Strategies for Planning and Implementing the Conversion of SAS® Applications from Version 5 to Version 6

Harriet Janes, SAS Institute Inc., Cary, N.C.
Sarah Dietrich, SAS Institute Inc., Cary, N.C.

ABSTRACT

This paper discusses what you should be doing and thinking about as you develop and maintain Version 5 applications so your installation can convert to Version 6 with a minimum amount of resources and problems. It is directed toward managers who will be administering this conversion and technicians who will be responsible for implementing the conversion. The use of SAS® tools that can simplify this conversion process, including the SAS® macro and autocall facilities, and SAS/DMI software is also discussed.

INTRODUCTION

This paper covers a brief overview of what applications will need to be converted for Version 6 and how to prepare your Version 5 applications for future conversion. After the overview, there will be a discussion about some programming techniques and tools that might be useful in Version 5 when preparing your applications for conversion.

Please note that the approach that follows is only a suggested approach that will have to be adapted to local standards and conventions. As in every conversion effort, it is critical that you have a backup of all data and source code that will be modified. Keep accurate, detailed records of your conversion effort. The authors recommend that you maintain a notebook or a journal of what you have done, any problems that may occur, and any steps that need to be completed later. After you have completed the conversion process, this journal will be a very important document for any questions that may arise as a result of the conversion; you can also use it to provide your management with a report of all the work that has been completed.

This paper emphasizes applications that use the SAS/AF® and SAS/FSP® software. These techniques include the use of the SAS macro and autocall facilities. Following this discussion, tools specific to the OS and CMS host systems will be discussed in detail. Under OS there will be a discussion of: how SAS/DMI® software can be used to convert SAS data libraries under the OS host system, how to convert applications to use the SAS autocall facility, and how and why to convert allocate statements to LIBNAME statements. Under the CMS host system there is similar discussion of how to convert applications to use the SAS autocall facility, and several other changes necessary for Version 6 applications.

WHAT APPLICATIONS NEED TO BE CONVERTED

A Version 5 format SAS data library may contain five different types of files:

- SAS data set (MEMTYPE=DATA)
- Catalog (MEMTYPE=CAT)
- Graphics Catalog (MEMTYPE=GCAT)
- IML Work Space (MEMTYPE=IMLWK)
- ETS Models (MEMTYPE=MODEL)

The Version 5 compatibility engine provides read and write access to SAS data sets only (the first type of file). All other types of files will have to be converted to Version 6 format before they can be used. This can be done using the V5TOV6 procedure. This paper does not discuss the details of the conversion procedure, but, rather, the ways to prepare for using this procedure in the future. If your applications consist of only SAS data sets, you will not have to convert your data files until you want to take advantage of some of the new features of Version 6, such as indexing and data compression.

HOW TO PREPARE APPLICATIONS FOR CONVERSION

Separate Data Files From Other Files

Assuming that you have more than just SAS data sets in your SAS data libraries, what things should you be thinking about for existing and future applications?

If there are applications that share the same data files and that may be converted at different times, it is necessary to separate the SAS data files from all other types of files. Until all applications are converted that reference the same data file, the data file should not be converted to a Version 6 format. However, to use a catalog under Version 6, it must be converted. A discussion of how to separate these files is discussed in detail in the section that relates to the specific host system.

Convert SAS/AF Programs to Macros and Store Externally in a SAS Autocall Library

To aid in converting the SAS/AF programs that reference the data files that are to be separated from catalogs, you may want to convert all these programs to macros and store them externally in a SAS autocall library. How these programs can be converted to use the SAS macro and autocall facility is discussed separately under both the OS and CMS host systems. In this manner, you can easily manipulate your SAS/AF programs with the least impact on your user community.

Using the SAS Autocall Facility

To compile a macro without the SAS autocall facility, it must be created within the current program or brought in with a %INCLUDE statement before you could invoke the macro. When the system option MAUTOSOURCE is in effect, invoking a macro that has not been defined causes the SAS autocall facility to search the autocall library or libraries for a member with the same name as the macro requested. If the member exists, the SAS autocall facility performs the following tasks:

- brings the source statements in the member into the current SAS program. The member must include a definition for the macro you invoked and can include other SAS statements and macro definitions.
- processes all the source statements and compiles all the macros in the member only once.

- executes the macro you requested.

To create a macro that can be called automatically, you must follow two steps. First, create a SAS autocall library. Creating the library is discussed under the appropriate host system discussion that follows. Second, create a member in the library containing the source statements for the macro. The name of the member must be the same name as the macro that you plan to call.

To associate a SAS autocall library with a SAS program, you must follow these steps. First, associate the library with your SAS job or session as described in a later section. Second, make sure that the SAS system option MAUTOSOURCE is in effect at SAS invocation.

For a complete discussion of the SAS autocall facility in Version 5, refer to Chapter 6, "The Autocall Facility" in the *SAS Guide to Macro Processing, Version 5 Edition*.

PROGRAMMING TECHNIQUES AND TOOLS FOR THE OS HOST SYSTEM

Locating SAS Data Libraries

Through an enhanced version of SAS/DMI, available on request from SAS Institute Inc., it is possible to locate all Version 5 and Version 6 SAS data libraries. The main menu from the enhanced SAS/DMI Catalog Super Locate Application is shown in **Screen 1**. As you can see, this panel can be used to locate many types of files. However, this discussion only looks at the panels pertaining to SAS data libraries.

```
------------------------------ CATALOG SUPER LOCATE ------------------------------
OPTION  ===>

  0  TEMPORARY- Manipulate temporary file created by other options.
  1  DSNAME   - List DSnames which are not in a generation data group (GDG),
                GDG Summaries, and VSAM Objects.
  2  GDG      - List DSnames which are part of a GDG.
  3  EMPTY    - List empty alias indices, GDGs with no cataloged generations,
                and VSAM clusters not on cataloged volumes.
  3F          - Like 3 but IDCAMS statements clean-up catalog written to
                temporary file. (Save temporary file with option 0.6)
  4  SAS DATA - List DSnames which are DASD pre-SAS V6 data libraries.
  4F          - SAS statements to examine, release, copy, and convert pre-SAS V6
                DASD data libraries written to temporary file. (Submit with 0.1)
  4V6         - List DSnames which are DASD SAS V6 data libraries.
  4V6F        - SAS statements to ACCESS members of SAS V6 DASD libraries
                written to temporary file. (Submit with 0.1V6)
  5  RELEASE  - List DSnames of overallocated DASD files.
  5F          - SAS statements to release space from overallocated DASD
                files, including SAS data libraries, written to temporary
                file. (Submit with 0.1)
Please enter OPTION or press END key to exit.
```

Screen 1 Main Menu from SAS/DMI Catalog Super Locate Application

By selecting option 4F, several SAS/DMI panels display that allow you to tailor the selection process to your specific needs. You see the panels with the default values provided by option 4F. These values can be changed or additional values can be added. One of the important features of the Catalog Super Locate Options panel is that you can specify the selection name mask. Selections can range from all data libraries at your site to locating a specific data library. Here you see the selection of all files that begin with the prefix SASHML. You can also select by volume serial number or device type. In addition, you can select where to direct your output. All the information you need to supply is shown specified in **Screen 2**. Note, that these panels use the standard ISPF convention of preceding input fields with a => and output fields with a :.

```
------------------- CATALOG SUPER LOCATE OPTIONS ---------------------
 COMMAND ===>

Print Options Associated with Option 4F
  S Non-GDG DSnames _ VSAM Objects _ Alias Indices _ Empty Aliases
  _ GDG Summaries   S GDG DSnames  _ Empty GDGs    _ Missing Clusters
Print Values
  Output To      => FILE     (SAS|ISPF|BOTH|FILE)
  File Skeleton  => DMISLOC4 (DMISLOC|DMISLOC1-DMISLOC6|Your Skeleton)
Print Options
  _ Catalog Info   _ Volume Info    _ Data Set Info    S Space Info
Selection Values
  Name Mask        => SASHML.
  Class            => DASD     (Blank|TAPE|DASD|ARCHIVE|?)
  Device           =>          (See HELP for List of Valid Devices)
  Volume           =>          (Blank|Volser|Volser Prefix)
  Extended Select  => YES      (YES|NO)
Title => Search all Version 5 data sets Used within last two weeks
Workarea Size      => 839.5 k
Pause after        => 1000     catalog entries examined.

 Please press ENTER key to continue or END to return to previous panel.
```

