August 1971

CAL TSS Report

Howard Sturgis

July 1971

University of California
Computer Center
Berkeley

August 1971

Introduction

The purpose of this report is two-fold: first, to make available detailed information concerning the present status of CAL TSS, and second, to provide a basis for management discussion of the future course of development for CAL TSS. Since the TSS staff is small, not all projects can be pursued at once.

CAL TSS is a large-scale, time-shared operating system for the CDC 6400. It has been under development since the summer of 1968; the original goals of the system were to provide

1) a facility to support interactive languages for the use of students and other 'light' users of computers.

2) a facility for the development and use of large programs; and

3) a system for the development and use of large, sophisticated, and interactive programming languages.

On March 16,1971 CAL TSS, with all of its major modules included, was made available to the user community as a demonstration system. This version has many rough edges and some missing pieces, but is being improved.

At present the system can support a maximum of 15 teletypes, of which all can be in the BASIC subsystem, or some can be using the Editor and SCOPE Simulator. It is expected that, in September, with all of ECS, the system will be able to support about 50 teletypes, all in BASIC, or most in BASIC and some using the Editor and SCOPE Simulator which is a significant fraction of the load on the A machine.

This report contains a detailed discussion of what the current version of the system can do and its problems; it is accompanied by a number of appendices.[1]

---

[1] Appendix B was contributed by Vance Vaughan.

August 1971

Current Facilities and Capacity

At the present time CAL TSS provides:

BASIC	An interactive compiler.
SCOPE Simulator	A subsystem that will run many programs which run on the CDC SCOPE operating system. In particular it supports the FORTRAN compiler, the COMPASS assembler and the SNOBOL compiler of the A machine.
Editor	An interactive subsystem for the preparation, examination and modification of text files, (which may be submitted as input files to SCOPE Simulator, or may be output files from runs of the SCOPE Simulator).
Line Printer Card Reader	Subsystems for printing output files on the high-speed line printer and reading card decks on the high-speed card reader.
BCPL	A programming language for sophisticated programmers. Intended as a system programmers language.
Permanent Files	Files which may be kept within the system from day to day. (A program does not have to be re-entered each time it is run.)
File Protection and Sharing	Programs written by one user may be used by others if the writer desires.

The present system will support a maximum of 15 teletypes. These 15 teletypes can all be in the BASIC subsystem or in the Editor. A few could be using the SCOPE Simulator instead of BASIC/Editor, but as the number of people simultaneously using SCOPE goes up, the total number of users that can be supported goes down. The system has supported 6 users simultaneously interacting with a FORTRAN program run under SCOPE.

Comparison with the Batch System

It is difficult to compare TSS with the batch system. One of the few statistics available about the batch system is the number of jobs run per day. This goes up to about 4000 at times of heavy load. Unfortunately, most of the jobs run on the batch system contribute very little to the total time used and are generally thought to be small, short student jobs. One of the purposes of TSS was to support this kind of use.

Experience indicates that a student, with a little experience using the system, can do the equivalent of 4 batch jobs during 1 hour at a teletype. This estimate assumes that most of the information obtained

in the 4 batch jobs relates to syntactical errors, or minor bugs, which in fact seems to be the case for most student jobs run on the batch system. Thus, 40 teletypes, used by students, can provide the equivalent of 160 students jobs per hour on the batch system (4000 batch jobs per day = 166 jobs per hour).

For large jobs on the Batch machine the problems are different. Large jobs use facilities of the batch system that CAL TSS currently does not provide; e.g., access to magnetic tape, (CAL TSS has available only one drive for short periods of time) and direct access to ECS.

As far as computational efficiency is concerned, CAL TSS is quite competitive. For a job that is mainly CPU, it has very little overhead. For a job that has a lot of disk IO, it has some overhead, possibly about 50%. (The batch jobs will also have PPU overhead, but the total cost under batch is less.)

As an example of the ability to perform large tasks on TSS, the new version of the system was written and debugged on the old version, and contains at least 50,000 lines of COMPASS code.

Until the system has forced disk swapping (available late this year), it will not be able to run large numbers of users with large jobs.

Current Problems

Inability to charge for use of the system.

> This is the most serious problem. However, the essentials of the accounting system should be completed this summer. Once this has been done, the system can at least begin charging for connect time. The ability to charge for other resources, such as CPU time and ECS space, can probably also be installed this summer. Since ECS space is the most precious resource of the system, it should be the main resource charged for. It is now possible to charge for permanent disk space, but no administrative procedures have been set up. By the end of the summer the main problems with charging should be administrative rather than system.

Inconvenient access to permanent files

> This problem is caused by a number of things, one of which is the fact that most software was written for an earlier version of the system. It will be reduced when the software has been modified to

make use of new facilities in this version of the system.

No forced disk swap

Forced disk swap should be completed late this year. The main effect of its absence is a reduction in the number of simultaneous 'heavy' users the system can support; the current system cannot force a user with a lot of ECS space to give it up. There are a number of mechanisms to encourage giving up space, and they are used by most of the large subsystems currently available. However, a user can write his own programs that will take space and not release it, as well as typing commands at his teletype that obtain space, and then not releasing it.

Poor documentation

The system needs three levels of documentation: a cookbook on how to run FORTRAN and BASIC, a description for the sophisticated programmer who wishes to write his own subsystems, and finally an internal description for the programmers that maintain the system. An attempt has been made to produce the second level of documentation, and a document is available at some cost from the Computer Center Library. A very poor cookbook has been available for the last few months. A far better version is now in manuscript form and should be available in a month. There is practically no internal documentation of the new version of the system. There is some out-of-date documentation for the older parts of the system.

