Expression syntax

1 Introduction

Arbitrary user-defined expressions can be used for specifying, in the form of a time-varying symbolic function, the value of various parameters of many control devices. Typical examples are the “Math function” option in the f(u) device and the “Function” option in the History data tab of a control device. This document describes the syntax to be used when specifying a control device user-defined expression.

2 Expression

A control device data expression is an arbitrary combination of operators, numerical values, named values, and functions, assembled according to a set of formatting rules.

3 Formatting

Expressions are assembled using the following formatting rules:

- blanks are optional and may appear anywhere between elements of an expression

  exception: a blank separates a literal operator from an operand in the absence of other punctuation
  example: NOT b a blank is necessary between NOT and b
  NOT(b>3) no blank is needed between NOT and b

- multiple blanks are interpreted as single
- interpretation of keywords and names is case-sensitive

Expressions are entered through the generic text area boxes appearing in device data forms. An example with the f(u) device is shown in Figure 1. The following rules must be enforced when using such text area boxes:

- Although the text area box allows entering any number of characters without wrapping, it is strongly recommended to apply a manual wrap (press enter key) to force the appearance of the entire expression within the box, so that the horizontal elevator bar is not activated.
- Internally the data lines are automatically wrapped not to exceed a limit of 90 characters per Netlist line. The wrapping procedure is however applied blindly and may create syntax errors, that is why manual wrapping is strongly recommended in addition to being more readable.
- The trailing blanks of a line are automatically eliminated during the Netlist creation. The leading blanks are kept. When a manual wrap is applied, it must account for these conditions to present the correct syntax. If the expression is “NOT b” and the enter key is pressed after the blank character following “NOT” then the following line must start with a blank character. This is not needed in cases where

blanks are optional since in such cases the syntax decoder can automatically locate boundaries between elements of an expression.

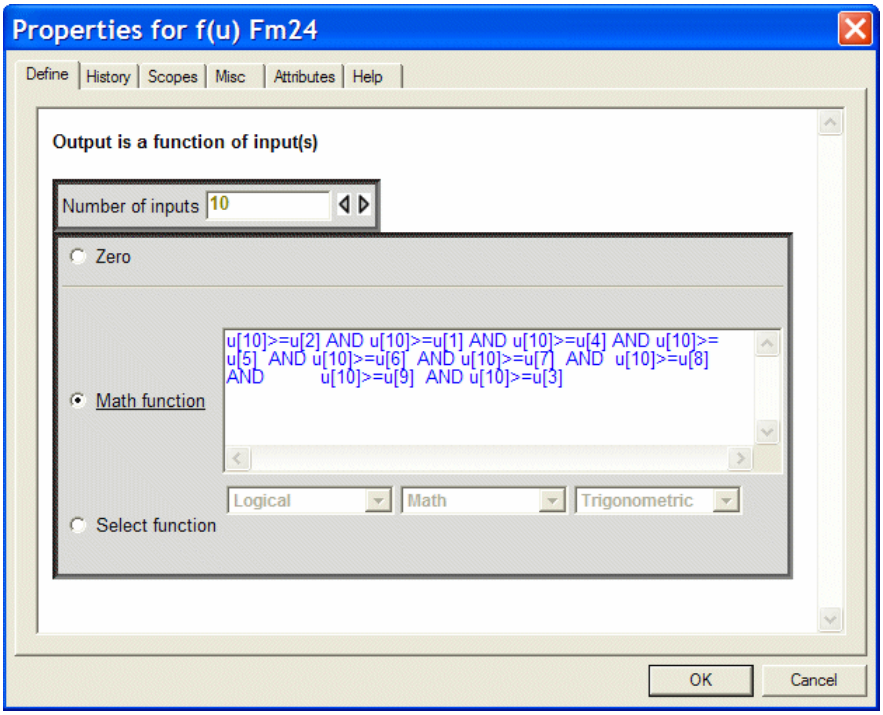

Figure 1 Expression usage example

4 Operators

Control device data expressions can use the following unary and binary operators:

operator	*description*	*use example*	*priority*	*type*
+	unary plus	+x	8	unary left
-	unary minus	-x	8	unary left
**, ^	power	x ** y	7	binary
MODULO	modulo	x MODULO y	7	binary
ATAN2	atan2	y ATAN2 x	7	binary
*	multiply	x * y	6	binary
/	divide	x / y	6	binary
+	add	x + y	5	binary
-	subtract	x - y	5	binary
==, .EQ.	is equal	x == y	4	binary
/=, ~=, .NE.	is equal	x ~= y	4	binary
>=, .GE.	is greater or equal	x >=y	4	binary
>, .GT.	is greater	x > y	4	binary
<=, .LE.	is less or equal	x <= y	4	binary
<, .LT.	is less	x < y	4	binary
NOT, ~, .NOT.	unary logical not	~x	3	unary left
AND, .AND.	logical and	x AND y	2	binary
NOR	logical nor	x NOR y	2	binary
OR, .OR.	logical or	x OR y	1	binary
NAND	logical nand	x NAND y	1	binary
XOR	logical xor	x XOR y	1	binary
(	start sub-expression	(expr)	0	unary left
)	end sub-expression		0	unary right

