THE IMPLEMENTATION OF A STUDENT INFORMATION SYSTEM FOR THE SLOAN SCHOOL PLACEMENT OFFICE

by

PETER GUSTAV HAAG

SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR

THE DEGREE OF

MASTER OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1975

Signature of Author.....
Alfred P. Sloan School of Management

Certified by.....
Thesis Supervisor

Accepted by.....
Chairman, Departmental Committee on Graduate Students

ABSTRACT

THE IMPLEMENTATION OF A STUDENT INFORMATION SYSTEM

FOR THE SLOAN SCHOOL PLACEMENT OFFICE

by PETER GUSTAV HAAG

Submitted to the Alfred P. Sloan School of Management on 9 May 1975 in partial fulfillment of the requirements for the degree of Master of Science

This thesis describes the design and implementation of an information system for the Sloan School Placement Office. The system, residing on Sloan's Prime 300 minicomputer, is a general purpose relational data-base management system. There are two primary data-bases, one containing information on current and past Sloan students, the other having information about companies that recruit at Sloan or have hired Sloan graduates. An interactive command language provides the interface between the user and the Fortran programs which handle the files. Using this language the user can generate various derivative data-bases and produce quick answer or report output. An emphasis has been placed on designing a system which is powerful, modular, and expandable, yet easy to use for a non-programmer.

Thesis Supervisor: Peter P. Chen
Title: Assistant Professor of Management Science

ACKNOWLEDGEMENTS

The author would like to thank several people who have been helpful in this project. First, Professor Chen for agreeing to supervise the work and for his helpful discussions. Second Fehmi Sami whose cooperation and discussions helped in defing the scope and nature of this work. Finally, John Woodward whose expertise provided crucial programming assistance.

TABLE OF CONTENTS

LIST OF EXHIBITS

1. INTRODUCTION

1.1 PURPOSE OF THE PROJECT

This project has a number of purposes. Foremost is the goal of providing a system that is useful to the Placement Office. The Sloan Placement Office Information System (SPOIS) is an actual implementation, not just a proposal. Frequent contact with the office enabled a realistic assessment of their desires, and the system was designed to meet these requirements.

The second goal was to get first-hand experience in the actual implementation of a Management Information System. Having designed several systems as classroom exercises the need was felt to actually carry one out.

A third objective was to gain experience working with a minicomputer, something this author had never done previously.

1.2 BACKGROUND

1.2.1 EVOLUTION

The need for and concept of a Placement Office information system was first cosidered in a project for the course 15.565, Management Information Technology II in the fall of 1974. Two Masters students, Bill Baggeroer and Jack Fox, examined various informational requirements and flows in the Sloan School (Ref.1, q.v.). They considered three main areas, the Placement Office, the Masters Program Office, and the Dean's Office. They concluded that the value of a computerized system to the latter two offices would be dubious because access to the files was infrequent and some hard-copy information (e.g. students' transcripts and letters of recommendation) would have to be kept anyway. Little computation was ever done, so it would be unlikely that the effort of creating the data-base for each year's incoming class would be sufficiently justifiable.

This left the Placement Office as the prime candidate for a system. Baggeroer and Fox concluded with a list of information that might be included in such a system and a proposal that a system be implemented. The Placement Office

was definitely interested, and Baggeroer and Fox expressed a hope that their preliminary proposal become a reality. The detailed design and implementation, they suggested, would make a good thesis topic. It was at this that this author picked up the project and decided to implement it as a Master's thesis.

In the beginning of 1975 as work was just starting Bill Baggeroer brought up the fact that another student, Fehmi A. Sami was interested in a similar project for his bachelor's thesis in the Department of Electrical Engineering and Computer Science. In a meeting with Fehmi, Bill Baggeroer, Prof. Chen, and Prof. Madnick the various possibilities were discussed. Since two people were interested in the same topic a concensus was clearly necessary. During the meeting it was, therefore, decided to split up the work, which was certainly enough for two people, yet to keep in close contact to assure each person the benefit of the other's work. Fehmi would work closely with the PLacement Office to determine specific needs and desires. He would translate these into a sort of "ideal" system. He would also look into some of the theoretical questions about the data-base design. The results of his work are documented in his thesis, "A Computer Based Student Information System-Sloan School of Management", (Ref.2)

This author would work at the implementation level,

considering choice of computer system and language and the detailed design and actual implementation of the information system. This work is documented in this report.

1.2.2 OTHER SYSTEMS

There are at MIT several other information systems that have some bearing on this project. They are discussed briefly now.

The Sloan School has two versions of an alumni system. The old version is a card system backed up on tape, while the new system, written in January 1974, is entirely a tape system. The old system has ten cards per alumnus and is complete for all of Sloan's (approximately 2000) S.M. and PhD alumni. The new system is an expanded version of the old one. The file contains a 563-character record for each alumnus and is updated with transaction cards in a batch job.

The system is used primarily for printing an annual alumni directory and various mailing lists. Eventually the new version will also be used for statistical reporting, but the data has not yet been gathered. Being a batch sequential system, however, severely limits its use to the type of things mentioned and prevents any kind of creative interaction with the data-base.

The Sloan administration has another system called the Sloan Information System, which it uses to prepare faculty load reports, cost reports, and enrollment reports. This is also a batch system and, like the Alumni System, is run on

the MIT Information Processing Center's IBM 370 via a Remote Job Entry terminal at the East Campus Computer Facility.

One system which is interactive is the Departmental Information System (DIS), designed and implemented by the School of Engineering, and used by most of its departments. It serves a function similar to that of the Sloan Information System, namely keeping current data on faculty, staff, T.A.'s, and R.A.'s. (But not students in general). The DIS resides on the Multics System (for more on Multics, see section 3.1.1) and uses RDMS (see section 4.1 for more about RDMS).

The Placement Office is getting another system built for it on Sloan's Prime computer. This is a reservation system for students to use in signing up for job interviews with recruiters who come to Sloan. This is being done by an undergraduate student, Margaret Krupp, as her bachelor's thesis for the Department of Electrical Engineering and Computer Science (Ref.3).

2 REQUIREMENTS AND GOALS

2.1 GENERAL DESIGN GOALS

In order to determine the success of an information system one must establish a set of goals before beginning any actual design. This allows a more objective evaluation during and at the end of the implementation. Whereas the purpose of this project (see section 1.1) refers to the reasons for undertaking it in the first place, design goals are a set of criteria the system should meet once it is completed. The main goals were:

(1) The system should be designed to give the Placement Office what it wants. Thus, the Placement Office must be involved in design and implementation from the beginning.

(2) The user should be isolated from the operating system as much as possible. He should not have to leave the system to perform any operations.

(3) The programs should be commented well enough so that they are virtually self-documenting.

(4) The system should be modular so that it is

(a) easy to debug

(b) easy to expand or modify

(c) easy to understand and maintain

(5) The system must be easy to use, interactive, and

patient with the user. It should prompt him when he insufficiently or incorrectly specifies a command.

(6) Help should be available on-line.

(7) The system should be cheap to maintain. Specifically, the data-base should be easy to update.

(8) The system should allow the people in the Placement Office to:

(a) save time doing the tasks that were previously done by hand

(b) do things that were previously infeasible.

2.2 PLACEMENT OFFICE REQUIREMENTS

The original concept as proposed by Baggeroer and Fox, was for an information system dealing with current students. Discussions with the Director of Placement, Leslie Hruby, however, indicated that she was satisfied with her current method of interacting with students. Her real interest in a data-base system was in generating a variety of statistics and reports on alumni. Examples of these include breakdowns of alumni by concentration, industry they are working in, geographic location and two-dimensional crossbreaks of numbers and percentages of students for varios pairs of parameters.

Thus, the system as used by the Placement Office will deal, initially at least, mainly with alumni. It is hoped that the flexibility of the system will encourage expansion into other areas such as current students.

3. COMPUTER SYSTEMS CONSIDERED

The cost-effectiveness goal strongly suggested the use of Sloan's own PRIME minicomputer on which time is available very cheaply. A complete review, however, requires examining the other possible systems. If for any reason the SPOIS had to be moved (e.g. it outgrew the Prime or Sloan got rid of the Prime) such a summary would prove useful.

There are three systems at MIT which would be candidates for supporting the SPOIS. These are considered here one at a time.

3.1 MULTICS

Multics is a powerful large-scale time-sharing system at MIT implemented on a Honeywell 6180 mainframe. It has many advantages in that it is quite secure, has a sophisticated file handling capability, and routinely supports specially written subsystems. In addition, Multics is where ROMS, upon which the present system has been modelled, resides. The major disadvantage of Multics is its high cost. The Placement Office does not have the hundreds or thousands of dollars that would be necessary to implement, maintain, and use their desired system on Multics. In future years, when

the use of an interactive data-base system has become an established and important part of the Placement Office, it may be desirable to upgrade to Multics, but for the present this is not feasible. A second problem is that despite its high cost Multics has been losing money, and there has been some talk of discontinuing the service entirely.

3.2 IPC 370

The MIT Information Processing Center (IPC) operates an IBM 370/168 which it is in the process of converting to VS2. The 370 supports IBM's Time Sharing Option (TSO). Though slightly less expensive to use than Multics, TSO is not nearly as flexible or powerful. The main advantages of the 370 are the powerful batch capability and the fact that the current Sloan Alumni System is run on it. This latter fact would be an advantage if some interaction between the current Alumni System and the SPOIS were desired.

3.3 PRIME 300

In the summer of 1974 the Sloan School purchased a Prime 300 minicomputer. Located in the basement of Sloan, it is operated by the Center for Information Systems Research, a group consisting of Sloan faculty members. Currently, the usage charge is $2.50 per hour of connect time with no

additional charges for any other resource used. Each Sloan student can receive, upon request, a 20-hour time allotment for his own personal use. In addition, time on the system is available for research and course work.

The following Prime features make it the most desirable choice for the SPOIS:

(1) It is cheap.

(2) It supports interactive usage.

(3) It is on site at the Sloan School and under Sloan's control.

(4) It has adequate security features.

(5) It has adequate on-line storage capability.

The hardware and software features will now be discussed in more detail.

3.3.1 PRIME HARDWARE

A sketch of the hardware configuration is given in Exhibit 1 (Page 41). A summary is presented here:

CPU: Fairly fast and powerful for a minicomputer. Typical machine instruction execution time is one to three microseconds.

Memory: 64K 16-bit words. Access time of 750 nanoseconds.

Disks: Two disk drives with one permanent and one removable platter on each. Each platter holds 1.5 million words for a total capacity of 6 million words.

Terminals: All are hard-wired teletypes (TTY 33's). One operator's TTY and three for the users.

Data Sets: Two Western Electric 103 A data sets for dial up.

Printer: Serial Centronics printer; 132 print positions; 165 characters per second.

Card Reader: Reads 300 cards per minute. No punch.

The devices are run through a multi-line communications controller which can handle up to 16 terminals or data sets. It can also take another printer, another card reader, and additional disks.

3.3.2 PRIME SOFTWARE AND FILE SYSTEM

The operating system being used is DOS/VM, version 7. Through the use of demand paging each user has up to 64K words of virtual memory. The user environment is entirely time-sharing; there is no batch capability.

The system currently supports the following languages:

-Prime Macro Assebler

-Basic (interpreter)

-Fortran IV (with extensions)

-Dynamo

Gasp is being acquired.

