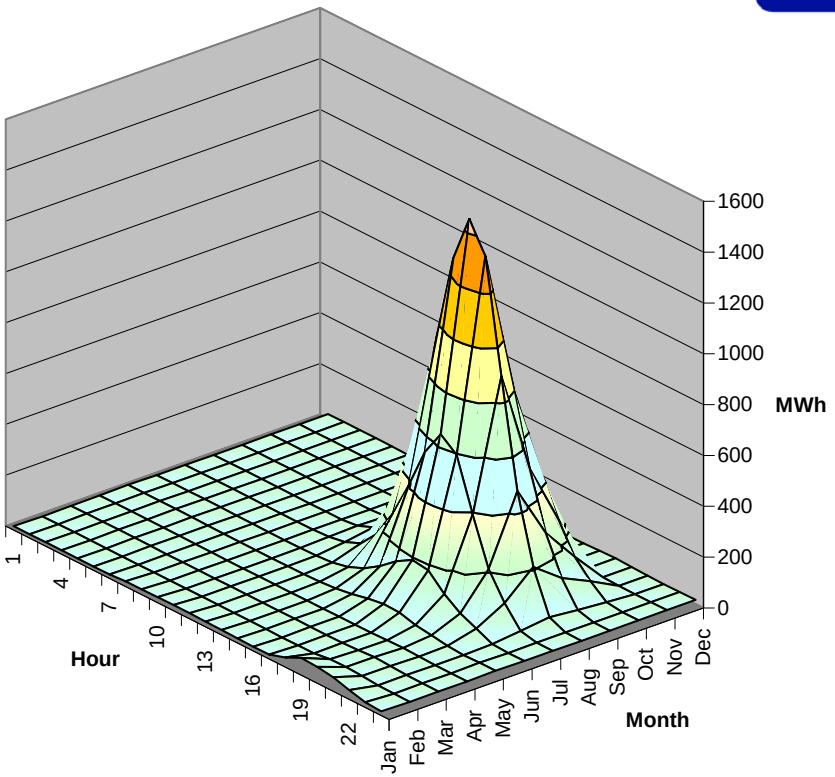


ELECTRIC POWER RESEARCH INSTITUTE

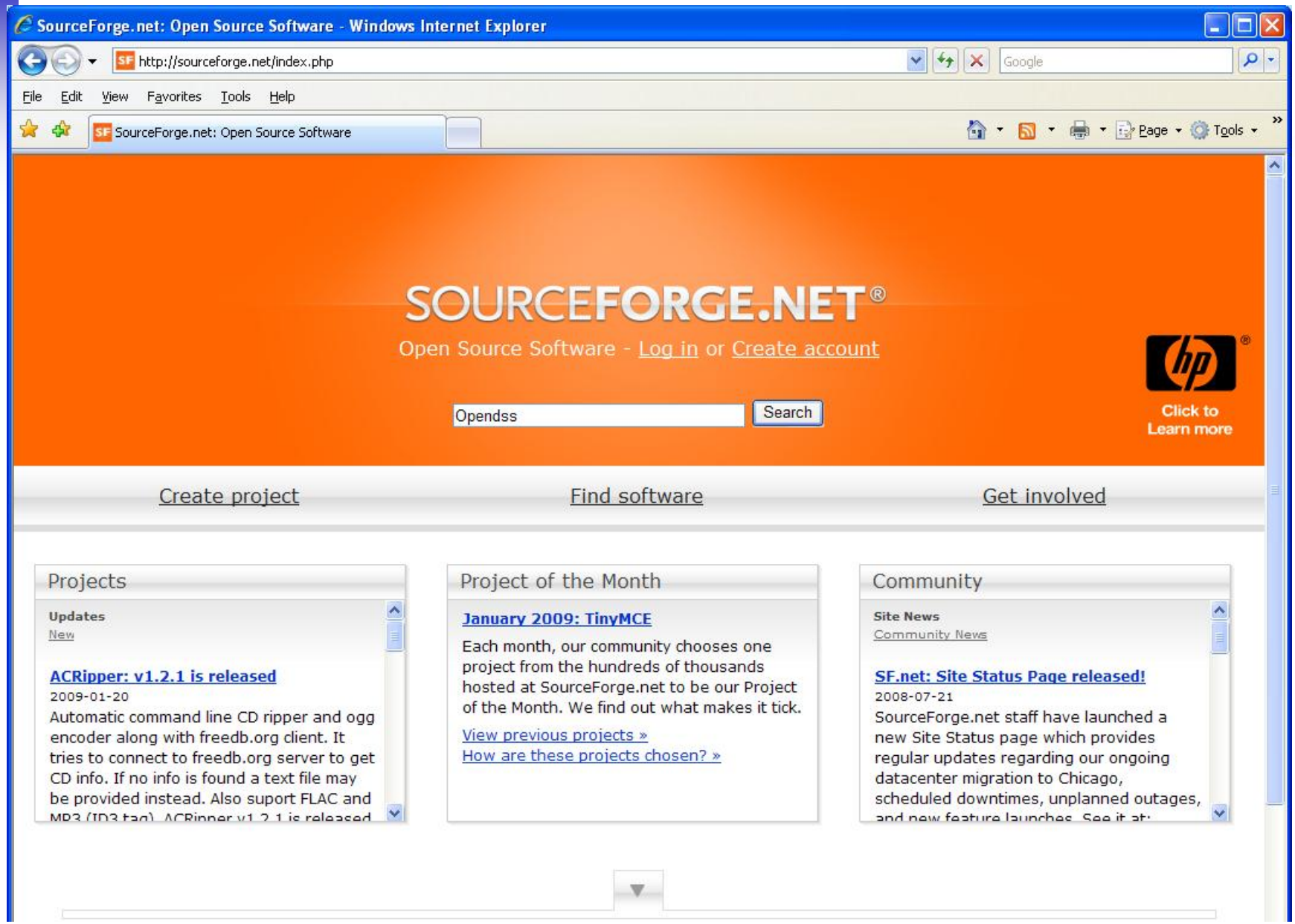
OpenDSS
Level 2 Training

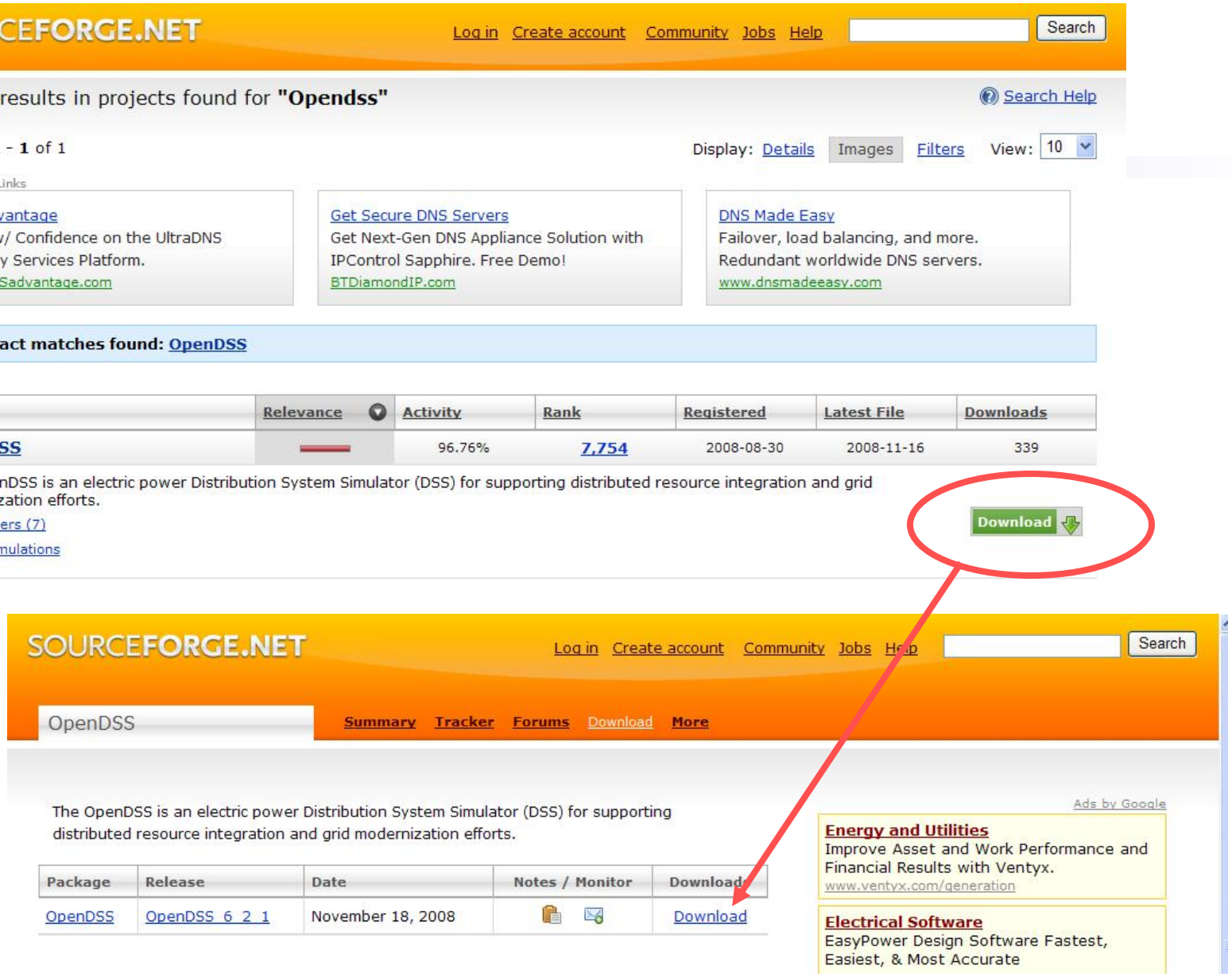
27 April 2009

Roger Dugan
rdugan@epri.com

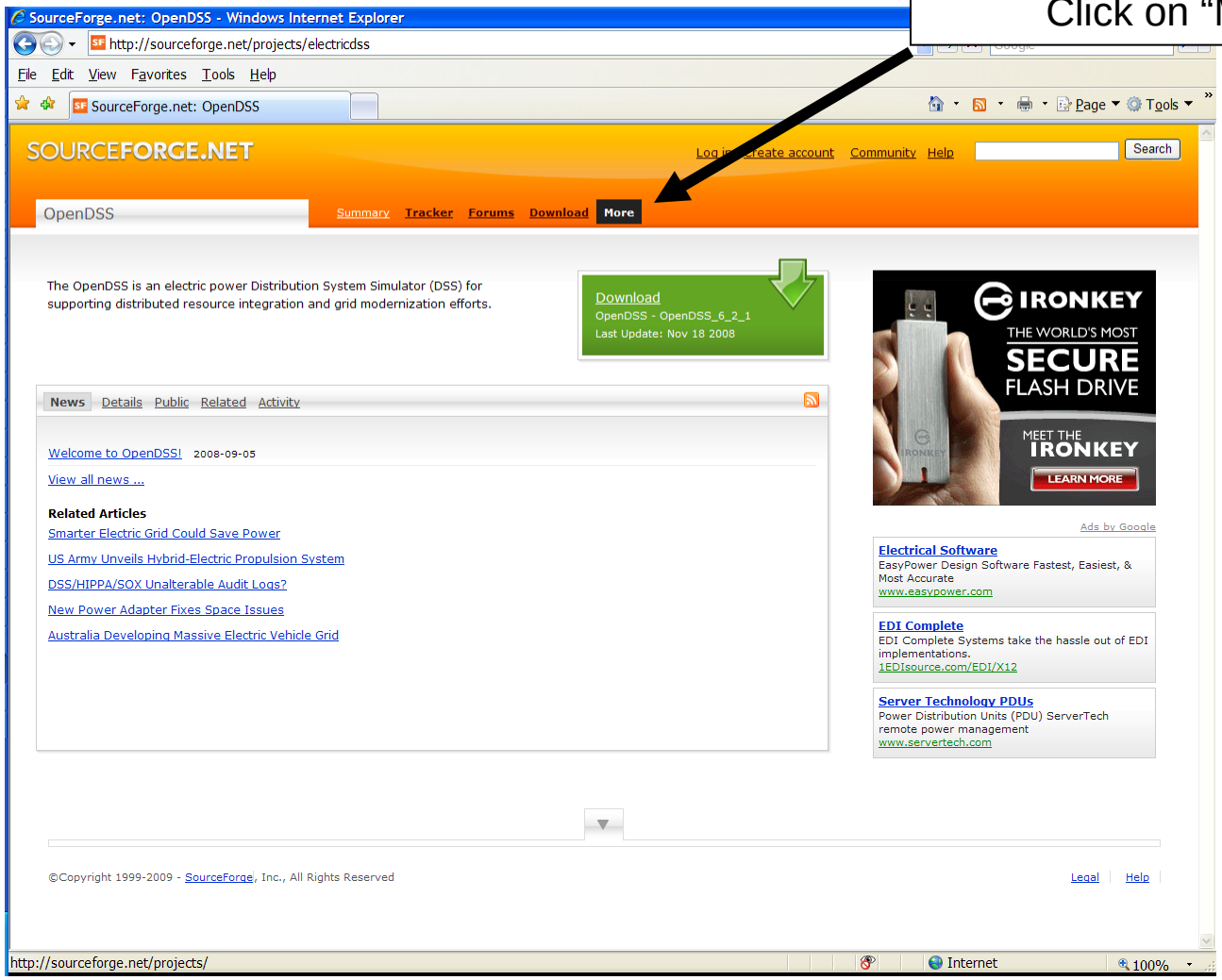
EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Getting Started: Installation & Basic Usage



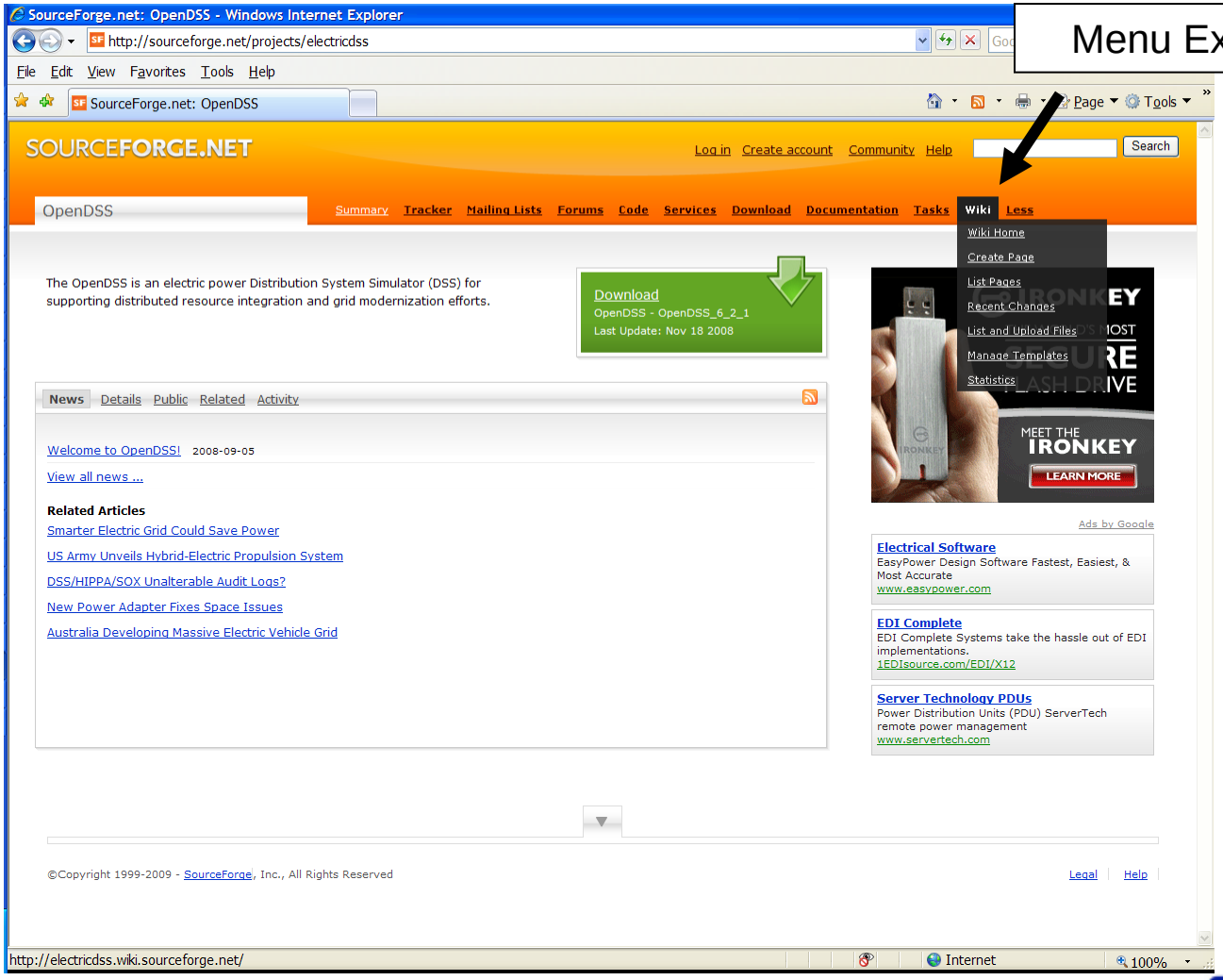


Finding the Wiki …



EPRI ELECTRIC POWER RESEARCH INSTITUTE

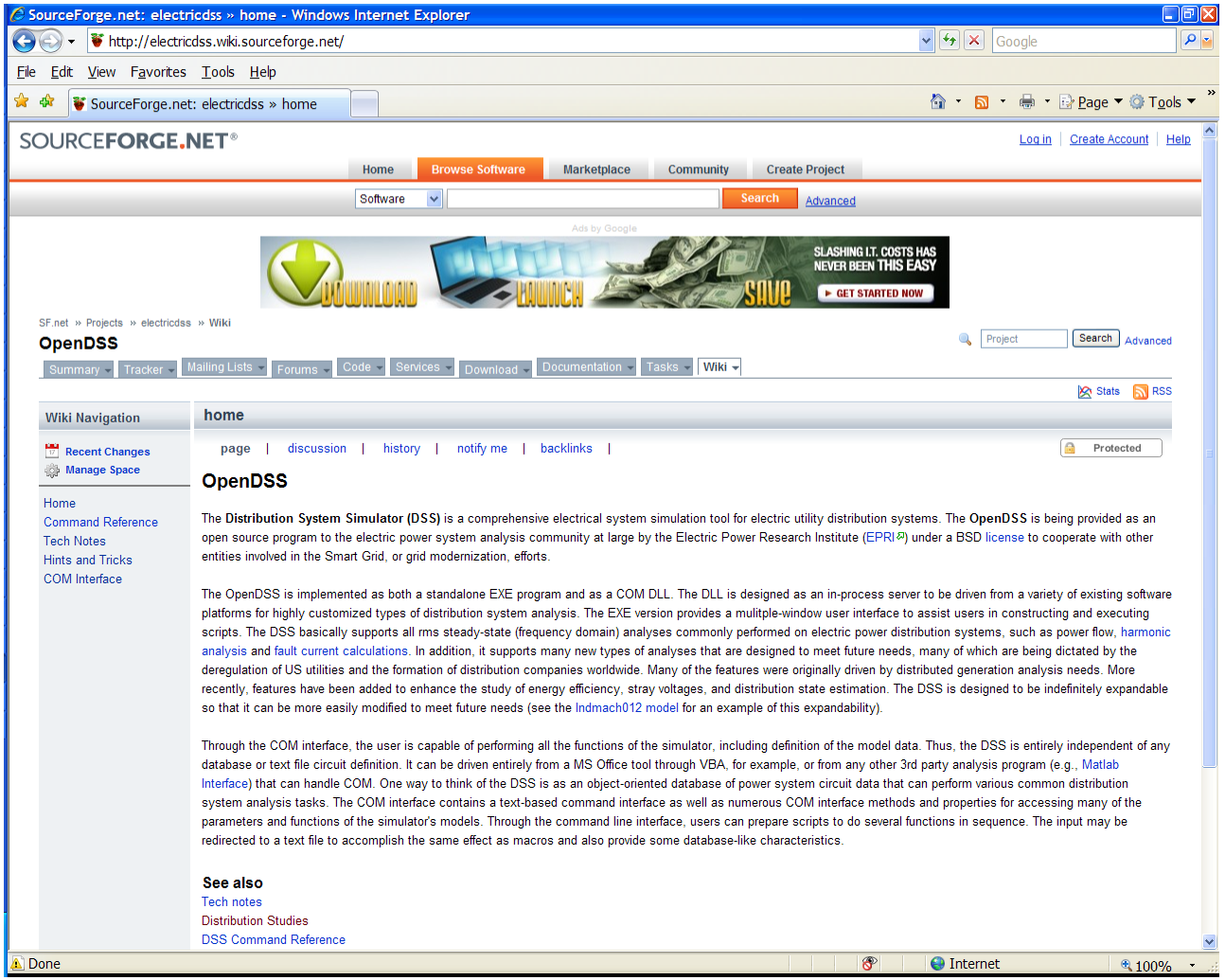
Finding the Wiki, cont'd



Menu Expands; Select Wiki

EPRI ELECTRIC POWER RESEARCH INSTITUTE

Wiki Home Page (Latest documentation)



Release Versions Vs. Source Code

- Release versions are posted irregularly
- You can keep up with the latest changes by accessing the source code and building the latest version
  - Some of the docs on the Wiki apply only to latest changes
- Compilers
  - Delphi 2007 (full IDE)
    - This is what we use for development
  - Turbo Delphi (Free)
    - https://downloads.embarcadero.com/free/turbodelphi

Accessing the SourceForge.Net Source Code Repository with TortoiseSVN

- Install a 32-bit TortoiseSVN client from tortoisesvn.net/downloads.
- Recommendation: From the TortoiseSVN General Settings dialog and click the last check box, to use "_svn" instead of ".svn" for local working directory name.

