Sample Utility Macros
Ross Z. Merlin, Pinkerton Computer Consultants Inc.

This presentation is a collection of general-purpose MACROS:

ANY, NOTANY -- shorthand for multi-value "IF" statements
INDEXBLD -- build an index dataset and index functions (FORMATs) to facilitate direct access of a SAS dataset
MAKEFMT -- make a SAS FORMAT from data in a SAS dataset
OBS -- ascertain the number of observations in a SAS dataset without counting
SEARCH -- search a field for user specified values, display all obs that match
TYPE -- determine the type (CHARACTER or NUMERIC) of a SAS variable
USAGE -- access SAS Usage Notes
VARNAME -- display variable names and labels
VALIDATE -- validate a response and generate a label for %GOTO
WP -- reformat text for inclusion in SAS listings

ANY and NOTANY are macros that allow for a shorter notation in multi-value "IF" statements.
In COBOL, one can code: IF (CHANNEL=2 OR 11 OR 13 OR 45) ... but in SAS one must code: IF (CHANNEL=2 OR CHANNEL=11 OR CHANNEL=13 OR CHANNEL=45) ...
Similarly, %NOTANY(CHANNEL,2 11 13 45) would expand to:
(CHANNEL='2 AND CHANNEL=11 AND CHANNEL=13 AND CHANNEL=45).
Two limitations apply: 1) character values can not have imbedded blanks; 2) each character value in the VALLIST must be enclosed in quotes.

%MACRO ANY(VAR,VALLIST);
%*PURPOSE: SHORTHAND TO GENERATE PART OF SAS "IF" STATEMENT;
%*USE: %ANY(X,I 2 3 9) GENERATES (X=I OR X=2 OR X=3 OR X=9);
%*LIMITATIONS: VALUES CAN NOT HAVE IMBEDDED BLANKS. CHARACTER VALUES MUST BE IN QUOTES.;
%LET POS=1;
%LET VAL=%SCAN(&VALLIST,1);
(&VAR=&VAL
%LET POS=2;
%LET VAL=%SCAN(&VALLIST,&POS);
%DO %UNTIL(&VAL=);
%IF &VAL=" THEN &VAR=&VAL;
%LET POS=%EVAL(&POS+1);
%LET VAL=%SCAN(&VALLIST,&POS);
%END;
)%MEND ANY;

%MACRO NOTANY(VAR,VALLIST);
%*PURPOSE: SHORTHAND TO GENERATE PART OF SAS "IF" STATEMENT;
%*USE: %NOTANY(X,I 2 3 9) GENERATES (X'!=I OR X'!=2 OR X'!=3 OR X'!=9);
%*LIMITATIONS: VALUES CAN NOT HAVE IMBEDDED BLANKS. CHARACTER VALUES MUST BE IN QUOTES.;
%LET POS=1;
%LET VAL=%SCAN(&VALLIST,1);
(&VAR=&VAL
%LET POS=2;
%LET VAL=%SCAN(&VALLIST,&POS);
%DO %UNTIL(&VAL=);
%IF &VAL=" THEN AND &VAR'==&VAL;
%LET POS=%EVAL(&POS+1);
%LET VAL=%SCAN(&VALLIST,&POS);
%END;
)%MEND NOTANY;

VARNAME lists the variable names and their labels from a SAS dataset. This might be useful in a macro-driven system where the user just needs to know which variables are available, but doesn't need all the other info that comes from PROC CONTENTS.

%MACRO VARNAME(DATA=,OUT=ZZV) / STMT;
%IF &DATA= %THEN %DO;
%PUT ENTER THE SAS DATASET NAME;
%INPUT DATA;
%END;
%IF &DATA= %THEN %DO;
%PUT THE SAS DATASET NAME;
%INPUT DATA;
%END;
%PUT THE SAS DATASET NAME: %DATA;
%PUT LABELS LIST FOR &DATA;
PROC PRINT DATA=ZZV; RUN;
%MEND;

USAGE makes access to SAS Usage Notes slightly easier. Instead of typing the ALLOCATE statement (CMS: FILEDEF) and the PROC FSBROWSE... statement, the user simply types %USAGE; Symbolic substitution inside single quotes is shown.

