Development of time shared basic system processor for Hewlett Packard 2100 series computers
by John Sidney Shema

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree of
MASTER OF SCIENCE in Electrical Engineering
Montana State University

Abstract:
The subject of this thesis is the development of a low cost time share BASIC system which is capable of providing useful computer services for both commercial and educational users.

The content of this thesis is summarized as follows:First, a review of the historical work leading to the development of time share principles is presented. Second, software design considerations and related restrictions imposed by hardware capabilities and their impact on system performance are discussed. Third, 1000D BASIC operating system specifications are described by detailing user software capabilities and system operator capabilities. Fourth, TSB system organization is explained in detail. This involves presenting TSB system modules and describing communication between modules. A detailed study is made of the TSB system tables and organization of the system and user discs. Fifth, unique features of 1000D BASIC are described from a functional point of view. Sixth, a summary of the material presented in the thesis is made. Seventh, the recommendation is made that error detection and recovery techniques be pursued in order to improve system reliability.

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Montana State University, I agree that permission for extensive copying of this thesis for scholarly purposes may be granted by my major professor, or, in his absence, by the Director of Libraries. A program listing of the 1000D Time Share BASIC system is not submitted as part of this thesis since program development was performed by a private business corporation which considers the listing as proprietary information. Arrangements must be made with a representative of Western Telecomputing Corporation of Bozeman, Montana in order to obtain a program listing for evaluation. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature John Sidney Illema

Date February 15, 1974

DEVELOPMENT OF TIME SHARED BASIC SYSTEM PROCESSOR FOR HEWLETT PACKARD 2100 SERIES COMPUTERS

by

JOHN SIDNEY SHEMA

A thesis submitted to the Graduate Faculty in partial fulfillment of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

P. E. Ulbrich
Head, Major Department

Donald K Weaver
Chairman, Examining Committee

Henry L. Parsons
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

March, 1974

ACKNOWLEDGMENT

The development of any major computer software operating system is a lengthy and complex process requiring the cooperation and creative talents of many individuals. The author wishes to thank the programmers and system analysts at Hewlett Packard Company for their excellence in designing and documenting the HP 2000B time shared BASIC system which served as a starting point for this investigation.

The writer is deeply grateful to Western Telecomputing Corporation for supplying the facilities and equipment to develop and debug 1000D BASIC. He is especially thankful to Dr. Donald K. Weaver Jr. for his backing of the project and many suggestions for improvement.

J.S.S

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The subject of this thesis is the development of a low cost time share BASIC system which is capable of providing useful computer services for both commercial and educational users.

The content of this thesis is summarized as follows: First, a review of the historical work leading to the development of time share principles is presented. Second, software design considerations and related restrictions imposed by hardware capabilities and their impact on system performance are discussed. Third, 1000D BASIC operating system specifications are described by detailing user software capabilities and system operator capabilities. Fourth, TSB system organization is explained in detail. This involves presenting TSB system modules and describing communication between modules. A detailed study is made of the TSB system tables and organization of the system and user discs. Fifth, unique features of 1000D BASIC are described from a functional point of view. Sixth, a summary of the material presented in the thesis is made. Seventh, the recommendation is made that error detection and recovery techniques be pursued in order to improve system reliability.

CHAPTER I

INTRODUCTION

Time sharing is a technique of computer system software design which allows many users, located at remote terminals, to have simultaneous access to a single central computer. Due to the extremely fast processing speed of the computer in comparison with that of the terminals, the computer is able to share its resources among the users. This makes it appear as if each terminal was directly connected to its own separate computer. Even if one terminal is running a program requiring large amounts of computer time, the other user terminals are required to wait only a few seconds for the computer to process their programs.

The objective of this thesis is to present an overall description of one specific time share system. This system was developed by Western Telecomputing Corporation in order to provide low cost computer services for both commercial and educational organizations in Montana

HISTORICAL DEVELOPMENT OF TIME SHARE PRINCIPLES

The introduction of the minicomputer in the late 1960's along with its associated low cost-high performance peripheral

equipment, caused the data processing industry to undergo a massive change. Early computer facilities emphasized the need to create powerful computer centers whose power was measured in terms of physical dimensions such as number of central processors, amount of main core memory available, and number of peripherals attached. Such thinking lead to the creation of multi-million dollar computer systems capable of handling tremendous quantities of problems and data. However, the cost of such systems was prohibitive to smaller companies and educational institutions.

With today's technologies of microminiaturization and large scale integration of circuits having reduced the size and cost of computers, a new philosophy of data processing has emerged. Rather than have one extremely powerful computer facility to handle all problems, it is now more economical to have several small processors, each handling a specific problem. In fact, in some cases the best solution to a specific problem is to design a special purpose processor using state of the art techniques rather than attempt to program a general purpose computer to perform the same function.[1]

Historically, much data processing was accomplished by programmers writing programs to a particular customer's specifications,

---

[1] Kampe, Thomas W., "The Design of a General-Purpose Microprogram-Controlled Computer with Elementary Structure," in Computer Structures: Readings and Examples, pp. 341-347.

entering the program with data cards into a batch processing facility, and returning hours later to see if any error had occurred. Omission of even a single comma could mean the run produced no results. Additionally, time was required to analyze the printout to determine if the algorithm written had actually performed the desired task. If not, several more program runs, each printing diagnostic information would be required before the program ran error free. Such a procedure was acceptable after a program had been fully written and made free of logic errors. The program could then be run on a routine basis. However, this procedure is unacceptable to the programmer since little, if any, interaction with the program was allowed.

Another problem which was not handled by the large computing system was that of the small or medium sized program requiring only several seconds of computer processing time. In most cases, the person waited several hours for his program to be run on the computer even though actual execution time was less than a minute for the entire program. In many cases, the computer's central processor was idle, while it waited for the results of a previous computation to be printed.

Numerous attempts were made to solve the problem of obtaining maximum utilization of a computer's resources in the early 1960's. One of the most effective solutions was that of a time share system developed at Dartmouth College by John G. Kemeney, Chairman of the

Department of Mathematics, and Thomas E. Kurtz, Director of Computation Center. Their idea, which was first proposed in 1962, was to design a program language which would be both simple to learn yet powerful enough to perform complicated tasks. The language was implemented in 1964 and was termed BASIC, for Beginner's All-purpose Symbolic Instruction Code. Up to 40 simultaneous users were allowed to communicate with the BASIC Language processor directly through keyboard-printer terminals. A master program, called the supervisory control or executive, kept track of all communication between the users and computer. It decided who was to run next, how long they were allowed to run, and performed the appropriate task called upon.

An important feature of the Dartmouth BASIC system was that the user was not required to be in the proximity of the computer in order to run a program. Terminals could communicate directly with the computer over regular telephone circuits. Thus, computer service was never farther away than the nearest telephone.

The major advantage of a time share computer system over a batch system is the ability of the programmer to communicate directly with a "virtual processor" and have each line of code examined immediately upon entry into the computer. This allows for on-line debugging of programs which is not possible on most batch systems. Since the computer is being "shared" among many customers, the cost

of computer services can be shared among the various customers. This allows even small businesses to afford computer services. Finally, communication with a time-share computer is on a real-time basis. Data can be entered and analyzed immediately, with results printed on the user terminal in a matter of seconds rather than hours. This capability of time share gives a company the ability to analyze data on sales, inventory, expenses, etc. and receive up to the minute company reports, such as cash flow or sales forecasting.

The Dartmouth BASIC system, capable of supporting many terminals, required a substantial investment in terms of processors, discs, and peripheral equipment. One of the first breakthroughs in terms of providing a low cost but powerful BASIC Language processor was the announcement by Hewlett Packard Company of their minicomputer-based 2000A Time Shared BASIC System in 1969. The 2000A System could generate up to 16 channels of BASIC for an investment of approximately $100,000. The 2000B System generated up to 32 channels of BASIC for about the same initial investment. The only major problem was that the initial system had limited disc space and additional disc space was very expensive.

