In addition to the changes needed to correct the HP 2000A TSB System Version E, the following new features have been added:

1. **All lines printed on the ASR-35 by the commands REPORT and DIRECTORY will be terminated by XOFF CR LF instead of CR LF.**

2. **The HP7970 Magnetic Tape Unit may be substituted for the HP 3030 Magnetic Tape Unit. Users of HP 7970 should type:**

   ```
   MAGTAPE - select code *
   ```

   when entering MAG TAPE commands. (The * must appear if SC #0)

3. **If a HP 7970 is being used, the status command will print an * after the Magtape select code.**

Reasons for HP 2000A TSB changes were to correct the following:

1. **Using = signs inside parameters to functions can cause fatal errors when the program is run. e.g.** A = INT (A=0)

2. **Redimensioning matrices to total size > 32767 can cause fatal errors. e.g.** MATA = ZER (1000, 1000)

3. **In the CAT command, random shipping within the directory can occur at the end of each line. This is non-fatal.**

4. **If a user hangs up his phone or exhausts his log on time simultaneously with typing CR or break, he may crash the system.**

5. **If a user in tape mode hangs up, the next user to come on that port is still in tape mode.**

6. **Disc errors occurring in the loader are printed incorrectly.**

7. **At the end of a magtape SLEEP, a verify procedure has been added. The following message is printed:**

   ```
   VERIFY?
   ```

   A response of NO will result in the system printing DONE and halting. A response of YES will cause the tape to be reread and checked for validity. If it is valid, the system will print DONE and halt. Otherwise, it will print TAPE BAD OR TOO SHORT, and halt. The dump can be restarted by pressing RUN.
USER SWAP TRACKS
(16 OR 32 TRACKS)

LIBRARY STORAGE
(REMAINING TRACKS)

SYSTEM I.D. TABLE
AVAILABLE DISC TABLE
(1 TRACK)

DIRECTORY
(1 TRACK)

COPY OF CORE
RESIDENT
SYSTEM
(3 TRACKS)

SYSTEM DISC
RESIDENT
Routines
(2 TRACKS)

LOGICAL DISC 0
SLEEP COMMAND

PURPOSE: PROVIDE FOR SYSTEMATIC SHUTDOWN OF SYSTEM.

FORMAT: SLEEP — CHARACTER STRING (MESSAGE)

EFFECT:
1. MESSAGE SENT TO ALL ACTIVE USERS
2. ALL USERS ARE DISCONNECTED
3. THE CURRENT SYSTEM IS DUMPED TO DISC
4. LIBRARY TRACKS ARE PACKED FOR EFFICIENCY PURPOSES.
5. IF MAGNETIC TAPE IS PRESENT, THE SYSTEM IS DUMPED TO MAG TAPE
6. COMPUTER HALTS

NOTE: SLEEPING A SYSTEM TYPICALLY REQUIRES 5 MINUTES.
STA

IDT 0/21/000 1160  ADT 0/21/019 0304

DIREC
0/22/000 1424  0/23/000 2272  0/24/000 1784  0/25/000 1568

SYSTEM 0/00/000 0/01/000 0/02/000 0/03/000 0/04/000

USERS
0/05/014 0/06/002 0/07/000 0/08/000 0/09/000 0/10/001 0/11/005 0/12/00
0/13/000 0/14/032 0/15/000 0/16/005 0/17/000 0/18/000 0/19/000 0/20/00

MAG= 17  PHONES= 16  DISC= 14-128 14-128 14-128 00-300

TRACKS
0 0000000000000000000000000000000000000000000000000000000000000000
1 0000000000000000000000000000000000000000000000000000000000000000
2 0000000000000000000000000000000000000000000000000000000000000000
3 1111111111111111111111111111111111111111111111111111111111111111

SAMPLE STATUS
OVERVIEW OF INTERNAL TIME SHARING OPERATION

I. Core Map Elements
A. Scheduler (TSB heart) - Schedules/ initiates/ suspends/ terminates tasks.
B. BASIC interpreter - syntax checking; RUN, LIST, PUNCH
   1. Re-enterable processor - i.e. one copy shared among all users
C. Swap area
   1. User swap area (1240B → (1240B + 5440), each user swapped in/out between (the one) in-core swap area and that user's particular disc swap track.
   2. System library program work space (1244B → (1244B + 15440)
D. System Library Program, (SLIP) Overlay Area
   System routine execute area (user & operator commands)
   1. Programs only read in -- i.e. no need to swap out.
   2. SLIP's run to completion, or self suspension (during output).
E. Drivers
   1. MPXR - Multiplexes input/output from/to the 16 user teletypes.
   2. Disc Driver - uses DMA 3,7 (2,6 not used)
   3. Console (TTY35) - handled separately from users.
      Requires tty board.
   4. Power Fail - System should be fully restored upon power failure.
F. TTY tables - in-core tables containing user buffer information, and scheduling status.
   1. One for each user.

II. Disc Map Elements
A. Swap tracks - 1/user
B. Core-resident copy of system
   1. Mainly for re-loading
   2. Portions also used for SLEEP
C. System Library Programs (disc-resident)
D. DIRECTORY track (s) - 1/64 tracks (i.e. each additional 64 tracks require another DIRECTORY track).
E. IDT/ADT (I.D. table, available disc table) track.
F. Remainder available for user files/programs.
III. Interaction
A. Swap tracks to/from swap area
B. SLIP's to SLIP overlay area
C. DIRECTORY, IDT, ADT to/from SLIP work-area (user swap area) when being examined/modified.

[Scheduler function (objective): To schedule, initiate, suspend, or terminate tasks]

IV. Tasks
A. Def: A body of programming work to be executed by the CPU. e.g.:
   1. Syntax processing for user BASIC statement
   2. User commands (RUN, CAT, REN, ...)
   3. Operator commands (ROS, DIS, SLE, ...)
B. Each port (not user) associated with a task; plus operator console.
   1. Therefore, maximum of 16+1 ready tasks can request service, simultaneously.
C. Types of tasks
   1. Core-resident -
      a. Executed immediately: SCR, TAP, KEY
      b. Scheduled: Syntax processing, RUN, LIST, PUNCH.
      c. Use swap area for BASIC programs(swapped in/out).
   2. Disc-resident - all other (user and operator) commands.
      a. Executed in SLIP overlay area
      b. Use user swap area for:
         i. BASIC program (e.g. REN, APP,...)
         ii. Work space (examining/modifying DIRECTORY, IDT, ADT).
      c. No need to swap program out (because they run to completion, or self-suspension).

V. Scheduling - deciding who runs, when, and begin executed of appropriate task.
A. Only one task has control of CPU at any time. Therefore, when a task is ready to run, it is placed in a waiting list (the queue). Queue is list of those tasks ready to be executed.
B. Placement in queue depends on:
   1. When (first in, first out)
   2. Priority level (PLEV) - (queue is actually a queue of four sub-queues. i.e. It is "last-in, last-out" within any given priority level).
C. STAT - status of task determines its priority level. Possible statuses:
   1. idle
   2. I/O suspend
   3. Syntax processing
   4. RUN, LIS, PUN
   5. Other commands
D. Priority Philosophy: "Service the interactive user quickly by giving him high priority at the expense of long-running compute-bound programs" (i.e. Users are more concerned with slow syntaxing and slow interaction, than long execution time).

Priority Scheme: (PLEV):

0: BASIC syntax
   interactive programs
1: Short RUN programs
   LIS, PUN
2: Dis-resident routines (other user commands, operator commands)
4: Compute-bound programs
   (1 second time-slices).

