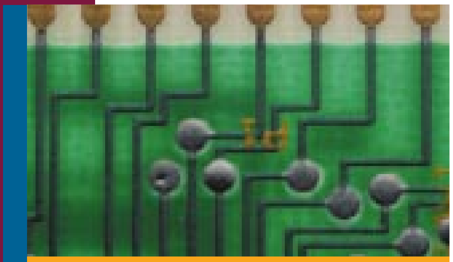

BY FRANK L. YAEGER

Using Symbols With DFSORT and ICETOOL

Release 14 of DFSORT provides a simple and flexible way to create and use symbols in your DFSORT and ICETOOL statements.

If you use DFSORT and its versatile ICETOOL utility, but hate figuring out the positions, lengths, formats and values for your fields and constants, help has arrived. Newly released DFSORT Release 14 gives you a simple and flexible way to create and use symbols in your DFSORT and ICETOOL statements. Symbols turn DFSORT's syntax into a high-level language!

This article presents an overview of how you can create and use DFSORT symbols for your own frequently used data. It also explains how to obtain IBM-created symbols and sample jobs for data associated with RACF, DFSMSrmm and DCOLLECT.

HOW SYMBOLS HELP

Symbols can help standardize your DFSORT applications and increase your productivity. You can use a symbol anywhere you would use a field or constant — in any DFSORT control statement or ICETOOL operator. DFSORT symbols can be up to 50 characters, are case sensitive and can include underscore characters. Thus, you can create meaningful, descriptive names for your symbols, such as Price_of_Item, making them easy to remember, read and understand.

A field symbol defines a field in terms of its position, length and format. A constant symbol defines a constant in terms of its literal, numeric or bit value. Once you make a symbol available, you free yourself from the sometimes tedious process of figuring out its position, length, format or value. No more confusion over offsets vs. positions and whether to add 4 for the RDW. No more recoding positions in statements for multiple DFSORT and ICETOOL jobs when you add, delete or rearrange fields in your datasets.

To illustrate how useful symbols can be, compare the DFSORT statements with and without symbols as shown in Figure 1.

IBM-CREATED SYMBOLS

To increase your productivity, IBM's DFSORT, RACF and DFSMS teams have already created DFSORT symbols and sample jobs for data associated with RACF, DFSMSrmm and DCOLLECT. For information on how you can obtain these symbols and sample jobs, visit the DFSORT home page at www.ibm.com/storage/dfsort/.

USING SYMBOLS

To use symbols with DFSORT and ICETOOL jobs, you just:

1. Create or obtain DFSORT symbol datasets that describe the data you want to process. Symbol datasets contain symbols that map the fields in your records and constants used for comparisons, titles, headings and so on. The symbols are specified in DFSORT's simple but flexible SYMNAMES statement format, which is described later in this article. You can easily add, delete or modify symbols using an editor, such as ISPF EDIT.

2. Include a SYMNAMES DD statement specifying the symbol datasets you want to use. You can use SYMNAMES to specify one symbol dataset or many concatenated symbol datasets.

3. Use the symbols from SYMNAMES in DFSORT control statements and ICETOOL operators. You can mix symbols (e.g., Last_Name) with

Figure 1: DFSORT Statements With and Without Symbols

```
** With Symbols
  SORT FIELDS=(Full_Name,A,Balance,D)
  INCLUDE COND=(Branches,EQ,West,AND,
                ERR_FLAG,NONE,Invalid,AND,
                (Balance,GT,Gift_Level#1,OR,
                Transactions,GT,High_Activity))

** Without Symbols
  SORT FIELDS=(24,40,CH,A,13,9,ZD,D)
  INCLUDE COND=(22,2,SS,EQ,C'01,95,18,22',AND,
                70,1,BI,NONE,X'FF',AND,
                (13,9,ZD,GT,250000,OR,
                68,2,PD,GT,200))
```