In 1972, Western Telecomputing Corporation started development of a Hewlett-Packard 2000B - based time shared BASIC system designed around a new low cost moving-head disc drive, the Hewlett Packard 7900A. Design specifications stated that the system should be capable

of supporting up to 16 channels of BASIC, meet most of HP2000B Language specifications, provide large quantities of disc storage, and have a total system cost of less that $40,000. The version of BASIC that was developed to meet these requirements was called the WTC-1000D Time Share BASIC System.

RELEVANCE OF PREVIOUS WORK

The WTC-1000D time shared BASIC system described in this thesis, is only one of the many variations upon the Dartmouth System as conceived by Kemeney and Kurtz. A study of all manufacturers of computers would show that each one has his own version of BASIC. Some of these are so powerful and have so many extensions that they tend to obscure the simplicity of the original language.

The system described is organized almost identically to that of Hewlett Packard 2000B. That particular implementation of BASIC was selected for two reasons. The first is that the source tapes were readily available. The second, and most important, was that the 2000B system had been documented and described in sufficient detail that modifications could be easily made and tested.

ORGANIZATION OF REMAINING CHAPTERS

Chapter 2 discusses restrictions imposed upon the design of the system in terms of hardware and performance requirements. A discussion of terminology is presented to acquaint the reader with various concepts underlying the principle of time sharing.

Chapter 3 is a presentation of the operating system specifications of 1000D BASIC. It presents an overview of the system from both the user's point of view and the system operator's point of view.

Chapter 4 describes TSB System organization. It presents an overview of TSB modules and describes communication between modules. This chapter describes the organization of TSB tables and explains their functions. Finally, an account of disc organization is given, both for system discs and user/library discs.

Chapter 5 is a description of unique capabilities of 1000D which were developed to enhance system operation or make the system better fit the needs of commercial customers.

The concluding chapter 6 gives a summary and conclusion of 1000D as a solution to providing economical computing services to both commercial and educational users. Finally, several recommendations for system improvement are presented and discussed.

CHAPTER 2

OPERATING SYSTEM REQUIREMENTS

In designing a computer operating system, two types of problems must be solved.[2] The first type deals with intrinsic problems or those which are inherent in the design itself. The designer must look at the function the software is to serve and evaluate the requirements needed to implement that function. A design consideration of 1000D BASIC was that it should perform a useful function for both commercial users and educational users. This decision immediately created several design problems. Commercial customers needed large files for storage of data and would be using programs requiring significant compute times. Increased file access was needed and this would tend to tie the system up. Educational users demanded rapid response time to programs which would be mostly input/output bound. Access to large data banks would not be required. Finally, a system of user security would be required to prevent unauthorized access to a user's private files by another user.

The second type of problem occurs when one attempts to solve the first, that is technological problems arise. These problems

---

[2] Walsh, Larry, 2000C Course Notes, p. 10.

can be looked upon as forming the constraints with which the system designer is forced to work. For example, limiting the amount of computer core memory implies an upper limit to user program length. Not all of the system coding may fit in memory so techniques must be derived to move blocks of coding into core when they are to be executed. These constraints force the system designer to carefully consider various algorithms for making maximum use of the system resources.

If the designer has decided upon a system language, BASIC in this case, communication with the system must be considered. Two levels of communication are required.[3] One is a command language. A processor for this language is then required. This processor is responsible for handling system access, for control of system resources, and for file and program control. The second level of communication is that of a program language. An interpreter for that language is required. This is the processor which executes program statements that control file manipulations, carries out calculations, and perform specific operations on data.

Other technological problems must be solved while planning the operating system. The hardware system must be separated by software procedures into a number of virtual processors for

---

[3] Ibid., p. 10.

simultaneous use by many users. Most importantly, such sharing of resources must be done in a way that makes sharing of resources "invisible" to all of the users. This places the requirement that the system be highly organized.

Sharing of resources implies that a master program must exist whose function is to determine which user will be allowed to execute a command or perform a calculation. Such a program is referred to as a scheduler. The time this program is running rather than a user program is called overhead time. Overhead is the time the system spends performing routines designed to implement the sharing of resources. It is time lost to the users since their programs are idle when the scheduler is operating. It is important that the overhead time be kept to an absolute minimum in a well organized system. Another function of the scheduler is to maximize utilization of system resources. This means the scheduler should attempt to keep each hardware subsystem busy and insure that the system does not "bottleneck" because one resource is overly busy while others remain idle.

Response time or the amount of time required for the system to respond to a user command will increase as system utilization increases. The system must wait for one user to complete a task before it can allow a new user to start its own task. The technique of "time-slicing" allows the computer to implement resource

sharing among the users and reduce response time. This procedure enables the computer to set an upper limit on the amount of time a user's program may execute in core. If the program exceeds this limit it is written back onto a disc from core and the next user's program is read from the disc into core and allowed to execute.

1000D SYSTEM SPECIFICATIONS

WTC-1000D BASIC was designed to run on a Hewlett Packard 2100 series computer. This section presents some of the design specifications and constraints which were placed on 1000D. Appendix A contains a complete list of all hardware required to operate the system.

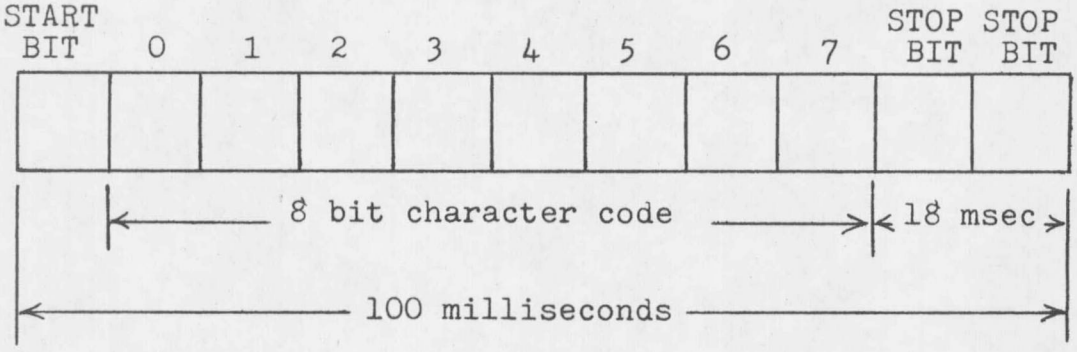
1000D was designed as a small-scale time-share system capable of supporting up to 16 simultaneous users. In order to keep response time of the system to less than five seconds, data communication transfer rate was set to be in the 10 to 30 character per second range. Faster rates would force the scheduler to spend most of its time character processing, leaving little time to execute user programs.

The most common time-share terminal in use at the time of system design was the ASR-33 teletype which operated at 10 characters per second (110 baud). This terminal communicated in a bit-serial

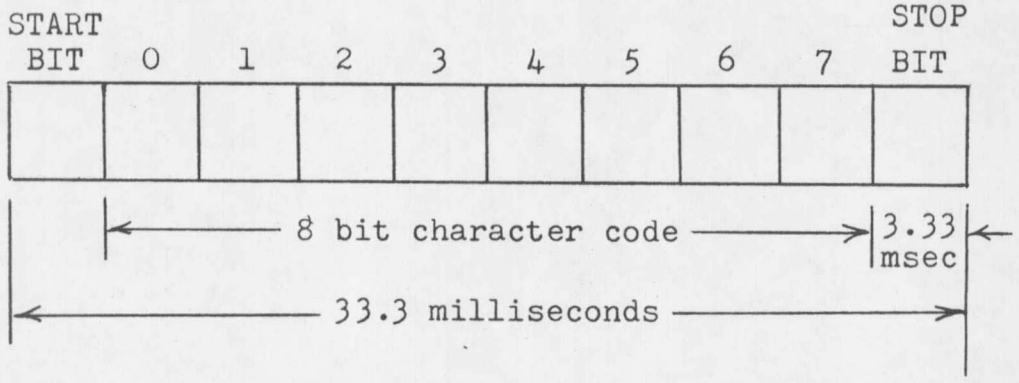
ASCII code. ASCII code was also the method of character representation internally within the HP 2100 series computers. Hence, no code conversion was required by the input/output processor. Figure 1 gives the format of the data words generated by a device operating at 10 characters per second and by one operating at 30 characters per second.

