Over Excitation Limiter OEL1

Over Excitation Limiter for Synchronous Machine Excitation Systems OEL1

Model supported by PSLF
Over Excitation Limiter OEL1

1) **Tpickup Parameter:**
The OEL1 limiter Runback is set based on the limiter timer set by the Tpickup parameter:
   a) If Tpickup > 0 then the field current limiter timer will have an inverse characteristic behavior. The Tpickup will be the delay in operation and will be activated when the field current exceeds Ifdset by 1 per unit.
   b) If Tpickup < 0 then the field current limiter will have a definite time characteristic and will be the time the field current must remain above the Ifdset to be activated.

2) **Runback Parameter:**
The Runback parameter set the behavior of the current or voltage limiter:
   a) If Runback > 0 then the voltage regulator reference is biased by the negative direction ramping at the rate if 1/Runback per unit per second as long as the field current or field voltage (as selected by Vfdflag) exceeds Ifcont.
      If the selected value falls under Ifcont then ramping is stopped and the voltage regulator reference remains frozen at the biased value.
   b) If Runback < 0 then the voltage regulator reference is biased and set to Runback per unit.
   c) If Runback = 0 then the generator Exciter is set to the Hard Limit of Ifcont.

3) The Hard Limiter:
The Hard limiter is set when the generator field current >= Ifdmax
   The Hard Limiter acts immediately to limit the excitation system output voltage and is set the following way by the Ifdmax parameter
   a) When Ifdmax > 0 then the Hard Limiter is set to Ifdmax.
   b) When Ifdmax < 0 then the Hard Limiter is set to Ifcont

4) Generator Trip:
   a) If Generator Current > Ifdmax for Tmax seconds or,
   b) If Generator Current > Tfdset for Tset seconds.

A pseudo code is available in the next page.
Pseudo Code of OEL1 algorithm:

HardLimit = Ifcont if Ifdmax <= 0 else HardLimit = Ifdmax
If fVfdflag = 0 Then InputSignal := FieldCurrent
Else InputSignal := EField
Function Local_DetermineVOEL(SetRunbackSeconds)
    If Runback < 0 then begin
        VOEL = Runback
        FrozenLimitToRunBack := true
    End
    Else Begin
        If SetRunbackSeconds >= 0 then begin
            VOEL := -1/(fRunBack)* SetRunbackSeconds + oldVOEL
            FrozenLimitToRunBack := true
        End
    End
End
Function LOCAL_GetRunBackSeconds
    If (TPickup < 0) and (AboveIfdInputSignalSetTime >= Abs(TPickup)) Then Begin
        result := Present Time - AboveIfdInputSignalSetTime- - Abs(TPickup)
    End
    Else If (fTPickUp > 0) Then Begin // Inverse time
        InverseCharacteristic TripReset(IO,local_NewTime);
        If InverseCharacteristic Tripped Then Begin // Meaning it got to one second
            result := InverseCharacteristic.TripTime
        End
    End;
End

If IfieldCurrent > Ifdmax then
    OEL1 is at HardLimit.
    If IfieldCurrent > Ifdmax for Tmax time then Trip the Unit.
End
If (IfieldCurrent > fIfdSet) Then Begin
    If IfieldCurrent > fIfdSet for Tmax time then Trip the Unit.
End;
If InputSignal > fIfdSet Then Begin
    AlreadyReset = false;
    If AboveIfdInputSignalSetTime = 0 then AboveIfdInputSignalSetTime = PresentTime
SetRunbackSeconds := PresentTime
If LOCAL_GetRunBackSeconds >= 0 then begin
   If (Runback = 0) and (Alarm = 0) then begin
      OEL1 set at HardLimit = Ifcont
   End
End

If Runback <> 0 then begin
   Local_DetermineVOEL(LOCAL_GetRunBackSeconds)
   If (fRunback <> 0) and (FrozenLimitToRunBack) and (Alarm = 0) then begin
      VOEL SendingSignal := true;
   End
End
Else Begin
   If (fRunback <> 0) and (FrozenLimitToRunBack) and (Alarm = 0) then begin
      VOEL SendingSignal := true;
   End
   If (fRunback <> 0) and (InputSignal > fIfcont) and (not AlreadyReset) and (Alarm = 0) then begin
      If LOCAL_GetRunBackSeconds >= 0 then begin
         LOCAL_DetermineVOEL(LOCAL_GetRunBackSeconds)
      End
   End
Else begin
   If (Runback <> 0) and (FrozenLimitToRunBack) then begin
      oldVOEL := fVOEL
      If (not AlreadyReset) then begin
         VOEL Frozen
      End
   End
AboveIfdInputSignalSetTime := Set to Zero;
InverseCharacteristic(Reset to Zero) // Reset to zero
If (not fAlreadyReset) then begin
   Reset
End;
AlreadyReset := true
RunbackisZeroActivated := false
End
End