;
      TERM;
;
 CH3 AWAIT(33), CNLQ3/USERF(1);
      ACT, 0, XX(2).NE.1, C3; IF S/W MON NOT ON THEN GO TO C3
      ACT,0,XX(2).EQ.1;
                              IF S/W MON ON THEN
      EVENT,23;
                                   WRITE S/W MON DATA TO DISK
      ACT/72:
 C3
      EVENT, 16;
      ACT/73;
      TERM:
;
CH4 AWAIT(34), CNLQ4/USERF(1);
      ACT, 0, XX(2).NE.1, C4; IF S/W MON NOT ON THEN GO TO C4
      ACT,0,XX(2).EQ.1;
                              IF S/W MON ON THEN
      EVENT,23;
                                   WRITE S/W MON DATA TO DISK
      ACT/74;
 C4
      EVENT, 16;
      ACT/75:
      TERM;
;
      AWAIT(35), CNLQ5/USERF(1);
CH5
     ACT, 0, XX(2).NE.1, C5; IF S/W MON NOT ON THEN GO TO C5
      ACT, 0, XX(2).EQ.1;
                               IF S/W MON ON THEN
      EVENT, 23;
                                   WRITE S/W MON DATA TO DISK
      ACT/76:
C5
      EVENT, 16;
      ACT/77;
      TERM:
;
ERR2 TERM;
                         ERROR ROUTINE TO GO HERE
;
      END;
INIT, 0, 600000;
MONTR, CLERA, 20., 20.0;
FIN;
```

D-90

С VERSION: 1.0 С С NAME: SETUP С FUNCTION: READS IN AVAILABLE HARDWARE EQUIPMENT С LIST, QUERIES USER ON CONFIGURATION С DESIRED AND THEN WRITES CONFIG FILE С FILES READ: TYPES С FILES WRITTEN: CONFIG С SUBROUTINES CALLED: NONE. С CALLING SUBROUTINES: NONE. С С AUTHOR: DAVID L. OWEN С HISTORY: N/A. С PROGRAM MAIN(TYPES, CONFIG) DIMENSION DSPEED(10), DCOST(10), TSPEED(10), TCOST(10), CSPEED(5), CCOST(5), RSPEED(5), RCOST(5), PCOST(5), PSPEED(5), 1 USPEED(5), UCOST(5), PART(20), ICHANL(5), IDISK(10,6), 1 ITAPE(10,6), IREAD(6), IPRINT(6), ISW MQ(5), IHW CNT(5)1 CHARACTER DNAME(10)\*20, TNAME(10)\*20, CNAME(5)\*20, RNAME(5)\*20, PNAME(5)\*20, UNAME(5)\*20, SANSWR\*1, HANSWR\*1 1 **REAL HSEC** PRINT \*, 'OPENING FILES' C\*\*\* OPEN INPUT FILE WHICH CONTAINS THE TYPES OF DEVICES OPEN(UNIT=15, FILE='TYPES', STATUS='OLD', ERR=990, IOSTAT=IOS, 1 ACCESS='SEQUENTIAL', FORM='FORMATTED') C\*\*\* OPEN CONFIG FILE WHICH WILL CONTAIN THE COMPUTER CONFIGURATION OPEN(UNIT=11, FILE='CONFIG', STATUS='NEW', ERR=990, 1 IOSTAT=IOS, ACCESS='SEQUENTIAL', FORM='FORMATTED') PRINT \*, FILES OPENED C\*\*\* **READ NUMBER OF DIFFERENT TYPES OF DISKS** READ(UNIT=15, FMT=100, ERR=999)NUMD C\*\*\* READ DISK TYPES IN DO 1 I=1,NUMD READ (UNIT=15, FMT=101, ERR=999)DNAME(I), DSPEED(I), DCOST(I) 1 CONTINUE C\*\*\* READ NUMBER OF DIFFERENT TYPES OF TAPE DRIVES READ(UNIT=15, FMT=100, ERR=999)NUMT C\*\*\* READ TAPE TYPES

-

C C

DATE: 5 DEC 1983

```
DO 2 I=1,NUMT
        READ (UNIT=15, FMT=101, IOSTAT=IOS, ERR=999)
              TNAME(I), TSPEED(I), TCOST(I)
     1
     CONTINUE
2
C*** READ NUMBER OF DIFFERENT TYPES OF CHANNELS
     READ (UNIT=15, FMT=100, ERR=999, IOSTAT=IOS)NUMC
C***
     READ CHANNEL TYPES
     DO 3 I=1,NUMC
     READ (UNIT=15, FMT=101, ERR=999, IOSTAT=IOS)CNAME(I),
     1
                    CSPEED(I),CCOST(I)
3
     CONTINUE
C*** READ NUMBER OF DIFFERENT TYPES OF CARD READERS
     READ (UNIT=15, FMT=100, ERR=999, IOSTAT=IOS)NUMR
C***
     READ CARD READER TYPES
     DO 4 I=1, NUMR
     READ (UNIT=15, FMT=101, ERR=999, IOSTAT=IOS)RNAME(I),
                   RSPEED(I),RCOST(I)
     1
     CONTINUE
4
C*** READ NUMBER OF DIFFERENT TYPES OF LINE PRINTERS
     READ (UNIT=15, FMT=100, ERR=999, IOSTAT=IOS)NUMP
C***
     READ PRINTER TYPES
     DO 5 I=1,NUMP
     READ (UNIT=15, FMT=101, ERR=999, IOSTAT=IOS)PNAME(I),
     1
                PSPEED(1), PCOST(1)
5
     CONTINUE
C*** READ NUMBER OF DIFFERENT TYPES OF CPUS
     READ (UNIT=15, FMT=100, ERR=999, IOSTAT=IOS)NUMU
C***
     READ CPUS TYPES
     DO 6 I=1.NUMU
     READ (UNIT=15, FMT=101, ERR=999, IOSTAT=IOS)UNAME(I),
                 USPEED(I), UCOST(I)
     1
6
     CONTINUE
     PRINT *, *
     PRINT *,"
***
C*** CPU S
C***
     QUERY USER ON TYPE OF CPUS
     PRINT *, WHICH TYPE OF CPU WOULD YOU LIKE
7
     DO 8 I=1,NUMU
     PRINT *, 1, \cdot, \cdot, uname(1), \cdot speed = \cdot, uspeed(1),
         COST = , UCOST(I)
     1
8
     CONTINUE
      PRINT *, * *
     PRINT *, ENTER TYPE (1,2,..) :
     READ *, ICPU
```

\_\_\_\_\_

```
IF ((ICPU .LT. 1) .OR. (ICPU .GT. NUMU)) THEN
       PRINT *, "ILLEGAL CPU TYPE"
       GOTO 7
     END IF
     PRINT *, ~ ~
C***
     QUERY USER ON NUMBER OF CPUS
55
     PRINT *, 'ENTER NUMBER OF CPUS DESIRED (MAX 99): '
     READ *, NCPUS
     IF ((NCPUS .GT. 99) .OR. (NCPUS .LT. 1)) THEN
       PRINT *, 'ILLEGAL NUMBER OF CPUS'
       GO TO 55
     END IF
     PRINT *, - -
C***
    QUERY USER ON TIME SLICE
56
     PRINT *, ENTER TIME SLICE (IN SECONDS): 1
     READ *, TSLICE
     IF ((TSLICE .LT .0001) .OR. (TSLICE .GT. 99999.)) THEN
      PRINT *, TIME SLICE RANGE .0001 TO 99999. SECONDS'
      GOTO 56
     END IF
     PRINT *. - -
     PRINT *, ~
C***
     MEMORY PARTITIONS
                                   ***
C***
     QUERY USER ON NUMBER OF MEMORY PARTITIONS
     PRINT *, ENTER NUMBER OF MEMORY PARTITIONS DESIRED (MAX 20)
9
     READ *, NPARTS
     IF ((NPARTS .GT. 20) .OR. (NPARTS .LE. 0)) THEN
       PRINT *, 'ILLEGAL NUMBER OF MEMORY PARTITIONS'
       GOTO 9
     END IF
     PRINT *, - -
C***
     QUERY USER ON SIZE OF MEMORY PARTITIONS
     PRINT *, ENTER PARTITION SIZES IN K
     DO 10 I=1,NPARTS
       PRINT *, 'PARTITION(',I,') SIZE (IN K) : '
       READ *, PART(I)
       PRINT *,
10
     CONTINUE
     PRINT *. - -
     PRINT *, ~ ~
C*** CHANNELS
                                            ***
```

```
C*** QUERY USER ON NUMBER OF CHANNELS
14
     PRINT *, 'ENTER NUMBER OF CHANNELS DESIRED (MAX 5) :'
     READ *, NCHAN
     IF ((NCHAN .LE. 0) .OR. (NCHAN .GT. 5)) THEN
        PRINT *, 'ILLEGAL NUMBER OF CHANNELS'
        GO TO 14
     END IF
     DO 13 K=1,NCHAN
        PRINT *, - -
C***
        QUERY USER ON TYPE OF CHANNELS
11
     PRINT *, WHICH TYPE WOULD YOU LIKE FOR CHANNEL(',K,'): '
        DO 12 I=1,NUMC
          PRINT *, I, . . , CNAME(I), DATA RATE = .,
            CSPEED(I), COST = , CCOST(I)
    1
12
        CONTINUE
        PRINT \star, ENTER TYPE (1, 2, ...)
        READ *, ICHAN
        IF ((ICHAN .LT. 1) .OR. (ICHAN .GT. NUMC)) THEN
           PRINT *, 'ILLEGAL CHANNEL TYPE'
           GOTO 11
        END IF
        ICHANL(K) = ICHAN
13
     CONTINUE
     PRINT *. 1
     PRINT *
***
C*** DISKS
C***
     QUERY USER ON NUMBER OF DISKS
15
     PRINT *, ENTER NUMBER OF DISKS/DRUMS DESIRED (MAX 10) :
     READ *, NDISK
     IF ((NDISK .LE. 0) .OR. (NDISK .GT. 10)) THEN
        PRINT *, ILLEGAL NUMBER OF DISKS'
        GO TO 15
     END IF
     DO 20 K=1,NDISK
        PRINT *, - -
C***
        QUERY USER ON TYPE OF DISKS
16
        PRINT *, WHICH TYPE OF DISK WOULD YOU LIKE'
        DO 17 I=1, NUMD
          PRINT *,I, . . ,DNAME(I), SPEED = .,DSPEED(I),
            COST = , DCOST(I)
    1
17
        CONTINUE
        PRINT *, 1
```

D-94

Ę

....

PRINT \*, ENTER TYPE (1,2,..) FOR DISK ',K,': ' READ \*, IDSK IF ((IDSK .LT. 1) .OR. (IDSK .GT. NUMD)) THEN PRINT \*, "ILLEGAL DISK TYPE" **GOTO 16** END IF IDISK(K, 6) = IDSKC\*\*\* GET CHANNEL CONNECTIONS PRINT \*, ENTER CHANNEL CONNECTIONS FOR DISK ',K PRINT \*, 0 - CHANNEL CONNECTIONS FOR DISK PRINT \*, 1 - CHANNEL CONNECTED TO DISK PRINT \*, 1 - CHANNEL CONNECTED TO DISK PRINT \*, DO 19 J=1,5 IF (J .LE. NCHAN) THEN 18 PRINT \*, CHANNEL ', J, ' (0/1): ' READ \*, JJ IF ((JJ .EQ. 0) .OR. (JJ .EQ. 1)) THEN IDISK(K,J) = JJELSE PRINT \*, 'ILLEGAL INPUT' **GOTO 18** END IF ELSE IDISK(K, J) = 0END IF 19 CONTINUE 20 CONTINUE PRINT \*, - -PRINT \*, \* C\*\*\* TAPES \*\*\* C\*\*\* QUERY USER ON NUMBER OF TAPES 21 PRINT \*, 'ENTER NUMBER OF TAPES DESIRED (MAX 10) :' READ \*, NTAPE IF ((NTAPE .LE. 0) .OR. (NTAPE .GT. 10)) THEN PRINT \*, ILLEGAL NUMBER OF TAPES' GO TO 21 END IF DO 26 K=1,NTAPE PRINT \*, 1 C\*\*\* QUERY USER ON TYPE OF TAPES 22 PRINT \*, WHICH TYPE OF TAPE WOULD YOU LIKE' DO 23 I=1, NUMT PRINT \*,I, `, TNAME(I), SPEED = `,TSPEED(I), COST = \$`,TCOST(I) 1

-

.....

```
23
        CONTINUE
        PRINT *.
        PRINT *, 'ENTER TYPE (1,2,..) FOR TAPE ',K, ': '
        READ *, ITAP
        IF ((ITAP .LT. 1) .OR. (ITAP .GT. NLMT)) THEN
           PRINT *, 'ILLEGAL TAPE TYPE'
           GOTO 22
        END IF
        ITAPE(K, 6) = ITAP
C***
        GET CHANNEL CONNECTIONS
        PRINT *, 'ENTER CHANNEL CONNECTIONS FOR TAPE ',K
        PRINT *, 0 - CHANNEL NOT CONNECTED TO TAPE
        PRINT *, 1 - CHANNEL CONNECTED TO TAPE
        PRINT *, · ·
        DO 25 J=1,5
         IF (J .LE. NCHAN) THEN
24
           PRINT *, 'CHANNEL ', J, ' (0/1): '
           READ *, JJ
           IF ((JJ .EQ. 0) .OR. (JJ .EQ. 1)) THEN
              ITAPE(K,J) = JJ
             ELSE
              PRINT *, 'ILLEGAL INPUT'
              GOTO 24
           END IF
         ELSE
           ITAPE(K,J) = 0
         END IF
25
        CONTINUE
26
     CONTINUE
     PRINT *, -
     PRINT *, *
C*** CARD READER
                                                ***
C***
        QUERY USER ON TYPE OF CARD READER
        PRINT *, WHICH TYPE OF CARD READER WOULD YOU LIKE"
27
        DO 28 I=1,NUMR
          PRINT *, I, '. ', RNAME(I), ' SPEED = ', RSPEED(I),
            \cos t = , RCOST(1)
    1
28
        CONTINUE
        PRINT *, ~ ~
        PRINT *, ENTER TYPE (1,2,..) OF CARD READER DESIRED ~
        READ *, IRDR
        IF ((IRDR .LT. 1) .OR. (IRDR .GT. NUMR)) THEN
           PRINT *, 'ILLEGAL CARD READER TYPE'
           GOTO 27
        END IF
        IREAD(6) = IRDR
```

```
C***
        GET CHANNEL CONNECTIONS
        PRINT *, 'ENTER CHANNEL CONNECTIONS FOR CARD READER'
        PRINT *, 0 - CHANNEL NOT CONNECTED TO CARD READER
        PRINT *,
                  1 - CHANNEL CONNECTED TO CARD READER
               . .
        PRINT *.
