Dynamic Simulator Interface Definition

Dr. Turhan Demiray

October 21, 2019

1 Symbolic Definition File

All mathematical models of the Dynamic Simulator are described by differential, switched algebraic and state-reset (DSAR) equations which is known as:

$$\dot{x} = f(x, y, z, \lambda) \tag{1}$$

$$\dot{z} = 0 \tag{2}$$

$$0 = g^{(0)}(x, y, z, \lambda) \tag{3}$$

$$0 = \begin{cases} g^{(i^-)}(x, y, z, \lambda) & y_{s,i} < 0 \\ g^{(i^+)}(x, y, z, \lambda) & y_{s,i} > 0 \end{cases} \quad \text{i = 1,..., s} \tag{4}$$

$$z^+ = h_j(x^-, y^-, z^-, \lambda) \quad y_{r,j} = 0 \quad \text{j = 1,..., r} \tag{5}$$

It can easily be seen, that (1) describes the differential equations, (3) and (4) the so called switched algebraic equations and (5) the state reset equations, where

- x are continuous dynamic states e.g. Rotor frequency (ω) of synchronous machines
- z are discrete states e.g. Tap position of the tap-changing transformer (tap).
- y are algebraic states e.g. Bus voltage magnitudes ($|V|$)
- λ are parameters e.g. Controller gains (K_{pss}), line reactances

of the system. The superscript $^-$ stands for pre-event values and $^+$ for post event values. At the beginning, the system behavior is described by the DAE given in (1) and (3). A transition to another set of DAE takes place if the corresponding transition condition is fulfilled. Such transition conditions are checked by means of so called *event variables* y_s and y_r. The y_s determine the switching events and y_r state reset events. An event is triggered by an element of y_s changing sign and/or an element of y_r passing through zero. By switching events, which are caused by y_s sign changes, the functional description of the system is changed from $g^{(i^-)}(x, y)$ to $g^{(i^+)}(x, y)$. If we look at the formulation (2) and (5), we see that the discrete states z are constant between events ($\dot{z} = 0$) and at state reset events caused by y_r the values of the discrete states change according to the state reset functions h_j. At state reset events, the values of dynamic states x are continuous, which is most often the case in physical systems. The equations (1)-(5) capture all the important aspects of a hybrid system.

The simulation kernel needs the models to be described in the DSAR structure, and for the numerical simulation each model has to provide the kernel with the model functions

f, g, h , with their partial derivatives f_x, f_y, g_x, g_y and the event variables y_s, y_r for event handling.

Definitions of the model descriptors f, g and h are normally known by the user. But sometimes the analytical calculation of the required partial derivatives (f_x, f_y, g_x, g_y) can be really time consuming. To ease the model creation for the user, an automatic code generation tool has been implemented called **tda_create_symbolically_normal**

The user simply writes the model equations in the required DSAR structure in a text file called *Symbolic Definition File* (SYMDEF), by defining the variables

- continuous dynamic states x
- discrete states z
- algebraic states y
  - inputs
  - external states
  - internal states
- event variables y_r and y_s
- parameters λ

and equations

- differential equations f
- switched-algebraic equations g
- state-reset equations h

tda_create_symbolically_normal processes the Symbolic Definition File of the model and creates a matlab code of the model with the necessary interface functions.

As an example, let us try to define the Symbolic Definition File of a Constant Gain Voltage Regulator depicted in Figure 1. As shown in Figure 1 the controller has following

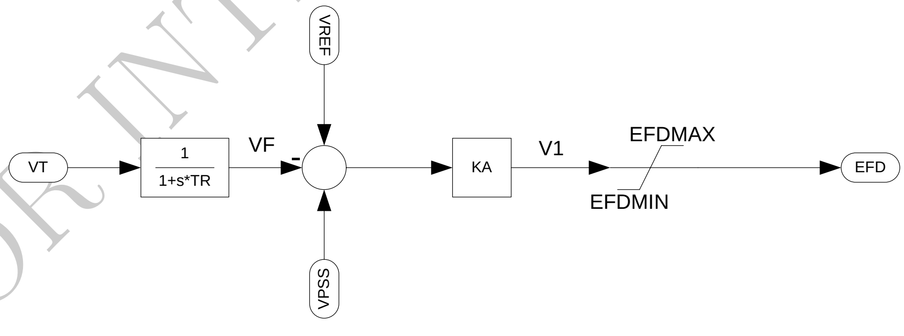


Figure 1: Constant Gain AVR with limits

variables:

