

Screen Control Language — the Undercover Sleuth of FSEDIT*

Robert F. Davis
Sverdrup Technology, Inc., AEDC Group
Arnold Engineering Development Center
Arnold Air Force Base, TN 37389

ABSTRACT

Entering data with the SAS® system can be enhanced by using Screen Control Language (SCL) programming with the FSEDIT procedure. This presentation outlines a number of useful SCL programming techniques that facilitate data entry and database management. In data entry, the topics covered will include double entry validation, using predetermined lists to check for data validity, automatic filling in of data fields, designation of required fields, and defining variable calculations. For management of databases that are accessible by multiple users, programming methods will be presented that will allow for the logging in of users, the validation of user and data entry permissions, and generation of audit trails.

INTRODUCTION

The integrity and validity of data in a database management system is of utmost importance. After all, "garbage in equals garbage out." The human factor is the biggest source of errors in getting data into a database. The ideas and techniques expressed here go a long way toward eliminating some of the error problems associated with data entry systems built with PROC FSEDIT. The PROC FSEDIT session allows for the creation and storage of customized data entry screens. Along with these screens, Screen Control Language (SCL) programs can also be stored. SCL can be used like a detective, to prevent human errors by alerting data entry personnel and aiding them in locating the error, as data are entered. The examples presented here have been tested on an IBM® 3090 mainframe with MVS operating system, under TSO using SAS 6.08, TS404.

THE EDIT SESSION

FSEDIT is a procedure in the SAS/FSP® product. When PROC FSEDIT is invoked with the 'screen=' option, as in Fig. 1, for the first time, a default screen image is displayed with all the variables in the data set specified. Figure 8 shows the structure of the data set used by the FSEDIT procedure in Fig. 1. Issuing the modify command brings up a selection menu. Menu option 2 allows for the creation of customized data entry screen(s). Figure 9 shows a customized screen used for this discussion. Menu option 3 opens the SCL source language editor window. SCL code can be typed in manually or included from an external source with the include command. Figure 2 illustrates the SCL code used for this discussion. SCL under PROC FSEDIT uses five dedicated labels for controlling execution. The FSEINIT section is executed only once during the session before any observations are displayed. The INIT section is executed once per observation before the observation is displayed. The MAIN section executes each time the user modifies a screen variable and presses the enter key. The control statement can be used to override this default behavior. The TERM section executes once for each observation just prior to moving to the next observation if the observation has been modified. The control statement can be used to override this default behavior. The FSETERM section executes when the user issues the end command to terminate the session.

FSEINIT

Five arrays are defined for later use with double entry verification. The 'names' array contains a list

* The research reported herein was performed by the Arnold Engineering Development Center (AEDC), Air Force Materiel Command. Work and analysis for this research were performed by personnel of Sverdrup Technology, Inc., AEDC Group, technical services contractor. Further reproduction is authorized to satisfy needs of the U. S. Government.

of all the variable names that are to be verified with double entry. Character variables must come first, followed by numeric variables. Arrays 'verify' and 'firsttime' are used as flag arrays to indicate when a double entry variable has been modified and when the verification data have been entered. These two arrays must be the same length as array 'names'. Array 'cflids' contains a list of all the character variables in array 'names' in the same order. The length of array 'cflids' must be the length of the longest variable in its list. Array 'nflids' contains all of the numeric variables in array 'names' in the same order. A control statement is issued with options always, label, and term. The always option allows the main section to be entered any time the enter key is pressed. The label option allows the code labeled with a screen variable name to be executed when the user modifies that variable and presses the enter key. This execution takes place before the main section is entered. The term option specifies that the term section will be executed whenever the user moves from the current observation, even if no screen variables were modified. The user id is obtained from the system with a symget to 'SYSJOBID'. A security data set (Fig. 5) is opened and an attempt to find the current user is made. If the user is not found, a show list command displays a message 'ACCESS DENIED'. When the user presses the enter key, a 'BYE' command is executed immediately, and the session is ended. If the current user id is found in the security file, the user name is retrieved and the access level is validated. Access levels can be programmed to allow or disallow input to specific screen variables. A protect command is issued for the screen variables 'keyid', 'totmatc', 'totmhr', and 'totprc'. These are calculated screen variables that no user is allowed to change. The security file is then closed. The user log (Fig. 6) is opened and updated to indicate the current user session type, user id, user name, and current date, and then closed. A link to "JRNFINIT" routine is issued. This routine is executed one time at the beginning of the edit session. It establishes the necessary temporary data sets and variables to record any changes made during the edit session in an audit journal (Fig. 3). The employees (Fig. 4) and valids (Fig. 7) data sets are opened for later use while observations are being changed. A call set statement to the employee's data set links its variables of the same name to the screen variables. At this point, the FSEINIT section is complete, and a return is issued, returning control to the edit session.

INIT

If the current observation is to be added or duplicated, the appropriate add or dup flag is set from the last command code received. A link to "JRN_INIT" is issued. This routine is executed to record a copy of the data from the current observation (dsid_old) before it is displayed. An exception to this is if a "DUP" command was issued, in which case, a record of all missing values is set instead of a copy. Later, in the "TERM" section, these data will be compared to see if any changes have been made. At that time, if any changes have been made, an appropriate entry is made in the audit journal. A return from "JRN_INIT" back to the "INIT" section is issued. The "INIT" section now issues a return and returns control to edit session.

