May 14, 1975

To the Graduate Council:

I am submitting herewith a thesis written by Teddy E. Sherrod entitled "A Computer Assisted Instruction Course to Present the H.A.S.P. System Operator Commands." I recommand that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Robert J. Plemmons, Major Professor

We have read this thesis
and recommend its acceptance:

John R. Ray

Accepted for the Council:

Vice Chancellor
Graduate Studies and Research

A COMPUTER ASSISTED INSTRUCTION COURSE TO PRESENT THE H.A.S.P. SYSTEM OPERATOR COMMANDS

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee

Teddy E. Sherrod

June 1975

ABSTRACT

The purpose of this study is to develop a Computer Assisted Instruction (CAI) course which presents the Houston Automatic Scheduling Priority (HASP) system operator control commands to potential computer operators. The goal of the course is to reduce the amount of on-the-job training required to bring a new computer operator to a satisfactory level of performance.

An argument might be advanced that the student operator could reach the same level of performance by simply reading the "HASP Operator's Guide." This is partly true but there are two strong points in favor of the Computer Assisted Instruction approach. First, by presenting the material via a computer terminal the student operator has the opportunity to experience using a device much like the computer operator's console. In addition, Computer Assisted Instruction allows the author of the course to guide the student through the material as each student progresses at his own rate. This form of instruction is interactive in that each section of course material that is presented to the student is followed by a question related to the material. The terminal then pauses, allowing the student to enter a response to the question. The response format is the same as that accepted by the computer to execute the

1207514

desired operator function. The question and answer section cf Computer Assisted Instruction allows an author to code statements which accept invalid answers to the questions. The author may then present hints and allow the student to attempt the command again. Computer Assisted Instruction allows the student operator to use this new knowledge without the need to experiment with operator commands in an online production environment.

Secondly, a student operator trained by reading a manual or by on-the-job instruction has no assurance that the knowledge he has gained is complete or that the material he has covered is sufficent. Computer Assisted Instruction eliminates the major portion of these problems by presenting a fixed amount of material to each student. The student's comprehension of the material is evaluated by an exam. Low test scores indicate areas that should be reviewed by the student before operating the computer.

This course is not limited to training new computer operators. Experienced operators have found that the course is a refresher to point out areas needing review. New operators seemed to have less trouble learning to operate the system after completing the Computer Assisted course than those operators who have not taken the course.

TABLE OF CONTENTS

CHAPTER	PAGE
INTRODUCTION	1
I. BACKGROUND	4
II. COURSE DESCRIPTION	11
MATERIAL COVERED	11
HASP OVERVIEW	12
HASP COMMAND FORM	14
MATERIAL FORMAT	15
MACRO GENERATION	17
III. COURSE FLOW	19
IV. CONCLUDING REMARKS	23
BIBLIOGRAPHY	25
APPENDIX	28
VITA	75

INTRODUCTION

The complexity of computer machinery and the need to keep operating costs at a minimum are two problems the computer industry faces today. Computer operators must be highly trained individuals familiar with the many different components which make up a computer system. This thesis addresses itself to one of those problem areas and that is the training of a computer operator in the use of the HASP operator commands. An extensive literature search reveals no completed works in this area of computer operator training.

A problem exists in the training of new computer operators. The new computer operator needs to learn the command language of the HASP system by being allowed to operate the computer; yet, while production work is being run, an inexperienced operator should not be allowed to experiment with certain operator commands. This thesis is the development of a Computer Assisted Instruction course which presents information about the HASP commands followed by questions covering the material presented. The material is presented at a computer terminal which is similar to the computer's console. This approach leaves little doubt as to the amount of material to which the new operator has been exposed.

The course structure consist of presenting material about a HASP command followed by questions covering the material just presented. The new operator enters a statement in response to the question. These statements are checked for syntax correctness in much the same manner as the HASP system decodes operator commands. Obvious errors in the statements are detected and additional information concerning the command format is given. This allows the new operator to reenter the command until it is the correct answer for the question.

This procedure also allows new operators to learn the HASP system commands without long periods of on-the-job training and without unnecessary experimentation with a production machine. The time required to train a novice operator is reduced since the trainee covers many different situations which he would probably not encounter in several days of operating a machine.

The thesis is divided into four chapters. Chapter I covers the history of CAI by following its early development in the 1920's to the many areas where CAI is used today. Chapter II looks at the operators HASP command language course developed in this thesis and covers a description of the features of the IBM COURSEWRITER III (2) language used to implement the course. A review of the basic building blocks for the course are covered in this

chapter. Chapter III addresses the use of the course and parameters used to control various options. The last chapter presents some ideas this author feels should be investigated to create additional operator training courses.

CHAPTER I

BACKGROUND

The evaluation of changing the format of instruction from the teacher-student relationship to computer-assisted instruction has not yet been completed. However, great progress is being made in reaching the goals of an efficient computer-assisted system. This chapter discusses and reviews the steps taken by various persons in order to reach this goal. Computer-assisted instruction is still in its infancy, so many findings have not been published, but an in depth study of the available literature on CAI has produced the information in this chapter. Some of the early contributions to CAI are reviewed in the first half of this chapter, while the second half reviews various CAI languages and present applications.

The closest ancestor of CAI before its inception was the textbook, which was invented and first published in 1596. The American Colonies published their first book of this kind in 1641, just three years after the establishment of the first American printing press. (3)

With the textbook, mass education was made possible because the teacher no longer had to be in individual contact with each student. However, where before the

teacher could adapt herself to the individual, the textbook could not. Because of this, there had to be a diverse and wide range of textbooks printed, all on the same subject. This presented a large economical problem to the school. Mass media of all kinds suffers from this same problem; it does not communicate on an individualized basis. (4)

Sidney L. Pressey, a psychologist at Ohio State University, was one of the first to attempt to overcome this situation and bring instruction to each individual level. Pressey invented several "teaching machines" in the 1920's which were comprised of a roll of paper and four keys. Multiple choice questions were printed on the roll of paper and the student answered the questions by punching one of the keys. If his answer was correct, the paper was turned to the next question. If he was wrong, the machine recorded the error and the student tried to answer correctly. Little notice of Pressey's invention was received from educators, and he was discouraged by the lack of recognition. In 1932 Pressey wrote that he was "regretfully dropping further work on these problems" and that he hoped that "enough has been done to stimulate other workers." (5)

However, other workers did not continue Pressey's work until the late 1940's when B. F. Skinner of Harvard University became interested and developed a machine that

would reinforce correct responses quickly in order to give the student a sense of accomplishment. Each of Skinner's questions was a small advance beyond the previous one and because the steps were small, the student's answers were usually correct, thereby reinforcing his sense of achievement. (6)

Darman A. Crowder developed a different approach to machine teaching a few years after Skinner's experimentation. He called it "intrinsic programming." The main difference between his machine and Skinner's was that Crowder's machine allowed large steps to be taken by the student. If the student was correct, he moved on to the next plateau, but if he was incorrect, he received remedial instruction on the material. This type of instruction is valuable in certain subject areas, such as history, but in a subject where the steps are important to the understanding of the student, such as math, the limitations are obvious. (9)

Programmed instruction (PI) became more popular in the early 1950's and the work of Skinner and Crowder was developed further. PI texts were written and properly tested and were found to be valuable in individualized instruction. "Among other benefits PI has demonstrated the ability to provide standardized instruction, decentralized training, and individualized pacing." (9) However,

although PI has proved valuable in instructive teaching, it has not been demonstrated to be as effective as human teachers.

Soon after the studies of the early 1950's, people became aware of the vast capacity of computers in programmed instruction. It was hoped that computers could overcome the basic problems and limitations of the teaching machines previously used. In 1958, the first experimentation combining the assets of computer technology with the limitations of programmed instruction was begun. At IBM's Thomas J. Watson Research Center, Yorktown Heights, New York, Gustave Rath and others developed a method whereby an IBM 650 Computer taught students binary arithmetic. About the same time, the System Development Corporation began a research effort using a computer to control projection devices. J. C. R. Licklider and John Surets at Balt, Beranek and Newman developed a third project in which a computer answered requests by students in an analytic geometry course.

Computer-assisted instruction developed quickly in the 1960's, advancing from about five major projects in 1961 to double that amount in 1963 and the curriculum packages doubled again by 1965. (9)

There are three modes or forms into which CAI can normally be classified. These are: drill, tutorial, or

dialogue. (7) The drill mode stresses repeated exercises and is the most basic type of instruction, because the material can be adjusted to the learner's competency level. The Palo Alto, California schools use this method quite successfully in the instruction of mathematics and language skills. (8) Fifteen elementary schools in New York City and the State University of New York at Stony Brook also use the drill mode of CAI.

The tutorial mode of CAI is more individualized and contains lecture, text, and examination material. This mode has been used quite often because it is easier for the layman to grasp. Pennsylvania State University is now offering course segments in mathematics, communication skills, and in various sciences using CAI. (10) One of the most advanced CAI centers is at Florida State University. They also use the tutorial mode for their CAI courses.

The highest form of CAI is the dialogue mode, in which the student is encouraged to participate by asking questions, directing discussion, and constructing solutions to problems. This mode requires the author to program the computer so that it will reply to the student's questions by taking into account both the programmed answer and the past conversation with the student. This mode has been developed by Dr. W. Feurgeig and others at Balt, Beranek,

and Newman, Incorporated. They developed a method whereby the dialogue mode assimilates examination and laboratory tests and the computer "discusses" tentative diagnoses with the student.

The programming languages used to implement a CAI monitor may be classified into four categories: 1. conventicnal complier languages, 2. adapted conventional complier languages, 3. interactive computing and display languages, and 4. specially devised instructional author languages. (11) The conventional complier languages and the adapted conventional complier languages such as FORTRAN, PL/1, ALGOL, COBOL, and LISP are difficult to use as CAI author languages. The main problem with these languages is that their complexity generally requires an experienced programmer to write course material. They were designed for scientific or business data processing and are inefficient for producing large amounts of text and accepting a wide range of student answers. The complexity of these languages gave rise to the interactive computing and display languages and specially devised instructional author languages. The remaining two classes of languages allow the author to produce large amounts of text and write interactive code which accept a wider range of student responses. Some of the languages in this class are: COURSEWRITER III, DIALOG, FOCAL, COPI, PILOT, INFORM, and

TUTOR. Generally these languages provide many more features than the class I and II languages which allow the author to maintain student records between sessions. The flow of the course in these languages is controlled by conditional branching depending on the author's intentions and the students answers.

Normally new operators are trained by an on-the-job training course. A thorough literature search reveals no completed works in the area of operator training by Computer Assisted Instruction. The operator training course discussed in this thesis was implemented using the IBM COURSEWRITER III language. A discussion of the course format is found in the next chapter.

CHAPTER II

COURSE DESCRIPTION

MATERIAL COVERED

The HASP system utilizes fourteen keywords to establish communications between the operator and the HASP program. The operator training course implemented in this thesis discusses the portion of the HASP system which is of interest to the console operator. HASP command formats are stated and several examples are given. Once the parameters of the course are covered, the student operator begins a series of several sections where each of the fourteen keywords are explained along with examples of the different command formats. Course flow then continues to a question and answer section where the novice operator enters commands in response to questions typed by the computer. Upon completion of the fourteen command sections, the student is branched to a comprehensive test where he is timed and scored. The scores from the test are presented in the last section and the student is allowed to "sign off" or branch to a section which he feels should be reviewed.

HASP OVERVIEW

HASP is a program which assumes control of certain devices and portions of the Operating System. HASP monitors and directs the flow of problem programs entering the system for execution. Jobs entering the system flow through five distinct steps which are under the control of the HASP system. Queues are established for each of the five steps (INPUT, EXECUTION, PRINT, PUNCH, and PURGE) and job priority within each queue is calculated based on assigned parameters. The queue priorities are generally determined by system resources required by a job in relation to requirements by other jobs in the queue.

The HASP system is capable of handling many different types of jobs in each queue. Perhaps the simplest way to understand how a job is processed in the HASP system is to follow a typical job through the five steps from input to purge. A job first enters the HASP system from an input device. This may be a hardware device such as a card reader or tape drive used to process physical input, or a logical reader used to process input streams which are data sets on disk or data sets constructed by executing programs.