  Then, to grab the files from SourceForge:

  1 - create a clean directory such as "c:\opendss"

  2 - right-click on it and choose "SVN Checkout..." from the menu

  3 - the repository URL is "https://electricdss.svn.sourceforge.net/svnroot/electricdss".
  - change the checkout directory if it points somewhere other than what you want.

Program Files

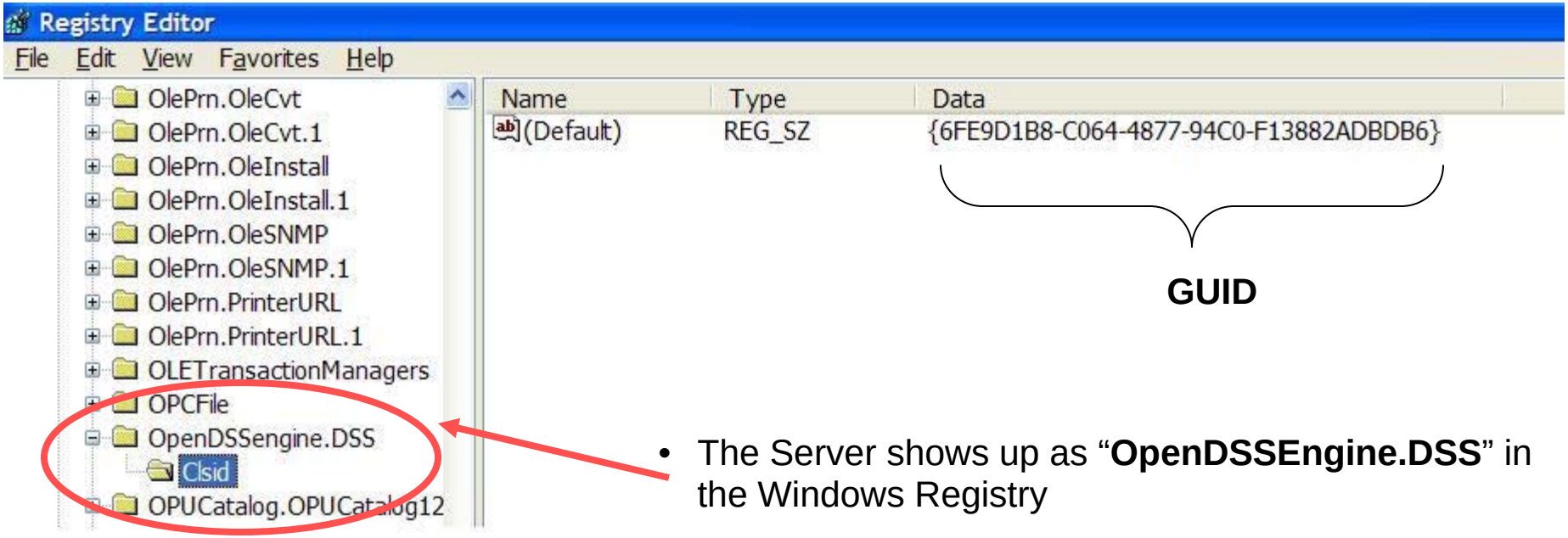
- OpenDSS.EXE — Standalone EXE
- OpenDSSEngine.DLL — In-process COM server
- KLUSolve.DLL — Sparse matrix solver
- DSSgraph.DLL — DSS graphics output

- Copy these files to the directory (folder) of your choice
  - Typically **`c:\OpenDSS`** *or* **`c:\Program Files\OpenDSS`**

- **If you intend to drive OpenDSS from another program, you will need to register the COM server**

Registering the COM Server

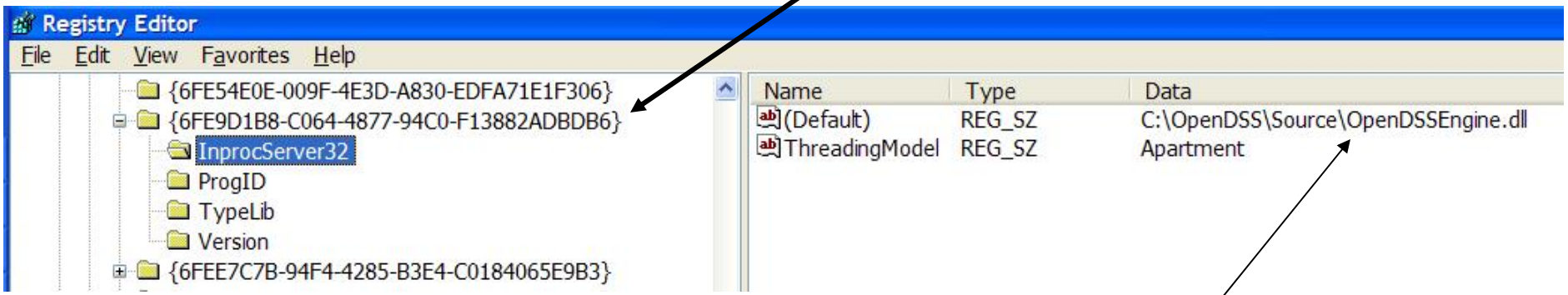
- In DOS window, change to the folder where you installed it and type:
  - **`Regsvr32 OpenDSSEngine.DLL`**



- The Server shows up as "**OpenDSSEngine.DSS**" in the Windows Registry

Registering the COM Server, cont'd

If you look up the GUID

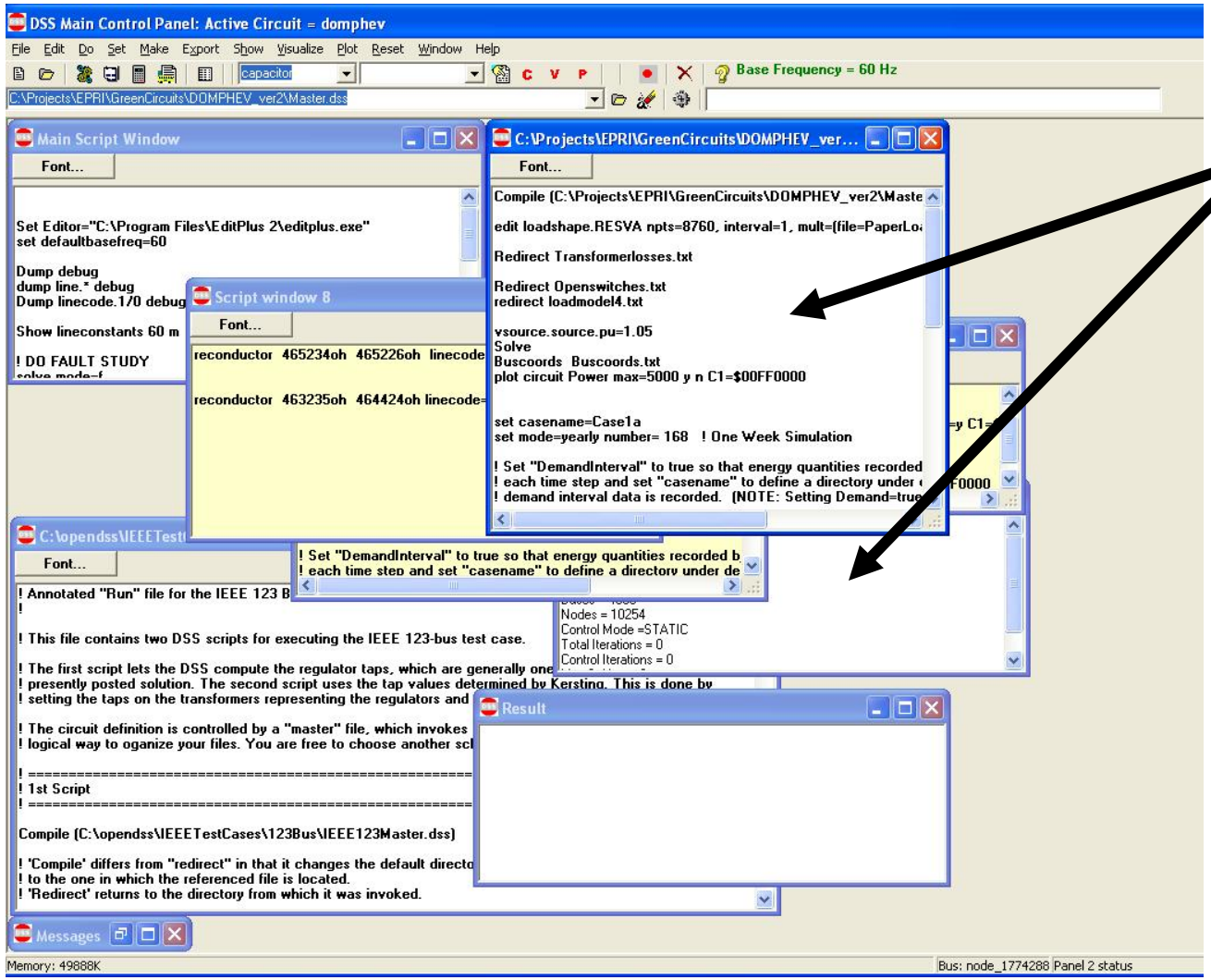


Points to OpenDSSEngine.DLL (In-process server, Apartment Threading model)