        DO 30 J=1,5
         IF (J .LE. NCHAN) THEN
29
           PRINT *, CHANNEL ', J, (0/1): '
           READ *, JJ
           IF ((JJ .EQ. 0) .OR. (JJ .EQ. 1)) THEN
              IREAD(J) = JJ
             ELSE
              PRINT *, 'ILLEGAL INPUT'
              GOTO 29
           END IF
         ELSE
           IREAD(J) = 0
         END IF
30
        CONTINUE
     PRINT *. *
     PRINT *, · ·
C*** LINE PRINTER
                                                ***
C***
        QUERY USER ON TYPE OF LINE PRINTER
        PRINT *, WHICH TYPE OF LINE PRINTER WOULD YOU LIKE"
31
        DO 32 I=1,NUMP
          PRINT *, I, . . , PNAME(I), SPEED = ., PSPEED(I),
            COST = $`,PCOST(I)
    1
32
        CONTINUE
        PRINT *. -
        PRINT *, ENTER TYPE (1,2,..) OF LINE PRINTER DESIRED *
        READ *, IPNTR
        IF ((IPNTR .LT. 1) .OR. (IPNTR .GT. NUMP)) THEN
           PRINT *, 'ILLEGAL LINE PRINTER TYPE'
           GOTO 31
        END IF
        IPRINT(6) = IPNTR
C***
        GET CHANNEL CONNECTIONS
        PRINT *, 'ENTER CHANNEL CONNECTIONS FOR LINE PRINTER'
        PRINT *, 0 - CHANNEL NOT CONNECTED TO LINE PRINTER
        PRINT *,
                 1 - CHANNEL CONNECTED TO LINE PRINTER
               . .
        PRINT *.
        DO 34 J=1,5
         IF (J .LE. NCHAN) THEN
           PRINT *, CHANNEL ', J, ' (0/1): '
33
          READ *, JJ
           IF ((JJ .EQ. 0) .OR. (JJ .EQ. 1)) THEN
```

57

IPRINT(J) = JJELSE PRINT \*, 'ILLEGAL INPUT' GOTO 33 END IF ELSE IPRINT(J) = 0END IF 34 CONTINUE PRINT \*. - -PRINT \*, \* C\*\*\* SOFTWARE MONITOR C\*\*\* QUERY USER IF HE WANTS TO USE SOFTWARE MONITOR PRINT \*, DO YOU WISH TO USE THE SOFTWARE MONITOR (Y/N) READ \*, SANSWR IF (SANSWR .EQ. 'Y') THEN PRINT \*, ENTER STARTING TIME OF S/W MONITOR PRINT \*, 'DAY HOUR MINUTE SECOND' READ \*, ISDAY, ISHOUR, ISMIN, SSEC PRINT \*, ENTER STOPPING TIME PRINT \*, DAY HOUR MINUTE SECOND READ \*, JSDAY, JSHOUR, JSMIN, SSSEC 35 PRINT \*, YOU CAN MONITOR UP TO FIVE OF THE FOLLOWING QUEUES' PRINT \*, 1. INPUT 2. HOLD 3. EXEC 4. OUTPUT PRINT \*, 11. TAPE 1 12. TAPE 2 13. TAPE 3 14. TAPE 4" PRINT \*, 15. TAPE 5 16. TAPE 6 17. TAPE 7 18. TAPE 8' PRINT \*, 19. TAPE 9 20. TAPE 10 21. DISK 1 22. DISK 21 PRINT \*, 23. DISK 3 24. DISK 4 26. DISK 6" 25. DISK 5 PRINT \*, 27. DISK 7 28. DISK 8 29. DISK 9 30. DISK 10' PRINT \*, '31. CHAN 1 32. CHAN 2 33. CHAN 3 34. CHAN 4' PRINT \*, 35. CHAN 5 36. NO QUEUE DESIRED' PRINT \*, - -DO 36 I=1,5 PRINT \*, ENTER QUEUE NUMBER : \* READ \*, IQ IF (((IQ .GE. 1) .AND. (IQ .LE. 4)) .OR. 1 ((IQ.GE. 11) .AND. (IQ.LE. 36))) THEN ISWMQ(I) = IQELSE PRINT \*, 'ILLEGAL QUEUE' **GOTO 35** END IF CONTINUE 36 END IF PRINT \*, PRINT \*, \*

```
************
C*** HARDWARE MONITOR
                                                  ***
C***
     QUERY USER IF HE WANTS TO USE HARDWARE MONITOR
     PRINT *, DO YOU WISH TO USE THE HARDWARE MONITOR (Y/N)
     READ *, HANSWR
     IF (HANSWR .EQ. 'Y') THEN
        PRINT *, ENTER STARTING TIME OF H/W MONITOR
        PRINT *, DAY HOUR MINUTE SECOND
        READ *, IHDAY, IHHOUR, IHMIN, HSEC
        PRINT *, 'ENTER STOPPING TIME'
        PRINT *, DAY HOUR MINUTE SECOND
        READ *, JHDAY, JHHOUR, JHMIN, SHSEC
75
        PRINT *, 'ENTER H/W MONITOR SAMPLE RATE (IN SECONDS) '
        READ *,RATEHW
        IF (RATEHW .GT. 99999) THEN
           PRINT *, DATE RATE TO LARGE, MAX RATE IS 999997
           GOTO 75
        END IF
37
        PRINT *, YOU CAN HAVE TWO TIMERS AND FIVE COUNTERS'
        PRINT *, CONNECTED TO THE FOLLOWING DEVICES:
      PRINT *, 1. CPU
                            2. READER
                                        3. PRINTER'
      PRINT *, 11. TAPE 1
                                                    14. TAPE 4'
                           12. TAPE 2
                                        13. TAPE 3
      PRINT *, 15. TAPE 5
                           16. TAPE 6
                                        17. TAPE 7
                                                    18. TAPE 8'
      PRINT *, 19. TAPE 9
                           20. TAPE 10 21. DISK 1
                                                    22. DISK 2'
     PRINT *, 23. DISK 3
                           24. DISK 4
                                       25. DISK 5
                                                    26. DISK 6'
     PRINT *, 27. DISK 7
                           28. DISK 8
                                       29. DISK 9
                                                    30. DISK 10'
     PRINT *, '31. CHAN 1
                           32. CHAN 2
                                                    34. CHAN 4"
                                       33. CHAN 3
      PRINT *, '35. CHAN 5
                           36. NO DEVICE DESIRED'
        PRINT *, 1
        PRINT *, ENTER NUMBER OF DEVICE FOR TIMER(1): *
        READ *, ICNT
        IF (((ICNT .GE. 1) .AND. (ICNT .LE. 3)) .OR.
    1
            ((ICNT .GE. 11) .AND. (ICNT .LE. 36))) THEN
           IHTIM1 = ICNT
         ELSE
           PRINT *, 'ILLEGAL DEVICE'
           GOTO 37
        END IF
        PRINT *, ENTER NUMBER OF DEVICE FOR TIMER(2): *
        READ *, ICNT
        IF (((ICNT .GE. 1) .AND. (ICNT .LE. 3)) .OR.
    1
            ((ICNT .GE. 11) .AND. (ICNT .LE. 36))) THEN
           IHTIM2 = ICNT
         ELSE
           PRINT *, 'ILLEGAL DEVICE'
           GOTO 37
        END IF
        DO 38 I=1,3
```
```
PRINT *, ENTER DEVICE NUMBER OF COUNTER(',I,'): '
           READ *, ICNT
           IF (((ICNT .GE. 1) .AND. (ICNT .LE. 3)) .OR.
     1
               ((ICNT .GE. 11) .AND. (ICNT .LE. 36))) THEN
              IHWCNT(I) = ICNT
            ELSE
              PRINT *, ILLEGAL DEVICE
              GOTO 37
           END IF
38
        CONTINUE
     END IF
     PRINT *,
     PRINT *, - -
C*** WRITE DATA TO CONFIGUARATION FILE
                                                ***
C*** NUMBER OF CPUS
     WRITE (UNIT=11, FMT=100, IOSTAT=IOS, ERR=999)NCPUS
***
     TYPE OF CPU
     WRITE (UNIT=11, FMT=101, IOSTAT=IOS, ERR=999)
            UNAME(ICPU), USPEED(ICPU), UCOST(ICPU)
     1
C***
    TIME SLICE
     WRITE (UNIT=11, FMT=112, IOSTAT=IOS, ERR=999)TSLICE
C*** WRITE NUMBER OF MEMORY PARTITIONS
     WRITE (UNIT=11, FMT=100, IOSTAT=IOS, ERR=999)NPARTS
C*** WRITE SIZE OF MEMORY PARTITIONS
     DO 40 I=1, NPARTS
        WRITE (UNIT=11, FMT=104, IOS TAT=IOS, ERR=999)I, PART(I)
40
     CONTINUE
C*** WRITE NUMBER OF CHANNELS
     WRITE (UNIT=11, FMT=100, IOSTAT=IOS, ERR=999)NCHAN
C*** WRITE CHANNEL'S NAME, SPEED AND COST
     DO 41 I=1,NCHAN
        WRITE (UNIT=11, FMT=101, IOSTAT=IOS, ERR=999)
              CNAME(ICHANL(I)), CSPEED(ICHANL(I)), CCOST(ICHANL(I))
     1
41
      CONTINUE
C*** WRITE NUMBER OF DISKS
     WRITE (UNIT=11, FMT=100, IOSTAT=IOS, ERR=999)NDISK
C*** WRITE DISK'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
      DO 42 I=1,NDISK
        WRITE (UNIT=11, FMT=105, IOSTAT=IOS, ERR=999)
          DNAME(IDISK(I,6)), DSPEED(IDISK(I,6)), DCOST(IDISK(I,6)),
     1
           (IDISK(I,K),K=1,5)
     1
42
      CONTINUE
C*** WRITE NUMBER OF TAPES
     WRITE (UNIT=11, FMT=100, IOSTAT=IOS, ERR=999)NTAPE
C*** WRITE TAPE'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
      DO 43 I=1.NTAPE
        WRITE (UNIT=11, FMT=105, IOSTAT=IOS, ERR=999)
```

7

```
D-100
```

```
1
            TNAME(ITAPE(1,6)), TSPEED(ITAPE(1,6)), TCOST(ITAPE(1,6)),
     1
            (ITAPE(I,K),K=1,5)
43
      CONTINUE
C***
      WRITE CARD READER'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
      WRITE (UNIT=11, FMT=105, IOSTAT=IOS, ERR=999)
            RNAME(IREAD(6)), RSPEED(IREAD(6)), RCOST(IREAD(6)),
     1
            (IREAD(K), K=1, 5)
     1
C*** WRITE LINE PRINTER'S NAME, SPEED, COST AND CHANNEL CONNECTIONS
      WRITE (UNIT=11, FMT=105, IOSTAT=IOS, ERR=999)
            PNAME(IPRINT(6)), PSPEED(IPRINT(6)), PCOST(IPRINT(6)),
     1
            (IPRINT(K), K=1, 5)
C*** WRITE IF SOFTWARE MONITOR IS DESIRED
      WRITE (UNIT=11, FMT=106, IOSTAT=IOS, ERR=999)SANSWR
C*** WRITE START TIME, STOP TIME, AND QUEUES IF DESIRED
IF (SANSWR .EQ. 'Y') THEN
         WRITE (UNIT=11, FMT=107, IOSTAT=IOS, ERR=999)
     1
                 ISDAY, ISHOUR, ISMIN, SSEC
         WRITE (UNIT=11, FMT=108, IOSTAT=IOS, ERR=999)
                 JSDAY, JSHOUR, JSMIN, SSSEC
     1
         WRITE (UNIT=11, FMT=109, IOSTAT=IOS, ERR=999)(ISW MQ(I), I=1, 5)
      END IF
C*** WRITE IF HARDWARE MONITOR IS DESIRED
      WRITE (UNIT=11, FMT=106, IOSTAT=IOS, ERR=999) HANSWR
C*** WRITE START TIME, STOP TIME, TIMERS AND COUNTERS IF DESIRED
IF (HANSWR .EQ. 'Y') THEN
         WRITE (UNIT=11, FMT=110, IOSTAT=IOS, ERR=999)
     1
                 IHDAY, IHHOUR, IHMIN, HSEC
         WRITE (UNIT=11, FMT=111, IOSTAT=IOS, ERR=999)
     1
                 JHDAY, JHHOUR, JHMIN, SHSEC
         WRITE (UNIT=11, FMT=102, IOSTAT=IOS, ERR=999)RATEHW
         WRITE (UNIT=11, FMT=109, IOSTAT=IOS, ERR=999)
     1
                 IHTIM1, IHTIM2, (IHWCNT(I), I=1,3)
      END IF
100
      FORMAT(12)
101
      FORMAT(A20,1X,F10.4,1X,F8.2)
102
      FORMAT('H/W MON SAMPLE RATE', T22, F10.4)
103
      FORMAT(15)
      FORMAT('PARTITION ', 12, T22, F10.4)
104
      FORMAT(A20, 1X, F10.4, 1X, F8.2, 1X, 5(1X, I1))
105
106
      FORMAT(A1)
      FORMAT('S/W MONITOR START', T22, 3(14, 1X), F8.4)
107
108
      FORMAT('S/W MONITOR STOP', T22, 3(14, 1X), F8.4)
109
      FORMAT(5(12, 1X))
      FORMAT('H/W MONITOR START', T22, 3(14,1X), F8.4)
110
      FORMAT('H/W MONITOR STOP', T22, 3(14, 1X), F8.4)
111
112
      FORMAT('TIME SLICE', T22, F10.4)
      GOTO 800
990
      PRINT *, 'ERROR IN OPENING FILE IOSTAT = ', IOS
      GO TO 800
      PRINT *, ERROR IN READ IOSTAT = , IOS
999
```

D-101

800 CONTINUE STOP END

•

G

D-102

.

Appendix E

••••

Ö

CPESIM II

Instructor Manual



# Contents

. . . . .

7

.

