

Model Interchange Format (MIF)

File: 58MIF.DOC
Created: August 10, 1995
Revised: September 12, 1995
Pages: 35
Author: Magne Myrtveit
ModellData AS
Status: DRAFT

1. Acknowledgments

I want to thank the System Dynamics Society for encouraging the definition and use of this standard.

I also thank Bob Eberlein for valuable input to the definition of the standard.

2. Background

The MIF standard is developed as a software independent way of representing dynamic simulation models. Typical uses of MIF will be for storing and transferring models. Examples include:

- Creation of model libraries
- Inclusion of model disks with text books
- Transfer of models using e-mail
- Automatic creation of models from other sources
- Automatic conversion of models to other formats, e.g. for inclusion in or use by other systems (e.g., authoring tools)

3. Requirements

MIF is designed to fill the following requirements.

- The bulk of existing system dynamic (SD) models should be possible to represent.
- Software packages should be allowed to include native information in MIF files. Native information is used to represent information that is not defined by the standard.
- MIF should be both backward and forward compatible. This means that models stored in a new version of MIF can be loaded into a tool supporting only an older MIF version, and vice versa.
- MIF files should transfer easily across (wide area) networks.

- MIF files should transfer easily across operating systems and hardware platforms, for example between Macintosh and MS-Windows.
- The MIF standard must allow for future expansions.
- Systems must be allowed to support only parts of the standard.
- MIF models should be possible to include in clipboard transfers, allowing for copy and paste between tools.

As a note, MIF is primarily not designed to be easily readable by humans. Visual appearance of MIF models (layout) is therefore not important to the standard.

4. Design guidelines

- The MIF standard is defined as a collection of features. A given model may use a subset – a given software may support a subset.
- Data in a MIF file is tagged. A tag is a label identifying information. Future expansion is done by adding more tags. Native information is included using tag names starting with an underscore.
- A MIF file consists of 7-bit ASCII text only¹.

5. Categories of SD models

The following existing software packages are referred to as a background for defining the standard:

Dynamo – Professional Dynamo Plus, version ? from Pugh Roberts.

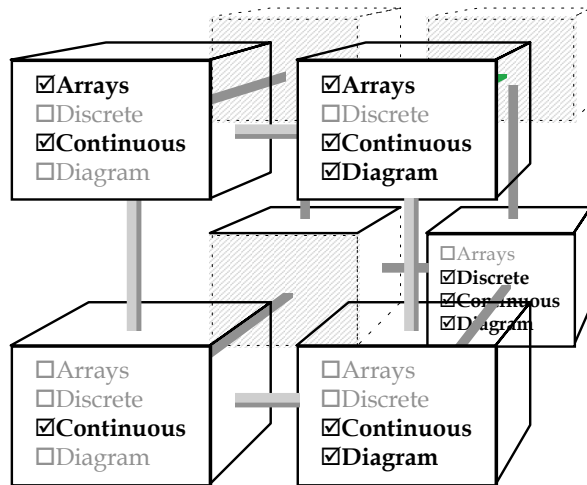
Powersim – Powersim version 2.0 from ModellData.

Stella – Refers to Stella version ? and Ithink version from High Performance Systems.

Vensim – Vensim version ? from Ventana.

¹Other characters are stored as escape sequences, as described in *Syntax definition of MIF*, page 27.

Current system dynamics models can be categorized in three dimensions, according to the figure below:



The left part holds models without diagram information, while the right hand side holds models with diagrams.

The bottom rows hold scalar models, while the top rows hold models using arrays.

The front holds continuous models, while the back holds models that also contain discrete variables (ovens, queues, conveyors).

Most current SD models fit into one of the five groups, listed below:

Group	Attributes				
	Examples	Continuous	Arrays	Diagram	Discrete
1	Scalar Dynamo models	●			
2	Dynamo models w/arrays	●	●		
3	Powersim, Stella, Vensim	●		●	
4	Stella w/queues, conveyors, ovens	●		●	●
5	Powersim, Vensim	●	●	●	

Group number 3 contains the bulk of existing models. By stripping off the diagram information, also the Dynamo software can *load* these models. Powersim, Stella and Vensim can both *load* and *store* models in this group.

For the above reasons version 1 of the MIF standard is defined for continuous, scalar models with graphical diagram information.

6. Comparison of SD tools

In the following the software tools Dynamo, Powersim, Stella and Vensim are compared in order to

identify a common denominator that can be the basis for version 1 of the MIF standard. Part of the discussion does not include Dynamo, as it does not include a graphical editor to create models.

6.1 Documents, views, windows

All the tools have an equations view (text view). In addition one or more graphic visualizations (diagram views) can be present (none for Dynamo).

Powersim has only one diagram view per model, but several windows can be opened on the same view. Each window has its own viewing options.

Stella has a stock and flow view and a mapping view. In the mapping view sectors and bundled connectors/flows can be displayed. The views of Stella are different graphical representations (diagrams) of the model.

Vensim has an arbitrary number of views. Each view is edited separately. As with Stella, each view is a separate diagram representation of a model.

The figure below displays a structure capturing the way documents are organized for all four tools.

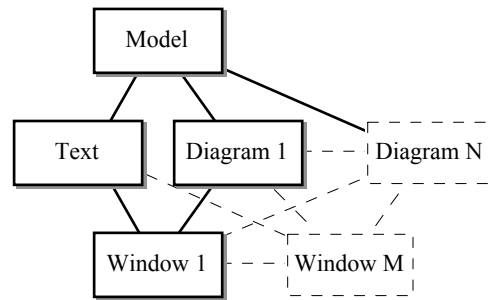
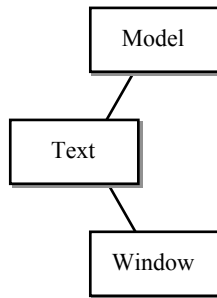


Figure 1: Structure of a model

Text and diagrams are “views” – that is, ways to present (visualize) a model in various ways. In particular, each diagram contains a set of symbols that used to represent static or dynamic information about a model. A window is used to display information contained in a view. Windows can filter information in various ways, for example by hiding certain symbol types. The user can switch between views to be displayed in a window.

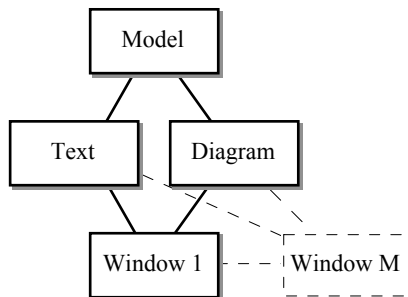
6.1.1 Dynamo documents



Dynamo does not have a graphical diagram editor, and only one window can be opened on a model. The figure to the left displays the structure of this system. It is easy to see how the structure is a special case of the structure in Figure 1.

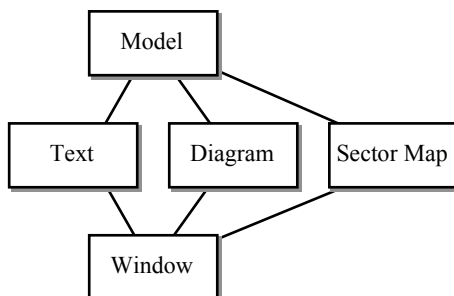
6.1.2 Powersim documents

Powersim has a textual and a graphical representation of a model. One or more windows can be opened. Each window can be switched between text view (equations) or diagram view. View options for both equations and diagram are stored along with each window. Again, the structure is a special case of Figure 1.



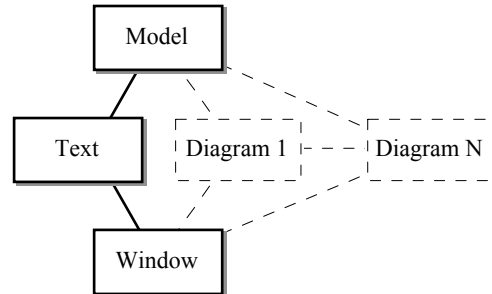
6.1.3 Stella documents

Stella has an equations view, an accumulator flow diagram and a sector map. The diagram and the map are alternative graphical representations of a model. One window is used to display either the text, the diagram, or the map. If view options for all three model representations are stored within the window, we get a structure that can be generalized to an arbitrary number of windows (as for Powersim). Thus, Stella documents are also a special case of Figure 1.



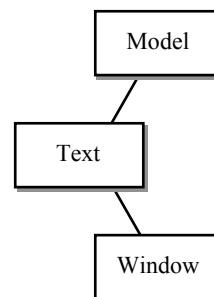
6.1.4 Vensim documents

Finally, the diagram below displays the way Vensim documents are organized. An arbitrary number of graphical views can be constructed for a model. In addition, the model equations can be viewed as text. One window is used to switch between views of the model.



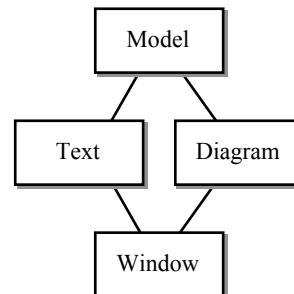
6.1.5 MIF compatible documents

For models without diagram information, the structure to the left should be used for MIF files.



Model documents with this structure can be opened by Dynamo and Vensim, and possibly by future versions of Powersim and Stella.

For models with graphical diagram information, documents should be organized as displayed below.



Documents in this format can be loaded by Dynamo, Powersim, Stella and Vensim, (given that a MIF reader is included with the tools).

Future versions of Dynamo may or may not be able to generate such models.

It should be noted that each tool may generate extra views (diagrams) and windows. Such extra information need not, however, be used by other tools reading the files. This means, for example, that:

- Powersim will not read the map view of a MIF file stored by Stella.
- Stella will only interpret the first window stored in a MIF file produced by Powersim.

- Powersim and Stella will only use the first diagram “view” of a Vensim model².
- Dynamo will only use the equations of a MIF model.

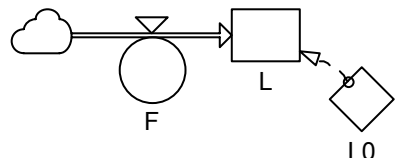
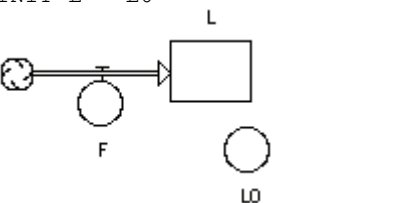
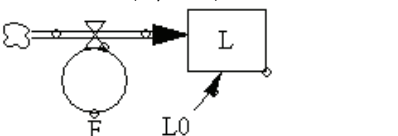
6.2 Equations

If we leave array variables out, all equations are variable definitions. (Possible other equation types include range definitions (FOR variables) and unit of measure definitions.)

Variables can be divided into two groups, levels and non-levels. The non-levels can be further subdivided in various ways, depending on software.

6.2.1 Levels

Level equations (and corresponding diagrams) are written like this in the various languages (L0 is the initial value of a level L, and F the flow.):

Software	Level equation and diagram
Powersim	$\text{level } L = L0 + dt * F$ 
Stella	$L(t) = L(t-dt) + (F) * dt$ $\text{INIT } L = L0$ 
Vensim	$L = \text{INTEG}(F, L0)$  <p>Note: This is one of many ways to create a Vensim diagram of a level with an inflow and explicit initialisation.</p>

6.2.2 Non-levels

For the remaining variable types we have the following example equations:

²The first Vensim view should be complete, i.e., all defined variables should be represented as symbols.

Type	Powersim	Stella	Vensim
const	const L0 = 3	L0 = 3	L0 = 3
aux	aux A = B*C	A=B*C	A = B*C
flow	aux F = A*2	...flows: F = A*2	F = A*2

Powersim uses a tag (text or icon) to identify variable type. Stella uses an iconic tag in equations view. The tag is not copied when exporting equations as text.

Vensim does not use tags – equation type is deduced from the right hand side of the equation. Dynamo uses one-letter tags.

Note that a Vensim equation like the following must be treated as a non-level (assuming L0 is not a constant):

$$L = L0 + \text{INTEGR}(F, 0)$$

This is because a level definition implicitly “freezes” the initial value. The following is OK for a level definition:

$$L = \text{INIT}(L0) + \text{INTEGR}(F, 0)$$

6.2.3 Table lookup functions

Let us look at how the tools can make a lookup into the graph below.

