The following users guide has been extracted and reworked from
http://www.lsu.edu/OCS/mainframe/mvs/tsodoc.html (Louisiana State University)

Using TSO/ISPF

Table of Contents

1. Introduction

The purpose of this document is to assist those who are learning to use TSO/SPF. TSO and SPF are two separate products, both of which are intended to help you in your work with an OS/390 computer. TSO (Time Sharing Option) is a command-driven software product which typically employs one-word or one-line commands, while SPF (System Productivity Facility) is a newer, full-screen, menu-driven product which both simplifies and extends the capabilities of TSO.

SPF allows you to perform almost all of TSO's functions by using menus and panels--and may simplify your work on the system by making it unnecessary to remember or key in longer TSO commands. Most experienced TSO/SPF users use a combination of the capabilities of TSO and SPF, choosing to use one product or the other depending on which one happens to be simpler or faster or clearer for the particular application.

2. Key TSO Concepts

There are only a few concepts which need to be understood before you can effectively use TSO or SPF. All of these have something to do with the concept of a data set. In TSO, the name "data set" is used to mean a file. A file is made up of a collection of one or more records. A record is composed of one or more fields. A field is composed of one or more characters. Figure 1 depicts each of these terms.

```
 field 1  field 2   field 3
    |       |          |
+------+---------+------------------+
| BOB  | 7665656 | ENGINEERING      | <--- record 1
+------+---------+------------------+
| TOM  | 7676756 | FORESTRY         | <--- record 2
+------+---------+------------------+
| FAYE | 7673421 | LAW              | <--- record 3
+------+---------+------------------+
```

Figure 1. A Data Set (i.e., a file)

Initially, you should know the difference between two types of data sets: "sequential data sets" and "partitioned data sets." A sequential data set contains records stored one after the other (like the records in the file of Figure 1). A partitioned data set contains one or more sequential files, all in one data set but separated (i.e., partitioned) from each other. In a partitioned data set, each individual file is referred to as a "member" of the data set. Figure 2 presents simple sketches of both a sequential data set and a partitioned data set. Note that the partitioned data set also includes a "directory" to identify and locate each data set member.

```
  +------------------------------------+
  |                                    |  SEQUENTIAL
  +------------------------------------+   DATA SET

  +---+---------+----+--------+------+
  |DIR|         |    |        |      |  PARTITIONED
  +---+---------+----+--------+------+   DATA SET
         |        |       |      |
     MEMBER 1     |   MEMBER 3   |
              MEMBER 2        MEMBER 4
```

Figure 2. A Sequential Data Set and a Partitioned Data Set

Data sets have names. Each data set name is composed of two or more parts or "qualifiers" separated by periods. For example, the data set name SYS3.MESSAGE.SEMINARS has three qualifiers and the data set name CSLSU.DATA has two. Each qualifier:

(1) must begin with an alphabetic character or national character ($,#,@), with the first letter of the first qualifier being an alphabetic character,
(2) normally contains only alphameric characters (A-Z,0-9), and
(3) cannot be longer than eight characters.

The first qualifier is referred to as the "high level" qualifier. When referring to your personal data sets (i.e., a data set whose first qualifier is your logonid), the high level qualifier need not be stated. If the high level qualifier is omitted, TSO automatically assumes that your logonid is the data set's high level qualifier. But how does TSO know if you have omitted the first qualifier? If a data set name is given without being enclosed in apostrophes, TSO assumes that it is one of your personal data sets and places your logonid at the front of the data set name. Data set names are said to be "fully qualified" if they are referred to by all their qualifiers.

EXAMPLE

Suppose a person's logonid is CSLSU and one of his data sets is named PROGRAM.FORT. If that data set is included in a TSO command, it could be written as either PROGRAM.FORT or as 'CSLSU.PROGRAM.FORT'. The fully qualified data set name is 'CSLSU.PROGRAM.FORT'.

Logonids that begin with a 'Y' belong to students (e.g., YCSLSU). Students operate in a limited TSO environment.

EXAMPLE

Suppose a student's logonid is $CSLSU and that he wants to create a data set whose last two qualifiers are PROGRAM.FORT. Suppose further that his classid is CS2134A. The data set created must be named 'CS2134A.$CSLSU.PROGRAM.FORT'. For this student, the first qualifier for all his data sets must be CS2134A and the second qualifier must be $CSLSU.

Partitioned data set MEMBERs also have names which follow the same rules as those given for individual qualifiers--one to eight alphameric characters beginning with an alphabetic character. Since members are parts of partitioned data sets, they must be referred to slightly differently than data sets themselves. Member names are enclosed in parentheses and placed after the name of the data set.

For example, if NUMBER1 is a member of the partitioned data set CSLSU.PROGRAM.FORT, it would be referred to as either PROGRAM.FORT(NUMBER1) or 'CSLSU.PROGRAM.FORT(NUMBER1)'. Another pair of important data set related terms are "cataloged" and "uncataloged." A data set is said to be cataloged if its name and location are listed in the system's master catalog. If its name and location are not listed in the master catalog, but it nevertheless resides on one of the system's disk packs, the data set is said to be uncataloged. The default on TSO is that all data sets are cataloged.

3. The Two Commandments of TSO

There are two TSO concepts regarding data sets which can be especially troubling for the new TSO user, but which must be understood before you can work successfully with TSO. They might even be called "The Two Commandments of TSO."

Commandment 1: Thou shalt ALLOCATE a data set before using it.
Before you can use a data set to store data or a program, you must first reserve sufficient space on a disk for that data set. The process is referred to as "allocation." That is, you must "allocate a data set." After you have allocated a data set, you may then begin to work with that data set. Section VIII explains the allocation process in detail.

Commandment 2: Thou shalt COMPRESS partitioned data sets regularly.
To compress a data set is to discard all old versions of data set members, to keep only the most recent version of all data set members, and to recoup the space used by the old versions. You cannot over-compress or squash data, no matter how many times you compress the data set. Note: Keeping old versions of members of a data set serves no useful purpose, since you cannot access these old members. They simply take up valuable space in your partitioned data set.

EXAMPLE

Suppose that a 100-line program is stored in a member of a partitioned data set. Suppose further that the program is run, found to have errors, edited to get rid of those errors, and stored again. TSO keeps both copies of the program (i.e., the member)--the new edited version and the original version. Next suppose that the newest version of the program is run and it also is found to contain errors. Again the program is corrected and saved. At this point, the partitioned data set contains three similar versions of the same program (approximately 300 lines of code)--only one of which can be accessed (the most recent).

The obvious problem in the above example is that each new version of the member requires an approximately equal portion of disk space--which has a limit. TSO does not automatically discard previous versions and then free the disk space for reuse. Rather, TSO waits indefinitely for you to tell it to get rid of the old member versions (by issuing a command to compress the data set). If no such command is given, the data set continues to fill up with outdated versions until eventually your allocated data set space is completely full. Section XIII explains how to compress a data set. You may think that these steps are awkward and that learning TSO is going to be really strange, but once these two concepts are understood, the remaining concepts are straightforward.

4. Logging On to TSO

Logging on to TSO requires that you know two things:

(1) your logonid (sometimes called your "userid") and
(2) your password

You will begin the logon process from a menu similar to the one shown in Figure 3. You simply follow the instructions given on the screen, substituting your unique logonid for "logonid".

Note: Throughout TSO, you must press the enter key after you type in a command. This must also be done during the log on process.

EXAMPLE

Our example user logging on to TSO has been assigned logonid CSLSU and password LEARN. Note that "T CSLSU" has been entered along side the "Select" arrow in Figure 3.

```
|    09:15:45  Aug 25 <STUDMENU, Page:001 of 001> Term=H1T40C58      |
|    ------------- LSU/SNCC ----------- PF1/PF13 ---------------     |
|    To LOGON to TSO <enter> ==> T logonid                           |
|    To LOGON to VM <enter> ==> V logonid                            |
|                                                                    |
|    Select ==> T CSLSU                           <--- user entry    |
|                                                                    |
|         L LOLA      - AVAILABLE   - Library On-Line Access         |
|         T TSO       - AVA 135/200 - Time Sharing Option            |
|         V VM        - AVAILABLE   - VM/CMS                         |
|                                                                    |
|    /B  # - BROWSE SELECTED BULLETIN TITLES BELOW.                  |
|                                                                    |
|         1 - SNCC Facilities and Computer Schedule                  |
|         2 - SNCC Seminar Schedule                                  |
```

Figure 3. Logon Menu Using CSLSU as the Logonid (Userid)

The logon screen shown in Figure 3 has a number of useful features. It shows the current time (09:15:45) and date (Aug 25), provides the network address of the terminal you are working on (H1T40C58), tells whether TSO and/or VM are available for use (plus telling the number of users logged on to TSO currently and the total number allowed to logon to TSO), and includes bulletin items. The bulletin items are of general interest and may be scanned without logging on (e.g., Select ==> /B 2 <enter> would cause the SNCC Seminar Schedule to be displayed on your screen).

Note: If a menu similar to Figure 3 is not on your monitor when you first sit down, you will need to take one or more steps to see that it appears.

(1) Case one occurs when a statement similar to the one below is present on your screen. To get the logon menu (Figure 3), you simply press the enter key.
CNWS095I - PRESS ENTER TO GET CL/MENU DISP; TERM=H1T40C58

(2) Case two occurs when the VM/LSU logo is shown. To get the logon menu (Figure 3), you type ===> VMEXIT <enter>.

After you enter your logonid and press <enter>, you will be presented with a nearly blank screen with the statement below shown at the top of the screen. ACF82004 ACF2, PASSWORD: It is here that you type your unique password and press <enter>. In our example above, the password given was LEARN. Be aware that--for security purposes--nothing you type in response to the password prompt will be seen on your screen.

Note: ACF2 is the assumed name of the security system which is used by the TSO system.

After entering your logonid and password, you will be "logged on," recognized as a TSO user. You will be shown several lines of messages, some of which are system generated and some of which may be announcements keyed in by SNCC staff members, followed by the TSO READY prompt. The last line should simply say READY, as shown below.

```
READY
```

Whenever the READY prompt appears, TSO is ready to receiveyour next command.

```
|       Note: One feature of TSO is that it allows you to spend as          |
|       much time viewing a particular screen as you like.                  |
|       However, it also informs you when there is more for you             |
|       to see.  TSO does this by placing three asterisks side-             |
|       by-side on the last line of the screen (e.g., ***).  To             |
|       move to the next screen--and to get rid of the three                |
|       asterisks--press the enter key.                                     |
```

EXERCISE 1: Logon to TSO using your unique logonid and password.