%MACRO USAGE(LIB=SYS3.) / STMT;
%TSO ALLOC F(USAGE)
DA(%STR(%' )&LIB.SAS.USAGE%STR(%'))
SHR REUSE;
PROC FSBROWSE DATA=USAGE.USAGE
SCREEN=USAGE.SCREEN; RUN;
%MEND USAGE;
INDEXBLD creates an index dataset and index functions (FORMATs) to facilitate direct access processing. The index dataset consists of a KEY, the relative record number of the first observation for that key value (X_FIRST), and the relative record number of the last observation for that key value (X_LAST). For example, this index dataset

```
SAMPLE  X_FIRST  X_LAST
A      1       10
B      11      15
C      16      22
D      23      29
```

shows that sample "A" occurs in observations 1 through 10, sample "B" occurs in observations 11 through 15 ... (the dataset from which the index is derived must be sorted by ascending key before %INDEXBLD is executed). Once the index dataset has been created, two formats are built by macro MAKEFMT (described below). These formats can be used to translate a key value into the X_FIRST or X_LAST value via the PUT function.

```
%MACRO INDEXBLD(XDSN=,DSN=,KEY=,FIRST=,LAST=,HELP=0) / STMT;
%IF (&XDSN= OR &DSN= OR &KEY= OR &FIRST= OR &LAST=) AND &HELP=0
%THEN %DO;
%PUT A REQUIRED PARAMETER IS MISSING;
%LET HELP=1;
%END;
%IF &HELP %THEN %DO;
%PUT INDEXBLD -- BUILD AN INDEX DATASET AND INDEX FUNCTIONS;
%PUT REQUIRED PARAMETERS:;
%PUT XDSN= NAME OF INDEX DATASET TO BE BUILT;
%PUT DSN= NAME OF DATASET TO BE INDEXED (SORTED BY KEY);
%PUT KEY= NAME OF THE VARIABLE TO BE THE INDEX KEY;
%PUT FIRST= NAME OF THE FORMAT (TO BE BUILT) THAT IDENTIFIES
%PUT XSTR( )THE FIRST OBS FOR A PARTICULAR KEY VALUE *;
%PUT LAST= NAME OF THE FORMAT (TO BE BUILT) THAT IDENTIFIES;
%PUT XSTR( )THE LAST OBS FOR A PARTICULAR KEY VALUE *;
%PUT * NOTE: IF KEY IS A CHARACTER VARIABLE, FIRST AND LAST;
%PUT XSTR( )MUST BEGIN WITH A DOLLAR-SIGN ($);
%PUT OPTIONAL PARAMETERS:;
%PUT HELP DISPLAY LIST OF PARAMETERS IF SPECIFIED;
%END;
%ELSE %DO;
DATA &XDSN;
  SET &DSN(KEEP=&KEY);
  BY &KEY;
  RETAIN X_FIRST;
  KEEP &KEY X_FIRST X_LAST;
  IF FIRST.&KEY THEN X_FIRST=_N_;
  IF LAST.&KEY THEN OD;
  X_LAST=_N_; OUTPUT;
END;
%MAKEFMT(NAME=&FIRST ,KEY=&KEY ,RESULT=X_FIRST ,DDNAME=SASLIB);
%MAKEFMT(NAME=&LAST ,KEY=&KEY ,RESULT=X_LAST ,DDNAME=SASLIB);
%END;
%MEND INDEXBLD;
```

Example of using an index dataset for direct access retrieval:
```
DATA;
SET DATABASE.INDEX;
IF SAMPLE='C' THEN DO;
  DO POINTER=X_FIRST TO X_LAST;
  SET DATABASE.UNIVERSE;
  OUTPUT;
END;
END;
RUN;
```

Example of using index functions for direct access retrieval:
```
DATA;
X_FIRST=PUT('C',$$FIRST.);
X_LAST=PUT('C',$$LAST.);
DO POINTER=X_FIRST TO X_LAST;
SET DATABASE.UNIVERSE;
OUTPUT;
END;
STOP;
RUN;
```

