BREXX/370 V2R5M2 Array Functions

The BREXX Array functions are an implementation outside the REXX standard. They allow more direct access to the items of an array compared to compound variables (stems). The definition overhead is also smaller, which allows larger arrays as with stems. For performance reasons, the internal checking of boundaries, limits and content is kept at a basic level, if exceeded the REXX script will most likely end with an 0C4.

I. String Array Functions

Why String Arrays? There is a performance and storage overhead with stems, the stem name must be located in a binary tree before the contents can be read. The allocation for content also contains a reserve in case a new version is a bit longer to avoid reallocation of the memory.

String Arrays have a pointer array addressing each content directly, adding a new item is therefore 2 times faster than in a stem, reading about 5 times faster. Another benefit is you can easily add low-level functions (written in C) to work on the arrays directly. SQSORT, SHSORT, SSEARCH, SSELECT are some examples.

SCREATE(size)

Creates a Source Array, returned is the Source Array Number, which must be used in various Source Array functions. The size refers to the maximum number of entries of the array. Exceeding the maximum might lead to an 0C4 or other abends.
Depending on virtual storage availability, you can have up to 32 different arrays.
For example, see SGET.

Returns the allocated **array-number** which can be used in subsequent array functions.

SSET(array-number,[item-index],string-value)

Sets a certain element of the array with a string value. The item index must not exceed the maximum size defined in the SCREATE function. If the item-index is not specified, the entry is added at the end of the array.
The item index must not exceed the maximum size defined in the SCREATE function. To minimize the overhead there is no checking of the limits in place. Exceeding it will cause an 0C4.

For example, see SGET.

SGET(array-number,item-index,[offset])

Gets (returns) an element of the array as a string value. If an offset is defined the returned value starts at it.
The item index must not exceed the maximum size defined in the SCREATE function. To minimize the overhead there is no checking of the limits in place. Exceeding it will cause an 0C4.

Example

```
smax=15
s1=screate(smax)
do i=1 to smax
   call sset(s1,i,right(i,3,'0')'. Record')
end
do i=1 to smax
   say sget(s1,i)
```

```
end
```

Result

```
001. Record
002. Record
003. Record
004. Record
005. Record
006. Record
007. Record
008. Record
009. Record
010. Record
011. Record
012. Record
013. Record
014. Record
015. Record
```

SSWAP(array-number,item-number-1, item-number-2)

Swaps the position of 2 elements in the array. As only pointers are moved a very fast function.

SCLC(array-number,item-number-1, item-number-2)

Compares 2 elements of the array. SCLC is much faster than loading both items and comparing it to the REXX level.

returns

<0	if item-1 < item-2
0	if item-1 < item-2
>0	if item-1 > item-2

Example

```
smax=15
s1=screate(smax)
do i=1 to smax
   call sset(s1,i,right(i,3,'0')'. Record')
end
do i=1 to smax
   say "Compare item "i" and 8, result: "sclc(s1,i,8)
end
```

Result

```
Compare item 1 and 8, result: -7
Compare item 2 and 8, result: -6
Compare item 3 and 8, result: -5
Compare item 4 and 8, result: -4
Compare item 5 and 8, result: -3
Compare item 6 and 8, result: -2
Compare item 7 and 8, result: -1
Compare item 8 and 8, result: 0
Compare item 9 and 8, result: 1
Compare item 10 and 8, result: 1
Compare item 11 and 8, result: 1
Compare item 12 and 8, result: 1
Compare item 13 and 8, result: 1
Compare item 14 and 8, result: 1
Compare item 15 and 8, result: 1
```

SQSORT(array-number,[ASCENDING/DESCENDING],[sort-offset])

Sorts an array using the quick sort algorithm in ascending or descending order, default is ascending. The sort-offset defines the sorting scope up to the end of the item, any substrings prior to it are not treated. If you define for example 5, the array is sorted at offset 5 (up to the rest of the item). The sort-offset defaults to 1.

This sort is 100-150 times faster than the BREXX quick sort running on stems.

Returns the number of sorted items.

Example

```
max=25
s1=SREAD("'pej.songs2'")
call slist s1
call sqsort(s1,'ASC',26)/* Sort Array S1 beginning column 26(song name) */
call slist s1            /* display sorted array */
call sfree s1
```

Result, song names are in sorted order

```
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   LED ZEPPELIN                STAIRWAY TO HEAVEN
00002   EAGLES                      HOTEL CALIFORNIA
00003   AC/DC                       BACK IN BLACK
00004   JOURNEY                     DON'T STOP BELIEVIN'
00005   PINK FLOYD                  ANOTHER BRICK IN THE WALL
00006   QUEEN                       BOHEMIAN RHAPSODY
00007   TOTO                        HOLD THE LINE
00008   KISS                        I WAS MADE FOR LOVIN' YOU
00009   BON JOVI                    LIVIN' ON A PRAYER
00010   NIRVANA                     SMELLS LIKE TEEN SPIRIT
00011   DEEP PURPLE                 SMOKE ON THE WATER
00012   METALLICA                   NOTHING ELSE MATTERS
00013   THE ROLLING STONES          (I CAN'T GET NO) SATISFACTIO
00014   BRUCE SPRINGSTEEN           BORN IN THE U.S.A.
00015   QUEEN                       WE WILL ROCK YOU
00016   LYNYRD SKYNYRD              FREE BIRD
00017   SURVIVOR                    EYE OF THE TIGER
00018   THE CLASH                   SHOULD I STAY OR SHOULD I GO
00019   JIMI HENDRIX                HEY JOE
00020   FLEETWOOD MAC               LITTLE LIES
00021   AC/DC                       HIGHWAY TO HELL
00022   THE POLICE                  ROXANNE

     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   THE ROLLING STONES          (I CAN'T GET NO) SATISFACTION
00002   PINK FLOYD                  ANOTHER BRICK IN THE WALL
```

```
00003    AC/DC                         BACK IN BLACK
00004    QUEEN                         BOHEMIAN RHAPSODY
00005    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
00006    JOURNEY                       DON'T STOP BELIEVIN'
00007    SURVIVOR                      EYE OF THE TIGER
00008    LYNYRD SKYNYRD                FREE BIRD
00009    JIMI HENDRIX                  HEY JOE
00010    AC/DC                         HIGHWAY TO HELL
00011    TOTO                          HOLD THE LINE
00012    EAGLES                        HOTEL CALIFORNIA
00013    KISS                          I WAS MADE FOR LOVIN' YOU
00014    FLEETWOOD MAC                 LITTLE LIES
00015    BON JOVI                      LIVIN' ON A PRAYER
00016    METALLICA                     NOTHING ELSE MATTERS
00017    THE POLICE                    ROXANNE
00018    THE CLASH                     SHOULD I STAY OR SHOULD I GO
00019    NIRVANA                       SMELLS LIKE TEEN SPIRIT
00020    DEEP PURPLE                   SMOKE ON THE WATER
00021    LED ZEPPELIN                  STAIRWAY TO HEAVEN
00022    QUEEN                         WE WILL ROCK YOU
```

SHSORT(array-number,[ASCENDING/DESCENDING],[sort-offset])

Sorts an array using the shell sort algorithm in ascending or descending order, default is ascending.

The sort-offset defines the sorting scope up to the end of the item, any substrings prior to it are not treated. If you define for example 5, the array is sorted at offset 5 (up to the rest of the item). The sort-offset defaults to 1.

This sort is 100-150 times faster than the BREXX shell sort running on stems.

SMERGE(array-number-1,array-number-2)

Merges 2 arrays into a new array, based on their sort order.

Returns the number of merged items.