T

La la La La La

List	of	Fi	gı	ır	e s	•	•	•	•	•	•	,	•	٠		•	•	•	•	•	•	•	•	•	•	•	•	E - 3
List	of	Ta	ьb	le	s	•	•	•	•	•	•	,	•	•		•	•	•	•		•	•	•	•	•	•	•	E – 4
1.0	In	tr	:00	duo	ct	ic	n		•	•			•	•		•	•	•	•		•	•	•	•	•	•	•	E – 5
		1.	1	C	PE	SI	M	I	Ι	0	u t	: D	ut		to	5	St	: u	ie	n t	t s			•				E-6
		1.	2	S	t u	d e	n	t	Ιr	מו	ut	: s	t	0		CP	E S	511	1	IJ	E			•			•	E - 7
		1.	3	S	t 11	d e	הי	t	Tr	ר די ח די	 11 f		fo	r	(	? <b>r</b>	ad	111	n o						_			E-8
		1	4	S					2	• P 2	n d		CP	- ज	s 1	ГM	1	Т	č	้ำก		1 2	+ 1	o n	•	÷		F-9
		1	5	יט ק	 	u C a 7		0 <del>-</del>			1.9		~ 1	2	0 . n		-	• •	0	<b>T</b> 1	u u	La		. •		•	•	F-9
		τ.		E é	a p	e I	•	01	8 4	1 11	12	a		U	11		•	•	•	•	•	•	•	•	•	•	•	E - 3
2.0	Wo	rk	10	рас	d	Ge	n	er	al	:1	o r	1	•	•		•	•	•	•		•	•	•	•	•	•	•	E-11
		2	1			1. 1	_			•_			_ ,					_										E-11
		2.	1	WO	οr	КI	. 0	ao		<b>P</b>	e c	:1	I 1	. C	a	[1	01	1	•	•	•	•	•	•	•	•	•	E-11
		2.	2	<b>P</b> 1	ro	gr	a		ES	ce	сu	10	10	n		•	:	•	•	•	•	•	•	•	•	٠	•	E-13
		2.	3	E	ve	nt		St	re	e a	m	F	11	e	]	SV	<b>S</b> 1	ſR	•		•	•	•	•	•	•	٠	E - 13
		2.	. 4	0	th	e r	•	CR	S	rr	C	)u	tŗ	u	t		•	٠	•	•	•	•	٠	٠	•	•	٠	E-11
3.0.	Co	nf	Eig	gui	ra	ti	lo	n	De	e f	ir	ıi	ti	. 0	n		•	•	•	4	•	•	•	•	•	•	•	E-28
		3	1	H	0 r	ፈъ	29	re	. / <	20	f t	- w	9 Y	• •		۰.	<u>_</u>			c f	- 11	<b>r</b> 0	г	) o f				F - 28
		3	2	ц. ц.		d 11	79	+		, D	л . Л		a i	. – . – .			2.	M/	20	4.4		r 0				•	•	E = 30
		2	2	це С.	a .		. a		-		u a	с С		רם י	w e 17 (		с г4	1		τ.	. 0	13		•	•	•	•	E 31
		э.		6	en	er	a	LI	ng	5	a	U	0.0	<b>U</b> .	с,	>	Γ 1		=	•		•	•	•	•	•	•	E-21
4.0.	Si	<b>m</b> u	114	at:	io	n	E	хe	cı	ıt	ic	n		•		•	•	•	•	•	•	•	•	•	•	•	•	E – 56
		4.	1	F	11	P	S	to	r	10	ρ							_										E - 56
		4	2	S	í m	1	2	+ i	0.1	- 0 1	C F v	• •	• • 11	-	1,	, . n	•	•				•			•			E-56
		4	2	м.	~ 4	4 6	 : .,,	1 -	~	•	ር ብ ጉወ		C 4		. 1	) <u>.</u> .	+ 4			, a		- т	•		•		•	E - 57
				ri (	u	τī	· y	<b>-</b> 1	8	L		•	91	щ	u I	d		. 01		n t		1			•	•	•	10.1
5.0.	Si	mυ	118	at:	10	n	0	u t	pι	ıt	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	٠	E-60
		5.	1	A 4	c c	011	10	t i	ne	,	Da	ıt	а	•													•	E-60
		5	2	S	o f	+ -	 7 @	re		s 1 n	n 1		- 0 r		D	. +	a		-									E - 60
		5	3	ц		7 -	79	 r -		 10			- L 0 F		יכ	 . +	а а		-			-		•				E = 60
				- 11 6						<b>4</b> U			~ -		~ ~		а.	٠	•	•		•	•	•	•	•		

E – 2

# List of Figures

1. ÷

Ċ

••••

Figu	ıre	Page
1.	CPESIM II System Structure	E-10
2.	Example of WRKDES File	E-26
3.	Cyber JCL to Execute CRSTR	E-27
4.	"User" Interface Procfil	E-53
5.	Typical TYPES File	E-54
6.	Typical Condes File	E – 5 5
7.	Cyber JCL to Load Dummy Files	E – 58
8.	Cyber JCL to Execute Simulation	E-59

# List of Tables

Ô

-

Tab	le									Page
1.	Workload Creation Input	• •	•	•	•	•	•	•	•	E-15
2.	Event Stream (EVSTR) Format .	• •	•	•	•	•	•	•	•	E - 2 2
3.	EVSTR Format For Job Records	• •	•	•	•	•	•	•	•	E-25
4.	Configuration Input	• •	•	•	•	•	•	•	•	E-32
5.	Types Input	• •	•	•	•	•	•	•	•	E-46
6.	ACTLOG Record Format	• •	•	•	•	•	•	•	•	E-61
7.	SFTMON Record Format	• •	•	•	•	•	•	•	•	E-62
8.	HRDMON Record Format		٠	•	•	•	•	•	•	E-63

## CPESIM II

#### Instructor Manual

## 1.0 Introduction

CPESIM II consists of a simulation of a major computer system and its operating environment. It was designed for use as a laboratory aid to allow students to apply Computer Performance Evaluation (CPE) tools and techniques to a "real" computer system. The use of an actual system for such studies is impractical for several reasons. First, due to the overhead and disruption caused by some measurement tools, their use in a real system for classroom purposes may be prohibited by computer center personnel. Second, if measurement data is available it may not be academically useful. That is, the system may be operating as it should (thus providing no problem solving opportunities), or it may contain several interacting problems which, although realistic, cannot practically be resolved by a student as a one-quarter course project. Third, if the data does allow the student to analyze a problem and recommend a solution, it is unlikely that the computer center will allow the implementation of the solution (or of several conflicting solutions from several students). CPESIM II solves these constraints be providing a simulated environment in which the student can gather whatever data is desired, analyze the data and recommend a solution, implement the solution (at an appropriate cost), and then analyze the modified system to verify whether or not the solution was effective.

CPESIM II consists of two parts. The heart of CPESIM II is a computer simulation (written in SLAM) of a large-scale computer system processing a workload, which executes on a host machine (e.g., the CYBER). Because it is a simulation, there are many simplifications and limitations which restrict its representation of reality. The second part of CPESIM II is a simulated operating environment for the computer system. This includes a computer-generated workload, a written scenario, budgetary constraints, and system data in a form that can be manipulated and analyzed by the student, playing the part of a CPE analyst and/or DP manager.

E – 5

The written scenario provides the student with operating details about the particular computer installation he or she is to investigate. The instructor configures the hardware, operating system, and workload to illustrate a particular problem or situation. Measurement data is then provided to the student as requested for analysis and decision making on a periodic basis (e.g., weekly). In order to force the student analyst to make tradeoffs, only a subset of the available data can be accessed during a given period, and an appropriate cost is associated with each student request.

## 1.1 CPESIM II Output To Students

•

Many types of information are available to the student to aid in the analysis. Some of these are free and automatic. Some must specifically be requested. There is a cost associated with many. Some are mutually exclusive. The student must choose what data he wants and how he wants to use it. This section briefly describes the outputs available and directs the student to further information.

Manufacturer's Data - Literature (of varying value) describing hardware and software features is available in printed form.

Installation Documentation - In-house documentation of the local configuration, parameter values, modifications, problems, etc., may be available in printed form. These are specific to a given problem and will be provided as a separate handout.

Accounting System Data - The computer's accounting system files are available at no cost to the student as disk files on the host computer. Such data is in raw form, and the format is defined in the manufacturer's literature for the appropriate operating system.

Hardware Monitor - A hardware monitor exists for the simulated computer. However, such a monitor may have to be purchased or leased. Particular probe points and sampling periods must be specified by the student. As with a real hardware monitor, there is no effect on system performance. The output data is available as a disk file on the host computer.

Software Monitor - A software monitor is available for the simulated computer. However, it may have to be purchased or leased. As with real software monitors, these monitors require memory, runtime, and other system overhead to operate. Output data is available as disk files on the host computer.

• • • •

Budget Reports - Periodic written summaries of the dollars spent will be available to the analyst.

## 1.2 Student Inputs to CPESIM II

•

One of the advantages of a simulation is that the students can make changes to the system. Thus, there are a number of inputs that the student may (or must) provide to CPESIM II (as well as some that he cannot). This section briefly describes the CPESIM II inputs available to the student.

Workload - Students have no control over the workload. The instructor does control the workload and may choose to vary it as the situation dictates (e.g., in response to a student initiated "user education program").

Interview Requests - Student analysts can submit written questions to installation personnel. Questions may or may not be answered. Charges for personnel time spent answering questions will be assessed to the student's account.

Monitor Purchase - Student analysts can submit purchase orders for the purchase of lease of any available hardware or software monitors.

Monitor Input Parameters - If the software monitor is desired then the students must provide the simulation the starting time, stopping time, and the queues to be monitored. If the hardware monitor is chosen then the starting time, stopping time, sample rate, counter probes, and timer probes must be specified. The student can input these parameters by using the "User" interface while making the configuration file.

Reconfiguration Requests - Students may request reconfiguration of existing hardware and operating system parameters. Personnel costs to reconfigure will be assessed. Degradation of system performance because of such changes will be frowned upon by installation management.

Purchase of System Options - Additional hardware or operating system modules may be purchased by submitting an appropriate purchase order. Costs will be assessed directly to the student account. Information on current cost and availability will be provided with each particular project.

Personnel Hiring - Requests for hiring additional installation personnel (e.g., for an additional shift) or for overtime authorization may be submitted. If approved, costs will be assessed directly to the student account.

## 1.3 Student Input For Grading

Of course, choice of grading technique is the prerogative of each individual instructor. However, this section describes one set of student inputs that may be used for that purpose.

Final Report - This represents the single most important input to the student's final grade. The report should be typewritten with drawings, tables, and graphs of professional quality. The report should consist of two parts. Part 1 is the Analyst's Report. This should include a clear statement of the problem as perceived, a systems analysis and description, stated hypothesis of the specific problem, analysis done to confirm (or deny) the hypothesis, recommended solution, and verification that the implemented solution solved the problem. Also, included should be a cost analysis and recommendations for the future. Part II is the Student's Critique. This should include an evaluation of the simulation as a learning tool along with a discussion of problems encountered and recommended changes.

Interim Reports - Various interim reports may be required throughout the quarter, depending on the scenario. These should be of the same quality as the final report and will count toward the project grade.

Software Tools - Some specific software tools (e.g., data reduction packages) may be required during the quarter. These will specifically count toward the project grade. For any other tools that the student might develop, the source code should also be submitted. These programs will influence the project grade.

Oral Presentation - Each team will be required to make a final oral presentation to the class and possibly an interim report as well. Although an important purpose of the oral report is to share each team's analysis and findings with the rest of the class, presentation is a graded part of the project and should be done in a

## professional manner.

## 1.4 Structure and CPESIM II Simulation

Figure 1 provides an overview of the CPESIM II system structure. The system can be divided into two sections. The first phase consists of generating two files, CONDES and EVSTR, needed by the simulation itself. CONDES is a file of parameters describing the simulated computer CONDES is created by a series of questions and system. answers from the user interface program (SETUP) prior to the simulation run. EVSTR is a file of synthetic job parameters, one record for each job, that provides a workload for the simulated system. Although EVSTR can come from anywhere (including the accounting log from a real computer), the program CRSTR provides a means of generating the workload from a set of workload description parameters, stored in file WRKDES. This program is very useful for generating artificial workloads for many uses other than CPESIM II. File WRKDES in turn, can be input from cards or from a permanent disk file.

The second part consists of the programs and files to execute the simulation itself. In the second phase the actual simulation programs, SIMS and SIMF, are used along with CONDES and EVSTR as input to generate performance data files ACTLOG and, optionally, HRDMON and SFTMON.

#### 1.5 Paper Organization

The remainder of this report discusses the two phases in greater detail. Chapters 2 and 3 discuss the workload generation and configuration definition of phase 1, respectively. Chapter 4 discusses the phase 2 execution of the simulation.



<u>с</u>.

Ø

•

Fig 1. CPESIM II System Structure

## 2.0 Workload Generation

This section describes the workload generation process. It should be noted that this procedure can be used independently of the rest of CPESIM II to generate workloads for any simulation such as statistical data analysis, simulated operating systems, etc. The output workload consists of a sequence of simulated "jobs" where each job consists of a job identifier, an arrival time, and a series of computer resource requirements (e.g., CPU time, memory, etc.). These job parameters are based on user specified frequency distributions and associated parameters for each resource. These distributions can be changed at any time throughout a standard 24 hour day. The system can generate up to seven days of workload, but the sequence of distributions must be identical for each day (i.e., each day is statistically the same).

### 2.1 Workload Specification

Table 1 defines the input card images necessary to generate a workload. The actual generation program (CRSTR) reads these images from a disk file with local filename WRKDES. Thus, the user can load WRKDES from an actual card deck, or can create or change this file interactively. A11 of the input fields on all the cards are in a free-form format. This means that the information on the input cards does not need to be in a particular column. However, the order of the data is critical. Most of the information required is formatted so that a key word indicates the contents of the fields which follow it. The end of a card terminates a value for a field, but has no other effects. All fields must be separated by at least one space. A11 numerical information is entered as "real" numbers. This means all numbers must have a decimal point and a zero. If a decimal point is not present, the code which edits the input will generally reject the number, or assume a decimal point - not necessarily where it ought to be. All input fields have default values which are used if an input value has an error. Otherwise all data fields in a card should be used. A card may be dropped from the input if it is not required. A card is the basic input unit, and it consists of a keyword followed by a series of data fields. These logical cards may actually occupy more than one physical card.

The first card in this file, which is optional, is the DAY card, depicted in Table 1A. This card merely specifies the number of days of workload to be generated. The second card is the SED card (Table 1B). This card specifies a seed for the random number generator. Program CRSTR uses this seed to generate several seeds for the different parameter distributions.

The third card of interest, shown in Table IC is the This card specifies a time (in decimal hours) at TIM card. which the workload generating distributions are to be changed. These records must be in ascending time sequence in the workload creation deck. The parameter cards are thus interspersed among the TIM cards. The parameters which are defined prior to the first TIM card go into effect at hour 0.00 and continue to be in control unless another record defining the parameter appears. If more than one definition of the same parameter appears, the last definition for that time period is used. Only distributions that are changed by cards following this one will actually be changed. Other distributions will continue as they were before the specified TIM card. Any number of TIM cards my be included in a single 24-hour day. This same pattern of distribution will be used for each day of the workload.

The final card type is the parameter card. One of these cards, described in Table 1D, is used to specify each distribution. The first field specifies the variable to which the distribution is applied. The parameter options for the first field are listed in Table 1E. The second field specifies the distribution type, and the following fields list the distribution parameters. The allowable distribution types and required parameters are listed in Table 1E. Some of the data on the logical cards is in the form of probability distributions. The distributions are sequences of numbers structured as follows:

probability-of-the value the value . .

probability-of-the value the value \*

The sequence of number pairs is followed by an asterisk to indicate the end of the sequence (all variable length sequences of information are followed by an asterisk). The sequence may be spread across as many physical cards as desired. There are two types of distributions which may be specified in this manner. They are a probability density function or a cumulative distribution function. If a probability density function is specified, then the probabilities must add up to 1.0. If a cumulative distribution function is specified, then the first entry must be for the zero probability (even it is zero), the probabilities must be specified in ascending order, and the last probability must be 1.0.

Figure 2 illustrates an example of the WRKDES file for generating one day's data. The interarrival time card with a mean of 481 minutes followed by a TIM 8.0 is used to keep any jobs from arriving before 8:00. Thereafter all distributions are defined. The TIM 16.0 followed by the INT card with mean of 481 results in no job arrivals after 1600.

## 2.2 Program Execution

The sequence of workload specification cards described in Section 2.1 must be presented to the generating program CRSTR as a disk file under local filename WRKDES by copying a card deck from INPUT or attaching an appropriate The generating program itself, CRSTR, must be permfile. attached, as well as the system SIMSCRIPT library SIM2LIB. The output file from CRSTR has local filename EVSTR, and should be requested as a permanent file and declared as an 80 character file to the Cyber Record Manager. The CRSTR program can then be executed. Finally, the output file EVSTR should be cataloged. Figure 3 illustrated the Cyber control cards used in this sequence. The relation of data and program files can be seen in Figure 1.

## 2.3 Event Stream File EVSTR

The output of the workload generating program is a file EVSTR which is basically a sequence of arriving jobs sorted in order of arrival time. However, this file is intended as an input event stream to the main CPESIM II program, and actually consists of three event types: POWER.UP, JOB.ARRIVAL, and SWITCH.OFF, as described in Table Each record is terminated by an "\*". The first record, 2. POWER.UP, and the last record, SWITCH.OFF, were used in the original CPESIM II, and are no longer used. They can be left in the file or be excluded as desired. The remaining records each represent an arriving job. The first field contains the character string JOB.ARRIVAL, and the remaining fields contain parameters for that job as defined in Table 2. The EVSTR file format is indicated in Table 3 for use by other programs. Note that this file is written by a SIMSCRIPT program, and arrival times less than one do not include a leading zero, hence the file may not be readable as is by a PASCAL program (if any jobs arrive prior to 1:00AM).

