Batch Processing Using SAS® under z/OS

Course Notes

Batch Processing Using SAS® under z/OS Course Notes was developed by Travis Masters and Mark Jordan. Editing and production support was provided by the Curriculum Development and Support Department.

Batch Processing Using SAS® under z/OS Course Notes

Book code E70238, course code MVSB9, prepared date 16Oct06 MVSB9_002

Table of Contents

Course Description

Batch Processing Using SAS® under z/OS Course Notes teaches you how to use ISPF under z/OS to create and submit SAS programs in batch mode and to view resulting output. You also learn basic Job Control Language (JCL) related to batch processing. This course is designed to be taught in approximately one hour. It is not offered publicly; it is available for on-site presentation only.

To learn more…

A full curriculum of general and statistical instructor-based training is available at any of the Institute's training facilities. Institute instructors can also provide on-site training.

For information on other courses in the curriculum, contact the SAS Education Division at 1-800-333-7660, or send e-mail to training@sas.com. You can also find this information on the Web at support.sas.com/training/ as well as in the Training Course Catalog.

For a list of other SAS books that relate to the topics covered in this Course Notes, USA customers can contact our SAS Publishing Department at 1-800-727-3228 or send e-mail to sasbook@sas.com. Customers outside the USA, please contact your local SAS office.

Also, see the Publications Catalog on the Web at support.sas.com/pubs for a complete list of books and a convenient order form.

Prerequisites

Prior experience with the Interactive System Production Facility (ISPF)/Program Development Facility (PDF) under z/OS is recommended. Some experience with SAS software will enhance your understanding of the ‘‘Common Programming Errors’’ section of the course.

General Conventions

This section explains the various conventions that may be used in presenting text, SAS language syntax, and examples in this book.

Typographical Conventions

You will see several type styles in this book. This list explains the meaning of each style:

UPPERCASE ROMAN	is used for SAS statements and other SAS language elements when they appear in the text.
italic	identifies terms or concepts that are defined in text. Italic is also used for book titles when they are referenced in text, as well as for various syntax and mathematical elements.
bold	is used for emphasis within text.
`monospace`	is used for examples of SAS programming statements and for SAS character strings. Monospace is also used to refer to variable and data set names, field names in windows, information in fields, and user-supplied information.
select	indicates selectable items in windows and menus. This book also uses icons to represent selectable items.

Syntax Conventions

The general forms of SAS statements and commands shown in this book include only that part of the syntax actually taught in the course. For complete syntax, see the appropriate SAS reference guide.

```
PROC CHART DATA = SAS-data-set;
      HBAR | VBAR chart-variables </ options>;
RUN;
```

This is an example of how SAS syntax is shown in text:

- **PROC** and **CHART** are in uppercase bold because they are SAS keywords.
- DATA= is in uppercase to indicate that it must be spelled as shown.
- *SAS-data-set* is in italic because it represents a value that you supply. In this case, the value must be the name of a SAS data set.
- **HBAR** and **VBAR** are in uppercase bold because they are SAS keywords. They are separated by a vertical bar to indicate they are mutually exclusive; you can choose one or the other.
- *chart-variables* is in italic because it represents a value or values that you supply.
- </ *options*> represents optional syntax specific to the HBAR and VBAR statements. The angle brackets enclose the slash as well as *options* because if no options are specified you do not include the slash.
- **RUN** is in uppercase bold because it is a SAS keyword.

Chapter 1 Getting Started with SAS Software under z/OS

1.1 Fundamental Concepts

Objectives

After completing this section you will know how to do the following:

- Enter and submit SAS programs in the Interactive System Production Facility (ISPF)/Program Development Facility (PDF).
- Save SAS programs.
- Use basic Job Control Language (JCL) statements.
- Copy JCL into your SAS programs.

continued...

3

Objectives

- Use the Spooled Data Set Facility (SDSF) or Interactive Output Facility (IOF) to view your SAS log and SAS output.
- Purge (release) held output files.

4

Even if you use ISPF, the commands issued to submit batch jobs can differ from the commands in this course. Record your location-specific information in the spaces provided throughout the course notes.

At my location, the development facility I use is

_____

At my location, to access ISPF or our development facility, I

_____

Program Development Facility

After you log on to z/OS, you can go to one of these locations:

- The READY prompt
- Your development facility menu
- An alternative menu that enables you to access your development facility

In this example, when you log on to z/OS, you go to the READY prompt. At the READY prompt, you enter SPF to access the ISPF Primary Option Menu.

```
____.____1____.____2____.____3____.____4____.____5____.____6____.____7____.____
IKJ56951I NO BROADCAST MESSAGES                          SCROLL ===> HALF
READY                                                    MAIN ==> UNLOCKED
===> SPF
```

6

ISPF Main Menu

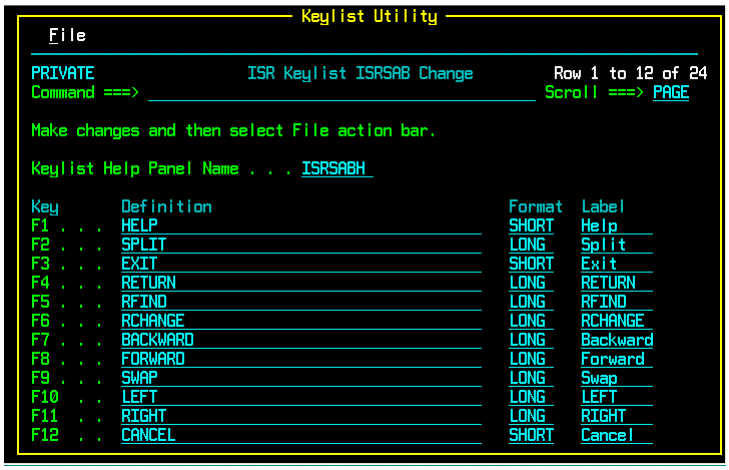
Enter a number on the Option line and press ENTER to select a menu item.

```
 Menu  Utilities  Compilers  Options  Status  Help
──────────────────────────────────────────────────────────────────────────────
                          ISPF Primary Option Menu
Option ===> ____________________________________________________________________

0  Settings      Terminal and user parameters          User ID . : EDU000
1  View          Display source data or listings       Time. . . : 15:20
2  Edit          Create or change source data          Terminal. : 3278
3  Utilities     Perform utility functions             Screen. . : 1
4  Foreground    Interactive language processing       Language. : ENGLISH
5  Batch         Submit job for language processing    Appl ID . : ISR
6  Command       Enter TSO or Workstation commands     TSO logon : LOGON
7  Dialog Test   Perform dialog testing                TSO prefix: EDU000
9  IBM Products  IBM program development products      System ID : PROD
10 SCLM          SW Configuration Library Manager      MVS acct. : EDU000,F
11 Workplace     ISPF Object/Action Workplace          Release . : ISPF 5.7

D  Tools         Display Development Tools Menu
I  IOF           Invoke IOF ("I." for Option Menu)
M  Master        Display Master Application Menu
U  User          Display User Application Menu


     Enter X to Terminate using log/list defaults
```

7

Function Keys

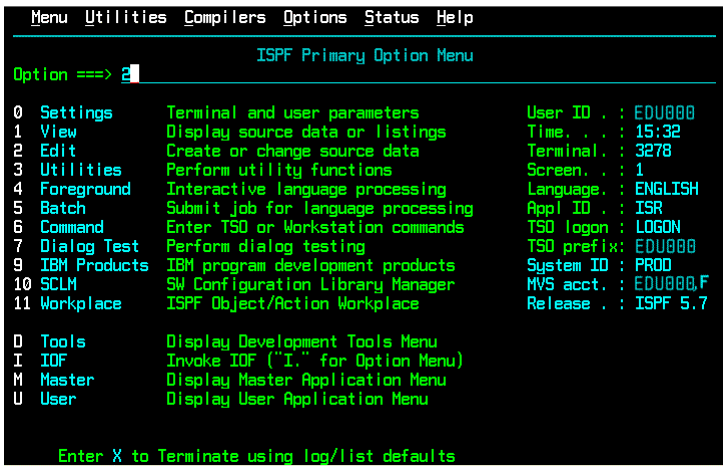
Type KEYS on the Option line and press ENTER to see function key definitions in an editable panel. Return to the ISPF Primary Option Menu by issuing the END command.



8

Entering SAS Programs

To enter a SAS program as a member of a partitioned data set (PDS), access the Edit Panel by typing 2 on the Option line and pressing ENTER.

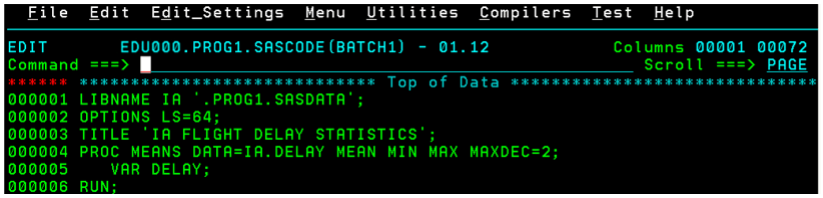


9

At my location, to access an Edit Entry Panel, I

Saving the Program

Enter the SAS program, then issue the END command to save the program and return to the Edit Entry Panel.



11

At my location, to save a SAS program and return to the previous panel, I

_____

Basic Job Control Language

To submit a SAS program in batch mode, just add Job Control Language (JCL). The specific JCL statements needed will depend on your location. This section explains the following JCL statements:

- JOB
- EXEC
- SYSIN

12

Basic Job Control Language

General form of the JOB statement:

```
//job-name JOB (acct-info),'Programmer Name'
```

job-name: Unique name for this job. Usually your user ID plus one character (a -z). Use up to eight alpha-numeric or national (@ $ #) characters. Jobs with the same name will not run concurrently.

acct-info: Optional. Indicates who to charge for the computer time used. Use a comma to skip.

