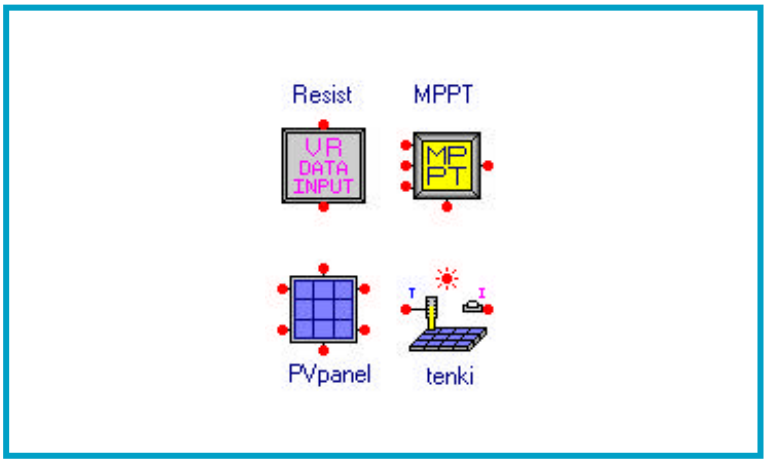
ATP における新素子作成法

― MODELS による素子の作成手順と作成例 ―

ATPDraw for Windows version 2.1 Manual



作成 ：山下　博史　（神戸市立工業高等専門学校　電気電子工学専攻）

Manual ’ s version 1.0

Last Upgrade ：2000/11/10 (Fri)

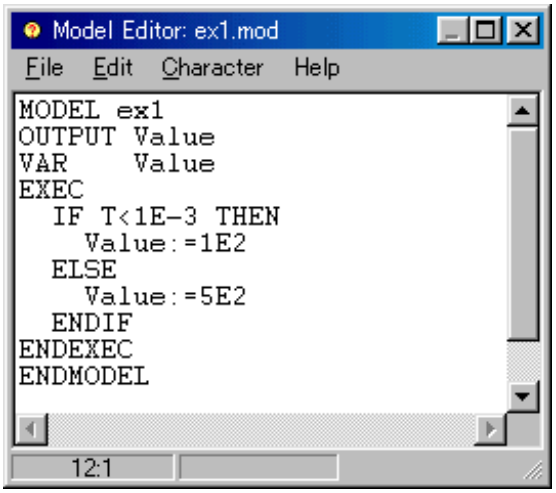
Reference ：ATPDraw for Windows version 2.1 Help (keyword ：MODELS)

例１ 単独電源の例

ATP 上で素子を作るためには、ATPDraw 上で使うアイコンである *.sup ファイルと、FORTRAN 言語に似た言語で書かれている命令のためのファイルである *.mod ファイル （モデルファイル）が必要となる。

① *.mod ファイルの作成

FORTRAN 言語に似た自由度の高い言語で書かれたこのモデルファイルは、ATPDrawの外で作成する。*.mod ファイルは次のように作成される。 作成においてはメモ帳などを使って作成しても良い。



例として、電源の出力を時間 T において、T<1E-3[s]時には出力を1E2=100V、それ以外の時は5E+2=500V を出力する素子を作成する。 プログラム中の

OUTPUT ：素子から出力される変数として定義する。

VAR ：プログラム中の変数を定義する。

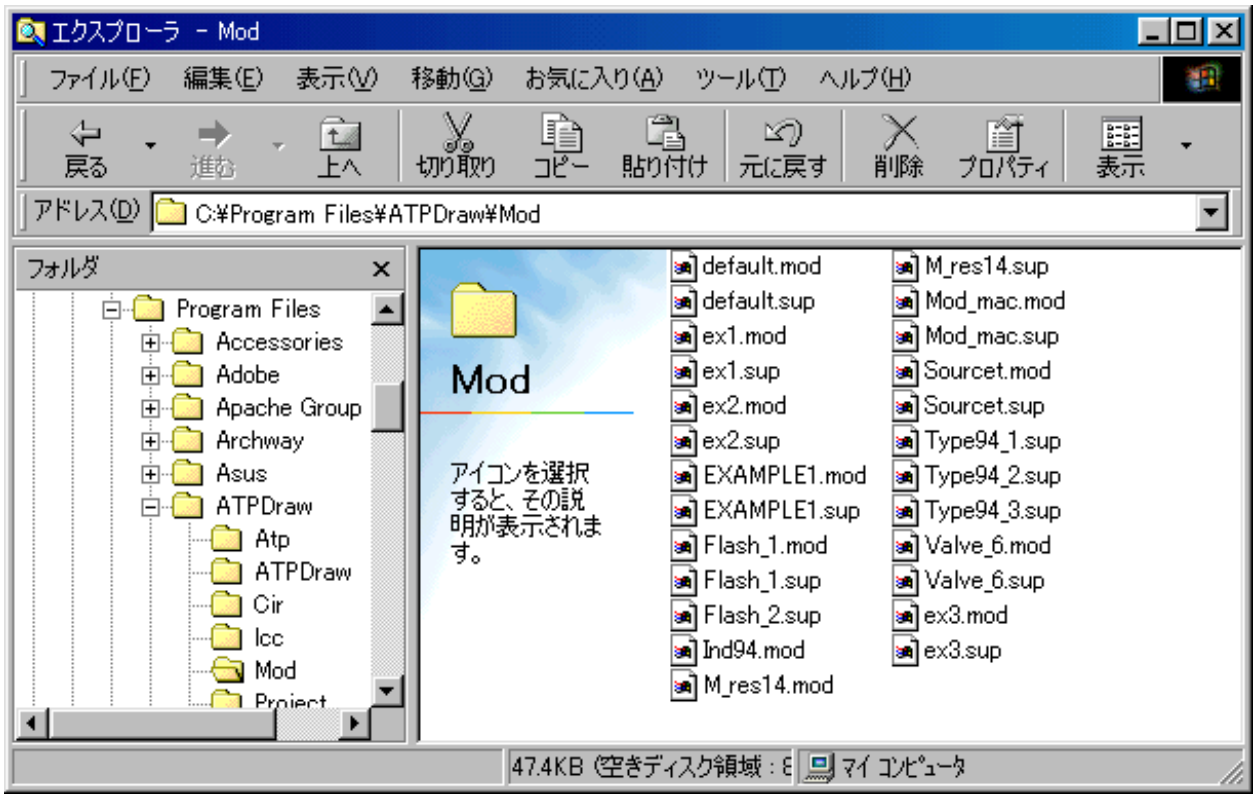
EXEC ：実際に計算などを実行する。 この中にプログラムを作成する。

IF に対して ENDIF、 EXEC に対して ENDEXEC 書かなければならない。model の名前として *.mod のファイル名と同じ名前をこのプログラムの名前として定義しておかなければいけない。 この例ではファイル名は ex1.mod とし、 プログラム中 model 名も ex1 にする。 そして最後に ENDMODEL を書けばファイルは完成する。 ファイル名の長さは 8文字以内で決めなければいけなく、さらに数字のみのファイル名やファイル名の先頭に数字を使うと ATP が動作しなくなるので注意する。

②ファイルの保存

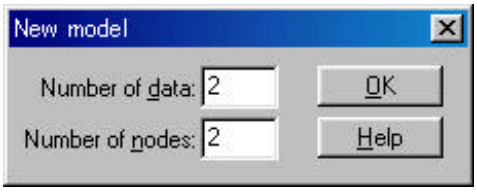
①で作成した *.mod ファイルを保存する。 保存する場所は図のように、

C:\Program Files\ATPDraw\Mod の場所に保存すればよい。この場所に置かなくても使用できるが、ATPDraw で使うときに便利なようにするためこの場所に設定した。



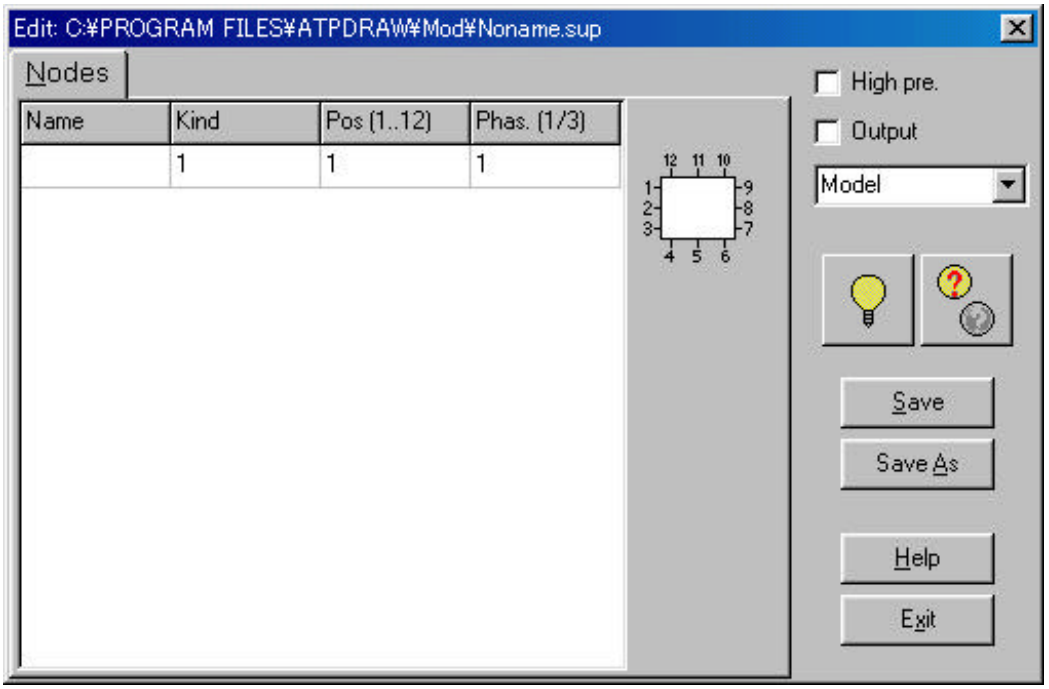
③ *.sup ファイルの作成

次に ATPDraw 上で使う素子を決定する。 まず始めに ATPDraw で、 Objects → Model → New sup-file を選ぶと次のような画面が出てくる。



ここで、作成する素子の形を決定する。 Number of data はデータの数、すなわちATPDraw 上で決定できるパラメータの数となるNumber of nodes は端子の数の総計である。 端子の数には入力 ・出力すべてを含めた物となる。 今の例においては Number of data は 0、Number of nodes は 1とする。

値を決定すると、 次のような画面となる。

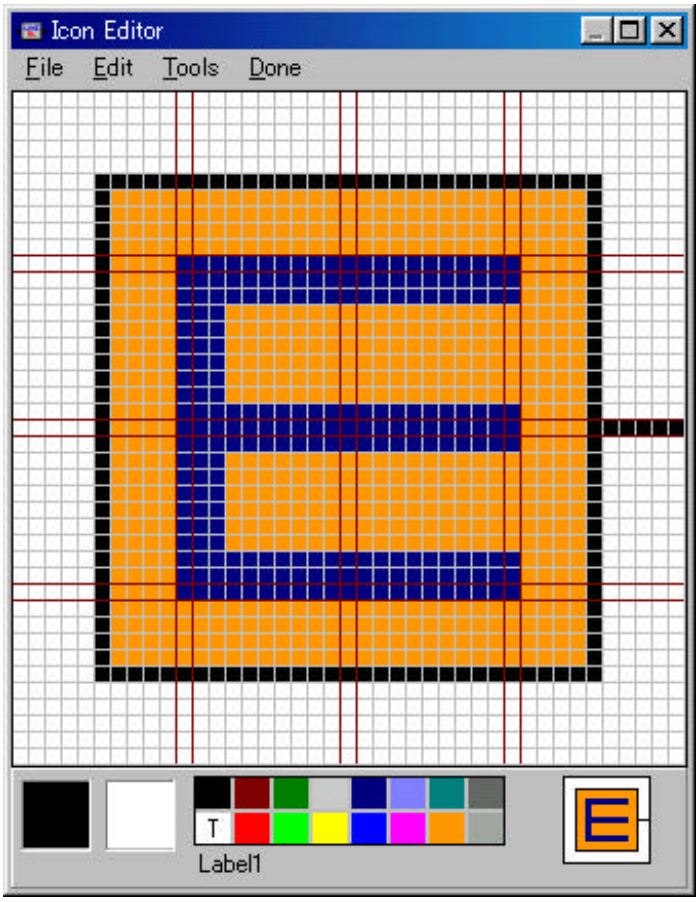


ここで、それぞれのパラメータ、 端子の出力点 ・性質を決定する。 この例では端子の名前 Name をValue とする。 Kind は端子の性質を決定するもので、 それは次のように決められる。

Kind	MODELS	Description	
0:		output	出力
1:	(i)	current input	入力 電流）
2:	(v)	voltage input	入力 電圧）
3:	(switch)	switch status input	
4:	(mach)	machine variable input	
5:	(tacs)	tacs variable input	TACS での入力
6:	(imssv)	imaginary part of complex steady-state node voltage	
7:	(imssi)	imaginary part of complex steady-state switch current	

この例では kind に 0を選ぶ。Pos は端子のポジションを決定する物で、その配置は図のようになっている。また Phas. はその入力されたものの相を示すもので、 ここでは単相を選ぶので 1とする。

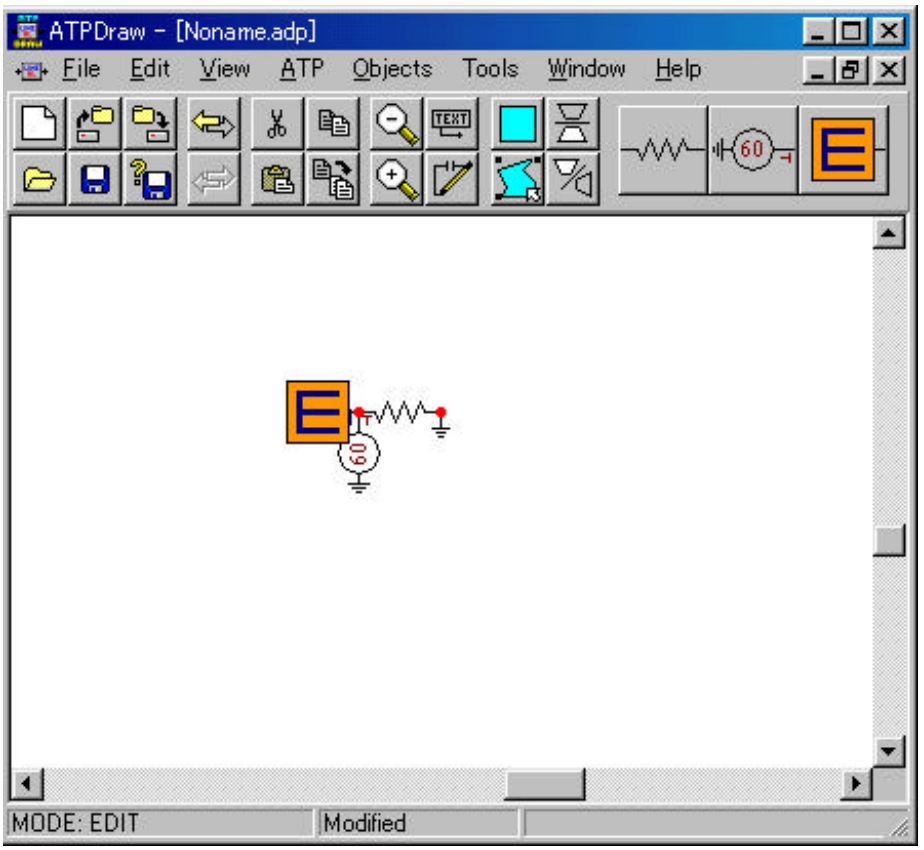
図中のここをクリックすると、 新しく作られた素子の絵柄を決定できる。 今参考として次のような絵柄とする。



Icon Editor を使って、作りたい素子の絵柄を決定する。 絵柄が書ければ Done をクリックするとその絵柄が更新される。以上によって新しい素子となる*.sup ファイルが作成できた。 よってこのファイルを保存する。 ここで注意することは、*.mod ファイルを作成したときのファイル名と同じにしなければならない。

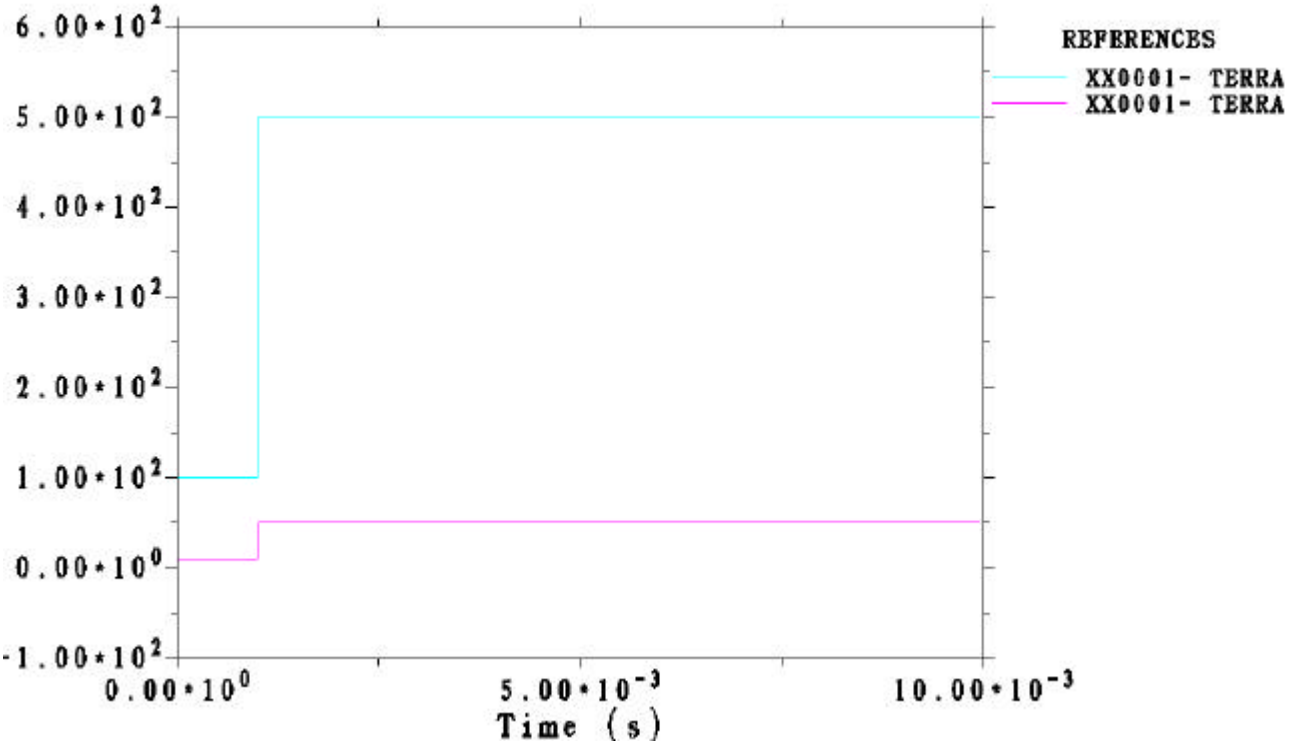
④実際の使い方

実際に作成した素子を使うには、 マウスを右クリック→ MODELS を選び、 作成した *.sup ファイル (この例の場合は ex1.sup）を選ぶ。 するとATPDraw 上に現れる。 次に、 実際に素子を駆動するためにはTacsSource が必要なので、 選び出してきて接続する。出力側には抵抗 (10Ω）を接続し、 抵抗における電圧・電流を見た。