There are also various utilities for sorting, output spooling, and a Real Time Operating System (RTOS). Logical

file management is hierarchic, similar to that used by Multics. Refer to Exhibit 2, page 42 for a diagram. There are basically two elements in this structure, files and file directories. Files contain programs and data. Directories list the addresses of files and/or other directories. At the root of all these is a directory called the Master File Directory (MFD). Below this are various User File Directories (UFD), such as user projects and system modules. Projects themselves list directories corresponding to individual users. Users typically have files in their directories, but they can also create sub-directories to a very deep level. Each directory has two passwords associated with it, one for the owner and one for non-owners. Each file has two access keys set by the directory owner for himself and for non-owners. There are three types of access to a file, read, write, and delete/truncate. The key specifies which combination of these types of access are legal. Currently, for example, for the HAAG UFD the owner has complete access to all files whereas non-owners have only read access to the programs but complete access to all data files.

Files and directories are stored in 448-word records of which the first 8 are inaccessible to the user. Directories take up one record, while files may occupy up to 32767 records. These records are chained together with forward and

backward pointers and to the user look like one long sequential address space.

4. RELATIONAL DATA-BASES

An overview of relational data-base concepts and definitions of terms is given in the introduction to the User's Guide to the SPOIS (Appendix, pages 61-66). Please read that section before continuing.

4.1 RDMS

The Relational Data Management System (RDMS) is a general purpose data management system written in PL/I and residing on Multics. The first section of the RDMS Reference Guide (Ref.5), entitled RDMS Design Principles, provides a good introduction to relational data-bases in general and RDMS specifically. RDMS significantly influenced the design of the SPOIS, which is why it is being discussed here. Another reason is that future improvements in the SPOIS would be likely to upgrade it in the direction of RDMS.

In RDMS refnos are completely invisible to the user. A set of data strategy modules, which the user can, if he wishes, specify, assign all refnos. In addition to the operations available in the SPOIS, RDMS has many more sophisticated commands. Examples of these are cartesian product, the composition of two relations, and the

intersection of relations. The mathematical background of RDMS is evidenced by the terminology. After this brief discussion of RDMS, let us now turn to a description of the actual design of the SPOIS.

4.2 DATA-BASE DESIGN OF THE SPOIS

A file, you will recall, has two columns, the first containing refnos, the second containing text. A relation , on the other hand, is stored as a matrix of refnos. In the SPOIS these entities are stored in a fixed format which is now described.

Files are stored as follows on the Prime's disk:

Word 0: FI (two ASCII characters)*
Word 1: Number of Records (NREC)
Word 2: Length of the text field in words (Q)
Words 3-38: One line of text description of the file
(Up to 72 characters)
Words 39- : Individual records.

The format of individual records is:

First Word: Refno
Next Q Words: Text

Total file length is 39+(Q+1)*NREC

Relations are stored as follows:

Word 0: RE (two ASCII characters)
Word 1: Number of records
Word 2: Number of fields (maximum of 30) (Q)

---

* Prime's 16-bit words can hold either a single integer or two 8-bit ASCII characters.

Words 3-38: One line of text description of the file
(Up to 72 characters)
Words 39-41: Name of field 1 (6 ASCII characters)
Words 42-44: Name of field 2 (6 ASCII characters)
.
.
.
(Number of such names=number of fields=Q)

Individual records begin in word 38+3*Q

Each record contains Q words containing one refno each. The sequence of the refnos within a record is the same as the sequence of the field names.

The total length of a relation is 39+(3*Q)+(NREC*Q)

The text in word 0 provides a quick, easy way of checking on the dataset type. This is necessary to make sure that the user has not attempted a command not meant for that dataset type.

The text in words 3-38 allows the user to describe the relation for his own mnemonic use. It also provides a heading for printout.

The SPOIS consists of a set of programs which deal with such datasets.

There is one unique dataset in the system, named DBINFO (Data-Base Information). This is structured as follows:

Word 0: Number of datasets in the system (NDBR)
Words 1-3: Name of first dataset (6 ASCII characters)
Word 4: Type of first dataset (2 ASCII characters)
Type can be: RE-relation
FI-file
PR-print file
DU-dump file

Words 5-7: Name of second dataset
Word 8: Type of second dataset
  •
  •
  •

The size of DBINFO is 1+(4*NDBR) words

DBINFO is used only for the user's sake. The actual Fortran programs do not need to reference it, since they determine the existence of datasets with a DOS subroutine called SEARCH.

5. THE SPOIS AT THE PROGRAMMING LEVEL

This chapter contains a description of the SPOIS at the programming level. For a description of how to use the system refer to the User's Guide in the Appendix (page 59).

5.1 THE FORTRAN PROGRAMS

There are three levels of programs in the SPOIS. The first of these is the Command Processor Program (CPP). This is the main program, the one the user is in immediately after he types R SPOIS. When the user is in this program he is in what is called command mode. The system has typed COMMAND-- and is awaiting a command. When the user types something in the CPP hunts through its list of legal commands and their abbreviations. If the command is found the CPP calls the appropriate subroutine and transfers any arguments the user has entered via an 18-word array in common.

A command of END causes a call to EXIT, a system subroutine which returns control to DOS. DOS will respond with OK, . A command of HELP causes a branch to the section of the CPP that prints out the list of legal commands. A listing of the CPP is presented in Exhibit 3, page 43.

There is generally one Fortran subroutine per SPOIS command. When control is passed to the subroutine it first checks to see if all arguments of the command have been properly specified. If so, it proceeds to perform the operation, which may involve asking the user some additional questions. If no or insufficient arguments have been specified it asks him to supply the missing information, then continues as above. If an incorrect argument has been specified the routine either asks the user to correct his error or it returns control to the CPP, depending on the specific routine. Sometimes, in fact, an error is enough to abort the CPP and return to DOS. This could happen if the program was attempting to read numeric data and encountered non-numeric input. Future versions of this system should catch those errors. One command subroutine, NEWFIL, is presented in Exhibit 4, page 46.

Finally, there are utility subroutines that are called by the command subroutines to perform certain common tasks. One command subroutine, ORDER, serves a dual function in that it is also called by other command subroutines such as NEWFIL. An example of a utility program, GETNUM, is given in Exhibit 5, page 51. This is the routine that handles the free format input.

The logic of the programs per se need not be discussed, since it is fairly straightforward. The rest of the programs

have not been included since that would serve no purpose save adding to the bulk of this report. Anyone who is interested in these programs can obtain copies from the SPOIS administrator. The next two pages list all the programs currently comprising the SPOIS together with a brief description thereof.

Fortran Programs in the SPOIS

CNAME-Implements the command CNAME, changes dataset names.

COPY-Implements command COPY, which copies a dataset.

CPP-Command Processor Program.

CREATE-Implements command CREATE,creates derivative datasets.

DELETE-Implements command DELETE, deletes datasets.

DUMP-Implements command DUMP,prints relations' refnos.

EDIT-Implements command EDIT, modifies a relation.

FIELDS-Implements command FIELDS,lists fields of a relation.

GETNUM-Utility for free format input.

INDBIN-Initializes DBINFO. Used only at system generation time.

INFO-Implements command INFO,lists names of all datasets.

KEYCOM-Contains mnemonic keys for SEARCH and PRWFIL.
Inserted in virtually all routines using $INSERT.

MEAN-Implements command MEAN,gets mean of a column.

MEDIAN-Implements command MEDIAN,gets median of a column.

MERGE-Implements command MERGE,merges two relations.

MFUTIL-Modify Utility called by EDIT and
MODFILE to add and delete records.

MODDBI-Modify DBINFO. Utility called by DELETE,
NEWFILE,NEWREL,CNAME,CREATE, and MERGE.

MODFIL-Implements MODFILE command,modifies files.

NEWFIL-Implements NEWFILE command, creates new files.

NEWREL-Implements NEWREL command,creates new relations

from terminal or card input.

ORDER-Implements command ORDER,called by other command subroutines, puts datasets in order.

PRINT-Implements PRINT command,prints dataset on terminal or disk for spooling.

RECORD-Implements command RECORDS,supplies number of records in a dataset.

SUM-Implements command SUM,gets sum of a column.

In keeping with the design goal of good documentation, several standards have been used. These are listed here:

(1) Use of many comments, both in-line and normal Fortran comment lines.

(2) Whenever possible, the same variable name is used throughout all programs for the same uses. Some of the common ones are:

RELNAM(3)-Name of a relation
FILNAM(3)-Name of a file
FRNAME(3)-Name of a file or relation
FUNIT,FUNIT1,FUNIT2-DOS logical file unit numbers
COMMAN(18)-Holds one line of SPOIS commands
NREC-Number of records in a relation or file
Q or LENGTH-Record size (in words)
DESCRI(36)-Holds one-line file or relation description
BUFF(.)-Buffer for input/output
PBUFF-Pointer to BUFF
PRNTFL(3)-Name of print or dump file
IOUNIT-Fortran unit number for I/O
I,J,IT,QP-temporary variables for looping,etc.

(3) All statement numbers appear in sequential order.

(4) Extensive use of CONTINUE statements.

The Fortran programs do call some system subroutines that should be explained, since they are unique to the Prime. This is done in the next section.

5.2 SYSTEM SUBROUTINES

There are several fairly powerful system subroutines without which the implementation of the SPOIS would have been significantly more difficult. These are discussed here.

SEARCH: This routine performs all the details involven in opening, closing, deleting,truncating, and rewinding datasets. Opening a dataset involves creating a memory buffer into which the dataset (or portions of it) is to be read. Upon opening a dataset one specifies whether it will be used for reading, writing, or both. It is even possible to open one's UFD for reading and writing, and this is done in the CNAME command.

PRWFIL: This one is used to position, read, and write data. Positioning can be to an absolute address or relative to the current position and can be done before or after reading or writing. The number of words to be transferred to or from an array in the program is also specified.

READ and WRITE: These do the same as PRWFIL, but with no explicit positioning capability. In fact, they call PRWFIL to perform the operation.

GETERR: After an error has occurred in the use of a system

subroutine this function can be used to retrieve the system Error Vector, which contains an error code and useful status information.

COMANL: This routine waits for the user to input a line in standard format. The format is zero to three commands and zero to nine octal integers. The line is then stored in an 18-word system array containg first the three commands (3 words of 6 ASCII characters each), then the nine numbers.

CMREAD: Retrives the 18-word system command line and stores it in an array in the program. Calling COMANL followed by CMREAD is the mechanism used in the CPP and all the command subroutines to obtain commands from the user. An 18-word array called COMMAN exists in common and is used by the CPP to pass command lines to the subroutines.

6. SAMPLE TERMINAL SESSION

Exhibit 6 shows a sample terminal session. This session shows but a few of the SPOIS commands, but should be enough to give the flavor of what using the system is like. Comments appear on the terminal output, so there is no need to say anything more here.

7. EVALUATION

At this point an evaluation of the system, in terms of the original design goals, should be undertaken. Let us review these point by point:

(1) Give the Placement Office what it wants.

The Office now has a system that can do many of the things that Leslie Hruby wanted. The system allows extensive data manipulation and printouts of the resultant relations. It does not yet have the capability to do crossbreaks. It was decided that implementing the basic functions had a higher priority than specialized reports. Each new report would be one extra program, which somebody else could easily write.

(2) User isolated from the operating system.

This goal has not yet been completely met, since the user must leave the SPOIS to get control of the card reader or spool output. Also, if the system is expecting fixed format numeric input and gets any non-numeric input the user will be thrown out of the SPOIS. Finally, if a user inputs non-octal numbers when issuing a command he will likewise be evicted.

The problems with input can be solved straighforwardly

with additional user-written Fortran routines. The problem of having to leave the system when using the devices is more difficult. It may never be solved for the card reader, but it is possible that a systems program could be created or modified for spooling from a Fortran program. For the present, however, these problems must be lived with.

