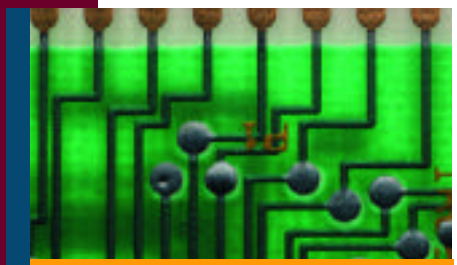

BY FRANK L. YAEGER

Year 2000: Sorting Special Indicators With DFSORT's E61

Special indicators such as 000000 and 999999 mixed in with real dates can pose Year 2000 sorting problems. An E61 routine can be an effective way to handle these problems. By using the information DFSORT passes as well as information you read from a dataset, you can create flexible E61 routines to handle different century windows and all the types of dates you use.

A DFSORT customer asked the following question:

"I am involved in a Year 2000 engagement at a major Australian Bank. When date is used as the major sort field, it is common practice here to use '000000' in the date field (yymmdd) to sort or force a header record to the front of a dataset and '999999' to force the trailer (or summary) record to the back. Using DFSORT's windowing capability for records with dates beyond 991231 would mean that header and trailer records will not appear at the beginning and end of the file — is this right? Can you give me any tips on easy ways to overcome this?"

When real dates such as 991231 and 001029 are mixed with special indicators such as 000000 and 999999, it can pose problems for Year 2000 sorting. This article will demonstrate how E61 routines can be used with DFSORT to solve these problems. Examples will demonstrate how a little programming in an E61 routine can have a big effect on the results of sorting date fields of various types and formats. Although I will concentrate on sorting here, keep in mind that E61 routines can be used for merging as well.

WHAT'S THE PROBLEM?

To understand the problem, let's look at some records with a C'yymmdd' field starting in position 1. This field contains real dates before January 1, 2000 as well as special indicators:

```
990518  -> 05/18/1999
999999  -> HIGH INDICATOR
970805  -> 08/05/1997
000000  -> LOW INDICATOR
```

Sorting these records with

```
SORT FIELDS=(1,6,BI,A)
```

works just fine, giving us the following as output:

```
000000  -> LOW INDICATOR
970805  -> 08/05/1997
990518  -> 05/18/1999
999999  -> HIGH INDICATOR
```

Now let's look at what happens when we add a real date after January 1, 2000:

```
990518  -> 05/18/1999
999999  -> HIGH INDICATOR
970805  -> 08/05/1997
000000  -> LOW INDICATOR
000106  -> 01/06/2000
```

We can no longer use 1,6,BI for the date field because it doesn't do the windowing necessary to treat 00 as 2000. As discussed in previous articles (see "References"), we would normally use DFSORT's Y2C or Y2S format and Y2PAST option to handle the windowing, as shown here:

```
SORT FIELDS=(1,2,Y2C,A,3,4,BI,A)
OPTION Y2PAST=1950
```

But both Y2C and Y2S give us the following incorrect output:

```
970805  -> 08/05/1997
990518  -> 05/18/1999
999999  -> HIGH INDICATOR
000000  -> LOW INDICATOR
000106  -> 01/06/2000
```

Although the real dates are now sorted correctly, the special indicators are not. Y2C and Y2S formats only deal with the

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

two-digit year, not with the entire date. Handling real dates and special indicators for all possible date types (e.g., yymmdd, mmddyy, ddmmyy, yyddd, dddyy, yymm, mmyy, etc.) in CH/ZD and PD formats would require a very large number of new date formats and is thus not supported directly.

Y2S can handle binary zeroes, blanks, and binary ones as special indicators in the two-digit year, but cannot handle a special indicator spread over an entire date. Both Y2C and Y2S use windowing for yy values of 00 and 99, treating them as 2000 and 1999, respectively. We need a way to avoid windowing for 000000 and 999999. The solution is to use an E61 routine.

