

Load Characteristic LDEV1

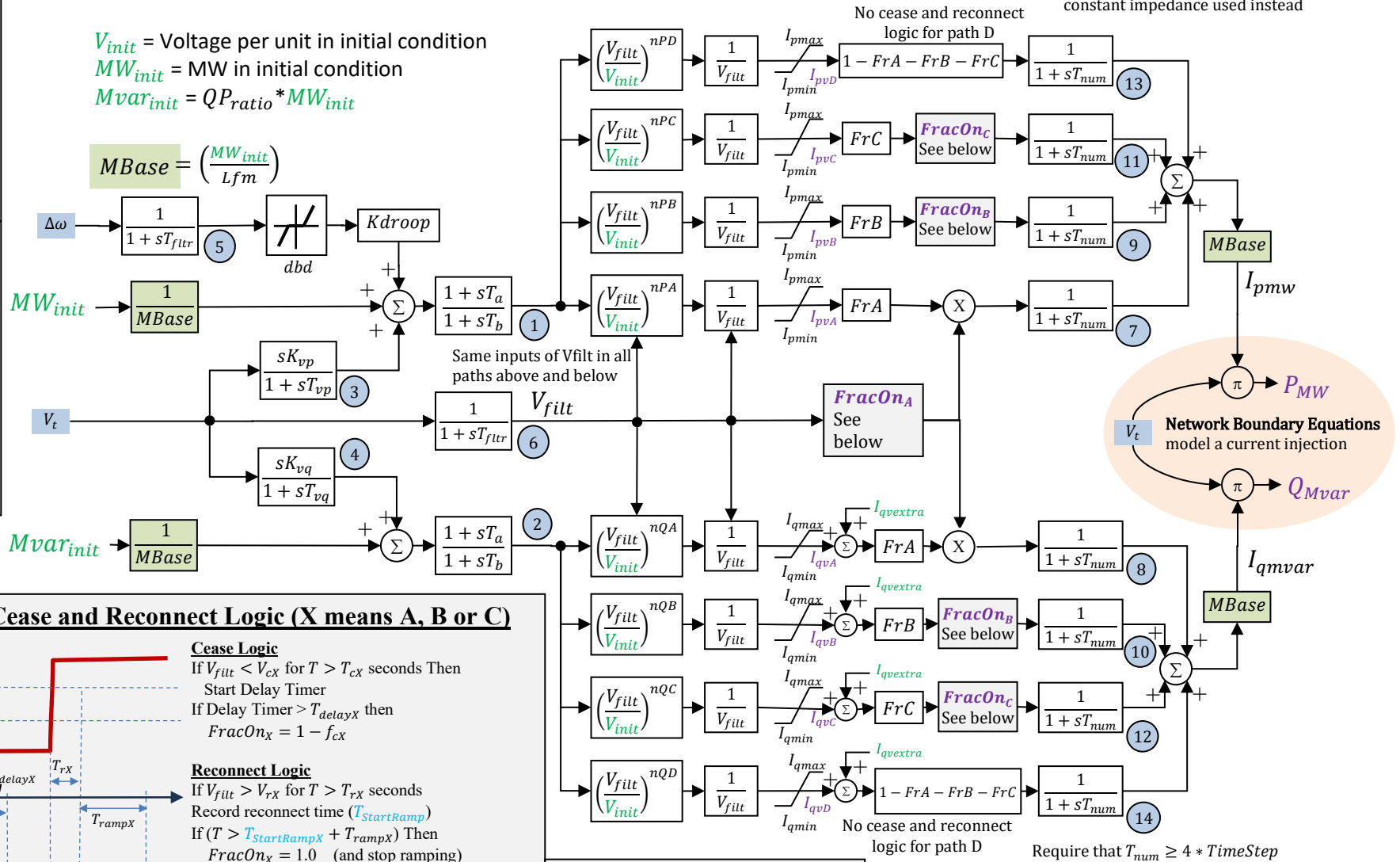
States
 1 - Pctrl
 2 - Qctrl
 3 - dP
 4 - dQ
 5 - wfilt
 6 - Vfilt
 7 - IpA
 8 - IqA
 9 - IpB
 10 - IqB
 11 - IpC
 12 - IqC
 13 - IpD
 14 - IqD

