

Productivity Tool created for Tax System Conversion Using
BASE SAS software's MACRO facility

Clarence W. Jackson
City of Dallas Revenue and Taxation Department
Dallas, Texas

This paper describes the conditions and SAS coding used in the evaluation, conversion, and auditing of a recent project we undertook at the City of Dallas in changing to a different TAX collections system.

For each computer program written, a section of code is needed to define the input file, and describe the fields of data on the file. In SAS, this is done in a DATA step using "INFILE" and "INPUT" statements. If this is done using COBOL record descriptions, then there needs to be a translation to the logical field positions if you would like to use SAS in its most efficient INPUT method of column pointing. This effort should not be wasted without documenting the field layout of the file for later programming. It is possible to use the document as an input tool for SAS jobs using BASE SAS software's MACRO facility, as I have done.

Among the things that needed to be done to convert our data to the format was to document the files of both systems. Our files were well documented with information useful in writing SAS programs such as field positions, format and length. Their file documentation consisted of COBOL copybooks with no layout of field positions. As any programmer of SAS knows that when you are working with COBOL record layouts, you must calculate the field positions of fields that you want to use. I was faced with having to document their files for our purposes, then write programs to verify and print their sample data. There were some 57 VSAM files, about 999 variables, and about 2 weeks to complete the task. We have experienced a lot of "downsizing" from 5 analysts to 1 (me), so I was faced with this task alone, and I had to do it in addition to my regular duties. This was during the Christmas season, and the end of the year is busy in the Tax Department. With no choice in doing this, I decided to let SAS help me write this code.

My first approach was to let SAS read the COBOL record layouts and create the "Input" statements. But the code to create the code was beginning to get long, so I put it on the shelf to be completed later.

Dejected, I began defining the physical record layouts from the COBOL logical layouts. As I did the first file,

```
01 ACCTREC.  
  05 ACCT-NUM      pic(20).  
  = Pos=1-20, fmt=$20.
```

it hit me that if I could put the definitions into a SAS dataset or some other file and read the data in the MACRO table, I could get it to help me write my code. Better still, maybe I could get SAS to read the raw data also! This is what I did.

I needed to define my files, so as I read in a COBOL description for the sample record file SAMPLE1,

Productivity Tool created for Tax System Conversion Using
BASE SAS software's MACRO facility

SAMPLE1

```
*01 ACCTREC.
    05 ACCT-NUM.
        10 ACCT-NO          PIC X(20).
        10 TAX-YEAR         PIC 9(3)    COMP-3.
```

I put on the document,

SAMPLE1

```
*01 ACCTREC.
    05 ACCT-NUM.
1-20      10 ACCT-NO          PIC X(20).          $20.
21-22     10 TAX-YEAR         PIC 9(3)    COMP-3.    PD2.
```

The completed document (see appendix A) will serve two purposes:

- (1) to supply documentation for the Information Services Programmers that would be maintaining the system when it was brought in-house, and
- (2) to be used by mine and other user's SAS jobs to use in infile/input statements.

The completed document for all 57 files document was printed, and also copied to a PDS member (cjvsamrr).

The next step was simply to write code that would read the PDS with the record layouts (cjvsamrr) for the data with the definitions to ensure that only description-lines were read, and use the POS, Format, and name variables as input statements, and the file name to get the file. Printing the data was easy after that. (see appendix B and C)

The job successfully read each of the 57 files, created infile/input statements, and printed each file's data with variable labels from the COBOL field names!

Completed in 9 days, this MACRO has allowed me to be more productive in other phases of the conversion project while being innovative. It has saved me a lot of time in writing code, as I can use a very small amount of code to do a lot of work, and has allowed everyone needing to see a dump of the records being converted to the new system by just putting in the file name (see appendix D). I have ideas of expanding this further, and any ideas from the reader will be appreciated.

Other benefits resulting from this exercise include:

- (1) Documentation of programs input data, record layout documents has been enhanced while reducing the effort needed.
- (2) Changes to file layouts and the programs using those files don't create a panic with me anymore. I just make the corrections/changes to the record layout document, copy the document to the PDS, run the SAS record-layout job written to do phase 1 testing distribute the printout to the people using my system, and go on to other things. (see appendix E)

Productivity Tool created for Tax System Conversion Using
BASE SAS software's MACRO facility

RECAP:

- (1) If you read raw data, establish a record layout library on a file.
I used a PDS, but it could be tape, disk, QSAM or VSAM.
- (2) Create a MACRO to read your record layout file. This was actually
easier than I thought it would be, and I'm sure you can do it to.
I put the BUILD Macro in the AUTOCALL library so that it could be
used easier.