The input/output processing routine required at the worst case .15 milliseconds to process a single character. At an input speed of 10 characters per second, the routine has two bit lengths to process a character without loss of data, or 18 milliseconds. It is obvious that at least 100 channels could be processed at this speed. For terminals operating at 30 characters per second, timing is more critical. The computer has only one bit length or 3.33 milliseconds to process a character from each channel without loss of data. Calculations show that the computer can process at best, only 22 channels at 30 characters per second. Therefore, a full system of high speed terminals would tend to run much slower than a full system of slow speed terminals since most of the computer's time would be spent processing characters.

Since all communication from users to the system would be over telephone connections, a second design consideration was to allow system control of the telephone circuits. There were three reasons for doing so. First, a person dialing a wrong number could



Data Word at 10 Characters per Second.



Data Word at 30 Characters per Second.

Figure 1. Terminal Data Formats.

tie up one of the channels, preventing others from accessing the computer. To solve this problem, the computer allows only a certain number of seconds for the caller to log on the system. If he fails to do so, the line is disconnected. Second, if the phones are on a rotary call-in basis, a user can never be certain he will be able to access a particular channel if he is accidentally disconnected. A data set monitor in the system will log the user off the computer in the event his telephone carrier signal is lost. Third, a user's telephone is automatically placed back on hook when he logs off, making the channel immediately available to other users.

The amount of user space available for programming is very important to the customer and system designer. A customer would like to have unlimited program space. The system designer is aware that longer program areas mean the system will run slower since it takes longer to swap users in and out of core. A compromise was made in 1000D setting maximum program length to 4500 words, approximately 500 BASIC statements. For longer programs, the user was given the ability to "chain" programs together, resulting in virtually unlimited program length.

A final design consideration was that of selecting an optimum amount of disc storage. Fixed head discs were ruled out since they were very expensive and provided limited storage. However, access time was very fast, in the 5 millisecond range.

A decision was made to utilize the Hewlett Packard 7900 and 7901 moving-head disc drives. Access time was in the 35 millisecond range, fast for a moving head disc, and storage capacity of a single disc was roughly four times that of a fixed head disc. Each moving-head disc channel provided 203 tracks of 6144 words per track for a total channel capacity of over 1.2 million words. This was enough storage for approximately 960 medium-sized programs. 1000D allows up to four disc channels for a maximum capacity of over 4.8 million words or approximately 3840 programs.

CHAPTER III

OPERATING SYSTEM PROCEDURES

The language processor of 1000D BASIC is the same as that for HP2000B.[4] This is an extended version of the original Dartmouth BASIC as first described by Kemeney and Kurtz.[5] This chapter summarizes the main features of BASIC from both the user's and operator's view.

USER SOFTWARE CAPABILITIES

Extended BASIC enables the user to perform complex matrix operations in a single statement. Table 1 lists these operations and defines their function.

The working size of a matrix is defined by the program through the use of DIMENSION statements. For example, DIM A (5,3), creates a matrix of five rows and three columns. These are maximum limits which may be dynamically redefined in the program using the matrix IDN, ZER, CON, READ, or INPUT functions. In the above example, MAT

---

4 Hewlett Packard Company, 2000B: A Guide to Time Shared BASIC, 205 pp.

5 Kemeney, John G., and Kurtz, Thomas E., BASIC Programming, 122 pp.

TABLE 1. BASIC MATRIX OPERATIONS.

Operation	Example	Description
Addition	MAT C=A+B	Add matrix B to matrix A. Put result in matrix C.
Subtraction	MAT C=A-B	Subtract matrix B from Matrix A. Result in matrix C.
Multiplication	MAT C=A*B	Multiply matrix A by matrix B. Multiply matrix B by scalar A. Result in matrix C.
Inversion	MAT C=INV(A)	Generates matrix C as the inverse matrix of A.
Transposition	MAT C=TRN(B)	Generate C as the transpose matrix of B.
ZERO	MAT A=ZER	Set all elements of matrix A equal to zero.
Identity	MAT A=IDN	Generates A as the identity matrix.
Initialize	MAT B=CON	Sets all elements of matrix B to one.
READ	MAT READ C	Reads data from program DATA statements into matrix C.
Input	MAT INPUT B	Reads in elements of matrix B from user terminal.
Print	MAT PRINT A	Causes all elements of matrix A to be printed on user terminal.

Note: MAT READ may also input data from a user disc data file. MAT PRINT may write data to a user disc data file.

A = CON (2,2), redefines A to be a matrix of two rows and two columns. Matrices can only be redefined as being smaller than that set up by the DIM statement. The maximum number of elements allowed in a matrix is 2000. An array of more than 2000 elements will produce an error condition.

BASIC contains special variables and language elements for manipulating string quantities. These are alphanumeric values rather than numeric values. String variables are defined in the system by declaring a single letter (A through Z) followed by the dollar sign ($). The physical length of the string variable must be declared in a dimension statement. For example, DIM A$(36), sets up a string variable A$ of 36 elements. Maximum string length is 72 elements. A string value is a set of from 1 to 72 characters enclosed in quotes. Most printing and non-printing characters of the ASCII code set are valid string characters. During execution of a program, the logical length of the string is set. This is the count of the actual number of characters in the string. For example, the statement, A$ = "SAMPLE STRING", has a logical length of 13 characters even though the physical length of A$ was specified as 36 elements.

Substrings may be referenced by the use of subscripted variables. In the above example, A$(2,6) gives the second through sixth character or "AMPLE". BASIC allows string, substrings, and string variables to be used with relational operators. They are com-

pared and ordered as entries are in a dictionary. For example, the following statements may be written:

```
100  IF  A$ = "TEST"  THEN 400
200  IF  A$(4,6) ≤ B$(7,8)  THEN 500
```

The first statement will cause the program to go to statement 400 only if string A$ has four elements which are exactly "TEST".

The second example will go to 500 if the fourth and fifth elements of A$ are alphabetically less than the seventh and eighth elements of B$. Thus, BASIC allows string values to be read, printed, compared, and sorted with very little programming effort required.

Two special features of extended BASIC are the ability to specify length of the terminal print line and ability to access the system clock. Use of the WIDTH command allows the user to inform the system that the terminal may print only 20 columns of data per line or up to 132 columns per line. Use of the TIM(X) function allows a program to obtain the current minute, hour, julian day, and year from the system real-time clock. This information is useful in identifying multiple runs of the same program or for reporting run times of various programs.

A serious limitation of 1000D BASIC is its rather small user program area of 4500 words. Since some of this area is used for program overhead, program lengths generally have a maximum length of 3500 words. This is sufficient to store a BASIC program of roughly

500 statements. If arrays are dimensioned, the user loses two words of program space for each element in the array. Commercial programs are typically very long and utilize large arrays for data storage. Use of the BASIC CHAIN statement allows the programmer to divide one large program into many smaller segments which are stored on the disc. As each segment of coding is needed, it is called off the disc by the currently executing program. Parameters and other values may be passed from one program to the next through COMMON or data files. Use of the CHAIN statement gives BASIC the capability of handling programs of virtually unlimited length.

All users have access to two levels of libraries. The first level is that of the system public library. These are programs and files saved on the disc by the master user, A000. Users may get these programs from the disc, list and punch them, modify them, and execute them. User A000 may protect any program in the public library. This makes them "run-only" to all other users. Such a procedure would be used to prevent unauthorized copying of proprietary software. A user also has access to a second level of libraries, an individual user library. Programs and files stored in an individual library are accessible only by the owner.

The most powerful program function of BASIC is its versatile file handling capability. Users may create data files on the disc to allow for direct manipulation of large volumes of data. Files

may be of two forms: sequential or random access.

Sequential files are treated as serial access storage devices. A program starts reading or writing data as a serial list of data items from the physical beginning of the file and continues on an element by element basis until either the physical end of file is encountered or an end of data condition occurs. During a serial file write operation, the new data element is written immediately following the previous data element. To retrieve an item from the file, the program must start at the beginning of the file and read through all items until it comes to the desired data element.