Programmer Name: Maximum of 20 characters, enclosed in quotes. Printed on banners for this job.

13

Basic Job Control Language

General form of the EXEC statement:

```
//step-name EXEC procedure-name
```

step-name: A unique name for this step (optional).

procedure-name: The procedure name used at your location to invoke the SAS System (for example, SAS9).

14

Basic Job Control Language

General form of the SYSIN statement:

```
//SYSIN DD DSN=
```

or

```
//SYSIN DD *
```

DD indicates a Data Definition statement.

DSN= points to the partitioned data set member or other z/OS file that contains your SAS program.

* indicates program code will follow in same file

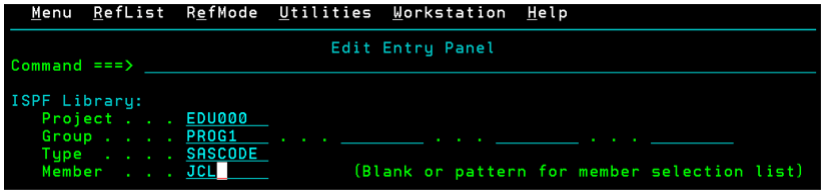
15

For more complete JOB, DD, and EXEC statement syntax, see the optional Advanced JCL section.

Basic Job Control Language

You can store JCL as a member of the same partitioned data set in which you store your SAS programs.

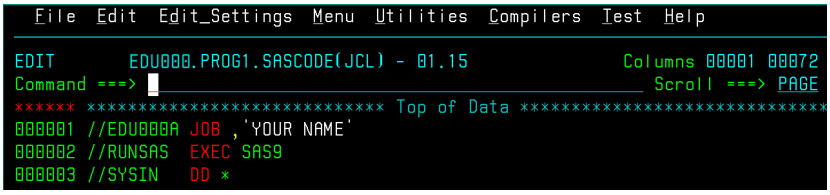
To access your JCL, enter the necessary information.



16

Basic Job Control Language

This example uses only the most basic JCL statements, making it easy to re-use. Issue the END command to return to the ISPF menu.



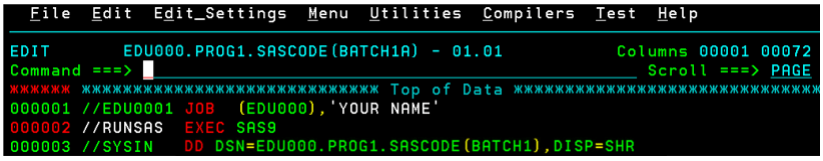
17

This file will serve as your JCL template. Copy it into your SAS program so that the SAS statements come after the SYSIN DD* statement prior to submitting your code in batch.

SYSIN Statement

There are three ways to use the SYSIN statement to identify the SAS program to process:

1. Use DSN= to point to the file that contains your program.



continued...

18

SYSIN Statement

2. Use the SAS mMacro statement %INCLUDE to bring SAS code into the SAS session from a file or PDS member.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1B) - 01.02            Columns 00001 00072
Command ===> _____________________________________ Scroll ===> PAGE
****** ********************************* Top of Data ***********************************
000001 //EDU000A JOB ,'PROGRAMMER NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004
000005 %INCLUDE 'EDU000.PROG1.SASCODE(BATCH1)';
```

continued...

19

If your stored program contains JCL (or if you do not know whether the program contains JCL), use the JCLEXCL option in the %INCLUDE statement. For example:

```
%include 'userid.PROG1.SAScode(mypgm)'/ jclexcl;
```

SYSIN Statement

3. Use DD * to indicate that SAS code follows the JCL statements in the same file.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1C) - 01.01          Member BATCH1 copied
Command ===> _____________________________________ Scroll ===> PAGE
****** ********************************* Top of Data ***********************************
000001 //EDU0001 JOB  (EDU000),'YOUR NAME'
000002 /*JOBPARM FETCH
000003 //RUNSAS  EXEC SAS9
000004 //SYSIN   DD *
000005
000006 LIBNAME IA '.PROG1.SASDATA';
000007 OPTIONS LS=64;
000008 TITLE 'IA FLIGHT DELAY STATISTICS';
000009 PROC MEANS DATA=IA.DELAY MEAN MIN MAX MAXDEC=2;
000010    VAR DELAY;
000011 RUN;
```

20

Copying JCL

If your basic JCL file uses //SYSIN DD * and is stored in the same partitioned data set (PDS) as your SAS programs, you can easily copy the JCL into SAS programs for batch submission.

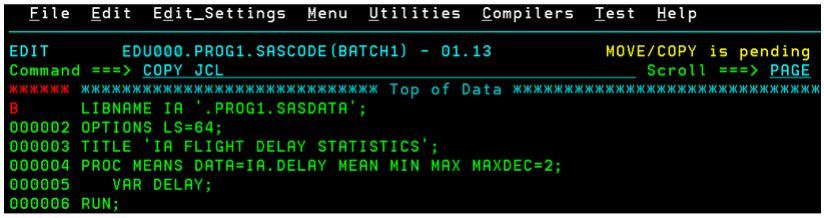
21

Copying JCL

Access the SAS program.

Place the line command B (Before) on the line number of program line 000001.

Type COPY on the command line with the member name where the JCL is saved and press ENTER.



22

At my location, to access a SAS program, I

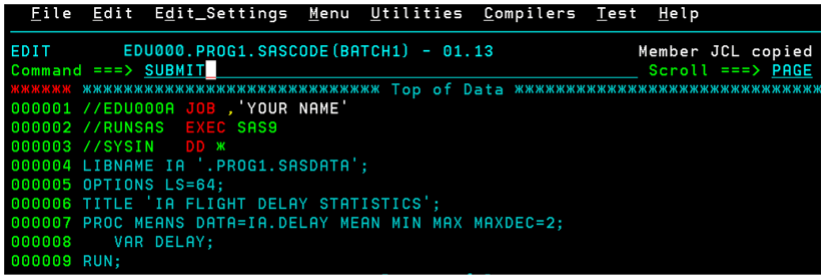
_____

At my location, to copy one file to another, I

_____

Submitting the Program

Regardless of the SYSIN statement you use to identify the SAS program, issue the SUBMIT command to submit the program for execution in batch.



23

Submitting the Program

ISPF acknowledges your submission by displaying a message like this near the bottom of the screen:

```
IKJ56250I JOB EDU000A(JOB06461) SUBMITTED
***
```

In this example, EDU000A is the name of the job and JOB06461 is the unique job identifier. It's a good idea to write down the job identifier, as you might need to refer to it later when accessing the job output files.

Three asterisks indicates "press ENTER to continue."

Return to the ISPF menu by issuing the END command.

24

At my location, to submit a SAS program, I

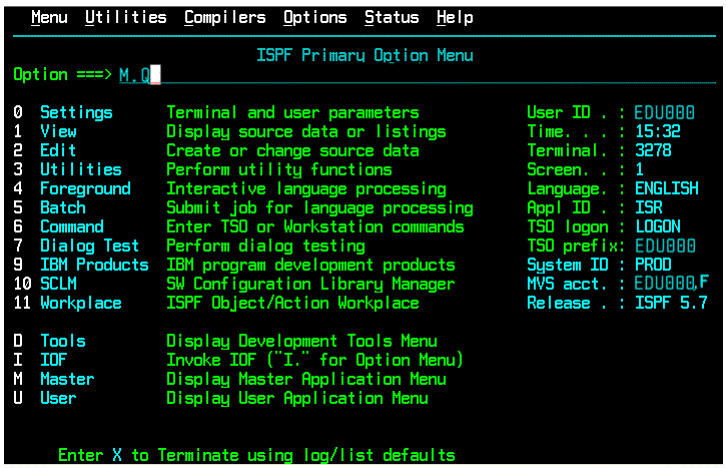
_____

After submitting the program, to return to the main ISPF menu, I

_____

Viewing the Log and Output - SDSF

The Spooled Data Set Facility (SDSF) is one way to view your log and output on a z/OS system. To access SDSF, issue the ISPF M.Q command.



25

At my location, to view the log and output, I

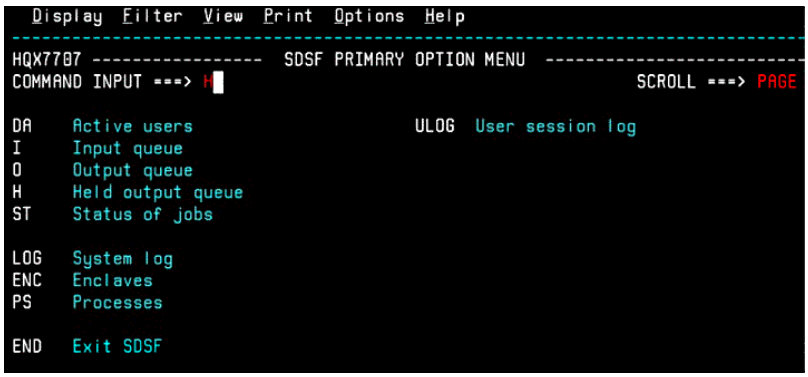
_____

To access that output facility, I

_____

Viewing the Log and Output - SDSF

To display held SAS jobs, enter H in the SDSF PRIMARY OPTION MENU.