Example

```
max=10
s1=SCREATE(max)        /* Create a String Array called S1 */
s2=SCREATE(max)        /* Create a String Array called S2 */
do i=1 to max
   call sset(s1,i,right((max-i+1),4,'0')' A Record') /* Add new Record in
Array S1 at position i */
   call sset(s2,i,right((max-i+1),4,'0')' B Record') /* Add new Record in
Array S1 at position i */
end
say "Source Array S1"
say "----------------"
call slist s1
say "Source Array S2"
say "----------------"
call slist s2
s3=smerge(s1,s2)            /* Merge Array S1 and S2 into S3 */
say "Source Array S3"
say "---------------"
call slist s3
return
```

Result

```
Source Array S1
---------------
00001    0010 A Record
00002    0009 A Record
00003    0008 A Record
00004    0007 A Record
00005    0006 A Record
00006    0005 A Record
00007    0004 A Record
00008    0003 A Record
00009    0002 A Record
00010    0001 A Record
Source Array S2
---------------
00001    0010 B Record
00002    0009 B Record
00003    0008 B Record
00004    0007 B Record
00005    0006 B Record
00006    0005 B Record
00007    0004 B Record
00008    0003 B Record
00009    0002 B Record
00010    0001 B Record
Source Array S3
---------------
00001    0001 A Record
00002    0001 B Record
00003    0002 A Record
00004    0002 B Record
00005    0003 A Record
00006    0003 B Record
00007    0004 A Record
00008    0004 B Record
00009    0005 A Record
00010    0005 B Record
00011    0006 A Record
00012    0006 B Record
00013    0007 A Record
00014    0007 B Record
00015    0008 A Record
00016    0008 B Record
00017    0009 A Record
00018    0009 B Record
00019    0010 A Record
00020    0010 B Record
```

SREVERSE(array-number)

reverses the order of an array, the first item becomes the last item, the last item the first item, etc. The reverse takes place in the specified array. There is no new array created. The reverse process is very quick as just the string addresses are swapped, not the string content.

Returns the number of elements of the array.

Example

```
smax=10
s1=screate(smax)
do i=1 to smax
   call sset(s1,i,right(i,6,'0')". Record")
end
say "Original"
say "--------"
call slist s1
call sreverse(s1)
say "Reversed"
say "--------"
call slist s1
call sfree(s1)
EXIT 0
```

Result

```
Original
--------
00001   000001. Record
00002   000002. Record
00003   000003. Record
00004   000004. Record
00005   000005. Record
00006   000006. Record
00007   000007. Record
00008   000008. Record
00009   000009. Record
00010   000010. Record
Reversed
--------
00001   000010. Record
00002   000009. Record
00003   000008. Record
00004   000007. Record
00005   000006. Record
00006   000005. Record
00007   000004. Record
00008   000003. Record
00009   000002. Record
00010   000001. Record
```

SFREE(array-number,[KEEP})

Removes the specified array and all its entries. All storage allocations are freed. If **KEEP** is specified all items are freed, but the array itself (array-number) remains allocated.

For example, see SWRITE.

SWRITE(array-number,dsn/ddname)

Writes all entries of the specified array into an external dataset.

The dataset can be either a fully qualified Dataset Name or a pre-allocated DD Name.

returned is the number of written entries.

For example, see SREAD.

SREAD(dsn/ddname<,size-of-array>)

Reads all entries of an external dataset into a new String Array. The dataset can be either a fully qualified Dataset Name or a pre-allocated DD Name. The optional parameter size-of-array is recommended for large datasets. If omitted the size of the array grows dynamically to accommodate the content.

returned is the newly created Array number.

Example:

```
s1=sread("'pej.songs'")      /* import CSV formatted DSN */
s2=screate(sarray(s1))       /* create formatted version */
do i=1 to sarray(s1)
   parse value sget(s1,i) with band':'song
   call sset(s2,i,left(band,25)song)
end
call slist s2
say swrite(s2,"'pej.songs2'")' Entries exported'
```

The contents of pej.songs, list of 20 best rock songs (not rated by me):

```
LED ZEPPELIN:     STAIRWAY TO HEAVEN
EAGLES:     HOTEL CALIFORNIA
AC/DC:     BACK IN BLACK
JOURNEY:     DON'T STOP BELIEVIN'
PINK FLOYD:     ANOTHER BRICK IN THE WALL
QUEEN:     BOHEMIAN RHAPSODY
TOTO:     HOLD THE LINE
KISS:     I WAS MADE FOR LOVIN' YOU
BON JOVI:     LIVIN' ON A PRAYER
NIRVANA:     SMELLS LIKE TEEN SPIRIT
DEEP PURPLE:     SMOKE ON THE WATER
METALLICA:     NOTHING ELSE MATTERS
THE ROLLING STONES:     (I CAN'T GET NO) SATISFACTION
BRUCE SPRINGSTEEN:     BORN IN THE U.S.A.
QUEEN:     WE WILL ROCK YOU
LYNYRD SKYNYRD:     FREE BIRD
SURVIVOR:     EYE OF THE TIGER
THE CLASH:     SHOULD I STAY OR SHOULD I GO
JIMI HENDRIX:     HEY JOE
FLEETWOOD MAC:     LITTLE LIES
AC/DC:     HIGHWAY TO HELL
THE POLICE:     ROXANNE
```

Result of fetched DSN:

```
     Entries of Source Array: 1
Entry   Data
-------------------------------------------------------
```

```
00001    LED ZEPPELIN                   STAIRWAY TO HEAVEN
00002    EAGLES                         HOTEL CALIFORNIA
00003    AC/DC                          BACK IN BLACK
00004    JOURNEY                        DON'T STOP BELIEVIN'
00005    PINK FLOYD                     ANOTHER BRICK IN THE WALL
00006    QUEEN                          BOHEMIAN RHAPSODY
00007    TOTO                           HOLD THE LINE
00008    KISS                           I WAS MADE FOR LOVIN' YOU
00009    BON JOVI                       LIVIN' ON A PRAYER
00010    NIRVANA                        SMELLS LIKE TEEN SPIRIT
00011    DEEP PURPLE                    SMOKE ON THE WATER
00012    METALLICA                      NOTHING ELSE MATTERS
00013    THE ROLLING STONES             (I CAN'T GET NO) SATISFACTION
00014    BRUCE SPRINGSTEEN              BORN IN THE U.S.A.
00015    QUEEN                          WE WILL ROCK YOU
00016    LYNYRD SKYNYRD                 FREE BIRD
00017    SURVIVOR                       EYE OF THE TIGER
00018    THE CLASH                      SHOULD I STAY OR SHOULD I GO
00019    JIMI HENDRIX                   HEY JOE
00020    FLEETWOOD MAC                  LITTLE LIES
00021    AC/DC                          HIGHWAY TO HELL
00022    THE POLICE                     ROXANNE
22 Entries exported
```

SLIST(array-number,[from],[to],[heading])

Prints the array content. With the optional from and to parameters, you can limit the range of entries to be printed. The optional heading parameter is printed in the heading line.

For example, see SREAD and others

SSEARCH(array-number,search-string, from,[“CASE”/”NOCASE”])

Searches in a String Array for a certain string and returns the index number. For repeated searches, you can use the from parameter.

Returns index position if found, or zero.

Example

```
s1=sread("'pej.songs2'")
ssc="ON"
ssi=ssearch(s1,ssc)  /* Search string ON in array */
do while ssi>0
   say "Found at "ssi": "sget(s1,ssi)
   ssi=ssearch(s1,ssc,ssi+1)
end
```

Result

```
Found at 4: JOURNEY                         DON'T STOP BELIEVIN'
Found at 9: BON JOVI                        LIVIN' ON A PRAYER
Found at 11: DEEP PURPLE                     SMOKE ON THE WATER
Found at 13: THE ROLLING STONES              (I CAN'T GET NO) SATISFACTION
```

SSELECT(array-number,search-1 ,[search-2,…,search-99])

Creates a subset of the array when an entry matches one of the specified search strings in a new array. There are up to 99 search strings allowed. The search is case-sensitive.

Returns the newly created array.