(3) Self-documenting programs.

About 50% of all lines of the programs are comments. This should be quite sufficient for documentation purposes.

(4) Modular system.

It is probably fair to say that this goal has been met. It is certainly quite easy to add or delete commands, since each command corresponds to one program.

(5) Easy to use, interactive, patient system.

This goal, too, has been met. The variety of ways in which commands can be input (without any or all arguments, abbreviations) the on-line HELP commands, and the constant interaction make this a very usable system, even for somebody who knows very little about computers.

(6) Cost-effective system.

Although this goal has not been tested, it is very likely

that it will be met. Updating the data-base for new students should take very little keypunching. Once the data is in a file or relation it will be easy to shape it to one's needs. A more time-consuming task would be to update alumni records, but that is not an issue yet.

(7) Help Placement Office save time, do new things.
The SPOIS certainly allows the Office to do new things. Whether it will actually save time is another question which will depend on the extent of use, familiarity with the system, and future improvements on it.

8. THE FUTURE OF THE SPOIS

8.1 MAINTENANCE

This factor is still open, since it depends heavily on how and how much the system is used.

First of all, it is recommended that once the system is in a semi-permanent form all the source and object programs be moved to a removable disk pack or a diskette. This would leave on-line only the system datasets and the SPOIS memory image load module itself. This move would help relieve the load on the currently overloaded disk (typically 95% full, sometimes 100%, in which case no new datasets can be created).

As far as updating the data-base is concerned it should be possible to hire students to do this work. This would not involve much money and could come out of the Placement Office budget or general administrative funds.

8.2 EXTENSIONS AND MODIFICATIONS

The system as it now stands offers much opportunity for future work. While it stands on a sound foundation there is much that can be done to improve it both for the Placement

Office and as a general purpose data management system. Here are a few suggestions:

(1) Replace the COMANL subroutine with a user written routine that accepts more commands (say six) and decimal numbers. This would make it easier to specify several of the commands.

(2) Allow larger refnos than the current limit of 32767, which is the largest number in a 16-bit word.

(3) Add additional report programs for the Placement Office and implement some of the more sophisticated ROMS commands.

(4) Write interfaces that would permit card reader assignment and spooling to be performed form within a Fortran program. This would make a good research or thesis topic.

(5) Make the code more efficient; put more variables in common.

(6) Add strategey routines to make the refnos invisible to the user.

(7) Allow all input to be free format. Check all input data to prevent a user from ever being thrown out of the SPOIS.

One final warning: If this system is made too much larger it will exceed certain Prime loader limits. There are ways around this, but they can also lead to problems. It may be necessary to break up the system into subsystems should this problem occur.

REFERENCES

(1) Baggeroer, William L. and Fox, John M., Student Information System, Sloan School of Management, Preliminary Analysis and Design, Term Project, 15.565, 17 December 1974

(2) Sami, Fehmi A., A Computer Based Student Information System - Sloan School of Management, S.B. Thesis for the Department of Electrical Engineering and Computer Science, 9 May 1975

(3) Krupp, Margaret, A Placement Office Student Reservation System, S.B. thesis for the Department of Electrical Engineering and Computer Science, 9 May 1975

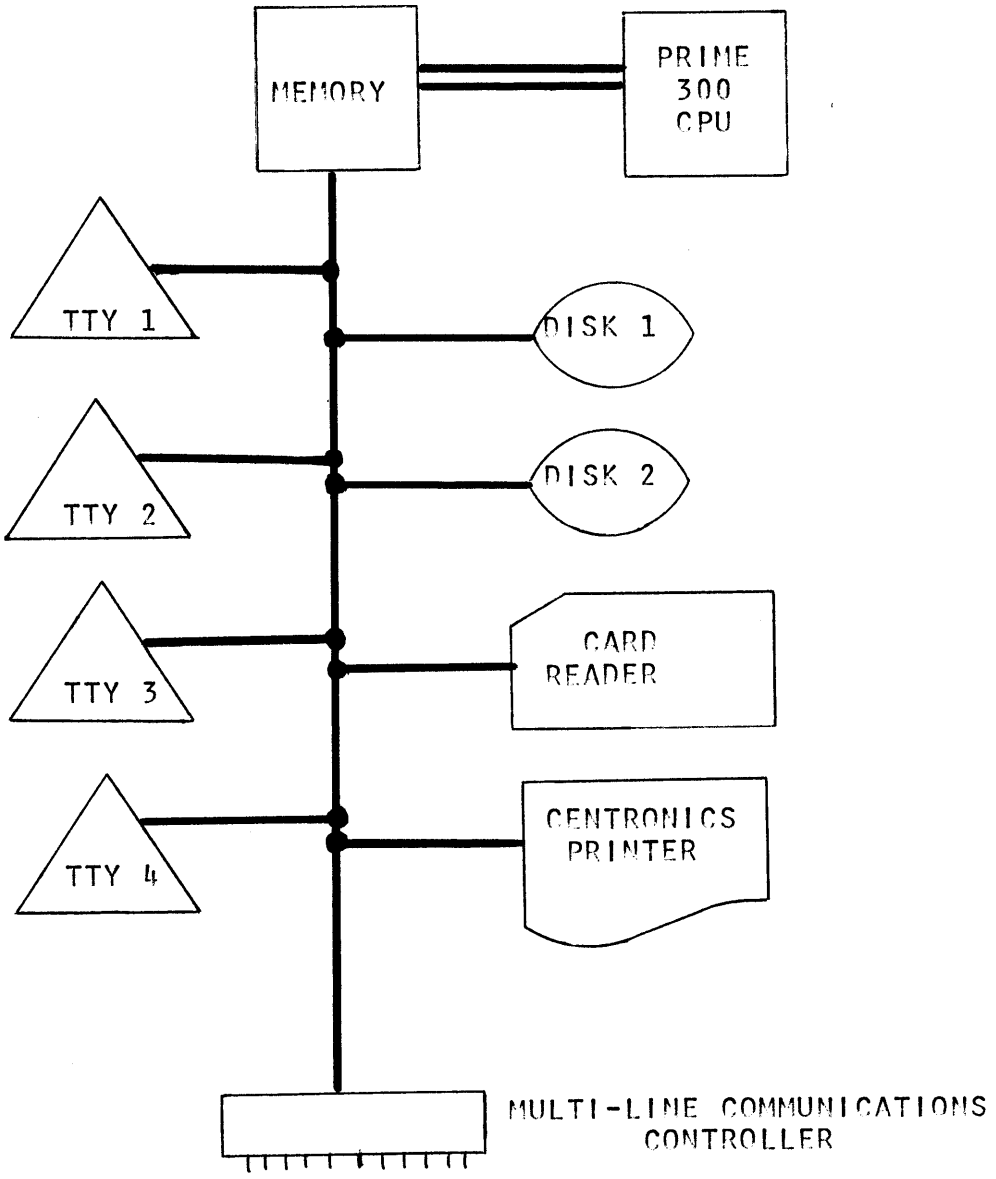
(4) Honeywell, Multics Users' Guide, November 1973

(5) RDMS Reference Guide

(6) PRIME Computer, Inc., Disk Operating System and Virtual Memory System Operating System Reference Guide, 1 August 1974

(7) PRIME Computer, Inc., Fortran IV Language Reference Manual, Revision B, July 1974

(8) MIT Project MAC, Multics Programmers' Manual, Revision 15, 30 November 1973



Note: TTY 1 is the operator's terminal.
TTY 2-4 are the user terminals.

