MAST 语法规则简介

一、 MAST 语言的分层结构：MAST 语言是采用分层结构的，关于分层结构将在模板一节中作简单的介绍。

二、 在 MAST 语言中，将涉及到文件命名规则，模板的引用，在程序中的注解，续行符，以及实数的表达式。
程序的注解符＃，续行符\，实数的表达与一般程序相同，但下列符号表示一个实数
a,f,p,n,u,mkmeg,g,t

三、 MAST 语言是用来建立仿真对象的模型，而这种模型又主要是供仿真器使用的，因此在 MAST 建模语言中的变量与其它语言相比有自己的特点，这些特点反映在该语言的许多方面，在变量类型上它有几个比较特殊的变量类型，它们是仿真器变量，系统变量，val 类型的变量；另外它还有状态变量，参数和 argument 变量。
状态变量与其它程序是相同的，它反映?a class="innerlink" href="doc.php?action=viewtitle=氖" target="_blank">氖且桓隼肮 5.牧浚???梢允且桓鲇械丿坏牧浚?部梢允且桓鑫薜丿坏牧俊 T 谄渌?芏喑绦蛑凶刺?恳话闱榭鲂露际鞘?至浚??赣 AST 语言中状态量也可能是模拟量。
在 MAST 语言中所用到的参数或 argument 变量都是一种数字，它们是数字形式的一种，在 MAST 中数字的形式将在下面作一个简单的介绍。

四、仿真器变量(simulator variable)是系统事先定义好了的一组变量，其作用是在模板和仿真器之间传递信息，也是程序对仿真器进行控制的一个手段。模板通过读取这类变量的值来获知系统的状态，通过对这类变量赋值来达到控制仿真器的目的，但并不是所有这类变量都能进行赋值操着。这类变量仿真器能识别，因此在使用时事先不需要申明，只需要引用事先定义好的变量名即可。以下是仿真器变量。
与直流分析有关的仿真器变量 与频域分析有关的仿真器变量 与时域分析有关的仿真器变量 与系统控制有关的仿真器变量
dc_domain freq_domain time_domain time
dc_init freq time_init next_time
dc_start freq_mag tr_start step_size
dc_done freq_phase tr_done statistical
time_step_done
1、与系统直流分析有关的仿真器变量
dc_domain 是反映直流仿真时的一个变量，仿真器在作直流分析时将该变量的值赋为 1，在其它情况下该变量的值为 0。
dc_init、dc_start 反映的都是仿真器开始作直流分析时的变量，在开始作直流分析时该变量的值为 1，在其它情况下该值为 0。
但这两个量也有不同的地方，dc_init 是系统在作第一次直流分析时其值为 1，而在 dc_start 是系统在作每一次直流分析时将其值赋为 1。
dc_done 变量是在直流分析完成时将该变量值赋为 1，在其它情况下该值为 0。
2、与频域分析有关的仿真器变量
freq_domain 系统在作频域分析时将该变量赋值为 1，在其它情况下将该值赋为 0。
freq 变量描述的是在作频域仿真时需要计算的仿真频率，该变量是一个只读变量，可以读取该变量的值来知道系统当前的状态。在进行直流分析和时域分析时该变量的值为 0。

freq_mag、freq_phase 这两个变量与系统的频域分析有关，通常情况下在作频域分析时这两个变量的值为 1，即在作频域分析时既要作幅频特性，也要作相频特性。但这两个变量也可以通过赋值来改变其值。例如，在系统作频域分析时，将 freq_mag 变量的值赋为 0，则系统在作频域分析时就只作相频特性而不作幅频特性，系统作直流分析和时域分析时该变量的值为 0。

3、与时域分析有关的仿真器变量

time_domain 系统在作时域分析时将该变量赋值为 1，在其它情况下将该值赋为 0。

time_init、tr_start 反映的都是仿真器开始作时域分析时的变量，在开始作时域分析时该变量的值为 1，在其它情况下该值为 0。但这两个量也有不同的地方，time_init 是系统在作第一次直流分析时其值为 1，而在 tr_start 是系统在作每一次直流分析时将其值赋为 1。

tr_done 变量是在时域分析完成时将该变量值赋为 1，在时域分析其它时刻为 0。系统作直流分析、频域分析时其值也为 0。

Time_step_done 在作暂态分析时，每一个仿真步长完成时该变量的值将被赋为 1，在其它情况下该变量的值为 0。该变量是一个只读变量，该变量不能通过程序赋值。