Files may also be used in a random access mode. This enables the program to break the file into a series of logical subfiles that can be modified independently of each other. Each subfile is a block of 128 words.

Data may be written into a file in either numeric form or string variable form. String and numeric data may be randomly mixed in any file. Use of the TYP(X) function enables the program to identify what type of data the next file item is during a file read. Numeric data requires two words of storage per number whereas two string characters may be stored in a single word.

File lengths are variable from 1 to 48 sectors. Each sector holds 128 words of data. Maximum file length is 6144 words. Each program may reference up to 16 files at one time. This is equivalent

to having immediate access to 98,304 words of additional data storage. Any program may access an unlimited number of files during execution through the ASSIGN statement. This statement allows a new file to be referenced in the place of a previously defined file in the program.

The system allows users simultaneous access of files. The first user program to reference a file is allowed read/write privileges. All other users accessing the same file are allowed read-only access. Users whose account identification codes begin with an A (A000 through A999) always are allowed read/write access to the same files.

This section has described some of the more important capabilities of extended BASIC. A complete description of user capabilities may be found in the WTC-1000D BASIC user's guide.

SYSTEM OPERATOR SOFTWARE CAPABILITIES

The system operator is responsible for maintaining the TSB system, adding new accounts, modifying existing accounts, and removing old accounts from the system. These functions are accomplished through the system operator's console using special system commands and through the user account A000.

System control through user A000 is restricted to two functions.

Only programs stored under that account become a part of the public library, accessible to all users. Placing programs and files in this library is the primary function of A000. The second function involves maintenance of a particular program in the public library, the "HELLO" program. This program is automatically run on the user's terminal following a successful attempt to log on TSB. HELLO is a standard BASIC program whose function is to describe the system, give the current date and time, and print important messages from the system operator to the users.

Primary system control takes place through operator commands entered on the system console. One function performed is that of limiting access of system resources by the users. The system operator controls resource allocation to up to 562 user accounts, the maximum allowed by the system. This is accomplished by setting restrictions on each account through operator commands. Each account is assigned a unique ID code and password required for system access. The operator also specifies a maximum number of minutes of connect time allowed each user and additionally, a maximum number of sectors of disc storage the user is allowed. The password, maximum connect time and maximum disc storage allowed may be modified at any time by the system operator. If required, an entire account may be removed from the system by an operator command.

The system operator may request TSB to generate reports

useful for evaluating system utilization. The ROSTER command generates a listing of currently active channels as shown in Figure 2.

```
ROSTER
....  ....  S231  ....  A000  A000  ....  C246
K341  ....  ....  ....  ....  B260  .... ....
```

Figure 2. Example of ROSTER Command

The first line lists, from left to right, the users logged on terminals 0 through 7. User S231 is on channel 2, user A000 is on channels 4 and 5, and user C246 is on channel 7. The second line list users on channels 8 through 15. User K341 is on channel 8. User B260 is on channel 13. Four dots indicate an inactive channel.

A listing of all accounts with the total connect time used to date and disc space used to date may be obtained with the REPORT command. This command is used to generate a report for account purposes. An example of the REPORT command output is given in Figure 3.

REPORT

ID	TIME	DISC	ID	TIME	DISC	ID	TIME	DISC
A000	00007	02798	A001	15339	02314	A010	01723	00003
C021	00000	00000	C060	00767	00010	C140	00583	00125
C240	00666	00082	C241	00311	00027	C242	00106	00088
S231	05267	00186	S232	00099	00091	S239	00043	00047
S300	03406	00292	U010	00311	01241	Z623	20001	01024

Figure 3. Example of REPORT Command

ID specifies the account identification code of the user. TIME is the total connect time used in minutes and DISC is the amount of disc storage used in sectors.

The DIRECTORY command is used whenever the operator wishes to obtain a listing of library programs and files. The operator may specify what type of directory is desired, a complete listing of all programs and files on the discs or simply a directory for an individual account.

Figure 4 gives an example of a directory for a single user.

DIR - S232

ID	NAME	DATE	SUB	TR/SEC	LEN	
S232	*ALFA1	355/73	0	092/25	03	
	*ALFA2	004/74	0	092/22	03	
	*ALFA3	004/74	0	102/07	03	
	*ALPHA	006/74	0	092/18	04	F
	CODE	353/73	0	139/00	06	
	BA	009/74	0	092/04	14	C
	MCDA	011/74	0	166/20	14	
	WORK	023/74	0	117/02	08	C

Figure 4. Example of DIRECTORY Command

The account identification code is listed under the column ID. The program or file name appears under NAME, the date the entry was last referenced is given under DATE. The disc channel number is printed under SUB and the disc address of the entry in the form track number/sector number is printed under the column TR/SEC. The length of the program or file in sectors is printed under LEN. If the entry is a file, an "F" is printed. A "C" prints if the program has been saved in a semi-compiled form. A "P" indicates the entry has been protected. If none of the above appears, the entry is a program.

Programs and files which have not been referenced since a particular date may be removed from the system with the PURGE command. The disc space recovered is returned to the system. The system operator may also move programs and files from one disc to another with the MOVE command. This feature is useful when attempting to recover programs from a defective disc.

Finally, the system operator has the ability to initiate an orderly shutdown of the TSB system through the SLEEP command. Back-up discs may be written using utility programs after TSB has stopped. These backup discs may be used to restore the system in the event of a system failure.

CHAPTER IV

TSB SYSTEM ORGANIZATION

This chapter describes in detail the various modules which comprise TSB and the communication which takes place between these modules. Appendix B is a core map of the system. It shows how the modules are distributed throughout core. Since almost all communication is carried out in tables, a description of the format and function of important TSB tables is given. Finally, organization of the system and user/library discs is discussed, since most TSB tables are disc resident.

TSB SYSTEM MODULES

Time-share BASIC is written in several independent modules which link together through core and disc resident tables and through system linkage variables in core. This section describes the various system modules, defines their function, and lists linkage rules for communication with other modules. Core locations for these modules are given in Appendix B. Ten modules will be discussed: (1) base page, (2) power fail processor, (3) input/output buffers, (4) BASIC language processor, (5) disc criver, (6) system console driver, (7) multiplexor driver, (8) clock driver, (9) scheduler, and (10) system library overlay programs.

Base Page. The first 1024 words of memory are directly addressable by instructions located in any other part of memory. For this reason, most system links are located on base page. Memory locations below 100_8 contain interrupt linkages to input/output processing routines and general temporary storage locations used by various programs. The system equipment table starts at address 100_8 and constitutes the core resident information about the system. Immediately following the equipment table are system status variables and input/output driver temporaries. Some of these linkages are bit flags, each bit in the word referring to a single port or channel on the system. Figure 5 gives an example of an arbitrary bit-flag word.

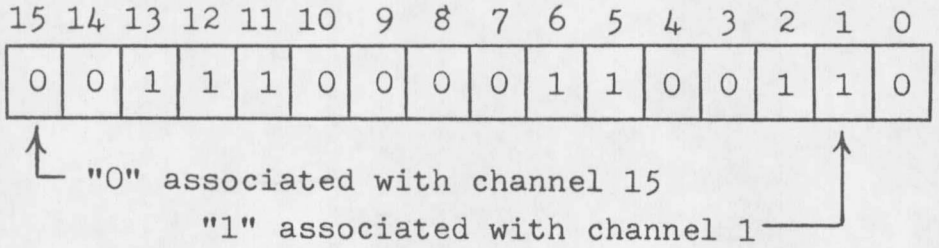


Figure 5. Bit-Flag Representation

From figure 5 it can be seen that a value of "0" or "1" may be associated with a particular channel by setting the bit in the word

to the appropriate state. A test of the state of the bit is used to determine whether or not a specific condition exists on a particular port. Table 2 lists base page links, defines whether they are word or bit flags, and describes their function.

The remaining words on base page not contained in user area are linkages to numerous subroutines located throughout memory and general use constants shared by various routines.