Exhibit 1: PRIME 300 Hardware Configuration

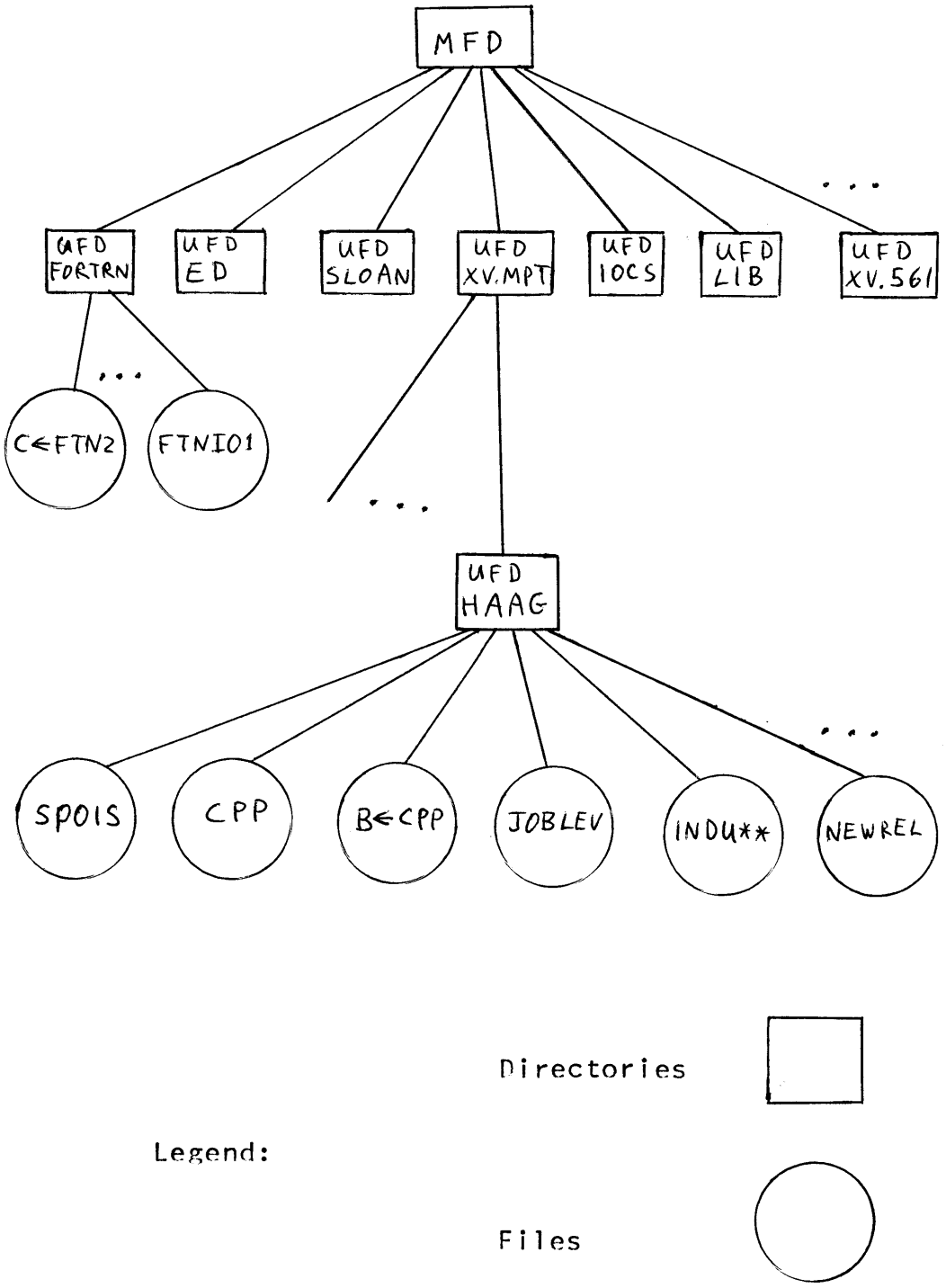


Exhibit 2: PRIME Logical File Structure

EXHIBIT 3: The CPP

```
C                    CPP
C  COPYRIGHT (C) 1975 PETER G.  HAAG
C
C      THIS IS THE COMMAND PROCESSOR PROGRAM(CPP) FOR THE
C  SLOAN PLACEMENT OFFICE INFORMATION SYSTEM(SPOIS).   FOR A DE-
C  SCRIPTION OF HOW TO USE THE SYSTEM REFER TO THE SPOIS USER'S GUIDE.
C  IF YOU RUN INTO TROUBLE RUNNING THIS PROGRAM YOU CAN GET
C  ASSISTANCE AT ANY TIME BY TYPING HELP.
C      THE CPP IS THE MASTER FORTRAN PROGRAM IN THE SPOIS.   IT IS
C  THE INTERFACE BETWEEN THE USER AND THE ACTUAL FORTRAN
C  PROGRAMS.   THE CPP CALLS OTHER PROGRAMS IN RESPONSE TO COMMANDS
C  ISSUED BY THE USER.   IT IS THESE PROGRAMS WHICH MANIPULATE
C  THE ACTUAL FILES.
C
C
       INTEGER
       COMMON COMMAN(18)  /* HOLDS ONE LINE OF COMMANDS */
       DIMENSION  TABLE1(3,20),  /* HOLDS FULL COMMAND NAMES  */
      2          TABLE2(20)     /* ABBREVIATED COMMAND NAMES    */
C              COMNUM          /* NUMBER OF COMMANDS AVAILABLE */
C
C TO ADD OR DELETE COMMANDS YOU MUST:
C (1)CHANGE THE DIMENSIONS OF TABLE1 AND TABLE2
C (2)CHANGE THE VALUE OF COMNUM IN ITS DATA STATEMENT
C (3)MODIFY TABLE1 & TABLE2 IN THEIR DATA STATEMENTS
C (4)ADD OR DELETE THE IF STATEMENTS AFTER STATEMENT 200
C (5)MODIFY FORMAT 8005 APPROPRIATELY
C
       DATA  TABLE2/'HE','CR','ME','PR','NF',
      1  'OR','ED','MF','FI','RE',
      2  'EN','DE','IN','NR','CN','CO','DU','MN','MD',
      3  'SU'/
C
       DATA  TABLE1/'HELP  ','CREATE','MERGE ',
      1  'PRINT ','NEWFIL','ORDER ','EDIT  ',
      2  'MODFIL','FIELDS','RECORD',
      3  'END   ','DELETE','INFO  ','NEWREL','CNAME ','COPY  ',
      4  'DUMP  ','MEAN  ','MEDIAN','SUM   '/
C
       DATA COMNUM/20/
C
C*****CODE BEGINS HERE******
C
       I=:207207
       WRITE(1,5) I,I,I,I,I
5      FORMAT(5A2)
       WRITE(1,10)
10     FORMAT(///'WELCOME TO THE SPOIS.'/'IF AT ANY TIME YOU'
      1,' REQUIRE ASSISTANCE TYPE HELP.')
20     WRITE(1,30)
30     FORMAT('COMMAND--')
       CALL COMANL
       CALL CMREAD(COMMAN)
```

EXHIBIT 3 (cont.)

```
      DO 60 I=1,COMNUM
      IF (COMMAN(1).EQ.TABLE2(I)) GOTO 200
      IF (COMMAN(1).EQ.TABLE1(1,I).AND.
     1    COMMAN(2).EQ.TABLE1(2,I).AND.
     2    COMMAN(3).EQ.TABLE1(3,I)) GOTO 200
60    CONTINUE
65    WRITE(1,70)
70    FORMAT('ILLEGAL INPUT,TRY AGAIN--')
      GOTO 20
C
C
200   CONTINUE
      IF(I.EQ.1) GOTO 8000  /*GOTO HELP SECTION*/
      IF(I.EQ.2) CALL CREATE
      IF(I.EQ.3) CALL MERGE
      IF(I.EQ.4) CALL PRINT
      IF(I.EQ.5) CALL NEWFIL
      IF(I.EQ.6) CALL ORDER
      IF(I.EQ.7) CALL EDIT
      IF(I.EQ.8) CALL MODFIL
      IF(I.EQ.9) CALL FIELDS
      IF(I.EQ.10) CALL RECORD
      IF(I.EQ.11) CALL EXIT
      IF(I.EQ.12) CALL DELETE
      IF(I.EQ.13) CALL INFO
      IF(I.EQ.14) CALL NEWREL
      IF(I.EQ.15) CALL CNAME
      IF(I.EQ.16) CALL COPY
      IF(I.EQ.17) CALL DUMP
      IF(I.EQ.18) CALL MEAN
      IF(I.EQ.19) CALL MEDIAN
      IF(I.EQ.20) CALL SUM
      GO TO 20
C
C
8000  CONTINUE
      WRITE(1,8005)
8005  FORMAT(//'THE FOLLOWING ARE THE LEGAL '
     1,'COMMANDS AND THEIR ABBREVIATIONS:'//
     3 'CNAME(CN)----CHANGE A DATA-BASE NAME'/
     X 'COPY(CO)-----MAKE A COPY OF A DATA-BASE'/
     3 'CREATE(CR)---CREATE A NEW RELATION FROM OLD ONES'/
     4 'DELETE(DE)---DELETE AN EXISTING FILE OR RELATION'/
     X 'DUMP(DU)-----PRINT OUT THE REFNOS OF A RELATION'/
     X 'EDIT(ED)-----MODIFY A RELATION,DISPLAY A RECORD'/
     5 'END(EN)------GETS YOU OUT OF THE SPOIS'/
     6 'FIELDS(FI)---LIST THE FIELDS OF A RELATION'/
     7 'HELP(HE)-----GETS YOU THIS LIST HERE'/
     X 'INFO(IN)-----LIST RELATIONS,FILES & PRINT FILES'/
     X 'MEAN(MN)-----CALCULATE MEAN OF A COLUMN OF A RELATION'/
     X 'MEDIAN(MD)---CALCULATE MEDIAN OF A COLUMN OF A RELATION'/
     8 'MERGE(ME)----MERGE 2 EXISTING RELATIONS'/
     9 'MODFIL(MF)---MODIFY AN EXISTING FILE'/
```

EXHIBIT 3 (cont.)

```
2 'NEWFILE(NF)--CREATE A NEW FILE'/
3 'NEWREL(NR)---CREATE A NEW RELATION FROM CARDS OR INPUT'/
4 'ORDER(OR)----PUT A FILE OR RELATION IN ASCENDING ORDER'/
5 'PRINT(PR)----PRINT A FILE OR RELATION'/
X 'RECORDS(RE)--GIVES NUMBER OF RECORDS IN A FILE OR RELATION'/
X 'SUM(SU)------CALCULATE SUM OF A COLUMN OF A RELATION'//
6 'TO LEARN MORE ABOUT INDIVIDUAL COMMANDS TYPE THE'/
7 'COMMAND FOLLOWED BY HELP'//)
 GOTO 20
 CALL EXIT
 END
```

Exhibit 4: Newfil

```
C               NEWFILE
C  COPYRIGHT (C) 1975 PETER G.  HAAG
C
C  WITH THIS PROGRAM THE USER CREATES NEW FILES
C  THE SYNTAX IS:
C                                  ( HELP )
C  (NEWFILE)    (FILENAME)    (TERMINAL)
C  (  NF    )    ( HELP    )    ( CARDS  )
C                                ( BOTH   )
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
      SUBROUTINE NEWFIL
C
      INTEGER
      COMMON  COMMAN(18)    /* HOLDS A LINE OF COMMANDS */
      INTEGER FILNAM(3),    /* HOLDS FILE NAME          */
     2        FORM(7),      /* HOLDS FORMAT SPEC        */
     3        ERROR,        /* HOLDS ERROR CODE         */
     4        Q,            /* # OF WORDS IN ALPHA FIELD */
     5        Q1,Q2,        /* 1ST & 2ND DIGITS OF Q    */
     6        DESCRI(36),   /* HOLDS 1-LINE DESCRIPTION  */
     7        BUFF(40),     /* OUTPUT BUFFER            */
     8        PBUFF,        /* POINTER TO BUFF          */
     9        PFILE(2),     /* POINTER TO FILE LOC.     */
     1        FUNIT,
     2        STATUS(3)     /* HOLDS STATUS VECTOR FOR T$CMPC */
C
$INSERT KEYCOM
C
      DATA FORM/'( I5,1X,   A2)'/
      DATA PFILE/2*0/
      DATA BUFF/40*0/
      DATA FUNIT/1/
C
C     BEGINNING OF CODE
C
      PBUFF=LOC(BUFF)
      DO 100 J=1,15
100   COMMAN(J)=COMMAN(J+3)  /* SHIFT SECOND ARG.  TO FIRST */
      IF (COMMAN(1).NE.'  ') GOTO 300
      WRITE(1,200)
200   FORMAT('NAME OF NEW FILE AND INPUT MODE--')
      CALL COMANL
      CALL CMREAD(COMMAN)
300   IF(COMMAN(1).EQ.'HE'.AND.COMMAN(2).
     1   EQ.'LP') GOTO 8000
C
500   CONTINUE
      DO 600 J=1,3      /* READ FILENAME INTO ARRAY FILNAM */
600   FILNAM(J)=COMMAN(J)
C  NOW CHECK TO MAKE SURE FILE DOESN'T ALREADY EXIST
      CALL SEARCH(OPNRED,FILNAM,FUNIT,$800)
      CALL SEARCH (CLOSE,FILNAM,FUNIT,0)
```

Exhibit 4 (cont.)

```
      WRITE (1,700)
700   FORMAT('FILE ALREADY EXISTS.  TO MODIFY AN '
     1    ,'EXISTING FILE USED MODFILE COMMAND')
      RETURN
C
800   CALL GETERR(ERROR,1)
      IF (ERROR.EQ.'SH') GOTO 1000 /* SH=FILE NOT FOUND */
      WRITE (1,900) FILNAM
900   FORMAT('UNREVOVERABLE ERROR IN SEARCHING FOR ',3A2)
      RETURN
C
C  IF WE GET HERE, FILE WASN'T FOUND, SO WE'RE OK
C
1000  CONTINUE
C  NOW SHIFT INPUT MODE ARGUMENT INTO FIRST POSITION
      DO 1100 J=1,12
1100  COMMAN(J)=COMMAN(J+3)
1110  IF (COMMAN(1).EQ.'HE') GOTO 8000
      IF (COMMAN(1).EQ.'TE') GOTO 2000
      IF (COMMAN(1).EQ.'BO') GOTO 3000
      IF (COMMAN(1).EQ.'CA') GOTO 3000
      IF (COMMAN(1).EQ.'  ') GOTO 1140
      WRITE (1,1120)
1120  FORMAT ('ILLEGAL INPUT MODE. SPECIFY TERMINAL OR CARDS ',
     1       'OR BOTH')
1140  WRITE (1,1160)
1160  FORMAT ('WHAT IS THE INPUT MODE?')
      CALL COMANL
      CALL CMREAD(COMMAN)
      GOTO 1110
C
C  THIS SECTION HANDLES THE 'TERMINAL' AND 'BOTH' CASES
C
2000  CONTINUE
2020  WRITE (1,2030)
2030  FORMAT('HOW MANY CHARACTERS WIDE WILL THE ALPHABETIC'
     1       ,' FIELD BE--')
      READ (1,2040) J
2040  FORMAT (I2)
      IF (1.LT.J.AND.J.LT.66) GOTO 2055
      WRITE (1,2050)
2050  FORMAT('# OF CHARACTERS MUST BE BETWEEN 1 AND 66')
      GOTO 2020
2055  Q=(J+1)/2
      Q1=Q/10
      Q2=Q-10*Q1
      FORM(5)=LS(:260+Q1,8)+:260+Q2
C  WE NOW HAVE THE FOLLOWING VARIABLES:
C    Q=# OF WORDS IN ALPHA FIELD
C    Q1= FIRST DIGIT OF Q
C    Q2= SECOND DIGIT OF Q
C    FORM(5) CONTAINS ASCII VERSION OF Q1,Q2
```

