BSORT

BSORT is a high speed sort utility for sorting BASIC arrays. Any type of array (integer, single/double precision or string) may be sorted. Sorts can be performed on one and two dimensional arrays. The following syntax is used to initiate a sort. **Note:** "Integer Numbers" refers to integer variables or constants.

SYSTEM"RUN BSORT NUM%,*IND%,PSA(x),parm,parm,...,parm"
SYSTEM"RUN BSORT $STRVAR$"

NUM%	Number of elements to sort (an integer number).
*IND%(x)	Optional single dimension integer array. If not used, re-ordering of elements will occur in the array being sorted. If used, the sort will generate an index array containing element numbers of the sorted array, and no re-ordering of "sorted arrays" will occur.
PSA(x)	Primary sort array. An optional <+> or <-> may precede the array name to indicate the direction (ascending or descending order) of the sort. If not specified, <+> is assumed. A declaration tag (!,#,$,%) must be used for any array specified. A subscript must be specified, representing the first element number to be sorted. It must be an integer number.

Optional parameters are as follows:

SSA(x)	Secondary sort array. If used, a <+> or <-> must precede the array name. The sort key used will include corresponding information from the primary and secondary arrays. Any re-ordering of the primary array will cause a corresponding re-ordering of the secondary array. More than one may be used. A subscript is required if the secondary array is two dimensional.
TA	Tag array. Any re-ordering in the primary array will cause a corresponding re-ordering in a tag array. A tag array cannot be preceded by a <+> or <->, and may only appear after all secondary array definitions. More than one may be used.
(s,n)	Mid-string information. Valid only with STRING arrays. If specified, it must immediately follow the array information, and cannot be used with tag arrays. If specified, the sort key will begin at position s in the string, for n characters, where s and n are integer numbers.
$STRVAR$	Optional non-array string variable containing the sort parameters. Must be used if the length of the sort command (i.e. the number of chars. within the quote marks) exceeds 79.

BSORT can be used to perform many different sorting tasks from within a BASIC program. Only variables and arrays that have been "established" can be used; BSORT cannot allocate memory for variables or arrays. If an un-dimensioned array is used in a sort command, an error will be generated. The following examples illustrate the many different types of sorts which can be performed.

Sorting a Single Dimension Array

Sorting a single dimension array is the simplest type of sort. An integer, single/double precision, or string array may be sorted. In order to sort a one dimension array, two parameters must be passed to BSORT: the number of elements to be sorted, and the starting position in the primary sort array. As an example, assume that the following string array exists in memory:

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
SMITH	JONES	BROWN	WILLIAMS	JOHNSON	GREEN

To sort the array, this command would be used, with the results shown below.

SYSTEM"RUN BSORT 6,+A$(1)"

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
BROWN	GREEN	JOHNSON	JONES	SMITH	WILLIAMS

In the example, the number of elements to be sorted was specified as six, with the sort being performed on array A$ (the primary sort array), starting at element 1.

In this type of sort, a re-ordering of elements takes place. The sequence is in ascending order, so that the value of A$(1)<A$(2)<A$(3) etc. The plus sign <+> in front of the primary sort array indicates that the direction of the sort is to be in ascending order. In this case, the plus sign is optional; if the primary sort array appears without a sign preceding it, the sort will be in ascending order.

Sorting may also be performed so that the re-ordering is in descending order. This is accomplished in the sort command by using a minus sign <-> in front of the primary sort array. Thus, after executing the sort command:

SYSTEM"RUN BSORT 6,-A$(1)"

the value of A$(1) would be "WILLIAMS"; the value of A$(6) would be "BROWN".

Note that in the above examples, the number of elements to be sorted (6) and the starting array position (1) were specified as integer constants. Any integer constant which needs to be passed to BSORT can be specified as a simple (non-array) variable. The only restriction in using a variable as a value passed to the sort utility is that it must be an integer type, with the type declaration tag (%) explicitly present. DEF statements (e.g. DEF INT) may be used; but the variable as used in the sort command must have a type declaration tag.

Arrays used in BSORT must have a type declaration tag. In the above examples, an error would occur if the commands DEF STR A:DIM A(6) were issued and the following sort invoked, since the <$> declaration tag was not present.