Figure 2: Symbol Dataset – ACCOUNTS.SYMBOLS

```
* Symbols for the fields and constants of ACCOUNTS
RDW,1,4
   Record_Length,=,2,bi
   SKIP,2
Account_Number,*,8,ch
Balance,*,9,zd
   Gift_Level#1,250000    2500.00
   Gift_Level#2,500000    5000.00

* Branch_Location and Branches are the same field with
* different formats.
Branch_Location,*,2,ch
   California,'01'
   Oregon,'95'
   Washington,'18'
   Arizona,'22'
   Florida,'16'
   Alabama,'25'
   North_Carolina,'92'
Branches,=,2,SS
   West,'01,95,18,22'
   South,'16,25,92'

* First_Name and Last_Name are subfields of Full_Name
Full_Name,*,40,ch
   Last_Name,=,20,ch
   First_Name,*,20,ch
SKIP,2        Not used
Type,*,2,ch
   Checking,'CH'
   Money_Market,'MM'
   Certificate,'CD'
Transactions,*,2,pd
   High_Activity,200
ERR_FLAG,*,1,bi
   Invalid,x'FF'
     Bad_Check,x'80'
     Bad_Credit,x'40'
     No_Funds,x'20'
* Alternate forms for No_Funds
     No_Funds_A,b'..1.....'
     No_Funds_B,B'00100000'
Other_Accounts,*   Variable information
```

regular fields (e.g., 20,5,CH) and constants (e.g., C'Yaeger').

DFSORT will read SYMNAMES and use the symbols it contains to "transform" your "statements with symbols" to "statements without symbols" by performing symbol substitution. DFSORT will then use the transformed statements (that is, the statements without symbols) as if you had specified them directly.

Typically, you would set up a symbol dataset to map the record layout (that is, the fields and constants) of each dataset you process frequently with DFSORT or ICETOOL. For example, Figure 2 shows a sample symbol dataset named ACCOUNTS.SYMBOLS that contains symbols for a variable-length (VB) dataset named ACCOUNTS. You would use the symbols from ACCOUNT.SYMBOLS in DFSORT and ICETOOL statements that process ACCOUNTS. Then, any time you changed the record layout of ACCOUNTS (e.g., by rearranging fields), you would make a corresponding change to ACCOUNTS.SYMBOLS. That way, you wouldn't have to change your jobs that use ACCOUNTS when you changed its record layout; DFSORT would use your symbols to automatically give you the correct new positions. This would save you time and help you avoid errors.

SYMNAMES AND SYMNOUT DD STATEMENTS

To use symbol processing in your DFSORT or ICETOOL jobs, just include a SYMNAMES DD statement pointing to one or more symbol datasets you want to use (concatenation is allowed). A symbol dataset must have LRECL=80 and RECFM=F or RECFM=FB. It can be a sequential dataset, a partitioned member or a DD * dataset.

To print your original SYMNAMES statements and the symbol table DFSORT builds from them, include a SYMNOUT DD statement. RECFM=FBA and LRECL=121 will be used for the SYMNOUT dataset, which would typically be SYSOUT=*. It's a good idea to include a SYMNOUT dataset until your SYMNAMES statements are debugged.

SYMNAMES STATEMENTS

A SYMNAMES statement can be a symbol statement, keyword statement, comment statement (starts with * in position 1) or blank statement (blanks in

Figure 3: DFSORT Job With Symbols

```
//DSYM JOB ...
//EXTRACT  EXEC PGM=ICEMAN
//SYSOUT   DD SYSOUT=*
//SYMNAMES DD DSN=ACCOUNTS.SYMBOLS,DISP=SHR
//SYMNOUT  DD SYSOUT=*
//SORTIN   DD DSN=ACCOUNTS,DISP=SHR
//CAEXTR1  DD DSN=CA.EXTRACT1,DISP=(NEW,CATLG,DELETE),
//  SPACE=(CYL,(20,20)),UNIT=SYSDA
//CAEXTR2  DD DSN=CA.EXTRACT2,DISP=(NEW,CATLG,DELETE),
//  SPACE=(CYL,(20,20)),UNIT=SYSDA
//SYSIN    DD *
  SORT FIELDS=(Account_Number,A,Type,A)
  INCLUDE COND=(Branch_Location,EQ,California)
  OUTFIL FNAMES=CAEXTR1,CONVERT,
    OUTREC=(Account_Number,X,Type,X,Balance,M18,X,
       Transactions,M10,X,ERR_FLAG,HEX)
  OUTFIL FNAMES=CAEXTR2,
    OUTREC=(RDW,Account_Number,First_Name,Last_Name,
       Other_Accounts)
/*
```

positions 1 through 80). ACCOUNTS.SYMBOLS contains all four types of SYMNAMES statements.

