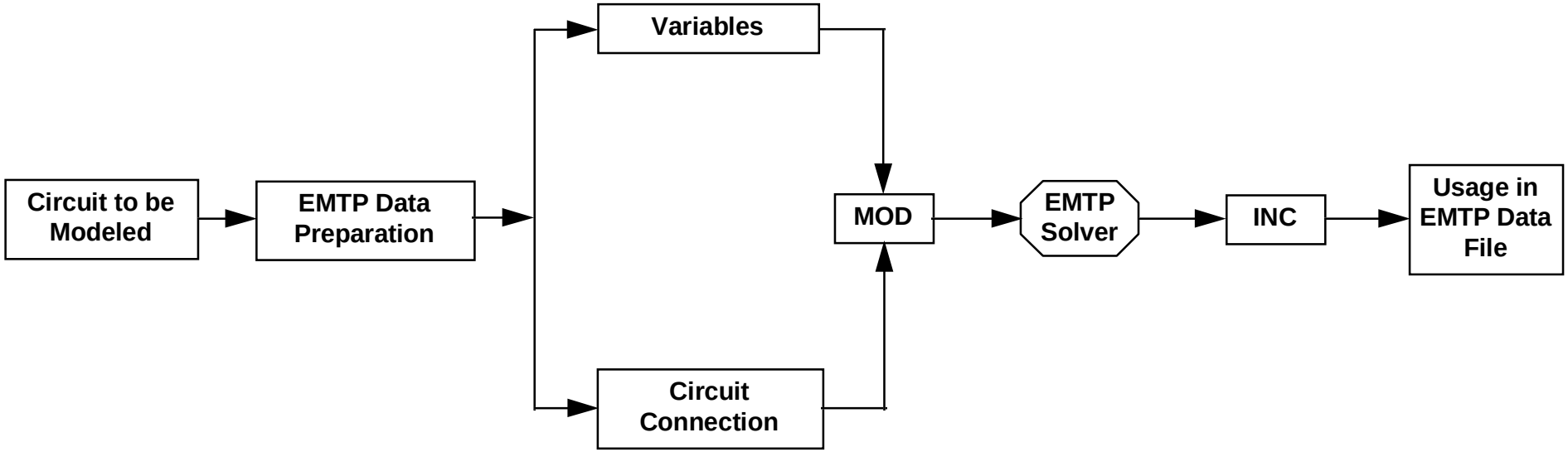
Introduction to EMTP Data Modules

- Overview of EMTP Data Module (EDM) Development:
  - Creating a data module
  - Data module usage
  - Examples



EMTP Data Modules - When and Why?

- In general, a user should have already developed the equivalent model in an EMTP data case, and then wish to convert the data into a reusable data module.
- If a component/model is only going to be used once, the user will need to determine the usefulness of converting it into a module.
- A significant advantage of module usage is that they can be shared by a group, with one person responsible for development / maintenance and the others having access to the model.

EMTP Data Modules - Overview

- Grouping of Input Data (/ Cards)
  - Grouping of Data Logically, Rather than by Rules
- Argument Substitution
  - Character Strings (Bus Names), Numbers
- Transportability and Reliability
- Ease of Use
  - $INCLUDE Special Request Card

Rule Book: 19.1

Grouping of Input Data

- Grouping of EMTP data is accomplished using "/" cards:
  - Method allows data to be grouped logically, rather than with the traditional EMTP structure:

```
Miscellaneous Data Cards
Branch Data
BLANK End of Branch Data
Switch Data
Blank End of Switch Data
etc.
```

  - Allows large complex data sets (i.e. PWM inverter/motor) to be divided into smaller more manageable groups of data.
  - Six levels of nesting are allowed in data modules.

Argument Substitution

- Data modules provide the facility to substitute variables (arguments), rather that simply inserting a portion of data into the larger data case.
- Concept is similar to programming procedures or functions.

```
Function Root (iRoot, value)
Dim iRoot as Integer
Dim value as Real
   Root = value^iRoot
End Function
```

- Arguments may be strings or numeric quantities.

