V119 (String Sort) Subroutine

Description

V119 is an Assembly Language commuter module that supports eight different types of operations depending on the first parameter passed. V119 can be used to

- sort a sequence of logical records in a single dynamic array.
- perform multi-level sorting against two or more fields in the logical records.
- perform multi-level sorting with different justification and sort direction (ascending, descending) for each field.

In essence, V119 is a string sort utility. The substrings to be sorted are delimited with record marks (@RM ASCII 255), and are thus logical records. The records usually contain only the data to be sorted. Each separate sort element (logical field) is separated from the previous one with a field mark (@FM ASCII 254).

Data that is not part of the sort string (called satellite data) can be included at the end of the fields to be sorted. The most common example of satellite data is a record key.

Syntax

V119 *(FUNCTION_CODE, SORT_FILE, BYS, JUSTS, WORK, FLAG)*

16-bit OpenInsight Remarks

V119 can sort strings up to a total string length of 65,533 bytes. If the data to be sorted exceeds this length, V119 can be used to create and merge individual blocks of sort data. The blocks are temporarily stored in an operating system file.

The process of using blocks of data requires a series of discrete operations: initializing a sort file, loading it with blocks of data to be sorted, merging the blocks, sorting the blocks, extracting the sorted data, and deleting the temporary file. Each of these operations requires a separate call to V119.

You can choose any operating system file you wish to hold blocks of data. However, the system function Get.Sort.File( ) can be used to return a unique filename for this purpose. Get.Sort.File( ) uses the environment setting for the sort path, and creates a temporary sort filename that includes the station number.

Parameters

V119 accepts a total of six parameters. The first parameter is a one-letter code that determines what V119 is to do. It is the only parameter that is always required.

V119 accepts the following parameters:

<table>
<tr><th>Function Code</th><th>Description</th></tr>
<tr><td>D</td><td>causes the external sort file to be deleted.
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>SORT_FILE</td><td>Set to the drive, path, and filename of the operating system file.</td></tr>
<tr><td>BYS</td><td>Null.</td></tr>
<tr><td>JUSTS</td><td>Null.</td></tr>
</table></td></tr>
<tr><td>E</td><td>Indicates an extract operation to return satellite data from the sort array. This is useful when the sort operation has been performed with record keys as satellite data. A list of record keys is then returned.
<table>
<tr><th>Parameter</th><th>Description</th></tr>
<tr><td>SORT_FILE</td><td>Null.</td></tr>
<tr><td>BYS</td><td>Null.</td></tr>
<tr><td>JUSTS</td><td>Null.</td></tr>
<tr><td>WORK</td><td>A dynamic array in the same format as in the sort operation. WORK returns the last fields of each record separated by field marks.</td></tr>
<tr><td>FLAG</td><td>Null.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>I</td><td>Initializes an operating system file that is used to hold sort data temporarily while sorting.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>Set to the drive, path, and filename of the operating system file.</td></tr><tr><td>BYS</td><td>Null.</td></tr><tr><td>JUSTS</td><td>Null.</td></tr><tr><td>WORK</td><td>Null.</td></tr><tr><td>FLAG</td><td>Zero if there was a problem in file i/o and non-zero if the operation was successful.</td></tr></table></td></tr>
<tr><td>L</td><td>Indicates a list operation. This operation is similar to the Extract operation. This operation gets the keys back from the merged external sort file in blocks of 32K.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>The drive, path, and filename of the external sort file.</td></tr><tr><td>BYS</td><td>Null.</td></tr><tr><td>JUSTS</td><td>Null.</td></tr><tr><td>WORK</td><td>Passed null. Returns a 32K block of field mark delimited of keys.</td></tr><tr><td>FLAG</td><td>Zero if there was a problem in file i/o and non-zero if the operation was successful.</td></tr></table></td></tr>
<tr><td>M</td><td>Indicates a merge operation. This operation is used to fully sort the external sort file after all of the sorted blocks of data have been written to it.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>The drive, path, and filename of the external sort file.</td></tr><tr><td>BYS</td><td>The same value as when the data was sorted during the Sort operation.</td></tr><tr><td>JUSTS</td><td>The same value as when the data was sorted during the Sort operation.</td></tr><tr><td>WORK</td><td>Null.</td></tr><tr><td>FLAG</td><td>Zero if there was a problem in file i/o and non-zero if the operation was successful.</td></tr></table>
Note: This operation immediately doubles the size of the operating system file. If disk space is tight, this is the most vulnerable operation.</td></tr>
<tr><td>S</td><td>Sorts an array of records.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>Null.</td></tr><tr><td>BYS</td><td>A or D for each sort field. A represents an ascending sort for the field and D represents a descending sort. For example, if BYS equals ADA, V119 does an ascending sort on the first field in WORK, descending on the second field, and ascending on the third field even though there may be satellite data.</td></tr><tr><td>JUSTS</td><td>Works the same way as BYS but refers to the justification of each sort field and consequently whether an alpha or numeric sort is performed. L and R are the only valid characters representing left and right justification, respectively.</td></tr><tr><td>WORK</td><td>A dynamic array of data to be sorted. Each piece of data to sort is separated by a record mark (ASCII 255). A final record mark is required, but not a beginning one.
If a multiple level sort is to be performed, fields are defined within each record separated by field marks (ASCII 254). The first field is sorted first (the primary sort), the second field is sorted next, and so forth.