Example

```
s1=sread("'pej.songs2'")
call slist s1
s2=sselect(s1,'ON','OF','EE')    /* Search ON, OF, EE in array */
say copies('-',32)
say 'Selected'
say copies('-',32)
call slist s2
call sfree(s1)
call sfree(s2)
```

Result

```
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   LED ZEPPELIN                    STAIRWAY TO HEAVEN
00002   EAGLES                          HOTEL CALIFORNIA
00003   AC/DC                           BACK IN BLACK
00004   JOURNEY                         DON'T STOP BELIEVIN'
00005   PINK FLOYD                      ANOTHER BRICK IN THE WALL
00006   QUEEN                           BOHEMIAN RHAPSODY
00007   TOTO                            HOLD THE LINE
00008   KISS                            I WAS MADE FOR LOVIN' YOU
00009   BON JOVI                        LIVIN' ON A PRAYER
00010   NIRVANA                         SMELLS LIKE TEEN SPIRIT
00011   DEEP PURPLE                     SMOKE ON THE WATER
00012   METALLICA                       NOTHING ELSE MATTERS
00013   THE ROLLING STONES              (I CAN'T GET NO) SATISFACTION
00014   BRUCE SPRINGSTEEN               BORN IN THE U.S.A.
00015   QUEEN                           WE WILL ROCK YOU
00016   LYNYRD SKYNYRD                  FREE BIRD
00017   SURVIVOR                        EYE OF THE TIGER
00018   THE CLASH                       SHOULD I STAY OR SHOULD I GO
00019   JIMI HENDRIX                    HEY JOE
00020   FLEETWOOD MAC                   LITTLE LIES
00021   AC/DC                           HIGHWAY TO HELL
00022   THE POLICE                      ROXANNE
--------------------------------
Selected
--------------------------------
     Entries of Source Array: 1
Entry   Data
-------------------------------------------------------
00001   JOURNEY                         DON'T STOP BELIEVIN'
00002   QUEEN                           BOHEMIAN RHAPSODY
00003   BON JOVI                        LIVIN' ON A PRAYER
00004   NIRVANA                         SMELLS LIKE TEEN SPIRIT
00005   DEEP PURPLE                     SMOKE ON THE WATER
```

```
00006   THE ROLLING STONES                     (I CAN'T GET NO) SATISFACTION
00007   BRUCE SPRINGSTEEN                      BORN IN THE U.S.A.
00008   QUEEN                                  WE WILL ROCK YOU
00009   LYNYRD SKYNYRD                         FREE BIRD
00010   SURVIVOR                               EYE OF THE TIGER
00011   FLEETWOOD MAC                          LITTLE LIES
```

SCHANGE(array-number,from-1,to-1[,from2,to2[,from3,to3]])

Changes the content of the array (line by line), from-1 is replaced by to1, from2 by to2, etc. If multiple change parameters are specified, a subsequent change may re-change a previous change.

returned is the number of changes performed.

SSUBSTR(array-number,from-column,[length],[INTERNAL/EXTERNAL])

Creates an array with the substring of each line (according to the SUBSTR REXX function). EXTERNAL (default) creates a new array with the substring results. INTERNAL works on the existing array.

returned is the array number that has been created/used.

SCOUNT(array-number,search-string-1[,search-string-2[,search-string-3…]])

Counts the lines containing the search strings. Multiple occurrences of a search string in a line are not counted, but hits of additional search strings on a line will be counted.

returned is the number of lines containing the search strings

SDROP(array-number,drop-string-1[,drop-string-2[,drop-string-3…]])

Drops lines containing the drop strings. An empty drop string is treated to drop empty lines.

returned is the number of lines containing the search strings

SSPLIT(string-to-split,delimiter-chars)

SPLIT splits a string into lines and stores them in a SARRAY. The optional delimiter table defines the split character(s), which shall be used to separate the lines. The delimiter string may consist of more than one character. This function is useful if you have file content in one string containing the line-feed character. SSPLIT returns the array number created.

SEXTRACT(array-number,begin-lino,end-lino)

SEXTRACT extracts lines of a SARRAY. The first parameter is the line to begin, second is the last line to be extracted, it is not the number of lines. End-lino defaults to the last line of the source array.

SCUT(array-number,begin-string,end-string,[from-line],[NO-DELIMITER/DELIMITER])

SCUT extracts lines of a SARRAY. **If NO-DELIMITER is specified, t**he extraction starts with the lines after the begin-string and ends with the line before the end string is found. If DELIMITER is specified, the delimiter lines are included. The default is NO-DELIMITER.

For example, we have the following SARRAY (s1):

```
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   Record 1
00002   Record 2
00003   Record 3
00004   Record 4
00005   Record 5
00006   From Here
00007   Data 1
00008   Data 2
00009   Data 3
00010   End
00011   Record 6
00012   Record 7
00013   Record 8
00014   Record 9
00015   Record 10
15 Entries
```

And the following REXX

```

s2=sextract(s1,"From Here","End")
call slist s2
```

Result

```
     Entries of Source Array: 1
Entry   Data
-------------------------------------------------------
00001   Data 1
00002   Data 2
00003   Data 3
3 Entries
```

SARRAY (array-number)

Returns information about the Source Array. The following BREXX variables are set:

```
sarrayhi       highest element number set in the array
sarraymax      maximum entries available
sarrayADDR     address of the Source Array
```

Returns the highest array entry.

II. Integer Array Functions

A. Simple Integer Array

ICREATE(elements,mode)

Creates an integer array with the size elements. Returned is the array number to be used to address the array with **ISET** and **IGET.** You can have up to 64 integer arrays. Depending on the virtual storage they may contain 1 million elements and more. Accessing integer arrays is very fast as there is no overhead compared to STEM variables.

Elements number entries available

mode is the initialization type

Element-Number	index of an element
NULL	elements are set to 0
DESCENT	index of the element in reverse order
SUNDARAM	prime numbers (Sundaram algorithm)
PRIME	prime numbers (sieve of Eratosthenes)

If the mode is not set the array remains uninitialized.

Returns the allocated **array-number** which can be used in subsequent array functions.

ISET(array-number,element-number,integer-value)

Sets a certain element of an array with an integer value.

IGET(array-number,element-number)

Gets (returns) a certain element of an array with an integer value.

IADD(array-number,row,column,integer-value)

Adds an integer value to a certain element of the array.

ISUB(array-number,row,column,integer-value)

subtracts an integer value from a certain element of the array.

IAPPEND(array-1,array-2)

Create a new array by appending array-1 by array-2.

Returns the newly created **array-number** which can be used in subsequent array functions.

IARRAY(array-number)

Returns the highest array index set in the integer array

ILIST(array-number,[from],[to],[heading])

Prints the array content. With the optional from and to parameters, you can limit the range of entries to be printed. The optional heading parameter is printed in the heading line.

B. Integer Matrix

The integer matrix is based on an integer array, the rows and columns are internally translated into the position in the array.

IMCREATE(rows,columns)

Creates an integer matrix containing the specified number of rows and columns. The matrix is initialized with zeros.

Returns the allocated **array-number** which can be used in subsequent array functions.

IMSET(array-number,row,column,integer-value)

Sets a certain element of the matrix to an integer value.

IMGET(array-number,row,column)

Gets (returns) a certain element of the matrix.

IMADD(array-number,row,column,integer-value)

Adds an integer value to a certain element of the matrix.

IMSUB(array-number,row,column,integer-value)

subtracts an integer value from a certain element of the matrix.

IARRAY(array-number,'ROW'/'COLUMN)`

Returns the number of rows or columns of the matrix.

IFREE(array-number)

Frees a defined integer array or matrix.

III. Float Array

FCREATE(elements,mode)

Creates an float array with the size elements. Returned is the array number to be used to address the array with **FSET** and F**GET.** You can have up to 64 integer arrays.

Returns the allocated **array-number** which can be used in subsequent array functions.

FSET(array-number,element-number,float-value)

Sets a certain element of an array with a float value.

FGET(array-number,element-number)

Gets (returns) a certain element of the float array.

FARRAY(array-number)

Returns the highest array index set in the float array

FLIST(array-number,[from],[to],[heading])

Prints the array content. With the optional from and to parameters, you can limit the range of entries to be printed. The optional heading parameter is printed in the heading line.

FFREE(array-number)

Frees a defined float array.

IV. Linked List functions

LLCREATE()

Creates a Linked List, returned is the Linked List Number(llist-number) which must be used in various Linked List operations.
The Linked List is bidirectional. You can have up to 32 different Linked Lists, depending on the virtual storage availability.
Returns the allocated **linked-list-number** which can be used in subsequent linked list functions.

LLFREE(llist-number)

Removes the Linked List and all its entries. All storage allocations are freed.

LLCLEAR(llist-number)

Clears (removes) the Linked List entries, but the list header remains intact. From there you can add new entries to it.

LLADD(llist-number,"entry-text")

Adds a new entry (llentry) at the end of the Linked List and links up the previous entry with a forward and the new entry backward reference. If the operation is successful a pointer (llpointer) to the new entry is returned. If the operation fails a return code < 0 is returned.

The internal current pointer (llcurrent) is set to the new entry and can be used in subsequent Linked List operations.

Example see LLINSERT

LLDEL(llist-number,[llist-pointer])

Removes an entry, defined by the current entry or the specified llist-pointer (llpointer. If the operation was successful the internal current pointer (llcurrent) is set to the next entry, if there is no one, to the last element. Returned will be the internal current pointer (llcurrent). If the operation fails a return code < 0 is returned.

