## 2.170 StrDigCalc - Arithmetic operations with datatype stringdig

### Usage

`StrDigCalc` is used to perform arithmetic operations (+, -, *, /, %) on two positive digit strings in the same way as numeric arithmetic operations on positive integer values.

This function can handle positive integers above 8 388 608 with exact representation.

### Basic examples

The following example illustrates the function `StrDigCalc`.

See also *More examples on page 1410*.

**Example 1**

```
res := StrDigCalc(str1, OpAdd, str2);
```

`res` is assigned the result of the addition operation on the values represented by the digital strings `str1` and `str2`.

### Return value

Data type: `stringdig`

`stringdig` is used to represent big positive integers in a string with only digits.

This data type is introduced because the data type `num` cannot handle positive integers above 8 388 608 with exact representation.

### Arguments

```
StrDigCalc (StrDig1 Operation StrDig2)
```

`StrDig1`

*String Digit 1*

Data type: `stringdig`

String representing a positive integer value.

`Operation`

*Arithmetic operator*

Data type: `opcalc`

Defines the arithmetic operation to perform on the two digit strings. Following arithmetic operatons of data type `opcalc` can be used; `OpAdd`, `OpSub`, `OpMult`, `OpDiv` and `OpMod`.

`StrDig2`

*String Digit 2*

Data type: `stringdig`

String representing a positive integer value.

*Continues on next page*

2.170 StrDigCalc - Arithmetic operations with datatype stringdig
*RobotWare - OS*
*Continued*

### Program execution

This function will:

- Check only digits 0...9 in `StrDig1` and `StrDig2`
- Convert the two digital strings to `long integers`
- Perform an arithmetic operation on the two `long integers`
- Convert the result from `long integer` to `stringdig`

### More examples

The following examples illustrate the function `StrDigCalc`.

**Example 1**

```
res := StrDigCalc(str1, OpSub, str2);
```

`res` is assigned the result of the substration operation on the values represented by the digital strings `str1` and `str2`.

**Example 2**

```
res := StrDigCalc(str1, OpMult, str2);
```

`res` is assigned the result of the multiplication operartion on the values represented by the digital strings `str1` and `str2`.

**Example 3**

```
res := StrDigCalc(str1, OpDiv, str2);
```

`res` is assigned the result of the division operation on the values represented by the digital strings `str1` and `str2`.

**Example 4**

```
res := StrDigCalc(str1, OpMod, str2);
```

`res` is assigned the result of the modulus operation on the values represented by the digital strings `str1` and `str2`.

### Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable `ERRNO` will be set to:

| Name | Cause of error |
|---|---|
| `ERR_INT_NOTVAL` | Input values not only digits or modulus by zero |
| `ERR_INT_MAXVAL` | Input value above 4294967295 |
| `ERR_CALC_OVERFLOW` | Result out of range 0...4294967295 |
| `ERR_CALC_NEG` | Negative substraction, that is, StrDig2 > StrDig1 |
| `ERR_CALC_DIVZERO` | Division by zero |

### Limitations

`StrDigCalc` only accepts strings that contain digits (characters 0...9). All other characters in `stringdig` will result in error.

This function can only handle positive integers up to 4 294 967 295.

*Continues on next page*

### Syntax

```
StrDigCalc '('
  [ StrDig1 ':=' ] < expression (IN) of stringdig > ','
  [ Operation ':=' ] < expression (IN) of opcalc > ','
  [ StrDig2 ':=' ] < expression (IN) of stringdig > ')'
```

A function with a return value of the data type `stringdig`.

### Related information

| For information about | See |
|---|---|
| Strings with only digits. | *stringdig - String with only digits on page 1685* |
| Arithmetic operators. | *opcalc - Arithmetic Operator on page 1625* |

## 2.171 StrDigCmp - Compare two strings with only digits

### Usage

`StrDigCmp` is used to compare two positive digit strings in the same way as numeric compare of positive integers.

This function can handle positive integers above 8 388 608 with exact representation.

### Basic examples

The following examples illustrate the function `StrDigCmp`.

**Example 1**

```
VAR stringdig digits1 := "1234";
VAR stringdig digits2 := "1256";
VAR bool is_equal;
is_equal := StrDigCmp(digits1, EQ, digits2);
```

The variable `is_equal` will be set to `FALSE`, because the numeric value `1234` is not equal to `1256`.

### Return value

Data type: `bool`

TRUE if the given condition is met, FALSE if not.