Others
 1 - Pmw
 2 - Qmvar
 3 - IpvA
 4 - IqvA
 5 - FracOnA
 6 - IpvB
 7 - IqvB
 8 - FracOnB
 9 - IpvC
 10 - IqvC
 11 - FracOnC
 12 - IpvD
 13 - IqvD

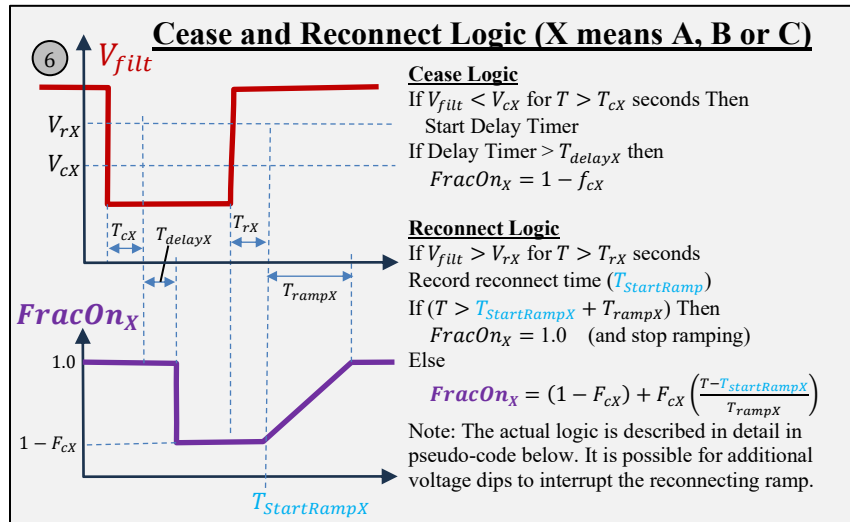
Load Characteristic LDEV1 Electronic Load Model (EV Chargers, Computer Loads, ...)

V_{init} = Voltage per unit in initial condition
 MW_{init} = MW in initial condition
 $Mvar_{init} = QPratio * MW_{init}$

$$MBase = \left(\frac{MW_{init}}{Lfm} \right)$$



If ($V_{init} < V_{cx}$ for any path)
 OR ($V_{init} < MinVoltLoad$)
 OR ($MW_{init} = 0$)
 then this model will be ignored, and a constant impedance used instead



$FracOn_X$ represents a fraction of the respective portion of the load that is operating based on the cease and reconnect logic. Note that fraction D does not have cease and reconnect logic.