VI. Scheduler Functions
A. Task scheduling = placing in queue
   1. BASIC line, CR Syntax
   2. User command, CR Command
   3. Operator command, CR Command
   4. Disconnection Bye
B. Task initiation = begin execution of top-of-queue
C. Task suspension = removing from queue temporarily
D. Task re-scheduling = placing back in queue
   1. return from I/O wait
E. Task termination = removing from queue
   1. Normal task end (incl. BYE, SLE)
   2. User abort
F. Miscellaneous scheduler functions:
   1. Bump top-of-queue, due to higher priority request
   2. Time-out RUN jobs. = placing job at bottom of queue
Address

8

TSB CORE MAP

Base Page

1240

User Swap Area

and

Slip Work Space

14000

Basic

Interpreter

30734

Drivers

33015

Scheduler

37300

System Library Program

Overlay Area

BBDL

Disc 30734

Power Fail 31062

Console 31336

MPXR 32123
CORE RESIDENT TASKS

Base Page

BASIC Program

BASIC Interpreter

Drivers

Executive

SLIP Overlay Area

BBDL

Scheduled

Syntax Processing

RUN

LIS

PUN

SCR

TAP

KEY

Executed Immediately
DISC RESIDENT TASKS

Base Page

BASIC Program
or
Work Space

BASIC Interpreter

Drivers

Scheduler

SLIP

BBDL
QUEUE INSERTION
ACCORDING TO TIME & PLEV

TO BE INSERTED: 8 0 3 0 9 1 7 4 4 0 1 2 2 1

QUEUE:

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8 0</td>
<td>8 0</td>
<td>8 0</td>
<td>8 0</td>
<td>8 0</td>
<td>8 0</td>
<td>8 0</td>
</tr>
<tr>
<td></td>
<td>3 0</td>
<td></td>
<td>3 0</td>
<td></td>
<td>3 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9 1</td>
<td></td>
<td>9 1</td>
<td></td>
<td>9 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>7 4</td>
<td></td>
<td>7 4</td>
<td></td>
<td>7 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 2</td>
<td></td>
<td>1 2</td>
<td></td>
<td>1 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 1</td>
<td></td>
<td>2 1</td>
<td></td>
<td>2 1</td>
<td></td>
</tr>
</tbody>
</table>

PLEV
PORT NO.
THE MULTIPLEXOR
HP TELEPRINTER MULTIPLEXER INTERFACE

- Permits up to 16 Teleprinters or Bell system data sets, or any combination of the two, to be connected to a 2116B.

- Provides for bit serial transfer of data between the computer and the external device.

- Permits simultaneous input and output.

- The card contains and 880 HZ oscillator which is used by software to establish the sampling rate of the input lines.

- Due to its relatively rapid, asynchronous interrupt rate, it should be assigned a relatively high priority I/O address.

OPTION 01

- Auto disconnect option - allows individual status for each data set to be controlled by computer. This provides protection against "housewife" calls.

- Also, if an input port is inactive for a time established by computer software, the computer causes a disconnect. The assumption here is that the operator at the remote terminal has completed transmissions, but has not properly terminated the data set.
START BIT 8 INTERRUPTS
EACH DATA BIT 8 INTERRUPTS
STOP BITS 16 INTERRUPTS
TOTAL INTERRUPTS/CHARACTER = 88

MARK "1"

SPACE "0"

INTERRUPTS 12

INTERRUPTS 12
11 - MULTIPLEXOR INTERRUPT DRIVER SENSES BEGINNING OF CHARACTER TTY #1
120 - MID SENSES BEGINNING OF CHARACTER FOR TTY #2 AND ALSO THAT IT MUST
SAMPLE SECOND BIT OF CHARACTER FORM TTY #1
144 - MID SENSES BEGINNING OF CHARACTER FOR TTY #3 AND ALSO TO SAMPLE
5TH DATA BIT OF TTY #1 AND TO DO NOTHING WITH TTY #2
156 - SAMPLE BIT 4 FOR TTY #2 AND SAMPLE BIT #1 FOR TTY #3 AND DO
NOTHING FOR TTY #1
172 - END OF CHARACTER FOR TTY #1; SAMPLE BIT 6 OF TTY #2; SAMPLE BIT 3
OF TTY #3
MPX DRIVER

1. DETERMINE WHEN A START CHARACTER BIT HAS BEEN SENT FROM MULTIPLEXOR

2. ONCE DETERMINED, SET UP WHEN TO SAMPLE SUCCEEDING BITS

3. HOW TO DISTINGUISH BETWEEN SUCCEEDING BITS
   a) DATA BITS
   b) PARITY BITS
   c) STOP BITS

4. ONCE DISTINGUISHED, PERFORM APPROPRIATE ACTION
   a) PACK BIT
   b) IGNORE
   c) IGNORE

5. HOW TO ECHO EACH BIT – START BIT INCLUDED

6. WHEN IS CHARACTER COMPLETE

7. BUFFER A LINE

8. HANDLE CHARACTERS OF SPECIAL SIGNIFICANCE

9. OUTPUT SCHEME

10. DETECT SUCCESSFUL ABORT REQUEST AND SIGNAL EXECUTIVE
<table>
<thead>
<tr>
<th>OCTAL</th>
<th>WORD NO.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>BTIM</td>
<td>Counter for interrupts between bit sampling (initially 4; 8 for each succeeding).</td>
</tr>
<tr>
<td>1</td>
<td>CHAR</td>
<td>Packing/unpacking of bits.</td>
</tr>
<tr>
<td>2</td>
<td>BCNT</td>
<td>Counter for number of bits within a character (10 on input, 11 on output).</td>
</tr>
<tr>
<td>3</td>
<td>MASK¹</td>
<td>24 (port #); e.g.: bit 4 set for port 4.</td>
</tr>
</tbody>
</table>
| 4     | CCNT     | Input: not used  
Output: negative number of characters to be transmitted. |
| 5     | BPNT     | Input: address² of where next character goes.  
Output: address² of character currently being output. |
| 6     | BSTR     | Input: address² of first character of line currently being input.  
Output: address² of last character in buffer. |
| 7     | BHED     | Input: address² of first character of first line not yet processed (significant only during tape made input).  
Output: not used. |
| 10    | BGIN¹    | Address² of beginning of physical buffer. |
| 11    | BEND¹    | Address² of first character beyond physical buffer. |
| 12    | LADR¹    | Address of LADDR entry. |

¹ A fixed parameter.  
² Character-address: bits 15-1 are word address; bit 0 is word-half (0 = left half, 1 = right half).
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISC</td>
<td>Dis-address of swap area.</td>
</tr>
<tr>
<td>PROG</td>
<td>Address of last core location used in swap area. Relevant only when user is not in core. (PBPTTR contains last address when user is in core (updated by interpreter).)</td>
</tr>
<tr>
<td>ID</td>
<td>User = I.D. 0 = no user on this port.</td>
</tr>
<tr>
<td>NAME</td>
<td>Program name.</td>
</tr>
<tr>
<td>PHON</td>
<td>Phone timing: value of DATIM+1 necessary to force disconnection.</td>
</tr>
<tr>
<td>TIME</td>
<td>DATIM, DATIM+1 upon logon.</td>
</tr>
<tr>
<td>ABCN</td>
<td>Counter for user abort request (114 msec).</td>
</tr>
<tr>
<td>CLOC</td>
<td>RUN timeout counter (1 second slices).</td>
</tr>
<tr>
<td>RSTR</td>
<td>Starting address when a task is first scheduled, or re-scheduled when returning from I/O suspend (PREG contains restart address when task is otherwise interrupted).</td>
</tr>
<tr>
<td>STAT</td>
<td>Status</td>
</tr>
<tr>
<td>-2</td>
<td>System disconnect</td>
</tr>
<tr>
<td>-1</td>
<td>User abort request</td>
</tr>
<tr>
<td>0</td>
<td>Idle</td>
</tr>
<tr>
<td>1</td>
<td>Aborting</td>
</tr>
<tr>
<td>2</td>
<td>Input suspend</td>
</tr>
<tr>
<td>3</td>
<td>Output suspend</td>
</tr>
<tr>
<td>4</td>
<td>Syntax processing</td>
</tr>
<tr>
<td>5-60</td>
<td>Command processing STAT number corresponds to COMTAE E.g.: 5=RUN, 6=LIST, 7=PUN, etc.)</td>
</tr>
<tr>
<td>LINK</td>
<td>Links queue: address of LINK word of next (lower) port on queue.</td>
</tr>
</tbody>
</table>
31 PLEV  Priority level, when on queue.
0 Syntax
   Returning from I/O suspend
   System library routines when they reach
   the top-of-queue.
1 RUN, LIST, PUNCH
2 System library routines until they reach the
   top-of-queue.
4 Compute-bound (RUN) programs.
<table>
<thead>
<tr>
<th>TSB VERSION E, TTY TABLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>BTIM</td>
</tr>
<tr>
<td>CHAR</td>
</tr>
<tr>
<td>BCNT</td>
</tr>
<tr>
<td>MASK</td>
</tr>
<tr>
<td>CCNT</td>
</tr>
<tr>
<td>BPNT</td>
</tr>
</tbody>
</table>
REFER TO THE MULTIPLEXOR NOTES FOR DESCRIPTIONS OF THE TELETYPING TABLES.
DISC-RESIDENT TABLES