Example

```
ll1=llread("'pej.songs2'")          /* Create Linked List */
call lllist ll1
call llset(ll1,"POSITION",3)        /* set to 3. Entry    */
call lldel(ll1)                     /* remove AC/DC       */
call lllist ll1
call llfree ll1
```

Result

```
     Entries of Linked List: 0  (0)
Entry Entry Address      Next     Previous       Data
-------------------------------------------------------
    1      3061c8      306258            0    LED ZEPPELIN                  STAIRWAY TO HEAVEN
    2      306258      3062e8       3061c8    EAGLES                        HOTEL CALIFORNIA
    3      3062e8      306378       306258    AC/DC                         BACK IN BLACK
    4      306378      306408       3062e8    JOURNEY                       DON'T STOP BELIEVIN'
    5      306408      306498       306378    PINK FLOYD                    ANOTHER BRICK IN THE WALL
    6      306498      306528       306408    QUEEN                         BOHEMIAN RHAPSODY
    7      306528      3065b8       306498    TOTO                          HOLD THE LINE
```

```
     8     3065b8      306648      306528    KISS                          I WAS MADE FOR LOVIN' YOU
     9     306648      3066d8      3065b8    BON JOVI                      LIVIN' ON A PRAYER
    10     3066d8      306768      306648    NIRVANA                       SMELLS LIKE TEEN SPIRIT
    11     306768      3067f8      3066d8    DEEP PURPLE                   SMOKE ON THE WATER
    12     3067f8      306888      306768    METALLICA                     NOTHING ELSE MATTERS
    13     306888      306918      3067f8    THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
    14     306918      3069a8      306888    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
    15     3069a8      305498      306918    QUEEN                         WE WILL ROCK YOU
    16     305498      306a38      3069a8    LYNYRD SKYNYRD                FREE BIRD
    17     306a38      306ac8      305498    SURVIVOR                      EYE OF THE TIGER
    18     306ac8      305458      306a38    THE CLASH                     SHOULD I STAY OR SHOULD I GO
    19     305458      305658      306ac8    JIMI HENDRIX                  HEY JOE
    20     305658      306b58      305458    FLEETWOOD MAC                 LITTLE LIES
    21     306b58      305618      305658    AC/DC                         HIGHWAY TO HELL
    22     305618           0      306b58    THE POLICE                    ROXANNE
Linked List contains 22 Entries
       List counter  22 Entries
Current active Entry 305618
     Entries of Linked List: 0 (0)
Entry Entry Address      Next     Previous       Data
-------------------------------------------------------
     1     3061c8      306258           0    LED ZEPPELIN                  STAIRWAY TO HEAVEN
     2     306258      306378      3061c8    EAGLES                        HOTEL CALIFORNIA
     3     306378      306408      306258    JOURNEY                       DON'T STOP BELIEVIN'
     4     306408      306498      306378    PINK FLOYD                    ANOTHER BRICK IN THE WALL
     5     306498      306528      306408    QUEEN                         BOHEMIAN RHAPSODY
     6     306528      3065b8      306498    TOTO                          HOLD THE LINE
     7     3065b8      306648      306528    KISS                          I WAS MADE FOR LOVIN' YOU
     8     306648      3066d8      3065b8    BON JOVI                      LIVIN' ON A PRAYER
     9     3066d8      306768      306648    NIRVANA                       SMELLS LIKE TEEN SPIRIT
    10     306768      3067f8      3066d8    DEEP PURPLE                   SMOKE ON THE WATER
    11     3067f8      306888      306768    METALLICA                     NOTHING ELSE MATTERS
    12     306888      306918      3067f8    THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
    13     306918      3069a8      306888    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
    14     3069a8      305498      306918    QUEEN                         WE WILL ROCK YOU
    15     305498      306a38      3069a8    LYNYRD SKYNYRD                FREE BIRD
    16     306a38      306ac8      305498    SURVIVOR                      EYE OF THE TIGER
    17     306ac8      305458      306a38    THE CLASH                     SHOULD I STAY OR SHOULD I GO
    18     305458      305658      306ac8    JIMI HENDRIX                  HEY JOE
    19     305658      306b58      305458    FLEETWOOD MAC                 LITTLE LIES
    20     306b58      305618      305658    AC/DC                         HIGHWAY TO HELL
    21     305618           0      306b58    THE POLICE                    ROXANNE
Linked List contains 21 Entries
       List counter  21 Entries
Current active Entry 306378
```

LLINSERT(llist-number,"entry-text"[,llist-pointer])

Inserts a new entry (llentry) **before** the current entry or the specified llist-pointer. All link information from the predecessor and successor entries is updated.

If the operation is successful a pointer (llpointer) to the inserted entry is returned. If the operation fails a return code < 0 is returned.

The internal current pointer (llcurrent) is set to the new entry and can be used in subsequent Linked List operations.

Example

```
ll1=llread("'pej.songs2'")           /* Create Linked List */
say copies('-',32)
say "Run Through Linked List"
say copies('-',32)
say llget(ll1,"FIRST")
do while llset(ll1,"NEXT")>0
   say llget(ll1)
end
call llset(ll1,"POSITION",2)       /* set to 1. Entry    */
call llinsert(ll1,"CREAM                         I AM SO GLAD")
call lllist ll1
call llfree ll1
```

Result

```
--------------------------------
Run Through Linked List
--------------------------------
LED ZEPPELIN                  STAIRWAY TO HEAVEN
EAGLES                        HOTEL CALIFORNIA
AC/DC                         BACK IN BLACK
JOURNEY                       DON'T STOP BELIEVIN'
PINK FLOYD                    ANOTHER BRICK IN THE WALL
QUEEN                         BOHEMIAN RHAPSODY
TOTO                          HOLD THE LINE
KISS                          I WAS MADE FOR LOVIN' YOU
BON JOVI                      LIVIN' ON A PRAYER
NIRVANA                       SMELLS LIKE TEEN SPIRIT
DEEP PURPLE                   SMOKE ON THE WATER
METALLICA                     NOTHING ELSE MATTERS
THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
QUEEN                         WE WILL ROCK YOU
LYNYRD SKYNYRD                FREE BIRD
SURVIVOR                      EYE OF THE TIGER
THE CLASH                     SHOULD I STAY OR SHOULD I GO
JIMI HENDRIX                  HEY JOE
FLEETWOOD MAC                 LITTLE LIES
AC/DC                         HIGHWAY TO HELL
THE POLICE                    ROXANNE
     Entries of Linked List: 0 (0)
Entry Entry Address     Next     Previous      Data
-------------------------------------------------------
    1     305258      305138          0    LED ZEPPELIN                  STAIRWAY TO HEAVEN
    2     305138      3052e8     305258    CREAM                         I AM SO GLAD
    3     3052e8      305378     305258    EAGLES                        HOTEL CALIFORNIA
    4     305378      305408     3052e8    AC/DC                         BACK IN BLACK
    5     305408      305498     305378    JOURNEY                       DON'T STOP BELIEVIN'
    6     305498      305528     305408    PINK FLOYD                    ANOTHER BRICK IN THE WALL
    7     305528      3055b8     305498    QUEEN                         BOHEMIAN RHAPSODY
    8     3055b8      305648     305528    TOTO                          HOLD THE LINE
    9     305648      3056d8     3055b8    KISS                          I WAS MADE FOR LOVIN' YOU
   10     3056d8      305768     305648    BON JOVI                      LIVIN' ON A PRAYER
   11     305768      3057f8     3056d8    NIRVANA                       SMELLS LIKE TEEN SPIRIT
   12     3057f8      305888     305768    DEEP PURPLE                   SMOKE ON THE WATER
   13     305888      305918     3057f8    METALLICA                     NOTHING ELSE MATTERS
   14     305918      3059a8     305888    THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
   15     3059a8      305a38     305918    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
   16     305a38      304818     3059a8    QUEEN                         WE WILL ROCK YOU
   17     304818      305ac8     305a38    LYNYRD SKYNYRD                FREE BIRD
   18     305ac8      305b58     304818    SURVIVOR                      EYE OF THE TIGER
   19     305b58      3047d8     305ac8    THE CLASH                     SHOULD I STAY OR SHOULD I GO
   20     3047d8      3049d8     305b58    JIMI HENDRIX                  HEY JOE
   21     3049d8      305be8     3047d8    FLEETWOOD MAC                 LITTLE LIES
   22     305be8      304998     3049d8    AC/DC                         HIGHWAY TO HELL
   23     304998           0     305be8    THE POLICE                    ROXANNE
Linked List contains 23 Entries
       List counter  23 Entries
Current active Entry 305138
```

LLGET(llist-number[option/llist-pointer])

Returns the entry referred by the option or internal current pointer, or the specified llist-pointer. The internal current pointer (llcurrent) is not changed.