# 2.4 Other CRSTR Output

.

In addition to the event stream file, CRSTR also prints a workload description to the line printer (Cyber OUTPUT file). This information reflects the input WRKDES information, nicely formatted, as well as the total number of jobs created and the last day jobs arrived.

۰.

٠.

# Table 1-A WORKLOAD CREATION INPUT; DAY CARD (OPTIONAL)

. .

Ċ,

....

. . .

FIELD	CONTENTS	VALUE(S)	DEFAULT	COMMENTS
1	Record Type	DAY	(Req)	NOTE 1
2	Num of Days	Integer	7.0	NOTE 2

NOTE 1: Indicates a day card. Used to identify the number of days the workload is to be created.

NOTE 2: The number of days a workload is to be created for.

# Table 1-B WORKLOAD CREATION INPUT; SEED CARD (OPTIONAL)

FIELD	CONTENTS	VALUE(S)	DEFAULT	COMMENTS
1	Record Type	SED	(Req)	NOTE 1
2	SEED	REAL	0.0	NOTE 2

NOTE 1: Indicates a random number generator seed card. Used to specify different starting seed for workload random number generation.

NOTE 2: The starting seed to be used.

67

# Table 1-C WORKLOAD CREATION INPUT; TIME CARD (OPTIONAL)

FIELD	CONTENTS	VALUE(S)	DEFAULT	COMMENTS
1	Record Type	TIM	(Req)	NOTE 1
2	Cutoff/Start Hour	Real	0.0	NOTE 2

NOTE 1: Indicates a time change input card. Used to delimit sections of a day when distrobutions apply.

-

NOTE 2: The hour at which the distributions prior to this card are replaced by the distributions which follow it. If a distribution is not replaced, it continues to control the output parameters. All distributions are initialized to default values at hour 0.0. When the input deck is read, the time of day is initialized to 0.0.

# Table 1-D WORKLOAD CREATION INPUT; PARAMETER CARDS

FIELD	CONTENTS	VALUE(S)	DEFAULT	COMMENTS
1	REC TYPE	See Table 1-E	(Required)	
2	Distribution Name	See Table 1-F	EXPONENT	NOTE 1
3	Distribution Parameters	See Table l-F	1.0	
NOTE 1	The distributi indicated by t	on which govern the record type.	s the param	eter

G

Table 1-EPARAMETERVALUESFORFIELD 1OFPARAMENTERCARDS

5

21

۰.

\_\_\_\_

.

PARAMETER NAME	COMMENTS
INT	Interarrival time in minutes
CPU	CPU time in seconds
MEM	Memory required in 1K blocks
PR I	Priority in integer values greater than l. Higher priority = lower value.
MTS	Number of allocatable devices
CRD	Number of input cards
LIN	Number of output lines
DIS	Number of system disk blocks
TAP	Number of allocated device blocks.

# Table 1-F DISTRIBUTIONS AVAILABLE FOR WORKLOAD DESCRIPTION

. .

•••

....

6

DISTRO NAME	PARAMETERS	COMMENTS
BETA	XPOWER X-1POWER	Generates a beta dísto- buted real numbers where
		XPOWER = power of x and X-1POWER = power of (1-x)
BINOMIAL	TRIALS PROBJUC	Generates integers repre- senting the number of suc- cess in TRAILS independent trials when the probabil- ity of success is PROBSUC.
ERLANG	MEAN K	Generates Erlang distri- buted numbers with MEAN and K
EXPONENT	MEAN	Generates Exponentially distributed numbers with MEAN.
GAMMA	MEAN K	Generates a Gamma distibuted numbers with MEAN, and H
LOGNOR MA L	MEAN ATDDEV	Generates Lognormally dis- tributed numbers with MEAN and standard deviation STDDEV.
NORMAL	MEAN STDDEV	Generates Normally distri- buted numbers with MEAN and standard deviation, STDDEV

# Table 1-F DISTRIBUTIONS AVAILABLE FOR WORKLOAD DESCRIPTION (CONTINUED)

Ó

DISTRO NAME	PARAMETERS	COMMENTS
POISSON	MEAN	Generates Poisson distri- buted numbers with MEAN.
UNIFORM	L"AL HVAL	Generates Uniformly distr buted numbers form LVAL t HVAL.
WEIBULL	SHAPE SCALE	Generates numbers from a Weibull distribution with SHAPE parameter and a SCA parameter.
STEP	VPROB VAL	Generates numbers from th values specified by VAL with probabilities VPROB. The VPROB values must for a probability distributio function or cumulative di tribution function. A * must terminate the string of numbers.
LSTEP	VPROB VAL	Generates numbers linearl interprated from the valu specified by VAL with pro abilities VPROB. The VPR values must form a valid cumulative distribution function. A * must termi ate the string of numbers

# Table 2-A EVENT STREAM (EVSTR) FORMATPOWER UP EVENT (Note 1)

•

्र**ः** •्र

5

FIELD	CONTENTS	VALUE(S)	COMMENTS
1	Event Type	POWER.UP	Indicates the starting time of the computer
2	Start Up Time	Real = O	The time at which the computer is to start processing in hours
3	Record Term.	*	
NOTE 1: the	This card is n	o longer u	sed by CPESIM II to start
	computer. It can be left in	will not a the EVSTR	ffect the simulation an! file.

# Table 2-B EVENT STREAM (EVSTR) FORMATJOB ARRIVAL EVENT

۰.

1

67

FIELD	CONTENTS	VALUE(S)	COMMENTS
1	Event Type	JOB • ARRIV	AL Indicates arrival of a job for the computer to process.
2	Arrival time	Real O	Arrival time in hours.
3	Job Name	Integer	Unique job indentifier.
4	CPU Time	Integer	CPU time required in seconds.
5	Memory	Integer	Blocks of memory needed.
6	Priority	Integerl	Job priority. Lower value = higher priority.
7	Allocatable Devices	Integer	Number of devices required.
8	Cards	Intger	Number of input cards.
9	Lines	Integer	Number of lines outputed
10	Disk Blocks	Integer	Number of disk blocks job reads/writes.
11	Allocatable Device Blocks	Integer	Number of blocks job reads/writes to allocat- ed devices.
12	Record Term	*	

E-23

 $\mathcal{A} = \mathcal{A} + \mathcal{A} + \mathcal{A}$ 

# Table 2-C EVENT STREAM (EVSTR) FORMATSWITCH OFF EVENT (Note 1)

FIELD	CONTENTS	VALUE(S)	COMMENTS
1	Event Type	SW ITCH.OFF	Indicates the stopping time of the computer
2	Start Up Time	Real = 0 1	The time at which the computer is to stop processing in hours
3	Record Term.	*	

NOTE 1: CPESIM II no longer uses this card to process the end of the job stream. The simulation ignores this card and can be left in EVSTR.

-

•••••

Table 3 EVSTR Forma	t F	or .	Job	Records
---------------------	-----	------	-----	---------

FIELD	VARIABLE	FORMAT
1	JOB.ARR IVAL	Al2 (left justified)
2	Arrival Time	F10.4,1X
3	Job Name	110,1X
4	CPU Time	13,1X
5	Memory	I3,1X
6	Priority	12,1X
7	Alloc. Devices	12,1X
8	Cards	I4,1X
9	Lines	15,1X
10	Disk Blocks	I4,1X
11	Alloc. Device Blocks	I4,1X
12	*	1A

6

.

. . . .

SED 680 DAY 1.0 INT STEP 1.0 481.0 \* **TIM 8.0** INT EXPONENT 3.0 CPU EXPONENT 160.0 MEM LSTEP 0.0 0.1 10.0 16.0 0.19 20.C 0.19 25.0 0.029 30.0 0.32 35.0 0.38 50.0 0.57 63.0 0.76 70.0 0.76 100.0 0.855 124.0 0.95 150.0 0.95 190.0 1.0 205.0 × PRI STEP 1.0 2.0 \* MTS STEP 1.0 0.0 \* CRD EXPONENT 10.0 LIN EXPONENT 10.0 DIS EXPONENT 1000.0 TAP STEP 1.0 0.0 \* TIM 13.0 CPU EXPONENT 500.0 TIM 16.0 INT STEP 1.0 481.0 \*

•.•

Fig 2. Example of WRKDES File



Ġ

Fig 3. Cyber JCL to Execute CRSTR

## 3. Configuration Definition

In addition to an input workload, the CPESIM II simulator also needs to know the current system configuration. This section describes how the user specifies the configuration. The simulation program uses a configuration file with the local name CONDES. This file can be entered from a remote terminal. The CONDES file contains information on the hardware devices (disks, CPU, etc.), hardware interconnections, operating system parameters (partition sizes, etc), hardware monitor connections, and software monitor parameters. This section discusses each of these areas.

## 3.1 Hardware/Software Architecture Definition

The first records of interest in the CONDES file define the hardware, operating system parameters, and interconnections of the computer system being simulated. These records are defined in Table 4A through 4K, and must be entered in the file in the order listed in Table 4L. These records, as indicated in the first field of each, deal with the CPU, with memory, with each IOC (I/O channel), and with each I/O device.

The first three records deal with the cpu. The first record indicates the number of cpus that will be in the simulation. The second contains the cpu name, the cpu ratio which is used to simulate relative speeds of different processors, and the cpu cost. If the ratio has a value of 1.0 each job will require the same amount of CPU time as is specified in that job's resource requirement in the EVSTR file. A faster processor would be simulated by using a CPU ratio of a less than 1.0. Thus this parameter is the ratio of the actual processing time to the requested processing time. Typically this value would be initially set to 1.0, then varied on later runs to reflect replacement of the CPU with a faster or slower (and cheaper) CPU. The third record, sets the round robin time quantum. This parameter sets the number of seconds that each job will execute before yielding the CPU to another job. Note that if a job requests an I/O before this time is reached it will yield the CPU at that time. By setting this parameter to a very large number, FIFO service can be modeled. CAUTION: Since this simulation distributes disk and tape I/Os evenly across a job's runtime, if the average time between I/O requests is less than the round robin time quantum, then for most jobs the latter parameter becomes irrelevant, and the system will behave as though it uses FIF0.

The second set of records describes the memory partitions of the simulated computer. The first card lists the number of memory partitions that the system will have (maximum of 20 partitions). The number of memory partitions also determines the multiprogramming level. Each job (except the job scheduler) is allocated a memory partition prior to being released into the execute queue. Since the job scheduler is always core resident in system memory, the multiprogramming level is one plus the number of memory partitions. After the number of memory partitions is entered, the size of each partition is entered (one per record). The size of the memory partitions are entered in terms of 1024 byte blocks. The order of the memory partitions IS important. When the job scheduler executes it tries to fit the job in the first free memory partition. If the partition is not large enough it looks at the next partition, followed by the next, until a partition of sufficient size is found or there are no more available partitions. Because of this searching algorithm the partitions should be listed in ascending order by size. By having them listed in this way, the smallest available partition will be used for each particular job. Also note, if a job requires more memory than is available in the largest memory partition, the job will sit in the hold queue until the end of the simulation.

The next set of records define the Input/Output Controllers (IOCs). The first record defines the number of controllers that the simulation will have. A maximum of five IOCs can be defined in any one simulation run. The next records specify the IOC name, the data transfer rate, and the cost. The rate will be in terms of standard data blocks per second. The data rate must be specified as an integer number of data blocks since fractions of blocks are not allowed. The IOC data transfer rate is the maximum number of blocks that an IOC can handle from I/O devices at one particular time. For example, if an IOC with a transfer speed of 100 blocks per second is connected to three devices with device transfer speeds of 40 blocks per second, two of those devices could be active simultaneously with no delay, while a transfer over the third device would have to put in the IOC queue.

After the definition of the IOCs, the I/O devices are defined. The first group of I/O devices defined are unallocatable I/O devices (disks and drums). The first record will specify the number of unallocated devices (maximum of 10). The following records will define each individual I/O device. The first field is the name of the device. The second field is the data transfer rate. This field must be specified in terms of the amount of time (in

seconds) required to transfer a 1K block of data. The third field represents the cost of the device. The last five fields define the connection of the device with the IOCs. A "1" in field 1 will indicate that the device is connected to the the first channel, where as a "0" will indicate the device is not connected to the IOC. The next 4 fields represent IOCs 2 through 5, respectively. There must be one channel definition card for each allocatable device.

The next group of I/O device definition cards define the allocatable I/O devices (tapes). The allocatable I/O devices are defined in exactly the same manner as the unallocatable I/O devices.

The third type of I/O device defined is the card reader. The first field is the name of the card reader. The second and third fields define the speed and cost of the card reader. The card reader's speed is defined in terms of cards read per minute. The simulation then calculates the transfer rate by dividing the 1K block size by the card record size (80 bytes). This result is then truncated to an integer and divided into the card reader speed. The result is now in terms of how long it takes to load a 1K buffer. This result is then multiplied by 60 to get the transfer rate in seconds instead of minutes. The last five fields define the interface between the reader and the IOCs. The format for the channel connections is the same as the unallocatable devices.

The last I/O device defined is the line printer. The line printer's name, speed, and cost are represented by the first three fields. The line printer's speed is defined in terms of lines per minute. Like the card reader, the transfer rate of the printer is calculated by finding the number of 132 byte lines that can fit in the 1K buffer and dividing by the number of lines per minutes. This result is then converted to a transfer rate in terms of seconds. The last five fields define the channel connections in the same way as the previous I/O devices.

#### 3.2 Hardware and Software Monitors

The present configuration of the simulation requires that the software and hardware monitors be defined in the CONDES file. The software monitor requires four cards to define its parameters. A 'Y' in the first card tells the simulation if the software monitor is desired. If the software monitor is desired then the next three cards will define the monitor's parameters. (NOTE: If the software monitor is not desired then the next three cards are not inserted into the CONFIG file and the hardware monitor cards follow the 'N' card for the software monitor.) The second and third cards define the starting and stopping time of the monitor. These cards have four fields which correspond to the desired day, hour, minute and second for the starting or stopping time of the monitor. NOTE: The starting time is the time at which the monitor is entered into the hold queue. The actual monitor trace will not start until the monitor is executed by the cpu. The fourth card defines the five queues which will be monitored. The software monitor can monitor any queue in the simulated system. The number of the desired queue is entered into each of the five fields. The queue numbers are listed in Table 4-J. If less than five queues are to be monitored any illegal queue number can be substituted in one of the queue fields.

The definition of the hardware monitor is similar to the software monitor. The first card tells the simulation if the hardware monitor is desired. If the monitor is desired the next two cards specify the starting and stopping times. The last card defines the two timer probes and the three counter probes. These probes can be connected to the cpus, IOCs and I/O devices. If one or more probes is not desired, then an illegal probe connection point is entered into that probe's field. Legal probe points are listed in Table 4-K.