Screen 2 Panel to Specify SAS/DMI SUPER CATALOG OPTIONS

One of the options that you may select from the panel shown in **Screen 2** is extended data set selection. If you answer YES, then a panel is displayed as shown in **Screen 3**. Selections by date are entered in two forms. You can use either the DDMONYYYY form or a relative date form for a specific date. To specify a relative date, enter a plus or minus sign followed by the number of intervals from today's date followed by the interval length. Use a period to specify that all dates be selected.

You can choose from the following selection criteria:

- creation date
- date of last use
- expiration date
- amount of tracks actually used
- amount of tracks free.

If you set a parameter equal to zero for the amount of free tracks used, you can locate all data sets that have been created but never used. All selection criteria are shown in **Screen 3**.

```
------------------- EXTENDED DATA SET SELECTION --------------------------
 COMMAND ===>

Title =>
Extended Selection Operators and Values
  Creation Date    EQ  .          (DDMONYYYY | Relative Date | Period)
  Last Use Date    EQ  .          (DDMONYYYY | Relative Date | Period)
  Expiration Date  EQ  .          (DDMONYYYY | Relative Date | Period)
  Organization     EQ  DA DAU
                       (DA DAU GS IS ISU PO POU PS PSU ?)
  Record Format    EQ  U

                       (F FA FB FBA FBM FBS FBSA FBSM FM FSA FSM U UA UM
                        V VA VB VBA VBM VBS VBSA VBSM VM VS VSA VSM ?)
  Record Length    EQ  -4         (Number | -4 | *Number | Period)
  Block Size       EQ  .          (Number | Period )
  Extents          EQ  .          (Number in range 0 - 16 | Period)
  Secondary Tracks EQ  .          (Number | Period)
  Allocated Tracks EQ  .          (Number | Period)
  Used Tracks      GT  0          (Number | Period)
  Free Tracks      EQ  .          (Number | Period)
Valid operators are EQ, NE, GT, LT, GE, and LE. Today is 22FEB1990 / 90.053
Please press ENTER key to continue or END to return to previous panel.
```

Screen 3 Panel to Specify SAS/DMI EXTENDED DATA SET SELECTION

Following one pass through the selection process, you go again to the main menu from the Catalog Super Locate application. You can select any of the following options for additional selection. Realize that you have not done any processing against the selected files. Based on your selection you have either created reports or the

statements necessary to process the selected files. When you have selected all the files to process, as illustrated in **Screen 4**, use option 0 to begin processing.

```
------------------------- CATALOG SUPER LOCATE -------------------------------
OPTION  ===> 0

 0  TEMPORARY- Manipulate temporary file created by other options.
 1  DSNAME   - List DSnames which are not in a generation data group
               (GDG), GDG Summaries, and VSAM Objects.
 2  GDG      - List DSnames which are part of a GDG.
 3  EMPTY    - List empty alias indices, GDGs with no cataloged
               generations, and VSAM clusters not on cataloged volumes.
 3F         - Like 3 but IDCAMS statements clean-up catalog written to
               temporary file. (Save temporary file with option 0.6)
 4  SAS DATA - List DSnames which are DASD pre-SAS V6 data libraries.
 4F         - SAS statements to examine, release, copy, and convert
               pre-SAS V6 DASD data libraries written to temporary file.
               (Submit with 0.1)
 4V6        - List DSnames which are DASD SAS V6 data libraries.
 4V6F       - SAS statements to ACCESS members of SAS V6 DASD
               libraries written to temporary file. (Submit with 0.1V6)
 5  RELEASE  - List DSnames of overallocated DASD files.
 5F         - SAS statements to release space from overallocated DASD
               files including SAS data libraries, written to temporary
               file. (Submit with 0.1)
Please enter OPTION or press END key to exit.
```

Screen 4 Main Menu from SAS/DMI Catalog Super Locate Application

Separating SAS Data Files from Other Types of Files

Once all the data libraries are selected by the criteria that you specified on the previous panels, you can choose what you want to do with the information. By choosing option 0 of the main panel (refer to **Screen 1**) the panel shown in **Screen 5** displays and allows choices on ways to manipulate the files that you have selected.

```
------------------- MANIPULATE TEMPORARY FILE --------------------------------
OPTION  ===> 1

  1   Submit Temporary File to SAS V5 System for Execution
 1V6  Submit Temporary File to SAS V6 System for Execution
      V6 CLIST Name => SAS606
      CLIST Options =>
        SAS Options => AUTOEXEC=ISPCTL1 IS=*ENDSAS;* NONEWS NOOPLIST NOSOURCE
  2   Submit Temporary File to JES for Batch Execution (Provide JCL
      with Opt 4)
  3   Browse Temporary File
  4   Edit Temporary File
  5   Copy Temporary File to ISPF List Data Set
  6   Copy Temporary File to DSname Entered Below
      =>
  7   Empty Temporary File

Temporary File DDname: ISPCTL1 DSname:
SYS90059.T121546.RA000.SASHML.R0000002

Please enter OPTION or press END key to exit.
```

Screen 5 Panel to Manipulate Temporary File

By choosing option 1, you can submit the temporary file to the SAS Version 5 system for execution. A new panel displays allowing you to manipulate the data libraries selected on the previous panels. The panel shown in **Screen 6** contains various options available to process the data libraries.

```
------------------- DASD V5 DATA LIBRARY MENU --------------------------------
OPTION  ===>

       Old Library: SASHML.CONSULT.USAGE
            Volume: SAS821 TRKS per CYL: 15       Device: 3380
           Extents: 1    Secondary TRKS: 15       SPACE: (CYL,(7,1))
   Allocated TRKS: 105       Used TRKS: 97    Free TRKS: 8

    New library =>
   Release to USED space plus => 10    => %    (%,TRKS, or CYLS)
   COPY/CONVERT File Types:  S DATA  S CAT  S GCAT  S IMLNK  S MODEL

1    DATASETS of old library         1N  DATASETS/ACCESS of new library
2    RELEASE of old library          2N  RELEASE of new library
3    CONTENTS of old library         3N  CONTENTS of new library
4    Display old space by file type  4N  Display new space by file type
5    Delete old library              5N  Delete new library
6V5  Allocate new V5 library         6V6 Allocate new V6 library
7V5  COPY files to new V5 library    7V6 Convert files into new V6 library
                                     8V6 Generate V5TOV6 statements to
X    End This Application Immediately    convert files into new V6 library

Please enter OPTION or press END key to get next selected library.
```

Screen 6 Panel Submit Temporary File to SAS Version 5 for Execution

You can use this panel to

- use PROC DATASETS to display all the data files for the old library and delete any files not currently being used.
- release (with PROC RELEASE) space from the old library.
- display how much space is being used by each file type and for each of the selected files.
- allocate a new Version 5 library, if needed, with space already determined based on which types of files are selected.
- copy all file types selected from old library into new library.
- access any other ISPF application, using a split screen. You may want to split the screen and use ISPF/PDF option 3.14 to search program source for all references to the old library and make changes to the source based on the newly created Version 5 library.
- test your newly modified applications by splitting the screen and invoking Version 5 of the SAS System.