5. SPF Initialization and Invocation

SPF is an IBM software product used in conjunction with TSO that utilizes "menus" to allow users to readily select the tasks they wish to perform. Before you can use SPF, you must first issue the SPFPROF command from the READY prompt as shown below:

```
READY              <--- TSO prompt
SPFPROF    <enter>
```

The SPFPROF command needs to be entered only once for each new logonid. After it has been entered one time, you will forevermore be allowed to use SPF simply by typing SPF as shown below:

```
READY
SPF       <enter>
```

The SPF command will take you to the SPF PRIMARY OPTION MENU shown in Figure 4.

```
| ------------ LSU SNCC's ISPF/PDF PRIMARY OPTION MENU ------------  |
| OPTION ===>                                                        |
|                                                    USERID - CSLSU  |
|   0  ISPF PARMS   - Specify terminal/user parms    TIME   - 14:07  |
|   1  BROWSE       - Display source data/output     TERM   - 3278   |
|   2  EDIT         - Create or change source data   PF KEYS- 12     |
|   3  UTILITIES    - Perform utility functions      PREFIX - CSLSU  |
|   4  FOREGROUND   - Invoke lang. proc. in foreground               |
|   5  BATCH        - Submit job for language processing             |
|   6  COMMAND      - Enter TSO command or CLIST                     |
|   7  DIALOG TEST  - Perform dialog testing                         |
|   8  LM UTILITIES - Perform library management utility functions   |
|   C  CHANGES      - Display summary of changes for this release    |
|   T  TUTORIAL     - Display information about TSO/SPF              |
|   X  EXIT         - Terminate ISPF using log and list defaults     |
|                                                                    |
| These are extensions to the standard ISPF and ISPF/PDF.            |
|   I  IOF          - Interactive Output Facility ("I." for menu).   |
|   L  LSU          - LSU/SNCC maintained extensions.                |
|   S  SPECIAL      - Your Department's extensions - panel DEPTMAIN  |
|   U  USER         - Your own extensions          - panel USERMAIN  |
```

Figure 4. SPF PRIMARY OPTION MENU

As you become more familiar with SPF, you will probably want to use more of its capabilities. This introductory document, however, concentrates its attention on only four SPF options: 2 (EDIT), 3 (UTILITIES), 6 (COMMAND), and I (IOF).

| Note: An easy way to move around from screen to screen (i.e., |
| from option to option) in SPF is to type an equal sign |
| followed by the option you wish to invoke. For example, |
| after typing =2 <enter>, control is transferred to SPF 2, |
| EDIT. Once in EDIT, you could move directly to SPF |
| option 6 by typing =6 <enter> on the COMMAND line. This |
| facility saves you the trouble of returning to the |
| SPF PRIMARY OPTION MENU before moving on to another SPF |
| option. Once you are more familiar with SPF, you will |
| find this feature quite handy. |

EXERCISE 2: Invoke SPF (initialize if necessary).

6. Keyboard

When working with SPF, you will find that a number of keys are useful, especially the Program Function (PF) keys, whose default definitions are shown in Figure 5.

1 SPF HELP	2 SPLIT SCREEN	3 END (SAVE)
4 RETURN TO SPF MENU	5 REPEAT FIND	6 REPEAT CHANGE
7 SCROLL UP	8 SCROLL DOWN	9 SWAP SCREENS
10 SCROLL LEFT	11 SCROLL RIGHT	12 RETURN TO CMD LINE

Figure 5. Program Function Keys

Other Important Keys:

PA1	ATTENTION. Terminates processing of TSO command.
PA2	RESHOW. Redisplays the contents of the screen. May be useful if you have accidentally pressed the CLEAR key.
ENTER	SUBMIT TSO COMMAND for processing. TSO syntax requires that the ENTER key be pressed before a TSO command is issued.
DEL	DELETE a character.
INS MODE	INSERT a character. (Note: To perform an insert from within edit mode, you may have to delete one or more characters at the end of the line.)
<--'	NEW LINE. Move the cursor to the next line.
-->|	TAB. Move the cursor to the next field.
Arrows	The four directional arrows which control cursor movement.
CLEAR	CLEAR the screen.
ERASE EOF	ERASE END-OF-FIELD. Erase from the cursor to the end of the logical line shown on the screen.
RESET	RESET the keyboard. Turn insert mode off, free the keyboard, etc.

SHIFT	Used to get the "top" symbol for each key on the keyboard and to release the LOCK key.
LOCK	CAPS LOCK. Sets entire keyboard to point to the "top" symbol on each key. To release, tap the SHIFT key one time.

EXERCISE 3: Check your Program Function Keys against the PF key definitions shown in Figure 5 (which are the defaults) by typing 0.3 on the option line of the SPF PRIMARY OPTION MENU and pressing <enter>. After you have made your comparisons, return to the SPF PRIMARY OPTION MENU by pressing the PF4 (Program Function 4) key.

7. Allocating a Data Set

In Section IV, the concept of data set allocation was discussed. Allocation is the process of reserving space on disk for the storage of data, text, programs, etc. To allocate a data set using SPF, you must first be within the UTILITIES section (SPF 3), which is the choice displayed on the SPF PRIMARY OPTION MENU shown in Figure 6.

```
| ------------ LSU SNCC's ISPF/PDF PRIMARY OPTION MENU ------------ |
| OPTION ===> 3                                                     |
|                                                  USERID - CSLSU   |
|   0  ISPF PARMS   - Specify terminal/user parms   TIME   - 14:07   |
|   1  BROWSE       - Display source data/output    TERM   - 3278    |
|   2  EDIT         - Create or change source data  PF KEYS- 12      |
|   3  UTILITIES    - Perform utility functions     PREFIX - CSLSU   |
|   4  FOREGROUND   - Invoke lang. proc. in foreground               |
|   5  BATCH        - Submit job for language processing             |
|   6  COMMAND      - Enter TSO command or CLIST                      |
|   7  DIALOG TEST  - Perform dialog testing                         |
|   8  LM UTILITIES - Perform library management utility functions   |
|   C  CHANGES      - Display summary of changes for this release    |
|   T  TUTORIAL     - Display information about TSO/SPF              |
|   X  EXIT         - Terminate ISPF using log and list defaults     |
|                                                                   |
| These are extensions to the standard ISPF and ISPF/PDF.           |
|   I  IOF          - Interactive Output Facility ("I." for menu).   |
|   L  LSU          - LSU/SNCC maintained extensions.                |
|   S  SPECIAL      - Your Department's extensions - panel DEPTMAIN  |
|   U  USER         - Your own extensions          - panel USERMAIN  |
```

Figure 6. Choosing UTILITIES from SPF PRIMARY OPTION MENU

After the UTILITIES section has been selected, the next step is to choose option 2, DATASET, as shown in Figure 7.

```
| ------------------ UTILITY SELECTION MENU ----------------------- |
| SELECT OPTION ===> 2                                               |
|                                                                    |
|   1 LIBRARY    - Library utility:                                  |
|                         Print index listing or entire data set     |
|                         Print, rename, delete, or browse members   |
|                         Compress data set                          |
|   2 DATASET    - Dataset utility:                                  |
|                         Display data set information               |
|                         Allocate, rename, or delete entire data set|
|                         Catalog or uncatalog data set              |
|   3 MOVE/COPY - Move, copy, or promote members or data sets        |
|   4 DSLIST    - Data set list:                                     |
|                         Print or display list of data set names    |
|                         Print or display VTOC information          |
|   5 RESET     - Reset statistics for members of ISPF library       |
|   6 HARDCOPY  - Initiate hardcopy output                           |
|   7 VTOC      - Display or print VTOC entries for a DASD volume    |
|   8 OUTLIST   - Display, delete, or print held job output          |
|   9 COMMANDS  - Create/change an application command table         |
|  10 CONVERT   - Convert old format menus/messages to new format    |
|  11 FORMAT    - Format definition for formatted data Edit/Browse   |
```

Figure 7. Choosing DATASET from UTILITY SELECTION MENU

Note: In SPF, instead of typing 3 <enter> at the SPF PRIMARY OPTION MENU and 2 <enter> at the UTILITY SELECTION MENU, you can combine these steps by entering 3.2 on the option line of the SPF PRIMARY OPTION MENU. This approach can be used with any option that has a sub-menu.

Once you are within option 3.2, you must specify the name and characteristics of the data set you wish to allocate.

EXAMPLE

Suppose that you wish to allocate a partitioned data set named CSLSU.PROGRAM.PASCAL. The data set is to contain numerous Pascal programs, all of which are to be relatively short.

To accomplish the data set allocation, you must complete two screens. The first requires that you choose the A (allocate) option and specify the data set name as shown in Figure 8. The New Line (bent arrow) key is used to move the cursor to each subsequent data entry position.

```
| ---------------------- DATA SET UTILITY ------------------------  |
| OPTION ===> A                                                      |
|                                                                    |
|   A - Allocate new data set         C - Catalog data set           |
|   R - Rename entire data set        U - Uncatalog data set         |
|   D - Delete entire data set                                       |
|   blank - data set information                                     |
|                                                                    |
| ISPF LIBRARY                                                       |
|    PROJECT ===> CSLSU                                              |
|    GROUP   ===> PROGRAM                                            |
|    TYPE    ===> PASCAL                                             |
|                                                                    |
| OTHER PARTITIONED OR SEQUENTIAL DATA SET:                          |
|     DATA SET NAME     ===>                                         |
|     VOLUME SERIAL     ===>        (If not cataloged, required for  |
|                                    option "C")                     |
|     DATA SET PASSWORD ===>        (If password protected)          |
```

Figure 8. Specifying Data Set Name (first screen to complete when allocating a data set)

After you press <enter>, the next display screen is similar to Figure 9. On this screen, you must specify the attributes of the data set you wish to allocate. Throughout SPF, you move the cursor from field to field with the New Line key, typing over information already in the spaces if they are not the values you wish. The values in Figure 9 are appropriate for our example data set, CSLSU.PROGRAM.PASCAL.

```
| ------------------ ALLOCATE NEW DATA SET ----------------------  |
| COMMAND ===>                                                     |
|                                                                  |
| DATA SET NAME: CSLSU.PROGRAM.PASCAL                              |
|                                                                  |
|   VOLUME SERIAL      ===> USER08    (Blank for authorized vol)   |
|   GENERIC UNIT       ===>           (Generic group name or addr) |
|   SPACE UNITS        ===> TRKS      (BLKS, TRKS, or CYLS)        |
|   PRIMARY QUANTITY   ===> 10        (In above units)             |
|   SECONDARY QUANTITY ===> 0         (In above units)             |
|   DIRECTORY BLOCKS   ===> 5         (Zero for sequential DS)     |
|   RECORD FORMAT      ===> FB                                      |
|   RECORD LENGTH      ===> 80                                      |
|   BLOCK SIZE         ===> 6320                                    |
```

Figure 9. Specifying Data Set Characteristics (second screen to complete when allocating a data set)

The meanings of each of the categories in the ALLOCATE NEW DATA SET panel as shown in Figure 9 deserve some discussion.

VOLUME SERIAL refers to the particular disk pack which your data is to be stored on. The system (SMS) will pick the volume for you.

GENERIC UNIT refers to a type of data storage device (such as DISK) and may typically be ignored.

SPACE UNITS refers to the measurement unit to be used in allocating disk space: blocks, tracks, or cylinders. A track (TRKS or TRACK), the unit typically chosen, is one concentric circle on one platter of a disk pack. On IBM 3380 disks (or equivalent) one track can store 47,476 bytes of data (or nearly 600 80-byte records).

PRIMARY QUANTITY refers to the initial allocation amount, such as 10 tracks

SECONDARY QUANTITY refers to the amount TSO will add to your allocation if your primary allocation amount is insufficient. It is suggested that you put a zero here to simplify your data set handling.

DIRECTORY BLOCKS refers to the space which must be set aside if the data set being allocated is a partitioned data set. As a rule of thumb, you will need one directory block for every four members of your data set. In our example above, five directory blocks are allocated, which implies that we expect to have 20 or fewer members in the data set. If the data set being allocated is any type other than partitioned, the DIRECTORY BLOCKS quantity should be zero.

RECORD FORMAT, RECORD LENGTH, and BLOCK SIZE refer to the manner in which individual records are to be stored and retrieved. Typical values for each are defined below.

```
 RECORD FORMAT -- FB  (fixed block)
                 FBA (fixed block but including
                      a carriage control column)

 RECORD LENGTH -- 80 for input files
                 133 for output files

 BLOCK SIZE    -- Even multiples of the RECORD LENGTH
                 Suggest:  6320 for input files
                           7448 for output files
```

After the data set characteristics have been specified and the enter key pressed, SPF will again display the DATA SET UTILITY screen. At this point, the data set CSLSU.PROGRAM.PASCAL has been allocated and cataloged (listed in the master catalog). A message at the top right of the screen states DATA SET ALLOCATED. To return to the SPF PRIMARY OPTION MENU, press the PF4 key.

Note: The easiest way to allocate a new data set is to first "show" the characteristics of a similar data set which has already been allocated and then to modify its characteristics to fit your situation. This is accomplished by listing the name of the "similar" data set on the DATA SET UTILITY screen, leaving the OPTION area blank, and then pressing the enter key.