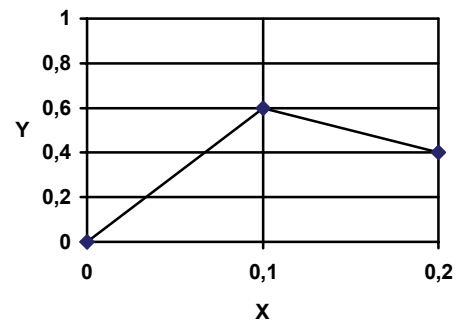


Figure 2: Example graph

6.2.3.1 Graphs in Dynamo

Dynamo specifies graphs using table functions, which look like this:

$$\text{TABLE}(\text{Vector}, X, X_{lo}, X_{hi}, Dx)$$

Dynamo has four table lookup functions, with different interpolation and/or asymptotes.

Function	Interpolation	Asymptotes
TABLE	Linear	None
TABPL	Polynomial	None
TABHL	Linear	Horizontal
TABXT	Linear	Linear

No asymptotes means that an error will occur if X is outside of the interval X_{lo} to X_{hi} .

Note that TABPL assumes that the vector argument has extra room at the end for internal use by the function.

The graph in Figure 2 can be expressed like this in Dynamo.

Graph referring to variable T

```
A Y = TABLE(T(*), X, 0, 0.2, 0.1)
T T(*) = 0, 0.6, 0.4
```

6.2.3.2 Graphs in Powersim

Powersim uses vectors to define y-values of fix points of a lookup function.

```
GRAPH(X, X0, Dx, Vector)
```

The vector argument can be either a literal or a vector variable. The fix points are evenly spaced in the X-direction.

The parameters X, X₀, and Dx can be any scalar expressions.

The following graph functions are defined:

Function	Interpolation	Asymptotes
GRAPH	Linear	Horizontal
GRAPHCURVE	Polynomial	Linear
GRAPHLINAS	Linear	Linear
GRAPHSTEP	Horizontal	Horizontal

The example in Figure 2 can be expressed like this:

Alt. 1: Literal graph

```
aux Y=GRAPH(X, 0, 0.1, [0, 0.6, 0.4])
```

Alt. 2: Graph referring to variable T

```
aux Y = GRAPH(X, 0, 0.1, T)
const T = [0, 0.6, 0.4]
```

In the above equations, “Min:0;Max:1” can be added as a comment inside the brackets.

6.2.3.3 Graphs in Stella

Stella lists pairs of fix points for the graph. Arguments – excluding the independent variable – must be literals.

```
GRAPH(X)
((X0, Y0), (X1, Y1), (X2, Y2), ... )
```

The fix points are evenly spaced in the X-direction. Stella supports both linear and horizontal interpolation (no distinction is made in equations view).

The input parameter (X) must be a variable. A graph function cannot be part of an expression, i.e., a graph must represent the entire right hand side of an equation.

The example in Figure 2 can be expressed like this:

Literal graph

```
Y = GRAPH(X)
(0.00, 0.00), (0.1, 0.6), (0.2, 0.4)
```

6.2.3.4 Graphs in Vensim

Vensim lists pairs of fix points in a separate table variable. Fix points can be nonuniformly spaced along the X-dimension.

The example in Figure 2 can be expressed like this:

Alt. 1: Graph referring to variable T

```
y = t(x)
t((0,0), (0.1,0.6), (0.2,0.4))
```

Alt. 2: Graph referring to variable T

```
y=TABLE(t, x, 0, 0.2, 0.1)
t=0,0.6,0.4
```

Alternative 1 is a form that must be converted to uniformly spaced points along the x-axis when loaded by Dynamo, Powersim or Stella. This process can result in a large number of fix points. As an example the following definition of t would result in 1000 evenly spaced fix points!

```
t=((0,1),(1,1),(100,1000),(1000,100000))
```

6.2.3.5 Conclusion on graphs

Limitations imposed by the various tools:

- Stella does not support arrays (vectors).
- Stella requires the independent axis to be given as a variable (not an expression).
- Stella does not support graphs as part of an expression.
- Stella requires the list of Y-values to be literals.
- Dynamo and Vensim does not support literal arrays (vectors) as arguments to the table lookup functions.
- Dynamo, Powersim and Stella do not support variably spaced fix points in the X-direction.
- Dynamo and Vensim do not support horizontal (discrete) interpolation.

The table lookup functions (at level one of MIF) should be defined with the following properties:

- Input variable (X) should be defined as a variable reference (not an expression).
- A graph function must define the entire right hand side of an equation.
- Only linear interpolation and linear asymptotes are supported.
- No use of separate vector variables.
- List of Y values for fix points defined as a list of literals.
- List of X-values for fix points defined as literals with fixed lower limit (X0) and fixed distance (Dx).

Below is the definition of a graph function that fulfills the above list of requirements:

```
GRAPH(X, X0, Dx, [Y0, Y1, ...Yn-1])
```

X must be the name of a (scalar) variable. X0, Dx, Yi must be literal numbers. The GRAPH function cannot be used as part of another expression. Brackets are used to group the Y values into a list (literal vector).

Dynamo and Vensim require a separate variable for holding Y0, Y1, ...Yn-1. When loading a MIF file, these tools should generate a new table variable to hold this list. When storing a MIF file, Dynamo and Vensim should expand the vector inline into the GRAPH function, and optionally output a native tag defining the name of the table variable that was used in the graph.

As an example, the Dynamo definition

```
A Y = TABXT(T(*), X, 0, 0.2, 0.1)
T T(*) = 0, 0.6, 0.4
```

would be represented like this in MIF (the syntax used below is explained starting on page 27):

```
{\var
  {\name Y}
  {\def GRAPH(X, 0, 0.1, [0, 0.6, 0.4])}
  {\_tablename T}
}
```

6.2.4 Resulting equations format

An equations format with only one equation per variable is preferred in setting the MIF standard.

More equation types will appear in future version of MIF.

The syntax of a level definition and other definitions should be similar. Both Powersim and Vensim has one equation per variable, and a similar syntax for levels

and non-levels. (This was also true for Stella until the equations format was changed.)

As a conclusion, variable equations will be identified as objects with the following attributes:

Attribute	Explanation
Name	Variable name (left hand side of equation)
Definition	Variable definition (right hand side of equation)
Documentation	String describing the variable
Unit of measure	String defining the unit of measure
Type	One of the following: level, auxiliary, constant. (Type can be deduced from <i>definition</i> .)
Scale	Default minimum and maximum value of the variable when displayed e.g., in a graph. The min/max values can be specified by the user or set to the lowest/highest value taken on by the variable during the most recent simulation.

6.3 Parameters to functions

Functions normally take scalar arguments than can be literals, variables or expressions. Some functions put restrictions on their parameters. Below is a list of parameter types.

Parameter	Explanation
<i>normal</i>	Any valid expression.
<i>literal</i>	The parameter must be a literal number
<i>variable</i>	The parameter must be a variable reference
<i>start-up</i>	The initial value of the parameter will be used for the entire simulation
<i>computational</i>	The parameter will be evaluated only when necessary
<i>delayed</i>	The parameter value will influence future results of the function using the parameter.

In addition, some functions take optional parameters. Optional parameters can be left out from right to left.

The various software tools interpret the above parameter types slightly different.

Computational parameters are supported only by Powersim. The other tools handle computational parameters as normal parameters.

Powersim uses a separate delayed link in connection with *delayed parameters*. Delayed parameters in Powersim must be variables. Powersim allows for circular definitions if a delayed link is involved in the circle. Functions using delayed parameters include: DELAYINF, DELAYMTR, DELAYPPL.

The GRAPH functions use a *literal vector argument*. This does not mean that vectors variables need to be supported by the tools in order to use the GRAPH function.

6.4 List of functions

In order for a MIF model to load as unchanged as possible into the tool that was used to create the model, care has been taken to include as many functions from the various tools as possible.

We also want to avoid putting unnecessary limitations on the functions a user can use in a given tool when creating MIF models. The only function categories that are not defined by this MIF standard, are array functions and functions on discrete variables (conveyors, ovens, queues).

When loading a MIF file, unsupported functions should be mapped into equivalent expressions (or macros). Future releases of the software tools are encouraged to include MIF functions that are not possible to represent in current versions of the tools.

- - Subtraction

Syntax A - B
 In A, B - Any number.
 Result The difference between A and B.
 Support Dynamo, Powersim, Stella, Vensim

- - Unary Minus

Syntax - A
 In A - Any number.
 Result A negated, i.e., (0 - A).
 Support Dynamo, Powersim, Stella, Vensim

! - Factorial

Syntax A!
 In A - Any number.
 Result Not-a-number (=?), if A is less than 0.

1, if A is zero.

$1*2*3*4*...*A$, if A is an integer.

Integrate(F) with $F = \text{TIME}^A * \text{EXP}(-\text{TIME})$, start time equal to 0.0, and stop time equal to infinity, if none of the above rules can be applied.

Example The following expression is true:

$$3! = 6$$

Support Powersim

% - Percent

Syntax A%

In A - Any number.

Result A divided by 100.

Example The following is true:

$$A/100 = A\%$$

Note Stella's % operator behaves different, and is listed under MOD.

Support Powersim

* - Multiplication

Syntax A * B

In A, B - Any number.

Result A multiplied by B.

Support Dynamo, Powersim, Stella, Vensim

+ - Addition

Syntax A + B

In A, B - Any number.

Result The sum of A and B.

Support Dynamo, Powersim, Stella, Vensim

+ - Unary plus

Syntax + A

In A - Any number.

Result A unchanged.

Support Dynamo, Powersim, Stella, Vensim

/ - Division

Syntax A / B

In A - Any number.

B - Any number except 0.

Result A divided by B.
Support Dynamo, Powersim, Stella, Vensim

< - Less Than

Syntax $A < B$
In A, B - Any number.
Result True if $A < B$ and False otherwise.
Support Powersim, Stella, Vensim

<= - Less Than or Equal To

Syntax $A \leq B$
In A, B - Any number.
Result True if $A \leq B$ and False otherwise.
Support Powersim, Stella, Vensim

<> - Not Equal To

Syntax $A \neq B$
In A, B - Any number.
Result True if $A \neq B$ and False otherwise.
Note See note for =.
Support Powersim, Stella, Vensim

= - Equal To

Syntax $A = B$
In A, B - Any number.
Result True if $A = B$ and False otherwise.
Note Testing for equality should be avoided if possible, since no computer is able to represent all possible floating point numbers accurately. This means that even if the results of two expressions should theoretically be the same, this may not be the case when executed on a computer. The following is an example of a "dangerous" expression:

IF(TIME = 5, B, C)

Support Powersim, Stella, Vensim

> - Greater Than

Syntax $A > B$
In A, B - Any number.
Result True if $A > B$ and False otherwise.
Support Powersim, Stella, Vensim

>= - Greater Than or Equal To

Syntax $A \geq B$
In A, B - Any number.
Result True if $A \geq B$ and False otherwise.
Support Powersim, Stella, Vensim

^ - Number Raised to a Power

Syntax $A \wedge B$
In A, B - Any non-negative number.
Result Returns the value of A raised to the power of B, e.g., AB.
Note The expression $A \wedge 0$ is 1 for all values of A. The expression $0 \wedge B$ is 0 for all values of B.
Support Dynamo, Powersim, Vensim

Dynamo $A ** B$

ABS - Absolute Value

Syntax ABS(X)
In X - Any number.
Result The absolute value of X.
Example The expression $ABS(-3) = 3$ is true.
Support Dynamo, Powersim, Stella, Vensim

AND - Conjunction, Logical And

Syntax A AND B
In A, B - Any number.
Result True if both A and B are True, and False otherwise.
Note A related (but not equivalent) way of expressing A AND B is:
 $A * B$.
Support Powersim, Stella, Vensim
Vensim :AND:

ARCCOS - Arcus Cosine

Syntax ARCCOS(Value)
In Value from interval [-1..1].
Result Angle defined by arcus sine of input value. Angle lies in the interval [-Pi/2 .. Pi/2].
Support Powersim

ARCSIN - Arcus Sine

Syntax ARCSIN(Value)
In Angle - Angle in radians.
Result Angle defined by arcus cosine of input value. Angle lies in the interval [0 .. Pi].
Support Powersim

ARCTAN - Arcus Tangent

Syntax ARCTAN(Value)
In Value.
Result Angle defined by arcus tangent of input value. Angle lies in the interval $<-\pi/2 .. \pi/2>$.
Support Powersim, Stella, Vensim

ATSTART - Test for Beginning of Simulation

Syntax ATSTART
Result True at the very beginning of the simulation, and False otherwise.
Support Powersim

AVG - Average

Syntax AVG(X1, X2, ...XN)
In X1, X2,...XN - Any number (N >= 1)
Result The mean of the arguments, as defined by the expression:
 $(X1, X2, ...XN)/N$
Note At least one argument must be specified.
Support Powersim, Stella
Stella MEAN(X1, X2, ...XN)

BOOL - Convert Number to Boolean

Syntax BOOL(X)
In X - Any number.
Result Returns 1 if ROUND(X) is not zero, and 0 otherwise.
Example The following expression is true:
 $BOOL(X) = (ROUND(X) \neq 0)$
Support Powersim

CEIL - Round Number Up to Nearest Integer

Syntax CEIL(X)
In X - Any number.