HASP places the input on a system spooling device, assigns a job number, and enters pertinent information about the new job in the correct HASP queue. The job

resides in this queue until it is selected for execution. Execution is the second step as the job flows through the system. Once the job is in execution HASP monitors the job and supplies input as required and retrieves print and punch output data. HASP places the job in the print queue once execution is completed to await selection for print. Selection for print is based on the job priority as assigned by HASP. In the print step all system messages and print data sets created during execution are printed. Next the job is placed in the punch queue provided punch data sets were created during execution. Following printing and punching the job is placed in the purge queue. Here all spool area occupied by the job is returned to the system for use by subsequent jobs.

A job passes through the five steps sequentially, waiting in each step until selection. Since the HASP system may run many jobs concurrently, the operator must be capable of entering HASP commands quickly and correctly to effeciently utilize the system.

HASP COMMAND FORM

The computer operator communicates with the HASP system by use of the HASP operator commands. These commands fall into categories such as job queue commands, job list commands, device list commands, system commands, display commands, and remote job entry commands. Each command is in the form:

```
$ Verb Operand1,Operand2,...,Operandn
```

Where:

$	-	The HASP system uses the dollar sign ($) character for HASP command identification.
Verb	-	A keyword or keyword abreviation which indicates the type of action to be taken.
Operands	-	The operands are used to modify the command verb or supply additional information to identify the job or system facility to be acted upon. (1)

The HASP system allows a great deal of flexibility in entering commands. The text of a command may contain embedded blanks, upper and lower case characters, and for the most part be misspelled. All that is required is that enough information be in the text to identify the job or facility desired. The dollar sign ($) must precede the

command verb to identify the text as a HASP command. Table I contains each of the HASP verbs covered in this CAI course.

MATERIAL FORMAT

The CAI course is constructed in course blocks. Each block presents a HASP command by covering the description of the command verb and then giving several examples of the command usage along with a description of the system action taken when the command is entered. The course begins a question and answer session after presenting the description and examples. The questions are typed and the terminal pauses for the student operator's response. The responses are decoded in much the same manner as the HASP system decodes an operator command. As obvious errors are detected, additional information is typed about the correct answer. The new operator is allowed to make another attempt at answering the question correctly. Since it would be impossible to allow for all incorrect responses, the course allows three unanticipated answers before typing the correct answer. Once the operator types the correct answer the course continues to the next question.

Scores are maintained for each HASP verb. At the end of a command verb section, totals for the section are presented to the student. The totals along with the command keyword are stored in auxiliary storage. The

TABLE I

HASP COMMAND VERBS (1)

COMMAND	DEFINITION	OPERAND TYPES
$A	RELEASE	ALL JOBS OR SPECIFIC JOBS
$B	BACKSPACE	PRINTERS
$C	CANCEL	DEVICE FUNCTIONS OR JOBS
$D	DISPLAY	DISK, UNITS, LINES, REMOTES, MESSAGES, JOBS, QUEUES, ACTIVITY, INITIATORS, OR OUTSTANDING REQUESTS
$E	RESTART	DEVICE FUNCTIONS, OR JOBS IN EXECUTION
$F	FORWARD SPACE	PRINTERS
$H	HOLD	ALL JOBS OR SPECIFIC JOBS
$I	INTERRUPT	PRINTERS
$N	REPEAT	DEVICE FUNCTION
$P	STOP (AFTER CURRENT FUNCTION)	DEVICE, INITIATOR, SYSTEM, OR JOB
$R	ROUTE OUTPUT	BY ROUTING GROUP OR JOB
$S	START	DEVICE, INITIATOR, OR SYSTEM
$T	SET	DEVICE, INITIATOR, JOB, PCB IMAGE, OR SYSTEM JOB NUMBER BASE
$Z	HALT (IMMEDIATE)	DEVICE

operator may review the totals for the section before continuing to the next section. Storing of scores in auxiliary storage allows the monitor of the course to review each operator's programs over the course material.

After all command sections have been covered, the operator is presented with a fifty question test. All commands are covered by the test in no particular order. In this section the response of the operator is checked for validity to the question asked. If the command is correct a response of OK is typed. An invalid response is indicated by typing INVALID COMMAND. The test is scored for correct answers and timed. The test section is followed by a list of scores for each of the fourteen HASP commands and the number of questions asked concerning each command. The student is then allowed to branch back to a section for review or "sign off" the course.

MACRO GENERATION

While outlining the course and reviewing the various statements which would appear several times in the text, the author developed a number of "MACROS" which would generate the statements at desired locations. This method relieved the author of the laborious task of coding the same statements each time they appear in the course. COURSEWRITER III provides a means of "MACRO" generation

similar to the one used in this course. A listing of the course and "MACROS" are found in the appendix.

CHAPTER III

CCURSE FLOW

The course flow is contrclled by a combination of parameters, switches, and counters. Table II contains the COURSEWRITER III parameters used for this course. A one (1) indicates the parameter is in effect while a zero (0) indicates the parameter option is off. Future application of this ccurse may require a change of course parameters. Ihis would be accomplished by replacing the local parameter statement found in the "input" section of the course.

In addition to course flow controlled by parameters, the author has used the switches available in the CCURSEWRITER III language. Each section is assigned a switch with an initial setting cf zero. As a student enters a section of the course the switch is set to one. If the student branches back to this section at a later time a message will be typed that the section has been entered previously and he is given the option to branch to another section.

By setting parameter nine to zero, the student is allowed to enter branching commands at any time. This allows the student to study a section as long as he feels it is necessary and then branch to another section.

TABLE II

COURSE PARAMETERS (2)

Parameters	Meaning
0	Translate student response to lowercase.
1	Remove space characters from student response.
2	Remove line feed and return characters from student response.
3	Remove backspace and immediately preceding character from student response.
4	Remove tab characters from student response.
5 - 7	Reserved.
8	Student requests for help are not allowed.
9	Student requests for go to are not allowed.
10	Message HELP NCT ALLOWED is sent to all help requests.
11	Message GO TC NOT ALLOWED is sent to all go to requests.
12	With no un statement at the end of an answer group, operate as if a un with the text "INCORRECT" were present.
13	This bit on causes the system to restart a student who has been signed off at the label defined in return register 5.
14	This bit on branches the student to the label defined in return register 4 when he uses the go to command. The label is validity checked.
15	This bit on prevents polling for ca, wa, aa, or un op codes following pr's (see "Operation Code Execution Summary").
16	This bit on stores the student record at each sign-on return point rather than at sign-off time.
17	This bit on causes a branch using return register 3 contents if a branch is requested to a label that is not in the course.
18	This bit on during student status causes the contents of counter 30 to be typed.
19	This bit on causes the normal carrier return sent to the terminal following a student response to be cmitted.
20	Inhibit student comments from being recorded.

TABLE II (continued)

Parameter	Meaning
21	This bit on causes counter 29 to contain the number of characters in the student response including the EOB.
22	This bit on causes the message PRESS ENTER TO CONTINUE to be displayed on the last line of a display terminal screen when a screen erase is encountered in the course.
23	This bit on causes a code designating the input device on the display system which was used to enter the response. The code is stored in counter 1 and is: Code Meaning 1 Keyboard 2 Program Function Key (PA 1) 3 Light Pen 4 Badge Reader
24	This bit on allows the student to use paging if that facility is supported by the Coursewriter system.
25	This bit on allows the student to have continuous delayed hardcopy if that facility is supported by the Coursewriter system.
26	This bit on allows the student to have selective delayed hardcopy if that facility is supported by the Coursewriter system.
27	This bit on allows the student to have immediate hardcopy if that facility is supported by the Coursewriter system.
28 - 29	Reserved.
30	Inhibit execution of au and fp op code statements.
31	This bit on means a tranmission error occurred on the last student response.

Branching is allowed in all sections except the comprehensive test section. Once a student enters the test section, the test must be completed.

CHAPTER IV

CONCLUDING REMARKS

Experienced operators who participated in evaluating the course elected to take the comprehensive test first. Low scores were not anticipated by the experienced operators. Six people with an average of one year's operating experience averaged a score of seventy-eight. They were surprised at the amount of information they had either not known or had forgotten. Students who proceeded sequentially through the course averaged a score of approximately eighty-three on the comprehensive test. They have commented that their first contact with the operator's console was more like an extension of the course rather than being a totally new experience. Both experienced and inexperienced operators indicated that additional courses should be developed that would expose novice operators to the entire system.

As indicated in the introduction to this thesis, this course development is only one part of the information a novice operator should understand before operating a computer. Additional courses are needed which cover the Operating System (OS) commands, a course on general operator techniques, and various production job streams.

The few hours an operator spends at a terminal taking a CAI course will greatly reduce the number of hours of on-the-job training. Experienced operators could use the course as a review at regular intervals and reacquaint themselves with seldom used commands and operating techniques.

Authors of future courses related to computer operator training should look at the different options available with CAI. An extensive use of macros was employed in this course. The future courses should be constructed in such a way as to release the author from the laborious task of coding repetitive CAI instructions. Possibilities exist in constructing a course such that questions are built in a random order by use of a random number generator. Selected keyword phrases would be placed in a course dictionary and the questions selected from these phrases. Answers would be built at the same time as the questions by having a link between a phrase and the correct answer to the phrase. Wrong answers would be constructed in the same manner as correct answers. A program which uses this technique has been successfully used in the College of Law at the University of Tennessee. Computer Assisted Instruction courses in the training of computer operators are only limited to the author's ability to construct the courses in a manner which is instructive and interesting to new operators.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. IBM Corporation, "HASP OPERATOR'S GUIDE," HASP SYSTEM, (1973), 689-772.

2. IBM Corporation, COURSEWRITER III Version 3 Author's Guide, (1973).

3. Gloria M. Silvern and Leonard C. Silvern, "Computer-assisted instruction: specification of attributes for CAI programs and programmers," Proceedings of 21st National Conference, Association for Computing Machinery Washington 1966), 57.

4. Lawrence M. Stolurow, Computer-assisted Instruction, Hatward Computing Center, Technical Report No. 2 (Cambridge 1967), 1-2.

5. Sidney L. Pressey, "A Third and Fourth Contribution Toward the Coming 'Industrial Revolution' in Education," School and Society, Vol. 36 (November 19, 1932), 672.

6. B. F. Skinner, "The Science of Learning and The Art of Teaching," The Harvard Educational Review, Vol. 24 (January-December, 1954), 87.

7. Karl L. Zinn, A Basic Reference Shelf on Interactive Use of Computers for Instruction (Stanford, 1968), 4.

8. Patrick Suppes, "The Uses of Computers in Education," Scientific American, Vol. 215 (September, 1966), 207.

9. Donald E. McGoucan, "Review of the Literature," Computer-Assisted Instruction: Status Feasibility Study, Thesis, University of Tennessee (1970), 10-38.

10. H. S. Berger, "The Pennsylvania State University Computer-Assisted Instruction," Inquiry, Vol. 2, No. 2 (June, 1967), 2-4.