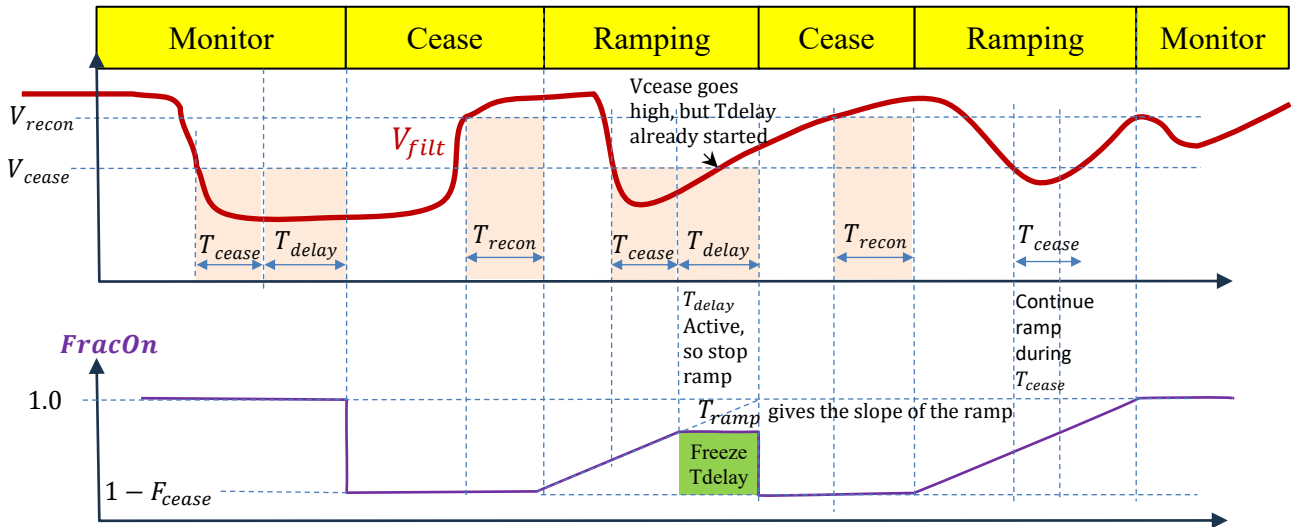
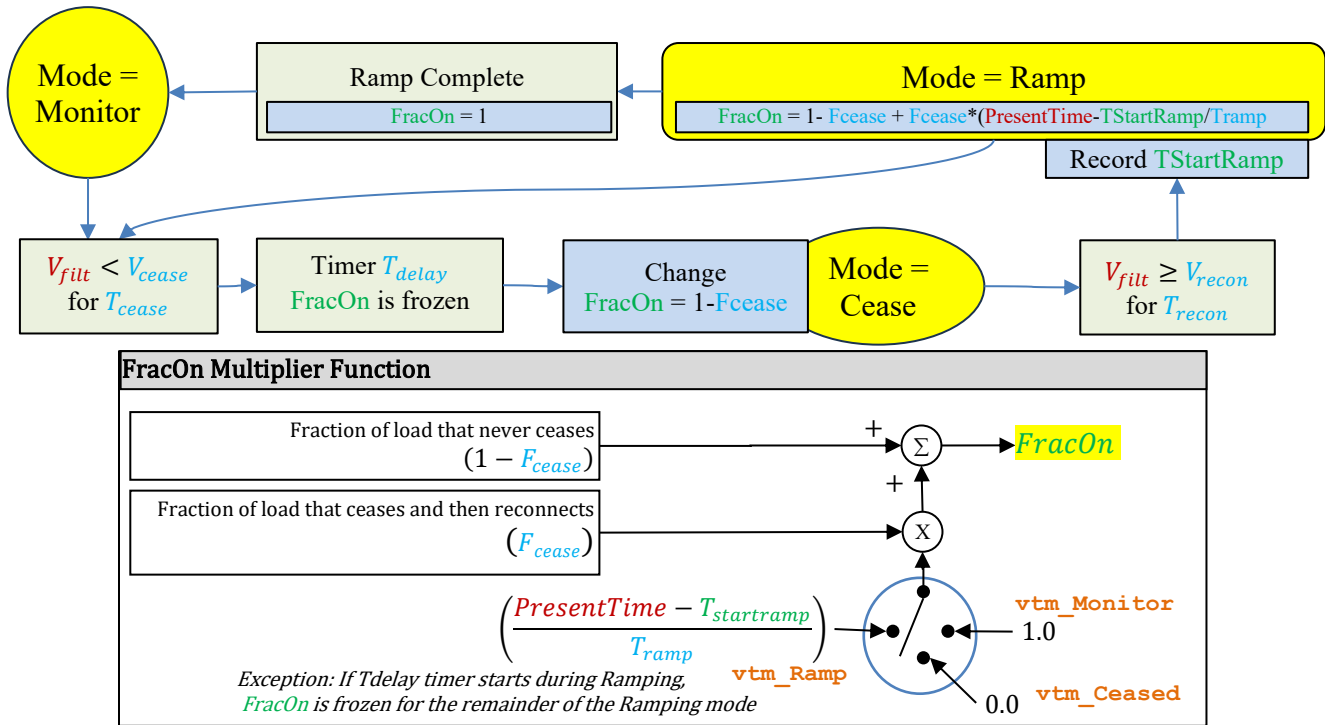
If used inside a modular load model, then $I_{qvextra} = 0$ and mismatch of Mvar is handled during the initialization of the CompLoad components.