Result Returns the smallest integer that is greater than or equal to X.
Support Powersim

CLIP - IF Greater Than or Equal To

Syntax CLIP(P, Q, R, S)
In P, Q, R, S - Any numbers.
Result IF(R >= S, P, Q)
Note Same as FIFGE
Support Dynamo

COS - Cosine

Syntax COS(Angle)
In Angle - Angle in radians.
Result The cosine of Angle.
Support Dynamo, Powersim, Stella, Vensim

COSH - Hyperbolic Cosine

Syntax COSH (Value)
In Value.
Result The hyperbolic cosine of input value.
Support Powersim, Vensim

COSWAVE - Periodic Cosine Wave

Syntax COSWAVE(Amplitude, Period)
In Amplitude - Amplitude of the wave.
Period - Period of the wave.
Result A time-dependent cosine wave, defined by the equation:
 $COSWAVE(A, P) = A * COS(TIME/P)$
Note Function depends on TIME.
Support Powersim, Stella

DEGTOGRAD - Convert Degrees to Gradients

Syntax DEGTOGRAD(Angle)
In Angle - Angle in degrees.
Result The equivalent of Angle measured in gradients, as defined by the equation:
 $DEGTOGRAD(A) = A*400/360$
Support Powersim

DEGTORAD - Convert Degrees to Radians

Syntax DEGTORAD(Angle)

In	Angle - Angle in degrees.		averaging time equal to DelayTime, and a given Initial value for the delay.
Result	The equivalent of Angle measured in radians, as defined by the equation: $DEGTORAD(A) = A * \pi / 180$	Restrict	Initial must be omitted (because of Dynamo).

Support Powersim

DELAYINF - N-th Order Information Delay

Syntax DELAYINF(Input, DelayTime[, Order=1[, Initial(Order)=Input]])

In Input - Variable to be delayed (delayed parameter).
 DelayTime - Delay time measured in the time unit of the simulation.
 Order - Positive integer specifying the order of the delay. It defaults to 1 if not specified (optional start-up parameter).
 Initial - Initial delay value specified as a vector expression with Order number of elements. The elements of Initial default to the value of Input if not specified.
 If Initial has fewer elements than Order, the last element is replicated for the remaining values (optional start-up parameter).

Result The n-th order exponential information delay of Input, using an exponential averaging time equal to DelayTime, a given delay order of Order, and a given Initial value for the delay.

Restrict Order must be 1 or 3.
 Initial must be scalar (because of Stella).

Support Powersim, Stella

Stella SMTHN(Input, DelayTime, Order, Initial)

DELAYINF1 - First Order Information Delay

Syntax DELAYINF1(Input, DelayTime[, Initial=Input])

In Input - Variable to be delayed (delayed parameter).
 DelayTime - Delay time measured in the time unit of the simulation.
 Initial - Initial delay value. Initial defaults to the value of Input if not specified (optional start-up parameter).

Result The first order exponential information delay of Input, using an exponential

Support Dynamo, Stella, Vensim
 Dynamo SMOOTH(Input, DelayTime)

Stella SMTH1(Input, DelayTime[, Initial])

Vensim SMOOTH(Input, DelayTime)
 SMOOTHI(Input, DelayTime, Initial)

DELAYINF3 - Third Order Information Delay

Syntax DELAYINF3(Input, DelayTime [, Initial[(3)]=Input])

In Input - Variable to be delayed (delayed parameter).

DelayTime - Delay time measured in the time unit of the simulation.

Initial - Initial delay value. Initial defaults to the value of Input if not specified (optional start-up parameter).

Result The third order exponential information delay of Input, using an exponential averaging time equal to DelayTime, a given delay order of Order, and a given Initial value for the delay.

Restrict Initial must be scalar (because of Stella).
 Initial must be omitted (because of Dynamo).

Support Dynamo, Stella, Vensim

Dynamo DLINF3(Input, DelayTime)

Stella SMTH3(Input, DelayTime[, Initial])

Vensim SMOOTH3(Input, DelayTime)
 SMOOTH3I(Input, DelayTime, Initial)

DELAYMTR - N-th Order Material Delay

Syntax DELAYMTR(Input, DelayTime[, Order=1[, Initial(Order)=Input]])

In Input - Variable to be delayed (delayed parameter).

DelayTime - Delay time measured in the time unit of the simulation.

Order - Positive integer specifying the order of the delay (optional start-up parameter with default equal to 1).

	Initial - Initial delay value specified as a vector expression, with Order defining the number of elements. The elements of Initial default to the value of Input if not specified. If Initial has fewer elements than Order, the last element is replicated for the remaining values (optional start-up parameter).
Result	The n-th order exponential material delay of Input, using an exponential averaging time of DelayTime, a given delay Order, and a given Initial value for the delay.
Restrict	Order must be 1 or 3. Initial must be scalar (because of Stella). Initial must be omitted (because of Dynamo).
Support	Powersim

DELAYMTR1 - First Order Material Delay

Syntax	DELAYMTR1(Input, DelayTime [, Initial=Input])
In	Input - Variable to be delayed (delayed parameter). DelayTime - Delay time measured in the time unit of the simulation. Initial - Initial delay value (optional start-up parameter).
Result	The first order exponential material delay of Input, using an exponential averaging time of DelayTime, and a given Initial value for the delay.
Restrict	Initial must be omitted (because of Dynamo).
Support	Dynamo, Vensim
Dynamo	DELAY1(Input, DelayTime)
Vensim	DELAY1(Input, DelayTime) DELAY1I(Input, DelayTime, Initial)

DELAYMTR3 - Third Order Material Delay

Syntax	DELAYMTR3(Input, DelayTime [, Initial[3]=Input])
In	Input - Variable to be delayed (delayed parameter). DelayTime - Delay time measured in the time unit of the simulation.

	Initial - Initial delay value (optional start-up parameter).
Result	The third order exponential material delay of Input, using an exponential averaging time of DelayTime, and a given Initial value for the delay.
Restrict	Initial must be scalar (because of Stella). Initial must be omitted (because of Dynamo).
Support	Dynamo, Vensim
Dynamo	DELAY3(Input, DelayTime)
Vensim	DELAY3(Input, DelayTime) DELAY3I(Input, DelayTime, Initial)

DELAYPPL - Pipeline Delay

Syntax	DELAYPPL(Input, DelayTime[, Initial=Input])
In	Input - Variable to be delayed (delayed parameter). DelayTime - Delay time measured in the time unit of the simulation (start-up parameter). Initial - Initial delay value (optional start-up parameter with default equal to Input).
Result	The value of Input at DelayTime time units earlier in the simulation. During the first DelayTime time units of the simulation, the values specified by Initial are returned (Initial is a vector with one element per time step for a period equal to DelayTime).
Support	Powersim, Stella, Vensim
Stella	DELAY(Input, DelayTime, Initial)
Vensim	DELAY_FIXED(Input, DelayTime, Initial)

DELAYPPLINF - Variable Time Information Pipeline Delay

Syntax	DELAYPPLINF(Input, DelayTime, MaxDelayTime [, Initial=Input])
In	Input - Value to delayed (delayed parameter). DelayTime - Delay time measured in the time unit of the simulation. MaxDelayTime - Maximum delay time measured in the time unit of the simulation.

This variable is used, in combination with the simulation time step to determine the maximum size of the internal storage that is used by the function.

Initial - Initial delay value (optional start-up parameter with default equal to Input).

Result The "infinite" order information delay of Input with delay time DelayTime.

Note The difference between DELAYPPLINF and DELAYPPL is that DELAYPPLINF will adjust to a changing DelayTime during simulation (while DELAYPPL uses the initial value of DelayTime for the entire simulation).

The difference between DELAYPPLMTR and DELAYPPLINF lies in the transient response to a change in DelayTime (see Delay Functions).

Use DELAYPPL if the delay time is constant.

Support Powersim

DELAYPPLMTR - Variable Time Material Pipeline Delay

Syntax DELAYPPLMTR(Input, DelayTime, MaxDelayTime [, Initial=Input])

In Input - Value to delayed (delayed parameter)

DelayTime - Delay time measured in the time unit of the simulation.

MaxDelayTime - Maximum delay time measured in the time unit of the simulation.

This variable is used, in combination with the simulation time step to determine the maximum size of the internal storage that is used by the function.

Initial - Initial delay value (optional start-up parameter with default equal to Input).

Result The "infinite" order material delay of Input with delay time DelayTime.

Note The difference between DELAYPPLMTR and DELAYPPL is that DELAYPPLMTR will adjust to a changing DelayTime during simulation (while DELAYPPL uses the

initial value of DelayTime for the entire simulation).

The difference between DELAYPPLMTR and DELAYPPLINF lies in the transient response to a change in DelayTime (see Delay Functions).

Use DELAYPPL if the delay time is constant.

Support Powersim

DERIVN - N-th Order Time Derivative

Syntax DERIVN(Input[, Order=1])

In Input - Expression to be derived.

Order - Order of derivation. Default value is 1 (optional start-up parameter).

Result Returns N-th Order time derivative of Input.

Note Function result depends on previous values of Input.

Support Powersim, Stella

DIVZ0OP - Division with Zero Result for Zero Denominator

Syntax A DIVZ0OP B

In A, B - Any number.

Result IF(B<>0, A/B, 0)

See also DIVZ0

Support Powersim

DIVZ0 - Division with Zero Result for Zero Denominator

Syntax DIVZ0(A, B)

In A, B - Any number.

Result IF(B<>0, A/B, 0)

See also DIVZ0

Support Vensim

Vensim ZIDZ(.) -- ?

DIVZ1OP - Division with Unit Result for Zero Denominator

Syntax A DIVZ1OP B

In A, B - Any number.

Result IF(B<>0, A/B, 1)

Support Powersim

DIVZX - Division with Explicit Result for Zero Denominator

Syntax DIVZX(A, B, X)
 In A, B, X - Any number.
 Result IF(B<>0, A/B, X)
 Support Powersim, Vensim
 Vensim XIDZ(,,) -- ?

EULER - Sample at Start of Time Step

Syntax EULER(X)
 In X - Any variable (computational parameter).
 Result The value of X at the beginning of the current time step.
 Note X must be a variable (expressions and literals are not allowed).
 This function is sometimes useful with higher order integration methods, where several computations are performed between each time step. If we want a value (e.g., a rate) to be constant during the intermediate calculations of a higher order integration, EULER may be used.
 Support Powersim

EXP - Exponential of Number

Syntax EXP(X)
 In X - Any number.
 Result Returns the exponential of X (e raised to the power of X (ex)).
 Note This is the inverse function of LN, i.e., LN(EXP(X)) = X.
 Restrict: $-174 \leq X \leq 174$ (Dynamo)
 Support Dynamo, Powersim, Stella, Vensim

EXPRND - Exponential Distribution

Syntax EXPRND ([Mean=1[, Seed]])
 In Mean - Mean value of distribution (optional parameter with default equal to one).
 Seed - Initialization of random number generator (optional start-up parameter with random default value).
 Result Generates a series of exponentially distributed random numbers with a mean of Mean.