Accessing the COM Server

- In MATLAB:
  - `DSSobj = actxserver('OpenDSSEngine.DSS');`
- In VBA:
  - ```
    Public DSSobj As OpenDSSEngine.DSS
    Set DSSobj = New OpenDSSEngine.DSS
    ```
- In PYTHON:
  - `self.engine = win32com.client.Dispatch("OpenDSSEngine.DSS")`
    ```

OpenDSS Standalone EXE User Interface

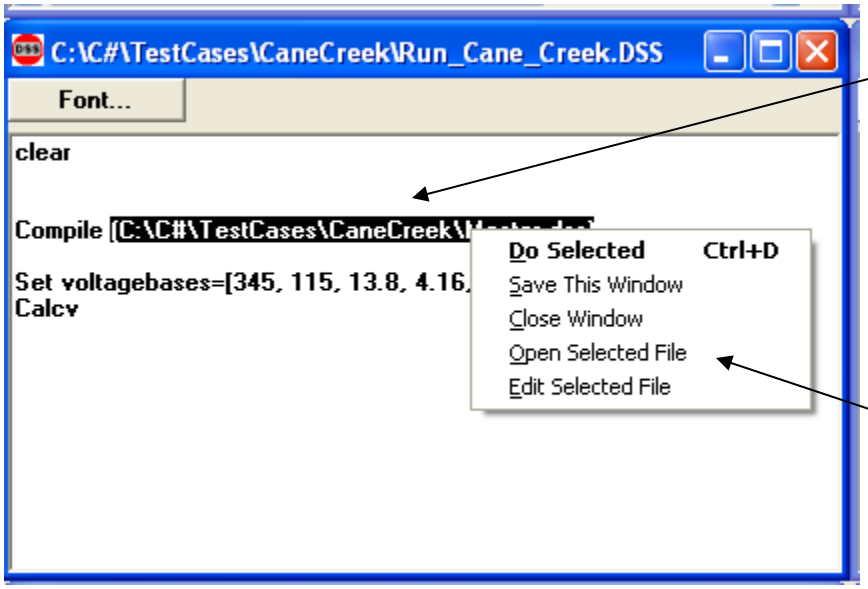


Multiple script windows

Any script window may be used at any time.

EPRI ELECTRIC POWER RESEARCH INSTITUTE

Executing Scripts in the Stand-alone EXE



Select all or part of a line

Right-Click to get this pop-up menu

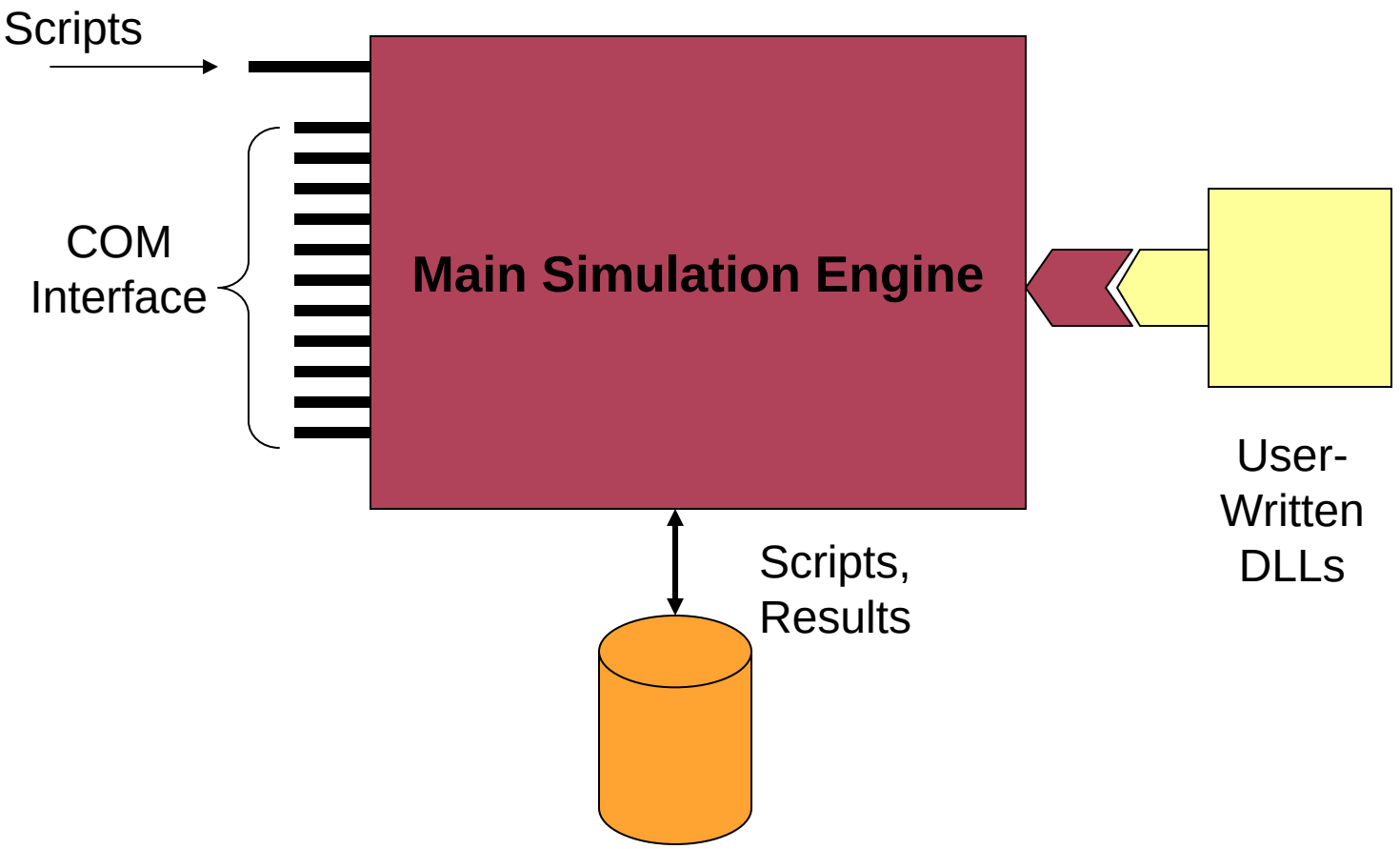
DSS executes selected line or opens selected file name

Any script window may be used at any time.

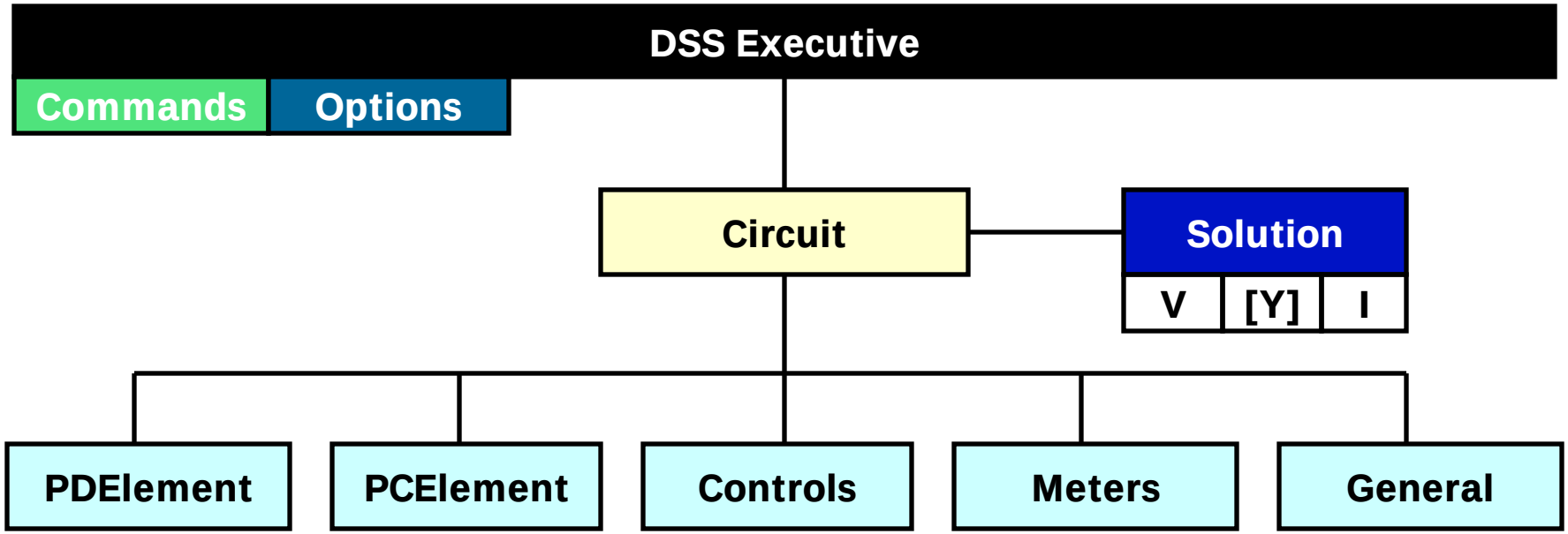
EPRI | ELECTRIC POWER RESEARCH INSTITUTE

DSS Structure

DSS Structure



DSS Object Structure



Line
Transformer
Capacitor
Reactor

Load
Generator
Vsource
Isource

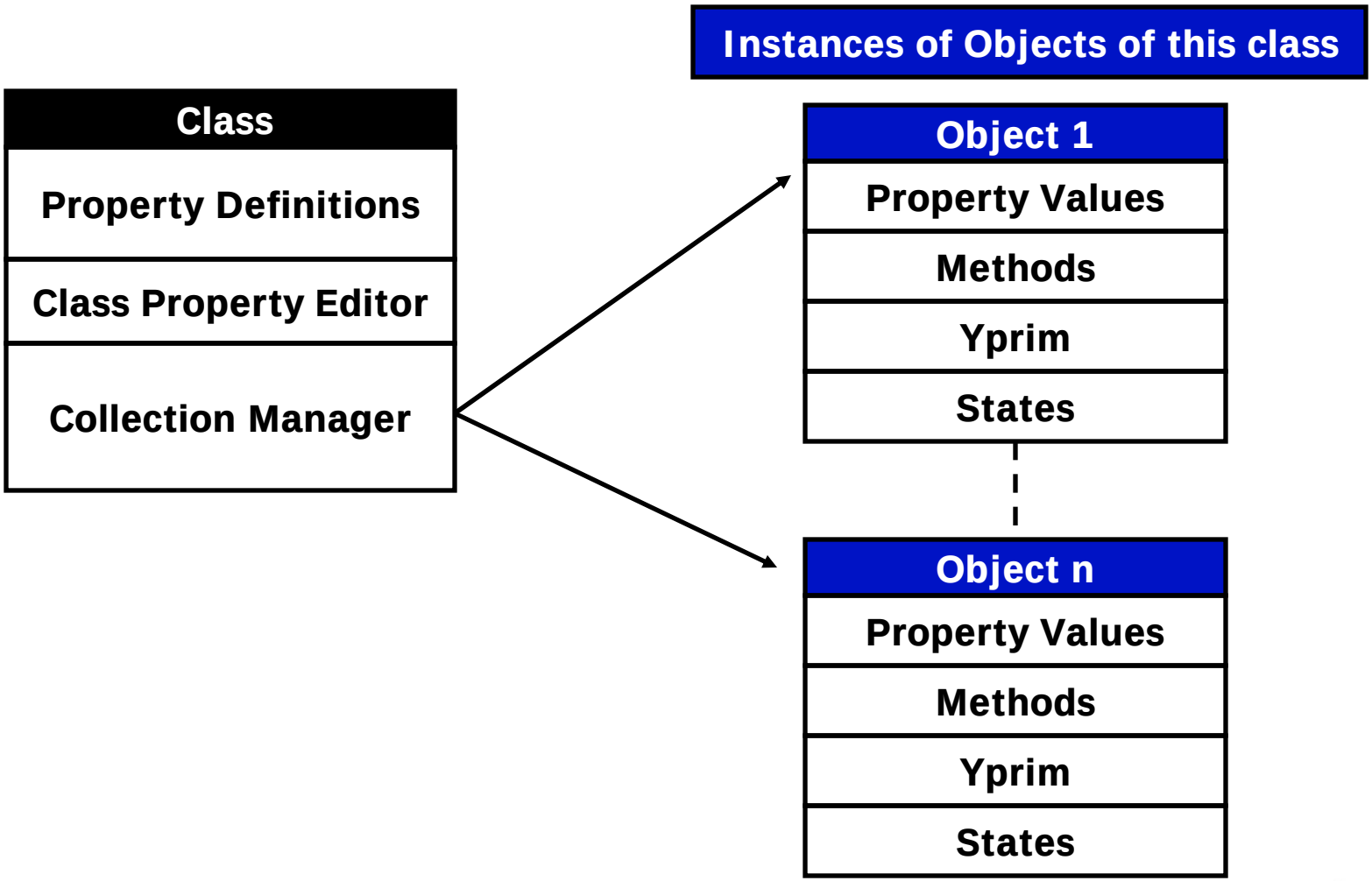
RegControl
CapControl
Relay
Reclose
Fuse

Monitor
EnergyMeter
Sensor

LineCode
LineGeometry
WireData
LoadShape
GrowthShape
Spectrum
TCCcurve

EPRI ELECTRIC POWER RESEARCH INSTITUTE

DSS Class Structure



DSS Classes (as of 2009)

- Power Delivery (PD) Elements
  - Line
  - Transformer
  - Reactor
  - Capacitor
- Power Conversion (PC) Elements
  - Load
  - Generator
  - Vsource
  - Isource
- Control Elements
  - RegControl
  - CapControl
  - Recloser
  - Relay
  - Fuse
- Metering Elements
  - Monitor
  - EnergyMeter
  - Sensor
- General
  - LineCode
  - LineGeometry
  - Loadshape
  - Growthshape
  - Wiredata
  - Spectrum
  - TCC Curves

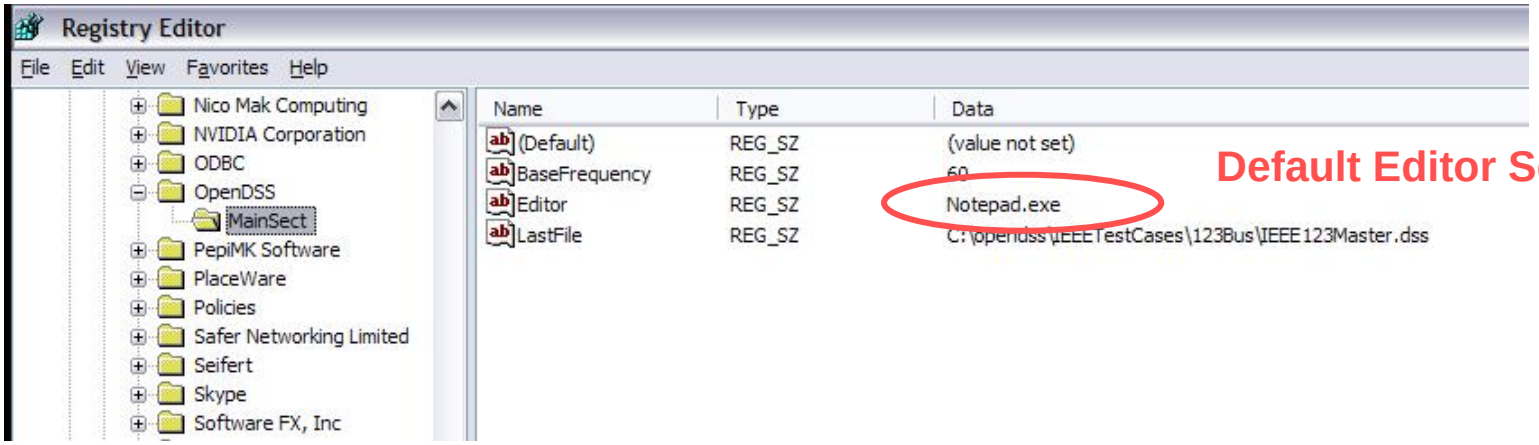
Organizing Your User Interface

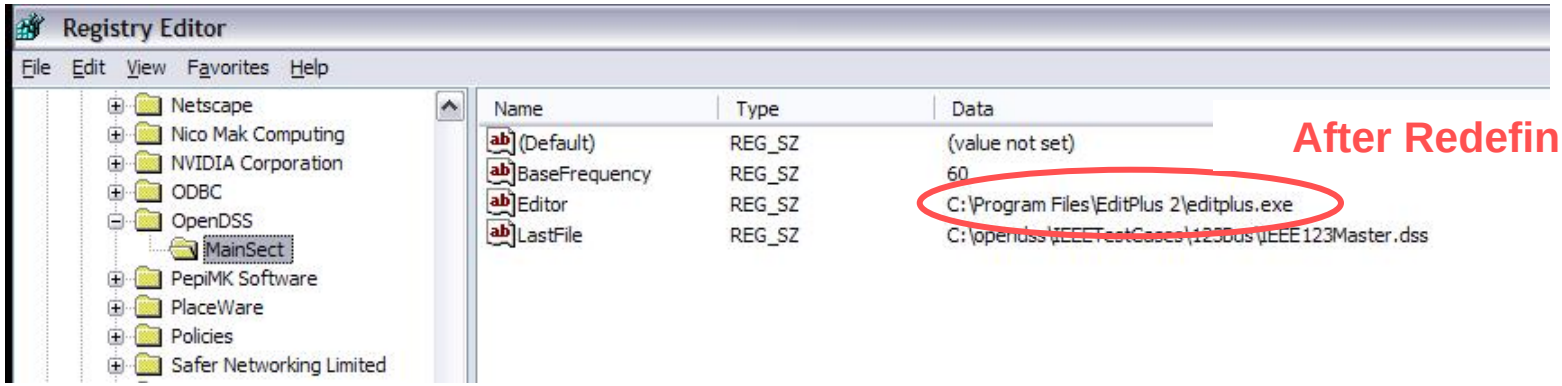
Organizing Your Main Screen

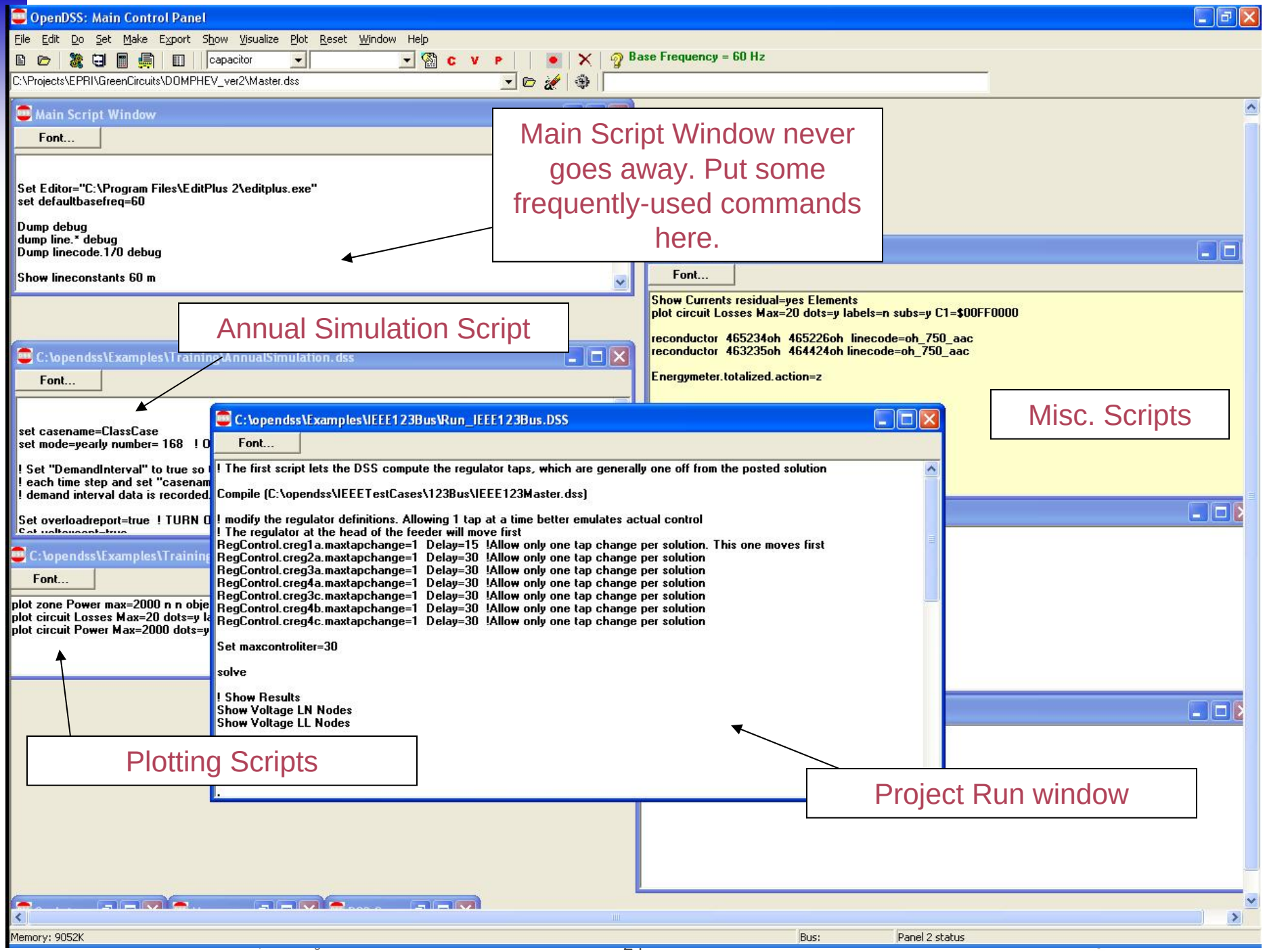
- The OpenDSS saves all windows on the main screen
- The appear where you left them when you shut down
- The next time you start up, you can resume your work
- Values are saved in a file (*OpenDSS.ini*) saved in the OpenDSS.exe folder
  - Note: You can update the program simply by copying in new exe and dll files.
  - Do not overwrite the ".ini" file if you want to preserve your workspace
    - However, if the .ini file gets corrupted, you may simply delete it.
- It is a good idea to come up with a comfortable way to organize your script windows …

OpenDSS Registry Entries

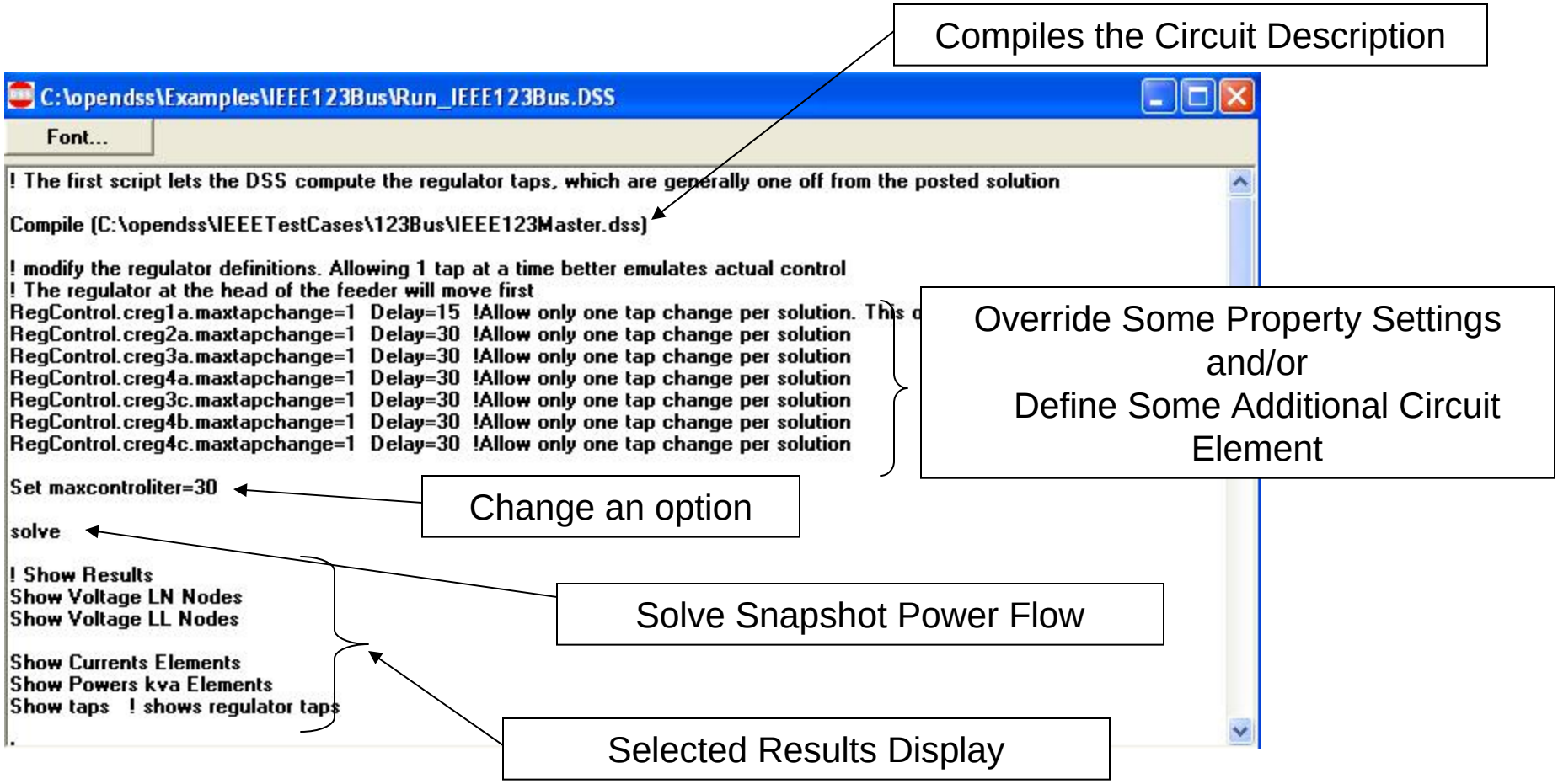
- Certain persistent values are saved to the Windows Registry upon exiting the program






OpenDSS: Main Control Panel
File Edit Do Set Make Export Show Visualize Plot Reset Window Help
capacitor
Base Frequency = 60 Hz
C:\Projects\EPRI\GreenCircuits\DOMPHEV_ver2\Master.dss
Main Script Window
Font...
Set Editor="C:\Program Files\EditPlus 2\editplus.exe"
set defaultbasefreq=60
Dump debug
dump line.* debug
Dump linecode.1/0 debug
Show lineconstants 60 m
Main Script Window never goes away. Put some frequently-used commands here.
Annual Simulation Script
C:\opendss\Examples\Training\AnnualSimulation.dss
set casename=ClassCase
set mode=yearly number= 168
! Set "DemandInterval" to true so
! each time step and set "casenam
! demand interval data is recorded.
Set overloadreport=true ! TURN O
C:\opendss\Examples\Training
plot zone Power max=2000 n n obje
plot circuit Losses Max=20 dots=y l
plot circuit Power Max=2000 dots=y
Plotting Scripts
C:\opendss\Examples\IEEE123Bus\Run_IEEE123Bus.DSS
! The first script lets the DSS compute the regulator taps, which are generally one off from the posted solution
Compile (C:\opendss\IEEETestCases\123Bus\IEEE123Master.dss)
! modify the regulator definitions. Allowing 1 tap at a time better emulates actual control
! The regulator at the head of the feeder will move first
RegControl.creg1a.maxtapchange=1 Delay=15 !Allow only one tap change per solution. This one moves first
RegControl.creg2a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg3a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4a.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg3c.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4b.maxtapchange=1 Delay=30 !Allow only one tap change per solution
RegControl.creg4c.maxtapchange=1 Delay=30 !Allow only one tap change per solution
Set maxcontroliter=30
solve
! Show Results
Show Voltage LN Nodes
Show Voltage LL Nodes
Project Run window
Show Currents residual=yes Elements
plot circuit Losses Max=20 dots=y labels=n subs=y C1=$00FF0000
reconductor 465234oh 465226oh linecode=oh_750_aac
reconductor 463235oh 464424oh linecode=oh_750_aac
Energymeter.totalized.action=z
Misc. Scripts
Memory: 9052K
Bus:
Panel 2 status

Organizing Run Scripts



Organizing Master File

So Compile Doesn't Fail