Note: The SPF 3.2 option not only allows you to allocate a data set, but also allows you to Rename (R), Delete (D), Catalog (C), and Uncatalog (U) a data set. As such, it is one of the most frequently used SPF panels.

EXERCISE 4: Allocate a 10-track partitioned data set that could contain up to 20 Pascal programs. Use your logonid and the same names for the GROUP and TYPE fields as were used in the EXAMPLE earlier in this section (Figures 8 and 9).

8. Creating (Editing) a Data Set

To create (or edit) a program data set, choose option 2 from the SPF PRIMARY OPTION MENU as shown in Figure 10.

```
| ------------ LSU SNCC's ISPF/PDF PRIMARY OPTION MENU ------------ |
| OPTION ===> 2                                                      |
|                                                   USERID - CSLSU   |
|   0  ISPF PARMS   - Specify terminal/user parms   TIME   - 14:07   |
|   1  BROWSE       - Display source data/output    TERM   - 3278    |
|   2  EDIT         - Create or change source data  PF KEYS- 12      |
|   3  UTILITIES    - Perform utility functions     PREFIX - CSLSU   |
|   4  FOREGROUND   - Invoke lang. proc. in foreground               |
|   5  BATCH        - Submit job for language processing             |
|   6  COMMAND      - Enter TSO command or CLIST                     |
|   7  DIALOG TEST  - Perform dialog testing                         |
|   8  LM UTILITIES - Perform library management utility functions   |
|   C  CHANGES      - Display summary of changes for this release    |
|   T  TUTORIAL     - Display information about TSO/SPF              |
|   X  EXIT         - Terminate ISPF using log and list defaults     |
|                                                                    |
| These are extensions to the standard ISPF and ISPF/PDF.            |
|   I  IOF          - Interactive Output Facility ("I." for menu).   |
|   L  LSU          - LSU/SNCC maintained extensions.                |
|   S  SPECIAL      - Your Department's extensions - panel DEPTMAIN  |
|   U  USER         - Your own extensions          - panel USERMAIN  |
```

Figure 10. Choosing EDIT from SPF PRIMARY OPTION MENU

The EDIT - ENTRY PANEL, shown in Figure 11, will be displayed. If you have used the edit option previously and specified an 'ISPF Library', note that the screen contains the data set name of the last data set referenced. Using the New Line key to move the cursor to the appropriate screen positions, you type in the complete data set name.

EXAMPLE

Suppose the first Pascal program that goes into
CSLSU.PROGRAM.PASCAL has a member name of EXAMPLE1 (Figure 11).

```
|  ---------------------- EDIT - ENTRY PANEL ----------------------  |
|  COMMAND ===>                                                      |
|                                                                    |
|  ISPF LIBRARY:                                                     |
|    PROJECT ===> CSLSU                                              |
|    GROUP   ===> PROGRAM    ===>         ===>         ===>          |
|    TYPE    ===> PASCAL                                             |
|    MEMBER  ===> EXAMPLE1        (Blank for member selection list)  |
|                                                                    |
|  OTHER PARTITIONED OR SEQUENTIAL DATASET:                          |
|    DATA SET NAME  ===>                                             |
|    VOLUME SERIAL  ===>          (If not cataloged)                 |
|                                                                    |
|  DATA SET PASSWORD ==>          (If password protected)            |
|  PROFILE NAME     ===>          (Blank defaults to data set type)  |
|  INITIAL MACRO    ===>                                             |
|  MIXED MODE       ===>          (Specify YES or NO)                |
|  FORMAT NAME      ===>                                             |
```

Figure 11. Specifying the File to be Edited (Created)
(first screen to complete when creating data set 'CSLSU.PROGRAM.PASCAL(EXAMPLE1)')

After typing the data set name and pressing the <enter> key, you will be taken to a clear SPF EDIT screen, such as the one shown in Figure 12. Take a close look at the screen layout. There is some heading information near the top of the screen (including a COMMAND line), a TOP OF DATA line, a largely empty area (called the "data area"), and a BOTTOM OF DATA line.

```
|  EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) ------- COLUMNS 001 0072  |
|  COMMAND ===>                                   SCROLL ===> CSR   |
|  ****** ****************** TOP OF DATA **************************  |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ''''''                                                            |
|  ****** **************** BOTTOM OF DATA *************************  |
```

Figure 12. Clear SPF EDIT Screen

The name of the data set and member being edited are shown near the top left of the screen. The columns currently being shown are listed near the top right of the screen. Directly under the column numbers, SCROLL ===> CSR is shown. This tells how far up or down in the data set you will scroll when you press the PF8 (forward) or PF7 (backward) keys. In Figure 12, the current setting is CSR, but it could be set at one PAGE (to move a full screen each time) or HALF (to indicate a half-screen move each time). CSR (cursor) indicates movement within the data set to the position of the cursor within the data area--or by a full screen if the cursor is outside the screen's data area. At the left side of the data area, six apostrophes begin each line. This area is called the "prefix area." Actual data values cannot be entered in the column adjacent to the apostrophes.

To enter a program, you should position the cursor in the
first data entry position, key in the first line of your
program or data, and then press the New Line and Tab keys in
succession to move the cursor to the beginning of the next
line. You repeat the sequence until you have completed keying
in the program--then press the <enter> key.

EXAMPLE

Suppose the first Pascal program is a sequence that adds
two numbers and then writes the two numbers and their sum.
Our example uses the data values of 12 and 7.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
| COMMAND ===>                                     SCROLL ===> CSR   |
| ****** ****************** TOP OF DATA ***************************   |
| '''''' PROGRAM ADD(INPUT,OUTPUT);                                  |
| '''''' VAR FIRST,SECOND,SUM: INTEGER;                              |
| '''''' BEGIN                                                       |
| '''''' READ (FIRST,SECOND);                                        |
| '''''' SUM = FIRST + SECOND;                                       |
| '''''' WRITE (FIRST,SECOND,SUM);                                   |
| '''''' END.                                                        |
| '''''' $ENTRY                                                      |
| '''''' 12   7                                                      |
| ''''''                                                             |
| ''''''                                                             |
| ''''''                                                             |
| ****** **************** BOTTOM OF DATA **************************   |
```

Figure 13. Example Program Using the SPF EDIT Screen

After you have finished entering the program, you press the PF12 key to move the cursor to the COMMAND line. To SAVE your program, there are five options:

(1) Type SAVE <enter> on the COMMAND line. This saves your program and leaves you in the SPF editor so that you can make further changes to your program (or data) if you wish.

(2) Type END <enter> on the COMMAND line. This saves your program and returns you to the EDIT - ENTRY PANEL screen (Figure 11).

(3) Press the PF3 key. The default setting of the PF3 key has the same meaning as typing END.

(4) Type RETURN <enter> on the COMMAND line. This saves your program and returns you to the SPF PRIMARY OPTION MENU screen (Figure 4).

(5) Press the PF4 key. The default setting of the PF4 key has the same meaning as typing RETURN <enter>.

Note: If for some reason you decide that you do not want to keep the changes you have made (since the last time you saved the file), you can cancel all those changes by typing CANCEL <enter> on the COMMAND line.

Note: If you try to SAVE a data set member from within SPF 2 and get a D37 error message instead, you are being told that your data set has insufficient space to save the member. This probably means that your partitioned data set needs to be compressed. To compress the data without losing your most recent set of changes, "split the screen" (which allows you to work with two parts of SPF at the same time) by pressing the PF2 key and perform the data set compression process as described in Section XIII. After the data set has been compressed, you will probably be able to save your changes without further difficulties. The steps required to split the screen are defined concurrently with the discussion of IOF in Section XII. You need not read either section at this point, but you should be aware that you can free yourself from this dilemma if it occurs.

Sometime during your program entry session, you may have pressed the <enter> key. If so, you probably noticed that the screen suddenly changed rather dramatically. The apostrophes to the left of the data area either changed to numbers (if data had been entered on that particular line) or vanished altogether. A different approach must be used when entering data in this mode.

One option is to insert one blank line at a time into the data set. This approach is typically a little slower since it causes each line to be processed individually by the computer (rather than having several lines of code processed all at once as in the approach described above). To accomplish this, simply place an I anywhere in the "prefix area" and press <enter>.

EXAMPLE

Suppose that you wanted to add the Pascal statement WRITE ('...THE OUTPUT...'); just after the SUM := FIRST + SECOND; statement. Figure 14 shows the first step in the line insertion process.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
| COMMAND ===>                                     SCROLL ===> CSR   |
| ****** ***************** TOP OF DATA **************************    |
| 000100 PROGRAM ADD(INPUT,OUTPUT);                                  |
| 000200 VAR FIRST,SECOND,SUM: INTEGER;                              |
| 000300 BEGIN                                                       |
| 000400 READ (FIRST,SECOND);                                        |
| I 0500 SUM := FIRST + SECOND;                                      |
| 000600 WRITE (FIRST,SECOND,SUM);                                   |
| 000700 END.                                                        |
| 000800 $ENTRY                                                      |
| 000900 12   7                                                      |
| 001000                                                             |
| 001100                                                             |
| 001200                                                             |
| ****** **************** BOTTOM OF DATA *************************   |
```

Figure 14. First Step in Line Insert Process
(Note "I" in prefix area of the fifth data entry line.)

One blank line is then inserted in the program just below the line marked with an I and the cursor is placed in the same column as the first character in the previous line as shown in Figure 15.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
| COMMAND INPUT ===>                               SCROLL ===> CSR   |
| ************************ TOP OF DATA **************************    |
| 000100 PROGRAM ADD(INPUT,OUTPUT);                                  |
| 000200 VAR FIRST,SECOND,SUM: INTEGER;                              |
| 000300 BEGIN                                                       |
| 000400 READ (FIRST,SECOND);                                        |
| 000500 SUM := FIRST + SECOND;                                      |
| ''''''  _                                                          |
| 000600 WRITE (FIRST,SECOND,SUM);                                   |
| 000700 END.                                                        |
| 000800 $ENTRY                                                      |
| 000900 12   7                                                      |
| 001000                                                             |
| 001100                                                             |
| 001200                                                             |
| *********************** BOTTOM OF DATA *************************   |
```

Figure 15. Result of Placing an I in the Prefix Area

A second option is to insert several blank lines and then follow the same procedure as was originally described in terms of program data entry. For example, to insert 5 blank lines at the same place that one blank line was inserted above, simply type I5 in the prefix area instead of I and press <enter>.

Note: The Waterloo Pascal compiler requires that the data set being compiled not be line numbered. Therefore, before saving the data set, you should issue the UNNUM command from the EDIT - ENTRY PANEL COMMAND line (e.g., COMMAND ===> UNNUM <enter>). This command will un-number the lines in the data set member.
To tell whether a data set member is numbered or unnumbered, you can look at the line numbers in the prefix area. If the the first line number is 000001, then the data set member is in UNNUM mode. If the first line number is 000100, then it is in NUM mode.

Regardless of how changes are made to the program, when all the changes are complete, save the modifications using any one of the five options mentioned previously in this section.

Note: Numerous helpful "prefix area" commands (formally called "line commands") are available in SPF to aid the data set editing process. These include insert (I), delete (D), copy (C), move (M), and repeat (R), as well as several others. A more complete list plus descriptions is included in Appendix A.

Note: Numerous helpful "primary" commands (those commands which can be typed on the COMMAND Line, as opposed to the prefix area) are available in SPF to also aid the data set editing process. These include FIND (F) and CHANGE (C), as well as several others. A more complete list plus descriptions is included in Appendix B.

EXERCISE 5: Type in the example program shown in Figure 13 in your data set (i.e., your logonid.PROGRAM.PASCAL) under the member name EXAMPLE1. After you have finished entering the program, add the Pascal statement:

```
    WRITE ('...THE OUTPUT...');
```

on the line just after the SUM := FIRST + SECOND; statement. Save the data set member.

9. Printing a Data Set

To get a printout (paper copy) of your program, you can use the TSO command PRINTDA. There are two approaches to getting into the TSO command mode.

(1) Exit from SPF by typing =X <enter> from the COMMAND line within SPF.

(2) Press PF4 to return to the SPF PRIMARY OPTION MENU and then select option 6 (COMMAND). SPF 6 is actually TSO-under-SPF.

This document assumes you chose the second approach to get into the TSO command mode (i.e, that you will be working from SPF option 6).

EXAMPLE

Suppose you wanted to print a copy of the program stored in 'CSLSU.PROGRAM.PASCAL(EXAMPLE1)'.

