VHDL语言基础（三）

单击输入您的封面副标题

VHDL语言的基本语法

VHDL语言提供了一系列顺序语句和并行语句

- 顺序语句的用来实现模型的算法部分
- 并行语句用来表示黑盒子的连接关系

结构体是由一系列的并行语句构成

- 每个并行语句表示一个功能单元
- 多个功能单元构成一个结构体

```
ARCHITECTURE 结构体名 OF 实体名 IS
        ——说明部分
BEGIN
        ——并行语句A
        ——并行语句B
        ——并行语句C
END 结构体名;
```

并行语句

进程语句（PROCESS）

信号赋值语句（SIGNAL ASSIGNMENT）

块语句（BLOCK）

元件调用语句（COMPONENT）

端口映射语句（PORT MAP）

生成语句（GENERATE）

过程调用语句（PROCEDURE CALL）

并行语句

1.并行信号赋值语句

- 一个并行信号赋值语句代表着对该信号赋值的等价的进程语句

普通并行信号赋值语句

条件信号赋值语句

选择信号赋值语句

普通并行信号赋值语句

```
信号量 <= 敏感信号量表达式;
```

- 是VHDL语言最基本的语句之一
- 用在进程内部时，是作为顺序语句出现的
- 用在结构体中，进程的外部时，是作为并行语句出现的

```
Q <= A AND B AND C;
```

并行信号赋值语句等效为一个进程

```
PROCESS(A, B, C)
BEGIN
        Q <= A AND B AND C;
END PROCESS;
```

条件信号赋值语句

```
目的信号量 <= 表达式1 WHEN 条件1 ELSE
              表达式2 WHEN 条件2 ELSE
              表达式3 WHEN 条件3 ELSE
                ……
              ELSE 表达式n;
```

- 每个表达式后面都跟有用WHEN所指定的条件，如果满足该条件，则该表达式值代入目的信号量
- 如果不满足条件，则再判断下一个表达式所指定的条件
- 最后一个表达式可以没有条件，它表明，在上述表达式所指明的条件都不满足时，则将该表达式的值代入目标信号量

```
LIBRARY  IEEE；
USE  IEEE.STD_LOGIC_1164.ALL；
ENTITY  MUX4  IS
PORT（
        I0，I1，I2，I3：  IN  STD_LOGIC；
        A，  B：  IN  STD_LOGIC；
        Q：  OUT  STD_LOGIC）；
END  MUX4；
ARCHITECTURE  RT  OF  MUX4  IS
        SIGNAL  SL：  STD_LOGIC_VECTOR（1  DOWNTO  0）；
BEGIN
        SEL <=  B&A；
        Q <=  I0  WHEN  SEL=“00”  ELSE
              I1  WHEN  SEL=“01”  ELSE
              I2  WHEN  SEL=“10”  ELSE
              I3  WHEN  SEL=“11”  ELSE
             ‘X’；
END  RT；
```

可以等效为用IF语句描述的进程

```
ARCHITECTURE  RT  OF  MUX4  IS
        SIGNAL  SL：  STD_LOGIC_VECTOR（1  DOWNTO  0）；
BEGIN
        SEL <=  B&A；
        PROCESS(SEL, I0, I1, I2, I3)
        BEGIN
                IF SEL=”00” THEN
                        Q <= I0;
                ELSIF SEL=”01” THEN
                        Q <= I1;
                ELSIF SEL=”10” THEN
                        Q <= I2;
                ELSIF SEL=”11” THEN
                        Q <= I3;
                ELSE
                        Q <= ‘X’;
                END IF;
        END PROCESS;
END  RT；
```