### Arguments

```
StrDigCmp (StrDig1 Relation StrDig2)
```

`StrDig1`

*String Digit 1*

Data type: `stringdig`

The first string with only digits to be numerical compared.

`Relation`

Data type: `opnum`

Defines how to compare the two digit strings. Following predefined constants of data type `opnum` can be used `LT, LTEQ, EQ, NOTEQ, GTEQ` or `GT`.

`StrDig2`

*String Digit 2*

Data type: `stringdig`

The second string with only digits to be numerical compared.

### Program execution

This function will:

- Check that only digits 0...9 are used in `StrDig1` and `StrDig2`
- Convert the two digital strings to `long integers`
- Numerically compare the two `long integers`

*Continues on next page*

### Error handling

The following recoverable errors are generated and can be handled in an error handler. The system variable ERRNO will be set to:

| Name | Cause of error |
|---|---|
| ERR_INT_NOTVAL | Input values not only digits |
| ERR_INT_MAXVAL | Value above 4294967295 |

### Limitations

StrDigCmp only accepts strings that contain digits (characters 0...9). All other characters in stringdig will result in error.

This function can only handle positive integers up to 4 294 967 295.

### Syntax

```
StrDigCmp '('
  [ StrDig1 ':=' ] < expression (IN) of stringdig > ','
  [ Relation ':=' ] < expression (IN) of opnum > ','
  [ StrDig2 ':=' ] < expression (IN) of stringdig > ')'
```

A function with a return value of the data type bool.

### Related information

| For information about | See |
|---|---|
| String with only digits | *stringdig - String with only digits on page 1685* |
| Comparison operators | *opnum - Comparison operator on page 1626* |
| File time information | *FileTimeDnum - Retrieve time information about a file on page 1221* |
| File modify time of the loaded module | *ModTimeDnum - Get file modify time for the loaded module on page 1302* |

## 2.172 StrFind - Searches for a character in a string

### Usage

`StrFind` (*String Find*) is used to search in a string, starting at a specified position, for a character that belongs to a specified set of characters.

### Basic examples

The following example illustrates the function `StrFind`.

**Example 1**

```
VAR num found;
found := StrFind("Robotics",1,"aeiou");
```

The variable `found` is given the value 2.

```
found := StrFind("Robotics",1,"aeiou"\NotInSet);
```

The variable `found` is given the value 1

```
found := StrFind("IRB 6400",1,STR_DIGIT);
```

The variable `found` is given the value 5.

```
found := StrFind("IRB 6400",1,STR_WHITE);
```

The variable `found` is given the value `4`.

### Return value

Data type: `num`

The character position of the first character at or past the specified position that belongs to the specified set. If no such character is found, string length +1 is returned.

### Arguments

```
StrFind (Str ChPos Set [\NotInSet])
```

`Str`

*String*

Data type: `string`

The string to search in.

`ChPos`

*Character Position*

Data type: `num`

Start character position. A runtime error is generated if the position is outside the string.

`Set`

Data type: `string`

Set of characters to test against. See also *Predefined data on page 1415*.

`[\NotInSet]`

Data type: `switch`

Search for a character not in the set of characters presented in `Set`.

*Continues on next page*

## Syntax

```
StrFind '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Set ':=' ] <expression (IN) of string>
  ['\' NotInSet ] ')'
```

A function with a return value of the data type `num`.

## Predefined data

A number of predefined string constants are available in the system and can be used together with string functions.

| Name | Character set |
|---|---|
| STR_DIGIT | <digit> ::=<br>0 \| 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 |
| STR_UPPER | <upper case letter> ::=<br>A \| B \| C \| D \| E \| F \| G \| H \| I \| J<br>\| K \| L \| M \| N \| O \| P \| Q \| R \| S \| T<br>\| U \| V \| W \| X \| Y \| Z \| À \| Á \| Â \| Ã<br>\| Ä \| Å \| Æ \| Ç \| È \| É \| Ê \| Ë \| Ì \| Í<br>\| Î \| Ï \| 1)\| Ñ \| Ò \| Ó \| Ô \| Õ \| Ö \| Ø<br>\| Ù \| Ú \| Û \| Ü \| 2) \| 3) |
| STR_LOWER | <lower case letter> ::=<br>a \| b \| c \| d \| e \| f \| g \| h \| i \| j<br>\| k \| l \| m \| n \| o \| p \| q \| r \| s \| t<br>\| u \| v \| w \| x \| y \| z \| à \| á \| â \| ã<br>\| ä \| å \| æ \| ç \| è \| é \| ê \| ë \| ì \| í<br>\| î \| ï \| 1) \| ñ \| ò \| ó \| ô \| õ \| ö \| ø<br>\| ù \| ú \| û \| ü \| 2) \| 3) \| ß \| ÿ- |
| STR_WHITE | <blank character> ::= |

## Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.173 StrLen - Gets the string length

### Usage

`StrLen` (*String Length*) is used to find the current length of a string.

