第五章 PLC-2

三、数据类型和地址区

四、指令与编程

五、程序结构简介

三、数据类型和地址区

1. 基本数据类型

可支持二进制、十进制、十六进制、BCD码、ASCII码、时间、日期等格式的数据

常见的数据类型	描述	位数	常数举例	编程举例
BOOL	二进制位	1	TRUE/FALSE	I3.2
BYTE	字节	8	B#16#2F	IB3，由I3.0~I3.7组成的一个字节
WORD	无符号字	16	W#16#247D	MW100，MB100,MB101组成的字
DWORD	无符号双字	32	DW#16#149E857A	MD100
LWORD	长字	64	LW#16#ADAC1EF5	
INT	有符号整数	16	-362	-32768~32768
DINT	有符号长整数	32	L#23	-2147483648~2147483647
REAL	IEEE浮点数	32	20.1234	
S5TIME	SIMATIC时间	16	S5T#1H3M50S	
TIME	IEC时间	32	T#1H3M50S	
LTIME	IEC时间	64	T#11350d20h25m14s830ms652µs315ns	
CHAR	ASCII字符	8	‘2A’	
……				

2、常用的地址区

地址区域	可以访问的地址单位	符号及标识方法	说明	
过程映像输入区	位 · 字节 · 字 · 双字	I · IB · IW · ID	对应输入模块端口	
过程映像输出区	位 · 字节 · 字 · 双字	Q · QB · QW · QD	对应输出模块端口	
标志位存储区	位 · 字节 · 字 · 双字	M · MB · MW · MD	全局变量区	
数据块 自己定义的全局变量	位 · 字节 · 字 · 双字	DBX · DBB · DBW · DBD	用OPN DB打开	·不建议绝对地址寻址 ·确有需要，可修改数据块属性(非优化模式)
	位 · 字节 · 字 · 双字	DIX · DIB · DIW · DID	用OPN DI打开	
定时器	T	T		
计数器	C	C		
本地数据区	位 · 字节 · 字 · 双字	L · LB · LW · LD	局部变量	