选择信号赋值语句

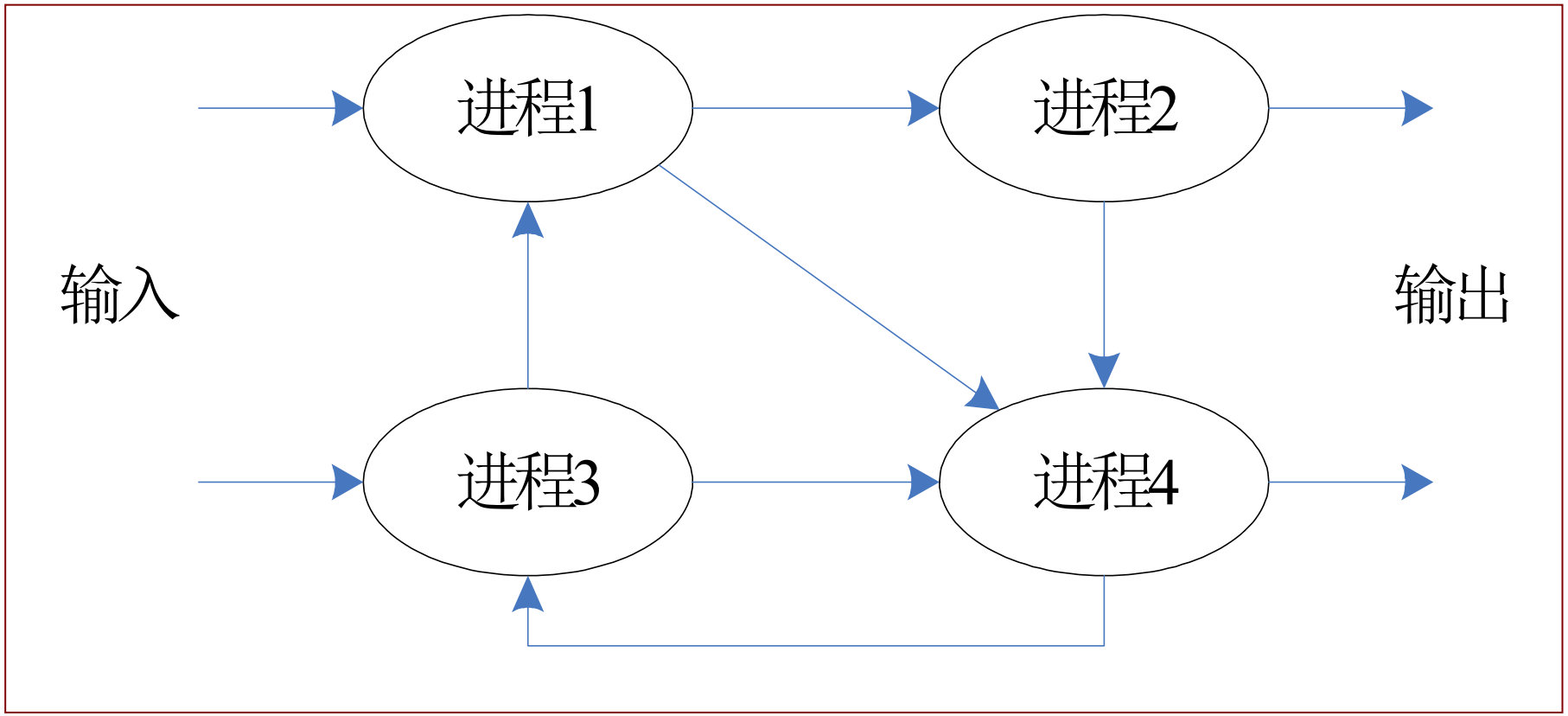
```
WITH   表达式   SELECT
        目的信号量 <=  表达式1  WHEN  条件1
                       表达式2  WHEN  条件2
                         ……
                       表达式n  WHEN  条件n;
```