WHAT'S AN E61 ROUTINE?

During processing DFSORT allows you to plug in your own programming logic by supplying various user exit routines. For example, you can use an E15 routine written in Assembler or COBOL to delete, change, or insert input records before sorting occurs. Likewise, you can use an E35 routine written in Assembler or COBOL to delete, change or insert output records after sorting occurs. User exit routines allow you to do specific or complex processing that goes beyond DFSORT's rich set of built-in functions. The *DFSORT Application Programming Guide* describes all of DFSORT's user exits in detail.

An E61 routine can change specific control fields before sorting occurs. DFSORT passes a copy of each requested control field to the E61 routine, which can change the value in the copy of the control field but not its length. DFSORT sorts with the changed copy of the control field, using ascending sequence and the format specified for the control field. The actual fields in your records are not changed, so you can manipulate fields any way you like in your E61 routine without worrying that the changes will affect your output records or other functions such as INCLUDE or SUM. Only sorting is affected by E61 routine changes. Your E61 routine must be written in Assembler.

To use an E61 routine, you supply DFSORT control statements such as

```
MODS E61=(E61RTN,400,EXIT)
SORT FIELDS=(21,6,BI,E,1,8,CH,A,11,5,BI,E)
```

In the first line you tell DFSORT to use your E61 routine with the MODS control statement. Your E61 routine is named E61RTN, is approximately 400 bytes long (a rough estimate is fine here), and is located in the library identified by an EXIT DD statement. If your E61 routine resides in the STEPLIB, JOBLIB or link library, you would omit the third parameter (ddname).

In the second line you tell DFSORT to pass a control field to your E61 routine with the SORT control statement. Specify E (E61) for the sequence parameter rather than A (ascending) or D (descending). In the example above, the first control field (21,6,BI) and third control field (11,5,BI) will be passed to your E61 routine. Every time DFSORT reads an input record, it calls your E61 routine for each E control field. With the SORT statement above, when DFSORT reads the first input record, it calls your E61 routine passing the 21,6,BI control field and then calls your E61 routine passing the 11,5,BI control field. DFSORT will do that for every input record.

Figure 1 shows the format of the information DFSORT passes to your E61 routine for each control field (CF). Register 1 has a pointer to this information. The CF number is 1 for the first A, D or E control field, 2 for the second control field, etc., although only the E control fields are passed to your E61 routine. For the SORT statement above, the following occurs:

- For the first CF (21,6,BI), DFSORT passes a CF number of 1, a pointer to the six bytes at position 21, and a CF length of 6.
- For the third CF (11,5,BI), DFSORT passes a CF number of 3, a pointer to the five bytes at position 11, and a CF length of 5.

Figure 1: Information Passed to E61

Byte 1	Byte 2	Byte 3	Byte 4
00	00	00	CF number
00	Pointer to copy of CF		
Not used	Not used	CF length	

HOW CAN E61 SOLVE THE DATE PROBLEM?

Let's start with the simplest case of using an E61 routine to handle dates with special indicators. Assume you have only C'yymmdd' date fields with C'000000' as a low indicator, C'999999' as a high indicator, and real dates. You want to sort the date fields in ascending order using a century window of 1950 to 2049 for the real dates. Your E61 routine has to change the copy of the date so that C'000000' sorts first, C'yymmdd' values sort next as appropriate 19yy and 20yy dates, and C'999999' sorts last. Figure 2 shows an E61 routine named E61EX1 with logic to handle this simple case. Figure 3 shows some sample JCL to use this E61 routine. E61EX1 uses a hardcoded value of X'yy' to determine the start of the century window as 19yy (e.g., X'50' for 1950). It has logic to return encoded dates to DFSORT as follows:

```
C'000000' -> C'0',X'F0',C'0000' for the low indicator
C'yymmdd' -> C'1',X'yy',C'mmdd' for 19yy dates
             C'2',X'yy',C'mmdd' for 20yy dates
C'999999' -> C'9',X'F9',C'9999' for the high indicator
```