### Basic examples

The following example illustrates the function `StrLen`.

**Example 1**

```
VAR num len;
len := StrLen("Robotics");
```

The variable `len` is given the value `8`.

### Return value

Data type: `num`

The number of characters in the string (>=0).

### Arguments

```
StrLen (Str)
```

`Str`

*String*

Data type: `string`

The string in which the number of characters is to be counted.

### Syntax

```
StrLen '('
  [ Str ':=' ] <expression (IN) of string> ')'
```

A function with a return value of the data type `num`.

### Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID Instructions, Functions and Data types* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID Instructions, Functions and Data types* |

## 2.174 StrMap - Maps a string

### Usage

`StrMap` (*String Mapping*) is used to create a copy of a string in which all characters are translated according to a specified mapping.

### Basic examples

The following examples illustrate the function `StrMap`.

**Example 1**

```
VAR string str;
str := StrMap("Robotics","aeiou","AEIOU");
```

The variable `str` is given the value `RObOtIcs`.

**Example 2**

```
str := StrMap("Robotics",STR_LOWER, STR_UPPER);
```

The variable `str` is given the value `ROBOTICS`.

### Return value

Data type: `string`

The string created by translating the characters in the specified string, as specified by the "from" and "to" strings. Each character from the specified string that is found in the "from" string is replaced by the character at the corresponding position in the "to" string. Characters for which no mapping is defined are copied unchanged to the resulting string.

### Arguments

```
StrMap ( Str FromMap ToMap)
```

`Str`

*String*

Data type: `string`

The string to translate.

`FromMap`

Data type: `string`

Index part of mapping. See also *Predefined data on page 1418*.

`ToMap`

Data type: `string`

Value part of mapping. See also *Predefined data on page 1418*.

### Syntax

```
StrMap '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ FromMap ':=' ] <expression (IN) of string> ','
  [ ToMap ':=' ] <expression (IN) of string> ')'
```

A function with a return value of the data type `string`.

*Continues on next page*

2.174 StrMap - Maps a string
*RobotWare - OS*
*Continued*

### Predefined data

A number of predefined string constants are available in the system and can be used together with string functions.

| Name | Character set |
|---|---|
| STR_DIGIT | <digit> ::=<br>0 \| 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 |
| STR_UPPER | <upper case letter> ::=<br>A \| B \| C \| D \| E \| F \| G \| H \| I \| J<br>\| K \| L \| M \| N \| O \| P \| Q \| R \| S \| T<br>\| U \| V \| W \| X \| Y \| Z \| À \| Á \| Â \| Ã<br>\| Ä \| Å \| Æ \| Ç \| È \| É \| Ê \| Ë \| Ì \| Í<br>\| Î \| Ï \| 1)\| Ñ \| Ò \| Ó \| Ô \| Õ \| Ö \| Ø<br>\| Ù \| Ú \| Û \| Ü \| 2) \| 3) |
| STR_LOWER | <lower case letter> ::=<br>a \| b \| c \| d \| e \| f \| g \| h \| i \| j<br>\| k \| l \| m \| n \| o \| p \| q \| r \| s \| t<br>\| u \| v \| w \| x \| y \| z \| à \| á \| â \| ã<br>\| ä \| å \| æ \| ç \| è \| é \| ê \| ë \| ì \| í<br>\| î \| ï \| 1) \| ñ \| ò \| ó \| ô \| õ \| ö \| ø<br>\| ù \| ú \| û \| ü \| 2) \| 3) \| ß \| ÿ- |
| STR_WHITE | <blank character> ::= |

### Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.175 StrMatch - Search for pattern in string

### Usage

`StrMatch` (*String Match*) is used to search in a string, starting at a specified position, for a specified pattern.

### Basic examples

The following example illustrates the function `StrMatch`.

**Example 1**

```
VAR num found;

found := StrMatch("Robotics",1,"bo");
```

The variable `found` is given the value `3`.