#### 3.3 Generating a CONDES File

57

The CONDES file must have the records in the proper format and in the order listed in Table 4-L. These records can be entered by hand or they can be generated by using the "User" interface. The file SETUP provides this "USER" interface. This file reads the hardware descriptions from a file called TYPES. From the data provided by TYPES, the user will be queried about the types and quantities of various pieces of equipment. From the user's responses a CONDES file will be generated. This file can then be used as input to the simulation. Figure 4 shows the PROCFIL used to execute the "User" interface. The file TYPES must be stored on disk prior to its being called. Typically the instructor will control the TYPES file. He would list all the scenario hardware in this file. By controlling the file, the students can choose only "legal" pieces of equipment from which to define their system. The format for the TYPES file is listed in Table 5-A through 5-F. The order of the cards must be as specified in Table 5-G. Figure 6 illustrates the contents of a typical CONDES file generated by SETUP using the TYPES file listed in Figure 5.

# Table 4-A CONFIGURATION INPUT; CPU CARDS

# NUMBER OF CPUS CARD

••••

57

.

FIELD	VARIABLE	FORMAT	COMMENTS
1	Num of CPUs	12	

# CPU DESCRIPTION CARD

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of CPU	A20,1X	
2	CPU Ratio	F10.4,1X	Relative CPU speed. .5 twice as fast as 1.0
3	CPU Cost	F8.2,1X	Cost of each CPU in \$.

# CPU TIME SLICE CARD

FIELD	VARIABLE	FORMAT	COMMENTS
1	TIME SLICE	A20,1X	Indicates time slice card
2	Time Slice	F10.4	Round robbin time quantum.

## Table 4-B CONFIGURATION INPUT; MEMORY CARDS

۰.

67

· · ·

# NUMBER OF MEMORY PARTITIONS

FIELD	VARIABLE	FORMAT	COMMENTS
1	Number of	Ι2	Maximum of 20 memory
	Partitions		partitions

# MEMORY PARTITION SIZE CARD (One for Each Partition)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Partition	A20,1X	
2	Partition Size	F10.4	Size in terms of 1024 byte blocks quantum.
## Table 4-C CONFIGURATION INPUT; CHANNEL CARDS

· · · ·

#### NUMBER OF CHANNELS

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of Channels	I 2	Maximum of	5 channels

# CHANNEL DESCRIPTION CARD (One per Channel)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Channel	A20,1X	
2	Channel Data Rate	F10.4,1X	Number of 1K blocks channel can transfer in in one second.
3	Channel Cost	F8.2,1X	Cost of channel in \$.

## Table 4-D CONFIGURATION INPUT; DISK CARDS

inde hara -

5

## NUMBER OF DISKS

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of Disks	12	Maximum of 10 disks	

## DISK DESCRIPTION CARD (One per Disk)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Disk	A20,1X	
2	Disk Data Rate	F10.4,1X	Time in seconds to transfer a lK block of data.
3	Disk Cost	F8.2,1X	Cost of disk in \$.
4	Channel l Connection	11,1X	l indicates disk con- nected to channel l
5	Channel 2 Connection	11,1X	l indicates disk con- nected to channel 2
6	Channel 3 Connection	11,1X	l indicates disk con- nected to channel 3
7	Channel 4 Connection	11,1X	l indicates disk con- nected to channei 4
8	Channel 5 Connection	11,1X	l indicates disk con- nected to channel 5

### Table 4-E CONFIGURATION INPUT; TAPE CARDS

.

#### NUMBER OF TAPES

FIELD	VARIABLE	FORMAT	COMMENTS
1	Number of Tapes	12	Maximum of 10 tapes

## TAPE DESCRIPTION CARD (One per Tape)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Tape	A20,1X	
2	Tape Data Rate	F10.4,1X	Time in seconds to transfer a lK block of data.
3	Tape Cost	F8.2,1X	Cost of tape in \$.
4	Channel l Connection	11,1X	l indicates tape con- nected to channel l
5	Channel 2 Connection	11,1X	l indicates tape con- nected to channel 2
6	Channel 3 Connection	11,1X	l indicates tave con- nected to channel 3
7	Channel 4 Connection	11,1X	l indicates tape con- nected to channel 4
8	Channel 5 Connection	11,1X	l indicates tape con- nected to channel 5

E-36

## Table 4-F CONFIGURATION INPUT; CARD READER DESCRIPTION CARD

٠

57

•

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Card Reader	A20,1X	
2	Card Reader Data Rate	F10.4,1X	Number of cards read per minute
3	Card Reader Cost	F8.2,1X	Cost of reader in \$.
4	Channel l Connection	11,1X	. 1 indicates connected to channel 1
5	Channel 2 Connection	11,1X	l indicates connected to channel 2
6	Channel 3 Connection	11,1X	l indicates connected to channel 3
7	Channel 4 Connection	11,1X	l indicates connected to channel 4
8	Channel 5 Connection	11,1X	l indicates connected to channel 5

## Table 4-G CONFIGURATION INPUT; LINE PRINTER DESCRIPTION CARD

1

F

57

••••

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Line Printer	A20,1X	
2	Line Printer Data Rate	F10.4,1X	Number of lines printed per minute
3	Line Printer Cost	F8.2,1X	Cost of printer in \$.
4	Channel l Connection	11,1X	l indicates connected to channel l
5	Channel 2 Connection	11,1X	l indicates connected to channel 2
6	Channel 3 Connection	11,1X	l indicates connected to channel 3
7	Channel 4 Connection	11,1%	l indicates connected to channel 4
8	Channel 5 Connection	11,1X	l indicates connected to channel 5

E-38

#### Table 4-H CONFIGURATION INPUT; SOFTWARE MONITOR CARDS

an in the factor of the factor

SOFTWARE MONITOR EXECUTION CARD

FIELD	VARIABLE	FORMAT	COMMENTS
1	S/W Monitor	A 1	'Y' indicates that the
	Desired		monitor is desired.
			'N' indicates not
			desired.

SOFTWARE MONITOR START UP CARD (Required if S/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Starting Day	T22,I4,1X	Starting day of run.
2	Starting Hour	14,1X	Starting hour of run.
3	Starting Min.	14,1X	Starting minute of run.
4	Starting Sec.	F8.4	Starting second of run.

## SOFTWARE MONITOR TERMINATION CARD (Required if S/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Stopping Day	T22,I4,1X	Stopping day of run.
2	Stopping Hour	14,1X	Stopping hour of run.
3	Stopping Min.	I4,1X	Stopping minute of run.
4	Stopping Sec.	F8.4	Stopping second of run.

## Table 4-H CONFIGURATION INPUT; SOFTWARE MONITOR CARDS (Continued)

57

SOFTWARE MONITOR QUEUE CARD (Required if H/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Queue #1	12,1X	First queue to be monitored. See Table 4-J
2	Queue #2	12,1X	Second queue to be monitored. See Table 4-J
3.	Queue #3	12,1X	Third queue to be monitored. See Table 4-J
4	Queue #4	12,1X	Fourth queue to be monitored. See Table 4-J
5	Queue #5	12,1X	Fifth queue to be monitored. See Table 4-J

### Table 4-I CONFIGURATION INPUT; HARDWARE MONITOR CARDS

· ·· ····· ·····

7

**T** 

.

### HARDWARE MONITOR EXECUTION CARD

FIELD	VARIABLE	FORMAT	COMMENTS
1	H/W Monitor Desired	A 1	'Y' indicates that the monitor is resired.
	bebired		N' indicates not
			desired.

#### HARDWARE MONITOR START UP CARD (Required if H/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Starting Day	T22,I4,1X	Starting day of run.
2	Starting Hour	14,1X	Starting hour of run.
3	Starting Min.	I4,1X	Starting minute of run.
4	Starting Sec.	F8.4	Starting second of run.

#### HARDWARE MONITOR TERMINATION CARD (Required if H/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Stopping Day	T22,I4,1X	Stopping day of run.
2	Stopping Hour	I4,1X	Stopping hour of run.
3	Stopping Min.	14,1X	Stopping minute of run.
4	Stopping Sec.	F8.4	Stopping second of run.

Table 4-I CONFIGURATION INPUT; HARDWARE MONITOR CARDS (Continue)

. . . .

eleiteisenen en en en en

HARDWARE MONITOR SAMPLE RATE CARD (Required if H/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Sampler Rate	T22,F10.4	How ofter the monitor will record data and clear counter and timers

#### HARDWARE MONITOR PROBE CARD (Required if H/W Monitor Card is 'Y')

FIELD	VARIABLE	FORMAT	COMMENTS
1	Timer #1	12,1x	First timer probe con- nection. See Table 4-K
2	Timer #2	12,1X	Second timer probe con- nection. See Table 4-K
3	Count #1	12,1X	First counter probe con- nection. See Table 4-K
4	Count #2	12,1X	2nd counter probe con- nection. See Table 4-K
5	Count #3	12,1X	Third counter probe con- nection. See Table 4-K

QUEUE #	QUEUE NAME	QUEUE #	QUEUE NAME
1.	INPUT	2.	HOLD
3.	EXEC	4.	OUTPUT
11.	TAPE 1	12.	TAPE 2
13.	TAPE 3	14.	TAPE 4
15.	TAPE 5	16.	TAPE 6
17.	TAPE 7	18.	TAPE 8
19.	TAPE 9	20.	TAPE 10
21.	DISK 1	22.	DISK 2
23.	DISK 3	24.	DISK 4
25.	DISK 5	26.	DISK 6
27.	DISK 7	28.	DISK 8
29.	DISK 9	30.	DISK 10
31.	CHAN 1	32.	CHAN 2
33.	CHAN 3	34.	CHAN 4
35.	CHAN 5		
36.	NO QUEUE DESIRED		

## Table 4-J CONFIGURATION INPUT; SOFTWARE MONITOR QUEUES

.

5

5

QUEUE #	QUEUE NAME	QUEUE #	QUEUE NAME
1.	<b>C P</b> U	2.	READER
3.	PRINTER	11.	TAPE 1
12.	TAPE 2	13.	TAPE 3
14.	TAPE 4	15.	TAPE 5
16.	TAPE 6	17.	TAPE 7
18.	TAPE 8	19.	TAPE 9
20.	TAPE 10	21.	DISK 1
22.	DISK 2	23.	DIS< 3
24.	DISK 4	25.	DISK 5
26.	DISK 6	27.	DISK 7
28.	DISK 8	29.	DISK 9
30.	DISK 10	31.	CHAN 1
32.	CHAN 2	33.	CHAN 3
34.	CHAN 4	35.	CHAN 5

Table 4-K CONFIGURATION INPUT; HARDWARE MONITOR PROBE POINTS

34.CHAN 4336.NO DEVICE DESIRED

٠,٢

E-44

## Table 4-L CONFIGURATION IN UT; ORDER OF INPUT CARDS

CARD NAME

``·

Ţ

1

COMMENTS

NUMBER OF CPUS				
CPU DESCRIPTION				
CPU TIME SLICE				
NUMBER OF PARTITIONS				
MEMORY PARTITION SIZE	ONE	PER	PART	TITION
NUMBER OF CHANNELS				
CHANNEL DESCRIPTION	ONE	PER	CHAN	INEL
NUMBER OF DISKS				
DISK DESCRIPTION	ONE	PER	DISK	
NUMBER OF TAPES				
TAPE DESCRIPTION	ONE	PER	TAPE	
CARD READER DESCRIPT.				
LINE PRINTER DESCRIPT.				
S/W MONITOR EXECUTION				
S/W MONITOR START UP	MANI	DITOF	LY IF	EXECUTION WAS
	INDI	ECATE	ED BY	Y'IN EXECUTION CARD
S/W MONITOR	MANI	DITOR	LY IF	EXECUTION WAS
TERMINATION	IND	ICATE	ED BY	Y'IN EXECUTION CARD
S/W MONITOR QUEUE	MANI	DITOR	Y IF	F EXECUTION WAS
	IND	ICATE	ED BY	Y' IN EXECUTION CARD
H/W MONITOR EXECUTION				
H/W MONITOR START UP	MANI	DITOR	RY IF	F EXECUTION WAS
	IND	ICATE	ED BY	Y'IN EXECUTION CARD
H/W MONITOR	MAN	DITOR	RY IF	EXECUTION WAS
TERMINATION	IND	ICATE	ED BY	Y'IN EXECUTION CARD
H/W MONITOR SAMPLE	$M \sim N$	DITOF	RY IF	F EXECUTION WAS
RATE	IND	ICATH	ED B	Y' IN EXECUTION CARD
H/W MONITOR PROBE	MAN	DITOF	RY IF	EXECUTION WAS
	TND	TCATE	D BY	Y' IN EXECUTION CARD

## Table 5-A TYPES INPUT; DISK CARDS

.

NUMBER OF DISKS TYPES

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of Disks Types	12		

# DISK DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Disk	A20,1X	
2	Disk Data Rate	F10.4,1X	Time in seconds to transfer a lK block of data.
3	Disk Cost	F8.2,1X	Cost of disk in \$.

### Table 5-B TYPES INPUT; TAPE CARDS

57

NUMBER OF TAPE TYPES

FIELD	VARIABLE	FORMAT	COMMENTS
1	Number of Tape Types	I 2	

# TAPE DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Tape	A20,1X	
2	Tape Data Rate	F10.4,1X	Time in seconds to transfer a lK block of data.
3	Tape Cost	F8.2,1X	Cost of tape in \$.

### Table 5-C TYPES INPUT; CHANNEL CARDS

## NUMBER OF CHANNEL TYPES

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of Channel Types	I2 s		

# CHANNEL DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Channel	A20,1X	
2	Channel Data Rate	F10.4,1X	Number of lK blocks channel can transfer in in one second.
3	Channel Cost	F8.2,1X	Cost of channel in \$.

#### Table 5-D TYPES INPUT; CARD READER CARDS

NUMBER OF CARD READER TYPES

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of	12		
	Reader Types			

## CARD READER DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Card Reader	A20,1X	
2	Card Reader Data Rate	F10.4,1X	Number of cards read per minute
3	Card Reader Cost	F8.2,1X	Cost of reader in \$.

#### Table 5-E TYPES INPUT; LINE PRINTER CARDS

• • • • • •

6

NUMBER OF LINE PRINTER TYPES

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of	T2		
-	Printer Type	с- с		

LINE PRINTER DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of Line Printer	A20,1X	
2	Line Printer Data Rate	F10.4,1X	Number of lines printed per minute
3	Line Printer Cost	F8.2,1X	Cost of printer in \$.

Table 5-F TYPES INPUT; CPU CARDS

67

NUMBER OF CPU TYPES CARD

FIELD	VARIABLE	FORMAT	COMMENTS	
1	Number of	12		
	CPU types			

# CPU DESCRIPTION CARD (One Per Type)

FIELD	VARIABLE	FORMAT	COMMENTS
1	Name of CPU	A20,1X	
2	CPU Ratio	F10.4,1X	Relative CPU speed. .5 twice as fast as 1.0
3	CPU Cost	F8.2,1X	Cost of each CPU in \$.

Table 5-G TYPES INPUT; ORDER OF INPUT CARDS

والمعارفة والمحارثة

.

•

. . .