```
Clear

New Circuit.ExampleCircuit BasekV=138 pu=1.05 MVASC3 = 2000  MVASC1=2000

! Master file examples

! Library files
Redirect LineCode.dss
Redirect LoadShape.dss
Redirect GrowthShape.dss
Redirect TCC_Curve.dss
Redirect Spectrum.dss

! Circuit element descriptions are in a subdirectory "Feeders"
Redirect Feeders\Transformers.dss
Redirect Feeders\Branches.dss
Redirect Feeders\Loads.dss
Redirect Feeders\Capacitors.dss

Set Voltagebases=(69, 12.1, 4.16, 0.48)    ! define legal voltage bases
calcv       ! Abbrev for CalcVoltageBases

! Buses exit now so define coordinates
Buscoords buscoords.txt    ! Load bus x,y coordinates

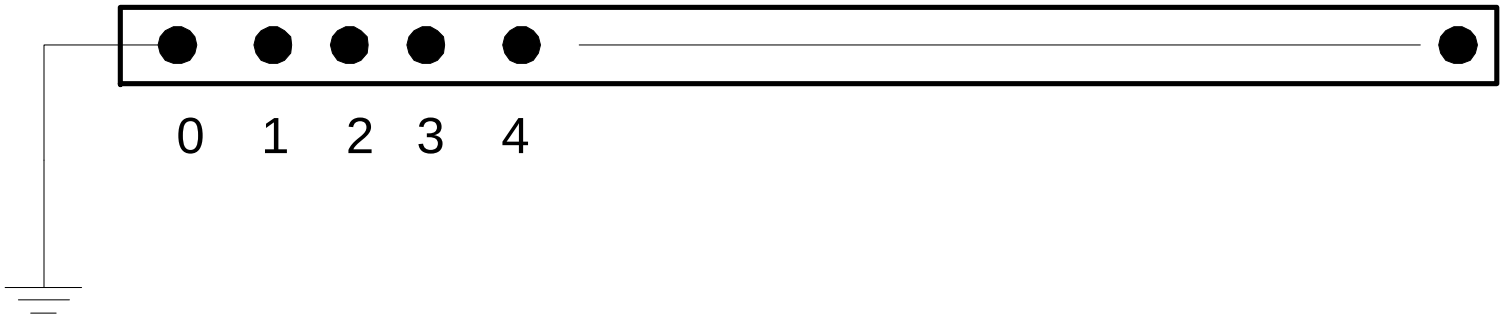
! Define energy meters after voltage bases so they will know voltage bases
Redirect EnergyMeter.dss