- 选择信号赋值语句类似于CASE语句
- 对表达式进行测试，当表达式取值不同时，将使不同的值代入目标信号量

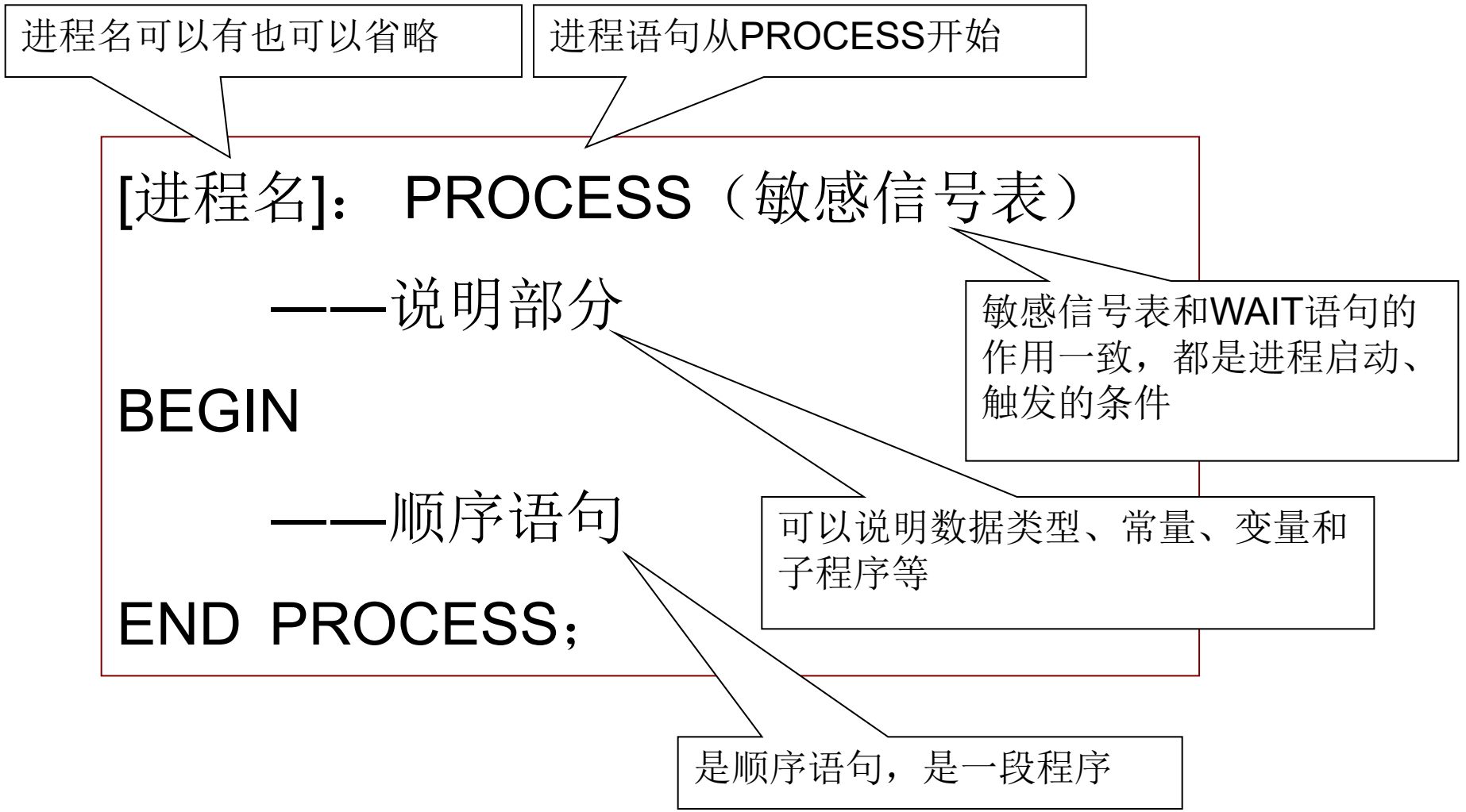
VHDL模型的最基本的表示方法

- 并行执行的进程（PROCESS）
  - 进程语句是并行语句，进程语句之间是并行关系
  - 进程语句内部则是由一组在整个模拟期间连续执行的顺序语句构成，是顺序执行的
  - 进程之间通过信号或共享变量进行通信
- PROCESS是描述硬件并行工作的最常用、最基本的语句