References:
"Efficient Data Retrieval -- Direct Access Using the Point Option", by Neil Howard and Linda Pickle Williams, SUGI '84 PROCEEDINGS. pages 294-298.
VALIDATE is a macro language utility that is used to compute the destination of a \%GOTO. It functions like the COMPUTED GOTO of FORTRAN. The three positional parameters are: the name of the macro variable to be tested (VARNAME); the list of valid values (VALLIST); and the list of labels (GOLIST) to GOTO depending on which value of VALLIST matches the value of VARNAME. GOLIST always has one more value than VALLIST; the "extra" label is the GOTO label if there is no match. Example:

\%TRYAGAIN: \%PUT ENTER THE TV CHANNEL;
\%INPUT CHANNEL;
\%GOTO \%VALIDATE(CHANNEL,2 4 5 7 9 11 13,BAHT DC DC DC DC BAHT TRYAGAIN);
\%BAHT: \%PUT CHANNEL IS FROM BALTIMORE; \%GOTO SLEEP;
\%DC: \%PUT CHANNEL IS FROM DC;
\%SLEEP: \%PUT TURN OFF THE TV;

\%MACRO VALIDATE(VARNAME,VALLIST,GOLIST);
\* THIS MACRO COMPARES A VALUE TO A LIST OF ACCEPTABLE VALUES,
\* AND RETURNS A LABEL TO WHICH THE CALLING CODE BRANCHES. BASED;
\* ON THE VALIDATE MACRO OF MICHAEL D. RHODES (SUGI 10, PAGE 341).
\* THIS VERSION BY ROSS Z. MERLIN, 06JAN1986;
\%LET VAL=.; \%* ONE VALUE FROM THE VALUE LIST;
\%LET POS=0; \%* POSITION IN VALUE LIST;
\%00 \*UNTIL(IVAL= OR IVAL=IIIVARNAME);
\%LET POS=%EVAL(IPOS+1);
\%LET VAL=%SCAN(IVALLIST,POS);
\% END;
\%IF IVAL= \%THEN \%DO;
\%PUT ERROR: "&VARNAME" IS AN INVALID RESPONSE;
\%PUT \%STR(VALID ANSWERS ARE: &VALLIST..);
\%END;
\%SCAN(&GOLIST,POS)
\%END;
\%END;

OBS finds the number of observations in a SAS dataset. This may be useful in macro driven systems where certain steps should be bypassed if an earlier step resulted in zero observations in a dataset. OBS uses the direct access option NOBS= to get the number of obs. without counting each one; the number is then stored in a macro variable named NOBS. NOBS should be made GLOBAL before \%OBS is executed. Example:

\%OBS(DATA=SAMPLE);
\%IF &NOBS=0 \%THEN \%DO;
\%DATA REPORT; \%MERGE SAMPLE CODEBOOK; \%BY CODE;
\%END;
\%ELSE \%PUT SORRY, NO DATA SELECTED.;

\%MACRO OBS(DATA= LAST ,FILE= LOG,MACVAR=NOBS,HELP=0)/STMT;
\%IF IHELP \%THEN \%DO;
\%PUT PURPOSE: FIND # OF OBS IN A SAS DATASET.;
\%PUT FEATURES: ANSWER MAY BE PRINTED TO A;
\%PUT FILE AND/OR STORED IN A MACRO VARIABLE.;
\%PUT USES DIRECT-ACCESS RATHER THAN COUNTING.;
\%PUT PARAMETERS AND DEFAULTS --;
\%PUT DATA= LAST -- SASDATASET NAME;
\%PUT FILE= LOG WHERE TO PRINT ANSWER;
\%PUT MACVAR=NOBS \%NAME OF MACRO VARIABLE;
\%PUT HELP IF SPECIFIED, LIST PARAMETERS ONLY.;
\%END;
\%ELSE \%DO;
\%DATA NULL;
\%SET NDATA NOBS=NOBS POINT=POINT;
\%ERROR=0;
\%IF FILE= "THEN \%DO;
\%FILE &FILE;
\%IF \&UPCASE(DATA)= LAST \%THEN \%LET DATA=\$SYSDSN;
\%PUT \"DATA HAS " NOBS " OBS."
\%END;
\%IF \&MACVAR= "THEN \%STR(CALL SYMPUT("&MACVAR",PUT(NOBS,12.)));
\%STOP;
\%RUN;
\%END;
\%END OBS;
MAKEFMT generates "PROC FORMAT" statements from observations in a SAS dataset. Techniques demonstrated by this macro include indirect macro variable references, the SYMPUT function, and the DQUOTE option. For further information and enhancements, see "Statement Style Macros" by Don Henderson, SUGI '84 PROCEEDINGS, pages 979 through 988.