MAIN

The "CONTROL ALWAYS" statement is executed in the "FSEINIT" section, allowing main to be entered each time the user presses the enter key. To begin with, a link to "JRN_MAIN" is issued. In this routine, the command line is interrogated and flags set to identify the command later. If the command is a delete, a link to "WRITEJRN" is issued to update the audit journal for the current observation as deleted. In the "WRITEJRN" section, the temporary data set for changes (dsid_chg) is set to all missing values. This causes the data comparison to be unequal, and all variables are written to the audit journal as deleted. A return statement continues the processing of the "MAIN" section. If the current observation has been added or modified and the current command on the command line is not "cancel" or "delete", the rest of the main section is executed; otherwise, it is skipped. A check is now made to see if any of the numeric variables listed for double entry verification have invalid values. If they have invalid values the program returns to the edit session where a system error message is displayed for the user. If the variable names in the 'names' array have been modified they are marked with a 1 in the 'verify' array for double entry verification. If an element of the 'verify' array is true and the corresponding element in the 'firsttime' array is not, that element of the 'firsttime' array is set to 1, and the corresponding screen variable is copied to a temporary character or numeric variable. A jump to "ENDCHECK" routine is executed. The next time this code is executed, 'verify' and 'firsttime' will be

true. At this point, 'verify' and 'firsttime' will be set to zero. The stored variable data will now be compared against the verification data. If they are unequal, an error flag is set and a jump to "ENDCHECK" routine is executed. At the "ENDCHECK" routine, if 'firsttime' is true, a message 'ENTER DATA AGAIN' FOR VERIFICATION' is set for display to the user. If error is true, a message 'DATA ENTRY DID NOT VERIFY, PLEASE TRY AGAIN' is set for display to the user. At this point, the total manhours, total material cost, and total project cost screen variables are calculated and set. A return is issued and control is now returned to the edit session.

TERM

The "CONTROL TERM" statement is executed in the "FSEINIT" section. This allows the "TERM" section to be entered any time the user attempts to move from the current observation. If the current observation has not been deleted or canceled, links are executed to check for required fields (phase and title). A description of these processes will follow in the "SCREEN VARIABLE LABELS" section. If the current observation was modified, added, or duplicated, a link to "JRN_TERM" is executed. In "JRN_TERM", if this observation was added or duplicated, the audit journal change code is set to 'AD'. The last key value used is now retrieved and incremented by one. This new key value is assigned to the current observation and stored back in the last key file. At this point a fall through to "WRITEJRN" section occurs. If the current observation was not deleted, the temporary changes data set is updated to the current screen contents. Then all the variables are retrieved from the temporary data sets and compared one at a time. If they are unequal, a record for each variable is written to the audit journal with the variable name, old and new values, date/time, key, user id, and change code. When the journal was opened in the "JRNFINIT" routine, a 'call set' statement linked the date/time and user id variables to the SCL variables. This allows the date/time and user id variables to be written to the journal without a 'put var' statement. A return is issued, returning control to the "TERM" section. Flags are now reset according to the current conditions, and a return is executed, returning control to the edit session.

FSETERM

This section is entered when the user issues the end command from the command line. This is a termination housekeeping routine. A link to "JRNFTERM" is executed where the journal and the two temporary data sets (old_recs and changes) are closed. The temporary data sets (old_recs and changes) are now deleted. A return statement brings us back to the "FSETERM" section where the employees and valids data sets are closed. A return is issued, and the edit session is terminated.

SCREEN VARIABLE LABELS

In the "FSEINIT" section a 'control label' statement is executed. This allows a section of code, labeled with the same name as a screen variable, to be executed any time it is modified. The following are descriptions of four screen variables set up in this application as labeled variables.

PHASE

Phase is a required field. A where clause is made from the current value of the screen variable 'phase'. This where clause is processed through the valids data set. If a match is found, this routine returns from where it came. If no match is found, a pull down box with a list of valid values is presented to the user. A message requesting the user to choose one value is displayed. When the user presses the enter key, the selection is transferred to the screen variable 'phase'. A return is executed back to where the routine came from. Figure 10 shows the screen image with a pull down box for valid phases.

JOB

If job number is entered, it must be entered with four digits including leading zeros. This routine checks to see that this condition exists. If this condition does not exist, a message is formatted for display to the user, and the cursor is positioned at the job field upon return. Figure 12 shows a screen image for an invalid job entry.

TITLE

This is a required field. If this field is blank, a message is formatted for display to the user upon return from this routine and the cursor is positioned to the 'title' field. Figure 11 shows the screen image of a title field found to be missing.

LASTNAME

The engineer's lastname is used to look up and fill in other variables on the edit screen. The lastname is used to make a where clause that searches the employee's data set for a match. If no match is found, a message is set to be displayed to the user and the cursor is positioned to the engineer's lastname field upon return. Figure 13 shows an example screen of this condition. If a match is found, a check is made to see if there is more than one lastname. If only one occurrence of the lastname is found in the employees data set, the record is fetched and a return is issued, returning control from where it was called. If more than one occurrence of lastname is found, a list is made of duplicate lastnames in the employee's data set and displayed to the user in a pull down box (Fig. 14). The user then must select one name. When the user presses the enter key, the selection is set in the screen variable. A call set is executed when the employees data set is opened in the "FSEINIT" section, linking the employees data set variables to the screen variable of the same name. The variables 'lastname', 'first', 'ext', 'emp_no', and 'ms' are automatically set on the edit screen by the fetch or locate statements. A return is issued, returning control to where this routine was called.