CARD NAME COMMENTS NUMBER OF DISK TYPES DISK DESCRIPTION One Per Type NUMBER OF TAPE TYPES TAPE DESCRIPTION One Per Type NUMBER OF CHANNEL TYPES CHANNEL DESCRIPTION One Per Type NUMBER OF READER TYPES CARD READER DESCRIPTION One Per Type NUMBER OF PRINTER TYPES LINE PRINTER DESCRIPTION One Per Type NUMBER OF CPUS CPU DESCRIPTION One Per Type



•

Fig 4. "User" Interface Procfil

E – 53

3			
DISK 1	0.0328	1000.00	
DISK 2	0.0656	500.00	
DRUM 1	.0108	3000.00	
TAPE 1 2	0.0725	250.00	
CHANNEL 1	500.00	1100.00	
CHANNEL 2	200.00	620.00	
	80.00	20 40	
CARD READER 1	1000.00	30.40	
2	1000.00	400.00	
PRINTER 2	1000.00	500.00	
LASER PRINTER	4000.00	3000.00	
2			
CPU I CPU 2	1.00	1110.00	
LTU Z	• 7 2	4000.00	

Fig 5. Typical TYPES File

E - 54

. . •

67

•

2					
CPU 1	1.0000	1110.00			
TIME SLICE	1.0000				
5					
PARTITION 1	4.0000				
PARTITION 2	100.0000				
PARTITION 3	100.0000				
PARTITION 4	200.0000				
PARTITION 5	700.0000				
2					
CHANNEL 2	200.0000	620.00			
CHANNEL 1	500.0000	1100.00			
2				_	
DISK 2	.0656	500.00	1 1	0	0
DRUM I	.0108	3000.00	0 1	0	0
TAPE 1	.0725	250.00	1 1	0	0
CARD READER 2	1000.0000	400.00	1 0	Õ	Ō
LASER PRINTER	4000.0000	3000.00	1 0	Õ	0
Y				-	-
S/W MONITOR START	0 12	0 0.0	000		
S/W MONITOR STOP	2 0	0 0.0	000		
3 1 11 31 36					
N					

ļ

at a company

6

• .

Fig 6. Typical CONDES File

#### 4.0 Simulation Execution

With the WRKDES and CONDES files available, the actual simulation can be run. Section 5 discusses the simulation output. However, since the output files must be cataloged as part of the execution run and are typically printed at that time, the JCL associated with the simulation output is presented in this section.

#### 4.1 File Storage

In order to allow students to develop and use software data analysis tools, the out ut files from the simulation should be stored as perm files. This output consists of three files for each student group. Because of directory limitations under NOS-BE, the following approach is suggested to prevent loss of the simulation output due to a full directory. A dummy perm file is created for each student group before the simulation run. Note that each could specify its own protection password at this point if such security were felt to be needed. Then at the end of the simulation run, the output files are cataloged as cycles 2, 3, and 4 of this dummy file. Typically a retention time of 5 days might be specified. This accomplishes two things. First, it is necessary to delete cycles 2, 3, and 4 before the next run, as the cyber is limited to five cycles of any filename. Second, it forces the students to decide what data is really worth saving in machine form, copying and saving those files themselves, and allowing unused data to leave the system. Figure 7 shows the JCL to create dummy files for three student groups.

#### 4.2 Simulation Execution

The JCL for a simulation run is shown in Figure 8. It is necessary to attach the network portion of the model (SIMS), the discrete portion of the model (SIMF), as well as the CONDES file and the EVSTR file. In order to record data to the accounting file, the hardware monitor file, and the software monitor file, three files have to be requested for permanent file space. After the execution of the model, the three files will be permanently stored on the disk. The data is now ready for further data reduction.

Typically (in the CPE course) one run is made per week. During the first few weeks, only one run per class is made while the students gather sufficient data for decision making. Once the students want to change the configuration, or look at different variables with the monitors, then a separate run is required for each group. Because of the run times involved, the running of the program should be done by the instructor.

#### 4.3 Modifying the Simulation Run Time

The simulation is presently set up to run from 0 to 600,000 seconds of simulated time (approximately 7 days). If it is desired to run the simulation at sum other amount of time then the file SIMS has to be modified. The only card that has to be modified is the "INIT" card, which is the third card from the end of the file. The INIT card has the following format:

#### INIT, start-time, stop-time;

where start-time and stop- ime are real variables representing the precise second that the simulation is to start and stop.



.

.

57

Fig 7. Cyber JCL to Load Dummy Files

E-58

.

OWE, T70, IO200, CM220000. T830617, OWEN, 55533 ATTACH, PROC, SLAMPROC, ID=AFIT, SN=AFIT. ATTACH, SIMS, SIMS. ATTACH, SIMF, SIMF. ATTACH, TAPE11, TAPE11. REQUEST, TAPE12, \*PF. REQUEST, TAPE13, \*PF. REQUEST, TAPE14, \*PF. ATTACH, TAPE15, CONFIG. FTN5(I=SIMF,ANSI=0).BEGIN, SLAMII, PROC, M=LGO, PMD=1, PL=10000, MAP=PART, I=SIMS. CATALOG, TAPE12, ACTLOG. CATALOG, TAPE13, HR DMON. CATALOG, TAPE14, SFTMON. \* E OR \*EOF

Fig 8. Cyber JCL to Execute Simulation

#### 5.0 Simulation Output

.

There are three basic output files generated by SIMS and SIMF. This section briefly addresses these three files.

#### 5.1 Accounting Data

The accounting data is generated directly by every run of the simulation. This file is under the local filename of TAPE12. There is one record for each job that has gone through the system. The record content and format is listed in Table 6. At the end of the simulation the local accounting file (TAPE12) is cataloged as ACTLOG. The file is now ready for post processing by the student.

#### 5.2 Software Monitor Data

Software monitor data is generated only if the CONDES file has requested the software monitor. The monitor records the length of time a job spends in a queue, where the job came from and where the job is going after exiting the queue. The monitor will record the data for all the jobs that go through the five queues specified in the CONDES file. The software monitor file will be cataloged as SFTMON at the end of the simulation run. The record content and format is listed in Table 7.

#### 5.3 Hardware Monitor Data

Hardware monitor data is generated only if the CONDES file has requested the hardware monitor. The hardware monitor records the amount of time that a timer probe was energized, and the number of times that a counter probe was energized. The counters and timers are reset after every hardware monitor sample rate cycle. At the end of the simulation the file is cataloged as HRDMON. The record content and format is listed in Table 8.

## Table 6 ACTLOG Record Format

· · · · ·

6

.

FIELD	VARIABLE	FORMAT
1	Arrival Time	1X,F15.4,1X
2	Job Name	F10.0,1X
3	CPU Time	F5.0,1X
4	Memory	F5.0,1X
5	Priority	F5.0,1X
6	Alloc. Devices	F5.0,1X
7	Cards	F6.0,1X
8	Lines	F6.0,1X
9	Disk Blocks	F6.0,1X
10	Alloc. Device Blocks	F6.0,1X
11	Job Type (Note 1)	F2.0,1X
12	CPU Time Used	F10.3,1X
13	I/O Time Used	F10.3,1X
14	Memory Size Used	F10.3,1X
15	Departure Time	F15.4
	-	
Note 1:	ACTLOG only records user	r jobs so Job Type is
	a luave 1 0	

Table / SFIMUN Record Form
----------------------------

10

Ø

FIELD	VARIABLE	FORMAT
1	Queue Name	1X,A15,1X
2	Time in Queue	F8.3,1X
3	Where Job Came From	A15,1X
4	Where Job is Going	A 1 5



Table o habmon accold format	Table	8	HRDMON	Record	Format
------------------------------	-------	---	--------	--------	--------

į

FIELD	VARIABLE	FORMAT	
1	Time of Recording	1X,F15.4,1X	
2	Timer # 1	F8.3,1X	
3	Timer # 2	F8.3,1X	
4	Counter # 1	15,1X	
5	Counter # 2	15,1X	
5	Counter # 3	15	



Sopendix F

[]

語言であるというという。これでもないです。

6

-

C

CPESIM II

Student Manual

# Contents

1

•

List	of F	igures	<b>.</b>	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	F – 3
List	of T	ables	٠	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	F – 4
I.	Int	roduct	ion	L	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	F <del>-</del> 5
	С	PESIM	II	0u	tp	ut	t	0	St	ud	er	ıts	;	•	•	•	•	•	•	•	F – 7
	S	tudent	: In	1 p u	ts	t	0	CP	ΕS	IM	]	II	•	•		•	•	•	•		F-8
	S	tudent	: In	pu	t	fo	r	Gr	ad	in	g	•	•	•	•	•	•	•	•	•	F – 1 0
II.	ABC	BITBU	JCKE	T	Coi	m p	ut	e r	S	y s	te	en	•	•	•	•	•	•	•	•	F-12
	I	ntrodu	ıcti	on	t	0	t h	e	A B	C	Co	mp	ut	er				•	•	•	F-12
	Н	ardwan	ce S	pe	ci	fi	сa	ti	оп	L		•	•		•		•	•	•	•	F-13
	S	oftwar	e S	pe	ci	fi	сa	ti	0 1	L		•				•					F-19
	Α	ccount	ing	D	a t i	a															F-28
	S	oftwar	e M	lon	ite	o r															F - 29
	Н	ardwa	te M	lon	it	o r		•	•	•	•	•	•	•	•	•	•	•	•	•	F - 32
III	Mod	ifying	g th	e.	A B	с	Co	αp	ut	er	Ċ	Con	ii	Eur	a t	io	n	•	•	•	F-34
	E	xecuti	ing	t h	e	Co	nf	ig	ur	at	ic	n	Pı	cog	ra	n					F-34
	Υ	rouble	сн	00	+ + .			5						0							F-35

# List of Figures

Fig	ure											Page		
1.	Typical	Job	Flow	•	•		•		•		•	•	•	F-21

.

6

and the second secon

# List of Tables

57

.

1

i

Tab	le														Page
1.	ACTLOG Record Format	•	•	•	•	٠	•	•	•	•	•	•	•	•	F – 28
2.	SFTMON Record Format	•	•	•	•	•	•	•	•	•	•	•	•	•	F-30
3.	Possible SFTMON Names	•	•	•	•	•	•	•	•	•	•	•	•	•	F-31
4.	HRDMON Record Format	•	•	•	•	•	•	•	•	٠	•	•		•	F-33

#### CPESIM II Student Manual

#### I. Introduction

CPESIM II consists of a simulation of a major computer system and its operating environment. It was designed for use as a laboratory aid to allow students to apply Computer Performance Evaluation (CPE) tools and techniques to a "real" computer system. The use of an actual system for such studies is impractical for several reasons. First, due to the overhead and disruption caused by some measurement tools, their use in a real system for classroom purposes may be prohibited by computer center personnel. Second, if measurement data is available it may not be academically useful. That is, the system may be operating as it should (thus providing no problem solving opportunities), or it may contain several interacting problems which, although realistic, cannot practically be resolved by a student as a one-quarter course project. Third, if the data does allow the student to analyze a problem and recommend a solution, it is unlikely that the computer center will allow the implementation of the solution (or of several conflicting solutions from several students). CPESIM II solves these constraints be providing a simulated environment in which the student can gather

F – 5

whatever data is desired, analyze the data and recommend a solution, implement the solution (at an appropriate cost), and then analyze the modified system to verify whether or not the solution was effective.

CPESIM II consists of two parts. The heart of CPESIM II is a computer simulation (written in SLAM) of a large-scale computer system processing a workload, which executes on a host machine (e.g., the CYBER). Because it is a simulation, there are many siplifications and limitations which restrict its representation of reality. The second part of CPESIM II is a simulated operating environment for the computer system. This includes a computer-generated workload, a written scenario, budgetary constraints, and system data in a form that can be manipulated and analyzed by the student, playing the part of a CPE analyst and/or DP manager.

The written scenario provides the student with operating details about the particular computer installation he or she is to investigate. The instructor configures the hardware, operating system, and workload to illustrate a particular problem or situation. Measurement data is then provided to the student as requested for analysis and decision making on a periodic basis (e.g., weekly). In order to force the student analyst to make tradeoffs, only a

F - 6
subset of the available data can be accessed during a given period, and an appropriate cost is associated with each student request.

### CPESIM II Output To Students

Many types of information are available to the student to aid in the analysis. Some of these are free and automatic. Some must specifically be requested. There is a cost associated with many. Some are mutually exclusive. The student must choose what data he wants and how he wants to use it. This section briefly describes the outputs available and directs the student to further information.

Manufacturer's Data - Literature (of varying value) describing hardware and software features is available in printed form.

Installation Documentation - In-house documentation of the local configuration, parameter values, modifications, problems, etc., may be available in printed form. These are specific to a given problem and will be provided as a separate handout.

Accounting System Data - The computer's accounting system files are available at no cost to the student as disk files on the host computer. Such data is in raw form, and the format is defined in the manufacturer's literature for the appropriate operating system.

Hardware Monitor - A hardware monitor exists for the

F - 7

simulated computer. However, such a monitor may have to be purchased or leased. Particular probe points and sampling periods must be specified by the student. As with a real hardware monitor, there is no effect on system performance. The output data is available as a disk file on the host computer.

Software Monitor - A software monitor is available for the simulated computer. However, it may have to be purchased or leased. As with real software monitors, these monitors require memory, runtime, and other system overhead to operate. Output data is available as disk files on the host computer.

Budget Reports - Periodic written summaries of the dollars spent will be available to the analyst.

## Student Inputs to CPESIM II

One of the advantages of a simulation is that the students can make changes to the system. Thus, there are a number `f inputs that the student may (or must) provide to CPESIM II (as well as some that he cannot). This section briefly describes the CPESIM II inputs available to the student.

Workload - Students have no control over the workload. The instructor does control the workload and may choose to vary it as the situation dictates (e.g., in response to a student initiated "user education program").

F - 8

Interview Requests ~ Student analysts can submit written questions to installation personnel. Questions may or may not be answered. Charges for personnel time spent answering questions will be assessed to the student's account.

Monitor Purchase - Student analysts can submit purchase orders for the purchase of lease of any available hardware or software monitors.

Monitor Input Parameters - If the software monitor is desired then the students must provide the simulation the starting time, stopping time, and the queues to be monitored. If the hardware monitor is chosen then the starting time, stopping time, sample rate, counter probes, and timer probes must be specified. The student can input these parameters by using the "User" interface while making the configuration file.

Reconfiguration Requests - Students may request reconfiguration of existing hardware and operating system parameters. Personnel costs to reconfigure will be assessed. Degradation of system performance because of such changes will be frowned upon by installation management.

Purchase of System Options - Additional hardware or operating system modules may be purchased by submitting an appropriate purchase order. Costs will be assessed directly to the student account. Information on current cost and

F – 9

availability will be provided with each particular project.

Personnel Hiring - Requests for hiring additional installation personnel (e.g., for an additional shift) or for overtime authorization may be submitted. If approved, costs will be assessed directly to the student account.

## Student Input for Grading

Of course, choice of grading technique is the prerogative of each individual instructor. However, this section describes one set of student inputs that may be used for that purpose.

Final Report - This represents the single most important input to the student's final grade. The report should be typewritten with drawings, tables, and graphs of professional quality. The report should consist of two parts. Part 1 is the Analyst's Report. This should include a clear statement of the problem as perceived, a systems analysis and description, stated hypothesis of the specific problem, analysis done to confirm (or deny) the hypothesis, recommended solution, and verification that the implemented solution solved the problem. Also, included should be a cost analysis and recommendations for the future. Part II is the Student's Critique. This should include an evaluation of the simulation as a learning tool along with a discussion of problems encountered and recommended changes.