进程模型图



进程语句（PROCESS）



```
ENTITY  DFF  IS
        PORT（
                D，CLK：  IN  BIT；
                Q：  OUT  BIT）；
END DFF；
ARCHITECTURE    BEHAVE  OF  DFF  IS
BEGIN
        P1：PROCESS（CLK）
        BEGIN
                IF  CLK'EVENT AND CLK = '1'  THEN
                        Q <=  D  AFTER  10  ns；
                END  IF；
        END  PROCESS  P1；
END  BEHAVE；
```

- PROCESS语句中含有敏感表，则等价于该进程语句内的最后一个语句是一个隐含的WAIT语句
- 含有敏感信号表的进程语句中不允许再显式出现WAIT语句
- 不含有敏感信号表的进程中可以有多个显式的WAIT语句
  - WAIT语句的执行会暂停进程的执行，直到敏感信号发生变化或某种条件满足为止

WAIT语句的格式

WAIT [ON 信号表] [UNTIL 条件] [FOR 时间表达式];

WAIT ——无限等待

WAIT ON ——等待敏感信号发生变化

WAIT UNTIL 表达式——等待表达式成立

WAIT FOR 时间表达式——等待一段由时间表达式指定的时间

```
PROCESS
        VARIABLE   TEMP：   BIT；
BEGIN
        TEMP  ：= A OR B；
        C <=   NOT   TEMP；
        WAIT   ON   A，   B；          -- 等待信号A或B的值改变
END   PROCESS；


WAIT  ON  A  FOR  50  ns; -- 等待信号A的值改变或已
                                过50ns的时间
WAIT  UNTIL  A = ‘0’；                -- 等待信号A的值为 ‘0’；
WAIT；                                -- 永远等待
```

进程语句的特点

- 一个结构体中可以有多个进程存在
- 进程之间并行运行，并可存取结构体或实体中所定义的信号；
- 进程内部的所有语句都是按顺序执行的；
- 为启动进程，在进程结构中必须包含一个显式的敏感信号量表或者一个WAIT语句；
- 进程之间的通信是通过信号传递来实现的

顺序语句

进程、过程和函数内部是顺序语句

完全按程序中出现的顺序执行各条语句

前面语句的执行结果可能直接影响后面语句的执行

- WAIT语句
- 变量赋值语句
- 信号赋值语句
- IF语句
- CASE语句
- LOOP语句
- NEXT语句
- EXIT语句
- RETURN语句
- NULL语句
- REPORT语句

顺序语句

1.变量赋值语句

- 变量的说明和赋值限定在顺序区域内
  - 只能在进程、函数和过程中
  - 信号值可以赋给变量，变量的值也可以赋给信号
- 变量的赋值是立即执行的

变量赋值语句的格式

目的变量 := 表达式；

```
SIGNAL   SIG ：  BIT：= ‘0’；
        ……
PROCESS
        VARIABLE   EVENT_ON_SIG ： INTEGER ： = 0；
BEGIN
        WAIT   ON   SIG；
        EVENT_ON_SIG ： = EVENT_ON_SIG  +  1；
END   PROCESS；
```

变量值只是在进程或子程序中使用，无法传递到进程之外

顺序语句

2.信号赋值语句

- 是VHDL语言中进行行为描述的最基本的语句
- 信号的更新是在所有进程被执行并挂起后

变量赋值语句的格式

目的信号量 <= 信号量表达式

A <= B；　　A得到B的值

A <= B AFTER 5 ns；　　当B发生变化5ns以后才被代入到信号A

顺序语句

3. IF语句

- 根据所指定的条件来确定执行哪些语句
- 书写格式通常可以分为三种类型

格式1

```
IF   条件   THEN
       ——顺序语句
END   IF;
```

- 如果条件成立，则执行IF语句所包含的顺序处理语句
- 如果条件不成立，程序将跳过IF语句所包含的顺序处理语句，而向下执行IF语句后的语句
- 这种描述可以用来描述D触发器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY DFF IS
PORT（
        CLK，D： IN STD_LOGIC;
        Q: OUT STD_LOGIC）;
END DFF;
ARCHITECTURE BEHAVE OF DFF IS
BEGIN
        PROCESS（CLK）
        BEGIN
                IF（CLK'EVENT AND CLK = '1'）THEN
                        Q <= D;
                END IF;
        END PROCESS;
END BEHAVE;
```

格式 2