The user swap area begins at approximately 1300_8. The first four words are used to save run-time variables when the user is swapped out of core to the disc. These four locations store the A-register, B-register, Extend bit, Overflow bit, and Program counter when the user's program time-slice is exhausted. The next block of memory stores subroutine return addresses of the interrupted program. General usage information follows. The first word of available program space in the user's swap area is approximately 1774_8.

Power fail processor. If the computer is running when a power failure occurs, control passes to the power fail processor which saves the current machine and input/output system status. A flag is set to indicate that this status has actually been saved. The program then halts the computer.

When power is restored, the flag is checked to determine whether or not the power up process can be started. When sufficient power

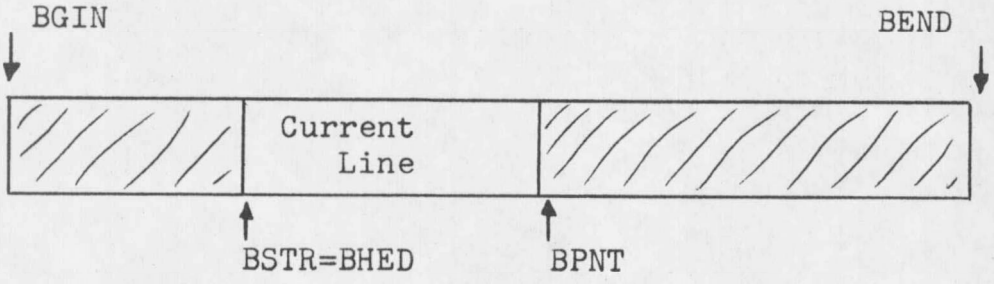
TABLE 2. BASE PAGE LINKS

NAME	TYPE	COMMENTS
MAIN	Word	Points to word 0 of the teletype table of user program currently in core. If none present, MAIN = 0.
LIB	Word	Points to the location in the command table of the disc address of the library routine in core. LIB = 0 when none is present.
PLEX	Bit	Bit = 1, port is full duplex. 0 = half duplex.
UNABT	Bit	Bit = 1, no abort allowed.
ABTRY	Bit	Bit = 1, abort attempted.
ENDSK	Word	0 if disc not busy.
TIMEF	Word	1 when current program is timed.
MPCOM	Bit	Bit = 1 for multiplexor message to scheduler.
IOTOG	Bit	Bit = 1 for input; 0 = output.
TAPEF	Bit	Bit = 1 for channel in TAPE mode.
CHNFG	Bit	Bit = 1 for CHAIN running.
HFLAG	Bit	Bit = 1 for $HELLO running.
TERR	Bit	Bit = 1 if error occurred while reading tape.
CFLAG	Bit	Bit - 1 if program is in compiled mode.
PHM	Bit	Bit = 1 if carriage return received.
PHT	Bit	Phones are timing when bit = 1.
PHR	Word	Number of seconds allowed to log on system.

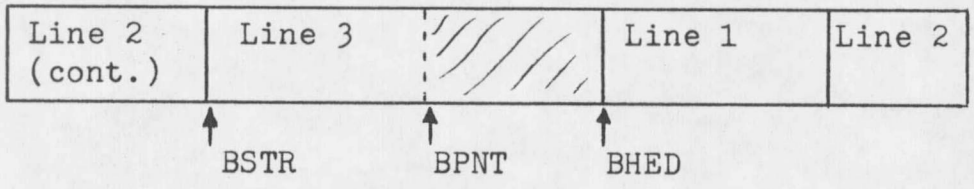
returns, the program restarts the computer using the saved system status. This insures that BASIC will be restored to the same operating state as existed prior to the power failure.

Input/Output buffers. Each channel on the time share system has a 50 word (100 character) input/output buffer for communication between the multiplexor and TSB. Each of these is a "circular" buffer, having a physical start and end address, and a logical start and end address. The time share system always communicates directly with these buffers. Output operations merely place characters in the appropriate slot in the buffer. Input retrieves a character from the buffer. Figure 6 shows the three cases which are allowed to exist within the I/O buffers.

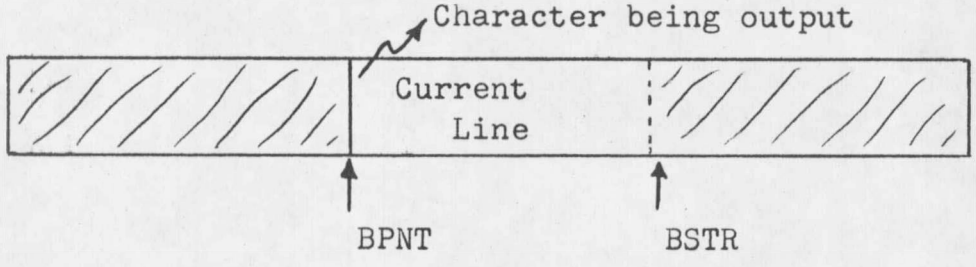
In each of the three cases shown in Figure 6, the variable BGIN is a fixed value pointing to the start address of the user's physical buffer. BEND is the name of a variable pointing to the first character following the physical buffer. It is a constant value equal to BGIN+50. While in input mode, BPNT points to the character location into which the next character will be deposited. During output, it points to the location of the character currently being transmitted. BSTR points to the first character of the most recent buffer during input. On output, it points to the location into which the next character will be deposited. BHED is used only during input. It points to the first character in the first logical buffer.



(A) Channel in normal input mode.



(B) Channel in TAPE mode. User has been Queued.



(C) Channel in output mode.

Figure 6. I/O Buffer States

BASIC language processor. The BASIC language processor of WTC-1000D TSB is essentially the same as that for HP2000B. This is a completely re-entrant program which is responsible for syntax checking, program compilation and decompilation, and for execution of BASIC programs. A complete BASIC processor description is beyond the scope of this thesis. Complete documentation of the HP2000B processor may be obtained directly from Hewlett Packard Company.[6]

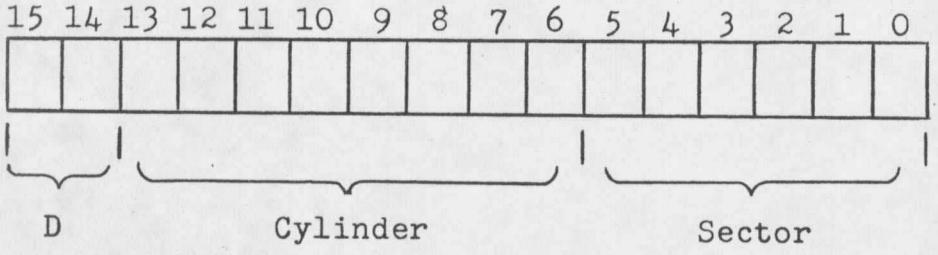
Disc driver. The TSB disc driver is responsible for all data transfer to and from the system discs. Any module of the system may call the disc driver to perform a disc transfer. Three parameters are required for each disc transfer. The first is the disc address in the A-register given in standard format as shown in Figure 7.

The disc address word is interpreted automatically by the disc driver. The driver determines how bits 15 and 14 are to be decoded depending on which disc drive is being used with the system.

The second parameter passed to the disc driver is the core address from which data is to be read or into which data is to be written. This parameter is passed in the B-register. Bit 15 is set to one for a read from disc to core. Bit 15 is zero for a transfer from core to disc.

---

[6] Crandall, Ron, and McEvoy, Dennis, 2000B Time Shared BASIC Internal Maintenance Specifications, pp. 171-226.



D-field (HP7900 disc drive)

Bit 15 = drive number (0-1).
Bit 14 = 0 for cartridge disc within drive.
Bit 14 = 1 for fixed disc within drive.

D-field (HP7901 disc drive)

Bits 15-14 = drive number (0-3).

Cylinder (HP7900 and HP7901)

Bits 13-6 = track number (0-202).

Sector (HP7900 and HP7901)

Bits 5-0 = sector number (0-47).

Figure 7. Disc Address Word

The third parameter is the number of words of data to be transferred. This value is given as a negative number in the base page link entry WORD.

Upon initiation of a disc transfer, the variable ENDSK is set to one. It is cleared by the disc driver upon completion of data transfer. This value is tested by each module prior to performing a disc transfer to insure that the disc is available for use.