3、输入/输出模块端口对应的I/O地址

每一通道均配置独立的地址，应用程序根据地址实现IO操作

开关量I/O模块 每个通道的地址占1位(1bit)，16通道模块的地址占2字节，32通道模块地址占4字节

模拟量I/O模块 每个模拟量地址为16位(2字节)，8通道模块的地址占16字节，2通道模块地址占4字节

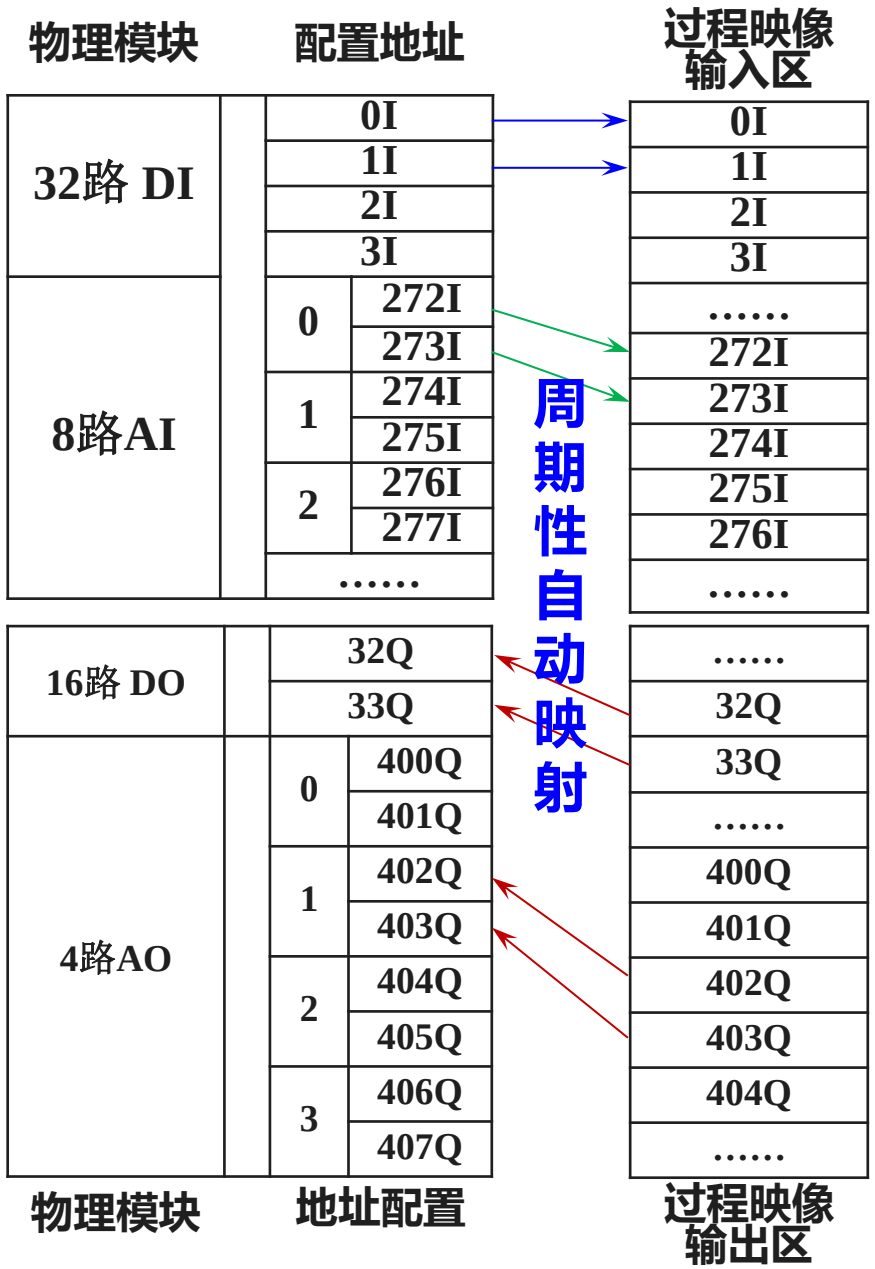
I/O地址生成方式： 在配置硬件时，系统自动提供缺省地址(推荐使用，通常地址是够用的)，也可手动设置地址

上图所示，组态了一个系统，包含：PM、CPU、DI 16路、DQ16路、AI 8路、AQ 4路

DI值范围： 0~32766
DQ值范围：0~32766
AI值范围： 0~32752
AQ值范围：0~32760

寻址范围足够大，一般场合都够用！

过程映像输入/输出示意：



读取通道0：A %I0.0
读取通道7：A %I0.7
……

读取通道0：L %IW272
读取通道1：L %IW274
……

输出至通道0：= %Q32.0
输出至通道1：= %Q32.1
……

输出至通道0：T %QW402
输出至通道3：T %QW406
……

注意事项：

- 输入、输出地址是不同的（I、Q）
- 模块地址在硬件配置时生成或指定
- 支持绝对地址访问，用 % 开头
- 支持变量名称访问，不容易搞错
- 建议自己定义变量名称，容易记
- 端口状态与映像区周期性自动映射
- 非必要建议优先对映像区读写
- 亦可直接、即读即写硬件端口
- 即读即写指令中应加:P

如 **L %IW272:P**

T %QW400:P


DI 16x24VDC HF_1 [DI 16x24VDC HF]
常规
IO 变量
系统常数
文本
常规
模块参数
输入 0 - 15
常规
组态概览
输入
I/O 地址
I/O 地址
输入地址
起始地址: 0 .0
结束地址: 1 .7
等时同步模式
组织块: --- (自动更新)
过程映像: 自动更新


DI 16x24VDC HF_1 [DI 16x24VDC HF]
常规
IO 变量
系统常数
文本
名称
类型
地址
Bool %I0.0
Bool %I0.1
Bool %I0.2
Bool %I0.3
Bool %I0.4
Bool %I0.5
Bool %I0.6
Bool %I0.7
Bool %I1.0
Bool %I1.1
Bool %I1.2