- continuous dynamic states: x_1
- inputs: V_T,V_{PSS},V_{REF}
- external states: E_{FD}

- internal states: V_F,V_1
- event variables: ev_{min} and ev_{max}
- parameters: T_R, K_A, E_{FD}^{min} and E_{FD}^{max}

The template for a SYMDEF file shown below. The symbolic definition file has different sections.

- Section starting with the label ***definitions:*** contains all the variables and parameters of the model with theri specific types.
- In the section starting with the label ***f_equations:***, the first order ordinary differential equations f of the model are given, where the [***dt()***] stands for the derivative operator.
- Section starting with the label ***g_equations:*** comprises all the switched-algebraic equations g of the model. The g-equations must have unique names (e.g. g_1, g_2, g_3).
- And finally, section starting with the label ***h_equations:*** comprehends the state reset equations h of the model.

In the definitions section, different keywords are used to define the correctly the variables and parameters of the model, namely:

- continuous dynamic states ... x ... [***dynamic_states***]
- inputs ... y_{ext} ... [***inputs***]
- external states ... y_{ext} ... [***external_states***]
- internal states ... y_i ... [***internal_states***]
- event variables ... y_r, y_s ... [***events***]
- parameters ... λ ... [***parameters***]

The Symbolic Definition File of a Constant Gain Voltage Regulator could be defined as:

Note: The exclamation mark "!" signs in front of event variable tells the simulator kernel to stop and to re-initialize the system if this event (a sign change in this variable) is recognized. The $+/-$ signs in front of event variable gives the direction of the sign change, when the event is triggered. A $+$ sign means, an event will be triggered if the event variable changes sign from $-$ to $+$.

Another possibility for writing your models in a SYMDEF file is to use predefined macros. There is a library of predefined macros which eases the model description for the user by using standard transfer function.

For example the same controller written with the macros would look like as follows:

2 Blocks for Symdef

1.1 Integrator

Transferfunction:

$$y_1 = \frac{K}{s} \cdot u_1$$

Function call:

y1 = integ(u1,x1,K,min,max,expr)

Function arguments:

y1 ....output
u1 ....input
x1 ....state
K......................................parameter
min ....lower limit [“none” if no lower limit exists lower limit exists]
max..................................upper limit [“none” if no lower limit exists upper limit exists]
expr ....if empty, automatic calculation of initial value of x1, if not the initial value of x1 is calculated such that expr=0 is fulfilled.

Limit Type:Non-windup

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
dynamic_states dV=1
internal_states V=1 U1
parameters K=10 VMIN V1MAX

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = V - integ(U1,dV,K,VMIN,VMAX) % with min max limits

g1 = V - integ(U1,dV,K,VMIN,VMAX,dV-2.15) % initialize such that
                                            (dV-2.15)=0 -> dV=2.15
```

1.2 LAG

Transferfunction:

$$y_1 = \frac{K}{1+sT} \cdot u_1$$

Function call:

y1 = lag(u1,x1,K,T,min,max)

Function arguments: y1 *output*

u1 ....input
x1 ....state
K.....................................parameter
T.....................................parameter
min ....lower limit [“none” if no lower limit exists lower limit exists]
max.................................upper limit [“none” if no lower limit exists upper limit exists]
expr ....if empty, automatic calculation of initial value of x1, if not the initial value of x1 is calculated such that expr=0 is fulfilled.

Limit Type:Non-windup

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
dynamic_states dV=1
internal_states V=1 U1 Y1
parameters K=10 T=0.01 VMIN V1MAX

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = V - lag(U1,dV,K,T,VMIN,VMAX) % with min max limits

g1 = V - lag(U1,dV,K,T,VMIN,VMAX,Y1-U1) %
```