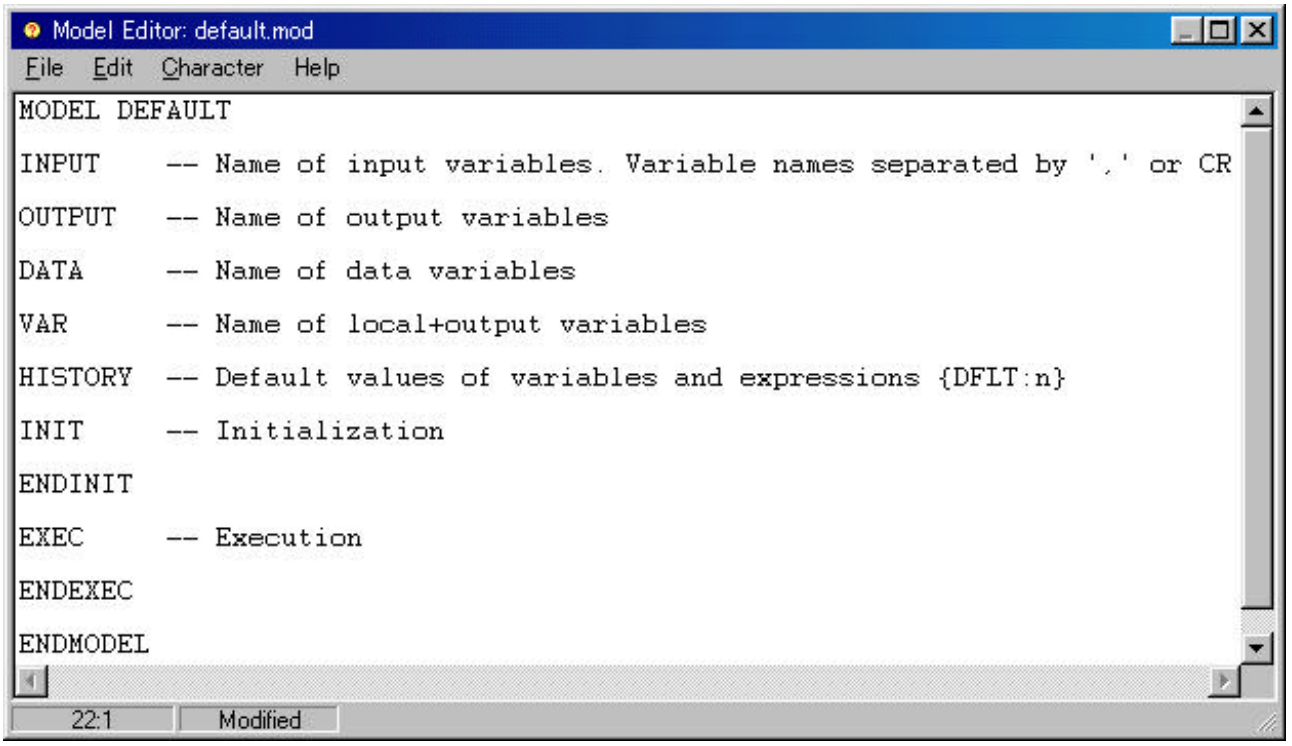


⑤結果

抵抗は 10Ωで時間に応じて 100V と500V が加わるのだから（E-3 秒で変更）、 抵抗には 10A と50A が流れることになる。実行の結果より、抵抗の電流、電圧は次のように得られた。図の水色の線は電圧値を示し、 ピンク色の線は電流を示す。



.mod ファイルの作成としてテキストから作成したが、.mod ファイルを作る方法として、次の方法でも良い。まず Objects → Model → New mod-file を選ぶと、 次のようなファイルが出てくる。



これより、 FORTRAN 言語によってプログラムを作成する。
図よりこの手順で書く。

MODEL DEFAULT ：DEFAULT のところにファイル名を書く。 *.sup のファイル名と同じになるように設定。save したときの名前が後で書き換えられるので、ここで設定しなくてもファイル名を決めるときに設定しても良い。

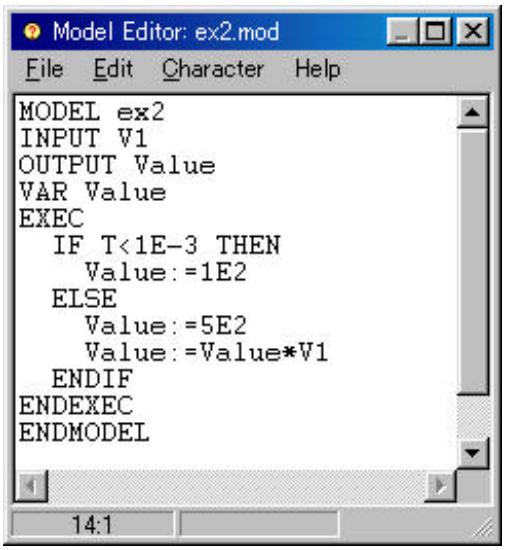
INPUT ：入力端子の変数名を設定
OUTPUT ：出力端子の変数名を設定
DATA ：ATPDraw 上で変化できるパラメータを決定
VAR ：プログラム中で使う変数を設定
HISTORY ：変数や式などの初期状態などを決定
INIT ：変数の初期値を設定
ENDINIT ：初期値設定の最後
EXEC ；実際のプログラムを書く場所。 ここに FORTRAN でプログラムを作成する。
ENDEXEC ：実行プログラムの最後

ENDMODEL ：モデルファイルの最後

プログラムが書けたら、 前の方法と同じように *.mod ファイルにファイル名を書いて保存する。

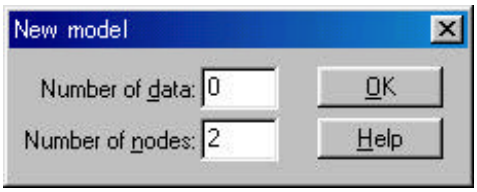
例２ 外部入力端子を持つ電源の例

①*.mod ファイルの作成

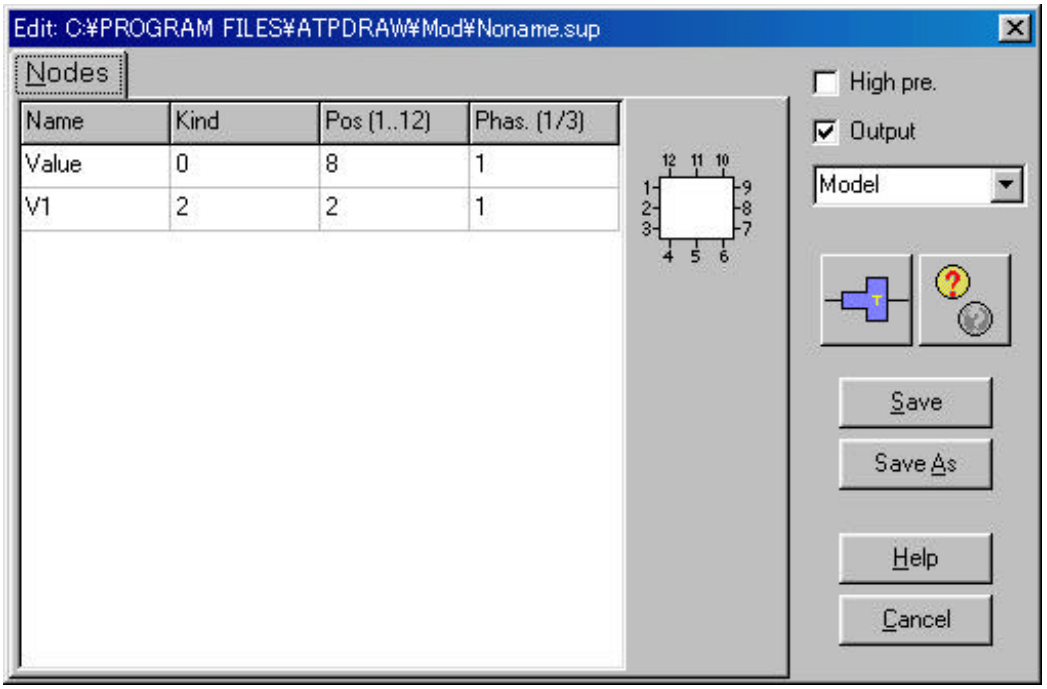


例として、電源の出力を時間Ｔにおいて、T<1E-3秒時には出力を1E2=100V、それ以外の時は5E+2=500V　500 × V1=1000V を出力する素子を作成する。 V1は入力端子である。
プログラム中の
INPUT ：素子に入力される変数として定義する。
OUTPUT ：素子から出力される変数として定義する。
VAR ：プログラム中の変数を定義する。
EXEC ：実際に計算などを実行する。この中に計算させるプログラムを作成する。

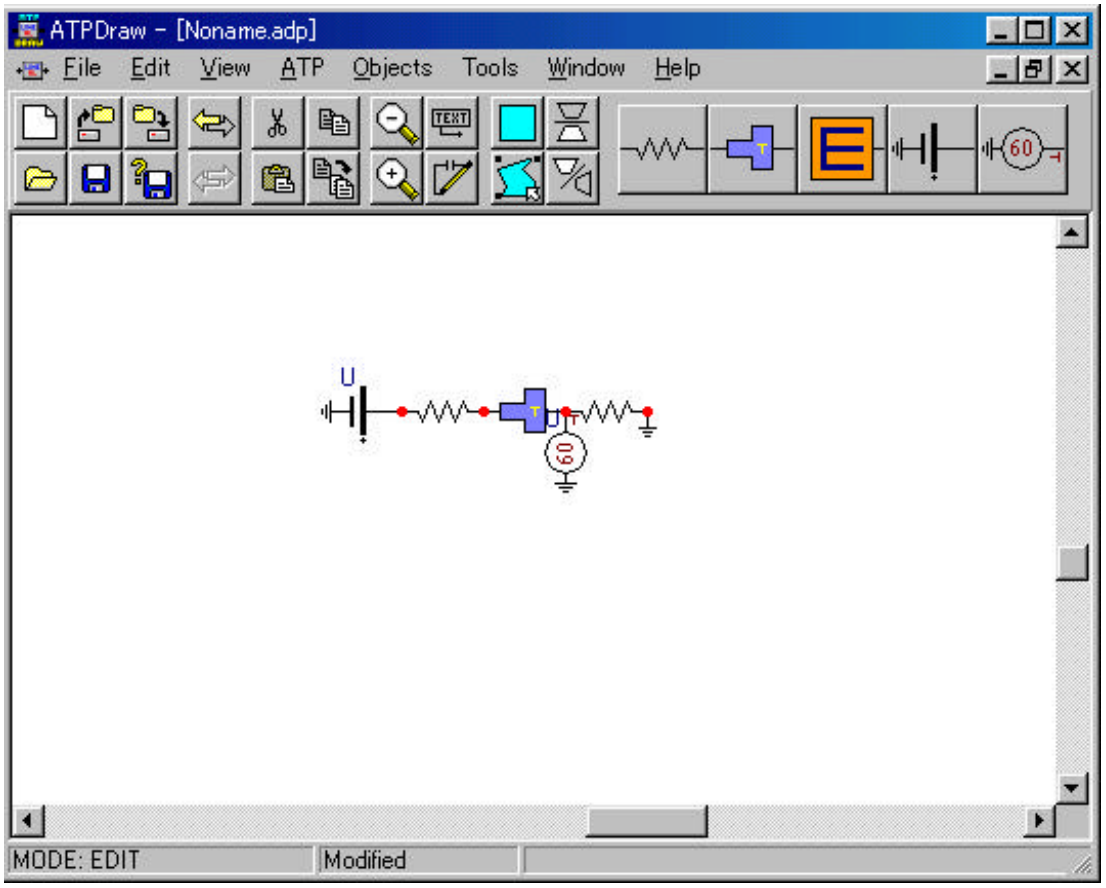
②*.sup ファイルの作成
前の例と同じように、パラメータなどを決定していく。



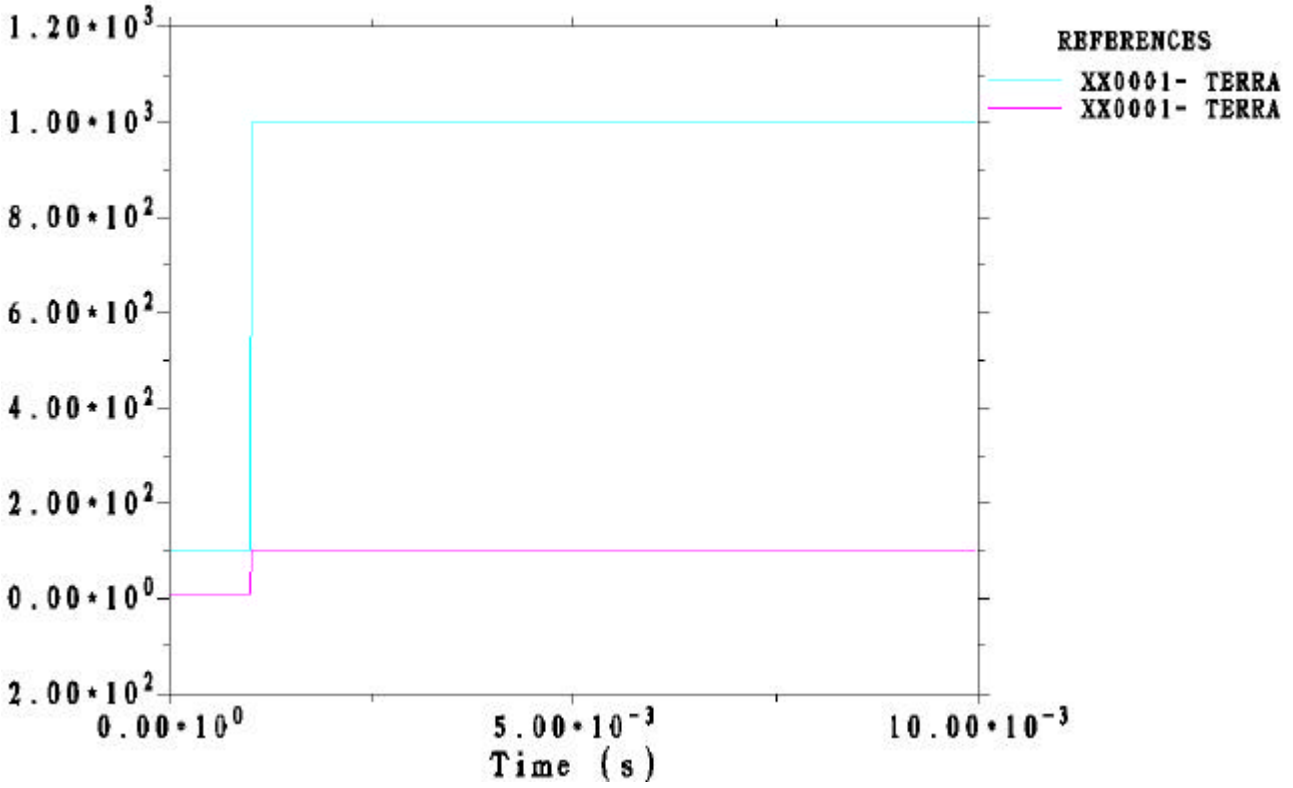
ここで、作成する素子の形を決定する。今の例においては Number of data は0、Number of nodes は入力と出力の端子がそれぞれ1つづつなので２となる。



それぞれのパラメータを決定すると次のようになる。出力端子 Value は Kind は０　Pos は８とし、入力端子は Kind は２　電圧の入力とする)、Pos は 2とし、各相は単相として Phas. は 1と決定する。また素子の絵柄も自作する。それが完了したらファイルを保存し、*.mod のファイルと同じファイル名で保存する。この例では ex2.sup とした。

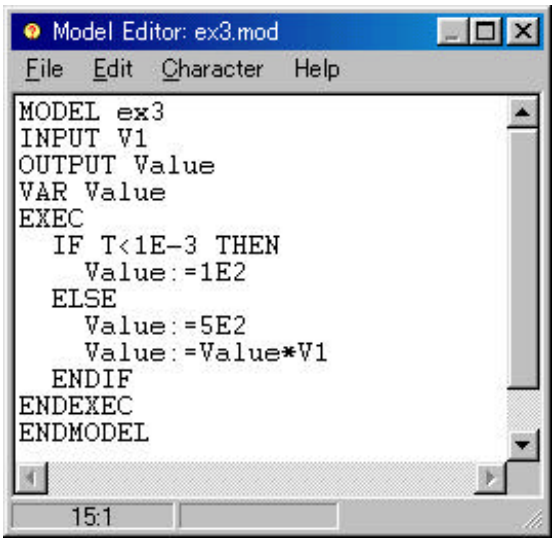


回路中で、 入力電圧は V1= 2Vとし、 抵抗は入力側、 出力側ともに10Ωとした。 ここで入力側の電源に抵抗を接続しなければ計算できないことがわかった。 計算結果は時間によって100V、 1000V を出すようになっている事がわかる。 また電流は時間によって10A、100A 出力していることがわかる。グラフ中で水色の線は電圧を示し、 ピンク色の線は電流を示す。



例３ TACS 電源を使用して入力する電源の例

① *.mod ファイルの作成



例として、例 2と同じように、電源の出力を時間 T において、T<1E-3 秒時には出力を1E2=100V、それ以外の時は 5E+2=500V 500 × V1=1000V を出力する素子を作成する。V1 は入力端子である。V1 には TACS から電源を供給する。プログラム中の