```
| -------------------- TSO COMMAND PROCESSOR ----------------------  |
| ENTER TSO COMMAND OR CLIST BELOW:                                   |
|                                                                     |
| ===> PRINTDA PROGRAM.PASCAL(EXAMPLE1)                               |
|                                                                     |
|                                                                     |
```

Figure 16. Print a Copy of the Contents of 'CSLSU.PROGRAM.PASCAL(EXAMPLE1)'

Recall that typing PROGRAM.PASCAL(EXAMPLE1) is the same as typing 'CSLSU.PROGRAM.PASCAL(EXAMPLE1)' since TSO will automatically place your logonid at the beginning of the data set name if you type in a name without it being enclosed in apostrophes. If you want to print the entire partitioned data set, you could type PRINTDA PROGRAM.PASCAL <enter>.

EXERCISE 6: Print a copy of the program created in Exercise 5.

10. Running a Program

Running a program is actually a three-step process. To be "run," a program must be compiled, linked, and executed. The compile step translates the program statements (called "source code") to machine language (which the computer can understand). The link step combines (links) the compiled, machine language version of the program to the language libraries for the language being used (e.g., the Pascal language libraries) in readiness for processing. Finally, the linked package (code + libraries) is executed.
There are two ways to run a program using SPF, in the foreground or in the background. Foreground processing implies interactive processing, which typically means that you enter a single command to cause your program to be processed. Background processing implies batch processing. Batch processing requires that you use Job Control Language (JCL) statements to process your program.
There are advantages to each type of program processing. For example, interactive (foreground) programs are appropriate when data input values need to be entered in response to program prompts, such as those required by automatic teller machines. Batch (background) processing is particularly appropriate when the program is to wait until non-demand time before being run.

FOREGROUND PROCESSING:
The most appropriate place to issue commands which invoke foreground program processing is from TSO. You would move to SPF option 6 (COMMAND), if you wanted to run a program in the foreground.
To invoke the Waterloo Pascal compiler, you would type the command WPASCAL followed by the name of the data set containing the Pascal program and press <enter>. Program output is sent only to the screen when the command is issued in this form.

EXAMPLE

Suppose you wish to run program ADD (Figure 15) using foreground processing with the results displayed on the screen only. Figure 17 shows the command required.

```
| -------------------- TSO COMMAND PROCESSOR ---------------------- |
| ENTER TSO COMMAND OR CLIST BELOW:                                 |
|                                                                   |
| ===> WPASCAL  PROGRAM.PASCAL(EXAMPLE1)                            |
|                                                                   |
|                                                                   |
```

Figure 17. To Run Program ADD in the Foreground with Data Displayed on Screen Only

After the WPASCAL command is issued, your program will be displayed on the screen. When the screen prompts you with "DEBUG?," type GO and press <enter>.
To have the program output saved, you must send the output to a data set--which must already exist (i.e., must have been allocated via SPF option 3.2; see Section VIII for details).

This is accomplished by including the PGMLIST option, when issuing the WPASCAL command as shown in Figure 18.

EXAMPLE

Suppose you wish to run program ADD in foreground mode and wish to save the results in a sequential data set named PROGRAM.OUT. In this example, it is assumed that you have already allocated the sequential data set CSLSU.PROGRAM.OUT, making sure that you used '0' for the number of DIRECTORY BLOCKS since the data set is sequential. (Remember, Figures 8 and 9 show the data allocation process.)

```
| ------------------- TSO COMMAND PROCESSOR ---------------------  |
| ENTER TSO COMMAND OR CLIST BELOW:                                 |
|                                                                   |
| ===> WPASCAL PROGRAM.PASCAL(EXAMPLE1) PGMLIST(DSN(PROGRAM.OUT))   |
|                                                                   |
|                                                                   |
```

Figure 18. To Run Program ADD in the Foreground with the Output Saved in Data Set CSLSU.PROGRAM.OUT

Again, your program will be displayed on the screen. When you are prompted with "DEBUG?," type GO and press <enter>. After the above command has been processed, you may browse the output data set CSLSU.PROGRAM.OUT via SPF option 1.

Note: At this point, BROWSE (option 1) has not been discussed. It is easy to use. The first panel is similar to the EDIT panel (Figure 11). After you have specified the data set you wish to browse (e.g., CSLSU.PROGRAM.OUT), you may then view the contents of the data set using the scroll keys (PF7 and PF8). No changes to the data set can be made using the BROWSE option.

BACKGROUND PROCESSING:
Batch jobs are typically submitted from SPF option 2, EDIT. Before a program can be submitted, appropriate Job Control Language (JCL) statements must be added to the data set. Once the JCL has been added to the data set, you may submit the job for processing by typing SUBMIT <enter> on the COMMAND line.

EXAMPLE

Suppose logonid CSLSU (Department number - 1234, Project number - 56565) wishes to run program ADD in the background (i.e., as a batch job).

First, you must add the JCL using EDIT (SPF 2) so that the data set member looks like the one in Figure 19. The JCL used in this example is explained in Appendix E.
Then you submit the job for background processing by typing SUBMIT <enter> on the COMMAND line as is also shown in Figure 19.

Note: This file must also be UNNUMbered before being SUBMITted for processing to the Waterloo Pascal compiler. As stated previously, this is accomplished by typing UNNUM on the COMMAND line and pressing the <enter> key.

Note: Output from this program will automatically be held
(because of the MSGCLASS=S parameter). Section XII
discusses how to view and print held output.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
| COMMAND ===> SUBMIT                              SCROLL ===> CSR   |
| ************************* TOP OF DATA **************************** |
| 000001 //CSLSU1  JOB (1234,56565),'CSLSU - STUBBS',NOTIFY=CSLSU,   |
| 000002 //            CLASS=Q,MSGCLASS=S                            |
| 000003 //************ SAMPLE WATERLOO PASCAL JOB *************     |
| 000004 //       EXEC WPASCAL                                       |
| 000005 $JOB                                                        |
| 000006 PROGRAM ADD(INPUT,OUTPUT);                                  |
| 000007 VAR FIRST,SECOND,SUM: INTEGER;                              |
| 000008 BEGIN                                                       |
| 000009    READ (FIRST,SECOND);                                     |
| 000010    SUM := FIRST + SECOND;                                   |
| 000011    WRITE (FIRST,SECOND,SUM);                                |
| 000012 END.                                                        |
| 000013 $ENTRY                                                      |
| 000014 12 7                                                        |
| 000015 $$                                                          |
| 000016 //                                                          |
| *********************** BOTTOM OF DATA *************************** |
```

Figure 19. Submitting Program ADD to Run in the Background

EXERCISE 7: Run the program in logonid.PROGRAM.PASCAL(EXAMPLE1)
in the foreground. Send the output to the screen
only.
EXERCISE 8: Run the program in logonid.PROGRAM.PASCAL(EXAMPLE1)
in the foreground. Save the output in an existing
sequential data set named logonid.PROGRAM.OUT.
Note: For a refresher on how to allocate the
sequential data set logonid.PROGRAM.OUT, see
Section VIII. Make sure that you specify 0
for the DIRECTORY BLOCK attribute.
EXERCISE 9: Run the program in logonid.PROGRAM.PASCAL(EXAMPLE1)
in background mode.

Note: When adding the JCL to your program's
source code, use your logonid and
include the NOTIFY= yourlogonid option on
your JOB statement.
Note: You should have been given your department
and project numbers at the same time you
received your logonid and password. If
not, you must use LSU's security system
software called ACF2 to find these numbers.
Appendix D explains how to use ACF2 to find
Accounting Information (which includes your
department and project numbers).

11. Viewing and Printing Output Data Sets and Program Results

After a background job has been submitted for processing, the
easiest way to follow the job's progress--to view program
results, to examine the job's JCL, and to view possible
program errors--is to split the screen (using the PF2 key) and
then type I <enter> on the COMMAND line of the SPF PRIMARY
OPTION MENU. Figure 20 shows the way your screen will look
after you have pressed the PF2 key to split the screen.

Note: To swap screens while in an SPF split screen session,
you need only press the PF9 key. PF9 is a toggle key
which allows you to move back and forth between the
two screens.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------ LSU SNCC's ISPF/PDF PRIMARY OPTION MENU ------------  |
| OPTION ===> I                                                      |
|                                                    USERID - CSLSU  |
|   0  ISPF PARMS   - Specify terminal/user parms    TIME   - 14:07  |
|   1  BROWSE       - Display source data/output     TERM   - 3278   |
|   2  EDIT         - Create or change source data   PF KEYS- 12     |
|   3  UTILITIES    - Perform utility functions      PREFIX - CSLSU  |
|   4  FOREGROUND   - Invoke lang. proc. in foreground                 |
|   5  BATCH        - Submit job for language processing               |
|   6  COMMAND      - Enter TSO command or CLIST                       |
|   7  DIALOG TEST  - Perform dialog testing                           |
|   8  LM UTILITIES - Perform library management utility functions    |
|   C  CHANGES      - Display summary of changes for this release      |
|   T  TUTORIAL     - Display information about TSO/SPF                |
|   X  EXIT         - Terminate ISPF using log and list defaults       |
|                                                                    |
| These are extensions to the standard ISPF and ISPF/PDF.            |
|   I  IOF          - Interactive Output Facility ("I." for menu).    |
|   L  LSU          - LSU/SNCC maintained extensions.                  |
|   S  SPECIAL      - Your Department's extensions - panel DEPTMAIN  |
|   U  USER         - Your own extensions          - panel USERMAIN  |
```

Figure 20. Invoking IOF (Interactive Output Facility)
from a Split Screen Session

After typing I and pressing <enter>, the IOF JOB LIST MENU,
shown in Figure 21, will be displayed. You choose the job you
wish to view by typing on the COMMAND line the number displayed
beside the JOBNAME, then press the <enter> key. Figure 21
shows the IOF JOB LIST MENU when only one background job has
been run.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| --------------------- IOF JOB LIST MENU ----------------------     |
| COMMAND ===> 1                                  SCROLL ===> SCREEN |
| -------------------------- OUTPUT JOBS -------------------------   |
| ------JOBNAME---JOBID--ACCT-HELD-LOGONID--AFTER--DEST---GRPS-REC-  |
| _   1 CSLSU1    J8190            CSLSU            STUBBS             |
|                                                                    |
|                                                                    |
```

Figure 21. IOF JOB LIST MENU, Choosing to View Job No. 1

After job number 1 has been selected for viewing, the IOF JOB
SUMMARY, shown in Figure 22, is displayed. The JOBNAME, JOBID
number, time and day the job was run, and the default print
DESTination are shown near the top of the table. The Return
Code (RC) (i.e., the degree of error severity) for the GO
(execution) step and the cataloged PROCedure (WPASCAL in this
case) used to process the job are shown next. The bottom part
of the table lists specific information about each job step.
Each part can be viewed by typing an S in the far left column

of the table beside the particular DDNAME you want to view.