Interim Reports - Various interim reports may be

F - 1 0

required throughout the quarter, depending on the scenario. These should be of the same quality as the final report and will count toward the project grade.

Software Tools - Some specific software tools (e.g., data reduction packages) may be required during the suarter. These will specifically count toward the project grade. For any other tools that the student might develop, the source code should also be submitted. These programs will influence the project grade.

Oral Presentation - Each team will be required to make a final oral presentation to the class and possibly an interim report as well. Although an important purpose of the oral report is to share each team's analysis and findings with the rest of the class, presentation is a graded part of the project and should be done in a professional manner.

# II. <u>American Business Computer (ABC)</u> BITBUCKET Computer System

## Introduction to the ABC Computer

. .•

وثم فربية بماله المستحسب فيست

The ABC computer system was designed to provide flexiblity in providing custom computing power to any organization. This system consists of a variety of hardware and software options, which can be configured in many ways to tailor the BITBUCKET to the computing needs of the individual installation. This document provides abbreviated specifications of both the hardware components and the software options available to the ABC customer.





.

### Hardware Spectifications

## Model 2000 CPU

The Model 2000 represents ABC's standard Central Processing Unit. The Model 2000 provides for an extensive instruction set, with an average execution speed of 4 microseconds per instruction.

#### Model 3000 CPU

The Model 3000 provides the same capabilities as the Model 2000 except for a faster speed. The Model 3000 average execution time is 3 microseconds.

## Model 20 Core Memory

The primary main memory unit for the BITBUCKET system is the Model 20, 96K word module. This multi port memory allows connection to up to 5 IOCs and up to 99 CPUs. The module can be expanded to have a up to 20 partitions of 1000K words apiece.

# Models 3110 and 3117 Input/Output Controllers

The ABC Input/Output Controllers (IOCs) are sophisticated I/O channels, each capable of interfacing several peripheral devices to main memory.

Model 3110 is a byte-multiplexed channel capable of multiplexing up to 10 devices, each with a maximum transfer rate of 2.5k bytes per second. The primary function of this channel is to provide an overlapped interface for the system's card readers and line printers.

Model 3117 is a high speed multiplexor channel capable of multiplexing several block-oriented devices such as magnetic tapes, disks, and drums. Up to ten high speed devices can be connected to each 3177. Each device can transfer data at a rates up to 500K bytes/second. The channel itself is capable of transferring 500K bytes per second, and can multiplex the transfer of multiple blocks as long as the sum of the data rates of the devices concerned are within the transfer rate of the channel. This controller requires interconnection to an accessed disk or drum during speed and rotational latency. In the case of high speed drums (e.g. Model 2700) the controller is dedicated during the entire delay period. For the slower disks (e.g. Model 2714), the wait time can be multiplexed between several disks as long as the total transfer rate of the multiple disks is less than the maximum transfer rate of the IOC.

## Model 2700 Drum

Fixed head per track Rotational speed - 3600 rpm Capacity - 7K per track - 25 Active tracks Transfer speed - 410K bytes/second

F = 14

## Model 2714 Disk

-Consists of a controller and one single-spindle disk drive, connects directly to IOC

-Data storage capacity of each drive is 20.48 million bytes

-Access arrangement: each drive provides access to 200 recording cylinders via a column-type access mechanism with 20 vertically aligned read/write heads, 1 per disk surface. Each cylinder position provides access to 102,400 bytes of storage.

-Disk pack: each drive accommodates one IBM 2316 or equivalent disk pack which contains 11 disks and provides 20 recording surfaces

-Head position time:

Track to Track - 30 ms Average - 74 ms Max - 156 ms -Transfer rate: 312,000 bytes per second -Rotational speed: 3600 rpm

# Model 6051 Magnetic Tape Drive

The Model 6051 has been one of ABC's standard workhorse models for many years. Although of lower density and slower speed than more recent entries, the 6051's low cost and high reliability account for its continued popularity in many systems. Since the Model 6051 uses industry standard 9 track 1/2 inch magnetic tape, it utilizes the same physical medium as all other ABC tape drives, thus requiring an inventory stock of only one tape type.

Tape: Standard 1/2-inch magnetic tape 2400-ft reels.
Tracks : Nine-track
Density: 800 bits-per-inch
Encoding: NRZ
Speed: 37.5 inch/sec
Transfer rate: 30,000 bytes/sec
Gap size: 0.5 inch between blocks
Physical Dimensions: 19" x 48"

# Model 6110 Magnetic Tape Drive

•

The Model 6110 is ABC's standard high speed 9-track tape drive. The high transfer rate of 120,000 bytes per second combined with reliablity of 800 bpi density make the Model 6110 an effective tape storage drive. ABC's patented Quicloc mechanism speeds tape mounting.

> Tape: Standard 1/2-inch magnetic tape 2400-ft reels.
> Tracks : Nine-track
> Density: 800 bits-per-inch
> Encoding: NRZ
> Speed: 150.0 inch/sec

-Transfer rate: 120,000 bytes/sec -Gap size: 0.5 inch between blocks -Physical Dimensions: 19" x 36"

### Model 7001 Magnetic Tape Drive

The Model 7001 is ABC's top of the line high speed 9-track tape drive. Although actual effective data storage capacity depends on block size used, due to the need of a .4 inch interblock gap, the Model 7001 allows a capacity of 40 million bytes. ABC's patented Quicloc mechanism speeds tape mounting. The Model 7001 has proven itself as a highly reliable tape drive in many installions.

-Tape: Standard 1/2-inch magnetic tape

2400-ft reels.

-Tracks : Nine-track -Density: 1600 bits-per-inch -Encoding: NRZ -Speed: 150.0 inch/sec <u>Transfer rate: 240,000 bytes/sec</u> -Gap size: 0.4 inch between blocks -Physical Dimensions: 19" x 42"

Model 2920 Card Reader

17

Standard card reader reads industry standard 80 column cards. Read rate is 1000 cards per minute. The input rack holds 5000 cards.

Model 1200 Line Printer

C . . . . .

••••

Standard tractor-feed train line printer. Prints 1000, 132 character lines a minute.

## SOFTWARE SPECIFICATION

There are two differenc operating systems that can be run on the ABC computer. These operating systems can handle either a single cpu configuration or a multiple cpu configuration.

## Batch Uniprocessor Multi-Programming System (BUMPS)

BUMPs is ABC's multiprogramming operating system intended for use in a batch environment with a single processor. BUMP's comes as standard equipment for the ABC computer. This operating system is described in more detail in the following sections.

# System Jobs

The BUMPs operating system consists of a nucleus which is core resident, requires 96K words, and includes such tasks as memory loading, input/output control, and processor scheduling. In addition there are three system programs which <u>DO</u> have impact on the flow of user programs. The Job Scheduler, although core resident (in the 96K nucleus), must go through the execute queue like any other job. The Input Spooler and the Output Spooler are not normally core resident and must acquire resources in the Hold queue like any other user job. The operation of these programs and the resources they require is described in greater detail below.

Job Flow

A typical user program gets into the system through the input spooler (Figure 1). After the job is spooled it is placed into the hold queue to await resources. The job scheduler examines the jobs in the hold queue, according to the job priorities and the time the job entered the hold queue, and allocates memory partitions and allocatable I/O devices to each job. After the job is allocated it proceeds to the execute queue. When the cpu is free it takes the highest ranking job in the execute queue and performs a cpu burst. Job ranking in the execute queue is done by job type. The job scheduler gets the highest priority, followed by the software monitor start-up job, output spooler, input spooler and user jobs, respectively. The cpu burst lasts until one of four things happen: an J/O was issued for an allocatable I/O device, an I/O was requested issued for an unallocatable device, a timeout occurred, or the job finished. If an allocatable device I/O request occurred then the job gets placed in the smallest channel queue which is connected to the requested I/O device. The I/O is performed, the channel is freed, and the job is placed back into the execute queue. If an unallocatable device I/0occurred the job gets placed in the proper device queue. Once the device has been acquired the job is placed in the smallest channel queue which is connected to the device.



7

F-21

۰. ۰.

• . The I/O is then performed. After completion of the I/O the channel and the device are freed and the job is placed back into the execute queue. If a timeout occurred then the job is placed back into the execute queue to wait for another time slice. If the job is completed the job is placed in the output queue and waits for the output spooler to print the job. Also at that time, the memory partition used and the allocatable devices assigned to the job are released back to the operating system.

#### Input Spooler

The input spooler takes the user's programs in the card reader and spools them onto the disk. When a new job arrives at the input queue the operating system checks to see if the input spooler is already in memory. If the spooler is not in memory it is placed in the hold queue and must compete for resources and cpu time like any other job. The input spooler requires a memory partition of 4K before proceeding to the execute queue. When the input spooler acquires the cpu, it first schedules an I/O to read the 8C byte cards into a 1K system buffer. After the buffer is loaded the spooler is placed back into the execute queue. The next time the spooler gets the cpu the buffer is spooled to the disk. This two step process continues until the entire job is spooled. At that time the operating system checks to see if there are any more jobs in the input queue. The input spooler continues until all the jobs in the input queue are spooled. The input spooler is then released from the system and the memory partition is returned to the operating system. The input spooler requires .001 seconds of cpu time every time it acquires the cpu.

## Output Spooler

The output spooler takes the job's output fil- which is stored on disk and prints it. Whenever a job arri ... at the output queue, the operating system checks to see ~he output spooler is loaded. If necessary the output spooler will be placed in the hold queue. It will wait for a memory partition of at least 4K before it will proceed to the execute queue. When the output spooler first executes it will perform an I/O which will load a lK system buffer from the disk file. The spooler will be placed back into the execute queue so that it can print the buffer in the form of 132 byte lines. This process continues until the entire job has been printed. Like the input spooler, the output spooler continues to spool until its associated queue is empty. In order to set up a transfer, the output spooler consumes .001 seconds of cpu time every time it acquires the cpu.

## Job Scheduler

.

The job scheduler allocates memory partitions and  $\mathbf{I}$  allocatable I/O devices (tapes) to jobs in the hold queue. The job scheduler is loaded in the execute queue (if it is not already) every time a new job arrives in the hold queue or when resources are freed (after each user's job, spooler or software monitor completion). The job scheduler is always core resident as part of the operating system and does not count against the multiprogramming level. When the job scheduler obtains the cpu, it looks at every job in the hold queue to see if it can assign resources. Resources are assigned to jobs according to a priority system. The job scheduler ranks the jobs according to the job's priority attribute. This attribute is part of the jou's input parameters. If there is a tie, the jobs are arranged hold queue is analyzed according to this priority scheme. The job scheduler first che:ks to see if a memory partition is free which can handle the job. If a memory partition is free the scheduler thens checks to see if there are enough allocatable I/O devices free. Only if both conditions are met, is the job assigned resources and taken from the hold queue and placed into the execute queue. The jcb scheduler requires 3 cpu seconds every time that it executes and no I/O time.

## Static Partition Memory Management

With this memory manager, the user may specify up to 20 memory partitions of various size. Partitions are defined in terms of 1K work increments. The memory manager users a First Fit algorithm. When the job scheduler executes it starts with first free memory partition, checking to see if the job will fit. If necessary, the job scheduler will check all free memory partitions.

## Process Scheduler

The process scheduler selects jobs from the execute queue for the cpu to perform a cpu burst. The process scheduler uses a priority round robin scheme. Jot; in the execute queue are ordered first by their job type and then by the time they entered the queue. The job scheduler has the highest priority, followed by the the software monitor start-up job, output spooler, input spooler and user jobs, respectively. When the cpu becomes free the process scheduler takes the highest priority, longest waiting job out of the execute queue. The job will execute until its time quantum is used up, a I/O was issued or the job completes. If a time out occurs the job will be placed back of the execute queue. If a I/O occurs before the time-out, the job will do the I/O then go to the back in the execute

queue. The process scheduler is part of the operating system nucleus and consumes no visible resources.

## I/O System

To facilitate I/O data flow, all data transfers are done in fixed 1K blocks. The high speed multiplexed ABC Input/Output Controllers (IOCs) allow concurrent block transfers to or from several devices if the sum of the device effective transfer rates is within the transfer rate capability of the IOC itself. Each I/O request results in the transfer of a block of data from the specified devices, even if only a small portion of the block is requested. In order to make such a transfer, both the device and the IOC must be available. When an I/O is issued, the job first acquires the appropriate I/O device. If the device is busy, it must wait in the device queue. The device queues use a FCFS algorithm. After the job seizes the device, it tries to allocate an IOC which is connected to the device. If all the IOCs are running at capacity, then the job will wait in the smallest of the channel queues. Only after the I/Odevice and the channel are free is the data transfer started. Where possible, the I/O scheduler tries to distribute usage across the system of disks and drums.

Each job can have 4 types of I/O. First it must read all of the spooled cards from the disk. It must write all output printer lines to the disk file (for latter

transfer by the Output Spooler). In addition each job requires a number of tape I/Os and a number of disk/drum I/Os.

## Batch Multiprocessor Multi-Programming System (BMMPS)

The BMMPS operating system is exactly like the BUMPS operating system, except for the multiple processors. BMMPS can have a maximum of 99 cpus connected at one time. There is still only one execute queue from whic. all the cpus acquire jobs to process. All the cpus in the system are the same and can process any job.

## ACCOUNTING DATA

57

 $\tilde{\mathbf{x}}$ 

The ABC computer records accounting data on every user job that goes through the computer. The accounting file contains one record per job. The contents and the format of the accounting file (ACTLOG) is listed below in Table 1.

Table 1 ACTLOG Record Format

FIELD	VARIABLE	FORMAT	
1	Arrival Time	1X,F15.4,1X	
2	Job Name	F10.0,1X	
3	CPU Time	F5.0,1X	
4	Memory	F5.0,1X	
5	Priority	F5.0,1X	
6	Alloc. Devices	F5.0,1X	
7	Cards	F6.0,1X	
8	Lines	F6.0,1X	
9	Disk Blocks	F6.0,1X	
10	Alloc. Device Blocks	F6.0,1X	
11	Job Type (Note 1)	F2.0,1X	
12	CPU Time Used	F10.3,1X	
13	I/O Time Used	F10.3,1X	
14	Memory Size Used	F10.3,1X	
15	Departure Time	F15.4	
	-		

Note 1: ACTLOG only records user jobs so Job Type is always 1.0.

## SOFTWARE MONITOR

The software monitor is like any other job in the ABC computer system. It is entered into the system and must compete for computer resources. The monitor is loaded into the hold queue at the user specified starting time. It sits in the hold queue until a partition of at least 4K is available. It is then placed in the execute queue until it can acquire the cpu. When the monitor acquires the cpu it starts the tracing of jobs through five user specified queues. The software monitor then releases the cpu back to the operating system. Once the trace has started the monitor will record data about every job that enters one of the five queues. The queue name, the time spent in the queue, where the job came from, and where the job is going, is recorded every time a job leaves a queue. When the software monitor is monitoring queues and recording data, it puts an extra burden on the system. The software monitor causes the cpu to run at 95% efficiency. When the scheduled stopping time of the monitor occurs, the memory partition is freed and the cpu goes back to running at full efficiency. The software monitor writes its data to a file called SFTMON which is available for post processing. The contents and format of the records are list in Table 2.