Once Version 6 is installed at your site you could use this same panel to

- run PROC ACCESS against a Version 6 library to find out information about files contained in the library. PROC ACCESS is a SAS Version 6 tool for managing SAS libraries. It allows you to delete, rename, and list members in a SAS data library, and also provides specialized functions for specific members.
- release (with PROC RELEASE) any unused space on a Version 6 library.
- run PROC CONTENTS to find more specific information about contents of data files.
- display space information on the Version 6 library by file type.
- delete the library.
- allocate a Version 6 library with space allocations determined by the selection criteria for the Version 5 library.
- run the V5TOV6 procedure interactively and copy all file types that have been selected.

- delete files on the new Version 6 library.

This process is done interactively, but at any time you can interrupt the process and continue at a later time. The temporary file still contains all the necessary information about the files that were located, and you can save them into a permanent file for processing later.

Converting Version 5 SAS/AF Applications to Use the SAS Autocall Facility

To convert your SAS/AF applications to use the SAS autocall facility, you need to create a macro from the program screen and store the macro externally in a SAS autocall library. For example, if your site has installed the SAS autocall library provided by SAS Institute Inc., you can simply specify the MAUTOSOURCE option and invoke those macros as needed. The CLIST supplied by SAS Institute contains an AUTOS option that can be used to access one or more of your SAS autocall libraries. Use their names as the value of the AUTOS option in the SAS command, as in

```
sas autos('''company.autocall.lib''')
```

or

```
sas autos('''finance.lib1.autoxx'' ''company.autocall.lib''')
```

if you need to concatenate several libraries.

Note that you should not convert any SAS/AF PROGRAM entries to use the SAS autocall facility if they contain conditional execution indicators, a pound sign (#), or are computer-based training applications containing these indicators. Version 6 cannot properly convert this type of PROGRAM entry if it is stored externally in the SAS autocall facility.

Refer to **Screen 7** for an example of how a SAS/AF PROGRAM entry could be converted.

```
Command ===>                                                  Text Editor
                                                              Col 001 079
          Put an X beside the data base you want to edit:
                &s1 Student     &s2 IDB
------------------------------------------------------------------------------
 %MACRO EDITIT;
 %TSO ALLOC FI(SAVE) DA('SASADM2.EDSALES.MASSDB')  OLD REUSE ;
 %IF &SYSRC=0 %THEN %DO; * Data set is not in use;
   %IF %upcase(&s1)=X    %THEN %DO;
      PROC FSEDIT DATA=SAVE.STUDENTS SCREEN=SAVE.STUDENTS;
      RUN;
   %END;
   %ELSE %upcase(&S2)=X %THEN %DO;
      PROC FSEDIT DATA=SAVE.idb SCREEN=SAVE.idb ;
      RUN;
   %END;
 %ELSE %DO;
    %put Data set is in use by another user, try again later;
    %put Press enter when three stars appear.;
     %input;
 %END;
%MEND EDITIT;
%EDITIT;
 PROC DISPLAY C=SYS.SYSTEM.MAIN.MENU; RUN;
```

Screen 7 Version 5 SAS/AF PROGRAM Entry before Conversion to SAS Autocall Facility

By using the code given in **Example 1** in **Appendix 1**, you could convert this code to use the SAS autocall facility by creating a member by the name of EDITIT and also giving the macro the same name. Note that you would need to make sure that if you are using macro variables from on the PROGRAM entry that you assign an associated SAS macro variable as shown in **Screen 8** .

```
                Attributes for PROGRAM  & SYSTEM screens
Command ===>

Please use the scroll commands or function keys to review the fields.

Field Name: S1     Length:   3  Row:   4  Column:  23

    Type: CHAR     Caps: X  Just: L  Pad: _  Protect: _  Non-display: _

  Format:               Informat:                          Required: _

    List:

    Help: ........ ........          Associated SAS Macro variable: S1

Field Name: S2     Length:   3  Row:   5  Column:  23

    Type: CHAR     Caps: X  Just: L  Pad: _  Protect: _  Non-display: _

  Format:               Informat:                          Required: _

    List:

    Help: ........ ........      Associated SAS Macro variable: S2
```

Screen 8 Version 5 SAS/AF PROGRAM Entry Attribute Definition Panel

Screen 9 shows what the PROGRAM entry looks like after conversion to the SAS autocall facility.

```
Command ===>                                                  Text Editor
                                                              Col 001 079
          Put an X beside the data base you want to edit:
                &s1 Student   &s2 IDB
------------------------------------------------------------------------------
 %EDITIT;
 %DISP;
```

Screen 9 Version 5 SAS/AF PROGRAM Entry after Conversion to the SAS Autocall Facility

If you design your macros properly, you could pass a parameter at SAS invocation and call whichever version of SAS data libraries you wanted at the time. More specifically, if you included a release in your naming conventions for both SAS data libraries and SAS macro libraries, you would not need to change the actual SAS/AF macros themselves when you moved to Version 6.

The program in **Example 2** in **Appendix 1** shows how once you have your programs stored in a SAS autocall library, you can use SAS programs to manipulate your programs in various ways.

Converting Allocate Statements to LIBNAME Statements

It is not necessary to convert allocate statements to LIBNAME statements unless you want to take advantage of some of the new features that Version 6 offers. However, once your application is running on Version 6, by specifying a library engine on a LIBNAME statement, your application will use less resources and be more efficient.

In Version 6, in order to use files in a SAS data library, you must perform two steps:

1. allocate the library
2. assign the library to a library engine.

You perform both these activities through the LIBNAME statement. Briefly, engines provide the ability to access a variety of different types of files as if they were SAS data sets. One of the engines is the Version 5 compatibility engine, which allows Version 6 access to Version 5 data sets on disk. Version 6 can recognize a Version 5 format data set without specifically specifying an engine on the LIBNAME statement, but without the LIBNAME statement the data set does not appear in LIB/DIR windows until referenced. If you do not convert your allocate statements to LIBNAME statements in Version 6, you will need to issue a LIBNAME CLEAR statement before each TSO FREE FILE statement so the FREE command can be recognized.

The LIBNAME statement is currently available in Version 5 in a simpler form than it is in Version 6. In Version 5 an allocate statement such as,

```
%TSO ALLOC F(SAVE) DA('USER.SAS.SASDATA') OLD ;
```

would be replaced by

```
%LIBNAME SAVE 'USER.SAS.SASDATA' DISP=OLD;
```

If you are using %TSO statements and are checking the return code using &SYSRC, and you convert to %LIBNAME, you can use &SYSLIBRC to check the return code.

PROGRAMMING TECHNIQUES AND TOOLS FOR THE CMS HOST SYSTEM

Separating Data Files from Other Types of Files

It is standard in the MIS department of SAS Institute Inc. to use a variety of separate minidisks. For example, the order entry data files are on a commonly named minidisk for each user. Also, 298 could be defined as the common minidisk for order entry data files for all userids. The screen catalogs associated with these data files are stored on another minidisk that is shared by all users. The users also share the program source code, which is located on still another minidisk.

Before these standards were established, a catalog was created for userid SASPUBA that contained a screen called DATAPUBA.ORDSCR and a data file called DATAPUBA.ORDER. The observations can be edited after specifying the following command:

```
PROC FSEDIT DATA=DATAPUBA.ORDER
SCREEN=DATAPUBA.ORDSCR
```

As more userids are added, the maintenance grows exponentionally because each time a request is made for a screen change the minidisk for each userid has to be accessed and modified. If each userid has a copy of the source code on the same minidisk with the data files and screens, then each program has to be changed when a request is made by the user.