11. Charles H. Frye, "CAI Languages: Capabilities and Applications," Datamation, Vol. 14 (September, 1968), 34.

APPENDIX

```
sign on
cs4330
insert after cs4330
 init
pr
de
ld 11111000101010000100011011110010/p
ld labels/r3
ld 0/c0-c30
ld 00000000000000000000000000000000/s
ld                                                    /b1,0,50
ld                                                    /b1,50,50
ld 20/c27
ld 200/c28
ld b1/b2
ld b1/b3
ld b1/b4
ld b1/b5
ld b1,0,100/a,0,100
ld b1,0,100/a,99,100
ld b1,0,100/a,198,100
ld b1,0,100/a,297,100
ld b1,0,100/a,396,100
ld b1,0,100/a,495,100
ld b1,0,100/a,594,100
ld b1,0,100/a,693,100
ld b1,0,100/a,792,100
ld b1,0,100/a,891,100
ld b1,0,100/a,990,34
 initen
 friend
GEN PR   25
qu This is a Computer Assisted Instruction (CAI) course to present
   the Houston Automatic Spooling Priority (HASP) system operations
   commands. The course is constructed to give information concerning
   a HASP command, followed by a few examples, and then questions
   related to the material just covered. The questions allow for
   a few wrong answers; however if you are unable to answer correctly
   in three tries the answer will be typed. After each correct answer
   is entered, the course will move to the next question or section.
   Scores for each section will be displayed when the last question
   is answered. A comprehensive test follows the command descriptions
   and in this section the course will allow one reply per question.
   That answer will be checked much the same way as the HASP system
   would decode the command. A score of correct and incorrect
   responses will be presented at the end of the test to indicate
   any command section which should be reviewed.
   If you make a typing error while entering the command simply
   type ### and press the return key. The system will allow the
   answer to be entered again without counting the typing error as
   an incorrect answer.
 name
qu Please type your first name and we will start the course.
ep
ld 0/p0
ca(w) &
ld n &no& &n't& &wrong&/b5
ld y &correct& &right& &ok& &yes&/b4
```

```
ld b0,0,25/b2,1,25
qu Is '
ld 1/p0
ty b2
ty ' Your first name?
ep
ca(w) b5
ld 0/p0
ty Let's try again.
br name
ca(c,w) b4
ld b1/b4
ld b1/b5
ld b2/a,1,10
un Please type 'yes' or 'no'.
ld 1/p0
qu b2
   ---That's an Interesting name.
qu If you are ready to learn all about the HASP commands please type
   yes.
GEN EP
ca yes
wa no
ty Well all that I can say is Please sign off and come back later.
un Please type yes or no.
 frend
 prelab
pr
de
qu This course is constructed with several major and minor sections.
   Each major section is a letter abbreviation for a HASP command
   verb. At any time in the course the command GO TO xtotal
   (x being the letter of a major section. Ex. ctotal, ztotal) may
   be issued.  This command will branch the course to a section of
   code which prints the totals for the current section and zeros
   the registers used to accumulate scores. After the scores have
   been presented, there will be an opportunity to branch back to
   this section to chose the next Hasp command for study or continue
   sequentially through the course material. This procedure allows you
   to terminate a question and answer section once you feel a
   command has been covered adequately.
   Please press enter to continue.
 labels
pr
de
ld 0/c10-c15
ld              /b2,50,14
ty Below are the labels for this course:
ty LABEL    CONTENTS
   init     Course parameters are initialized.
   friend   Terminal operations are explained.
   labels   Course labels are presented.
   hasp     A short overview of the HASP system is presented.
   a        Release
   b        Backspace
   c        Cancel
   d        Display
   e        Restart
```

```
   f         Forward space
   h         Hold
   i         Interrupt
   n         Repeat
   p         Stop (after current function)
   r         Route output
   s         Start
   t         Set
   z         Halt (immediatly)
   Test      Comprehensive test covering all commands.
ty Press enter to continue.
pr
qu Please enter the label you wish to branch to or type x.
GEN EP
ca init
br init
ca friend
br friend
ca labels
br prelab
ca hasp
br hasp
ca a
br a
ca b
br b
ca c
br c
ca d
br d
ca e
br e
ca f
br f
ca h
br h
ca i
br i
ca n
br n
ca p
br p
ca r
br r
ca s
br s
ca t
br t
ca z
br z
ca test
br test
ca scores
br scores
ca redo
br redo
ca x
br hasp
```

```
un You have made an error in typing the label. Please retry.
ad 1/c6
br hasp/c6,qe,3
 labend
 hasp
GEN PR   27
qu INTRODUCTION
   HASP is a program which, when started by the operator, assumes
   control of selected devices and portions of the Operating System
   (OS) for the purpose of managing the subsequent flow of jobs
   submitted for execution. Under normal processing, jobs flow
   through five distinct major functions of HASP as follows:
   1. INPUT          Jobs are read into the system:
                     Each job, made up of JOB CONTROL
                     LANGUAGE (CL) and optional input data
                     cards, enters the system and is saved
                     on direct access storage (SPOOL volumes)
                     for later high speed retrieval.
   2. EXECUTION      Jobs are submitted to OS for execution:
                     as each job is selected for execution,
                     the JCL cards are retrieved and submitted
                     to an OS READER/INTERPRETER for initiation
                     by OS. During execution, each job is
                     monitored; input data is provided and
                     print/punch data created by the job,
                     along with SYSTEM messages, is saved
                     on the SPOOL volumes for later out-
                     put.
   3. PRINT          Print output for jobs is printed:
                     The SYSTEM messages and print data sets
                     created during execution are printed.
   4. PUNCH          Punch output for jobs is punched:
                     The punch data sets created during execu-
                     tion are punched.
   5. PURGE          Jobs are removed from the system:
                     Upon completion of all processing required
                     for a job, the SPOOL volume space and all
                     HASP resources associated with the job are
                     made available for re-use.
   Although each job entering the system passes sequentially through
   each function, one function at a time, all HASP functions may run
   concurrently when sufficient jobs are available for processing.
   PRIORITY QUEUEING AND SCHEDULING
   As each HASP function completes processing a job, the job is
   placed in a queue in order of HASP scheduling priority along
   with other jobs to wait for the next function. A new job to
   process is then selected from the queue of eligible jobs.
   Since jobs are in priority order on the queue, high priority
   jobs will be selected for processing in preference to lower
   priority jobs. The net effect is that high priority jobs will
   spend less time in the system than low priority jobs.
   To illustrate HASP processing of jobs, the following example
   traces a job through the system:
   Job A enters the system and is assigned HASP job number 100.
   Jobs 1 through 99 have entered the system previously and are
   being processed by other functions, queued for processing, or
   have been deleted from the system. Assuming that job 100 is
   placed in the Class A execution queue along with jobs 97, 98,
```

and 99 and is highest priority, the HASP initiator will select job 100 for OS execution when the next Class A job is selected. Job 100 is placed in the print queue upon completion of execution. Again assuming that the queue contains jobs 70, 71, 73, 80, and 92 and job 100 is highest priority, job 100 will be selected for printing when the printer is free from processing the previous job. After being printed, job 100 is then queued for punch. If the punch queue is empty and the punch is available, the job will be immediately selected for punching. After all punching for job 100 has completed, the job is then queued for purging and, when selected, is removed from the system.

HASP OPERATOR COMMANDS

Through the use of HASP operator commands the operator may communicate with the HASP SYSTEM for the purpose of displaying information, controlling the flow of jobs within the system, and controlling HASP SYSTEM facilities which are used in processing of jobs. Each HASP command falls into one of the following categories:

1. JOB QUEUE COMMANDS - Commands which search the HASP job queue and display or alter the status of jobs without regard for the job identity.
2. JOB LIST COMMANDS - Commands which search the HASP job queue and display or alter the status of jobs based upon the identity of the job(s).
3. MISCELLANEOUS JOB COMMANDS - Job commands which apply to a single job by identity.
4. DEVICE LIST COMMANDS - Commands which control the HASP peripheral devices.
5. SYSTEM COMMANDS - Commands which control the status of the HASP SYSTEM or the submission of jobs to OS/360 for execution.
6. MISCELLANEOUS DISPLAY COMMANDS - Commands which provide informative responses but do not belong to the other categories.
7. REMOTE JOB ENTRY COMMANDS - Commands associated almost exclusively with HASP remote job entry.

The following sections provide sufficient information for operator control of the HASP SYSTEM for that time period after the initial response to the HASP request for initialization options.

ENTERING HASP COMMANDS - GENERAL

HASP commands have the following form:

```
$verb   operand1,operand2...,operandn
```

Where:

- $ - HASP command identification character--all commands to the HASP SYSTEM start with the $ character.
- verb - HASP command verb--a single character verb which describes the general action which is to be taken (see TABLE 1). A longer form of the verb may be used which is partially compatible with former versions of the HASP SYSTEM.
- operands - HASP command operands--operands are used to modify the verb of the command or identify the job or system facility to be acted upon. Commas are used to separate operands when more than one operand is used.

NOTE: If more operands are entered than the command is designed to handle, the additional operands will either be ignored or be concatenated to the last acceptable operand and handled as one.

The HASP command structure allows for a great amount of flexibility in entering the text of the command. The following rules apply:

1. FOR TEXT OUTSIDE PAIRED APOSTROPHES:
   A. All alphabetic characters may be entered in upper or lower case.
   B. Blanks may be inserted at any point in the command after the initial $ for operator convenience.
   C. Apostrophes may appear in the text of the command as a text character; however, each apostrophe text character must appear in duplicate.
2. FOR TEXT INSIDE PAIRED APOSTROPHES:
   All characters must appear as required by the individual command. Text apostrophes must appear in duplicate.
3. Key words for operands may, for the most part, be misspelled. It is only necessary to enter enough information to identify the job or facility desired.

The following examples illustrate the above rules:

1. $r all, rmt 4, local
   $RALL,RMT4,LOCAL
2. $dm4,'If your job''s output is deleted, resubmit'
   $DM4,'IF YOUR JOB'S OUTPUT IS DELETED, RESUBMIT'
3. $a all or $a a
   $AA

NOTE: The first line of each example represents the operator's input. The second line represents first character of each operand underlined.

TABLE 1 HASP COMMAND VERBS

COMMAND	DEFINITION	OPERAND TYPES
$A	RELEASE	ALL JOBS OR SPECIFIC JOBS
$B	BACKSPACE	PRINTERS
$C	CANCEL	DEVICE FUNCTIONS OR JOBS
$D	DISPLAY	DISK, UNITS, LINES, REMOTES, MESSAGES, JOBS, QUEUES, ACTIVITY, INITIATORS, OUTSTANDING REQUESTS
$E	RESTART	DEVICE FUNCTIONS, OR JOBS IN EXECUTION
$F	FORWARD SPACE	PRINTERS
$H	HOLD	ALL JOBS OR SPECIFIC JOBS
$I	INTERRUPT	PRINTERS
$N	REPEAT	DEVICE FUNCTION
$P	STOP (AFTER CURRENT FUNCTION)	DEVICE, INITIATOR, SYSTEM, OR JOB
$R	ROUTE OUTPUT	BY ROUTING GROUP OR JOB
$S	START	DEVICE, INITIATOR, OR SYSTEM
$T	SET	DEVICE, INITIATOR, JOB, FCB IMAGE, OR SYSTEM JOB NUMBER BASE
$Z	HALT (IMMEDIATE)	DEVICE

ty The above information is taken directly from the IBM HASP OPERATOR'S GUILD. You should read through this manual after completing the course.

PLEASE PRESS ENTER TO CONTINUE.

endhas
a
pr(r)
de
qu The HASP system uses the letter a for the RELEASE symbol. $a may be used to release all jobs held by a previous $h command or jobs may be released one job at a time. A single job is released from the hold queue by either using the job number assigned by HASP or by using the job name. Examples of both of these formats are shown below. As an operator, you will be required to release jobs for execution which the system has held. A job is automatically placed into a hold queue if the resources required by the job are not online. Releasing a job from one of these hold queues would be done by the $a command.
On certain occasions, the operator may wish to release all jobs held by a hold all command. This is accomplished by entering $aall or $aa. Below are some examples of the release command.

COMMAND	SYSTEM RESPONSE	ACTION TAKEN
$aa or $aall	queue released	All jobs are released that were held by a $hall command.
$aj500	job(s) released	Here a job that was held by the system is released.
$a'myjob'	job(s) released	The job name 'myjob' is use to release the job.
$aj500-525	job(s) released	The job in the system with numbers 500 thru 525 will be released by this command.
$aj100,201,5	job(s) released	A maximum of 5 job numbers may be listed in one release command.