INPUT ：素子に入力される変数として定義する。

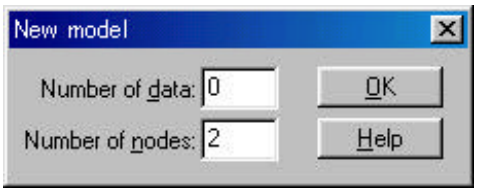
OUTPUT ：素子から出力される変数として定義する。

VAR ：プログラム中の変数を定義する。

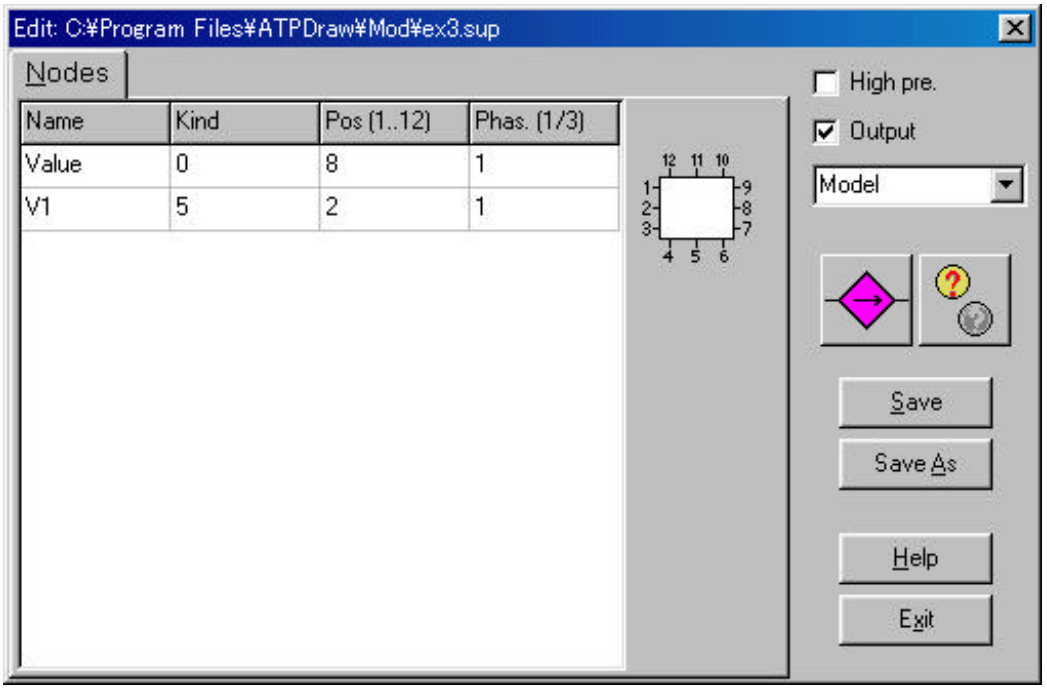
EXEC ：実際に計算などを実行する。この中にプログラムを作成する。

② *.sup ファイルの作成

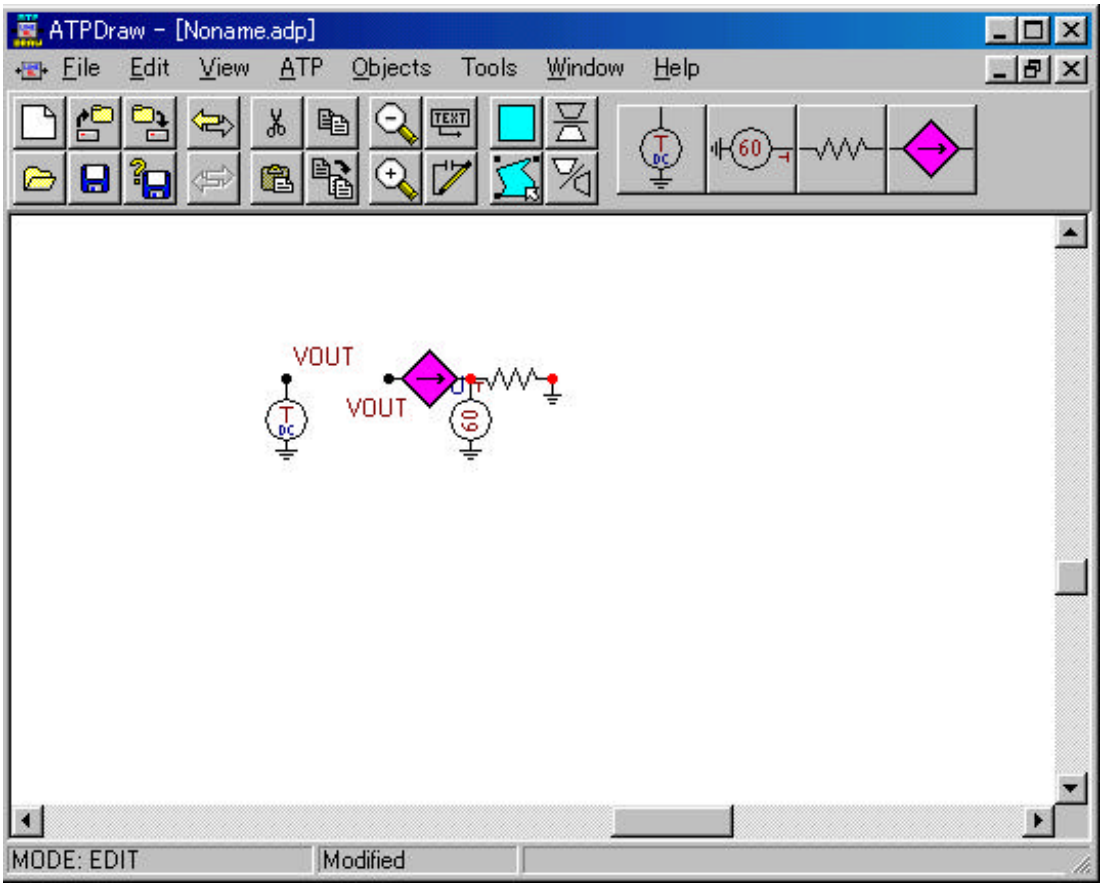
前の例と同じように、パラメータなどを決定していく。



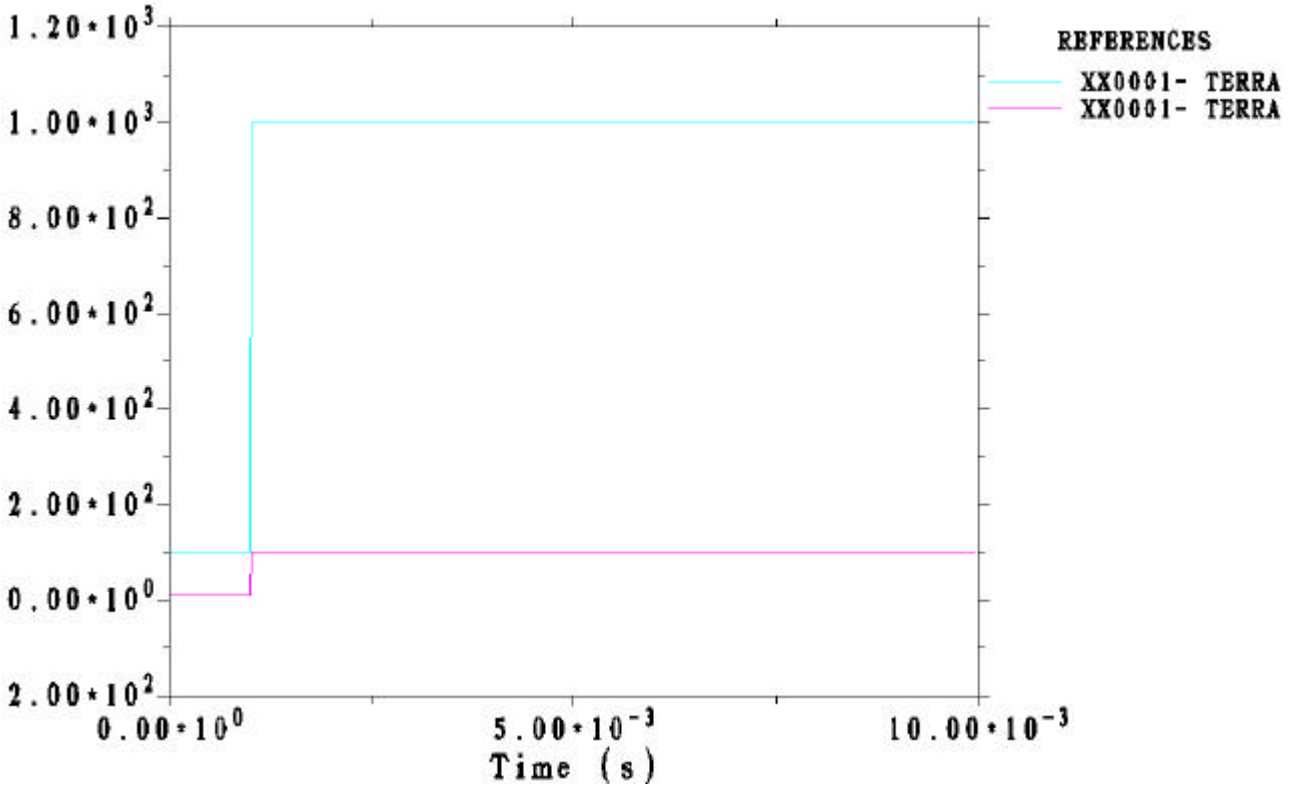
ここで、作成する素子の形を決定する。今の例においては例 2の時と同じになるので、Number of data は 0、Number of nodes は入力と出力の端子がそれぞれ 1つづつなので 2となる。



それぞれのパラメータを決定すると次のようになる。出力端子 Value は Kind は 0、Pos は 8とし、入力端子は Kind は 5 (TACS の入力とする)、Pos は 2とし、各相は単相として Phas. は 1と決定する。また素子の絵柄も自作する。それが完了したらファイルを保存し、*.modのファイルと同じファイル名で保存する。この例においてファイルの名前は ex3.sup とした。

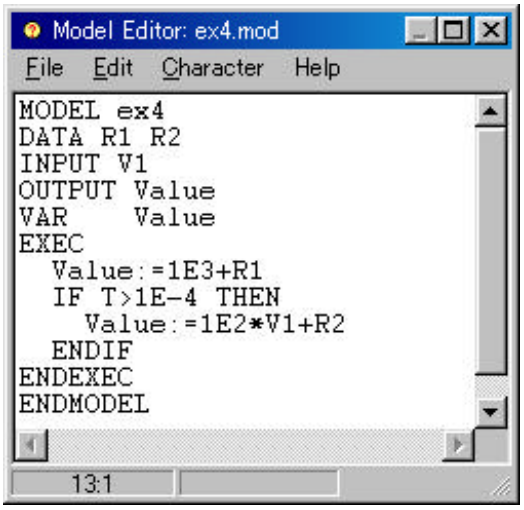


回路中で、入力電圧はVOUT= 2Vとし、抵抗は入力側、出力側ともに１０Ωとした。ここで入力側の電源に抵抗を接続しなければ計算できないことがわかった。計算結果は時間によって100V、1000Vを出すようになっている事がわかる。また電流は10A、100Aを出力している。ここでTACSの入力を素子の入力端子に入れるためには、TACSの出力端子と新しい素子の端子の名前が同じとなるようにすればよい。atpファイルを作るときに”Same name on different nodes “** ” ”（** :端子の名前）という忠告が出るが、これは2つの端子の名前を同じ名前としたためであるが、そのまま実行すると計算できる。よって計算結果を示すと次のようになる。



例４ TACS 電源を使用して入力し、 パラメータを変更できる電源の例

① *.mod ファイルの作成



例として、電源の出力を時間Ｔにおいて、T<1E-3秒時には出力を1E2+R1=100+R1、それ以外の時は1E+2×V1+R2=100×V1+R2を出力する素子を作成する。V1は入力端子である。V1にTACSから電源を供給する。 R1 とR2 は ATPDraw 上で変更できるパラメータである。 プログラム中の

INPUT ：素子に入力される変数として定義する。

OUTPUT ：素子から出力される変数として定義する。

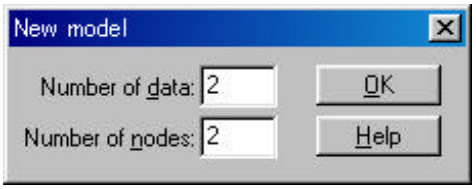
ＤＡＴＡ ：パラメータの変数を定義する。 ここで定義した変数が ATPDraw 上で変更できる。

VAR ：プログラム中の変数を定義する。

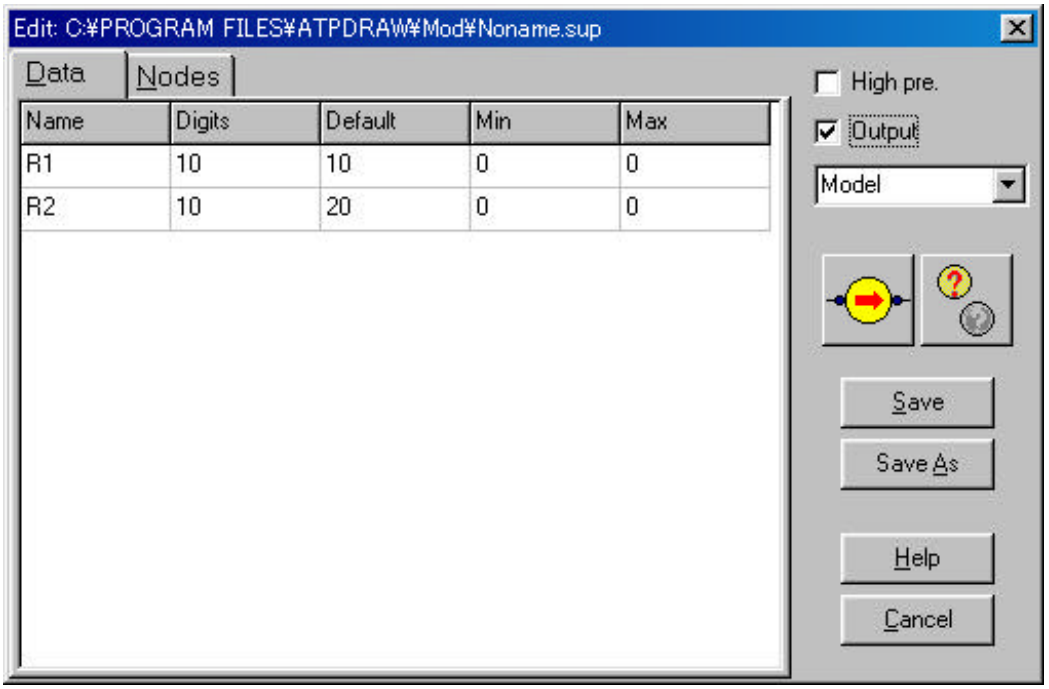
EXEC ：実際に計算などを実行する。 この中にプログラムを作成する。

② *.sup ファイルの作成

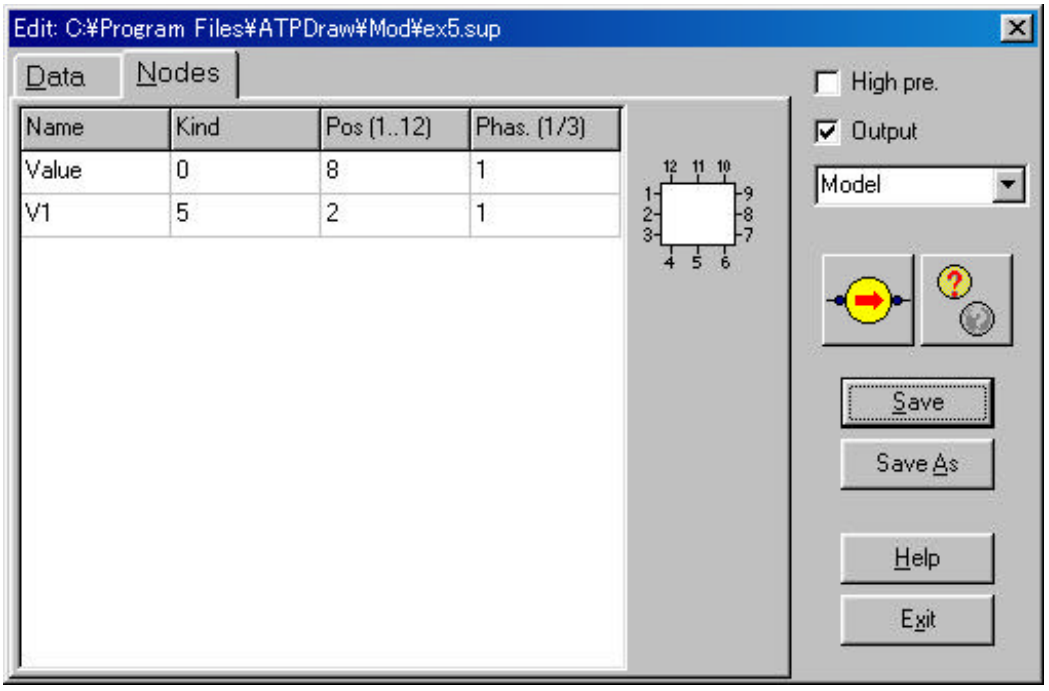
前の例と同じように、 パラメータなどを決定していく。



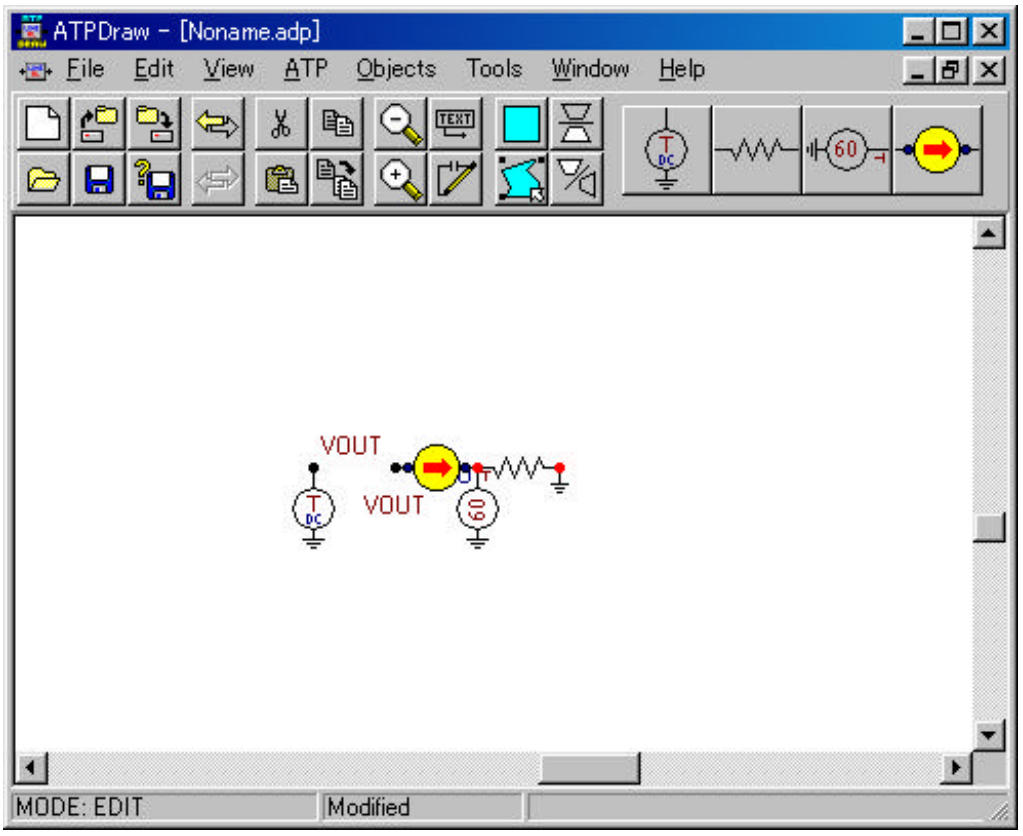
ここで、作成する素子の形を決定する。今の例においては、パラメータとして２つ定義するので、Number of data は２、 Number of nodesは入力と出力の端子がそれぞれ1つづつなので２となる。



