SAS® Macros for Converting Between ARC/INFO® Import/Export Files and SAS Data Sets

Barbara L. Jackson, Oak Ridge National Laboratory[1]

Abstract

ARC/INFO is a widely used geographic information system by Environmental Systems Research Institute (ESRI). ARC/INFO has powerful spatial analysis and presentation capabilities, but its data management and statistical analysis capabilities are limited. The SAS macros described here facilitate exchange of data between ARC/INFO and SAS.

The SAS macro, ARCTOSAS, creates a SAS data set from an ARC/INFO sequential export file. The SAS data set will have as many variables as there are INFO items, with identical or abbreviated names and appropriate data types.

The SAS macro, QUICKARC, creates a sequential file from a SAS data set. This file can be imported directly into ARC/INFO for use in the geographic information system. All the variables in the SAS data set are included as items in the INFO data set, with the appropriate data types and the same names.

The macros run under SAS version 6 for the PC/DOS and VMS operating systems, and version 5.18 or higher for MVS operating systems.

QUICKARC for SAS Version 6

The QUICKARC macro creates a flat, .E00-type file that can be imported by ARC/INFO from a member of a SAS version 6 data set. (There is also a SAS version 5.18 QUICKARC program that will run on the IBM mainframe.) Some features of QUICKARC are the following:

1. The file name of the .E00 file appears in the first record of the .E00 file. The name of the SAS data set appears in the third record of the .E00 file. This feature will help to keep track of the origin of data sets.

2. All the variables in the SAS member are included as INFO items in the output file. All SAS variables will have their same name in the INFO data set.

3. All numeric variables in the SAS data set will be floating (8-byte) INFO items, regardless of their SAS length. The INFO display format will be (10.4).

4. All character variables in the SAS data set will be character items in the INFO data set, with the same length.

5. SAS numeric variables with a date format will be floating (8-byte) in the INFO data set. Since the SAS date value is not useful in INFO, variables for year, month, and day should be created in SAS as needed.

6. Missing values in the SAS data set will be written as blank in the .E00 file. (OPTIONS MISSING=' '; appears in the macro.) ARC/INFO will read in the blank values as zeros. If this is not acceptable, the missing values in the SAS data set should be modified before invoking the QUICKARC macro.

To summarize, the SAS data set should have exactly the variables and observations of interest with the desired name, type, length, and missing value specification before invoking the QUICKARC macro. It is very tedious to alter the .E00 file after it is created. Instead, the SAS data set should be modified and the QUICKARC macro invoked again.

TSO Interactive

The QUICKARC macro can be "included" into the edit screen of your TSO SAS session in the following manner: Supply the appropriate allocate statements for the output .E00 file, the SAS data set if necessary, and the file for the QUICKARC macro. Code the call statement for QUICKARC with the appropriate arguments, and "submit".

1. Logon to TSO.

2. Invoke SAS.

3. Submit the following to allocate the QUICKARC macro:
   libname arcprog '.arc.prog';

4. Submit the following to allocate the E00 output file:
   filename e00 '.myfile.e00' new lrecl(80);

5. Submit the following to allocate your (existing) SAS data set:
   libname mysas '.mysas.sas' old;

6. Create the SAS data set member(s) with exactly the variables (length and type) and observations desired.

7. On the command line for the edit screen:
   include arcprog(quickarc)

(The entire QUICKARC macro listing will appear in the edit screen.)

Hit the SUBMIT PF key or enter the submit command on the command line. This will compile the QUICKARC macro so that it can be used anytime during the rest of your SAS session.

8. Invoke the macro with a call statement and submit:
   %QUICKARC(SASDATA=mysas.abc,FILE=e00,
   WORKFILE=work2,DUMMY=arra);

where
SASDATA is assigned the name of the input SAS data set member,
FILE is the ddname of the output E00 flat file,
WORKFILE is a (one-level) SAS work data set,
DUMMY is a four-letter (or less) SAS variable name that does not appear in the SASDATA data set.

The default argument values are shown below:

```
%QUICKARC(SASDATA=sasin,FILE=e00,
 WORKFILE=work2,DUMMY=arra);
```

If a parameter is omitted, the default value will be used. The SAS log screen will display the progress of the macro. The output screen will display work data sets.

