ReadNext...By Statement

Description

Extracts the next key from a selected list of keys (a cursor), until the list is exhausted.

Syntax

ReadNext *variable* Using *cursorvar* **By** *direction* Then | Else *statements*

Parameters

The ReadNext...By statement has the following parameters.

<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>Variable</td><td>Assigned a key from a select list. Each time ReadNext...By executes, the next available key from the list is assigned to variable.</td></tr>
<tr><td>Cursorvar</td><td>Contains a cursor variable. Cursor variables are initialized with a Select...By statement.</td></tr>
<tr><td>Direction</td><td>Indicates the direction in which the keys are read (ascending or descending) and whether the select list is terminating or non-terminating. If the list is non-terminating, the value of Status() is set to 1 after the last key has been read. The parameter direction must be an integer between 0 and 3. You can use alpha codes in the ReadNext...By command for clarity. Possible values:
<table>
<tr><th>Value</th><th>Code</th><th>Meaning</th></tr>
<tr><td>0</td><td>AT</td><td>Ascending Terminating.</td></tr>
<tr><td>1</td><td>AN</td><td>Ascending Nonterminating.</td></tr>
<tr><td>2</td><td>DT</td><td>Descending Terminating.</td></tr>
<tr><td>3</td><td>DN</td><td>Descending Nonterminating.</td></tr>
</table>
</td></tr>
<tr><td></td><td>Alphabetic codes are literal.</td></tr>
<tr><td>Then</td><td>The statement(s) following Then are executed if a row key is read successfully.</td></tr>
<tr><td>Else</td><td>The statement(s) following Else are executed if a key cannot be read from the cursor. This is the case, for example, when the list has been exhausted. The Status() function indicates the reason for the false branch, and the system variable @FILE_ERROR contains detail about the nature of the error. The Else branch is not typically used in nonterminating ReadNext...By statements, since the list is never exhausted. However, a Then or Else is still required in order for ReadNext to compile properly. Because ReadNext...By loads @RECORD, @ID and @DICT when resolving a latent list, any program using these variables for its own purposes should save them to local variables before using ReadNext...By.</td></tr>
</table>

Caution: If you exit a ReadNext loop before exhausting the select list, you must clear the select list using ClearSelect. Otherwise, your results may be unpredictable.

See also

ClearSelect, Select...By, Read

Example: Processing the CUSTOMERS table with a cursor.

```
declare function Set_FSError
open "CUSTOMERS" To CUSTOMERS_TABLE else
status = Set_FSError()
return
end

/* Select...By initializes a Cursor with a sorted list of customer records. A ReadNext and Read loop reads each
record in turn,
passing the customer records to a local subroutine (not shown) for processing. Note the use of the Cursor
keyword and the Cursorvar. */

CURSOR_NO = ""
Select "CUSTOMERS" By "STATE" setting CURSOR_NO else
  status = Set_FSError()
  Return
end
Done = 0
loop
ReadNext @ID using CURSOR_NO  else Done = 1
Until Done Do
  read @RECORD From CUSTOMERS_TABLE, @ID else
    status = Set_FSError()
    return
  end
  * processing logic here ...
  GoSub PROCESS
Repeat
return 0

PROCESS:
   /* process the row */
return
```

Example Using a Terminating Cursor

```
Reading from a table, using Cursors:
/* This program initializes a Cursor, then uses ReadNext...By to move backward in the list. */
CURSOR_NO = 0
ClearSelect CURSOR_NO
Select "CUSTOMER" By "CITY" Setting CURSOR_NO Else
  status = Set_FSError()
  Return
End
Done = 0
Loop
  ReadNext @ID Using CURSOR_NO By "DT" Else
    Done = 1
  End
Until Done Repeat
/* A select list is established using Cursor 2. The ReadNext...By direction is initialized to 2 (ascending
terminating). */
Printer On
CURSOR_NO = 2
ClearSelect CURSOR_NO
GoSub GET_SORT_LIST
Select FILE By SORT_LIST Using CURSOR_NO Else
  status = Set_FSError()
  return
end
Done = 0
RN_Mode = 2
loop
  readnext @ID Using CURSOR_NO By RN_Mode else Done = 1
until done
  read @RECORD From FILE_HANDLE, @ID then
    print @ID "L#10" : " " :
    print @RECORD<1> "L#10"
  end
repeat
Printer Off
```