SYSTEM"RUN BSORT 6,A(1)"

When sorting an array, any part of the array may be sorted for any number of elements. Assuming from the above examples that A$ is dimensioned to have 7 elements {A$(0) through A$(6)}, the following sort can be executed.

```
NM%=4:PO%=2
SYSTEM"RUN BSORT NM%,A$(PO%)"
```

This sequence of commands would cause elements 2 through 5 to be sorted in ascending order, and would leave elements 0,1 and 6 untouched.

An error will be generated if sorting is forced beyond the dimensioned length of the array. In the above example, if PO% is 2, an error will be generated if NM% is assigned a value greater than 5.

Using Secondary Sort Arrays

More than one array may be used to determine the results of a sort operation. Secondary sort arrays can be specified after the primary sort array, and will be included in the sort (i.e. they will aid in determining the direction of the sort and will be re-ordered in conjunction with the primary sort array).

For example, assume that the following arrays are currently active in memory.

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
SMITH	JONES	JONES	WILLIAMS	JOHNSON	JONES

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)
SAMMY	BILLY	BETTY	RICHARD	CHARLES	BOBBY

The array A$ represents a list of last names, while the array F$ contains the corresponding first name. It is desired to create a sorted list of these names in ascending order, where the first name is used to determine the sorted order when the last names are the same. The following sort command may be used to accomplish this task, with the results shown below.

```
SYSTEM"RUN BSORT 6,A$(1),+F$
```

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
JOHNSON	JONES	JONES	JONES	SMITH	WILLIAMS

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)
CHARLES	BETTY	BILLY	BOBBY	SAMMY	RICHARD

The array F$ is a secondary sort array. It is used in the sorting process to determine the sorted order when a direct match is found in the primary array. When a secondary sort array is specified, a direct correlation between elements in the primary array is assumed. Any re-ordering which occurs in the primary array will also occur in the secondary array. Thus in this example, the first names were carried along in the sort with the last names, and any exact matches on the last names caused the first names to be sorted.

There are some points that need to be made with respect to syntax and usage of secondary sort arrays. When dealing with a single dimension secondary array, it must be separated from the primary array with a comma. Additionally, no subscript number is required. Any re-ordering which occurs will be done according to the element number in the primary array (i.e. element I in the primary array corresponds to element I in the secondary array). Furthermore, secondary arrays must be dimensioned as high as the largest sorted element number in the primary array. For example, if a primary array is dimensioned to have 50 elements (0-49) and a secondary array is dimensioned to have 10 elements (0-9), a sort using both arrays could only be performed up to and including element nine. An error will be generated if the sort should go beyond the highest allowable element number of either the primary or secondary array.

Unlike primary arrays, use of a direction sign (+ or -) is mandatory when specifying a secondary array. The direction of the sort in a secondary array does not have to match that of the primary array. Using the above arrays (A$ and F$), the following sort command would produce the results shown below.

SYSTEM"RUN BSORT 6,+A$(1),-F$"

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
JOHNSON	JONES	JONES	JONES	SMITH	WILLIAMS

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)
CHARLES	BOBBY	BILLY	BETTY	SAMMY	RICHARD

Note that the directioning of the secondary sort array (in descending order) did not affect the re-ordering of the primary array (ascending order). However, any exact matches in the primary array caused the secondary array (first name array) to be sorted in descending order.

Using Multiple Secondary Arrays

The concept of using more than one secondary array does not differ greatly from using just one secondary array. In terms of syntax, commas must separate subsequent secondary arrays. The same restrictions also apply when more than one secondary array is used (i.e. the mandatory direction sign and the dimensioned lengths of the arrays).

The important point to note is that the order in which the arrays are entered on the sort command line may have an effect on the results of the sort. That is, the first secondary array specified will be the first array used to determine the results of the sort. As an example, examine the three arrays that are shown on the next page.

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)	A$(7)
SMITH	BROWN	JONES	JONES	JONES	JONES	JONES

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)	F$(7)
SAMMY	ROBBY	JOHN	JAKE	JOHN	HERB	HERM

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)
1001	1002	1003	1004	1005	1006	1007

Array A$ contains last names, Array F$ contains first names and Array I% contains ID numbers. Consider the results of the following sort command (shown below).