9. Copy the E00 file to the host computer that has the ARC/INFO software. The following example is for copying from an IBM/MVS® mainframe to a VAX/VMS® using INTERLINK®.

   Log on to the VAX and type the command:

   ```
   $convert/truncate/fdl=anye00.fdl hostibm::"[uid]myfile.e00"
       vaxfile.e00
   ```

Note: The anye00.fdl file can be created from an .e00 file using the ANALYZE/RMS_FILE/FDL command on the VAX.

10. Import the vaxfile.e00 to ARC/INFO.

    ```
    $ARC
    arc: IMPORT INFO VAXFILE.E00 ARCFILE.EXT
    ```

    where
    VAXFILE.E00 is the SAS output file
    ARCFILE.EXT is the INFO data set.

ARCTOSAS for SAS Version 6

The ARCTOSAS macro creates a SAS (version 6) data set from an ARC/INFO export (.E00) file. (There is also an ARCTOSAS program for SAS version 5.18 that runs on the IBM mainframe.) Some features of ARCTOSAS include the following:

1. The SAS data set that is created will have as many variables as there are INFO items in the .E00 file. The names of the SAS variables will be identical or close to the INFO item names. If a set of unique SAS variable names cannot be created from the set of INFO item names, the SAS variable names will be VAR1-VARn, where n is the number of items.

2. The LABEL for the SAS data set will be the name of the .E00 file and the name of the originating ARC file.

3. The LABELs for the SAS variables will be the exact corresponding INFO item names.

4. The length and type of the SAS variables are shown below:

--INFO item--	--SAS variable--
binary (4,8)	numeric (8)
char (1-4096)	char ($1-$200)
date (8)	date (8)
floating (4,8)	numeric (8)
integer (4,8)	numeric (8)

5. The SAS data set will have the same number of observations as the .E00 file.

6. The ARCTOSAS macro will print a PROC CONTENTS and 10 observations of the SAS data set.

Create the ARC/INFO EXPORT File

The ARCTOSAS reads as input an ARC/INFO export file. This file will typically have a name extension of .E00. The export file must be created in ARC. The following example is for ARC/INFO running on a VAX with VMS:

1. Log on to the VAX.

2. Invoke ARC/INFO.
   $ARC

3. Execute the EXPORT command
   arc: EXPORT INFO ARCFILE.EXT OUTNAME NONE

where
   a. ARCFILE.EXT is the INFO data set,
   b. OUTNAME is the name of the export file (OUTNAME.E00),
   c. NONE specifies no compression.

4. Exit from ARC.
   $arc: quit

5. Copy the export file to the computer that has the SAS software. Below are some hints for copying the .E00 file to another computer.

   a. Copy to IBM mainframe with MVS operating system over INTERLINK.

The .E00 file has record length of 80, but is stored as variable length records of undefined length on the VAX. The IBM mainframe with MVS operating system requires that the length of the records be defined as 80 bytes with fixed length. Therefore, you can use the Convert command on the VAX to copy the file to the IBM and, at the same time, convert it to fixed length. Or you can use the Copy command and use switches to specify the length of the file.

```
$convert/truncate/fdl=fix80.fdl
 arcfile.e00 nodename::"[uid]ibmfile.e00"
```

or

```
$copy arcfile.e00
nodename::"[uid]ibmfile.e00/fixed:80/pad:20"
```

uid is your user id.
nodename is the name of the IBM host.
The fix80.fdl file can be created from an existing 80-byte, fixed length record file using the Analyze/rms_file/fdl command.

   b. Copy to an IBM-compatible personal computer. See the documentation for your communications software for downloading the .E00 file to your PC.

TSO Interactive

The ARCTOSAS macro can be "included" into the edit screen of your TSO SAS session. Supply the appropriate allocate statements for the .E00 export file, the SAS data set, and the file containing the ARCTOSAS macro. Code the call statement for ARCTOSAS with the appropriate arguments, and "submit".