DIRECTORY:

The DIRECTORY is a table which contains all necessary information about each program or file in the system library. It resides on the disc and may occupy from 1 to 4 disc tracks, depending upon how many discs there are on the system. A core resident table called DIREC contains information on the DIRECTORY itself.

A directory entry consists of 8 words and has the following format:

<table>
<thead>
<tr>
<th>WORD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>USER I.D.</td>
</tr>
<tr>
<td>1</td>
<td>PROGRAM OR FILE</td>
</tr>
<tr>
<td>2</td>
<td>NAME</td>
</tr>
<tr>
<td>3</td>
<td>UNUSED</td>
</tr>
<tr>
<td>4</td>
<td>DATE</td>
</tr>
<tr>
<td>5</td>
<td>DISC ADDRESS</td>
</tr>
<tr>
<td>6</td>
<td>-LENGTH IN WORDS</td>
</tr>
</tbody>
</table>
I.D. TABLE

The I.D. table (IDT) is a disc resident table which contains one 8-word entry for each I.D. code on the system. The entries are kept sorted according to the I.D. codes. An entry has the following format:

WORD: 0 user I.D.
       1-3 password (filled with 0's if fewer than 6 characters)
       4 time allowed (in minutes)
       5 time used (in minutes)
       6 disc allowed (in sectors)
       7 disc used (in sectors)

Words 4 - 7 are 16 bit quantities with values between 0 and 65535.
AVAILABLE DISC 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. An entry has the following form:

WORD
Ø disc address
1 length in sectors

Entries are sorted according to word Ø. Each entry may refer to as much as one full track, and no two consecutive entries ever refer to two adjacent disc areas (two tracks are not considered to be adjacent).

Besides the entries for unallocated areas, there is also one ADT entry for each of the five tracks on which the system itself resides, and for each of the sixteen tracks allocated for user swapping.

FUSS TABLE:

The FUSS table is a 128 word table which resides on the disc.

FUSS is divided into 16 sections of 8 words each. The 8 words in each section are the disc addresses of the user files currently being accessed by the user corresponding to that table. Addresses with bit 7=1 indicates the user has read only access.

The purpose of maintaining this table is to:

1) prevent simultaneous write access by two users to one file

2) prevent KILLing a file when some user has access to it
**DISC-RESIDENT TABLES**

### DIRECTORY

8 wd. entry/file, prog.

<table>
<thead>
<tr>
<th>User id</th>
<th>prog or file name</th>
<th>date</th>
<th>d-address</th>
<th>-len (wds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177777</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177777</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177777</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177777</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>177777</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(B(15) = 1: protected)

(B(15) = 1: file)

(Sorted by 1st 4 words without bit 15's)

**DATE:**

8

<table>
<thead>
<tr>
<th>year</th>
<th>day</th>
</tr>
</thead>
</table>

**IDT**

8 wd. entry/user

<table>
<thead>
<tr>
<th>User id</th>
<th>password</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>16 bit unsigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>time max(min)</td>
</tr>
<tr>
<td>time used</td>
</tr>
<tr>
<td>disc max(sect)</td>
</tr>
<tr>
<td>disc used</td>
</tr>
</tbody>
</table>

(Sorted by 1st 4 words)

**ADT**

2 wd. entry/space

<table>
<thead>
<tr>
<th>d-address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>-len (sect)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

177777

0/

(System tracks are 0-le)

(Sorted by 1st words)

**FUSS**

8 wd. entry/port

<table>
<thead>
<tr>
<th>file 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>file 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

(port 0)

<table>
<thead>
<tr>
<th>file 8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

(port 1)

<table>
<thead>
<tr>
<th>file 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>disc-addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>
EQUIPMENT TABLE (EQT)
(CORE RESIDENT)

DIREC
DIREC is a core resident table which contains information about the
disc directory. It has the following structure:

WORD
φ  length in words of first directory track
1 - 4  same as first 4 words of first directory track
5  unused
6  disc address of first directory track
7 -13 same as 0 - 6 but applied to 2nd directory track
14 -20 same as 0 - 6 but applied to 3rd directory track
21 -27 same as 0 - 6 but applied to 4th directory track

The disc address of a directory is always sector φ of a track. Each
directory track may contain as many as 5440 words = 85 sectors = 680
directory entries.

IDLOC:  disc address of IDT table
IDLEN:  negative length in words of IDT
ADLOC:  disc address of ADT table
ADLEN:  negative length in words of ADT
TRAX:  this is a table of which disc tracks are physically avail-
able to the system. Locations 140 - 143 correspond to disc
φ, 144 - 147 to disc 1 etc. Track φ of disc φ is repre-
sented by bit φ of 140, etc. A bit is φ when the track is
available, 1 when unavailable.

?TBL  There is one word in this area for each of the four discs.
When the word is zero, the particular disc does not exist.
Otherwise bits 15:8 contain the number of sectors/track, bits
7:6 the disc prefix, and bits 5:0 the high priority select
code. The prefix is used by the disc driver as the high
order 2 bits of the 8-bit track address.
MAGSC: high priority select code for mag. tape; if nonexistent, MAGSC = Ø

PHSC: select code for auto disconnect board, if nonexistent, PHSC = Ø

PHR: = 10 X number of seconds allowed for user to log on; applicable only if PHSC ≠ 0
THE EQUIPMENT TABLE (EQT)

<table>
<thead>
<tr>
<th>label</th>
<th>Octal Care</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIREC</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>first 4 words of track</td>
</tr>
<tr>
<td></td>
<td>d-addr</td>
</tr>
<tr>
<td></td>
<td>for DIRECTORY Ø</td>
</tr>
<tr>
<td></td>
<td>DIRECTORY 1</td>
</tr>
<tr>
<td></td>
<td>DIRECTORY 2</td>
</tr>
<tr>
<td></td>
<td>DIRECTORY 3</td>
</tr>
<tr>
<td>IDLOC</td>
<td>134</td>
</tr>
<tr>
<td>IDLEN</td>
<td>135</td>
</tr>
<tr>
<td>ADLOC</td>
<td>136</td>
</tr>
<tr>
<td>ADLEN</td>
<td>137</td>
</tr>
<tr>
<td>TRAX</td>
<td>140</td>
</tr>
<tr>
<td></td>
<td>Which Tracks are Locked</td>
</tr>
<tr>
<td></td>
<td>for logical disc Ø (1 =&gt; corresponding track is locked)</td>
</tr>
<tr>
<td></td>
<td>logical disc 1</td>
</tr>
<tr>
<td></td>
<td>logical disc 2</td>
</tr>
<tr>
<td></td>
<td>logical disc 3</td>
</tr>
<tr>
<td>?TBL</td>
<td>160</td>
</tr>
<tr>
<td></td>
<td>disc Ø</td>
</tr>
<tr>
<td></td>
<td>disc 1</td>
</tr>
<tr>
<td></td>
<td>disc 2</td>
</tr>
<tr>
<td></td>
<td>disc 3</td>
</tr>
<tr>
<td>MAGSC</td>
<td>164</td>
</tr>
<tr>
<td>PHSC</td>
<td>165</td>
</tr>
<tr>
<td>PHR</td>
<td>166</td>
</tr>
<tr>
<td></td>
<td>phon sc</td>
</tr>
<tr>
<td></td>
<td>mag sc</td>
</tr>
</tbody>
</table>

All lengths in negative words
"d-a" -- disc-address
"sc" -- select code

Sectors/track

10* sec allowed for logon

prefix -- high order 2 bits of track addr

high SC
TSB WORD FORMATS

**ID's**

- **14 0**: Letter followed by a binary number.
  - \( A=1, B=2, \ldots, Z=32 \_ \_ \_ \_ \)

**Disc-Addresses**

- **13 8 6 0**: Track, sector, logical disc number.