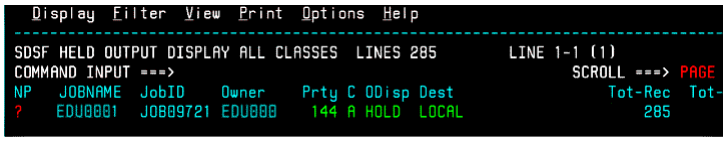


26

At my location, to display held SAS jobs, I

Viewing the Log and Output - SDSF

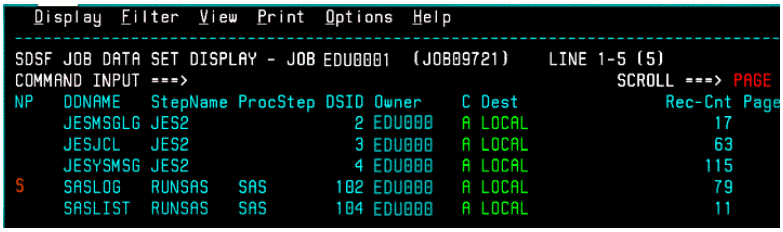
To view the log and output, enter ? in the selection column beside the JobName.



27

Viewing the Log and Output - SDSF

To view the log, enter S in the selection column beside SASLOG.



28

At my location, the SAS log is named

_____

At my location, the SAS output is named

_____

At my location, to view the log and output, I

_____

Viewing the Log and Output - SDSF

The SAS log indicates that

- the SAS LIBREF IA was assigned successfully
- the MEANS procedure created a report.

```
 Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY EDU0001  JOB09721  DSID   102 LINE 50       COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> PAGE
2          LIBNAME IA '.PROG1.SASDATA';
NOTE: Libref IA was successfully assigned as follows:
      Engine:        BASE
      Physical Name: EDU000.PROG1.SASDATA
3          OPTIONS LS=64;
4          TITLE 'IA FLIGHT DELAY STATISTICS';
5          PROC  MEANS DATA=IA.DELAY MEAN MIN MAX MAXDEC=2;
6             VAR DELAY;
7          RUN;

NOTE: There were 635 observations read from the data set
2                         The SAS System
                                    10:53 Tuesday, August 8, 2006

      IA.DELAY.
NOTE: The PROCEDURE MEANS printed page 1.
NOTE: The PROCEDURE MEANS used 0.10 CPU seconds and 7184K.
```

29

Viewing the Log and Output - SDSF

To view the output, enter S in the selection column beside SASLIST.

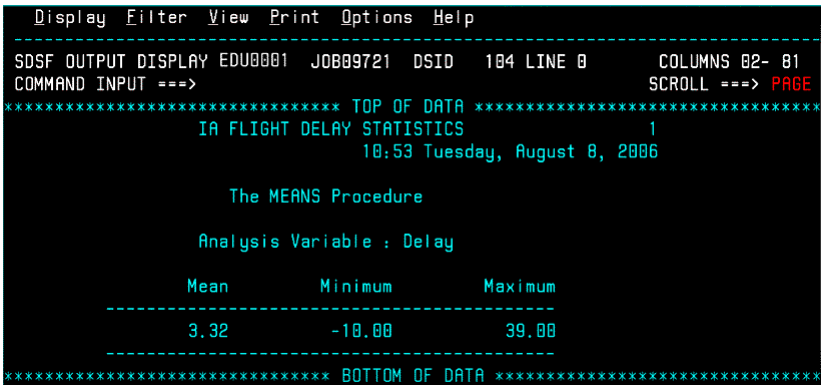
```
 Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF JOB DATA SET DISPLAY - JOB EDU0001  (JOB09721)    LINE 1-5 (5)
COMMAND INPUT ===>                                            SCROLL ===> PAGE
NP    DDNAME   StepName ProcStep DSID Owner    C Dest              Rec-Cnt Page
      JESMSGLG JES2                2 EDU000   A LOCAL                 17
      JESJCL   JES2                3 EDU000   A LOCAL                 63
      JESYSMSG JES2                4 EDU000   A LOCAL                115
      SASLOG   RUNSAS   SAS      102 EDU000   A LOCAL                 79
S     SASLIST  RUNSAS   SAS      104 EDU000   A LOCAL                 11
```

continued...

30

Viewing the Log and Output - SDSF

If necessary, scroll forward to view the entire output. Issue the END command to return to the SDSF menu.



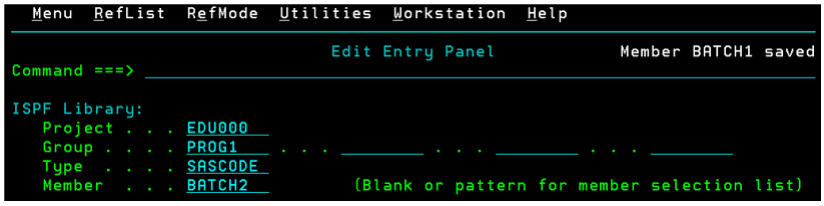
31

Continue to issue the END command to return to the ISPF Primary Option Menu.

Submitting Another Program

To write a PROC step to generate a tabular report:

1. Access an Edit Entry Panel (using ISPF 2).

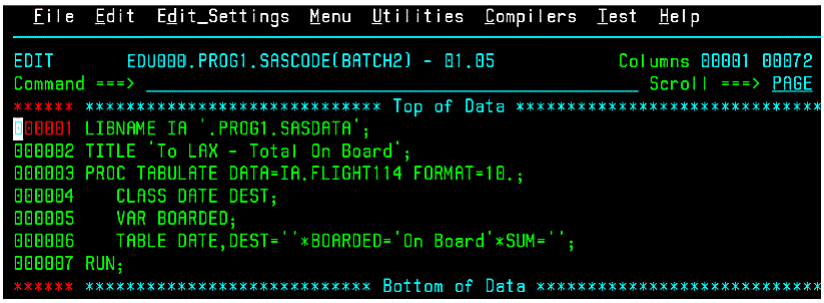


continued...

32

Submitting Another Program

2. Enter the SAS program.

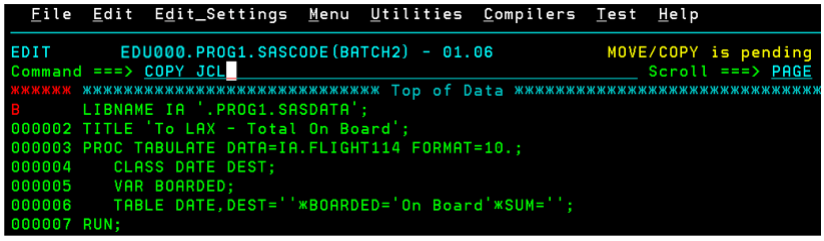


continued...

33

Submitting Another Program

3. Copy the JCL before the SAS program.



continued...

34

Submitting Another Program

4. Issue the SUBMIT command.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH2) - 01.06            Member JCL copied
Command ===> SUBMIT                                        Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU000A JOB ,'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004 LIBNAME IA '.PROG1.SASDATA';
000005 TITLE 'To LAX - Total On Board';
000006 PROC TABULATE DATA=IA.FLIGHT114 FORMAT=10.;
000007    CLASS DATE DEST;
000008    VAR BOARDED;
000009    TABLE DATE,DEST=''*BOARDED='On Board'*SUM='';
000010 RUN;
```

35

Viewing the Log and Output - IOF

The IOF facility is another way to view your log and output. To access IOF, type I on the command line and press ENTER. In the IOF Job List Menu, type an S (Select) next to the output file you want to view.

```
------------------------------ IOF Job List Menu -------------( 1 )------------
COMMAND ===>                                              SCROLL ===> SCREEN
--------------------------------- Output Jobs ---------------------------------
-------JOBNAME--JOBID---ACT-STAT-OWNER----JOB RESULT-HELD-DAY--TIME-PAGES-HELD-
S     1 EDU0001  J000045       0 MAJORD   MAXCC    0  351 222 13:25
```

Useful IOF commands include:

B - Browse entire job as a single file

C - Cancel (deletes the file, similar to **P**urge in SDSF)

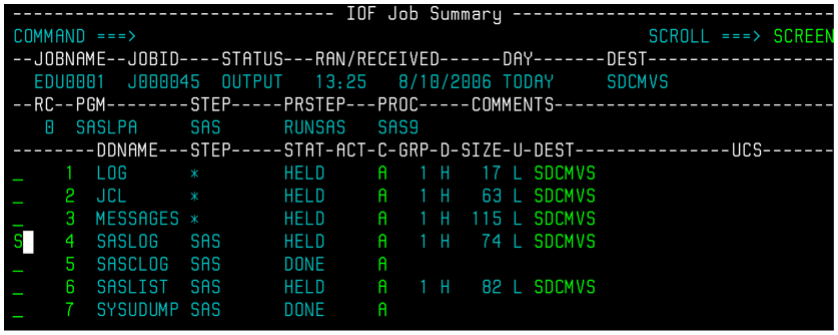
PC - Show why job will not print

S - Select for review

36

Viewing the Log and Output - IOF

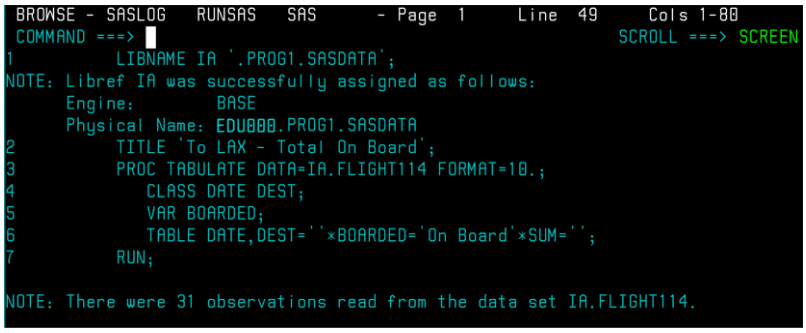
Use the IOF Job Summary menu to select individual files for viewing.



37

Viewing the Log and Output - IOF

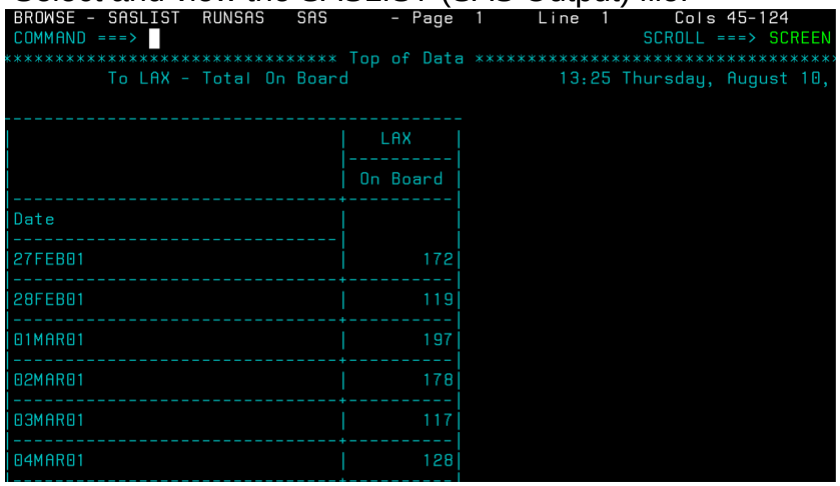
Review the SASLOG for errors, then type END on the command line to return to the IOF Job Summary Menu.



38

Viewing the Log and Output - IOF

Select and view the SASLIST (SAS Output) file.



When finished, issue the END command until you get to the ISPF Primary Option Menu.

39

At my location, the SAS log is named

_____

At my location, the SAS output is named

_____

At my location, to view the log and output, I

_____

1.2 Combining and Editing SAS Programs

Combining SAS Programs

- You can use one of the following three methods to combine SAS programs to submit as one job:
  - Use multiple DSN= parameters.
  - Use multiple %INCLUDE statements.
  - Copy the steps into one file and edit it.
- You will accomplish this using the ISPF Editor.

41

Multiple DSN= Parameters

Access the JCL that contains the DSN= parameter.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1A) - 01.01            Columns 00001 00072
Command ===>                                                Scroll ===> PAGE
ЖЖЖЖЖ ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ Top of Data ЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖЖ
000001 //EDU0001 JOB  (EDU000),'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD DSN=EDU000.PROG1.SASCODE(BATCH1),DISP=SHR
```