The use of C'0', C'1', C'2'and C'9' in the first byte of the returned date ensures that the special indicators and years before and after the start of the century window will sort correctly. The use of X'yy' in the second byte for the real dates ensures that dates in the same century will sort correctly. The key to a successful E61 routine is the logic to encode the control fields so they will sort in the desired order. Keep in mind that the encoded control field must be the same length as the original control field. You can usually find several encoding schemes that work; just pick an efficient one that you like. For best performance, try to minimize the number of instructions your E61 routine executes, since it will be called one or more times for each input record.

HOW CAN E61 DO MORE?

E61EX1 has the following limitations (well, I said it was simple):

- The century window is hardcoded. You need a different E61 routine for each century window you want to use.

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

- The century window is fixed. E61EX1 doesn't allow for a sliding century window.
- E61EX1 only handles C'yymmdd' dates. It can't handle other types of CH dates (e.g., C'yyddd', C'mmddyy', etc.) or any type of PD date (e.g., P'yymmdd', P'yyddd', P'mmddyy', etc.).
- E61EX1 only sorts dates in ascending order; it can't sort them in descending order.

You can overcome all of these limitations by putting additional logic in your E61 routine. The CF number, CF address, and CF length that DFSORT passes to your E61 routine will help. In addition, you can have your E61 routine read a record that contains any information you need about the century window or the date fields.

Special indicators such as 000000 and 999999 mixed in with real dates can pose Year 2000 sorting problems.

Figure 2: Source for the E61EX1 Routine

```
E61EX1    CSECT
*==> CHANGE EQU BELOW TO X'yy'       FOR START OF CENTURY WINDOW
CWSTART   EQU   X'50'                START OF CENTURY WINDOW = 1950
          USING   E61EX1,12          SHOW BASE REG
          STM     14,12,12(13)       SAVE REGS
          LA      12,0(0,15)         SET BASE REG
          ST      13,SAVE61+4        SAVE BACKWARD POINTER
          LA      14,SAVE61          SET FORWARD POINTER
          ST      14,8(13)               IN SAVE AREA
          LR      13,14              SET OUR SAVE AREA
          LR      3,1                GET PARMLIST POINTER
          USING   PARML,3            SHOW PARMLIST BASE REG
          L       4,PARMPTR          GET POINTER TO CURRENT CF
          USING   CF,4               SHOW CURRENT CF BASE REG
* ENCODE   C'yymmdd' AS C'i',X'yy',C'mmdd'
          CLC     DATE,=CL6'000000'  IF ZEROES,
          BE      GOBACK                 KEEP C'000000' (LOW INDICATOR)
          CLC     DATE,=CL6'999999'  IF NINES,
          BE      GOBACK                 KEEP C'999999' (HIGH INDICATOR)
          PACK    WORK,DATE(2)       PACK yy -> X'0yyF'
          LH      1,WORK             GET yy AS X'0yyF'
          SRL     1,4                GET yy AS X'yy'
          STC     1,DATE+1           SET X'yy' AS SECOND BYTE OF DATE
          MVI     DATE,IND19         INDICATE 19yy IN FIRST BYTE OF DATE
          CLI     DATE+1,CWSTART     IF 19yy IS CORRECT,
          BNL     GOBACK                 KEEP IT FOR OUTPUT YEAR
          MVI     DATE,IND20         INDICATE 20yy IN FIRST BYTE OF DATE
GOBACK    L       13,4(,13)
          LM      14,12,12(13)       RESTORE REGS
          BR      14                 RETURN
SAVE61    DS      18F                REGISTER SAVE AREA
WORK      DS      CL2                WORK AREA
IND19     EQU     C'1'               C'1' FOR 19yy INDICATOR
IND20     EQU     C'2'               C'2' FOR 20yy INDICATOR
          LTORG
PARML     DSECT                      PARMLIST PASSED TO E61
          DS      3C                     NOT USED
PARMNUM   DS      C                      NOT USED (CF NUMBER)
PARMPTR   DS      A                      ADDRESS OF CF
          DS      2C                     NOT USED
PARMLEN   DS      H                      NOT USED (CF LENGTH)
CF        DSECT                      CURRENT E CF
DATE      DS      CL6                    DATE - C'yymmdd'
          END
```

