LOOKUP

LOOKUP provides a mechanism for retrieving a value from an inline table.

PBASIC1: **LOOKUP** *index*, (*value0*, *value1*, *value2*, …), *variable*
PBASIC2: **LOOKUP** *index*, [*value0*, *value1*, *value2*, …], *variable*

Using **LOOKUP** the value at the index'th position will be moved to *variable*. If *index* is beyond the table bounds (table elements minus one) then *variable* is not changed. Note that in PBASIC the **LOOKUP** table is zero-indexed, i.e., the first a value in the table is at position zero. With *N* elements in the **LOOKUP** table the valid range of *index* is 0 to *N*-1.

Spin has two variants of **LOOKUP**.

Spin: *variable* := **LOOKUPZ**(index : value0, value1, value2, ...)
variable := **LOOKUP**(index : value1, value2, value3, ...)

Based on the syntax diagrams above you may be lead to believe that **LOOKUPZ** is the near-direct replacement of PBASIC **LOOKUP**. It is, and yet there is an important difference that must be considered for some programs: in PBASIC an out-of-range *index* will not change the output variable, while with Spin an out-of-range *index* will write zero to the output variable. This can cause confusion when using **LOOKUPZ** if one or more of the table elements is zero.

Consider this PBASIC code:

```
char = "?"
LOOKUP index, ["ABCDEFGHIJ"], char
```

If *index* is in the range of zero to nine then *char* will be changed to the appropriate letter, otherwise it will be left as is ("?").

Here is one possible translation to Spin:

```
if (index => 0) and (index =< 9)
  char := lookupz(index : "ABCDEFGHIJ")
else
  char := "?"
```

Note that if the table does not contain zero as one of the elements then we could also do this:

```
char := lookupz(index : "ABCDEFGHIJ")
if (result == 0)
  result := "?"
```

The behavior of Spin's **LOOKUP** is identical to **LOOKUPZ** except that the first value in the table is at index one.