Please press enter to begin a drill using the release command.
aqu1
rd
de
qu Suppose you are running the system and 3 jobs are in the hold queue and each job uses the same disk pack. The pack has been mounted so that all that needs to be done is release the jobs. The job numbers are 156 201 and 42. Please enter one command to release these jobs.
GEN EP
ca(w) $aj156,201,42 $aj201,42,156 $aj42,201,156 $aj42,156,201
cb(w) $aj201,156,42 $aj156,42,201
ty Very good. You understand how to release multiple jobs with only one command. The system would type: Job(s) released
ad 1/c10
GEN CL $release&
GEN WL release&
GEN WL aj&
wa(w) $a156,201,42 $a201,156,42 $a42,201,156 $a42,156,201
wa(w) $a156,201,42 $a156,42,201
ty You have omitted the j from your answer. The j signals the HASP system that a job number follows. Please try again.
ad 1/c11
wa(w) $aj15620142 $aj20115642 $aj42201156 $aj42156201
wa(w) $aj20115642 $aj15642201
ty You have omitted the comma which separates the job numbers. The system would type: INVALID COMMAND Please try again.
ad 1/c11

```
wa(1) $a'&
ty The quote is not used to release jobs when a job name is used.
   Please retype the command.
ad 1/c11
GEN UN
GEN UC    $aj156,201,42
 aqu2
pr
de
qu For the next command please release all jobs held by a previous
   hold all command.
GEN EP
ca(1) $aa&
ty Very good. $aall and $aa are correct commands.
ad 1/c10
GEN CL    $release&
GEN WL    release&
GEN WA    aa
GEN WA    aall
wa $aq
ty This may seem correct but the system is expecting a key word in
    place of the letter q.
ad 1/c11
GEN UN
GEN UC    $aa
 aqu3
pr
de
qu This question may require a little thinking before you answer it
   so be careful. Enter a single command which will release every
   hold job in the system.
GEN EP
ca $aj1-999
ty If you got this one correct the first time you are doing very well.
ad 1/c10
wa(1) $aa&
ty A $hall command was not entered so a $aa command will have no
   effect on a job held by the system. Consider releasing the jobs by
    their HASP job numbers. Please try again.
ad 1/c11
GEN CL    $release&
GEN WL    release&
GEN WA    aj1-999
wa(1) $aj1-**
ty HASP job numbers range from 1 thru 999. Please try again.
ad 1/c11
GEN UN
GEN UC    $aj1-999
 aqu4
pr
de
qu So far we have been releasing jobs only by their number. Generally
   the HASP number is the information an operator needs to release a
   job but sometimes a person from the front office will call the
   machine room and request that a certain job be run next. The HASP
   system allows for a job to be released by job name if there are not
   two or more jobs in the system by the same name.  This allows
   the operator to release a job without first having to display the
```

```
   job number. So all the front office has to do is give the job name.
   For the next command assume the director phones and asks that a job
   by the name of BILLS be released for execution. What command do you
    enter?
GEN EP
ca $a'bills
ty Very Good! There will be a little something extra in your pay
    envelope this month.
ad 1/c10
GEN CL   $release&
GEN WL   release&
GEN WA   a'bills
wa $ajbills
ty The j is used in a HASP command to signal that a number follows.
    Please try again.
ad 1/c11
UN Very funny. You have just punched a hole in the middle of your
    employment record. However; we will allow another try.
ad 1/c12
GEN UN
GEN UC   $a'bills
 aqu5
pr
de
qu Please enter the command to release job number 43.
GEN EP
ca $aj43
ty The system will respond with OK
ad 1/c10
GEN WA   aj43
GEN CL   $release&
GEN WL   release&
wa $a'43
ty The quote sign is only used when a job is being released by jobname
ad 1/c11
GEN UN
GEN UC   $aj43
 atotal
GEN  L1
GEN  TY
GEN  L2
 aend
 b
pr(r)
de
qu $b is the HASP letter used for the backspace command. This command
      is a device list command and the system will only take action
      on a device which is currently active. The command may be used
      to backspace 1 page, a specified number of pages, or an entire
      data set. Below are some examples of the command with an
      explanation of what function the command performs.
          $bprt1,10    printer 1 will be backspaced 10 pages.
          $bprt2,d     printer 2 will backspace to the beginning of the
   data set.
          $brm2.pr1    remote 2 printer 1 will backspace 1 page.
      Some points to keep in mind about the backspace command are:
      1. If the number of pages are not entered, the printer will
    backspace 1 page.
```

```
    2. Printing is resumed as soon as the backspace is completed.
    3. Multiple devices may be backspaced with one comand
    4. If the beginning of the data set is encountered during a
  backspace command, printing will resume at the beginning of the
  current data set.
    5. The command will have no effect if the device is not active.
    6. A $b command issued to a punch unit will have the effect of
  restarting the current function.
                        *** Please press enter to continue ***
 bqu1
rd
de
qu Since the above description fully outlined the limits of the
   backspace command, it is time to put all of this knowledge to work
   Please enter a command which would recover the last 15 pages printed
    on printer one. The pages were damaged by being wrapped around the
   printer ribbon.
GEN EP
ca(1) $b&prt1,15
ty very good
ty b2
ty  you are going to be an expert HASP operator in no time.
ad 1/c10
GEN WA    bprt1,15
wa(1) $bprt*,15
ty Check the printer number again and retype the command.
ad 1/c11
wa(1) $b&prt115
ty The comma is missing. The computer would not know what to do with
    this command.  Please try again.
ad 1/c11
GEN UN
GEN UC    $bprt1,15
 bqu2
pr
de
qu Please backspace the job currently printing at remote 3. Also,
   assume that the operator called and said that the job should be
   restarted at the beginning of the data set. He was printing a set
   or pressure sensitive labels and the entire set should be redone.
   The job is printing on printer 2.
GEN EP
ca(1) $b&rm3.pr2,d
ad 1/c10
ty The computer would respond with OK indicating you had entered the
   command correctly. We will now move on to the next exercise.
GEN WA    brm3.pr2,d
wa(1) $b&rm*.pr2,d
ty Check the number of the remote station you are working with and try
   again.  If the printer at some other remote was active and you
   entered this command the system would backspace that printer.
ad 1/c11
wa(1) $b&rm3.pr1,d
ty Congratulations, you have just backspaced the printer which was
   printing our checks and some people will get two checks this week.
   Now please reread the question and try again.
ad 1/c11
wa(1) $b&rm3.pr2
```

```
ty This command would only backspace the printer one page. Please try
   again.
ad 1/c11
wa(1) $e&
ty This command would restart the entire job. That was not the request
   of the remote 3 operator. The job may have several data sets.
ad 1/c11
wa(1) $n&
ty This command would repeat the job. The operator only wants to
   restart at the beginning of the current data set.
ad 1/c11
GEN UN
GEN UC    $brm3.pr2,d
 bqu3
pr
de
qu If the beginning of a data set is encountered while a printer is
   being backspaced, what action will the system take?
   a. The printer will continue to backspace until the desired number
      of pages has been backspaced or the beginning of job is reached.
   b. The printer will start printing at the beginning of the data
      set regardless if the number of pages to be backspaced has been
      satisified or not.
   c. The system will notify the operator that a beginning of data set
      has been encountered and allow the operator to issue the next
      command for that printer.
         Please enter only an 'a' 'b' or 'c'
GEN EP
ca b
tv Very Good
ty b2
ty Your correct answer score is increasing constantly.
ad 1/c10
wa a
ty   Answer a is not correct. Please try again.
ad 1/c11
wa c
ty Come on now. I was only pulling your leg. Try again.
ad 1/c11
un Please enter only an  a   b  or  c
 bqu4
pr
de
qu The HASP command $bpun1,20 will have which effect on the punch?
   a. The current job punching will be requed and restarted.
   b. The punch will backspace 25 cards.
   c. The punch will be deleted.
   d. The punch unit will be backspaced 20 cards.
         Please answer only 'a' 'b' 'c' or 'd'
GEN EP
ca a
ty I see that you remembered point 6 about the backspace command.
ad 1/c10
wa b
ty What does 25 cards have to do with $bpun1,20? Please try agin.
ad 1/c11
wa c
ty Does not compute. Wrong answer.  Try again please.
```

```
ad 1/c11
wa d
ty No. See a above. Try again.
ad 1/c11
un Please enter only a letter.
 bqu5
pr
de
qu For the last backspace command please backspace printer one at the
   main center three pages. They were damaged by the printer.
GEN EP
ca(1) $b&prt1,3
ty After you review your totals we will move on to the next section.
ad 1/c10
GEN WA   bprt1,3
wa(1) $b&prt1
ty This command will only backspace the printer one page. Try again.
ad 1/c11
wa(1) $b&prt13
ty The comma is missing from your answer. Please try again.
ad 1/c11
wa(1) $b&prt1,*
ty How many pages are we backspacing? Please try again.
ad 1/c11
GEN UN
GEN UC   $bprt1,3
 btotal
pr
de
GEN L1
GEN TY
GEN L2
 bend
 c
pr(r)
de
qu $c is the HASP symbol for the cancel command. This command may be
      used in many different forms as the examples below indicate.
      The cancel command is used to terminate current functions on
      a device such as a printer, card reader, or punch unit; or
      jobs may be cancelled by using their job name.
      Command                  Action taken by the system
      $cj100                   Job number 100 will be removed from its
                               current queue and placed in the next
                               queue. If the job was awaiting execution
                               before the command was entered the job
                               will be placed in the print queue. If
                               the job was executing it will be placed
                               in the purge queue. Once the job ends
                               execution it will be purged and the
                               programmer will not receive printed
                               output.
      $cj110,297,30            Jobs 110 237 and 300 will be cancelled.
                               Their status after this command is
                               entered will depend on the queue they
                               were in before the command was entered.
                               A maximum of five (5) job numbers may
                               be listed in one cancel command.
```

$c'myjob'	Job may be cancelled by using the job name if the number is not known. If there are multiple jobs in the system with the same name, then the jobs and their numbers will be displayed. The operator will then have to cancel the job by the HASP job number.
$crdr1	The HASP system allows the operator to cancel the current function on a device provided the device is active. The command acts as a device list command so up to five (5) devices may be cancelled by entering only one command. This command will cancel reader 1. The abbreviation rdr is the only letters the system will decode. The job which is on the reader will be flushed and the JCL will be printed.
$crm6.pr2	This is the command format to cancel a device at a remote station. Once again notice the abbreviation for the remote and the device must not contain any characters other than the HASP symbols which identify them. This command will delete the job which is currently printing at remote 6 on printer 2.

The examples above cover the general formats of the cancel command. After looking at each command and the comments to the right, press enter and a group of questions will be typed so that replies may be checked. Try to do the questions without looking back at the examples.

```
 cqu1
pr
de
qu Assume you are operating the system and one of the local programmers
    request that job number 953 be purged. What command would you enter
   ?
GEN EP
ca(1) $c&j&953
cb(1) $c&j&953,953
ty Correct. Now let's move on to something a little harder.
ad 1/c10
GEN CL   $cancelJ&953&
GEN WA   cj953
GEN WL   c&j&953
wa $c953
ty The letter j is missing from your answer. Please try again.
ad 1/c11
wa(1) $c&j&
ty Please recheck the number of the job to be cancelled and try again.
ad 1/c11
GEN UN
GEN UC   $cj953
 cqu2
pr
de
qu The last command is probably used the most of all cancel
   command forms. For the next command assume that the programmer
```

```
   did not know the job number but the job name is ROCKET. What is
   the one HASP command that would delete this job if the job is
   currently executing?
GEN EP
ca(1) $c&'rocket
ty VERY GOOD
ty b2
ty We are making a lot of headway.
GEN WA    c'rocket
GEN WL    c&'rocket
wa $cjrocket
wb(1) $c&jrocket
ty The j following the c indicates that a number follows. In this case
   the job is to be cancelled by job name. Please try again.
ad 1/c11
wa(1) $delete&
ty The word delete is acceptable by the system but $c is much shorter.
   Please reenter the command using $c.
ad 1/c16
GEN UN
GEN UC    $c'rocket
 cqu3
pr
de
qu Recall that the cancel command may be used to cancel more than one
   job at a time and that a job is moved from one queue to the next
   by the $c command, then what is the one HASP command which will
   purge a job from the system if it is currently awaiting execution?
   Think about this for a minute and you may get it correct on the
   first try. I almost forgot, the job number is 3
GEN EP
ca(1) $c&j&3,3
ty VERY VERY GOOD.
ad 1/c10
GEN CL    $delete&
GEN WA    cj3,3
GEN WL    c&j&3,3
wa(1) $c&j&3
ty This command would only move the job from the awaiting execution to
   the awaiting print queue. The job would not be completely deleted
   from the system. Please try again.
ad 1/c11
wa(1) $c&'&
ty The apostrophe is only used for jobnames. Here we know the number.
ad 1/c11
GEN UN
GEN UC    $cj3,3
 cqu4
pr
de
qu So far the questions have related only to job numbers or jobnames.
   The cancel command will now be used to delete devices.
   Please type the command to delete the job printing on printer 1.
GEN EP
ca(1) $c&prt1
ty Correct. A message would be printed on the last line of output that
   the print was deleted by operator.
ad 1/c10
```

```
GEN WA    cprt1
GEN WL    c&prt1
wa(1) $c&prt,1
wa(1) $c&prt.1
ty The device and number are not separated by special symbols.
ad 1/c11
GEN UN
GEN UC    $cprt1
 cqu5
pr
de
qu New operators seem to have a lot of problems when it comes to
   working with the remote stations. This is partly caused by the
   exact forms of the Hasp commands which must be used. The format of
   the command is $ verb rm#.device, where # is the remote number.
   Please enter a command to cancel the job printing on remote 2
   printer 1.
GEN EP
ca(1) $c&rm2.pr1
ty Very Good. Now let's see how you do on the next question.
ad 1/ c10
GEN WL    c&rm2.pr1
wa(1) $c&rm2,&
ty Look closely at the command format and try again.
ad 1/c11
GEN UN
GEN UC    $crm2.pr1
 ctotal
pr
de
GEN L1
GEN TY
GEN L2
 cend
 d
pr(r)
de
qu The HASP system display command is easily the most involved and
   complicated of all the HASP commands. Since the command is
   complicated, the following discussion will be rather brief and
   in the form of an outline. The display verb is abbreviated by the
   letter d and the objects of the verb are  Disk, Units, Lines,
   Remotes, Messages, Jobs, Queues, Activity, Initiators, or
   Outstanding Request. You should become familiar with this command
   by consulting the HASP Operator's Guide. The command is used to
   display the status of verb's object. Below are a few examples of
   the display command.
   COMMAND          ACTION
   $dd              All volume serials and addresses for online disks
                    are displayed.
   $dn              The status of all HASP controlled, non direct
                    access devices attached to the system will be
                    displayed along with their hardware addresses.
   $dlinex          The line number x will be displayed along with
                    the remote station, hardware address, and current
                    status of the device. If an address is not
                    assigned to line, the address field will contain
                    '***'.
```

