STRING type variables

STRING also has two subtypes, just like CHAR. The old, *"old-school"* STRING, which describes the text with ASCII characters, and WSTRING, which uses WCHAR characters with two bytes per character. Both types are suitable for storing text, which can be extremely useful for communication, especially in HMI connections.

For both types, the first two positions show the maximum length of the given STRING and the current length it has been filled with. One position equals one byte for STRING, and one word for WSTRING.

Name	Address	Display format	Monitor value
"example".tString	P#DB9.DBX0.0	String	'lama!'
	%DB9.DBB0	Hex	16#08
	%DB9.DBB1	Hex	16#05
"example".tString[1]	%DB9.DBB2	Character	'l'
"example".tString[2]	%DB9.DBB3	Character	'a'
"example".tString[3]	%DB9.DBB4	Character	'm'
"example".tString[4]	%DB9.DBB5	Character	'a'
"example".tString[5]	%DB9.DBB6	Character	'!'
"example".tString[6]	%DB9.DBB7	Character	'$00'

In the example above, taken from the PLC status, I entered the phrase "lama!" into an 8-byte STRING variable. The first two bytes contain the maximum length of the STRING (8) and the current length (5), followed by the phrase as our message.

If I change the display format to hexadecimal for the characters, I see the ASCII code for each letter.

Name	Address	Display format	Monitor value
"example".tString	P#DB9.DBX0.0	String	'lama!'
	%DB9.DBB0	Hex	16#08
	%DB9.DBB1	Hex	16#05
"example".tString[1]	%DB9.DBB2	Hex	16#6C
"example".tString[2]	%DB9.DBB3	Hex	16#61
"example".tString[3]	%DB9.DBB4	Hex	16#6D
"example".tString[4]	%DB9.DBB5	Hex	16#61
"example".tString[5]	%DB9.DBB6	Hex	16#21
"example".tString[6]	%DB9.DBB7	Hex	16#00

That is, the letter "l" is ASCII 16#6C, and "a" is ASCII 16#61, ... For **WSTRING**, this assignment appears like this:

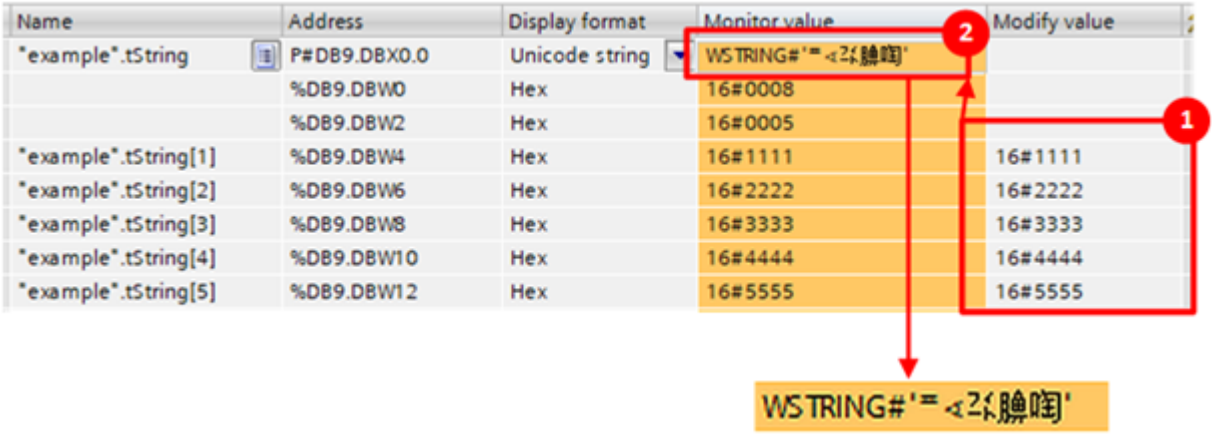
Name	Address	Display format	Monitor value
"example".tString	P#DB9.DBX0.0	Unicode string	WSTRING#'lama!'
	%DB9.DBW0	Hex	16#0008
	%DB9.DBW2	Hex	16#0005
"example".tString[1]	%DB9.DBW4	Character	'$00l'
"example".tString[2]	%DB9.DBW6	Character	'$00a'
"example".tString[3]	%DB9.DBW8	Character	'$00m'
"example".tString[4]	%DB9.DBW10	Character	'$00a'
"example".tString[5]	%DB9.DBW12	Character	'$00!'
"example".tString[6]	%DB9.DBW14	Character	'$00$00'

The *"$00I"* content type is due to the nature of UNICODE, as *"simple"* characters do not fill the entire UCS-2 space. It is clear that while we counted the positions per byte above, in this case each position occupies a word. The first two words here also contain the maximum length of the STRING (8) and the current length (5).

The same definition is given in hexadecimal form as follows:

Name	Address	Display format	Monitor value
"example".tString	P#DB9.DBX0.0	Unicode string	WSTRING#'lama!'
	%DB9.DBW0	Hex	16#0008
	%DB9.DBW2	Hex	16#0005
"example".tString[1]	%DB9.DBW4	Hex	16#006C
"example".tString[2]	%DB9.DBW6	Hex	16#0061
"example".tString[3]	%DB9.DBW8	Hex	16#006D
"example".tString[4]	%DB9.DBW10	Hex	16#0061
"example".tString[5]	%DB9.DBW12	Hex	16#0021
"example".tString[6]	%DB9.DBW14	Hex	16#0000

If we fully fill in the UCS-2 word field, we can see what the *"non-simple characters"* look like. In the first step, I entered longer codes in the word variables per character (1), and from this the "example" WSTRING (2) was displayed:



To sum it all up:

Type	**Length**	**Character encoding**	**Length (characters)**	**Example**
STRING	2 byte + text	CHAR, ASCII	0 .. 254 byte / character	'lamaPLC', STRING#'lamaPLC'
WSTRING	2 word + text	WCHAR, UNICODE	0 .. 16382 word / character	WSTRING#lamaPLC

More information: TIA Datatypes: S7 data types summary table