Transportability/Reliability & Ease of Use

- Each data module resides in a separate data file - once development and testing is completed it requires no maintenance.
- Possibility of unknown errors is reduced.
- Modules can be reused in the same data case.
- Modules can be reused in different studies.
- Modules are easily incorporated using “$INCLUDE” cards.

EDM Preamble / Declaration

- A module may have three types of declarations, ‘ARG’, ‘NUM’, and ‘DUM’.
  - ‘ARG’ argument
  - ‘NUM’ numeric constants
  - ‘DUM’ internal or dummy variables

```
ARG TERML,                                          -    ; NODES
ZERORE, ZEROINDUCTAN, POSIRE, POSIINDUCTAN,         -    ; SEQUENCE IMP.
AMPLITUDES, TSTARTTIME, TSTOPTIMES, RDAMP1               ; V, TSTART, TSTOP
C
NUM                                                 -    ;
ZERORE, ZEROINDUCTAN, POSIRE, POSIINDUCTAN,         -    ; SEQUENCE IMP.
AMPLITUDES, TSTARTTIME, TSTOPTIMES, RDAMP1               ; V, TSTART, TSTOP
C
DUM INTERA, INTERB, INTERC                               ; INTERNAL SOURCE
```

EDM Preamble - Special Characters

- Special characters:

‘-’	continuation character
‘;’	indicate rest of line is comment
‘#’	indicate imbedded blanks
‘_’	indicate imbedded blanks in numeric field
‘?’,’@’	one character substitution in column 80

External Variables - Type ARG

- Type ‘ARG’ stands for argument and is used to pass all external variables to the module.
- All external variables (numeric or character string) must be declared as type ‘ARG’.
- Variable type ‘ARG’ does not justify. It replaces a pattern as given in the argument list.
- A variable can be padded with blanks using the special character ‘#’.

EDM Guidelines - Node Name Creation

- Single bus and three-phase node name examples:

```
BUS01A - generates a name of:   BUS01A
                                .....^

BUS##A - generates a name of:   BUS  A
                                .....^
```

```
C <---Nodes--><---Refer--><-ohms<---mH<---uF
C <-Bus1<-Bus2<-Bus3<-Bus4<----R<----L<----C
  FAULTA                    0.001
  FAULTB                    0.001
  FAULTC                    0.001
C .....^.....^.....^.....^.....^.....^.....^
```

```
C <---Nodes--><---Refer--><-ohms<---mH<---uF
C <-Bus1<-Bus2<-Bus3<-Bus4<----R<----L<----C
  BUS01A                    0.001
  BUS01B                    0.001
  BUS01C                    0.001
C .....^.....^.....^.....^.....^.....^.....^
```

Numeric Variables - Type NUM

- Type ‘NUM’ is a subset of type ‘ARG’ and informs the program to expect a numeric string instead of an alphanumeric string.
- These variables replace numeric fields in the EMTP data.
- Users must refer to the actual EMTP rules - specify the width of the variable as the full width of the numeric field.
- Users may use the special character ‘_’ to imbed blanks in the field.
- In the “$INCLUDE” statement, users may specify a numeric string that is less than the variable width - If this occurs the quantity is right justified.

EDM Guidelines - Numeric Quantities

- For example: RDAMP1 refers to a resistor from nodes BUS01 to BUS02

```
C <---Nodes--><---Refer--><-Ohms<---mH<---uF
C Bus1->Bus2->Bus3->Bus4-><----R<----L<----C
  BUS01ABUS02A              RDAMP1
C .....^.....^.....^.....^.....^.....^.....^
```

- When the module is used: ‘230KV’, ‘SEND1’, and ‘200.00’ are used

```
C <---Nodes--><---Refer--><-Ohms<---mH<---uF
C Bus1->Bus2->Bus3->Bus4-><----R<----L<----C
  230KVASEND1A              200.00
C .....^.....^.....^.....^.....^.....^.....^
```

Dummy Variables - Type DUM

- Type ‘DUM’ refers to internal variables of a module.
- Variables of type ‘DUM’ must have a six character width and should begin with an alphanumeric character.
- An internal variable is replaced with a unique name created by the program.
- The program currently has 6000 dummy variables available.