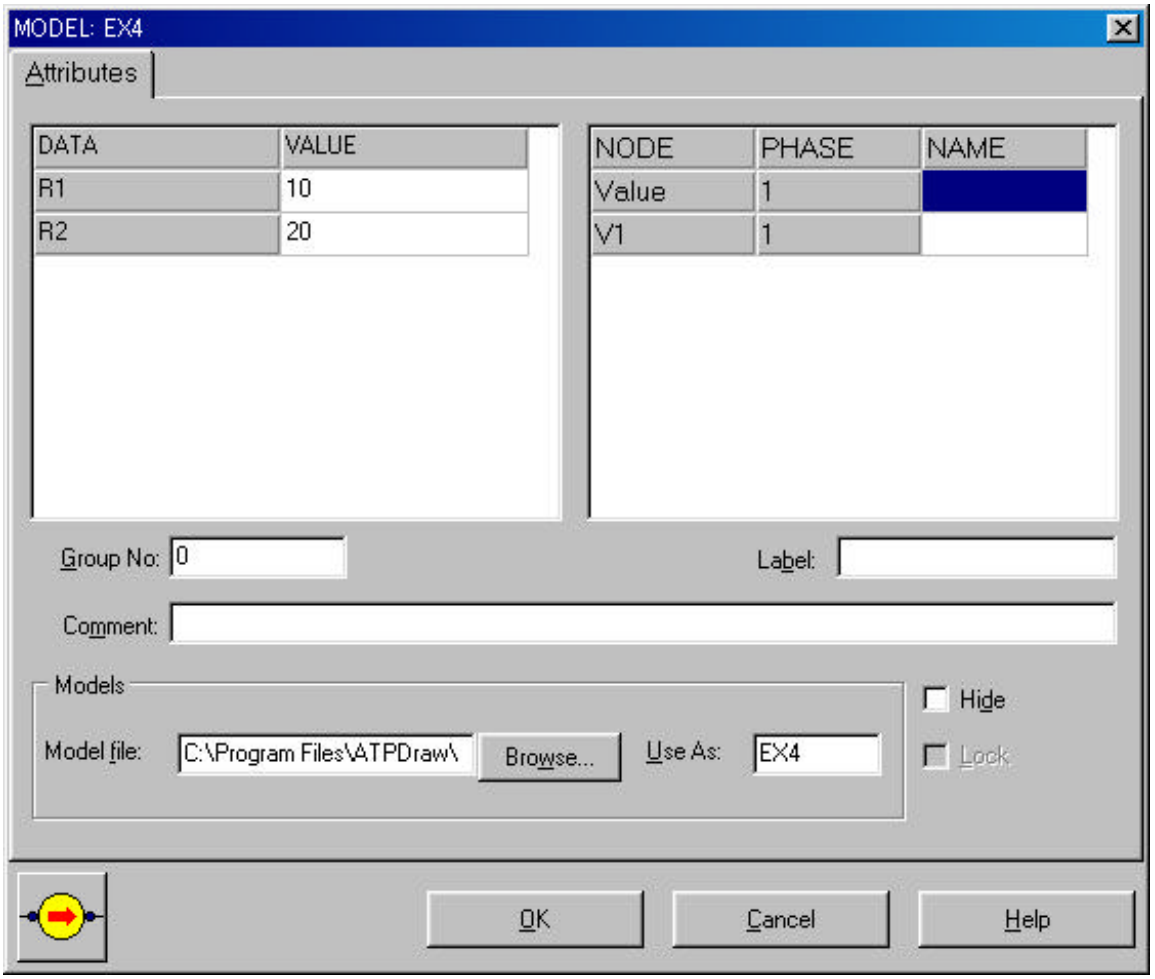
まず、 Data の決定を行う。 Name は ATPDraw 上で使うパラメータとなる。 この例では R1 とR2 の 2つを定義する。 Digits はダイアログボックス中の開コンポーネントにおいて取りうる最大のビット数、 Default はその変数が持つ初期値を決定する。 Min とMax は時間などの変数の場合において、 その値が出力される時間の最初と最後を設定する。 ある値だけを入力する場合には Min とMax は同じで0にする。



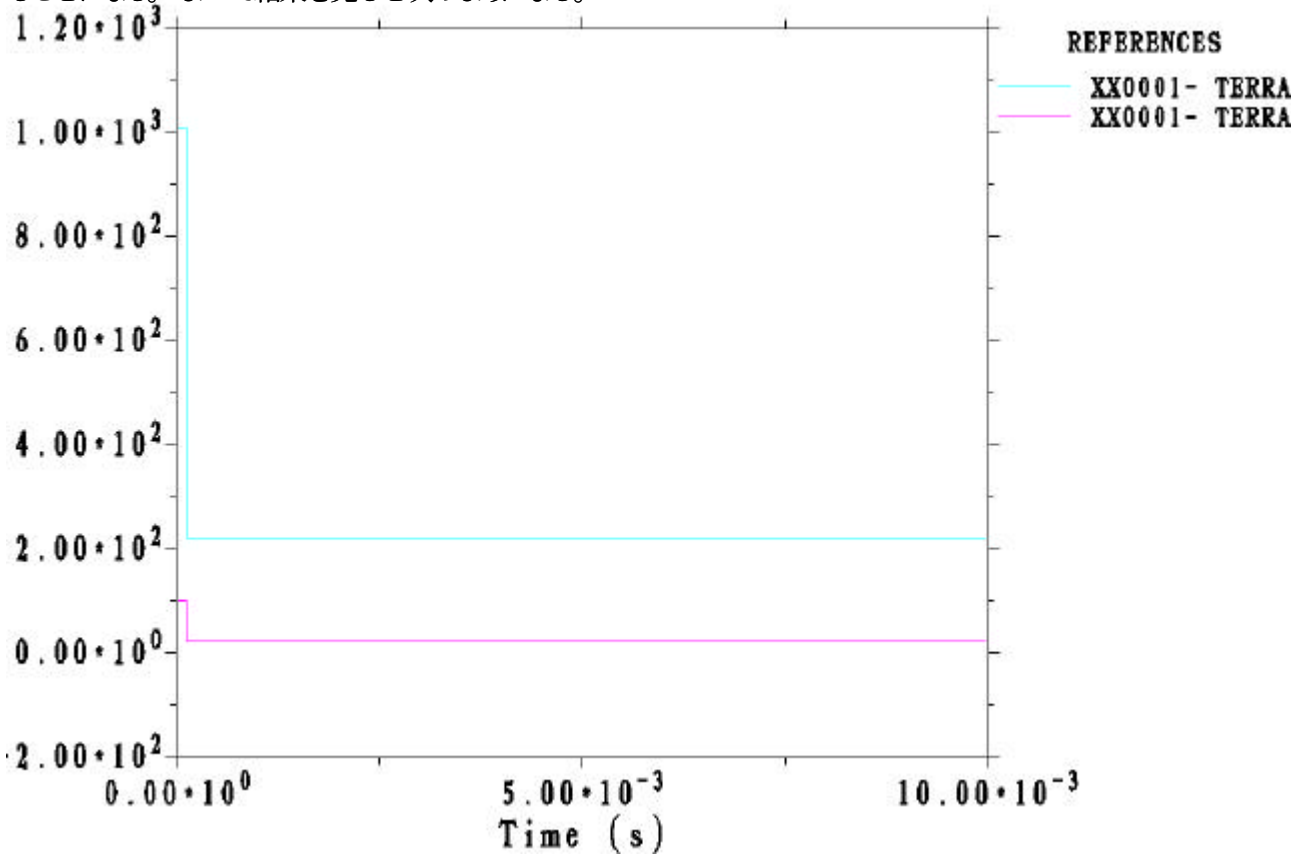
次に Nodes のパラメータを決定すると図のようになる。出力端子 Value は Kind は 0、Pos は 8とし、入力端子 V1 は Kind は 5 (TACS の入力とする)、Pos は 2とし、各相は単相として Phas. は 1と決定する。 また素子の絵柄も自作する。 それが完了したらファイルを保存し、 *.mod のファイルと同じファイル名で保存する。 この例ではファイル名をex4.sup とした。



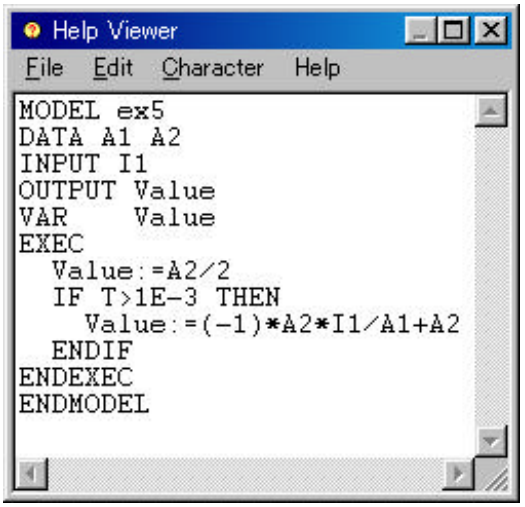
回路は例 3の場合と同じとなる。 ここで新しく作った素子について見てみると、



R1 とR2 のパラメータを次のように決定する。R1 を10、R2 を20 とすると、出力は1E3+R1=1000+10=1010V、電流は1010 ÷ 10=101A、また1E-4秒からは 1E+2 × V1+R2=100 × 2+20=220V、電流は220 ÷ 10=22A 発生することになる。よって結果を見ると次のようになる。

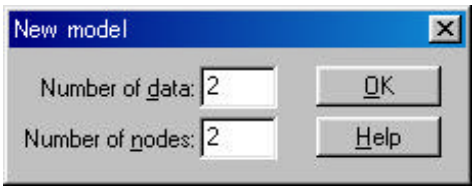


例５ 分散電源の例　線形の特性を持つ分散電源の例）
①*.mod ファイルの作成

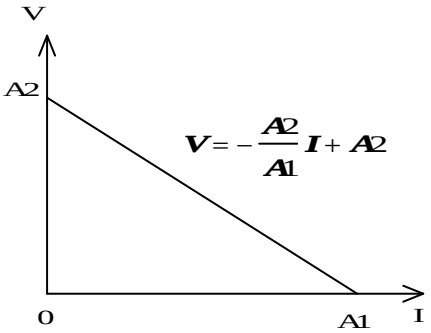


例として、電源の出力を時間Tにおいて、T<1E-3秒時には出力をR2/2、それ以外の時は電源の特性から決まる式から導出する。V1は入力端子である。 I1に素子から取り出した出力電流を素子に供給する。 A1とA2はATPDraw上で変更できるパラメータである。 プログラム中の
INPUT ：素子に入力される変数として定義する。
OUTPUT ：素子から出力される変数として定義する。
DATA ：パラメータの変数を定義する。 ここで定義した変数がATPDraw上で変更できる。
VAR ：プログラム中の変数を定義する。
EXEC ：実際に計算などを実行する。 この中にプログラムを作成する。

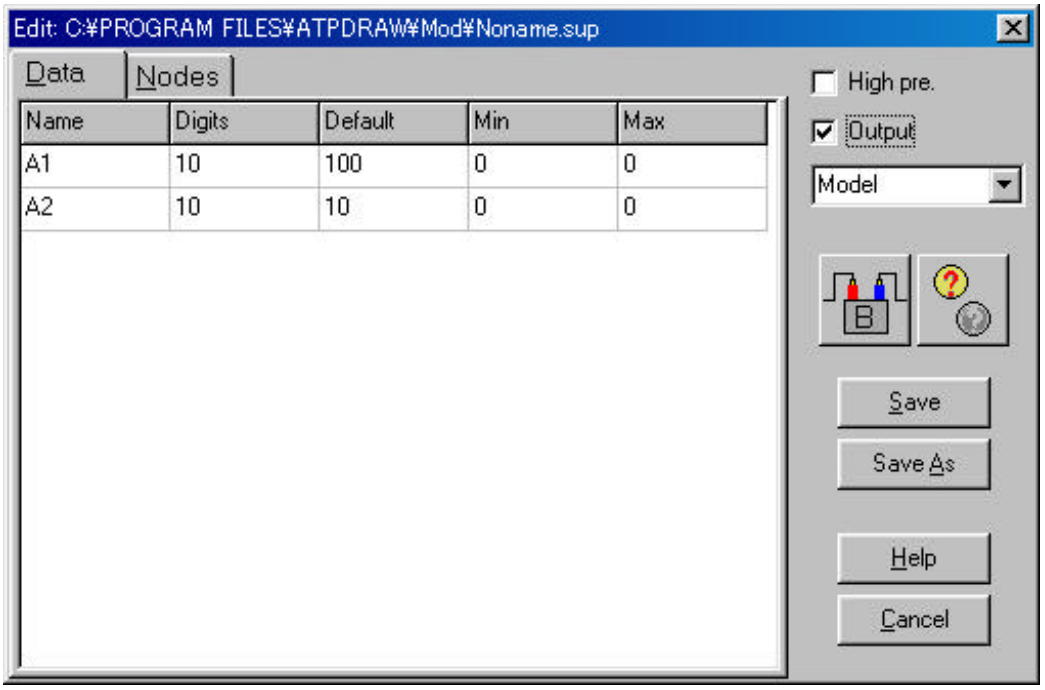
②*.sup ファイルの作成
前の例と同じように、 パラメータなどを決定していく。

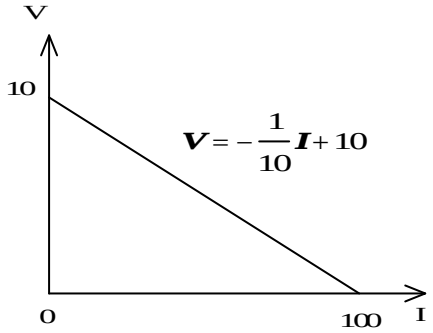


ここで、作成する素子の形を決定する。今の例においては、パラメータとして２つ定義するので、Number of dataは２、Number of nodesは入力電流と出力電圧の端子がそれぞれ1つづつなので２となる。

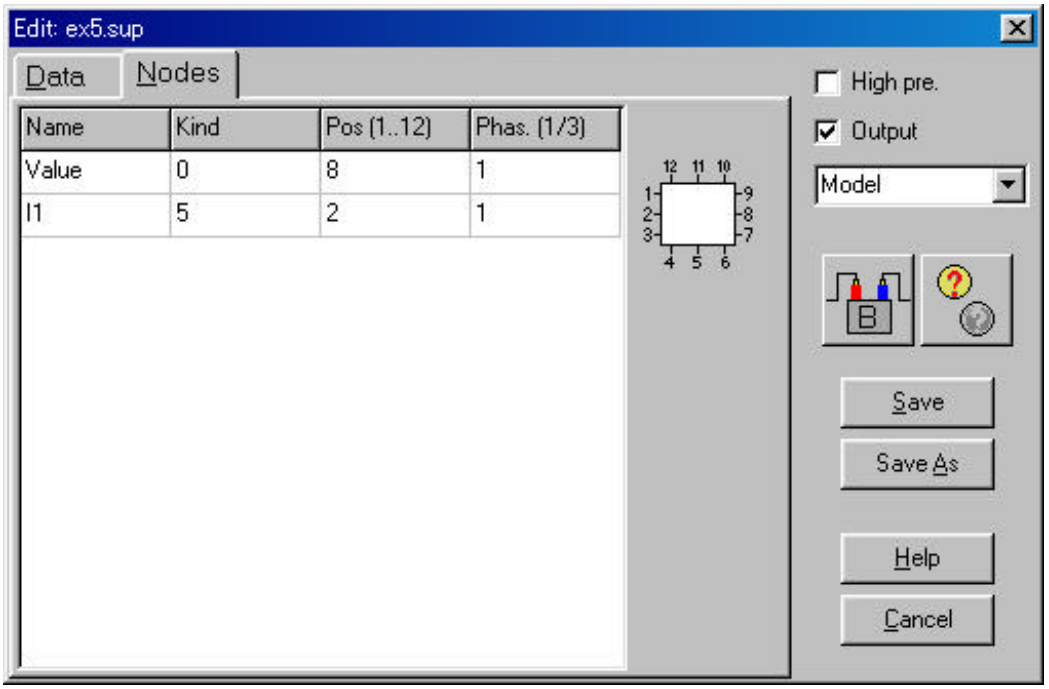


作成する電源の特性としては、図のように決定する （これは例として取り上げた特性であるので、 実際のVI方程式では横軸が電圧V、縦軸が電流Iとし、方程式をV=で解いたものを使う）。A1とA2が決まればその方程式より求められる特性が得られる。この素子について最初はR2/2の初期値を与え、 その後方程式から電圧値を決定し、 負荷抵抗の影響を持つ電源の特性を解析していく。

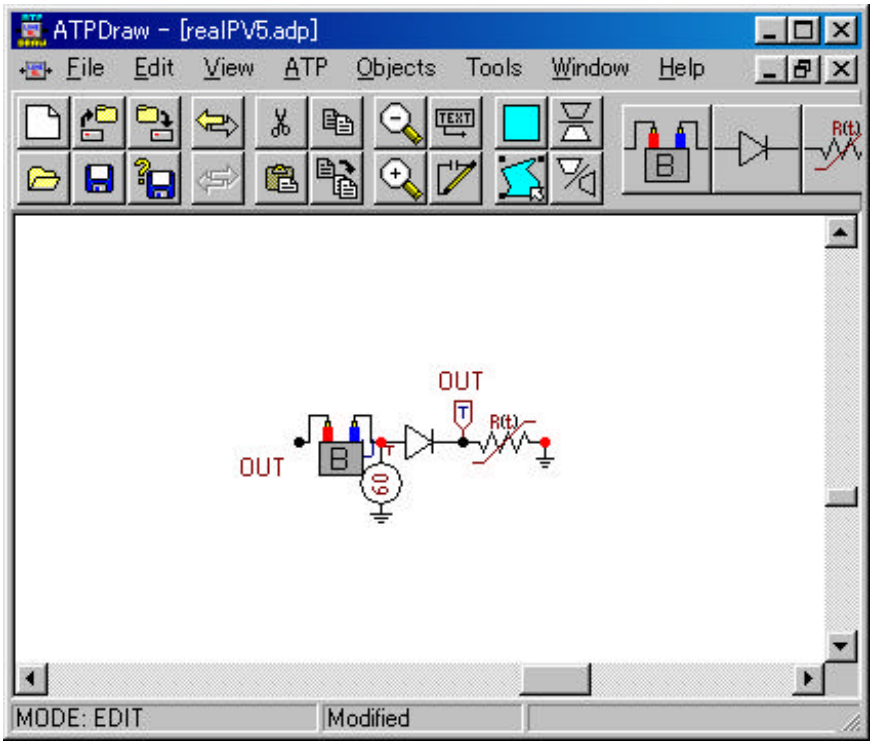




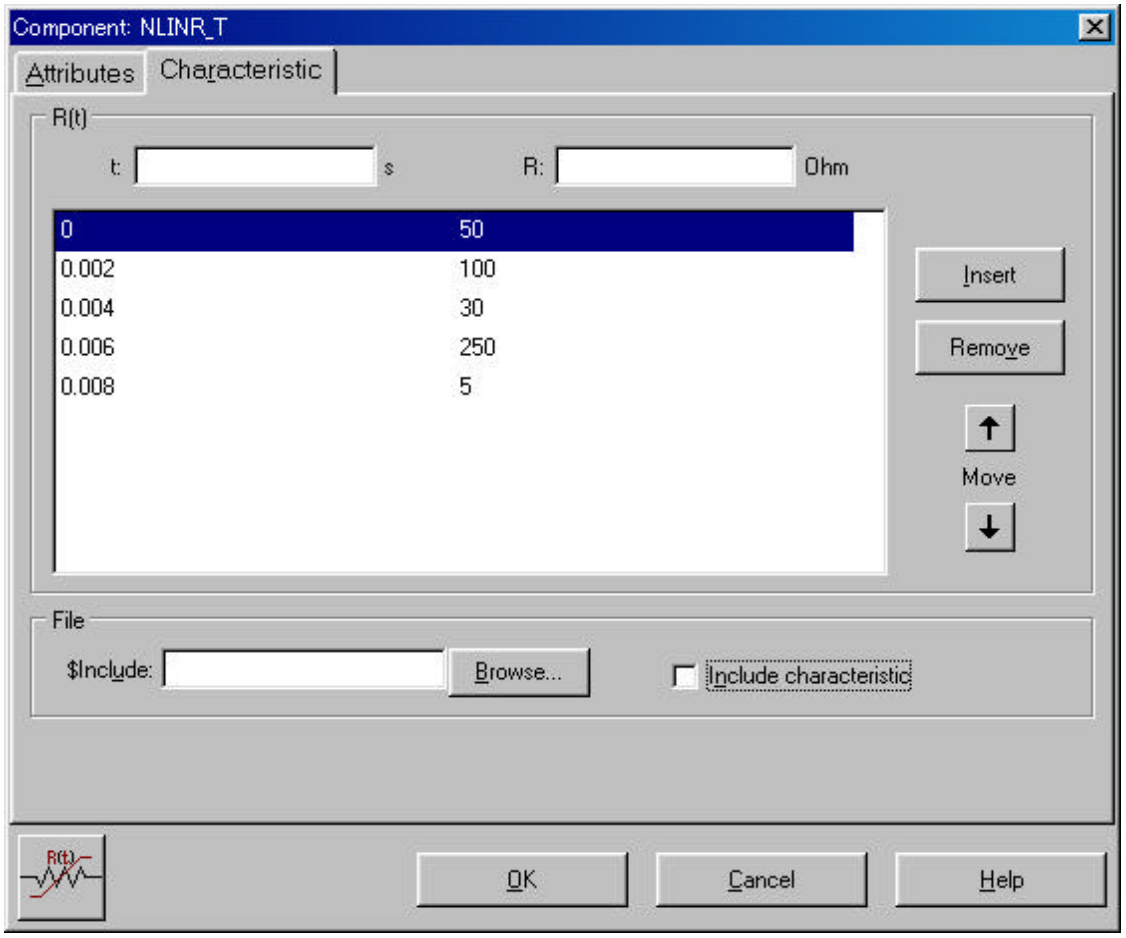
まず、Data の決定を行う。Name は ATPDraw 上で使うパラメータとなる。この例では A1 と A2 の 2 つを定義する。Digits はダイアログボックス中の開コンポーネントにおいて取りうる最大のビット数。Default はその変数が持つ初期値を決定する。Min と Max は時間などの場合において、その値が出力される時間の最初と最後を設定する。今の場合下の特性のような分散電源を作りたいと思うので、A1=100、A2=10 とした。