Options:

NEXT sets it to the next element after llcurrent in the Linked List chain. If llcurrent was the last element 0 is returned.

PREVIOUS sets it to the previous element of llcurrent in the Linked List chain. If llcurrent was the first element 0 is returned.

FIRST	sets it to the first element in the Linked List.
LAST	sets it to the last element in the Linked List.

Example

```
ll1=llread("'pej.songs2'")            /* Create Linked List */
say copies('-',32)
say "Run Through Linked List"
say copies('-',32)
say llget(ll1,"FIRST")
do while llset(ll1,"NEXT")>0
   say llget(ll1)
end
call llfree ll1
```

Result

```
--------------------------------
Run Through Linked List
--------------------------------
LED ZEPPELIN                  STAIRWAY TO HEAVEN
EAGLES                        HOTEL CALIFORNIA
AC/DC                         BACK IN BLACK
JOURNEY                       DON'T STOP BELIEVIN'
PINK FLOYD                    ANOTHER BRICK IN THE WALL
QUEEN                         BOHEMIAN RHAPSODY
TOTO                          HOLD THE LINE
KISS                          I WAS MADE FOR LOVIN' YOU
BON JOVI                      LIVIN' ON A PRAYER
NIRVANA                       SMELLS LIKE TEEN SPIRIT
DEEP PURPLE                   SMOKE ON THE WATER
METALLICA                     NOTHING ELSE MATTERS
THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
QUEEN                         WE WILL ROCK YOU
LYNYRD SKYNYRD                FREE BIRD
SURVIVOR                      EYE OF THE TIGER
THE CLASH                     SHOULD I STAY OR SHOULD I GO
JIMI HENDRIX                  HEY JOE
FLEETWOOD MAC                 LITTLE LIES
AC/DC                         HIGHWAY TO HELL
THE POLICE                    ROXANNE
```

LLSET(llist-number,option[,sub-option])

Changes the internal current pointer according to the specified option and returns it as a pointer.

Options:

NEXT	sets it to the next element after llcurrent in the Linked List chain. If llcurrent was the last element 0 is returned.
PREVIOUS	sets it to the previous element of llcurrent in the Linked List chain. If llcurrent was the first element 0 is returned.
FIRST	sets it to the first element in the Linked List.
LAST	sets it to the last element in the Linked List.

POSITION	sets it to n.th entry, as defined in sub-option. If the specified number is not available it is set to the last entry.
CURRENT	returns the current internal current pointer.
ADDRESS	sets it according to the address defined in the sub-option.

LLCOPY(llist-number,[from],[to],[existing-list],[”list-name”])

Creates a copy of the Linked List. If an existing linked-list is specified, the entries are added after its existing entries.

from	(optional) starts the copying process at from.th entry.
to	(optional) ends the copying process with to.th entry.
existing-list	(optional) appending an existing Source Array, else a new one will be created
list-name	(optional) names the new/appended Link List

returned is the newly created or appended Linked List Number(llist-number)

Example

```
max=10
ll1=llcreate()                         /* Create Linked List */
ll2=llcreate()                         /* Create Linked List */
call time('r')
do i=1 to max
   adr=lladd(ll1,i". Record")
end
call llList ll1
do i=1 to 5
   adr=lladd(ll2,i". Entry")
end
call llList ll2
ll3=llcopy(ll1,,,ll2,"Copied")
call llList ll3
```

Result

```
     Entries of Linked List: 0  (UNNAMED)
 Entry Entry Address     Next     Previous       Data
 --------------------------------------------------------
     1      2e3258      2e3278           0    1. Record
     2      2e3278      2e3298      2e3258    2. Record
     3      2e3298      2e32b8      2e3278    3. Record
     4      2e32b8      2e32d8      2e3298    4. Record
     5      2e32d8      2e32f8      2e32b8    5. Record
     6      2e32f8      2e3318      2e32d8    6. Record
     7      2e3318      2e3338      2e32f8    7. Record
     8      2e3338      2e3358      2e3318    8. Record
     9      2e3358      2e3378      2e3338    9. Record
    10      2e3378           0      2e3358    10. Record
 Linked List contains 10 Entries
        List counter  10 Entries
 Current active Entry 2e3378

     Entries of Linked List: 1  (UNNAMED)
 Entry Entry Address     Next     Previous       Data
```

```
  --------------------------------------------------------
      1      2e3398      2e33b8           0   1. Entry
      2      2e33b8      2e33d8      2e3398   2. Entry
      3      2e33d8      2e33f8      2e33b8   3. Entry
      4      2e33f8      2e3418      2e33d8   4. Entry
      5      2e3418           0      2e33f8   5. Entry
Current active Entry 2e3418
Linked List contains 5 Entries
       List counter  5 Entries

     Entries of Linked List: 1 (Copied)
Entry Entry Address     Next    Previous       Data
-------------------------------------------------------
    1      2e3398      2e33b8           0   1. Entry
    2      2e33b8      2e33d8      2e3398   2. Entry
    3      2e33d8      2e33f8      2e33b8   3. Entry
    4      2e33f8      2e3418      2e33d8   4. Entry
    5      2e3418      2e3458      2e33f8   5. Entry
    6      2e3458      2e3478      2e3418   1. Record
    7      2e3478      2e3498      2e3458   2. Record
    8      2e3498      2e34b8      2e3478   3. Record
    9      2e34b8      2e34d8      2e3498   4. Record
   10      2e34d8      2e34f8      2e34b8   5. Record
   11      2e34f8      2e3518      2e34d8   6. Record
   12      2e3518      2e3538      2e34f8   7. Record
   13      2e3538      2e3558      2e3518   8. Record
   14      2e3558      2e3578      2e3538   9. Record
   15      2e3578           0      2e3558   10. Record
Linked List contains 15 Entries
       List counter  15 Entries
Current active Entry 2e3578
```

LLENTRY(llist-number [,llist-pointer]))

Dumps the details of an entry either defined by the internal current pointer (llcurrent) or the llist-pointer.

```
---------------------------------------------
Linked List Entry
---------------------------------------------
Address  326f18
Data     42. Record
Next     326f58
Previous 326ed8
```

LLLIST(llist-number[,from],[to])

Outputs a detailed list of all entries on a Linked list.

```
     Entries of Linked List: 0
Entry Entry Address     Next    Previous       Data
-------------------------------------------------------
    1      326458      326498           0   1. Record
    2      326498      3264d8      326458   2. Record
    3      3264d8      326518      326498    3. Record
    4      326518      326558      3264d8    4. Record
    5      326558      326598      326518    5. Record
    6      326598      3265d8      326558    6. Record
    7      3265d8      326618      326598    7. Record
    8      326618      326658      3265d8    8. Record
    9      326658      326698      326618    9. Record
   10      326698      3266d8      326658    10. Record
   11      3266d8      326718      326698    11. Record
   12      326718      326758      3266d8    12. Record
…
```

LLDETAILS(llist-number,option)

Output statistics on the Linked List.

Options:

COUNT	returns the number of current entries in the Linked List.
ADDED	returns the number of added/inserted entries in the Linked List.
DELETED	returns the number of deleted entries in the Linked List.
LIST	returns the listed number of current entries in the Linked List. For this reason, it runs through the entire Linked List and counts the entries. LIST and COUNT should be equal, else there are inconsistencies in the Linked List.
FULL	Print all available information

```
CALL LLDETAILS(0,’FULL’)

Attributes of Linked List 0
-------------------------------------------------------
Entry Count     9999
     Listed     9999
      Added     10000
    Deleted     1
Current Pointer 326ed8
```

LLDELINK(llist-number[,llist-pointer])

Similar to LLDEL an entry defined by the current entry or the specified llist-pointer is removed from the Link List but is kept in storage as an orphan, which might be later inserted in a different position in the same or a different Linked List. This is a fast way of moving elements.