Examples EXPRND generates exponentially distributed random numbers with mean equal to 1.
 EXPRND(5) is the same as 5*EXPRND, both generating a series of exponentially distributed numbers with a mean of 5.

Support Powersim, Stella
 Vensim RANDOM_EXPONENTIAL() -- no args!

FALSE - Logical False

Syntax FALSE
 Result The value zero.
 Support Powersim

FIFGE - First If Greater Than or Equal To

Syntax FIFGE(P, Q, R, S)
 In P, Q, R, S - Any numbers.
 Result IF(R >= S, P, Q)
 Note Same as CLIP.
 Use IF instead.
 Support Dynamo

FIFZE - First If Third is Zero

Syntax FIFZE(P, Q, R)
 In P, Q, R - Any numbers.
 Result IF(R = 0, P, Q), which is the same as IF(R, Q, P).
 Note Same as Dynamo's SWITCH.
 Use IF instead.
 Support Dynamo

FLOOR - Round Number Down to Nearest Integer

Syntax FLOOR(X)
 In X - Any number.
 Result Returns the largest integer that is less than or equal to X.
 Examples FLOOR (3.14) = 3
 FLOOR (-5.5) = -6

Support Powersim

FORECAST - Value Forecasting

Syntax FORECAST (Input, PastTime, FutureTime[, Initial=0])

In Input - Value to be predicted.

PastTime - Positive number determining the averaging time used in computing the trend in Input, measured in the time unit of the simulation.

FutureTime - Non-negative number determining how far into the future a forecast is going to be, measured in the time unit of the simulation.

Initial - Initial trend value (optional start-up parameter with default equal to zero).

Result The forecaster value of Input at a time FutureTime into the future. The function computes the first order exponential average of Input by using an averaging time of PastTime, and then extrapolates the trend a distance equal to FutureTime into the future.

Note The function depends on previous values of its first parameter.

Support Powersim, Stella

FRAC - Fraction of Number

Syntax FRAC(X)

In X - Any number.

Result The fractional (decimal) part of X.

Note The relationship between X, INT(X) and FRAC(X) is the following (for all values of X):

$$X = \text{INT}(X) + \text{FRAC}(X)$$

Examples $\text{FRAC}(3.14) = 0.14$
 $\text{FRAC}(-5.5) = -0.5$

Support Powersim

FV - Future Value

Syntax FV (Rate, Periods, Payment, PresentValue)

In Rate - Rate per period.
 Periods - Number of periods.
 Payment - Periodic payment.
 PresentValue - Present value.

Result Future value (see Introduction to Financial Functions)

Support Powersim, Stella

GRADTODEG - Convert Gradients to Degrees

Syntax GRADTODEG (Angle)

In Angle - Angle in gradients.

Result The equivalent of Angle measured in radians, as defined by:

$$\text{GRADTODEG}(A) = A * 360 / 400$$

Support Powersim

GRADTORAD - Convert Gradients to Radians

Syntax GRADTORAD (Angle)

In Angle - Angle in gradients.

Result The equivalent of Angle measured in radians, as defined by:

$$\text{GRADTORAD}(A) = A * \text{PI} / 200$$

Support Powersim

GRAPHLINAS - Linear Graph with Linear Asymptotes

Syntax GRAPHLINAS (X, X1, Dx, Y(N))

In X - Any number (independent variable).
 X1 - Value of X corresponding to the first sample, i.e., Y(1).
 Dx - Increment between sampled X-values. Must be positive.
 Y - Vector with at least one element.

Result GRAPHLINAS is used to express a function by giving a set of function values for a series of equally spaced input values, as listed below:

<u>Input</u>	<u>Output</u>
X1	Y(1)
X1+Dx	Y(2)
X1+2*Dx	Y(3)
...	...
X1+(N-1)*Dx	Y(N)

If X is less than X1 then the value is extrapolated on the basis of Y(1) and Y(2), i.e., a line through the two first points of the sample.

If X is greater than X1 + (N-1)*Dx then the value is extrapolated on the basis of the last two points in the sample, i.e., Y(N-1) and Y(N). This means that the function

uses linear asymptotes based on the two most extreme points at both edges of the sample.

Restrict Dynamo and Vensim require a separate table variable to create a table lookup function. Table variables will not be exported to the MIF file. Instead the table elements will be expanded inline into the [...] part of the GRAPHLINAS function.

Stella does not support table lookup as part of an expression.

Stella requires X to be a variable.

Stella requires X1 and Dx to be literals.

Support Dynamo, Powersim, Stella, Vensim

Dynamo TABXL(T, X1, X1 + (N-1)*Dx, Dx)
T = Y(1), Y(2), ..., Y(N-1), Y(N)

Vensim TABXL(T, X1, X1 + (N-1)*Dx, Dx)
T = Y(1), Y(2), ..., Y(N-1), Y(N)

Stella Generated by Become Graph.

HIVAL - Highest Simulated Value

Syntax HIVAL(X)

In X - Any number.

Return The highest value of the expression X so far in the simulation.

Note Function depends on previous values of its parameter.

Support Powersim

HYPOT - Hypotenuse

Syntax HYPOT(X, Y)

In X, Y - Any number.

Result The square root of $X^2 + Y^2$, i.e., the length of the hypotenuse of a square-angled triangle. This may also be expressed as: $SQRT(X^2+Y^2)$.

Support Powersim

IF - Arithmetic If

Syntax IF(Condition, Value1, Value2)

In Condition - Logical value (True or False).
Value1 - Any numeric expression (computational parameter).

Value2 - Any numeric expression (computational parameter).

Result The value of Value1 is evaluated and returned if Condition is True. The value of Value2 is evaluated and returned otherwise.

Support Dynamo, Powersim, Stella, Vensim

Powersim Will compute only one of Value1 or Value2.

Dynamo SWITCH(Value2, Value1, Cond)
FIFZE(Value2, Value1, Cond)

Stella IF Cond THEN Value 1 ELSE Value2

Vensim IF_THEN_ELSE(Cond, Value1, Value2)

INFINITY - Infinitely Large Positive Number

Syntax INFINITY

Result A value that is used to represent a number that is too large to be represented by the computer.

Note Use -INFINITY to denote an infinitely large negative number.

The largest number that may be stored by Powersim is 1.0E+300.

Support Powersim

INIT - Initial Value

Syntax INIT(X)

In X - Any numeric expression (computational start-up parameter).

Result The initial value of X.

Note The function depends on previous value of its parameter.

X will only be evaluated during the initialization stage of the simulation, and the resulting value will be returned for the rest of the simulation.

Restrict X must be a level variable (because of Stella)

Support Powersim, Stella, Vensim

Stella INIT(X)

Vensim INITIAL(X)

INT - Integer Part of Number

Syntax INT(X)

In X - Any number.

Result Returns the integer part of X.

Note The relationship between X, INT(X) and FRAC(X) is the following (for all values of X):

$$X = \text{INT}(X) + \text{FRAC}(X)$$

Examples INT(3.14) = 3
 INT(-5.5) = -5

Support Powersim, Stella, Vensim

Vensim INTEGER(X)

INTEGRATE - Integration

Syntax INTEGRATE(X[, Init=0])

In X - Variable to be integrated over time (delayed parameter).
 Init - Initial value (optional start-up parameter).

Result Init plus the integrated value of X, i.e., Init plus the sum Xt/Dt for all time steps from the beginning of the simulation to the current time.

Note The function depends on previous values of its parameter.

Restrict Powersim: X must be a variable (expressions and literals are not allowed), and in the diagram X must be connected to the current variable using a delayed link.

Support Powersim, Vensim

Vensim INTEGR(X, Init)

Powersim INTEGRATE(X) -- should add optional Init

ISGT - Greater Than

Syntax ISGT(A, B)

In A, B - Any numbers.

Result Same as $A > B$.

Note Use > instead.

Support Stella

Stella SWITCH(A, B) = ISGT(A, B)

LN - Natural Logarithm

Syntax LN(X)

In X - Any positive number.

Result The natural logarithm of X.

Support Dynamo, Powersim, Stella, Vensim

Dynamo LOGN(X)

Stella LOGN(X)

LOG - Base N Logarithm

Syntax LOG(X[, Base=10])

In X - Any positive number.
 Base - Base of logarithm (optional parameter with default equal to 10).

Result The base N logarithm of X.

Support Powersim, Vensim

LOG10 - Base 10 logarithm

Syntax LOG10(X)

In X - Any positive number.

Result The base 10 logarithm of X.

Support Stella

LOVAL - Lowest Simulated Value

Syntax LOVAL (X)

In X - Any number.

Result The lowest value of the expression X so far in the simulation.

Note The function depends on previous values of its parameter.

Support Powersim

MAX - Maximum

Syntax MAX(X1, X2, ...XN)

In X1, X2, ...XN - Any number.

Result The maximum of the arguments.

Restrict N=2 (because of Dynamo and Vensim)

Support Dynamo, Powersim, Stella, Vensim

MIN - Minimum

Syntax MIN(X1, X2, ...XN)

In X1, X2, ...XN - Any number.

Result The minimum of the arguments.

Restrict N=2 (because of Dynamo and Vensim)

Support Dynamo, Powersim, Stella, Vensim

MOD - Remainder of Division

Syntax A MOD B

In A - Any number.
B - Any number except zero.

Result The remainder of A / B, defined as the value R such that:
 $A = B * k + R$
where k is an integer and $ABS(R) < ABS(B)$.

See also MODULO

Support Powersim, Stella

Stella A % B

MODULO - Remainder of Division

Syntax MODULO(A, B)

In A - Any number.
B - Any number except zero.

Result The remainder of A / B, defined as the value R such that:
 $A = B * k + R$
where k is an integer and $ABS(R) < ABS(B)$.

See also MOD

Support Stella, Vensim

Stella MOD(A, B)

Vensim MODULO(A, B)

MONTECARLO

Syntax MONTECARLO(Probability[, Seed])

In Probability - Any number between 1 and 100.

Result $RANDOM(0, 100, Seed) \leq Probability * Timestep$

Support Stella

NAN - Not a Number

Syntax NAN

Result A value that is used to represent an invalid number.

Support Powersim

NORMAL - Normal Distribution

Syntax NORMAL ([Mean=0[, Deviation=1[, Seed]])

In Mean - Mean value of distribution, with default equal to 0 (optional parameter).
Deviation - Deviation of distribution, with default equal to 1 (optional parameter).
Seed - Initialization of random number generator (optional start-up parameter with random default value).

Result Generates a series of normally distributed random numbers with a mean of Mean and a standard deviation of Deviation.

Examples 5 + NORMAL generates normally distributed random numbers with mean = 5 and standard deviation = 1.
 $20 + 3 * NORMAL$ generates normally distributed random numbers with mean = 20 and standard deviation = 3.
NORMAL(20, 3) generates the same distribution as the one above.
NORMAL(5) generates normally distributed random numbers with mean = 5 and standard deviation = 1.

Support Dynamo, Powersim, Stella

Dynamo NORMRN(Mean, Deviation)

Vensim RANDOM_NORMAL() -- no args!

NOT - Negation

Syntax NOT A

In A - Logical value, True or False.

Result True if A is False, and False otherwise.

Example The following is true: NOT 1 = 0; and NOT 0 = 1.

Note A related way of expressing NOT A is: 1 - A.

Support Powersim, Stella, Vensim

Vensim :NOT:

NPV - Net Present Value

Syntax NPV(Payment, InterestRate)

In Payment - Input variable.
InterestRate - Rate per time unit.

Result The Net Present Value

Note What do we do with Stella's *Initial?*

Support Powersim, Stella

Stella NPV(Input, InterestRate[, Initial])

OR - Logical Or

Syntax A OR B

In A, B - Logical value, True or False.

Result True if at least one of A or B is True, and False otherwise.

Note A related way of expressing A OR B is: A + B - A * B.