By implementing this standard, all screen catalogs are placed on one minidisk that can be shared by multiple users, which allows all userids access to identical screens. Maintenance is reduced because only one copy exists and modifications are simplified. Please note that the program source code will need to be modified to reference the minidisk containing the screen catalogs. For example, all screens relating to the Publications CMS applications are located on a minidisk with the FILETYPE of PUBSCR. The following command can be issued to access the observations in the above example:

```
PROC FSEDIT DATA=DATAPUBA.ORDER
SCREEN=PUBSCR.ORDSCR
```

In converting to Version 6 different file types may be converted at different times. The screen catalogs must be converted to use Version 6 while the data files can still be read in Version 5. However, Version 5 and Version 6 format SAS data files cannot reside on the same minidisk if they have the same file type.

The standard of separating catalogs to different minidisks is one way of handling the conversion of different catalogs at different times. It also allows a technician to locate with ease the screen catalogs for conversion. When you put screen catalogs on a separate minidisk, it is easy to issue the V5TOV6 procedure for conversion of all the screens at one time. When you are ready to use Version 6, define a second minidisk with a unique file type for the converted Version 6 screen catalog from the V5TOV6 procedure. Once the conversion is complete, rename the file type for the Version 6 catalog to the same name as the Version 5 catalog. By defining this new minidisk you can keep the Version 5 screen catalog intact.

After the screen catalog is converted to Version 6, you can modify the screen catalog to take advantage of Version 6 enhancements. By having both the Version 5 and Version 6 screen catalogs in place, users experience minimal interruption while Version 6 enhancements are added. Keeping a Version 5 and Version 6 catalog also allows parallel testing, and if a situation arises under Version 6, then users can easily be moved back to Version 5 by switching the minidisk the user is linked to. By having a separate minidisk for the screen catalog, no program source code changes are needed when converting from Version 5 to Version 6.

Four points should be emphasized here.

- separating catalogs to different minidisks makes it easy to locate information for daily use as well as conversion
- separating catalogs to different minidisks reduces maintenance time by minimizing the number of screen and program source code changes
- separating catalogs allows conversion to occur at different times
- creating a new minidisk for the output from the V5TOV6 procedure makes it easy for technicians to determine what version a user is running by checking to see which minidisk the user is linked to as well as providing a parallel testing environment and an opportunity to implement some of the Version 6 enhancements while still converting other parts of the same system.

Converting Applications that Use Macros To Use the Autocall Facility

The autocall facility allows the developer to store macro source statements in a maclib, to associate the maclib with the SAS session, and to invoke the macros as needed. The autocall facility eliminates the need to create the macro in the program or to bring the macro into the program with a %INCLUDE statement before you can invoke the macro. Using this facility, a developer can associate a macro with each function key in an application; the macro is loaded and executed only when the user selects the function key option.

Once the SAS autocall library (or maclib) is established and the macro source statements moved to the maclib, the maclib can be associated with the SAS session. One way to accomplish this goal is to modify the SAS EXEC.

The SAS EXEC supplied by SAS Institute for the CMS Operating System issues a FILEDEF command to associate the autocall maclib supplied with base SAS software with the filedef

SASAUTOS. If your site has installed the autocall libraries for the products licensed, and if the necessary changes were made to the SAS EXEC during Installation, you can invoke the macros in the autocall library or libraries without making changes to your SAS program. For example, suppose your site licenses base SAS, SAS/GRAPH®, SAS/ETS®, and SAS/SHARE® software products and installs the autocall library for each product. Since a FILEDEF command is given for base SAS in the SAS EXEC, a queue does not need to be defined. In the SAS EXEC, a stack of the SAS product maclib filenames are built as follows:

```
SAS EXEC Comment
/* Build stack with all possible SAS product maclib filenames.  */
/* If you have your own macro autocall maclib, be sure to add a */
/* queue for it.                                                */

  queue 'SASGAUTOS'          /* SAS/GRAPH */
  queue 'SASEAUTOS'          /* SAS/ETS   */
  queue 'SASHAUTOS'          /* SAS/SHARE */
  'FILEDEF SASAUTOS DISK SASBAUTO MACLIB * (PERM'
  GLMACS=SASBAUTO
```

By concatenating your maclib with the other autocall maclibs in the SAS EXEC, you can invoke your personal macros in exactly the same way you invoke the macros supplied by SAS Institute. To concatenate personal maclibs such as SASTEST, PUBTEMP, and PUBAUTOS in front of the maclibs in the above example, you must create a modified SAS EXEC, making sure that a queue is established for each maclib as shown below.

```
SAS EXEC Comment
/* Build stack with all possible SAS product maclib filenames.  */
/* If you have your own macro autocall maclib, be sure to add a */
/* queue for it.                                                */

  queue 'SASTEST'            /* personal maclib */
  queue 'PUBTEMP'            /* personal maclib */
  queue 'PUBAUTOS'           /* personal maclib */
  queue 'SASGAUTOS'          /* SAS/GRAPH */
  queue 'SASEAUTOS'          /* SAS/ETS   */
  queue 'SASHAUTOS'          /* SAS/SHARE */
  'FILEDEF SASAUTOS DISK SASBAUTO MACLIB * (PERM'
  GLMACS=SASBAUTO
```

The SAS EXEC creates the needed FILEDEFS for each maclib and issues the CMS GLOBAL command based on the maclib filenames queued. Please contact your SAS Software Consultant regarding procedures for modifying the SAS EXEC at your site.

The MAUTOSOURCE system option must be in effect for you to invoke autocall macros. This option can be specified at SAS invocation or in an OPTIONS statement. When a macro is called and the SAS system option MAUTOSOURCE is in effect, the system searches the appropriate maclib for a member with the same name as the macro called. The SAS System searches the maclibs in the order they appear in the EXEC and uses the first member found with the requested name. When the SAS System finds the member, it is loaded and executed.

The SAS EXEC in the previous example was modified to reference three maclibs (SASTEST, PUBTEMP, and PUBAUTOS), which create

- a test library for testing all changes before putting them into production
- a temporary library consisting of daily changes only
- a production maclib that cannot be edited.

The naming conventions for personal maclibs is at the discretion of the user. Below is an explanation of the purpose of each maclib and the guidelines used for updating them.

SASTEST Maclib

The SASTEST maclib is used exclusively for testing purposes. It is an empty (dummy) maclib residing on the production minidisk. As stated earlier, the SAS EXEC was modified so that the SAS System would search this maclib first for the called macro. A programmer wanting to test a macro would copy the empty SASTEST maclib from the production disk to his A disk and copy the macro he wanted to test from the A disk into the maclib. (Later, additional execs will be discussed that serve this particular purpose.) By executing the SAS System with the MAUTOSOURCE system option and calling the macro, the programmer can easily test the changes made because this version of code only resides on the A disk of the tester. At the same time the user who is linked to the production disk and would not want to pick up test data is Instead referencing the empty SASTEST maclib on the production disk.

PUBTEMP Maclib

The PUBTEMP maclib is used exclusively for putting new code into production during the day. The PUBTEMP maclib is the second maclib searched when a macro is called. When a macro is invoked the SAS System first looks in the SASTEST maclib and if there is no member with the same name as the macro called, the SAS System searches the PUBTEMP maclib. The developer, after testing new code in the SASTEST maclib, logs onto the production userid (which is the owner of the minidisks the maclibs reside on) and puts the changes into the PUBTEMP maclib. At night a batch job runs and updates the production maclib (PUBAUTOS) with the changes from PUBTEMP. The job ends by clearing out the PUBTEMP maclib so the programmer has an empty MACLIB to work with the next day.

