LamaPLC: Simatic S7 SCL commands: Trigonometric / math functions

ABS

The function returns the value of the “ABS” (absolute value) math function.

_FC_ Result := **ABS** (Value);

Value: function input (integers, floating-point numbers)
Result: the return value of the function (integers, floating-point numbers)

```
2
3  #result_real := ABS(-123456.789);
4
5  #result_lreal := ABS(-123456.789);
6
7  #result_dint := ABS(-123456.789);
8
9  #result_int := ABS(-123456.789);
10
```

	#result_real	123456.8
	#result_lreal	123456.789
	#result_dint	123457
	#result_int	30409

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete. For INT, the conversion overflowed.

>> Back to LamaPLC main menu (SCL commands)

ABS_x

On the TIA portal, you can specify the type of variable used for the function by entering ABS_:INT, DINT, SINT, LINT, REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

COS / ACOS

The function returns the value of the “COS” (cosie) or “ACOS” (arccosine) trigonometric function.

_FC_ Result := **COS** (Value);
_FC_ Result := **ACOS** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
#result_real := COS(0.8);

#result_lreal := COS(0.8);

#result_int := COS(0.8) * 100;
```

#result_real	0.6967067
#result_lreal	0.696706709347165
#result_int	70
COS	0.696706709347165

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

COS_x / ACOS_x

On the TIA portal, you can specify the type of variable used for the function by entering COS_: REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

EXP

The function returns the value of the “EXP” (exponent from the base e (e = 2.718282)) math function.

_FC_ Result := **EXP** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```

#result_real := EXP(0.8);

#result_lreal := EXP(0.8);

#result_int := EXP(0.8) * 100;
```

#result_real	2.225541
#result_lreal	2.22554092849247
#result_int	223
EXP	2.22554092849247

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

EXP_x

On the TIA portal, you can specify the type of variable used for the function by entering EXP_: REAL, LREAL.

PLCs: S7-1500, S7-1200, S7-400, S7-300

>> Back to LamaPLC main menu (SCL commands)

FRAC

The function returns the value of the “FRAC” (fraction) math function.

_FC_ Result := **FRAC** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
18  #result_real:= FRAC(123.4567);
19
20  #result_lreal := FRAC(123.4567);
21
22  #result_int := FRAC(123.4567) * 10;
```

▶	#result_real	0.4567
▶	#result_lreal	0.456699999999998
▶	#result_int	5

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

FRAC_x

On the TIA portal, you can specify the type of variable used for the function by entering FRAC_: REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

LIMIT

The “Set limit value” instruction limits the value of the parameter IN to the values of the parameters MN and MX. The value of the parameter MN may not be greater than the value of the parameter MX.

_FC_ Result := **LIMIT** (MN := minimum, IN := input, MAX := maximum, ENO ⇒ operation enable);

MN, IN, MX, Result: If the value of the IN parameter fulfills the condition MN ⇐ IN ⇐ MX, it is returned as the result of the instruction. If the condition is not fulfilled and the IN input value is less than the MN low limit, the value of the MN parameter is returned as the result. If the high limit MX is exceeded, the value of the MX parameter is returned as the result. (Integers, S7 Times, Date types, Floating-point numbers)

ENO: If the value at the MN input is greater than at the MX input, the result is the value specified at the IN parameter and the enable output ENO (BOOL) is “FALSE” (see 2nd example).

In the example below, the input (66) is greater than the maximum (44), so the result is the maximum:

```
1
2  #in_1_int := 33;
3  #in_2_int := 66;
4  #in_3_int := 44;
5
6  #result_int := LIMIT(MN := #in_1_int,
7                       IN := #in_2_int,
8                       MX := #in_3_int,
9                       ENO => ENO);
10
11 IF ENO THEN  // no error
12     #result_int := #no_error;
13 ELSE
14     #result_int := #error;
15 END_IF;
```

#in_1_int	33
#in_2_int	66
#in_3_int	44
#result_int	44
#in_2_int	66
#in_3_int	44
ENO	TRUE
Result	TRUE
#result_int	1
#result_int	

In the example below, the maximum (44) is less than the minimum (55), so the operation is invalid (ENO = FALSE):

