# MSX Article


*Resizable arrays*

## Summary

This article aims at presenting a little trick used on sprites, which makes the use of strings to handle a resizable vector of numbers.

## 1- Introduction

An array is defined as a list containing data of the same type, disposed sequentially in memory and having a fixed size, defined at creation time.

Before learning how an array works, let's see how a Basic variable is stored in the MSX memory [1].

The variables are stored in a memory area starting from the address pointed by VARTAB (&HF6C2-&HF6C3) and ending at the address pointed by STREND (&HF6C6-&HF6C7).

Each variable stored in the memory has this basic structure [1]:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| T | N | N | V | V | V | V | V | V | V | V |

Where:

- T – Variable type:
  - 02 – integer (2 bytes).
  - 03 – string (3 bytes).
  - 04 – single precision (4 bytes).
  - 08 – double precision (8 bytes).
- N – Variable name. Only 1 or 2 characters can be used.
- V – Value. Ranges from 2 to 8 bytes.

Each time a variable is mentioned, the Basic interpreter searches for the respect name inside the variables area.

The following program in Basic illustrates how a variable is stored in memory.

```
10 DEFINT A-B
20 AB=2
30 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
40 PRINT"END  - BYTE - CHAR"
50 PRINT"------------------"
60 FOR F=0 TO 4
70 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
80 NEXT F
```

The variable “AB” is an integer type and takes 2 bytes.

After running the program, the following data is shown.

```
END  - BYTE - CHAR
------------------
80C3 -  02  -
80C4 -  41  -  A
80C5 -  42  -  B
80C6 -  02  -
80C7 -  00  -
```

A string has the following format:

| T | N | N | C | E | E |
|---|---|---|---|---|---|

Where:

- T – Variable type. The value is 3 (string).
- N – Variable name. Only 1 or 2 characters can be used.
- C – String length in bytes.
- E – Pointer to the first character.

The next program shows how the “A$” variable is stored in memory.

```
10 A$="MSX"
20 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
30 PRINT"END  - BYTE - CHAR"
40 PRINT"------------------"
50 FOR F=0 TO 5
60 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
70 NEXT F
80 C=PEEK(P+3)
90 P=PEEK(P+5)*256 + PEEK(P+4)
100 PRINT:PRINT"Reading the string:":PRINT
110 PRINT"END  - BYTE - CHAR"
120 PRINT"------------------"
130 FOR F=0 TO C-1
140 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
150 NEXT F
```

Exit:

```
END  - BYTE - CHAR
------------------
8199 -  03  -
819A -  41  -  A
819B -  00  -
819C -  03  -
819D -  09  -
819D -  80  -  ♠

Reading the string:
```

```
END  - BYTE - CHAR
------------------
8009 -  4D  -  M
800A -  53  -  S
800B -  58  -  X
```

Arrays may have one dimension or more. Arrays with two or more dimensions are called matrix. They are stored in memory as follows:

| T | N | N | C | C | D | $I_1$ | $I_1$ | ... | $I_d$ | $I_d$ | Data |
|---|---|---|---|---|---|---|---|---|---|---|---|

Where:

- T – Variable type.
- N – Variable name.
- C – Structure length, from “D” to the end of “Data”.
- D – Number of dimensions.
- $I_1$ – Size of dimension 1 (add 1 to declared size, once ranges from 0 to N).
- $I_d$ – Size of dimension d (add 1 to declared size, once ranges from 0 to N).
- Data – Vector or matrix data.

The next Basic program prints the “VT” array memory data.

```
10 DEFINT V
20 DIM VT(2,1)
30 VT(0,0)=1:VT(1,0)=2:VT(2,0)=3
40 VT(0,1)=4:VT(1,1)=5:VT(2,1)=6
50 P=PEEK(&HF6C3)*256 + PEEK(&HF6C2)
60 PRINT"END  - BYTE - CHAR"
70 PRINT"------------------"
80 FOR F=0 TO 43
90 PRINT HEX$(P+F) + " -  " + RIGHT$("00"+HEX$(PEEK(P+F)),2) + "  -  " +
CHR$(PEEK(P+F))
100 NEXT F
```

Exit:

```
END  - BYTE - CHAR
------------------
8123 -  02  -
8124 -  56  -  V
8125 -  54  -  T
8126 -  11  -
8127 -  00  -
8128 -  02  -
8129 -  02  -
812A -  00  -
812B -  03  -
812C -  00  -
```

```
812D -  01  -
812E -  00  -
812F -  02  -
8130 -  00  -
8131 -  03  -
8132 -  00  -
8133 -  04  -
8134 -  00  -
8135 -  05  -
8136 -  00  -
8137 -  06  -
8138 -  00  -
```

For this example, the structure length is 17 bytes (&H11). The structure length in this example were calculated as follows:

length = d + $I_1$ + $I_2$ + data

To calculate the data length:

17 = 1 + 2 + 2 + data
data = 17 - 5 = 12

As each cell from the matrix uses 2 bytes, we conclude that “Data” has 12/2 or 6 cells.