There are two major advantages to updating the PUBTEMP maclib rather than updating the production maclib (PUBAUTOS). First, the PUBTEMP maclib is much smaller than the master maclib, making it easier and faster for the programmer to work with. Updating and replacing a large maclib can be time consuming. Second, it is more likely that the production maclib could be accidentally deleted if there are frequent updates made to it by many different developers.

PUBAUTOS Maclib

PUBAUTOS is the production maclib. All updated macros and new code end up in this maclib. The maclib is large, and replacing it takes a great deal of time. Therefore, it is best to work with the PUBTEMP maclib (which is empty each morning) and update PUBAUTOS from PUBTEMP each evening with a production job.

Here, we introduce some EXECs developed to assist in retrieving and adding members to the maclibs. The following EXECs are much faster than using SPF to edit the maclib.

GETMEM EXEC	extracts a given number from a given maclib and puts it on your A disk with a FILETYPE of SAS. If the member does not exist in the maclib, CMS returns an error "open error code 08."
ADDMEM EXEC	puts SAS code into a given maclib. If the maclib does not exist, it creates one.
MADDMEM EXEC	is used while logged onto the production userid only. It puts SAS code into a given maclib and erases the code from your A disk. If the maclib does not exist, it does not create one. In addition, this EXEC renames the macro being replaced and puts it out on our production disk, providing us a backup; it is also a

reference to the old code should it be needed.

As an example, let's modify a macro named SALES.

To retrieve a copy of the SALES SAS macro from the production maclib use the following command:

```
GETMEM PUBTEMP SALES
```

The PUBTEMP maclib is checked first because a programmer might have changed the SALES macro today. If CMS returns an "open error code 08"then we know the macro has not been updated today. Instead, specify the following command to get the macro from the production maclib (PUBAUTOS):

```
GETMEM PUBAUTOS SALES
```

The GETMEM EXEC makes a copy of the macro SALES from SASAUTOS, putting it on the A disk and naming it SALES SAS (SALES for the name of the macro and it is always given a FILETYPE of SAS). The SALES SAS program can now be edited, making the necessary changes. When the changes are complete it is added to the SASTEST maclib by using the command:

```
ADDMEM SASTEST SALES
```

ADDMEM EXEC makes a copy of the SALES SAS file on the A disk, putting the copy in the SASTEST maclib. It leaves the SALES SAS file on the A disk so that further changes can be made to the program code if necessary. To test the changed SALES macro in the SASTEST maclib, execute SAS with the MAUTOSOURCE options and invoke the SALES macro. When finished testing, log onto the production userid (which is the owner of the minidisks that the maclibs reside on). Once on the production userid, link to the location of the SALES SAS program that was tested and use the CMS COPY command.

```
COPY SALES SAS C = = A
```

This copies the SALES SAS program over to the production A minidisk. Use the CMS COPY command again to copy the PUBTEMP maclib from the production disk to the A disk (we don't want to change the same maclib that users are accessing).

```
COPY PUBTEMP MACLIB C = = A
```

Use the MADDMEM EXEC command specified below to copy the SALES SAS program from the A disk into the PUBTEMP maclib.

```
MADDMEM PUBTEMP SALES
```

Before the MADDMEM EXEC updates the PUBTEMP maclib, it stores a copy of the SALES program on the production minidisk. It is a good idea to keep the "before changed" copy of the SALES macro in case you need to refer to it at a later date.

Last, replace the newly updated PUBTEMP maclib back to the production minidisk so the user can access the "latest" updates. Use the command

```
CREPMAC PUBTEMP SALES
```

The CREPMAC EXEC copies the PUBTEMP maclib over to the production disk after renaming the old one. All you need to do is reaccess the production disk to get the latest SALES macro.

Using the autocall facility not only improves performance but also simplifies the system design process, maintenance, and the support resulting from having the macro program code reside in one location, the maclib. Implementing modularity, by structuring the systems using the autocall facility, provides the developers a single reference point for all system code. As a result, complexity of the systems is reduced.

Sample code for the GETMEM, ADDMEM, MADDMEM, CREPMAC EXECs is given in **Example 1** in **Appendix 2**.

Adding An Equal Sign between Value Options for CMS SAS Invocation

Value options are one type of SAS system option. Value options have many possible settings, which are specified with an identifying keyword followed by a value. Under Version 5, an equal sign must be used between the keyword and the value in an OPTIONS statement, but is optional on the SAS command. For example,

Options statement	`options linesize=72;`
SAS commanmd	`sas (linesize 72`
SAS command	`sas (linesize=72`

In Version 6, an equal sign **must** be used for value options specified on the SAS command. An exec was written by MIS to add the equal sign between the value options specified on the SAS command. The SAS tool is shown in **Example 2** in **Appendix 2**.

CONCLUSION

At your site you should consider having the first conversion done by your most experienced SAS programmers. They should document their conversion experience and share this information with others responsible for conversion. Your management will have to decide the order that SAS applications are converted based on the benefits to be gained from SAS Version 6. You can receive immediate benefits and simplify your conversion to Version 6 by implementing now the techniques discussed in this paper.

APPENDIX 1: MVS EXAMPLES

The BUILD procedure in batch mode can be used to either print a hardcopy of your SAS/AF programs or send the output to a data set. **Example 1** shows the Version 5 PROC BUILD syntax used to output all the program screens to a file. This enhancement to the BUILD procedure is documented in Chapter 2 of SAS Technical Report P-146 (1986). You must have an existing form in a catalog referenced by the DDNAME SASUSER, and it must be specified with the disposition of OLD. In this application, there was no OCL specified in the form E3700, only the minimum information was specified in the form. You may specify a select list following the PROC BUILD statement.

Example 1

```
//STEP1 EXEC SAS518
//SASUSER DD DSN=user.SAS.SASUSER,DISP=OLD
//OLD DD DSN=USER.SAS.OUTPUT1,DISP=(NEW,CATLG),  <== CREATE OUTPUT FILE
// VOL=SER=XXXXX,SPACE=(TRK,(5,5),RLSE),
//OCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//SAVE DD DSN=user.MEOIAV5.SYSTEM,DISP=SHR

 PROC BUILD  C=SAVE.SYSTEM  FORM=E3700 PRTFILE=OLD;
/*
//STEP2 EXEC SAS518
//IN DD DSN=user.SAS.OUTPUT1,DISP=SHR
//OUT  DD DSN=&&TEMP,UNIT=OISK,DISP=(NEW,PASS),
// SPACE=(TRK,(10,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)

  /* Read OUTPUT1 file from previous step and separate all SAS/AF    */
  /* programs into separate members to be put into a PDS library in  */
  /* the third step. Note that a %MACRO member_name and              */
  /* %MEND member_name are created for you                           */
DATA _NULL_;
LENGTH LINE $ 72 LMEM MEMBER $ 8;
IF _N_=1 THEN LINE=REPEAT('-',72);
RETAIN ;
INFILE IN END=EOF;
```

```
INPUT TEXT $CHAR132.;
FILE OUT;
IF INDEX(TEXT,'.PROGRAM')¬=0 THEN DO;
   MEMBER=LEFT(SCAN(TEXT,1,'.'));
   IF LMEM¬=MEMBER THEN DO;
     IF LMEM¬=' ' THEN
     PUT '$MEND ' LMEM ' ;';
     PUT './' @9 'ADD NAME=' MEMBER;
     PUT '$MACRO ' MEMBER ' ;';
     FLAG2=0;
   END;
   FLAG=1;
   LMEM=MEMBER;
END;
IF FLAG=1 AND SUBSTR(TEXT,2,72)=LINE THEN FLAG2=1;
IF FLAG=1 AND FLAG2=1 AND SUBSTR(TEXT,2,72)¬=LINE AND
   INDEX(TEXT,'.PROGRAM')=0 AND SUBSTR(TEXT,1,1)¬='1' THEN
   PUT TEXT $CHAR80.;
/*
//STEP3 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSPRINT DD SYSOUT=A
//SYSUT2 DD DSN=user.GEN.MACROS,DISP=SHR
//SYSIN DD DSN=&&TEMP,DISP=(OLD,DELETE)
```