1. Log on to TSO.

2. Invoke SAS.

3. Submit the following to allocate your SAS data set:
   libref mysas '\arc\gis';

4. On the command line of the edit screen:
   include '\arc\arctosas.sas'

(The entire ARCTOSAS macro listing will appear in the edit screen.)

Hit the SUBMIT PF key, or enter the submit command on the command line. This will compile the ARCTOSAS macro so that it can be used anytime during the rest of your SAS session.

5. Invoke the macro with a call statement and submit:

```
%ARCTOSAS(INFILE='info.e00',OUTSAS=mysas.member,
WORKFILE=work1,DUMMY=dummy);
```

where

a. INFILE specifies the file name of the ARC/INFO export file, 'C:\ARCDIR\FILE.E00'.
b. OUTSAS specifies the one-level (temporary) or two-level (permanent) SAS output data set.
c. WORKFILE specifies a SAS work data set (temporary).
d. DUMMY specifies a five-character or less SAS variable name that is not the name of an INFO variable in the .E00 data set.

The default values are shown below:

```
%ARCTOSAS(INFILE='info.e00',OUTSAS=outsas,
WORKFILE=work1,DUMMY=dummy);
```

If a parameter is omitted from the argument list, the default value will be used.

The SAS log screen will display the progress of the macro. The output screen will display work data sets.

The macro will print a PROC CONTENTS and 10 observations of the SAS data set to the output screen. You may continue the SAS session as desired.

1. SAS is a registered trademark of SAS Institute, Inc., Cary, NC, USA.
2. ARC/INFO is a registered trademark of Environmental Systems Research Institute (ESRI), Redlands, CA, USA.
3. INTERLINK is a registered trademark of Computer Sciences, Inc., Freemont, CA, USA.
4. IBM and MVS are registered trademarks of International Business Machines Corporation.
5. VAX and VMS are registered trademarks of Digital Equipment Corporation.

[1]Managed by Martin Marietta Energy Systems, Inc. under contract DE-AC05-84OR21400 with the U.S. Department of Energy.