Table 2 SFTMON Record	l Format	
-----------------------	----------	--

Ű7

FIELD	VARIABLE	FORMAT	COMMENTS
1	Queue Name	1X,A15,1X	See Note l
2	Time in Queue	F8.3,1X	In seconds
3	here Job Came From	A15,1X	See Note 1
4	Where Job is Going	A15	See Note 1

NOTE 1: See Table 3 for possible contents of these variables.

.....

.

# Table 3 Possible SFTMON Names

. ا Ö

1

.

VARIABLE NAME	COMMENTS
JOB ARRIVAL	
INPUT OUFUE	
EVEC OUFUE	
ABVI CROOT	Tenut Species empirel in eveter
ARVL SPOOL	Autout Spooler arrival in system.
ARVL OSPOOL	C/W Monitor errival in system.
ARVL S/WMOM	S/W MOULEOF AFFIVAL IN System.
ARVL JSCHED	Job Scheduler arrival in execute queue
JOB FINISHED	JOD COmpleted.
TAPE I QUEUE	
TAPE Z QUEUE	
TAPE 3 QUEUE	
TAPE 4 QUEUE	
TAPE 5 QUEUE	
TAPE 6 QUEUE	
TAPE / QUEUE	
TAPE 8 QUEUE	
TAPE 9 QUEUE	
TAPE 10 QUEUE	
DISK 1 QUEUE	
DISK 2 QUEUE	
DISK 3 QUEUE	
DISK 4 QUEUE	
DISK 5 QUEUE	
DISK 6 QUEUE	
DISK 7 QUEUE	
DISK 8 QUEUE	
DISK 9 QUEUE	
DISK 10 QUEUE	
CHAN 1 QUEUE	Channel 1 Queue.
CHAN 2 QUEUE	
CHAN 3 QUEUE	
CHAN 4 QUEUE	
CHAN 5 QUEUE	
CPU	
GENERAL CHAN	Used only in "Where job is going",
	because channel # not determine yet.
SPOOL QUEUE	Input Spooler queue.
OSPOOL QUEUE	Output spooler queue.

## HARDWARE MONITOR

Unlike the software monitor, the hardware monitor does not use up the ABC computer resources. The hardware monitor has its own timers, counters, and data recording devices. It is an event driven monitor which can be connected to any I/O device, channel, or cpu. The hardware monitor has two timer probes and three counter probes. These probes along with the starting time, the stopping time, and the sample data rate are specified in the configuration file. One limitation in the hardware monitor is that the monitor treats multiple cpus as one. The timer or counter connected to the cpus will be pulsed every time any cpu is activated. The output of the hardware monitor is written to a data file, called HRDMON, which is available for post processing. The record contents and format is listed in Table 4.

Table 4 HRDMON Record Format

-----

1.1222

.

••••

FIELD	VARIABLE	FORMAT
1	Time of Recording	1X,F15.4,1X
2	Timer # 1	F8.3,1X
3	Timer # 2	F8.3,1X
4	Counter # 1	15,1X
5	Counter # 2	15,1X
6	Counter # 3	15

## III. Modifying the ABC Computer Configuration

.

The ABC computer can be modified to any legal configuration by using a simple question and answer program. This program will from the responses, generate a configuration file. This file is then used as input to the simulation.

## Executing the Configuration Program

. . .

The execution of the configuration setup program is a simple four step process. By following these four steps the configuration file will be generated and automatically catalog as a permanent file.

Step	1	(sign on to the Cyber)
Step	2	ATTACH, PROCFIL, SETUP, ID=EE752
Step	3	BEGIN, SETUP
Step	4	(Answer questions as they appear on the screen)

F - 34

#### Trouble Shooting

The generation of the configuration file should work under normal circumstances. If an error does occur it is probably one of three things:

1. Character variables have to be enclosed in a single quote ('). Make sure you answer 'Y' or 'N' when it is appropriate.

2. If you get an error that states that there is no room for the file called CONFIG, it means that you already have 5 copies of the file. Since the Cyber can only hold five copies at one time, you need to purge one of the files. After you have purged a copy of the CONFIG file, type "RETURN, SETUP, CONFIG, TYPES". You are now ready to start again.

3. The third possible error that might occur is if you get disconnected from the terminal while in the middle of the process. If this happens return all your files and start again. If you do not return the files then the procfil might abort because it will try to attach the files again. Captain David L. Owen was born on 21 November 1956, in Cincinnati, Ohio. He graduated from high school in 1975, and attended the United States Air Force Academy from which he received the degree of Bachelor of Science in 1979. Upon graduation, he received a commission in the USAF. He served as a manager of simulation software for the 6595th Shuttle Test Group and the Shuttle Activation Task Force, Vandenberg AFB, CA, until entering the School of Engineering, Air Force Institute of Technology, in June 1982.

> Perminate Address: 4939 Fields Ertel Rd Cincinnati, OH 45241

#### VITA

. . .

UNCLASSIFIED						
	REPORT DOCUM	IENTATION PAG	E ,	101	1. 1. 1	1.
A REPORT SECURITY CLASSIFICATION		16. RESTRICTIVE	MARKINGS			
24. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION	AVA _ABILITY	OF REP	ORT	
UNCLASSIFIED		Approved	for public	rele	ease;	
B. DECLASSIFICATION/DOWNGHADING SCHE	DULE	cistribut	ion unlimit	ted.		
PERFORMING ORGANIZATION REPORT NUM	MBER(S)	5. MONITORING OF	RGANIZATION F	REPORT	NUMBER	5)
AFIT/GCS/EE/83D-16						
A NAME OF PERFORMING ORGANIZATION	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONI	TORING ORGAN	NIZATIO	NC	
School of Engineering	AFIT/EN					
ic. ADDRESS (City, State and ZIP Code)		76. ADDRESS (City,	State and ZIP Co	ode)		
Air Force Institute of Tech Wright-Patterson AFB, Ohio	noloay 45433					
. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT	INSTRUMENT I	DENTIF	ICATION N	UMBER
ic. ADDRESS (City, State and ZIP Code)		10. SOURCE OF FU	NDINC NOS.			
		PROGRAM ELEMENT NO.	PR DJECT NO.		TASK NO.	WORK UN NO.
11. TITLE (Include Security Classification)		-				
11. TITLE (Include Security Classification) See Box 19					·	
11. TITLE (Include Security Classification) See Box 19 12. PERSONAL AUTHOR(S) David L. Owen Cant USAE		-			<u> </u>	
11. TITLE (Include Security Classification) See Box 19 12. PERSONAL AUTHOR(S) David L. Owen, Capt. USAF 13b. TYPE OF REPORT NS. Those is	COVERED	14. DATE OF REPO	RT (Yr., Mo., De	21	15. PAGE C	
11. TITLE (Include Security Classification) See Box 19 12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF 13. TYPE OF REPORT MS Thesis FROM 16. SUPPLEMENTARY NOTATION	COVERED	14. DATE OF REPO 1983 Dece	BT (Yr., Mo., Do: mter		15. PAGE C	OUNT
11. TITLE (Include Security Classification) See Box 19 12. PERSONAL AUTHOR(S) David L. Owen, Capt. USAF 13a. TYPE OF REPORT IS. TYPE OF REPORT MS Thesis FROM 16. SUPPLEMENTARY NOTATION	COVERED	14. DATE OF REPO 1983 Dece	BT (Yr., Mo., De mter Particul Cr Frida E. Color Dean for E. color Dean for E. color	VL3 .ch and	15. PAGE C 	OUNT
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM         16. SUPPLEMENTARY NOTATION         7.       COSATI CODES         EVEL D.       CROUP	COVERED TO 	14. DATE OF REPO 1983 Decen	RT (Yr., Mo., De mter Fight D. Well Dean for E An Force Holes	۲/ ۱۹۹۰ - ۲۵۱ ۱۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰ ۱۹۹۰ - ۲۹۹۰۰ - ۲۹۹۰ - ۲۹۹۰۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰۰ - ۲۹۹۰ - ۲۹۹۰ - ۲۹۹۰۰ - ۲۹۹۰	15. PAGE C 3,1 Professional character (A bibcd High ber	OUNT 2 EVS Y Development iC)
11. TITLE (Include Security Classification)         See Box 19         2. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         ISA TYPE OF REPORT         MS Thesis         FROM         6. SUPPLEMENTARY NOTATION         7.       COSATI CODES         FIELD       GROUP         SUB. GR.	TO 18. SUBJECT TERMS ( Computer Perfo Computer Scier	Continue on reverse if n Dr ance Evalua	BT (Yr., Mo., Da mber Fritten in Gold Dean for E An Porce hold ecomogatoralises tion	W13 W13 Ach atel Comp Simu	15. PAGE C 31 Professional Charles IA Machine Uter Sir Lation	COUNT Count Co
1. TITLE (Include Security Classification)         See Box 19         2. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         3a. TYPE OF REPORT         MS Thesis         FROM         6. SUPPLEMENTARY NOTATION         7.         COSATI CODES         FIELD         GROUP         SUB. GR.	18. SUBJECT TERMS ( Computer Perfo Computer Scier	Continue on reverse if n Dr ance Evalua nc- Education	BT (Yr., Mo., De mter Portrod for Dean for K. and An Torce had ecomogal autidea tion	Win and and and comp Simu	15. PAGE C .3.1 Professional Pr	Development ICI Trulation
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13b. TIME         MS Thesis         FROM	COVERED TO 18. SUBJECT TERMS ( Computer Perfor Computer Scier and identify by block number TER SYSTEM SIMUL E EVALUATION US.	14. DATE OF REPO 1983 Decen Continue on reverse if n Dr: ance Evalua ac- Education er/ ATION FOR	BT (Yr., Mo., De mter Period Cr Dean for A act an Force had ecomogal Publication tion	Wia ach abd Comp Simu	15. PAGE C 	Development ic) r) mulation
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	COVERED TO 18. SUBJECT TERMS ( Computer Perfor Computer Scier Ind identify by block number TER SYSTEM SIMUL E EVALUATION US Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua nc-: Education er) ATION FOR	BT (Yr., Mo., De mber Forward for a Dean for a color an force for ecomognication tion	WLA Acta and artificiary Comp Simu	15. PAGE C 3.1 Professional Pro	Development ic) r) rulation
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	18. SUBJECT TERMS ( Computer Perfo Computer Scier and identify by block number TER SYSTEM SIMUL E EVALUATION US. Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n Dr: ance Evalua ac-: Education er/ - TION FOR	BT (Yr., Mo., De mter Period Cr Dean for A con an Porce house econogai Publican tion	Wia ach airl Comp Simu	15. PAGE C 	COUNT 2 Count 2 Count Cou
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	COVERED TO 18. SUBJECT TERMS ( Computer Perfor Computer Scier and identify by block number TER SYSTEM SIMUL E EVALUATION US. Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua hc-: Education er; 27ION FOR	BT (Yr., Mo., Do mter Provide Cr. Dean for E. Col Dean for E. Col Dean for E. Col Construction Construction tion	WL3 WL3 arruy Comp Simu	15. PAGE C 	OUNT 2 Powelopment iC) nulation
11. TITLE (Include Security Classification)         See Box 19         12. PERSONAL AUTHOR(S)         David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM         10. SUPPLEMENTARY NOTATION         17.         COSATI CODES         FIELD       GROUP         SUB. GR.         19. ABSTRACT (Continue on reverse if necessary ar         Title:       CPESIM II: A COMPUTER PERFORMANCH         Thesis Chairman:       Thomas C.	COVERED TO 18. SUBJECT TERMS ( Computer Perfor Computer Scier ad identify by block number TER SYSTEM SIMUL E EVALUATION US. Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua nc-: Education er) ATION FOR	BT (Yr., Mo., De mber Forward for a Dean for a for an force for ecomognic utilities tion	WLA ach and ar/puby Comp Simu	15. PAGE C 3.1 Professional Pro	COUNT 2 2 2 2 2 2 2 2 2 2 2 2 2
11. TITLE (Include Security Classification) See Box 19         12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF ISD. TIME MS THESIS         13b. TIME MS THESIS         15c. SUPPLEMENTARY NOTATION         17.         17.         18. SUPPLEMENTARY NOTATION         17.         17.         18. SUPPLEMENTARY NOTATION         17.         17.         18. SUPPLEMENTARY NOTATION         17.         17.         17.         18. SUPPLEMENTARY NOTATION         17.         18. SUPPLEMENTARY NOTATION         17.         18. ABSTRACT (Continue on reverse if necessary or Title: CPESIM II: A COMPUT COMPUTER PERFORMANCH         Thesis Chairman: Thomas C.	18. SUBJECT TERMS ( Computer Perfo Computer Scier and identify by block number TER SYSTEM SIMUL E EVALUATION US. Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua ac- Education er/	BT (Yr., Mo., De mter Period Cr Dean for been an Porce heat ecomogated tides tion	Wia ach aid Comp Simu	15. PAGE C 	OUNT 2 Development IC) Trulation
11. TITLE (Include Security Classification) See Box 19         12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	COVERED         18. SUBJECT TERMS (         Computer Performed identify by block number         Computer Scier         Ind identify by block number         TER SYSTEM SIMULE         E EVALUATION US         Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua nc-: Education er) - TION FOR	BT (Yr., Mo., De mter Partied of A Provide Cr Provide Cr Dean for Free An Processon tion	With and array Comp Simu	15. PAGE C 	OUNT 2
11. TITLE (Include Security Classification) See Box 19         12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF         13b. TYPE OF REPORT         13b. TYPE OF REPORT         13b. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	COVERED         18. SUBJECT TERMS (         Computer Performed identify by block number         Computer Scier         Ind identify by block number         TER SYSTEM SIMULE         E EVALUATION US.         Hartrum	14. DATE OF REPO 1983 Decen Continue on reverse if n or ance Evalua hc-: Education er; 21. ABSTRACT SEC	BT (Yr., Mo., Da mber Period br m Period br m Den for Local An Porce how tion	VLA Avaa and argerby Comp Simu	15. PAGE C 	OUNT 2 2 2 2 2 2 2 2 2 2 2 2 2
11. TITLE (Include Security Classification) See Box 19         12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	IB. SUBJECT TERMS (         COMPUTER Performed identify by block number         TER SYSTEM SIMULE         E EVALUATION US.         Hartrum         ACT         I D TIC USERS	14. DATE OF REPO 1983 Decen Continue on reverse if n Drance Evalua act Education er/ ATION FOR 21. ABSTRACT SEC UNGLAS	BT (Yr., Mo., De mter Period Er Ergin E. Moto An Porce home ecomogaceutides tion URITY CLASSIF SSIFIED		15. PAGE C 	OUNT 2-1 10013 Development ICI Trulation
11. TITLE (Include Security Classification) See Box 19         12. PERSONAL AUTHOR(S) David L. Owen. Capt. USAF         13a. TYPE OF REPORT         13b. TIME         MS Thesis         FROM	COVERED         TO         18. SUBJECT TERMS (         Computer Performed identify by block number         Computer Scier         Ind identify by block number         TER SYSTEM SIMULE         E EVALUATION US.         Hartrum         ACT         I D TIC USERS	14. DATE OF REPO         1983 Decen         Continue on reverse if n         Dr: ance Evalua         c-: Education         err         Continue on FOR         21. ABSTRACT SEC         UNCLAS         22b TELEPHONE N         (Include Area Cl	BT (Yr., Mo., De mter Provide Craw In the Action Dean for Erect an Processi ecomogated tion URITY CLASSIF SSIFIED TUMBER ode	Comp Simu	15. PAGE C 	DUNT 2 2 2 2 2 2 2 2 2 2 2 2 2

4

CECURITY OF ACCULICATION OF THE PLOT