1.3 DERLAG

Transferfunction:

$$y_1 = \frac{K \cdot sT_D}{1 + sT_D} \cdot u_1$$

Function call:

y1 = derlag(u1,x1,K,TD,min,max)

Function arguments:

y1 ....output
u1 ....input
x1 ....state
K......................................parameter
TD....................................parameter
min ....lower limit [“none” if no lower limit exists lower limit exists]
max..................................upper limit [“none” if no lower limit exists upper limit exists]

Limit Type:Windup

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
dynamic_states dV=1
internal_states V=1 U1
parameters KA=10 TA=0.01 VMIN V1MAX

%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = V - derlag(U1,dV,K,TD,VMIN,VMAX) % with min max limits
```

1.4 LEADLAG

Transferfunction:

$$y_1 = K \cdot \frac{1 + sT_Z}{1 + sT_N} \cdot u_1$$

Function call:

y1 = leadlag(u1,x1,K,TZ,TN,min,max)

Function arguments:

y1 ....output
u1 ....input
x1 ....state
K.......................................parameter
TZ ....parameter
TN ....parameter
min ....lower limit [“none” if no lower limit exists lower limit exists]
max...................................upper limit [“none” if no lower limit exists upper limit exists]

Limit Type:Windup

Example:

```
%--------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------
dynamic_states dV=1
internal_states V=1 U1
parameters K=10 TZ=0.01 TN=0.01 VMIN V1MAX

%--------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------
g1 = V - leadlag(U1,dV,K,TZ,TN,VMIN,VMAX) % with lower and upper limits
```

1.5 Rationale Transfer NTH Order

Transferfunction:

$$y_1 = \frac{b_0 + b_1 \cdot s + b_2 \cdot s^2 + ... + b_N \cdot s^N}{a_0 + a_1 \cdot s + a_2 \cdot s^2 + ... + a_N \cdot s^N} \cdot u_1$$

Function call:

y1 = rN(u1,x1,x2,...,xN,A0,A1,A2…AN,B0,B1,B2…BN,min,max)
N = {2,3,4}

Function arguments:

y1 ....output
u1 ....input
x1-xN ....state
A0-AN ....parameter
B0-BN ....parameter
min ....lower limit ["none" if no lower limit exists lower limit exists]
max.................................upper limit ["none" if no lower limit exists upper limit exists]

Limit Type:Windup

Example:

```
%------------------------------------------------------------------------
definitions:
%------------------------------------------------------------------------
dynamic_states x1 x2 x3 ..XN=1
internal_states V=1 U1
parameters A0=1 A1 …… AN B0 B1 B2 ……BN VMIN V1MAX

%------------------------------------------------------------------------
g_equations:
%------------------------------------------------------------------------
g1 = V – r2(U1,x1,x2,A0,A1,A2,B0,B1,B2,VMIN,VMAX)
g1 = V – r3(U1,x1,x2,x3,A0,A1,A2,A3,B0,B1,B2,B3,VMIN,VMAX)
g1 = V – r4(U1,x1,x2,x3,x4,A0,A1,A2,A3,A4,B0,B1,B2,B3,B4,VMIN,VMAX)
```

1.6 TABLE

Transferfunction:

y1 = lookup(u1)

Function call:

y1 = table(u1,X1,Y1,X2,Y2,……XN,YN)

Function arguments:

y1 ....output
u1 ....input
K......................................parameter
X1 Y1 ....point pairs

Example:

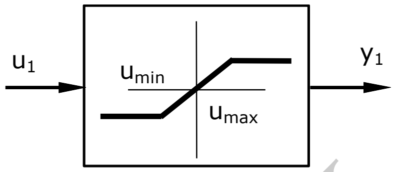
```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states U1=1 Y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = Y1 - table(U1,0.0,0.0,1.0,0.8,1.1,0.85) % with lower and upper limits
```

1.7 Limit

Transferfunction:

$$u_1 \geq u_{\max} \qquad y_1 = u_{\max}$$
$$u_{\min} < u_1 < u_{\max} \qquad y_1 = u_1$$
$$u_1 \leq u_{\min} \qquad y_1 = u_{\min}$$



Function call:

y1 = limit(u1,min,max)

y1 = limit(u1,min,max, nolimitforinit)

Function arguments:

y1 ....output
u1 ....input
min ....lower limit [“none” if no lower limit exists lower limit exists]
max.................................upper limit [“none” if no lower limit exists upper limit exists]
nolimitforinit ....if 1 limits will be ignored during initialization

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1=1 y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - limit(u1,0.0,1.0) % with lower and upper limits
g1 = y1 - limit(u1,0.0,1.0,1) % with lower and upper limits and the limits
will be ignored during initialization
```