! Don't do Solve here ... better to do it in Run File
```

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Circuit Modeling Basics

DSS Bus Model



Referring to Buses and Nodes

Bus1=*BusName.1.2.3.0*

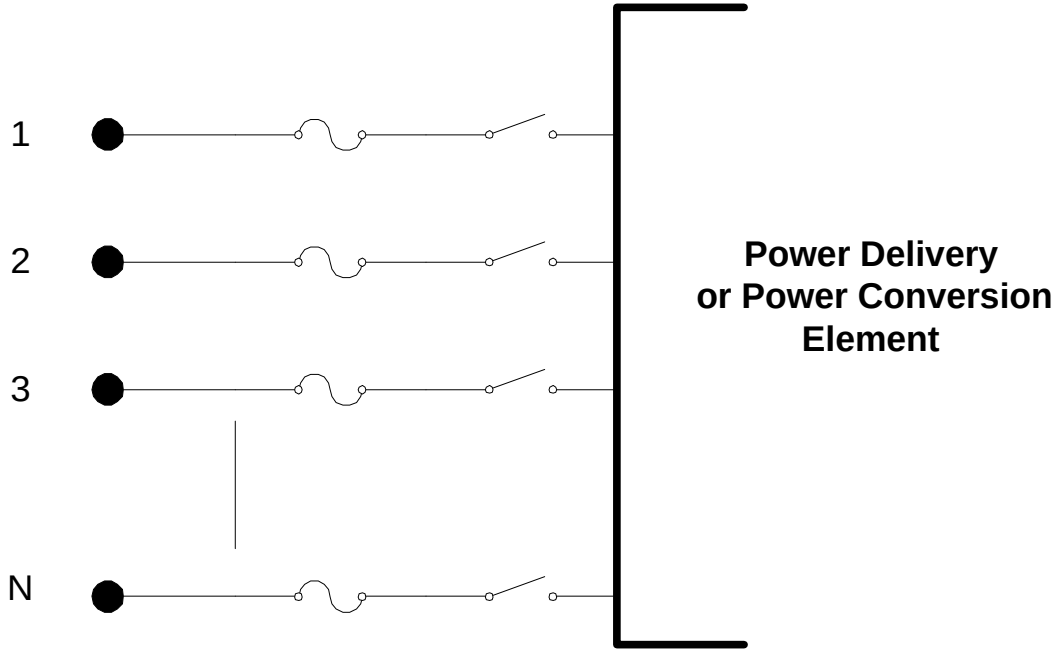
(This is the default for a 3-phase circuit element)

Shorthand notation for taking the default

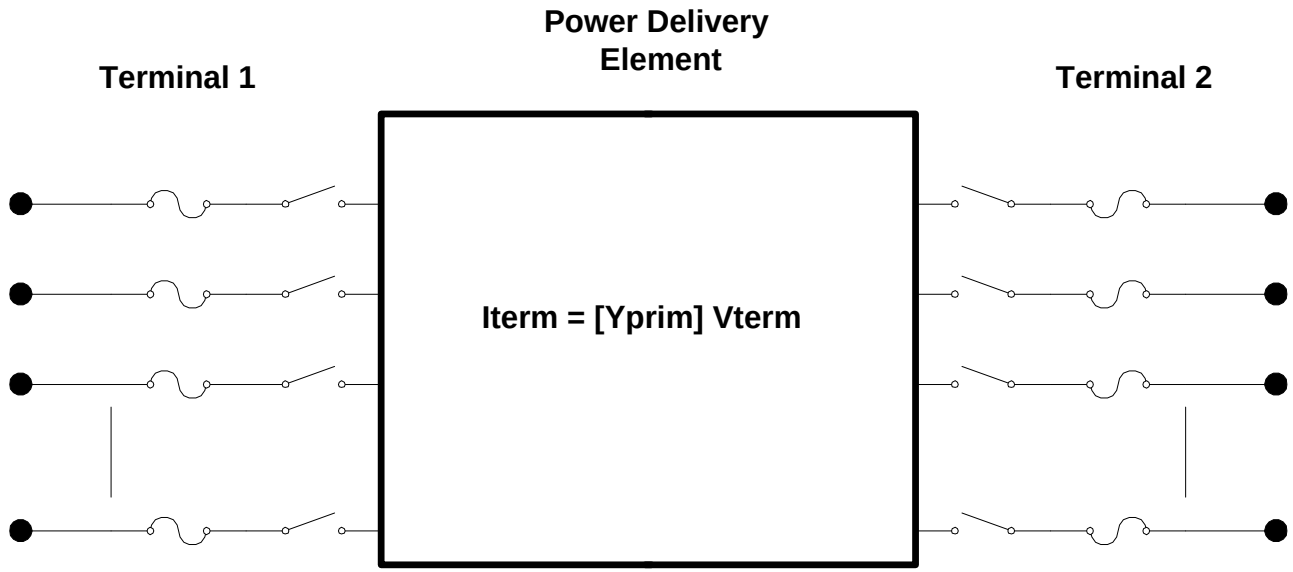
Bus1=*BusName*

Note: Sometimes this can bite you (e.g. – Transformers, or capacitors with ungrounded neutrals)

DSS Terminal Definition

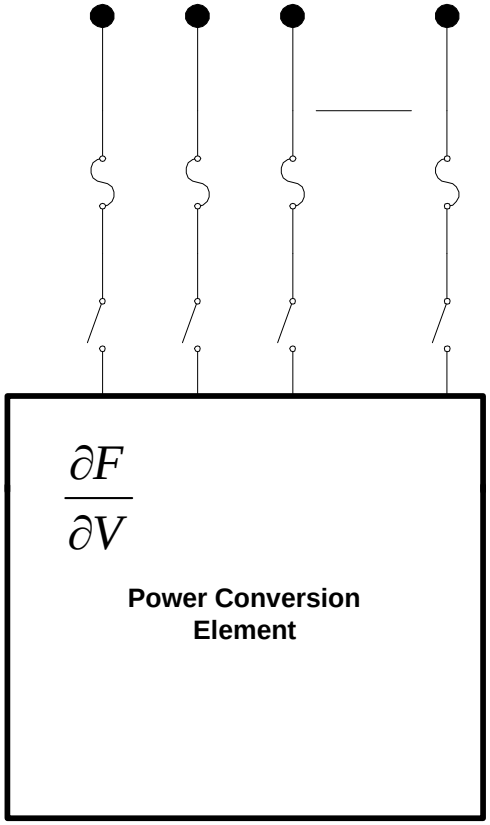


Power Delivery Elements

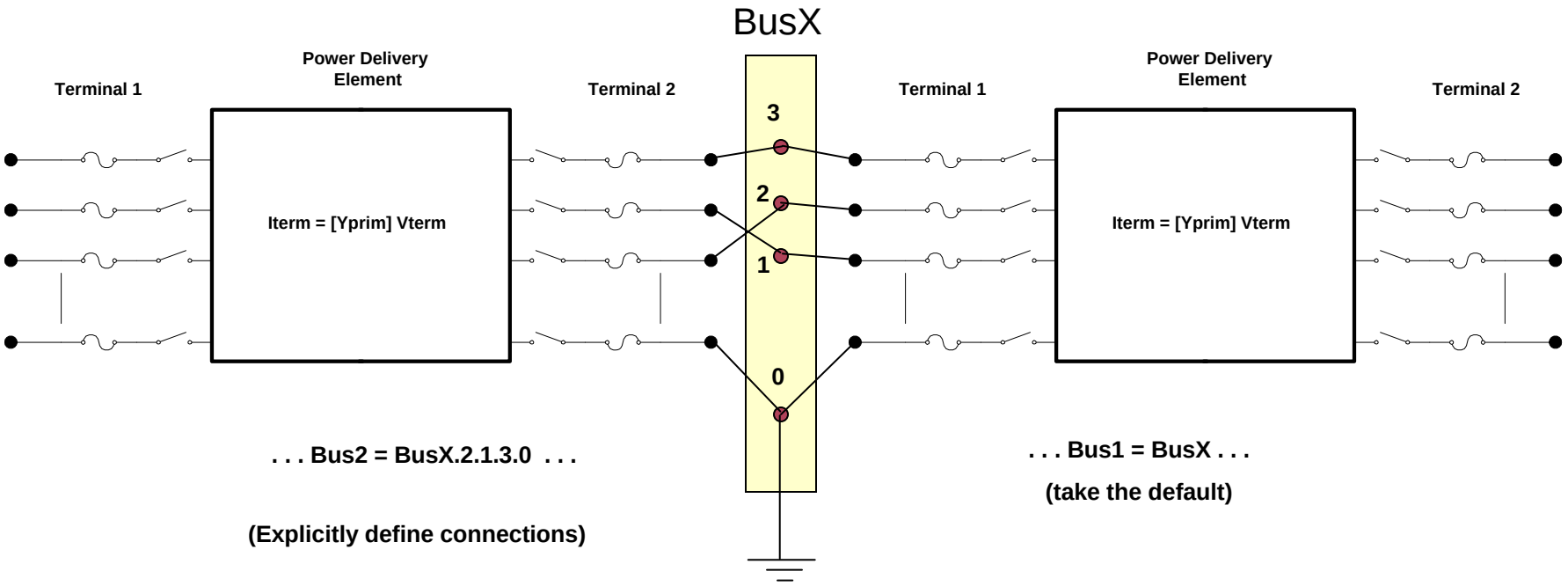


Power Conversion Elements

$I_{Term}(t) = \mathbf{F(V}_{Term}, [State], t)$



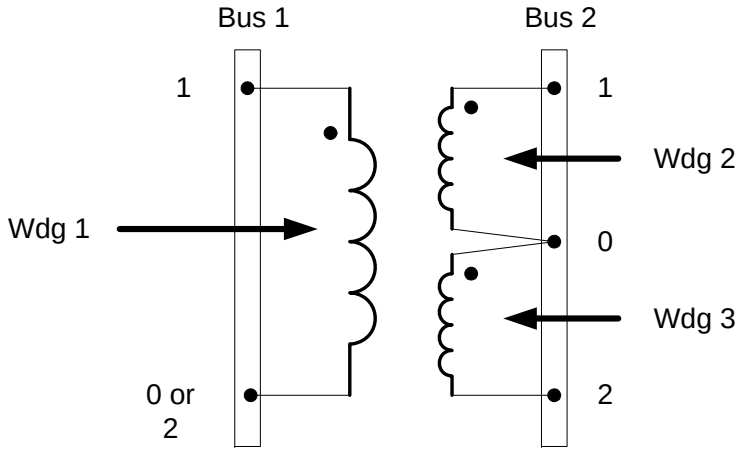
Circuit Elements are Connected together at the Nodes of Buses



DSS Convention: A *Terminal* can be connected to only one *Bus*. You can have any number of *Nodes* at a bus.

Connections for 1-Phase Residential Transformer

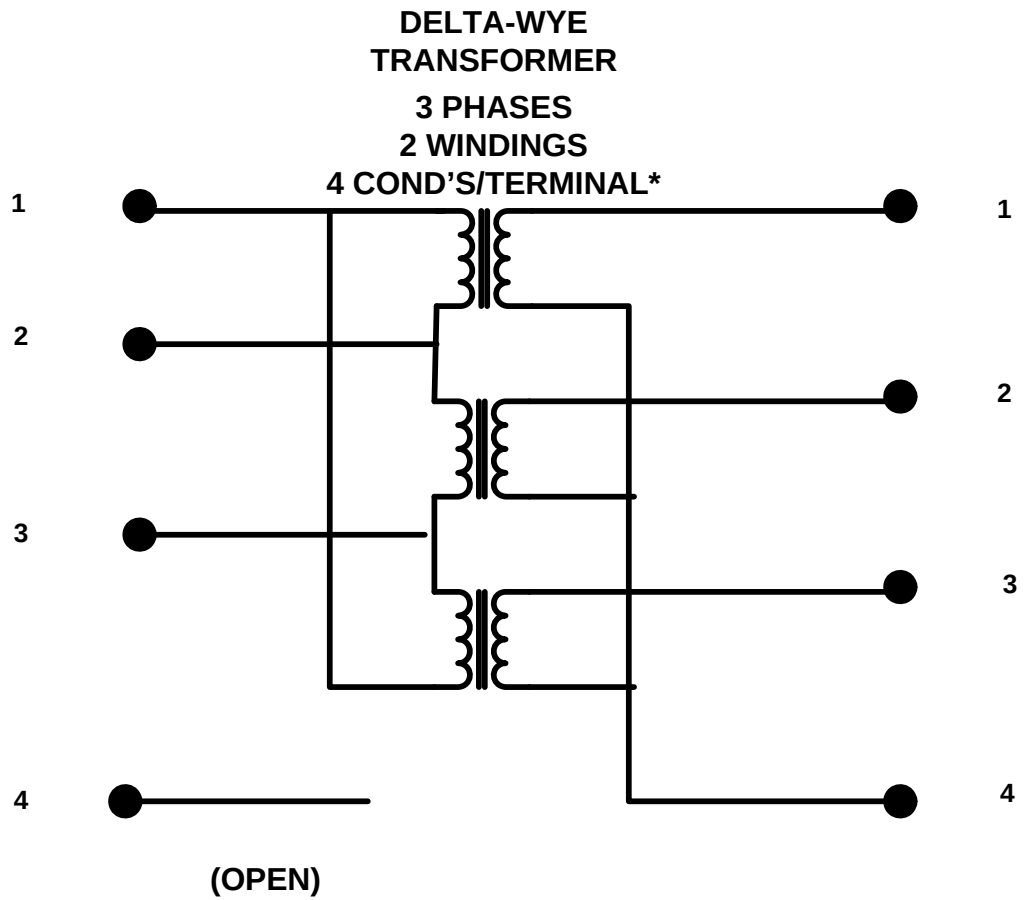
```
! Line-to-Neutral Connected 1-phase Center-tapped transformer
New  Transformer.Example1-ph  phases=1  Windings=3
~ Xhl=2.04  Xht=2.04  Xlt=1.36  %noloadloss=.2
~ Buses=[bus1.1  bus2.1.0   bus2.0.2]    !!! Note polarity
~ kVs=[7.2 .12  .12]     ! ratings of windings
~ kVAs=[25 25 25]
~ %Rs = [0.6  1.2  1.2]
~ conns=[wye wye wye]    ! default
```



Center-Tapped 1-Phase Transformer Model

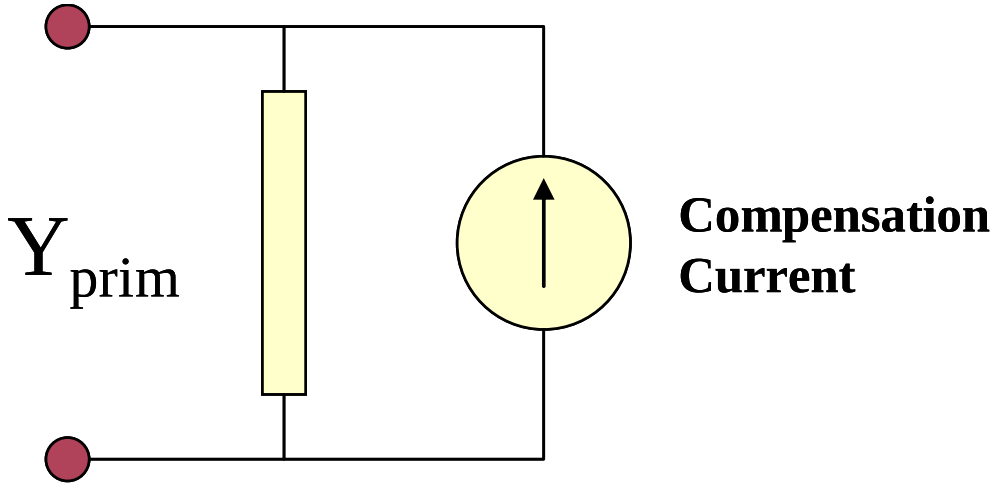
All Terminals of a Circuit Element Have Same Number of Conductors

3-Phase Transformer



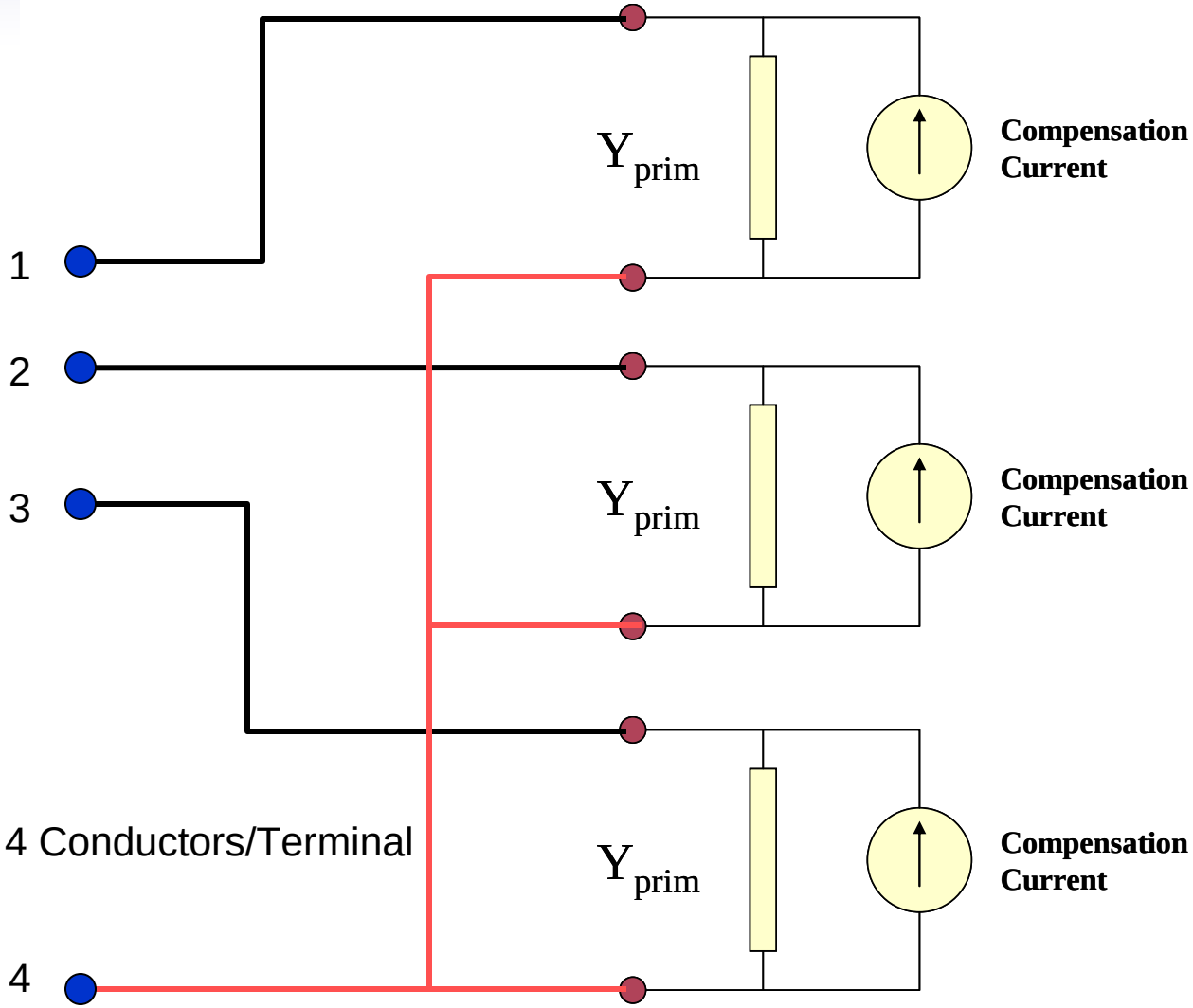
*** MUST HAVE THE SAME NUMBER OF CONDUCTORS FOR EACH TERMINAL**

Load (a PC Element)

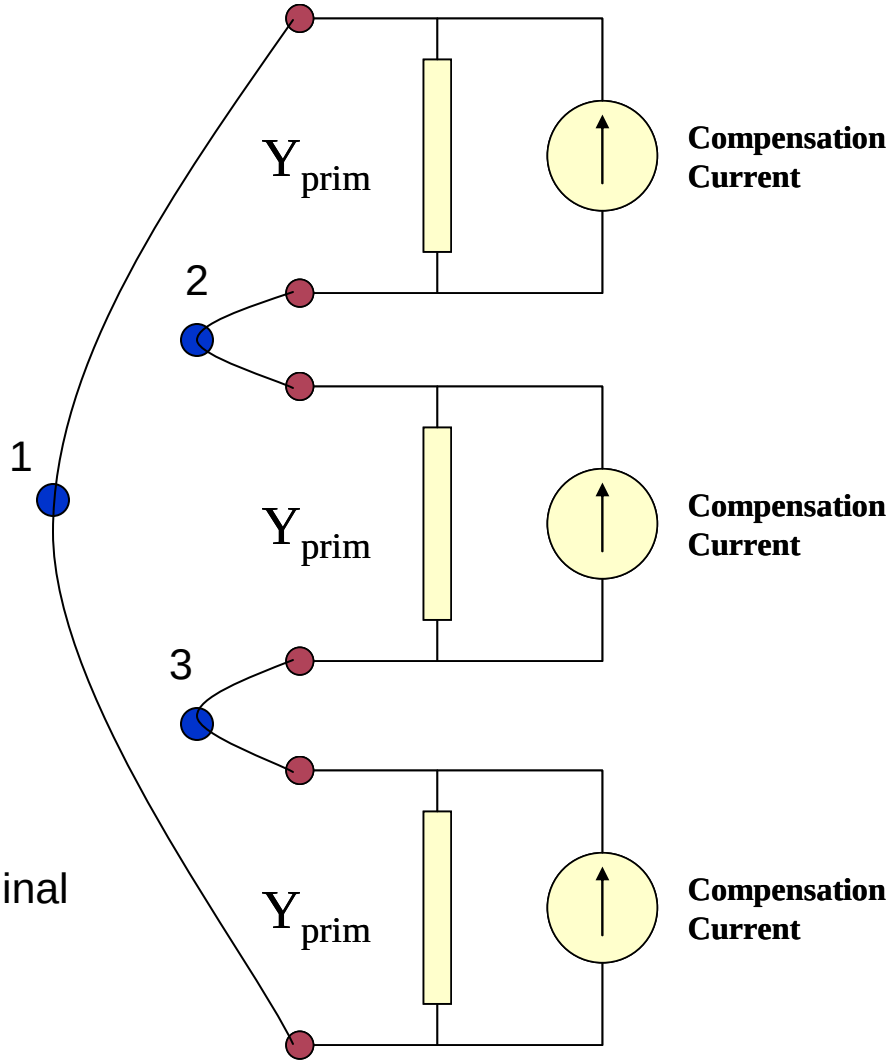


(One-Line Diagram)

Load - 3-phase Y connected

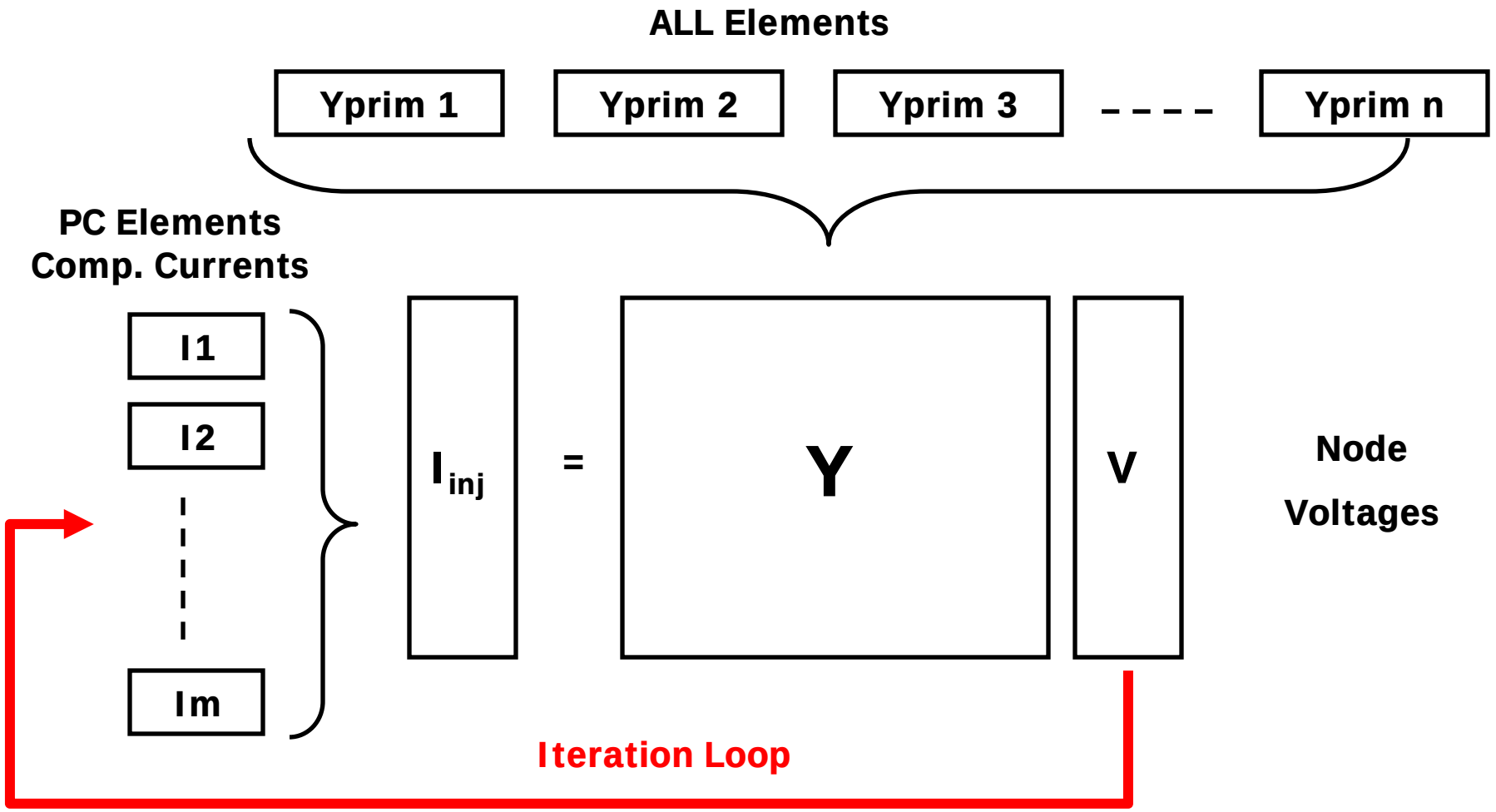


Load - 3-phase Delta connected



3 Conductors/Terminal

Putting it All Together



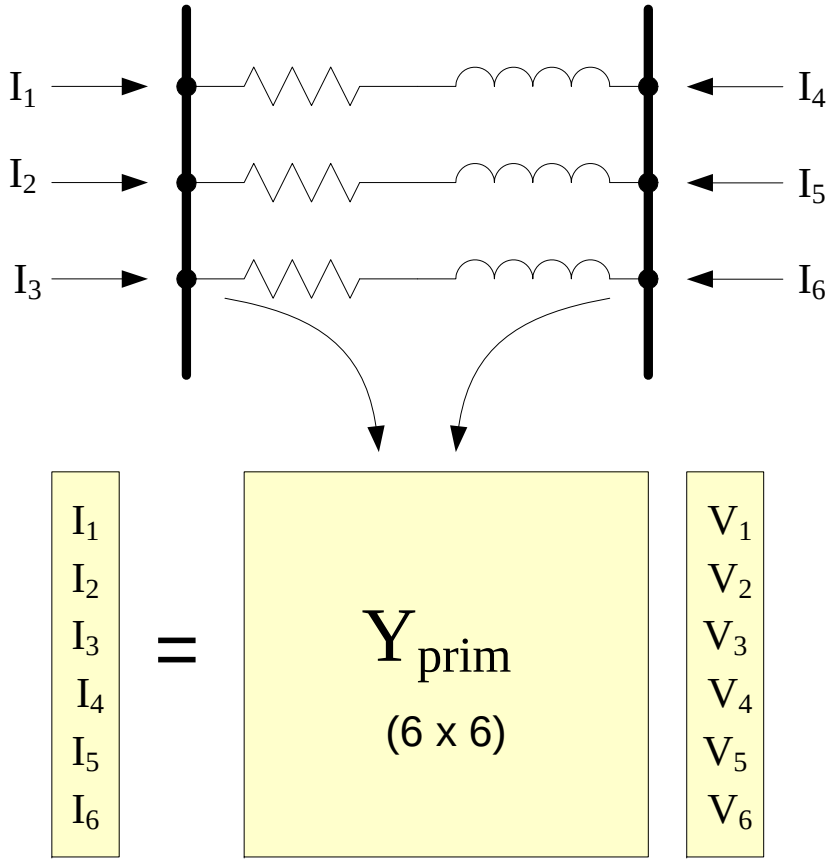
Solution Speed

- Distribution systems generally converge quite rapidly with this method.
- The OpenDSS program seems to be on par with the faster commercial programs – or faster
- It is set up to run annual simulations easily
  - Our recommendation:
    - *Err on the side of running more power flow simulations*
    - That is, don't worry about the solution time until it proves to be a problem
    - That reveals more information about the problem.

How Do You Get Currents and Power If You Only Solve for Node Voltages?

- One thing that troubles some users who are accustomed to other ways of solving power flows is how the branch currents (and powers) are determined when only the Node voltages and Compensation currents are known.
- If the Y matrix is properly formed, and convergence is achieved, the currents will be correct (obey Kirchoff's law at nodes)
- Currents and powers are determined by post processing
- Power criteria are matched by converging with the specified Load criteria
  - i.e., compensation currents

Computing Currents in a Branch



Yprim

- You can obtain the Primitive Y matrix for each element a number of ways (after a Solve)
- Dump command
  - **Dump class.name debug**
    - Or, Dump Class.* debug
- Script
  - **Show Yprim** ! Of active element
  - **Export Yprims** ! All Yprims
- COM Interface
  - **V = DSSCircuit.ActiveElement.NumPhases**

Possible Source of Error!