SYSTEM"RUN BSORT 7,A$(1),+F$,-I%"

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)	A$(7)
BROWN	JONES	JONES	JONES	JONES	JONES	SMITH

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)	F$(7)
ROBBY	HERB	HERM	JAKE	JOHN	JOHN	SAMMY

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)
1002	1006	1007	1004	1005	1003	1001

The primary sort occurred on the last name (ascending order). If an exact match occurred in the last name, the first name was used to determine the order (ascending order). If two people had identical first and last names, the ID number was then used, and sorted in descending order.

If the secondary arrays were "switched" on the command line, the results obtained would be quite different. Observe this sort command and associated results.

SYSTEM"RUN BSORT 7,A$(1),-I%,+F$"

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)	A$(7)
BROWN	JONES	JONES	JONES	JONES	JONES	SMITH

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)	F$(7)
ROBBY	HERM	HERB	JOHN	JAKE	JOHN	SAMMY

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)
1002	1007	1006	1005	1004	1003	1001

Note that the last names did sort in ascending order. However, since the I% array appears immediately after the primary array, any identical match found on the last name caused the next sort criteria to be taken from the I% array, with the results being determined in descending order.

Using Tag Arrays

In addition to using secondary arrays, Tag Arrays may be specified on a sort command line. They are similar to Secondary sort arrays, with the exception that the information contained in them has no bearing on the results of the sort. If a tag array is used, any re-ordering which occurs in the primary sort array will also occur in a tag array.

Tag arrays are distinguished from secondary arrays by the lack of a direction sign. If an array (other than the primary array) has no direction sign, it is taken to be a tag array. If both tag and secondary arrays are to be used, ALL secondary arrays must be defined on the sort command line prior to any tag array definitions. Subsequent tag arrays must be separated by commas, and no subscript number is required.

Consider a last name - first name example, where the following arrays have been defined in memory.

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
JONES	JONES	JONES	WILLIAMS	JOHNSON	JONES

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)
ROBIN	BILLY	BETTY	RICHARD	CHARLES	BOBBY

A typical sort command which makes use of F$ as a tag array could be represented by the following, with the results shown below.

```
SYSTEM"RUN BSORT 6,A$(1),F$"
```

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)	A$(6)
JOHNSON	JONES	JONES	JONES	JONES	WILLIAMS

F$(1)	F$(2)	F$(3)	F$(4)	F$(5)	F$(6)
CHARLES	ROBIN	BILLY	BOBBY	BETTY	RICHARD

Note that the last names are sorted in correct ascending order. However, since F$ was used as a tag array, it did not affect the results of the sort, and only a re-ordering of the data occurred. The order of the items shown in the F$ array is related to the re-ordering of the A$ array. Whenever exact data matches occur (as is the case with the name JONES), re-ordering is done in a random fashion. If additional sort information is required, it should be specified as a secondary sort array. If information is only to be "moved along with" sorted information, it may be specified as a tag array.

MID$ Sorting

When sorting string arrays, an optional MID$ (mid string) parameter may be specified with primary and/or secondary arrays. This will allow sorting to be done on a string array using only a specified part of the string. If re-ordering is performed, the entire string element will be "moved".

As an example, consider the following two string arrays.

L$(0)	L$(1)	L$(2)	L$(3)	L$(4)
D. BROWN	R. SMITH	T. JONES	R. SMITH	P. JONES

F$(0)	F$(1)	F$(2)	F$(3)	F$(4)
BR, DALE	SM, ROB	JO, TERRY	SM, RICH	JO, PETE

Array L$ contains a first name initial, followed by a period, a space and a last name. Array F$ contains the first two characters of the last name, followed by a comma, a space, and the first name. With a sort to be done in the order of last name - first name, this sort command could be used, with the results shown below.

SYSTEM"RUN BSORT 5,L$(0)(4,7),+F$(5,6)

L$(0)	L$(1)	L$(2)	L$(3)	L$(4)
D. BROWN	P. JONES	T. JONES	R. SMITH	R. SMITH

F$(0)	F$(1)	F$(2)	F$(3)	F$(4)
BR, DALE	JO, PETE	JO, TERRY	SM, RICH	SM, ROB

In this sort command, the primary sort array is the L$ array, while F$ is a secondary array. Both arrays are sorted in ascending order.