- parentheses can be used for modifying the default precedence order
- in the absence of parentheses, operators with higher precedence are applied first
  ex: a * -b + c is calculated as (a*(-b)) + c
- conversion between numerical and logical value is applied automatically as needed

>0	is converted to	true
<=0	is converted to	false
true	is converted to	1
false	is converted to	0

5 Numerical values

In an expression, any of the following representations can be used for specifying a numerical value:

value type	*value expression*	*examples*
integer_value	one or more digits 0-9	8, 026
signed_integer_value	[sign] integer_value	8, -32
decimal_value	[integer_value] [. [integer_value]]	8, 8., 8.03, .03
signed_decimal_value	[sign] decimal_value	8.03, -8.03
E_value	decimal_value [E signed_integer_value]	8.03E-32

- no embedded blanks are allowed in the above character sequences
- the [ ] notation indicates an optional element of a sequence

6 Named values

In a control device data expression (also called text-area expression), symbolic references to a value can be used in the following forms:

reference	*refers to*	*example*
signal name	value of any signal visible in a the local circuit	2*a – SQRT(b)
parameter name	value of any parameter visible in the local subcircuit	0.5*COS(#omega#*t)
input name	value of any input to an f(u)	2*u[1] – SQRT(u[2])

An example of reference to a local circuit signal is given by the illustrative design of Figure 2. The CSIG of control device C1 is used in lim1 for creating a limiter function f(t). The limiter data of lim1 is shown in Figure 3.

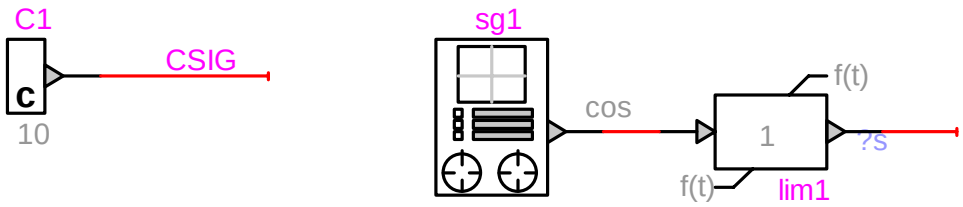

Figure 2 Using a reference to a local circuit signal

The low limit is set to –CSIG-1 and the high limit is set to CSIG+1. This is legal if the CSIG signal is available with its name (visible) within the scope of the current circuit or subcircuit.

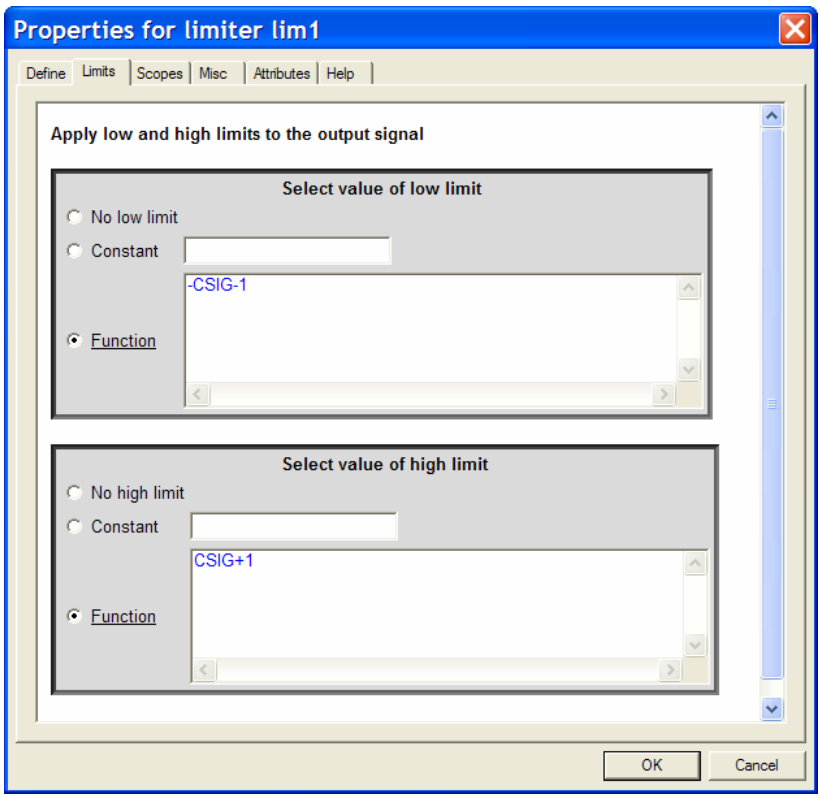

Figure 3 The lim1 data for Limits

It is very important to remember that only the visible (Visible checkbox in right-click Name menu) signal names are controlled by the user and thus can be properly referenced. When a signal name is invisible in the design picture it is allowed to use its default naming method. In the current version of EMTPWorks, its name may be modified during the internal signal name keeping procedures and consequently a reference to its name will be lost and become illegal.