```
IF   条件   THEN
      ——顺序语句
ELSE
      ——顺序语句
END   IF;
```

- 指定的条件满足时，将执行THEN和ELSE之间所界定的顺序处理语句
- 当IF语句所指定的条件不满足时，将执行ELSE和END IF之间的顺序处理语句
- 用条件来选择两条不同程序执行的路径
- 描述的典型电路是二选一电路

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX2 IS
PORT（
        A，B，SEL：IN STD_LOGIC;
        C: OUT STD_LOGIC）;
END MUX2;
ARCHITECTURE BEHAVE OF MUX2 IS
BEGIN
        PROCESS（A，B，SEL）
        BEGIN
                IF（SEL = '1'）THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END BEHAVE;
```

格式 3

```
IF   条件   THEN
     ——顺序语句
ELSIF   条件   THEN
     ——顺序语句
          ……
ELSIF   条件   THEN
     ——顺序语句
ELSE
     ——顺序语句
END   IF；
```

- 设置了多个条件，当满足所设的多个条件之一时，就执行该条件后跟的顺序处理语句
- 如果所有设置都不满足时，则执行ELSE和END IF之间的顺序处理语句
- 描述的典型电路是多选一电路

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX4 IS
PORT（
        INPUT： IN STD_LOGIC_VECTOR（3 DOWNTO 0）;
        SEL： IN STD_LOGIC_VECTOR（1 DOWNTO 0）;
        C： OUT STD_LOGIC）;
END MUX4;
ARCHITECTURE BEHAVE OF MUX4 IS
BEGIN
        PROCESS（INPUT， SEL）
        BEGIN
                IF（SEL = “00”）THEN
                        C <= INPUT（0）;
                ELSIF（SEL = “01”）THEN
                        C <= INPUT（1）;
                ELSIF（SEL = “10”）THEN
                        C <= INPUT（2）;
                ELSE
                        C <= INPUT（3）;
                END IF;
        END PROCESS;
END BEHAVE;
```

顺序语句

4. CASE语句

- 从许多不同语句的序列中选择其中之一执行
- 常用来描述总线或编码译码的行为

CASE语句的格式

```
CASE   表达式   IS
    WHEN   条件表达式  =>
                        顺序语句;
END   CASE;
```

- 当CASE和IS之间的表达式的取值满足指定的条件表达式的值时，执行后跟的由符号 => 所指的顺序处理语句

```
ARCHITECTURE  BEHAVE  OF  MUX4  IS
        SIGNAL  SEL：  INTEGER；
BEGIN
        PROCESS（A，  B，  I0，  I1，  I2，  I3）
        BEGIN
                SEL <=  0；
                IF（A = ‘1’）THEN
                        SEL <=  SEL + 1；
                END  IF；
                IF（B = ‘1’）THEN
                        SEL <=  SEL + 2；
                END  IF；
                CASE  SEL  IS
                        WHEN  0  => Q <=  I0；
                        WHEN  1  => Q <=  I1；
                        WHEN  2  => Q <=  I2；
                        WHEN  3  => Q <=  I3；
                END  CASE；
        END  PROCESS；
END  BEHAVE；
```

四选一选择器

顺序语句

5. LOOP语句

- LOOP语句可以使程序能进行有规则的循环
- 重复模式有两种：WHILE和FOR

FOR循环