Author Info:

Clarence Jackson,
Sr. Systems Analyst,
Revenue and Taxation Dept.
City of Dallas, Texas
214-670-5798

References:

SAS Views: SAS Macro Language, Version 5 edition
SAS Language, Reference Version 6, 1st edition
SAS Companion for MVS Environment
SAS Guide to VSAM Processing
SAS Programming Tips: A Guide to Efficient SAS Processing

Appendix A - Input file document

Input file 'sys7.control(cjvsamrr)' used to store the file and field layouts for use in the BUILD macro.

```
# TEST1
  *01 TEST-RECORD.
    05 SK-KEY-AREA.
      1      10 SK-CLASS          PIC X(01).          $1.
      2-6    10 SK-ACCOUNT       PIC 9(9)  COMP-3.     PD5.
      7-8    10 SK-YEAR          PIC 9(3)  COMP-3.     PD2.
      9-10   10 SK-SEQ           PIC 9(3)  COMP-3.     PD2.
      11-12  10 SK-TRL-SEQ       PIC 9(3)  COMP-3.     PD2.
      13-14  10 SK-TRANSCODE     PIC 9(3)  COMP-3.     $2.
      15-18  10 SK-TA-ID         PIC X(4).          $4.
    05 SK-DATA-AREA.
      19-144 10 SK-PAYMENTS-AREA PIC X(126).         $126.
      145-210 10 SK-RECORDS-AREA PIC X(66).          $66.
              10 SK-ITEM-PAYS REDEFINES SK-ITEMPAY-AREA.
    15 TEST-PAY-KEY.
      145-150 17 RECORDS-ID      PIC 9(11)  COMP-3.   PD6.
              17 ALT-KEY.
      151-156 20 PAYMENT-NO      PIC 9(11)  COMP-3.   PD6.
      157-162 20 SEQ-NUM         PIC 9(11)  COMP-3.   PD6.
      163-166 15 PAID-DATE       PIC 9(6)    COMP-3.   PD4.
      167-170 15 DUE-DATE        PIC 9(6)    COMP-3.   PD4.
      171-175 15 ATT-FEE         PIC S9(7)V99 COMP-3.   PD5.2
      176-180 15 PEN-INT-AMT     PIC S9(7)V99 COMP-3.   PD5.2
      181-186 15 TAX-AMT         PIC S9(9)V99 COMP-3.   PD6.2
      187-192 15 PAID-AMT        PIC S9(9)V99 COMP-3.   PD6.2
      193-196 15 TRANS-TYPE      PIC X(4).          $4.
      15 TRANS-REDEF REDEFINES TRANS-TYPE.
      193-194 20 TRANS-CODE1     PIC XX.           $2.
              20 TRANS-CODE-RED REDEFINES TRANS-CODE.
      193      15 X-TRANS        PIC X.            $1.
      194      15 FILLER         PIC X.            $1.
      195-196 20 TRANS-ACTIONS   PIC XX.           $2.
              20 TRANS-ACT-RED REDEFINES TRANS-ACTION.
      195      25 TRANS-A        PIC X.            $1.
      196      25 TRANS-B        PIC X.            $1.
      197-202 15 WHERE-ID        PIC X(6).         $6.
      203-206 15 OPER-ID         PIC X(4).         $4.
      207-210 15 SYSTEM-DATE     PIC 9(6)    COMP-3.   PD4.
```

Appendix A - Input file document

```
# SAMPLE1
  *01 ACCT-DTL.
    05 DTL-ACCT-NO.
1-20      10 ACCT-NO      PIC X(20).      $20.
21-22      10 YEAR      PIC 9(3)      COMP-3.      PD2.
23-27      05 ACRES      PIC 9(5)V9(4) COMP-3.      PD5.4
28-32      05 LAND      PIC S9(9)      COMP-3.      PD5.
33-37      05 IMPR      PIC S9(9)      COMP-3.      PD5.
38         05 FUNNY-ACCT PIC X.      $1.
39         05 BAD-FLAG  PIC X.      $1.
40         05 EXEMPT    PIC X.      $1.
41         05 HS        PIC X.      $1.
42         05 SR        PIC X.      $1.
43         05 DISAB     PIC X.      $1.
44         05 SOLAR     PIC X.      $1.
45         05 EX-A      PIC X.      $1.
46         05 EX-B      PIC X.      $1.
47-49      05 DAV      PIC S9(5)      COMP-3.      PD3.
50-53      05 DTL-CHANGE-DATE PIC 9(6)      COMP-3.      PD4.
54-57      05 DTL-CHANGE-OPID PIC X(4).      $4.
```