## 2- Modifying the array size

An array cannot have its size changed after the creation. But, when necessary, we may create a new array with the desired size and copy the content from the old array to the new one. In order to remove the garbage from memory, use the Basic command ERASE to delete the old array. E.g.: ERASE VT.

Another alternative for that is the use of resizable arrays for certain cases. As mentioned on the previous chapter, string are array of characters (1 byte). Nevertheless, they can be resized automatically through an attribution or the use of a Basic function designed for strings.

The functions in Basic that handle strings are:

- LEFT$(S, N) – returns a substring of “S” with “N” bytes length, from the left.
- RIGHT$(S, N) – returns a substring of “S” with “N” bytes length, from the right.
- MID$(S,P,N) – returns a substring of “S” with “N” bytes length, starting from “P”.
- LEN(S) – Returns the string “S” length.

Besides these functions, there are some operations like the concatenation (joint) of characters or strings, which uses the “+” operator to join two strings. E.g.:

```
10 A$ = “ABC”
20 A$ = A$ + “D”
```

The variable “A$” now holds the string “ABCD”.

In the next example, we will remove the last character of the string:

```
30 A$ = LEFT$(A$, LEN(A$)-1)
```

The advantage of using strings is the fact that they always have the exact size to store the data. Nevertheless, strings only allows to store characters in it. We must remember that characters have 1 byte size. In that case, it is only possible to store 1-byte data in each cell.

Once strings deals with characters, we must convert numeric data to character using the STR$(val) Basic command. By doing that, we are able to store 1-byte numbers in a string.

Suppose a list of 10 numbers, ranging from 1 to 10. The next Basic program will store these numbers in a string.

```
10 FOR F=1 TO 10
20 L$ = L$ + CHR$(F)
30 NEXT F
```

If you want to read a data located at position P in a string, use the Basic function MID$. Notice that the value must be converted back to number using the Basic function ASC.

```
10 FOR F=1 TO 10
20 L$ = L$ + CHR$(F)
30 NEXT F
40 FOR P=1 TO 10
50 V = ASC(MID$(L$, P, 1))
60 PRINT V
70 NEXT P
```

## 3- The trick applied on sprites [2]

The Basic instruction “SPRITE$(n) = string” defines the sprite object that appears on the screen. The variable “n” is the sprite identification, ranging from 0 to 255 at 8x8 mode and 0 to 63 at 16x16 mode. The “string” is the array that holds the sprite shape design.

Each string character contains the information about one line of the sprite. The bit 1 indicates that the pixel corresponding to that region must be drawn. In the other hand, the pixel 0 indicates that the pixel will be not drawn.

The next example shows how to configure the “string” to draw the letter “M” at 8x8 mode.

```
A$ = CHR$(&b01000010)
B$ = CHR$(&b01100110)
C$ = CHR$(&b01011010)
D$ = CHR$(&b01011010)
E$ = CHR$(&b01000010)
F$ = CHR$(&b01000010)
G$ = CHR$(&b01000010)
H$ = CHR$(&b01000010)

SPRITE$(1) = A$+B$+C$+D$+E$+F$+G$+H$
```

Notice that the bits 0 and 1 defines the drawing shape. The numbers “1” were colored in red to highlight the “M” shape.

The 8x8 sprite mode stores the data in a string with 8 bytes length, while the 16x16 mode takes 32 bytes to store the data.

## 4- Credits

This article was written by Marcelo Silveira on January 2017.
E-mail: flamar98@hotmail.com

References:

[1] – MSX Top Secret, Edison Moraes, available at: http://www.msxtop.msxall.com.
[2] – Article: “Sprites and Gravity”, Marcelo Silveira, available at: http://marmsx.msxall.com.

Note: this is a translation from the original article titled “Vetor de Tamanho Variavel”, in portuguese, written by the same author.