Exhibit 4 (cont.)

```
      WRITE (1,2060)
2060  FORMAT('PLEASE TYPE A ONE LINE(UP TO 72 CHARACTERS)',
     1       'DESCRIPTION OF THE FILE')
      READ (1,2070) DESCRI
2070  FORMAT (36A2)
      IF (COMMAN(1).EQ.'BO') GOTO 4550
      WRITE (1,2080)
2080  FORMAT('NOW BEGIN ENTERING LINES IN FIXED FORMAT'/
     1       'DO YOU WANT AN EXPLANATION--')
2085  READ (1,2090)I
2090  FORMAT (1A2)
      IF (I.EQ.'NO') GOTO 6130
      IF (I.EQ.'YE') GOTO 2110
      WRITE(1,2100)
2100  FORMAT ('ANSWER YES OR NO PLEASE--')
      GOTO 2085
2110  WRITE (1,2120)
2120  FORMAT(/'FORMAT IS: AN INTEGER UP TO 32767, THEN,'/
     1  'BEGINNING IN COLUMN 7, YOUR TEXT.  TO END INPUT,'/
     2  'ENTER A ZERO AS THE FIRST AND ONLY CHARACTER OF THE LINE.'/)
      IOUNIT=1
      GOTO 6130
C
C
3000  CONTINUE
3090  WRITE (1,3100)
3100  FORMAT('HAVE YOU ASSIGNED CR1? IF YOU ANSWER NO I'/
     1   'WILL TAKE YOU OUT OF THE SPOIS SO YOU CAN DO SO')
      READ (1,3200) J
3200  FORMAT(1A2)
      IF (J.EQ.'NO') CALL EXIT
      IF (J.EQ.'HE') GOTO 8000
      IF (J.EQ.'YE') GOTO 3300
      WRITE (1,3250)
3250  FORMAT('PLEASE ANSWER YES, NO , OR HELP')
      GOTO 3090
3300  IF (COMMAN(1).EQ.'BO') GOTO 2000
C
C  THIS SECTION HANDLES THE HEADER IN THE 'CA' CASE (CARDS)
C
4000  CONTINUE
C  WE USE T$CPMC BECAUSE READ(3,...) DOESN'T WORK
C  ARGUMENTS ARE: 1=>IGNORED
C  PBUFF=>POINTER TO BUFFER INTO WHICH INPUT IS READ
C  40=># OF WORDS TO READ FROM CURRENT CARD
C  :40000=>READ CARD IN ASCII
C  STATUS=>STATUS VECTOR FOR ERROR CODE
      CALL T$CMPC(1,PBUFF,40,:40000,STATUS)
      DECODE(80,4100,BUFF) BUFF(1),J1,(DESCRI(J),J=1,36)
4100  FORMAT(1A2,T4,I2,T7,36A2)
      BUFF(3)=J1
      IF (BUFF(1).EQ.'FI') GOTO 4200
      WRITE (1,4150)
```

Exhibit 4 (cont.)

```
4150  FORMAT('FIRST CARD DOES NOT SPECIFY FI IN COLUMNS 1-2')
      RETURN
4200  IF (BUFF(3).GE.1.AND.BUFF(3).LE.66) GOTO 4500
      WRITE (1,4250)
4250  FORMAT('INCORRECT CHARACTER COUNT SPECIFIED IN COLUMNS 4-5'/
     1    'RANGE IS BETWEEN 1 AND 66 INCLUSIVE')
      RETURN
C
C
4500  CONTINUE
C  DETERMINE INPUT FORMAT:
      Q=(BUFF(3)+1)/2
      Q1=Q/10
      Q2=Q-10*Q1
      FORM(5)=LS(:260+Q1,8)+:260+Q2
4550  IOUNIT=3
6130  CONTINUE
C
C WE NOW OPEN FILNAM FOR READING AND WRITING
      CALL SEARCH (OPNWRT,FILNAM,FUNIT,0)
      CALL PRWFIL(PWRITE,FUNIT,PBUFF,39,0,0)
C  THIS PUTS DUMMY DATA IN WORDS 0-38.  WE DON'T REALLY
C CARE WHAT IS IN BUFF SINCE IT WILL BE OVERWRITTEN LATER
C ANYWAY.  THIS ALSO POSITIONS FILE POINTER TO FIRST DATA
C RECORD (WHICH WE HAD TO DO ANYWAY).
      ITEMP=Q+1
      DO 6150 ICOUNT=1,10000
      IF (COMMAN(1).EQ.'TE') GOTO 6135
      CALL T$CMPC(1,PBUFF,40,:40000,STATUS)
      DECODE (80,FORM,BUFF) (BUFF(J),J=1,ITEMP)
      GOTO 6140
6135  READ (1,FORM) (BUFF(J),J=1,ITEMP)
6140  IF (BUFF(1).EQ.0) GOTO 6200
      CALL PRWFIL(PWRITE,FUNIT,PBUFF,Q+1,0,0)
6150  CONTINUE
6200  CONTINUE
      PFILE(1)=0
      PFILE(2)=0
      BUFF(1)='FI'
      BUFF(2)=ICOUNT-1
      BUFF(3)=0
      DO 6250 J=1,36
6250  BUFF(J+3)=DESCRI(J)
      CALL PRWFIL(PWRITE+PREABS,FUNIT,PBUFF,39,PFILE,0)
      CALL SEARCH(CLOSE,FILNAM,FUNIT,0)
      CALL MODDBI(FILNAM,'FI','AD') /* UPDATE DIRECTORY */
      WRITE(1,6400)
6400  FORMAT('DONE.  THE FILE WILL NOW BE ORDERED NUMERICALLY')
      DO 6500 J=1,3
6500  COMMAN(J+3)=FILNAM(J)
      CALL ORDER(COMMAN)
      RETURN
C
```

```
C
C
8000  CONTINUE
      WRITE (1,8100)
8100  FORMAT('  COMMAND NEWFILE OR NF'/
     1 'THE SYNTAX IS:'/
     2 '                          (TE)'/
     3 '(NEWFILE)  (FILENAME) (CA)'/
     4 '  (NF)         (HELP)  (BO)'/
     5 '                          (HE)'//
     6 '  THIS COMMAND CREATES A NEW FILE WITH THE GIVEN FILENAME'/
     7 'INPUT CAN BE ACCEPTED FROM THE TERMINAL(TE),FROM CARDS(CA)'/
     8 'OR BOTH(BO).  FOR TERMINAL INPUT YOU WILL BE ASKED FOR ALL'/
     9 'INFO NEEDED.  FRO CARD SPECIFY THE FILENAME AND THE SYSTEM'/
     1 'DOES THE REST.  FOR BOTH YOU SUPPLY THE RECORD LENGTH AND'/
     2 'FILE DESCRIPTION FROM THE TERMINAL AND YOUR DATA IS ON'/
     3 'CARDS.  FOR CA AND BO YOU MUST 1)LEAVE THE SPOIS.  2)TYPE'/
     4 'AS CR1.  3)REENTER THE SPOIS.  4)ENTER THE NEWFILE COMMAND'/
     5 'AND APPROPRIATE SUBCOMMANDS.  5)LEAVE THE SPOIS.  6)TYPE'/
     6 'U CR1.'/)
      RETURN
      END
```

```
      SUBROUTINE GETNUM(N)
C
C  COPYRIGHT (C) 1975 PETER G.  HAAG
C
C  THIS ROUTINE READS AN INTEGER FROM THE TERMINAL.  THE SYMBOL
C  #(POUND SIGN) TYPED IMMEDIATELY AFTER THE NUMBER CANCELS IT.
C  ANY OTHER CHARACTER IS A DELIMITER.  THE ROUTINE ALSO
C  CHECKS TO SEE THAT THE PRIME MAXIMUM INTEGER OF 32767
C  HAS NOT BEEN EXCEEDED.
C  NOTE: # IS OCTAL 243.  DIGITS 0 THRU 9 ARE OCTAL 260 THRU
C  271 IN ASCII.
C
      INTEGER CHAR
C
100   N=0
      DO 200 J=1,73
      CALL T1IN(CHAR)
      CHAR=CHAR-:260
      IF(CHAR.GE.0.AND.CHAR.LE.9) GOTO 250
C  THIS FINDS THE FIRST DIGIT
200   CONTINUE
250   N=CHAR
300   DO 500 J=1,73
      CALL T1IN(CHAR) /* READS ONE CHARACTER FROM THE TTY
      CHAR=CHAR-:260
      IF(CHAR.EQ.-13) GOTO 100 /* CANCEL THIS NUMBER
      IF(CHAR.LT.0.OR.CHAR.GT.9) GOTO 600
      IF(N.LT.3276) GOTO 400
      IF (N.GE.3277) GOTO 800
      IF(CHAR.GT.7) GOTO 800
400   N=10*N+CHAR
500   CONTINUE
600   RETURN
800   WRITE(1,810)
810   FORMAT('MAXIMUM NUMBER IS 32767.  YOURS WILL BE IGNORED',
     1 '.  REENTER IT')
      GOTO 100
      END
```

Exhibit 5: GETNUM

```
R SPOIS <------This gets you into the SPOIS
GO



WELCOME TO THE SPOIS.          <------Introductory message
IF AT ANY TIME YOU REQUIRE ASSISTANCE TYPE HELP.
COMMAND-- <------System now ready for your first command
INFO <------Let's list all datasets in the SPOIS

SPOIS INFO:
INDUST-FI-LIST OF INDUSTRIES
JOBLEV-FI-JOB LEVEL OF RESPONSIBILITY
GEOGRA-FI-WORLDWIDE GEOGRAPHIC REFERENCE CODES
FUNCT -FI-FUNCTIONAL JOB AREAS -- 2 MAY 1975
CONCEN-FI-CONCENTRATION CODES -- 2 MAY 1975
IHOPE -RE-RELATION IS CALLED IHOPE BECAUSE I HOPE THIS WORKS
NAMES -FI-TEST FILE WITH STUDENT NAMES
IHOP**-PR-


COMMAND--
PR CONCEN TE<-Print the file CONCEN at the terminal


          FILE CONCEN

CONCENTRATION CODES -- 2 MAY 1975

    1 MIS
    2 ORGAN. STUDIES
    3 FINANCE
    4 OPERATIONS
    5 INTERNAT. MGT.
    6 MARKETING
    7 OPERATIONS RES.
    8 MPC
    9 SYSTEMS DYN.
   10 INDUSTRIAL REL.
   20 OTHER


COMMAND--
HELP<------Let's get a list of all legal commands
```

