SECTION 8

Arrays

8.1 ARRAYS

An array consists of a group of elements, each of which contains a data item. The individual elements are selected by numerical subscripts. The rank of an array is the number of subscripts used to refer to the elements of the array. The rank may be any integer from zero to seven, inclusive.

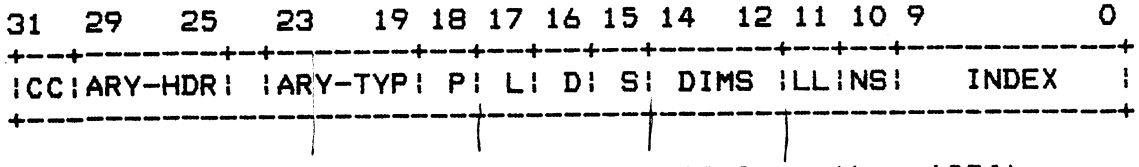
The lowest value for any subscript is zero; the highest value is a property of the array. Each dimension has a size, which is the lowest number which is too great to be used as a subscript. For example, for a one-dimensional array of five elements, the size of the one and only dimension is five, and the acceptable values of the subscript are the integers zero through four.

Any array may have an array leader. An array leader is like a one-dimensional ART-Q array which is prepended to the main array. Hence an array which has a leader acts like two arrays joined together except that the indexing scheme is different for array leaders. The array leader words can be stored into and examined by special accessors, different from those used for the main array. The leader is always fully boxed; leader words can always hold any kind of Lisp object, regardless of the type or rank of the main part of the array.

Many high-level Explorer data types are implemented using arrays. These include hash tables, flavor data structures, and other structures created by DEFSTRUCT.

An array object is represented as a word of DTP-ARRAY. The pointer field point to an array header word of data type DTP-ARRAY-HEADER. The array may also indirectly point to its ARRAY-HEADER through an intermediate forwarding structure. An array object pointing to a a DTP-HEADER-FORWARD indicates that the array has been structure-forwarded. The HEADER-FORWARD pointer field will then point to an ARRAY-HEADER or to another HEADER-FORWARD. The structure forwarding process, which occurs when an array needs to be grown larger than its original size, is described in the Internal Storage Formats section.

The format of an ARRAY-HEADER is shown in Figure 8-1. These fields are discussed in subsequent subsections.



INDEX	<0:9>	=	Index length (if less than 1024)
NS	<10>	=	Named Structure Flag
LL	<11>	=	Long Length Flag
DIMS	<12:14>	=	Number of Dimensions
S	<15>	=	Simple Bit
D	<16>	=	Displaced Bit
L	<17>	=	Has Leader Bit
P	<18>	=	Physical Bit
ARY-TYP	<19:23>	=	Array Type

Figure 8-1 Array Header Word

8.1.1 Array Type.

The array type field indicates the type of the array. The array type determines the type of data that may be stored in the array and how this how this data is accessed.

There are many types of arrays. Some types of arrays can hold Lisp objects. These are called Q arrays. In a Q Array each memory word contains a Lisp object; hence all memory words in a Q array are boxed. Most of the Q arrays allow their cells to hold arbitrary Lisp objects, although some numeric arrays limit element contents to various numeric types.

Other types of arrays can only contain integers stored as raw bit patterns and often packed several to a 32-bit memory word. These are the bit arrays, whose data portions are unboxed. Some numeric arrays are also unboxed. Bit array elements (those which are smaller than 32 bits) are stored right-to-left within each word; the first element of an ART-4B ARRAY, for example, would be stored right-justified, beginning at bit 0 of the 32-bit word.

The array types are known by a set of symbols whose names begin with "ART-" (for ARray Type). They are summarized in Table 8-1 and are also discussed in the Explorer Lisp Reference manual. The symbol ARRAY-TYPES contains a list of all array types.

Table 8-1 Array Types

Code	Type	Boxed?	Bits/Element
0	ART-ERROR	N/A	N/A
1	ART-1B	No	1
2	ART-2B	No	2
3	ART-4B	No	4
4	ART-8B	No	8
5	ART-16B	No	16
6	ART-32B	Yes	32
7	ART-Q	Yes	32
8	ART-Q-LIST	Yes	32
9	ART-STRING	No	8
10	ART-STACK-GROUP-HEAD	Yes	32
11	ART-SPECIAL-PDL	Yes	32
12	ART-HALF-FIX	No	16
13	ART-REG-PDL	Yes	32
14	ART-DOUBLE-FLOAT	No	64 *
15	ART-SINGLE-FLOAT	No	32
16	ART-FAT-STRING	No	16
17	ART-COMPLEX-DOUBLE-FLOAT	No	128 *
18	ART-COMPLEX	No	64 *
19	ART-COMPLEX-SINGLE-FLOAT	No	64 *
20	ART-FIX	Yes	32

* These types require 2 or 4 words per element.