System console driver. The operation of the system console driver is determined by two flags located on base page, T35F1 and T35F2. Table 3 describes the operation of the console as signified by these flags.

TABLE 3. SYSTEM CONSOLE STATES

T35F1 STATE	T35F2 STATE	CONSOLE DRIVER OPERATING MODE
0	0	Driver is accepting input (normal state).
0	-1	1) input command received and being processed, or 2) output terminated from a system command which is to be reinitiated.
-1	0	Normal output occuring.
-1	-1	Outputting, at the end of which the current system command will be reinitiated.

The console driver is always in the input wait mode unless processing a command or printing. System modules may call this driver by placing the number of characters to be printed in the A-register with bit 15=0 if a carriage return/linefeed is to be appended. The B-register is set to the core address of the output buffer. If bit 15=1 in the core address word, punching will occur in addition to printing.

Multiplexor driver. The multiplexor driver is responsible for handling data communication between the user terminals and the input/output buffers. Communication with up to 16 terminals is handled by the WTC-200B hardware multiplexor. Complete programming specifications may be found in the WTC-200B reference manual.[7]

Output to the multiplexor driver is performed on a character by character basis using the output a character subroutine OUTCH. The character to be output is placed in bits 6-0 of the A-register. The address of word 0 of the user's teletype table is placed in the B-register. Data is then transferred using a JSB OUTCH,I instruction.

The OUTCH subroutine places characters into a user's buffer until it is filled (99 characters), at which point the user is suspended. Since BASIC is fully re-entrant, this poses no problems

---

[7] Western Telecomputing Corporations, Instruction Manual for the WTC-200B Communication Multiplexor, pp. 48.

for that module. Other modules of the system are not allowed to suspend a user since they are not re-entrant. These modules must wait until the buffer is empty before they attempt to output data to any channel. Furthermore, they must never let OUTCH suspend their operation. Most of these routines typically only partially fill the user's output buffer then call a special suspension routine which saves the current run-status.

Input from a user channel is allowed only when the user is in idle or input status, or while entering a program under TAPE control. Upon completion of input (carriage return received) a test is made to determine if the channel is in TAPE mode. If it is, the appropriate MPCOM bit is set to indicate to the scheduler that input is ready for processing. If not, a bit in PHM is set which causes the clock driver to set the appropriate MPCOM bit 200 milliseconds later. This prevents the teletype from getting out of synchronism with the multiplexor driver. The scheduler will initiate the appropriate action upon detecting the MPCOM bit as being set.

While a program is running and not printing data, the port is set to the idle mode during which input is allowed to occur. If the "BREAK" key is pressed for at least 1/10 of a second, the multiplexor driver will set an abort request condition to stop the program. While a program is printing, the port is set to the full duplex mode. This allows data to be input even while data is being

output to that channel. Pressing the "BREAK" key during output will also set the abort condition.

The multiplexor driver recognizes when an abort attempt has been requested. It checks the port's status to determine if an abort is allowed. If the user is running a library program other than Library or Catalog, the abort attempt is ignored since the routine may be in the process of updating system tables. At other times when aborting could cause loss of system information, the port's UNABT bit is set. When an attempt to abort is made, the driver will not set the stop condition but will set a bit in ABTRY. Routines which set UNABT always call ABCHK when aborts will no longer cause any problems. ABCHK tests ABTRY and aborts the user if the channel's bit was set.

Clock driver. The clock driver is the central point through which control is passed to the scheduler. Entry comes to the driver whenever the clock (real time scalar) interrupts. This event occurs every 100 milliseconds. Besides giving control to the scheduler, the clock driver also performs the following tasks: During normal system operations, the console teletype motor is turned off. Logging operations and user messages automatically turn the motor on and set up a two second delay to allow the motor time to reach operating speed. The clock driver is responsible for initiating output on the system console after the two seconds has elapsed.

When a user terminal is in the input mode and receives a carriage return, the multiplexor driver sets up a 200 millisecond delay. The clock driver tests each channel for this condition and sets the port's MPCOM bit after the delay has timed out.

The clock driver also keeps the BASIC time of day clock. This is a real-time clock which keeps track of second of day, hour of day, day of year, and century. Finally, the clock gives control to the scheduler.

Scheduler. The basic philosophy of the TSB scheduling algorithm is to provide short response times for short, interactive jobs at the possible cost of delays in longer running jobs.[8] The implementation of this involves a queue of jobs to be run arranged according to a priority scheme. The queue is a list of from 1 to 18 entries, each entry pointing to the next entry, and the last entry pointing back to the first. The 18 entries in the queue are the 16 LINK words in the user teletype table, a LINK word in the console teletype table, and a queue head. At the head of the queue are the locations MLINK, MILNK+1, and MLINK+2. These entries are always on the queue. The priority of the queue is stored in MLINK+2. It is always 77777_8 and is the last entry in the queue. MLINK+1 is the pointer to the first

---

8 Crandall, and McEvoy, op cit., pp. 17-19.

entry on the queue. In the case of an empty queue, MLINK+1 points to itself. Each entry on the queue has a priority no greater in value than that of the one it points to. In adding a new entry to the queue, the new entry is placed just ahead of the first entry with a larger priority. Figure 8 gives an example of a queue containing users one, four, and seven in addition to the system console.

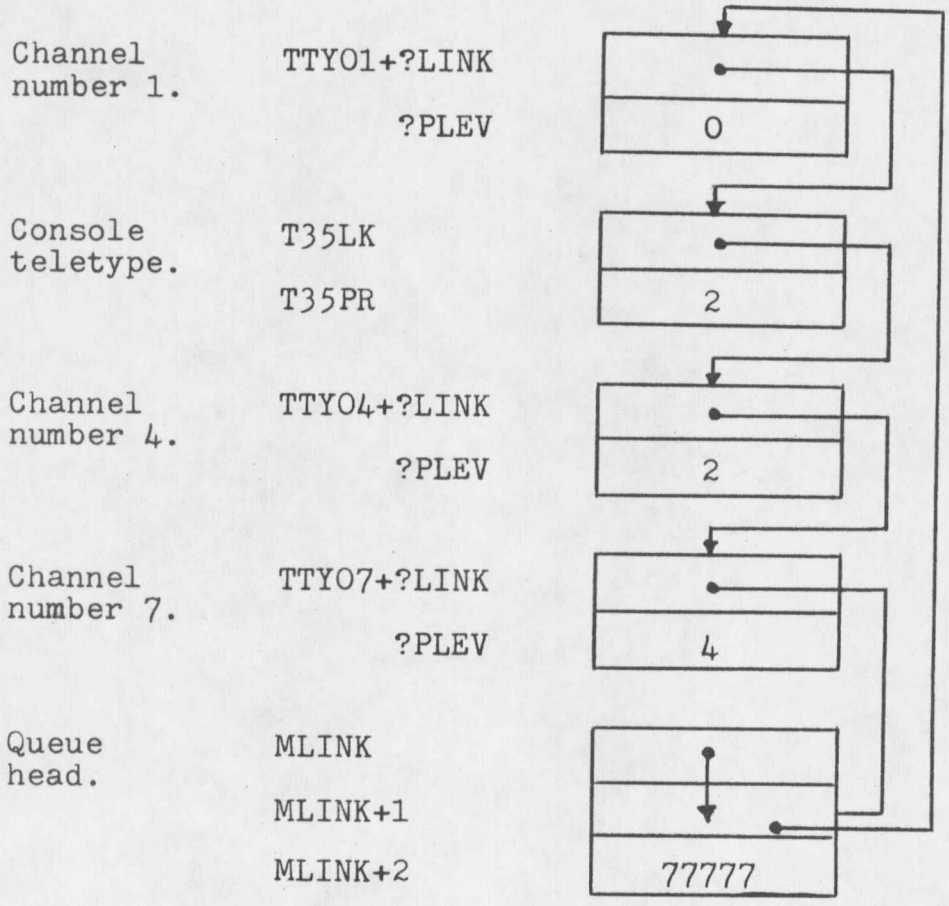


Figure 8. System Queue

The scheduler uses the following rules to assign priorities:

1. Job priorities assigned upon initial placement in queue:
   A. SYNTAX and jobs returning from I/O suspend :0
   B. BASIC commands (RUN,LIST,PUNCH) :1
   C. DISC Resident commands (GET,SAVE,etc.) :2
2. Jobs are reassigned priorities as follows:
   A. Jobs of priority two are reassigned priority zero upon reaching the top of the queue.
   B. RUN jobs which have exceeded their time sliced are assigned priority four and repositioned in the queue. Each job is assigned a time slice of one second.