$drmtx	Remote x and the devices attached will be displayed along with each device status and hardware address.
$drmts	This command displays all remote devices, status and hardware addresses.
$dm3,'text'	The text is displayed at remote 3.
$dmr-rr,'text'	All remotes within the number range r - rr will receive the message.
$djxxx	The name and current status of job xxx will be displayed.
$d'myjob'	The same as above except the object of the display verb is a jobname. In either case if the job is not in the system a response of JOB(S) NOT FOUND will be printed.
$dq	All jobs in the HASP queues will be summarized and displayed in order of class.
$dq,xeqb	Jobs in queue b will be summarized and displayed.
$dq,prt	Jobs in the print queue will be displayed.
$dq,hold	The total number of jobs in the hold queue will be summarized and displayed.
$dq,0,prt	The total number of jobs awaiting print at the main center will be displayed.
$dq,1-3,pun	This command will show the total number of jobs awaiting punch at remotes 1 2 and 3.
$dn	The $dn command is the same as the $dq command above except that all job names are displayed. The same formats as above may be used with the $dn command; therefore, they will not be listed again here.
$da	A display of all active functions will be listed. This list includes jobs being executed, output devices, input devices, and remote lines along with descriptive information.
$df	Jobs in the special forms queue will be displayed listing the special forms number and the output device type.
$df,1-6	Same as above except awaiting at the remotes.
$di	This command lists all the HASP initiators, the status of each, and the job class the initiator will execute.
$dix	Display status of initiator x only.
$dr	The HASP outstanding reply message numbers will be displayed.

By now I am sure you see what I was talking about when I said the display command was very involved. The next thing to do is answer some questions about the display command. Please press enter to continue.

```
 dqu1
pr
de
qu For lack of a better place to start asking questions let's just
   start with the display disk command. Please enter the command to
   display all online direct access storage devices.
GEN EP
ca(1) $d1&
ty Excellent. You could also display dog, dirt, ditch, etc and
   have the same effect as your command.
ad 1/c10
```

```
GEN WL    dd&
GEN CL    $display&
GEN UN
GEN UC    $dd
 dqu2
pr
de
qu Display line 6
GEN EP
ca(1) $dl&ne6
ty The system would respond with LINE 6 ACTIVE or LINE 6 INACTIVE
ad 1/c10
GEN CL    $display&
GEN WL    d&
wa(w) $dl6 $dln6 $dli6 $dlin6
ty The system will respond with JOB NOT FOUND. Please try again.
ad 1/c11
GEN UN
GEN UC    $dlne6
 dqu3
pr
de
qu Please display remote station 2.
GEN EP
ca(w) $drm2 $drmt2 $dremote2
ty Valid entries are: $drm2 $drmt2 $dremote2 plus a few more.
ad 1/c10
wa $dr2
ty The system would respond with the requests which are outstanding.
   Can you see why? Please try again.
ad 1/c11
GEN WL    d&
GEN CL    $display&
GEN UN
GEN UC    $drmt2
 dqu4
pr
de
qu The above command displays one remote. A $drmts would display all
   remote stations, the device addresses, and the activity of each
   device. We now want to move on to the next display command and
   that is the command to display a message at the remotes.
   Please write a message to remotes 1 2 3 4 5 and 6 with one
   command.
GEN EP
ca(1) $dm1-6,&
ty Very Good.
ty b2
ty that display command was pretty tough.
ad 1/c10
wa(1) $drm&
ty You are displaying a message at remote stations. This command
   would display the remotes. Please retry.
ad 1/c11
GEN CL    $display&
GEN WL    d&
GEN UN    UN
GEN UC    $dm1-6,'text'
```

```
 dqu5
pr
de
qu Display job 937.
GEN EP
ca(1) $dj&937
ty JOB  937  ARDVARK AWAITING EXECUTION B  PRIO 9  T- 2
   very good.
ad 1/c10
wa(1) $d&,&
ty A comma should not appear in this display command. Please retry.
ad 1/c11
GEN CL   $display&
GEN WL   d&
GEN UN
GEN UC   $dj937
 dqu6
pr
de
qu Display the number or a job with the name of APPLE.
GEN EP
ca $d'apple
ty Correct.
ad 1/c10
wa $dapple
ty The single quote ' is missing. Please retry.
ad 1/c11
GEN CL   $display&
GEN WL   d&
GEN UN
GEN UC   $d'apple
 dqu7
pr
de
qu Type the command to display a total of all hold jobs in the queue.
GEN EP
ca(1) $dq,h&
ty Very good.
ad 1/c10
wa $dhold
ty The system would not recognize this command. Try again.
ad 1/c11
wa $dqhold
ty The comma is missing from the command. Retry.
ad 1/c11
GEN CL   $display&
GEN WL   d&
GEN UN
GEN UC   $dq,hold
 dqu8
pr
de
qu We have covered a number of display commands and most have been
   very simple so let's try one that is just a little harder. Using
   only one command, display the names of jobs awaiting print at
   remotes 1 2 3 and 4.
GEN EP
ca $dn,1-4,prt
```

```
ty Good Good Great you got it correct. The system would list the
   names of all jobs awaiting print at the remotes.
ad 1/c10
wa(1) $dn,1,2&
ty The remote numbers are not separated by commas. Try a range of
   numbers.
ad 1/c11
wa(1) $d&q&
ty The names of the jobs were requested not just the number awaiting.
ad 1/c11
GEN CL    $display&
GEN WL    d&
wa $dn,1-4prt
ty A comma must be used before the printer abbreviation.
ad 1/c11
GEN UN
GEN UC    $dn,1-4,prt
 dqu9
pr
de
qu What command will display the activity on the system?
GEN EP
ca $da
ty Correct.
ad 1/c10
wa(1) $d&q&
ty The dq command will only display the queues. Please try again.
ad 1/c11
wa(1) $d&n&
ty This command will display the names of all jobs in the system.
   Please retry.
ad 1/c11
wa da
ty This OS command will display the core activity but we are
   interested in all HASP activity. Please try again.
ad 1/c11
GEN CL    $display&
GEN UN
GEN UC    $da
 dqu10
pr
de
qu Just a little longer. We are getting close to the end of the
   section on display commands. Display the special forms jobs
   awaiting print at remote 6.
GEN EP
ca $df,6
ty Correct.
ad 1/c10
wa $df6
ty The system will respond with FORMS QUEUE EMPTY but it will be
   referring to the forms queue for the main center. Retry.
ad 1/c11
wa $df
ty This command will display the forms queue for the main center.
ad 1/c11
GEN CL    $display&
GEN WL    d&
```

```
GEN UN
GEN UC    $df,6
 dqu11
pr
de
qu Display the initiators.
GEN EP
ca $di
ty Very Good.
ad 1/c10
wa(1) $d&init&
ty Invalid command. The letter i must be used. Please retry.
ad 1/c11
GEN CL    $display&
GEN WL    d&
GEN UN
GEN UC    $di
 dqu12
pr
de
qu Now for the last one. Display any HASP request.
GEN EP
ca(1) $dr&
ty Correct
ty b2
ty You have earned a gold star for getting this far.
ad 1/c10
GEN CL    $display&
GEN WL    d&
GEN UN
GEN UC    $dr
 dTOTAL
pr
de
GEN L1
GEN TY
GEN L2
 dend
 e
pr
de
qu The next HASP command to be taken under consideration is the
   restart command. The restart command is used for devices and jobs
   in execution. The HASP symbol for restart is the letter e.
   The command is a device list command allowing a maximum of five (5)
   devices to be restarted by entering one command. Some examples are
   below:
   COMMAND                    SYSTEM ACTION
   $eprt1                     The current job printing on printer 1
                              will be returned to the printer queue and
                              requeued for printing. The last line
                              printed on the printer will state that
                              the printer has been restarted by the
                              operator. When the job is printed the
                              next time the entire job will be printed
                              from the beginning. This command is
                              useful if the printer has malfunctioned
                              and some of the output was lost.
```

$erm6.pr2	This command will do the same as the above command except printer 2 at remote 6 will be the printer affected.
$epun1,prt2	The punch at the main center will be restarted at the same time printer 2 will be restarted.

Something to remember about this command is that the current jobs are returned to the output queue and go through the Hasp priority selection routine for output. This means that the job may not be the next job to be outputed on the device and also the job may be outputed on another device if there is more than one device type on the system. Also consider that if you do not want the output to come up next, a $hjxxx or $pnnn command should be entered before the device is restarted. (xxx is a HASP job number and nnn is a HASP device name).

Recall, from the brief discussion above, that the $e command may be used with a job which is executing. The job may be restarted by job name or by the HASP job number. The job will be returned to the awaiting executing queue once the current execution is completed. Now let's look at some of the advantages and disadvantages of this command in reference to jobs in execution.

1. The command is only valid while a job is currently executing.
2. The entire job stream will be executed again.
3. No printed output will be produced by the first execution of the job.
4. All permanent data sets will remain from the first time the job was run. This may mean that the job will terminate due to a JCL error the second time the job is run because of duplicate names on a system volume.
5. The job may be held, restarted, and cancelled in that order so that the job may be run at a later time without rereading the program job stream. This is useful if a tape is missing.

From the above comments, the operator must be very familiar with the job stream before restarting a job.