REFERENCES

"SAS Language Reference, Version 6." First Edition.

"SAS Procedures Guide, Version 6." Third Edition.

"SAS Technical Report P-222." Changes and Enhancements to Base SAS Software, Release 6.07.

"SAS Technical Report P-242." SAS Software: Changes and Enhancements, Release 6.08.

"SAS/FSP Software Usage and Reference, Version 6." First Edition.

"SAS Screen Control Language Usage, Version 6." First Edition.

"SAS Screen Control Language Reference, Version 6." First Edition.

SAS Technical Report P-199. "Using SAS Screen Control Language in Release 6.06."

SAS Technical Report P-216. "SAS/AF Software, SAS/FSP software, and SAS Screen Control Language: Changes and Enhancements, Release 6.07."

"SAS/SHARE Usage and Reference, Version 6." First Edition.

SAS Technical Report P-212. "SAS/SHARE Software for the MVS Environment, Release 6.07."

ACKNOWLEDGMENTS

SAS is a registered trademark of SAS Institute Inc.
SAS campus Drive, Cary, N.C. 27513
IBM is a registered trademark of IBM Corporation

The code dealing with the Audit Journaling was taken from the SAS Observations publication Vol. 1 #1 pages 31-38 (4Q91:31-38) and modified for use with this application.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC. */
/* SOURCE CODE NAME: EDIT */
/* AUTHOR: ROBERT F. DAVIS */
/* PURPOSE: THIS CODE STARTS THE FSEDIT SESSION. */
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
PROC FSEDIT DATA=PRJ.PROJECTS SCREEN=PRJ.PRJSCR;
DATA _NULL_;
LIBNAME PRJ CLEAR;
RUN;
QUIT;

```

Figure 1. Edit session startup.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC. */
/* SOURCE CODE NAME: FSEDTSCL */
/* AUTHOR: ROBERT F. DAVIS */
/* PURPOSE: THIS IS A SCREEN CONTROL LANGUAGE PROGRAM */
/* THAT RUNS IN THE BACKGROUND OF THE FSEDIT */
/* SESSION. */
*****/
FSEINIT:
  LENGTH COMMAND $3;
  LENGTH USERID $6 LASTCODE $2 NAME $20;
  /*NOTE: CTEMP MUST HAVE A LENGTH EQUAL TO THE LENGTH */
  /* OF CFLDS ARRAY.*/
  LENGTH CTEMP $35;
  LENGTH NTEMP 8;
  /* ARRAY OF VARIABLE NAMES FOR DOUBLE ENTRY VERIFICATION.*/
  /*LIST ALL CHARACTER VARIABLES FIRST FOLLOWED BY NUMERICS.*/
  ARRAY NAMES {*} $8 ('TITLE' 'JOB' 'MHR' 'MHR' 'MATC'
    'FY' 'PRI');
  /* NOTE: ARRAYS VERIFY AND FIRSTIME MUST BE THE SAME */
  /* LENGTH AS THE ARRAY "NAMES" */
  ARRAY VERIFY {*} V1-V7;
  ARRAY FIRSTIME {*} F1-F7;
  /* NOTE: ALL CHARACTER FIELDS IN NAMES MUST BE LISTED IN */
  /* ARRAY CFLDS IN THE SAME ORDER. CFLDS LENGTH */
  /* MUST BE THE LENGTH OF THE LONGEST FIELD IN THE */
  /* LIST.. */
  ARRAY CFLDS {*} $35 TITLE JOB;
  /* NOTE: ALL NUMERIC FIELDS IN NAMES MUST BE LISTED IN */
  /* ARRAY NFLDS IN THE SAME ORDER. */
  ARRAY NFLDS {*} MHR MHR MATC FY PRI;

  CONTROL ALWAYS LABEL TERM;
  USERID=SYMGGET('SYSJOBID');

  /*OPEN THE SECURITY DATA SET*/
  DSID_SEC=OPEN('PRJ.SECURITY','I');
  IF (DSID_SEC=0) THEN _MSG_ =SYMSMSG();
  RC=LOCATE(DSID_SEC,VARNUM(DSID_SEC,'BADGE'),USERID);
  IF (RC LE 0) THEN DO;
    RC=SHOWLIST(' ','ACCESS DENIED');
    CALL EXECMDI('BYE');
  END;

  USERNAME=GETVARC(DSID_SEC,VARNUM(DSID_SEC,'NAME'));
  LEVEL=GETVARC(DSID_SEC,VARNUM(DSID_SEC,'SEC_LEV'));
  IF (USERID EQ '000001' AND LEVEL EQ 1) THEN UNPROTECT _ALL_;
  /* THIS USER ID HAS FULL EDIT ACCESS*/
  ELSE PROTECT _ALL_; /*ALL OTHERS HAVE NO EDIT ACCESS*/
  PROTECT KEYID TOTMATC TOTMHR TOTPRJC;
  RC=WHERE(DSID_SEC);
  RC=CLOSE(DSID_SEC);