Example 2

Once you have your programs stored in a SAS autocall library, you can use a program such as the following to create listings, make changes to current macros, and create a new library for Version 6 processing. Note that by using the OPENMEM function you can open multiple members of a PDS in one DATA step.

```
// EXEC SAS518
//VERSION5 DD DSN=user.EDUC.MACROS,DISP=SHR

   /* locate all members of the PDS library */

DATA DIRECTRY;
      INFILE VERSION5 RECFM=U BLKSIZE=256 LRECL=256; COL=3;
      INPUT COUNT PIB2. @; IF COUNT<14 THEN STOP;
      DO WHILE(COL<COUNT);
      INPUT @COL MEMBER $8. +3 IND PIB1. @;
      IF MEMBER>'99999999' THEN STOP;
      COL=COL+12+2*MOD(IHD,32);
      OUTPUT; END;
      KEEP MEMBER;

   /* open PDS library with the use of the OPENMEM function and   */
   /* find all references to allocate statements, libnames or     */
   /* any SAS/FSP references                                      */

DATA _NULL_;SET DIRECTRY;
   RC=0;
   INFILE VERSION5 UNBUFFERED END=EOF;
   CALL OPENMEM(MEMBER,RC);
   FILE PRINT;
   DO WHILE(¬EOF);
      INPUT TEXT $CHAR80.;
      IF INDEX(UPCASE(TEXT),'ALLOC')¬=0 OR
         INDEX(UPCASE(TEXT),'LIBNAME')¬=0 THEN
          PUT /'MEMBER: ' MEMBER /' ' TEXT;
       IF INDEX(UPCASE(TEXT),'FSEDIT')¬=0 OR
          INDEX(UPCASE(TEXT),'LETTER')¬=0 OR
          INDEX(UPCASE(TEXT),'SCREEN')¬=0 OR
          INDEX(UPCASE(TEXT),'C=')¬=0 OR
          INDEX(UPCASE(TEXT),'CAT=')¬=0 OR
          INDEX(UPCASE(TEXT),'CATALOG=')¬=0 THEN
              PUT TEXT;
    END;
```

APPENDIX 2: CMS EXAMPLES

Example 1

GETMEM EXEC Extracts a member from a maclib and puts a copy on your A disk with a FILETYPE of SAS.

```
 /* EXTRACT MEMBER FROM ANY MACLIB      */
 /* AND COPY TO YOUR A MINI DISK        */
 /* SYNTAX   GETMEM MEMBER MAC          */
 /* WHERE MEMBER IS THE MEMBER OF THE   */
 /* MAC (MACLIB)                        */
 PARSE UPPER ARG  MEMBER MAC

IF MEMBER='' | MAC='' THEN DO;
    SAY 'SYNTAX : GETMEM MEMBER MAC';
    EXIT;
END;

'STATE ' MAC ' MACLIB *'
IF RC¬=0 THEN DO;
    SAY 'THE MACLIB ' MAC ' IS NOT FOUND';
    EXIT;
END;

'FILEDEF INMOVE DISK ' MAC ' MACLIB * (MEMBER'  MEMBER
'FILEDEF OUTMOVE DISK' MEMBER 'SAS A'
'MOVEFILE'
XXRC=RC
'FILEDEF INMOVE CLEAR'
'FILEDEF OUTMOVE CLEAR'
EXIT XXRC
```

ADDMEM EXEC Puts SAS CODE in a maclib. If the maclib does not exist, it creates one.

```
 /*-------------------------------------------------------------*/
 /*  THIS EXEC WILL PUT CODE OF FILETYPE SAS INTO A GIVEN   */
 /*  MACLIB.                                                */
 /*-------------------------------------------------------------*/
 /*                                                         */
 /* SYNTAX:   ADD FN MAC                                    */
 /*                                                         */
 /*       WHERE    FN IS THE FILENAME OF THE PROGRAM       */
 /*                FILETYPE IS SAS                         */
 /*                MAC IS THE NAME OF THE MACLIB           */
 /*-------------------------------------------------------------*/
TRACE OFF

PARSE UPPER ARG FN MAC;

 /* CHECK SYNTAX */
IF FN='' | MAC='' THEN DO;
    SAY 'SYNTAX: FN MAC';
    EXIT;
END;

 /* CHECK IF FILE EXISTS */
SET CMSTYPE HT
STATE FN SAS A
RCC=RC
SET CMSTYPE RT
IF RCC¬=0 THEN DO;
    SAY 'WE CANNOT FIND THE FILE' FN SAS ' ON YOUR A DISK';
    EXIT;
END;

 /* CHECK IF MACLIB EXITS */
XX=0;
SET CMSTYPE HT
'STATE ' MAC ' MACLIB *'
RCC=RC
SET CMSTYPE RT
IF RCC¬=0 THEN DO;
    SAY 'WE CANNOT FIND THE MACLIB ' MAC;
    SAY  'CREATING MACLIB ';
    XX=1;
END;
 /* CHECK IF FILE EXISTS ON THE A DISK WITH FILETYPE OF COPY */
SET CMSTYPE HT
STATE FN COPY A
RCC=RC
SET CMSTYPE RT
IF RCC=0 THEN DO;
    SAY 'WE NEED TO RENAME YOUR SAS FILE TO HAVE A FILETYPE OF COPY';
    SAY 'SO THAT WE CAN ADD THE FILE TO THE MACLIB.  YOU ALREADY   ';
    SAY 'HAVE THIS FILE ON YOUR A DISK AND YOU NEED TO EITHER DELETE';
    SAY 'OR RENAME THIS FILE.';
    EXIT;
END;

 /* RENAME THE FILETYPE OF THE NEW SAS CODE TO COPY AND ADD TO MACLIB */
RENAME FN SAS A FN COPY A
IF XX=0 THEN DO;
   MACLIB REP MAC FN;
   END;
```

```
IF XX=1 THEN DO;
   MACLIB GEN  MAC FN;
   END;
IF RC=4 THEN DO;
    SAY 'THE SAS SOURCE CODE ' FN ' WAS NOT IN ' MAC ' MACLIB.';
    SAY 'THE SAS SOURCE CODE WAS ADDED TO THE ' MAC ' MACLIB.';
END;

RENAME FN COPY A FN SAS A
```