Returned is the address of the orphaned entry.

If the operation was successful the internal current pointer (llcurrent) is set to the next entry, if there is no one, to the last element.

The example is contained in the LLLINK sample.

LLLINK(llist-number,llist-pointer)

Links an orphaned entry to the Linked List **prior** to the current entry and sets the pointers accordingly.

If the operation was successful, the internal current pointer (llcurrent) is set to the newly inserted entry.

Example

```
max=10
ll1=llcreate()                      /* Create Linked List */
ll2=llcreate()                      /* Create Linked List */
do i=1 to max
   adr=lladd(ll1,i". Record")       /* add new entry      */
end
call llList ll1
posadr=llset(ll1,"POSITION",7)      /* set to 7. Entry    */
deladr=lldelink(ll1,posadr)         /* DELINK it          */
say "is now de-linked,ADDR "d2x(deladr)
call llList ll1
say "Insert one entry to LL2 "d2x(llinsert(ll2,"1. Entry"))
call llList ll2
say "LINK into new LList "d2x(llLink(ll2,deladr))
call llList ll2
```

Result

```
     Entries of Linked List: 0 (UNNAMED)
Entry Entry Address     Next    Previous      Data
-----------------------------------------------------
    1     2db238      2db258           0   1. Record
    2     2db258      2db278      2db238   2. Record
    3     2db278      2db298      2db258   3. Record
    4     2db298      2db2b8      2db278   4. Record
    5     2db2b8      2db2d8      2db298   5. Record
    6     2db2d8      2db2f8      2db2b8   6. Record
    7     2db2f8      2db318      2db2d8   7. Record
    8     2db318      2db338      2db2f8   8. Record
    9     2db338      2db358      2db318   9. Record
   10     2db358           0      2db338   10. Record
Linked List contains 10 Entries
       List counter  10 Entries
is now de-linked,ADDR 2DB2F8
     Entries of Linked List: 0 (UNNAMED)
Entry Entry Address     Next    Previous      Data
-----------------------------------------------------
    1     2db238      2db258           0   1. Record
    2     2db258      2db278      2db238   2. Record
    3     2db278      2db298      2db258   3. Record
    4     2db298      2db2b8      2db278   4. Record
    5     2db2b8      2db2d8      2db298   5. Record
    6     2db2d8      2db318      2db2b8   6. Record
    7     2db318      2db338      2db2d8   8. Record
    8     2db338      2db358      2db318   9. Record
    9     2db358           0      2db338   10. Record
```

```
Linked List contains 9 Entries
       List counter  9 Entries
Insert one entry to LL2 2DB3D8
     Entries of Linked List: 1 (UNNAMED)
Entry Entry Address     Next    Previous      Data
-------------------------------------------------------
    1      2db3d8           0           0   1. Entry
Linked List contains 1 Entries
       List counter  1 Entries
LINK into new LList 2DB2F8
     Entries of Linked List: 1 (UNNAMED)
Entry Entry Address     Next    Previous      Data
-------------------------------------------------------
    1      2db2f8      2db3d8           0   7. Record
    2      2db3d8           0           0   1. Entry
Linked List contains 2 Entries
       List counter  2 Entries
```

LLSORT(llist--number,[ASCENDING/DESCENDING],[sort-offset])

Sorts the Linked List using the quick sort algorithm in ascending or descending order, default is ascending.

The sort offset defines the sorting scope up to the end of the item, any substrings before it are not treated. If you define for example 5, the array is sorted at offset 5 (up to the rest of the item). The sort-offset defaults to 1.

returned is the Linked List Number(llist-number), it is the same as the entry list.

Example

```
ll1=llread("'pej.songs2'")
call llList ll1
call llsort ll1
call llList ll1    /* sort from column 1, band name */
call llfree ll1
```

Result

```
     Entries of Linked List: 0 (0)
Entry Entry Address     Next    Previous      Data
-------------------------------------------------------
    1      3371c8      337258           0   LED ZEPPELIN                 STAIRWAY TO HEAVEN
    2      337258      3372e8      3371c8   EAGLES                       HOTEL CALIFORNIA
    3      3372e8      337378      337258   AC/DC                        BACK IN BLACK
    4      337378      337408      3372e8   JOURNEY                      DON'T STOP BELIEVIN'
    5      337408      337498      337378   PINK FLOYD                   ANOTHER BRICK IN THE WALL
    6      337498      337528      337408   QUEEN                        BOHEMIAN RHAPSODY
    7      337528      3375b8      337498   TOTO                         HOLD THE LINE
    8      3375b8      337648      337528   KISS                         I WAS MADE FOR LOVIN' YOU
    9      337648      3376d8      3375b8   BON JOVI                     LIVIN' ON A PRAYER
   10      3376d8      337768      337648   NIRVANA                      SMELLS LIKE TEEN SPIRIT
   11      337768      3377f8      3376d8   DEEP PURPLE                  SMOKE ON THE WATER
   12      3377f8      337888      337768   METALLICA                    NOTHING ELSE MATTERS
   13      337888      337918      3377f8   THE ROLLING STONES           (I CAN'T GET NO) SATISFACTION
   14      337918      3379a8      337888   BRUCE SPRINGSTEEN            BORN IN THE U.S.A.
   15      3379a8      336258      337918   QUEEN                        WE WILL ROCK YOU
   16      336258      337a38      3379a8   LYNYRD SKYNYRD               FREE BIRD
   17      337a38      337ac8      336258   SURVIVOR                     EYE OF THE TIGER
   18      337ac8      336218      337a38   THE CLASH                    SHOULD I STAY OR SHOULD I GO
   19      336218      336418      337ac8   JIMI HENDRIX                 HEY JOE
   20      336418      337b58      336218   FLEETWOOD MAC                LITTLE LIES
   21      337b58      3363d8      336418   AC/DC                        HIGHWAY TO HELL
   22      3363d8           0       337b58    THE POLICE                    ROXANNE
Linked List contains 22 Entries
       List counter  22 Entries
Current active Entry 3363d8
     Entries of Linked List: 0 (0)
Entry Entry Address      Next     Previous      Data
-------------------------------------------------------
    1      300a38      3009a8            0    AC/DC                         BACK IN BLACK
    2      3009a8      300918       300a38    AC/DC                         HIGHWAY TO HELL
    3      300918      300888       3009a8    BON JOVI                      LIVIN' ON A PRAYER
    4      300888      3007f8       300918    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
    5      3007f8      300768       300888    DEEP PURPLE                   SMOKE ON THE WATER
    6      300768      337b58       3007f8    JOURNEY                       DON'T STOP BELIEVIN'
    7      337b58      337ac8       300768    KISS                          I WAS MADE FOR LOVIN' YOU
    8      337ac8      337a38       337b58    NIRVANA                       SMELLS LIKE TEEN SPIRIT
    9      337a38      3379a8       337ac8    PINK FLOYD                    ANOTHER BRICK IN THE WALL
   10      3379a8      337918       337a38    QUEEN                         BOHEMIAN RHAPSODY
   11      337918      337888       3379a8    TOTO                          HOLD THE LINE
   12      337888      3363d8       337918    EAGLES                        HOTEL CALIFORNIA
   13      3363d8      336418       337888    FLEETWOOD MAC                 LITTLE LIES
   14      336418      3377f8       3363d8    JIMI HENDRIX                  HEY JOE
   15      3377f8      336218       336418    LED ZEPPELIN                  STAIRWAY TO HEAVEN
   16      336218      337768       3377f8    LYNYRD SKYNYRD                FREE BIRD
   17      337768      3376d8       336218    METALLICA                     NOTHING ELSE MATTERS
   18      3376d8      337648       337768    QUEEN                         WE WILL ROCK YOU
   19      337648      3375b8       3376d8    SURVIVOR                      EYE OF THE TIGER
   20      3375b8      336258       337648    THE CLASH                     SHOULD I STAY OR SHOULD I GO
   21      336258      337528       3375b8    THE POLICE                    ROXANNE
   22      337528           0       336258    THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
Linked List contains 22 Entries
       List counter  22 Entries
Current active Entry 337528
```

LLWRITE(llist-number,dsn/ddname)

Writes all entries of a Linked List into an external dataset.