### Return value

Data type: `num`

The character position of the first substring, at or past the specified position, that is equal to the specified pattern string. If no such substring is found, string length +1 is returned.

### Arguments

```
StrMatch (Str ChPos Pattern)
```

`Str`

*String*

Data type: `string`

The string to search in.

`ChPos`

*Character Position*

Data type: `num`

Start character position. A runtime error is generated if the position is outside the string.

`Pattern`

Data type: `string`

Pattern string to search for.

### Syntax

```
StrMatch '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Pattern ':=' ] <expression (IN) of string> ')'
```

A function with a return value of the data type `num`.

*Continues on next page*

2.175 StrMatch - Search for pattern in string
*RobotWare - OS*
*Continued*

**Related information**

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.176 StrMemb - Checks if a character belongs to a set

### Usage

`StrMemb` (*String Member*) is used to check whether a specified character in a string belongs to a specified set of characters.

### Basic examples

The following example illustrates the function `StrMemb`.

**Example 1**

```
VAR bool memb;
memb := StrMemb("Robotics",2,"aeiou");
```

The variable `memb` is given the value `TRUE`, as `o` is a member of the set `"aeiou"`.

```
memb := StrMemb("Robotics",3,"aeiou");
```

The variable `memb` is given the value `FALSE`, as `b` is not a member of the set `"aeiou"`.

```
memb := StrMemb("S-721 68 VÄSTERÅS",3,STR_DIGIT);
```

The variable `memb` is given the value `TRUE`, as `7` is a member of the set `STR_DIGIT`.

### Return value

Data type: `bool`

`TRUE` if the character at the specified position in the specified string belongs to the specified set of characters.

### Arguments

```
StrMemb (Str ChPos Set)
```

`Str`

*String*

Data type: `string`

The string to check in.

`ChPos`

*Character Position*

Data type: `num`

The character position to check. A runtime error is generated if the position is outside the string.

`Set`

Data type: `string`

Set of characters to test against.

### Syntax

```
StrMemb '('
  [ Str ':=' ] <expression (IN) of string> ','
  [ ChPos ':=' ] <expression (IN) of num> ','
  [ Set ':=' ] <expression (IN) of string> ')'
```

*Continues on next page*

2.176 StrMemb - Checks if a character belongs to a set
*RobotWare - OS*
*Continued*

A function with a return value of the data type `bool`.

### Predefined data

A number of predefined string constants are available in the system and can be used together with string functions.

| Name | Character set |
|---|---|
| `STR_DIGIT` | <digit> ::=<br>0 \| 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 |
| `STR_UPPER` | <upper case letter> ::=<br>A \| B \| C \| D \| E \| F \| G \| H \| I \| J<br>\| K \| L \| M \| N \| O \| P \| Q \| R \| S \| T<br>\| U \| V \| W \| X \| Y \| Z \| À \| Á \| Â \| Ã<br>\| Î \| Ï \| 1)\| Ñ \| Ò \| Ó \| Ô \| Õ \| Ö \| Ø<br>\| Ù \| Ú \| Û \| Ü \| 2) \| 3) |
| `STR_LOWER` | <lower case letter> ::=<br>a \| b \| c \| d \| e \| f \| g \| h \| i \| j<br>\| k \| l \| m \| n \| o \| p \| q \| r \| s \| t<br>\| u \| v \| w \| x \| y \| z \| à \| á \| â \| ã<br>\| ä \| å \| æ \| ç \| è \| é \| ê \| ë \| ì \| í<br>\| î \| ï \| 1) \| ñ \| ò \| ó \| ô \| õ \| ö \| ø<br>\| ù \| ú \| û \| ü \| 2) \| 3) \| ß \| ÿ- |
| `STR_WHITE` | <blank character> ::= |

### Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.177 StrOrder - Checks if strings are ordered

### Usage

`StrOrder` (*String Order*) compares two strings (character by character ) and returns a boolean indicating whether the two strings are in order according to a specified character ordering sequence.

### Basic examples

The following examples illustrate the function `StrOrder`.

**Example 1**

```
VAR bool le;

le := StrOrder("FIRST","SECOND",STR_UPPER);
```

The variable `le` is given the value `TRUE`, because "F" comes before "S" in the character ordering sequence `STR_UPPER`.

**Example 2**

```
VAR bool le;

le := StrOrder("FIRST","FIRSTB",STR_UPPER);
```

The variable `le` is given the value `TRUE`, because `"FIRSTB"` has an additional character in the character ordering sequence (no character compared to `"B"`).