**YEAR**

- **24**: Year
  - **+ 1**

**DATIM**

- **600*min+10*sec+1/10sec-36000**: Hour of year

**DATIM+1**

- **600*min+10*sec+1/10sec-36000**: 1/10 sec of hour.
DESCRIPTIONS

Select code
(which group of 64 tracks)
SECTORS/TRACK
Track Availability

EQT

Select Code, or Ø

Mag Tape

Select Code, or Ø

Auto-Disconnect
In fig. 1 are four groups of commands and their associated error diagnostics. For each group, indicate which TSB table is accessed to determine the error conditions of that group.

**DIRECTORY**

The DIRECTORY contains information about all the files and programs in the system.

1. Where is it maintained?
2. Why?

Each entry in the DIRECTORY describes a particular file or program.

3. Consider the GET command. What information must each entry contain in order for it to work?
4. There is no confusion between identically named programs for different users. What does this imply about the argument of the search in GET?
5. Consider the PURGE command. What information must each entry contain in order for it to work?

**fig. 1**

**Group 1**

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE</td>
<td>DUPPLICATE ENTRY</td>
</tr>
<tr>
<td>GET</td>
<td>NO SUCH PROGRAM</td>
</tr>
<tr>
<td>KILL</td>
<td>NO SUCH PROGRAM</td>
</tr>
<tr>
<td>PROTECT</td>
<td>NO SUCH PROGRAM</td>
</tr>
<tr>
<td>OPEN</td>
<td>DUPPLICATE ENTRY</td>
</tr>
</tbody>
</table>

**Group 2**

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE</td>
<td>SYSTEM OVERLOAD</td>
</tr>
<tr>
<td>OPEN</td>
<td>SYSTEM OVERLOAD</td>
</tr>
</tbody>
</table>

**Group 3**

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAVE</td>
<td>FILE SPACE FULL</td>
</tr>
<tr>
<td>HELLO</td>
<td>ILLEGAL ACCESS</td>
</tr>
<tr>
<td></td>
<td>NO TIME LEFT</td>
</tr>
<tr>
<td>OPEN</td>
<td>FILE SPACE FULL</td>
</tr>
</tbody>
</table>

**Group 4**

<table>
<thead>
<tr>
<th>Command</th>
<th>Error Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>KILL</td>
<td>FILE IN USE</td>
</tr>
</tbody>
</table>
6. Somehow we must prevent GETing files. How can we do this?
7. How can we prevent LISTing protected programs, or accessing protected files?
8. For efficiency in searching, how is the DIRECTORY ordered?
9. What operator command lists (almost) all of the information in the DIRECTORY?
10. What doesn't it list?

**IDT**

Each entry in the IDT contains information associated with each user in the system.

For the commands listed below, indicate what information the IDT must contain in order for the command to work:

1. HELLO
2. SAVE
3. How should the IDT be structured?

**ADT**

Each entry in the ADT describes an available area on the discs.

1. How can such an area be defined?
2. What commands access the ADT?
I. Schedule summary
   A. Tasks compete for CPU control
   B. Ready tasks are queued
   C. Placement on queue by:
      1. Priority level - each task is queued ahead of all lower priority tasks
      2. Time - each task is queued behind all equal and higher priority tasks
   D. Priority level determined by status of task
   E. Functions of scheduler

II. Task statuses

III. Task priorities

IV. The queue

V. Macro flow chart as implementation of necessary scheduler functions

VI. Changes in task status/priority

VIII. Example
SCHEDULER FUNCTIONS

Task Scheduling = placing in queue
1. BASIC line, CR Syntax
2. User command, CR Command
3. Operator command, CR Command
4. Disconnection BYE

Task initiation = begin execution of top-of-queue

Task suspension = removing from queue, temporarily
1. I/O suspend - normal, or forced (by some SLIP's)

Task re-scheduling = placing back on queue
1. Return from I/O suspend

Task termination = removing from queue
1. Normal task end (incl. BYE, SLE)
2. User abort

Task interruption = remove from top-of-queue (but still in queue)
1. Bump due to higher priority task
2. Time out RUN jobs - place on bottom
STAT Values

-2  System disconnect (only with PHONES)
    - Not logged on in time
    - Lost carrier
    - BYE

-1  User abort request
    - Break key held $\geq 114$ msec

0    Idle

1    Aborting

2    Input wait
    - Task requests input

3    Output wait
    - Buffer is full on output
    - Library routine

4    Syntax Processing

$\geq 5$  Command - value corresponds to command's position in COMTABLE

STAT transitions

a  $\rightarrow$  -2  Disconnection
-1  $\rightarrow$  1  Abort request detected by scheduler
0  $\rightarrow$  -2, or $\geq 4$  Initiate task or disconnection
1  $\rightarrow$  0  "STOP" message complete
2,3  $\rightarrow$  previous STAT  Task returning from I/O suspend
$\geq 4$  $\rightarrow$  0  Task completion
$\geq 5$  $\rightarrow$  2,3  Into I/O suspend

"a" is anything
PLEV values

0  Syntax processing
   Return from I/O suspend
   Disc-resident routines when at top-of-queue
1  RUN (initially), LIST, PUNCH
2  Disc-resident routines before reaching top-of-queue
4  Compute-bound programs