Symbol Statements

Each symbol in SYMNAMES must be described using a symbol statement. A symbol statement looks like this:

```
symbol,value <optional remark>
```

Leading blanks are allowed before the symbol, so use indentation to aid readability. In ACCOUNTS.SYMBOLS, Last_Name and First_Name are indented to show they are subfields of Full_Name, and each constant symbol is indented to show the field symbol it's associated with.

A symbol can be one to 50 characters consisting of uppercase and lowercase letters (A through Z, a through z), underscore (_), dollar sign ($), at sign (@) and number sign (#). Numbers (0 through 9) can be used for the second and subsequent characters. Symbols are treated as case-sensitive: Frank, FRANK and frank are three different symbols.

Symbol Statements For Constants

A symbol statement for a constant looks like this:

```
symbol,constant <optional remark>
```

You can use any character string, hexadecimal string, bit string or decimal number recognized in DFSORT or ICETOOL statements as the constant. The constant in a symbol statement can be specified as one of the following:

- A character string in the form 'string', C'string' or c'string'. You can use the three forms interchangeably. In ACCOUNTS.SYMBOLS, West is a character string.
- A hexadecimal string in the form X'string' or x'string'. You can use the two forms interchangeably. In ACCOUNTS.SYMBOLS, Invalid is a hexadecimal string.
- A bit string in the form B'string' or b'string'. You can use the two forms interchangeably. In ACCOUNTS.SYMBOLS, No_Funds_A and No_Funds_B are two different types of bit strings.
- A decimal number in the form n, +n or -n. You can use n and +n interchangeably. In ACCOUNTS.SYMBOLS, Gift_Level#1 is a decimal number.

Symbol Statements for Fields

A symbol statement for a field looks like this:

```
symbol,field <optional remark>
```

The field in a symbol statement can be specified as p,m,f (position, length and format), p,m (position and length) or p (position only).

Position (p) can be a number, an asterisk (*) or an equal sign (=). An asterisk (*) assigns the next position to p. It allows you to map consecutive fields in your records without having to compute their actual positions or recompute their positions when you add, remove or rearrange fields. In ACCOUNTS.SYMBOLS, Balance has an asterisk (*) to show it starts immediately after Account_Number. An asterisk (*) can also be used to create mappings of contiguous fields using concatenated symbol datasets.

An equal sign (=) assigns the previous position to p. It allows you to map subfields without specifying their actual positions. In ACCOUNTS.SYMBOLS, Last_Name has an = to show it starts at the same position as Full_Name.

Length (m) can be a number or an equal sign (=). Format (f) can be any format recognized in DFSORT or ICETOOL statements or an equal sign (=). An equal sign (=) assigns the previous length or format to m or f, respectively. You can specify p,m,f for your field symbols and then use them in DFSORT statements where p,m is required. DFSORT will cleverly substitute p,m rather than p,m,f when appropriate. For example, if you use these DFSORT statements with symbols from ACCOUNTS.SYMBOLS:

```
SORT FIELDS=(Type,A)
SUM FIELDS=(Balance)
OUTREC FIELDS=(RDW,Type,15:Balance)
```

DFSORT will transform them to the following format:

```
SORT FIELDS=(66,2,CH,A)
SUM FIELDS=(13,9,ZD)
OUTREC FIELDS=(1,4,66,2,15:13,9)
```

Note that DFSORT automatically substituted p,m,f for the SORT and SUM fields and p,m for the OUTREC fields, as required by its syntax rules.

Keyword Statements

Keyword statements can help you map the fields in your records by letting you set a starting position, skip unused bytes and align fields on specific boundaries. The available keyword statements are:

- **POSITION,q** - sets the next position and previous position to q for use with * and = in a subsequent field symbol. For example,

  ```
  POSITION,8
  Syma,*,2,FI
  ```

  assigns position 8 to Syma.