**Example 3**

```
VAR bool le;

le := StrOrder("FIRSTB","FIRST",STR_UPPER);
```

The variable `le` is given the value `FALSE`, because `"FIRSTB"` has an additional character in the character ordering sequence (`"B"` compared to no character).

### Return value

Data type: `bool`

`TRUE` if the first string comes before the second string (`Str1` <= `Str2`) when characters are ordered as specified.

Characters that are not included in the defined ordering are all assumed to follow the present ones.

### Arguments

```
StrOrder ( Str1 Str2 Order)
```

`Str1`

*String 1*

Data type: `string`

First string value.

`Str2`

*String 2*

*Continues on next page*

2.177 StrOrder - Checks if strings are ordered
*RobotWare - OS*
*Continued*

Data type: `string`

Second string value.

`Order`

Data type: `string`

Sequence of characters that define the ordering. See also *Predefined data on page 1424*.

## Syntax

```
StrOrder '('
  [ Str1 ':=' ] <expression (IN) of string> ','
  [ Str2 ':=' ] <expression (IN) of string> ','
  [ Order ':=' ] <expression (IN) of string> ')'
```

A function with a return value of the data type `bool`.

## Predefined data

A number of predefined string constants are available in the system and can be used together with string functions.

| Name | Character set |
|---|---|
| STR_DIGIT | <digit> ::=<br>0 \| 1 \| 2 \| 3 \| 4 \| 5 \| 6 \| 7 \| 8 \| 9 |
| STR_UPPER | <upper case letter> ::=<br>A \| B \| C \| D \| E \| F \| G \| H \| I \| J<br>\| K \| L \| M \| N \| O \| P \| Q \| R \| S \| T<br>\| U \| V \| W \| X \| Y \| Z \| À \| Á \| Â \| Ã<br>\| Ä \| Å \| Æ \| Ç \| È \| É \| Ê \| Ë \| Ì \| Í<br>\| Î \| Ï \| 1)\| Ñ \| Ò \| Ó \| Ô \| Õ \| Ö \| Ø<br>\| Ù \| Ú \| Û \| Ü \| 2) \| 3) |
| STR_LOWER | <lower case letter> ::=<br>a \| b \| c \| d \| e \| f \| g \| h \| i \| j<br>\| k \| l \| m \| n \| o \| p \| q \| r \| s \| t<br>\| u \| v \| w \| x \| y \| z \| à \| á \| â \| ã<br>\| ä \| å \| æ \| ç \| è \| é \| ê \| ë \| ì \| í<br>\| î \| ï \| 1) \| ñ \| ò \| ó \| ô \| õ \| ö \| ø<br>\| ù \| ú \| û \| ü \| 2) \| 3) \| ß \| ÿ- |
| STR_WHITE | <blank character> ::= |

## Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.178 StrPart - Finds a part of a string

### Usage

`StrPart` (*String Part)* is used to find a part of a string, as a new string.

### Basic examples

The following example illustrates the function `StrPart`.

**Example 1**

```
VAR string part;
part := StrPart("Robotics",1,5);
```

The variable `part` is given the value `"Robot"`.

### Return value

Data type: `string`

The substring of the specified string which has the specified length and starts at the specified character position.

### Arguments

```
StrPart (Str ChPos Len)
```

`Str`

*String*

Data type: `string`

The string in which a part is to be found.

`ChPos`

*Character Position*

Start character position. A runtime error is generated if the position is outside the string.

`Len`

*Length*

Data type: `num`

Length of string part. A runtime error is generated if the length is negative or greater than the length of the string, or if the substring is (partially) outside the string.

### Syntax

```
StrPart '('
[ Str ':=' ] <expression (IN) of string> ','
[ ChPos ':=' ] <expression (IN) of num> ','
[ Len ':=' ] <expression (IN) of num> ')'
```

A function with a return value of the data type `string`.

### Related information

| For information about | See |
|---|---|
| String functions | *Technical reference manual - RAPID overview* |

*Continues on next page*

2.178 StrPart - Finds a part of a string
*RobotWare - OS*
*Continued*

| For information about | See |
|---|---|
| Definition of string | *string - Strings on page 1683* |
| String values | *Technical reference manual - RAPID overview* |

## 2.179 StrToByte - Converts a string to a byte data

### Usage

`StrToByte` (*String To Byte*) is used to convert a string with a defined byte data format into a byte data.

### Basic examples

The following example illustrates the function `StrToByte`.