PLEV transitions

1  $\rightarrow$ 4  RUN job timed-out
0  $\rightarrow$ 4  RUN job (that returned from I/O suspend) timed-out
2  $\rightarrow$ 0  SLIP reaching top-of-queue
     $\rightarrow$ 0  Return from I/O suspend
     $\rightarrow$ 2  (BYE)  Disconnection
<table>
<thead>
<tr>
<th>STAT</th>
<th>PLEV</th>
<th>MEANING</th>
<th>SET BY</th>
<th>WHEN</th>
<th>RESET TO</th>
<th>PLEV</th>
<th>BY</th>
<th>WHEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>OFF 0</td>
<td>DISCONNECT 3</td>
<td>PHONES</td>
<td>1. NOT LOGGED IN IN TIME, OR 2. CARRIER LOST FOR 1 SEC.</td>
<td>-2</td>
<td>2</td>
<td>SCHEDULER</td>
<td>DETECTION OF FLAG FROM PHONES INDICATING DISCONNECT</td>
</tr>
<tr>
<td>-1</td>
<td>OFF 0</td>
<td>USER ABORT REQUEST</td>
<td>MPXR ABORT TIMER</td>
<td>BREAK KEY &gt; 114 MSEC</td>
<td>1</td>
<td>OFF 0</td>
<td>SCHEDULER</td>
<td>DETECTION OF FLAG FROM USER INDICATING ABORT REQUEST</td>
</tr>
<tr>
<td>0</td>
<td>OFF 0</td>
<td>IDLE</td>
<td>SCHEDULER</td>
<td>TASK TERMINATION</td>
<td>NORMAL STATE - CAN GO TO -2, 24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>OFF 0</td>
<td>ABORTING</td>
<td>SCHEDULER</td>
<td>WHEN TYPING &quot;STOP&quot;</td>
<td>0</td>
<td>OFF 0</td>
<td>MPXR OUTPUT ROUTINE</td>
<td>COMPLETION OF OUTPUT</td>
</tr>
<tr>
<td>2</td>
<td>PREVIOUS STAT</td>
<td>INPUT WAIT</td>
<td>SCHEDULER</td>
<td>TASK REQUESTS INPUT</td>
<td>PREVIOUS STAT</td>
<td>0</td>
<td>SCHEDULER</td>
<td>DETECTION OF FLAG FROM MPXR INDICATING OR RECEIVED</td>
</tr>
<tr>
<td>3</td>
<td>PREVIOUS STAT</td>
<td>OUTPUT WAIT</td>
<td>SCHEDULER OUTPUT ROUTINE</td>
<td>OUTPUT BUFFER IS FULL</td>
<td>PREVIOUS STAT</td>
<td>0</td>
<td>SCHEDULER</td>
<td>DETECTION OF FLAG FROM MPXR INDICATING ONLY 10 CHARACTERS LEFT IN BUFFER</td>
</tr>
<tr>
<td>4</td>
<td>OFF 0</td>
<td>SYNTAX</td>
<td>SCHEDULER</td>
<td>DETECTION OF FLAG FROM MPXR INDICATING CP, &amp; A NUMERATED LINE DURING IDLE</td>
<td>0</td>
<td>OFF 0</td>
<td>SCHEDULER</td>
<td>SYNTAX COMPLETION</td>
</tr>
<tr>
<td>5</td>
<td>OFF 0</td>
<td>LIST PUNCH</td>
<td>SCHEDULER</td>
<td>1. DETECTION OF FLAG FROM MPXR INDICATING OR, &amp; UNNUMBERED LINE DURING IDLE, OR 2. DETECTION OF FLAG FROM MPXR INDICATING RETURN FROM I/O SUSPEND</td>
<td>0</td>
<td>OFF 0</td>
<td>SCHEDULER</td>
<td>TASK COMPLETION</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>SCHEDULER</td>
<td>RUN 4 LIST PUNCH</td>
<td>0</td>
<td>OFF 0</td>
<td>SCHEDULER</td>
<td>I/O SUSPEND</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
<td>CORRESPONDING COMMAND</td>
<td>0</td>
<td>OFF 0</td>
<td>SCHEDULER</td>
<td>I/O SUSPEND</td>
</tr>
</tbody>
</table>

**NOTES:**
1.-flag is bit set in MPXON.
2. MPXR accepts no input (incl. BREAK) until acknowledging LF is output by task.
3. ANY STATUS TASK CAN ENTER DISCONNECT.
4. RUN JGOS' PLEV CHANGE TO 4 UPON TIME RUN-OUT. STAT REMAINS SAME.
5. ALTHOUGH STAT REMAINS -2, SCHEDULER TREATS DISCONNECT AS 'BYE' COMMAND.
EXPLANATION OF THE QUEUE SEQUENCE EXAMPLE

The example is one of a TSB system with 7 ports (2,3,6,7,8,9,10) logged on. Each frame depicts the status (STAT) and priority (PLEV) of each user as well as the order of the queue at that time. The queue is the top group of entries with the top-of-queue on top. Each port entry shows the port number in the left box, that user’s status in the middle box, and his PLEV in the left box. The time between frames is strictly relative.

1 - 8 is doing syntax.
2 - 10 requests syntax processing. Note: does not receive CPU control immediately due to 8 not being finished.
3 - 8 finishes. 10 receives control.
4 - 9 requests RUN.
5 - 10 finishes.
6 - 9 gets interrupted by (higher-priority) syntax job of 8.
7 - 7 types a line of syntax (bumping 9 further).
8 - 8 finishes. 7 get control.
9 - 7 finishes. 9 resumes where it was interrupted.
10- 10 requests LIST. 6 requests KILL.

11- 9’s RUN job gets timed-out. i.e. total time that it was on top-of-queue (frames 5,9,10) was 1 second. 9 gets reassigned a PLEV of 4 and placed on bottom of queue. 10 receives control.
12- 3 requests syntax, interrupting 10 and seizing control.
13- 2 requests syntax, and gets queued under 3.
14- 3 finishes. 2 receives control
15- 2 finishes. 10 resumes the LIST program. This fills the buffer and 10 is placed in output suspend (off the queue).
16- Notice that 10’s STAT upon output suspend (i.e. LIST) is saved in PLEV (which has no relevance when a user is off the queue). 6’s KILL request has acquired control, and its PLEV is changed to 0 (so that it may run to completion or self-suspension).

17- 7 requests syntax, but gets queued under 6.
18- 6 finishes.
19- 10 returns from output wait (10 characters left in buffer) and gets queued with 0 PLEV.

20- 9's user has pressed the break-key for more than 114 msec. The multiplexor changes his STAT and signals the scheduler. In the meantime, the Phones logic has detected a lost carrier on 6, changes STAT and also signals scheduler.

21- The scheduler detects 9's -1 PLEV, dequeues him, and changes his STAT to 1 while "STOP" is being typed. The scheduler also detects 6's -2 PLEV and queues him as if he requested BYE (the only difference is that his STAT remains -2).

22- 10 goes into output wait again.

23- The MPXR output routine detects the last character has been output for port 9 and STAT was 1 (i.e. the "STOP" abort message is complete) and resets STAT to the idle state. Also, 8's RUN job has encountered an INPUT statement and goes to input wait (STAT (=5) is saved). 6 receives control and the BYE routine is run with 0 PLEV.

24- 6 finishes and 8's user has responded.

25- 2 requests syntax and 10 returns from I/O suspend; but notice that both get queued under 8's RUN job due to 8's high priority upon returning from I/O suspend.

26- But 8 gets timed-out again (frames 24 +25 > 1 second) and becomes a miserable low-STAT task again.

And so on ...
<table>
<thead>
<tr>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>syn</td>
<td>0</td>
<td>2</td>
<td>syn</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>syn</td>
<td>0</td>
<td>10</td>
<td>LIS</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>LIS</td>
<td>1</td>
<td>6</td>
<td>KIL</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>KIL</td>
<td>2</td>
<td>9</td>
<td>RUN</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>RUN</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>7</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>syn</td>
<td>0</td>
<td>10</td>
<td>LIS</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>LIS</td>
<td>0</td>
<td>10</td>
<td>LIS</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>RUN</td>
<td>4</td>
<td>8</td>
<td>RUN</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>6</td>
<td>-2</td>
<td>6</td>
<td>-2</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>SYN</td>
<td>LIS</td>
<td>RUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>25</th>
<th>SYN</th>
<th>LIS</th>
<th>RUN</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Consider...

Suppose someone dials up TSB. The phones' logic begins timing him. He types in his HELLO command, but during the start bit of the carriage return, the Phones' logic times him out. What Happens?
What? No TBG?

Assume a TSB system has a malfunctioning time-based generator, in that it does not give any interrupts. Analyze what effect this would have on the system if the bad board doesn't break the priority string; and if it does. What symptoms would you expect?

On one of the four systems, replace the TBG board with a jumper board, and prove your theories. Try different types of RUN programs, any commands you suspect would operate improperly, and (if possible) on the system with auto-disconnect (lab B), try dialing up and check the effects.
The system currently provides 1 second time-slices for compute-bound RUN programs. It may be advantageous for CAI to increase this to 2 seconds. How can such a mod be made (i.e., generate the mod)?

You may wish to try it on one of the systems.
PHONES' LOGIC:
The Phones logic constantly monitors the input from the disconnect board to detect any "ringing" indications, i.e. status changes from 1 to 0. If the software detects a "ringing" signal from a data set, it responds by setting the appropriate bit in PHO which when outputted to the disconnect board makes the data terminal ready and answers the data set.

Once a "ringing" has been detected, the required response time plus the present time is put into the user's TTY table ?PHON Entry and the appropriate bit in PHT corresponding to the user's TTY port number is set to indicate that this particular user is being timed. Each time the phones' logic is invoked by the scheduler, and the bit in PHT is still set, the current time (DATIM) is checked to see if it equals the user's PHON entry. If it doesn't equal, nothing is done. If it does equal, the user has run out of time and a hangup must be initiated.