The dataset can be either a fully qualified Dataset Name or a pre-allocated DD Name.

returned is the number of written entries.
An example is contained in LLREAD

LLREAD(dsn/ddname)

Reads all entries of an external dataset. The dataset can be either a fully qualified Dataset Name or a pre-allocated DD Name.

returned is the newly created Linked List Number(llist-number).

Example

```
ll1=llread("'pej.songs2'")
call lllist ll1
say "Records written: "llwrite(ll1,"'pej.temp'")
```

Result

```
     Entries of Linked List: 0 (0)
Entry Entry Address      Next     Previous      Data
-------------------------------------------------------
    1      3371c8      337258            0    LED ZEPPELIN                  STAIRWAY TO HEAVEN
    2      337258      3372e8       3371c8    EAGLES                        HOTEL CALIFORNIA
    3      3372e8      337378       337258    AC/DC                         BACK IN BLACK
    4      337378      337408       3372e8    JOURNEY                       DON'T STOP BELIEVIN'
    5      337408      337498       337378    PINK FLOYD                    ANOTHER BRICK IN THE WALL
    6      337498      337528       337408    QUEEN                         BOHEMIAN RHAPSODY
    7      337528      3375b8       337498    TOTO                          HOLD THE LINE
    8      3375b8      337648       337528    KISS                          I WAS MADE FOR LOVIN' YOU
    9      337648      3376d8       3375b8    BON JOVI                      LIVIN' ON A PRAYER
```

```
   10      3376d8      337768      337648    NIRVANA                        SMELLS LIKE TEEN SPIRIT
   11      337768      3377f8      3376d8    DEEP PURPLE                    SMOKE ON THE WATER
   12      3377f8      337888      337768    METALLICA                      NOTHING ELSE MATTERS
   13      337888      337918      3377f8    THE ROLLING STONES             (I CAN'T GET NO) SATISFACTION
   14      337918      3379a8      337888    BRUCE SPRINGSTEEN              BORN IN THE U.S.A.
   15      3379a8      336358      337918    QUEEN                          WE WILL ROCK YOU
   16      336358      337a38      3379a8    LYNYRD SKYNYRD                 FREE BIRD
   17      337a38      337ac8      336358    SURVIVOR                       EYE OF THE TIGER
   18      337ac8      336318      337a38    THE CLASH                      SHOULD I STAY OR SHOULD I GO
   19      336318      336518      337ac8    JIMI HENDRIX                   HEY JOE
   20      336518      337b58      336318    FLEETWOOD MAC                  LITTLE LIES
   21      337b58      3364d8      336518    AC/DC                          HIGHWAY TO HELL
   22      3364d8           0      337b58    THE POLICE                     ROXANNE
Records written: 22
```

V. Matrix Functions

MCREATE(rows,columns)

Creates a (Float) matrix with size [rows x columns]. Returned is the Matrix number to be used in various matrix operations. You can have up to 128 matrixes, depending on the virtual storage available. Accessing a matrix ys is very fast as there is no overhead compared to STEM variables.

Returns the allocated **matrix-number** which can be used in subsequent matrix functions.

MSET(matrix-number,row,column,float-value)

Sets a certain element of the matrix with a float value.

MGET(matrix-number,row,column)

Gets (returns) a certain element of the matrix.

MMULTIPLY(matrix-number-1,matrix-number-2)

Multiplies 2 matrices and creates a new matrix, which is returned. Input matrices remain untouched. The format of matrix-1 is [rows x columns], therefore the format of matrix-2 must be [columns x rows]. The format of the result matrix is rows x rows.

MINVERT(matrix-number)

Inverts the given matrix and creates a new matrix, which is returned. The input matrix must be squared and remains untouched. The format of the result matrix remains the same as the input matrix.

MTRANSPOSE(matrix-number)

Transposes the given matrix and creates a new matrix, which is returned. The input matrix remains untouched. If the format of the input matrix is [rows x columns] then the result matrix is columns x rows.

MCOPY(matrix-number)

Copies the given matrix and creates a new matrix, which is returned. The input matrix remains untouched. Formats of both matrices are equal.

MNORMALISE(matrix-number,mode)

Normalises the given matrix and creates a new matrix, which is returned. The input matrix remains untouched. The formats of both matrices are equal.

mode	**STANDARD**	row is normalized to mean=0 variance=1
	ROWS	row value is divided by the number of rows

MEAN	row value is normalized to mean=0, variance remains unchanged

MDELROW(matrix-number,row-number[,row-number[,row-number…]])

Copies the given matrix without the specified rows-to-delete as a new matrix, which is returned. The input matrix remains untouched.

MDELCOL(matrix-number,col-number[,col-number[,col-number…]])

Copies the given matrix without the specified columns-to-delete as a new matrix, which is returned. The input matrix remains untouched.

MPROPERTY(matrix-number[,"FULL"/"BASIC"])

Returns the properties of the given matrix in BREXX variables:

_rows	number of rows of matrix
_cols	number of columns of matrix.

If **FULL** is specified additionally the following stem variables are returned:

```
_rowmean.column-i       mean of rows of column-i
_rowvariance.column-i   variance of rows of column-i
_rowlow.column-i        lowest row value of column-i
_rowhigh.column-i       highest row value of column-i
_rowsum.column-i        sum of row value of column-i
_rowsqr.column-i        sum of squared row value of column-i
_colsum.row-i           sum of column values of row-i
_colsqr.row-i           sum of squared column values of row-i
```

MSCALAR(matrix-number,number)

Multiplies each element of a matrix with a number (float). The result is stored in a new matrix, which is returned. The input matrix remains untouched.

MADD(matrix-number-1, matrix-number-2)

Adds each element of a matrix-1 with the same element of matrix-2. The result is stored in a new matrix, which is returned. The input matrix remains untouched. Matrix-1 and matrix-2 must have the same dimensions.

MSUBTRACT(matrix-number-1, matrix-number-2)

Subtracts each element of a matrix-2 from the same element of matrix-1. The result is stored in a new matrix, which is returned. The input matrix remains untouched. Matrix-1 and matrix-2 must have the same dimensions.

MPROD(matrix-number-1, matrix-number-2)

Multiplies each element of a matrix-1 with the same element of matrix-2. The result is stored in a new matrix, which is returned. The input matrix remains untouched. Matrix-1 and matrix-2 must have the same dimensions.

MSQR(matrix-number)

Squares each element of the matrix. The result is stored in a new matrix, which is returned. The input matrix remains untouched.

MINSCOL(matrix-number,)

Inserts a new column as the first column. The initial first column becomes the second column, etc. The result is stored in a new matrix, which is returned. The input matrix remains untouched.

MFREE([matrix-number/integer-array-number,“MATRIX”/”INTEGER-ARRAY”])

Frees the storage of allocated matrices and/or integer arrays. If no parameter is specified all allocations are freed. To release a specific matrix or integer-array the matrix-number or integer-array-number must be used as the first parameter, followed by the type to release.

VI. Conversions between String Arrays, Linked Lists, and STEMS.

STEM2S("stem-name.")

Copies a stem variable into a Source Array, **stem-name.0** must contain the number of items.

The copy process takes stem-name.1, stem-name.2, … up stem-name.n (where n is contained in stem-name.0) and copies it into a String Array.

Returned is the number of the String Array.

Example

```
xmax=1000
do i=1 to xmax
   fred.i=i". record"
end
FRED.0=xmax
say "Set Time  "time('e')
call time('r')
s1=stem2s("fred.")
say "Copy Time "time('e')
call slist s1,xmax-10,xmax
```

Result

```
Set Time  0.130996
Copy Time 0.066642
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00990   990. record
00991   991. record
00992   992. record
00993   993. record
00994   994. record
00995   995. record
00996   996. record
00997   997. record
00998   998. record
00999   999. record
01000   1000. record
```

S2STEM("array-number","stem-name.")

Copies a SARRAY into a stem

Returned is the number of the items in the stem (String Array).

Example

```
smax=1000
s1=screate(smax)
do i=1 to smax
   call sset(s1,,"Record "i)
end
call slist s1,smax-10,smax
call time('r')
call s2stem(s1,"Fred.")
say "S2STEM "time('e')
```

```
do i=smax-10 to smax
   say i fred.i
end
```

Result

```
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00990   Record 990
00991   Record 991
00992   Record 992
00993   Record 993
00994   Record 994
00995   Record 995
00996   Record 996
00997   Record 997
00998   Record 998
00999   Record 999
01000   Record 1000
S2STEM 0.253646
990 Record 990
991 Record 991
992 Record 992
993 Record 993
994 Record 994
995 Record 995
996 Record 996
997 Record 997
998 Record 998
999 Record 999
1000 Record 1000
```

S2IARRAY

Copies a SARRAY into an integer array.