Seed Key	First Name	Last Name
DUM	DUM000	DUM999
DMU	DMU000	DMU999
MDU	MDU000	MDU999
MUD	MUD000	MUD999
UMD	UMD000	UMD999
UDM	UDM000	UDM999

EDM Body

- The input formats and rules that need to be followed while building a MOD file are very similar to those while developing a regular EMTP data file.
- A major difference between a MOD file and a regular input file is that in the MOD file general character strings are used to replace all the determined node names and numerical values specified in the regular input file.
- In addition, all of the used character strings need to be declared as proper types.

EDM Body - Data Sorting

- Data sorting - "/" cards:

/BRANCH	indicate branch data
/SWITCH	indicate switch data
/SOURCE	indicate source data
/TACS	indicate TACS data
/OUTPUT	indicate output data
/PLOT	indicate plot data
/REQUEST	indicate special request card
/STATISTICS	indicate statistics request cards
/END MODULE	indicate end of a data module

EDM Body - '$' Request Cards

- '$' cards request the program to:

  [1] Read another input file

  [2] Set file attributes, such as directory

  [3] Modify the contents of card images using argument list

  [4] Change the 'seed key' of internal variables

EDM Guidelines - Converting MOD to INC

- Process for Creating Modules:

  [1] Write Module File (*.MOD)

  [2] Convert to Include File (*.INC)

```
MODULE
D:\EMTPFILE\WORKBOOK\VOLUME2\3PHEQUIV.MOD
D:\EMTPFILE\WORKBOOK\VOLUME2\3PHEQUIV.INC
STOP
```

  [3] Use Module via $INCLUDE Request Card

```
$PREFIX, D:\EMTPFILE\WORKBOOK\VOLUME2\
$SUFFIX, .INC
```

Module Usage

- $INCLUDE
  - Intructs program to read another input file:

```
$INCUDE   filename  argument, argument, argument, - ; comment
          argument, argument                        ; comment
```

- $PREFIX / $SUFFIX
  - Informs the program of common file attributes (path, extension):

```
$PREFIX  prefix name
$SUFFIX  suffix name
```

- BLANK Cards

Module Usage - Blank Cards

- Blank cards are required to terminate an EMTP data classes.
- Standard blank cards may not be used within modules and data files using modules.
- The keywork “BLANK ENDS” is used at the end of the data file.

```
BLANK ENDS TACS
BLANK ENDS BRANCH
BLANK ENDS SWITCH
BLANK ENDS SOURCE
BLANK ENDS OUTPUT
BLANK ENDS PLOT
BEGIN NEW DATA CASE
BLANK END OF ALL DATA CASES
```

EDM Guidelines - Some General Rules

- There are several important rules to follow when creating modules. They include:

  [1] Each data block must start with the proper '/' card

  [2] A '$' card can be part of a data block

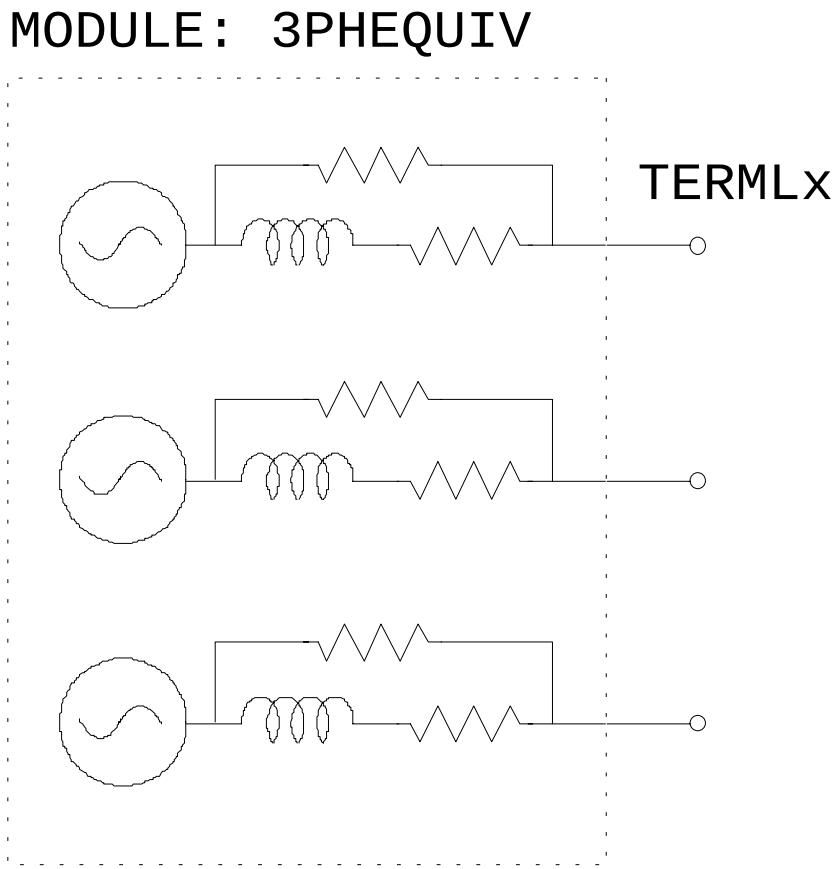
  [3] If an '$INCLUDE' card is embedded in a data block then it must be followed with the proper '/' card