Poor availability of teletypes

At present there are 32 teletypes connected to the machine. Only 8 of these are available to the public, and these are the teletypes used by the system staff for work on the system. In order to support a large student load there should be a reasonable number of publicly available teletypes in a small number of locations. Also, if there are many more teletypes connected to the machine than can be supported at one time, there will be difficult administrative problems scheduling use of these teletypes.

Poor availability of the system

The system is now up for general use between 2 PM and 6 PM weekdays. Between the hours of 8 AM and 8 PM, 4 hours are used by the SCOPE system staff for maintenance of the Batch system. 4 more hours are used by the TSS staff for development of the new system. Now that maintenance work has been moved to the new system, these 4 hours of TSS development should be reduced. A total of about 1 hour is lost in the switching of the machine between TSS and the SCOPE staff.

Improvements to be made in the Near Future

The following are changes which the TSS staff already knows how to make. Some are in progress, all are expected by September.

1) The charging algorithm is being implemented with obvious benefits.

2) The user software (Editor, BASIC, SCOPE) is being modified to give more convenient access to files, making the system easier to use.

3) The user software is being modified to keep its swapped ECS requirements low, which will result in an increase in capacity.

Long-Term Improvements

These improvements involve more coding than those previously mentioned, and it is difficult to predict when they will be available. However, the staff knows how to implement them; it is just a matter of time.

1) Forced disk swapping which will enable the system to support more large subsystems and long-running programs simultaneously.

2) Background access to peripherals will make it easy to get large volumes of data into and out of the system.

3) New user subsystems and improvements to old subsystems will continue to make the system more convenient for more users.

August 1971

APPENDIX A

Capacity in September

In this appendix an attempt is made to predict the September capactiy of the system under various hypothetical conditions. The most easily understood description of capacity would be a statement of the number of logged in teletypes the system can support. Unfortunately there are a number of problems with this form of description.

The fact that the system is designed to ultimately run things other than logged in teletypes poses a technical problem. In particular it is expected that it will support a background batch system. Therefore the calculations will involve the number of disk processes, herein called processes, that the system will support. Each logged in teletype will be represented by one process, but other processes may also be active.

It is conceivable that a system with a very large number of logged on teletypes could be developed. However, there would be long delays in response to requests for service since at any given time, a large number of teletypes would be waiting for the completion of a previous request for service.

Actually, having several teletypes waiting for service at one time can increase the efficiency of the system in at least two ways. First, if the requests are in fact being processed in a time-shared manner and are generating disk activity, then this activity will increase the efficiency of the disk I/O. Second, since requests will arrive from the various teletypes at random, in order to keep the computer busy most of the time there will be times when several requests will be pending. This last will not even require that the requests be serviced in a time-shared manner.

Given a stable system and stable software, increasing the number of teletypes will have the following effects. When the number is small and the users of these teletypes do not interfere with each other, the amount of useful computing will go up linearly with the number of teletypes. A user at one teletype will not see a decrease in performance.

As the number of teletypes gets large enough for the users to interfere with one another, they will see a decline in individual performance, but the total amount of useful computing will still go up, although not as fast. Finally, the total amount of computing will level off. Above some number of teletypes there will be a small increase in utilization of the computer and a large increase in the total time wasted by the users at the teletypes.

The number of teletypes at which the various effects described above occur depends heavily on the use the individual users are making of the system. At one extreme the user may only want to use the BASIC subsystem to write and debug a fairly simple student type program. He will be spending a good part of his time puzzling out diagnostic messages typed at his teletype and will be making requests for service at a very low rate. At the other extreme is the user with a CPU bound program of very large size that runs for hours. He may make only one request for service, but what a request.

It is generally assumed that the system has two broad classes of users. The first is exemplified by the BASIC user mentioned above, except that he is more experienced and makes new requests relatively soon after the last is finished. The second is what is called an Editor-SCOPE user. He is typically in one of three modes. He is either 1) constructing or modifying a source program using the text editor subsystem; 2) compiling or assembling his program; or 3) running his own program, either under the SCOPE Simulator, or as a subsystem by itself.

When in Editor mode, the Editor-SCOPE user puts very little load on the system, probably less than an average BASIC user. In the other two modes he puts an unknown, but probably large load on the system. Only experience and measurement of the system will show how service is affected by the presence of some given number of such users.

The raw data used in the following capacity calculations include observed performance of the existing system, fixed parameters in the system code, known improvements in the system code that will be made this summer and similar improvements that will occur but have not yet been discovered.

ECS Space

Available ECS space is the current major limitation on the number of logged in teletypes. ECS space is absorbed in three general ways: first, by a sytem overhead independent of the number of logged in teletypes; second, by an overhead for each logged in teletype, independent of whether that teletype is getting service or not; and third, by space needed for those teletypes receiving service at a given moment.

The following paragraphs attempt to estimate these three sizes from various system parameters, and to estimate the number of logged in teletypes CAL TSS can support, considering ECS only. In the discussion all numbers are in decimal and K stands for about 1000. Most parameters from the system have an accuracy between 1 part in 10 and 1 part in 100. As these parameters were obtained from the system in late June and early July, they do not reflect current values. The term "processes" refers to logged in teletypes. Each logged in teletype

will be represented by one process, but other processes will exist; notably for a background batch system.