3. 数据块

对数据块的基本认识：

- 数据块是PLC最主要的数据存储区，需要用户自己定义，相当于C语言**全局变量**
- 主要存放IO结果、中间运算结果和状态、与其他设备或系统交互的数据等
- 块内数据可定义bool、byte、int、real、date、time等多种数据类型
- 数据块应先定义、后访问，可根据需要定义不同数据块
- 默认定义为优化的数据块，优化的数据块只能访问符号地址（变量名称，自己定义）
- 可通过更改块属性为标准的数据块，标准的数据块可访问绝对地址（哪种好用自己体会）
- 数据块分为共享数据块和背景数据块，本质都是全局变量
- 还支持ARRAY DB等特殊类型数据块， ARRAY DB块内仅包含一个数组
- 每个CPU允许定义的数据存储区大小、数量与CPU型号有关（一般都是够用的）

共享数据块定义： 定义的数据可以被任何块读写访问

背景数据块定义：附属于某个FB块、数据结构与FB输入输出数据格式完全相符的数据块，可以理解为FB批量赋值的数据块，也可以当作一般DB块用

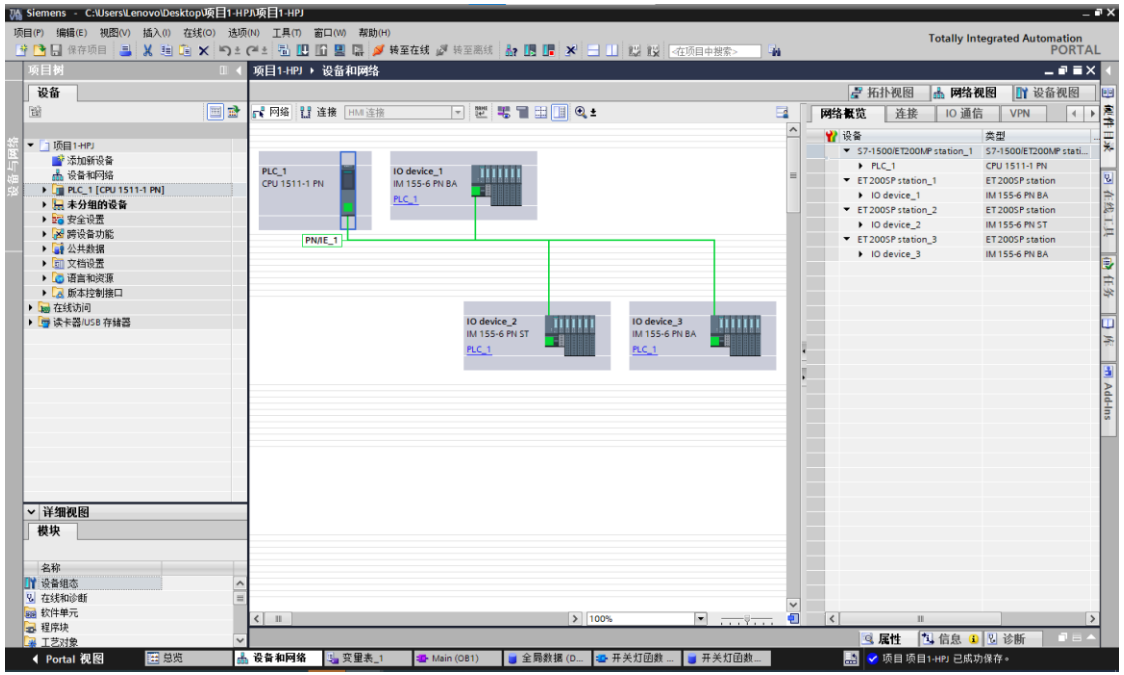
四、指令与编程

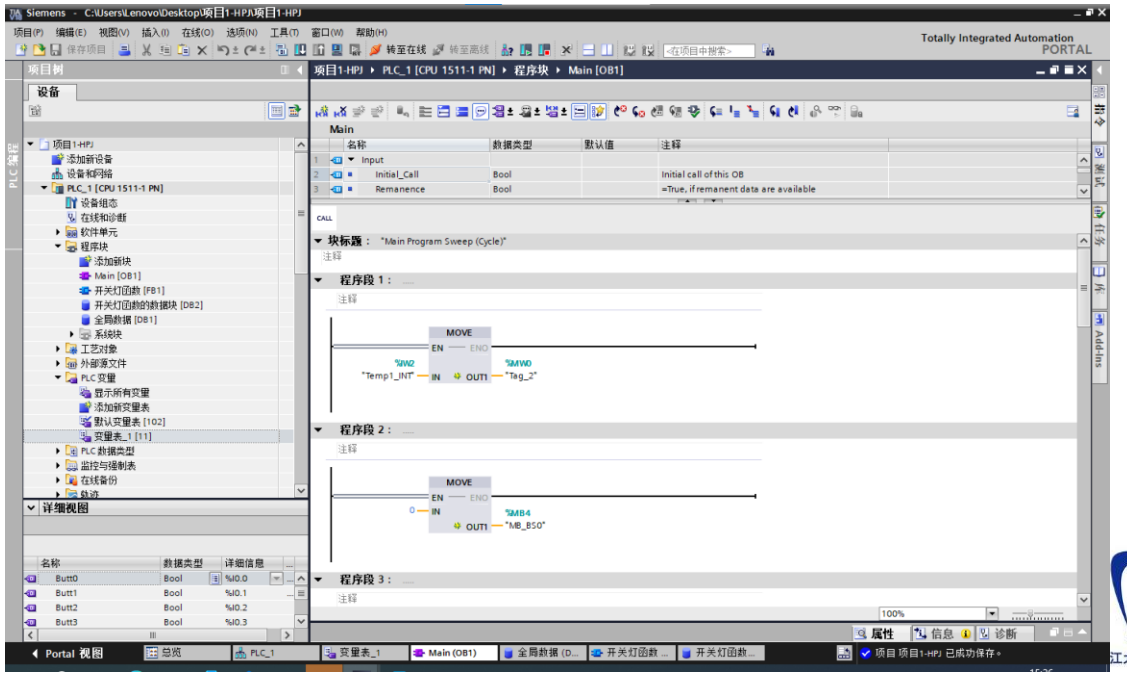
1. PLC的编程语言概述

IEC 61131-3：是由国际电工委员会IEC于1993年制定IEC 61131国际标准第三部分，用于规范PLC、DCS、IPC、CNC、SCADA的编程系统的标准，为工业自动化控制系统的软件设计提供**标准化编程语言**的国际标准。

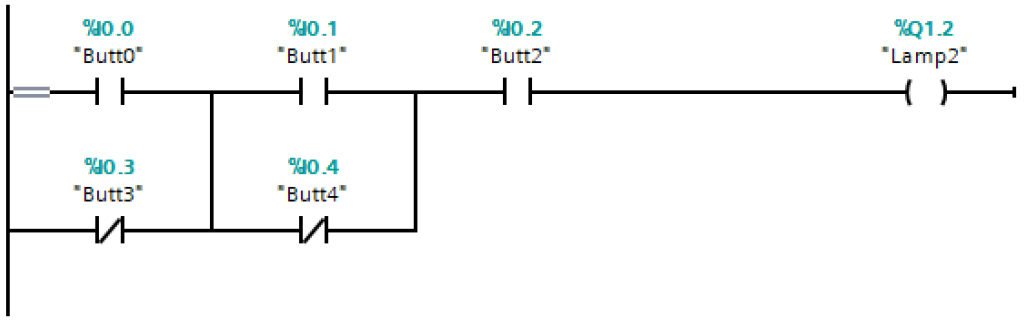
S7-1500系列PLC用户程序的开发软件包：TIA软件(博途)中的 STEP 7，符合IEC有关标准

编程语言包括：LAD(梯形图)、STL(语句表)、FBD(功能块图)、SCL(结构化语言，如类Pascal语言)、GRAPH(图表化语言)等，LAD和FBD可以互相转化，其他不可互相转化