Exhibit 6: Sample Terminal Session

```
THE FOLLOWING ARE THE LEGAL COMMANDS AND THEIR ABBREVIATIONS:

CNAME(CN)----CHANGE A DATASET NAME
COPY(CO)-----MAKE A COPY OF A DATASET
CREATE(CR)---CREATE A NEW RELATION FROM OLD ONES
DELETE(DE)---DELETE AN EXISTING FILE OR RELATION
DUMP(DU)-----PRINT OUT THE REFNOS OF A RELATION
EDIT(ED)-----MODIFY A RELATION,DISPLAY A RECORD
END(EN)------GETS YOU OUT OF THE SPOIS
FIELDS(FI)---LIST THE FIELDS OF A RELATION
HELP(HE)-----GETS YOU THIS LIST HERE
INFO(IN)-----LIST SOME OR ALL SPOIS DATASETS
MEAN(MN)-----CALCULATE MEAN OF A COLUMN OF A RELATION
MEDIAN(MD)---CALCULATE MEDIAN OF A COLUMN OF A RELATION
MERGE(ME)----MERGE 2 EXISTING RELATIONS
MODFIL(MF)---MODIFY AN EXISTING FILE
NEWFILE(NF)--CREATE A NEW FILE
NEWREL(NR)---CREATE A NEW RELATION FROM CARDS OR INPUT
ORDER(OR)----PUT A RELATION IN ASCENDING ORDER
PRINT(PR)----PRINT A FILE OR RELATION
RECORDS(RE)--GIVES NUMBER OF RECORDS IN A FILE OR RELATION
SUM(SU)------CALCULATE SUM OF A COLUMN OF A RELATION

TO LEARN MORE ABOUT INDIVIDUAL COMMANDS TYPE THE
COMMAND FOLLOWED BY HELP

COMMAND--                    <-- Let's learn more about NF command
NF HELP
  COMMAND NEWFILE OR NF
THE SYNTAX IS:
                          (TE)
(NEWFILE)  (FILENAME)     (CA)
  (NF)        (HELP)      (BO)
                          (HE)

  THIS COMMAND CREATES A NEW FILE WITH THE GIVEN FILENAME
INPUT CAN BE ACCEPTED FROM THE TERMINAL(TE),FROM CARDS(CA)
OR BOTH(BO). FOR TERMINAL INPUT YOU WILL BE ASKED FOR ALL
INFO NEEDED. FRO CARD SPECIFY THE FILENAME AND THE SYSTEM
DOES THE REST. FOR BOTH YOU SUPPLY THE RECORD LENGTH AND
FILE DESCRIPTION FROM THE TERMINAL AND YOUR DATA IS ON
CARDS. FOR CA AND BO YOU MUST 1)LEAVE THE SPOIS. 2)TYPE
AS CR1. 3)REENTER THE SPOIS. 4)ENTER THE NEWFILE COMMAND
AND APPROPRIATE SUBCOMMANDS. 5)LEAVE THE SPOIS. 6)TYPE
U CR1.
COMMAND--
```

Exhibit 6 (cont.)

```
NF <------We now issue NF command without arguments. The system
NAME OF NEW FILE AND INPUT MODE--   prompts us for them.
STUDEN
WHAT IS THE INPUT MODE?          Having forgotten the syntax, we
HE <-----------------------------abort command and ask for help
  COMMAND NEWFILE OR NF
THE SYNTAX IS:
                          (TE)
(NEWFILE)   (FILENAME)    (CA)
  (NF)         (HELP)     (BO)
                          (HE)

  THIS COMMAND CREATES A NEW FILE WITH THE GIVEN FILENAME
INPUT CAN BE ACCEPTED FROM THE TERMINAL(TE),FROM CARDS(CA)
OR BOTH(BO). FOR TERMINAL INPUT YOU WILL BE ASKED FOR ALL
INFO NEEDED. FRO CARD SPECIFY THE FILENAME AND THE SYSTEM
DOES THE REST. FOR BOTH YOU SUPPLY THE RECORD LENGTH AND
FILE DESCRIPTION FROM THE TERMINAL AND YOUR DATA IS ON
CARDS. FOR CA AND BO YOU MUST 1)LEAVE THE SPOIS. 2)TYPE
AS CR1. 3)REENTER THE SPOIS. 4)ENTER THE NEWFILE COMMAND
AND APPROPRIATE SUBCOMMANDS. 5)LEAVE THE SPOIS. 6)TYPE
U CR1.
COMMAND--
NF STUDEN TE <------Now we're ready to create a new file
HOW MANY CHARACTERS WIDE WILL THE ALPHABETIC FIELD BE--
16
PLEASE TYPE A ONE LINE(UP TO 72 CHARACTERS)DESCRIPTION OF THE FILE
STUDENT NAMES--SAMPLE TERMINAL SESSION
NOW BEGIN ENTERING LINES IN FIXED FORMAT
DO YOU WANT AN EXPLANATION--
YES

FORMAT IS: AN INTEGER UP TO 32767, THEN,
BEGINNING IN COLUMN 7, YOUR TEXT. TO END INPUT,
ENTER A ZERO AS THE FIRST AND ONLY CHARACTER OF THE LINE.
10     ALPHA, JOHN <-------Here we begin entering lines
20     BETA, BETTY         without any prompting
30     DAHLEN, COLLEEN
40     GAMOW, GEORGE
50     KALER, TOM
0 <-------------------------A zero ends input
DONE. THE FILE WILL NOW BE ORDERED NUMERICALLY
STUDEN HAS BEEN ORDERED
COMMAND--
```

Exhibit 6 (cont.)

```
PR NAMES <------Let's print out a stored file
WHERE SHOULD I OUTPUT, TE OR DI?
TE


          FILE NAMES

TEST FILE WITH STUDENT NAMES

   10 ALPHA
   20 BETA
   30 GAMMA
   40 DELTA
   50 EPSILON
   60 LAMBDA
   70 MU
   80 OMEGA


COMMAND--
RECORD NAMES <------Let's find out how many records NAMES has
NAMES  CONTAINS     8 RECORDS <------Correct!
COMMAND--
NF NAMES <------We tried to create a new file called names.
FILE ALREADY EXISTS. TO MODIFY AN EXISTING FILE USED MODFILE COMMAND
C OMMAND--
NR HELP<------Now let's learn how to make a new relation
    COMMAND NEWREL OR NR
THE SYNTAX IS:

(NEWREL)  (RELATION-NAME)  (TE)
  (NR)        (HELP)       (CA)
                           (HE)

THIS COMMAND IS USED TO CREATE A NEW RELATION FROM
SCRATCH. INPUT CAN COME FROM CARDS(CA) OR FROM THE
TERMINAL(TE). HE AS THE THIRD ARGUMENT MEANS HELP
TO USE CARD INPUT YOU MUST 1)LEAVE THE SPOIS. 2)TYPE
AS CR1. 3)REENTER THE SPOIS. 4)USE THE NEWREL COMMAND
AFTER YOU ARE DONE, BE SURE TO UNASSIGN CR1.

COMMAND--
NR SAMPLE<------Now let's create a new relation
WHAT IS THE INPUT MODE?
TE<------Input will be from the terminal
 PLEASE TYPE A ONE LINE DESCRIPTION OF THE RELATION
SAMPLE RELATION FOR SAMPLE SESSION
NOW BEGIN ENTERING THE NAMES OF THE FIELDS IN YOUR
RELATION,UP TO 3 PER LINE. TO STOP, TYPE EN:
```

Exhibit 6 (cont.)

```
NAMES <-------------------------Relation will have four fields
CONCEN FUNCT <------------------
FUNCT EN <----------------------
SAMPLE HAS  4 FIELDS<-The system knows how to count
NOW BEGIN ENTERING LINES OF NUMBERS.
DO YOU WANT AN EXPLANATION?
YEP<-------Slang acceptable, as long as first two letters are YE
TYPE IN REFERENCE NUMBERS AS MANY AS YOU WANT ON A LINE.
NUMBERS MAY BE SEPARATED BY ANY CHARACTER YOU WISH EXCEPT
A DIGIT OR THE SYMBOL #. TYPING # IMMEDIATELY AFTER A
NUMBER CANCELS IT. FOR EACH NEW RECORD YOU WILL BE PROMPTED
BY THE SYMBOL >. TO STOP, TYPE 32767 AS THE FIRST
NUMBER OF A NEW RECORD.
>
10 1 85 52 >          <-------Free format input.Prompt character
20/2/72↑41!>          <-------is >.
30#31#30 7 62 0 > <-------# cancels previous number
40(3)=14...NOT SO. NEXT NUMBER IS 12 > <-letters ignored
NOTICE HOW LETTERS ARE IGNORED:DOES 50+3=62? NO
0<-> <-------fourth number of this record on a new line.
32767<------Stop input.

SAMPLE HAS     5 RECORDS.
COMMAND--
IN RE<---------Let's list all relations in the SPOIS

SPOIS INFO:
IHOPE -RE-RELATION IS CALLED IHOPE BECAUSE I HOPE THIS WORKS
SAMPLE-RE-SAMPLE RELATION FOR SAMPLE SESSION



COMMAND--
IN RE BR<---------Let's list them again, but without descriptions

SPOIS INFO:
IHOPE -RE-
SAMPLE-RE-



COMMAND--
DE IHOP**<---------Delete this print file
IHOP** HAS BEEN DELETED
COMMAND--
DE SAMPLE<-Delete our relation. For relations and files system
ARE YOU SURE YOU WANT TO DELETE SAMPLE? checks to make sure you
YEP                                     really want to delete
SAMPLE HAS BEEN DELETED          them and haven't made a mistake.
COMMAND--
```

Exhibit 6: Sample Terminal Session (cont.)

PR FUNCT TE ← Print out the functional job areas file.

```
          FILE FUNCT

FUNCTIONAL JOB AREAS -- 2 MAY 1975

   11 BANK TRUST DEPARTMENT
   12 BANK COMMERCIAL LOAN DEPT.
   13 BANK SECURITY PORTFOLIO DEPT.
   14 FINANCIAL PLANNING
   15 CONTROLLER,BUDGET,AUDITING
   16 TREASURER AREA
   17 CHIEF FINANCIAL OFFICER
   21 MARKETING RESEARCH
   22 NEW PRODUCT DEVELOPMENT
   23 SALES
   24 PRODUCT/BRAND MANAGEMENT
   25 MARKETING MANAGER/DIRECTOR
   31 MANUFACTURING
   32 PURCHASING,MATERIALS MGMT,INV. CONTROL
   33 DISTRIBUTION,TRAFFIC,TRANSPORT.
   34 QUALITY CONTROL & MAINTENANCE
   35 PRODUCTION PLANNING
   36 ENGINEERING
   40 GENERAL MANAGEMENT
   51 DATA PROCESSING & MIS
   52 SYSTEMS ANALYSIS/SYS. ENGINEERING/PROGRAMMING
   53 O.R./MANAGEMENT SCIENCE
   54 COMMUNICATIONS
   61 ENGINEERING
   62 RESEARCH
   71 INDUSTRIAL RELATIONS
   72 PERSONNEL
   73 ORGANIZATIONAL DEVELOPMENT
   81 CORPORATE PLANNING
   82 ECONOMICS
   83 PUBLIC RELAS
   84 LEGAL
   85 CONSULTANT
   86 ADMINISTRATIVE ASSISTANT
   87 MANAGEMENT TRAINEE
   90 OTHER


COMMAND--
OH NO ← Illegal input, but no harm done.
ILLEGAL INPUT,TRY AGAIN--
COMMAND--
```

Exhibit 6 (cont.)

```
PR FUNCT DI<---We output file FUNCT onto disk
YOUR OUTPUT WILL BE IN FUNC**  This creates the print
                               file called FUNC**

COMMAND--
EN<------We leave the SPOIS, so we can spool FUNC**

OK, SPOOL FUNC**<--This causes FUNC** to be printed out on
GO                    the Centronics printer.
YOUR SPOOL FILE IS PRNTIØ

OK, R SPOIS<------Now we reenter the SPOIS
GO



WELCOME TO THE SPOIS.
IF AT ANY TIME YOU REQUIRE ASSISTANCE TYPE HELP.
COMMAND--
IN PR<---List all print files

SPOIS INFO:
FUNC**-PR-



COMMAND--
DE FUNC**<---Now that we have spooled FUNC**, let's delete it
FUNC** HAS BEEN DELETED      to save disk space.
COMMAND--
END<-------We're done for now, so let's leave the SPOIS.
OK, LO<----Abbreviation for LOGOUT
XV.MPT  HAAG    (2) LOGGED OUT AT  Ø6'56 Ø5165<---
TIME USED= Ø0'21  Ø0'Ø6  Ø0'Ø0          Time: 6:56 AM. Date:16 May 1975
                  We were on for 21 minutes
OK,
```

Exhibit 6 (cont.)