Preliminary

The size of ECS at Berkeley is 500K. 200K of these words have been dedicated by the Computer Center to the A machine. The 300K assigned to the B machine are divided by the TSS system into a number of pools. 140K go into initial system overhead, the rest go into two pools that are divided up on a per process basis. 105K go into a 'swapped' ECS pool, and 64K go into a 'fixed' ECS POOL. The intention is that processes with a large amount of swapped ECS can spend part of their time with all of their associated swapped ECS moved onto the disk. (Hence the name, 'swapped' ECS.) This swapping procedure is expected to be implemented late this year.

Initial System Overhead

The 140K of intial system overhead mentioned above is determined by various system parameters, some of which will have to be changed as the number of processes is increased. In fact, some are probably too large now. The initial system overhead has a number of components, including system code, teletype I/O buffers, peripheral device I/O buffers and tables for the disk file system.

The system code resident in ECS is not expected to increase by any substantial amount. The teletype I/O buffers consume about 80 words per wired in teletype, of which there are now 24. Assuming that this will rise to 100, an additional 6K of initial overhead will be needed. The disk I/O buffers consume 20K. Since an analysis of disk traffic has not been made, it is not known whether this size is high or low. It is probably high. It should not have to be more than doubled with 50 processes. Increase in the number of peripherals on the machine is not anticipated, hence no change in the peripheral I/O buffer space.

The tables for the disk file system, other wise known as Disk Data Storage (DDS), are now 16K, and their size depends on the amount of swapped ECS in use. As a first approximation the necessary size should grow linearly with the number of processes. In fact this is probably a conservative estimate since with more processes the amount of swapped ECS per process should be less than now. Another consideration is the fact that the current amount of DDS appears to be too high. No more than a third of it has ever been seen to be in use. In fact, the system programmers involved claim that 8K would be enough to handle the current load.

Given that the current system is capable of supporting 15 processes, the following estimate is made as to how the initial system overhead should be divided into fixed cost and per process costs. The system code will remain unchanged; the teletype I/O buffers will be fixed

cost, but at 6K more than now; the possible extra 20K of disk I/O buffers for a total of 50 processes will be counted as .4K per process. The DDS, which is now 1K per process, will count as .8K per process. The initial system overhead, which was first given as 140K, has to be reduced by 16K for DDS and increased by 6K for the teletypes, leading to a fixed cost figure of 130K.

Fixed ECS

Each process is now guaranteed 4K of fixed ECS. In addition there is 4K of fixed ECS for the rare process that will need more than the guarantee. However, such a process has not yet been seen. At least one fix is known that will reduce the needed guarantee of fixed ECS to 3.5K and it will be in by September. An optimistic estimate is that at least another .5K can be saved, giving 3K.

Swapped ECS

Each process is now guaranteed 3.5K of swapped ECS. In addition there is 38K of swapped ECS held in reserve for processes that are momentarily doing large tasks, such as running the SCOPE Simulator. Further there is 18K of swapped ECS used to hold 'frozen' files, i.e., files that have been read into ECS and are held there without charge for all users. Examples include the code files for BASIC, the Editor and the SCOPE Simulator.

The guarantee of 3.5K was set so that all users could run BASIC or the Editor. The initial version of BASIC, as with all other software, stays at one fixed size once it has been called, even while waiting for teletype I/O. Most of the subsystems will be converted this summer to reduce their size while waiting for teletype I/O. Most will also be converted to vary their size as their needs vary in general. Once this has been done for BASIC, the guarantee can be reduced to that size which all subsystems can reach while waiting for teletype I/O, probably 2K at the most.

The parameter that is most difficult to estimate is the amount of additional swapped ECS needed to support those subsystems providing service at the moment. First there must be a certain minimum amount of this additional swapped ECS to support the largest size that any one process may want to attain, which is 32K. Any additional amount will provide increasingly better service to the processes. For this purpose, the additional amount is best figured on a per process basis. At the present time, given 15 processes, the additional swapped ECS comes to 2.5K per process. In fact, for the system as it now exists, this is probably too low. (The 15 processes have never all been in active use by heavy users.) However, with the subsystems reducing their space while waiting for teletype I/O, most of the use of swapped space will evaporate. However, again, since the BASIC users will now have to use this additional space whenever they are getting service,

there will be a demand that did not exist before. In what follows the additional swapped ECS is figured in several ways: 1) at the minimum to support the system, 32K; 2) at a small level of 1K per process; 3) at a moderate level of 2K per process; and 4) at a large level of 4K per process. To put these figures in perspective, at the small level, all processes could be in BASIC, with half in simultaneous execution of small BASIC programs, and there would be 0.2K per process left over.

Results

The following table represents the preceeding information and concludes with the calculated number of processes the system can support under various conditions. It includes the possibility of using all or some of the ECS now dedicated to the A machine. The table has 5 columns, representing conditions in late June and 4 possibilities in September. Included in parentheses is the hypcthetical case where the fixed ECS guarantee has been successfully reduced to 3K.