Figure 4: SYMNOUT Output

```
---- ORIGINAL STATEMENTS FROM SYMNAMES ----
* Symbols for the fields and constants of ACCOUNTS
RDW,1,4
   Record_Length,=,2,bi
   SKIP,2
Account_Number,*,8,ch
Balance,*,9,zd
   Gift_Level#1,250000    2500.00
   Gift_Level#2,500000    5000.00

* Branch_Location and Branches are the same field with
* different formats.
Branch_Location,*,2,ch
   California,'01'
   Oregon,'95'
   Washington,'18'
   Arizona,'22'
   Florida,'16'
   Alabama,'25'
   North_Carolina,'92'
Branches,=,2,SS
   West,'01,95,18,22'
   South,'16,25,92'

* First_Name and Last_Name are subfields of Full_Name
Full_Name,*,40,ch
   Last_Name,=,20,ch
   First_Name,*,20,ch
SKIP,2        Not used
Type,*,2,ch
   Checking,'CH'
   Money_Market,'MM'
   Certificate,'CD'
Transactions,*,2,pd
   High_Activity,200
ERR_FLAG,*,1,bi
   Invalid,x'FF'
     Bad_Check,x'80'
     Bad_Credit,x'40'
     No_Funds,x'20'
* Alternate forms for No_Funds
     No_Funds_A,b'..1.....'
     No_Funds_B,B'00100000'
Other_Accounts,*    Variable information

---------- SYMBOL TABLE ----------
RDW,1,4
Record_Length,1,2,BI
Account_Number,5,8,CH
Balance,13,9,ZD
Gift_Level#1,250000
Gift_Level#2,500000
Branch_Location,22,2,CH
California,C'01'
Oregon,C'95'
Washington,C'18'
Arizona,C'22'
Florida,C'16'
Alabama,C'25'
North_Carolina,C'92'
Branches,22,2,SS
West,C'01,95,18,22'
South,C'16,25,92'
Full_Name,24,40,CH
Last_Name,24,20,CH
First_Name,44,20,CH
Type,66,2,CH
Checking,C'CH'
Money_Market,C'MM'
Certificate,C'CD'
Transactions,68,2,PD
High_Activity,200
ERR_FLAG,70,1,BI
Invalid,X'FF'
Bad_Check,X'80'
Bad_Credit,X'40'
No_Funds,X'20'
No_Funds_A,B'..1.....'
No_Funds_B,B'00100000'
Other_Accounts,71
```

- **POSITION,symbol** - sets the next position and previous position to the position of the specified field symbol for use with * and = in a subsequent field symbol. POSITION,symbol can be used like the Assembler ORG instruction. For example:

```
Sym1,20,10,BI
Sym2,*,18,CH
Sym3,*
POSITION,Sym1
Sym4,*,6,ZD
Sym5,*,4,ZD
```

assigns position 20 to Sym4 (that is, Sym4 and Sym5 overlay Sym1).

Symbols can help standardize your DFSORT applications and increase your productivity.

- **SKIP,n** - skips n bytes for use with * in a subsequent field symbol.

- **ALIGN,x** - aligns the next position on a specific boundary for use with * in a subsequent field symbol. x can be H for halfword alignment, F for fullword alignment or D for double-word alignment.

SYMBOLS IN DFSORT STATEMENTS

You can use symbols in the following DFSORT control statements wherever you use constants ('string', C'string', X'string', B'string', n, +n or -n) and fields (p,m,f or p,m or p): INCLUDE, INREC, MERGE, OMIT, OUTFIL, OUTREC, SORT and SUM. Control statements in DFSPARM, SYSIN, SORTCNTL and the parameter list passed from a calling program can all use symbols.

When SYMNAMES is present, DFSORT transforms control statements with symbols to control statements without symbols, and uses the transformed statements as if you had specified them directly. DFSORT lists both the original statements and the transformed statements.