Figure 3: Sample JCL for the E61EX1 Routine

```
//EX1RUN EXEC  PGM=ICEMAN
//SYSOUT DD SYSOUT=*
//EXIT DD DSN=...  library containing the E61EX1 routine
//SORTIN DD DSN=...
//SORTOUT DD DSN=...
//SYSIN DD *
  SORT FIELDS=(15,6,BI,E,       C'yymmdd'
               5,8,CH,D,        Not a date
               25,6,BI,E)       C'yymmdd'
  MODS E61=(E61EX1,200,EXIT)
/*
```

Let's say you want your E61 routine to handle different fixed century windows and sort C'yymmdd', C'yyddd', P'yymmdd', and P'yyddd' dates with special indicators in ascending or descending sequence. Your E61 routine can use the passed CF length to determine the type and format of each date as follows:

- CF length of 3: P'yyddd'(X'yydddC')
- CF length of 4: P'yymmdd' (X'0yymmddC')
- CF length of 5: C'yyddd'
- CF length of 6: C'yymmdd'

You can have your E61 routine read an information record to tell you the following:

- the start of the fixed century window
- the sequence for each date field

Figure 4 shows an E61 routine named E61EX2 with the logic to handle two date fields for this more complicated case. Figure 3 shows some sample JCL to use this E61 routine. E61EX2 uses an information dataset that looks like this:

```
//INFO DD *
yy u,v w,x
/*
```

yy allows you to specify any starting year (00 to 99) for the century window. u is the relative number (one to nine) of the first E control field. v is the sequence (A for ascending or D for descending) to be used for the first E control field. w and x are the relative number and sequence for the second E control field. You can omit w,x if you only have one E control field.

E61EX2 opens the INFO dataset, reads and saves the information record, and closes