1.8 LV GATE

Transferfunction:

$$y_1 = \min(u1, u2)$$

Function call:

y1 = min(u1,u2)

Function arguments:

y1 ....output
u1 ....input
u2 ....input

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 u2 y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - min(u1,u2)
```

1.9 HV GATE

Transferfunction:

$$y_1 = \max(u1, u2)$$

Function call:

y1 = max(u1,u2)

Function arguments:

y1 ....output
u1 ....input
u2 ....input

Example:

```
%----------------------------------------------------------------------------
definitions:
%----------------------------------------------------------------------------
internal_states u1 u2 y1

%----------------------------------------------------------------------------
g_equations:
%----------------------------------------------------------------------------
g1 = y1 - max(u1,u2)
```

1.10 SATURATION QUADRATIC

Transferfunction:

if u_1 > max : y_1 = max

if u_1 < min : y_1 = min

otherwise $y_1 = u_1$

Function call:

y1 = saturation1 (u1,E1,SE1,E2,SE2)

Function arguments:

y1 ....output
u1 ....input
E1 ....parameter
SE1 ....parameter
E2 ....parameter
SE2 ....parameter

Example:

```
%-------------------------------------------------------------------------------
definitions:
%-------------------------------------------------------------------------------
internal_states u1 y1

%-------------------------------------------------------------------------------
g_equations:
%-------------------------------------------------------------------------------
g1 = y1 - saturation1(u1,E1,SE1,E2,SE2) %
```

1.11 STATIC EXCITER

Transferfunction:

Function call:

y1 = statexc (u1)

Function arguments:

y1 ....output
u1 ....input

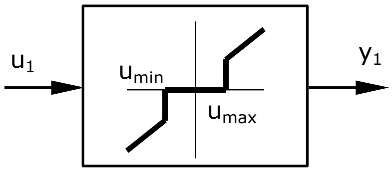
Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 y1

%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - statexc (u1) %
```

1.12 DEADBAND

Transferfunction:



$$u_1 \geq u_{max} \qquad y_1 = K_1 \cdot u_1 - K_2 \cdot u_{max}$$
$$u_{min} < u_1 < u_{max} \qquad y_1 = 0$$
$$u_1 \leq u_{min} \qquad y_1 = K_1 \cdot u_1 - K_2 \cdot u_{min}$$

Function call:

y1 = deadband (u1,K1,K2,min,max)

Function arguments:

y1 ....output
u1 ....input
E1 ....parameter
SE1 ....parameter
E2 ....parameter
SE2 ....parameter

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 y1
parameters K1 K2 min max
%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - deadband (u1,K1,K2,min,max) %
```

1.13 Absolute Value

Transferfunction:

$$u_1 > 0 \quad y_1 = u_1$$
$$u_1 < 0 \quad y_1 = -u_1$$

Function call:

y1 = absval (u1)

Function arguments:

y1 ....output
u1 ....input

Example:

```
%-----------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------
internal_states u1 y1