If used as a stand-alone model, then $I_{qvextra} = \frac{Q_{mvar} - Mvar_{init}}{MBase} * V_{init}$

Index	Parameter	Description
0	Lfm	Loading factor used to calculate MVAbase of model as MWinit/Lfm. Lfm<0.001 is treated as a 0.800
1	Tfltr	Voltage measurement time constant [s]
2	Dbd	Deadband on frequency response [pu]
3	Kdroop	Frequency droop [per unit]
4	Kvp	Proportional constant for active power washout
5	Tvp	Time constant for active power washout [seconds]
6	QPratio	Q/P ratio for MvarInit computation from MWInit. MvarInit = QPratio*MWInit
7	Kvq	Proportional constant for reactive power washout
8	Tvq	Time constant for reactive power washout [seconds]
9	Ta	Lead time constant [seconds]
10	Tb	Lag time constant [seconds]
11	FrA	Fraction of Type A (If FrA + FrB + FrC > 1.0 they are normalized to sum to 1.0)
12	FrB	Fraction of Type B (If FrA + FrB + FrC > 1.0 they are normalized to sum to 1.0)
13	FrC	Fraction of Type C (If FrA + FrB + FrC > 1.0 they are normalized to sum to 1.0)
14	nPA	Active Power Exponential for Type A
15	nQA	Reactive Power Exponential for Type A
16	nPB	Active Power Exponential for Type B
17	nQB	Reactive Power Exponential for Type B
18	nPC	Active Power Exponential for Type C
19	nQC	Reactive Power Exponential for Type C
20	nPD	Active Power Exponential for Type D
21	nQD	Reactive Power Exponential for Type D
22	FcA	Fraction that will cease for Type A ($0 \leq FcA \leq 1$)
23	VcA	Voltage threshold for cease logic for Type A [per unit]
24	TcA	Time delay for cease logic to be initiated for Type A [seconds]
25	TdelayA	Time delay to cease after detection for Type A [seconds]
26	VrA	Voltage threshold to initiate power ramp logic for Type A [per unit]
27	TrA	Time delay for ramp up reconnection logic to be initiated for Type A [seconds]
28	TrampA	Ramp up time for Type A [seconds]
29	FcB	Fraction that will cease for Type B ($0 \leq FcB \leq 1$)
30	VcB	Voltage threshold for cease logic for Type B [per unit]
31	TcB	Time delay for cease logic to be initiated for Type B [seconds]
32	TdelayB	Time delay to cease after detection for Type B [seconds]
33	VrB	Voltage threshold to initiate power ramp logic for Type B [per unit]
34	TrB	Time delay for ramp up reconnection logic to be initiated for Type B [seconds]
35	TrampB	Ramp up time for Type B [seconds]
36	FcC	Fraction that will cease for Type C ($0 \leq FcC \leq 1$)
37	VcC	Voltage threshold for cease logic for Type C [per unit]
38	TcC	Time delay for cease logic to be initiated for Type C [seconds]
39	TdelayC	Time delay to cease after detection for Type C [seconds]
40	VrC	Voltage threshold to initiate power ramp logic for Type C [per unit]
41	TrC	Time delay for ramp up reconnection logic to be initiated for Type C [seconds]
42	TrampC	Ramp up time for Type C [seconds]
43	Ipmax	Maximum Ip [per unit]
44	Ipmin	Minimum Ip [per unit]
45	Iqmax	Maximum Iq [per unit]
46	Iqmin	Minimum Iq [per unit]
47	ndelt	Time step subdivision factor (not used by PowerWorld)
48	Tnum	Time delay for outputs [seconds] (If < 4*TimeStep are treated as 4*TimeStep)