	Late June, early July	September minimum additional swapped ECS	September small additional swapped ECS	September moderate additional swapped ECS	September large additional swapped ECS
Overhead					
Initial	140K	130K	130K	130K	130K
Frozen files	18K	18K	18K	18K	18K
Swap ECS	0	32K	0	0	0
Total	158K	180K	148K	148K	148K
Per Process					
Disk I/O bf	0	.4K	.4K	.4K	.4K
DDS	0	.8K	.8K	.8K	.8K
Fixed ECS	4K	3.5K (3K)	3.5K (3K)	3.5K (3K)	3.5K (3K)
Swap ECS guarantee	3.5K	2.0K	2.0K	2.0K	2.0K
Additional	2.5K	0	1.0K	2.0K	4.0K
Total	10.0K	6.7K (6.2K)	7.7K (7.2K)	8.7K (8.2K)	10.7K (10.2K)
Number of Processes					
Total ECS	14	18 (19)	18* (19)*	17 (18)	14 (15)
300K	24	32 (35)	32 (35)	29 (30)	23 (24)
500K	34	47 (51)	45 (49)	40 (42)	32 (34)

---

* More than 1.0K of additional swapped ECS per process in order to meet minimum condition of 32K.

August 1971

Summary of Assumed Changes

A. An increase to 100 wired in teletypes.
B. An increase in DDS and disk buffers to accommodate more processes.
C. A .5K reduction on guaranteed fixed ECS space per process by a known change in the system. (A possible further .5K reduction in guaranteed fixed ECS space per process.)
D. A 1.5K reduction in guaranteed swapped ECS per process. (This depends on modifications to the user software.)
E. Some figure of additional swapped ECS. (This will depend heavily on user mix.)
F. Possible additional ECS from that dedicated to the A machine.

Disk Space

Disk space will be the limiting factor when the system has a large number of users. The disk on the 6400, a 6638, comes in two independent halves, each with a capacity of about 8 million words. The B machine has one of these halves.

Roughly 1 million words is used by the disk system itself as part of an algorithm designed to allow writes to occur at the current disk arm position. The other 7 million words must be divided into two groups. The first is temporary disk space used by a logged in teletype that will be released at logout time. This space contains such things as the assembler scratch files, and output files to be printed or examined with the Editor. The second is permanent space assigned to each user of the system and will hold such things as his source files.

If it is assumed that the system can support 50 simultaneous users for each 1 hour during a 10 hour day, every day of the week, then a user community of at least 500 users is needed. Most of these users would be expected to be students, with modest demands.

One method of slicing the pie is:

500 Medium Users 4K words each (Enough for a 10 page FORTRAN program or equivalent of 600 cards.)	2×10^6
Small Number of Large Users	2×10^6
40 Logged In Users At 32K temporary space	1.3×10^6
10 Logged In Users At 128K temporary space (will allow assembly of very large COMPASS programs)	1.3×10^6
	6.6×10^6

August 1971

CPU Time

At the current capacity of the system as limited by ECS and with the current user community, there is CPU time to burn. CPU time should be a problem at about 50 users, assuming that the same user mix continues. If this does in fact turn out to be the case, then 50 would be the ideal load. The current user community is weighted towards heavy users, making extrapolation difficult.

In any case, a few words about how much useful CPU time is delivered to the user. For a user performing small tasks, there is a large overhead, and he receives only a small percentage of directly useful CPU time. For large tasks the situation is different. As an experiment a very large (over 100 pages) machine language program on the CAL TSS and on the A machine system was assembled. One problem with a comparison of this sort is the multiplicity of assemblers available. All of them assemble essentially the same language, i.e. COMPASS, the original assembler on the SCOPE system provided by CDC.

TSS system - NOMPASS	105 seconds CPU time 140 seconds real time (No one else on)
A machine - COMPASS	128 seconds CPU time 128 seconds PPU time 140 seconds real time (No one else running)
A machine - MOMPASS	60 seconds CPU time 82 seconds PPU time 90 seconds real time (No one else running)

Observe that there is as much variation between the assemblers on the A machine as between the machines. (MOMPASS is thought to be the same assembler as NOMPASS.) Most machine language programmers on the A machine seem to use COMPASS. The Time-sharing staff uses NOMPASS on their system exclusively.

August 1971

APPENDIX B
(Vance Vaughan)

User Experience, User Capacity, and Charging Users

This appendix presents some user experience on CAL TSS and introduces some capacity and charge rate estimates based on this experience. The first part gives those facts and comments which have been gleaned from actual user experience. The second part is more tenuous and tries to evaluate the capacity of TSS as compared to the batch system. Finally, there is a section which introduces some data relevant to determining an appropriate charge structure for running student jobs on TSS.

User Experience

Two things should be noted about user experience on CAL TSS. Namely, there is not very much experience because the system has not been available very long, and most of what experience there is has been obtained on early versions of the system, i.e., it is obsolete. However, it is the only data available and has some value in defining what is useful about the system and what areas need improvements.

Current users of the system can be divided into three general categories:

1) Very experienced and sophisticated users (including the system staff).
2) Students from several classes (and their instructors).
3) Users with their own dedicated lines.

Category 1 provides a biased view (not always favorable), but some observations should be made anyhow. First, it should be noted that the current system itself has been developed on an early version of the system which has been in operation since Summer '69. The interactive debugging capacity of the system has been very helpful, possibly crucial, in the development of the system. Most of the so-called user subsystems, such as the Editor, BASIC, and SCOPE, have been developed and debugged on the system. Thus it is a known fact that CAL TSS provides an environment for the implementation, debugging and operation of sophisticated, interactive subsystems. Several developments of this sort can be and have been in progress simultaneously on the system without denying service to more conventional users.