```

Figure 2. Edit session screen control language.

```

/* OPEN USER LOG FOR UPDATE. */
USERLOG=OPEN('PRJ.USERLOG','U');
TYPE = 'EDIT';
DATE= DATETIME();
RC=APPEND(USERLOG);
CALL PUTVARN(USERLOG,VARNUM(USERLOG,'DATE'),DATE);
CALL PUTVARC(USERLOG,VARNUM(USERLOG,'NAME'),USERNAME);
CALL PUTVARC(USERLOG,VARNUM(USERLOG,'BADGE_NO'),USERID);
CALL PUTVARC(USERLOG,VARNUM(USERLOG,'TYPE'),TYPE);
RC=UPDATE(USERLOG);
RC=CLOSE(USERLOG);
LINK JRNFINIT;

```

```

/*OPEN THE EMPLOYEE DATA SET*/
DSID_EMP=OPEN('PRJ.EMP','I');
IF (DSID_EMP=0) THEN _MSG_ =SYMSMSG();
CALL SET(DSID_EMP);

```

```

/*OPEN THE FIELD VALADATION DATA SET*/
DSID_VAL=OPEN('PRJ.VALIDS','I');
IF (DSID_VAL=0) THEN _MSG_ =SYMSMSG();
RETURN;

```

```

INIT:
  IF (LASTCODE EQ 'AD') THEN ADFLAG=1;
  ELSE IF (LASTCODE EQ 'DU') THEN DUPFLAG=1;
  LINK JRN_INIT;
  RETURN;

```

```

MAIN:
  LINK JRN_MAIN;
  RCMOD=OBSINFO('MODIFIED');
  IF ((RCMOD OR ADFLAG) AND
    (COMMAND NE 'CAN' AND COMMAND NE 'DEL'))
  THEN DO;
    FLD_ERR=0;
    DO I=1 TO DIM(NFLDS);
      IF (FIELD('ERROR',NAMES(I+DIM(CFLDS)))) THEN FLD_ERR=1;
    END;
    IF (FLD_ERR) THEN RETURN;
    DO I=1 TO DIM(NAMES);
      IF (FIELD('MODIFIED',NAMES(I))) THEN VERIFY(I)=1;
    END;
    DO I=1 TO DIM(NAMES);
      IF (VERIFY(I)) THEN DO;
        IF (NOT FIRSTIME(I)) THEN DO;
          FIRSTIME(I)=1;
          IF (I LE DIM(CFLDS)) THEN DO;
            CTEMP=CFLDS(I);
            CFLDS(I)='';
          END;
        ELSE DO;
          NTEMP=NFLDS(I-DIM(CFLDS));
          NFLDS(I-DIM(CFLDS))=.;
        END;
        GO TO ENDCHECK;
      END;
    ELSE DO;
      FIRSTIME(I)=0;
      VERIFY(I)=0;
      IF (I LE DIM(CFLDS)) THEN DO;
        IF (CTEMP NE CFLDS(I)) THEN DO;
          ERROR=1;
          GO TO ENDCHECK;
        END;
      END;
    END;
  END;

```

Figure 2. Continued.

```

ELSE DO;
  IF (NTEMP NE NFLDS(I-DIM(CFLDS))) THEN DO;
    ERROR=1;
    GO TO ENDCHECK;
  END;
END;
END;
END;
END;
END;
ENDCHECK:
IF (I LE DIM(NAMES)) THEN DO;
  IF (FIRSTIME(I) OR ERROR) THEN DO;
    IF (I LE DIM(CFLDS)) THEN CFLDS(I)='';
    ELSE NFLDS(I-DIM(CFLDS))=.;
    RC=FIELD('CURSOR',NAMES(I));
    _STATUS_='R';
    IF (FIRSTIME(I)) THEN
      _MSG_='ENTER DATA AGAIN FOR VERIFICATION';
    ELSE DO;
      ERROR=0;
      _MSG_='DATA ENTRY DID NOT VERIFY, PLEASE TRY AGAIN';
    END;
  END;
END;
END;
TOTMHR=SUM(MHRC,(MHR*6)); /* SURCHARGE FOR MAN HOURS */
TOTMTC=MATC*1.0825; /* TAX ON MATERIALS */
TOTPRJC=SUM(TOTMHR,TOTMTC); /* TOTAL PROJECT COST */
END;
RETURN;

```

TERM:

```

RC=OBSINFO('DELETED');
IF (COMMAND NE 'CAN' AND RC NE 1) THEN DO;
  LINK PHASE;
  LINK TITLE;
  RCMOD=OBSINFO('MODIFIED');
  IF (RCMOD EQ 1) OR (ADFLAG) OR (DUPFLAG) THEN DO;
    LINK JRN_TERM;
  END;
  IF (DUPFLAG) THEN DUPFLAG=0;
  ELSE IF (ADFLAG) THEN ADFLAG=0;
END;
ELSE DO;
  ADFLAG=0;
  DUPFLAG=0;
  DO I=1 TO DIM(NAMES);
    VERIFY(I)=0;
    FIRSTIME(I)=0;
  END;
END;
END;
RETURN;

```

FSETERM:

```

LINK JRNFTERM;
CALL CLOSE(DSID_EMP);
CALL CLOSE(DSID_VAL);
RETURN;

```

JRNINIT:

```

/* OPEN DATA SET FOR UPDATE. */
/* MAP SCL VARS TO DATA SET */
/* DATA VECTOR (DDV). */
JOURNAL=OPEN('PRJ.JOURNAL','U');
CALL SET(JOURNAL);