42

Multiple DSN= Parameters

Add a DSN= parameter for each of the individual SAS programs you want to add and submit.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1A) - 01.02            Columns 00001 00072
Command ===>                                                Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU0001 JOB  (EDU000),'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD DSN=EDU000.PROG1.SASCODE(BATCH1),DISP=SHR
000004 //        DD DSN=EDU000.PROG1.SASCODE(BATCH2),DISP=SHR
```

43

Multiple %INCLUDE Statements

Access the JCL that contains a %INCLUDE statement.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1B) - 01.02            Columns 00001 00072
Command ===>                                                Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU000A JOB ,'PROGRAMMER NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004
000005 %INCLUDE 'EDU000.PROG1.SASCODE(BATCH1)';
```

44

Multiple %INCLUDE Statements

Add a %INCLUDE statement for each SAS program you want to include and submit.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH1B) - 01.03           Columns 00001 00072
Command ===>                                               Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU000A JOB ,'PROGRAMMER NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004
000005 %INCLUDE 'EDU000.PROG1.SASCODE(BATCH1)';
000006 %INCLUDE 'EDU000.PROG1.SASCODE(BATCH2)';
```

- If your stored program contains JCL (or if you do not know if the program contains JCL), use the JCLEXCL option in the %INCLUDE statement.

45

Copying Steps

Access the Edit Entry Panel for your new program.

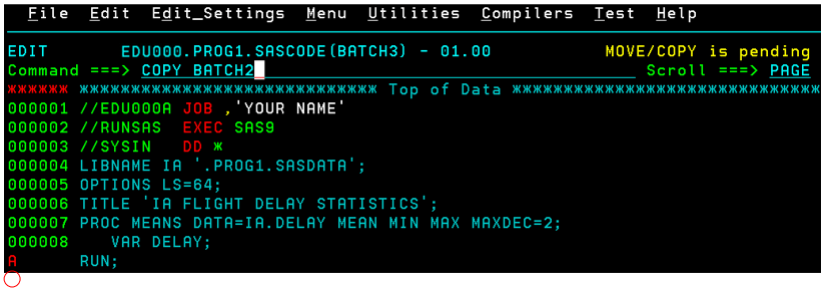
Copy in the first step using the COPY command.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH3) - 01.00           Columns 00001 00072
Command ===> COPY BATCH1                                   Scroll ===> PAGE
****** ********************************* Top of Data ********************************
******                                                                          ...
****** ******************************** Bottom of Data ********************************
```

46

Copying Steps

Repeat for the second program. Use the A line command to specify where the new text should start.



47

Text Editor Line Commands

Text editor line commands are used to modify program lines, and are typed over the text editor line numbers.

Line commands include:

In	Insert n lines
Dn	Delete n lines
Cn	Copy n lines
Mn	Move n lines
Rn	Repeat n lines
A	After (designates C, I, and M text destination)
B	Before (designates C, I, and M text destination)

48

Text Editor Block Commands

Text editor block commands modify several contiguous program lines all at once. Block commands are also typed over the text editor line numbers and include these:

DD	Delete block start / end
CC	Copy block start / end
MM	Move block start / end
RR	Repeat block start / end

A	After (designates CC and MM text destination)
B	Before (designates CC and MM text destination)

49

Copying Steps

Delete any duplicate steps and the duplicate JCL. (Use a block delete (DD) command.)

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH3) - 01.00       Block command incomplete
Command ===> _____________________________________________ Scroll ===> PAGE
****** ********************************* Top of Data **********************************
000001 //EDU000A JOB ,'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004 LIBNAME IA '.PROG1.SASDATA';
000005 OPTIONS LS=64;
000006 TITLE 'IA FLIGHT DELAY STATISTICS';
000007 PROC MEANS DATA=IA.DELAY MEAN MIN MAX MAXDEC=2;
000008    VAR DELAY;
000009 RUN;
DD     //EDU000A JOB ,'YOUR NAME'
000011 //RUNSAS  EXEC SAS9
000012 //SYSIN   DD *
DD     LIBNAME IA '.PROG1.SASDATA';
000014 TITLE 'To LAX - Total On Board';
000015 PROC TABULATE DATA=IA.FLIGHT114 FORMAT=10.;
000016    CLASS DATE DEST;
000017    VAR BOARDED;
000018    TABLE DATE,DEST=''*BOARDED='On Board'*SUM='';
000019 RUN;
```

50

Editing Your Program

Insert a blank line in the PROC TABULATE step using the Insert (I) command.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH3) - 01.00            Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU000A JOB ,'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004 LIBNAME IA '.PROG1.SASDATA';
000005 OPTIONS LS=64;
000006 TITLE 'IA FLIGHT DELAY STATISTICS';
000007 PROC MEANS DATA=IA.DELAY MEAN MIN MAX MAXDEC=2;
000008    VAR DELAY;
000009 RUN;
000010 TITLE 'To LAX - Total On Board';
000011 PROC TABULATE DATA=IA.FLIGHT114 FORMAT=10.;
000012    CLASS DATE DEST;
000013    VAR BOARDED;
I          TABLE DATE,DEST=''*BOARDED='On Board'*SUM='';
000015 RUN;
```