The example of Figure 4 demonstrates an illegal reference condition in a subnetwork. The control device lim2 is naming the signal INCONTROL in its Limits data. The INCONTROL signal is entering into the subnetwork through the INCONTROL pin and it is connected to the device Gain1 input. Since during data decoding INCONTROL will loose its name and take the name of signal connected to the pin INCONTROL from the level above this subnetwork, it will returned as not found when decoding the data of lim2. To make INCONTROL value become available for referencing within the current subcircuit scope, the user can connect it to a Gain device with a gain of 1. The signal exciting the Gain device has a unique name and it is within the local scope.

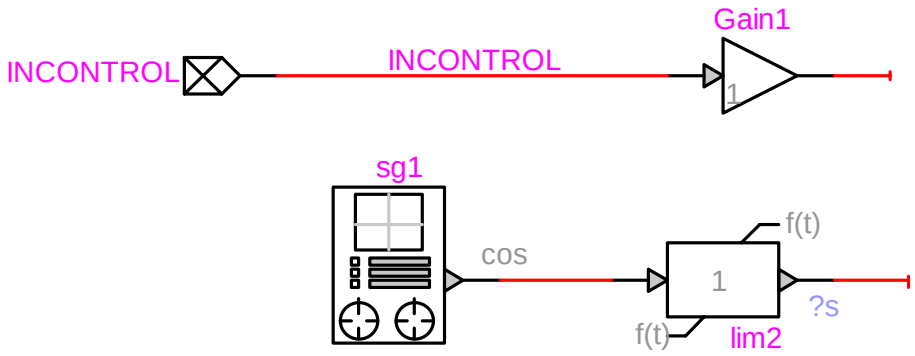

Figure 4 Illegal named value usage

There is however an exception to this condition. This is shown in Figure 5. This time the INCONTROL signal is floating, it is not connected to any control device. EMTP keeps an internal list of floating signal names and their connectivity, so it is able to find INCONTROL and connect it to its parent signal from the subcircuit above. This is also true for several subcircuit levels.

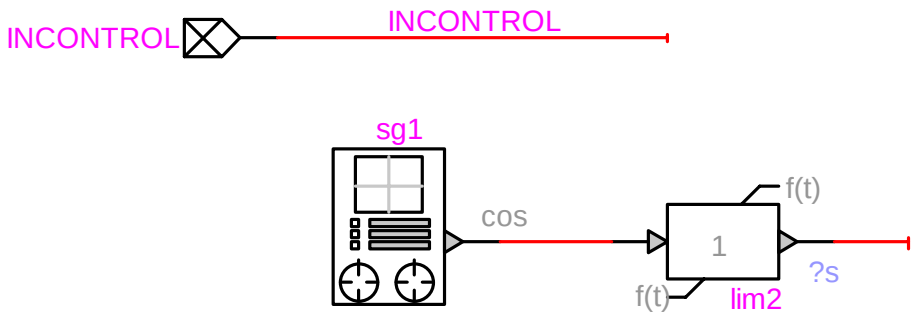

Figure 5 Legal named value usage for a floating pin signal

7 Pre-defined named values

In a control device data expression, references can be made by name to the following pre-defined values:

name	*description*
pi	the value pi
two_pi	the value pi*2
half_pi	the value pi/2
one_third_pi	the value pi/3
two_third_pi	the value pi*2/3
one_fourth_pi	the value pi/4
one_third	the value 1/3
two_third	the value 2/3
infinity	a large value
t	the local value of the simulation time
dt	the local value of the simulation time step

8 Pre-defined functions

In a control device data expression, references can be made by name to the following pre-defined functions:

function	*description*
SIN()	sine(rad)
COS()	cosine(rad)
TAN()	tangent(rad)
COTAN()	cotangent(rad)
ASIN()	arc sine, returns rad
ACOS()	arc cosine, returns rad
ATAN()	arc tangent, returns rad
SINH()	hyperbolic sine(rad)
COSH()	hyperbolic cosine(rad)
TANH()	hyperbolic tangent(rad)
ABS()	absolute value
SQRT()	square root
EXP()	exponential
LOG()	log natural
LOG10()	log base 10
CLEAN_ZERO()	when near zero, snap to zero
CLEAN()	when near integer, snap to integer
TRUNC()	truncate to integer
RECIP()	multiplicative inverse, limited to (-inf,+inf)
DEG()	radians to degrees
RAD()	degrees to radians
SIGN()	sign (-1, 0, +1)
NOT()	logical not function

- blanks can be inserted between the name of the function and the following "("

9 Pre-defined functions applied to all inputs of an f(u)

In the output expression of an f(u) device, the following notation indicates that the specified function is applied to all inputs at once, without having to specify each input individually:

function	*description*
[SUM]	sum of all inputs
[PROD]	product of all inputs
[AND]	AND of all inputs
[OR]	OR of all inputs
[XOR]	XOR of all inputs
[NAND]	NAND of all inputs
[NOR]	NOR of all inputs
[MIN]	minimum of all inputs
[MAX]	maximum of all inputs
[NORM]	Euclidian norm of all inputs

- ex: 2*[SUM] – [MIN] – [MAX]