MADDMEM EXEC Moves SAS code into a maclib after renaming any existing macro of the same name. Renamed macro is moved to the production disk.

```
 /*-------------------------------------------------------------*/
 /*  THIS EXEC WILL PUT CODE OF FILETYPE SAS INTO A GIVEN   */
 /*  MACLIB.                                                */
 /*-------------------------------------------------------------*/
 /*                                                         */
 /* SYNTAX:   ADD FN MAC                                    */
 /*                                                         */
 /*       WHERE    FN IS THE FILENAME OF THE PROGRAM        */
 /*                FILETYPE IS SAS                          */
 /*                MAC IS THE NAME OF THE MACLIB            */
 /*-------------------------------------------------------------*/
TRACE OFF

PARSE UPPER ARG FN MAC;

 /* CHECK SYNTAX */
IF FN='' | MAC='' THEN DO;
    SAY 'SYNTAX: FN MAC';
    EXIT;
END;

 /* CHECK MACLIB */
IF MAC¬='PUBTEMP' THEN DO;
    SAY 'THIS EXEC SHOULD ONLY BE USED ON PUBTEMP';
    EXIT;
END;

 /* CHECK IF FILE EXISTS */
SET CMSTYPE HT
STATE FN SAS A
RCC=RC
SET CMSTYPE RT
IF RCC¬=0 THEN DO;
    SAY 'WE CANNOT FIND THE FILE' FN SAS ' ON YOUR A DISK';
    EXIT;
END;

 /* CHECK IF MACLIB EXITS */
SET CMSTYPE HT
'STATE ' MAC ' MACLIB A'
RCC=RC
SET CMSTYPE RT
IF RCC¬=0 THEN DO;
    SAY 'WE CANNOT FIND THE MACLIB ON THE A DISK' MAC;
    EXIT;
END;
 /* CHECK IF FILE EXISTS ON THE A DISK WITH FILETYPE OF COPY */
SET CMSTYPE HT
STATE FN COPY A
RCC=RC
SET CMSTYPE RT
IF RCC=0 THEN DO;
    SAY 'WE NEED TO RENAME YOUR SAS FILE TO HAVE A FILETYPE OF COPY';
    SAY 'SO THAT WE CAN ADD THE FILE TO THE MACLIB.  YOU ALREADY   ';
    SAY 'HAVE THIS FILE ON YOUR A DISK AND YOU NEED TO EITNER DELETE';
    SAY 'OR RENAME THIS FILE.';
    EXIT;
END;

 /* COPY CURRENT SAS PROGRAM IN MACLIB AS BACKUP */
ACC 130 C
'FILEDEF INMOVE DISK ' MAC ' MACLIB * (MEMBER' FN
FILEDEF OUTMOVE DISK FN ZZZZZZ C
MOVEFILE
   BACKUP
ERASE  FN OOOOSAS C
RENAME FN OOOSAS C FN OOOOSAS C
RENAME FN OOSAS  C FN OOOSAS  C
RENAME FN OSAS   C FN OOSAS   C
 /* RENAME CURRENT VERSION FROM MACLIB */
RENAME FN ZZZZZZ C FN OSAS    C
ERASE FN SAS C
COPYFILE FN SAS A FN SAS C

 /*RENAME THE FILETYPE OF THE NEW SAS CODE TO COPY AND ADD TO MACLIB*/
RENAME FN SAS A FN COPY A
MACLIB REP MAC FN
IF RC=4 THEN DO;
    SAY 'THE SAS SOURCE CODE ' FN ' WAS NOT IN PUBTEMP  MACLIB.';
    SAY 'THE SAS SOURCE CODE WAS ADDED TO THE PUBTEMP  MACLIB.';
END;
 /* DISCARD FROM A DISK */
ERASE FN COPY A

FILEDEF INMOVE  CLEAR
FILEDEF OUTMOVE CLEAR

'ACC 130 C/C'
```

CREPMAC EXEC Copies a file to the production disk after renaming any existing file of the same name.

```
  /* REXX */
trace off
  PARSE upper ARG FN FT FM
  /* REPLACE FILES ON THE C DISK  */
  /* OLD FILES ARE RENAMED SO THEY DON'T GET CLOBBERED.*/
  /* Reminds user to update history of program changes.*/
say;
say '??? Did you change your header.... if not use HEADER SAS'
say '??? In MACROS header must go after %MACRO '
say;

  ACC 130 C

 if fn='PUBTEMP' & ft='MACLIB' then
  do
   erase pubtemp maclib25 C
    do I=25 to 0 by -1
      if i=0 then
        rename pubtemp maclib c pubtemp maclib0 c0
       else
        rename pubtemp 'maclib'i-1 c pubtemp 'maclib'i c0
    end
  end

  else
   do
    erase  fn 'OOO'ft c
    rename fn 'OO'ft  c   fn 'OOO'ft  c0
    rename fn 'O'ft   c   fn 'OO'ft   c0
    rename fn    ft   c   fn 'O'ft    c0
   end

   'fcopy' fn ft 'a = =  C (OLDD'
 /* ERASE FROM A DISK SO THAT WE WILL NOT FILL UP WITH JUNK */
    if rc = 0 then
      erase fn ft a
     'acc 130 c/c'
```