August 1971

Classes that have Used CAL TSS

Course Number	Quarter	Number Students	Language/ Subsystem	Instructor
CS120A	Fall 70	8	SNOBOL (SCOPE)	Mrs. Gould
CS120B	Winter 71	12	SNOBOL (SCOPE)	Mrs. Gould
CS1	Spring 71	60+	BASIC	Mrs. Gould
CS100	Spring 71	?	BASIC	Prof. Meissner
CE298-?	Spring 71	?	FORTRAN (SCOPE)	Prof. Glassey

The source of Category 2 experience is summarized in the table. The following comments are based on reports submitted by Mrs. Gould covering each of her three classes and on a telephone conversation with Prof. Glassey. I have been unable to contact Prof. Meissner. Mrs. Gould's reports are available for inspection.

First, the positive side. These comments are just the sort of thing that is supposed to be good about time-sharing, but it is nice to see that the users have noticed them.

1) The students get their assignments done more quickly (in terms of their time, not necessarily machine time) on TSS than on batch.
2) The students like Time-Sharing and generally prefer it to the batch mode.
3) The instructors feel that the immediate feedback is very valuable to learning.
4) The instructors note that projects are possible in the interactive environment which would otherwise be impossible.
5) Some students commented on the usefulness of the more extensive character set available on TSS.

Now, on the negative side. These complaints are commented on point by point.

1) Inadequate/non-existent documentation. This, combined with quirks and problems of early versions of the system, has caused inexperienced users a lot of difficulty in using the system. Prof. Glassey noted that whenever his students got off the specific set of instructions provided for them, they were frequently unable to continue or recover without logging off the system. Some staff time has recently been diverted from programming to preparing documents. Only experience will tell to what extent this documentation will alleviate the problems, but some improvement is inevitable.

2) System is hard to use. As with the documentation problem, this stems from the fact that the staff has been too busy gluing the system together to be able either to educate the user community or take minor wrinkles out of the system. This situation has only begun to ease recently, and the user interface has already improved markedly.

Actually, the student users have done surprisingly well in the face of system peculiarities. The next round of students will be spared a lot of the hassles which their predecessors faced and will presumably be even more productive.

3) Lack of available teletypes. This situation stems from two sources. One is the limited number of teletypes which the system can support simultaneously and is extensively discussed elsewhere in this report. The other is the lack of physical teletypes where students can use them. The whole problem of teletypes has been clouded by uncertainty as to the future of the system and lack of administrative policy on allocating the multiplexor ports. Now that the system is becoming productive, these problems will have to be solved. It seems that for instruction purposes, it is essential that labs with a goodly number of teletypes be established where students can gain access to them.

4) An interface to the batch system is lacking. This is part of a larger problem of allocating peripheral devices among users. It is the same old story; heavy demands on the staff by more fundamental parts of the system have prevented development of the peripheral interface beyond what is absolutely necessary for the development of the system. It should be noted that Mrs. Gould developed an ad hoc method of getting cards punched to drive the plotter for her 120B class. There are no fundamental problems here, it is only a matter of time and emphasis.

5) Delayed echo of characters. Prof. Glassey commented that he was able to type fast enough to cause the echo of characters to fall behind his typing. This seems to be an inevitable consequence of the full duplex mode of character transmission. Some compensatory comfort may be derived from the confidence inspired by the fact that the echoed character has come back from the computer, allowing the user to know that the line is not garbling what he types.

6) Restricted hours of availability. Because the hardware has been used for system development, both TSS and Remote Batch, the system has only been available to users four hours a day. TSS development requirements should taper off soon, and the possibility of scheduling more user hours, especially during the day, should be seriously considered.

7) Noise. Mrs. Gould notes that 4 model 33 teletypes in one normal room create an annoying noise level. This problem must be seriously considered if labs with many teletypes are to be established.

August 1971

Capacity

The problem of comparing a time-sharing environment to a batch-processing environment is very involved. Here, questions about the relative merits and demerits of two environments from the user's point of view are ignored. Instead, an attempt is made to compare the capability of the two types of system to handle a given user load. No attempt is made to compare the perfcrmance of the two systems for all types of jobs, which would be very hard to do. So this is further restricted by defining a 'student' assignment and then estimating the 'batch equivalent' capacity of TSS to support such student assignments.

A student assignment is characterized as follows:

1) The actual computing requirements are very small, only a few seconds of CP time.

2) The completion of the assignment on the batch system requires several runs, first to remove trivial syntax errors from the source code, then to remove bugs.

Statistics such as the average number of runs required to complete an assignment on the batch system are not known, nor is the average time per assignment at the ccnsole. Thus, there are no hard figures on which to base a comparison of batch and time-sharing even for the restricted kind of use under discussicn here. However, assume that in one hour at the teletype, a student can accomplish the same amount of work on an assignment as he could in fcur runs on the batch system. This figure has grown out of the experience of instructors and staff associated with student use of the system, but is impossible to justify precisely. No attempt is made to justify it beyond saying that it has consistently been regarded as ccnservative by everyone with whom it has been discussed.

On the basis of the figure

1 console hcur = 4 batch runs,

one can compute the batch equivalent capacity of TSS in jobs per day from

jobs = 4 x number of active teletype x hours TSS available.

Given that TSS will support 40 active teletypes and is up 12 hours a day, then TSS can handle the equivalent of

4 x 40 x 12 = 1920 student jobs per day.

August 1971

This is astonishing, to say the least. The batch load is around 2000 jobs per day, up to an occasional high of 4000+.

The figure of 1920 student jobs per half day indicates that TSS is highly competitive with the batch system in the processing of this kind of work, but the figure is so surprising and tenuous that no further speculation on its ramifications will be made here. Experience will provide the final answer, but at least there is reason to be optimistic.

