Model Equations

Version 1.0.0

# Block: UEL\_OEL\_Router\_1\_Type (UEL\_OEL\_Router\_1)

* Inputs: Voel, Vuel
* Outputs: voel1, vuel1, voel2, vuel2, voel3, vuel3
* Parameters: Voel\_in, Vuel\_in
* Initialisation and Simulation equations

if Vuel\_in>=1.5 then

vuel1 := Vuel;

else

vuel1 := -99999;

end if;

if (Vuel\_in>=0.5) and (Vuel\_in<1.5) then

vuel2 := Vuel;

else

vuel2 := -99999;

end if;

if Vuel\_in>=0.5 then

vuel3 := 0;

else

vuel3 := Vuel;

end if;

if Voel\_in>=1.5 then

voel1 := Voel;

else

voel1 := 99999;

end if;

if (Voel\_in>=0.5) and (Voel\_in<1.5) then

voel2 := Voel;

else

voel2 := 99999;

end if;

if Voel\_in>=0.5 then

voel3 := 0;

else

voel3 := Voel;

end if;

# Block: SCL\_Router\_1\_Type (SCL\_Router\_1)

* Inputs: Vscl\_sum, Vscl\_uel, Vscl\_oel
* Outputs: vscl\_sum, vscl\_uel\_to1, vscl\_oel\_to1, vscl\_uel\_to2, vscl\_oel\_to2
* Parameters: Vscl\_in
* Initialisation and Simulation equations

if Vscl\_in>0.5 then

vscl\_sum := 0;

else

vscl\_sum := Vscl\_sum;

end if;

if (Vscl\_in>=0.5) and (Vscl\_in<1.5) then

vscl\_uel\_to1 := Vscl\_uel;

else

vscl\_uel\_to1 := -99999;

end if;

if (Vscl\_in>=0.5) and (Vscl\_in<1.5) then

vscl\_oel\_to1 := Vscl\_oel;

else

vscl\_oel\_to1 := 99999;

end if;

if Vscl\_in>=1.5 then

vscl\_uel\_to2 := Vscl\_uel;

else

vscl\_uel\_to2 := -99999;

end if;

if Vscl\_in>=1.5 then

vscl\_oel\_to2 := Vscl\_oel;

else

vscl\_oel\_to2 := 99999;

end if;

# Block: add\_1

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

===================================

# Block: add3\_1

* Inputs: u1, u2, u3
* Outputs: y
* Parameters: k1, k2, k3
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2 + k3\*u3;

# Block: add\_2

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: HV\_gate\_1\_Type (HV\_gate\_1)

* Inputs: vuel\_to\_B, yi2, vscl\_uel\_B
* Outputs: yo
* Parameters:
* Initialisation and Simulation equations

yo := max(vuel\_to\_B, max(yi2, vscl\_uel\_B));

# Block: LV\_gate\_1\_Type (LV\_gate\_1)

* Inputs: voel\_to\_B, yi2, vscl\_oel\_B
* Outputs: yo
* Parameters:
* Initialisation and Simulation equations

yo := min(voel\_to\_B, min(yi2, vscl\_oel\_B));

# Block: AsymmetricalLimiter\_1\_Type (AsymmetricalLimiter\_1)

* Inputs: u
* Outputs: y
* Parameters: V\_min, V\_max
* Initialisation and Simulation equations

y := if u > V\_max then V\_max else if u < V\_min then V\_min else u;

# Block: LeadLagTgTf (leadLagTgTf\_1)

* Inputs: u, x0\_extTfTg
* Outputs: y
* Parameters: Tf, Tg, x0, InitCondition
* Internal variables: x
* Initialisation equations

// Initialise

if InitCondition == 1 then //Initialise from internal parameter, x0

x := x0;

elseif InitCondition == 2 then //Initialise from external input, x0\_extTfTg

x := x0\_extTfTg;

end if;

y := (1-(Tg/Tf))\*x+(Tg/Tf)\*u;

* Simulation equations

x := previous(x) + (interval()/Tf)\*(previous(u)-previous(x));

y := (1-(Tg/Tf))\*x+(Tg/Tf)\*u;

# Block: add\_3

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: add\_1\_1

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: gainKpa\_1

* Inputs: u
* Outputs: y
* Parameters: Kpa, inverseGain
* Initialisation and Simulation equations

if inverseGain then

y := u/Kpa;

else

y := Kpa \* u;

end if;

# Block: AsymmetricalLimiter\_2\_Type (AsymmetricalLimiter\_2)

* Inputs: u
* Outputs: y
* Parameters: Vr\_min
* Initialisation and Simulation equations

y := if u < Vr\_min then Vr\_min else u;

# Block: add\_1\_2

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: AsymmetricalLimiter\_3\_Type (AsymmetricalLimiter\_3)