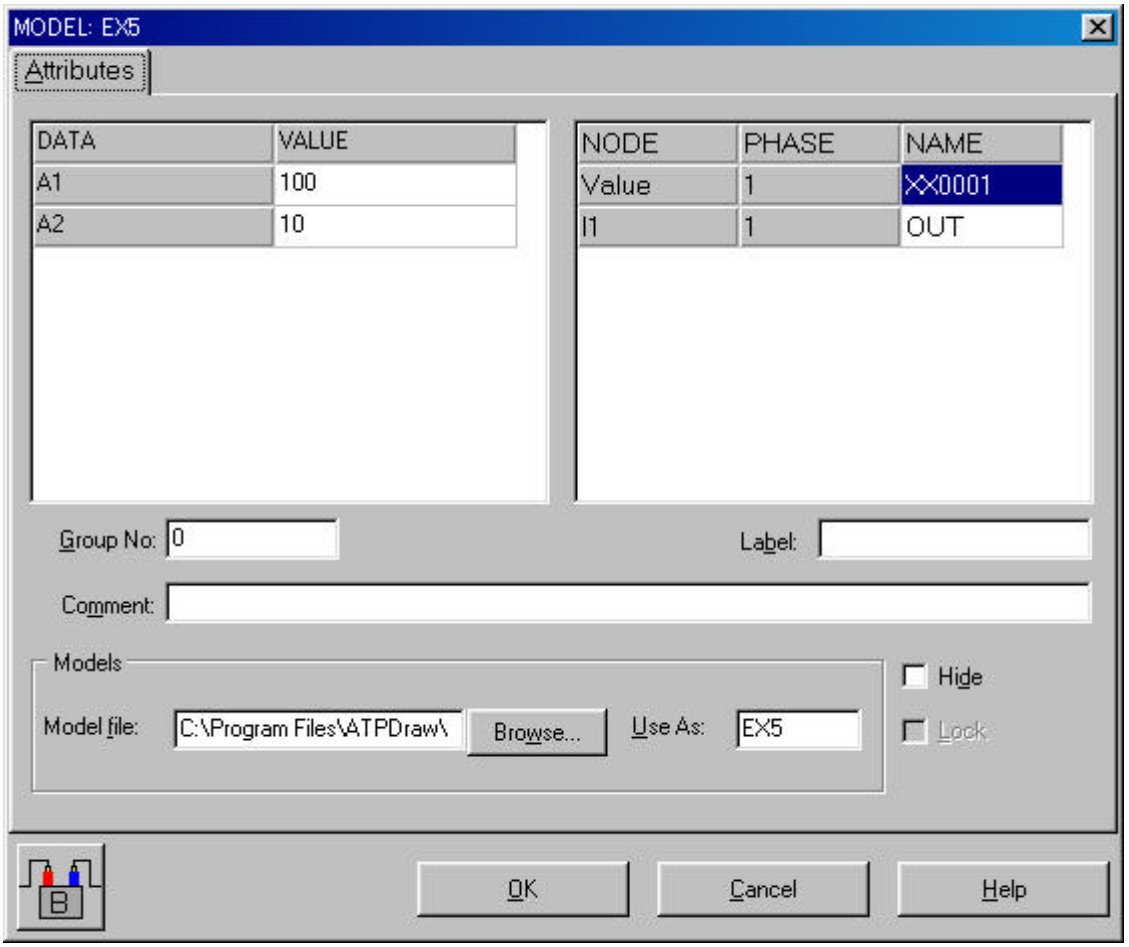
Nodes のパラメータを決定すると次のようになる。出力端子 Value は Kind は 0、Pos は 8とし、入力端子 I1 は Kind は 5（TACS の入力とする）、Pos は 2とし、各相は単相として Phas. は 1と決定する。また素子の絵柄も自作する。それが完了したらファイルを保存し、*.mod のファイルと同じファイル名で保存する。この例でファイル名は ex5.sup とした。



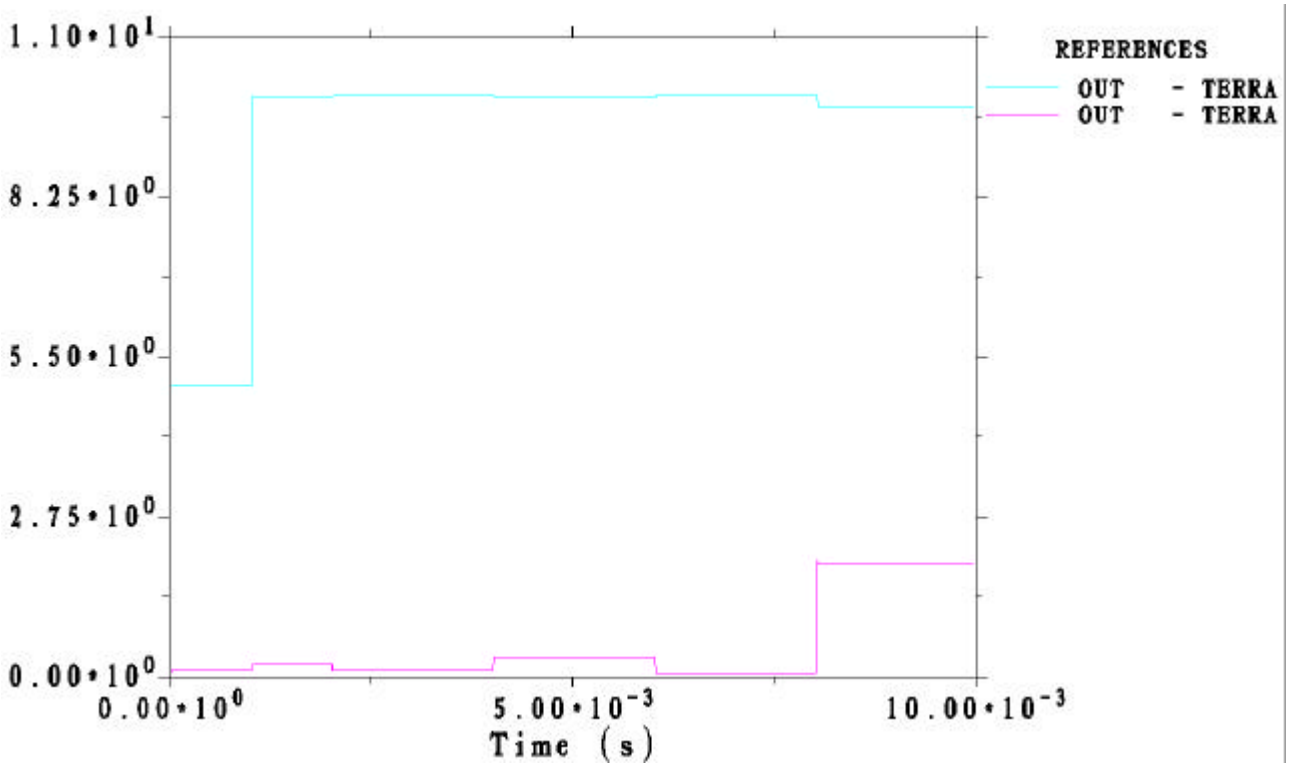
図のように回路を作成する。出力端には電流を見るために TACS のプローブが接続されているが、diode やスイッチなど TACS 素子を接続しなければ動作しないことに注意する。今の場合直流を取り扱っているので、diode を接続した後に TacsProbe を接続しなければならない。TacsProbe は電流を見るので、Type は 91とする。また、可変負荷を図のように時間的に定義する。



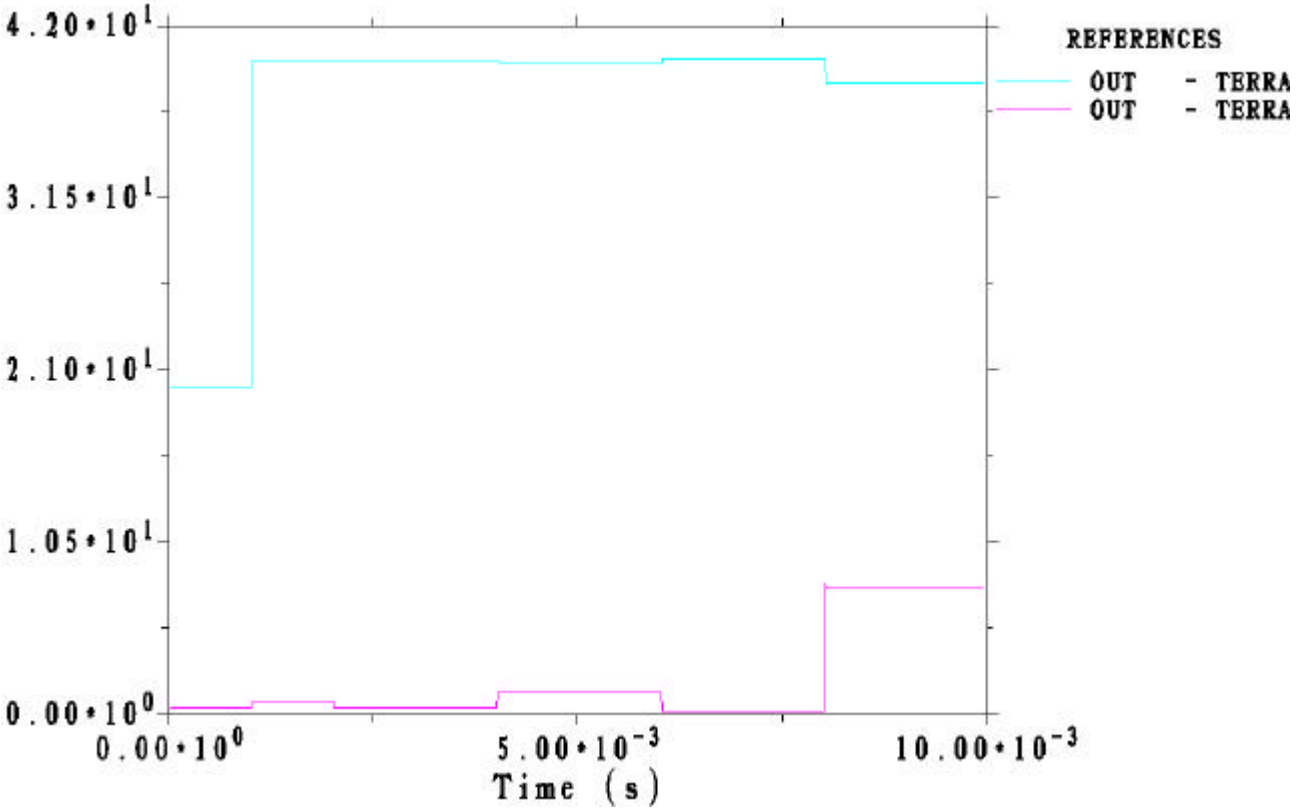
電源の特性を決めるA1、A2の値は図のように決定した。



以上より計算を行い、結果を示す。負荷が変動するにつれてVI特性式より求められる電圧値・電流値になることがわかる。また、負荷が大きくなるにつれて電圧値は増加し、電流は減少する事がわかる。



また電源の特性を変化させるため、A1=200、A2=40とすると、次のような結果がでる。初め初期値を40÷2=20としているので20Vをとる。そしてVI方程式より決まる値をそれぞれ取っていく。図中の水色の線は電圧、ピンク色の線は電流を示す。

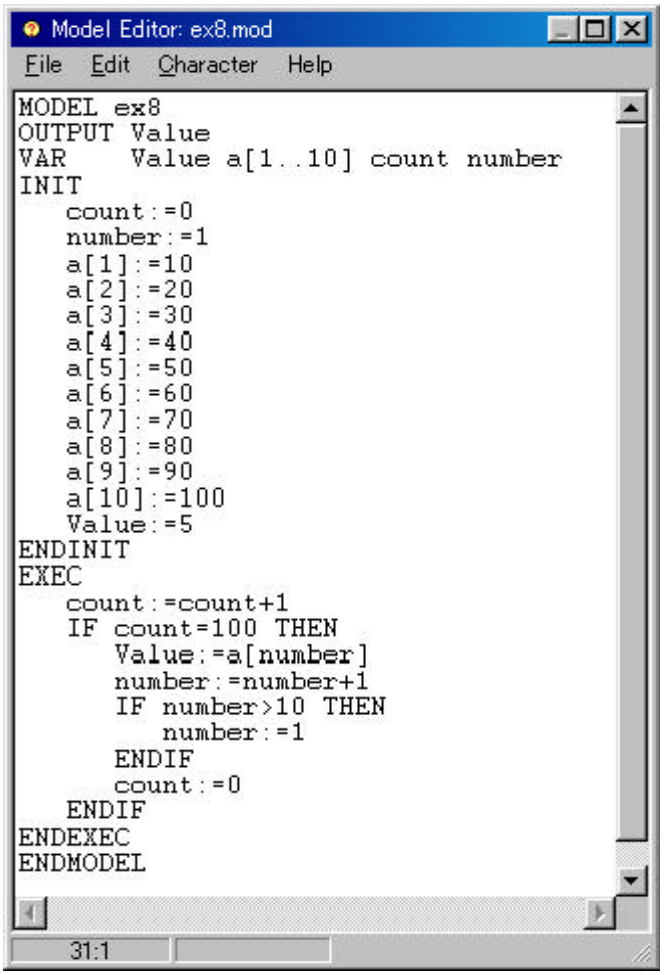


以上により、負荷の影響を持つ分散電源が構成できる。これを応用してその他の電源も構成できる。

例６ データを読み込む素子の作成　(part1)

　ここでは、model 中に配列として組み込まれたデータをある時間間隔で読み出す素子を作成する。

①*.mod ファイルの作成



例として、電源の出力を時間 T において、 100カウント (時間刻み幅× 100[s]) 毎に配列の中にあるデータを順番に読み出すプログラムを作成する。Value は出力を示す。count は出力間隔を決定する変数であり、numberは出力する配列の順番を示す。 初期値として出力 Value は 5とした。