EXAMPLE

   Suppose you wish to view the program output (i.e., SYSPRINT
   step) from the example batch job submitted in Section XI.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------------------ IOF JOB SUMMARY ----------------------- |
| COMMAND ===>                                   SCROLL ===> SCREEN |
| -------------------------- OUTPUT JOBS ------------------------- |
| ---JOBNAME---JOBID--STATUS---RAN/RECD--DAY-----DEST---COPIES----- |
|    CSLSU1    J8190  OUTPUT   13:02   9/04/86   STUBBS    1        |
| --RC--STEP----PROCSTEP--PROC------COMMENTS---------------------- |
|    0  GO                WPASCAL                                  |
| --------DDNAME--STEP----STAT-ACT-GRP-CLS-RECS--DEST---FORMS-COPY- |
| _    1  LOG      *      HELD          S    11         1380        |
| _    2  JCL      *      HELD          S    15         1380        |
| _    3  MESSAGES *      HELD          S    38         1380        |
| S    4  SYSPRINT *      HELD          S    17         1380        |
```

Figure 22. IOF JOB SUMMARY, Program Output Available for Viewing

Once a file has been selected, you may scroll through it by
using the PF7 (UP) and PF8 (DOWN) keys. You may also move
directly to the next file in the list by typing N <enter> on
the COMMAND line within the particular job STEP or to the
previous file in the list by typing PR <enter> on the COMMAND
line within the particular job STEP.
IOF is especially handy because it not only allows you to
view program output, but also allows you to print one or more
output files at different locations.

EXAMPLE

   Suppose logonid CSLSU wishes to print the program output
   from the example batch job discussed in Section XI at the
   STUBBS Hall output facility.

Since CSLSU's default print DESTination is STUBBS (as shown
under DEST in Figure 23), it is an easy process to print one
or more sections of the program. Simply type an N beside the
DDNAME or DDNAMEs you are interested in printing and press
<enter>. The sections which have an N by their DDNAME will be
printed at the default print destination. Figure 23 shows
that the JCL and SYSPRINT sections are to be printed.
The N command, which is short for sNap, is the simplest IOF
print command. Unfortunately, it does not work if the desired
print location is something other than the default.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------------------ IOF JOB SUMMARY -----------------------  |
| COMMAND ===>                                  SCROLL ===> SCREEN  |
| -------------------------- OUTPUT JOBS -------------------------  |
| ---JOBNAME---JOBID--STATUS---RAN/RECD--DAY-----DEST---COPIES-----  |
|    CSLSU1    J8190  OUTPUT   13:02   9/04/86   STUBBS    1         |
| --RC--STEP----PROCSTEP--PROC------COMMENTS----------------------  |
|    0  GO                WPASCAL                                    |
| --------DDNAME--STEP----STAT-ACT-GRP-CLS-RECS--DEST---FORMS-COPY-  |
| _    1  LOG      *      HELD          S    11          1380          |
| N    2  JCL      *      HELD          S    15          1380          |
| _    3  MESSAGES *      HELD          S    38          1380          |
| N    4  SYSPRINT *      HELD          S    17          1380          |
```

Figure 23. IOF JOB SUMMARY, Printing JCL and SYSPRINT at STUBBS
(Where STUBBS is the Default DESTination)

EXAMPLE

Suppose logonid CSLSU wishes to print the program output
from the example batch job discussed in Section XI at the
output facility in CEBA.

A two-step process is required to accomplish this. First, the
DEST parameter value on the line beside SYSPRINT must be
changed to CEBA since the default DESTination is STUBBS and
the CLS (short for MSGCLASS) parameter value on the line
beside SYSPRINT must be changed to A as in Figure 24. Press
the <enter> key to make that change.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------------------ IOF JOB SUMMARY -----------------------  |
| COMMAND ===>                                  SCROLL ===> SCREEN  |
| -------------------------- OUTPUT JOBS -------------------------  |
| ---JOBNAME---JOBID--STATUS---RAN/RECD--DAY-----DEST---COPIES-----  |
|    CSLSU1    J8190  OUTPUT   13:02   9/04/86   STUBBS    1         |
| --RC--STEP----PROCSTEP--PROC------COMMENTS----------------------  |
|    0  GO                WPASCAL                                    |
| --------DDNAME--STEP----STAT-ACT-GRP-CLS-RECS--DEST---FORMS-COPY-  |
| _    1  LOG      *      HELD          S    11          1380          |
| _    2  JCL      *      HELD          S    15          1380          |
| _    3  MESSAGES *      HELD          S    38          1380          |
| _    4  SYSPRINT *      HELD          A    17  CEBA   1380          |
```

Figure 24. IOF JOB SUMMARY, Changing SYSPRINT DEST to CEBA and CLS
Parameter to A (Step 1 of Printing Process
when Desired Print DESTination is Not the Default)

Next, type a P (Print) in the far left column of the table
beside the DDNAME SYSPRINT as shown in Figure 25 and press
<enter>. The SYSPRINT output file will then be printed at
CEBA.

Note: You can print more than one file at a time, remembering
that if the desired destination is not the default, you
must first make appropriate changes in the IOF SUMMARY
and then print the files. The steps are: (1) type the
printer DESTination for each file, change the CLS
parameter to A, and press <enter>; (2) type P next to

the DDNAME entry or entries you wish to print and press
<enter> again to have the sections printed.

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------------------ IOF JOB SUMMARY ------------------------- |
| COMMAND ===>                                   SCROLL ===> SCREEN  |
| -------------------------- OUTPUT JOBS --------------------------- |
| ---JOBNAME---JOBID--STATUS---RAN/RECD--DAY-----DEST---COPIES-----  |
|    CSLSU1    J8190  OUTPUT   13:02   9/04/86   STUBBS    1         |
| --RC--STEP----PROCSTEP--PROC------COMMENTS------------------------ |
|    0  GO                WPASCAL                                    |
| --------DDNAME--STEP----STAT-ACT-GRP-CLS-RECS--DEST---FORMS-COPY-  |
| _    1  LOG      *      HELD          S    11          1380        |
| _    2  JCL      *      HELD          S    15          1380        |
| _    3  MESSAGES *      HELD          S    38          1380        |
| P    4  SYSPRINT *      HELD MOD      A    17  CEBA   1380        |
```

Figure 25. IOF JOB SUMMARY, Printing SYSPRINT Output at CEBA
(Step 2 of Printing Process When Desired Print
DESTination is Not the Default)

When you have finished working with IOF, you simply type
=X <enter> on the COMMAND line (see Figure 26), which will end
this split screen session and will return you to your original
EDIT session on the first screen (see Figure 27).

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
|  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  |
| ------------------------ IOF JOB SUMMARY ------------------------- |
| COMMAND ===> =X                                SCROLL ===> SCREEN  |
| -------------------------- OUTPUT JOBS --------------------------- |
| ---JOBNAME---JOBID--STATUS---RAN/RECD--DAY-----DEST---COPIES-----  |
|    CSLSU1    J8190  OUTPUT   13:02   9/04/86   STUBBS    1         |
| --RC--STEP----PROCSTEP--PROC------COMMENTS------------------------ |
|    0  GO                WPASCAL                                    |
| --------DDNAME--STEP----STAT-ACT-GRP-CLS-RECS--DEST---FORMS-COPY-  |
| _    1  LOG      *      HELD          S    11          1380        |
| _    2  JCL      *      HELD          S    15          1380        |
| _    3  MESSAGES *      HELD          S    38          1380        |
| _    4  SYSPRINT *      HELD PRT      A    17  CEBA   1380        |
```

Figure 26. End the Split Screen Session by Issuing the '=X' command

```
| EDIT --- CSLSU.PROGRAM.PASCAL(EXAMPLE1) -------- COLUMNS 001 0072  |
| COMMAND ===>                                     SCROLL ===> CSR   |
| ************************* TOP OF DATA **************************   |
| 000001 //CSLSU1  JOB (1234,56565),'JOE USER',NOTIFY=CSLSU,         |
| 000002 //             CLASS=Q,MSGCLASS=S                           |
| 000003 //************ SAMPLE WATERLOO PASCAL JOB *************     |
| 000004 //       EXEC WPASCAL                                       |
| 000005 $JOB                                                        |
| 000006 PROGRAM ADD(INPUT,OUTPUT);                                  |
| 000007 VAR FIRST,SECOND,SUM: INTEGER;                              |
| 000008 BEGIN                                                       |
| 000009    READ (FIRST,SECOND);                                     |
| 000010    SUM := FIRST + SECOND;                                   |
| 000011    WRITE (FIRST,SECOND,SUM);                                |
| 000012 END.                                                        |
| 000013 $ENTRY                                                      |
| 000014 12 7                                                        |
| 000015 $$                                                          |
| 000016 //                                                          |
| *********************** BOTTOM OF DATA *************************   |
```

Figure 27. Original EDIT Session

Note: You could have printed all the files related to job
CSLSU1 by issuing the command PO CSLSU1 from within SPF
at option 6 (or outside SPF from a READY prompt) rather
than using the P command within IOF.

```
READY
PO  CSLSU1  <enter>
```

To minimize confusion concerning which job output you
wish to print, you should make it a habit to change the
jobname on the JOB statement each time that you submit a
batch job (e.g., CSLSU1, CSLSU2, CSLSU3, etc.).
The PO (for PrintOut) command is used to print "held"
job output. Job output was "held" (for viewing and
possible printing) in this example because the parameter
MSGCLASS=S was included on the JOB statement. To have
job output sent directly to the printer, you may change
the MSGCLASS parameter value to A (i.e., MSGCLASS=A)
on your JOB statement.

EXERCISE 10: Print the LOG, the JCL, the MESSAGES, and the
SYSPRINT program output (using Option I) for the
batch job you ran in Exercise 9. Use the
printer nearest you.

12. Compressing a Partitioned Data Set

In Section IV, the concept of partitioned data set compression
was discussed. Compression is the process of freeing the
unusable disk space allocated to a partitioned data set. To
compress a data set using SPF, you must first be within the
UTILITIES section (SPF 3), which is the choice displayed on
the SPF PRIMARY OPTION MENU shown in Figure 28. (Note: It is
the same SPF option used when allocating a data set (Figure 6).)

```
| ------------ LSU SNCC's ISPF/PDF PRIMARY OPTION MENU ------------  |
| OPTION ===> 3                                                     |
|                                                   USERID - CSLSU  |
|   0  ISPF PARMS   - Specify terminal/user parms    TIME   - 14:07  |
|   1  BROWSE       - Display source data/output     TERM   - 3278   |
|   2  EDIT         - Create or change source data   PF KEYS- 12     |
|   3  UTILITIES    - Perform utility functions      PREFIX - CSLSU  |
|   4  FOREGROUND   - Invoke lang. proc. in foreground                |
|   5  BATCH        - Submit job for language processing              |
|   6  COMMAND      - Enter TSO command or CLIST                      |
|   7  DIALOG TEST  - Perform dialog testing                          |
|   8  LM UTILITIES - Perform library management utility functions    |
|   C  CHANGES      - Display summary of changes for this release     |
|   T  TUTORIAL     - Display information about TSO/SPF               |
|   X  EXIT         - Terminate ISPF using log and list defaults      |
|                                                                   |
| These are extensions to the standard ISPF and ISPF/PDF.           |
|   I  IOF          - Interactive Output Facility ("I." for menu).   |
|   L  LSU          - LSU/SNCC maintained extensions.                 |
|   S  SPECIAL      - Your Department's extensions - panel DEPTMAIN  |
|   U  USER         - Your own extensions          - panel USERMAIN  |
```

Figure 28. Choosing UTILITIES Section from SPF PRIMARY OPTION MENU

After the UTILITIES section has been selected, the next step
is to choose option 1, LIBRARY, as shown in Figure 29.