Student Users and the Charging Algorithm

The problem of charging for use of the machine in such a way as to provide best possible service at the user end while providing necessary revenue at the Computer Center end is many faceted. Here are published some figures on the cost of running students on the batch system, and some inferences about possible charging algorithms and rates for TSS are drawn.

Figures supplied by the Computer Science Department give the cost per student per quarter for computer time as varying between $17 and $62, depending on the type of course.

The exact nature of the charging algorithm to be used for TSS is not yet specified, but the limited number of teletype hours available on the system seems to indicate that connect time should be charged for in order to avoid having access to the machine clogged up. If the algorithm charged only for connect time, a rate of $1-2 per hour would provide a healthy chunk of time for each student based on the current figures of $17-62 per student. But on the assumption of 40 active teletypes, this would only provide $40-80 revenue per hour from the terminals. The difference between this figure and the hourly revenue required from the machine would have to be made up by:

1) Charging for some other service, such as background batch.
2) Charging for some other resource, perhaps CP time or ECS space.

This would produce additional income from other users without substantially increasing cost to students.

APPENDIX C

Status of System Code

As the system is constructed in layers, the status of each layer will be discussed individually.

ECS System

This is the base layer of the system. The original design for this layer was completed in late Fall of 1968, and the major components were working in Summer 1969. In the light of the design work for the Disk System in Fall 1969 and Spring 1970, some portions of the ECS system were re-designed. During 1970 some of the uncompleted components in the original design were completed, notably the ECS compacter. Also, during 1970 some of the re-designed components were completed, in particular the new block parameter passing conventions and the parameter return conventions. Nearing completion is the new call-stack logic. This will require some slight modification in higher levelcode, and hence its completion requires careful integration with the higher levels of the system. Unstarted as yet is the fancy scheduler; work is expected to begin on it late this summer.

Also part of the ECS system, a sort of sublayer, is the interrupt system, which interfaces external devices to the rest of the system. Modification will probably be needed here to support more than the one tape drive currently supported. The interrupt portion of the ECS system has given no trouble since summer 1970, and no work has been done on it since then.

Although the ECS system has been very reliable over the last year, at present it has one known bug that causes system crashes, but on an average of less than once a month at the current load. So far there is insufficient information to diagnose the bug, and each time it occurs more code is inserted into the system to diagnose the trouble the next time it occurs.

Disk System

This portion of the system manages disk files and directories. It permits a portion of a file to exist in ECS and the rest to be on the disk. The detailed design of this level was begun in Fall 1969 and continued into Spring 1970. The basic components of this layer became operational in Spring 1971. It is now in about the same state of completion as the ECS level was in Summer 1969. A number of the components in the original design are not yet available, and some are present in an incomplete form. The major missing component is forced disk swapping., which is expected late this year.

August 1971

Command Processor

This is the highest layer of the system that must be maintained as protected code. The design was started in Summer 1970 for a quick and dirty system expected that September. (The so-called September System.) That design gradually evolved intc the version now available. Most of the features expected to be seen by an ordinary user are now completed. Missing features include acccunting, protected subprocess descriptors, and the ability tc construct user processes not tied to particular teletypes. These features shculd be completed late this summer. The most difficult one is accounting and has been in progress several mcnths.

User Software

This top most layer of the system is nc less important than any other part of the system. It is the layer that users confront most of the time. Since this layer of the system is not protected system code, it is possible for anyone to write parts of it, which has led to a very diffuse responsibility for this layer. Full responsibility, even for the necessary portions, does not lie within the project proper.

All of the code now existing at this level was first written for an early version of the system. The command processor for the new system contains facilities to allow the old code to be run with only minor changes. A number of less minor changes have now become necessary for two reascns: ECS space control and permanent file naming. Since no one has overall responsibility for this code, it is difficult to get these changes made.

Currently available user software includes: the SCOPE Simulator, the BASIC interactive compiler, the Editcr, the line printer and card readers, and the BCPL compiler. Some of the demonstration programs are also available; most were the prcducts of some Computer Science Department classes. These include an interactive haiku program and a conversational psychoanalyst.

August 1971

APPENDIX D

Current Allocation of Staff

This appendix gives the primary assignment, by portion of system, of the current staff. Most of the staff will be working full-time this summer, but will drop back to part-time in the fall. The figures are in terms of full-time equivalents for the summer. Most of the staff work on other projects besides their primary assignments.

Howard Sturgis

Director of the project. Chief architect of the Command Processor, but has now turned most of the coding over to two other programmers. He now spends most of his time supervising and working on overall design. This report itself has taken a considerable amount of his time over the last few weeks.

Vance Vaughan

Principal task, ECS system. Other tasks include preparation of documentation for users. Most of his time the last month was spent on documentation.

Dave Redell

Disk system. All of his time now is spent on implementing new features and removing bugs. He is currently saddled with a large body of code which he did not write.

Paul McJones

Disk system and Directory system. He is the chief architect of the directory system. Most of the intended features of the directory system now work, so he is moving over to the disk system to help Dave Redell. Also, he spends time on the SNOBOL system for use on the Batch machine.

Bill Bridge

Command Processor. Most of his work is now concentrated on the accounting system. He is chief architect of the BASIC system.

Gene McDaniel

Command Processor. Most of his work in the past has been as an assistant to Vance Vaughan on the ECS system, but he is now moving over to replace Howard Sturgis on the Command Processor. Most of his work in the near future will involve improvements to help with

the maintenance of the system, such as the ability to construct slave test user prccesses.