  [4] The first eleven (11) characters of a line in a module must not be 'BLANK ENDS'. This pattern is a keyword and is used at the end of the EMTP data file.

  [5] The body of a module must end with a '/ENDMODULE' card, followed by a '$EOF' card.

Module Example #1 - 3PHEQUIV

- Three-phase source equivalent - **3PHEQUIV**



Module 3PHEQUIV - Argument List

Argument	Description	Type	Length
TERML	Output terminal name 5 characters + (A)(B)(C)	ARG	5
ZERORE	Zero sequence source resistance (Ω)	NUM	6
ZEROINDUCTAN	Zero sequence source inductance (mH)	NUM	12
POSIRE	Positive sequence source resistance (Ω)	NUM	6
POSIINDUCTAN	Positive sequence source inductance (mH)	NUM	12
AMPLITUDES	Voltage source (peak voltage, phase-to-ground)	NUM	10
TSTARTTIME	Start time (sec), -1.0 for steady-state	NUM	10
TSTOPTIMES	Stop time (sec), 9999. for entire simulation	NUM	10
RDAMP1	Parallel damping resistor (Ω)	NUM	6
COM	Comment	N/A	N/A
INTERA(B)(C)	Internal nodes for voltage source connection	DUM	6

Module 3PHEQUIV - .MOD File

```
ARG TERML,                                        -   ; ELECTRIC NODES
ZERORE, ZEROINDUCTAN, POSIRE, POSIINDUCTAN,       -   ; SEQUENCE IMP.
AMPLITUDES, TSTARTTIME, TSTOPTIMES, RDAMP1            ; V, TSTART, TSTOP
C
NUM                                               -   ;
ZERORE, ZEROINDUCTAN, POSIRE, POSIINDUCTAN,       -   ; SEQUENCE IMP.
AMPLITUDES, TSTARTTIME, TSTOPTIMES, RDAMP1            ; V, TSTART, TSTOP
C
DUM INTERA, INTERB, INTERC                            ; INTERNAL SOURCE CONN.
C
/BRANCH
C <---Nodes--><---Refer--><-Ohms<---------mH
C <-Bus1<-Bus2<-Bus3<-Bus4<----R<----------L
51TERMLAINTERA            ZEROREZEROINDUCTAN
C .....^.....^.....^.....^.....^..........^  Zero Sequence
52TERMLBINTERB            POSIREPOSIINDUCTAN
C .....^.....^xxxxxxxxxxx.....^..........^  Positive Sequence
53TERMLCINTERC
C .....^.....^xxxxxxxxxxx.....^..........^
C
C <---Nodes--><---Refer--><-Ohms<---mH<---uF<-------------------------------Out
C Bus1->Bus2->Bus3->Bus4->><----R<----L<----C                                V
  TERMLAINTERA            RDAMP1
  TERMLBINTERB            RDAMP1
  TERMLCINTERC            RDAMP1
C .....^.....^.....^.....^.....^.....^.....^xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx^
C
/SOURCE
C <--Bus<I<-----Ampl<-----Freq<----Phase<-------A1<------T1><---Tstart<----Tstop
14INTERA 1AMPLITUDES      60.0       0.0                  TSTARTTIMETSTOPTIMES
14INTERB 1AMPLITUDES      60.0    -120.0                  TSTARTTIMETSTOPTIMES
14INTERC 1AMPLITUDES      60.0     120.0                  TSTARTTIMETSTOPTIMES
C .....^.^.........^.........^.........^.........^.........^.........^.........^
/ENDMODULE
$EOF
```