The MID$ information immediately follows the subscript position for the primary array. It is enclosed within parentheses, and consists of two integer numbers. The first number specifies the position in the string where the sort criteria begins. Here, the strings in the L$ array are to be sorted starting with the fourth character (i.e. the first character of the last name). The second number tells the sort utility the number of characters to include in the sort from the starting position. Here, seven characters of each string (starting at position four of the string) will comprise the sort key for the primary sort array.

Similarly, MID$ information has been supplied with the secondary sort array. Using the F$ array, the sort key will begin at position 5 (i.e. the first character of the first name) in each element of the array, and extend for 6 characters into each string. Thus, the two arrays are sorted in the order of last name - first name.

Several points need to be made with respect to MID$ sort information. It must always immediately follow the last piece of information associated with the array (i.e. no comma separator is used). When sorting single dimension arrays, this will come after the closing parenthesis of the subscript number for the primary array, and after the declaration tag of the secondary array.

BSORT will NOT check to see if the MID$ values are valid for any string, with the exception that they must not exceed 255. If the starting MID$ position exceeds the entire length of the string in question, a "null" value will be used for that particular element of the array. If the starting MID$ position is within the string, but the number of characters to use for sort criteria is more than what is remaining in the string, only the remaining characters will be used.

For example, A$(1)="HI ", A$(2)="BYE" and A$(3)="THIS IS THE END". The following sort commands will produce the results shown below.

Sort Command	**Ordering of Elements**
1. SYSTEM"RUN BSORT 3,A$(1)(1,3)"	2,1,3
2. SYSTEM"RUN BSORT 3,A$(1)(2,4)"	3,1,2
3. SYSTEM"RUN BSORT 3,A$(I)(3,2)"	1,2,3

In example 1, the first 3 characters of each string are used in the sort to determine the results. In example 2, the second through fifth characters of each string are used. In example 3, characters three and four of each string are used. Since the first array position only has a length of two characters, its sort value is "null", and so it was sorted "first" (in ascending order).

Generating an Index Array

The previous examples illustrated methods by which arrays were sorted into either ascending or descending order. With those sorts, the array data was re-ordered, so that physical access of the array (by ascending/descending element numbers) was required-to see the-sorted results. In some cases (such as reading data into an array from a random access file) it may not be desirable to actually re-order an array when "sorting". Thus, BSORT may also be used to generate index arrays. BSORT will initialize the index array to contain the element numbers of the array to sort. The sort will re-order the index array, so that the values in the index array will form a list of pointers to the "sorted" elements of the primary array. For example, assume that the following arrays are currently in memory:

P$(1)	P$(2)	P$(3)	P$(4)	P$(5)	P$(6)	P$(7)
WILLIAMS	SMITH	JONES	BROWN	GREEN	JOHNSON	RICH

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)
0	0	0	0	0	0	0

The sort command listed below could be used to create the index array I%.

SYSTEM"RUN BSORT 7,*I%(1),P$(1)"

P$(1)	P$(2)	P$(3)	P$(4)	P$(5)	P$(6)	P$(7)
WILLIAMS	SMITH	JONES	BROWN	GREEN	JOHNSON	RICH

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)
4	5	6	3	7	2	1

Notice that the sort command did not alter the primary sort array (P$). However, the values in the index array (I%) changed to reflect proper access order of the primary array. Since I%(1) has a value of 4, the fourth element of the P$ array is the first element to access if ascending sorted order is desired. It is now simple to print the P$ array contents in sorted order, using I% as an index.

```
FOR L%=1 TO 7                        FOR L%=1 TO 7
PRINT P$(I%(L%))        or           M%=I%(L%):PRINT P$(M%)
NEXT L%                              NEXT L%
```

When using an index sort, the index array must immediately follow the number of items to sort on the command line, and must be preceded by an asterisk <*> which indicates that an index array is to be generated. The index array MUST be a one dimension integer type with the declaration tag used explicitly. A subscript number must be used with the index array; it will indicate the starting position of the indexed information. This subscript may be either an integer constant or a simple integer variable. Finally, the index array must be dimensioned large enough to contain all index values generated - the number of items sorted. Failure to adhere to these guidelines will more than likely generate an error.