A few questions of the restart command will follow.

```
 equ1
pr
de
qu Please enter a command to restart printer 1.
GEN EP
ca $eprt1
cb(1) $e&prt1
ty Very Good. The printer will stop at current page, print a line
   stating that the printer was restarted, and return the job to the
   print queue.
ad 1/c10
GEN WA    eprt1
GEN CL    $rest&
GEN UN
GEN UC    $eprt1
 equ2
pr
de
qu This question is a little harder so answer with caution.
   Please restart remote 1 which is line 1.
GEN BP
ca(1) $el&ne1
```

```
ty Excellent. If the entire remote station is to be restarted it must
   be restarted by the line number associated with that remote.
ad 1/c10
wa(1) $er&
ty You have fallen into the trap of trying to restart the remote.
   Try restarting the line.
ad 1/c11
GEN WA    elne1
GEN WL    e&
GEN CL    $rest&
GEN UN
GEN UC    $elne1
 equ3
pr
le
qu Please enter a command to requeue job 537. The job is currently
   executing.
GEN EP
ca $ej537
ty Very Good.
ad 1/c10
wa(1) $j&
ty It will not be necessary to hold the job only restart it.
ad 1/c11
wa $e537
ty The j is missing from your answer. Please try again.
ad 1/c11
wa(1) $e'&
ty The single quote is only used with the job name. Here only the job
   number is known. Please try again.
ad 1/c11
GEN WA    ej537
GEN CL    $rest&
GEN UN
GEN UC    $ej537
 etotal
GEN L1
GEN TY
GEN L2
 eend
 f
pr(r)
le
qu $f is the symbol used for FORWARD spacing a HASP printer device.
      The command has two options. The printer may either be FORWARD
      spaced a given number of pages or an entire data set.
      The following are examples of the FORWARD space command:
      $fprt2,d       Entering this command will cause printer 2 to
                     be forward spaced to the next data set.
      $fprt1,10      This command will space printer 1 forward by
                     10 pages.
   Some points to remember about the forward space command are:
      1. If the amount of forward spacing is not specified (that is
         the number of pages or data set) then the printer is forward
         spaced 1 (one) page.
      2. If the device is not active no action will be taken.
      3. If the device being forward spaced is a punch unit, the
         effect will be to RESTART THE CURRENT FUNCTION.
```

```
      4. The forward space command is a device list command and up to
         5 (five) units may be forward spaced with one command.
      5. A maxium of 9999 pages may be forward spaced by one command.
                          *** Please Press Enter To Continue ***
 fqu1
pr
de
qu Now that you know all there is to know about the FORWARD space
   command, it is time to use this new knowledge as if you were
   operating the computer. Please enter the command to FORWARD space
   printer one ten pages.
ep
ed all
ca(1) $f&pr&t&1,10
ty Very Good. I will add 1 to your score.
ad 1/c10
GEN WA   fprt1,10
wa(1) $f&pr&t&110
ty You have omitted the comma before the number of pages. Please try
    again.
ad 1/c11
GEN UN
un The correct answer is $fprt1,10  Please type it.
ad 1/c14
 fqu2
qu The last command was pretty easy, especially since the answer was
   an example in the description of the forward space command.
   Now let's move on to something a little more difficult. Please
   enter the command to skip to the next data set for the job
   currently being printed at REMOTE station 2 on printer 1.
ep
ed all
ca(1) $f&rm2.pr1,d
ty Very Good.
ty b2
ty   That was a tough one and you got it correct.
ad 1/c10
GEN WA   frm2.pr1,d
wa $frm2,d
ty Which printer at remote 2 are you forward spacing. Try again.
ad 1/c11
wa $fpr1,d
ty Which remote station are you working with? Please try again.
ad 1/c11
wa $frm2.pr1
ty This command would forward space printer 1 at remote 2 but for
   only one page. We want to skip the entire current data set.
ad 1/c11
GEN UN
un The correct answer is $frm2.pr1,d   Please type it.
ad 1/c14
 fqu3
qu What effect will the following command have on the job punching
   on punch 1 at the main center?   $fpun1,10
     A.  The punch unit will be forward spaced 10 cards.
     B.  The current job will be deleted.
     C.  The current job will be returned to the punch queue and
         the entire job will be repunched.
```

```
   Please answer 'A'  'B'  or  'C'
ep
ed all
ca c
ty I am glad to see you were not fooled by choice A.
ad 1/c10
wa a
ty Choice A may seem correct at first glance, but consider the problem
   involved in skipping a few cards in a deck and try again.
ad 1/c11
wa b
ty If the punch is to be deleted the $c command would be used.
ad 1/c11
un Please enter only an 'A'  'B'  or  'C'
ad 1/c13
 fqu5
qu For the next forward space command question please enter the
   command to skip to the next data set of the job printing on
   printer 3 at the main center.
ep
ed all
ca(1) $f&pr&t&3,d
ty Very Good
ty b2
ty  Now we will move on to the next HASP command.
ad 1/c10
GEN WA   fprt3,d
wa(1) $f&prt*,d
ty Look back and check the printer you are working with. If one of
   the other printers was active and you used its number the command
   would still be valid and someone would be missing part of their
   output.  Please try again.
ad 1/c11
wa(1) $f&prt3
ty We want to skip an entire data set. The above command would only
   skip one page. Please try again.
ad 1/c11
GEN UN
un The correct answer is $fprt3,d  Please type it.
ad 1/c14
 ftotal
rd
ie
GEN L1
GEN TY
GEN L2
 fend
 h
pr
de
qu The HASP hold command is simple in that the command is used to
   hold jobs in one queue to keep the job from going to the next
   queue. This may be done by a $hall command to hold all jobs in
   the system or holding individual jobs by number or name. Below
   are a few examples and then a few questions.
   COMMAND          SYSTEM ACTION
   $ha              The system will respond with OK and all jobs
                    currently in the system will be held. Another
```

```
                form of this command is $h all.
   $hj731       Job 731 will be placed in hold. If the job is
                executing, the job will be held for print once
                it ends execution.
   $h'carpool   The job carpool will be held. Using the job name
                as the object of a HASP verb saves time if the
                job number is not known.
   Once the hold all command has been issued, the jobs may be
   released by a release all command or the jobs may be released
   one job at a time by number or name.
   Press enter and a few short questions will be asked about the
   HASP hold command. The letter h is the abbreviation for the verb.
 hqu1
pr
de
qu Please hold all jobs in the system.
GEN EP
ca(1) $ha&
ty OK.
ad 1/c10
wa(1) $h&q&
ty This is an invalid command. You want to hold all jobs. Retry.
ad 1/c11
GEN CL   $hold&
GEN WL   h&
wa(1) $hj1-&
ty Now this is going to cause many many problems. The jobs will be
   held but when they are released a $aall command will not release
   them. Now a $aj1-999 will release the jobs held but this command
   will release every job in the system. There is a better way.
   Please try again.
ad 1/c11
GEN UN
GEN UC   $hall
 hqu2
pr
de
qu Hold the job in the system by the name of check.
GEN EP
ca $h'check
ty correct.
ad 1/c10
wa(1) $h&j&
ty The letter j is only used when a job number follows.
ad 1/c11
wa(1) $h&,&
ty A comma does not appear in this command.
ad 1/c11
GEN  CL  $hold&
GEN  WL  h&
GEN  UN
GEN  UC  $h'check
 hqu3
pr
de
qu Now for one quick final hold command question. Use one command
   to hold jobs 1 thru 15.
GEN EP
```

```
ca $hj1-15
ty Correct. Now on we go.
ad 1/c10
wa(1) $h&a&
ty There may be more than 15 jobs in the system. Please retry.
ad 1/c11
wa(1) $h&,&
ty A comma should not appear in this command. Try a range of numbers.
ad 1/c11
GEN CL    $hold&
GEN WL    h&
GEN UN
GEN UC    $hj1-15
 htotal
GEN L1
GEN TY
GEN L2
 hend
 i
pr(r)
de
qu $i is the HASP symbol used for the INTERRUPT command. The command
      is only valid in reference to a printer device. The current
      function on the HASP printer device, if active, will be check-
      pointed and terminated. The job is returned to the HASP output
      queue to be rescheduled. The job will be placed in the queue
      and solicited for printing depending on the job priority. Once
      the job is selected for printing, the print will start one page
      back of where the job was interrupted. A continue separator page
      will be printed at the beginning of the job. If the job is to be
      held for printing at a later time a $h command should be issued
      before the printer is interrupted. This will place the job in
      the print hold queue so that printing will not start right away
      on the job  just interrupted.
      IF the device is a punch unit the job will be restarted from the
      beginning of the job. Please press enter to continue
 iqu1
pr
de
qu Since the interrupt command is simple in format only a couple of
   questions will be asked to give an idea of how the command is used
   Please type the command to interrupt the job printing on printer 1.
GEN EP
ca $iprt1
ty correct.
ad 1/c10
GEN CL    $interrupt&
GEN WA    iprt1
GEN WL    interrupt&
wa(1) $i&,prt1
ty A comma should not appear in the interrupt command. Please retype.
ad 1/c11
GEN UN
GEN UC    $iprt1
 iqu2
pr
de
qu Please enter a command to interrupt the job printing at remote 6
```

```
   on printer 2.
GEN EP
ca $irm6.pr2
ty VERY GOOD. The commands for the remote stations are just a little
   more complex than commands for the main center.
ad 1/c10
GEN CL   $interrupt&
GEN WL   interrupt&
GEN WA   irm6.pr2
wa(1) $i&rm6,pr2
ty The remote station and printer is separated by a period. Please
   type the command again.
ad 1/c11
wa(1) $ir&e6.p&r2
ty The abbreviations for remote & printer (rm6.pr2) are required.
ad 1/c11
GEN UN
GEN UC   $irm6.pr2
 iqu3
pr
ie
qu Sometimes as an operator you will be asked to bring the system down
   as soon as possible. The printers may have very long jobs printing
   to where several minutes may be required to allow the job to
   complete. Suppose a 40K line job has just started on printer 2 and
   your supervisor requests that the system be brought down ASAP.
   The command $pprt2 would be entered and then the printer would be
   interrupted. What command should be entered?
GEN EP
ca $iprt2
ty Good. We will now go on to the next command.
ad 1/c10
GEN CL   $interrupt&
GEN WA   iprt2
GEN WL   interrupt&
wa $iprt1
ty Printer 2 is the printer to be interrupted. Please try again.
ad 1/c11
wa(1) $z&
ty This command would halt the printer but not requeue the print.
GEN UN
GEN UC   $iprt2
 itotal
GEN L1
GEN TY
GEN L2
 iend
 e
pr(r)
ie
qu The next HASP command we will cover is the repeat ($n) command.
   This command is simple because it is only used in reference to
   output devices. These output devices being HASP printer and punch
   units. The $n command is used to produce a second copy of a job
   which is currently printing. The last line printed on the
   first copy reads PRINTER REPEATED BY OPERATOR. As an operator
   you will be asked from time to time to produce a second copy of
   output. This is the format of the command.
```

```
   COMMAND          SYSTEM RESPONSE      ACTION TAKEN
   $nprt1                OK              The current job will continue
                                         to print on printer 1. After
                                         the end page the job will be
                                         queued for print again.
   $npun1                OK              The punch will produce two
                                         complete copies of the punched
                                         deck.
   Please press enter to continue. There will only be 2 questions.
 nqu1
pr
ie
qu Please enter the command to get a second copy of the job currently
   printing on printer 2.
GEN EP
ca $nprt2
ty Ok
ad 1/c10
GEN WL   repeat&
GEN WA   nprt2
wa(1) $restart&
wb(1) $e&
ty Using this command would cause the printer to restart the job
   before the first copy was completed. Please try again.
ad 1/c11
wa $rprt2
ty The above is an invalid command. Please try again.
ad 1/c11
GEN UN
GEN UC   $nprt2
 nqu2
pr
ie
qu Please produce a second copy of the job punching on unit 1.
GEN EP
ca $npun1.
ty VERY Good. We will take a quick look at the scores and move on.
ad 1/c10
GEN WL   repeat&
GEN WA   npun1
GEN CL   $repeat&
wa(1) $npu1
ty The 3 letters must be used to name the punch (pun). Try again.
ad 1/c11
wa $nunit1
ty The abbreviation for the punch (pun1) must be used. Please retype.
ad 1/c11
GEN UN
GEN UC   $npun1
 ntotal
GEN L1
GEN TY
GEN L2
 nend
 p
pr(r)
de
qu The HASP letter p is used for the stop or drain command verb.
```

The verb may have as its object a DEVICE, INITIATOR, SYSTEM, or JOBS. The object of the verb p is marked as stopping, or draining. This means that the current operation(s) will be completed and no additional action will be taken on that verb object until a $s command has been issued. The best way to cover this command is by several examples along with a description of the action taken by the system.

COMMAND	SYSTEM ACTION
$pprt1	The current job printing on printer 1 will be completed and no additional job will be printed on that printer until a $sprt1 is issued.
$ppun1	The punch will be drained the same as the printer above.
$pi2	Initiator 2 will not execute any additional job after the current job finishes.
$pi	All initiators will be drained.
$pj10	Job 10 will be deleted after the current function is completed. This command is useful say if the jobs printed output is desired but not the punched output.
$p	A command of $p without an object assumes that the entire HASP system is to close down as soon as an ALL AVAILABLE FUNCTION COMPLETE message is issued.