Support Powersim, Stella, Vensim

Vensim :OR:

PCT - Convert Number to Percent

Syntax PCT(X)

In X - Any number.

Result X * 100

Support Powersim, Stella

PI - Trigonometric Constant Pi

Syntax PI

Result Value of Pi, i.e., 3.141592654...

Support Powersim, Stella

PMT - Periodic Payment

Syntax PMT(Rate, Periods, PresentValue, FutureValue)

In Rate - Discount rate per period.
Periods - Number of periods.
PresentValue - Present value.
FutureValue - Future value.

Result Periodic payment.

Support Powersim, Stella

POISSON - Poisson Distribution

Syntax POISSON ([Mean=1[, Seed]])

In Mean - Mean value of distribution (optional parameter with default equal to 1).
Seed - Initialization of random number generator (optional start-up parameter with random default value).

Result Generates a series of random numbers according to the Poisson distribution, with a mean of Mean.

Note Stella must store POISSON(M) as POISSON(M*DT)/DT and load POISSON(X) as POISSON(X/DT)*DT.

Support Powersim, Stella

Vensim RANDOM_POISSON() -- no args!

POLY - Polynomial

Syntax POLY(X, A0 [, A1, ..., An])

In X - Any number.

A0...An - Any number (polynomial coefficients).

Result The polynomial function of X, as defined by the expression:

Support Powersim

PULSE - Periodic Pulse

Syntax PULSE(Volume, First, Interval)

In Volume - Pulse volume (computational parameter).

First - Time of first pulse, measured in the time unit of the simulation.

Interval - Time interval between pulses, measured in the time unit of the simulation.

Result Volume/TIMESTEP or 0, depending on the current TIME and the values of

First and Interval. A pulse occurs at the time First, and every time Interval thereafter.

Note The function depends on TIME and TIMESTEP.

The relationship between PULSE and PULSEIF is the following:

$PULSE(V, F, I) = PULSEIF(TIMECYCLE(F, I), V)$

Dynamo's PULSE function behaves different, and is listed under TIMECYCLE.

Support Powersim, Stella

Stella PULSE(Volume[, First, Interval])

PULSEIF - Conditional Pulse

Syntax PULSEIF(Condition, Volume)

In Condition - Condition True or False determining if pulse is going to occur.

Volume - Pulse volume (computational parameter).

Note The function depends on TIMESTEP.
 Result Zero if Condition is False and Volume/TIMESTEP otherwise.

Support Powersim

PV - Present Value

Syntax PV(Rate, Periods, Payment, FutureValue)

In Rate - Rate per period.
 Periods - Number of periods.
 Payment - Periodic payment.
 FutureValue - Future value (optional parameter with default equal to zero).

Result Present value.

Support Powersim, Stella

RADTODEG - Convert Radians to Degrees

Syntax RADTODEG(Angle)

In Angle - Angle in radians.
 Result The equivalent of Angle measured in degrees, as defined by the equation:

$$\text{RADTODEG}(A) = A * 180 / \text{PI}$$

Support Powersim

RADTOGRAD - Convert Radians to Gradients

Syntax RADTOGRAD (Angle)

In Angle - Angle in radians.
 Result The equivalent of Angle measured in degrees, as defined by the equation:

$$\text{RADTOGRAD}(A) = A * 200 / \text{PI}$$

Support Powersim

RAMP - Linear Function

Syntax RAMP(Slope, First)

In Slope - Slope of the function.
 First - Time to start ramp.

Result 0 if TIME < First, and (Slope * TIME - First) otherwise, as defined by the following equation:

$$\text{RAMP}(S, F) = \text{IF}(\text{TIME} < F, 0, (\text{TIME} - F) * S)$$

Support Dynamo, Powersim, Stella

Dynamo RAMP(Slope, Time)

Stella RAMP(Slope[, Time])

RANDOM - Uniform Distribution

Syntax RANDOM([Min=0[, Max=Min+1[, Seed]])

In Min - Minimum value to be returned (optional parameter with default equal to 0).

Max - Maximum value to be returned (optional parameter with default equal to Min+1).

Seed - Initialization of random number generator (optional start-up parameter with random default value).

Result Uniformly distributed random number between Min and Max.

Examples 5 + RANDOM returns uniformly distributed random numbers between 5 and 6. This may also be expressed as: RANDOM(5).

20 + 3 * RANDOM returns uniformly distributed random numbers between 20 and 23. This can also be expressed as: RANDOM(20, 23)

Support: Dynamo, Powersim, Stella, Vensim

Dynamo Min + (NOISE()+0.5) * (Max-Min)

Stella RANDOM(Min, Max[, Seed])

Vensim Min + RANDOM_0_1() * (Max-Min)

ROUND - Round Number to Nearest Integer

Syntax ROUND(X)

In X - Any number.

Result X rounded to the nearest integer, according to the following formula:

$$\text{ROUND}(X) = \text{IF}(X \geq 0, \text{FLOOR}(X+0.5), \text{CEIL}(X-0.5))$$

Support Powersim, Stella

SAMPLE - Periodic Sample

Syntax SAMPLE(Input, First, Interval[, Initial=0])

In Input - Any numeric expression to be sampled (computational parameter).

First - First sampling time measured in the time unit of the simulation.

Interval - Sampling interval measured in the time unit of the simulation.

	Initial - Value returned by SAMPLE until the first sampling time (optional start-up parameter with default equal to zero).
Result	SAMPLE is periodically set equal to Input and retains this value until the next sampling time. The first sample is taken at time First, and new samples are taken every Interval time units thereafter. SAMPLE returns the value of Initial until the first sample time.
Note	The function depends on TIME and on previous values of its first parameter.
Restrict	First should be set equal to STARTTIME (because of Dynamo).
Support	Dynamo, Powersim
Dynamo	SAMPLE(Input, Interval, Initial)

SAMPLEIF - Conditional Sample

Syntax	SAMPLEIF(Condition, Input[, Initial=0])
In	Condition - Conditional value True or False, which determines whether a sample is to be taken. Input - Any numeric expression to be sampled (computational parameter). Initial - Value to be returned by SAMPLEIF until the first time Condition is True (optional start-up parameter with default equal to zero).
Result	The value of Input at the most recent sampling time. Before the first sampling time, the value of Initial is returned.
Note	The function depends on previous values of its second parameter.
Support	Powersim, Vensim
Vensim	SAMPLE_IF_TRUE(Cond, Input, Initial)

SIGN - Sign of Number

Syntax	SIGN (X)
In	X - Any number.
Result	+1 if X is positive, -1 if X is negative, and 0 otherwise.
Example	The following is true: $SIGN(X) = IF(X < 0, -1, IF(X > 0, 1, 0))$
Support	Powersim

SIN - Sine

Syntax	SIN(Angle)
In	Angle - angle in radians.
Result	The sine of Angle.
Support	Dynamo, Powersim, Stella, Vensim

SINH - Hyperbolic Sine

Syntax	SINH(Value)
In	Value.
Result	The hyperbolic sine of input value.
Support	Powersim, Vensim

SINWAVE - Periodic Sine Wave

Syntax	SINWAVE(Amplitude, Period)
In	Amplitude - Amplitude of the wave. Period - Period of the wave.
Result	A time-dependent sine wave, defined by the equation: $SINWAVE(A, P) = A * SIN(TIME/P)$
Note	The function depends on TIME.
Support	Powersim, Stella

SQRT - Square Root

Syntax	SQRT(X)
In	X - Any non-negative number.
Result	The square root of X.
Example	The relationship between SQRT and ^ is: $SQRT(X) = X ^ 0.5$
Support	Dynamo, Powersim, Stella, Vensim

STARTTIME - Start Time of Simulation

Syntax	STARTTIME
Result	The start time of the simulation, as defined in the Simulation Setup dialog box.
Support	Stella, Powersim

STDDEV - Standard Deviation

Syntax	STDDEV (X1, X2, ...Xn)
In	X1, X2, ...Xn - Any number.
Result	The standard deviation of the arguments, as defined by:
Support	Powersim

STEP - Step Function

Syntax	STEP(Height, StepTime)
In	Height - Numeric expression determining step height. StepTime - Numeric expression determining time of step.
Result	Zero if TIME is less than StepTime, and Height otherwise.
Support	Dynamo, Powersim, Vensim

STOPTIME - Stop Time of Simulation

Syntax	STOPTIME
Result	The stop time of the simulation, as defined in the Simulation Setup dialog box.
Support	Powersim

SUM - Sum of expressions

Syntax	SUM(X1, X2, ...XN)
In	X1, X2, ...XN - Any number.
Result	The sum of the arguments.
Note	Powersim's SUM function behaves different, and is not possible to represent by the other tools. It is therefore not part of the MIF standard.
Support	Stella
Stella	SUM(X1, X2, ...)

SWITCH - First If Third is Zero

Syntax	SWITCH(P, Q, R)
In	P, Q, R - Any numbers.
Result	IF(R = 0, P, Q), which is the same as IF(R, Q, P).
Note	Stella's SWITCH function behaves different, and is listed under ISGT. Same as Dynamo's FIFZE. Use IF instead.
Support	Dynamo

TAN - Tangent

Syntax	TAN(Angle)
In	Angle - Angle in radians.
Result	The tangent of Angle.
Support	Powersim, Stella, Vensim

TANH - Hyperbolic Tangent

Syntax	TANH(Value)
In	Value.
Result	The hyperbolic tangent of input value.
Support	Powersim, Vensim

TIME - Current Time of Simulation

Syntax	TIME
Result	The current time of the simulation, starting at the value of STARTTIME, and incremented by TIMESTEP for each step of the simulation.
Note	The function depends on TIME (obviously).
Support	Dynamo (variable) Powersim, Stella, Vensim (variable)

TIMECYCLE - Test for Cyclic Time or Time Interval

Syntax	TIMECYCLE(First, Interval[, Duration=0[, Height=1]])
In	First - First time to check for. Interval - Time between intervals to check for. Duration - Length of interval (optional parameter with default equal to zero). Height - Value to be returned when inside time interval (optional parameter with default equal to one).
Result	Value if current simulation time is within an interval from First + k * Interval to First + k * Interval + Duration, where k is a non-negative integer.
Note	The function depends on TIME and TIMESTEP. When testing against TIME the size of the time step is taken into account.
Support	Dynamo, Powersim
Dynamo	PULSE(Height, Duration, First, Interval)

TIMEIS - Test for Given Time or Time Interval

Syntax	TIMEIS(PointInTime[, Duration=0])
In	PointInTime - Numeric expression determining time to be tested for.

Duration - Duration of time interval to test for (optional parameter with default value equal to zero).

Result True if current time lies in the interval PointInTime to PointInTime+Duration.

Support Powersim, Vensim

Vensim PULSE(PointInTime, Duration)

TIMESTEP - Time Step of Simulation

Syntax TIMESTEP

Result The time step of the simulation, as defined in the Simulation Setup dialog box.

Support Powersim, Stella

Dynamo DT

Stella DT

TREND - Trend Extrapolation

Syntax TREND(Input, AveragingTime[, Initial=0])

In Input - Numeric expression to be examined over time.

AveragingTime - Averaging time.

Initial - Initial value of the TREND function (optional start-up parameter that defaults to zero).

Result The first order exponential average change rate of Input, using the given

AveragingTime. The value is expressed as the relative change in Input per time unit.

The function depends on previous values of its first parameter.

Support Powersim, Stella

TRUE - Logical True

Syntax TRUE

Result The value one.

Note When used in a condition, any value which rounds to something different from zero is regarded to be true.

Support Powersim

XOR - Exclusive Logical Or

Syntax A XOR B

In A, B - Logical value, True or False.

Result True if one, and only one, of A and B is True, and False otherwise. In other words:

(A AND NOT B) OR (NOT A AND B)

or simply:

BOOL(A) <> BOOL(B)

Note When using <> and = on logical values, it is a good practice to use BOOL on the values first. This ensures that True values are set to one. The example below illustrates this point, as the two expressions do not produce the same result:

3 = 4 -- is false










BOOL(3) = BOOL(4) -- is true

A related way of expressing A XOR B is: A + B - 2 * A * B.