Appendix 1 - QUICKARC MACRO

```
%MACRO QUICKARC(SASDATA=sasin,FILE='arcfile.e00',
WORKFILE=work2, DUMMY=arra);
  %* this macro creates an ARC export file (.e00) from a SAS data
  set.
  SASDATA is one- or two- level sas name of input data set.
  FILE is name of the output (flat) arc/info import file.
  WORKFILE is name of temporary SAS work data set (should be
  one-level).
  DUMMY is a SAS variable name of four characters or less that
  does not appear in the SASDATA data set.
  default values for arguments are shown in the argument list;

OPTIONS MISSING = ' ';
%*---CHKSET----------------------------------------------------;
%MACRO CHKSET(INDATA);
%* this macro is a PC version of the macro to check
  to see if a SAS data set exists.
  INDATA is one- or two-level SAS data set name (CAPS or lower
  case).
  the macro variable &DAMSG will have value BAD if no data set
  &DAMSG will have value OK if data set exists;
  %LOCAL MEMNAME LIBNAME;
  %GLOBAL DAMSG;
  %LET DAMSG=BAD;
  %IF %INDEX(&INDATA,.) GT 0 %THEN %DO;
  %LET LIBNAME
  =%QUOTE(%UPCASE(%QUOTE(%SCAN(&INDATA,1,.))));
  %LET MEMNAME
  =%QUOTE(%UPCASE(%QUOTE(%SCAN(&INDATA,2,.))));
  %END;
  %ELSE %DO;
  %LET LIBNAME=WORK;
  %LET MEMNAME
  =%QUOTE(%UPCASE(%QUOTE(&INDATA)));
  %END;
  data cout;
  run;
  proc contents data=&INDATA noprint out=cout
  (keep=memname);
  data _null_;
  set cout;
  if memname = "&MEMNAME" then call symput('DAMSG','OK');
  stop;
  run;
%MEND CHKSET;
%*------------------------------------------------------------;
%CHKSET(&SASDATA);
%IF &DAMSG=BAD %THEN %DO;
  %PUT '********';
  %PUT "SAS data set &SASDATA does not exist -- QUICKARC
  terminating";
  %PUT '********';
  %END;
  %ELSE %DO;
  %*
  create output data set from proc contents of SASDATA;
  proc contents data=&SASDATA out=&WORKFILE;
%*
  create some macro variables from information in contents data
  set.
  NUMVARS has the number of variables in the data set
  ARCREC has the length in bytes of the logical data record
    for FILE
  NUMRECS  number of 80-byte physical records in a logical
  record
  LOGLEN   length of logical data record
  NUMOBS   number of data observations;
  data &WORKFILE;
  set &WORKFILE end = eof;
  retain arcrec loglen boundary;
  if type = 1 then length = 8;  /*  length is always 8 for numeric*/
  laglen = lag(length);
  if _n_ = 1 then do;
    arcrec = length; /* initialize arcrec */
    boundary = 14;  /* first boundary is always 14  */
    if type = 1 then loglen = 24; /*logical length 24 for numeric*/
    else loglen = length;        /*logical len =length for char*/
    end;
    else do;
    arcrec = arcrec + length;   /* increment arcrec */
    boundary = boundary + laglen*10;  /* len of prev var times
    10*/
    if type = 1 then loglen = loglen + 24;  /* 24 for numeric var*/
      else loglen = loglen + length;
    end;
    if eof then do;
      call symput('NUMVARS',(put(_n_,3.)));
      call symput('ARCREC',(put(arcrec,4.)));
      call symput('LOGLEN',(put(loglen,4.)));