Pseudo code detail for Cease and Reconnect Logic

Cease and Reconnect logic uses a timer to control transitions between 3 modes: Monitor, Ceased and Ramp. While in the Ramp Mode, the $FracOn$ variable is updated each timestep to perform a ramp from the Ceased Value of $(1 - F_{cease})$ back up to 1.00. The output of the cease and reconnect logic is a multiplier **FracOn** the specifies what fraction of the respective portion of load remains online.



Variable Names in the Pseudo-Code Below

Fcease, **Vcease**, **Tcease**, **Tdelay** : input parameters control Ceasing
Vrecon, **Trecon**, **Tramp** : input parameters control Reconnecting
Vfilt : State variable of the filtered voltage
FracOn : Multiplier representing the fraction of the load that remains connected
Mode : 3 possible modes: **vtm_Monitor**, **vtm_Ceased**, **vtm_Ramp**
TimerActive : 4 possibilities determine if any timer is active (only one timer can be active)
 vtt_None, **vtt_Tcease**, **vtt_Tdelay**, **vtt_Trecon**
Timer : Float number used to keep track of timing for Tcease, Tdelay and Trecon
TStartRamp : Float number used to keep track of timing during ramp

Initialize the following before simulation starts

```
FracOn = 1.0
Mode = vtm_Monitor
TimerActive = vtt_None
Timer = 0
StartRamp = 0
```

Run the following at each time step

```
If Fcease > 0 Then // Ignore all this stuff if Fcease <= 0
  If (Mode == vtm_Ramp)
    AND (TimerActive <> vtt_Tdelay) // FracOn is frozen during Tdelay
  Then // update the value of FracOn
    If PresentTime >= TStartRamp + Tramp Then
      FracOn = 1
      Mode = vtm_Monitor
    Else
      FracOn = 1 - Fcease + Fcease*(PresentTime - TStartRamp)/Tramp
    EndIf
  EndIf

  // Tcease and Tdelay timers are counting the same whether we are in Monitor or Ramp
  If Mode <> vtm_Ceased Then // handled in both Monitor and Ramp modes
    If TimerActive <> vtt_Tdelay Then
      If (Vfilt >= Vcease) Then
        TimerActive = vtt_None
      ElseIf (TimerActive <> vtt_Tcease) Then
        Timer = PresentTime
        TimerActive = vtt_Tcease
      EndIf
    EndIf

    If (TimerActive = vtt_Tcease) AND (PresentTime - Timer) >= Tcease Then
      Timer = PresentTime
      TimerActive = vtt_Tdelay
    EndIf

    If (TimerActive = vtt_Tdelay) AND (PresentTime - Timer) >= Tdelay Then
      TimerActive = vtt_None
      Mode = vtm_Cease
      FracOn = 1 - Fcease
    EndIf
  Else // (Mode == vtm_Cease)
    If (Vfilt < Vrecon) Then
      TimerActive = vtt_None
    ElseIf (TimerActive <> vtt_Trecon)
      Timer = PresentTime
      TimerActive = vtt_Trecon
    EndIf

    If (TimerActive = vtt_Trecon) AND (PresentTime - Timer) >= Trecon Then
      TimerActive = vtt_None
      If Tramp == 0 then // skip over Ramp Mode and jump to Monitor
        Mode = vtm_Monitor
        FracOn = 1
      Else
        Mode = vtm_Ramp
        TStartRamp = PresentTime
      EndIf
    EndIf
  EndIf
EndIf // If Fcease > 0
```