Keith Standiford

External I/O. He is the chief architect of the disk dump, load and recover package. He is now working on numerous small packages to help with system maintenance.

APPENDIX F

Current Projects

The following is a list of projects now under way, or which will begin sometime this summer.

Swapped ECS Space

All existing user software must be converted to control its use of swapped ECS space since it was originally written for an early version of the system in which such control was not possible. Swapping ECS space is the major factor which will determine how many users performing non-trivial tasks the system can support at one time. Since there is no one person directly responsible for all of this software, the conversion is a slow process.

New Naming Conventions

All existing software, which was written for an earlier system with different conventions, must be converted to use the new naming conventions. In order to provide reasonable access to permanent files, the software must be changed to use the new conventions.

BASIC Editor

BASIC now uses a version of the system text editor. This editor uses different conventions from the editors available in most BASIC systems. In fact the current BASIC system executes statements in order of their beginning line number, rather than their order in the text which results in a very confusing situation. A new editor will be written for BASIC that will be similar to other BASICs.

The above were all projects at the user software level; the following are projects at the system level.

Accounting

It is necessary to charge users for the use of the system. At present this is not done. During construction of the lower levels of the system a number of facilities were installed to be used by the projected accounting system. These facilities generally accumulate local charges of various sorts, e.g., swapped ECS space-time.

A full scale accounting system is being written which will: 1) maintain within the system a running total of remaining funds for each user; 2) appropriatelydecrement these totals at the end of each teletype session; and 3) record the resources used by the user for this session in a transaction file which will be used by the Computer Center accounting staff to send a record of use of the system to each user.

August 1971

Disk Space Control

At present the system is unable to control the use of the disk for either temporary files or permanent files. This space control should be in effect early this summer. At that time it will become possible to charge individual users for files kept permanently on the disk. (Completed as of 7/26/71 at the disk system level. The Command Processor now uses this facility to control permanent files.)

ECS System Call Stack Logic

A process is composed of a number of subprocesses that can call one another, passing and returning parameters. A number of imperfections were discovered in the original design, and a redesign was done last summer. The new version should be available late this summer. Since this change will entail changes in the code in higher layers, it can not be turned on until the old BEAD system is finally removed and the appropriate changes in the new system have been completed. These changes are thought to be minor.

Fancy ECS System Scheduler

In order to permit processes with a small amount of processing to get good service, a fancy scheduler for the ECS system was designed at the beginning of the project but has never been implemented due to other projects of greater importance. Work should begin on this scheduler late this summer and be completed late this year. As the load on the system increases, the scheduler becomes of greater importance since it essentially provides the character by character interaction ability of the system.

Forced Disk Swapping

Programs exist which will compute for long times between teletype interactions. These programs will hold large amounts of ECS while computing, thus preventing more interactive programs which have released space from continuing. The forced disk swap is the system's method of preventing this situation. Work on this facility will begin this summer and should be completed late this year.

Moving System Maintenance to New System

Most system work is now done on the old BEAD system. A move to the new system has been postponed for various reasons. At present the last obstacle is the lack of disk space control, but as soon as that is working the move will be made. (This is an obstacle because when disk space control is turned on, user directories must be remade and files moved from old directories to new ones. The fewer files that have to be moved the better.) The only sense in which this is a project is

that it will take some time and some reorganization of procedures. (Completed as of 7/26/71.)

Numerous Internal System Projects

Besides the projects described above, there are a number of individually small projects under way to improve various characteristics of maintenance of the system. One such project is to write user software for the new system to perform various tasks needed in the maintenance of the system, e.g., a program to construct a dead start system tape and a program to allow examination of post mortem tape dumps taken after system crashes or while testing new system code. Another project is to permit the construction of user processes containing new pieces of system code for test. It is expected that several system programmers could be testing new code in quite different sections of the system at the same time and independently. One more project is to permit the examination, on the fly, of the system part of a user process that has hung up and is no longer listening to the teletype. (These last two projects will provide most of the system features needed to implement the proposed batch system; see Appendix F, Future Projects.)

The dead start system tape program and the post mortem dump tape examination program have been completed as of 7/26/71.

APPENDIX F

Possible Future Projects

There are a number of projects which have been discussed, but for which no work is contemplated in the immediate future, i.e., this summer.

Trivial User Mode

Currently when a user logs in, he immediately acquires quite a lot of machinery to handle his requests for service. This machinery is assigned to him from the time he logs in to the time he logs out. If he is performing a large task, such as running the SCOPE Simulator, the machinery is a small part of his cost. When he is doing small tasks, such as editing, the machinery amounts to about half of his costs. When he is doing nothing, the cost is enormous.

It has been proposed to introduce into the system a new state, Trivial User Mode, in which the costs per user are very small. It has also been suggested that some small tasks could even be performed in this state by multiplexing them through a larger, special pseudo-user.

This mode would be introduced at the command processor level in the system, thus involving no changes at the fundamental levels. It is estimated that it may require 6 man months to implement.

This mode will greatly increase the number of simultaneous users, while reducing the power of the machine for those users in the trivial mode. It is in this way that KRONOS, for example, attains its large number of simultaneous users.

The main objection to this proposal is that it destroys the concepts of an individual process handling services for an individual user, thus allowing the writing of user software without having to consider multiplexing users. It will be possible for users to write their own software which multiplexes users and test it without System Programmer intervention.

Background Batch System

The major problem with the current version of the system is that it sits idle for most of the time. There are a number of ways to get more useful time out of it. The first is to increase the number of simultaneous users.