The above examples cover most uses of the p verb. We will now look at a few questions and allow you to respond with hopefully the correct command.

```
 pqu1
pr
de
qu Printer 2 is currently printing a job.  Enter a command which will
   cause the printer to stop as soon as the job finishes.
GEN EP
ca $pprt2
ty Excellent
ty b2
ty you are becoming quite an operator.
ad 1/c10
wa $prt2
ty The second p is missing.  Please try again.
ad 1/c11
GEN WA    pprt2
GEN CL    $drain&
GEN CL    $stop&
GEN UN
GEN UC    $pprt2
 pqu2
pr
de
qu Now enter one command which will drain printers 1 and 2.
GEN EP
ca(w) $pprt1,prt2  $pprt2,prt1
ty Very Good.  The p verb is a device list command and a maximum of
   five (5) devices may be drained by using one command.
ad 1/c10
GEN WL    p&
GEN CL    $drain&
GEN CL    $stop&
wa(w) $pprt1 $pprt2
```

```
ty This would only drain one of the printers.  Please drain both with
   only one command.
ad 1/c11
wa $pprt1prt2
ty The comma must separate the two devices.  Please retry.
ad 1/c11
GEN UN
GEN UC    $pprt1,prt2
 pqu3
pr
de
qu Now assume that you would like to run a job in the high area of
   core in a MVT system. You must first stop any additional jobs
   from beginning execution. What HASP command would be entered?
GEN EP
ca $pi
ty Very Good. The system would respond with OK.
ad 1/c10
wa $p
ty This command would drain the entire system. That is not the answer.
ad 1/c11
wa(1) $h&
ty This command might hold all the jobs but what if additional job
   were read in after the $h command was issued? There is a better
   way. Please try again.
ad 1/c11
GEN WA    pi
GEN CL    $stop&
GEN CL    $drain&
GEN UN
GEN UC    $pi
 pqu4
pr
de
qu A real quick one now. What command will drain the entire HASP
   system.
GEN EP
ca $p
ty OK  Next are your totals for this section.
ad 1/c10
wa $pall
ty NO. Try again.
ad 1/c11
GEN CL    $drain&
GEN CL    $stop&
GEN WA    p
wa $phasp
ty This command would stop the HASP program. All you want to do is
   drain the system.
ad 1/c11
GEN UN
GEN UC    $p
 ptotal
GEN L1
GEN TY
GEN L2
 pend
 r
```

```
pr(r)
de
qu The HASP route command is used to route partial output for a job,
   an entire job, or an entire group of jobs. The format of the
   command is:        $r type, for-id  where:
         type:     ALL  all output for the specified job(s) is routed
                   PRT  printed output is to be routed.
                   PUN  punched output is to be routed.
         for-id:   JOBn The designed output for job n is to be routed
                   Local The designated output for all jobs in the
                   system with local output will be routed to-id.
                   Device The designated output for all jobs in the
                   system and scheduled for a device will be routed
                   RMx The designated output for all jobs in the
                   system routed to RMx will be routed to-id
         to-id:    Local  jobs specified in the for-id will be routed
                   to local devices.
                   Device jobs are routed from for-id to this device
                   RMx jobs are routed from for-id to remote x
   Notes:  1. Jobs may be routed to remotes which do not exist.
           2. RMT0 is equivalent to specifying Local.
   Examples of the route command:
   COMMAND                    SYSTEM ACTION
   $ra11,job10,rmt6           All output associated with job 20 will
                              be routed to remote 6.
   $rprt,rm2,local            All the print currently in the queue for
                              remote 2 will be routed to local printers
   $rpun,j437,local           All punch output for job 437 will be
                              routed to a local punch. The print for
                              job 437 will go to the designated remote.
                              Notice that a j may be used in place of
                              the word job.
   Please press enter to continue to the questions on the route verb.
 rqu1
pr
de
qu Please route the print output for job 576 to remote 6.
GEN EP
ca(1) $rprt,j&576,rmt6
ty Excellent
ad 1/c10
GEN WA   rprt,j576,rmt6
GEN CL   $route&
wa(1) $ra11&
ty This command would route all output. Only the print is to be routed
ad 1/c11
wa(1) $rprtj&576r&6
ty Commas must separate the fields of the route command. Please retry.
ad 1/c11
GEN UN
 rqu2
pr
de
qu Route all output awaiting print at remote 2 to the local printers.
GEN EP
ca $rall,rm2,local
ty Very Good. You seem to be learning the route command.
ad 1/c10
```

```
GEN CL    $route&
GEN WL    route&
wa(1) $rprt&
ty The output may consist of punched cards too. Please try again.
ad 1/c11
GEN UN
GEN UC    $rall,rm1,local
 rqu3
pr
de
qu Route all jobs from remote 2 to remote 1.
GEN EP
ca $rall,rm2,rmt1
ty OK. We will now take a look at your scores for this section.
ad 1/c10
GEN CL    $route&
GEN WL    rall&
wa $rall,rm1,rmt2
ty You have the jobs going in the wrong direction. Please try again.
GEN UN
GEN UC    $rall,rm2,rmt1
 rtotal
pr
de
GEN L1
GEN TY
GEN L2
 rend
 s
pr(r)
ie
qu The HASP start command is used to start input and output devices,
   HASP initiators, and the entire HASP system.  The letter s is used
   for the verb abbreviation. Below are examples of the different
   start commands.
   COMMAND.          ACTION TAKEN BY THE SYSTEM
   $srdr1            Reader 1 is started. This command is required after
                     a device has been drained by the $p command. Each
                     device must be started if the device was not ready
                     when the HASP system was started. A device not
                     ready is automatically drained when $s command
                     is entered to start the HASP system.
   $slne1,xyz        Line 1 with password xyz is started.
   $si               All initiators are started.
   $si3              Initiator 3 is started.
   $sprt1,pun2       Printer 1 and punch 2 are started by the same
                     command. A maximum of 5 devices may appear as
                     objects to one start command.
   $stpe1,380        An input tape on unit 380 is started. If the unit
                     address has been specified in a previous command
                     the unit address should not be used.
   $scon1            The specified HASP console will be started for all
                     logical console classes of messages with all levels
                     of importance.
   The few examples above are representative of the start command
   formats. The following questions cover the most frequently used
   forms of the start command. Please press enter to continue.
 squ1
```

```
pr
de
qu Please start reader 2.
GEN EP
ca $srdr2
ty ok
ad 1/c10
wa(1) $srea&
ty Please use the rdr abbreviation for reader.
ad 1/c11
GEN CL   $start&
GEN WL   sr&
wa $srdr1
ty Reader 2 is the reader to be started. Please retry.
ad 1/c11
GEN UN
GEN UC   $srdr2
 squ2
pr
de
qu Start line 3 with a password of physics.
GEN EP
ca $sline3,physics
ty very good. The password is 0 to 8 characters. A line may not
   have a password.
ad 1/c10
wa(1) $sline&
ty Please use the abbreviation for line (lne).
ad 1/c11
wa(1) $sline3ph&
ty The comma which separates the line number and the password is
   missing.
ad 1/c11
GEN CL   $start&
GEN WL   s&
GEN UN
GEN UC   $sline3,physics
 squ3
pr
de
qu Please start console 1 and console 2 with one command.
GEN EP
ca(w) $scon1,con2 $scon2,con1
ty Excellent.
ad 1/c10
wa(1) $scons&
ty You must use the abbreviation con for console in this command.
ad 1/c11
GEN CL   $start&
GEN WL   s&
GEN UN
GEN UC   $scon1,con2
 squ4
pr
de
qu Please start all HASP initiators.
GEN EP
ca $si
```

```
ty Very Good
ty b2
ty You are starting to show progress.
ad 1/c10
wa(1) $s&all&
ty No. That command will not do.  Please try again.
ad 1/c11
GEN CL    $start&
wa(1) $sinit&
ty The letter i is all that is needed. Please try again.
ad 1/c11
GEN WA    si
GEN UN
GEN UC    $si
 squ5
pr
de
ty When IPLing the system or starting the HASP program, the line
   ENTER HASP REQUESTS will be printed on the console. The operator
   reply is $s.    This will start the HASP system. Please press
   enter to continue.
 stotal
GEN L1
GEN TY
GEN L2
 send
 t
pr(r)
de
qu The set command is represented by the letter t. This command is
   very involved and complicated in that the objects of the set verb
   are many with several fields. For this reason only the more common
   forms of the command will be shown and it is recommended that the
   student read the section covering the HASP set command found in
   the HASP OPERATOR'S GUIDE before completing this course.
   A brief description of set command with an explanation of the
   symbols commonly used follow.
   $t reader   ,  a = authority
                  h
      printer  ,  c = 1
      punch          carriage
               ,  f = rx...
                     sx...
                     ax...
                     n
               ,  s = yx...
               ,  t = train
               ,  s = yx...
                     nx...
      console  ,  level
               ,  class
               ,  rx...
               ,  a = authority
      con,level,class,class,...
   Each object of the set command verb will be explained briefly.
   $t reader,hx... The h places all jobs read by the reader into hold
            a=authority   allows HASP command to be submitted by the
                          designated reader.
```

0 - display only
1 - system control
2 - device control
4 - job control

Some examples are:

COMMAND	SYSTEM RESPONSE	ACTION TAKEN
$trdr2,h	OK	subsequent jobs are held.
$trdr1,A=4	OK	job control commands allowed.
$trm6.rdr1,h	OK	Jobs from remote 6 are held.

```
$t printer  , c = 1        single space printer output
   punch
            , F = rx...    Reset printer or punch to standard forms
              = n          Set printer or punch to a forms number n
            , s = yx...    Allow separator pages for output devices
              nx...        Disallow separator pages on output.
```

Examples of the above are:

COMMAND	ACTION TAKEN BY THE SYSTEM
$tprt1,F=0216	Printer 1 is set to print forms 0216.
$tpun1,F=7001	Punch 1 is set to punch forms 7001.
$tprt1,F=r	Printer 1 is returned to standard forms set up.
$tprt2,S=n	HASP separator pages will not be printed
$trm1.pr1,c=1	Remote 1 printer 1 will single space the remainder of the job currently printing.

```
$t console  , level   Level is a number 0-15 which specifies the
                      highest operator message level to eliminate
                      for the designated console.
            , class   The logical console class of messages the
                      specified console is to display in addition
                      to current classes.
                      LOG    - log console messages
                      ERROR  - error messages
                      UR     - unit record messages
                      TP     - HASP RJE line messages
                      TAPE   - tape console messages
                      MAIN   - main operator console messages
                      OS     - OS WTO messages
```

Now a few examples of the set console ($tcon) command.

COMMAND	ACTION TAKEN BY THE SYSTEM
$tcon1,15	The console will stop receiving messages.
$tcon1,reset	The console will start receiving all messages.
$tcon1,main,ur,tp	The console will receive only ur, tp, and main messages.

One other form of the set console command is: $tcon,level,class,...
The con parameter indicates that HASP is to set the level of message output for the logical console classes passed to OS consoles. The level and class options are the same as above. Recall that the description given here for the set command is only partial and the student should read the HASP OPERATOR'S GUIDE for a complete discussion of the set command.