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

Figure 4: Source for the E61EX2 Routine

```
E61EX2    CSECT
          USING   E61EX2,12         SHOW BASE REG
          STM     14,12,12(13)      SAVE REGS
          LA      12,0(0,15)        SET BASE REG
          ST      13,SAVE61+4       SAVE BACKWARD POINTER
          LA      14,SAVE61         SET FORWARD POINTER
          ST      14,8(13)              IN SAVE AREA
          LR      13,14             SET OUR SAVE AREA
          LR      2,1               GET PARMLIST POINTER
          USING   PARML,2           SHOW PARMLIST BASE REG
          L       3,PARMPTR         GET POINTER TO CURRENT CF
          USING   CF,3              SHOW CURRENT CF BASE REG
          LH      4,PARMLEN         GET CURRENT CF LENGTH
          BCTR    4,0               SUBTRACT 1 FOR EX INSTRUCTIONS
          TM      FLAGS,OPNDNE      IF OPEN DONE,
          BO      PROCFLD               SKIP OPEN/GET/CLOSE
          OPEN    (INFODCB,(INPUT)) OPEN INFO DATA SET
          OI      FLAGS,OPNDNE      SHOW OPEN DONE
          GET     INFODCB,INFORCD   GET THE INFORMATION RECORD
          CLOSE   INFODCB           CLOSE INFO DATA SET
          NI      INNUM1,X'0F'      CHANGE FIRST CF NUMBER TO BINARY
          NI      INNUM2,X'0F'      CHANGE SECOND CF NUMBER TO BINARY
          PACK    WORK,INCWSTRT     PACK START OF CW -> X'0yyF'
          LH      1,WORK            GET INPUT YEAR AS X'0yyF'
          SRL     1,4               GET INPUT YEAR AS X'yy'
          STC     1,CWSTART         SET X'yy' AS START OF CW
          USING   CFINFO,5          SHOW CURRENT CF INFORMATION BASE
PROCFLD   LA      5,INSEQ1          POINT TO SEQUENCE FOR FIRST CF
          CLC     INNUM1,PARMNUM    IF FIRST INFO CF NUMBER MATCHES,
          BE      CKFMT                 USE ITS INFORMATION
          LA      5,INSEQ2          POINT TO SEQUENCE FOR SECOND CF
          CLC     INNUM2,PARMNUM    IF SECOND INFO CF NUMBER MATCHES,
          BE      CKFMT                 USE ITS INFORMATION
          B       GOBACK            NO MATCH - LEAVE CF UNCHANGED
CKFMT     CLI     PARMLEN+1,4       CHECK LENGTH OF CF
          BL      PD3CKSP               IF 3, IT'S P'yyddd'
          BE      PD4CKSP               IF 4, IT'S P'yymmdd'
CHFMT     EX      4,CHCMP0          IF ZEROES,
          BE      CKSEQ                 KEEP C'0...0' (LOW INDICATOR)
          EX      4,CHCMP9          IF NINES,
          BE      CKSEQ                 KEEP C'9...9' (HIGH INDICATOR)
* ENCODE   C'yymmdd' AS C'i',X'yy',C'mmdd'
* ENCODE   C'yyddd'  AS C'i',X'yy' C'ddd'
          PACK    WORK,DATE(2)      PACK yy -> X'0yyF'
          LH      1,WORK            GET yy AS X'0yyF'
          SRL     1,4               GET yy AS X'yy'
          STC     1,DATE+1          SET X'yy' AS SECOND BYTE OF DATE
          MVI     DATE,CIND19       INDICATE 19yy IN FIRST BYTE OF DATE
          CLC     DATE+1(1),CWSTART IF 19yy IS CORRECT,
          BNL     CKSEQ                 KEEP 19yy INDICATOR
          MVI     DATE,CIND20       INDICATE 20yy IN FIRST BYTE OF DATE
          B       CKSEQ             CHECK SEQUENCE
PD3CKSP   CLC     DATE(3),PZROS+1   IF ZEROES,
          BE      CKSEQ                 KEEP X'00000C' (LOW INDICATOR)
          CLC     DATE(3),PNINES+1  IF NINES,
          BE      CKSEQ                 KEEP X'99999C' (HIGH INDICATOR)
* ENCODE   X'yydddC' AS X'iyyddd'
          MVC     WORK(1),DATE      COPY X'yy'
          SLR     1,1               CLEAR WORK REG
          ICM     1,7,DATE          GET X'yyddd0'
          SRL     1,4               GET X'0yyddd'
          STCM    1,7,DATE          PUT IT BACK
          B       PDCOM             DO COMMON CODE
PD4CKSP   CLC     DATE(4),PZROS     IF ZEROES,
          BE      CKSEQ                 KEEP X'0000000C' (LOW INDICATOR)
          CLC     DATE(4),PNINES    IF NOT NINES,
          BNE     PD4DT                 ITS A REAL DATE
          OI      DATE,PINDHIGH     ENCODE AS X'9999999C' (HIGH INDICATOR)
          B       CKSEQ             CHECK SEQUENCE
* ENCODE   X'0yymmddC' AS X'i0yymmdd'
PD4DT     L       1,DATE            GET X'0yymmddC'
          SRL     1,4               GET X'00yymmdd'
          ST      1,DATE            PUT IT BACK
```

(Continued on next page.)

the INFO dataset. It uses the yy value in the information record to determine the starting year of the fixed century window.