- If the branch is extremely short (impedance is very low), currents may be incorrectly computed
  - Convergence tolerance is generally 0.0001 pu
  - Voltage solution will be correct enough
- 64-bit math is used throughout
  - You have a fair amount of leeway
  - **However, if voltages at both ends of branch are nearly the same, you will be taking the difference between two nearly equal numbers and the multiplying it by a large number (very high conductance)**
    - **This will magnify any error**
- Do not use impractically short branches

EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Advanced Topics

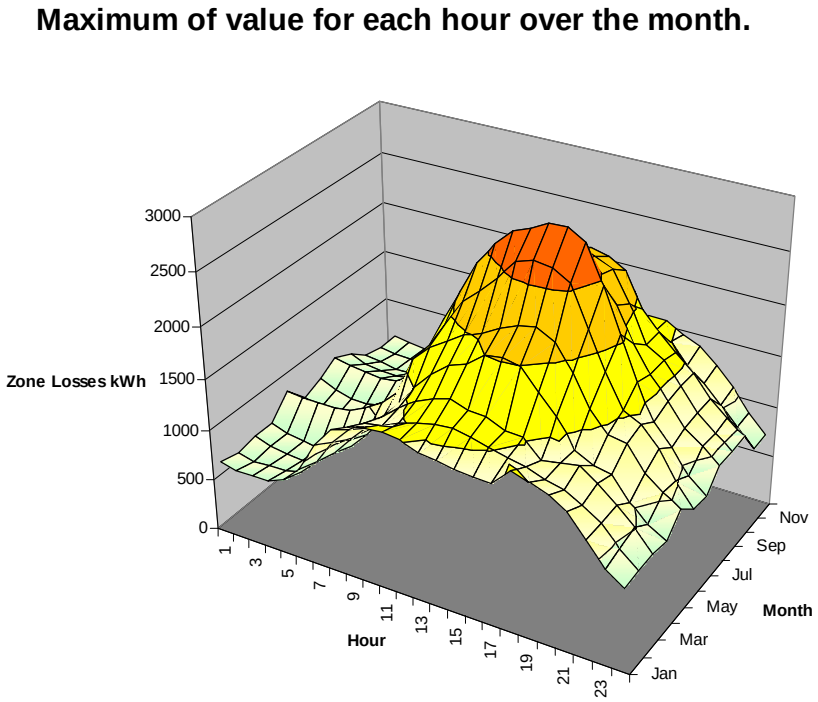
EPRI | ELECTRIC POWER RESEARCH INSTITUTE

Plotting

Ways to Plot

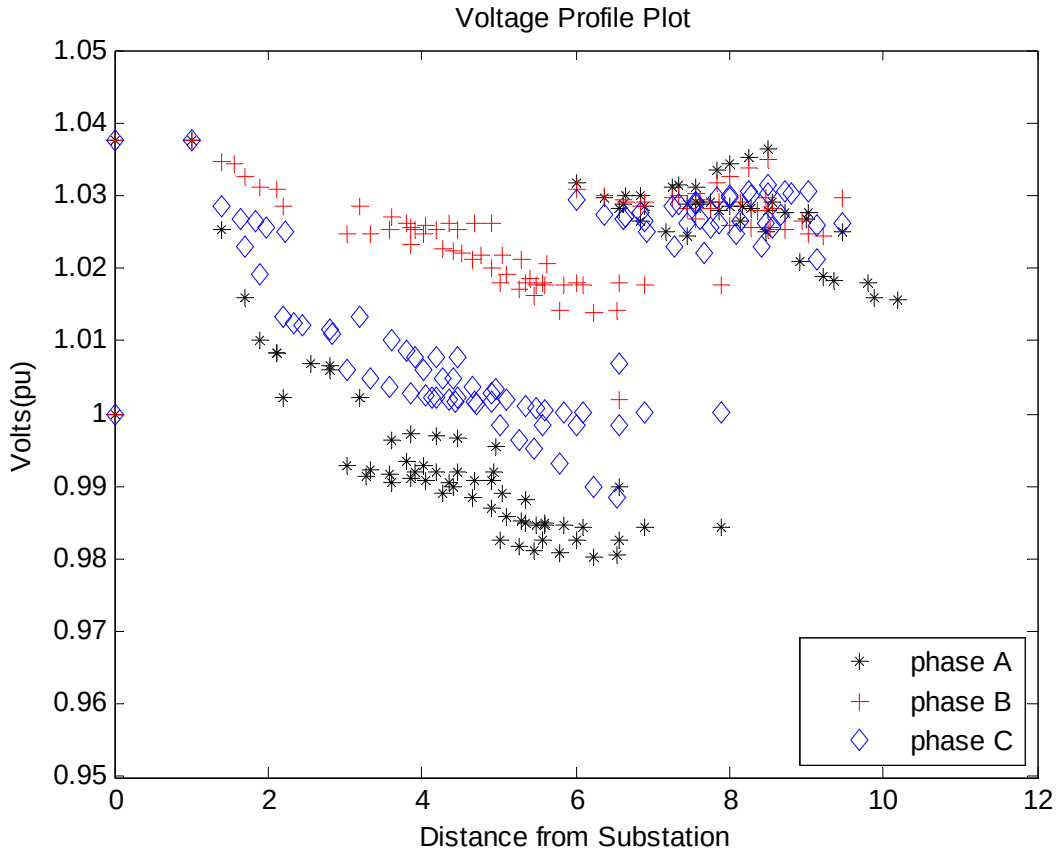
- Use the built-in plotting capabilities

- Plot in an external program, such as Excel or MATLAB

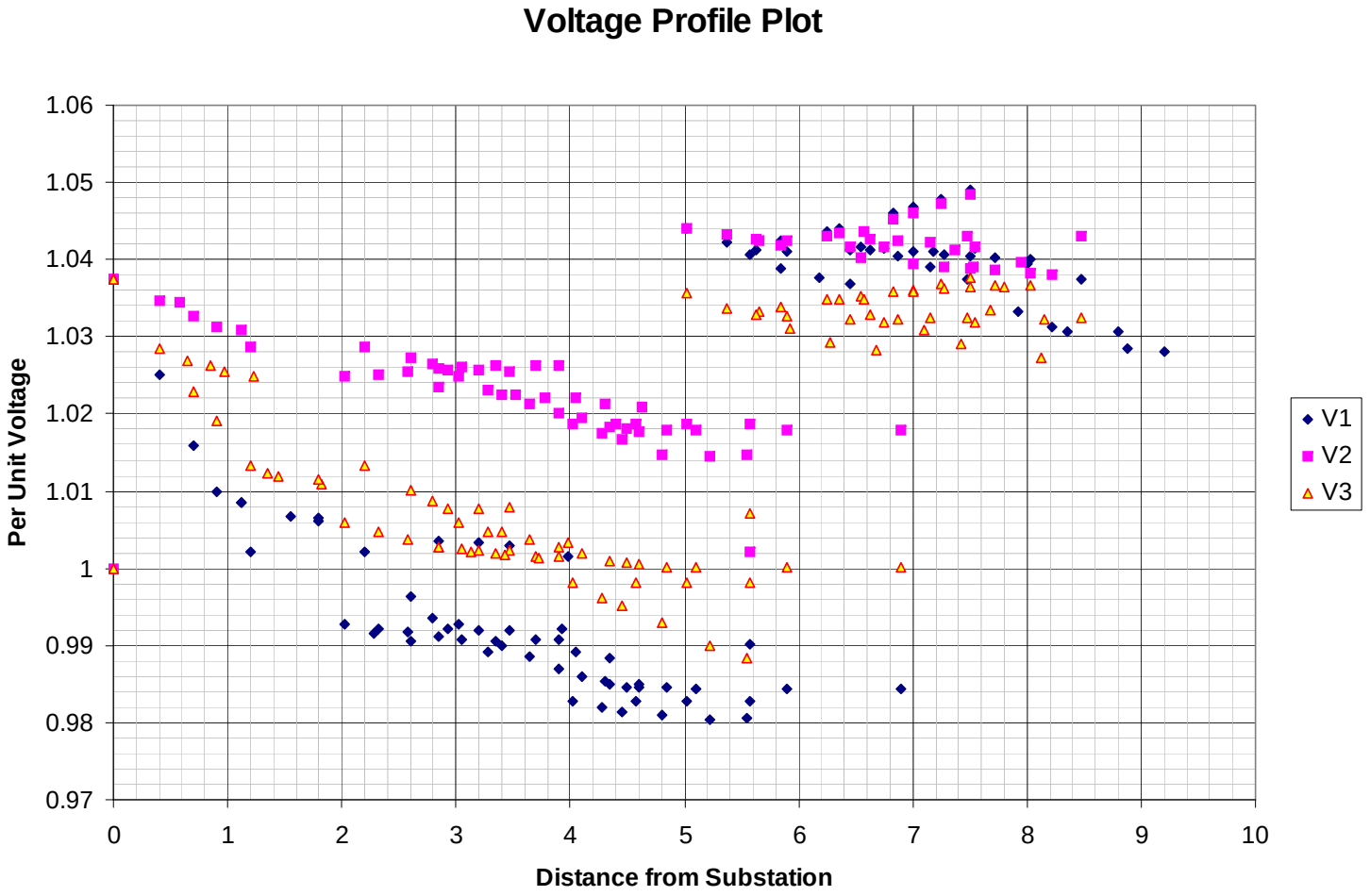


From Excel
(See Example)

From Matlab …



From Excel …



The Plot Command

- **Type =** {Circuit | Monitor | Daisy | Zones | AutoAdd | General (bus data) }
- **Quantity =** {Voltage | Current | Power | Losses | Capacity | (Value Index for General, AutoAdd, or Circuit[w/ file]) }
- **Max =** {0 | value corresponding to max scale or line thickness}
- **Dots =** {Y | N}
- **Labels =** {Y | N}
- **Object =** [metername for Zone plot | Monitor name | File Name for General bus data or Circuit branch data]
- **ShowLoops =** {Y | N} (default=N)
- **R3 =** pu value for tri-color plot max range [.85] (Color C3)
- **R2 =** pu value for tri-color plot mid range [.50] (Color C2)
- **C1, C2, C3 =** {RGB color number}
- **Channels=(**array of channel numbers for monitor plot)
- **Bases=(**array of base values for each channel for monitor plot). Default is 1.0 for each. Set Base= after defining channels.
- **Subs={**Y | N} (default=N) (show substations)
- **Thickness=**max thickness allowed for lines in circuit plots (default=7)
- **Buslist=[**Array of Bus Names | File=filename] (for Daisy plot)

The Plot command, cont'd

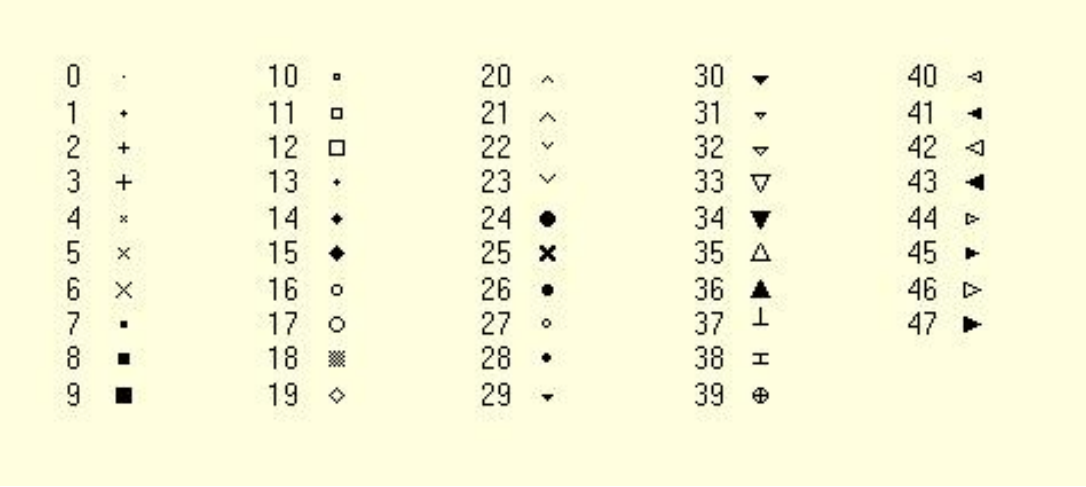
- Power and Losses in kW.
- C1 used for default color (RGB).
  - Hex Format: $00FF00000
- C2, C3 used for gradients, tri-color plots.
- Scale determined automatically if Max = 0 or not specified.

- Examples:
- Plot type=daisy quantity=power max=5000 dots=N !! Generators by default
- Plot daisy power 5000 dots=N Buslist=[file=MyBusList.txt]
- Plot circuit quantity=7 Max=.010 dots=Y Object=branchdata.csv
- Plot General Quantity=2 Object=valuefile.csv

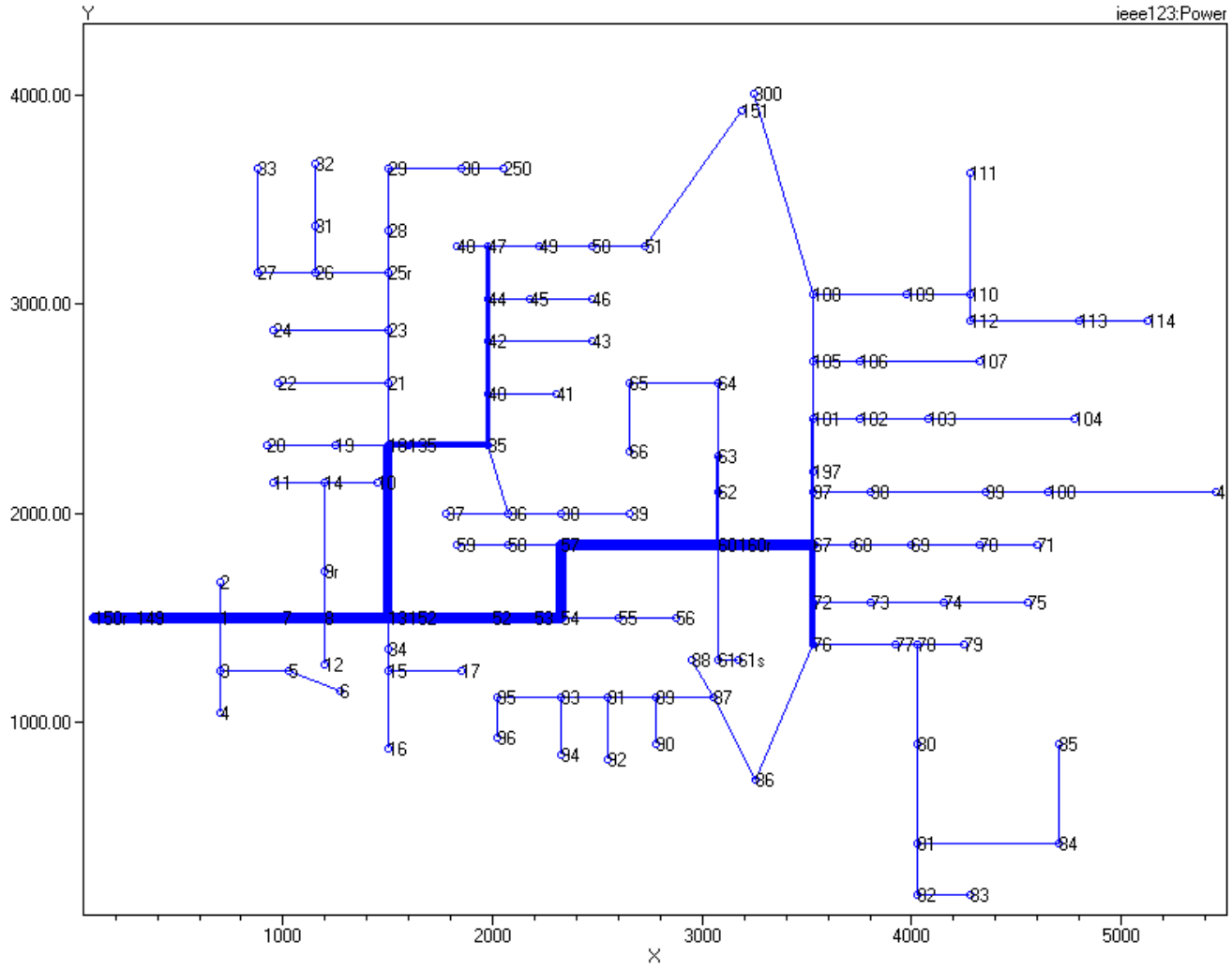
Commands/Options Associated with Plot

- AddMarker Bus=busname code=nn color=$00FF0000 size=3
- Set Nodewidth = nn
- Set MarkerCode = nn

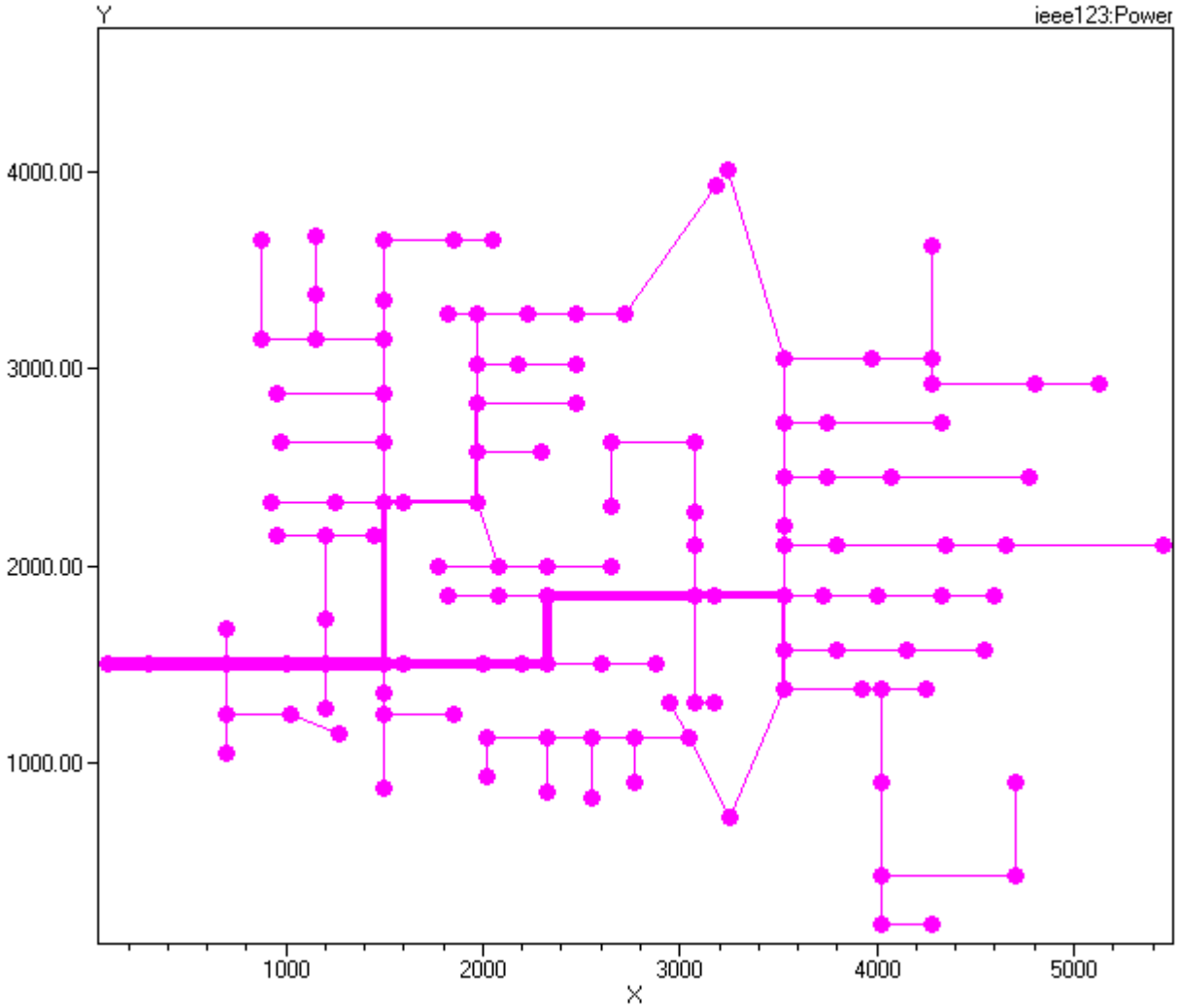
Marker Codes



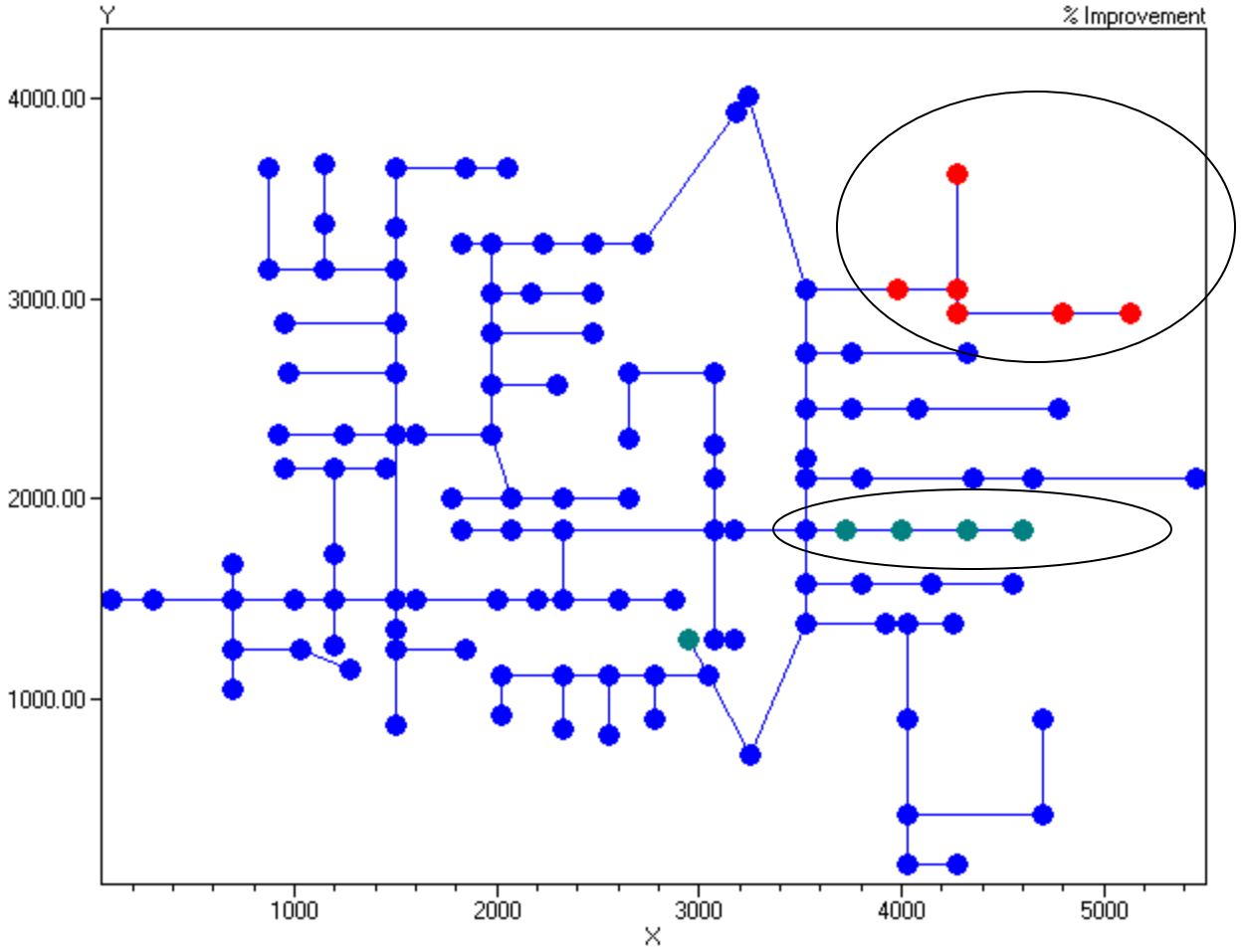
plot circuit Power max=1000 dots=y labels=y C1=$00FF0000



set nodewidth=3 markercode=24
plot circuit Power Max=2000 dots=y labels=n subs=n C1=$00FF00FF

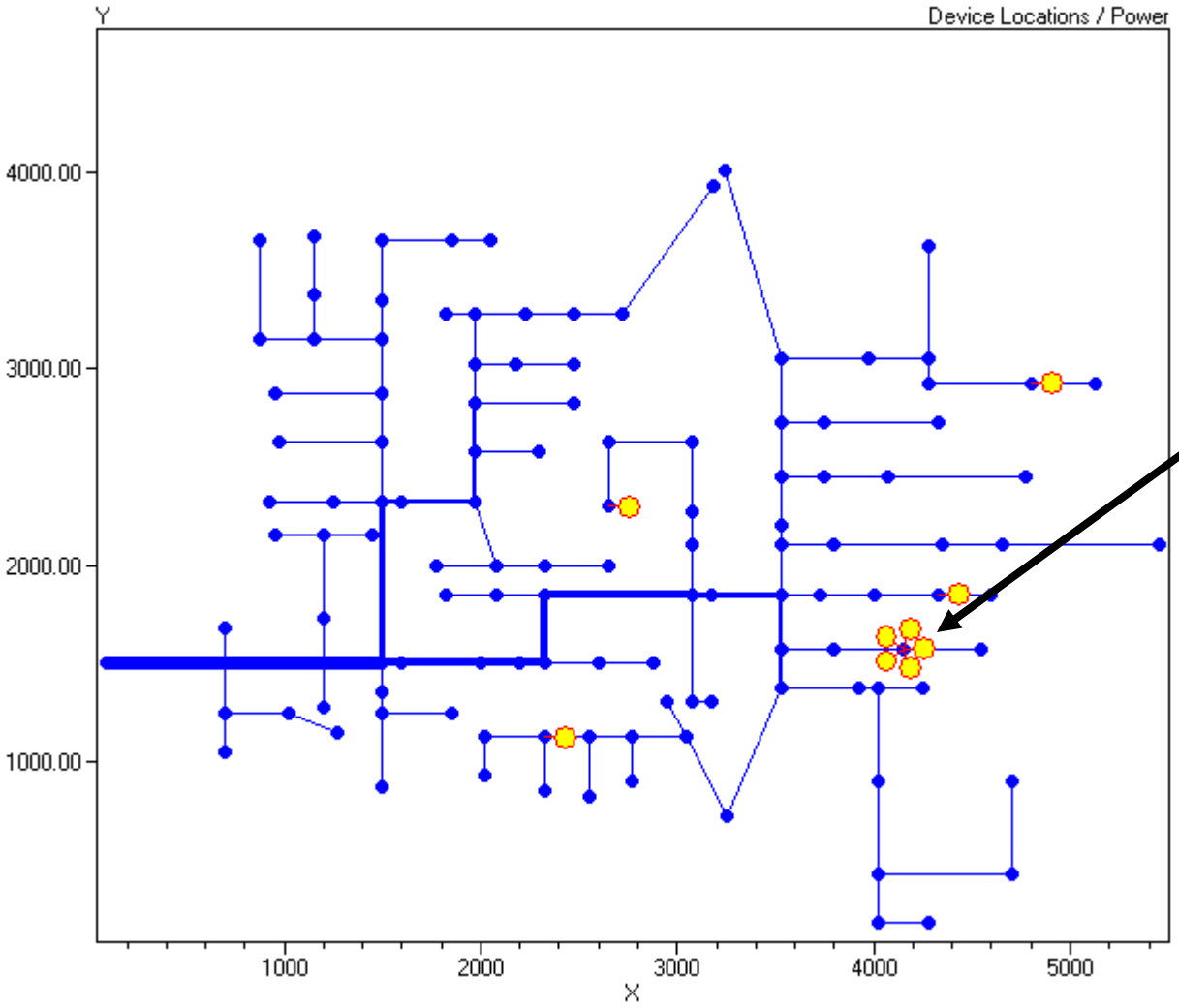


Set Genkw=100
set mode=autoadd
solve
Set nodewidth=7
plot Auto 3 dots=y labels=n subs=n C1=16711680 C2=8421376 C3=255 R3=0.95 R2=0.9



Possibly best areas for adding DG

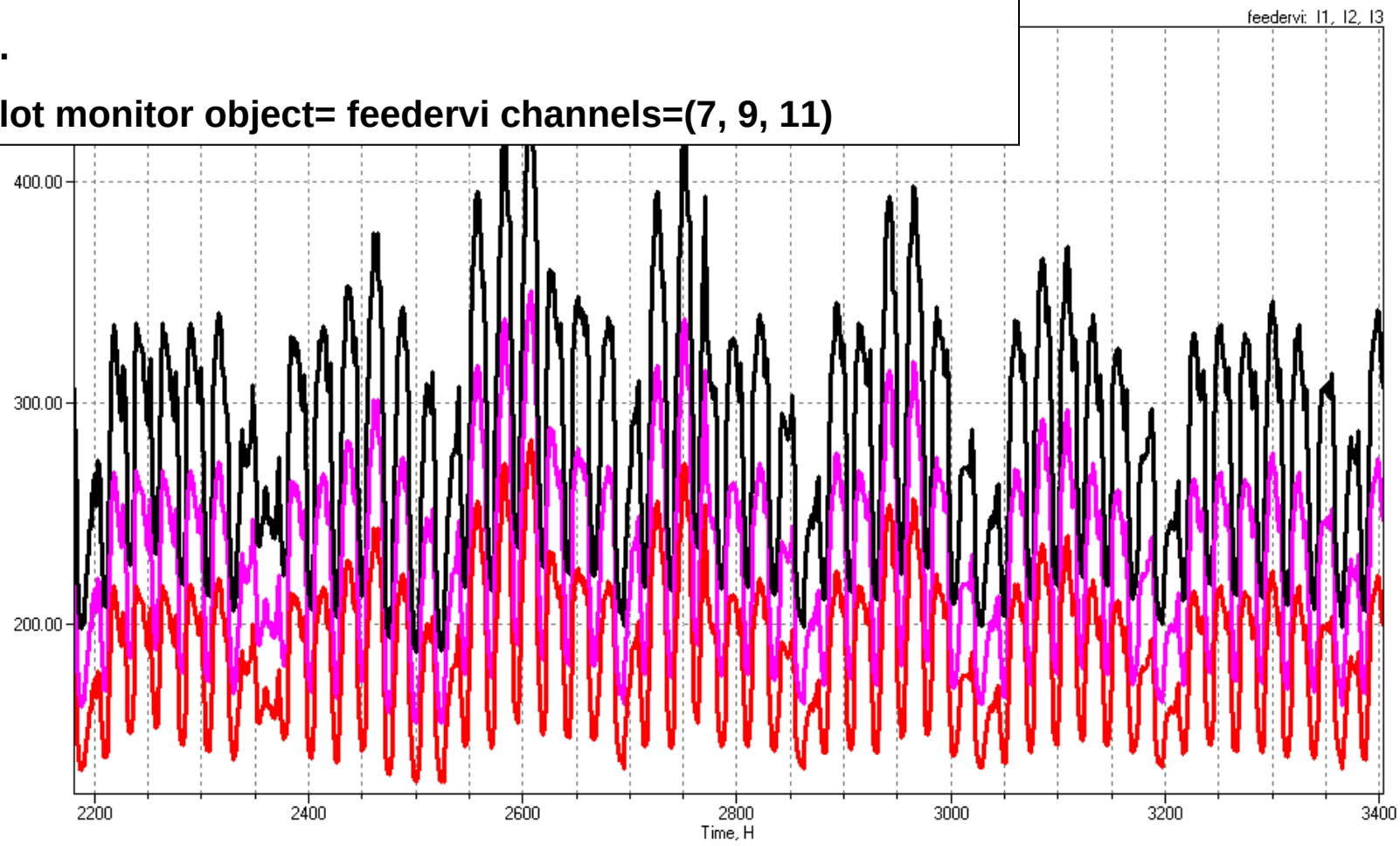
Set nodewidth=1 daisysize=2
plot daisy Power max=2000 y n C1=$00FF0000



Need 500 kW here

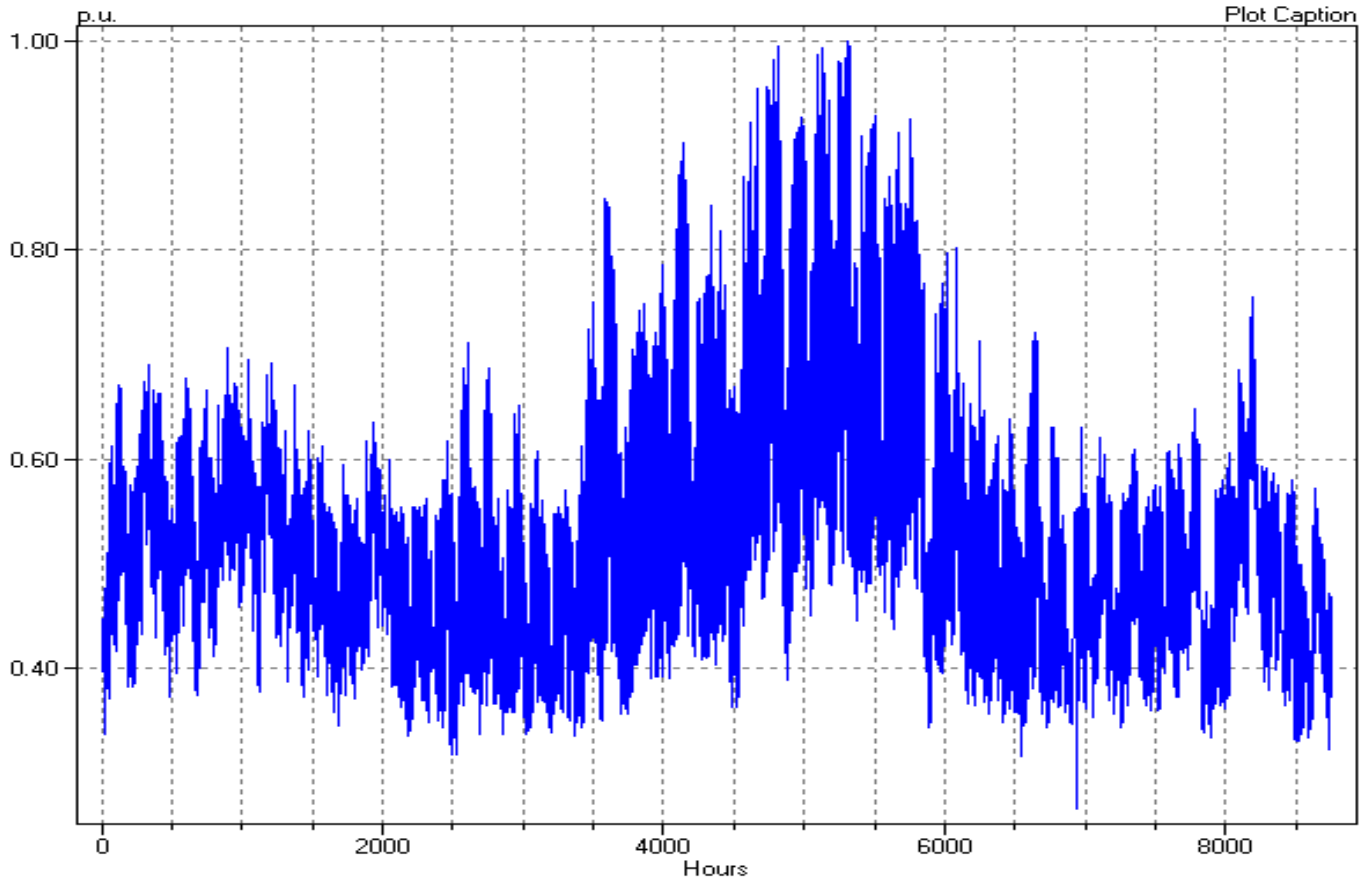
Monitor Plot Of Feeder Currents

