Pascal II: Operand Formats (3 of 4)

Characters:

Characters, by ASCII definition, are simply integers between 0 to 255, inclusive. Characters take up one word of storage. The ASCII value of the character is stored in the least significant byte. The most significant byte is not used by Pascal and should be ignored.

```
    15  .  .  .  .  .  .  8   7  .  .  .  .  .  .  0  <== 16 bits
           unused                    ASCII
```

Example: the character "A" has an ASCII value of 65 (hexadecimal 41). The binary representation is:

```
        MSB x  x  x  x  x  x  x  x   0  1  0  0   0  0  0  1 LSB
            <----- not used------>       4   (hex)    1
```

Characters can be passed either as actual parameters (by value) or as Var parameters (by address).

Strings:

A string is a packed array of characters that can be from one to 256 bytes long. The first byte of a string always contains a number from 0 to 255; this number indicates the length of the string. One character is stored per byte, and the string ends on a word boundary--that is, if the last character in the string is the first byte of a new word, the other byte of the word is also reserved and not used by the string.

Each character of the string can be accessed in a packed array of characters; you cannot, however, access the length byte (the 0th element). Doing so causes the message "Value Range Error" to be displayed.

Example: The string "ABCD" has a length of 4. It looks like this:

```
          S[4]       S[3]       S[2]       S[1]       S[0]
   MSB 01000100   01000011   01000010   01000001   00000100 LSB
         "D"        "C"        "B"        "A"         4
```

Pascal always passes strings by address, since strings' lengths may vary.

Pointers:

Address pointers are unsigned integers that occupy 1 word of storage. Their format is identical to that of integers, except that their values may range from 0 to 65535. The value of a pointer, in this implementation of Pascal, is the memory address of the object being described.

Example: The address of AN0 (one of the annunciator ports) is hex C058 (49240 decimal). This address is stored as:

```
        MSB 1  1  0  0  0  0  0  0   0  1  0  1  1  0  0  0 LSB
            <--------> <--------->   <--------> <-------->
                C           0            5           8
```

Pointers, like integers, may be passed by value or by reference (as a Var parameter).

Long Integers:

Long integers are a special type of variable, first defined at UCSD as part of their extensions to the Pascal language. They are primarily used to handle calculations involving numbers which (a) cannot be represented accurately in floating point (real) format, and (b) are too large to store in integer format.

Long integers are stored in BCD (binary coded decimal)--one digit per nybble. One entire word is reserved for the sign of the long integer, and the variable must end on a word boundary. Four digits can be contained in one word, so the smallest definable long integer takes up two words of memory. The numbers are padded with leading zeroes when necessary to fill up the last word. The sign is 0 if positive and 255 if negative. (One byte is used for the sign.)

To illustrate the structure of long integers, let's take a specific example: the long integer -123456 takes 3 words: one for the sign, and two for the digits (since they are stored in multiples of 4). This long integer is stored in the following format:

```
            <-------- each digit is one nybble --------->
        MSB 6   5   4   3    2   1   0   0    0   0   F   F LSB
            <--  word -->    <--  word -->    < sign word >
```

A long integer should always be passed by address, since its length depends on its definition.
Apple Tech Notes

Keywords: <None>

==========

19960215 11:05:19.00
Tech Info Library Article Number: 688