4、与系统控制有关仿真器变量

time 变量是仿真器在作时域分析时的仿真过程中所确定的时间，它是由仿真步长决定的，它只有在作时域分析时有效，在作频域和直流分析时该值为 0。该变量时一个只读变量，程序不能对该变量赋值。

next_time 该变量的值决定了仿真器的下一个仿真时间。在通常情况下，在作时域分析时，相对与某时间点而言，仿真器要进行的下一个时间点是由仿真步长决定的，在变步长计算中，系统的仿真步长又是由计算误差决定的，那是在 next_time 变量没有赋值的情况下，如果对 next_time 变量赋值，则仿真器下一步计算的时间点由仿真器所在的当前时间再加上 next_time 变量的值。在使用该变量时需要对该变量赋值。

step_size 该变量决定了仿真器的步长，在定步长仿真中该步长作为仿真器仿真的步长，在变步长仿真中，该步长作为仿真开始时的起始步长。

statistical 该变量反映了系统是否在作统计分析，如 Monte 分析，参数扫描分析等。当系统在作统计分析时，该变量的值为 1，系统没有作统计分析时该变量的值为 0。

四、 系统变量

系统变量时系统数学模型中的变量，它们可以 across 类型的变量，也可以是 var 类型的变量，也可以是 ref 变量。

var 变量是与 val 变量相对应的，而 ref 变量仅是 var 类型变量的一种形式，如果一个变量需要 var 变量的特征而它又必须定义成连接点，则这时因将 var 类型的变量定义成 ref 变量。

如前所述，所谓建模就是建立一系列描述系统特性的方程，在 MAST 语言中需要建立的描述系统特征的方程数必须与 var 类型的变量数相等。

五、 val 类型的变量

在 MAST 语言中，val 类型的变量是一个中间变量，在模型中需要建立中间变量时可以将其定义为 val 变量，与 val 变量相对应是 var 变量，var 变量就是非中间变量，在模板将非中间变量定义为 var 变量。对于系统而言，只有在需要的时候系统才对 val 类型的变量赋值，对 val 类型变量赋值将在 values 段中进行。关于 values 段的描述将在模板中介绍。

六、 申明

在前面讲述了一些关于 MAST 语言的变量，但是在使用这些变量时必须首先申明这些变量，

在 MAST 语言中，不仅包括变量的申明，而且还包括单位申明、指针申明、连接点申明、argument 申明、参数申明。

1、 单位申明：单位申明包括两种情况，一种是模拟量的单位申明，另一个是数字量的申明。单位申明的关键词 unit，对于模拟量的单位定义的语法规则是

unit{“缩写”,” 全称”,” 描述” } 名字

unit{“rpm”,”revolutions/minute”,”argular velocity”} w

对于数字量而言，其单位定义语法规则为

unit{“名”,” 值”,” 符号”,” 图形”,

“名”,” 值”,” 符号”,” 图形”}名字

对于数字量而言，该数字量有几种状态，在单位定义中就应该有几行。

unit state {l4_0, "0", "0", "low.1",

l4_1, "1", "1", "high.1",

l4_x, "x", "x", "middle.1",

l4_z, "x", "z", "middle.1"} logic_4 =l4_1

在每一行中的第一列为该变量代表这一状态的名称，第二列为该变量这一状态的布尔代数值，第三列可以是任何字符，它表示该变量在这一状态在输出打印时的符号，第四列的表达式为 symbol.type，它表示该变量的这一状态在输出图形的特性。关于 symbol 和 type 的意义如下表所示。

symbol 图形

low low line

middle mid level line

high high line

unknown low and high line

style 线的类型 颜色

1 Solid Black

2 Long dash line Dark blue

3 Dash line Red

4 2dot_2dash line Purple

5 Dot_dash line Dark green

6 Dot line brown

2、 指针申明(pin)