```
| ------------------ UTILITY SELECTION MENU ---------------------- |
| SELECT OPTION ===> 1                                             |
|                                                                  |
|   1 LIBRARY    - Library utility:                                |
|                         Print index listing or entire data set   |
|                         Print, rename, delete, or browse members |
|                         Compress data set                        |
|   2 DATASET    - Dataset utility:                                |
|                         Display data set information             |
|                         Allocate, rename, or delete entire data set|
|                         Catalog or uncatalog data set            |
|   3 MOVE/COPY - Move, copy, or promote members or data sets      |
|   4 DSLIST    - Data set list:                                   |
|                         Print or display list of data set names  |
|                         Print or display VTOC information        |
|   5 RESET     - Reset statistics for members of ISPF library     |
|   6 HARDCOPY  - Initiate hardcopy output                         |
|   7 VTOC      - Display or print VTOC entries for a DASD volume  |
|   8 OUTLIST   - Display, delete, or print held job output        |
|   9 COMMANDS  - Create/change an application command table       |
|  10 CONVERT   - Convert old format menus/messages to new format  |
|  11 FORMAT    - Format definition for formatted data Edit/Browse |
```

Figure 29. Choosing LIBRARY Utilities from UTILITY SELECTION MENU

Once you are within option 3.1, you must type a C on the
COMMAND line and specify the name of the partitioned data set
you wish to compress.

EXAMPLE

Suppose that you wish to compress the partitioned data set
named CSLSU.PROGRAM.PASCAL. Figure 30 shows what you
should type before pressing <enter>.

```
| --------------------- LIBRARY UTILITY  -------------------------|
| OPTION ===> C                                                   |
|                                                                 |
|   blank - Display member list       B - Browse member           |
|   C - Compress data set             P - Print member            |
|   X - Print index listing           R - Rename member           |
|   L - Print entire data set         D - Delete member           |
|   I - Data set information          S - Data set information(short)|
|                                                                 |
| ISPF LIBRARY                                                    |
|    PROJECT ===> CSLSU                                           |
|    GROUP   ===> PROGRAM      ===>        ===>        ===>       |
|    TYPE    ===> PASCAL                                          |
|    MEMBER  ===>          (If option "P", "R", "D", or "B" selected)|
|    NEWNAME ===>          (If option "R" selected)               |
|                                                                 |
| OTHER PARTITIONED OR SEQUENTIAL DATA SET:                       |
|     DATA SET NAME     ===>                                      |
|     VOLUME SERIAL     ===>        (If not cataloged, required for |
|                                    option "C")                  |
|     DATA SET PASSWORD ===>        (If password protected)        |
```

Figure 30. Screen to Complete to Compress a Partitioned Data Set

After you press <enter>, you should receive a message that the compress was successful. To return to the SPF PRIMARY OPTION MENU, press the PF4 key.

Note: The SPF 3.1 option not only allows you to compress a data set, but also allows you to print the entire data set (L), to list data set information (I) and to Browse (B), Print (P), Rename (R), or Delete (D) a member of a data set. It is a frequently used SPF panel.

Note: Recall that if you get a D37 error message when trying to save a data set member from within SPF 2, you are being told that your data set has insufficient space to save the member. This probably means that your partitioned data set needs to be compressed. To compress the data set without losing your most recent changes, split the screen by pressing the PF2 key and perform the data set compression process as described in this section.

EXERCISE 11: Compress the partitioned data set that you created in EXERCISE 5.

13. TSO Initialization

Each time you logon to TSO, a start-up procedure is executed automatically. This procedure must be stored in a data set member named 'logonid.CLIST(BEGIN)', where 'logonid' is naturally your unique logonid. The procedure initializes your TSO session environment and connects you with important system libraries (i.e., system data sets). However, before the procedure can be executed, it must first be created.
You can create the data set and member by simply entering the TSO command shown below (from READY or SPF 6).
either USER01, USER02, ..., or USER10 (the names of the ten

```
COPY 'SYS2.SHARE.CLIST(BEGIN)' CLIST(BEGIN) SPACE(1) TRACK
    UNIT(PERM)
```

You should type the entire statement on one line and follow it by pressing the <enter> key. (When entering a TSO command, TSO will automatically move your cursor to the next line if you run out of room. That is, it is not necessary for you to be concerned with such matters as where the line ends.) You do not have to allocate this data set beforehand since it uses the COPY command and includes the SPACE, TRACK, and UNIT options. The TSO COPY command can allocate and copy into a data set.

Note: Student users of TSO (i.e., those with $logonids) will need to coordinate this step with their instructors, who may wish them to connect with a particular class CLIST rather than having their own. Even if student users are allowed to have their own CLIST, the TSO COPY statement shown above must be modified slightly for student logonids. The statement in this case, which should be entered on one line and followed by pressing the <enter> key, should be as follows:

```
   COPY 'SYS2.SHARE.CLIST(BEGIN)' 'classid.$logonid.CLIST(BEGIN)'
  SPACE(1)  TRACK  UNIT(PERM)
```

Note: A number of budgetary units have their own disk packs and should not use the long term PERM disk packs (although using USER77 and USER78, the general-usage, temporary disk packs is fine). These groups (and the names of their disk packs) are:

Arts and Sciences	-- ASCS01
Chemistry	-- CHEM01
Physics	-- PHYS01, PHYS02
Division of Research Services	-- DRS001
Experimental Statistics	-- STAT07, STAT08

EXERCISE 12: Create a BEGIN CLIST for yourself. Edit it so that instead of saying:

```
     TSOUSER 'NAME - UP TO 20 CHAR'
```

it will say your logonid and default printer destination (e.g., TSOUSER 'CSLSU - STUBBS'). What you put within apostrophes on the TSOUSER statement will be printed on the cover sheet of your printouts. Later, you may wish to add more to the BEGIN member of your CLIST, but for now, this is fine.

14. Logging Off

Before you can logoff TSO, you must first exit SPF. To exit SPF and return to TSO's READY prompt, you can either enter =X from the COMMAND line of any SPF option or successively press the PF3 key. Once READY appears on your screen, you know that you are out of SPF (though still in TSO) and can then logoff. This is accomplished simply by typing LOGOFF and pressing <enter>.