```

```

/* CREATE OLD_RECS AND CHANGES DATA SETS */
/* BY USING MASTER DATA SET AS MODEL */
/* OPEN BOTH TEMPORARY DATA SETS FOR UPDATE. */
CALL NEW('OLD_RECS','PRJ.PROJECTS',1,N);
DSID_OLD=OPEN('OLD_RECS','U');
CALL SET(DSID_OLD);
CALL NEW('CHANGES','OLD_RECS',1,N);
DSID_CHG=OPEN('CHANGES','U');
CALL SET(DSID_CHG);

```

```

/* _NVAR= # OF VARS IN DATA SET */
_NVAR=ATTRN(DSID_OLD,'NVAR');

```

```

/* READ FIRST RECORDS */
RC=FETCHOBS(DSID_OLD,1);
RC=FETCHOBS(DSID_CHG,1);

```

```

/* INITIALIZE VARS FOR PUTVAR */
NUM_CHNG=VARNUM(JOURNAL,'VARCHNG');
NUM_OLD=VARNUM(JOURNAL,'OLDVALUC');
NUM_OLDN=VARNUM(JOURNAL,'OLDVALUN');
NUM_NEWC=VARNUM(JOURNAL,'NEWVALUC');
NUM_NEWN=VARNUM(JOURNAL,'NEWVALUN');
NUM_KEY1=VARNUM(JOURNAL,'KEY1');
NUM_CODE=VARNUM(JOURNAL,'CODE');
NUMOKEY1=VARNUM(DSID_OLD,'KEYID');
RETURN;

```

JRN_INIT:

```

/* UPDATE OLD_RECS TO NULL RECORD */
IF (LASTCODE EQ 'DU') THEN DO;
  RC=DELOBS(DSID_OLD);
  RC=APPEND(DSID_OLD,'NOSET');
END;
/* UPDATE W/ CURRENT VALUES FROM MASTER */
ELSE RC=UPDATE(DSID_OLD);
RETURN;

```

JRN_MAIN:

```

/* RETURN COMMAND IN UPPER CASE */
COMMAND=WORD(1,'U');
/* LOOK AT FIRST THREE CHARACTERS */
COMMAND=SUBSTR(COMMAND,1,3);
SELECT(COMMAND);
/* USER DELETED RECORD */
WHEN('DEL')
DO;
  CODE='D';
  LASTCODE='DE';
/* UPDATE THE JOURNAL DATA SET FOR DELETED RECORD */
LINK WRITEJRN;
END;
/* USER CANCELLED RECORD */
WHEN('CAN')
DO;
  CODE='';
  LASTCODE='';
END;

```

Figure 2. Continued.

Figure 2. Continued.

```

/* USER ADDED RECORD */
WHEN('ADD')
DO;
  CODE='A';
  LASTCODE='AD';
END;

/* USER DUPLICATED RECORD */
WHEN('DUP')
DO;
  CODE='C';
  LASTCODE='DU';
END;
OTHERWISE
DO;
  CODE='C';
  LASTCODE=' ';
END;
END; /* END SELECT CLAUSE */
RETURN;

JRN_TERM:
/* CHECK ADD OR DUP FLAG AND SET CODE FOR ADDS */
/* UPDATE KEY FOR NEW RECORD */
IF (ADFLAG OR DUPFLAG) THEN DO;
  CODE='AD';
  DSID_KEY=OPEN('PRJ.LASTKEY','U');
  IF (DSID_KEY=0) THEN _MSG_=SYSMSG();
  RC=FETCHOBS(DSID_KEY,1);
  NUMLKEY1=VARNUM(DSID_KEY,'LASTKEY');
  KEYID=GETVARN(DSID_KEY,NUMLKEY1)+1;
  CALL PUTVARN(DSID_KEY,NUMLKEY1,KEYID);
  RC=UPDATE(DSID_KEY);
  RC=CLOSE(DSID_KEY);
END;

WRITEJRN:
/* CREATE THE JOURNAL OBSERVATION IF NECESSARY */
DATETIME=DATETIME(); /* NOT USED IN THIS PROGRAM */
/* IS WRITTEN TO JOURNAL ENTRIES */
/* UPDATE CHANGES DATA WITH CURRENT SCREEN VALUES */
IF (CODE NE 'D') THEN RC=UPDATE(DSID_CHG);
/* DELETE COMMAND, SET CHANGES DATA TO MISSING VALUES */
ELSE DO;
  RC=DELOBS(DSID_CHG);
  RC=APPEND(DSID_CHG,'NOSET');
END;
DO _J=1 TO _NVAR;
  IF VARTYPE(DSID_OLD,_J) EQ 'N' THEN DO;
    _NUM1=GETVARN(DSID_OLD,_J);
    _NUM2=GETVARN(DSID_CHG,_J);
    /* LET CODE REFLECT NUMERIC CHANGE */
    SUBSTR(CODE,2,1)='N';
  END;
  ELSE DO;
    _CHAR1=GETVARN(DSID_OLD,_J);
    _CHAR2=GETVARN(DSID_CHG,_J);
    /* LET CODE REFLECT CHARACTER CHANGE */
    SUBSTR(CODE,2,1)='C';
  END;
  IF ((_CHAR1 NE _CHAR2) OR (_NUM1 NE _NUM2)) THEN DO;
    RC=APPEND(JOURNAL);
    CALL PUTVARN(JOURNAL,NUM_CHNG,VARNAM(DSID_OLD,_J));
    IF (_CHAR1 NE _CHAR2) THEN DO;