51

Editing Your Program

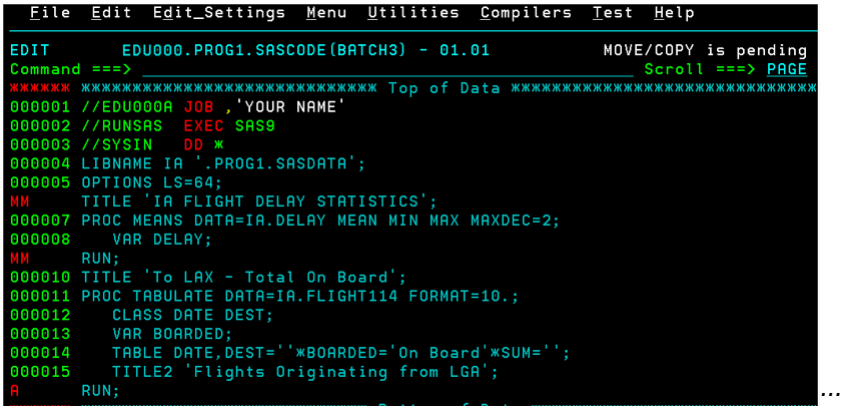
In the blank line inserted, add a TITLE2 statement.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       EDU000.PROG1.SASCODE(BATCH3) - 01.00            Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ********************************* Top of Data ********************************
000001 //EDU000A JOB ,'YOUR NAME'
000002 //RUNSAS  EXEC SAS9
000003 //SYSIN   DD *
000004 LIBNAME IA '.PROG1.SASDATA';
000005 OPTIONS LS=64;
000006 TITLE 'IA FLIGHT DELAY STATISTICS';
000007 PROC MEANS DATA=IA.DELAY MEAN MIN MAX MAXDEC=2;
000008    VAR DELAY;
000009 RUN;
000010 TITLE 'To LAX - Total On Board';
000011 PROC TABULATE DATA=IA.FLIGHT114 FORMAT=10.;
000012    CLASS DATE DEST;
000013    VAR BOARDED;
000014    TABLE DATE,DEST=''*BOARDED='On Board'*SUM='';
''''''    TITLE2 'Flights Originating from LGA';
000015 RUN;
```

52

Editing Your Program

Move PROC MEANS and its TITLE after the PROC TABULATE statement with block move (MM) and After (A) commands.

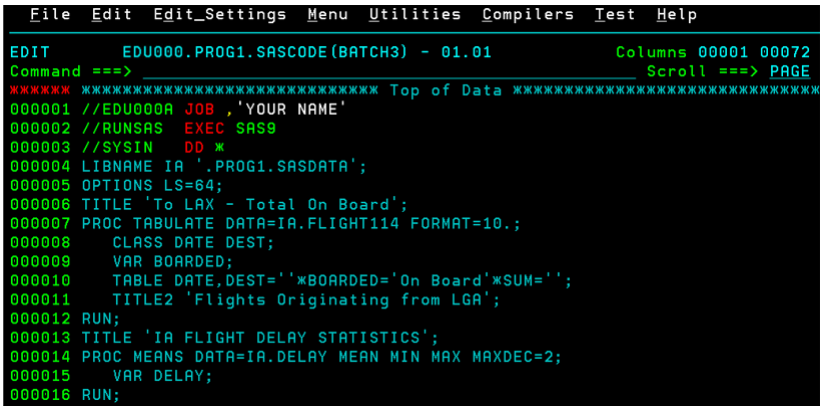


...

53

Editing Your Program

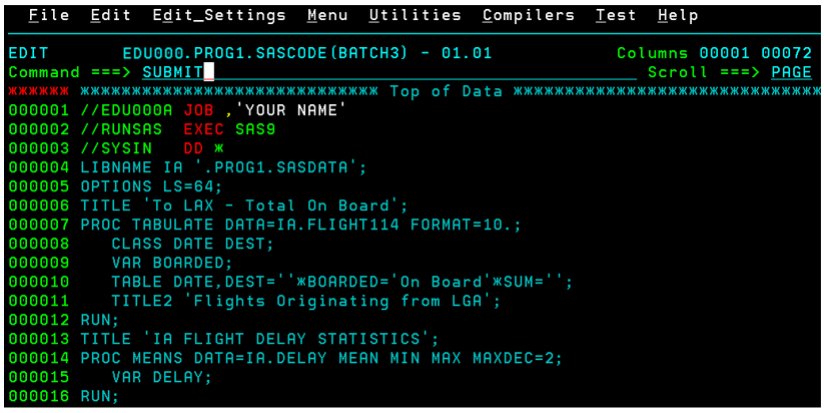
PROC MEANS is now after PROC TABULATE.



54

Submit the Combined SAS Program

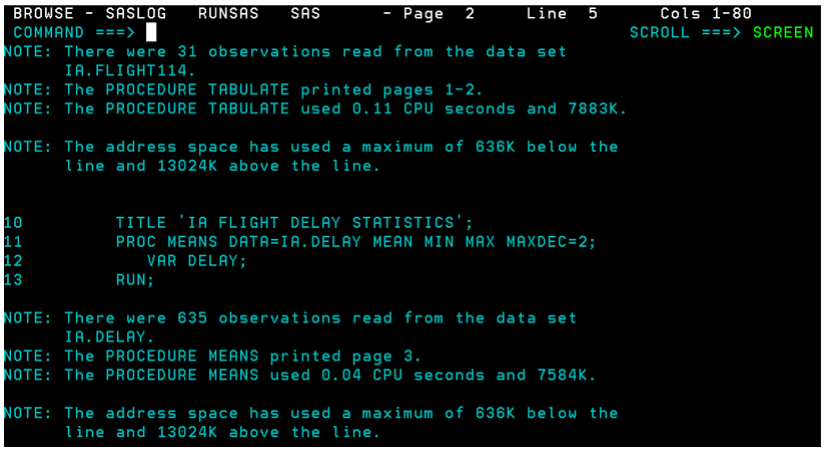
Submit the program.



55

View the Results

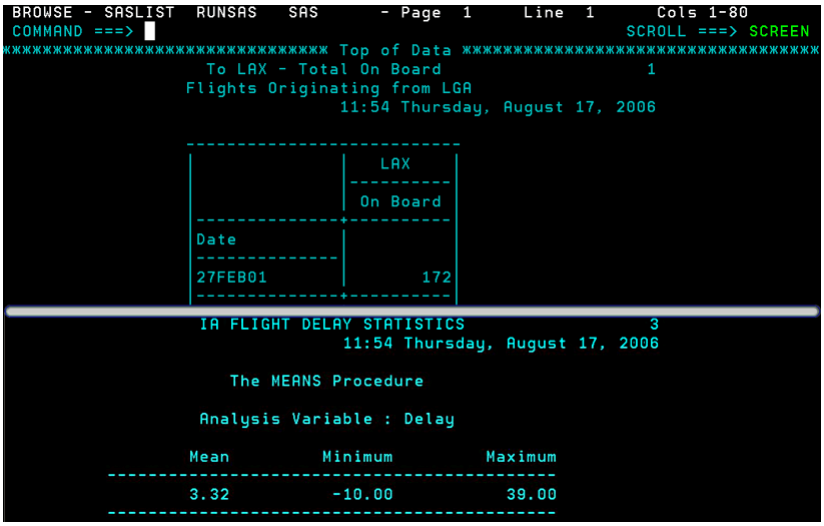
View the SAS log for the combined program in SDSF or IOF. Scroll forward to see the entire log.



56

Viewing the Results

View the SAS output in SDSF or IOF. Scroll forward and side-to-side to see the output from all the steps.



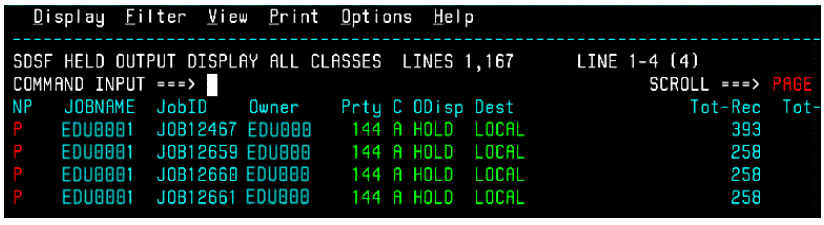
57

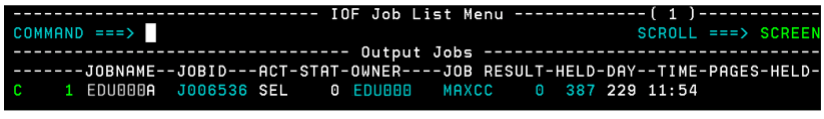
Releasing Held Files

After you view the output, you should purge it.

For SDSF, enter P in the selection field beside the file.

For IOF, enter C in the selection field beside the file.





58

At my location, to purge or release held file, I

Exercises

The files containing the programs are stored as members of a partitioned data set (PDS) named *userid*.PROG1.SASCODE. In the partitioned data set are several SAS programs and a member called JCL, which contains basic JCL statements.

1. Submitting Your First Program

a. Access the JCL. Supply your name and user ID. Return to the main menu.

b. Access the program named C02EX1. The program contains a DATA step that creates a SAS data set named AIRPORTS and a PRINT procedure step that prints the data set.

c. Place an asterisk before the INFILE statement on line 2

d. Delete the asterisk before the INFILE statement on line 3

e. Copy the JCL before the SAS program.

f. Submit the program.

g. Save your changes and return to the main menu.

h. Using IOF, SDSF, or your output facility, view the SAS log and SAS output.

2. Submitting a Second Program

a. Access the file named C02EX2. The file contains a PROC CHART step that produces a vertical bar chart of flight departure delays.

b. Copy the JCL before the SAS program and submit it.

c. Save your changes and return to the main menu.

d. Using IOF, SDSF, or your output facility, view the SAS log and SAS output.

3. Combining the Programs

a. Open an Edit Entry Panel for a new member in the PDS. Name the new member C02MYEX3.

b. Copy in the C02EX1 file.

c. Copy in the C02EX2 file after the C02EX1 program steps.

d. Delete any duplicate JCL and redundant SAS code (LIBNAME statement).

e. Submit the C02MYEX3 program.

f. Save the program and exit the editor.

Don't forget to save! You will use this program in a future exercise.

g. Using IOF, SDSF, or your output facility, view the SAS log and SAS output.

h. Release all held files.

4. Editing an Existing Program

The program C02EX4 contains a DATA step and both PRINT and MEANS procedure steps. The DATA step creates a SAS data set named DFWLAX. The PROC MEANS step produces a report with minimum, average, and maximum number of First Class and Economy passengers on the flights. The PROC PRINT step produces a listing report of the DFWLAX data set.

Make the following modifications:

a. Add this TITLE statement to the PROC MEANS step after the VAR statement. Be sure to enter both quotation marks.

```
title 'Average, Minimum, & Maximum Passenger Loads';
```

b. Move the PROC MEANS step after the PROC PRINT step.

c. Add the following TITLE statement to the PROC PRINT step. Be sure to enter both quotation marks.

```
title 'Passenger Data for DFW/LAX Flights';
```

d. Submit the program and view the SAS log and SAS output. The program should first produce the listing report, which contains all variables and all observations in the DFWLAX data set, and second the basic statistical report, which shows the average, minimum, and maximum number of passengers in First Class and Economy. Each report should have a different, appropriate title.

1.3 Common Programming Errors

Objectives

After completing this section, you will be able to

- recognize SAS error and warning messages
- identify
  - missing semicolons
  - unbalanced quotation marks
- use the ISPF text editor to correct errors.

61

Syntax Errors

When you make errors such as misspelling SAS keywords, forgetting semicolons, or specifying invalid options, the SAS System prints the following in the SAS log:

- the word ERROR or WARNING
- the location of the error
- a message explaining the error.

62

Syntax Errors

Example:

Submit the BATERROR program, read the messages in the SAS log, and correct any errors.