```
READY               <--- TSO prompt
LOGOFF <enter>        <--- user entry
```

Control returns to the standard LSU/SNCC menu screen (Figure 3).

EXERCISE 13. Logoff of TSO.

APPENDIX A -- COMMON EDIT "PREFIX" COMMANDS

To try out some of the following prefix commands, position the cursor in the line number (prefix) area on the left side of the screen while you are in SPF option 2 (EDIT), type the command and press <enter>.

I	Insert one line.
I3	Insert 3 lines.
D	Delete one line.
D3	Delete 3 lines.
DD	Delete a group of lines ending with the next line also marked with DD.
C	Copy one line before (B) or after (A) another line marked with an A or a B line command.
C2	Copy 2 lines (this line and the next line) before or after another line marked with an A or a B line command.
CC	Copy a group of lines beginning with this line and ending with the next line also Marked with CC before or after another line marked with an A or a B line command.
M	Same as C, except it moves the line rather than copying it.
M2	Same as C2, except it moves the lines instead of copying them.
MM	Same as CC, except it moves the group of lines.
R	Repeat a line.
R4	Repeat a line 4 times.
RR	Repeat a series of lines ending with the next line marked with RR.
RR2	Same as RR, but repeat the series of lines twice.
TE	Set up the screen for text entry. Press "new line" to move to the next line. Press <enter> to end text entry mode.
TS	Split the line at the point where the cursor is positioned when <enter> is pressed.
TF	Flow the text evenly to the line length of the data set.
TF70	Same as TF, but end each line at column 70 or before.
COLS	Display a line indicating column positions. Delete the COLS line with the D prefix area command.
BNDS	Display and allow changes to bounds.
TABS	Display and allow changes to tab settings.
)10	Shift the data on the line to the right 10 columns.
(10	Shift the data on the line to the left 10 columns.
))5	Shift the data on the group of lines ending in the next line marked with)) to the right 5 columns.
((5	Shift the data on the group of lines ending in the next line marked with ((to the left 5 columns.
>5	Shift the data - after the first blank - on the line to the right 5 columns.
<5	Shift the data - after the first blank - on the line to the left 5 columns.
>>5	Shift the data - after the first blank - on the group of lines ending in the next line marked with a >> to the right 5 columns.
<<5	Shift the data - after the first blank - on the group of lines ending in the next line marked with a << to the left 5 columns.

APPENDIX B -- COMMON EDIT "PRIMARY" COMMANDS

The following list of commands are entered on the primary COMMAND line while in EDIT. (Some of the commands may also be used on the COMMAND line elsewhere in SPF, such as in BROWSE.)

SAVE	Save the data set (or data set member) and remain in EDIT mode.
CANCEL	Exit EDIT, but do not save any changes made to the data set during the current EDIT session. (Note: You are actually editing a copy of the data set.)
PROFILE	Display the profile, or characteristics, of this data set. Once displayed, the characteristics may be

	changed by typing over them.
CAPS OFF	Change the profile for this data set so that subsequent characters are not converted to uppercase. The profile CAPS ON/OFF is automatically set depending on the ontents of a new data set.
NULLS ON	Modify the profile for this data set so that the end of every line is filled with nulls instead of blanks. This makes it much easier to insert characters within lines. Especially useful if data set includes text.
HEX ON/OFF	Change the profile for this data set to display/not display the hexadecimal characters for each character as well.
RESET	Remove extraneous information lines, such as the three lines of profile display or a column positions line.
FIND string	Find and display the next occurrence of the string. Surrounding apostrophes are necessary if the string contains any blanks or certain special characters (such as * or "). As is, the FIND does not distinguish capital letters (i.e., 'Xxx', 'XXX', and 'xxx' are the considered the same string). The FIND can be repeated by pressing the RFIND (repeat find) key (PF5).
F c'Xxx'	Find the next occurrence of the string 'Xxx'. Specifying a string as a character string enables FIND to distinguish between uppercase and lowercase letters (i.e., 'Xxx', 'XXX', 'xxx' are not considered the same string).
F xxx WORD	Find the next occurrence of the string 'xxx' that is not part of another word.
F xxx PREV	Find the previous occurrence of the string 'xxx'.
F xxx 1 7	Find the next occurrence of the string 'xxx' in columns 1 through 7. The string must be completely contained within these columns.
CHANGE xxx yyy	Change the next occurrence of the string 'xxx' to 'yyy'. Special characters are handled the same as for FIND. The CHANGE can be repeated by pressing the RCHANGE (repeat change) key (PF6).
C xx yy ALL	Change all occurrences of 'xx' to 'yy'. Be careful when doing this. (You might want to SAVE the data set before using this command.)
COPY mem	Copy a member of the same partitioned data set into this member. Use an A or a B line command (in prefix area) to indicate where the member should be placed.
COPY	Copy a part of another member within the same partitioned data set or copy all or part of a member of another partitioned data set or copy all or part of a sequential data set. When you press <enter> a "copy from" menu will be displayed for you to enter the name of the data set and possibly the member you want to copy and, optionally, the lines you want to copy from that data set or member. First, use an A or a B line command to indicate where the copied lines should be placed.
CREATE mem	Create a new member of this partitioned data set. Use the C (or CC) line command to indicate what to copy into the other member.
CREATE	Create a new member of this partitioned data set, copy into an existing member of any partitioned data set, or copy into a sequential data set. Use the C (or CC) line command to indicate what to copy into the other member or data set. When you press <enter> a menu will be displayed for you to enter the name of the data set and/or member you want to copy into. (The data set

	must already exist.)
SUBMIT	Submit a background job. The data set must contain appropriate Job Control Language. You will want to have NOTIFY=logonid on your JOB statement so that you will be notified when your job ends. You may also want to have MSGCLASS=S on your JOB statement. MSGCLASS=S will cause the output of your job to be placed in the held output queue so that you can view it from IOF (SPF option I), which was described earlier in this document.
TABS ALL/OFF	Turns on/off hardware tabbing so that the tabs you set via the TABS line command will/will not be in effect.

Press the HELP key (PF1) while in EDIT to refresh your memory whenever you wish.

Sequence numbers are automatically created (and renumbered when the data set is saved) unless you set the profile to NUMBERS OFF. The numbers are placed in the last eight positions for fixed length records; the first eight positions for variable length records.

APPENDIX C -- COMMON TSO COMMANDS

ALLOCATE (ALLOC)	Allocate a new data set or link an existing data set to a particular ddname (or file name) to be read by a program. Ex1. ALLOC DS(NEW.DATA) NEW UNIT(PERM) SPACE(10) BLOCK(6320) Ex2. ALLOCATE FILE(FT08F001) DATASET('CSLSU.DATA')
ALLOW	Grant ACF2 (security) authorization for another logonid to use your department and project numbers. Ex. ALLOW CSLSU
BLKSIZE	Determine optimal blocksizes for disk data sets. In the example below, 80 represents the data set's record length. Ex. BLKSIZE 80
CANCEL	Cancel one of your background jobs. Ex. CANCEL CSLSU1
COMPARE	Compare the contents of two data sets. List any differences. Ex. COMPARE 'CSLSU.VERSION1' 'CSLSU.VERSION2'
COMPRESS (C)	Compress a partitioned data set. Ex. C 'CSLSU.PDS'
CONC	Concatenate your CLIST library with SYS2.CMDPROC (the system CLIST library) or a user specified CLIST library. Ex. CONC
COPY	Copy a sequential data set, partitioned data set (PDS), or PDS member to a new or existing data set or PDS member. Ex. COPY OLD.DATA NEW.DATA
DSAT	Display data set attributes. Ex. DSAT 'CSLSU' Ex. DSAT CLIST
DUSER (DU)	Display free space on the USER0x volumes. Ex. DUSER
DVOL	Display free space on a disk volume. Ex. DVOL USER08
EDIT (E)	Edit a data set using the TSO Editor. (This is normally used only when SPF is not available on a terminal.) Ex. EDIT (BEGIN) CLIST
FREE	Free an allocated file. Ex. FREE DATASET('CSLSU.DATA')
FREEALL	Free all dynamically allocated files except for excluded (FA) ddname list.

	Ex. FREEALL
HELP (H)	Obtain online help information for a TSO command. Ex. HELP DSAT
IOF	Facility for viewing and printing "held" job output. Ex. IOF CSLSU1
LIST (L)	List a data set (or PDS member) at your terminal. Ex. LIST 'CSLSU.PGM.FORT'
MANUAL	Display list of printable documentation and, optionally, print a manual. (From SPF, use option L.DOC) Ex. MANUAL
MEMBERS (MEM)	Display names of members of a partitioned data set. Ex. MEMBERS PDS.DATA
MMDEL	Delete a member(s) from a PDS. Ex. MMDEL CLIST (MEM1,MEM2)
OQ	Display job status information. Ex. OQ CSLSU1 JOBNAME
OSDEL	Delete a disk data set. Ex. OSDEL PGM.DATA
PERMIT	Grant ACF2 (security) authorization for another logonid to access your data set(s). Ex. PERMIT CSLSU
PLOTDD	Allocate files for Benson plotting. Ex. PLOTDD
PRINTDA	Print or punch a disk data set. Ex. PRINTDA PGM.DATA
PRINTOUT (PO)	Move held job output to a disk data set and (optionally) print. Omit NOPRINT to print output. Omit KEEP to have job output purged from held output queue after printing. Ex. PRINTOUT CSLSU1 NOPRINT KEEP
REL	Release some of the unused tracks in a data set. In the example below, 10 tracks are released. Ex. REL PDS.DATA 10
RLSE	Release all unused space in a disk data set. Ex. RLSE 'CSLSU.PGM.FORT'
RENAME (REN)	Rename a disk data set. Ex. RENAME OLD.NAME NEW.NAME
SEARCH	Search a PDS for a specified string. Ex. SEARCH 'CSLSU.PGM.FORT' 'VECTOR'
SEND (SE)	Send a message to another logonid. Ex. SEND 'MESSAGE' USER(CSLSU)
SPF	Invoke SPF. Ex. SPF
TSOSORT	Sort a data set from TSO. Ex. TSOSORT FROM.DATA TO.DATA 'FIELDS=(1,4,CH,A)'
TSOUSER	Specify TSO user identification. Statement usually placed in BEGIN member of user's CLIST to denote print destination. Ex. TSOUSER 'CSLSU - STUBBS'
USA	Grant ACF2 (security) authorization to SNCC User Services to use your department and project numbers. (Not required for classwork logonids.) Ex. USA
USERS	Display logonids of users logged on to TSO. Ex. USERS
USP	Grant ACF2 (security) authorization to SNCC User Services to access your data sets. (Not required for classwork logonids.) Ex. USP

Application Program Processing Commands:

ASM	Assemble a VS Assembler program. Ex. Assemble the code stored in member EXP1 in data set CSLSU.PGM.ASM. User CSLSU is logged on. ASM is assumed to be the last data set qualifier. ASM PGM(EXP1) An object module will be written into data set CSLSU.PGM.OBJ. If the data set CSLSU.PGM.OBJ does not exist prior to this command, the data set created will be sequential. If the data set CSLSU.PGM.OBJ alreay exists prior to this command, the type (sequential or partitioned) will be the original type defined. If the existing data set is sequential, the object module will be written into the sequential data set. If the existing data set is partitioned, the object module will be written into member TEMPNAME in the existing partitioned data set. Existing data in a sequential data set or in member TEMPNAME of a partitioned data set will be overwritten.
ASMG	Assemble an Assembler G program. Ex. Assemble the code stored in member EXP1 in data set CSLSU.PGM.ASM. User CSLSU is logged on. ASM is assumed to be the last data set qualifier. ASMG PGM(EXP1) An object module will be written into data set CSLSU.PGM.OBJ. If the data set CSLSU.PGM.OBJ does not exist prior to this command, the data set created will be sequential. If the data set CSLSU.PGM.OBJ alreay exists prior to this command, the type (sequential or partitioned) will be the original type defined. If the existing data set is sequential, the object module will be written into the sequential data set. If the existing data set is partitioned, the object module will be written into member TEMPNAME in the existing partitioned data set. Existing data in a sequential data set or in member TEMPNAME of a partitioned data set will be overwritten.
ASMH	Assemble an Assembler H program. Ex. Assemble the code stored in member EXP1 in data set CSLSU.PGM.ASM. User CSLSU is logged on. ASM is assumed to be the last data set qualifier. ASMH PGM(EXP1) An object module will be written into data set CSLSU.PGM.OBJ. If the data set CSLSU.PGM.OBJ does not exist prior to this command, the data set created will be sequential. If the data set CSLSU.PGM.OBJ alreay exists prior to this command, the type (sequential or partitioned) will be the original type defined. If the existing data set is sequential, the object module will be written into the sequential data set. If the existing data set is partitioned, the object module will be written into member TEMPNAME in the existing partitioned data set. Existing data in a sequential data set or in member TEMPNAME of a partitioned data set will be overwritten.

FORT

Compile a FORTRAN G1 program.
Ex. Compile the FORTRAN source code stored in member
EXP1 in data set CSLSU.PGM.FORT. User CSLSU is
logged on. FORT is assumed to be the last data set
qualifier.

```
   FORT PGM(EXP1)
```

An object module will be written into member EXP1 in
data set CSLSU.PGM.OBJ, and the compiler output will
be written into member EXP1 in data set
CSLSU.PGM.LIST. If data set CSLSU.PGM.OBJ or
CSLSU.PGM.LIST already exists prior to issuing the
FORT command, it must be a partitioned data set. If
member name EXP1 already exists in data sets
CSLSU.PGM.OBJ and/or CSLSU.PGM.LIST, the existing
data will be overwritten.

FORTV

Compile a VS FORTRAN program.
Ex. Compile the FORTRAN source code stored in member
EXP1 in data set CSLSU.PGM.FORT. User CSLSU is
logged on. FORT is assumed to be the last data set
qualifier.

```
   FORTV PGM(EXP1)
```

An object module will be written into member EXP1 in
data set CSLSU.PGM.OBJ, and the compiler output will
be written into member EXP1 in data set
CSLSU.PGM.LIST. If data set CSLSU.PGM.OBJ or
CSLSU.PGM.LIST already exists prior to issuing the
FORT command, it must be a partitioned data set. If
member name EXP1 already exists in data sets
CSLSU.PGM.OBJ and/or CSLSU.PGM.LIST, the existing
data will be overwritten.

LINK

Link-edit facility.
Ex. Link the VS FORTRAN object module stored in member
EXP1 in data set CSLSU.PGM.OBJ to the VS FORTRAN
library. User CSLSU is logged on. OBJ is assumed
to be the last data set qualifier.

```
LINK PGM(EXP1) FORTLIB LIB('SYS1.VFORTLIB')
```

Note: Omit the LIB(SYS1.VFORTLIB) if program was
compiled with G1 FORTRAN compiler.

A load module will be written into member EXP1 in
data set CSLSU.PGM.LOAD. If data set CSLSU.PGM.LOAD
exists prior to issuing the LINK command, it must be
a partitioned data set. If data is already stored
in a member named EXP1, the existing data will be
overwritten.

PLI

Compile a PL/I program.
Ex. Compile the PL/I code stored in member EXP1 in data
set CSLSU.PGM.PLI. User CSLSU is logged on. PLI is
assumed to be the last data set qualifier.

```
   PLI PGM(EXP1)
```

An object module will be written into data set
CSLSU.PGM.OBJ. If the data set CSLSU.PGM.OBJ does
not exist prior to this command, the data set
created will be sequential. If the data set
CSLSU.PGM.OBJ alreay exists prior to this command,
the type (sequential or partitioned) will be the
original type defined.

If the existing data set is sequential, the object
module will be written into the sequential data set.
If the existing data set is partitioned, the object
module will be written into member TEMPNAME in the
existing partitioned data set. Existing data in a
sequential data set or in member TEMPNAME of a
partitioned data set will be overwritten.

RUN

Compile and execute an Assembler XF, COBOL, FORTRAN G1, or
PL/I Optimizer program.
Ex. "Run" member EXP1 in data set CSLSU.PGM.FORT. User
CSLSU is logged on. In this example, FORT is
assumed to be the last data set qualifier.
RUN PGM(EXP1) FORT
An object module will be written into member EXP1 in
data set CSLSU.PGM.OBJ, and the compiler output will
be written into member EXP1 in data set
CSLSU.PGM.LIST. If data set CSLSU.PGM.OBJ or
CSLSU.PGM.LIST exists prior to issuing the FORT
command, it must be a partitioned data set. If
member name EXP1 already exists in data sets
CSLSU.PGM.OBJ and/or CSLSU.PGM.LIST, the existing
data will be overwritten.

Note: If the source code had been COBOL, the run
statement would have been RUN PGM(EXP1) COBOL
with similar modifications for the other source
code language possibilities.

SASCP

Execute SAS interactively. SAS statements may be
entered following this command.
Ex. SASCP
Type /* <enter> to exit SASCP.

SASFILES

Allocate the files necessary to run SAS. Must precede
(SASF) SASCP command if executing SAS interactively. (Note:
If you frequently run SAS, you should place the SASFILES
command in the BEGIN member of your CLIST data set.)
Ex. SASF

XCTLA

Execute a program stored in load module form.
Ex. Execute the load module stored in member EXP1 in
data set CSLSU.PGM.LOAD. User CSLSU is logged on.
XCTLA EXP1 LIB(PGM.LOAD)

WATFIV

Compile and execute a WATFIV (Waterloo FORTRAN IV)
(W) program in the foreground.
Ex. Compile and execute in the foreground the FORTRAN
source code stored in member EXP2 of data set
CSLSU.PGM.FORT. User CSLSU is logged on.