%-----------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------
g1 = y1 - absval(u1) %
```

1.14 SIGNUM

Transferfunction:

Function call:

y1 = absval (u1)

Function arguments:

y1 ....output
u1 ....input

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 y1

%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - signval(u1) %
```

1.15 MAGNITUDE

Transferfunction:

Function call:

y1 = magnitude (u1,u2,K)

Function arguments:

y1 ....output
u1 ....input
u2 ....input
K.......................................parameter

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 u2 y1
parameters K1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - magnitue(u1,u2,K1) %
```

1.16 SATURATION1

Transferfunction: Quadratic Saturation Function

Function call:

y1 = saturation (u1, E1,SE1,E2,SE2)

Function arguments:

y1 ....output
u1 ....input
E1 ....parameter
SE1 ....parameter
E2 ....parameter
SE2 ....parameter

Example:

```
%-------------------------------------------------------------------------
definitions:
%-------------------------------------------------------------------------
internal_states u1 y1
parameters K1

%-------------------------------------------------------------------------
g_equations:
%-------------------------------------------------------------------------
g1 = y1 - saturation1(u1,E1,SE1,E2,SE2) %
```

1.17 AND

Transferfunction:

y1 = 1 if all inputs are larger than 0.5

otherwise y1 = 0

Function call:

y1 = and (u1,u2,….un)

Function arguments:

y1 ....output
u1..un ....input

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - and(u1,u2,u3) %
```

1.18 BOOL AND

Transferfunction:

assumes that all inputs are boolean variables 0 or 1

y1 = 1 if all inputs are 1

otherwise y1 = 0

Function call:

y1 = bool_and (u1,u2,….un)

Function arguments:

y1 ....output
u1..un ....input

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - bool_and(u1,u2,u3) %
```

1.19 OR

Transferfunction:

y1 = 1 if one of the inputs are larger than 0.5

otherwise y1 = 0

Function call:

y1 = or(u1,u2,….un)

Function arguments:

y1 ....output

u1..un ....input

Example:

```
%--------------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%--------------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------------
g1 = y1 - or(u1,u2,u3) %
```

1.20 BOOL OR

Transferfunction:

assumes that all inputs are boolean variables 0 or 1

y1 = 1 if one of the inputs is 1

otherwise y1 = 0

Function call:

y1 = bool_or(u1,u2,….un)

Function arguments:

y1 ....output
u1..un ....input

Example:

```
%----------------------------------------------------------------------------
definitions:
%----------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%----------------------------------------------------------------------------
g_equations:
%----------------------------------------------------------------------------
g1 = y1 - bool_or(u1,u2,u3) %
```

1.21 SELECT

Transferfunction:

if u1 > 0.5

y1 = u2

else

y1 = u3

Function call:

y1 = select(u1,u2,u3)

Function arguments:

y1 ....output

u1..u3 ....input

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - select(u1,u2,u3) %
```

1.22 Bool Select

Transferfunction:

assumes that u1 is a boolean variable 0 or 1

if u1==1

y1 = u2

else

y1 = u3

Function call:

y1 = bool_select(u1,u2,u3)

Function arguments:

y1 ....output
u1..u3 ....input

Example:

```
%--------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------
internal_states u1 u2 u3 y1

%--------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------
g1 = y1 - bool_select(u1,u2,u3) %
```

1.23 ISEQUAL

Transferfunction:

```
if u1==u2
    y1 = 1
else
    y1 = 0
```

Function call:

```
y1 = isequal(u1,u2)
```

Function arguments:

y1 ....output
u1..u2 ....inputs

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 u2 y1

%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - isequal(u1,u2) %
```

1.24 LESS ZERO

Transferfunction:

```
if u1<0
    y1 = 1
else
    y1 = 0
```

Function call:

y1 = less_zero(u1)

Function arguments:

y1 ....output
u1 ....inputs

Example:

```
%------------------------------------------------------------------------------
definitions:
%------------------------------------------------------------------------------
internal_states u1 y1