```
000001 //EDU0001 JOB  (&SYSUID),'MY NAME'
000002 /*JOBPARM FETCH
000003 //RUNSAS  EXEC SAS9
000004 //IA DD DSN=&SYSUID..PROG1.SASDATA,DISP=SHR
000005 //SYSIN   DD *
000006 PROC SORT DATA=IA.FLIGHT114 OUT=FLIGHT;
000007   BY DATE;
000008 RUN;
000009 TITLE 'To LAX - Total On Board';
000010 PROC TABULATE DATA FLIGHT FORMAT=10.;
000011    CLASS DATE DEST;
000012    VAR BOARDED;
000013    TABLE DATE,DEST*BOARDED='On Board'*SUM='';
000014 run;
000015
000016 TITLE 'Listing of Flights - Flight 114';
000017 PROC PRINT DATA=FLIGHT;
000018 RUN;
```

63

Interpreting Messages

View the SAS log.

```
 BROWSE - SASLOG   RUNSAS   SAS      - Page  1     Line  45     Cols 1-80
 COMMAND ===> _                                          SCROLL ===> SCREEN
NOTE: The initialization phase used 0.12 CPU seconds and 6094K.

NOTE: The address space has used a maximum of 636K below the line and 7160K abov

1          PROC SORT DATA=IA.FLIGHT114 OUT=FLIGHT;
2            BY DATE;
3          RUN;

NOTE: There were 31 observations read from the data set IA.FLIGHT114.
NOTE: The data set WORK.FLIGHT has 31 observations and 13 variables.
NOTE: The PROCEDURE SORT used 0.02 CPU seconds and 6584K.
```

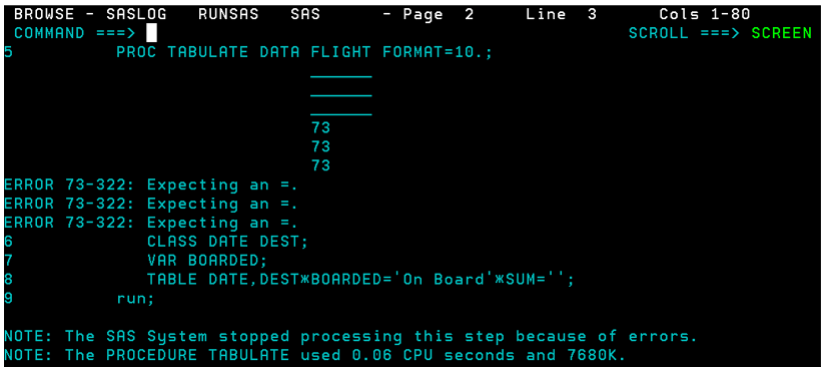
64

At my location, to view the SAS log, I

_____

Interpreting Messages

Scroll forward to view the entire SAS log.



65

At my location, to scroll forward, I

_____

Interpreting Messages

Analysis of the log indicates:

- The data set IA.FLIGHT114 was sorted successfully into the output data set WORK.FLIGHT.
- the TABULATE procedure failed:
  “Error 73-322 – Expecting an =.”
  - SAS expected to find ‘=‘ after the DATA key word in the PROC TABULATE statement.
  - The statement must be corrected and the job re-submitted.

66

Correcting the Program

Open the BATERROR program for editing.

- Edit line 10 to read:
  PROC TABULATE DATA=FLIGHT FORMAT=10.;
- Resubmit the code and recheck the results.

67

Omitting a Semicolon

Another very common programming error is omitting a semicolon at the end of a statement.

Example: In the BATSEMI program, Line 6 does not end with a semicolon.

```
000001 //EDU0001 JOB  (&SYSUID),'YOUR NAME'
000002 /*JOBPARM FETCH
000003 //RUNSAS  EXEC SAS9
000004 //IA DD DSN=&SYSUID..PROG1.SASDATA,DISP=SHR
000005 //SYSIN   DD *
000006 DATA FLIGHT
000007   SET IA.FLIGHT114;
000008 RUN;
000009 TITLE 'To LAX - Total On Board';
000010 PROC TABULATE DATA=FLIGHT FORMAT=10.;
000011    CLASS DATE DEST;
000012    VAR BOARDED;
000013    TABLE DATE,DEST*BOARDED='On Board'*SUM='';
000014 RUN;
```

68

Omitting a Semicolon

The error messages in the SAS log may not appear germane to the actual programming error.

```
  Display  Filter  View  Print  Options  Help
 ------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY EDU0001  JOB23399  DSID   102 LINE 49       COLUMNS 02- 81
 COMMAND INPUT ===>                                            SCROLL ===> PAGE
1          DATA FLIGHT
2            SET IA.FLIGHT114;
             ___
             ___
             ___
             56
             56
             56
ERROR 56-185: SET is not allowed in the DATA statement when option DATASTMTCHK=C
ERROR 56-185: SET is not allowed in the DATA statement when option DATASTMTCHK=C
ERROR 56-185: SET is not allowed in the DATA statement when option DATASTMTCHK=C
              DATA statement, or use DATASTMTCHK=NONE.
              DATA statement, or use DATASTMTCHK=NONE.
              DATA statement, or use DATASTMTCHK=NONE.

3          RUN;

NOTE: The SAS System stopped processing this step because of errors.
NOTE: The DATA statement used 0.00 CPU seconds and 6387K.
```

- Check the statements above the error message for a missing semicolon.

69

Unbalanced Quotation Marks

Some syntax errors can cause SAS to misinterpret SAS statements in your program.

Example: In program BATQUOTE, Line 9 has unbalanced quotation marks (opened with “, but closed with ‘).

```
000001 //EDU0001 JOB  (&SYSUID),'YOUR NAME'
000002 /*JOBPARM FETCH
000003 //RUNSAS  EXEC SAS9
000004 //IA DD DSN=&SYSUID..PROG1.SASDATA,DISP=SHR
000005 //SYSIN   DD *
000006
000007 PROC PRINT DATA=IA.DFWLAX;
000008    VAR DATE FLIGHT DEST FIRSTCLASS ECONOMY;
000009 TITLE1 "LAX Flights';
000010 TITLE2 "People On Board";
000011 RUN;
```

70

Unbalanced Quotation Marks

SAS log warnings about quoted strings (such as “quoted string too long” or “meaning of an identifier after a quoted string”) usually result from unbalanced quotes.

```
SDSF OUTPUT DISPLAY EDU0001  JOB24530  DSID   102 LINE 50       COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> PAGE
2          PROC PRINT DATA=IA.DFWLAX;
3             VAR DATE FLIGHT DEST FIRSTCLASS ECONOMY;
4          TITLE1
4        | "
           _
           _
           _
           49
           49
           49
4        | LAX Flights';
NOTE 49-169: The meaning of an identifier after a quoted string may change in a
             a quoted string and the succeeding identifier is recommended.
5          TITLE2 "People On Board";
WARNING: The TITLE statement is ambiguous due to invalid options or unquoted tex
6          RUN;
```

71

Unbalanced Quotation Marks

Output titles not as expected may indicate that the unbalanced quotes are in a TITLE statement.

```
  Display  Filter  View  Print  Options  Help
--------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY EDU0001  JOB24530  DSID   104 LINE 0        COLUMNS 02- 81
COMMAND INPUT ===>                                            SCROLL ===> PAGE
********************************* TOP OF DATA ***********************************
LAX Flights';                                                        TITLE2 P


                                                                        First
                                  Obs     Date    Flight    Dest      Class