```
[标号]：FOR　循环变量　IN　离散范围　LOOP
        ——顺序语句；
END　LOOP　[标号]；
```

- 循环变量的值在每次循环中都将发生变化
- IN后跟的离散范围表示循环变量在循环过程中依次取值的范围

I是循环变量，它可取值1，2，…，9共9个值

```
ASUM：  FOR   I   IN   1   TO   9   LOOP
        SUM  =  I  +  SUM;
END   LOOP   ASUM;
```

实现1～9的累加运算

注：

- 循环变量在信号说明和变量说明中都不能出现
- 信号和变量也不能代入到循环变量中

```
LIBRARY   IEEE;
USE   IEEE.STD_LOGIC_1164.ALL;
ENTITY   PARITY   IS
PORT（
        INPUT ：IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
        OUTPUT ：OUT  STD_LOGIC）；
END   PARITY;
ARCHITECTURE  BEHAVE OF PARITY IS
BEGIN
        PROCESS(INPUT)
                VARIABLE TEMP : STD_LOGIC;
        BEGIN
                TEMP := ‘0’;
                FOR I IN 0 TO 7 LOOP
                        TEMP := TEMP XOR INPUT(I);
                END LOOP;
                OUTPUT <= TEMP;
        END PROCESS;
END BEHAVE;
```

8位奇偶校验电路

WHILE循环

```
[标号]：  WHILE   条件   LOOP
        ——顺序语句；
END   LOOP   [标号];
```

- 如果条件为“真”，则进行循环，如果条件为“假”，则结束循环

```
I ：= 1;
SUM ：= 0;
ABC：  WHILE（I < 1 0 ）LOOP
        SUM ：= I + SUM;
        I ：= I + 1;
END   LOOP   ABC;
```

循环控制变量I的递增是通过算式I：=I+1实现的

顺序语句

6. NEXT语句

- 在LOOP语句中NEXT语句用来跳出本次循环

格式

```
NEXT  [标号]  [WHEN条件];
```

- NEXT语句执行时将停止本次迭代，而进入到下一次新的迭代
- NEXT后跟的标号表明下一次迭代的起始位置，WHEN条件表明NEXT语句执行的条件
- 如果NEXT语句既无标号也无WHEN条件说明，那么只要执行到该语句，就立即无条件地跳出本次循环，从LOOP语句的起始位置进入下一次循环

```
FOR I IN 0 TO MAX LOOP
        IF（DONE（I） = TRUE）THEN
                NEXT;
        ELSE
                DONE（I）：= TRUE;
        END IF;
        Q（I） <= A（I） AND B（I）;
END LOOP;
```

顺序语句

7. EXIT语句

- LOOP语句中使用的循环控制语句
- 执行EXIT语句将结束循环状态，从LOOP语句中跳出，结束LOOP语句的正常执行

格式

```
EXIT  [标号]  [WHEN条件];
```

- 如果EXIT语句含有条件，条件为真时，从LOOP语句中跳出，条件为假时，则继续LOOP循环
- 如果EXIT语句含有标号，则跳到标号处继续执行

```
L1：  FOR  I  IN  10  DOWNTO  1  LOOP
        L2：  FOR  J  IN  0  TO  I  LOOP
                EXIT  L2  WHEN  I = J；
                MATRIX（I，J）：= I *（J + 1）；
        END  LOOP  L2；
END  LOOP  L1；
```

顺序语句

8. RETURN语句

- 是一段子程序结束后，返回主程序的控制语句
- 用来结束函数和过程的执行

格式

```
RETURN [表达式];
```

- 过程中的RETURN语句必须是无条件的，不能有表达式
- 函数语句中的RETURN语句必须有表达式 ,函数的结束必须使用RETURN语句

顺序语句

9. NULL语句

•表示无任何动作，执行NULL语句只是使程序走到下一个语句

格式

```
NULL;
```

•经常用在CASE语句中，用来表示在某种情况下不需要做任何动作

```
case controller_command is
        when forward => engage_motor_forward;
        when reverse => engage_motor_reverse;
        when idle => null;
end case;
```