**Example 1**

```
VAR string con_data_buffer{5} := ["10", "AE", "176", "00001010",
     "A"];
VAR byte data_buffer{5};
data_buffer{1} := StrToByte(con_data_buffer{1});
```

The content of the array component `data_buffer{1}` will be 10 decimal after the `StrToByte ...` function.

```
data_buffer{2} := StrToByte(con_data_buffer{2}\Hex);
```

The content of the array component `data_buffer{2}` will be 174 decimal after the `StrToByte ...` function.

```
data_buffer{3} := StrToByte(con_data_buffer{3}\Okt);
```

The content of the array component `data_buffer{3}` will be 126 decimal after the `StrToByte ...` function.

```
data_buffer{4} := StrToByte(con_data_buffer{4}\Bin);
```

The content of the array component `data_buffer{4}` will be 10 decimal after the `StrToByte ...` function.

```
data_buffer{5} := StrToByte(con_data_buffer{5}\Char);
```

The content of the array component `data_buffer{5}` will be 65 decimal after the `StrToByte ...` function.

### Return value

Data type: `byte`

The result of the conversion operation in decimal representation.

### Arguments

```
StrToByte (ConStr [\Hex] | [\Okt] | [\Bin] | [\Char])
```

`ConStr`

*Convert String*

Data type: `string`

The string data to be converted.

If the optional switch argument is omitted, the string to be converted has `decimal` (Dec) format.

`[\Hex]`

*Hexadecimal*

Data type: `switch`

*Continues on next page*

2.179 StrToByte - Converts a string to a byte data
*RobotWare - OS*
*Continued*

The string to be converted has `hexadecimal` format.

`[\Okt]`

*Octal*

Data type: `switch`

The string to be converted has `octal` format.

`[\Bin]`

*Binary*

Data type: `switch`

The string to be converted has `binary` format.

`[\Char]`

*Character*

Data type: `switch`

The string to be converted has `ASCII` character format.

## Limitations

Depending on the format of the string to be converted, the following string data is valid:

| Format | String length | Range |
|---|---|---|
| Dec .....: ’0’ - ’9’ | 3 | "0" - "255" |
| Hex .....: ’0’ - ’9’, ’a’ -’f’, ’A’ - ’F’ | 2 | "0" - "FF" |
| Okt .....: ’0’ - ’7’ | 3 | "0" - "377" |
| Bin .....: ’0’ - ’1’ | 8 | "0" - "11111111" |
| Char ....: Any ASCII character | 1 | One ASCII char |

RAPID character codes (for example, “\07” for BEL control character) can be used as arguments in `ConStr`.

## Syntax

```
StrToByte '('
  [ConStr ':=' ] <expression (IN) of string>
  ['\' Hex ] | ['\' Okt] | ['\' Bin] | ['\' Char]
  ')'
```

A function with a return value of the data type `byte`.

## Related information

| For information about | See |
|---|---|
| Convert a byte to a string data | *ByteToStr - Converts a byte to a string data on page 1139* |
| Other bit (byte) functions | *Technical reference manual - RAPID overview* |
| Other string functions | *Technical reference manual - RAPID overview* |

## 2.180 StrToVal - Converts a string to a value

### Usage

`StrToVal` (*String To Value*) is used to convert a string to a value of any data type.

### Basic examples

The following example illustrates the function `StrToVal`.

See also *More examples on page 1429*.

**Example 1**

```
VAR bool ok;
VAR num nval;
ok := StrToVal("3.85",nval);
```

The variable `ok` is given the value `TRUE` and `nval` is given the value `3.85`.

### Return value

Data type: `bool`

`TRUE` if the requested conversion succeeded, `FALSE` otherwise.

### Arguments

```
StrToVal ( Str Val )
```

`Str`

*String*

Data type: `string`

A string value containing literal data with format corresponding to the data type used in argument `Val`. Valid format as for RAPID literal aggregates.

`Val`

*Value*

Data type: `ANYTYPE`

Name of the variable or persistent of any data type for storage of the result from the conversion.

All type of value data with structure atomic, record, record component, array or array element can be used. The data is unchanged if the requested conversion failed because the format don't correspond to the data used in argument `Str`.

### More examples

More examples of the function `StrToVal` are illustrated below.

**Example 1**

```
VAR string str15 := "[600, 500, 225.3]";
VAR bool ok;
VAR pos pos15;

ok := StrToVal(str15,pos15);
```

*Continues on next page*