```
   WATFIV PGM.FORT(EXP2)
```

Type RUN <enter> when you see the 'CMD:' prompt.
Type EXIT <enter> to end the sequence.

WPASCAL

Compile and execute a Waterloo Pascal program in the
foreground.
Ex. Compile and execute in the foreground the Pascal
source code stored in member TEST of data set
CSLSU.PGM.PASCAL. User CSLSU is logged on.

```
   WPASCAL PGM.PASCAL(TEST)
```

Type GO <enter> when you see the 'DEBUG?' prompt.

APPENDIX D -- USING ACF2 TO FIND ACCOUNTING INFORMATION

ACF2 (Access Control Facility 2) is LSU's security system
software. It is mentioned in this document because you may need
to use ACF2 to find your department and project numbers. Your
department and project numbers are required in the JOB statement
of any JCL you intend to submit. Just follow the instructions
listed below to determine these two important numbers.

(1) From SPF option 6 or from a READY prompt, type
in the command ACF <enter>.
(2) Type LIST logonid <enter>, where logonid is your
personal logonid (CSLSU in our example).
(3) Find the ACCOUNTING row (the last row displayed).
(4) Record your department and project numbers.
(5) Type END <enter> to exit from ACF2.

In our example in Figure 31, the logonid is CSLSU, the
department number is 1234, and the project number is 56565.

```
|  -------------------- TSO COMMAND PROCESSOR --------------------- |
|  ENTER TSO COMMAND OR CLIST BELOW:                               |
|                                                                  |
|  ===> acf  <enter>                                               |
|                                                                  |
|                                                                  |
|   ACF                                                            |
| list CSLSU <enter>                                               |
|   CSLSU         BRAA       CSLSU JOHN DOE                        |
|      ....                                                        |
|   ACCOUNTING    DEPT-NO(1234) PROJ-NO(56565)                     |
| end <enter>                                                      |
|                                                                  |
|                                                                  |
```

Figure 31. Using ACF2 to Find Department and Project Numbers

The JOB statement below (which was included in the batch job
first shown in Figure 19) shows the department and project
numbers located using ACF2.

```
//CSLSU1 JOB (1234,56565),'CSLSU - STUBBS',NOTIFY=CSLSU,
//          CLASS=Q,MSGCLASS=S
```

APPENDIX E -- JCL AS USED IN THIS DOCUMENT

JCL stands for Job Control Language and is responsible for
linking your program to the computer's operating system.
A key point to remember regarding Job Control Language is that
all JCL statements must be in uppercase. In running the example
program shown in Figure 19, the JCL statements used were a JOB
statement, a comment statement, and an EXEC statement. Only
these three statements will be discussed in this document.
The first JCL statement in every job is the JOB statement.
The JOB statement originally shown in Figure 19 is presented
below along with a number of comments.

```
//CSLSU1 JOB (1234,56565),'CSLSU - STUBBS',NOTIFY=CSLSU,
//          CLASS=Q,MSGCLASS=S
```

(1) Columns 1 and 2 always have // in them.
(This is true for all JCL statements.)

(2) The jobname always begins in column 3. It is a good idea to use your logonid plus an additional character for your jobname (CSLSU1 in our example).

(3) The word JOB always comes after the jobname. Be sure that there is a space before and after JOB. The most common error in JCL is to leave out the space after JOB.

(4) The accounting information follows the space after JOB. It tells the computer which account to charge. The first number is the department number and the second number is a unique project number. The two numbers are separated by a comma and enclosed in parentheses ((1234,56565) in our example).

(5) The name field follows the accounting information. You may put your name, or any other kind of identification here, as long as it does not exceed 20 characters. It must be enclosed in quotes if a blank is included ('CSLSU - STUBBS' in our example).

(6) Commas, not blanks, separate the keyword parameters from each other. Keyword parameters follow the name field. In our example, three keyword parameters are used: NOTIFY, CLASS, and MSGCLASS.

(7) The NOTIFY keyword parameter tells the machine to send messages regarding the job to the specified logonid. Such messages tell that the job has completed, or abended (abnormally ended), or whatever. Usually, your logonid is used here, but you could place someone else's logonid in this position if you wanted them to be notified (NOTIFY=CSLSU in our example).

(8) To continue the JOB statement on a second line, you need the // in columns 1 and 2 as usual, then continue the JCL statement anywhere between column 4 and column 16. A comma must follow the last keyword value on the previous line (e.g., NOTIFY=CSLSU,).

(9) The CLASS keyword parameter (if it is used) requests one of the batch monitor or express job classes. Q is the express class for short (i.e., five seconds or less) jobs which require no tape mounts (e.g., CLASS=Q).

(10) The MSGCLASS keyword parameter specifies what happens to the output of a job. If 'S' is specified, the job output is held for viewing at a terminal and possible printing (e.g., MSGCLASS=S). If 'A' is specified, job output is sent straight to the printer.

The second JCL statement (the third line) originally shown in Figure 19 is a comment statement. It is presented below along with a number of remarks.

```
//************ SAMPLE WATERLOO PASCAL JOB *************
```

(1) Even for JCL comments, columns 1 and 2 always have // in them. (This is true for all JCL statements.)

(2) For a JCL comment, a '*' is always in column 3. After the '*' in column 3, the statement can include any combination of symbols.

The third JCL statement (the fourth line) originally shown in Figure 19 is an EXEC statement. EXEC is short for execute, so EXEC followed by some program name implies that the program is to be executed. An EXEC statement is presented below along with a number of comments.

```
//      EXEC WPASCAL
```

(1) Columns 1 and 2 always have // in them.
(This is true for all JCL statements.)

(2) The word EXEC must begin somewhere between column 4 and column 16 inclusive. If you follow the EXEC statement with WPASCAL, the computer expects to compile, link, and execute a Waterloo Pascal program.

The LAST JCL statement should be // as shown in Figure 19. This brief statement tells the system that it has come to the end of your job.

Note: For more thorough documentation on JCL, you can view or print the "Introduction to JCL" and "Introduction to JCL, Part II" documents by typing L.DOC.3 <enter> from the COMMAND line on the SPF PRIMARY OPTION MENU.

APPENDIX F -- SQUARE BRACKETS

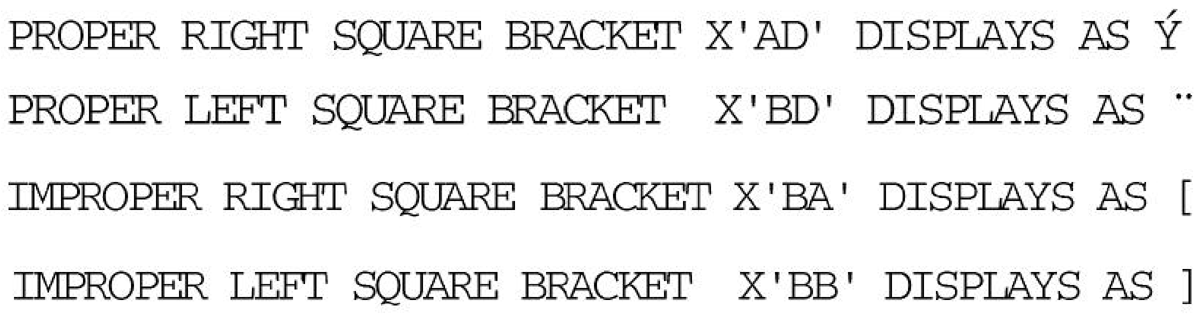
Square brackets (keyboard characters [and]) may become an issue when developing your C code. At issue is the EBCDIC hex value of the characters on the OS/390 system. The C/C++ compiler is expecting a certain hex value for those characters. It’s possible to generate the wrong hex values if you use a 3270 emulator to either transfer a file containing square brackets or type the characters into a file in a TSO session.

Square bracket hex values for square brackets in ASCII and EBCDIC:

Character	**ASCII hex**	**Proper EBCDIC hex**	**Improper EBCDIC hex**
Left Square ([ )	x’5B’	x’AD’	x’BA’
Right Square ( ])	x’5D’	x’BD’	x’BB’

Square brackets in files sent to the OS/390 system with the FTP ASCII option will result in the proper x’AD’ and x’BD’ EBCDIC characters on the OS/390 system. The C++ compiler will accept them.

You can avoid any problems with square brackets easily enough if you transfer the files using FTP to the OS/390 system. But here’s the twist. If you FTP the code to the OS/390 system, and then use a 3270 emulator to view the file, you won’t see square brackets. You’ll see Ý and ¨ that looks like a a capital Y with an accent mark over it and a double-quote symbol. A 3270 emulator ISPF text edit session in hex mode, as shown in Figure 5, displays the proper and improper characters and their ECBDIC values. The square brackets are fine; the 3270 emulator is just displaying them using different characters.


PROPER RIGHT SQUARE BRACKET X'AD' DISPLAYS AS Ý

PROPER LEFT SQUARE BRACKET X'BD' DISPLAYS AS ¨

IMPROPER RIGHT SQUARE BRACKET X'BA' DISPLAYS AS [

IMPROPER LEFT SQUARE BRACKET X'BB' DISPLAYS AS]

You can avoid any problems with square brackets easily enough if you transfer the files using FTP to the OS/390 system. But here's the twist. If you FTP the code to the OS/390 system, and then use a 3270 emulator to view the file, you won't see square brackets. You'll see Ý and ¨ that looks like a a capital Y with an accent mark over it and a double-quote symbol. A 3270 emulator ISPF text edit session in hex mode, as shown in Figure 5, displays the proper and improper characters and their ECBDIC values. The square brackets are fine; the 3270 emulator is just displaying them using different characters.

If you view the file contents with a Telnet or Rlogin session, the square brackets will look just fine. What you *don't* want to do is overtype the characters with the 3270 emulator. This will result in the *improper* x'BA' and x'BB' EBCDIC characters being placed in the file. The 3270 emulator will show them as "[" and "]", but the C++ compiler will flag them as errors. The bottom line is this: Use FTP to transfer to the OS/390 system the files with square brackets. If you use a 3270 emulator to view the file, be aware the brackets will look funny, but in reality, they are just fine. And, don't use the 3270 emulator to overtype the characters. This will result in the wrong characters being placed in the file.

APPENDIX G -- FILE TRANSFER

Most 3270 emulators offer a file transfer capability OS/390 host to PC or PC to OS/390 host.

The file transfer has to be done in TSO mode. It uses the IND$FILE facility. This works in TSO mode, but usually not in ISPF mode.

The OS/390 file name has to be enclosed by single quotes, e.g.

'aaa.bbb.ccc(ddd)''

Before attempting a file transfer, you should ensure that the host session is in a suitable state for the transfer to occur. If the host session is not in a suitable state, incoming system messages, notes, or files, will interrupt the file transfer operation and cause it to fail, often with misleading symptoms. Ensure that the following is completed:

- Messages have been set off, as follows:
  - PROFILE NOINTERCOM (in an MVS/TSO environment)
- The cursor in the host session is positioned at the READY prompt.

SNA file transfer block size can be set to 32,000 bytes provided the host system file transfer module IND$FILE is at Version 1.1.1 or higher. You can find out what release of IND$FILE is in use by typing IND$FILE?. If you use an unsuitable block size, you may get the following symptoms:

- Small files (less than 2KB) will transfer without problems.
- Larger files will fail part way through the transfer with a message indicating that the host has not responded.

A quick reminder of the syntax required can be obtained from:

- Chapter 3. Transferring Files in the Personal Communications AS/400 and 3270 Version 4.1 for DOS Reference, SC31-8265

The FTP server comes as part of the TCP/IP stack. It can be started using the SYS1.PROCLIB data set or using the UNIX shell. To perform ET/390 actions from the VisualAge for Java IDE, you must set the Sbdataconn parameter in the /etc/ftp.data data set on the host correctly to define the conversion between EBCIDIC and ASCII code pages. See the OS/390 eNetwork Communications Server V2R5 TCP/IP Implementation Guide, SG24-5227, for more information on how to define this parameter.