Figure 3 shows a sample DFSORT job that uses the symbols in ACCOUNTS.SYMBOLS. Since a SYMNOUT DD statement is present, DFSORT lists the original

Figure 5: ICETOOL Job With Symbols

```
//TSYM JOB ...
//ANALYZE EXEC PGM=ICETOOL
//TOOLMSG DD SYSOUT=*
//DFSMSG DD SYSOUT=*
//SYMNAMES DD DSN=ACCOUNTS.SYMBOLS,DISP=SHR
// DD *
* Local constants for titles and headings
West_Title,'Western Branch Accounts by Type'
South_Title,'Southern Branch Accounts by Type'
Type_Heading,'Type of Account'
Count_Heading,'Number of Accounts'
/*
//SYMNOUT DD SYSOUT=*
//IN      DD DSN=ACCOUNTS,DISP=SHR
//TWEST   DD DSN=&T1,DISP=(,PASS),SPACE=(CYL,(25,5)),UNIT=SYSDA
//TSOUTH  DD DSN=&T2,DISP=(,PASS),SPACE=(CYL,(25,5)),UNIT=SYSDA
//LWEST   DD SYSOUT=*
//LSOUTH  DD SYSOUT=*
//TOOLIN  DD *
*  Show number of accounts with high transaction rate
   RANGE FROM(IN) ON(Transactions) HIGHER(High_Activity)
*  Separate out records from Western and Southern branches
   COPY FROM(IN) USING(CPY1)
*  Show number of accounts of each type in *estern branches
   OCCURS FROM(TWEST) LIST(LWEST) BLANK -
     DATE TITLE(West_Title) PAGE -
     HEADER(Type_Heading) ON(Type) -
     HEADER(Count_Heading) ON(VALCNT)
*  Show number of accounts of each type in Southern branches
   OCCURS FROM(TSOUTH) LIST(LSOUTH) BLANK -
     DATE TITLE(South_Title) PAGE -
     HEADER(Type_Heading) ON(Type) -
     HEADER(Count_Heading) ON(VALCNT)
/*
//CPY1CNTL DD *
*  Extract records from Western branches
   OUTFIL FNAMES=TWEST,INCLUDE=(Branches,EQ,West)
*  Extract records from Southern branches
   OUTFIL FNAMES=TSOUTH,INCLUDE=(Branches,EQ,South)
/*
```

symbols and the symbol table as shown in Figure 4.

SYMBOLS IN ICETOOL STATEMENTS

You can use symbols in the following ICETOOL operators wherever you can use constants ('string', n, +n or -n) and fields (p,m,f or p,m): DISPLAY, OCCUR, RANGE, SELECT, STATS, UNIQUE and VERIFY. Operators in TOOLIN and in the parameter list passed from a calling program, and DFSORT control statements in xxxxCNTL and DFSPARM, can all use symbols.

When SYMNAMES is present, ICETOOL transforms ICETOOL and DFSORT statements with symbols to statements without symbols, and uses the transformed statements as if you had specified them directly. ICETOOL lists both the original statements and the transformed statements. Figure 5 shows a sample ICETOOL job that uses the symbols in ACCOUNTS.SYMBOLS. Since a SYMNOUT DD statement is present, ICETOOL lists the original symbols and the symbol table as shown in Figure 4.

SUMMARY

DFSORT Release 14 can improve your productivity significantly. By using symbols in your DFSORT and ICETOOL jobs, you'll be able to do the following:

- give descriptive, meaningful names to your fields and constants, making them easy to remember, read and understand
- free yourself from figuring out positions, lengths, formats and values
- stop worrying about offsets vs. positions and RDWs
- change your record layouts without changing your jobs
- obtain and use IBM-created symbols and sample jobs for data associated with RACF, DFSMSrmm and DCOLLECT

DFSORT symbol processing can help you standardize your DFSORT and ICETOOL applications.

REFERENCES

The *DFSORT Application Programming Guide Release 14* (SC33-4035-18) (available online from the DFSORT home page) has complete details on DFSORT symbols and all of the DFSORT control statements and ICETOOL operators.

The DFSORT home page (www.ibm.com/storage/dfsort/) has information on the new features of DFSORT R14, including symbols.

NaSPA member Frank L. 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 symbols, Year 2000 features, OUTFIL and ICETOOL. He is also responsible for the DFSORT home page on the Web. Frank can be reached via email at fyaeger@vnet.ibm.com.