If the user successfully logs on in the required time interval, the timing mechanism (PHT) in the phones logic is cleared in the HELLO Library routine and the user is allowed on the system.

The software continues to sample the status line; if a 0 is still present after the data set has been answered, the data set is transmitting data to the computer and nothing is done.

These are the ways in which the user may be disconnected:

a) carrier is lost
b) user signs off
c) user is timed out

a) When carrier is lost, the phones logic detects this because the input from the disconnect board changes from 0 to 1. When the software detects this transition, it knows it's a line dropout and initiates a 1 sec time interval via the user's PHON TTY table Entry, sets the user's PHT bit and allows the time-out logic of phones' timing to handle the dropout disconnect.

b) When the user signs off (BYE), the BYE routine sets the user's bit in PHT and sets his ?PHON entry to a 4 second time interval and allows the time-out logic of phones' timing to handle the sign off disconnect. The reason for the 4 second interval is so as to allow the standard log off message to be outputted the user's TTY.

c) The user is timed out when the time in his ?PHON TTY table entry is equal to or less than the time contained in DATIM. When this occurs, the user's status is set to a -2 (system disconnect) and the appropriate TTY bit is set in MPCOM to inform the scheduler of the user's status. When the scheduler detects the system disconnect status for that user, it schedules the user for the BYE Library routine.

When the BYE routine is called in to handle this user, the user may be in two states.
1) he has already typed BYE (signed off), 2) he has not signed off. If he has already signed off, his I.D. entry in the TTY table had been cleared from his previous BYE processing. So that, when BYE is entered again for him, the I.D. contained in the system TTY table entry is zero which signifies the user received his log off message and it is O.K. now to disconnect him from the system. The disconnect is accomplished by clearing his timing bit (PHT) and setting his PHO bit to one which is the hang up signal to the disconnect board.

If the user had not signed off and BYE was scheduled by the Scheduler it was because the user had timed out either for 1 second (line drop out) normal response time for a log on had elapsed. Therefore, the user was either in core or not in core. If he wasn't in core he is disconnected then. If he was in core (i.e., the case of a line drop out) 4 seconds must elapse before he is disconnected so as to allow the standard output message to be completed.
Each bit corresponds to \(2^N\) TTY number.

**PHL**

Set at load time to TTY status of disconnect board. For purposes of simplification, assume nothing was happening with the auto disconnect board at sleep time. Therefore, when the system was awakened, the TTY's are in their quiescent state, i.e. PHL is set to all ones. PHL is updated, such that, it will reflect the most recent input from disconnect board.

Each bit corresponds to \(2^N\) TTY number. Each time the phones' logic is invoked by the schedule, PHN is updated so as to reflect the current input from the disconnect board.

Each bit corresponds to \(2^N\) TTY number. When the phones' logic finishes with all the TTY's that need update in relation to the disconnect board, it outputs the word PHO. This word is the output communication to inform the board of any status change. The quiescent state for PHO is all 1's.

**PHONES'BOARD LEVELS**

Input = Ø implies carrier or ring
Input = 1 implies neither
Output = Ø answer the phone
Output = 1 hang phone up

Each bit corresponds to \(2^N\) TTY number set to the user's port which is currently being timed. The user can be timed for three reasons:

1. Normal response timing to log on system
2. Timing because carrier was lost
3. Second time out due to time required to output standard sign off message to user's TTY
SELECTED LIBRARY ROUTINES
HELLO

1. LOGS OFF OLD USER IF HE HASN'T DONE SO.
   a) REINITIALIZE NEW USER'S PORT PUSS TABLE ENTRY TO ZERO
   b) OUTPUT LOG OFF MESSAGE TO SYSTEM TTY
   c) SETUP PHONES TIMING FOR NEW USER

2. READ IN IDT TABLE AND CHECKS FOR VALID NEW ID.

3. CHECK FOR VALID PASSWORD IF ID CORRECT.

4. CHECK THAT THE TIME USED TO DATE IS LESS THAN THE ALLOWED.

5. USER HAS SUCCESSFULLY LOGGED ON, THEREFORE DISCONTINUE ANY
   PHONES TIMING BY SETTING THE BIT IN PHT CORRESPONDING TO THE
   NEW USER'S PORT NUMBER.

6. RUN ANY "HELLO" PROGRAM WHICH IS IN PUBLIC LIBRARY, OTHERWISE
   PRINT "READY" MESSAGE TO ALLOW USER ACCESS TO SYSTEM.

7. IF "HELLO" PROGRAM PRESENT, TRANSFER CONTROL TO BASIC.
HELLO EXERCISE:

ATTACHED TO THIS EXERCISE IS A LISTING OF THE PUBLIC LIBRARY ROUTINE $HELLO WHICH IS RUN EACH TIME A USER SUCCESSFULLY LOGS ON. ACCORDING TO THE HELLO FUNCTION OF THE TSB SYSTEM, THE $HELLO PROGRAM IS WRITTEN INTO THE USER'S SWAP AREA AND EXECUTED, HOWEVER THE PROGRAM NEVER RESIDES IN THE USER'S SWAP AREA AFTER IT IS EXECUTED, WHY?
STOP

GET-$HELLO

LIST
HELLO

8 PRINT
10 PRINT " HP IN-HOUSE SERVICE BUREAU"
15 PRINT " WELCOMES YOU TO"
20 PRINT " SYSTEM/1"
30 PRINT
31 PRINT " PREVENTIVE MAINTENANCE"
40 PRINT " DAILY FROM 0800 TO 0815 PDT"
50 PRINT
55 PRINT "** DENOTES SPECIAL MESSAGES"
56 PRINT
65 PRINT " Y D A ER"
70 END
HELLO-BYE EXERCISE:

An interesting occurrence happens when a user logs on and types echo-off and then logs on again without typing BYE. For this exercise, go up to a terminal and try it. What happened, why, and how do you fix it?
GET

1. TRANSLATE NAME OF PROGRAM FROM USER'S INPUT. IF PRECEDED
   BY $ SET UP FOR A000 SEARCH; OTHERWISE SET FOR SEARCHING ON
   USER'S ID.

2. PERFORM DIRECTORY SEARCH. PRINT ERROR IF NOT FOUND.

3. ERROR IF ENTRY FOUND IS FILE (BIT 15 OF WORD 2 OF ENTRY IS 1).

4. CHECK THAT THE PROGRAM WILL FIT INTO THE USER AREA. THIS IS
   NECESSARY IN CASE A PROGRAM WHICH WAS SAVED UNDER AN OLD VERSION
   OF THE SYSTEM CAN NO LONGER FIT WITH THE CURRENT VERSION.

5. UPDATE THE DATE OF THE USER'S PROGRAM (PURGE).

6. SET RUN-ONLY BIT IF RUN-ONLY PROGRAM UNLESS USER IS A000.

7. READ IN PROGRAM AND UP DATE CERTAIN FLAGS FOR BASIC, THEN EXIT.
GET

STORE USER'S ID IN LTEMP

STORE USER'S PROGRAM NAME IN LTEMP (1:3)

IS REQUEST FOR PUBLIC LIB ROUT.
   YES
      SET USER'S ID TO A000 (JUST FOR SEARCH)
   NO

USING DLOOK-IS REQUEST PROGRAM THERE?
   NO
      PRINT ERROR MESSAGE LIBER
   YES

IS PROGRAM A FILE?
   YES
      PRINT ERROR MESSAGE LIBER
   NO

WILL PROGRAM FIT INTO USER'S SWAP AREA?
   NO
      PRINT ERROR MESSAGE LIBER

GET 13

UPDATE DATE OF PROGRAM IN DIRECTORY

IS USER REAL USER A000?
   NO
   CLEAR RUN ONLY BIT
   YES

READ IN PROGRAM

DO HOUSEKEEPING FOR BASIC

EXIT
KILL

1. TRANSLATE THE PROGRAM OR FILE NAME AND PERFORM A DIRECTORY SEARCH. FAIL TO ERROR ROUTINE IF ILLEGAL NAME OR THE SEARCH FAILS.