System Library Overlay Programs. The last 256 words of non-protected memory (37300_8 - 37677_8) are reserved for execution of system library overlay programs. These are programs which are disc resident and called into memory only when needed to perform a special function, such as log a user on or off the system. These programs read into the overlay area and either run to completion or self suspension during output. Parameters required to restart the overlay program are saved in the user's work area. These programs use the user swap area of 4500 words as a system routine work area. Tables are built and modified in this area by the overlay programs. Table 4 lists the system overlay programs and describes their functions.

TABLE 4. SYSTEM LIBRARY OVERLAY PROGRAMS

NAME	DESCRIPTION
FUSS Table	Table indicating files usage by the users.
FILES	Processes FILES statements in user programs.
CHAIN	Processes CHAIN statement syntax.
CHAIN overlay	Executes CHAIN statement.
ASSIGN	Processes ASSIG statement syntax.
ASSIGN overlay	Executes ASSIGN statement.
SAVE	Stores a program in the user library.
CSAVE	Stores a program in semi-compiled form.
SUPERSAVE	Called by SAVE, CSAVE, and OPEN to redistredistribute the disc directory on a full disc to gain disc space.
GET	Retrieves a program from the library.
APPEND	Adds a program from the disc onto a program in user area.
HELLO	Logs a user on the system.
BYE	Logs a user off the system.
KILL	Removes a program or file from the library.
RENUMBER	Reassigns the sequence numbers of a user program.
NAME	Gives a program a name.
CATALOG	Prints the names and lengths of all programs and files in the user's library.

TABLE 4 (Continued)

NAME	DESCRIPTION
LIBRARY	Prints the names and length of all programs and files in the public library.
WIDTH	Assign terminal print length from 20 to 132 columns.
DELETE	Deletes sections of a user's program.
TIME	Prints a listing of terminal time used since log on, and total time used for the account.
DISC	Prints a listing of disc space used to date, and total disc space allowed for the account.
MESSAGE	Sends a one line message from the user terminal to the operator's console.
PROTECT	Used by A000 to make programs read-only.
UNPROTECT	Allows full access to a previously protected program. (A000 only).
OPEN	Creates a data file of the length specified by the user.
LENGTH	Prints the length (in words) of the user's program.
ECHO	Sets a terminal to full or half duplex.
REPORT	Prints a report of all user accounts, giving total connect time and disc space used.
DIRECTORY	Prints a report of programs and files on the TSB discs.
STATUS	Prints a report describing where TSB tables are stored.

TABLE 4 (Continued)

NAME	DESCRIPTION
ROSTER	Produces a listing indicating active ports on TSB.
DATE	Changes the status of TSB's real time clock.
RESET	Resets terminal connect times to a specified value.
CHANGE	Changes parameters in user ID table.
SLEEP	Initiates a systematic shutdown of the TSB system.
NEWID	Adds an account to TSB.
KILLID	Removes an account from TSB.
MOVE	Moves a program or file from one disc subchannel to another.
PURGE	Removes programs and files from TSB which haven't been referenced since a specified date.
PHONES	Enables/disables the phone logic.
FORMAT	Formats a user disc.
COPY	Copies one disc subchannel to another disc subchannel.
DISC	Adds a new disc subchannel to TSB or removes a disc subchannel from TSB.
ERROR TABLES	Six overlay programs containing error message text.

TSB SYSTEM TABLES

System tables provide the cohesive framework that holds system operation together. This section describes the function and organization of five system tables: (1) user ID tables, (2) directory, (3) disc availability table, (4) files usage table, and (5) command table.

ID Table. The ID table (IDT) is a disc resident table which contains one 8-word entry for each user ID code in the system. The entries are kept sorted according to the ID codes. A000 would be the first code, Z999 the last. Each entry has the form shown in Table 5.

TABLE 5. ID CODE ENTRY FORMAT

WORD	CONTENTS
0	User ID code.
1-3	PASSWORD.
4	Time allowed in minutes.
5	Connect time used to date.
6	Disc allowed in sectors.
7	Disc used to date.

The user ID is the internal representation of the user identification code such as A010. Bits 14 through 10 identify the ID letter (A = 1, B = 2, ..., Z = 32_8) and bits 9 through 0 identify the ID number (0-999). Password is a one to six character string. Zeros are appended if less than six characters long. Time allowed and used is given in minutes. Disc allowed and used is given in sectors. Words four through seven are 16-bit integers in the range 0 to 65535. The following two words, located on base page, refer to the IDT:

IDLOC = disc address of IDT.

IDLEN = length in words of IDT.

Directory. The directory is a table which contains all necessary information about each program or file in the system library. There are two directories on each disc. A directory entry consists of eight words and has the format shown in Figure 9.

Word	Contents	Comments
0	User ID Code	
1	Program	Bit 15=1 if protected
2	or File	Bit 15=1 if entry is a file
3	Name	Bit 15=1 if semi-compiled
4	Start of program ptr.	Starting core address
5	Date	Date entry last referenced
6	Disc address	
7	-length in words	

Figure 9. Directory Entry Format

The user ID is the system representation of the user ID code in the same format as in IDT. Words one through three are the name of the program or file in ASCII characters, packed two characters per word. If fewer than six letters, spaces (40_8) are appended. The start of program pointer is used when "Getting" programs to tell where in core to start reading in the program. It is used to allow COMMON data to be passed between programs. Date is the day-of-year and year the program or file was last accessed. It is used by the PURGE command to remove infrequently used programs and files from the system. The disc address in standard disc format and program length are given for system access to the entry.

Entries are made in the directory alphabetically according to words zero through three. There are two pseudo entries in the directory table to denote the beginning and end of the directory. They have the form as shown in Figure 10.

0	0
1	0
2	0
3	0
4	0
5	177777
6	0
7	0

First Entry

0	177777
1	177777
2	177777
3	177777
4	0
5	177777
6	0
7	0

Last Entry

Figure 10. Directory Pseudo-Entries

A core resident table called DIREC exists on base page. It contains information about the disc directories. Its structure has the form shown in Table 6.

TABLE 6. BASE PAGE DIREC FORMAT

Word	Contents
0	-length in words of first directory track.
1-4	Same as first four words of first disc directory track.
5	Unused.
6	Disc address of first directory track.
7-13	Same as 0-6 but applied to second directory track.
14-20	Same as 0-6 but applied to third directory track.
21-27	Same as 0-6 but applied to fourth directory track.
28-34	Same as 0-6 but applied to fifth directory track.
35-41	Same as 0-6 but applied to sixth directory track.
42-48	Same as 0-6 but applied to seventh directory track.
49-55	Same as 0-6 but applied to eighth directory track.

A disc address of zero implies that there is no such directory track. When word 0 is zero, words 1-4 are meaningless. DIREC values are assigned values as given in Table 7.

TABLE 7. DIREC VALUES

Word	Contents
0-13	Directory information for disc subchannel 0.
14-27	Directory information for disc subchannel 1.
28-41	Directory information for disc subchannel 2.
42-55	Directory information for disc subchannel 3.

Disc Availability Table. The available disc table (ADT) is a disc resident table which contains one two-word entry for each area of the disc which is unallocated. Figure 11 shows the format of an ADT entry and the terminating pseudo entry.