```
New Monitor.FeederVI Line.l115 1  Mode=0
...
Plot monitor object= feedervi channels=(7, 9, 11)
```



LoadShape Plot

(Special plot in EXE version only)



EnergyMeter Object

EnergyMeter

- Perhaps the most complex object presently in the DSS
- Emulates an actual energy meter
  - Except it can measure things elsewhere in the meter *zone*.
- Has multiple registers
  - Registers cleared on
    - reset meters (or reset)
    - set mode = ….
    - Set year=
  - Two types: accumulators and “drag hand”

EnergyMeter Registers (Jan 2009)

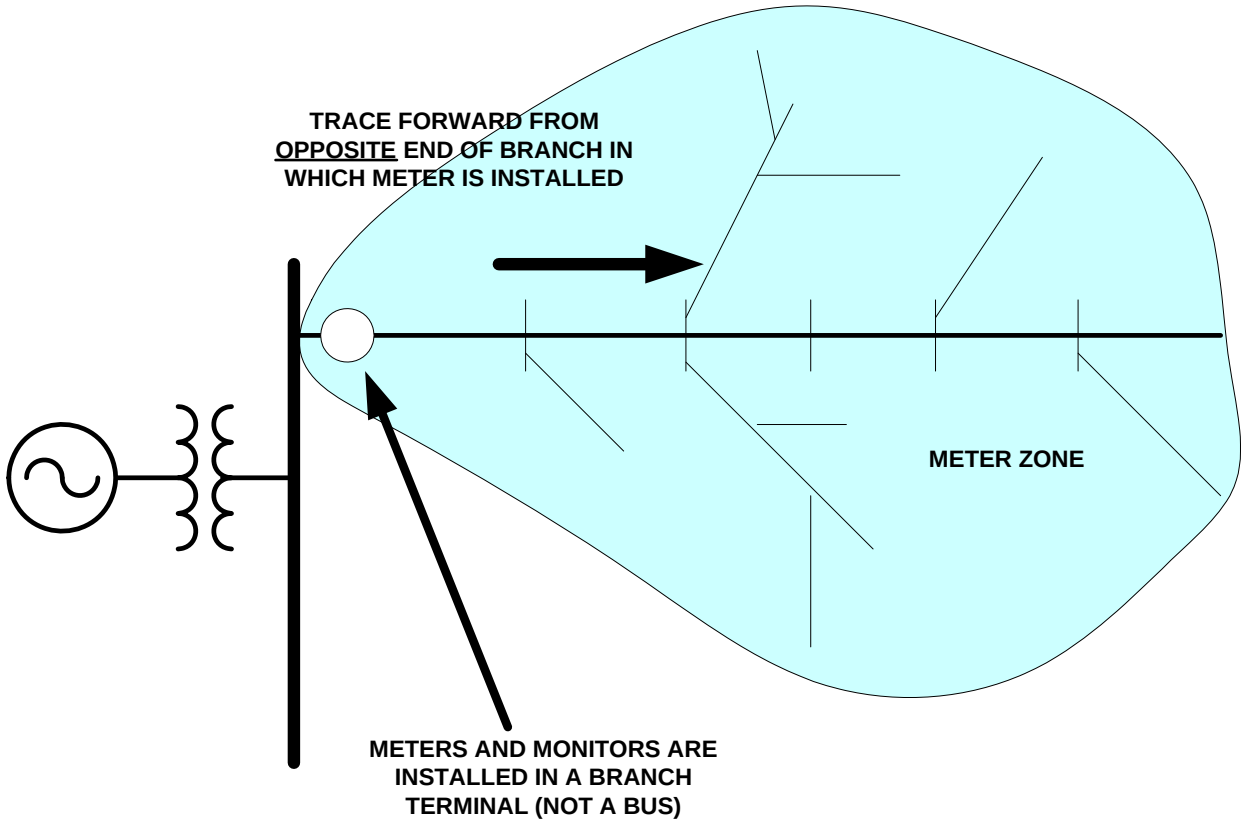
1. KWh at the meter location.
2. Kvarh at the meter location.
3. Maximum kW at the meter location.
4. Maximum kVA at the meter location.
5. KWh in the meter zone.
6. Kvarh in the meter zone.
7. Maximum kW in the meter zone.
8. Maximum kVA in the meter zone.
9. Overload kWh in the meter zone, normal ratings.
10. Overload kWh in the meter zone, emergency ratings.
11. Energy Exceeding Normal (EEN) in the loads in the meter zone.
12. Unserved Energy (UE) in the loads in the meter zone.
13. Losses (kWh) in power delivery elements in the meter zone.
14. Reactive losses (kvarh) in power delivery elements in the meter zone.
15. Maximum losses (kW) in power delivery elements in the meter zone.
16. Maximum reactive losses (kvar) in power delivery elements in the meter zone.
17. Load Losses kWh. I2R Losses in power delivery elements
18. Load Losses kvarh. I2X Losses in power delivery elements
19. No Load Losses kWh in shunt elements, principally transformers.
20. No Load Losses kvarh in shunt elements.
21. Max kW Load Losses during the simulation
22. Max kW No Load Losses during the simulation
23. Line Losses: Losses in LINE elements.
24. Transformer Losses: Losses in TRANSFORMER elements.
25. Line Mode Line Losses (3X Pos and neg seq losses)
26. Zero Mode Line Losses (3X zero sequence losses)
27. 3-phase Line Losses
28. 1- and 2-phase Line Losses
29. Gen kWh
30. Gen kvarh
31. Gen Max kW
32. Gen Max kVA
33. Aux1 (used for segregating losses by voltage level)
34. Aux2
35. Aux3
36. Aux4
37. Aux5
38. Aux6
39. Aux7

Meter Zone

- Collection of circuit elements “downline” from meter.
- Only element in DSS that knows about radial circuits
- Zone is established first time solution is executed
  - May be more time-consuming than actual solving for very large circuits.
  - Rebuilt whenever bus list is rebuilt
- EnergyMeter and Monitor objects are installed in a branch terminal
  - **`New Energymeter.example Element=Line.Line1 Terminal=1`**

Meter Zone, cont'd

- Zone is traced from the opposite end of the branch



Meter Zone, cont'd

- Plotting Meter Zone
  - **`plot zone Power max=2000 n n object=(metername) C1=$00FF0000`**
- Showing Meter Zone
  - **`Show zone metername`**
- Zone dump
  - **`energymeter.metername.action=zonedump`**
  - **`Or`**
  - **`Edit energymeter.metername  action=zonedump`**

Some Things That Require a Meter Zone

- Loss Analysis
- Excess load analysis
- Plotting zones if different colors
- Distance from substation (distance from meter)
- Reconductor Command (needs to trace back)

Monitor or Meter?

- **Monitor** measures quantities only where it is located
  - Takes a sample of quantity
  - Voltage and current (several options)
  - Powers
  - Transformer taps
  - State vars

- **EnergyMeter** measures power and integrates some, samples others
  - Samples quantities throughout its zone

Introduction to Driving the COM Server from another Application

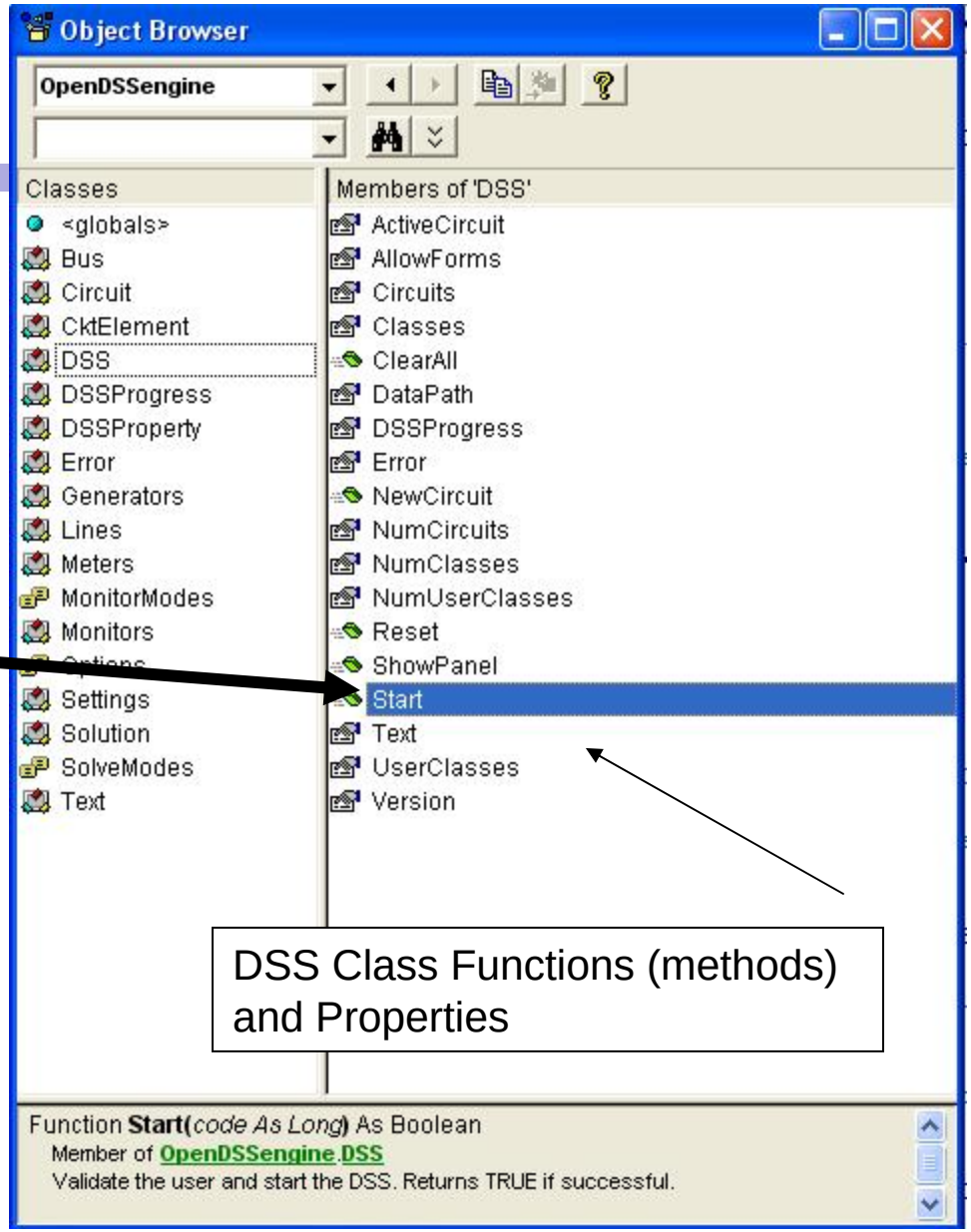
Active objects concept

- There is one registered In-Process COM interface:
  - *OpenDSSEngine.DSS*
    - That is, the DSS interface is the one you instantiate
    - The DSS interface creates all the others.
- The interfaces generally employ the idea of an **ACTIVE object**
  - Active circuit,
  - Active circuit element,
  - Active bus, etc.
  - The interfaces generally point to the active object
    - To work with another object, change the active object.

DSS Interface

This interface is instantiated upon loading OpenDSSEngine.DSS and then instantiates all other interfaces

Call the Start(0) method to initialize the DSS



Instantiate the DSS Interface and Attempt Start