      call symput('NUMOBS',(put(nobs,8.)));
```

```
      call symput('NUMRECS',left(put(ceil(loglen/80),5.)));
      end;
    keep length name type boundary;
  run;
proc print data=&WORKFILE;
%PUT number of variables is **&NUMVARS**;
%PUT length of arc records is **&ARCREC**;
%PUT length of physical data records is **&LOGLEN**;
%PUT number of phys records in a logical record is
**&NUMRECS**;
%PUT number of data records is **&NUMOBS**;
  %*
  write header information and variable information to FILE
  create macro variables
    MVi variable name
    FMi output format of variable
    STi starting position of variable in output record
    Wli width of output variable
    ;
  data _null_;
    set &WORKFILE;
    file &FILE;
    retain startpos;
    if _n_ = 1 then do;
      put 'EXP 0' @8 "&FILE";
      put 'IFO 2';
      put @1 "&SASDATA" @36 "&NUMVARS" @40 "&NUMVARS"
        @43 "&ARCREC" @49 "&NUMOBS";
      startpos = 1;
      end;
    call symput('MV'||left(put(_n_,5.)),name);
    call symput('ST'||left(put(_n_,8.)),left(put(startpos,8.)));
    if type=1 then do;                  /* numeric variable */
      call symput('FM'||left(put(_n_,5.)),'E24.0');
      put name $8. @17 ' 8-1' boundary 5. '-1 10 4 60-1 -1
      -1-1' @67 _n_ 3.;
      call symput('WI'||left(put(_n_,5.)),'24');
      startpos = startpos + 24;
      end;
      else if type=2 then do;           /* character variable */
        call symput('FM'||left(put(_n_,5.)),
          '$char'||compress(put(length,5.))||'.');
        call symput('WI'||left(put(_n_,5.)),left(put(length,8.)));
        startpos = startpos + length;
        put name $8. @17 length 3. '-1' boundary 5. '-1 ' length
        3. '-1 20-1 -1 -1-1' @67 _n_ 3.;
        end;
    run;

%DO i = 1 %TO &NUMVARS;
  %PUT mv&i=*&&MV&i* fm&i=*&&FM&i* st&i=*&&ST&i*
  wi&i=*&&WI&i*;
  %END;
%*----PUTSTMTS----------------------------------------;
%MACRO PUTSTMTS;
  %* this macro generates put statements for writing data records;
  %LOCAL i;
  %DO i = 1 %TO &NUMRECS;
    put (arra%EVAL((&i-1)*80+1) - arra%EVAL(&i*80) ) ($char1.);
    %END;
%MEND PUTSTMTS;
%*---------------------------------------------------;
%*
  write data records to FILE
  there are NUMRECS physical records for each logical record;
  data _null_;
    set &SASDATA end=eof;
    file &FILE mod;
    attrib &DUMMY.x length=$200;
    array &DUMMY{%EVAL(&NUMRECS*80)} $1
      &DUMMY.1-&DUMMY%EVAL(&NUMRECS*80) ;
    %* load variables into array 1 character at a time;
    %DO i = 1 %TO &NUMVARS;
      &DUMMY.x = put(&&MV&i,&&FM&i);
      do j = 1 to &&WI&i;
        &DUMMY{&&ST&i + j -1} = substr(&DUMMY.x,j,1);
        end;
      %END;
    %PUTSTMTS;
    if eof then do;
      put 'EOI';
      put 'EOS';
      end;
run;
run;
%END;
%MEND QUICKARC;

 /* calling example for SAS PC /*
*options mprint ;
*libname arc 'c:\sas\arci\';      /* define libname */
*%quickarc(SASDATA=arc.data1,FILE='arcfile2.e00');    /* call
quickarc */
*run;
```