```

```

/* SAVE OLD VALUE */
  CALL PUTVARN(JOURNAL,NUM_OLDC,_CHAR1);
  /* SAVE NEW VALUE */
  CALL PUTVARN(JOURNAL,NUM_NEWC,_CHAR2);
  _CHAR1=' '; _CHAR2=' ';
END;
ELSE DO;
  /* SAVE OLD VALUE */
  CALL PUTVARN(JOURNAL,NUM_OLDN,_NUM1);
  /* SAVE NEW VALUE */
  CALL PUTVARN(JOURNAL,NUM_NEWN,_NUM2);
  _NUM1=.; _NUM2=.;
END;

/* ADD OTHER VARIABLES THAT ARE NOT IN */
/* CHANGES DATA SET IF NEW OBSERVATION */
IF (SUBSTR(CODE,1,1) EQ 'A') THEN KEY1=KEYID;
/* GET KEY FROM LAST KEY USED */
/* ELSE GET KEY FROM OLD_RECS */
ELSE KEY1=GETVARN(DSID_OLD,NUMOKEY1);
CALL PUTVARN(JOURNAL,NUM_KEY1,KEY1);
CALL PUTVARN(JOURNAL,NUM_CODE,CODE);
RC=UPDATE(JOURNAL);
END;
END;
RETURN;

JRNFTERM:
/* CLOSE DATA SETS, FSEDIT TERMINATES */
RC=CLOSE(JOURNAL);
RC=CLOSE(DSID_OLD);
RC=CLOSE(DSID_CHG);

/* DELETE TEMPORARY WORK DATA SETS */
RC=DELETE('OLD_RECS');
RC=DELETE('CHANGES');
RETURN;

/*BEGIN SCREEN VARIABLE LABELS*/
PHASE:
WC1S1='VPHASE NE ' || ' ';
RC=WHERE(DSID_VAL,WC1S1);
IF (RC GT 0) THEN
DO;
  _MSG_=SYSMSG() || ' PHASE';
  RETURN;
END;
RC=LOCATEC(DSID_VAL,VARNUM(DSID_VAL,VPHASE'),PHASE);
IF (RC LE 0) THEN DO;
  CALL WREGION(15,10,9,60);
  PHASE=DATA1STC(DSID_VAL,VPHASE PH_DESC',
    'VALID PHASES, PLEASE CHOOSE ONE');
END;
RC=WHERE(DSID_VAL);
RETURN;

JOB:
L=LENGTH(JOB);
IF (L NE 0 AND L LT 4) THEN DO;
  _STATUS_='R';
  CURSOR JOB;
  _MSG_='PLEASE ENTER ALL 4 DIGITS INCLUDING LEADING ZEROS';
END;

```

Figure 2. Continued.

Figure 2. Continued.

```

RETURN; TITLE:
  IF (TITLE EQ '') THEN DO;
    CURSOR TITLE;
    _STATUS_='R';
    _MSG_='THIS IS A REQUIRED FIELD';
  END;
RETURN;

LASTNAME:
NAME=LASTNAME;
WC1S1=SCAN(LASTNAME,1) EQ ' ' || NAME || '';
RC=WHERE(DSID_EMP,WC1S1);
NUM=ATTRN(DSID_EMP,'ANY');
IF (NUM EQ 0) THEN
  DO;
    _MSG_='NO SUCH ENGINEER'S NAME FOUND';
    _STATUS_='R';
    CURSOR LASTNAME;
    ALARM;
    RC=WHERE(DSID_EMP);
    RETURN;
  END;
COUNT=0;
DO WHILE (FETCH(DSID_EMP) NE -1);
  COUNT+1;
  IF (COUNT EQ 1) THEN NOTEID=NOTE(DSID_EMP);
END;
IF (COUNT EQ 1) THEN
  DO;
    RC=POINT(DSID_EMP,NOTEID);
    RC=FETCH(DSID_EMP);
    IF (RC GT 0) THEN
      DO;
        _MSG_=SYSMSG();
        RC=WHERE(DSID_EMP);
        RETURN;
      END;
    END;
  ELSE
  DO;
    CALL WREGION(10,15,10,80);
    CVAL=DATALISTN
      (DSID_EMP,EMP_NO LASTNAME FIRST EXT MS);
    RC=LOCATEN(DSID_EMP,VARNUM(DSID_EMP,EMP_NO),CVAL);
    IF (RC LE 0) THEN
      DO;
        _MSG_=SYSMSG();
        RC=WHERE(DSID_EMP);
        RETURN;
      END;
    END;
  RC=WHERE(DSID_EMP);
RETURN;

```

Figure 2. Concluded.

```

/*****
COMPANY: SVERDRUP TECHNOLOGY, INC.
SOURCE CODE NAME: PRJJOJ
AUTHOR: ROBERT F. DAVIS
PURPOSE: THIS PROGRAM INITIALIZES AN AUDIT JOURNAL.
THIS SHOULD BE RUN ONLY ONCE TO ESTABLISH THE INITIAL
DATA SET.. THERE AFTER IT IS KEPT UP BY THE SCREEN
CONTROL LANGUAGE PROGRAMING IN THE DATA BASE ENTRY
SYSTEM.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.JOURNAL;
  ATTRIB KEY1 LENGTH=8 LABEL='RECORD IDENTIFIER';
  ATTRIB CODE LENGTH=$2 LABEL='CHANGE CODE';
  ATTRIB VARCHING LENGTH=$8 LABEL='CHANGE VARIABLE NAME';
  ATTRIB OLDVALUN LENGTH=8 LABEL='OLD NUMERIC VALUE';
  ATTRIB NEWVALUN LENGTH=8 LABEL='NEW NUMERIC VALUE';
  ATTRIB OLDVALUC LENGTH=$50 LABEL='OLD CHARACTER VALUE';
  ATTRIB NEWVALUC LENGTH=$50 LABEL='NEW CHARACTER VALUE';
  ATTRIB DATETIME FORMAT=DATETIME18. INFORMAT=DATETIME18.
    LABEL='CHANGE DATE';
  ATTRIB USERID LENGTH=$6 LABEL='ID OF USER THAT MADE CHANGE';
RUN;
QUIT;