There may be fields in the record that are not sorted (satellite data), because only as many fields will be sorted as there are characters in the BYS or JUSTS parameters. All sort fields must be contiguous at the beginning of each record. All satellite data must be contiguous at the end of each record. There is no way to skip the sorting of a field.

WORK returns in the same format as it was passed, but the records are in sorted order</td></tr><tr><td>Flag</td><td>Null</td></tr></table></td></tr>
</table>

<table>
<tr><td>V</td><td>Denotes a value operation. This operation is like the List operation in that it returns data in 32K blocks from an external sort file after it has been merged. However, the data returned is like that of a sort operation rather than an extract operation.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>The drive, path, and filename of the external sort file.</td></tr><tr><td>BYS</td><td>Null.</td></tr><tr><td>JUSTS</td><td>Null.</td></tr><tr><td>WORK</td><td>Null when passed. Returns a 32K block of sorted records by whatever criteria were specified when the external sort file was merged.</td></tr><tr><td>FLAG</td><td>Returns zero if there was a problem in file i/o and non-zero if the operation was successful.</td></tr></table></td></tr>
<tr><td>W</td><td>Indicates a write operation. This is used to write a block of sort data to the temporary sort file.

<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>SORT_FILE</td><td>The drive, path, and filename of the operating system file.</td></tr><tr><td>BYS</td><td>Null.</td></tr><tr><td>JUSTS</td><td>Null.</td></tr><tr><td>WORK</td><td>A dynamic array to write. This is most likely the return value of WORK from a sort operation. Thus, the external sort file becomes a series of sorted blocks of data.</td></tr><tr><td>FLAG</td><td>Returns zero if there was a problem in file i/o and non-zero if the operation was successful.</td></tr></table></td></tr>
</table>

See Also

Get.Sort.File()

Example: Sorting by One Column

```
/* Sort by Customer Number (Right Justified), Ascending , then Descending */

declare subroutine V119
/* Customer Number, Region, Customer Name, Total Sales */
Record1 = "42" : @fm : "West" : @fm : "Acme Corporation"    : @fm : 5000
Record2 = "1"  : @fm : "East" : @fm : "Zeta Corporation"    : @fm : 200
Record3 = "3"  : @fm : "East" : @fm : "Midland Corporation" : @fm : 3500
Record4 = "2"  : @fm : "West" : @fm : "Orland Corporation"  : @fm : 300
SortData = Record1 : @rm : Record2 : @rm : Record3 : @rm : Record4 : @rm

/*  ascending sort  */
Bys   = 'A'
Justs = 'R'
V119('S', '', Bys, Justs, SortData, '')

/*  descending sort */
Bys  =  'D'
V119('S', '', Bys, Justs, SortData, '')
```

Example: Sorting by Two Columns

```
/* Sort by Region (Ascending-Left), then Total Sales (Descending-Right) */

declare subroutine V119
/* Region, Total Sales, Customer Number, Customer Name */
Record1 = "West" : @fm : 5000 : @fm : "42"     : @fm : "Acme Corporation"
Record2 = "East" : @fm :  200 : @fm : "1"      : @fm : "Zeta Corporation"
Record3 = "East" : @fm : 3500 : @fm : "3"      : @fm : "Midland Corporation"
Record4 = "West" : @fm :  300 : @fm : "2"      : @fm : "Orland Corporation"
SortData = Record1 : @rm : Record2 : @rm : Record3 : @rm : Record4 : @rm

Bys   = 'AD'
Justs = 'LR'
V119('S', '', Bys, Justs, SortData, '')
```

Example: External Sort Logic

```
/*   External sort  */
declare subroutine V119 , msg
declare function Get.Sort.File
$insert logical

/* Same data as Sorting by Two Columns Example */
Record1 = "West" : @fm : 5000 : @fm : "42"     : @fm : "Acme Corporation"
Record2 = "East" : @fm :  200 : @fm : "1"      : @fm : "Zeta Corporation"
Record3 = "East" : @fm : 3500 : @fm : "3"      : @fm : "Midland Corporation"
Record4 = "West" : @fm :  300 : @fm : "2"      : @fm : "Orland Corporation"
SortData = Record1 : @rm : Record2 : @rm : Record3 : @rm : Record4 : @rm
Bys   = 'AD'
Justs = 'LR'

flag = ''
/*  get a unique sort file name and initialize it */
SortFile = Get.Sort.File()
V119('I', SortFile, '', '', '', flag)
IF flag else
   TEXT = 'Error initializing sort file.'
   call msg(@window, TEXT)
  return FALSE$
END

/*  write the data to SortFile.  May be called multiple times
if data > 32K. Need to write in chunks.      */
V119('W', SortFile , '', '', SortData, flag)
if flag else
   TEXT = 'Error writing to sort file.'
   msg(@window, TEXT)
   gosub delete_sortfile
   return FALSE$
end

/*  Use the 'M' argument to sort the external file, after data has been written. */
V119('M', SortFile , Bys , Justs , '', flag)

/*  Use the 'L' argument to retrieve a list of sorted keys.
Use the 'E' argument to retrieve the last field, in sorted order.
*/
V119('L', SortFile , Bys , Justs , SortData , flag)
V119('E', SortFile , Bys , Justs , SortData , flag)

/*  Don't forget to delete the sort file when done!  */
Delete_SortFile:
V119('D', SortFile , '', '', '', flag)
IF FLAG ELSE
  TEXT = 'Error deleting sort file ': SORT_FILE
  call MSG(@window, TEXT)
END
```