Returned is the array number of the created array (Integer Array).

S2FARRAY

Copies a SARRAY into a float array.

Returned is the array number of the created array (Float Array).

S2LL(array-number,[from],[to],[existing-linked-list],[“list-name”])

Copy a String Array into Linked List.

from	(optional) starts the copying process at from.th entry.
to	(optional) ends the copying process with to.th entry.
existing-list	(optional) appending an existing Linked List, else a new one will be created
list-name	(optional) name of the new/appended Linked List

returned is the Linked List Number(llist-number)

Example

```
s1=sread("'pej.songs2'")
call sList s1
ll2=s2ll(s1,,,,"LL Songs")
call llList(ll2)
call sfree(s1)
call llfree(ll2)
```

Result

```
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   LED ZEPPELIN                  STAIRWAY TO HEAVEN
00002   EAGLES                        HOTEL CALIFORNIA
00003   AC/DC                         BACK IN BLACK
00004   JOURNEY                       DON'T STOP BELIEVIN'
00005   PINK FLOYD                    ANOTHER BRICK IN THE WALL
00006   QUEEN                         BOHEMIAN RHAPSODY
00007   TOTO                          HOLD THE LINE
00008   KISS                          I WAS MADE FOR LOVIN' YOU
00009   BON JOVI                      LIVIN' ON A PRAYER
00010   NIRVANA                       SMELLS LIKE TEEN SPIRIT
00011   DEEP PURPLE                   SMOKE ON THE WATER
00012   METALLICA                     NOTHING ELSE MATTERS
00013   THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
00014   BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
00015   QUEEN                         WE WILL ROCK YOU
00016   LYNYRD SKYNYRD                FREE BIRD
00017   SURVIVOR                      EYE OF THE TIGER
00018   THE CLASH                     SHOULD I STAY OR SHOULD I GO
00019   JIMI HENDRIX                  HEY JOE
00020   FLEETWOOD MAC                 LITTLE LIES
00021   AC/DC                         HIGHWAY TO HELL
00022   THE POLICE                    ROXANNE
     Entries of Linked List: 0 (LL Songs)
Entry Entry Address     Next     Previous      Data
-------------------------------------------------------
    1     337138      3371c8           0    LED ZEPPELIN                  STAIRWAY TO HEAVEN
    2     3371c8      337258      337138    EAGLES                        HOTEL CALIFORNIA
    3     337258      3372e8      3371c8    AC/DC                         BACK IN BLACK
    4     3372e8      337378      337258    JOURNEY                       DON'T STOP BELIEVIN'
    5     337378      337408      3372e8    PINK FLOYD                    ANOTHER BRICK IN THE WALL
    6     337408      337498      337378    QUEEN                         BOHEMIAN RHAPSODY
    7     337498      337528      337408    TOTO                          HOLD THE LINE
    8     337528      3375b8      337498    KISS                          I WAS MADE FOR LOVIN' YOU
    9     3375b8      337648      337528    BON JOVI                      LIVIN' ON A PRAYER
   10     337648      3376d8      3375b8    NIRVANA                       SMELLS LIKE TEEN SPIRIT
   11     3376d8      337768      337648    DEEP PURPLE                   SMOKE ON THE WATER
   12     337768      3377f8      3376d8    METALLICA                     NOTHING ELSE MATTERS
   13     3377f8      337888      337768    THE ROLLING STONES            (I CAN'T GET NO) SATISFACTION
   14     337888      337918      3377f8    BRUCE SPRINGSTEEN             BORN IN THE U.S.A.
   15     337918      3362d8      337888    QUEEN                         WE WILL ROCK YOU
   16     3362d8      3379a8      337918    LYNYRD SKYNYRD                FREE BIRD
   17     3379a8      337a38      3362d8    SURVIVOR                      EYE OF THE TIGER
   18     337a38      336618      3379a8    THE CLASH                     SHOULD I STAY OR SHOULD I GO
   19     336618      3365d8      337a38    JIMI HENDRIX                  HEY JOE
   20     3365d8      337ac8      336618    FLEETWOOD MAC                 LITTLE LIES
   21     337ac8      336598      3365d8    AC/DC                         HIGHWAY TO HELL
   22     336598           0      337ac8    THE POLICE                    ROXANNE
Linked List contains 22 Entries
       List counter  22 Entries
Current active Entry 336598
```

LL2S(llist-number,[from],[to],[existing-array])

Copy a Linked List into a Source Array.

from	(optional) starts the copying process at from.th entry.
to	(optional) ends the copying process with to.th entry.
existing-array	(optional) appending an existing Source Array, else a new one will be created

returned is the Linked List Number(llist-number)

Example

```
max=8
ll1=llcreate()
do i=1 to max
   adr=lladd(ll1,i". Record")
end
call llList ll1
s1=ll2s(ll1)
say "Linked List copied into Source Array "s1
call slist(s1)
call llfree(ll1)
call sfree(s1)
```

Result

```
     Entries of Linked List: 0  (UNNAMED)
Entry Entry Address     Next     Previous       Data
-------------------------------------------------------
    1     2e3218      2e3238           0    1. Record
    2     2e3238      2e3258      2e3218    2. Record
    3     2e3258      2e3278      2e3238    3. Record
    4     2e3278      2e3298      2e3258    4. Record
    5     2e3298      2e32b8      2e3278    5. Record
    6     2e32b8      2e32d8      2e3298    6. Record
    7     2e32d8      2e32f8      2e32b8    7. Record
    8     2e32f8           0      2e32d8    8. Record
Linked List contains 8 Entries
       List counter  8 Entries
Current active Entry 2e32f8
Linked List copied into Source Array 0
     Entries of Source Array: 0
Entry   Data
-------------------------------------------------------
00001   1. Record
00002   2. Record
00003   3. Record
00004   4. Record
00005   5. Record
00006   6. Record
00007   7. Record
00008   8. Record
```

LL2STEM(“llist-number”)

Copies a Linked List into a stem

Returned is the number of the items in the stem (Linked List entries).

Example

```
max=1000
/* -------------------------------------------
 * Copy LLIST into STEM
 * -------------------------------------------
 */
```

```
LL1=LLCREATE("LLIST")
do i=1 to max
   call LLADD(LL1,'FRED 'i)
end
call time('r')
call ll2stem(LL1,'myStem.')
say "LL2STEM "time('e')
do i=mystem.0-10 to mystem.0
   say i mystem.i
end
```

Result

```
LL2STEM 0.195885
990 FRED 990
991 FRED 991
992 FRED 992
993 FRED 993
994 FRED 994
995 FRED 995
996 FRED 996
997 FRED 997
998 FRED 998
999 FRED 999
1000 FRED 1000
```

STEM2LL("stem-name.")

Copies stem into a Linked List, **stem-name.0** must contain the number of items.

The copy process takes stem-name.1, stem-name.2, … up stem-name.n (where n is contained in stem-name.0) and copies it into a Linked List.

Returned is the created Linked List number.

Example

```
max=1000
/* ------------------------------------------
 * Copy STEM into LLIST
 * ------------------------------------------
 */
do i=1 to max
   myStem.i=i". Record"
end
mystem.0=max
call time('r')
ll1=stem2ll('myStem.')
say "STEM2LL "time('e')
call lllist ll1,max-10,max
```

Result

```
STEM2LL 0.080318
     Entries of Linked List: 0 (UNNAMED)
Entry Entry Address     Next    Previous      Data
-------------------------------------------------------
  990      359498       3594d8       359458    990. Record
  991      3594d8       359518       359498    991. Record
  992      359518      359558      3594d8   992. Record
  993      359558      359598      359518   993. Record
  994      359598      3595d8      359558   994. Record
  995      3595d8      359618      359598   995. Record
  996      359618      359658      3595d8   996. Record
  997      359658      359698      359618   997. Record
  998      359698      3596d8      359658   998. Record
  999      3596d8      359718      359698   999. Record
 1000      359718           0      3596d8   1000. Record
Linked List address  34b218
Linked List contains 1000 Entries
       List counter  1000 Entries
Current active Entry 359718
```

BREXX/370 V2R5M2 Array Functions

Inhalt