Support Powersim

6.5 Variable symbols

The following variable symbols appear in SD diagrams:

Variables	Powersim	Stella	Vensim
Level			 (use any)
Auxiliary			 (use any)
Constant		(use auxiliary)	Constant (use any)
Snapshot			<Level>

Variable symbols refer to variable definitions in the equations part of the MIF file.

Vensim does not *enforce* consistency between symbol shape and variable type.

A Powersim constant is an auxiliary with a static definition (the value is independent of time).

Powersim and Stella display snapshots according to variable type. Vensim puts the name in angle brackets (<>) and *normally* omits the symbol shape.

Variable symbol attributes:

Attribute	Powersim	Stella	Vensim
Line color	16	2, 16, 256 depending on HW	64

Fill color			
<i>transparent</i>	X	X	X
<i>solid</i>			64
Line width	fixed (1)	fixed (1)	free: 0->
Size	4 steps	fixed	free
Name pos.			
<i>free</i>		X	
<i>outside</i>			
<i>inside</i>			X
<i>below</i>	X	x	X
<i>above</i>	X	x	X
<i>left</i>	X	x	X
<i>right</i>	X	x	X
Name font	fixed	fixed	free
Shape			
<i>none</i>			X (const.)
<i>by type</i>	X	X	X (default)
<i>box</i>	levels	levels	X (levels)
<i>clear box</i>			X
<i>circle</i>	auxiliaries	auxiliaries	X (aux.)
<i>hexagon</i>			X
<i>diamond</i>	constants		X
<i>triangle</i>			X
<i>up triangle</i>			X

	following: 'transparent' 'solid'	
Fill color		
<i>red</i>	0..255	255
<i>green</i>	0..255	255
<i>blue</i>	0..255	255
Name pos.		
<i>placement</i>	One of the following: 'outside' 'inside'	'outside'
<i>angle</i>	0..360	270 (only used when 'outside')
Name font	see RTF	system default
Shape	One of the following: 'none' 'box' 'clear box' 'circle' 'hexagon' 'diamond' 'triangle' 'up triangle'	-
<i>Auto shape</i>	0 (off) 1 (on)	1 (Vensim's "by type")
<i>Current shape</i>	One of the following: 'none' 'box' 'clear box' 'circle' 'hexagon' 'diamond' 'triangle' 'up triangle'	-
Snapshot, ghost	0 (no) or 1 (yes)	0

In storing variable symbols, the following fields will be defined by MIF.

Field	Values	Default
Type	'var'	'var'
Symbol id	Numeric ID of this symbol (local to each view)	-
Visible	0 (no) or 1 (yes)	1
Selected	0 (no) or 1 (yes)	0
Variable	Name of associated variable	-
Position		
<i>across</i>	1/20*1/72 inch	-
<i>down</i>	1/20*1/72 inch	-
Size		
<i>width</i>	1/20*1/72 inch	-
<i>height</i>	1/20*1/72 inch	-
Line width	1/20*1/72 inch	15 (3/4 pt)
Line color		
<i>red</i>	0..255	0
<i>green</i>	0..255	0
<i>blue</i>	0..255	0
Fill type	One of the	'transparent'

The figure below displays how the various measurements are applied to a variable symbol.

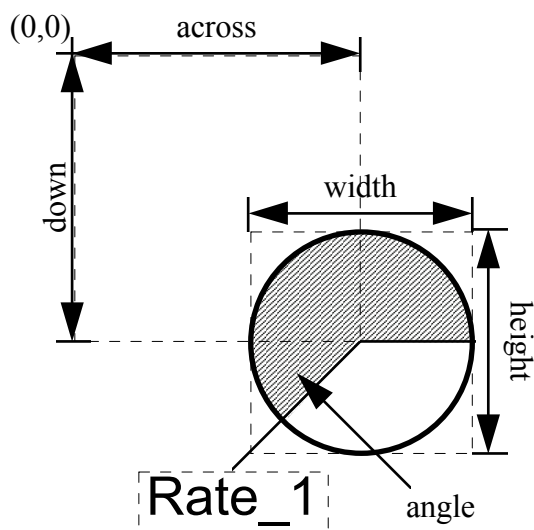


Figure 3: Variable symbol measurements

The gap between the variable shape and the name can be determined by each software. Powersim and Vensim will round the angle to the nearest multiple of 90. Powersim and Stella will force the variable name ‘outside’.

6.6 Arrows

The following building blocks are used to make links and flows:

Arrow	Powersim	Stella	Vensim
Source, sink	(must be part of flow, cannot be moved)	(must be part of flow, can be moved a little)	
Valve	(part of flow)	(part of flow)	(can be sized and rotated in steps of 90°)
Arrow			
Shape	determined by type	determined by type	arc polyline p. h/v-line
Style	fixed	fixed	several
Joint	must be part of link	must be part of link	(indep. symbol of type “comment” displayed)

			as an icon)
Link	(Three kinds: normal, init, delayed)		Use arrow
Flow-with-rate	use flow, link, aux.	Noname 3	Use arrow, valve, arrow, var.
Flow		part of flow-with-rate	Use arrow, valve, arrow

As can be seen from the above table, the various SD tools operate with different atomic building blocks for making links and flows.

Vensim’s approach is loosely connected to the traditional SD concepts of conserved (material) flows and nonconserved (information) flows. Only certain combinations of arrows, valves, comment symbols, and variable symbols represent valid SD flow diagrams in the traditional sense. The MIF format will support only such combinations.

6.7 Flows

A flow is a conserving link, that is, a link that moves a mass from one place to another.

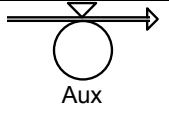
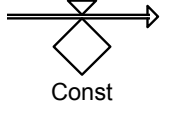
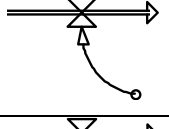
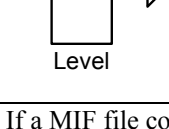
Flows must start and end in a cloud or a level variable.

The rate of mass per time unit is controlled by a variable, connected to the flow valve either through an information link or by direct association.

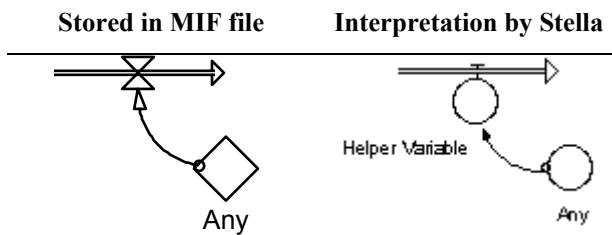
Below is a list of all valid flow end connections. (The valve is omitted, and will be discussed separately below.)

	cloud -> flow -> cloud
Level_1	cloud -> flow -> level
Level_1	level -> flow -> cloud
Level_1 Level_2	level -> flow -> level

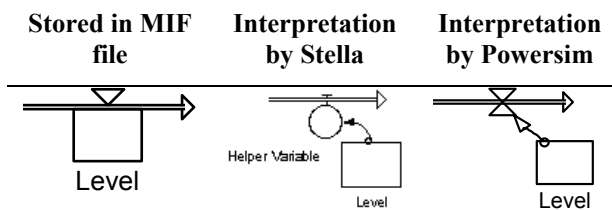
The flow rate must be controlled by a variable either directly or via an information link. Below all valid connections to a flow-valve are listed.

 Aux	auxiliary -> flow-valve
 Const	constant -> flow-valve
 Link	link -> flow-valve
 Level	level -> flow-valve

If a MIF file contains a link to flow-valve connection, Stella should automatically generate a rate variable (auxiliary) and point the link to that variable. The definition of the rate should be set equal to the name of the variable at the other end of the link. (Alternatively Stella can remove the link symbol and merge the rate variable symbol with the flow symbol. If the same variable is used to control N flows, N-1 helper variables must be created.)



If a MIF file contains a level to flow-valve connection (discouraged), Stella should generate a rate variable (auxiliary), add a link from the level to the rate, and define the rate equal to the name of the Level. Powersim should add a link from the level to the flow valve.



Below is a list of attributes that will be stored for a flow in the MIF file.

Field	Type	Default
Type	'flow'	'flow'
Symbol id	Numeric ID of this symbol	-

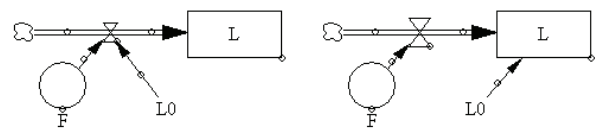
	(local to each view)	
Visible	0 (no) or 1 (yes)	1
Selected	0 (no) or 1 (yes)	0
Path	<i>p</i> followed by series of the following: line <i>p</i> arc <i>p1,p2</i> curve <i>p1,p2,p3</i> (see page 33)	-
Line width	1/20*1/72 inch	15 (3/4 pt)
Line color		
	<i>red</i>	0..255
	<i>green</i>	0..255
	<i>blue</i>	0..255
Source	Numeric ID of source symbol	-1 (interpret as cloud)
Destination	Numeric ID of destination symbol	-1 (interpret as cloud)
Rate	Numeric ID of symbol (variable or link) controlling valve	-
Valve Position		
	<i>across</i>	1/20*1/72 inch
	<i>down</i>	1/20*1/72 inch

A line style with two parallel lines is assumed.

Cloud symbols and valve symbols should not be stored in the MIF file.

Vensim can have several links (arrows) pointing at a valve. A MIF compatible Vensim model should not allow for more than one link pointing to a valve.

As an example, some Vensim model are drawn with both the flow rate and the level initialization pointing at the valve symbol. A MIF compatible Vensim model would point the rate at the valve and the initialization at the level.



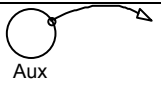

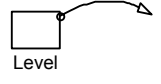

a) Not MIF compatible b) MIF compatible

When loading a model as displayed in alternative b, Stella would simply ignore the link from L0 to L. The rate F can be merged (moved) into the flow by Stella, as Stella does not support links-to-flow connections.

6.8 Links

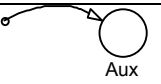
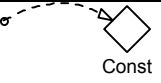
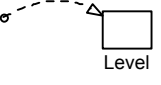

A link is non-conserving, that is, mass will not be moved through the link.

Below is a list of valid sources of a link. Several links can depart from the same symbol.

	auxiliary
	constant
	level
	joint (junction)

Below are the valid destinations of a link. Several links can point at the same symbol. Several links to a joint (Vensim) is discouraged.

(Stella does not seem to support joints.)

	auxiliary
	constant ³
	level ⁴
	joint ⁵

Below is a list of attributes that will be stored for a link in the MIF file.

Field	Type	Default
Type	'link'	'link'
Symbol id	Numeric ID of this symbol (local to each view)	–
Visible	0 (no) or 1 (yes)	1

³The source (e.g., A) of a link to a constant is used in a constant expressions, e.g. INIT(A). Powersim displays initialization links using dotted lines.

⁴Stella does not draw links to levels. Hence, links to levels should be skipped by Stella.

⁵Vensim supports several links pointing at the same joint. Stella and Powersim should make every link except the first point directly to the variables pointed at by the link(s) leaving the joint.

Selected	0 (no) or 1 (yes)	0
Path	<i>p</i> followed by series of the following: line <i>p</i> arc <i>p1,p2</i> curve <i>p1,p2,p3</i>	–
Line width	1/20*1/72 inch	15 (3/4 pt)
Line color		
	<i>red</i>	0..255
	<i>green</i>	0..255
	<i>blue</i>	0..255
Source	Numeric ID of source symbol (variable or joint)	–
Destination	Numeric ID of destination symbol (variable or joint)	–

The line style is assumed to be a solid line. Powersim will display links to levels as dotted lines.

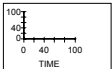
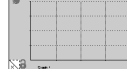

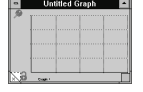
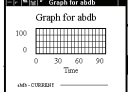
6.8.1 Delayed links

Powersim has some functions that accept delayed parameters. Links via delayed parameters can be used in a circle (feedback loop). A delayed link symbol is *required* for delayed parameters. Vensim and Stella should use normal links instead. Replacing delayed links with normal links can result in double links between two symbols. In that case the delayed link symbol should be dropped by Stella and Vensim.