0	Disc address
1	Length of Area

ADT Entry

0	177777
1	0

Terminating ADT Entry

Figure 11. ADT Entry Format

ADT entries are sorted according to word zero. Entries are originally set one for each track available on the disc with word one set to the maximum number of sectors on that track. As programs are stored on the discs, the ADT is searched for the first entry containing sufficient disc storage space. After the program has been stored, the ADT entry sector count will be decreased by the amount used. A length of zero sectors indicates a track has been filled and is no longer available for program storage. As programs and files are removed from the system, the released area is added back to the ADT. Two ADT entries containing disc space released in adjacent areas will be combined to form a single ADT entry.

Besides the entries for unallocated areas, there are ADT entries for each of the two loader tracks, five TSB system tracks, 16 user tracks, IDT track, ADT track, and two disc directory tracks. Word one of each of these entries is set to zero so they will not be allocated for program storage. In addition, during the loading process, all defective tracks are removed from the ADT. The terminating ADT entry as shown in Figure 11 denotes the end of the table. Since track zero is always allocated as a system track, any possible disc address is guaranteed to be bounded by two ADT entries. The following two words on base page refer to the ADT:

ADLOC = disc address of ADT.

ADLEN = -length of ADT in words.

Files Usage Table. The files usage table (FUSS) is a disc resident table used to control access of files by multiple users. It is divided into 16 subtables of 16 words per user. Each subtable lists the disc addresses of files currently being accessed by the corresponding user. FUSS is essential for preventing simultaneous write access by two users of the same file, except for privileged and semi-privileged users, and for preventing Killing a file when a user has access to it. The first user to access a file obtains write capability. All subsequent, but simultaneous, users get read-only access except for users A000 through A999 who always get read-write access.

A user's FUSS table entries are set by the FILES and ASSIGN routines. These are both called from BASIC programs. FUSS entries are cleared by BYE, HELLO, KILLID, and sometimes by KILL.

The disc address of FUSS may be obtained by the instruction:

```
LDA  FUSS,I
```

Command Table. The command table (COMTABLE) is a list of all user and system operator commands, containing their ASCII codings and core or disc addresses. The structure of COMTABLE is defined in Table 8.

TABLE 8. COMTABLE FORMAT

Section Label	Description
COM1	Codes for commands which are executed immediately by the system.
COM2	Codes for commands which are executed by BASIC.
COM3	User commands which are executed by disc resident programs.
COM4	System commands, all of which are executed by disc resident programs.
COM5	Core starting addresses for those commands which are listed under COM1 and COM2.
COM6	Disc addresses for those commands listed under COM3 and COM4. These addresses are filled in by the TSB Loader.

Each command in COMTABLE is recognized by the subroutine SCOM (search for command) only by its first three letters. This search routine takes the first three letters of any line input without a statement number and converts each letter into a number from 0 to 31_8, and then packs the three numbers into one word as three five bit bytes. An example of HELLO is given in Figure 12.

Letter:	H	E	L	L	O	
ASCII	110	105	114	X	X	X = ignored
Offset	101	101	101			
	7	4	13	difference		

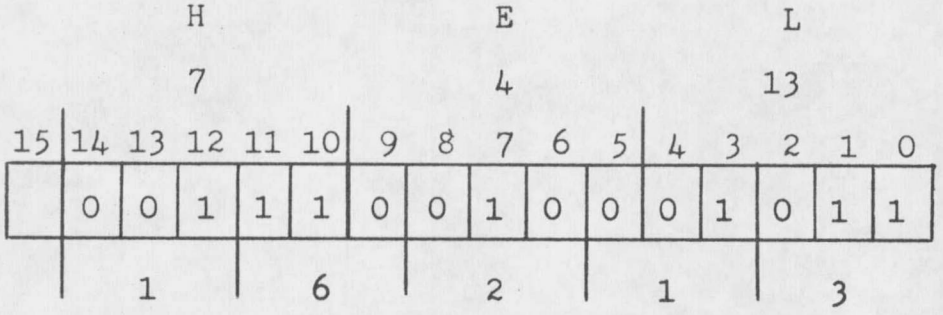


Figure 12. COMTABLE Entry Coding Format

As shown in Figure 21, the COMTABLE entry for HELLO is 16213_8. In addition, all operator commands have bit 15 set to one. This indicates the command is valid only from the system operator console. Some codes in COM2, COM3, and COM4 are set to minus one (all ones). These do not correspond to any possible three letter code. Their function is to generate room in COM5 and COM6 for core and disc addresses of routines called indirectly (such as FILES) or for tables (such as FUSS). In the case of CTAPR, the purpose is to generate a

status type for printing compiler tape errors without a direct command from the user.

DISC ORGANIZATION

1000D BASIC is designed to operate on either the HP7900A disc drive or the HP7901A disc drive. TSB can operate up to two HP7900A drives or four HP7901A disc drives. Either system provides for the same amount of total disc space. The disc drive type is specified during system loading. This section describes format of the HP7900 disc system, format of the HP7901 disc drive, system disc format, and user disc format.

HP7900 Disc Format. Two HP7900 disc drives may be used on TSB. Each drive contains a fixed disc and a removable disc. Discs are arranged into subchannels from zero to three. A minimum TSB system would consist of subchannel zero which corresponds to the first removable cartridge in disc drive one as shown in Figure 13.

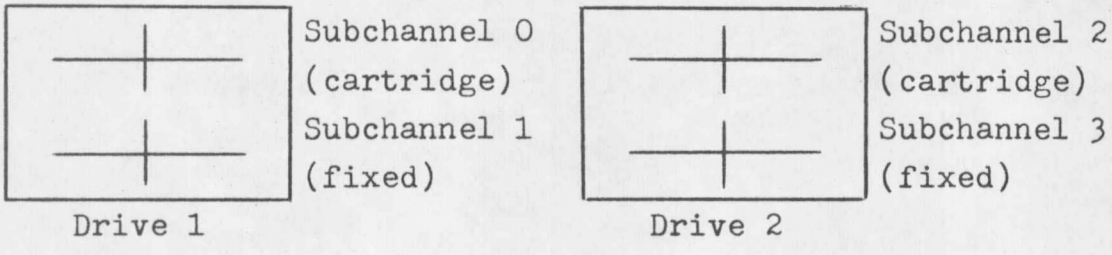


Figure 13. HP7900 Disc Organization

HP7901 Disc Format. Four HP7901 disc drives may be used on TSB, giving exactly the same configuration as exists on TSB with HP7900 disc drives. However, use of HP7901 drives allows greater flexibility in disc handling since all discs are removable cartridges. Figure 14 shows the organization of TSB using HP7901 drives.

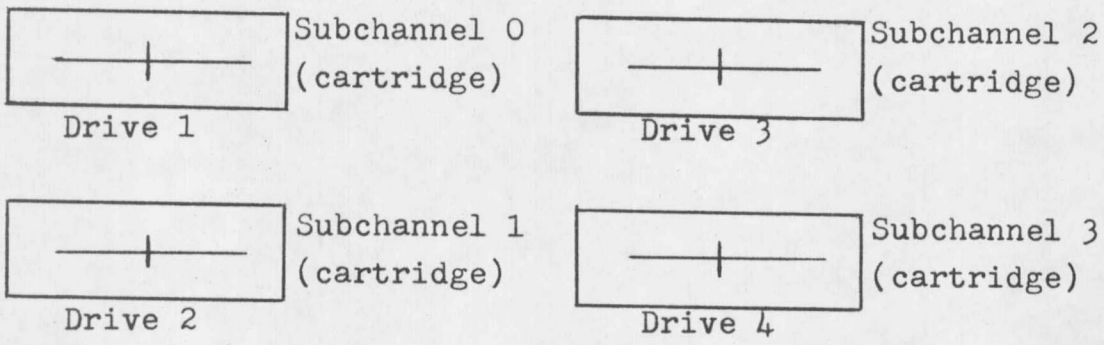


Figure 14. HP7901 Disc Organization

All discs are directly interchangeable between 7900 and 7901 disc drives. Discs in subchannels one through three are interchangeable and may be run on any other existing subchannel. However, the disc in subchannel zero may only be run in subchannel zero since TSB system and system tables are stored on that subchannel.

System Disc Organization. The disc in subchannel zero is referred to as the system disc and must be present on a minimum TSB system. It contains a copy of the TSB system, user swap tracks,