Regarding the subscript number used with the index array, in most cases it will be parallel to the subscript number specified in the sorted array. However, it is not mandatory that these two subscript numbers be the same. As an example, assume that the following integer array exists in memory.

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)	I%(8)	I%(9)	I%(10)
100	200	300	400	500	600	700	800	900	1000

Using this as an index array, the following sort command on the P$ array (see previous example) will produce the results shown.

SYSTEM"RUN BSORT 4,*I%(6),P$(2)"

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)	I%(6)	I%(7)	I%(8)	I%(9)	I%(10)
100	200	300	400	500	4	5	3	2	1000

This will sort four elements of the P$ array (elements 2 through 5), and store the index information in the I% array, starting at element 6. Note that elements 1-5 and 10 in the index array were unaffected by the sort. The numbers stored in the index array correspond to the element numbers that were sorted (2 through 5).

If the I% array was initially dimensioned to contain 11 elements (0-10), the following sort command would cause an error:

SYSTEM"RUN BSORT 4,*I%(8),P$(2)

The error would be caused by trying to store index information beyond the end of the index array (i.e. Element #11 of the I% array does not exist).

Indexed sorts may be performed using all of the previously defined sort parameters (i.e. MID$ and secondary arrays). The syntax for such sorting would remain the same, with the exception of the index array being specified in the sort command. No re-ordering will be done on any array used in an indexed sort. Thus, it would be meaningless to use tag arrays in an indexed sort; although no error would be generated, a tag array will not be affected by an indexed sort.

Sorting Two-dimensional Arrays

BSORT supports the use of two dimensional arrays as any type of array (primary, secondary, tag) in a sort command. This section will discuss several variations of using two dimensional arrays.

Throughout the documentation, examples have been given of various sort procedures using single dimension arrays. The array illustrations have always been depicted in a "horizontal" fashion, representing one row of information with multiple columns. Sorting a one dimension array implies that a row of the array (in this case the only row of the array) be used as the key information, with individual columns being re-ordered (or indexed) to satisfy the requirements of the sort.

This same concept can be carried over to two dimensional arrays. An individual row of the array is specified, from which the key (sort) information is retrieved. Additionally, a starting column number is specified, and the number of elements to be sorted represents the number of columns involved in the sort. If re-ordering is required, an entire column of data is "moved".

As an example, assume that this array (A$) has been established in memory.

	1	2	3	4	5
1	DALE	DAN	DON	DICK	DOCK
2	BROWN	JONES	SMITH	GREEN	PETERS
3	25	34	19	53	42
4	BOSTON	BUTTE	BALT	PHIL	PITT
5	03021	78654	23376	19769	16511
6	MA	MT	MD	PA	PA
7	REP	REP	CLIENT	ADV	STOCK

If it was desired to sort this array by last name in ascending order, the following sort command could be entered, with the results shown below.

SYSTEM"RUN BSORT 5,A$(2,1)"

	1	2	3	4	5
1	DALE	DICK	DAN	DOCK	DON
2	BROWN	GREEN	JONES	PETERS	SMITH
3	25	53	34	42	19
4	BOSTON	PHIL	BUTTE	PITT	BALT
5	03021	19769	78654	16511	23376
6	MA	PA	MT	PA	MD
7	REP	ADV	REP	STOCK	CLIENT

Several points can be drawn from this example. The total number of items to sort is 5. Row 2 is designated as containing the information to sort. The sort will begin at column 1 (in row 2) and continue for a total of 5 columns. If a re-ordering is to take place, all information in the given column is "moved" (essentially, the two columns involved in the re-ordering are "swapped").

If the A$ array were used as it appeared initially (see Example 1), and the following sort command was issued:

SYSTEM"RUN BSORT 2,A$(5,4)"

A swap of columns 4 and 5 would be performed. This sort would use information in row 5 as the key. The sort would begin at column 4, and include 2 columns (i.e. columns 4 and 5). Since 16511 is less than 19769, a re-ordering would occur.