Please press enter and a few short questions will be presented.

```
 tqu1
pr
le
qu Set console one to print only unit record messages.
GEN EP
ca $tcon1,ur
ty That one was pretty hard. The next one will be easier.
ad 1/c10
```

```
wa(l) $tcon&,1,&
ty This command will set all consoles at level 1. That is not what
   we wanted. Please try again.
ad 1/c11
wa(l) $tconsole&
ty The abbreviation for console (con) must be used.
ad 1/c11
GEN CL    $set&
GEN WL    tcon&
GEN UN
GEN UC    $tcon1,ur
 tqu2
pr
de
qu Printer 2 is printing one line per page. Please set the printer
   such that it will single space the output.
GEN EP
ca(l) $tprt2,c*1
ty Very Good.
ty b2
ty You are doing just fine.
ad 1/c10
wa(l)     $tprt1&
ty Did you read the question closely? Printer 2 is the one printing
   one line per page you just single spaced some good output.
ad 1/c11
GEN CL    $set&
GEN WL    tprt2,&
GEN UN
GEN UC    $tprt2,c=1
 tqu3
pr
de
qu Set printer  1 to print special forms # 6328. The printer is
   drained.
GEN EP
ca(l) $tprt1,f*6328
ty Excellent!
ad 1/c10
GEN CL    $set&
GEN WL    t&
GEN UN
GEN UC    $tprt1,r=6328
 tqu4
pr
de
qu Now set printer 1 back to standard forms.
GEN EP
ca $tprt1,f=r
ty The printer would start printing standard forms jobs if any were
   in the print queue.
ad 1/c10
wa(l) $tprt1,f=st&
ty You do not want to set the forms to standard. You want to reset
   the printer. Please try again.
ad 1/c11
GEN CL    $set&
GEN WL    tprt&
```

```
GEN UN
GEN UC   $tprt1,f=r
 tqu5
pr
de
qu Set reader 2 so that all jobs subsequently read will be placed
   into the hold queue.
GEN EP
ca(1) $trdr2,h&
ty The system will respond with OK.
GEN  CL   $set&
GEN  WL   trdr&
GEN  UN
GEN  UC  $trdr2,h
 ttotal
GEN  L1
GEN  TY
GEN  L2
 tend
 z
pr(r)
de
qu The last HASP operator command to be reviewed is the $z command.
   This is the symbol used by HASP to stop a device. The command
   differs from the $p command in that the device does not send or
   receive any additional information after the $z command is entered.
   The $p command allows the device to complete the current task
   before stopping the unit. The $z command is a device list command
   therefore five(5) devices may be stopped by entering one command.
   A $s command is required to start the device once it has been
   halted.  Below are some examples of the halt command.
   COMMAND                    ACTION TAKEN
   $zprt1                     Printer 1 will halt and the system will
                              respond with OK
   $zcon1                     No additional messages will be sent to
                              that console but the console will
                              continue to print until the buffer is
                              empty
   $zrdr2                     This command will halt reader 2 as some
                              person is reading their deck. The use of
                              this command is obvious if someone is
                              reading a deck of cards and you would
                              like to  meet them.  Simply enter $zrdr2
                              wait for a short time and then give them
                              a hand getting the reader going again.
                              $srdr2 will have to be entered after
                              a  sufficient amount of time.
   Please press enter and we will cover a couple of questions
   concerning the halt command.
 zqu1
pr
de
qu Please enter a command to halt reader 2.
GEN EP
ca $zrdr2
ty The card reader would stop before the deck is finished reading and
   the system would respond with OK. $srdr2 would start the reader
   again without losing any information.
```

```
ad 1/c10
wa(1) $hal&
ty The system will respond with OK but let's consider what you have
   just done. The $h is a valid HASP command and the al is a valid
   HASP verb argument so instead of halting reader 2 you entered a
   hold all command. To get out of this corner a $aa must be entered.
   Please try the halt command again.
ad 1/c15
GEN WA    zrdr2
GEN UN
GEN UC    $zrdr2
 zqu2
pr
de
qu Please enter the command to halt the punch unit 2 at the main
   center.
GEN EP
ca $zpun2
ty Very Good
ty b2
ty The punch unit would stop and as you know $spun1 would get
   it going again without losing any punched output.
ad 1/c10
wa(1) $hal&
ty The system will respond with OK but let's consider what you have
   just done. The $h is a valid HASP command and the al is a valid
   HASP verb argument so instead of halting reader 2 you entered a
   hold all command. To get out of this corner a $aa must be entered.
   Please try the halt command again.
ad 1/c15
wa(1) $zpunch&
ty The abbreviation for the punch unit must be used. Please try again
ty b2
ty .
ad 1/c11
GEN WA    zpun1
GEN UN
GEN UC    $zpun1
 zqu3
pr
de
qu For the last halt command please enter the command to halt printer
   3 at the main center.
GEN EP
ca $zprt3
ty Very Good. Now for the totals from this section and on we go.
ad 1/c10
wa(1) $hal&
ty The system will respond with OK but let's consider what you have
   just done. The $h is a valid HASP command and the al is a valid
   HASP verb argument so instead of halting reader 2 you entered a
   hold all command. To get out of this corner a $aa must be entered.
   Please try the halt command again.
ad 1/c15
GEN WA    zprt3
GEN UN
GEN UC    $zprt3
 ztotal
```

```
GEN L1
GEN TY
GEN L2
 zend
 test
GEN PR   26
qu The following is a test which covers the HASP command material
   in this course. Your answer should be in the shortest form
   that is use the letter for a command verb rather than the verb.
   This section will be timed; however correctness is the most
   important consideration while making a reply. Scores will be
   tabulated and presented at the end of this section. The test
   format is a question followed by your answer then the response
   as to the validity of the command.
         Please press enter to begin the test.
pr
de
ld t0/c8
ld 0/p14
ld 1/p8
ld 1/p9
ld 1/p11
qu Start printer one.
GEN AN   22 $s&prt1
qu Drain punch 2.
GEN AN   20 $p&pun2
qu Route all of job 537 to remote 6.
GEN AN   21 $r&a&,j&537,rm&6
qu Halt reader 2 immediately.
GEN AN   24 $z&rdr2
qu Set the systems job number counter to 1.
GEN AN   23 $tj&1
qu Release the job AMPEX from hold.
GEN AN   11 $a'ampex&
qu Backspace printer 3 to the beginning of the current data set.
GEN AN   12 $b&prt3,d
qu Send a message to remotes 1 2 3 4 telling them to sign off.
GEN AN   14 $d&m1-4,&
qu Check the status of job 186.
GEN AN   14 $d&j&186
qu Is job ORBIT in the system?
GEN AN   14 $d&'orbit
qu Forward space printer 1 of remote 6 10 pages.
GEN AN   16 $f&rm6.pr1,10
qu Start line 3.
GEN AN   22 $s&l&ne3
qu Repeat the print on printer 2.
GEN AN   19 $n&prt2
qu Cancel the punch output on punch 1.
GEN AN   13 $c&pun1
qu Restart line 6.
GEN AN   15 $e&l&ne6
qu Hold all jobs currently in the system
GEN AN   17 $h&a&
qu Interrupt the job printing on printer 4.
GEN AN   18 $i&prt4
qu Stop printer 1 immediately.
GEN AN   24 $z&prt1
```

```
qu Release all jobs in the system. A hold all command was not issued.
GEN AN    11 $a&j1-999
qu Restart printer 1 at remote 2.
GEN AN    15 $e&rm2.pr&1
qu Place jobs 536, 500, and 110 into hold.
GEN AN    17 $h&j&536,500,110
qu Repeat punch unit 1.
GEN AN    19 $n&pun1
qu Cancel job 10.
GEN AN    13 $c&j&10&
qu Set the priority of job 790 to 15.
GEN AN    23 $t&j&,p&15
qu Route all of the print output awaiting at remote 1 to local.
GEN AN    21 $r&prt,rm1,local
qu Forward space printer 2 10 pages.
GEN AN    16 $f&prt2,10
qu Backspace printer 2 10 pages.
GEN AN    12 $b&prt2,10
qu Drain line 2.
GEN AN    20 $p&l&ne2
qu Set printer 3 to print forms 6328.
GEN AN    23 $t&prt3,f*6328
qu Cancel the remaining print on printer 2.
GEN AN    13 $c&prt2
qu Display the HASP queues.
GEN AN    14 $d&q
qu Reset printer 3 to standard forms.
GEN AN    23 $t&prt3,f*r
qu Start  printer 1, reader 2, and punch 1 with one command.
GEN AN    22 $s&prt1,rdr2,pun1
qu Drain punch 2.
GEN AN    20 $p&pun2
qu Release job 291 for execution.
GEN AN    11 $a&j291
qu Display the functions active under HASP control
GEN AN    14 $d&a&
qu Route the print output awaiting print at remote 1 to remote 2.
GEN AN    21 $r&prt,rm&1,rm&2
qu Display the forms queue.
GEN AN    14 $d&f&
qu Place all jobs in hold which are read on reader 2.
GEN AN    26 $t&rdr2,h&
qu Display the names of jobs awaiting execution in queue A.
GEN AN    14 $d&n&,xeqa
qu cancel job GRAPH.
GEN AN    13 $c'graph&
qu Forward space printer 1 to the next data set.
GEN AN    16 $f&prt1,d
qu Set console 1 to accept unit record and tape mount messages.
GEN AN    23 $t&con1,ur,tp
qu Place all jobs in the system on hold.
GEN AN    17 $h&a&
qu Restart printer 3.
GEN AN    15 $e&prt3
qu Backspace printer 2 of remote 7 5 pages.
GEN AN    12 $b&rm7.pr&2,5
qu Halt printer 1.
GEN AN    24 $z&prt1
```

```
qu Display all remotes.
GEN AN    14 $d&rmt&
qu Job 12 is awaiting execution. Enter one command to purge the job.
GEN AN    12 $c&j&12,12
qu Release job 5.
GEN AN    11 $a&j&5
ld 0/p9
ld 0/p11
 testen
pr
de
ld t0/c7
qu This is the end of the test session. Press enter to receive your
   scores.
 endtes
 scores
GEN PR    28
sb c8/c7
mp 2/c9
ld The time required to take the test was       minutes. /b4,0,53
ld c7/b4,39,4
ld 0/p14
ld 0/p8
ld 0/p9
ty b4
ty
ld b3/b4
ld Your score on the test is       . /b4
ld c9/b4,27,3
ty b4
ld b3/b4
ty This is pretty good
ty b2
ty Below are the scores for each section covered on the test.
ty COMMAND    # CORRECT  # QUESTIONS
GEN LD    A 11 4
GEN LD    B 12 4
GEN LD    C 13 4
GEN LD    D 14 8
GEN LD    E 15 3
GEN LD    F 16 3
GEN LD    H 17 3
GEN LD    I 18 1
GEN LD    N 19 2
GEN LD    P 20 3
GEN LD    R 21 3
GEN LD    S 22 3
GEN LD    T 23 6
GEN LD    Z 24 3
ty
ty Any section that you feel you should review or any section with
   low test scores may be reviewed by typing GO TO R4.
ty If you do not feel that any sections need reviewing, Please press
   enter.
rd
de
qu This course is designed to give only an introduction to the HASP
   system. Your ability to become a competent operator depends on
```

```
   your desire to learn the system and all the components which
   are combined to allow the system to function properly. Simply
   being able to recite all the operator commands is not enough.
 scoend
sign off
end
```

GEN AN $command& ##

```
ep
ed all
ca(1) $command&
ty OK
ad 1/c##
br pr
```

GEN CL command

```
wa(1) command
ty The dollar sign ($) is missing from your answer.
   Please retype the command.
ad 1/c12
```

GEN EP

```
ep
ed
```

GEN L1

```
pr
ld c10/b2,50,2
ld c11/b2,52,2
ld c12/b2,54,2
ld c13/b2,56,2
ld c14/b2,58,2
ld c15/b2,60,2
```

GEN L2

```
ad c14/c13
ld c10/b4,2,3
ld c11/b4,11,3
ld c12/b4,19,3
ld c13/b4,35,3
ld c14/b4,43,3
ld CR/b4,51,1        /* CR = Carrage return character */
ty b4
ld 0/c10-c15
ld b3/b4,0,50
br pr/s9,0
id b2,50,13/a,(c27),13
ad 14/c27
ld r5/a,(c27),6
ad 1/c27
pr
qu You may type Go To R3 or press enter to continue.
```

GEN LD #1 #2 #3

```
ld #1/b4,3,1         /* single letter */
ld c#2/b4,14,3       /* counter number */
ld #3/b4,28,1        /* one digit number */
ld EOB/b4,4,1        /* EOB = End-of-Block character */
ld CR/b4,0,1         /* CR = Carrage return character */
ty b4
```

GEN PR

```
pr(r)
de
br pr/s##,0          /* ## switch number */
ty You have previously entered this section. If you wish
   to continue press enter. If not type Go To R4.
pr
de
ld 1/s##
```

GEN TY

```
ty The totals for this section are:
ty CORRECT   WRONG   UNANTICIPATED   MISSING   ANSWERS
ty ANSWERS   ANSWERS ANSWERS         $ SIGNS   GIVEN
```

GEN UC $command

```
un The correct answer is  $command  Please type it.
ad 1/c14
```

GEN UN

```
un I was unable to understand your answer. Please try again.
ad 1/c13
br re/s10,0
ld r5/a,(c28),6
br re/c28,ge,1000
ad 7/c28
ld b0/a,(c28),(c29)
un There is still something about your answer which I
   cannot decode. Please try again.
ad 1/c13
br re/s10,0
ld r5/a,(c28),6
br re/c28,ge,1000
ad 7/c28
ld b0/a,(c28),(c29)
```

GEN WA command&

```
wa(1) command&
ty Your command is correct except the dollar sign ($) is
   missing. Please retype the command.
ad 1/c12
```

GEN WL command&

```
wa(1) command&
ty The dallor sign ($) is missing from your answer.
   Please retype the command.
ad 1/c12
```

VITA

Teddy Edward Sherrod was born in Norfork County, Portsmouth, Virginia, on June 30, 1946. He graduated from Bearden High School, Knoxville, Tennessee in June 1964 and entered The University of Tennessee that same month. In May 1966 he joined the U. S. Army and served three years before returning to The University of Tennessee. Sherrod received a Bachelor of Science degree in Industrial Management under the Management Science Option in December 1972. The following January he entered the Master of Science program in Computer Science. He graduated from that program in June 1975.

While attending The University of Tennessee, he has been employed as a computer operator and assistant supervisor of computer operations. His present career plans include continuation in the Data Processing field.