Appendix B - Macros from library

This macro controls the labels used on SAS variables in the DATA step. It is called by the %BUILD macro.

```
%MACRO LABX ;  
%LOCAL NWJ;  
%DO NWJ=1 %TO &&MATCH&CWJ;  
    &&VNAME&NWJ="&&LABEL&NWJ"  
%END;;  
%MEND LABX;
```

This macro controls the column style of defining SAS variables in the INPUT statement. It is called by the %BUILD macro.

```
%MACRO INPX ;  
%LOCAL CAM;  
%DO CAM=1 %TO &&MATCH&CWJ;  
    @ &&POS&CAM &&VNAME&CAM &&FMT&CAM  
%END;;  
%MEND INPX;
```

Appendix C - BUILD Macro from library

This is the main macro BUILD. It calls in the file with the field layouts, and uses them to build a SAS data set from raw data files that are dynamically allocated using the FILENAME statement.

```
%MACRO BUILD      (INPUTDSN,INPUTVAR);
```

```
OPTIONS DQUOTE    NOMPRINT NOSOURCE NONOTES;
```

```
%LOCAL CWJ;
```

```
DATA _NULL_;  
  SET &INPUTDSN END=DONE;  
  CALL SYMPUT ('INVAR' || TRIM(LEFT(PUT(_N_,3))),  
              TRIM(LEFT(&INVAR)));  
  IF DONE THEN CALL SYMPUT('NUMOBS',_N_);  
RUN;
```

```
/* The following FILENAME statement assigns the file with the data used  
   for defining the fields. This file is referred to as the 'document  
   file'.  
;
```

```
FILENAME CJVS "SYS7.CONTROL(CJVSAMRR)" DISP=SHR;
```

```
/* The next two data steps reads the document file for the start and  
   stop positions, field names and format of the field.  
;
```

```
DATA CJA1 (KEEP=POSI VNAME FMT FILE1 FILE2 SEQ STARTPOS LABEL);  
  LENGTH NAME LABEL $ 30;  
  INFILE CJVS MISSEVER;  
  INPUT @1 X $1. @;  
  IF X='#' THEN DO; CAM=1;  
    INPUT @5 FILE1 $8. @15 FILE2 $8. ;  
    RETAIN FILE1 FILE2; DELETE; RETURN; END;  
  IF NOT(X IN ('1','2','3','4','5','6','7','8','9','0')) THEN DO;  
    INPUT; DELETE; RETURN; END;  
  ELSE DO;  
    INPUT @1 POSI $ LEV $ NAME $ @68 FMT $ ;  
  END;  
  CAM+1; IF CAM GT 99 THEN CAM=1;  
  LABEL=COMPRESS(NAME);  
  NAMES=SUBSTR((TRIM(LEFT(COMPRESS(NAME,'- .')))),1);  
  NAMEX=SUBSTR(NAMES,1,6) || TRIM(LEFT(PUT(CAM,$2.)));  
  VNAME=COMPRESS(NAMEX,'- .');  
  STARTPOS=SCAN(POSI,1);  
  STARTPOS=SCAN(POSI,1);  
  SEQ=_N_;  
RUN;
```

```
PROC SORT;      BY FILE1 SEQ; RUN;
```