E61EX2 uses the CF length to determine the type and format of each date as discussed above. It uses the CF length and the format it determines from it to check for the appropriate special indicators (e.g., CL6'0' for C'yymmdd', PL4'0' for P'yymmdd', etc.). E61EX2 uses the same encoding scheme as E61EX1 for CH dates. It encodes P'yyddd' (X'yydddC') dates like this:

```
X'00000C' -> X'00000C' for the low indicator
X'yydddC' -> X'1yyddd' for 19yy dates
             X'2yyddd' for 20yy dates
X'99999C' -> X'99999C' for the high indicator
```

and encodes P'yymmdd' X('0yymmddC') dates like this:

```
X'0000000C'-> X'0000000C' for the low indicator
X'0yymmddC'-> X'10yymmdd' for 19yy dates
           -> X'20yymmdd' for 20yy dates
X'0999999C'-> P'9999999C' for the high indicator
```

Note that the encoded values for the real dates are not valid PD values since they do not have signs in the last four bits. For the X'yydddC' case in particular, we had to eliminate the sign (C) to make room for the indicator (0, 1, 2, or 9) before yyddd. As a result, we cannot use PD format for these fields in the SORT statement and instead must use BI format so DFSORT will not treat the last four bits as a sign. In general, it's a good idea to use BI format for E control fields and encode them as bit values.

To determine whether to use ascending or descending sequence for the date, E61EX2 looks at the u,v (e.g., 2,A) and w,x (e.g., 3,D) values in the saved information record. If u matches the passed CF number, E61EX2 uses either ascending sequence if v is A or descending sequence if v is D. Likewise, for w,x. For descending sequence, E61EX2 just inverts the bits of the date. Since DFSORT always sorts E control fields in ascending sequence, inverting the bits will result in descending sequence instead.

One last note: In a real production environment, you would probably want to add error handling to your E61 routine. You could also change the information record and E61 routine logic to support any additional functions you need such as other date types (e.g., C'mmddyy', P'dddyy', etc.), sliding windows, more E fields, etc.

© 1998 Technical Enterprises, Inc. Reproduction of this document without permission is prohibited.