* Inputs: u
* Outputs: y
* Parameters: Vr\_max
* Initialisation and Simulation equations

y := if u > Vr\_max then Vr\_max else u;

# Block: add\_1\_3

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: gainKl\_1

* Inputs: u
* Outputs: y
* Parameters: Kl, inverseGain
* Initialisation and Simulation equations

if inverseGain then

y := u/Kl;

else

y := Kl \* u;

end if;

# Block: gainKh\_1

* Inputs: u
* Outputs: y
* Parameters: Kh, inverseGain
* Initialisation and Simulation equations

if inverseGain then

y := u/Kh;

else

y := Kh \* u;

end if;

# Block: LowPassFilterKiaTia (lowPassFilterKiaTia\_1)

* Inputs: u, x0\_ext
* Outputs: y
* Parameters: Kia, Tia, x0, InitCondition
* Internal variables: x
* Initialisation equations

if InitCondition == 1 then //Initialise from internal parameter, x0

x := x0;

elseif InitCondition == 2 then //Initialise from external input, x0\_ext

x := x0\_ext;

end if;

y := x;fTg;

end if;

y := (1-(Tg/Tf))\*x+(Tg/Tf)\*u;

* Simulation equations

x := previous(x)+(interval()/Tia)\*(Kia\*previous(u)-previous(x));

y := x;

# Block: UnitDelay (unitDelay\_1)

* Inputs: u
* Outputs: y
* Parameters: y\_start
* Initialisation equations

y := y\_start;

* Simulation equations

y := previous(u);

# Block: max\_1

* Inputs: u1, u2
* Outputs: y
* Parameters:
* Initialisation and Simulation equations

y := max(u1,u2);

# Block: max\_2

* Inputs: u1, u2
* Outputs: y
* Parameters:
* Initialisation and Simulation equations

y := max(u1,u2);

# Block: LeadLagTcTb (leadLagTcTb\_1)

* Inputs: u, x0\_ext
* Outputs: y
* Parameters: Tb, Tc, x0, InitCondition
* Internal variables: x
* Initialisation equations

if InitCondition == 1 then //Initialise from internal parameter, x0

x := x0;

elseif InitCondition == 2 then //Initialise from external input, x0\_ext

x := x0\_ext; //Efd\*(1-Kia)

end if;

y := (1-(Tc/Tb))\*x+(Tc/Tb)\*u;

* Simulation equations

x := previous(x) + (interval()/Tb)\*(previous(u)-previous(x));

y := (1-(Tc/Tb))\*x+(Tc/Tb)\*u;

# Block: add\_4

* Inputs: u1, u2
* Outputs: y
* Parameters: k1, k2
* Initialisation and Simulation equations

y := k1\*u1 + k2\*u2;

# Block: HV\_gate\_2\_Type (HV\_gate\_2)

* Inputs: vuel\_to\_C, yi7, vscl\_uel\_C
* Outputs: yo
* Parameters:
* Initialisation and Simulation equations

yo := max(vuel\_to\_C, max(yi7, vscl\_uel\_C));

# Block: LV\_gate\_2\_Type (LV\_gate\_2)

* Inputs: voel\_to\_C, yi8, vscl\_oel\_C
* Outputs: yo
* Parameters:
* Initialisation and Simulation equations

yo := min(voel\_to\_C, min(yi8, vscl\_oel\_C));

# Block: LowPassFilter1Ta (lowPassFilter1Ta\_1)

* Inputs: u, x0\_extXa
* Outputs: y
* Parameters: Ka, Ta, x0, InitCondition
* Internal variables: x
* Initialisation equations

if InitCondition == 1 then //Initialise from internal parameter, x0

x := x0;

elseif InitCondition == 2 then //Initialise from external input, x0\_extXa

x := x0\_extXa;

end if;

y := x;

* Simulation equations

x := previous(x)+(interval()/Ta)\*(Ka\*Ta\*previous(u)-previous(x));

y := x;

# Block: LimiterVrmaxmin (limiterVrmaxmin\_1)

* Inputs: u
* Outputs: y
* Parameters: Vr\_max, Vr\_min
* Initialisation and Simulation equations

y := if u > Vr\_max then Vr\_max else if u < Vr\_min then Vr\_min else u;

# Model initialisation

* Inputs: Efd\_init
* Outputs: x0\_lowPassFilter1Ta\_1, x0\_leadLagTcTb\_1, x0\_lowPassFilterKiaTia\_1
* Parameters: Kia
* Initialisation equations

x0\_lowPassFilter1Ta\_1 := Efd\_init;

x0\_leadLagTcTb\_1:=Efd\_init\*(1-Kia);

x0\_lowPassFilterKiaTia\_1:=Kia\*Efd\_init;