SORTKEY - Sort-Key Function

SORTKEY (*character-string*)

This system function is used to convert "incorrectly sorted" characters (or combinations of characters) into other characters (or combinations of characters) that are "correctly sorted" alphabetically by the sort program or database system.

Format/length: A253

Several national languages contain characters (or combinations of characters) which are not sorted in the correct alphabetical order by a sort program or database system, because the sequence of the characters in the character set used by the computer does not always correspond to the alphabetical order of the characters.

For example, the Spanish letter "CH" would be treated by a sort program or database system as two separate letters and sorted between "CG" and "CI" - although in the Spanish alphabet it is in fact a letter in its own right and belongs between "C" and "D".

Or it may be that, contrary to your requirements, lower-case and upper-case letters are not treated equally in a sort sequence, that letters are sorted after numbers (although you may wish them to be sorted before numbers), or that special characters (for example, hyphens in double names) lead to an undesired sort sequence.

In such cases, you can use the system function SORTKEY(*character-string*). The values computed by SORTKEY are only used as sort criterion, while the original values are used for the interaction with the end-user.

You can use the SORTKEY function as an arithmetic operand in a COMPUTE statement and in a logical condition.

As *character-string* you can specify an alphanumeric constant or variable, or a single occurrence of an alphanumeric array.

When you specify the SORTKEY function in a Natural program, the user exit NATUSK*nn* will be invoked - *nn* being the current language code (that is, the current value of the system variable *LANGUAGE).

You can write this user exit in any programming language that provides a standard CALL interface. The *character-string* specified with SORTKEY will be passed to the user exit. The user exit has to be programmed so that it converts any "incorrectly sorted" characters in this string into corresponding "correctly sorted" characters. The converted character string is then used in the Natural program for further processing.

The general calling conventions for external programs are explained in the description of the CALL statement.

For details on the calling conventions for user exits, see *User Exits*.

Example:

```
DEFINE DATA LOCAL
1 CUST VIEW OF CUSTOMERFILE
  2 NAME
  2 SORTNAME
END-DEFINE
...
*LANGUAGE := 4
...
REPEAT
  INPUT NAME
  SORTNAME := SORTKEY(NAME)
  STORE CUST
  END TRANSACTION
  ...
END-REPEAT
...
READ CUST BY SORTNAME
  DISPLAY NAME
END-READ
...
```

Assume that in the above example, at repeated executions of the `INPUT` statement, the following values are entered: "Sanchez", "Sandino" and "Sancinto".

At the assignment of `SORTKEY(NAME)` to `SORTNAME`, the user exit NATUSK04 would be invoked. This user exit would have to be programmed so that it first converts all lower-case letters to upper-case, and then converts the character combination "CH" to "C*x*" - where *x* would correspond to the last character in the character set used, i.e. hexadecimally H'FF' (assuming that this last character is a non-printable character).

The "original" names (`NAME`) as well as the converted names to be used for the desired sorting (`SORTNAME`) are stored. To read the file, `SORTNAME` is used. The `DISPLAY` statement would then output the names in the correct Spanish alphabetical order:

```
Sancinto
Sanchez
Sandino
```