%------------------------------------------------------------------------------
g_equations:
%------------------------------------------------------------------------------
g1 = y1 - less_zero(u1) %
```

1.25 Less Or Equal Zero

Transferfunction:

if u1<=0

y1 = 1

else

y1 = 0

Function call:

y1 = less_or_eq_zero (u1)

Function arguments:

y1 ....output

u1 ....inputs

Example:

```
%----------------------------------------------------------------------------
definitions:
%----------------------------------------------------------------------------
internal_states u1 y1

%----------------------------------------------------------------------------
g_equations:
%----------------------------------------------------------------------------
g1 = y1 - less_or_eq_zero (u1) %
```

1.26 Greater Zero

Transferfunction:

```
if u1>0
    y1 = 1
else
    y1 = 0
```

Function call:

```
y1 = greater_zero(u1)
```

Function arguments:

y1 ....output
u1 ....inputs

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 y1

%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - greater_zero(u1) %
```

1.27 GREATER OR EQUAL ZERO

Transferfunction:

```
if u1>=0
    y1 = 1
else
    y1 = 0
```

Function call:

y1 = greater_or_eq_zero (u1)

Function arguments:

y1 ....output
u1 ....inputs

Example:

```
%--------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------
internal_states u1 y1

%--------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------
g1 = y1 - greater_or_eq_zero (u1) %
```

1.28 DISRETE STATE CHANGE

Function call:

y1 = changestate (u1,u2,u3,z1)

Transferfunction:

if u1 changes from 0 to 1, the new value of the discrete state z1,will be set to u2.

y1 = z1

with u3 the initial value of the discrete state z1 is set

Function arguments:

y1 ....output
u1 ....input
u2 ....input
u3 ....input
z1 ....discrete state

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
dynamic_states z1
internal_states WAITDONE TAPPOSITION
parameters TAPSTEP=0.0125 TAP0=1
%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = TAPPOSITION - changestate(WAITDONE,z1+TAPSTEP,TAP0,z1)%
```

1.29 HIT VALUE

Function call:

y1 = hit(u1,value,direction,duration)

Transferfunction:

if u1 hits the "value" parameter when rising "direction=1" or "direction=-1"falling, an impluse .of "duration" will be generated

Function arguments:

y1 ....output
u1 ....input
value ....Parameter
direction............................Parameter
duration ....Parameter

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 VHITMIN VHITMAX
%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = VHITMIN - hit(u1,0.95,-1,0.001)% hit while falling
g2 = VHITMAX - hit(u1,1.1,1,0.001)% hit while rising
```

1.30 RELAY

Function call:

y1 = relay(u1,minval,maxval,minout,maxout)

Transferfunction:

If u1 is greater than an adjustable value maxval, the output y1 is maxout. If y1 was maxout, y1 changes to minout, if u1 becomes less than minval

Function arguments:

y1 ....output
u1 ....input
minval ....Parameter
maxval.............................Parameter
minout ....Parameter
maxout ....Parameter

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 y1
parameters minval maxval minout maxout
%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - relay(u1,minval,maxval,minout,maxout)
```

1.31 Pickup

Function call:

y1 = pickup (u1,delay)

Transferfunction:

A pickup of u1 from 0 to 1 is transferred to the output y1 delayed by a time interval "delay". If u1 resets to 0 before, y1 is not changed.

Function arguments:

y1 ....output
u1 ....input
delay ....Parameter

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 y1
parameters delay
%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - pickup(u1,delay)
```

1.32 RESET

Function call:

y1 = reset (u1,delay)

Transferfunction:

A reset of u_1 from 1 to 0 is transferred to the output y_1 delayed by a time interval Δt.

Function arguments:

y1 ....output
u1 ....input
delay ....Parameter

Example:

```
%-----------------------------------------------------------------------------
definitions:
%-----------------------------------------------------------------------------
internal_states u1 y1
parameters delay
%-----------------------------------------------------------------------------
g_equations:
%-----------------------------------------------------------------------------
g1 = y1 - pickup(u1,delay)
```

1.33 PickupReset

Function call:

y1 = pickupreset (u1,Tpickup,Treset)

Transferfunction:

Combination of the Pickup and reset blocks.