2. IF THE ENTRY TO BE FILLED IS A FILE, SEARCH THE FUSS TABLE TO SEE IF ANY OTHER USER HAS ACCESS TO THE FILE. IF SO, PRINT A MESSAGE AND TERMINATE. IF NOT, CLEAR THE USER'S SECTION OF FUSS.

3. DELETE THE ENTRY FROM THE DIRECTORY AND ADJUST DIREC.

4. SUBTRACT THE PROGRAM LENGTH FROM THE USER'S IDT ENTRY (SYSTEM ADMINISTRATION).

5. UPDATE ADT TO REFLECT ADDED DISC SPACE.

6. DECOMPILE USER IF FILE WAS KILLED. THIS GUARANTEES THAT ANY OLD REFERENCES TO FILE WILL DISAPPEAR.
**ECHO:**

The ECHO command is used to control the computer echo of teletype input. Echoing is determined by the user's bit in the word PLEX. Bit = 0 implies no echo, 1 implies echo. The user will want echoing if and only if his teletype is full duplex. The command format is:

- ECHO-ON for full duplex
- ECHO-OFF for half duplex
OPEN

1. TRANSLATE AND CHECK THE FILE NAME AND LENGTH.

2. CHECK THE IDT AND ADT TO SEE IF
   a) THE USER HAS ENOUGH DISC ALLOCATED TO HIM TO SATISFY THE COMMAND. (ADMINISTRATIVE PURPOSES)
   b) THERE IS AN AREA ON THE DISC WHICH IS LARGE ENOUGH TO ACCOMMODATE THE FILE. IF THERE IS AN AREA WAIT UNTIL ROOM IN DIRECTORY IS FOUND.

3. PERFORM A DIRECTORY SEARCH ON THE FILE NAME. IF FOUND, THIS IS A DUPLICATE ENTRY, SO TERMINATE.

4. IF DIRECTORY TRACK IS NOT FULL, INSERT NEW ENTRY.

5. IF DIRECTORY TRACK IS FULL, CALL IN SUPERSAVE TO RESTRUCTURE THE DIRECTORY AND INSERT THE ENTRY.

6. UPDATE THE IDT AND ADT APPROPRIATELY.

7. INITIALIZE THE FILE SO THAT A -1 (END-OF-FILE) IS AT THE BEGINNING OF EVERY SECTOR. WRITE THE FILE TO THE DISC AND THEN TERMINATE.
BYE:

1. SET THE USER'S PLEX BIT TO FULL DUPLEX.

2. IF USER ID = 0, THEN DISCONNECT PHONES TIMING AND TERMINATE BYE.

3. IF USER IN CORE, READ IN FUSS TABLE AND CLEAR USER'S ENTRY IN TABLE.

4. READ IN IDT TABLE AND COMPUTE HIS TIME USED TO UPDATE HIS IDT TABLE ENTRY.

5. CREATE A LOG-OFF MESSAGE FOR USER AND PUT IN SYSTEM TELETYPET I/O QUEUE.

6. REMOVE USER'S ID FROM TTY TABLE.

7. CALCULATE TIME USED AND OUTPUT TO USER'S TTY.

8. ENABLE AUTO DISCONNECT VIA PHT TO DISCONNECT USER FROM SYSTEM.
BYE EXERCISE

YOU HAVE IMPLEMENTED A MOD TO MAKE A PORT OR A PARTICULAR SEQUENCE OF PORTS HALF-DUPLEX IN THE MULTIPLEXOR DRIVER. CAN A SIMILAR MOD BE IMPLEMENTED IN THE BYE ROUTINE? STATE REASON(S) WHY IT CAN OR CANNOT.
1. Test if program has name. Fail if none.

2. Test if there is a program to save. Fail if none.

3. If the user's program is in compiled form (CFLAG bit=1), call DCMPL to put it into the form in which we will save it.

4. Read in IDT to see if the user has sufficient disc space allocated to save the program. Fail if not enough room.

5. Read in ADT to check to find first entry large enough to hold the program. Fail if one is not found.

6. Perform a directory search on the program to be saved. Fail if such an entry name already exists.

7. If the directory track is full, the SUPERSAVE library routine is called to attempt to reallocate the directory. If it's successful in its reallocation algorithm, the new directory entry is made. If SUPERSAVE fails in its attempt, a system overload message will appear on the user's TTY.

8. Insert a new directory entry into the directory if there is no need for SUPERSAVE.

9. Update the IDT to reflect the new amount of disc used for that user.

10. Update the ADT to reflect new length because of disc allocation for user's program.

11. Copy the user's program to its library area.
SAVE

does Program have a name?

YES

IS there a program to save?

YES

IS program compiled?

YES

DECOMPIL e & WRITE OUT TO DISC

NO

READ IN IDT & ADT TABLE

IS USER OUT OF ALL DISC SPACE?

YES

"FILE SPACE FULL" ERROR MESSAGE

LIBER

NO

"No Name ERROR" MESSAGE

LIBER

IS there an entry in ADT TABLE LARGE ENOUGH TO SAVE PROGRAM?

YES

"SYSTEM OVERLOAD MESSAGE"

LIBER

NO

IS SIMILAR ENTRY IN DIRECTORY?

YES

"DUPLICATE" ENTRY MESSAGE

LIBER

DLOOK

NO

SUPER SAVE

OK

INSERT IN DIRECTORY & UPDATE DIREC

OK

ERROR MESSAGE

UPDATE ADT & IDT

COPY USER PROGRAM TO ITS SAVED LOCATION

EXIT
SUPERSAVE:

SUPERSAVE (SAVE OVERLAY ROUTINE) IS CALLED BY THE SAVE AND OPEN ROUTINES WHENEVER THEY WANT TO MAKE A DIRECTORY ENTRY ON A TRACK THAT IS ALREADY FULL.

SUPERSAVE ATTEMPTS TO REDISTRIBUTE THE DIRECTORY TRACKS SO THAT THEY WILL BE AS EQUAL IN LENGTH AS POSSIBLE. THIS WILL GENERALLY PREVENT IT FROM BEING CALLED VERY FREQUENTLY. THE OPERATION IS AS FOLLOWS:

1. SCAN THROUGH DIREC AND DETERMINE THE TOTAL LENGTH OF ALL DIRECTORY TRACKS, AND ADD 8 FOR THE NEW ENTRY. IF ALL DIRECTORY TRACKS ARE FULL, EXIT THROUGH FAILURE LOCATION.

2. DIVIDE TOTAL DIRECTORY LENGTH BY NUMBER OF AVAILABLE DISC TRACKS TO DETERMINE THEIR NEW INDIVIDUAL LENGTHS.

3. NOW REDISTRIBUTE THE DIRECTORY TRACKS. THE BASIC IDEA OF THE ALGORITHM IS TO FILL THE SWAP AREA WITH AS MUCH OF THE DIRECTORY INFORMATION AS WE CAN, READING FROM THE BEGINNING AND THEN TO WRITE OUT AS MUCH AS WE CAN, ALWAYS MAKING SURE THAT WHEN WRITING, WE DON'T OVERLAY ANY PORTION THAT HASN'T BEEN READ YET.
SLEEPING AND LOADING
CATALOG

[LIB=CAT with A000 user]

1. Initialize user, 000000

2. Find entry ≈ sought after

3. DLOOK

4. End of directory track
   - yes: read in next non 0 track
   - no: Same user

5. Same user
   - yes: print name and length
   - no: DONE

6. Full tty line

7. Bump

8. Save buffer address of name in BHED

9. By Scheduler; rescheduled when 10 char left.
   using ptr saved in BHED

10. Read last name printed

11. Set user from tty table
Bump to next entry

Save i.d. in tty35 buffer (at T35BF+35)

Print Heading

Suspend

initialize 0000,000000

Suspend

D LOOK

Setup

Sets D LOOK search for id, name just printed

end of directory track

read in next non 0 track

i.d. neg.

i.d. diff. than last

convert & put in buf

DONE

print line

place blanks in id position

convert rest & put in buf

yes

no

yes

no
SLEEP

disallow further queuing

everybody in abort & stop all output

output SLE message to all

when IOTOG=-1

done?

yes

no

disconnect all

read in IDT

scan tty tables

done

done

for each active user

update IDT time

setup log-off mes.