USER'S GUIDE TO THE

SLOAN PLACEMENT OFFICE INFORMATION SYSTEM

First Edition

by PETER G. HAAG

May 1975

TABLE OF CONTENTS

INTRODUCTION

The Sloan Placement Office Information System (SPOIS) was written by Peter G. Haag in the spring of 1975 for his Master's thesis for the Sloan School. For a more thorough background and a more detailed system level description of the file structure and programs refer to that document.

The SPOIS is a fairly general relational data-base system. Though it was written for the Placement Office, it could, in fact, be used for other purposes. To use the system requires some understanding of relational data-base concepts. A summary will be given here.

Suppose we had an index that listed a group of students, their concentrations, and the functional job areas they were interested in. Such an index might look like this:

		Functional Job Area	
Name	Concentration	First Pref.	Second Pref.
Alpha, John	MIS	Consultant	Systems Analyst
Beta, Betty	Organ. Studies	Personel	General Management
Dahlen, Colleen	OR	Research	-
Gamow, George	Finance	Financial Planning	Loan Officer
Kaler, Tom	Finance	Research	-

If such an index were stored in a computer we would like to take advantage of the computer's capabilities to do various things with it. For example, we might want to switch the first two columns, or order the index by first preference job. If we had a similar index for another group of students we might want to merge the two.

The traditional way of storing this index is to store the actual text of each row. Unfortunately, this makes the operations we would like to do cumbersome. We therefore create the following FILES:

Name File

Number	Name
10	Alpha, John
20	Beta, Betty
30	Dahlen, Colleen
40	Gamow, George
50	Kaler, Tom

Concentration File

Number	Concentration
1	MIS
2	Organizational Studies
3	Finance
4	Operations
6	Marketing
7	O.R.
10	Labor Relations

Functional Job Areas File

Number	Area
12	Bank Loan Department
14	Financial Planning
40	General Management
52	Systems Analyst
62	Research
72	Personel
85	Consultant

We can now specify our index (which we will henceforth call a RELATION) in the following manner:

Name	Concentration	Functional Job Pref.1	Pref.2
10	1	85	52
20	2	72	40
30	7	62	0
40	3	14	12
50	3	62	0

Each of the columns or FIELDS of the relation is headed by the name of the file to which it corresponds. The reference numbers (or REFNOs) refer to the entries in the appropriate files. Note that even though we have two functional job columns we need only one file. Also, the Concentration file has entries which we do not use in this relation. We have put them in in case we later add a student who is in such a concentration. Finally, note that the numbers we have used are arbitrary except that when we order

a relation it will be according to these numbers. Thus, we have been careful to give the students numbers which, when ordered numerically, put the names in alphabetical order.

There is one case in which the refnos are not arbitrary, namely when the numbers themselves have meaning and there is no associated text. If, for example, you wanted to have a field called Salary in the relation you would simply add it on without creating a file for it. When the relation is printed out the refnos will be printed out as they are, since no file exists for them.

There are several commands in the SPOIS for doing arithmetic calculations on such fields of refnos. Actually, these commands can be used on any other field, but the results obtained would be meaningless.

One other term should be defined here, namely DATASET. This is used to refer to any collection of data. Two examples are files and relations. Two other examples are dump files (see DUMP command) and print files (see PRINT command).

There are several advantages of relational data-bases over conventional ones. Most important is the ease of manipulating the data. Since each relation is a matrix of numbers, it is very easy to sort by any column, switch columns, add and delete rows and columns, take subsets, etc. If the data had been stored in the conventional way such

manipulations would be terribly time-consuming and fairly difficult jobs.

Another advantage is that the use of refnos can save storage. This will be the case if a field has only a few possible values, as does Concentration in our example (imagine a relation with 1000 students listed; in comparison to this the number of possible concentrations is small). The concentration file is short and takes little storage. Also, every time a relation has more than one field containing the same type of data (i.e. where the refnos refer to the same file) space is saved, since only one file is needed.

A word or two about the use of refnos. Refnos are the key ingredient in relational data-base systems. It is possible, however, to make them invisible to the user. In such a case the user refers only to the data itself and the system determines the refnos. This requires a set of strategy modules that assign refnos such that the data in the files is arranged in the sequence the user wishes. It also requires that the user be extremely careful in entering his text the same way each time. It was decided that these considerations would make the system too complex. In the future such modules may be added.

Important Note: Currently all refnos must be integers between 1 and 32766 inclusive.

The SPOIS provides a set of commands with which the user

can create and modify files and relations. In addition, relations can be manipulated in various ways and reports can be generated. The use of these commands are described in this manual.

ENTERING AND LEAVING THE SPOIS

The normal way of getting access to the SPOIS will be to use one of the teletypes in the basement of Sloan, Room E52-081. These terminals are wired directly to the Prime computer. To establish a connection, simply turn the switch to LINE. Now simply copy the following sequence. Everything that is underlined is typed by the user; the computer does the rest:

```
     ?LOGIN XV.MPT HAAG
PASSWORD
CCDP
XV.MPT HAAG(2) LOGGED IN AT 14:31 05085
OK, R SPOIS

WELCOME TO THE SPOIS.
IF AT ANY TIME YOU REQUIRE ASSISTANCE TYPE HELP.
COMMAND--
```

Please note that both the login line and the password are subject to change, although the format is not. You should therefore check to determine what the exact login is.

After the last line is typed you can begin entering commands. When you are finished, type the following sequence:

```
COMMAND--
END

OK, LO
XV.MPT HAAG (2) LOGGED OUT AT 15°09 05085
TIME USED = 00°38  01°14  00°23
```

The 00°38 means you were on the system for 38 minutes. You now turn off the teletype and are done.

The system can also be used from other terminals, including video terminals and the IBM 2741 and Trendata 1000's. Currently this possibility is not always available; you should check with the operator to see if these terminals may be used. These all require you to dial up the Prime as follows:

(1) Turn on the power.

(2) Make sure the switch is set to COM, not LCL.

(3) On the data-set (the telephone-like device), press the button labelled TALK, and pick up the receiver.

(4) Dial the Prime (8-6008 from the video terminals, 8-6009 from the 2741's or Trendatas)

(5) When you hear a high pitched tone press the DATA button and hang up. The button should light up. If it does not, hang up and go back to step (3)

(6) Now login as above.

Logout is the same as for the teletype.

HELP AND INSTRUCTIONS

Now that you have logged into the SPOIS you can begin entering commands. This system is quite interactive, so that it is reasonably flexible to use. As the introductory message notes, if you ever need any information you can type HELP. If you are at command level (you are at command level whenever the computer has just typed the message COMMAND--) you can get a list of legal commands by typing HELP. To find out more about a specific command type the command followed by HELP, e.g.

NEWREL HELP

The system explains the uses and syntax of the command, then returns you to command level. You can type HELP at virtually any time. However, you cannot use HELP when you are inputting data or when the system asks you for numbers.

Another way in which the system is flexible is in the way it will accept commands. If you type only the command the system will ask you for the additional information it needs. At this point do <u>not</u> repeat the command. Just give it what it asks for. If you were to type in the command and the first argument the system would, if it needed it, ask you for the second argument.

Sometimes the system will require data from you, e.g. when you are creating or modifying files or relations. In these cases it will often ask you if you want instructions. You should answer YES if this is the first time you have used the command or have forgotten the format. The instructions tell you how to enter the data. If you are familiar with the command type NO. In the first case begin entering your data after the explanation is given. In the latter, begin entering it immediately.

ERRORS, PROBLEMS, ETC.

If you make a mistake in typing there are two ways to correct it. Typing a question mark (?) deletes everything you have typed so far on the line. After you type the question mark simply enter the correct line as though you were at the beginning of the line. Example:

```
NEFIL XY?NEWFIL XYZ
```

is entirely equivalent to

```
NEWFIL XYZ
```

If you only wish to delete one or a few characters you can use the double quotes ("). One such character will delete the previous character typed. Two will delete the previous two characters, and so forth. Thus:

```
OP"RDER TQP""EST
```

is exactly the same as

```
ORDER TEST
```

Note that once you have entered a line and hit the carriage return you can no longer change it.

In an emergency you can leave the the SPOIS by hitting CTRL-P (press the CTRL button and the P button simultaneously) on a teletype or hitting the ATTN (Attention) key on other terminals. The system will stop whatever it is doing and type QUIT, . You can now do one of two things. If you decide that your action was rash you can type S. This causes the system to resume what it was doing. If, on the other hand, you were justified, you can either logout or return to the SPOIS. To do the latter you should first type C ALL. This ensures that all files are closed and not just hanging in limbo. If you don't do this you may later get the message UNIT IN USE and be thrown out of the SPOIS. After typing C ALL you can enter the SPOIS as you did in the beginning, typing R SPOIS.

COMMANDS AND SYNTAX

The following sections describe the syntax and uses of the various commands. The way to enter a command is to type the command or its abbreviation followed by its arguments, if any. Commands and arguments are separated from each other by one or more spaces. Each command is six letters long and has a two letter abbreviation. Some commands are listed with more than six letters for mnemonic ease, but only the first six are significant. When you enter a command it will be checked to see if the first six letters agree with the six-letter name or the first two letters agree with the abbreviation. Thus, you can invoke the NEWFILE command (abbreviation NF) in any of the following ways:

```
NEWFILE
NEWFIL
NF
NEWFILECABINET
NFXYZ
NF123XGGG, etc.
```

In this manual and when using the HELP command the syntax of the commands will be specified in a standard format as in the following example:

```
(NEWREL)   (relation-name)    (TE)
  (NR)          (HELP)        (CA)
                              (HE)
```

For the items in parentheses you must choose one from each column. Upper-case items are entered literally, while for the lower-case item you must replace it with the appropriate name. Note that the teletype has no lower case, so you must resort to common sense or this manual to distinguish the two possibilities.

If you leave off the last one or two arguments the system will ask you for what it needs. Note, however, that the arguments are positional, so you could not leave out the first argument and enter the second. If you typed NEWREL TE the system would think you wanted to create a new relation named TE. It would then ask you for the second argument. Also, if your first argument is HELP the second one will be ignored.

Some commands have optional arguments. These will be surrounded by the symbols < and >.

HE as the second argument is always equivalent to HELP as the first. This is useful if you are in the middle of entering a command and suddenly realize that you are not quite sure how to use it.

When entering data there are two formats, fixed format and free format. In free format numeric input you enter as many numbers as you wish on a line, separating them by any non-numeric character except the symbol. This symbol entered

immediately after a number cancels it. The following two lines are equivalent in free format input:

```
323#503 751-7.5000/6#79
503  751  7  5000  79
```

For this type of input only the symbols " and ? do not cause any deletion; they are treated just like any other delimiter..