8.1.2 Array Leaders.

The array may optionally have an array leader which consists of a number of words BEFORE the array header. If the Has Leader Bit is set in the array header word, there is a leader present.

If there is a leader then the word immediately before the array header is a FIXNUM holding the number of array leader elements. Before that are the array leader elements, which may have any data type since any object can be stored in them. Finally, preceding the leader elements is a word of data type DTP-HEADER and header type %HEADER-TYPE-ARRAY-LEADER. The rest of the leader header word contains the total number of words in the leader (including the leader header and number-of-leader-elements words). The presence of the leader header is necessary for routines such as the garbage collecter which scan through memory in the forward direction. Note that leader elements are indexed backwards from the array header. The storage layout of an array with leader is shown in Figure 8-2.

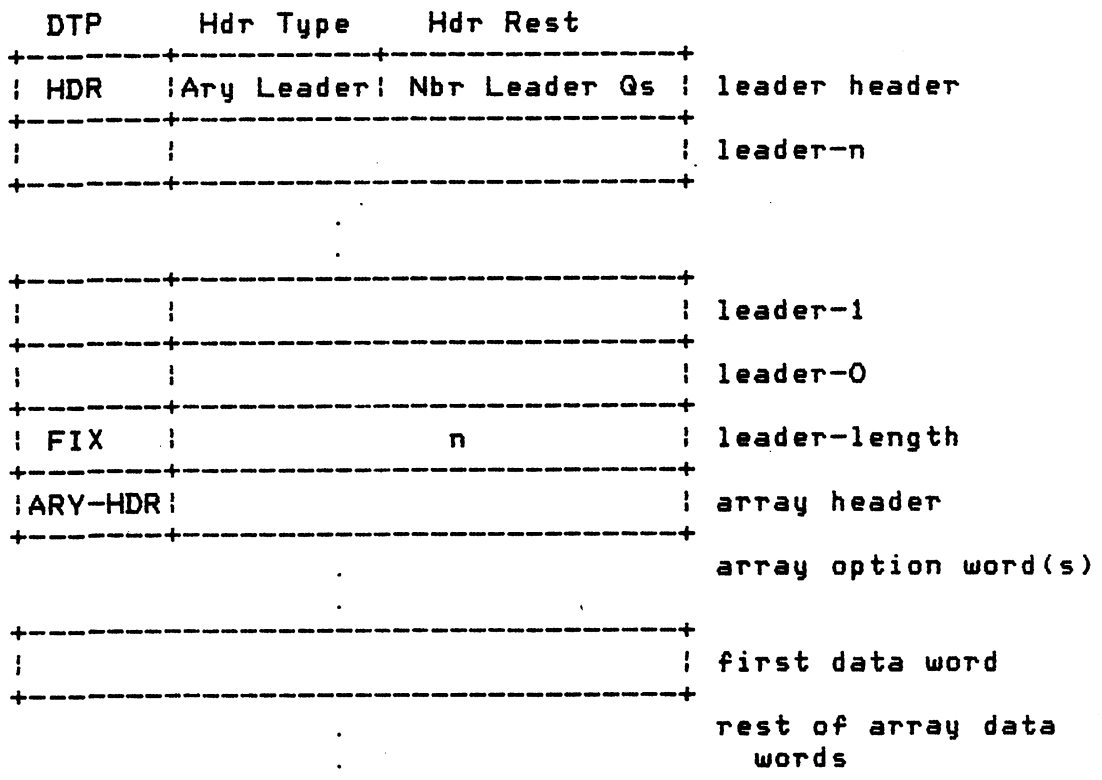


Figure 8-2 Array with Leader

8.1.3 Simple Bit.

The Simple Bit indicates that this is a simple array which can be accessed more efficiently. Simple arrays are one dimensional arrays which may or may not have leaders. They cannot be displaced, physical or long length.

Note that here the term simple is used only to distinguish a type of access that may be done by the virtual machine. It does not have same meaning as the Common Lisp data type SIMPLE-ARRAY.

8.1.4 Named Structure Flag.

The Named Structure FLAG is 1 to indicate that this array is an instance of a Named-Structure (probably defined with DEFSTRUCT with the :NAMED-STRUCTURE option). The structure name is found in array leader element 1 if the Has Leader Bit is set; otherwise it is in array element 0.

Named structures may be viewed as implementing a sort of user defined data typing facility. Certain system primitives, if handed a named structure, will obtain the name and obtain from that a function to apply, to perform the primitive.

8.1.5 Index Length Field.

The index length of an array is its number of data elements minus one. In a one dimensional array, it is the maximum value the index may take. In a multidimensional array, it is the product of the sizes of each of the dimensions.