Appendix C - BUILD Macro from library

```

DATA CJA2 (KEEP=POSI VNAME FMT FILE1 FILE2 SEQ STARTPOS LABEL);
LENGTH NAME LABEL $ 30;
INFILE CJVS MISSEVER;
INPUT @1 X $1. @;
  IF X='#' THEN DO; CAM=1;
    INPUT @5 FILE2 $8. @15 FILE1 $8. ;
    RETAIN FILE1 FILE2; DELETE; RETURN; END;
  IF NOT(X IN ('1','2','3','4','5','6','7','8','9','0')) THEN DO;
    INPUT; DELETE; RETURN; END;
  ELSE DO;
    INPUT @1 POSI $ LEV $ NAME $ @68 FMT $ ;
  END;
CAM+1; IF CAM GT 99 THEN CAM=1;
LABEL=COMPRESS(NAME);
NAMES=SUBSTR(((TRIM(LEFT(COMPRESS(NAME,'- .'))))),1);
NAMEX=SUBSTR(NAMES,1,6)||TRIM(LEFT(PUT(CAM,$2.)));
VNAME=COMPRESS(NAMEX,'- .');
STARTPOS=SCAN(POSI,1);
SEQ=_N_;
IF FILE1=' ' THEN DELETE;
RUN;

PROC SORT; BY FILE1 SEQ; RUN;

DATA CJAC;
  SET CJA1 (IN=A) CJA2 (IN=B); BY FILE1 ;
RUN;

PROC SORT; BY FILE1 STARTPOS; RUN;

%DO CWJ=1 %TO &NUMOBS;

DATA CJ&CWJ ; SET CJAC; BY FILE1;
  IF FILE1="&&INVAR&CWJ";
  IF VNAME=' ' THEN VNAME='FIL'||TRIM(LEFT(PUT(_N_,2.)));
RUN;

DATA _NULL_ ; SET CJ&CWJ END=DONE;
  CALL SYMPUT ('POS'||TRIM(LEFT(PUT(_N_,3.))),TRIM(LEFT(STARTPOS)));
  CALL SYMPUT ('VNAME'||TRIM(LEFT(PUT(_N_,3.))),TRIM(LEFT(VNAME)));
  CALL SYMPUT ('FMT'||TRIM(LEFT(PUT(_N_,3.))),TRIM(LEFT(FMT)));
  CALL SYMPUT ('LABEL'||TRIM(LEFT(PUT(_N_,3.))),TRIM(LEFT(LABEL)));
IF DONE THEN CALL SYMPUT('MATCH'||TRIM(LEFT(PUT(&CWJ,3.))),_N_);
*F THEN CALL SYMPUT('MATCHOB',_N_);
DELETE; RETURN;
RUN;

  %IF &&INVAR&CWJ=RPTSWK %THEN %DO;
FILENAME &&INVAR&CWJ "TAX7.QSAM.&&INVAR&CWJ" DISP=SHR; %END;
%ELSE %DO;
FILENAME &&INVAR&CWJ "TAX7.VSAM.&&INVAR&CWJ" DISP=SHR;
%END;
  %IF &&INVAR&CWJ=' ' %THEN %DO; &CWJ=&NUMOBS; %GOTO EXITO; %END;

PROC DATASETS; DELETE CJ&CWJ; RUN;

```

Appendix C - BUILD Macro from library

```

DATA &&INVAR&CWJ (DROP=RC);
  %IF &&INVAR&CWJ=RPTSWK %THEN %DO;
    INFILE &&INVAR&CWJ ; %END;
  %ELSE %DO;
    INFILE &&INVAR&CWJ VSAM FEEDBACK=FV;
  %END;
  INPUT %INPX;;;
  LABEL %LABX;;;
  RC='NO ' ;
  %IF &&INVAR&CWJ='RPTSWK' %THEN %DO;
IF (_IORC_ NE 0) OR (FV NE 0) OR (_FDBK_ NE 0) THEN DO;
  RC='YES'; FV=0; _ERROR_=0; _FDBK_=0; END;
%END;
RUN;

FILENAME &&INVAR&CWJ CLEAR;

DATA _NULL_;
SET &&INVAR&CWJ; IF _N_ EQ 1;
FILE LOG;
  PUT ///// @10 '+-----+'
           @10 '|                                     |'
           @10 "      SAS DATA SET &&INPUTVAR&CWJ IS CREATED  |"
           @10 '|                                     |'
           @10 '+-----+' ;

RUN;

%EXITO:
%END;

%MEND BUILD;

```

Appendix D - Batch program using the BUILD Macro

This is a sample program using Macro BUILD from an AUTOCALL library.

```
//CJbldtst JOB (TAX1,XXXX),CJAC.2BN,CLASS=s
//S1 EXEC SAS,HOLD=YES,OPTIONS='DQUOTE MAUTOSOURCE mprint'
//SASLOG DD COPIES=1,HOLD=&HOLD
//SASLIST DD COPIES=1,HOLD=&HOLD
//TAXMAC DD DSN=TAX7.SAS.MACROS,DISP=SHR
//SYSIN DD *
```

```
OPTIONS SASAUTOS=TAXMAC;
```

```
DATA BUILDFILE (KEEP=NAME);
```

```
  INFILE CARDS;
```

```
  INPUT @1 NAME $8.;
```

```
  LIST;
```

```
CARDS;
```

```
sample1
```

```
;
```

```
%BUILD    (BUILDFILE,NAME); PAGE;
```

```
//
```