%MACRO MAKEFMT(Data=, Key=Key, Result=Result,
                 Name=, Other=, DDName=Library, Help=0) / STMT;
%*----------------------------------------------~----- -----------*;
%* FROM SUGI 84 PROCEEDINGS, P. 980, MAKEFMT BY DON HENDERSON*;
%* MODIFIED BY ROSS Z. MERLIN, 31JUL84, TELEPHONE 202/632-0032 *
%* THIS MACRO CREATES A FORMAT MODULE FROM A SAS DATASET.*;
%* A "KEY=RESULT" STMT IS NOT GENERATED IF KEY-VALUE IS MISSING.*;
%* IF THE DATASET IS EMPTY, THEN A "NULL" FORMAT IS CREATED.*;
%*---------------------------------------------------------------*;
%IF &HELP %THEN %DO;
%PUT PARAMETER=DEFAULT MEANING;
%PUT --------------------------.--.--------.-------.-------.-------.-------.
%PUT DATA= LAST
%PUT KEY=KEY -
%PUT RESULT=RESULT-
%PUT NAME= NAME OF SAS DATASET FOR KEY AND RESULT;
%PUT VARIA8LE FOR FORMAT VALUE (CODED VALUE);
%PUT NAME= VARIABLE FOR FORMAT LABEL (DECODED VALUE);
%PUT NAME= FORMAT NAME - MUST START WITH "$" IF;
%PUT %STR( %PUT OTHER=
%PUT %STR( %PUT DDNAME=LIBRARY
%PUT HELP
%PUT %END; %ELSE %DO;
%IF &Key= OR &RESULT= OR &NAME= %THEN %DO;
%PUT REQUIRED KEYWORD PARAMETER NOT SPECIFIED;
%PUT KEY= &.key RESULT=&result NAME=&name;
%PUT WARNING: FORMAT (&name) COULD NOT BE MADE;
% END;
%ELSE %DO;
DATA NULL;
SET &DATA END=LASTREC;
IF SUBSTR("&other",1,1)=" " THEN DO;
   CALL SYMPUT('OTHER',SUBSTR("&other",2,L));
END;
%IF %SUBSTR(&name,1,1)="$" THEN %STR(IF &key="", THEN DO;);
%ELSE %STR(IF &key="", THEN 00;);
ENTRIES+1;
MVAR=' FMT'!!LEFT(PUT(ENTRIES,4.));
%IF %SUBSTR(&name,1,1)="$" THEN %STR(CALL SYMPUT(MVAR,PUT(&key,BEST.))!!'="''
   TRIM(&result))!!''');
%ELSE %STR(CALL SYMPUT(MVAR,'''!!TRIM(&key))!!''''
   TRIM(&result))!!'''));
END;
IF LASTREC THEN CALL SYMPUT('_ENTRIES',PUT(ENTRIES,4.));
RUN;
PROC FORMAT DDNAME=&DDNAME;
   VALUE &NAME=
   %IF &ENTRIES=0 %THEN %DO;
      %DO I=1 %TO &ENTRIES;
         &FMT &I=
         %END;
   %END;
%IF %QUOTE(&OTHER)=" " THEN OTHER="&other";
%END;
%ELSE %PUT WARNING: A DUMMY FORMAT HAS BEEN COMPILED.;
RUN;
%END; %END;
%MEND MAKEFMT;
SEARCH is an interactive utility to find all occurrences of user-specified character strings in the designated key field of a SAS dataset. Direct access is used to avoid setting the end-of-file flag, thereby allowing the user to search through the data many times for one execution of the macro. Example:

```sas
%search(data=telephone.exchange,key=loc);
ENTER WORD OR PHRASE, NULL LINE TO EXIT

EXCHANGE=281 LOC=VIENNA N=17
EXCHANGE=941 LOC=ANNANDALE N=57
***** 2 MATCHING ENTRIES

EXCHANGE=445 LOC=LANGLEY PARK N=22
EXCHANGE=450 LOC=STERLING PARK N=24
EXCHANGE=454 LOC=COLLEGE PARK N=97
***** 3 MATCHING ENTRIES

%MACRO SEARCH(data= LAST ,key=KEY)/STMT;
%TSO ALLOC F(TERMINAL) DA(*) REUSE;
DATA NULL;
INFILE TERMINAL MISSOVER;
PUT 'ENTER WORD OR PHRASE, ' 'NULL LINE TO EXIT';
INPUT WORD CHAR4.0.;
IF WORD='' THEN STOP;
DO PTR=I TO MAXOB;
   PUT _ALL_; COUNT+1;
END;
END;
COUNT=0;
RUN;
%MEND SEARCH;
```