プログラム中の

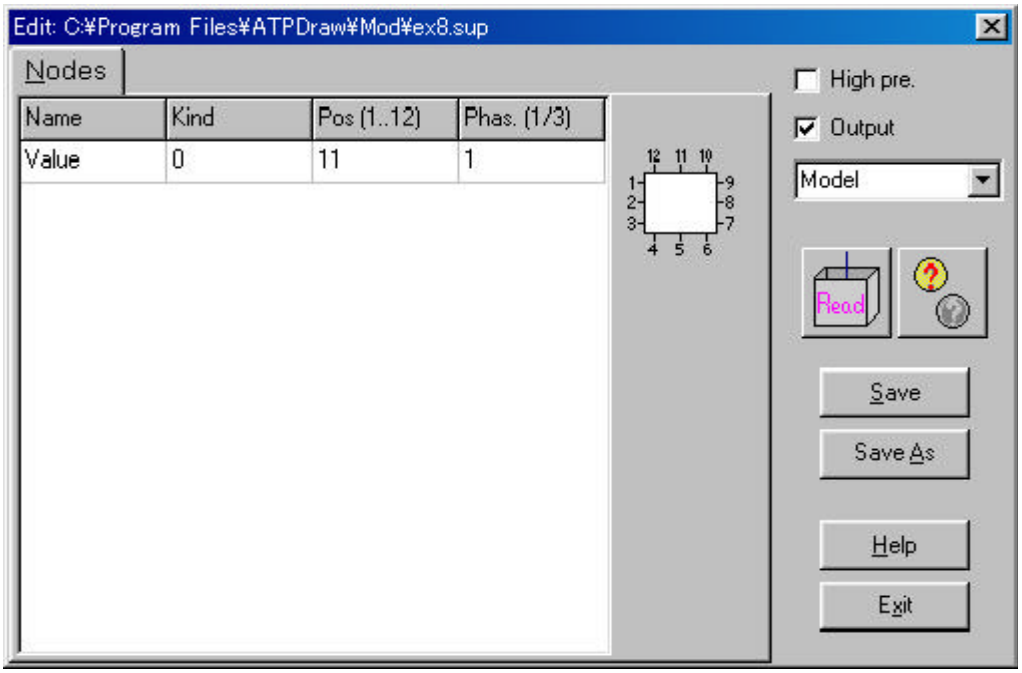
OUTPUT ：素子から出力される変数として定義。

VAR ：プログラム中の変数を定義する。

INIT ：初期値を設定する。

EXEC ：実際に計算などを実行する。この中にプログラムを作成する。

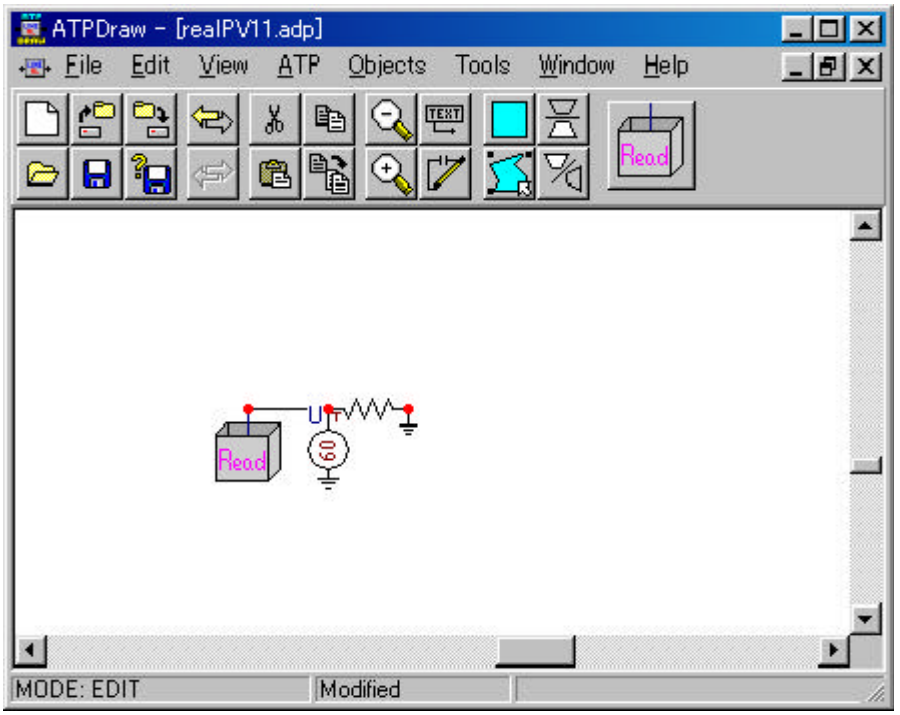
②*.mod ファイルの作成



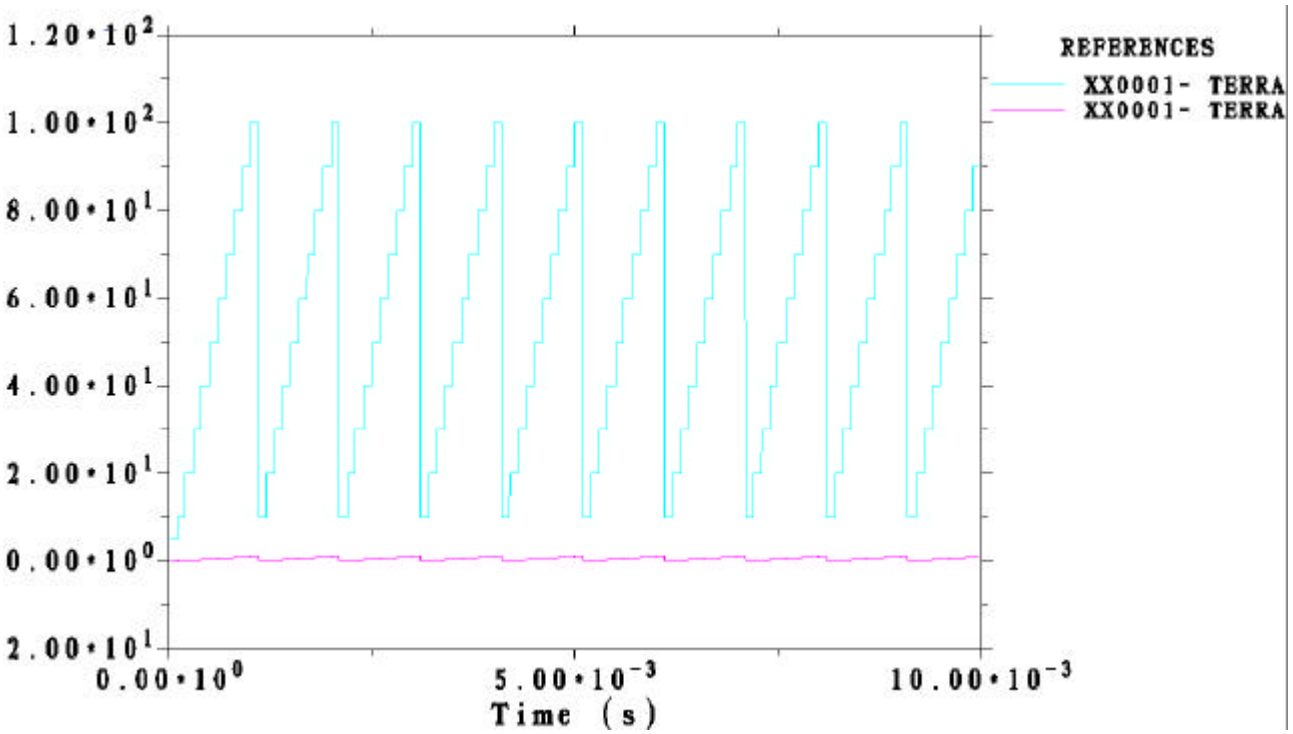
端子 Value は出力なので kind は 0、 Pos は 11とし、出力は単相なので Phas. は 1とした。この例でファイル名は ex8.sup とした。

③回路の作成

作った素子をもとに回路を下の図のように作成した。



回路中の抵抗は 100 Ωとし、 これよりシミュレーションを行うと次のような結果が得られた。

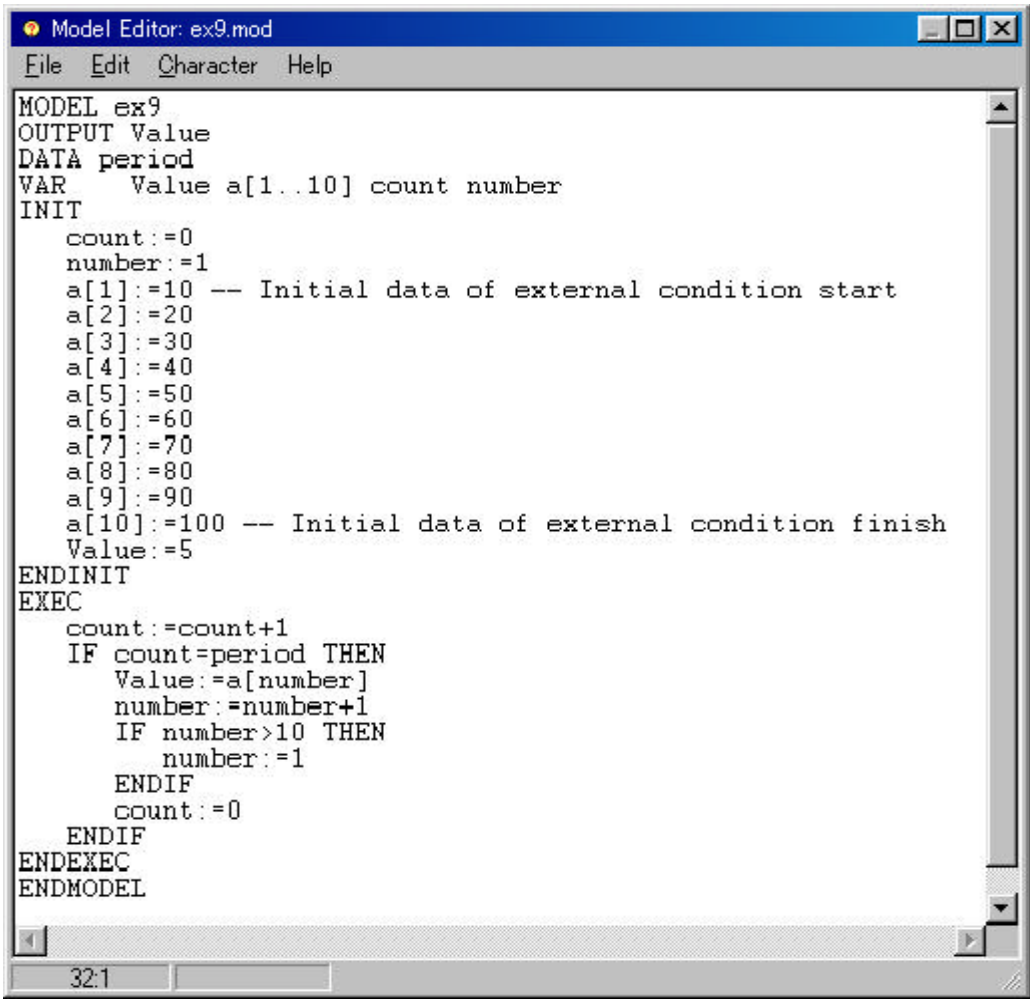


図中の水色の線は電圧を示し、 ピンク色の線は電流を示す。時間は0～0.01[s]で時間刻みは1E-6[s]なので、値の書き換えは100回行われることになる。その結果が正しく出力されているので、100回毎に値を出力する素子が作成できた。

例７ データを読み込む素子の作成　(part2)

　ここでは、model中に配列として組み込まれたデータをある時間間隔で読み出す素子を作成し、例６を改良して読み出す間隔を変更できるようにできる素子を作成する。

①*.mod ファイルの作成



例として、電源の出力を時間Tにおいて、カウント数（時間刻み幅× period（周期）[s]）毎に配列の中にあるデータを順番に読み出すプログラムを作成する。Valueは出力を示す。aはデータが入る配列であり、今回は10個の配列を設定した。countは出力間隔を決定する変数であり、numberは出力する配列の順番を示す。初期値として出力Valueは5とした。

プログラム中の

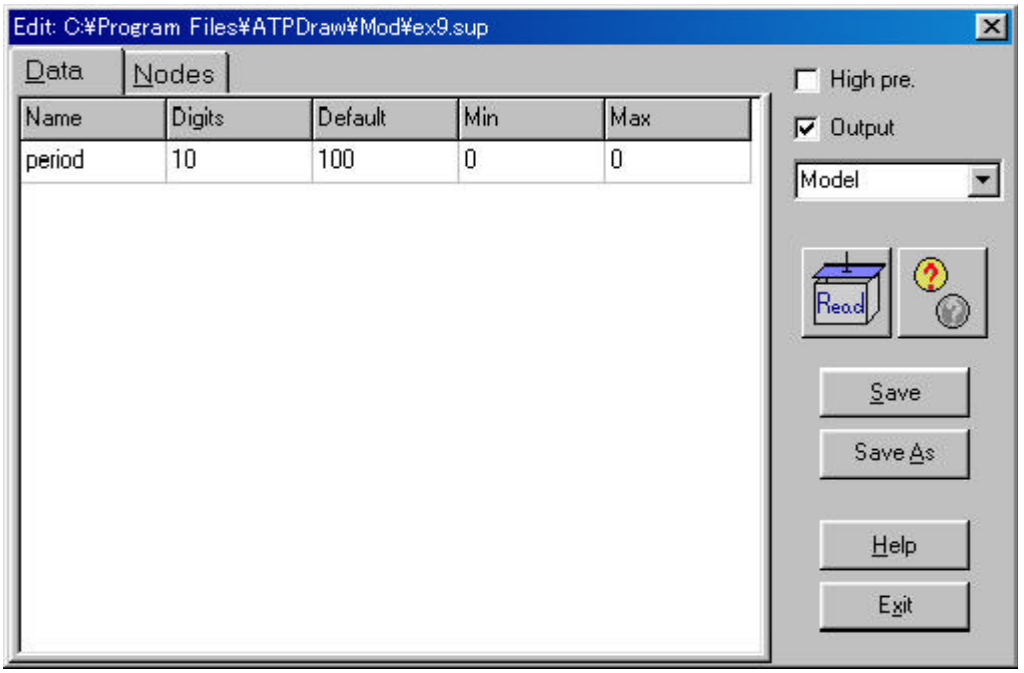
OUTPUT ：素子から出力される変数として定義。

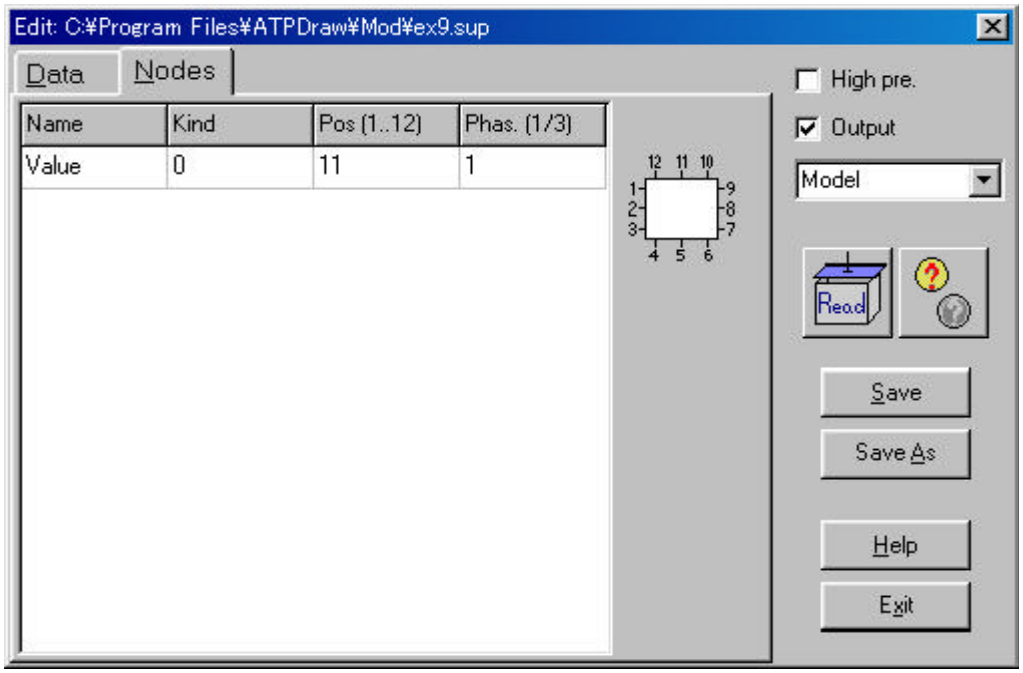
VAR ：プログラム中の変数を定義する。

INIT ：初期値を設定する。

EXEC ：実際に計算などを実行する。この中にプログラムを作成する。

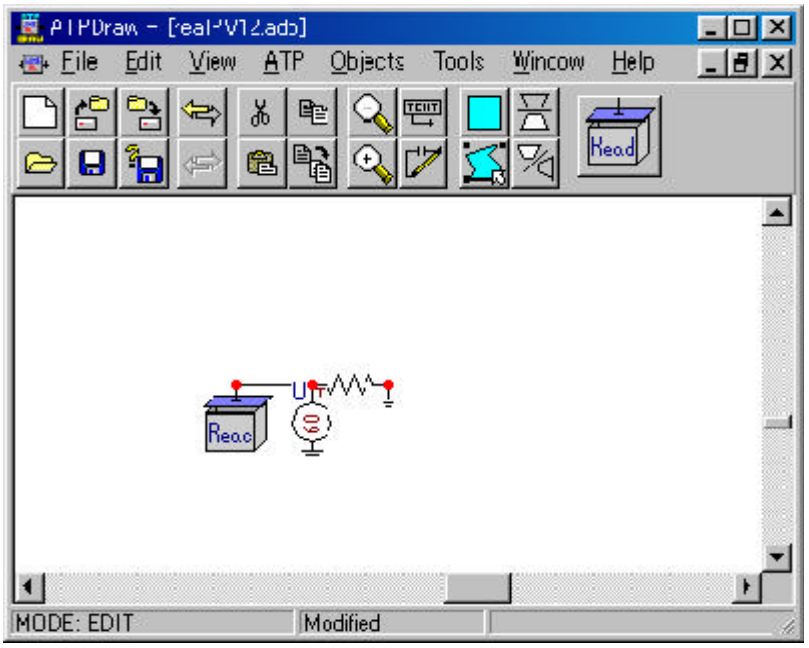
② *.mod ファイルの作成



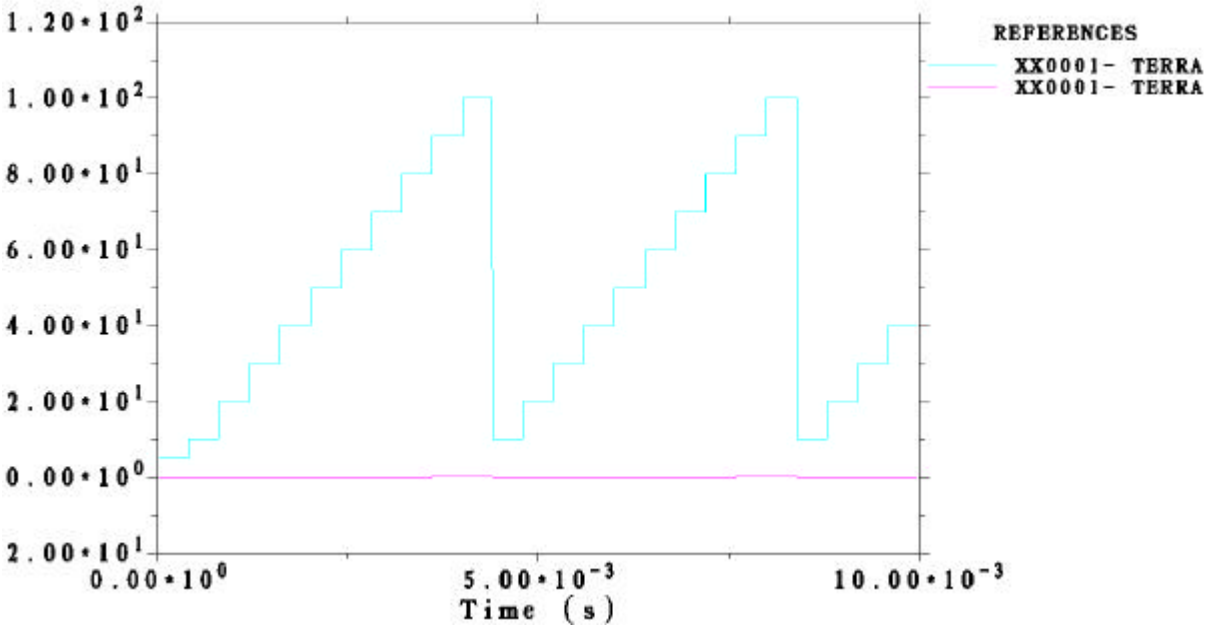


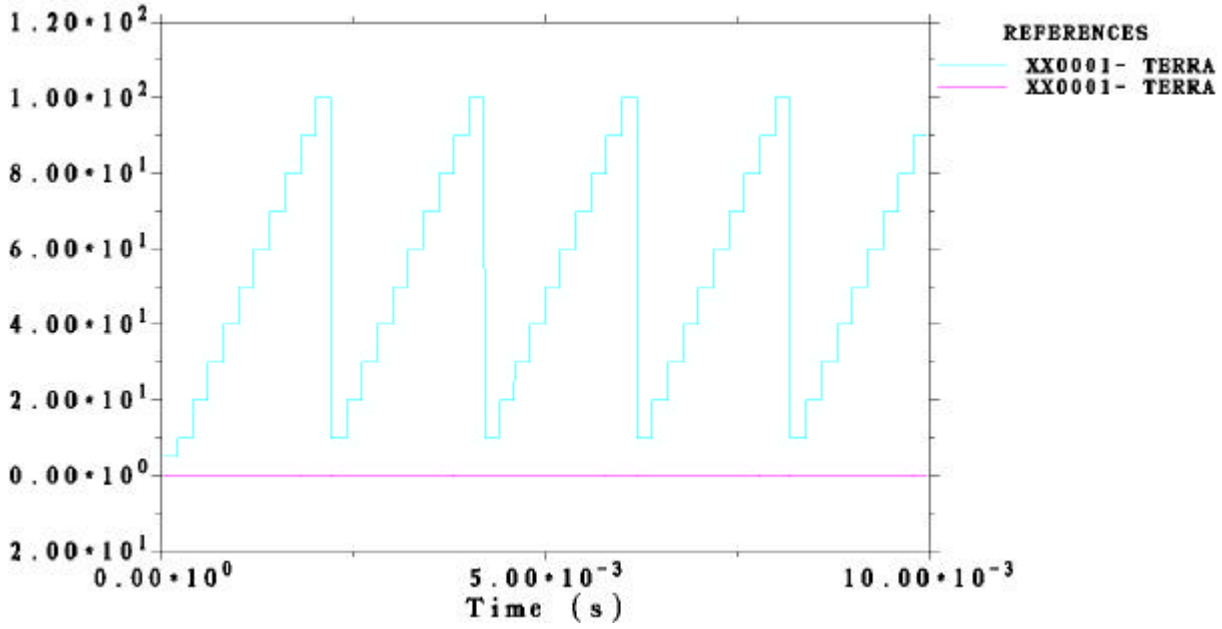
初期値として出力間隔　周期）period を設定し、 100 とした。端子 Value は出力なので kind は 0、 Pos は 11とし、 出力は単相なので Phas. は 1とした。この例でファイル名は ex8.sup とした。

③回路の作成



回路的には例 6の場合と同じとなる。よってシミュレーションを行うと次のようになった。ここでは周期を 200カウントと400カウントとしたものを示す。グラフ中の水色の線は電圧を示し、 ピンク色の線は電流を示す。 これより外部から取り出す時間間隔を変更できる素子を作成することができた。

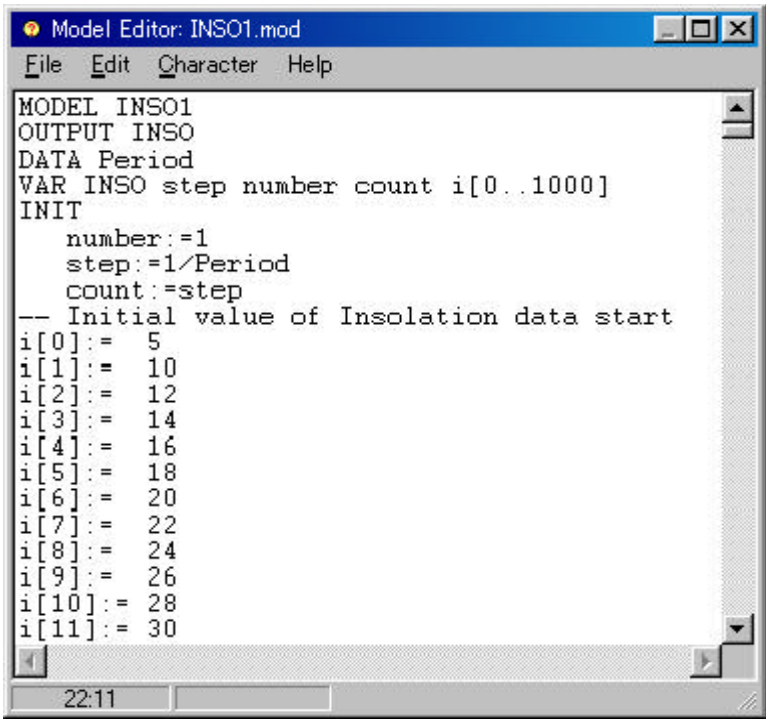


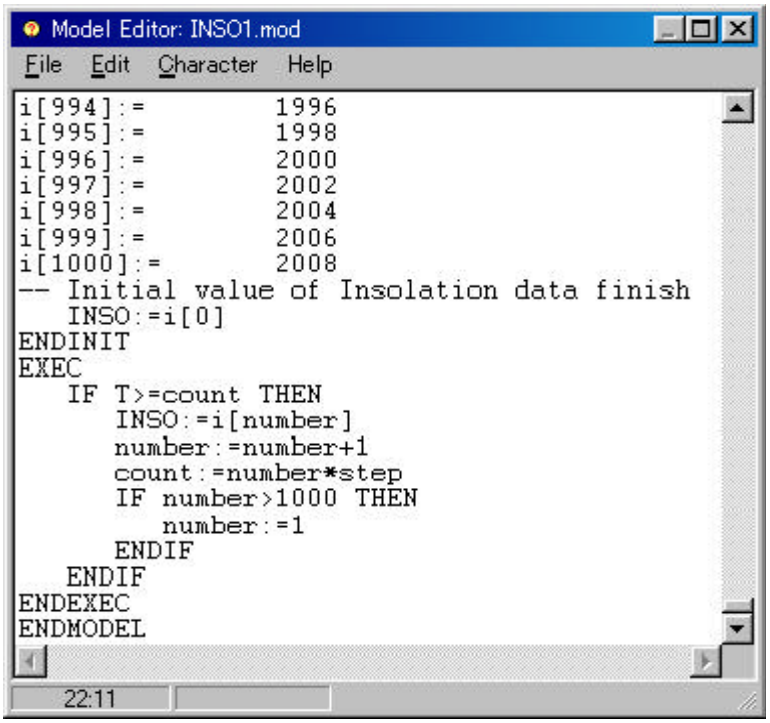


例８ データを読み込む素子の作成　(part3)

　ここでは、model中に配列として組み込まれたデータをある時間間隔で読み出す素子を作成し、 例７を改良して読み出す間隔を変更できるようにできる素子を作成する。またデータの個数をExcelを用いて作成して挿入する。

①*.mod ファイルの作成





例として、 電源の出力を時間Tにおいて、カウント数　(時間刻み幅×1/period　(周期:periodは周波数を示す))[s]）毎に配列の中にあるデータを順番に読み出すプログラムを作成する。 Valueは出力を示す。iはデータが入る配列であり、 今回は1000個の配列を設定した。countは出力間隔を決定する変数であり、 numberは出力する配列の順番を示す。 初期値として出力Valueは5とした。初期条件としてi[0]～i[1000]間で初期値を挿入しているが、 これはExcelで作成した。

プログラム中の

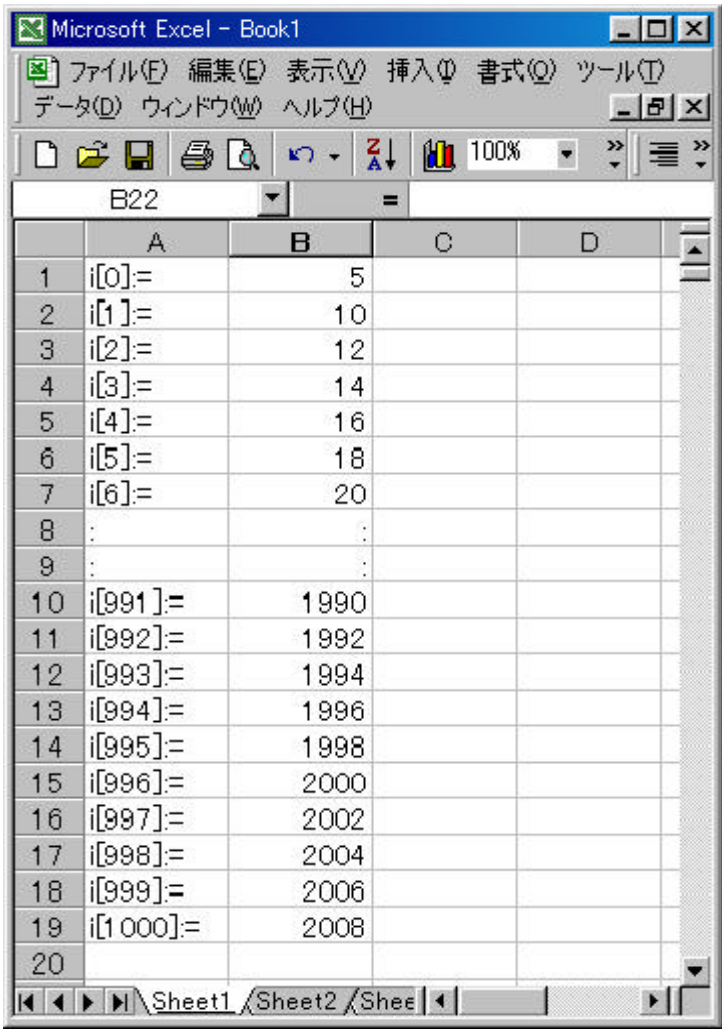
OUTPUT ：素子から出力される変数として定義。

DATA ：パラメータの変数を定義する。ここで定義した変数がATPDraw上で変更できる。

VAR ：プログラム中の変数を定義する。

INIT ：初期値を設定する。

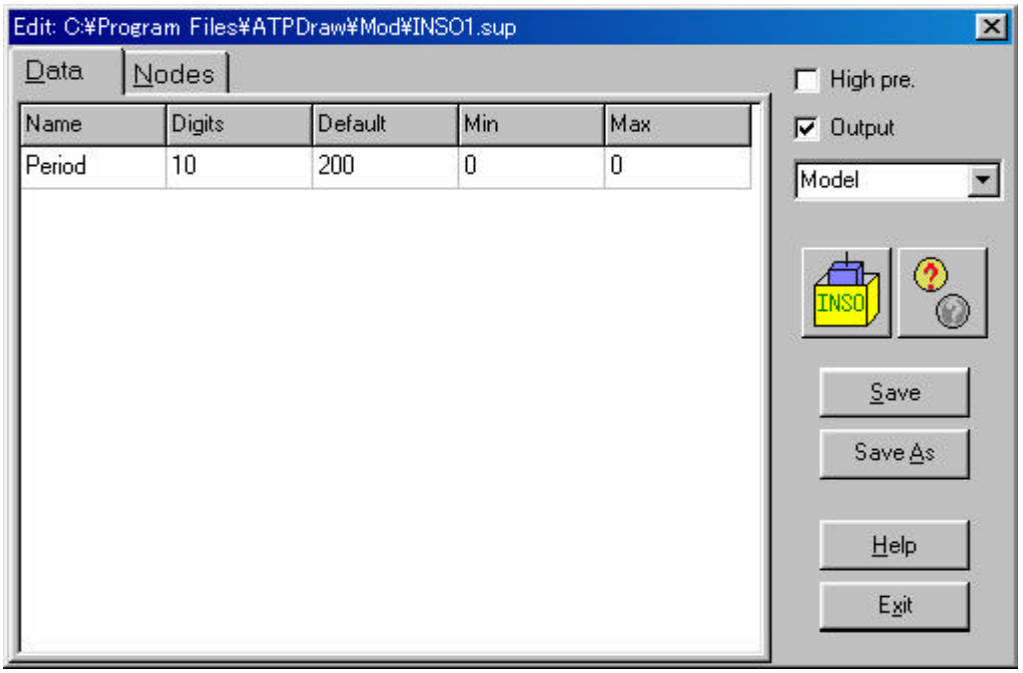
EXEC ：実際に計算などを実行する。 この中にプログラムを作成する。

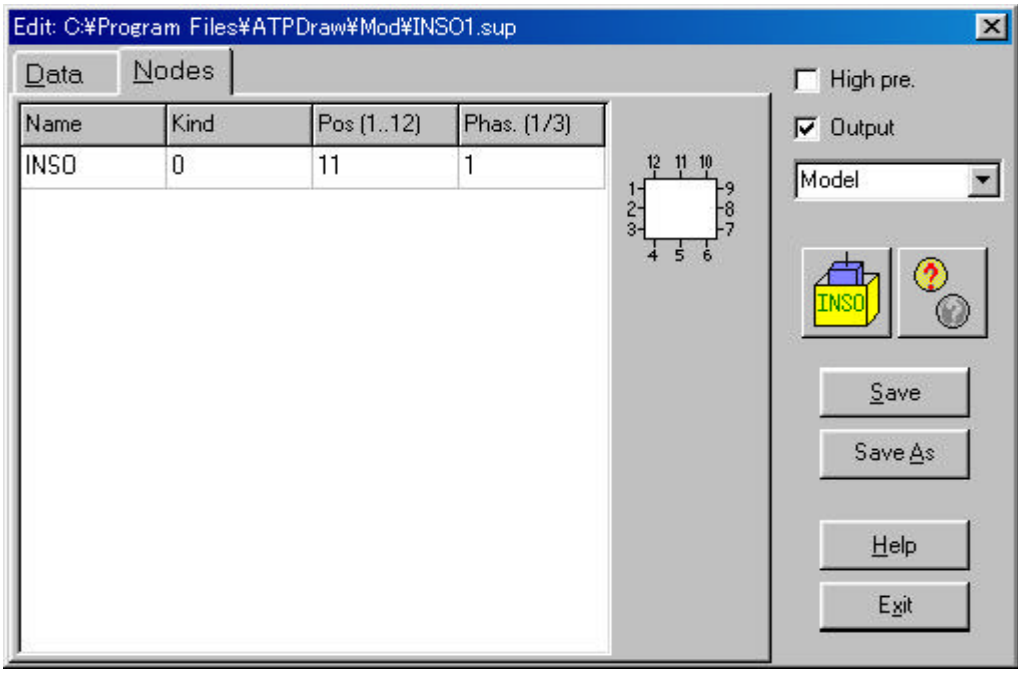


挿入したデータは Excel でこのように作成した。これをそのままコピーして張り付けても良いし、*.txt の形で保存してコピーして張り付けても良い。今の例では初期状態として i[0]:=5、i[1]:=10 から２ずつ足したものを配列に入れた。

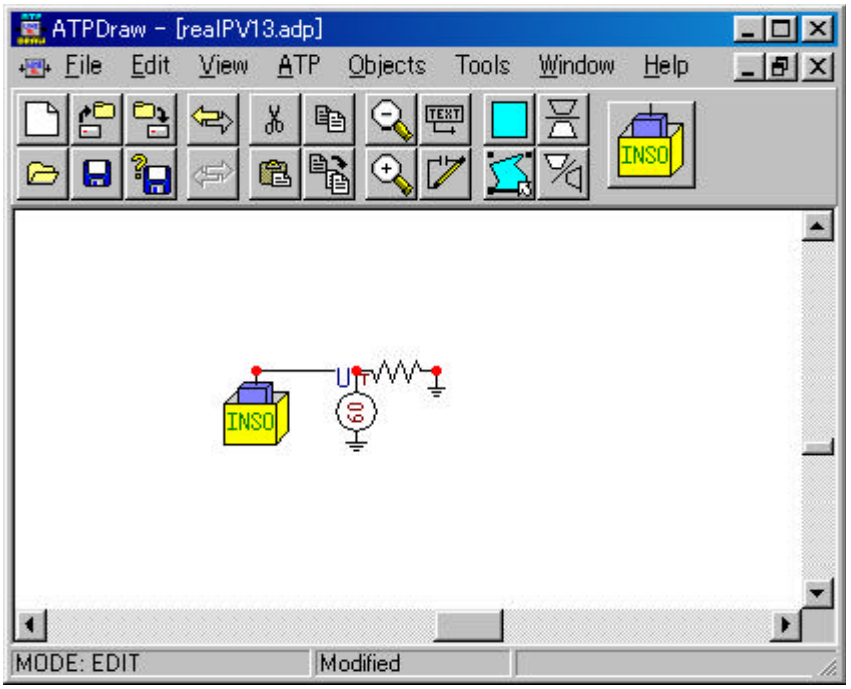
② *.sup ファイルの作成

次のように設定し、 ファイルをINSO.sup として保存した。



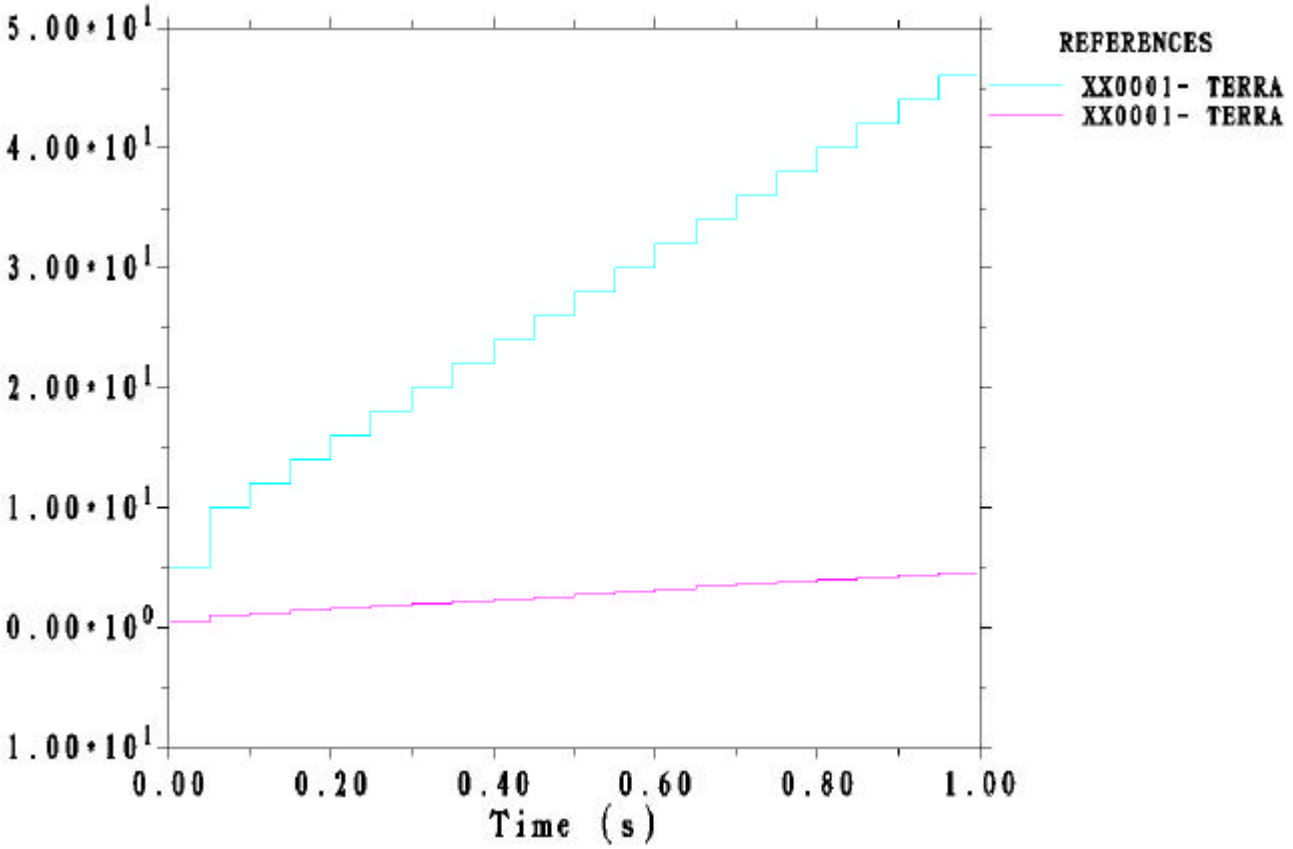


③回路の作成

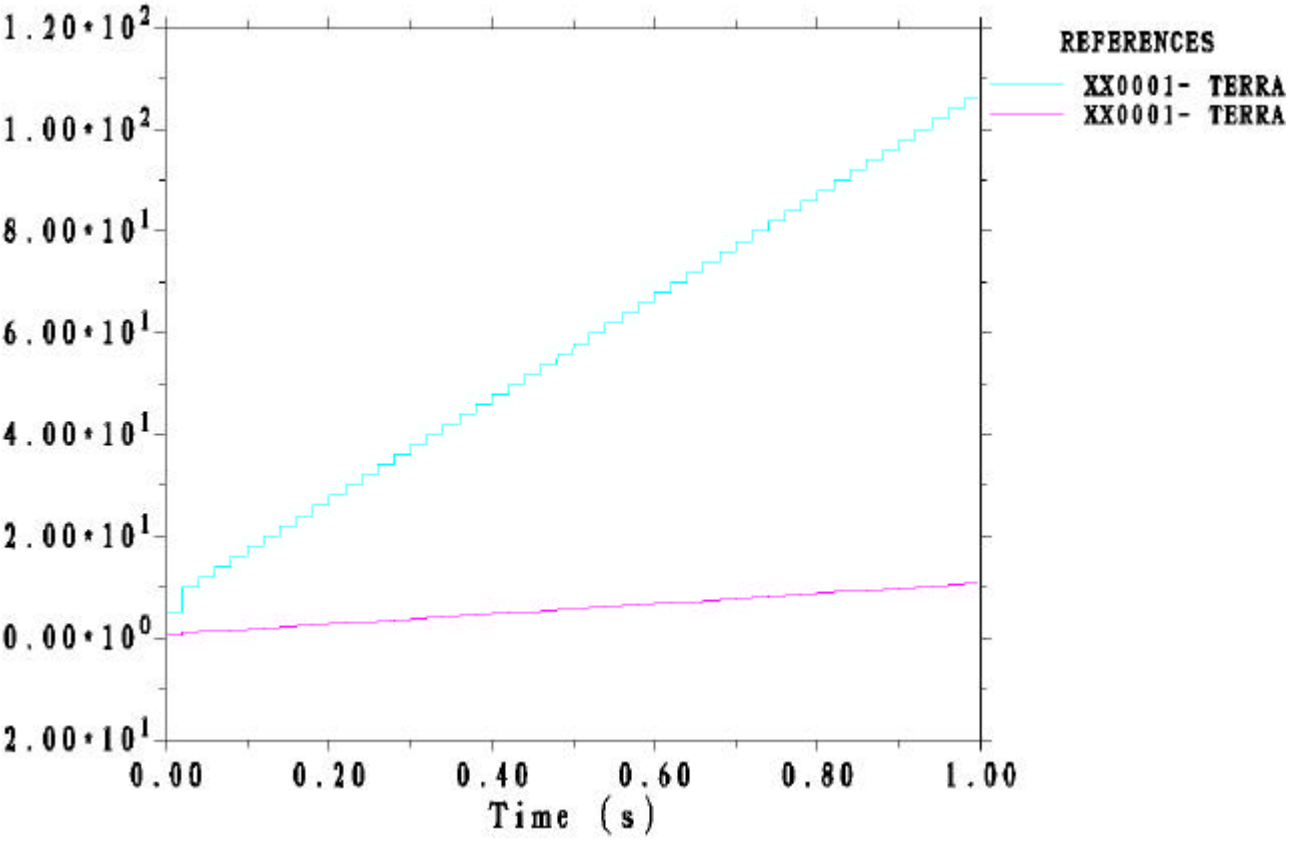


回路を次のように作成し、 外部からデータを読み出す周期(period[Hz])を設定してシミュレーションした結果を示す。回路的には例６･例７と同じである。シミュレーションは１秒まで行っているので、periodの数字だけ出力が変更されることになる。今の例ではperiod=20[Hz] 50[Hz]としたので、20ステップと５０ステップ変化する結果が得られているので、正しい素子が作成できた。

・20[Hz]のシミュレーション結果



・50[Hz]のシミュレーション結果



グラフ中の水色の線は電圧を示し、 ピンク色の線は電流を示す。結果により正しく素子が動作していることがわかった。

例９ データを読み込む素子の作成　(part4)

　ここでは、model中に配列として組み込まれたデータをある時間間隔で読み出す素子を作成し、 太陽電池の外部的特性として例８より実際のデータを代入してその結果を見る。またデータの個数をExcelを用いて修正して挿入する。

①*.mod ファイルの作成

Model Editor: INSO2.mod

File Edit Character Help

```
MODEL INSO2
OUTPUT INSO
DATA Period
VAR INSO step number count i[0..5000]
INIT
   number:=1
   step:=1/Period
   count:=step
-- Initial value of Insolation data start
i[1]:=  0.29688
i[2]:=  0.3125
i[3]:=  0.29688
i[4]:=  0.28125
i[5]:=  0.28125
i[6]:=  0.29688
i[7]:=  0.26563
i[8]:=  0.28125
i[9]:=  0.29688
i[10]:= 0.28125
i[11]:= 0.28125
i[12]:= 0.25
i[13]:= 0.26563
i[14]:= 0.28125
```

1:1

Model Editor: INSO2.mod

File Edit Character Help

```
i[2293]:=       0.51563
i[2294]:=       0.51563
i[2295]:=       0.5
i[2296]:=       0.48438
i[2297]:=       0.51563
i[2298]:=       0.53125
i[2299]:=       0.5
i[2300]:=       0.51563
i[2301]:=       0.51563
-- Initial value of Insolation data finish
   INSO:=i[1]
ENDINIT
EXEC
   IF T>=count THEN
      INSO:=i[number]
      number:=number+1
      count:=number*step
      IF number>2301 THEN
         number:=1
      ENDIF
   ENDIF
ENDEXEC
ENDMODEL
```

1:1

例として、電源の出力を時間Tにおいて、カウント数　(時間刻み幅×1/period　周期　:periodは周波数を示す)[s]）毎に配列の中にあるデータを順番に読み出すプログラムを作成する。Valueは出力を示す。iはデータが入る配列であり、今回は5000個の配列を設定した。countは出力間隔を決定する変数であり、numberは出力する配列の順番を示す。初期値として出力Valueはi[1]=0.29688とした。初期条件としてi[0]～i[2301]間で初期値を挿入しているが、これはExcelで*.txtでファイルを修正し、この形に作成した。

プログラム中の

OUTPUT　:素子から出力される変数として定義。

DATA　:パラメータの変数を定義する。ここで定義した変数がATPDraw上で変更できる。

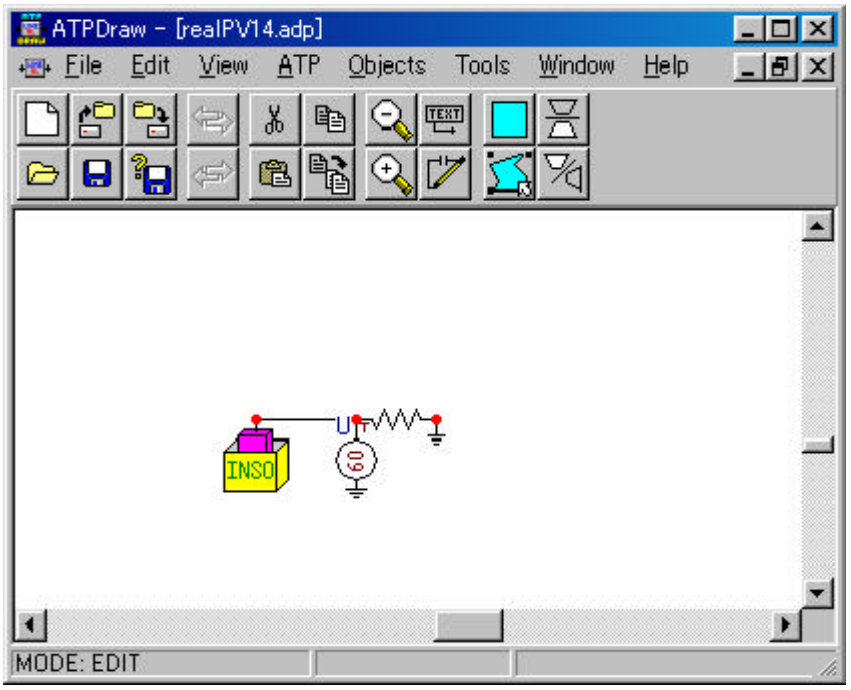
VAR　:プログラム中の変数を定義する。

INIT　:初期値を設定する。

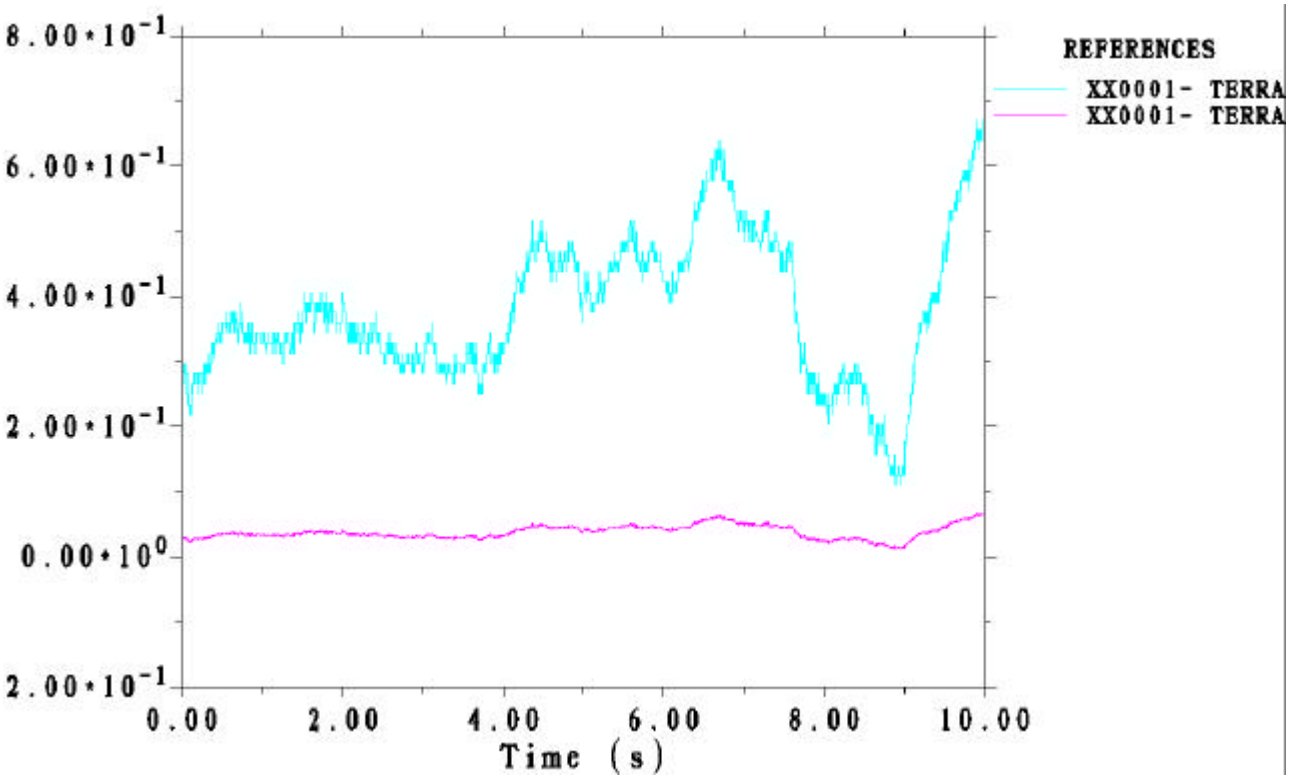
EXEC　:実際に計算などを実行する。この中にプログラムを作成する。

②回路の作成

*.sup ファイルは例8の場合と同じになるのでここでは省略した。 回路は次のように作成した



回路を次のように作成し、 外部からデータを読み出す周期(period[Hz])を設定してシミュレーションした結果を示す。回路的には例６・例７・例８と同じである。 接続した抵抗は10[Ω]として計算した。 シミュレーションは10秒まで行っているので、periodの数字だけ出力が変更されることになる。今の例ではperiod=200[Hz]としたので、２００ステップ×10秒＝2000回変化する結果が得られているので、 正しい素子が作成できた。



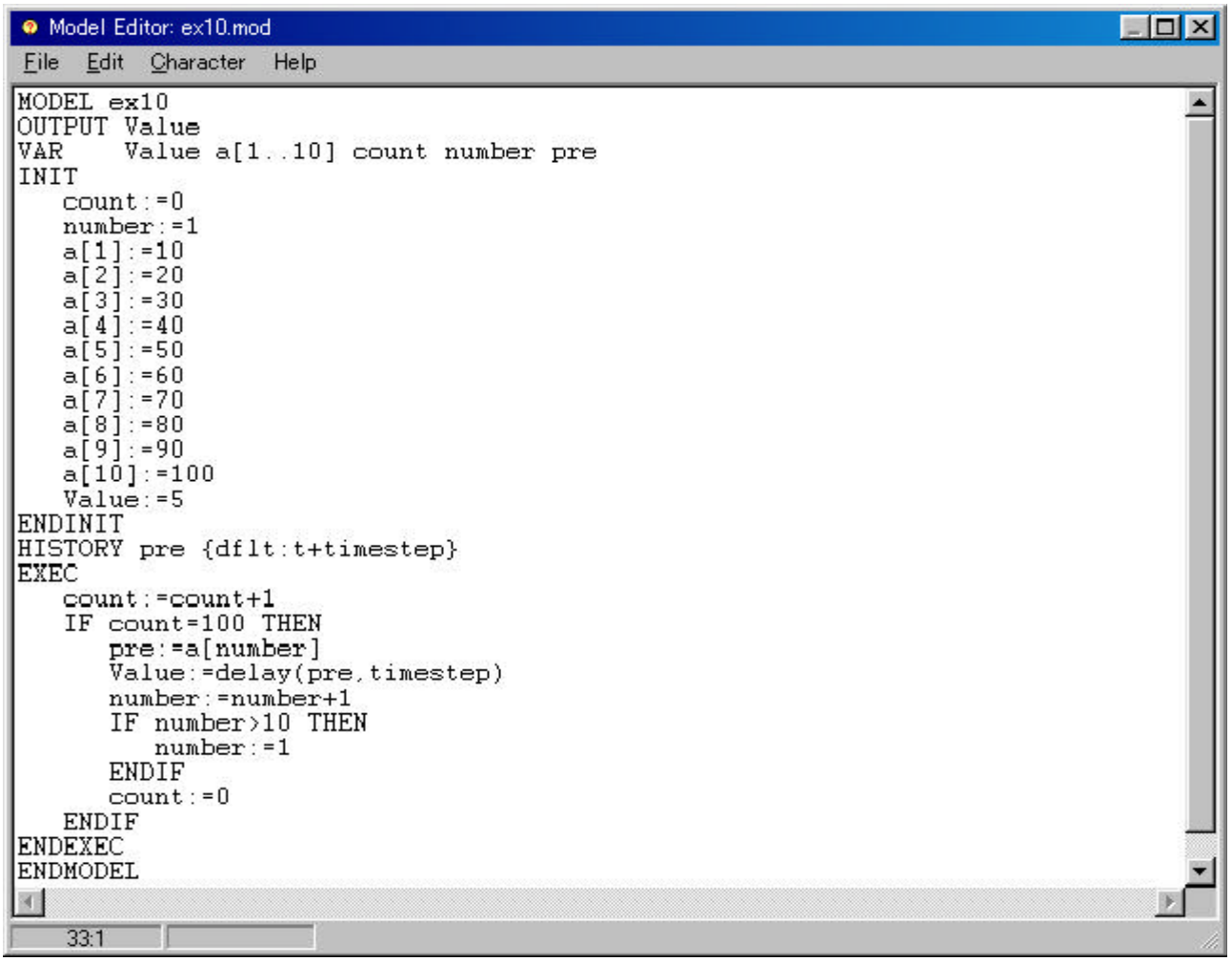
ここで水色の線は電圧を示し、青色の線は電流を示す。実際に条件を太陽電池に入れる場合には電圧のみを見たらいいので、 電圧を出力として加えるようにする。

例 10 信号を遅延させる素子

後に電力などを比較するために信号を遅延させる素子を作成する。

① *.mod ファイルの作成

プログラムは例 11で作ったものを使用し、遅延させる部分を加えて比較を行った。そのプログラムを次に示す。、



遅延を行うためには、遅延させる変数の定義と」、遅延の命令 delay が必要となる。プログラム中の、

HISTORY pre {dflt:t+timestep}

Value:=delay(pre,timestep)

がその部分に当たる。1行目は pre という変数を遅延させると定義し、2行目は pre をtimestepだけ遅らせるというものである。式を書くと pre(t-timestep)ということを表している。

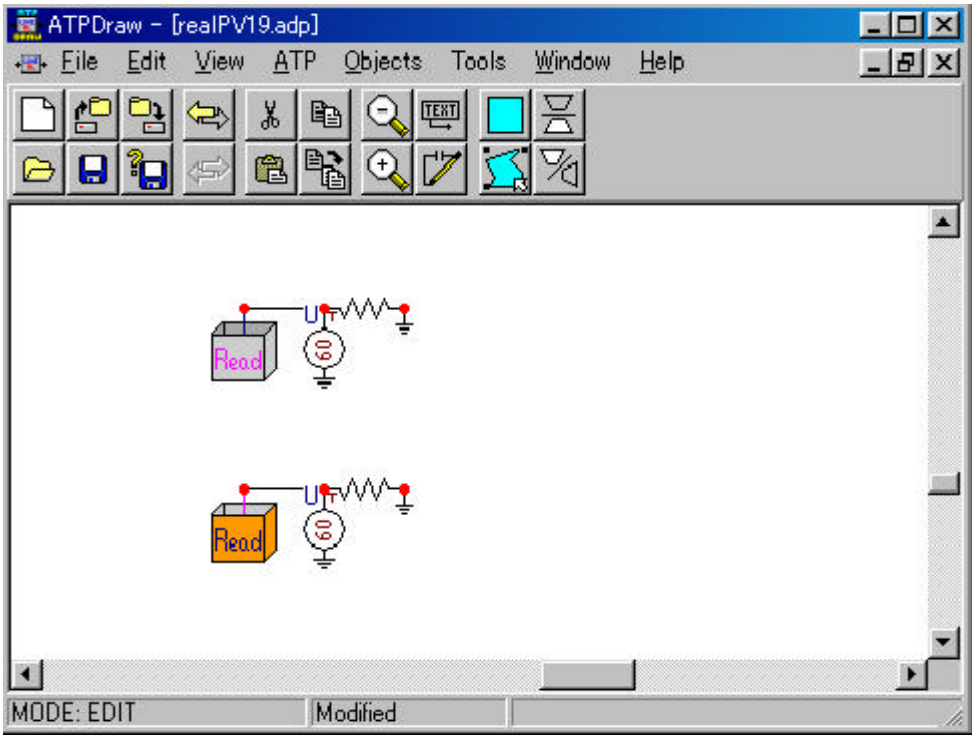
この例では、100カウント毎に配列の中にあるデータを読み出し、それの間は値を維持するというプログラムである。よって、元の信号を 100カウント送らせて出力することになる。

② *.sup ファイルの作成

*.sup ファイルの方は例 11の場合と同じとなるのでそのまま使用した。よってここでは説明を省略する。

③回路の作成

回路は次のように作成した。 回路の上側は元の値を出力する方で、 下側の方は同じ出力を１００カウント（1サイクル）遅らせて出力している。 負荷抵抗は 10 Ωを接続している。



④シミュレーション結果

よって計算した結果を示す。出力は両方とも電圧で、水色の線は上側の素子の出力、ピンク色の線は下側の素子の出力となっている。 よってこれより遅延が正しく行われていることがわかった。