There are two problems with this approach: one is that the system itself needs much improvement to increase the number it can log in; the second problem is that since users will make essentially random requests for service, in order to have the machine busy for all but a tiny fraction of time, it must have many users waiting for services

most of the time. One crude calculation indicated that to have the machine busy 90% of the time required that the average number of users waiting for service at a given time would be about 10.

One way to keep the machine busy without this effect is to have something in the 'background' to run when no console user is requesting service. It has been proposed that this background be a Batch System to run jobs read in at a card reader. A method for implementation of this Batch System has been proposed that requires no changes to the fundamental levels of the system, but requires some new facilities at the command processor level, which should be available at the end of summer, as they are also needed for improved maintenance of the system itself. These facilities basically consist of the ability for a user process to construct slave user processes. Once these have been implemented, the rest of the work will be at the user software level.

At the moment there are two undesirable side effects of the Batch System. Since the fancy ECS scheduler has not yet been implemented, all users will see a degradation in the character by character response of the system, which is seen by all users, since it is used by the line collector on all control characters. Second, since forced disk swapping is not yet installed, the Batch System will occupy an essentially constant amount of ECS, whether running or not, thus decreasing the maximum number of logged in teletypes.

New Language Processors

At the moment, all of CAL TSS's language processors, except BASIC and BCPL, run under the SCOPE Simulator, including the processors most heavily used by 'heavy' users: COMPASS, FORTRAN, SNOBOL. There are two bad effects of this arrangement. The first is that these processors are not re-entrant; each user has to supply ECS space to hold his own copy of the one he is using. Second, the SCOPE Simulator is in itself an expensive subsystem to run.

For the most frequently used processors, it is hoped that a new version could be written that would be re-entrant. In fact, at the beginning of the project, a small amount of time was spent in looking at the RUN FORTRAN Compiler, to estimate how difficult this would be. There are two parts to the problem. The first is to make the code of the compiler re-entrant. This was thought not to be difficult, since it would be easy to collect all the writable data areas in one or two areas, and the only other task was to handle the use of the RJ instruction, which would also be easy with the introduction of an internal transfer vector in writable area. The second part of the problem is to handle file IO. This is more difficult, and the proposed solution is to construct a cut down version of the SCOPE Simulator that supplies the SCOPE IO to facilities.

August 1971

Floating Line Printer and Card Reader Drivers

The present line printer and card reader driver software requires that the user TTY wait while the printing or reading is in progress. It has been proposed that floating processes be constructed to handle these functions so that a user's TTY would be freed for other tasks. This facility could very easily be incorporated in the proposed batch facility, and is probably the best way to go.

Peripheral Device Allocator

The original conception of the system included facilities to permit individuals to write their own programs for driving the peripheral devices. One of the advantages derived from this concept is that it would not be necessary to write drivers that could handle all possible needs. The system would only have to handle the usual cases. The ECS system is so designed that CAL TSS can hand out capabilities for a given device and taken them away later. The necessary code must yet be designed and written at the command processor level to allocate the devices to individual processes. (The current system assumes that the code in the individual processes for driving peripheral devices is friendly.)

File Communication with the A Machine

Users need a means of transfering programs to and from the A machine Batch System, for example, in order to debug a program on TSS and then run production on the Batch System. At present this transfer is accomplished by the use of magnetic tape, but is unsatisfactory for several reasons. First, TSS tape handling conventions are primitive at present. Second, the system has available only one tape drive, which will probably be needed for some kind of file archival system. Third, there is insufficient operator support to handle a large amount of tape mounting and demounting.

Therefore, some way to copy files between the two systems without having to go to some outside medium is needed. The original system proposal was to use a channel coupler between the two machines and have appropriate PPU programs in the two machines to transfer the files. In order to save money, the channel coupler was not obtained. A more recent proposal envisions that the transfer be done in some common part of ECS. This will probably be a slow transfer since it requires the monitoring of the transfer by the system code. (There is no way for one machine to interrupt the other.) The system will be able to monitor the transfer at some regular interval, say about once every 10 milliseconds. If a sufficiently large block of data can be transfered

at one time, a reasonable transfer rate should be attainable.

File Archival System

One of the major limitations on the size of the CAL TSS user community will be the total size of files that it can hold on a permanent basis. At present these files are all held on the disk, a relatively small storage device. One obvious solution is to require users to hold their files on magnetic tape, but again, this would put an intolerable burden on the operators for mounting and demounting tapes.

proposed solution to this problem is to permit users to 'Archive' their files. All archived files would be written successively on the same reel of magnetic tape, thus reducing the tape mounting costs. Requests to retrieve files would be collected over some time interval, in hopes that several requests could be satisfied by a single tape mounting.

Speed Up

The system was designed with the intention that, after it was running successfully, those parts which proved inefficient could later be rewritten to run faster. For example, the ECS system contains many objects and about 100 actions on those objects, but most of the time spent in the ECS system probably involves only about 10 actions. With a degree of concentration on these actions a substantial reduction in overhead should be realized. Similarly, the disk system currently involves 3 subprocesses that call each other. This arrangement paid off in ease of construction and debugging, but now these 3 subprocesses can be combined into one, thus eliminating the inter-subprocess call time, a substantial saving. Finally, it was originally intended to supply some form of direct access to ECS. When this has been provided, the disk system code can be easily modified to take advantage of this feature to reduce its overhead by possibly one-half. In fact, the disk system was written with this modification in mind, and will require only small changes in its code.

The combination of all of the above changes should reduce system overhead by about one half.