As Stella does not allow for circles, even when delay functions are used, delayed links should not be used to create circles. Use explicit flow-to-level combinations instead.

6.9 Reports

The following report objects are shared by the SD packages.

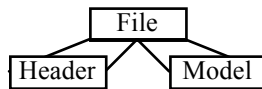
	Powersim	Stella	Vensim
Time graph (in place)			N/A
(icon)	N/A	(also has icon) 	N/A
(window)	N/A	Graph 1 	

		(also has icon)	
Time table (in place)	X	X	N/A
Time table (icon)	N/A	X	N/A
Time table (window)	N/A	X	X

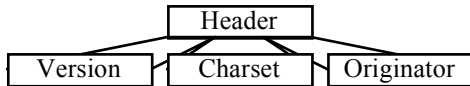
(incomplete...)

7. Structure of MIF files

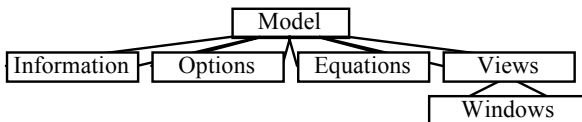
A MIF file is organized as displayed below.



The header contains information about the MIF file version, character set and originating software, as displayed below.



All models created with one version of a software will contain the same information in the MIF header. Information about the model itself is stored in the Model part. This part contains global information about the model (title, subject, author, etc.), settings for options of the software tool, model equations, and graphical views into the model.



The equations part contains a list of variable definitions. In future versions of the standard, index variables (for variables) and array dimensions will also be included here.

Views can be graphical or textual. A graphical view can contain objects that are variables, flows, links, graphs, tables, etc. A list of one or more windows is used to display views. A window can display one view at the time.

8. Language for syntax definition

In the following, the MIF syntax is described using a syntax based on the Backus-Naur Form:

Syntax	Meaning
--------	---------

'c'	A literal
<text>	A nonterminal.

a	The terminal control word a, without a parameter.
aN	The terminal control word a, with a parameter N.
a?	Item a is optional.
a+	One or more repetitions of item a.
a*	Zero or more repetitions of item a.
a b	Item a followed by item b.
a b	Item a or item b.
a & b	Item a and/or item b, in any order.

9. Syntax definition of MIF

The MIF standard is partially based on the Rich Text Format (RTF) standard, developed by Microsoft (1994). MIF is, however, not compatible with RTF, in any way.

Information is grouped in the MIF file using curly braces ({}).

Each group of data starts with a tag, which is a name preceded by a backslash, e.g. like this **\info**.

Identifiers are used to name variables and functions, etc. Identifiers can contain arbitrary characters (e.g., spaces and foreign language characters). Characters outside the range 0-9, a-z, A-Z must be quoted like this:

\a The character *a*.

Characters with ordinal numbers greater than 127 must be written using escape sequences, like this:

\hh A hexadecimal value, based on the specified character set (may be used to identify 8-bit values).

Colors are represented as rgb (red, green, blue) values, like this:

<rgb> **\rN & \gN & \bN**

9.1 The file group

At the top level a MIF file can be defined like this:

<file> '{ <header> <model> }'

9.2 The header group

The header contains global information about the file, including file version.

<header> **\mifN <charset> <originator>**
<charset> **\ansi | \mac**
<originator> '{ **\origN** ('dynamo' | 'stella' | 'powersim' | 'vensim') }'

Control Word	Meaning
--------------	---------

mif	Version of MIF file (0).
------------	--------------------------

ansi	ANSI character set, i.e., MS-Windows (default)
mac	Macintosh character set
orig	Originating software. Numeric argument holds software version times 1000.

Future versions of the MIF format may include identification for categories of models, e.g. `\array` for array models and `\discrete` for models with discrete variable types. Presence of diagram information is taken from the Views section of the Model part.

Header example, Powersim model version 2.01 for MS-Windows:

```
\mif0\ansi{\orig2010 powersim}
```

9.3 The model group

A model will have some global information in addition to the set of objects from which it is composed. Following the RTF standard, we define a model like this:

```
<model> <info?> <options>* <equations>?
        <view>* <window>*
<equations> { \equations <equation>* }
<view> { \view <viewtype> <viewsize?>
        <viewopt>* <symbol>* }
<window> {' \window <winsize> <winopt>*
        '}
```

Control Word	Meaning
equations	Group holding the equations of the model.
view	The model can have one or more views.
window	The model can have one or more windows. Each window displays a view.

9.4 The information group

This group contains title, subject, author, etc.⁶

```
<info> {' \info {' \title <string> '}' &
        {' \subject <string> '}' &
        {' \author <string> '}' &
        {' \operator <string> '}' &
        {' \keywords <string> '}' &
        {' \comment <string> '}' &
        {' \version <string> '}' &
```

⁶The definition can be taken directly from the RTF standard.

```
{' \doccomm <string> '}' &
{' \creatim <time> '}' &
{' \revtim <time> '}' &
{' \printtim <time> '}' &
{' \buptim <time> '}' &
{' \edminsN '}' & \logpixxN &
\logpixyN '}'
<time> \yrN \moN \hrN \minN
```

Control Word	Meaning
title	Title of document
subject	Subject of document
author	Author of document
operator	Person who last made changes to document
keywords	Selected key words for document
comment	Comments; text is ignored
version	Version number of document
doccomm	Comments displayed in Edit Summary Info dialog box
creatim	Creation time
revtim	Revision time
printtim	Last print time
buptim	Backup time
edmins	Total editing time in minutes
yr	Year
mo	Month
hr	Hour
min	Minute
logpixx logpixy	Screen resolution when document saved (pixels per inch)

An example of the information group follows:

```
{\info{\title The coffee cup} {\author
Arne-Helge Byrknes}}
```

9.5 Model options group

This group will hold global options for the document, including editing options, viewing options and page information. The group will normally contain native information (tags starting with an underscore (_)).

```
<options> <times> & <integr> & <pause> &
& \noedit & \nosave &
\nextlevN & \nextauxN &
```

`\nextconstN & \nextrateN & <printopt> &`
`\paperwN & \paperhN &`
`\margin & \margrN &`
`\margtN & \margbN &`
`\facignp & \gutterN &`
`\headeryN & \footeryN &`
`{' \header <string> '}` &
`{' \footer <string> '}`

from Start time of simulation.
to Stop time of simulation.
dt Time step of simulation.
unit Time unit of simulation.
runcnt Number of runs (1) to be simulated in sequence.

(Here number of runs can also be included if greater than one. The same goes for cyclic time.)

Control Word	Meaning
noedit	The user cannot edit the file, used for games.
nosave	The user cannot save the file, used for libraries.
nextlev	Number of next level (stock), auxiliary (converter), constant, rate variable when generating automatic names, e.g., Level_1, Level_2, etc.
nextaux	
nextconst	
nextrate	
paperw	Paper width in twips
paperh	Paper height in twips
margl	Left margin in twips
margr	Right margin in twips
margt	Top margin in twips
margb	Bottom margin in twips
facingp	Facing pages (off)
gutter	Extra margin space to allow for binding.
headery	Distance from top of page to header.
footery	Distance from bottom of page to footer.
header	Header text, can contain field codes for page numbering, etc.
footer	Footer text, can contain field codes for page numbering, etc.

9.6 Times group

This group specifies time horizon and time step of the simulation.

`<times> {' \times \fromN & \toN & \dtN & \runcntN & <timeunit> '}`
`<timeunit> {' \unit <ident> '}`

Control Word	Meaning
--------------	---------

9.7 Integration group

The integration method is specified by this group.

`<integr> { \integr \orderN & \varstep & \abserrN & \relerrN }`

Control Word	Meaning
order	Order of integration method
varstep	Variable step integration (off)
abserr	Absolute error limit
relerr	Relative error limit

The following integration methods are supported:

Integration method	Options
Euler	\order1
Runge Kutta 2	\order2
Runge Kutta 4	\order4
Runge Kutta 4 variable step	\order4 \varstep

Absolute and relative error limits may be specified for variable step integration.

9.8 Pause group

This group specifies automatic pauses during the simulation.

`<pause> {' \pause {' \firstsN & \everysN & \use '}` &
`{' \firsttN & \everytN & \use '}`

If omitted, no automatic pause will occur. A pause can be specified as intervals in terms of the time unit or the time step. The **use** tag determines which pause specification is active.

Control Word	Meaning
firsts	Time of first pause in steps
firstt	Time of first pause in time units
everys	Length of interval between pauses in steps

everyt Length of interval between pause in time units

use Use the pause specification if the group that holds this control word.

```
{' \dim <dimensions> '}' &
{' \def <expression> '}' &
{' \unit <unitexpr> '}' &
{' \doc <text> '}' &
{' \scale \minN & \maxN & \loN
& \hiN & \fixmax & \fixmin
& \yminN & \ymaxN '}' '}'
```

The following example will pause the simulation every time unit.

```
{\pause{\firsts0\everys10}{\firstt0\everyt1\use}}
```

This example will pause the simulation every ten time steps.

```
{\pause{\firsts0\everys10\use}
}{\firstt0\everyt1}}
```

9.9 The equations group

Each equation is defined by its own group. Only variable equations need to be specified. Unit of measure definitions can also be included.

`<equation> <unitdef>* <vardef>*`

9.10 Unit equations

Units of measure are used to document variables and optionally also to verify unit consistency for variable definitions.

```
<unitdef> {' \unit {' \name <identifier> '}' &
{' \def <unitexpr> '}' &
{' \doc <string> '}' '}'
<unitexpr> <unitfact> |
<unitfact> '*' <unitfact> |
<unitfact> '/' <unitfact> |
<unitfact> '^' <number>
<unitfact> <number> |
<name> |
{' (<unitexpr> '}'
```

A unit expression is composed from unit names and numbers that can be multiplied, divided or raised to a power. Examples include:

```
m/s
people/week
m/s^2
```

9.11 Variable equations

A variable must have a name. Unless undefined, the variable must also have a definition. The variable can be documented, and a value range (scale) can be given. An optional unit of measure can also be present. Future versions of the MIF format will include a dimensions group for specifying array variables.

`<vardef> {' \var {' \name <ident> '}' &`

Control Word	Meaning
var	Identifies a variable definition group.
name	Name of variable.
dim	Dimensions of variable, reserved for future array expansion.
def	Definition of variable.
unit	Optional unit of measure.
doc	Documentation of variable.
scale	Scale group.
min max	Minimum and maximum value of variable.
lo hi	Lowest and highest value of variable during latest simulation.
fixmin	If present, use min value as scaling. Otherwise use lo value.
fixmax	If present, use max value as scaling. Otherwise use hi value.
ymin ymax	Minimum and maximum scale of graph function's y-axis. (Powersim: Edit Graph; Stella: Become Graph).

The syntax of the right hand side of a variable definition is defined below:

```
<expr> <factor> |
<prefix op> <expr> |
<expr> <infix op> <expr> |
<expr> <postfix op>
<factor> <literal> |
<varref> |
<funcall> |
{' (<expr> '}'
<literal> <decimal number>
<varref> <ident>
<funcall> <ident> <parlist>?
<parlist> {' (<expr> (',' <expr> )* '}'
```

Unary operators:

Oper.	Prec.	Pos.	Purpose
+	8	prefix	Unary plus
-	8	prefix	Unary minus
NOT	8	prefix	Negation
!	8	postfix	Factorial
%	8	postfix	Percent

Binary operators:

Oper.	Prec.	Assoc.	Purpose
^	7	right	Raised to a power
*	6	left	Multiplication
/	6	left	Division
MOD	6	left	Remainder of division
DIVZ0	6	left	Division, zero if by zero
DIVZ1	6	left	Division, one if by zero
+	5	left	Plus
-	5	left	Minus
<	4	left	Less than
<=	4	left	Less than or equal to
>	4	left	Greater than
>=	4	left	Greater than or equal to
=	3	left	Equal to
<>	3	left	Not equal to
AND	2	left	Logical and
XOR	1	left	Logical exclusive or
OR	1	left	Logical or

In order to avoid ambiguity, parenthesis should be used when making expressions involving several different operators in a row.