When entering names in free format (e.g. field names using the NEWREL command) you can enter one to three names per line, using as many lines as necessary.

For fixed format you will be given instructions. Remember that if you use the ? to erase a line it will be as though you were back in column one. Similarly, a " will also shift the column you are in.

The command descriptions now follow. These descriptions are more thorough than those available with the HELP commands, but the latter will always be the most up-to-date.

CNAME--CN

(CNAME) (old-name) (new-name)

(CN) (HELP) (HELP)

Use this command to change the name of a relation or file. The contents remain unchanged.

COPY--CO

(COPY) (existing-dataset-name) (name of copy)

(CO) (HELP) (HELP)

This command is used to make a copy of an existing file or relation. The two will be identical except for the name. This is useful if you wish to make various changes to a dataset yet still keep a copy of the original.

CREATE--CR

(CREATE) (old-relation-name) (new-relation-name)

(CR) (HELP) (HELP)

This command is used to create a new relation using an existing one. The new relation will be a subset of the old one and both will be saved. After you have supplied the two

arguments the system will ask you for additional information. First it will ask you to type a one-line description (up to 72 characters) of the new relation. This description is printed out when you invoke the PRINT and INFO commands. You will then be asked to list the fields you wish to include in your new relation (Remember that these must be fields of the old relation). To determine which fields a relation has use the FIELDS command. Next, you will be asked to supply the refnos of the first and last records to be included in the new relation. These refnos must be in the first field of the relation. Your new relation will now contain only selected rows and columns of the old one.

DELETE--DE

(DELETE) (dataset-name)

(DE) (HELP)

This command is used to delete a relation, file, print file, or dump file. Once something is deleted the system in essence gets rid of it and forgets that it ever existed.

DUMP--DU

(DUMP) (relation-name) (TE)

(DU) (HELP) (DI)

(HE)

This command is used to print out, in matrix form, all refnos in a relation. The output will look similar to the relation depicted on page 5 of this manual. If the matrix doesn't fit across the width of the paper each record will be printed entirely, but on more than one line. Continuation lines are indented. The second argument tells where to put the output. TE specifies to print it at the terminal. DI means to create a disk dump file. In the latter case the name of the dump file will be the first four letters of the relation name with two plus symbols (++) added on to the end. No printed output will be generated automatically. To get output you must leave the SPOIS using the END command, then spool the dump file. This causes it to be printed out on the high-speed printer in Room E52-083. Suppose you wanted to dump a relation called MASTER. You would first type: DUMP MASTER DI . The system would respond: YOUR OUTPUT WILL BE IN MAST++ . You would then leave the SPOIS and type: SPOOL MAST++ . You can then reenter the SPOIS. It is a good idea to get rid of dump files after you have spooled them because they waste storage and can always be

recreated. To do this, use the SPOIS DELETE command.

EDIT--ED

(EDIT) (relation-name)

(ED) (HELP)

Use EDIT to modify a relation or display individual records. This is the system's only two-level command. Issuing this command gets you into edit mode. Until you leave this mode you will be working with one specific relation. After each subcommand the system will reply EDIT--. You can now issue the following subcommands:

(ADDREC)

(DELREC <field-name>)

(ADDFIELD field-name)

(DELFIELD field-name)

(DISPLAY <field-name>)

(SWITCH field1 field2)

(CHEAD)

(QUIT)

ADDREC: Use this subcommand to add records to the relation. After you have typed ADDREC the system will ask you to begin

entering records in free format. Simply type in the refnos. You will be prompted by the symbol > for each new record. To stop type 32767 as the first number of a new record.

DELREC: Use this subcommand to delete records. After you type DELREC the system asks you to begin entering refnos in free format. These are refnos in the first field of the relation unless you specified a field name, in which case that is the field checked for the refno. Keep entering numbers until you are through. To stop, enter 32767. The deleted record is the first one encountered whose refno matches the one you entered. In some cases there will be other records that match the refno; thus, you can enter the same number several times, if you wish.

ADDFIELD: This is used to add a field on to the relation. The field is added on to the right side of the relation. For each record the system types the text of the first field. You then supply, in free format, the refno for the field to be added. Entering 32767 at any time before all records have been updated cancels this subcommand. You can subsequently change the position of this field using the SWITCH subcommand.

DELFIELD: This subcommand deletes the specified field.

DISPLAY: Prints out the text and refnos of one or more records. After issuing this subcommand you will be asked to enter the refnos in free format. If you have not specified a field name the first field is searched for that refno. If you did specify a field, that is the one that will be searched. All records that match will be printed.

SWITCH: This switches the two fields specified.

CHEAD: Used to Change the descriptive HEADer. After you have entered this subcommand the system will print out the current header. You then enter either NC for No Change or a replacement header.

QUIT: This gets you out of edit mode and back to command level.

END--EN

(END)

(EN)

This command gets you out of the SPOIS. Use it when you are through or need to assign the card reader (see NEWFILE and NEWREL) or spool output (see DUMP and PRINT).

FIELDS--FI

(FIELDS) (relation-name)

(FI) (HELP)

This command lists, in order, the names of all the fields of a relation.

HELP--HE

(HELP)

(HE)

This command causes the system to print out a list of all the available commands with a one-line description of each.

INFO--IN

<FI>

(INFO) <RE>

(IN) <PR>

<OU>

<HE>

<AL>

This command is used to list your datasets. The first argument specifies which dataset type to list. FI=files only; RE=relations only; PR=print files only; DU=dump files only; AL=all types. HE means HELP. If you omit this argument you will get all types. BR, the second argument, means brief. If you do not specify BR, INFO will print out the text of the descriptive header for files and relations.

MEAN--MN

(MEAN) (relation-name) (field-name)

(MN) (HELP) (HELP)

Use this command to calculate the mean value of aa field of a relation. All records are used.

MEDIAN--MD

(MEDIAN) (relation-name) (field-name)

(MD) (HELP) (HELP)

Use this command to calculate the median value of a field of a relation. All records are used.

MERGE--ME

(MERGE) (relation1) (relation2)

(ME) (HELP) (HELP)

Use this command to merge the contents of one relation with another one. After you have entered the two arguments the system will respond with:

NAME OF MERGED RELATION?

You then enter a relation name. If the name you give is the same as that of one of the arguments, then the contents of the other argument are added to the named relation. This relation will then have the contents of both, while the other one remains unchanged. If you specify any other name a new relation will be created containing the contents of relation1 followed by the contnts of relation2. In this case you will also be asked to supply a one-line description of the new relation.

In order to merge two relations they must have exactly the same fields in the same order. Use the FIELDS command to determine which fields a relation has. If necessary, use SWITCH, ADDFIELD, or DELFIELD subcommand of the EDIT command to make relations compatible for MERGEing.

MODFILE--MF

(AR)

(MODFILE) (file-name) (DR)

(MF) (HELP) (CH)

(HE)

This command is used to modify the contents of a file. Records may be added or deleted and the descriptive header can be replaced. Records cannot be modified; however, the same effect can be achieved by first deleting a record then adding a replacement. If you specify AR (Add Records) the system will ask you to begin entering lines in fixed format. The format is the refno in columns 1-5 and the text beginning in column 7. The width of the text field was established when the file was first created and cannot be changed; thus, if you enter text that is too long it will be truncated. When you are done adding records enter a refno of zero with no text. The file will then be ordered automatically by refno.

If you specify DR (Delete Records) you will be asked to list in free format the refnos of all records to be deleted. Enter as many per line as you wish. To stop, enter a zero.

If you specify CH (Change Header) the system will type out the current header. You then enter either NC (for No Change) or a replacement line.

NEWFILE--NF

(TE)

(NEWFILE) (file-name) (CA)

(NF) (HELP) (BO)

(HE)

This command is used to create a new file. Input can be from the terminal (TE), from cards (CA), or from both (BO).

Terminal Input: The system will ask you first how many characters wide the text field will be (this must be from 1 to 66 characters). Then it will ask you to type a descriptive header. Then you begin entering records in the following fixed format: Refno anywhere in columns 1-5; text beginning in column 7. If you enter more characters than you specified, the text will be truncated. To stop, enter a refno of zero.

Card Input: Before using this option you must leave the SPOIS and type AS CR1. This reserves the card reader for your exclusive use. Now reenter the SPOIS and use the NEWFILE command with the CA option. The system will take it from there. Cards must be in the following fixed format:

Card 1-Columns 1-2: FI
Columns 4-5: one or two digit integer giving the number of characters in the text field
Columns 7-78: Description of the file

Cards 2,3,...-Columns 1-5: Refno
Columns 7-:Text (field size

specified on previous card)

Last card-Column 1: a zero

Both Input: As with cards, you must leave the SPOIS and type AS CR1. You will then be asked for the size of the text field and a one-line description of the file. Thereafter the system will read in the cards. The card deck is the same as for CA except that you leave off the first card. The card reader is located in Room E52-083. Ask the operator for assistance in using it. Make sure your deck is in the reader before invoking the command. After you are finished using the card reader you should leave the SPOIS and type U CR1. This frees the card reader so that other people can use it.

NEWREL--NR

(NEWREL) (relation-name) (TE)

(NR) (HELP) (CA)

(HE)

This command is used to create a new relation from scratch.

TE (Terminal input): The system will first ask for a one-line description of the relation. It then asks you to enter the names of all the fields in free format (see the

Commands and Syntax section of this manual). The order you enter them is the same order they will be stored in. You will then be asked to enter refnos in free format. Be sure to keep the order of the refnos the same as the order of the fields. After each record is complete the system will prompt you with a > symbol. To stop, enter the number 32767 as the first number of a new record.

CA (Card input): Before using this option make sure you have typed AS CR1 before entering the SPOIS. That command gives you exclusive use of the card reader. If you have not, leave the SPOIS and do so, then reenter the SPOIS and use the NEWREL command. The system takes care of the rest. Card format is as follows:

Card 1- Columns 1-2: RE
Columns 7-78: Description of the relation

Card 2- Columns 1-6, 8-13,15-20,... Field names in the desired order, left justified in the field. The last entry must be EN. Use up to three cards if necessary, but do not begin a new card until you have filled up the current one.

Subsequent cards- Columns 1-5,7-11,13-17,... Refnos. Use up to 3 cards per record if necessary, but do not begin a new card until you have filled up the current one.

Last card- Column 1: a zero

After you are done using the card reader you should leave the SPOIS and type AS CR1. This frees the card reader so that other people can use it.

ORDER--OR

(ORDER) (relation-name) (field-name)

(OR) (HELP) (HELP)

This command orders the specified relation by the refnos in the given field. Note that it is never necessary to order a file, since files are always maintained in order.

PRINT--PR

(PRINT) (relation-name) (TE)

(PR) (file-name) (DI)

(HELP) (HE)

This command is used to print out the contents of a file or relation. For relations the refnos are replaced with their corresponding text. The last argument specifies where the output is to go. TE causes the dataset to be printed out at the terminal. DI means that it is to be put on disk, in a print file. A print file can later be printed on the line printer in Room E52-083. The name of the print file will be the same as the dataset from which it came, except that the last two letters will be replaced with two asterisks (**). To output a print file on the high speed printer you must leave the SPOIS, then type SPOOL

print-file-name. It is a good idea to delete print files after they have been spooled, since they take up storage and can be recreated at any time.

RECORDS--RE

(RECORDS) (file-name)

(RE) (relation-name)

(HELP)

This command tells how many records the specified file or relation currently has.

SUM--SU

(SUM) (relation-name) (field-name)

(SU) (HELP) (HELP)

Use this command to calculate the sum of all the refnos in a field.

THE END