Function arguments:

y1 .... output
u1 .... input
Tpickup .... Parameter
Treset .... Parameter

Example:

```
%-------------------------------------------------------------------------------
definitions:
%-------------------------------------------------------------------------------
internal_states u1 y1
parameters Tpickup Treset
%-------------------------------------------------------------------------------
g_equations:
%-------------------------------------------------------------------------------
g1 = y1 - pickupreset(u1,Tpickup,Treset)
```

1.34 TIMER

Function call:

y1 = timer(start,stop)

Transferfunction:

if “start” variable changes from 0 to 1 a timer will start and y1 is the time value.

if “stop” variable changes from 0 to 1 a timer will stop and the time value will be reset to 0

Function arguments:

y1 ....output
start ....input
stop.................................input

Example:

```
%--------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------
internal_states TIMERVALUE TIMERSTART TIMERSTOP
%--------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------
g1 = TIMERVALUE - timer(TIMERSTART,TIMERSTOP)
```

1.35 SAMPLE DELAY

Transferfunction:

Samples the input every TSAMPLE seconds and an dthe output is equal to the delay input by NDELAY samples

Function call:

y1 = sampledelay(u1,TSAMPLE,NDELAY)

Function arguments:

y1 ....output
u1 ....input
TSAMPLE ....parameters
NDELAY.............................parameters

Example:

```
%---------------------------------------------------------------------------
definitions:
%---------------------------------------------------------------------------
internal_states u1 y1
parameters TSAMPLE NDELAY

%---------------------------------------------------------------------------
g_equations:
%---------------------------------------------------------------------------
g1 = y1 - sampledelay(u1, TSAMPLE,NDELAY) %
```

1.36 TABLE

Transferfunction:

approximates the input according to the 2 dimensional array X and Y

Function call:

y1 = table(u1,X1,Y1,X2,Y2,....XN,YN)

Function arguments:

y1 ....output
u1 ....input
X1 ....parameters
Y1 ....parameters
XN....................................parameters
YN....................................parameters

Example:

```
%-------------------------------------------------------------------------
definitions:
%-------------------------------------------------------------------------
internal_states u1 y1
parameters X1 Y1 X2 Y2 X3 Y3

%-------------------------------------------------------------------------
g_equations:
%-------------------------------------------------------------------------
g1 = y1 - table(u1, X1,Y1,X2,Y2,X3,Y3) %
```

1.37 PI CTRL

Transferfunction:

$$y_1 = \left(K_P + \frac{K_I}{s} \right) \cdot u_1$$

Function call:

y1 = pictrl(u1,x1,KP,KI,min|[none],max|[none])

y1 = pictrl(u1,x1,KP,KI,min|[none],max|[none],nolimitforinit=1)

y1 = pictrl(u1,x1,KP,KI,min|[none],max|[none],nolimitforinit=1,x0)

Function arguments:

y1 ....output
u1 ....input
x1 ....state
KP ....parameter
KI.....................................parameter
min ....lower limit [“none” if no lower limit exists lower limit exists]
max...................................upper limit [“none” if no lower limit exists upper limit exists]

Example:

```
%--------------------------------------------------------------------------
definitions:
%--------------------------------------------------------------------------
dynamic_states dV=1
internal_states V=1 U1
parameters Kp=10 Ki=0.1 VMIN V1MAX

%--------------------------------------------------------------------------
g_equations:
%--------------------------------------------------------------------------
g1 = V - pictrl(U1,dV,Kp,Ki,VMIN,VMAX) % with min max limits

g1 = V - pictrl (U1,dV,Kp,Ki,VMIN,VMAX, dV-2.15) % initialize such that
                                            (dV-2.15)=0 -> dV=2.15
```