Example 2: SAS System Options

```
/*********************************************************************/
/*                                                                   */
/*    PURPOSE: TO REFORMAT ANY CMS EXEC FILE WHICH INVOKES THE VERSION */
/*             5 SAS SYSTEM AND CONTAINS VERSION 5 SAS SYSTEM OPTIONS  */
/*             WHICH MAY OR MAY NOT REQUIRE EQUAL SIGNS                */
/*                                                                   */
/*      NOTES: REGARDLESS OF WHERE THE FILE IS LOCATED THE NEW         */
/*             FORMATTED CMS EXEC FILE WILL BE LOCATED ON THE 191      */
/*             NINIDISK. THE PROGRAM ALSO CHECKS THE FILE TO SEE IF IT */
/*             ACTUALLY INVOKES THE SAS SYSTEM AND IF THE FILE WAS     */
/*             WRITTEN IN REXX OR EXEC2.                               */
/*                                                                   */
/*********************************************************************/

TRACE OFF
CP SET EMSG OFF
VMFCLEAR
DO 5
    SAY ' '
    END
SAY 'This program was written in order to reformat any CMS Exec file'
SAY 'which invokes the SAS System and contains any Version 5 System'
```

```
SAY 'Options which may or may not require an equal sign (=).'
SAY ' '
SAY 'Regardless of where the file resides, the program will recreate'
SAY 'the formatted version of the file on your 191 minidisk.'
SAY ' '
SAY 'Please enter the FILENAME, FILETYPE, and FILEMODE of the CMS Exec'
SAY 'file you want reformatted.'
SAY ' '
SAY 'For example, enter ====> MYOLD EXEC A'
PULL FN FT FM

IF FN = ' ' THEN DO
    SAY ' '
    SAY 'You have to enter the name of the file you want reformatted'
    EXIT
    END

IF FT = ' ' THEN DO
    FT = EXEC
    FM = A
    END

IF FM = ' ' THEN
    FM = A

STATE FN FT FM
IF RC ¬= 0 THEN DO
    SAY ' '
    SAY 'The file:' FN FT FM 'does not exist.'
    EXIT
    END

IF FM ¬= A THEN DO
    STATE FN FT A
    IF RC = 0 THEN
        ERASE FN FT A
    END

/*********************************************************************/
/* CHECK TO SEE IF FILE ACTUALLY INVOKES SAS OR NOT AND IF IT DOES */
/* WHETHER OR NOT THE SAS OPTIONS ALREADY CONTAIN THE EQUAL SIGNS  */
/*********************************************************************/

CALL CHECK

/*********************************************************************/
/* CHECK TO SEE IF THE FILE SAS518 OPTIONS A EXIST OR NOT AND IF   */
/* DOES NOT CREATE THE FILE WHICH CONTAINS THE OPTIONS THAT REQUIRE*/
/* THE EQUAL SIGNS                                                 */
/*********************************************************************/

STATE SAS518 OPTIONS A
IF RC ¬= 0 THEN DO
    CALL OPTIONS
    END

/*********************************************************************/
/* CREATE AN ARRAY THAT CONTAINS ALL THE OPTIONS THAT REQUIRE AN   */
/* EQUAL SIGN                                                      */
 *********************************************************************/

MAKEBUF
EXECIO '*' DISKR SAS518 OPTIONS A 1 '(' FINIS
N = QUEUED()
DO I = 1 TO N
    PULL OPTION.I
    END
DROPBUF

/*********************************************************************/
/* CHECK THE FILE FOR REXX OR EXEC2 FORMAT AND LOOK FOR THE        */
/* INVOCATION OF THE SAS SYSTEM, OTHERWISE OUTPUT TEXT TO NEW FILE */
/*********************************************************************/

MAKEBUF
EXECIO '*' DISKR FN FT FM 1 '(' FINIS
Z = QUEUED()
REXXFLAG = OFF
DO Y = 1 TO Z
    PULL LINE
    IF Y = 1 THEN DO
        IF INDEX(LINE,'/*') ¬= 0 THEN
            REXXFLAG = ON
        END
    IF INDEX(SPACE(LINE,1),'EXEC SAS') ¬= 0 THEN DO
        NEXTLINE = OFF
        NEWLINE  = ' '
        SASLINE  = ON
        TEXT     = ' '
        X = INDEX(LINE,'(')
        SASCALL = STRIP(TRANSLATE(SUBSTR(LINE,1,X),' ',"'"))
        TEXT = STRIP(TRANSLATE(SUBSTR(LINE,X+1),' ',"'"))
        CALL DOIT
        DO WHILE (NEXTLINE = ON)
            Y = Y + 1
            PULL LINE
            NEWLINE = ' '
            NEXTLINE = OFF
            TEXT = ' '
            X = INDEX(LINE,"'")
            IF X ¬= 0 THEN
                TEXT = STRIP(TRANSLATE(LINE),' ',"'"))
            ELSE
                TEXT = STRIP(LINE)
            CALL DOIT
            END
        END
    ELSE DO
        "EXECIO 1 DISKW" FOO FT A "(FINIS STRING" LINE
        END
    END

/*********************************************************************/
/* FINISH UP BY ERASING THE OLD FILE IF IT EXISTS AND RENAMING THE */
/* NEW FILE AND TURN ERROR MESSAGES BACK ON                        */
/*********************************************************************/

STATE FN FT A
IF RC = 0 THEN
    ERASE FN FT A
RENAME FOO FT A FN FT A
DROPBUF
CP SET EMSG ON
EXIT

/*********************************************************************/
/* ROUTINE THAT PARSES OUT THE SAS OPTIONS AND DECIDES WHETHER OR  */
/* NOT TO ADD THE EQUAL SIGN TO THE OPTION                         */
/*********************************************************************/

DOIT:
IF SUBSTR(REVERSE(TEXT),1,1) = ',' THEN DO
    NEXTLINE = ON
    TEXT = TRANSLATE(TEXT,' ',',')
    END
ELSE
    NEXTLINE = OFF

J = WORDS(TEXT)

/*********************************************************************/
/* THE OPTION (MENFILL/NONEMFILL) DOES REQUIRE AN EQUAL SIGN IN    */
/* VERSION 6 BUT THE OUTPUT FROM PROC OPTIONS IN VERSION 5 DOES    */
/* NOT INCLUDE THE EQUAL SIGN IN THE MENFILL OPTION                */
/*********************************************************************/

DO I = 1 TO J
    WORD = STRIP(WORD(TEXT,I))
    IF WORD ¬= ' ' THEN DO
        DO K = 1 TO N
            IF WORD = OPTION.K | WORD = NEMFILL THEN
                WORD = STRIP(WORD)||'='
            END
        END
    ELSE
        I = J

    IF FLAG = ON THEN DO
        NEWLINE = STRIP(NEWLINE)||STRIP(WORD)
        FLAG = OFF
        END
    ELSE
        NEWLINE = STRIP(NEWLINE)||' '||STRIP(WORD)

    IF INDEX(WORD,'=') ¬= 0 THEN
        FLAG = ON
    END
```

```
IF SASLINE = ON THEN DO
    IF REXXFLAG = ON THEN
        LINE = "'"||SASCALL||STRIP(NEWLINE)||"'"
    ELSE
        LINE = SASCALL||STRIP(NEWLINE)
    SASLINE  = ' '
    END
ELSE DO
    IF REXXFLAG = ON THEN
        LINE = "'"||STRIP(NEWLINE)||"'"
    ELSE
        LINE = STRIP(NEWLINE)
    END

IF NEXTLINE = ON THEN
    LINE = LINE||','

"EXECIO 1 DISKW" FOO FT A "(FINIS STRING" LINE

RETURN

/*********************************************************************/
/* ROUTINE TO CREATE THE FILE THAT CONTAINS ALL THE SAS OPTIONS      */
/* WHICH REQUIRE THE EQUALS SIGN USING THE OUTPUT FROM PROC OPTIONS*/
/*********************************************************************/

OPTIONS:
MAKEBUF

QUEUE 'PROC OPTIONS; RUN;'
QUEUE 'ENDSAS;'

'EXEC SAS (LD NONOTES'

EXECIO '*' DISKR SAS SASLOG A 1 '(' FINIS

DO QUEUED()
    PULL LINE
    LINE = STRIP(SUBSTR(LINE,2,20))
    IF INDEX(LINE,'=') ¬= 0 THEN DO
        X = INDEX(LINE,'=')
        OPTION = SUBSTR(LINE,1,X-1)
        "EXECIO 1 DISKW" SAS518 OPTIONS A "(FINIS STRING" OPTION
        END
    END

DROPBUF

STATE SAS SASLOG A
IF RC = 0 THEN
    ERASE SAS SASLOG A

RETURN

/*********************************************************************/
/* ROUTINE TO CHECK THAT THE FILE DOES IN FACT INVOKE THE SAS        */
/* SYSTEM AND WHETHER OR NOT THE FILE ALREADY CONTAINS EQUAL SIGNS */
/* IN THE SAS OPTIONS                                                */
/*********************************************************************/

CHECK:
MAKEBUF
"EXECIO * DISKR" FN FT FM "1 (FINIS"
N = QUEUED()
CHECK = OFF
SASCHECK = OFF
DO I = 1 TO N
    PULL LINE
    IF INDEX(SPACE(LINE,1),'EXEC SAS') ¬= 0 THEN DO
        SASCHECK = ON
        IF INDEX(LINE,'=') ¬= 0 THEN
            CHECK = ON
        I = N
        END
    END

IF SASCHECK = OFF THEN DO
    SAY ' '
    SAY 'The file:' FN FT FM 'does not invoke the SAS System.'
    CP SET EMSG ON
    EXIT
    END

IF CHECK = ON THEN DO
    SAY ' '
    SAY 'The file:' FN FT FM 'already contains equal signs in the SAS Optior
    CP SET EMSG ON
    EXIT
    END

DROPBUF
RETURN
```

REFERENCES

SAS Institute Inc. (1985), *SAS/AF® User's Guide, Version 5 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1985), *SAS/FSP® User's Guide, Version 5 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1985), *SAS® Guide to Macro Processing, Version 5 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1986), SAS Technical Report: P-146 *Changes and Enhancements to the Version 5 SAS System.* Cary, NC: SAS Institute Inc., 1986.

SAS Institute Inc. (1986), *SAS® Companion for the CMS Operating System, 1986 Edition*, Cary, NC: SAS Institute Inc.

SAS Institute Inc. (1989), SAS Technical Report: P-195 *Transporting SAS® Files between Host Systems.* Cary, NC: SAS Institute Inc.

ACKNOWLEDGMENT

We want to express our gratitude to James R. Walker of Legent Corporation, the author of SAS/DMI software, for the contributions he made to the enhancement to the SAS/DMI softare.