Appendix 2 - ARCTOSAS MACRO

```
%MACRO ARCTOSAS(INFILE='info.e00',
OUTSAS=outsas,WORKFILE=work1, DUMMY=dummy);
%*
this macro creates a SAS data set from an ARC/INFO export file.

1. create an ARC/INFO export (.e00) file,
   $ARC
   arc: EXPORT INFO ARCFILE.EXT OUTNAME NONE

   where
   ARCFILE.EXT is ARC file,
   OUTNAME is export file name (OUTNAME.E00),
   NONE specifies no compression.

2. download the export (.e00) file to the PC.

3. a. invoke SAS on the PC.
   b. allocate SAS library, filenames.
   c. include the file that contains the ARCTOSAS macro.
   d. supply the ARCTOSAS macro call with the following
      arguments:
        INFILE is the ARC/INFO export file name
        OUTSAS is the one-level (temporary) or two-level
        (permanent) SAS output data set
        WORKFILE is a SAS work data set (temporary)
        DUMMY is a five-character or less SAS variable name that
        is not the name of an INFO variable in the .e00 data set
        (the default values are shown in the argument list of the
        macro statement).
   e. submit.

   the resulting SAS data set will have as many variables as there
   are INFO items. the macro will attempt to give SAS names that
   are identical or close to the INFO item names. if it fails, the
   SAS names will be VAR1-VARn.
   the LABEL for the SAS data set will be the value of the INFILE
   parameter (the .e00 data set name).
   the LABELS for the SAS variables will be the exact
   corresponding INFO item names.
```

```
   the length and type of the SAS variables are shown below:

      --info item--          --SAS variable--
      binary (4,8)           real (8)
      char (1-4096)          char ($1-$200)
      date (8)               date (8)
      floating (4,8)         real (8)
      integer (4,8)          real (8)

   good luck !

Program developed by Barbara Jackson, C&TD, (615) 574-8680,
2/90, Martin Marietta Energy Systems, Inc., P.O. Box 2008,
MS 6038, Oak Ridge, TN 37831-6038.
;

  data _null_;
    %GLOBAL NUMVARS INFOLINS DATALIN NUMDATA;
    infile &INFILE firstobs=3 obs=3;
%PUT you have chosen infile = &INFILE;
%PUT you have chosen outfile = &OUTSAS;
%PUT work file is &WORKFILE;
%PUT dummy variable name is &DUMMY;
    input numvars 39-42 numdata 52-56;
    /* numvars includes redefined items*/
    infolins = numvars + 3;        /* last information record */
    datalin = infolins + 1;        /* first data record */
    call symput('NUMVARS' ,left(put(numvars,5.)));
    call symput('INFOLINS',left(put(infolins,5.)));
    call symput('DATALIN' ,left(put(datalin,5.)));
    call symput('NUMDATA' ,left(put(numdata,5.)));
  run;
%PUT number of variables = &NUMVARS;
  data &WORKFILE;
    infile &INFILE firstobs=4 obs=&infolins;
    attrib sasname length=$8;
    attrib fmt length=$10;
    retain startcol endcol;
    input varname $16. arclen $3. @30 len $3. @34 dec_n $1. @36
    type 1.;
    nochars = length(varname);
    count = _n_;
    if _n_ = 1 then endcol = 0;
    sasname = substr(varname,1,8);
    %* replace invalid characters in sasname with '_' ;
    if indexc(sasname,'#-/*()\|$@!%&.?<>') gt 0 then
      sasname = translate
      (sasname,'_________________','#-/+()\|$@!%&.?<>');
    if indexc(sasname,"'") gt 0 then
      sasname = translate(sasname,'_',"'");
    %* assign a format value according to type
      assign the width of this field;
    select (type);
      when (1) do;                          /* date  */
        fmt = 'yymmdd8.';
        width = 8;
        end;
      when (2) do;                          /* char  */
        %*character variable is max length of 200;
        fmt = '$char'||compress(min(200,len))||'.';
        width = len;
        end;
      when (3) do;                          /*integer*/
        fmt = compress(arclen)||'.';
        width = arclen;
        end;
      when (5) do;                          /*binary */
        fmt = '11.'||left(dec_n);
        width = 11;
        end;
      when (6) if arclen='4' then do;       /*floating*/
        fmt = '14.'||left(dec_n);
        width = 14;
        end;
      else if arclen='8' then do;
        fmt = '24.'||left(dec_n);
        width = 24;
        end;
      end;
      %* assign the start and end column number for this field;
      startcol = endcol + 1;
      endcol = startcol + width -1;
%*
  sort the sasnames according to length of name and alphabetical
  order.  if any names are duplicates, then append an integer to
  the end of the name. a name may have up to 8 characters.
  ;
  proc sort data=&WORKFILE;
    by nochars sasname;
  data &WORKFILE;
    set &WORKFILE;
    retain i;
    if _n_ = 1 then i = 0;
    if sasname = lag(sasname) then do;
      i = i + 1;
      if nochars <= 7 then
        sasname = compress(sasname)||left(put(i,3.));
        else
        sasname = substr(sasname,1,7)||left(put(i,3.));
      end;
      else i = 0;
%*
  check to see if there are still duplicate sasnames
  if so, set a macro variable to flag this condition;