                                    1    14955     439      LAX         20
                                    2    14955     921      DFW         20
                                    3    14956     114      LAX         15
                                    4    14956     982      dfw          5
                                    5    14957     439      LAX         14
                                    6    14957     982      DFW         15
                                    7    14958     431      LaX         17
                                    8    14958     982      DFW          7
                                    9    14959     114      LAX          .
                                   10    14959     982      DFW         14
******************************** BOTTOM OF DATA *********************************
```

72

To resolve the problem in batch mode, find the missing quotation mark and insert one in the proper location in your code, then resubmit the program.

Exercises

The files containing the programs are stored as members of a PDS named *userid*.PROG1.SASCODE.

5. Correcting Errors

a. Submit the file named C02EX5. The file contains a DATA step, a PRINT procedure step, and a MEANS procedure step. The DATA step creates a SAS data set named DFW, the PROC PRINT step prints the data set, and the PROC MEANS step requests the average, minimum, and maximum pounds of mail on these flights.

b. View the SAS log and determine the error(s).

c. Correct the errors in the program and submit the corrected version.

d. View the SAS log and SAS output.

1.4 SAS Windowing Environment (Optional)

Objectives

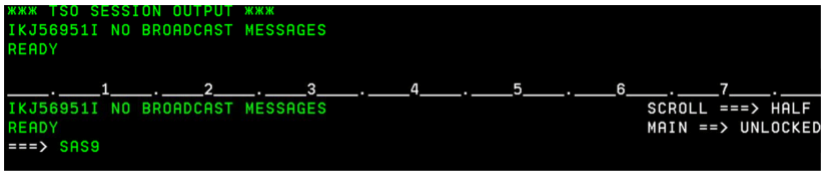
After completing this section, you should be able to

- use the SAS Windowing Environment to write, edit, and submit programs
- view your output in the SAS Windowing Environment
- recall and save your programs from within the SAS Windowing Environment.

75

Starting SAS in Interactive Mode

At the READY prompt, enter the name of the SAS cataloged procedure (usually the same one used to invoke SAS in a JCL EXEC statement):



SAS starts using the interactive Windowing Environment

76

To access the SAS windowing environment at my location, I

_____

Introduction to SAS Windowing Environment

The SAS Windowing Environment is a valuable tool for developing programs to be submitted in batch. There are three primary windows:

- the Program Editor window
- the Log window
- the Output window.

77

Introduction to SAS Windowing Environment

The Program Editor window corresponds to the Edit Entry Panel in your development facility. You can use the Program Editor window to

- write programs
- include programs from an external file
- edit programs
- submit programs.

The Log and Output windows correspond to SASLOG and SASLIST in batch mode.

You can issue commands in Windowing Environment windows to accomplish tasks.

78

Introduction to SAS Windowing Environment

- The Log window corresponds to the SASLOG file in batch mode.
- The Output window corresponds to the SASLIST file in batch mode.
- You can issue commands on the Windowing Environment command line to accomplish tasks.
- Programmable, context-sensitive function keys can be used to issue commands to the command line of the active window. Use the KEYS command to view and edit the function executed by each key.

79

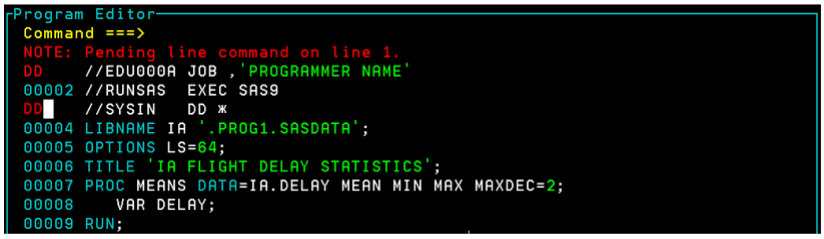
Including a SAS Program

If you have stored a SAS program in an external file, bring the program into the Program Editor window by issuing the INCLUDE command.

80

Preparing the Program for Submission

Add a RUN statement at the end of the program, if required. If the program contains JCL, delete it using the block delete (DD) command.

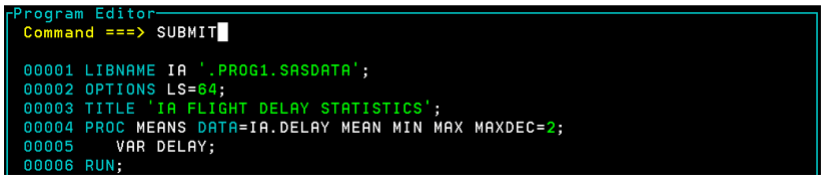


81

All of the text editor line commands that you learned for batch mode are available in the Program Editor window. Just as in ISPF, some commands can be submitted quickly and easily using function keys. To review and / or modify function key definitions, issue the KEYS command.

Submitting the SAS Program

When you have completed any necessary editing in Windowing Environment, submit the program with the SUBMIT command.



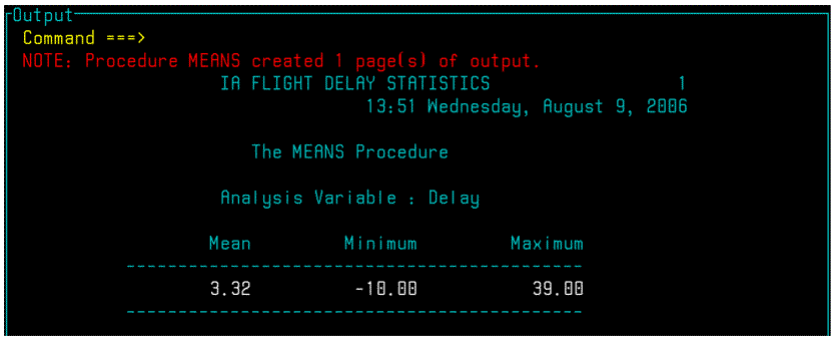
82

When developing SAS programs with very large data sets, system options can make the use of the SAS windowing environment more effective. For example, the OBS=5 system option stops each SAS process at the fifth observation in the referenced data sets.

Furthermore, if your DATA step works properly, you do not have to resubmit the DATA step to test your procedures on a temporary data set because the data remains available until the SAS session terminates.

Viewing the Output

The results of the PRINT procedure are displayed in the Output window. Issue the END command to return to the Program Editor window.



83

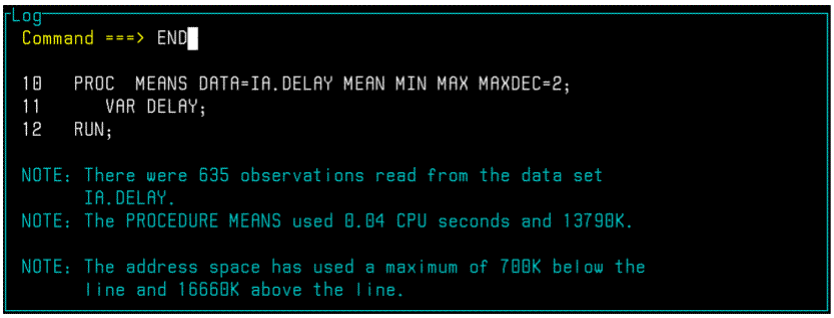
Viewing the Log

Issue the LOG command to make the Log window active.

84

Viewing the Log

Issue the END command to return to the Program Editor window.



85

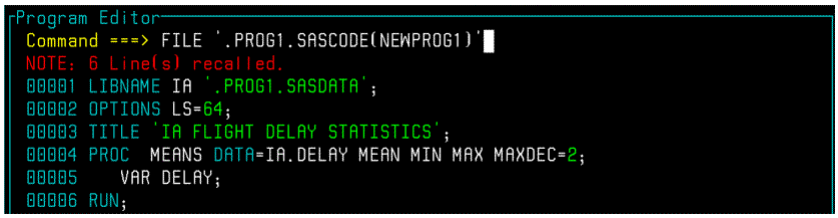
Recalling the Program

In the Program Editor window, issue the RECALL command to recall the program.

86

Saving the Program

After making changes to the program and removing any options that are not needed, save the program to an external file by issuing the FILE command.



The saved SAS program can be submitted in batch, using one of the methods discussed earlier.

87

Exiting the SAS Windowing Environment

To exit the SAS Windowing Environment and terminate your interactive SAS session, use the command ENDSAS. The command BYE is an alias for ENDSAS.

SAS terminates and returns you to the READY prompt.

88

Exercises

The files containing the programs are stored as members of a PDS named *userid*.PROG1.SASCODE.

6. Using the SAS Windowing Environment.

a. Include the file C02MYEX3.

b. Delete the JCL. Add this statement at the beginning of the program:

```
OPTIONS OBS=10;
```

Add a RUN statement at the end of the program, if required.

c. Submit the SAS program.

d. Recall the program.

e. Save the program to a new PDS member named C02MYEX6.

7. Batch Submitting Programs Created in the SAS DMS

a. Exit SAS DMS.

b. Return to your text editor.

c. Access the PDS member C02MYEX6.

d. Copy your JCL before the SAS program.

e. Delete the OPTIONS statement.

f. Submit the job in batch.

g. View your SAS log and SAS output.

h. Release all held files.

1.5 Additional JCL (Optional)

Objectives

After completing this section, you should be able to

- write more complex JCL statements
- better understand JCL syntax.

91

JCL Syntax

JCL involves a complex, rigidly applied set of rules.

A subset of the rules includes:

- Record length for JCL files must be 80 characters
- JCL statements
  - Must begin in column 1
  - Must be written completely in uppercase
  - Cannot extend past column 71
  - Usually begin with two slashes (//) except:
    - JOBPARM and Delimiter statements, which begin with a slash and an asterisk (/*)
    - Comments, which begin with two slashes and an asterisk (//*)

continued...

92

JCL Syntax

- Continue a JCL statement by ending the first line with a comma after a complete parameter.
- The continuation line must:
  - Begin with two slashes in columns 1-2
  - The first parameter must begin in column 4-16.
  - The line must end on or before column 71.

continued...

93

JCL Syntax

- Enclose lists of sub-parameters in parentheses, separated by commas.
- Omit a positional parameter or sub-parameter in a list by marking its place with a comma. If the last parameter is omitted, no comma is necessary.
- Omit the parentheses when using a single (first parameter or a keyword) sub-parameter from the list

94

JOB Statement

General form of the JOB statement:

```
//jobname JOB (acctg info),'name',CLASS=a,
//  NOTIFY=userid,MSGCLASS=b,
//  MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

jobname: A name for this job. Typically your user ID + one character (A-Z). Can be any combination of 8 or fewer alpha-numeric and @ $ # characters.

acctg info: Information to identify the person or group to be charged for the computer resources used by this job.

continued...

95

JOB Statement

```
//jobname JOB (acctg info),'name',CLASS=a,
//  NOTIFY=userid,MSGCLASS=b,
//  MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

name: A name (1-20 characters) that will be printed on the banner of the job's output. Enclose the text in single quotes.

a: The queue for this job, as defined by your System Administrator. Some queues run jobs at specified times, others may cancel jobs that use too many CPU seconds.

continued...

96

JOB Statement

```
//jobname JOB (acctg info),'name',CLASS=a,
//  NOTIFY=userid,MSGCLASS=b,
//  MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

userid: This user will be notified via broadcast message when the job finishes. The symbolic parameter &SYSUID may be substituted here.

b: designates the output queue (site specific)

n: controls printing of JCL:

0 – No JCL messages
1 – All JCL messages
2 – All JCL messages, except from cataloged procedures

continued...

97

JOB Statement

```
//jobname JOB (acctg info),'name',CLASS=a,
//  NOTIFY=userid,MSGCLASS=b,
//  MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

d: Controls data set allocation / disposition messages:

0 – No allocation messages unless the job abends
1 – All allocation messages

k: Specifies amount of storage a job can use in kilobytes or megabytes (Example: 2K or 5M)

continued...

98

JOB Statement

```
//jobname JOB (acctg info),'name',CLASS=a,
//  NOTIFY=userid,MSGCLASS=b,
//  MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

M: Total CPU minutes the job is allowed to run

S: Total CPU seconds the job is allowed to run

99

OUTPUT and JOBPARM Statements

Here you learn two ways to explicitly manage output:

- JCL OUTPUT statement
- JES2 JOBPARM statement

Consult the SAS Software Consultant or System Administrator at your site for the preferable method and parameter values for your system.