Assume once more that the A$ array exists in memory as shown in Example 1. It is desired to generate an index array (I%), where the information in row 3 is sorted in descending order. The following sort command would accomplish this, with the results shown below.

SYSTEM"RUN BSORT 5,*I%(l),-A$(3,1)"

I%(1)	I%(2)	I%(3)	I%(4)	I%(5)
4	5	2	1	3

Note that when indexing a two dimensional array, the column position of the sorted array is stored in the index array. The sorted array remains unchanged.

Using Two Dimensional Secondary and Tag Arrays

The concept behind sorting two dimensional arrays carries over to the use of two dimensional secondary and tag arrays. In both instances, the number of rows is insignificant. The number of columns in either a secondary or tag array must be as large (or greater than) the highest column number to be sorted in the primary array.

In the case of a tag-array, no subscript is required. Re-ordering of columns in the tag array will correspond to those re-ordered in the primary array. The entire column will be "moved", regardless of the number of rows in the array.

The same re-ordering rules apply to two dimensional secondary arrays. However, a subscript must be included with the secondary array. The subscript will be the row number from which key information is to be taken.

Let us assume that the following arrays exist in memory.

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)
BROWN	ADAMS	BROWN	ADAMS	BROWN

ARRAY B$

	1	2	3	4	5
1	PRES	VP	MGR	SALES	DIST
2	25	53	34	42	19
3	DALE	DOCK	DAN	DICK	DON

The A$ array is to be the primary array, and row three of the B$ array will be used as secondary sort information. The following sort command would yield these results (i.e. primary sort by last name, secondary sort by first name), with the sorted arrays being shown on the next page.

SYSTEM"RUN BSORT 5,A$(1),+B$(3)

A$(1)	A$(2)	A$(3)	A$(4)	A$(5)
ADAMS	ADAMS	BROWN	BROWN	BROWN

ARRAY B$

	1	2	3	4	5
1	SALES	VP	PRES	MGR	DIST
2	42	53	25	34	19
3	DICK	DOCK	DALE	DAN	DON

Note that any re-ordering which occurred in the primary array forced the entire (corresponding) column in the B$ array to be moved.

The same re-ordering will occur if a two dimensional tag array is used. A subscript is not required. Entire columns (regardless of the number of rows) will be re-ordered according to the corresponding re-ordering of the primary array.

When using two dimensional secondary arrays, the same array can be used more than once in a sort command, provided that the row specified is different in each case. As a matter of fact, if the primary array is two dimensional, a row different than the primary sort row may be specified as a secondary sort array. In the case of our first example dealing with two dimensional arrays, if it was desired to obtain a sort on this array primarily by last name (row 2) and secondarily by first name (row 1), the following sort command could be used:

SYSTEM"RUN BSORT 5,A$(2,1),+A$(1)"

This would have the affect of using row 2 of the A$ array as the primary sort information, while using row 1 of the same array as secondary sort information.

As a final point, it is permissible to use the MID$ function when dealing with two dimensional secondary arrays. As is the case with a one dimension primary array, the MID$ information would immediately follow the row subscript of the secondary array. The following example illustrates the syntax that would be used:

SYSTEM"RUN BSORT 10,XX%(2,5),+C$(3)(19,8)"

Here, XX% is the primary array. The sort will be on row 2 of this array, starting at column 5. It will extend for 10 columns (up to and including column 14). C$ will be used as the secondary array. Columns 5-14 of row 3 will be used. Within each of these elements, a MID$ will be performed, so that the string used in the sort will begin at position 19 of each element, and extend for 8 characters.

Using a Variable to Pass the Sort Command

Depending on the number of parameters specified, the length of an actual sort command may become quite large. A limitation does exist in that the total length of a SYSTEM command cannot exceed 79. For this reason, BSORT allows for the passing of sort parameters in a simple string variable. For example, this command utilizes information contained in PARM$ as the parameters to use for the sort.

```
PARM$ = "10,*II%(1),-AA$(4,1)(15,20),+AC#(3),-SD$(7)(13,8)"
SYSTEM "RUN BSORT $PARM"
```

In the command which initiates the sort, a <$> must precede the string variable containing the sort parameters. In using this type of sort command, the only limitation is that the length of the string cannot exceed 255 characters.