Module 3PHEQUIV - .INC File

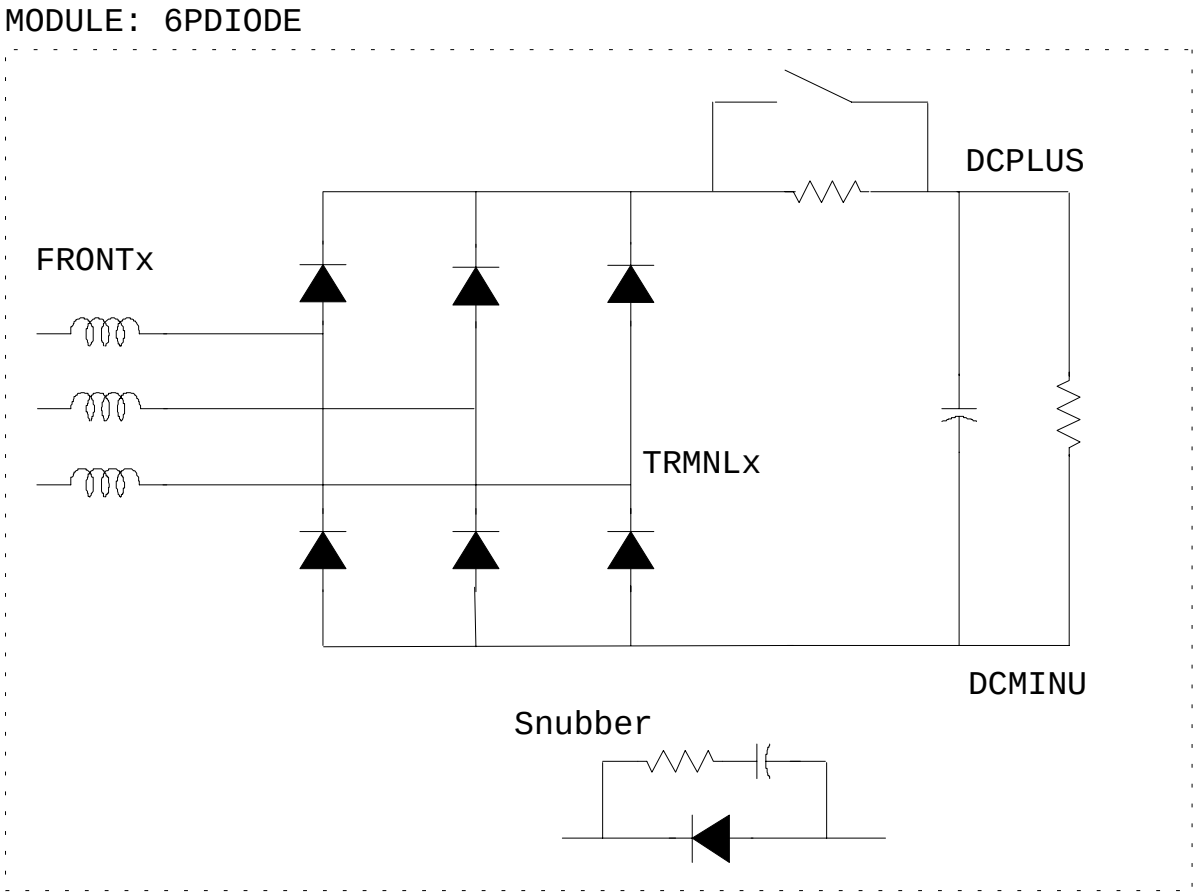
```
KEND    7   32   44   14    7   32   44   14    7   14    7   32   14    7   32
       14    7   32   14   20   70   80    8   20   70   80    8   20   70   80
        8
KTEX    1    0    0    1    1    0    0    1    1    1    1    0    1    1    0
        1    1    0    1    0    0    0    1    0    0    0    1    0    0    0
        1
C EMTP Data Module: 3PHEQUIV
C
C Description: Three-Phase Equivalent Source
C
C Created: 5/2/94 (TEG/LT)  Last Updated: 5/2/94 (TEG)
C
C Usage: 3PHEQUIV TERML, R0, L0, R1, L1, VPEAK, TSTART, TSTOP, RDAMP1 ; COM
C
/BRANCH
C <---Nodes--><---Refer--><-Ohms<---------mH
C <-Bus1<-Bus2<-Bus3<-Bus4<----R<----------L
51TERMLAINTERA            ZEROREZEROINDUCTAN
C .....^.....^.....^.....^.....^...........^  Zero Sequence
52TERMLBINTERB            POSIREPOSIINDUCTAN
C .....^.....^xxxxxxxxxxxx.....^...........^  Positive Sequence
53TERMLCINTERC
C .....^.....^xxxxxxxxxxxx.....^...........^
C
C <---Nodes--><---Refer--><-Ohms<---mH<---uF<-------------------------------Out
C Bus1->Bus2->Bus3->Bus4-><----R<----L<----C                                 V
  TERMLAINTERA            RDAMP1
  TERMLBINTERB            RDAMP1
  TERMLCINTERC            RDAMP1
C .....^.....^.....^.....^.....^.....^.....^xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx^
C
/SOURCE
C <--Bus<I<-----Ampl<-----Freq<----Phase<-------A1<------T1><---Tstart<----Tstop
14INTERA 1AMPLITUDES      60.0      0.0                    TSTARTTIMETSTOPTIMES
14INTERB 1AMPLITUDES      60.0   -120.0                    TSTARTTIMETSTOPTIMES
14INTERC 1AMPLITUDES      60.0    120.0                    TSTARTTIMETSTOPTIMES
C .....^.^.........^.........^.........^.........^........^.........^.........^
/ENDMODULE
$EOF **** END OF FILE ****
DATE AND TIME OF MODULE CONVERSION:    05/02/94  15.57.08
NUMBER OF ARGUMENTS DECLARED IN THIS MODULE:     9
NUMBER OF DUMMY VARIABLES DECLARED IN THIS MODULE:     3
```