TYPE determines the type attribute of a variable in a SAS dataset. A GLOBAL macro variable, "VARTYPE", is given the value "NUMERIC", "CHARACTER", or null. Features demonstrated by this macro include the special SAS names _NUMERIC_ and _CHARACTER_, "DO OVER" array processing, and the VNAME subroutine.

```sas
%MACRO TYPE(data= LAST ,var=,variable=,print=yes,help=0) / STMT;
%GLOBAL VARTYPE; %LET VARTYPE=;
%IF &HELP %THEN %00; CLEAR;
%PUT XSTR( ) MACRO "TYPE" IS USED TO DETERMINE THE TYPE ATTRIBUTE;
%PUT (CHARACTER OR NUMERIC) OF A VARIABLE IN A SAS DATASET.;
%PUT XSTR( ) THE VARIABLE AND ITS TYPE MAY BE PRINTED IN THE LOG, AND;
%PUT THE MACRO RETURNS THE VALUE "CHARACTER" OR "NUMERIC" OR NULL;
%PUT XSTR( );
%PUT PARAMETER=DEFAULT MEANING;
%PUT DATA= LAST SAS DATASET CONTAINING VARIABLE TO BE CHECKED;
%PUT VAR= VARIABLE NAME TO HAVE TYPE DETERMINED;
%PUT VARIABLE= (ALTERNATIVE PARAMETER SPELLING FOR VAR);
%PUT PRINT=YES PRINT VARIABLE NAME AND TYPE IN LOG;
%PUT HELP IF SPECIFIED, LIST PARAMETER INFO ONLY.;
%END;
%ELSE %00;
%IF &VAR= AND &VARIABLE= %THEN %00;
%PUT A REQUIRED PARAMETER, "VAR" OR "VARIABLE", WAS NOT SPECIFIED.;
%PUT MACRO "TYPE" CANNOT EXECUTE.;
%END;
%IF &VAR= AND &VARIABLE= %THEN %LET VAR=&VARIABLE;
%LET VAR=%UPCASE(&VAR);
DATA NULL; RESULT='
SET &DATA( OBS=1);
ARRAY NUM_NUMERIC_ ALF _CHARACTER_;
DO OVER NUM:
   CALL VNAME(NUM,RESULT);
   IF RESULT="&VAR" THEN CALL SYMPUT('VARTYPE','NUMERIC');
END;
DO OVER ALF:
   CALL VNAME(ALF,RESULT);
   IF RESULT="&VAR" THEN CALL SYMPUT('VARTYPE','CHARACTER');
END;
RUN;
%IF &PRINT=YES %THEN %00;
%IF &VARTYPE= THEN SPOT VARIABLE &VAR IS NOT IN DATASET &DATA;
%ELSE %PUT &VAR IS &VARTYPE;
%END;
%END;
%MEND TYPE;
```
WP is a word processing application. It would be invoked like a datastep whenever text is to appear in the listing. Instead of doing a "cut-and-paste" job, a complete report can be produced in one job-step. Cover letters, explanations of tables or graphs, conclusions... anytime neat text needs to be interspersed with other SAS output, use %WP.

To use %WP, the raw text is stored in a "flat file". Special codes edited into the text tell %WP what to do -- the default action is to justify the text (flow as many words as possible onto a line, then insert spaces to make an even right margin). The special codes -- commands -- are briefly described in the program comments. For more information, refer to "Word Processing Using SAS" by Ross Z. Merlin, SUGI '82 PROCEEDINGS, pages 241-242. For complete documentation, contact the author at the address given at the end of this paper.

%MACRO WP(NUMBER=NO,DATE=NO,TITLES=NO,OUTDD=PRINT,INDD=TEXT,PS=55,PAGESIZE=,LS=65,LINESIZE=,CENTER=YES,HELP=0)
%STMT;
%PUT NOTE: SAS/WP COPYRIGHT (C) ROSS Z. MERLIN 1982, 1985;
%IF &HELP %THEN %00;
%PUT PARAMETER=DEFAULT
%PUT -----------------
%PUT NUMBER=NO
%PUT DATE=NO
%PUT TITLES=NO
%PUT OUTDD=PRINT
%PUT INDD=TEXT
%PUT PS=55,PAGESIZE=55
%PUT LS=65,LINESIZE=65
%PUT CENTER=YES
%PUT %STR(
%PUT HELP
%END;
%ELSE %00;
MEANING -- PARAMETER FOR OPPOSITE EFFECT;
---------------------------~------------;
NO PAGE NUMBERS -- "YES" OR NULL;
NO DATE/TIME AT TOP OF PAGE -- "YES" OR NULL;
DONT PRINT SAS TITLES -- "YES" OR NULL;
DONAME FOR PROCESSED TEXT;
DONAME FOR RAW TEXT INPUT;
USE EITHER FORM -- MAY BE BETWEEN 20 AND 500;
USE EITHER FORM -- MAY BE BETWEEN 64 AND 132;
CENTER SAS TITLES -- "NO" FOR NOCENTER;
NULL MEANS USE SETTING IN EFFECT;
IF SPECIFIED, LIST PARAMETERS ONLY.;
%IF %UPCASE(&NUMBER)=NO %THEN %STR(OPTIONS NONUMBER;);
%IF %UPCASE(&DATE)=NO %THEN %STR(OPTIONS NODATE;);
%IF %UPCASE(&TITLES)=NO %THEN %LET TITLES=NOTITLES;
%ELSE %LET TITLES=;
%IF &PS= AND &PAGESIZE'= %THEN %LET PS=&PAGESIZE;
%IF &LS= AND &LINESIZE'= %THEN %LET LS=&LINESIZE;
%LET FMT=$CHAR&LS .. ;
%IF %UPCASE(&CENTER)=YES %THEN %STR(OPTIONS CENTER;);
%ELSE %IF %UPCASE(&CENTER)=ND %THEN %STR(OPTIONS NOCENTER;);
FDELETE FORMAT;
IF MADE A PERMENANT FORMAT, REMOVE FROM HERE *
VALUE $WPABBR 'Mr.', 'Ms.', 'Mrs.', 'Dr.', 'Ave.', 'St.', 'Ct.', 'Rd.', 'Ln.', 'Hq.'='?' OTHER=' '; RUN;
DATA NULL; * SAS/WP WORD PROCESSING USING SAS
* COMMANDS:
* $A ASIS (PRINT REST OF THIS LINE AS IS )
* $B BLANK LINE (END CURRENT LINE, LEAVE 1 BLANK BEFORE NEXT)
* $C CENTER (CENTER TEXT FROM REST OF THIS INPUT LINE)
* $1 INDENT 5 SPACES ON NEW LINE
* $P PAGE-EJECT (FORCED START ON NEW PAGE)
* $R RIGHT-SHIFT (SHIFT TEXT FROM THIS INPUT LINE TO RT SIDE OF PAGE)
* $T TITLE-DEFINITION -- REFER TO USER INSTRUCTIONS
* $T TEXT-NEXT (DON'T RECOGNIZE NEXT WORD AS COMMAND)
* AUTOMATIC FEATURES:
* TWO SPACES AFTER ":,!,?," OR PERIOD (.) (EXCEPT AFTER ABBREVS)
* PAGE-EJECT ON PAGE-OVERFLOW;
LENGTH REC RECJ TITLE Tl-T16 CENTRTXT $ ALS TESTWORD $ 50;
LENGTH INDENFLG TEXTNEXT TEST $ 1;
RETAI1N COL 1 WORDCNT TTLX HITT LX 0;
RETAI1N INDENFLG TEXTNEXT 'N' Tl-T16 REC REC '.';
ARRAY TTLS (TTLX) Tl-T16;
IFILE BINOD;
FILE OUTDD PRINT &TITLES N=PS PAGESIZE= &PS HEADER=PRINTHORS;
INPUT WORD :$50. @@;
IF WORD = ' ' THEN RETURN;
L=LENGTH(WORD); EOW=COL+L-I;
IF L = 2 AND TEXTNEXT = 'N' THEN DO: /*MOST COMMANDS ARE LENGTH 2*/
IF WORD = ' ' THEN DO: /*PRINT REST OF LINE AS IS */
IF REC='.' THEN PUT 01 REC &FMT ;/* OUTPUT INCOMPLETE */
INPUT 04 REC &FMT ; /* REC BEFORE "AS IS" TEXT */
PUT 01 REC &FMT ;
709
WORDCNT = 0; COL = 1; REC = ' '; INDENFLG = 'N';