100

OUTPUT Statements

OUTPUT statement with selected parameters:

```
//formname OUTPUT DEST=destid,
//   COPIES=x,DEFAULT=yn
```

formname: A maximum of eight alphanumeric and national (@ $ #) characters. Must start with a letter or national character.

destid: The destination printer ID.

continued...

101

OUTPUT Statements

```
//formname OUTPUT DEST=destid,
//    COPIES=x,DEFAULT=yn
```

x: Number of copies of the output to print.

yn: Valid values are YES or NO. Determines if the OUTPUT statement parameters are applied by default to ALL SYSOUT DD statements.

JOBPARM Statements

General form of the JOBPARM statement to control output:

```
/*JOBPARM DEST=dest,NAME='formname',COPIES=x
```

formname: A maximum of eight alphanumeric and national characters (@ $ #); cannot start with a number

destid: Destination printer ID

x: Number of copies of the OUTPUT to print

continued...

103

JOBPARM Statements

Special form of the JOBPARM statement to hold output:

```
/*JOBPARM FETCH
```

FETCH: Holds the job output in an output queue.

This JOBPARM statement is used in a JES2 environment

104

EXEC Statement

General form of the EXEC statement:

```
//stepname EXEC proc|PGM=prog,
//  REGION=k,TIME=s
```

stepname: Identifier name for this step (optional)

proc: A cataloged procedure (Example: SAS9)

prog: An application program name

continued...

105

EXEC Statement

```
//stepname EXEC proc|PGM=prog,
//  REGION=k,TIME=s
```

k: Amount of storage THIS STEP can use in Kbytes or Mbytes (Example: 2K or 5M)

S: Total CPU seconds THIS STEP is allowed to run

106

DD Statements

General form of the Data Definition (DD) statement for accessing existing data sets:

```
// ddname DD DSN=project.group.type,
//    DISP=disp
```

ddname: Alias for the file specified in this DD statement

project.group.type: Indicates the data set to use, such as a sequential file, PDS, or PDS member (member name in parentheses).

continued...

107

DD Statements

```
// ddname DD DSN=project.group.type,
//    DISP=disp
```

disp: Disposition for the file:

- SHR (share - multiple users, read-only)
- OLD (single user, write access)

continued...

108

DD Statements

General form of the Data Definition (DD) statement for creating new data sets:

```
//ddname DD DSN=project.group.type,
//   DISP=disp
```

disp: Disposition for the file:

- To create new files:
  - NEW (DEFAULT– create a new file)
  - MOD (if file exists, write over it; if not, create it)

continued...

109

DD Statements

General form of the Data Definition (DD) statement for creating new data sets:

```
// ddname DD DSN=project.group.type,
//   DISP=(d,n,a),SPACE=(t,(p,s),r),
//   UNIT=u,VOL=SER=v
```

project.group.type: A data set name.

d: The disposition for this data set. Valid values: NEW (default), MOD (write over existing file).

n: Action if job finishes normally. Some valid values: CATLG and DELETE.

a: Action if job abends. Valid values as for *n.*

continued...

110

DD Statements

```
// ddname DD DSN=project.group.type,
//    DISP=(d,n,a),SPACE=(t,(p,s),r),
//    UNIT=u,VOL=SER=v
```

t: Units for space allocation: TRK, CYL, or blocksize.

p: Number of units needed for primary extent.

s: Number of units for each secondary extent.

r: Optional. If coded (RLSE), unused space is released to the system at the end of the job.

continued...

111

DD Statements

```
// ddname DD DSN=project.group.type,
//    DISP=(d,n,a),SPACE=(t,(p,s),r),
//    UNIT=u,VOL=SER=v
```

u: Device type or group.

Examples:

- 380 (device type)
- SYSDA (group)

v: Volume to which the data set is written. If not specified, the system chooses a volume with space available.

112

DD Example: Creating a File

```
//NEWFILE DD DSN=USERID.NEW.FILE,
//   DISP=(NEW,CATLG,DELETE),
//   SPACE=(CYL,(5,2),RLSE),
//   UNIT=SYSDA,VOL=SER=SAS900
```

The above DD statement accomplishes these tasks:

- Assigns the alias NEWFILE to USERID.NEW.FILE
- Creates a new data set, which will be cataloged if the job finishes normally or deleted if the job abends

continued...

113

DD Example: Creating a File

```
//NEWFILE DD DSN=USERID.NEW.FILE,
//   DISP=(NEW,CATLG,DELETE),
//   SPACE=(CYL,(5,2),RLSE),
//   UNIT=SYSDA,VOL=SER=SAS900
```

In addition:

- The file allocates five cylinders initially, and two cylinders for each subsequent extent.
- Unused space is released at the end of the job.
- The file will be written to the SYSDA group, on a volume labeled SAS900.

114

DD Example: Concatenating Files

When your data are contained in more than one file or more than one member of a PDS, you can treat them all as one contiguous file by concatenating DD statements. For example:

```
//FLATFILE DD DSN=USERID.FILE1.DATA,DISP=SHR
//         DD DSN=USERID.FILE2.DATA,DISP=SHR
//         DD DSN=USERID.FILE3.DATA,DISP=SHR
```

In this example, the three files can be used as if they were actually one single, large file. Records from each file will be encountered in the order that the data sets were listed.

115

DD Example: Rerouting SAS Log and Output

DD statements can be used to route the SAS log and SAS output to a location other than the default.

```
//SASLOG DD DSN=USERID.FILE.LOG,DISP=OLD
//SASLIST DD DSN=USERID.FILE.LIST,DISP=OLD
```

116

DD Example: SYSIN

When used with the special label SYSIN, a DD statement indicates which file(s) contains SAS code to be executed.

```
//SYSIN DD DSN=EDU000.PROG1.SASCODE(C02S2D1),
//      DISP=SHR
//      DD DSN=EDU000.PROG1.SASCODE(C02S2D2),
//      DISP=SHR
```

117

DD Example: Creating a Libref in JCL

A JCL DD statement can be used instead of a SAS LIBNAME statement to allocate a SAS data library.

```
//IA DD DSN=&SYSUID..PROG1.SASDATA,DISP=SHR
```

- The libref IA is assigned to a file using the submitting user's USERID for the first node and PROG1.SASDATA for the remainder of the filename.
- A DD statement causes the job to wait for the file to become available before executing.
- When using a SAS LIBNAME statement to allocate the file, an error is generated if the file is not available when executed and the LIBREF assignment fails.

118

DD Example: Creating a Fileref in JCL

You can also use JCL DD statements to assign filerefs to flat files, in lieu of a SAS FILENAME statement.

```
//MYRAW DD DSN=EDU000.PROG1.RAWDATA,DISP=SHR
```

- The fileref MYRAW is assigned to the file EDU000.PROG1.RAWDATA.
- Again, the DD statement causes the job to wait for the file to become available before executing.
- When using a SAS FILENAME statement to allocate the file, an error is generated if the file is not available when executed and the fileref assignment fails.

1.6 Chapter Summary

Commands to use:

Function	In these examples	At my location
Access the development facility (ISPF)	SPF	
View the function key definitions	ISPF KEYS	
Access an EDIT -ENTRY panel	ISPF =2	
Save a SAS program	END	
Access a saved SAS program	ISPF =2	
Copy	COPY (with A or B)	
Submit a SAS program	SUBMIT	
IOF		
View the output queue	ISPF =I	
Release the held files	C in the selection field	
View the log and output	S in the selection field	
SAS log filename	SASLOG	
SAS output filename	SASLIST	
SDSF		
View the held output queue	ISPF =M.Q.H	
Release the held files	P in the selection field	
View the log and output	S in the selection field	
SAS log filename	SASLOG	
SAS output filename	SASLIST	

General form of the JOB statement:

```
// job-name JOB (accounting-information)
```

General form of the EXEC statement:

```
// step-name EXEC procedure-name
```

General form of the SYSIN statement:

```
// SYSIN DD DSN=
```

or

```
// SYSIN DD *
```

General form of the JOB statement with selected parameters:

```
// jobname JOB (acctg info),'name',CLASS=a,NOTIFY=userid,MSGCLASS=b,
// MSGLEVEL=(n,d),REGION=k,TIME=(m,s)
```

General form of the EXEC statement with selected parameters:

```
// step-name EXEC procedure | PGM=program,REGION=k,TIME=s
```

General form of the DD, or Data Definition, statement for using existing files:

```
// ddname DD DSN=userid.data.set,DISP=disp
```

General form of the DD statement for creating files:

```
// ddname DD DSN=userid.data.set DISP=(d,n,a),SPACE=(t,(p,s),r), UNIT=u,
// VOL=SER=v
```

General form of the OUTPUT statement with selected parameters:

```
// formname OUTPUT DEST=dest,DEPT='dept',NAME='name'
```

General form of the JOBPARM statement to route output to HOLD queue:

```
/* JOBPARM FETCH
```

General form of the JOBPARM statement with selected parameters:

```
/* JOBPARM DEST=dest DEPT='dept',NAME='form-name'
```

Resources for Further Study

- SAS OnlineDoc, *SAS 9.1.3 Companion for z/OS* on the Web at http://support.sas.com/onlinedoc/913/
- *Introduction to the New Mainframe: z/OS Basics*, Chapter 6, “Using JCL and SDSF”, on the Web at http://publibz.boulder.ibm.com/zoslib/pdf/zosbasic.pdf
- z/OS JCL Fifth Edition - Gary Deward Brown

121