```
1
2  #in_1_int := 55;
3  #in_2_int := 66;
4  #in_3_int := 44;
5
6  #result_int := LIMIT(MN := #in_1_int,
7                       IN := #in_2_int,
8                       MX := #in_3_int,
9                       ENO => ENO);
10
11 IF ENO THEN  // no error
12     #result_int := #no_error;
13 ELSE
14     #result_int := #error;
15 END_IF;
```

#in_1_int	55
#in_2_int	66
#in_3_int	44
#result_int	66
#in_2_int	66
#in_3_int	44
ENO	FALSE
Result	FALSE
#result_int	
#result_int	2

>> Back to LamaPLC main menu (SCL commands)

LIMIT_x

On the TIA portal, you can specify the types of variables used for limiting by entering LIMIT_x: Integers, S7 Times, Date types, Floating-point numbers.

Example:

```
#in_1_int := 33;
#in_2_int := 22;
#in_3_int := 44;

#result_int :=
LIMIT_INT (MN := #in_1_int,
           IN := #in_2_int,
           MX := #in_3_int,
           ENO => ENO);

IF ENO THEN  // no error
    #result_int := #no_error;
ELSE
    #result_int := #error;
END_IF;
```

#in_1_int	33
#in_2_int	22
#in_3_int	44
#result_int	33
LIMIT_INT	33
#in_2_int	22
#in_3_int	44
ENO	TRUE
Result	TRUE
#result_int	1
#result_int	

>> Back to LamaPLC main menu (SCL commands)

LN

The function returns the value of the “LN” (natural logarithm to the base e (e = 2.718282)) math function.

_FC_ Result := **LN** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
#result_real:= LN(0.8);

#result_lreal := LN(0.8);

#result_int := LN(0.8) * 10;
```

#result_real	-0.2231435
#result_lreal	-0.22314355131421
#result_int	-2

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

LN_x

On the TIA portal, you can specify the type of variable used for the function by entering LN_: REAL, LREAL.

PLCs: S7-1500, S7-1200, S7-400, S7-300

>> Back to LamaPLC main menu (SCL commands)

MAX

Get maximum

- A minimum of two and a maximum of 32 inputs can be specified at the instruction

_FC_ Result := **MAX** (IN1 := input 1, IN2 := input 2, IN3 := input 3);

input [2..32]: input values (Integers, S7 Times, Date types, Floating-point numbers)
Result: Get maximum (Integers, S7 Times, Date types, Floating-point numbers)

```
#in_1_lreal := 123456.789;
#in_2_lreal := 123456.788;
#in_3_lreal := 123456.787;

#result_lreal := MAX(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);

#result_real := MAX(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);

#result_dint := MAX(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);
```

#in_1_lreal	123456.789
#in_2_lreal	123456.788
#in_3_lreal	123456.787
#result_lreal	123456.789
#result_real	123456.8
#result_dint	123457

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

Example of DT type:

```
#in_1_dt := DT#2022-12-31-23:59:59.999;
#in_2_dt := DT#2022-04-11-15:12:00.00;
#in_2_dt := DT#2022-07-18-21:21:00.00;

#result_dt := MAX(IN1 := #in_1_dt, IN2 := #in_2_dt, IN3 := #in_3_dt);
```

#in_1_dt	DT#2022-12-31-23:59:59.999
#in_2_dt	DT#2022-04-11-15:12:00
#in_2_dt	DT#2022-07-18-21:21:00
#result_dt	DT#2022-12-31-23:59:59.999

>> Back to LamaPLC main menu (SCL commands)

MAX_x

On the TIA portal, you can specify the type of variable used for the function by entering MAX_:
Integers, S7 Times, Date types, Floating-point numbers

>> Back to LamaPLC main menu (SCL commands)

MIN

Get minimum.

- A minimum of two and a maximum of 32 inputs can be specified at the instruction

_FC_ Result := **MIN** (IN1 := input 1, IN2 := input 2, IN3 := input 3);

input [2..32]: input values (Integers, S7 Times, Date types, Floating-point numbers)
Result: Get minimum (Integers, S7 Times, Date types, Floating-point numbers)

```
#in_1_lreal := 123456.789;
#in_2_lreal := 123456.788;
#in_3_lreal := 123456.787;

#result_lreal := MIN(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);

#result_real := MIN(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);

#result_dint := MIN(IN1 := #in_1_lreal, IN2 := #in_2_lreal, IN3 := #in_3_lreal);
```

#in_1_lreal	123456.789
#in_2_lreal	123456.788
#in_3_lreal	123456.787
#result_lreal	123456.787
#result_real	123456.8
#result_dint	123457

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

MIN_x

On the TIA portal, you can specify the type of variable used for the function by entering MIN_: Integers, S7 Times, Date types, Floating-point numbers

>> Back to LamaPLC main menu (SCL commands)

SIN / ASIN

The function returns the value of the “SIN” (sinus) / “ASIN” (arcsine) trigonometric function.