(Figure 4 continued from page 28.)

```
          MVC     WORK(1),DATE+1  COPY X'yy'
PDCOM     CLC     WORK(1),CWSTART IF 19yy IS NEEDED,
          BNL     PDS19                SET 19yy INDICATOR
          OI      DATE,PIND20       INDICATE 20yy IN FIRST BYTE OF DATE
          B       CKSEQ             CHECK SEQUENCE
PDS19     OI      DATE,PIND19       INDICATE 19yy IN FIRST BYTE OF DATE
CKSEQ     CLI     CFSEQ,C'D'        IF DESCENDING SEQUENCE NOT NEEDED,
          BNE     GOBACK               RETURN
          EX      4,INVERT          INVERT BITS FOR DESCENDING SEQUENCE
GOBACK    L       13,4(,13)
          LM      14,12,12(13)      RESTORE REGS
          BR      14                RETURN
CHCMP0    CLC     DATE(*-*),CZROS   CHECK FOR CH ZEROES
CHCMP9    CLC     DATE(*-*),CNINES  CHECK FOR CH NINES
PD4CMP0   CLC     DATE(*-*),PZROS   CHECK FOR PD ZEROES
PD4CMP9   CLC     DATE(*-*),PNINES  CHECK FOR PD NINES
INVERT    XC      DATE(*-*),ONES    INVERT FOR DESCENDING SEQUENCE
INFODCB   DCB     DDNAME=INFO,DSORG=PS,MACRF=(GM)
SAVE61    DS      18F               REGISTER SAVE AREA
FLAGS     DC      X'00'             FLAGS
OPNDNE    EQU     X'80'             ON = OPEN DONE
CWSTART   DS      C                 START OF CENTURY WINDOW (X'yy')
WORK      DS      CL2               WORK AREA
INFORCD   DS      0CL80             INFORMATION RECORD
INCWSTRT  DS      CL2                  START OF CENTURY WINDOW (00-99)
          DS      C                    BLANK
INNUM1    DS      C                    FIRST CF NUMBER (1-9)
          DS      C                    COMMA
INSEQ1    DS      C                    FIRST CF SEQUENCE (A OR D)
          DS      C                    BLANK
INNUM2    DS      C                    SECOND CF NUMBER (1-9)
          DS      C                    COMMA
INSEQ2    DS      C                    SECOND CF SEQUENCE (A OR D)
          DS      CL70                 BLANKS
CZROS     DC      CL6'000000'       CH LOW INDICATOR
CNINES    DC      CL6'999999'       CH HIGH INDICATOR
PZROS     DC      X'0000000C'       PD LOW INDICATOR
PNINES    DC      X'0999999C'       PD HIGH INDICATOR
ONES      DC      6X'FF'            FOR INVERTING DATES
CIND19    EQU     C'1'              C'1' FOR CH 19yy INDICATOR
CIND20    EQU     C'2'              C'2' FOR CH 20yy INDICATOR
PIND19    EQU     X'10'             X'1' FOR PD 19yy INDICATOR
PIND20    EQU     X'20'             X'2' FOR PD 20yy INDICATOR
PINDHIGH  EQU     X'90'             X'9' FOR PD NINES INDICATOR
          LTORG
PARML     DSECT                     PARMLIST PASSED TO E61
          DS      3C                   NOT USED
PARMNUM   DS      X                    RELATIVE CF NUMBER
PARMPTR   DS      A                    ADDRESS OF CF
          DS      2C                   NOT USED
PARMLEN   DS      H                    CF LENGTH
CF        DSECT                     CURRENT E CF
DATE      DS      0C                   DATE - CAN BE C'yymmdd', C'yyddd',
*                                      P'yymmdd' OR P'yyddd'
CFINFO    DSECT                     INFORMATION FOR CURRENT CF
CFSEQ     DS      C                    CURRENT CF SEQUENCE (A OR D)
          END
```

Figure 5: Sample JCL for the E61EX2 Routine

```
//EX2RUN EXEC  PGM=ICEMAN
//SYSOUT DD SYSOUT=*
//EXIT DD DSN=...  library containing the E61EX2 routine
//SORTIN DD DSN=...
//SORTOUT DD DSN=...
//INFO DD *
50 2,A 3,D
/*
//SYSIN DD *
  SORT FIELDS=(1,8,CH,A,      Not a date
               12,3,BI,E,     P'yyddd'  -> 12,3,PD,A
               22,6,BI,E)     C'yymmdd' -> 22,6,CH,D
  MODS E61=(E61EX2,700,EXIT)
/*
```

SUMMARY

Special indicators such as 000000 and 999999 mixed in with real dates can pose Year 2000 sorting problems. An E61 routine can be an effective way to handle these problems. By using the information DFSORT passes as well as information you read from a dataset, you can create flexible E61 routines to handle different century windows and all the types of dates you use.

REFERENCES

DFSORT/MVS home page:
www.ibm.com/storage/dfsort/

DFSORT/MVS FTP site:
ftp://index.storsys.ibm.com/dfsort/mvs/

DFSORT Application Programming Guide, SC33-4035

"Year 2000 Compliance: New DFSORT Features" by Michael H. Carroll, ***Technical Support***, October 1997

"Sorting Out Two-Digit Year Dates" by Frank Yaeger, ***Technical Support***, December 1996 *ts*

NaSPA member Frank Yaeger is an IBM senior programmer. He has spent the last 16 of his 28 years with IBM designing and implementing functional enhancements to DFSORT such as ICETOOL, OUTFIL, and Year 2000 features. He is also responsible for the DFSORT home pages on the World Wide Web. Frank can be reached via email at fyaeger@vnet.ibm.com.

©1998 Technical Enterprises, Inc. For reprints of this document contact sales@naspa.net.