.... messages will be printed as scheduler is invoked by TBG

.... disallows input

.... clear MPCOM set T35LK to MLINK+1

write IDT out

set FUSS to Ø

read in base swap area write out to sector Ø

tty35 done?

no

checking T35FI

read in overlay

LIBRA
The diagram outlines a process involving ADT entries in a system, directory, IDT, ADT, and SWAP. The steps include:

1. Read in ADT
2. Check if locked. If yes, go to TLTAB (T) and bump track, then return to locked?
3. If locked is no, check if system track. If yes, go to track len avail.?
4. If system track is no and track len avail.?
   a. Write ADT
   b. Scan directory for prog. on this track
      i. If no more directory modified, write out
      ii. If more dir. track, return to track len avail.?
      iii. If yes, write directory new addr.
9. Update directory new addr.

**Variables:**
- SLET = track counter
- SLES = next available track addr. (1st pass)
- SLER = next available track addr. (2nd pass)
- SLEP = 1st pass sub-directory pointer
- SLEQ = 2nd pass sub-directory pointer
SLEEP OVERLAY - track packing
not SYSTEM, locked, or fully available track

S ← R ← T
P ← Q ← STAB

I ← DIRDØ

read direct 1
D ← D end
F ← Ø

BUMP 1

Decr D

file/prog not ≠ on T

D ← LIBD

trk(D):T

(P) ← d-addr(D)

save old d-addr in sub-directo

(no)

write directory I

d-addr. (D) ← S

update directory entr w/new d-add

F ← F+1

flag as directory

another directory track ≠

I:DIRD3

(P) ← S
P ← P+1

calc. new d-addr for next prog; save in sub direc.

A

go form dummy in-core track image.
SLEEP OVERLAY - Track packing (Cont'd.)

detailed view

...........P phts to past last sub-directory
Q phts to 1st sub-directory entry

P : Q

\[ R \leftarrow (Q+1) \]
\[ Q \leftarrow Q+2 \]

read in f/p of Q into approp.spo in core

old d-addr = (Q)
len = \([Q+1]-R\) *64 words
new d-addr = R

...........update new d-addr pntr, R
update sub-direct pntr, Q

write dummy core image to

update trk len table

read ADT

replace all refs to T w/1 ref to T

old d-addr = (Q)
len = \([Q+1]-R\) *64 words
new d-addr = R

...........update new d-addr pntr, R
update sub-direct pntr, Q

return

go process next track
SLEEP TAPE FORMAT

neg. numbers (Ø7 system)

Track 225 image
directory 1 image
[directory 2 image]

defines number of non-Ø directories

EOF System Segment Table TSB

Segments Lib Len Table (Disc Res 1) Dbisc Res.2

Version E: -5
2-13B
2008-2000B
4001B-4030B
14000B-37300B
TSB BBDL LOADING

S.A. = 37760B
HLT 77B
Protect loader, and RUN, w/switch 0 down

a) Obtained from disc:
   i) System
   ii) I.D. information
   iii) User library
   iv) Hardware configuration

b) Requires a SLEpt System on disc

TSB LOADER OPTIONS

1. Generation - S.A. = 2000B; NO LIBRARY
   Creates a system from scratch; no library, i.d's, etc.
   a) Configurations different than:
      - One 64 track disc/drum
      - NO MAG tape
      - NO PHONES
      must be indicated

2. Update - S.A. = 2000B; LIBRARY; NO MAG TAPE
   Updates the software (new versions, patches), preserving
   the user library
   a) Loads system from paper tape
   b) Preserves
      i) DIRECTORY and user programs/files
      ii) I.D information
      iii) Latest TSB hardware configuration
   c) Requires a SLEpt system on disc
   d) Cannot LOCK any system track. (system, swap, IDT/ADT, or DIRECTORY)
3. **Mag Tape Awake** - S.A. = 2000B; LIBRARY; MAG TAPE
Restores a system from a SLEEP tape
a) System, ID information, and user library from mag tape
b) Preserves SLEEP tape's hardware configuration
c) Requires a SLEEP system on mag tape. (the contents of the core or disc are irrelevant)
d) Can LOCK any hardware disc tracks (except 0)

4. **Emergency Resuscitation** - S.A. = 3000B
Attempts to restore a system after a failure
a) Loads system from paper tape
b) Attempts to preserve
   i) DIRECTORY and user programs/files
   ii) ID information
   iii) Hardware configuration
The success or failure of Emergency Resuscitation depends on whether the EQT and the tables are organized correctly and consistently. A knowledge of the cause and place of failure can determine this, in most cases.
BBDL
Reads track $\emptyset$, sector $\emptyset$ (disc boot) into core $\emptyset \rightarrow 77B$
inserts 'JMP 77B' in 77B, and jump to 77B.
77B overlayed with last word of $\emptyset$, $\emptyset$; which is a JMP
to status section of disc boot.

DISC BOOT
Status of disc transfer okay: HLT 77B [Operator now
protects BBDL and specifies system (SW0=0: TSB/SW0=1: DOS)]
DOS: Boot reads $\emptyset, 2$ into "DMS" and transfers to "DMS"
TSB: Boot reads $\emptyset, 1$ into "RT/TS" and transfers to "RT/TS"
("DMS" and/or "RT/TS" must have been correctly set)

TSB
("RT/TS")
SYSTEM
Reads: $\emptyset, 3$ into (OB to 12000B) TSB base & Loader
LOADER
 syst 2 into (14000B to 26500B) Core resident
 syst 3 into (26500B to 37300B) Core resident 2
(sys 2 and sys 3 must be set)
Transfers control to date/time section of loader.
SA = 2000B

 Incorrect

 Library

 "YES"

 MAG SC

 "SC"

 Read EQT
 FLG = "Mag"

 Remember # of dir. trks.

 "NO"

 Read EQT
 FLG = "SYSGEN"

 Clear EQT
 DIREC = null dir.

 SECT/TRK DO

 SECT/TRK

 Correct

 Set ?TBL & TRAX
 for just one disc

 Process any
 DISC modifications

 Incorrect

 Incorrect

 Scan in-core
 ADT Replace 0's w/trk len.

 Lock (any empty
 tracks) and
 UNLOCK

 "LOAD"

 Appropriate 20
 full tracks
 for system

 "YES"

 DOS

 Present

 "NO"

 Configure boot
 TSB sys loader → 0, 2

 Readd disc
 boot

 "READ I"
"RT/TS"(Boot) → TSB entry point

TSB sys loader → 0, 1 Boot → 0, 0

paper tape reload yes

I trk for IDT/ADT tracks for direc (previous) no

Mag yes

IDT+ADT > 5440 no

yes

Delete enuf non-Ø ADT entries

no

Mag

yes

Readt/write IDT

read TRLTB

Readt/write 11b images Store according to ADT. Remember d- addrs in TRLTB

Readt dir. images Update (via TRLTB Write

"TSB"

DATE/TIME Configure Phones

Write 3 system segments done for all users

Initialize: Swap tracks &TTY tables

Read (paper) system

Readt/write Utility progs

Readt system

no
BASIC FILE MANIPULATION ROUTINES
BASIC FILE RECORD MANIPULATION (RQSTR)

- compute record requested
- exists?
  - no: FDAT4
  - yes: Reset File Pointer
- Rec'd in already
  - yes: Write Request?
    - yes: exit
    - no: Write out Current
  - no: Read in Requested
- Write Request?
  - yes: exit
  - no: Store EOR