If the index length of an array is larger than will fit in the index length field in the header, the Long Length Flag is set and the index length is stored in the next memory word.

8.1.6 Number of Dimensions Field.

The number of dimensions (rank) of the array is stored in the Number of Dimensions field. This can be a value from 0 to 7 inclusive. Zero-dimension arrays are defined to have exactly one element. Therefore, all zero dimension arrays will have a zero, but will have one element.

Elements are stored in multidimensional arrays in row-major order. An array cannot change its number of dimensions when it is grown. Each dimension after one will cause a dimension index word to be allocated.

8.1.7 Displaced Bit.

A displaced array is an array that has all its data elements in a separate, non-contiguous area of memory. A displaced array may be displaced to another array or to an arbitrary address (specified a LOCATIVE or a FIXNUM). Thus, a displaced array can be used to point at the beginning of a REGION. This is done, for example by the #'REGION-BITS array and others like it which represent the special areas/regions in low virtual memory. They are ART-Q-LIST arrays displaced to the address which is the origin of the region they represent.

Arrays displaced to a virtual or physical address are called displaced-to-address arrays. In contrast, if the array is displaced to another array, it is known as an indirect array, and the array it is displaced-to is called the indirected-to array or just an indirected array. The indirected-to array has an index length of its own. Then the index length of the indirect array appears to be MIN(x,y) where x is the index length of the indirected-to array and y is the total number of elements in the indirect array. Computing MIN prevents referencing beyond the

actual end of the indirected-to array.

A displaced index offset is used as to skip over a number of data slots in the indirected-to array or in the displaced-to-address area before element access begins. This is only tricky with an indirect array displaced to an indirected-to array. In this case, whenever the indirected-to array is referenced, it is as if that array were being referenced with an index N higher. And this index offset N is expressed in terms of the element size specified in the indirect array (not in elements of the indirected-to array). In other words, the storage of the indirected-to array is treated as if it had the same element size as the indirect array; even if the two element sizes are in fact different. Proper data alignment becomes an important consideration when using displaced index offsets.

The displaced index offset is always considered to be one dimensional; it is added after all the dimensions have been multiplied out. The resulting index is checked against the computed index length of the indirected-to array, if present.

The Displaced Bit is set in all displaced arrays. Such an array will always allocate one array option word to describe the storage displaced to, and may also allocate a displaced index offset word.

8.1.8 Physical Bit.

If this is a physical array, then both the Physical Bit and the Displaced Bit are set. Physical arrays are displaced arrays which are displaced to a NuBus physical address. Physical arrays may only be made from the bit-array types.

A physical array will always allocate one array option word to describe the physical address displaced to and may also allocate a displaced index offset word.

8.1.9 Option Words.

Between the ARRAY-HEADER word and the start of the array data elements there may be a number of array option words. If present they will occur in the following order. Each option word is discussed separately.

Non-Displaced Arrays:
- o Long Length word
- o Dimension word(s)

Displaced Arrays:
- o Dimension word(s)
- o Displaced-To word
- o Index length word
- o Displaced Index Offset word

8.1.9.1 Long Length Word.

If the total number of array elements is too big for the header index field (1024 elements or larger) the index length is stored in a long length word as a FIXNUM (limiting the maximum array size 2**24 - 1 elements). In non-displaced arrays, the long length word always follows immediately after the header word and before any dimension words.

8.1.9.2 Dimension Words.

If the array has more than one dimension, then there is a block of <number of dims minus one> words immediately after any long length word. Each dimension word holds the size of one dimension. The first dimension word contains the size of the most rapidly varying subscript (that of the last dimension in the dimension list); the second dimension word contains the size of the second most rapidly varying subscript (that of the penultimate dimension in the dimension list); and so forth until n-1 dimension words. The size of the last dimension can be computed from the N-1 dimension sizes along with the total index length in the index length field or the long length word.

8.1.9.3 Displaced Array Option Words.

In a multidimensional displaced array, the N-1 dimension words always come first, immediately after the array header. What follows is another group of 2 or 3 option words consisting of a displaced-to word, an index length word, and possibly a displaced-index-offset word. The displaced-to word is an array, locative or integer for regular displaced arrays. It and any further option words following it are considered boxed. On physical arrays, however, the displaced-to word is a 32-bit address; hence it and any words following it must be considered unboxed.

Displaced arrays must keep track of two different array sizes: their own (for computing how much actual storage this displaced array "stub" takes); and the number of elements in the displaced-to storage area. To accomodate the latter, an index length word is allocated after the displaced-to word.

The index field in the ARRAY-HEADER of the displaced array itself is used to contain the number of option words, not counting the dimension words, that this displaced array has. Thus it will always be either a FIXNUM 2 or 3.

If there is a displaced index offset, its FIXNUM word will be placed last.