_FC_ Result := **SIN** (Value);
_FC_ Result := **ASIN** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
#result_real := SIN(0.8);

#result_lreal := SIN(0.8);

#result_int := SIN(0.8) * 100;
```

#result_real	0.7173561
#result_lreal	0.717356090899523
#result_int	72
SIN	0.717356090899523

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

SIN_x / ASIN_x

On the TIA portal, you can specify the type of variable used for the function by entering SIN_: REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

SQR

The function returns the value of the “SQR” (square) math function.

_FC_ Result := **SQR** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
2
3  #result_real := SQR(0.3);
4
5  #result_lreal := SQR(0.3);
6
7  #result_int := SQR(0.3);
8
9  #result_int := SQR(0.3) * 100;
10
```

#result_real	0.09
#result_lreal	0.09
#result_int	0
#result_int	9

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

SQR_x

On the TIA portal, you can specify the type of variable used for the function by entering SQR_: REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

SQRT

The function returns the value of the “SQRT” (square root) math function.

_FC_ Result := **SQRT** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

SQRT_x

On the TIA portal, you can specify the type of variable used for the function by entering SQRT_: REAL, LREAL.

```
17
18  #result_real:= SQRT_REAL(12345.56);
```

▶	#result_real	111.1106

>> Back to LamaPLC main menu (SCL commands)

TAN / ATAN

The function returns the value of the “TAN” (tangent) / “ATAN” (arctangent) trigonometric function.

_FC_ Result := **TAN** (Value);
_FC_ Result := **ATAN** (Value);

Value: function input (Floating-point numbers)
Result: the return value of the function (Floating-point numbers)

```
3  #result_real := TAN(0.8);
4
5  #result_lreal := TAN(0.8);
6
7  #result_int := TAN(0.8) * 100;
8
9
```

▶	#result_real	1.029639
▶	#result_lreal	1.02963855705036
▶	#result_int	103
	TAN	1.02963855705036

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

TAN_x / ATAN_x

On the TIA portal, you can specify the type of variable used for the function by entering TAN_ / ATAN_: REAL, LREAL.

>> Back to LamaPLC main menu (SCL commands)

EXP

The function returns the value of the “EXP” (exponent from the base e (e = 2.718282)) math function.

Result := EXP(Value);

Result: the return value of the function
Value: function input

```
#result_real := EXP(0.8);

#result_lreal := EXP(0.8);

#result_int := EXP(0.8) * 100;
```

#result_real	2.225541
#result_lreal	2.22554092849247
#result_int	223
EXP	2.22554092849247

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

COS

The function returns the value of the “COS” (cosinus) trigonometric function.

Result := COS(Value);

Result: the return value of the function
Value: function input

```
#result_real := COS(0.8);

#result_lreal := COS(0.8);

#result_int := COS(0.8) * 100;
```

#result_real	0.6967067
#result_lreal	0.696706709347165
#result_int	70
COS	0.696706709347165

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

SIN

The function returns the value of the “SIN” (sinus) trigonometric function.

Result := SIN(Value);

Result: the return value of the function
Value: function input

```
#result_real := SIN(0.8);

#result_lreal := SIN(0.8);

#result_int := SIN(0.8) * 100;
```

#result_real	0.7173561
#result_lreal	0.717356090899523
#result_int	72
SIN	0.717356090899523

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

>> Back to LamaPLC main menu (SCL commands)

TAN

The function returns the value of the “TAN” (tangent) trigonometric function.

Result := TAN(Value);

Result: the return value of the function
Value: function input

```
3  #result_real := TAN(0.8);
4
5  #result_lreal := TAN(0.8);
6
7  #result_int := TAN(0.8) * 100;
8
9
```

#result_real	1.029639
#result_lreal	1.02963855705036
#result_int	103
TAN	1.02963855705036

A yellow underline in the code indicates that the result of the function is not completely accurate for REAL and INT types. For LREAL type, precision is complete.

Simatic, SCL, TIA, Math, ABS, COS/ACOS, ACOS, EXP, FRAC, LIMIT, LN, MAX, MIN, SIN, ASIN, SQR, SQRT, TAN, ATAN

This page has been accessed for: Today: 1, Until now: 3

From:
http://lamaplc.com/ - **lamaPLC**

Permanent link:
http://lamaplc.com/doku.php?id=simatic:scl_commands_math

Last update: **2023/06/09 19:25**