```
Public Sub StartDSS()

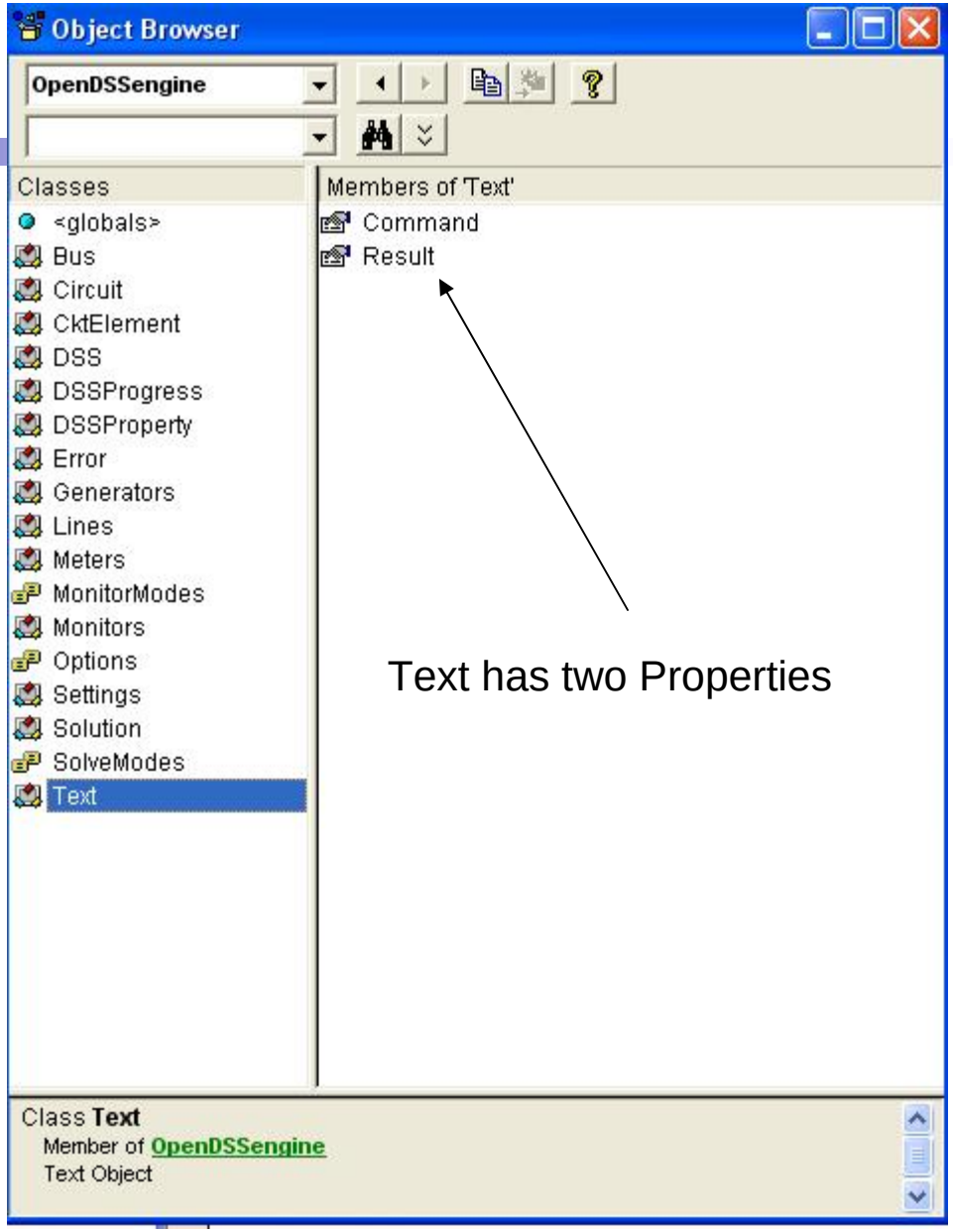
' Create a new instance of the DSS
    Set DSSobj = New OpenDSSengine.DSS
' Start the DSS
    If Not DSSobj.Start(0) Then
        MsgBox "DSS Failed to Start"
    Else
        MsgBox "DSS Started successfully“
      ' Assign a variable to the Text interface for easier access
        Set DSSText = DSSobj.Text
    End If


End Sub
```

COM Interface

Interfaces as Exposed by VBA Object Browser in MS Excel

Text interface is simplest



Text has two Properties

Assign a Variable to the Text Interface

```
Public Sub StartDSS()

' Create a new instance of the DSS
    Set DSSobj = New OpenDSSengine.DSS
' Start the DSS
    If Not DSSobj.Start(0) Then
        MsgBox "DSS Failed to Start"
    Else
        MsgBox "DSS Started successfully“
      ' Assign a variable to the Text interface for easier access
        Set DSSText = DSSobj.Text
    End If


End Sub
```

Now Use the Text Interface …

- You can issue any of the DSS script commands from the Text interface

```
‘ Always a good idea to clear the DSS when loading a new circuit
    DSSText.Command = "clear"
' Compile the script in the file listed under "fname" cell on the main form
    DSSText.Command = "compile " + fname
‘ Set regulator tap change limits for IEEE 123 bus test case
With DSSText
    .Command = "RegControl.creg1a.maxtapchange=1  Delay=15  !Allow only one tap change per solution.
   This one moves first"
    .Command = "RegControl.creg2a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg3a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4a.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg3c.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4b.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "RegControl.creg4c.maxtapchange=1  Delay=30  !Allow only one tap change per solution"
    .Command = "Set MaxControlIter=30"
  End With
```

Result Property

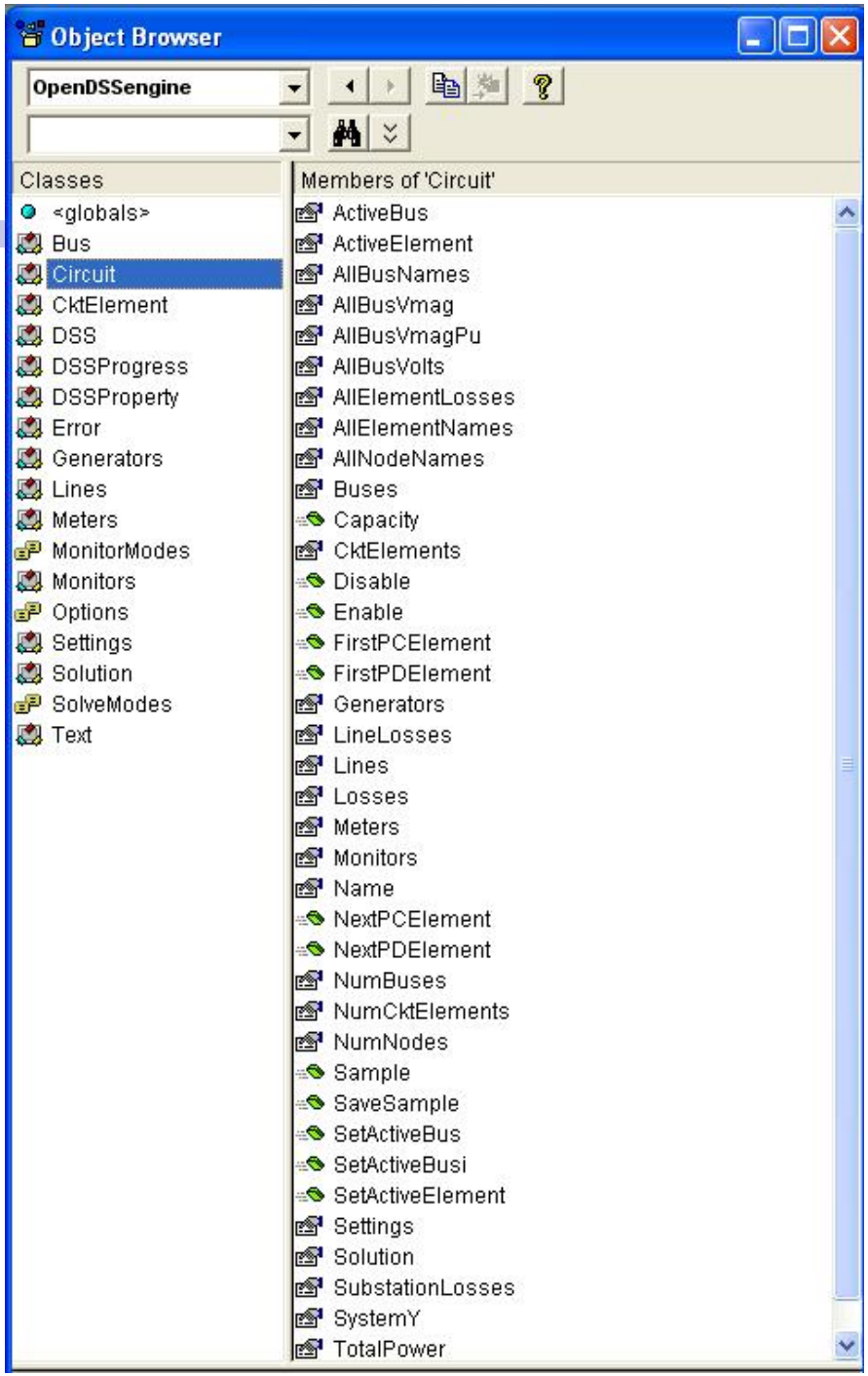
- The Result property is a Read Only property that contains any result messages the most recent command may have issued.
  - Error messages
  - Requested values

```
‘ Example: Query line length
DSSText.Command = “? Line.L1.Length”
S = DSSText.Result      ‘ Get the answer
MsgBox S          ‘ Display the answer
```

Circuit Interface

This interface is used to

1) Get many of the results for the most recent solution of the circuit
2) Select individual circuit elements in a variety of ways
3) Select the active bus
4) Enable/Disable circuit elements



Circuit Interface

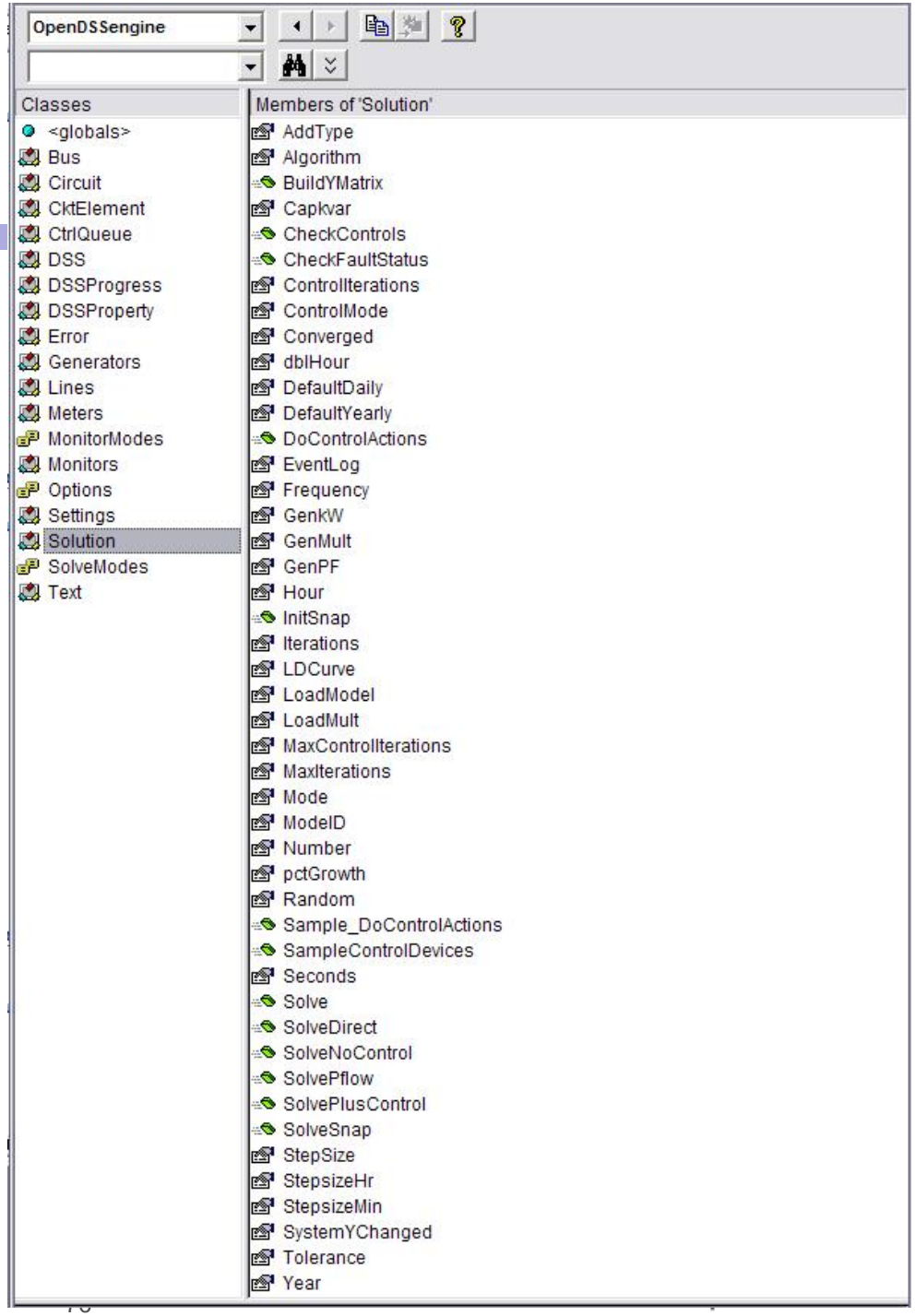
Since the Circuit interface is used often, it is recommended that a special variable be assigned to it:

```
Public DSSCircuit As OpenDSSengine.Circuit
…
DSSText.Command = “Compile xxxx.dss”
Set DSSCircuit = DSSobj.ActiveCircuit
DSSCircuit.Solution.Solve
…  ‘ Retrieving array quantities into variants
V = DSSCircuit.AllBusVmagPu
VL =DSSCircuit.AllElementLosses
```

Solution Interface

The Solution Interface is used to

1) Execute a solution
2) Set the solution mode
3) Set solution parameters (iterations, control iterations, etc.)
4) Set the time and time step size



Solution Interface

Assuming the existence of a DSSCircuit variable referencing the Circuit interface

```
Set DSSSolution = DSSCircuit.Solution
With DSSSolution
…
     .LoadModel=dssAdmittance
     .dblHour = 750.75
     .solve

End With
```

Use the With statement in VBA to simplify coding

CktElement Interface

This interface provides specific values of the Active Circuit Element

Some values are returned as variant arrays

```
V = DSSCircuit.ActiveElement.Powers

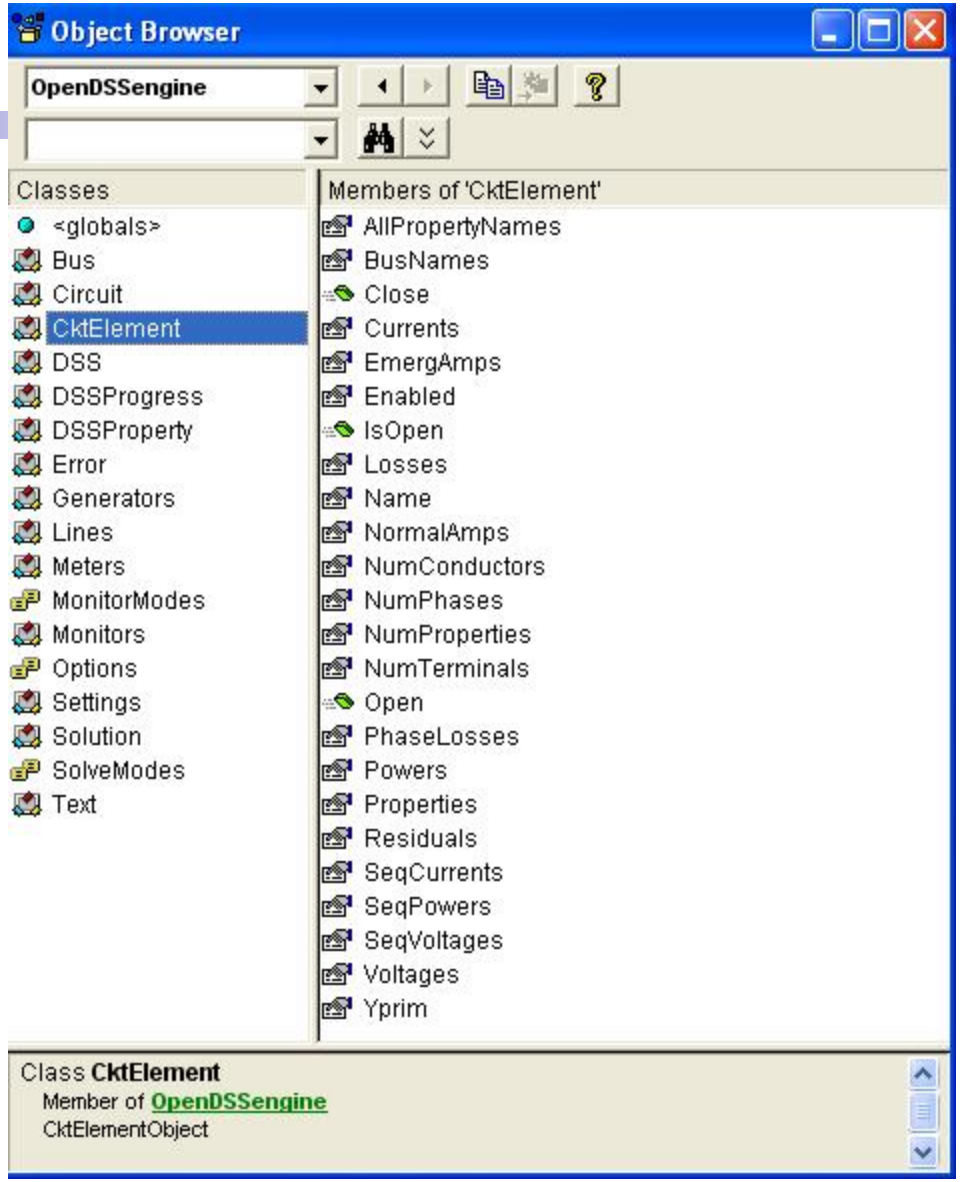
V = DSSCircuit.ActiveElement.seqCurrents

V = DSSCircuit.ActiveElement.Yprim
```

Other values are scalars

```
Name = DSSCircuit.ActiveElement.Name

Nph = DSSCircuit.ActiveElement.NumPhases
```



Properties Interface

This interface gives access to a String value of each public property of the active element

“Val” is a read/write property