CREATING HOT-KEYS FOR DATA ENTRY IN FSEDIT

Alexa Parliyan
Pfizer Inc., New York, NY

INTRODUCTION

While creating an FSEDIT screen for data entry, the users I was working with requested that I give them a "hot-key" for entering a value that they constantly enter: today's date. The techniques shown in this Coder's Corner paper provide two ways to designate a single key for entry of often-used values in an FSEDIT screen. This paper provides specific information for entering dates, however, these techniques may be modified and used for other values as well.

THE EXAMPLE

The FSEDIT screen I was creating required users to enter dates in a log-style screen. As each step in the process was completed, the user would enter the date:

Study:	ABC123
Date in:	01JUL96
Date reviewed:	02JUL96
Date finalized:	03JUL96

Since the dates were stored as SAS date values with the numeric informats of ddmmyy, the users were required to enter six-digit numbers for each date. If today's date was July 1, 1996, and several studies were to be logged, the user would need to go to each study observation and type "010796" in the desired field. Instead, what they wanted was to type simply the letter "t", or the function key F1, which would set the desired field to today's date.

A SPECIAL CASE: SAS DATE VALUES

Typically, if a value to be entered was in a constant range, the conversion could be hard-coded. For example, if a user had a choice of entering 1 for red, 2 for white and 3 for blue, the code could directly map the input values with the output via formatting or using computation variables on the FSEDIT screen.

However, the date values I used had numeric informats. If "1" was entered for a date field with an informat, it would not match the informat, and the field would not be updated. If there was no informat, SAS would require the actual SAS date value. Hence, an entry of "1" would yield "02JAN60".

METHOD 1: CREATING COMPUTATIONAL FIELDS

The first method of allowing users to enter one key to represent a value employs temporary screen variables that are of a different type than the date field to be updated. These character variables were created using the computational variable creation screen when I built the FSEDIT screen. While the updated field is a numeric SAS date value with numeric informat, the user is actually updating the character screen variables. The program checks to see if the screen variable is updated, then maps one to the other.

The code below will allow users to have any character key assigned to varying values, such as the letter "t" for today's date. It also allows users to enter dates with numeric informats to enter character dates, which are more intuitive.

```
/*  Three temporary computational variables     */
/*  must be created first, SCRVAR1-SCRVAR3      */

/* Initialize screen value with current value   */
INIT:
   scrvar1=compress(put(date1,date7.),'.');
   scrvar2=compress(put(date2,date7.),'.');
   scrvar3=compress(put(date3,date7.),'.');
return;

/* Map an entry of "t" to today's date          */
/* Map valid date value to numeric equivalent   */
MAIN:
  if modified(scrvar1) then do;
       if substr(scrvar1,1,1)='T' then do;
          date1=today();
          scrvar1 =put(today(),date7.);
       end;
       else if input(scrvar1,date7.) ne . then do;
            date1=input(scrvar1,date7.);
       end;
  end;

   repeat for date2 and date3......

return;

TERM:
return;
```

METHOD 2: USING THE SCL WORD FUNCTION

A more straight forward method of attaining the same results is to use the SCL *word* function. The *word* function will allow users to have function keys

assigned to varying values, such as today's date (or yesterday's date, or tomorrow's date). Using a function key eliminates the creation of temporary screen variables, and any mapping.

The following code sets a function key to the value SETDATE. The SCL code will then assign SETDATE to today's date in SAS date value on the field where the cursor is located.

```
/*  A FSEDIT.KEYS entry must be created in the  */
/*  the FSEDIT screen catalog that has SETDATE  */
/*  on one of the keys                          */

/* declare an array to reference values of      */
/* each of the date vars                        */

  array values{3} 8 date1 date2 date3;

/* create an SCL list with names of the date    */
/* vars to search on                            */
FSEINIT:
  control always;
  fldlist=makelist();
  rc=insertc(fldlist,'DATE1',-1);
  rc=insertc(fldlist,'DATE2',-1);
  rc=insertc(fldlist,'DATE3',-1);
return;

/* Test for SETDATE command. If executed and    */
/* cursor is on a date field, update the field  */
/* with today's date. It's assumed that the     */
/* numeric date fields have date formats        */
/* associated with them                         */
MAIN:
  command=word(1,'u');
  if command='SETDATE' then do;
     field=curfld();
     pos=searchc(fldlist,field);
     if pos then do;
        values{pos}=today();
        _msg_='Date '||field||'has been updated.';
     end;
     else _msg_='This is not a date field.';
  end;
return;

/* Delete the SCL list from memory              */
FSETERM:
  fldlist=dellist(fldlist);
return;
```

CONCLUSION

This Coder's Corner paper has shown two ways to allow more user-friendly FSEDIT screens. I have found that this feature has been a tremendous time-saver and very well-received by users.

AUTHOR

The author welcomes any comments or questions regarding this paper. Please contact:

Alexa Parliyan
Pfizer Inc.
235 East 42nd Street
New York, NY 10017

Tel: 212-573-1073
Fax: 212-916-7147
e-mail: parlia@pfizer.com