```

Figure 3. Code to create audit journal data set.

```

/*****
COMPANY: SVERDRUP TECHNOLOGY, INC.
SOURCE CODE NAME: PRJEMP
AUTHOR: ROBERT F. DAVIS
PURPOSE: THIS PROGRAM CREATES AN EMPLOYEE DATA
SET.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.EMP;
  ATTRIB LASTNAME LENGTH=$20 LABEL='EMPLOYEE LAST NAME';
  ATTRIB FIRST LENGTH=$20 LABEL='EMPLOYEE FIRST NAME';
  ATTRIB EXT LENGTH=8 LABEL='EMPLOYEE PHONE NUMBER';
  ATTRIB EMP_NO LENGTH=8 LABEL='EMPLOYEE NUMBER';
  ATTRIB MS LENGTH=8 LABEL='EMPLOYEE MAIL STOP';
  LASTNAME='BAKER'; FIRST='JOE'; EXT=1111; EMP_NO=000001;
  MS=2200; OUTPUT;
  LASTNAME='SMITH'; FIRST='JOHN'; EXT=2222; EMP_NO=000002;
  MS=2400; OUTPUT;
  LASTNAME='BAKER'; FIRST='HELEN'; EXT=1212; EMP_NO=000003;
  MS=2400; OUTPUT;
  LASTNAME='PIERSON'; FIRST='DONALD'; EXT=2323; EMP_NO=000004;
  MS=2600; OUTPUT;
  LASTNAME='CORTNER'; FIRST='DAVID'; EXT=4545; EMP_NO=000005;
  MS=2800; OUTPUT;
RUN;
QUIT;

```

Figure 4. Code to create employee data set.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC.
/* SOURCE CODE NAME: PRJSEC
/* AUTHOR: ROBERT F. DAVIS
/* PURPOSE:: THIS PROGRAM CREATES A SECURITY DATA SET.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.SECURITY;
ATTRIB NAME LENGTH=$20 LABEL='USER NAME';
ATTRIB BADGE LENGTH=$6 LABEL='BADGE NUMBER';
ATTRIB SEC_LEV LENGTH=8 LABEL='SECURITY LEVEL';
NAME='HELEN BAKER'; BADGE='000003'; SEC_LEV=1;
OUTPUT;
NAME='JOHN SMITH'; BADGE='000002'; SEC_LEV=2;
OUTPUT;
RUN;
QUIT;

```

Figure 5. Code to create security data set.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC.
/* SOURCE CODE NAME: PRJUSLOG
/* AUTHOR: ROBERT F. DAVIS
/* PURPOSE:: THIS PROGRAM INITIALIZES A USER LOG.
/* THIS SHOULD BE RUN ONLY ONCE TO ESTABLISH THE INITIAL
/* DATA SET. THERE AFTER IT IS KEPT UP BY THE SCREEN
/* CONTROL LANGUAGE PROGRAM IN THE DATA ENTRY SYSTEM.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.USERLOG;
ATTRIB NAME LENGTH=$20 LABEL='USER NAME';
ATTRIB DATETIME FORMAT=DATETIME18. LABEL='ENTRY DATE';
ATTRIB BADGE_NO LENGTH=$6 LABEL='ID OF USER';
ATTRIB TYPE LENGTH=$6 LABEL='EDIT OR BROWSE SESSION';
RUN;
QUIT;

```

Figure 6. Code to create user log data set.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC.
/* SOURCE CODE NAME: PRJVAL
/* AUTHOR: ROBERT F. DAVIS
/* PURPOSE:: THIS PROGRAM CREATES A DATA SET OF VALID
/* VALUES USED BY THE FSEDIT SCREEN CONTROL LANGUAGE.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.VALIDS;
ATTRIB VPHASE LENGTH=$2 LABEL='VALID PROJECT PHASES';
ATTRIB PH_DESC LENGTH=$60 LABEL='PROJECT PHASE DESCRIPTION';
VPHASE='P'; PH_DESC='PLANNING'; OUTPUT;
VPHASE='D'; PH_DESC='DESIGN'; OUTPUT;
VPHASE='NP'; PH_DESC='NEW PROJECT (FIRST YEAR EXECUTION)';
OUTPUT;
VPHASE='MY'; PH_DESC='MULTI YEAR EXECUTION'; OUTPUT;
RUN;
QUIT;

```

Figure 7. Code to create valid values data set.