RETURN;

END;

IF WORD = '$B' THEN DO; /* BLANK LINE */
  IF REC = ' ' THEN PUT 01 REC &FMT; /* OUTPUT INCOMPLETE REC */
  PUT; WORDCNT = 0; COL = 1; REC = ' '; /* BEFORE BLANK LINE */
  RETURN;
END;

IF WORD = '$C' THEN DO; /* CENTER TEXT FROM COLS 4 - ON */
  IF REC = ' ' THEN PUT 01 REC &FMT; /* OUTPUT INCOMPLETE REC */
  REC = ' '; /* BEFORE CENTERED TEXT */
  INPUT 04 CENTRTXT &FMT;
  LFTSTART = (8L - FLOOR(LENGTH(LEFT(CENTRTXT))))/2 + 1;
  SUBSTR(REC, LFTSTART) = LEFT(CENTRTXT);
  PUT 01 REC &FMT;
  REC = ' '; WORDCNT = 0; COL = 1; INDENFLG = 'N';
  RETURN;
END;

IF WORD = '$I' THEN DO; /* INDENT -- OUTPUT INCOMPLETE REC */
  IF REC = ' ' THEN PUT 01 REC &FMT; /* BEFORE NEW PARAGRAPH */
  WORD = ' ' /* USES 4 SPACES TO CAUSE INDENTATION */
  REC = ' '; WORDCNT = 0; COL = 1;
  EDW = 4; /* WITH COL=EDW+2, THIS WILL POINT TO COL 6 */
  INDENFLG = 'Y';
END;

IF WORD = '$P' THEN DO; /* PAGE EJECT -- OUTPUT INCOMPLETE REC */
  IF REC = ' ' THEN PUT 01 REC &FMT; /* BEFORE NEW PAGE */
  REC = ' '; WORDCNT = 0; COL = 1;
  PUT PAGE;
  RETURN;
END;

IF WORD = '$R' THEN DO; /* RIGHT SHIFT FROM COL 4 ON */
  IF REC = ' ' THEN PUT 01 REC &FMT;
  INPUT 04 REC &FMT;
  REC = RIGHT(REC);
  PUT 01 REC &FMT;
  REC = ' '; WORDCNT = 0; COL = 1; INDENFLG = 'N';
  RETURN;
END;

IF TEXTNEXT = 'Y'; /* SET FLAG */
RETURN;

END;

IF L = 3 AND SUBSTR(W1RD, 1, 2) = 'ST' AND TEXTNEXT = 'N' THEN DO;
  J = SUBSTR(W1RD, 3, 1); /* SHIFT DIRECTION INDICATOR */
  INPUT 05 TTLX TITLE &FMT;
  CENTRTXT = ' '; /* CENTRTXT IS TEMP HLDG AREA */
  IF TTLX = 0 THEN DO; /* SHIFT PROCESSING */
    IF J = 'L' THEN CENTRTXT = LEFT(TITLE);
    ELSE IF J = 'R' THEN CENTRTXT = RIGHT(TITLE);
    ELSE IF J = 'C' THEN DO;
      LFTSTART = (8L - FLOOR(LENGTH(LEFT(TITLE))))/2 + 1;
      SUBSTR(CENTRTXT, LFTSTART) = LEFT(TITLE);
    END;
    /* J = 'L' OR ANYTHING ELSE MEANS LEAVE AS-IS */
    ELSE CENTRTXT = TITLE;
    IF TTLX < 0 THEN H1TT1X = ABS(TTLX);
    ELSE IF TTLX > H1TT1X THEN H1TT1X = TTLX;
    TTLX = ABS(TTLX); TTLS = CENTRTXT; /* STORE TITLE */
  END;
  ELSE DO; /* (TTLX = 0) CLEAR TITLE ARRAY */
    H1TT1X = 0;
    DO OVER TTLS; TTLS = ' '; END;
  END;
  RETURN;
END;