在 MAST 语言中，所谓指针就是元件模板的对外连接点的一种形式，它也是模板对外连接的最常用的一种形式，定义指针实际上就是定义了元件的一种连接方式。

与指针有关的两个重要概念，就是指针中涉及到的单位和变量。

指针的定义总是与使用它们的单位联系在一起的，即在定义一个指针的同时就定义了与该指针相连接的单位。

指针的定义总是与一对 through 变量和 across 变量联系在一起的，即在定义一种指针的同时也就定义了一对 through 变量和 across 变量。定义指针的关键词是 pin。定义指针的语法规则如下：

pin identifer across uint1 through unit2 或

pin identifer through unit1 across unit2

eg

pin electrical through i across v
pin rotational through w across t
3、连接点申明
在 MAST 语言中，除了可以用指针类型来申明连接点外，还可以采用其它形式的连接点，连接点的通常形式有：
pin_type，state，var，ref。Pin_type 的定义已在前面讲述过了，关于其它类型的定义与相同类型变量的定义方法相同。
4、变量的申明
across 变量和 through 变量的申明是在申明指针时已经隐含申明了，对于这一类的变量不需要进行申明，而参数和 argument 变量都是一些数，它们的申明其实就是对数的申明。而对于其它的变量的关键词分别是 var,val,state,ref，simvar 它们的定义规则为
关键词 单位 变量
e.g
var unit id [,id]
val unit id [,id]
ref unit id [,id]
state unit id [,id]
state logic_4 inm
state v vin
state nu handle[2]

七、 数据
在 MAST 语言中还有两种变量，就是 argument 和参数，对于这两种变量而言，它们就是 MAST 语言中数据的一种。因此必须搞清楚 MAST 语言中的数据。
就总体而言，MAST 语言的数据类型有两种，一种是简单数据（**），另一种是组合数据(composite)。
1、简单数据：简单数据有三种类型，它们分别是 number，enum，string。它们分别是数值，枚举型数据和字符数据
e.g
number vcc=5, dc_input, rload=10k, cload
enum{_n,_p} bjt_type =_n
2、组合数据：组合数据有两种基本类型，一种是结构数据（struc），另一种是联合数据(union)。采用结构数据的目的是把一些相关变量构成一个整体，而采用联合变量的目的是希望一个变量在不同的时候处理不同类型的信息。
1) 结构数据的定义：定义结构数据用关键词 struc，其定义的语法为
关键词 名 结构体
e.g
struc fred{
enum{_n,_p} type
number is, beta, c
}
在上面的这个结构体中，struc 是结构体的关键词，fred 是这个结构的名称，这个结构体是由一个枚举数和三个实数构成。如果要对该结构数的引用格式如下：
struc fred bjt, m1, m2

在 Saber 库中的一些模型，在用到结构数据时常常用到如下几种形式

```
struc fred{
enum{_n,_p} type
number is, beta, c
}bjt
struc fred m1,m2
```

//

```
struc {
enum{_n,_p} type
number is, beta, c
}bjt,m1,m2
```

在最后一个例子中，没有名，它是申明了三个实体。

2) 联合数据的定义：定义结构数据用关键词 union，其定义的语法为

关键词 名 结构体

```
union source {
number dc
struc{
number mag
number phase
} ac
} input1=(dc=5),input2=(ac=())
```

3 数组

1）简单数组：简单数组是与简单数相对应的，即组成简单数组的各个元素都是简单数，简单数组定义的语法规则是

type id [subscripts]

在上式中，type 表示数组类型，id 是数组名，subscripts 表示该数组中的元素。在 MAST 语言中的数组与其它语言中的数组有相同的概念，即与数组有关包括数组的长度，数组的维数。在 MAST 语言中的一个特定就是数组的长度可以不定，它根据仿真时间的长短来确定数组的长度。关于对数组元素数的分配可以采用如下方式：

type id [lower:upper]

e.g

```
number id[0:2]
```

在上式中，lower 可以忽略，当忽略时，其值为 1。

```
number tc[2]
number samples[*,0:1]
number x[0:50,5,-1:1]
```

2) 联合数组：联合数组是与联合数相对应的，即组成联合数组的各个元素都是联合数，联合数组定义的语法规则是，首先定义一个联合类型的变量，然后再定义联合数组。如下例所示：

4 操作符

```
struc   sa_points{
number breakpoint, increment
}
struc sa_points svbe
```

, svbc

//
但是在 Saber 库中常常看到其它的几种形式
struc　{
number breakpoint, increment
} svbe
, svbc