```

/*****
/* COMPANY: SVERDRUP TECHNOLOGY, INC.
/* SOURCE CODE NAME: PROJECTS
/* AUTHOR: ROBERT F. DAVIS
/* PURPOSE:: THIS PROGRAM CREATES THE STRUCTURE FOR
/* THE PROJECTS DATA SET. DATA IS ENTERED IN THIS DATA
/* SET BY THE FSEDIT SESSION.
*****/
LIBNAME PRJ 'STI.SESUG96.PROJECTS';
DATA PRJ.PROJECTS;
ATTRIB TITLE LENGTH=$35 LABEL='PROJECT TITLE';
ATTRIB LASTNAME LENGTH=$20 LABEL='ENGINEER'S LASTNAME';
ATTRIB FIRST LENGTH=$20 LABEL='ENGINEER'S FIRST NAME';
ATTRIB EXT LENGTH=8 LABEL='ENGINEERS PHONE EXT.';
ATTRIB EMP_NO LENGTH=8 LABEL='EMPLOYEE NUMBER';
ATTRIB MS LENGTH=8 LABEL='ENGINEER'S MAIL STOP';
ATTRIB PHASE LENGTH=$2 LABEL='PROJECT PHASE';
ATTRIB JOB LENGTH=$4 LABEL='JOB NUMBER';
ATTRIB MHRS LENGTH=8 LABEL='PROJECT MANHOURS';
ATTRIB MHRC LENGTH=8 LABEL='PROJECT MANHOURS COST';
ATTRIB MATC LENGTH=8 LABEL='MATERIAL COST';
ATTRIB TOTMATC LENGTH=8 LABEL='TOTAL MATERIAL COST';
ATTRIB TOTMHRC LENGTH=8 LABEL='TOTAL MANHOUR COST';
ATTRIB TOTPRJC LENGTH=8 LABEL='TOTAL PROJECT COST';
ATTRIB FY LENGTH=8 LABEL='PROJECT FISCAL YEAR';
ATTRIB PRI INFORMAT=5.1 LABEL='PROJECT PRIORITY';
ATTRIB KEYID LENGTH=8
LABEL='UNIQUE KEY ID FOR EACH DATA RECORD';
SET PRJ.PROJECTS;
RUN;
QUIT;

```

Figure 8. Code to create projects data set.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>>		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
MAIL STOP:	2400	
JOB NO.:	0001	
PROJECT PHASE:	NP	
COMPLEX:	TUR	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	200	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	7200	
TOTAL PROJECT COST:	61325	
KEY ID:	1	

Figure 9. Basic data entry screen.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
THIS IS A REQUIRED FIELD		
PROJECT TITLE:	>	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
MAIL STOP:	2400	
JOB NO.:	0001	
PROJECT PHASE:	D	
COMPLEX:	HYP	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	200	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	7200	
TOTAL PROJECT COST:	61325	
KEY ID:	1	

Figure 11. Input screen showing a required field input.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
MAIL STOP:	2400	
JOB NO.:	0001	
PROJECT PHASE:	NP	
Select Data		
Command ==>		
VALID PHASES, PLEASE CHOOSE		
> P	PLANNING	
D	DESIGN	
NP	NEW PROJECT (FIRST YEAR EXECUTION)	
MY	MULTI YEAR EXECUTION	

Figure 10. Input screen showing pull down box of valid values.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
PLEASE ENTER ALL 4 DIGITS INCLUDING LEADING ZEROS		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
MAIL STOP:	2400	
JOB NO.:	> 1	
PROJECT PHASE:	D	
COMPLEX:	HYP	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	200	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	7200	
TOTAL PROJECT COST:	61325	
KEY ID:	1	

Figure 12. Input screen showing invalid entry for Job No.

FSEDIT PRJ.PROJECT		Obs 1
Command ==>		
NO SUCH ENGINEER'S NAME FOUND		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	>XAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
MAIL STOP:	2400	
JOB NO.:	0001	
PROJECT PHASE:	D	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	200	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	7200	
TOTAL PROJECT COST:	61325	
KEY ID:	1	

Figure 13. Input screen showing invalid engineer's name.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
ENTER DATA AGAIN FOR VERIFICATION		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	JOE	
ENGINEER EXT:	1111	
ENGINEER EMPLOYEE NO:	1	
MAIL STOP:	2200	
JOB NO.:	0001	
PROJECT PHASE:	D	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	> _____	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	6000	
TOTAL PROJECT COST:	60125	
KEY ID:	1	

Figure 15. Input screen showing double entry verification request.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	HELEN	
ENGINEER EXT:	1212	
ENGINEER EMPLOYEE NO:	3	
Select Data		
Command ==>		
Make one selection		
> 1 BAKER	JOE	1111 2200
3 BAKER	HELEN	1212 2400
TOTAL PROJECT COST:		61325
KEY ID:		1

Figure 14. Input screen showing pull down box for duplicate engineer's name.

FSEDIT PRJ.PROJECTS		Obs 1
Command ==>		
DATA ENTRY DID NOT VERIFY, PLEASE TRY AGAIN		
PROJECT TITLE:	PROJECT A	
ENGINEER LASTNAME:	BAKER	
ENGINEER FIRST:	JOE	
ENGINEER EXT:	1111	
ENGINEER EMPLOYEE NO:	1	
MAIL STOP:	2200	
JOB NO.:	0001	
PROJECT PHASE:	D	
PROJECT FISCAL YEAR:	1996	
PROJECT PRIORITY:	10.0	
MANHOURS:	> _____	
MANHOURS COST:	6000	
MATERIAL COST:	50000	
TOTAL MATERIAL COST:	54125	
TOTAL MANHOURS COST:	6000	
TOTAL PROJECT COST:	60125	
KEY ID:	1	

Figure 16. Input screen showing double entry not verified.