Example: $((A + B) * C) \bmod 10$

9.11.1 Comments

Comments that are included in expressions will be regarded as white space (space, tab, line feed). Comments should be enclosed in apostrophes (“”).

9.11.2 Unit of measure

Literals can be followed by a unit of measure specification, which is a text string enclosed inside curly braces.

Example: Speed + 10 {m/s}

9.12 The view group

A view can either be a text view (equations) or a diagram view (symbols). Each view is controlled by a set of options, and can contain a set of symbols (variable symbols, time graphs, bars, pictures, etc.).

```
<view> { \view <viewtype> \widthN &
          \heightN & \scaleonpause
          & \noautoreports &
          <viewopt>* <symbol>* }
<viewtype> \text | \diagram
```

Control Word	Meaning
view	Identifies a view group
text	The view is a text view
diagram	The view is a accumulator flow diagram
width height	Width and height of view area in twips. The values are optional, and can be obtained by examining the contents of the view.
scaleonpause	If present redisplay reports with scaleable axis whenever the simulation is paused, after having updated lo and hi values for variable values. See <i>The equations group</i> , page 30.
noautoreports	If present, automatic reports in connection to variable symbols are disabled.

9.13 The view options

```
<viewopt> {' \title <string> '}' &
           {' \fill <rgb> '}' &
           \showpages & \showrulers &
           \nohscroll & \novscroll &
           <font> & <iconsheet> &
           <diaopt>* | <eqopt>*
```

Control Word	Meaning
title	Title of view.
fill	Color of background.
showpages	Show page borders (off)
showrulers	Show rulers (off).
nohscroll	No horizontal scroll bar (on)
novscroll	No vertical scroll bar (on)
<iconsheet>	Global icon table, see below.

In defining symbols it is possible to refer to icons by index. The **iconsheet** holds information about the icons. (Used by Vensim.)

```
<iconsheet>   '{ \iconsheetN <icon>* }'
<icon>        '{ \icon <file>? <data> }'
<file>        '{ \file <string> }'
<data>        '{ \data <hexstring> }'
```

Control Word	Meaning
iconsheet	Table with N icons.
icon	Definition of an icon.
file	Name of file holding icon.
data	Data defining icon.

9.14 Diagram view options

Diagram and equations views have different options.

```
<diaopt>   '{ \grid \snap & \unitN & \distxN
             & \distyN & \angleN }' &
             '{ \symdfit <varsym> &
             <linksym> & <flowsym> &
             <reportsym> }' &
             '{ \symhide <varsym>* &
             <linksym>* & <flowsym>* &
             <reportsym>* }' &
             '{ \autoreports <varsym>* &
             <linksym>* & <flowsym>* &
             <reportsym>* }' &
             \showundef & \showfun &
             \zoomN
```

Control Word	Meaning
grid	Grid settings group.
snap	If present, snap to grid.
unit	Unit of measure used for distx and disty , 0=pixel, 1=cm, 2=inch, 3=point (1/72 inch).
distx disty	Distance between grid lines, measured according to unit .
angle	Angular grid, measured in degrees. Zero means that angular grid off.
symdfit	Default attributes (e.g., colour) of newly created symbols. A prototype of each symbol type can be listed here.
symhide	Group containing symbol types that should be hidden. As an example, a if a link symbol is present in this group, all links

should be hidden.

autoreports

Group containing prototypes for symbol types with auto report settings to be applied on a global basis. Any other settings are ignored.

showundef

Indicate if a variable is undefined, e.g., by adding a question mark.

showfun

Indicate use special functions (table lookup, stochastic, time dependent, etc.) inside variable symbols.

zoom

Percent zooming (100)

9.15 Equations view options

```
<eqopt>   '{ \varhide 'def' & 'doc' & 'dim' &
             'scale' & 'unit' }' &
             \showtimes & \icon
             ... sort equations by ...
```

Control Word	Meaning
varhide	List of parts of a variable definition that shall not be displayed in the equations view.
showtimes	Add time specification to the equations view.
icon	Use icons instead of text to label equations.

Other options: Print size (%). Printer selection. Not defined yet.

9.16 The window group

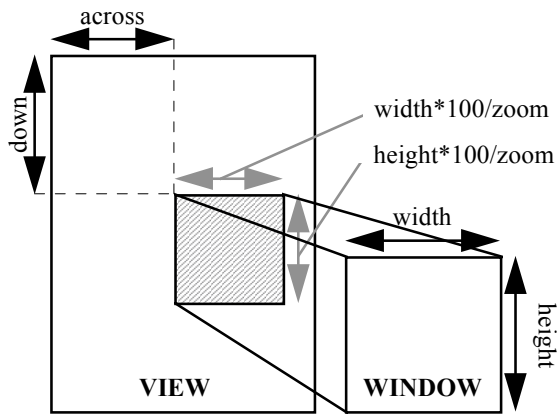
A window is used to display a view. It is possible to switch views. Therefore each window can hold options for many views. However, only one view is current, i.e., being displayed.

```
<window>   '{ \window \widthN \heightN &
             \maximize & \minimize
             <winopt>* }'
<winopt>   '{ \viewN \show? \downN
             \acrossN <viewopt>* }'
```

Control Word	Meaning
window	Identifies window group.
width height	Width and height of window in twips, in restored mode, i.e., not as maximized or minimized.

maximize	If present, the window is maximized.
minimize	If present, the window is minimized. (Should position of icon be specified here?)
view	Options for given view number, counting from one.
show	If present, this view is currently displayed in the window. Only one show control word should be present per window.
across down	Position of upper, left hand corner of window relative to view – measured in view coordinates. These values determine scrolling of window.

The mapping from view coordinates to window coordinates is depicted below.



9.17 The symbols

Symbols are identified by a type and a numeric identifier. Symbols can be visible or invisible; and part of the current selection or not.

Remaining attributes depend on the type.

```
<symbol> '{ \sym \idN \invisible &
           \selected <varattr> |
           <linkattr> | <flowattr> |
           <jointattr> | <repattr> }'
```

(Should we add clouds, valves, comments?)

Control Word	Meaning
sym	This is a symbol group.
id	Numeric identifier that is unique for this view.

invisible	If present, the symbol is to be hidden.
selected	If present, the symbol is part of the current selection.

Graphically, symbols have either closed or open shapes. Variable symbols and reports are examples of closed shapes, while links and flows are open shapes.

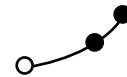
Shapes can be explicitly described using paths.

```
<path>      '{ \path <pt> <pt> ( <lineseg> |
             <arc> | <polyline> | <bezier>
             ) * }'
<lineseg>   '{ \line <pt> <pt> }'
<arc>       '{ \arc <pt> <pt> }'
<polyline>  '{ \polyline <pt> <pt> <pt>* }'
<bezier>    '{ \bezier <pt> <pt> <pt> <pt> }'
```

Control Word	Meaning
--------------	---------

path
Path describing flow. The two points identify the starting point of the path. The path primitives below display how a path is extended from its current point (displayed as an open circle).

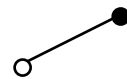
(path) arc
Arc segment of path. The current point is taken as the starting point. A point on the arc is given together with the ending point.



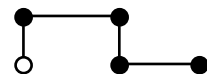
(path) bezier
Curve segment of path. Two handles and the ending point are given.



(path) line
Line segment of path. The ending point is given.



(path) polyline
Polyline segment of path. Bending points and ending point are given.



Drawing attributes for closed shapes are given below.

```

<cshape> '{ \cshape
          '{ \line \tr \wN & <rgb> '}' &
          '{ \fill \tr & <rgb> '}' &
          \auto & <path> &
          ( \none | \box | \circle |
            \hexagon | \diamond |
            \triangle | \uptriangle |
            <icon> | <bitmap> |
            <metafile> ) '}'
<icon>   '{ \iconN <file> & <data> '}'
<bitmap> '{ \bitmap <file> & <data> '}'
<metafile> '{ \metafile <file> & <data> '}'
<file>   '{ \file <string> '}'
<data>   '{ \data <hexstring> '}'

```

Control Word	Meaning
(line) tr	If present, no outline is drawn.
(line) w	Line width of outline in twips.
(line) <rgb>	Line color.
(fill) tr	If present the symbol is transparent, i.e., without fill color.
(fill) <rgb>	Fill color.
auto	Shape is determined by symbol type (default).
<path>	Shape is given by explicit path. The last point of the path is connected to the first using a straight line closing the path.
none	Shape is given predefined shape.
box	
circle	
hexagon	A clear box is represented as a box with a line that is transparent (tr).
diamond	
triangle	
uptriangle	The optional argument to icon is an index into a global icon lookup table. If <i>N</i> is omitted, the icon file or data should be given
icon	
bitmap	
metafile	
file	Name of file holding icon, bitmap, or metafile.
data	Data for current icon, bitmap, metafile. Will be used if file is not found when opening the document.

Drawing attributes for open shapes are given below.

```

<oshape> '{ \oshape
          '{ \line \wN & <rgb> '}' &

```

```

<path> '}'

```

Control Word	Meaning
(line) w	Line width of outline in twips.
(line) <rgb>	Line color.
<path>	Path describing shape. The path is not closed.

9.18 Variable attributes

Automatic reports (animation) can be set to graph, number and slider. See *Diagram view options*, page 32.

```

<varattr> '{ \type 'var' '}' &
          '{ \subtype ( 'level' | 'aux' |
                       'const' ) '}' &
          \rate & \snapshot & \acrossN &
          \downN & \widthN &
          \heightN & \autograph &
          \autonom & \autoslider &
          <cshape> &
          '{ \name <ident> '}' &
          '{ \nampos \inside & \angleN?
            <font> '}' '}'

```

Control Word	Meaning
type	Symbol type is variable.
subtype	Kind of variable.
rate	If present, the variable is used as a rate, i.e., controlling a flow.
snapshot	If present the symbol is a snapshot (alias, ghost, shadow).
across	
down	Position of center of symbol measured in twips.
width	
height	Size of symbol bounding box, excluding name that can be outside of symbol.
autograph	Display a graph, number or slider representing the symbol's current value.
autonom	
autoslider	
<cshape>	Shape of symbol.
name	Name of variable associated with symbol.
(nampos) inside	If present the name is placed inside of the symbol.
(nampos) angle	Angle from center of symbol to center of name text measured in

degrees. See Figure 3, page 24.

9.19 Flow attributes

```
<flowattr> '{ \type 'flow' }' &
           \srcN & \destN & \rateN &
           <oshape> &
           '{ \valve <pt> }'
```

Control Word	Meaning
type	Symbol type is flow.
src dest	Numeric id of source and destination symbol, which must be level (cloud if omitted).
rate	Numeric id of symbol controlling valve. The symbol must be a variable or a link.
<oshape>	Shape of flow.
valve	Position of valve symbol in twips.

Note that current tools only support poly lines for flow paths, and that lines must be either horizontal or vertical. This may be changed in the future.

9.20 Link attributes

```
<linkattr> '{ \type 'link' }' &
           \srcN \destN & \init & \delay &
           <oshape>
```

Control Word	Meaning
type	Symbol type.
src dest	Numeric id of source and destination symbol, which must be variables or joints.
init	If present, it is an initialization link.

delay If present, it is a delayed link.

<oshape> Shape of link.

9.21 Joint attributes

```
<jointattr> '{ \type 'joint' }' &
           '{ \src (<id> ';')* <id> }' &
           '{ \dest (<id> ';')* <id> }' &
           \acrossN & \downN & <cshape>
```

Control Word	Meaning
type	Symbol type is joint.
src	List of source link numbers, currently restricted to one.
dest	List of destination link numbers.
across down	Position of center of symbol measured in twips.
<cshape>	Shape of joint.

9.22 Report attributes

This group is not completed...

```
<repattr> '{ \type 'report' }' &
           '{ \title <string> }' &
           \acrossN & \downN & \widthN
           & \heightN
           ( <tableattr> | <graphattr> |
           <barattr> | <numberattr> |
           <pictureattr> | <lineattr> )
```

10. References

Microsoft Corporation 1994. *Rich Text Format (RTF) Specification*