TEXTNEXT = 'N'; /* RESET FLAG -- RESUME RECOGNITION OF COMMANDS */
IF EDW <= 8L THEN DO; /* IF WORD WILL FIT ON CURRENT LINE... */
  SUBSTR(REC, COL, L) = WORD;
  IF WORD = ' ' THEN WORDCNT + 1; /* CHECK FOR INDENTATION */
  COL = EDW + 2;
END;
ELSE DO;
/* DISTRIBUTE EXTRA SPACES WITHIN REC TO GET RECJ (JUSTIFIED) */
XTRASPCS = LSN - LENGTH(REC);
RECJ = ' '; /* MINIMUM 1 SPACE BETWEEN WORDS - Q=ADDITIONAL */
BTWN WDS = WORDCNT + 1; /* # OF SPACES BTWN EVERY WORD */
Q = FLTR(XTRASPCS / BTWN WDS); /* R= # OF WORDS THAT GET ONE */
R = MOD(XTRASPCS, BTWN WDS); /* LEFTOVER SPACE EACH. */
IF INDENTFLG = 'N' THEN START = 1;
ELSE START = 6;
STARTJ = START;
DO I = 1 TO BTWN WDS;
ENDWORD = INDEX(SUBSTR(REC, START), ' ') - 1;
SUBSTR(RECJ, STARTJ, ENDWORD) = SUBSTR(REC, START, ENDWORD);
TESTWORD = SUBSTR(RECJ, STARTJ, ENDWORD);
IF R = 0 AND I <= R /* EXTRA SPACE BEING PUT IN */
THEN STARTJ = STARTJ + ENDWORD + Q + 2;
ELSE /* Q+1 SPACES (BTWN EVERY WORD) BEING PUT IN */
STARTJ = STARTJ + ENDWORD + Q + 1;
START = START + ENDWORD + 1;
IF (TEST = '.' AND PUT(TESTWORD,$WPABBR.) = '?') OR
INDEXC(TEST, '!:?') = 1 THEN DO;
START = START + 1; /* DOUBLE SPACE AFTER ",,:!,?" */
STARTJ = STARTJ + 1; /* BUT NOT AFTER COMMON ABBREVS */
END;
END;
SUBSTR(RECJ, STARTJ) = SUBSTR(REC, START); /* GET THE REST */
PUT @1 RECJ &FMT; /* PRINT THE JUSTIFIED LINE */
WORDCNT = 0; REC = ' '; INDENTFLG = 'N';
SUBSTR(REC, L) = WORD;
IF WORD = '.' THEN WORDCNT + 1; /* CHECK FOR INDENTATION */
COL = L + 2;
END;
IF (SUBSTR(WORD, L, 1) = '.' AND
/* OBL-Space AFTER ".,,:!,?" */
PUT(WORD,$WPABBR.) = '?') OR INDEXC(SUBSTR(WORD, L, 1), '!:?') = 1
THEN COL = COL + 1; /* EXCEPT AFTER COMMON ABBRS */
RETURN;
PRNTHDRS: IF HITTLX = 0 THEN RETURN;
ELSE DO TTLX = 1 TO HITTLX; PUT @1 TTLS &FMT; END;
RETURN;
RUN;
%END; WP:

For further information, please contact:

Ross Z. Merlin 202/632-0032 or
Pinkerton Computer Consultants, Inc. 703/820-5571
1900 North Beauregard St., Suite 200
Alexandria, VA 22311

711