  proc sort data=&WORKFILE;
    by sasname;
  %LET FLAG=good;
  data _null_;
    set &WORKFILE;
    if sasname = lag(sasname) then
      call symput ('FLAG','bad');
run;
%*
  sort the workfile data set back to its orginal order;
  proc sort data=&WORKFILE;
    by count;
  proc print data=&WORKFILE;
%*
create macro variables
  MVi from sas variable names
  FMi from formats
  TYi from data type
  STi start column for this field logical record
  SCi start column for this field physical record
  Wli for width of this field
  SRi first 80-col record for this variable
  NRi SRi + 1
  LRi last 80-col record for this variable
  LAi label for this variable
  NUMBYTES total width of the logical record
  NUMRECS number of physical records in a logical record
  ENDDATA number of last physical data record
  ;
  data _null_;
    set &WORKFILE end = eof;
    call symput ('MV'||left(put(_n_,3.)),sasname);
```

```
    call symput ('LA'||left(put(_n_,3.)),varname);
    call symput ('FM'||left(put(_n_,3.)),fmt);
    call symput ('TY'||left(put(_n_,3.)),type);
    call symput ('ST'||left(put(_n_,5.)),left(put(startcol,8.)));
    sc1 = mod(startcol,80);
      if sc1 = 0 then sc1 = 80;
    call symput ('SC'||left(put(_n_,3.)),left(put(sc1,8.)));
    call symput ('WI'||left(put(_n_,3.)),left(put(width,8.)));
    call symput ('SR'||left(put(_n_,3.)),left(put(ceil(startcol/80),8.)));
    st1 = ceil(startcol/80) + 1;
    call symput ('NR'||left(put(_n_,3.)),left(put(st1,8.)));
    call symput ('LR'||left(put(_n_,3.)),
      left(put(ceil((startcol+width-1)/80),8.)));
    if eof then do;
      call symput ('NUMBYTES',endcol);
      call symput ('NUMRECS',compress(ceil(endcol/80)));
      call symput
        ('ENDDATA',ceil(endcol/80)*&NUMDATA+&INFOLINS);
      end;
run;
%DO i = 1 %TO &NUMVARS;
%PUT MVi *&&MV&i*  STi *&&ST&i*  SCi *&&SC&i*  Wli
*&&Wl&i*  SRi *&&SR&i*  NRi *&&NR&i*  LRi *&&LR&i*;
%END;
%*********************************************************;
%MACRO INPUTEM;
%* this macro generates an input statement;
  %IF &NUMRECS = 1 %THEN
    %STR(input &DUMMY.1 $char80.;);
    %ELSE    %STR(input    (&DUMMY.1-&DUMMY&NUMRECS)
      ($char80.););
%MEND INPUTEM;
%*********************************************************;
%MACRO GETREC(I);
%* this macro concatenates physical records together
  to make the big logical record;
  %LOCAL j;
  &DUMMY&&SR&I
  %IF &&LR&I > &&SR&I %THEN
    %DO j = &&NR&I %TO &&LR&I;
      ||&DUMMY&j
      %END;
%MEND GETREC;
%*********************************************************;
%MACRO VARS2;
%* this macro extracts variables from big logical record string;
  options symbolgen;
  %DO i = 1 %TO &NUMVARS;
    %IF &&TY&i = 1 %THEN
      %STR( var&i = input(translate(substr(%GETREC(&i),
        &&SC&i,&&Wl&i),'0',' '),&&FM&i); );
      %ELSE
        %STR(var&i = input(substr(%GETREC(&i),
          &&SC&i,&&WI&i),&&FM&i); );
    %END;
 options nosymbolgen;
 %MEND VARS2;
 %*********************************************************;
 %MACRO VARS;
 %* this macro extracts variables from big logical record string;
  %LOCAL i;
  %DO i = 1 %TO &NUMVARS;
    %IF &&TY&i = 1 %THEN
      %STR(var&i = input(translate(substr
        (&BIGREC,&&ST&i,&&WI&i),'0',' '),&&FM&i););
      %ELSE
        %STR(var&i =
          input(substr(&BIGREC,&&ST&i,&&WI&i),&&FM&i););
    %END;
%MEND VARS;
%*********************************************************;
%MACRO CONCAT;
%* this macro concatenates physical records together
  to make the big logical record;
  %LOCAL i;
  &DUMMY.1
  %IF &NUMRECS > 1 %THEN %DO i = 2 %TO &NUMRECS;
    ||&&DUMMY&i
    %END;
%MEND CONCAT;
%*********************************************************;
%MACRO LABELEM;
%* this macro generates a label statement;
  %LOCAL i;
  label
  %DO i = 1 %TO &NUMVARS;
    var&i = "&&LA&i"
    %END;
    ;
%MEND LABELEM;
%*********************************************************;
%MACRO RENAMEM;
%* this macro generates rename statement;
  %LOCAL i;
    rename
    %DO i = 1 %TO &NUMVARS;
      var&i = &&MV&i
      %END;
    ;
%MEND RENAMEM;
%*********************************************************;
%* read the arc/info input file and create SAS output file;
  data &OUTSAS (label = "&INFILE");
    infile &INFILE firstobs=&DATALIN flowover
      lrecl=80 linesize=80 obs = &ENDDATA;
    %LABELEM;          /* generate label statment */
    %INPUTEM;          /* input physical records */
    %VARS2;            /* input variables from string */
    drop &DUMMY.1-&DUMMY&NUMRECS;
    run;
  proc print data=&OUTSAS;
  proc contents data=&OUTSAS;
run;
%* if sasnames are valid rename all the variables;
%IF &FLAG=good %THEN %DO;
  data &OUTSAS (label = "&INFILE");
    set &OUTSAS;
    %RENAMEM;          /* rename variables to sasname */
  proc contents data=&OUTSAS;
  proc print data=&OUTSAS (obs=10);
  run;
%END;
run;
%MEND ARCTOSAS;

/* example call to ARCTOSAS */
*libname arc 'c:\sas\arci\';
*options mprint;
*%arctosas(INFILE='arcfile1.e00',OUTSAS=arctemp);
```