Module 3PHEQUIV - Usage

```
BEGIN NEW DATA CASE
$PREFIX, C:\DOC\MODULE\INCFILE\
$SUFFIX, .INC
C
C <-------------------------------------------------Misc Data-----------------
C
C ----Dt<---Tmax<---Xopt<---Copt
  50E-06 100.E-3
C .....^.......^.......^.......^
C -Iprnt<--Iplot<-Idoubl<-Kssout<-Maxout<---Ipun<-Memsav<---Icat<-Nenerg
    5001       3       1       1       1       0       0       2       0
C .....^.......^.......^.......^.......^.......^.......^.......^.......^
C
C <-------------------------------------------------Circuit Data--------------
C
C Source Equivalent
C 2000 kVA, 6% Transformer, 480 V secondary (assume positive and zero seq equal)
C
C Usage: 3PHEQUIV TERML, R0, L0, R1, L1, VPEAK, -
C                 TSTART, TSTOP, RDAMP1 ; COM
C
$INCLUDE 3PHEQUIV 480VB, 0.001, 0.018, 0.001, 0.018, 391.92, -
                   -1.0, 9999.9, 10.0
C
C <-------------------------------------------------Nodal Output--------------
C
/OUTPUT
C
C BUS-->BUS-->BUS-->
  480VBA480VBB480VBC
C .....^.....^.....^
BLANK ENDS BRANCH
BLANK ENDS SWITCH
BLANK ENDS SOURCE
BLANK ENDS OUTPUT
BLANK ENDS PLOT
BLANK ENDS CASE
```

Module Example #2 - 6PDIODE

- Three-phase, six-pulse diode bridge **- 6PDIODE**



System Example #1

- System example illustrating usage of modules 3PHEQUIV and 6PDIODE: