


```

*--      Result of .measure      --*
tdr      = 2.0290709983e-011
tdf      = 2.2386197167e-011
tdelay   = 2.1338453575e-011
idd      = -1.9822238709e-005
*-----*

```

```
C:\Design\tools\example\CHAPTER_8(DotMeasure)>
```

利用できる .measure コマンドの書式については、本章末の「.measure の書式について」を参照してください。

次に、.measure を用いてシミュレーション結果をデータ化する方法について説明します。図2に、インバータの論理しきい値電圧を求める回路例を示します。pMOSFET の W 値は、

```
.param Wp='@{1u,3u,0.25u}'
```

と指定されていますが、これにより、ns-spice は、Wp の値を、1u から 3u まで、0.25u ステップで変化させ、計9回のシミュレーションを連続実行します。実行結果をリスト2に示しますが、.param Wp=の行が変更されながら、各シミュレーションで .measure が実行されていることが分かります。

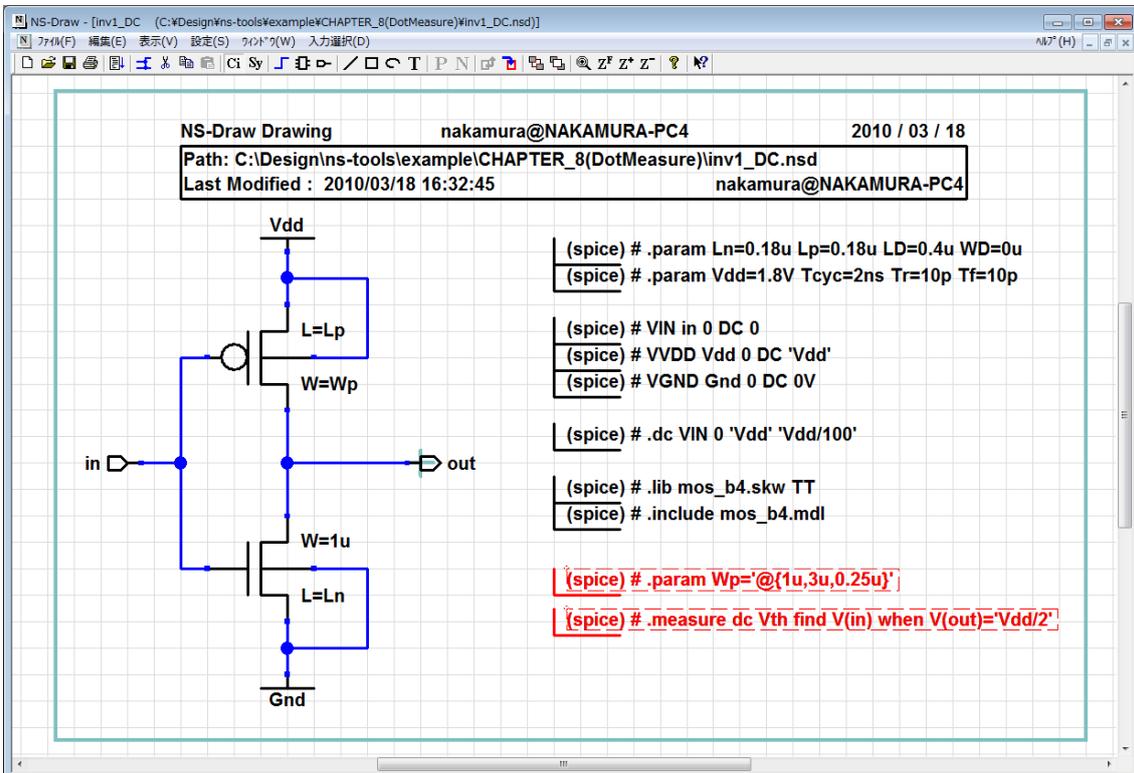


図2 .measure のサンプル2 (inv1_DC.nsd)

リスト1 ns-spice シミュレーションの連続実行結果 (コマンドプロンプト)

```
*****
```

```

** NS-Spice.                Ver. Mar 18 2010 (Nanodesign Corp.) **
*****
*-----*
*--          Param SCAN Analysis : 1( 1)    1/ 9  --*
*-----*
*----  SCAN Parameters  ----*
Line:45 -> .param wp='1e-006'
*-----*
*--          Result of .measure  --*
vth         = 0.7923449582
*-----*
*-----*
*--          Param SCAN Analysis : 1( 2)    2/ 9  --*
*-----*
*----  SCAN Parameters  ----*
Line:45 -> .param wp='1.25e-006'
*-----*
*--          Result of .measure  --*
vth         = 0.8258703817
*-----*
*-----*
*--          Param SCAN Analysis : 1( 3)    3/ 9  --*
*-----*
*----  SCAN Parameters  ----*
Line:45 -> .param wp='1.5e-006'
*-----*
*--          Result of .measure  --*
vth         = 0.854490207
*-----*
* . . .
* . . .
(中略)
* . . .
* . . .
*-----*
*--          Param SCAN Analysis : 1( 8)    8/ 9  --*
*-----*
*----  SCAN Parameters  ----*
Line:45 -> .param wp='2.75e-006'
*-----*
*--          Result of .measure  --*
vth         = 0.9529123191
*-----*
*-----*
*--          Param SCAN Analysis : 1( 9)    9/ 9  --*
*-----*
*----  SCAN Parameters  ----*
Line:45 -> .param wp='3e-006'
*-----*
*--          Result of .measure  --*
vth         = 0.9670423236
*-----*
*-----*

```

```

*--      Param SCAN Analysis  : Finished      --*
*--      measured.csv is created.             --*
*-----*

```

このシミュレーションの結果出力される波形ファイルには、9回分のシミュレーション波形が含まれており、それをプロット図が図3になります。

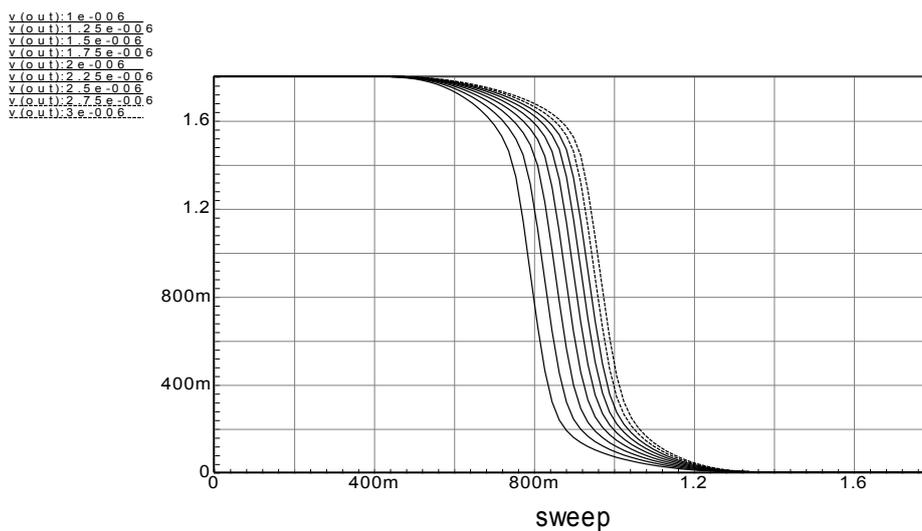
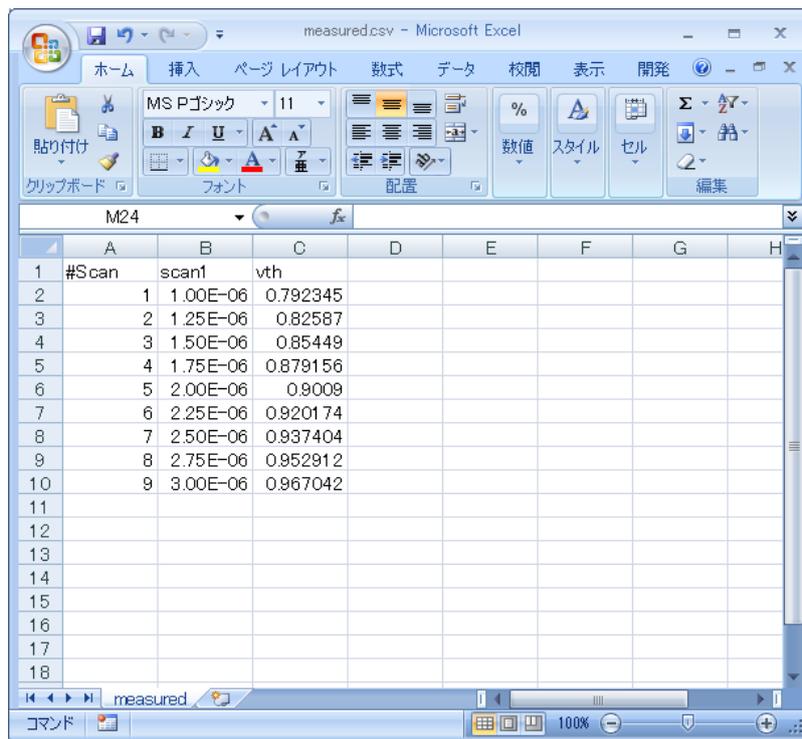


図3 シミュレーション結果（波形）

このとき、.measureの測定結果は、コマンドプロンプト画面に出力されるだけでなく、同時に、カレントフォルダ内に、measured.csvというファイル名で、CSV形式で保存されています。CSV形式のファイルは、一般的な表計算ソフトでそのまま開くことができます。図4に、Microsoft Excelでmeasured.csvを開いた例を示します。表計算ソフトのグラフ化機能を利用することで、図5に示すように直ちにデータをグラフ化することができます。



#Scan	scan1	vth
1	1.00E-06	0.792345
2	1.25E-06	0.82587
3	1.50E-06	0.85449
4	1.75E-06	0.879156
5	2.00E-06	0.9009
6	2.25E-06	0.920174
7	2.50E-06	0.937404
8	2.75E-06	0.952912
9	3.00E-06	0.967042

図 4 表計算ソフトによる CSV(measured.csv) ファイルのオープン

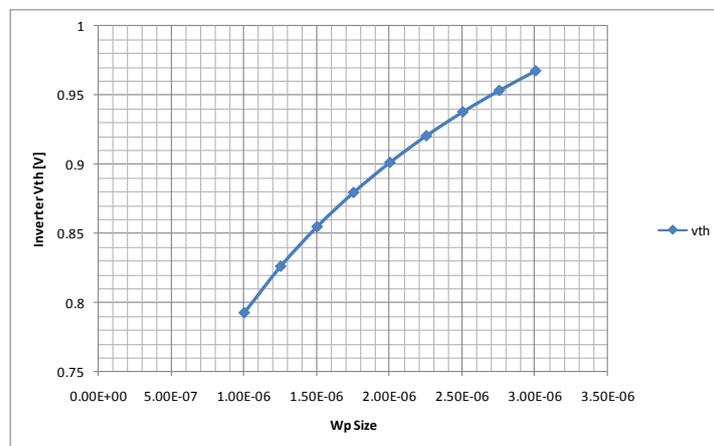


図 5 表計算ソフトによる CSV データのプロット

図 6 に、アンプの AC 解析の例を示します。電源電圧を変化させながら、直流利得 (DCGain) とユニティゲイン周波数 (UnityGainFreq)、位相余裕 (PhaseAtUGF) を計測しています。シミュレーションを実行すると、measured.csv が生成されますが、さらに表計算ソフトで読み込んだ後、グラフ化 (散布図) することで、図 7 のような結果が得られます。

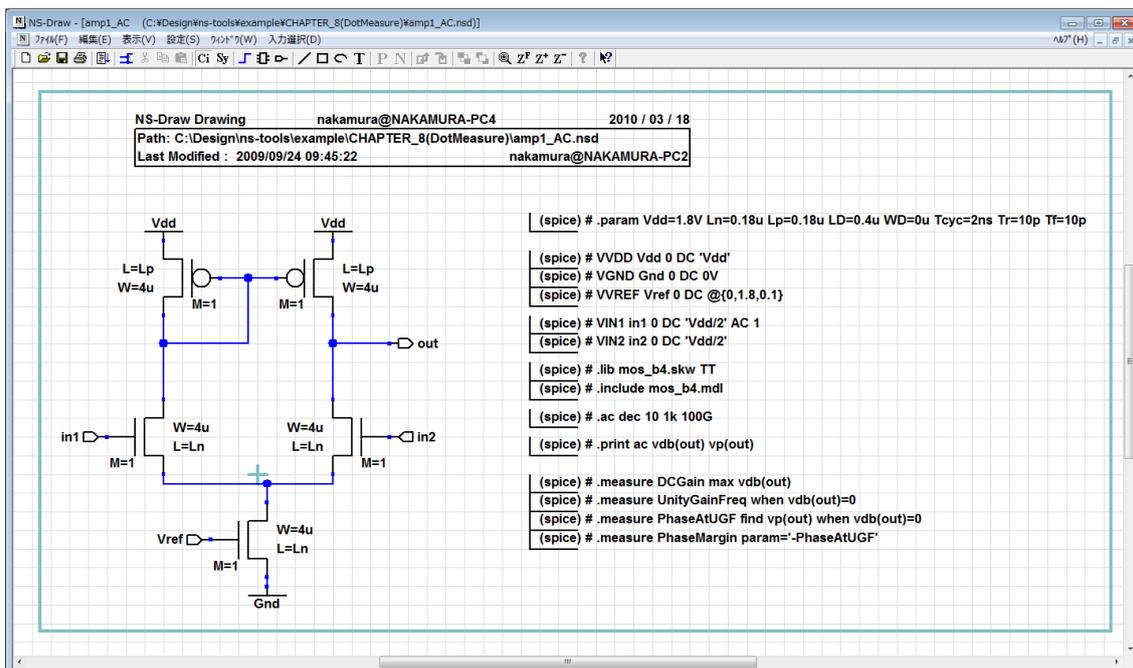


図6 AC解析における.measureの例(amp1_AC.nsd)

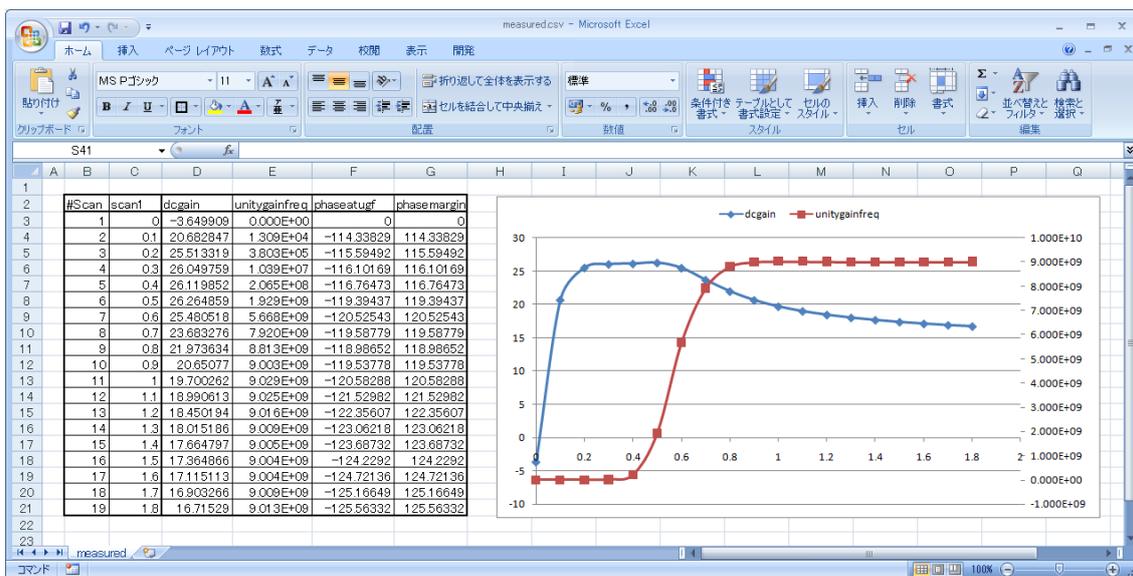


図7 表計算ソフトを利用したmeasured.csvデータのプロット

図8の例のように、一つのネットリスト内に、@{}指定を2つ以上設けることで、それらの全ての組み合わせのシミュレーションを実行することができます。図8の回路図では、RLを、10K, 33K, 100Kの3種、CLを10pから、15pまで、1pステップ(6回)、合計で18回のシミュレーションを行います。その結果のmeasured.csvを図9に示します。

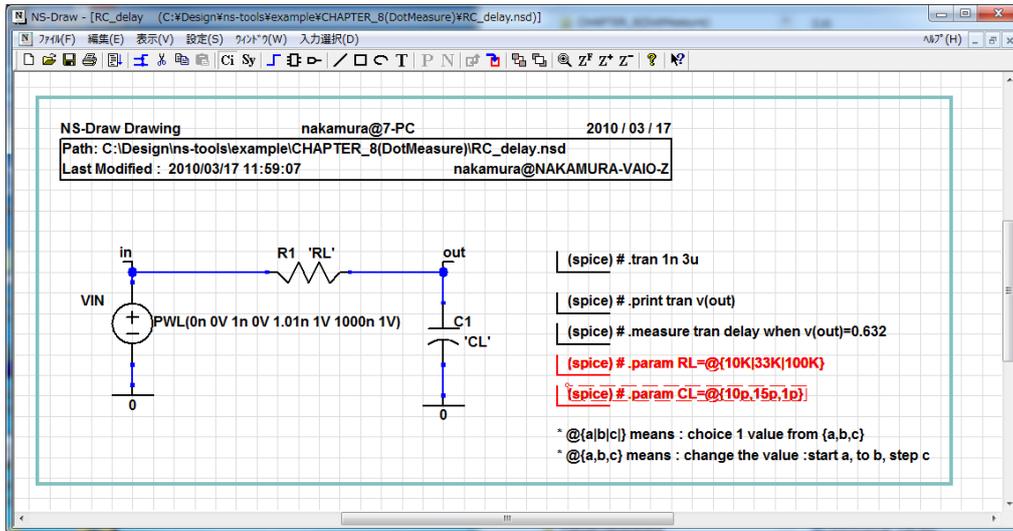


図8 スキャンパラメータの2重化

	A1	#Scan		
	A	B	C	D
1	#Scan	scan1	scan2	delay
2	1	1.00E-11	10k	1.01E-07
3	2	1.10E-11	10k	1.11E-07
4	3	1.20E-11	10k	1.21E-07
5	4	1.30E-11	10k	1.31E-07
6	5	1.40E-11	10k	1.41E-07
7	6	1.50E-11	10k	1.51E-07
8	7	1.00E-11	33k	3.31E-07
9	8	1.10E-11	33k	3.64E-07
10	9	1.20E-11	33k	3.97E-07
11	10	1.30E-11	33k	4.30E-07
12	11	1.40E-11	33k	4.63E-07
13	12	1.50E-11	33k	4.96E-07
14	13	1.00E-11	100k	1.00E-06
15	14	1.10E-11	100k	1.10E-06
16	15	1.20E-11	100k	1.20E-06
17	16	1.30E-11	100k	1.30E-06
18	17	1.40E-11	100k	1.40E-06
19	18	1.50E-11	100k	1.50E-06
20				
21				

図9 measured.csv(図8の回路のシミュレーション結果)

@{ }によるパラメータスキャンの書式について

(1) 列挙型 : @ {a|b|c|d|e|.....}

変更するパラメータ値を@ {}の中で、”|”で区切って順に列挙する。

例 : .param RL=@ {1K|10K|100K|1MEG|10MEG}

シミュレーションは、.param RL=1Kから、.param RL=10MEGまで、5回実行される

(2) 区間型1 (STEP) : @ {start, stop, step, STEP}

変更するパラメータ値を@ {}の中で、初期値 (start)、終値 (stop)、ステップ値 (step)での順で、カンマ”, ”で区切って列挙する。4番目の文字列 : STEPは省略可能。

例 : CL out 0 @ {10p, 100p, 10p}

シミュレーションは、10p, 20p, 30p, 40p, 50p, 60p, 70p, 80p, 90p, 100pの10回実行される

(2) 区間型2 (DIV) : @ {start, stop, ndiv, DIV}

変更するパラメータ値を@ {}の中で、初期値 (start)、終値 (stop)、区間分割数 (ndiv)での順で、カンマ”, ”で区切って列挙し、最後に、文字列 : DIVを記入する。

例 : CL out 0 @ {0, 100p, 10, DIV}

シミュレーションは、0p, 10p, 20p, 30p, 40p, 50p, 60p, 70p, 80p, 90p, 100pの11回実行される

(3) 区間型3 (LOG) : @ {start, stop, ndiv, LOG}

対数軸上で、当区間となるように値を変更する。@ {}の中で、初期値 (start)、終値 (stop)、10倍区間辺りの分割数 (ndiv)での順で、カンマ”, ”で区切って列挙し、最後に、文字列 : LOGを記入する。

例 : CL out 0 @ {1p, 1u, 4, LOG}

シミュレーションは、1p, 1.77p, 3.16p, 5.62p, 10p, 17.7p, 31.6p, 56.2p, 100pの9回実行される

.MEASUREの書式について

- SPICE の結果（波形）から、各種測定を行い数値出力します。
- トランジエント解析、AC 解析、DC 解析で利用可能
- 測定可能項目：
 - 1) 2つの波形の間の遅延、立ち上がり・立下り時間、周期の測定等
 - 2) 波形の平均値、RMS 値、最小値、最大値、peak to peak 値
 - 3) 値の測定：Find-When：特定の出力値になる時の入力値等
 - 4) 1～3) の測定結果の演算結果

1) 2つの波形の間の遅延、立ち上がり・立下り時間、周期の測定

書式：
 .MEASURE <TRAN | DC | AC> 出力名 TRIG 設定 TARG 設定
 (.MEAS と省略可)
 TRIG 設定：
 TRIG 変数名 VAL=trig_val <TD=time_delay> <CROSS=c | LAST>
 + <RISE=r | LAST> <FALL=f | LAST>
 または
 TRIG AT=値 (周波数または時間)
 TARG 設定：
 TARG 変数名 VAL=targ_val <TD=time_delay> <CROSS=c | LAST>
 + <RISE=r | LAST> <FALL=f | LAST>

[例 1] .measure tran tpd1 trig at=0ns targ v(out) val=0.5v cross=1
 (1 波形の遅延時間) out が、立ち上がりまたは立下りで、最初に 0.5V を切る点 (時間) を計測し、tpd1 という名前で出力する。
 [例 2] .measure tran tpd2 trig v(in) val=0.5V td=1ns rise=1 targ v(out) val=0.5v fall=1
 (2 波形間の遅延時間) in が 0.5V になる最初の立ち上がりの時点から、out が 0.5V になる 1 回目 (最初) の立下りまでの時間を計測し、tpd2 という名前で出力する。
 [例 3] .measure tran tcyc trig v(out) val=0.5V rise=2 targ v(out) val=0.5v rise=1
 (波形の周期) out が 0.5V になる 2 回目の立ち上がりから、次の立ち上がりまでの時間を計測し、tcyc という名前で出力
 [例 4] .measure tran tr trig v(out) val='VP*0.1' rise=last targ v(out) val='VP*0.9' rise=last
 (波形の立ち上がり時間) out が VP の 10% になる最後の立ち上がりから、90% になるまでの時間を計測し、tr という名前で出力

2) 波形の区間平均値、RMS 値、最小値、最大値、peak to peak 値

書式：
 .MEASURE <DC | AC | TRAN> 出力名 <AVG | MIN | MAX | PP | RMS | INTEG> 変数名 <FROM=val>
 <TO=val>
 演算の指定
 AVG：平均値、
 MIN：最小値
 MAX：最大値

PP : 振幅 (=最大値-最小値)
 RMS : 自乗平均値 (root mean square)
 INTEG : 積分値 (=平均値*区間長)

[例 1] `.measure tran idd_avg AVG i(vvdd) from=0ns to='tcyc'`
 (波形の平均値) 電圧源 vvdd の 0ns~tcyc の区間の平均電流を求め、idd_avg という名前で出力する。
 [例 2] `.measure tran out_pp PP v(out) from=0ns to='tcyc'`
 (波形の Peak-to-Peak 値) out の 0ns~tcyc の区間の PP 値を求め、out_pp という名前で出力する。

3) 値の測定 : Find-When : 特定の出力値になる時の入力値等

書式 :
`.MEASURE <DC|TRAN|AC> 出力名 WHEN 変数名=値 <TD = val>`
`+ < RISE=r | LAST > < FALL=f | LAST > < CROSS=c | LAST >`
 または、
`.MEASURE <DC|TRAN|AC> 出力名 WHEN 変数名 1=変数名 2 < TD=val >`
`+ < RISE=r | LAST > < FALL=f | LAST > < CROSS=c | LAST >`
 または、
`.MEASURE <DC|TRAN|AC> 出力名 FIND 変数名 1 WHEN 変数名 2=値 < TD=val >`
`+ < RISE=r | LAST > < FALL=f | LAST > < CROSS=c | LAST >`
 または、
`.MEASURE <DC|TRAN|AC> 出力名 FIND 変数名 1 WHEN 変数名 2=変数名 3`
`+ <TD=val > < RISE=r | LAST > < FALL=f | LAST > <CROSS=c | LAST>`
 または、
`.MEASURE <DC|TRAN|AC> 出力名 FIND 変数名 AT=値`

[例 1] `.measure ac unity_gain when vdb(out)=0`
 (ユニティゲイン周波数) 出力 out が、0db になる周波数を unity_gain という名前で出力する。
 [例 2] `.measure ac phase_margin find vp(out) when vdb(out)=0`
 (位相余裕) 出力 out が、0db になる時の、位相 (vp(out)) を phase_margin という名前で出力する。
 [例 3] `.measure dc threshold find v(in) when v(out)='VP/2'`
 (論理しきい値) 出力 out が、VP の 1/2 になる時の、V(in) を threshold という名前で出力する。

4) 1~3) の測定結果の演算結果

書式 :
`.MEASURE <DC|TRAN|AC> 出力名 PARAM='式'`

[例 1] `.measure tran freq param='1/tcyc'`
 (遅延時間→周波数変換) 他の .measure 文で求めた tcyc から周波数を求めて、freq という名前で出力する。式の中で引用できるのは、他の .measure 文の結果と、.param 文で指定された変数