不同编程语言的对比

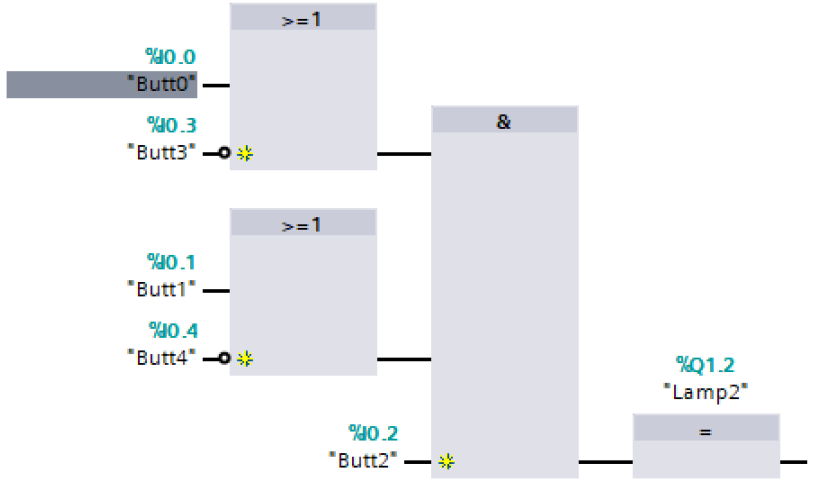


LAD

```
1        A(
2        O       "Butt0"         %I0.0
3        ON      "Butt3"         %I0.3
4        )
5        A(
6        O       "Butt1"         %I0.1
7        ON      "Butt4"         %I0.4
8        )
9        A       "Butt2"         %I0.2
10       =       "Lamp2"         %Q1.2
11
12
```

STL

```
A(
    O    "Butt0"
    ON   "Butt3"
    )
    A(
    O    "Butt1"
    ON   "Butt4"
    )
    A    "Butt2"
    =    "Lamp2"
```



FBD

程序段 6：

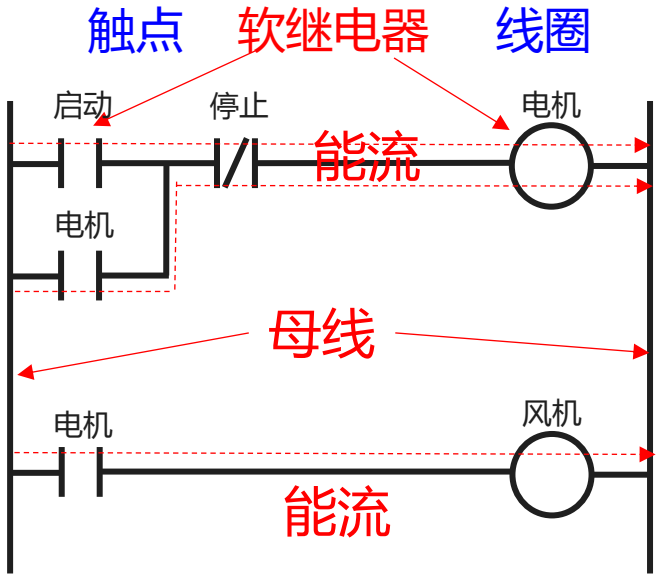
注释

```
IF ("Butt0" OR NOT "Butt3") AND ("Butt1" OR NOT "Butt4")AND "Butt2" THEN
    "Lamp2" := 1;
ELSE
    "Lamp2" := 0;
END_IF;
```

```
IF (“Butto” OR NOT “Butt3”) AND (“Butt1” OR NOT “Butt4”) AND “Butt2” THEN
   “Lamp2”:= 1;
ELSE
   “Lamp2”:=0;
END_IF;
```

SCL

LAD指令



软继电器：梯形图中某些编程元件沿用了继电器这一名称

能流：即“概念电流”，从左向右流动

母线：可想象左正右负的直流电源，右母线有时可以不画出

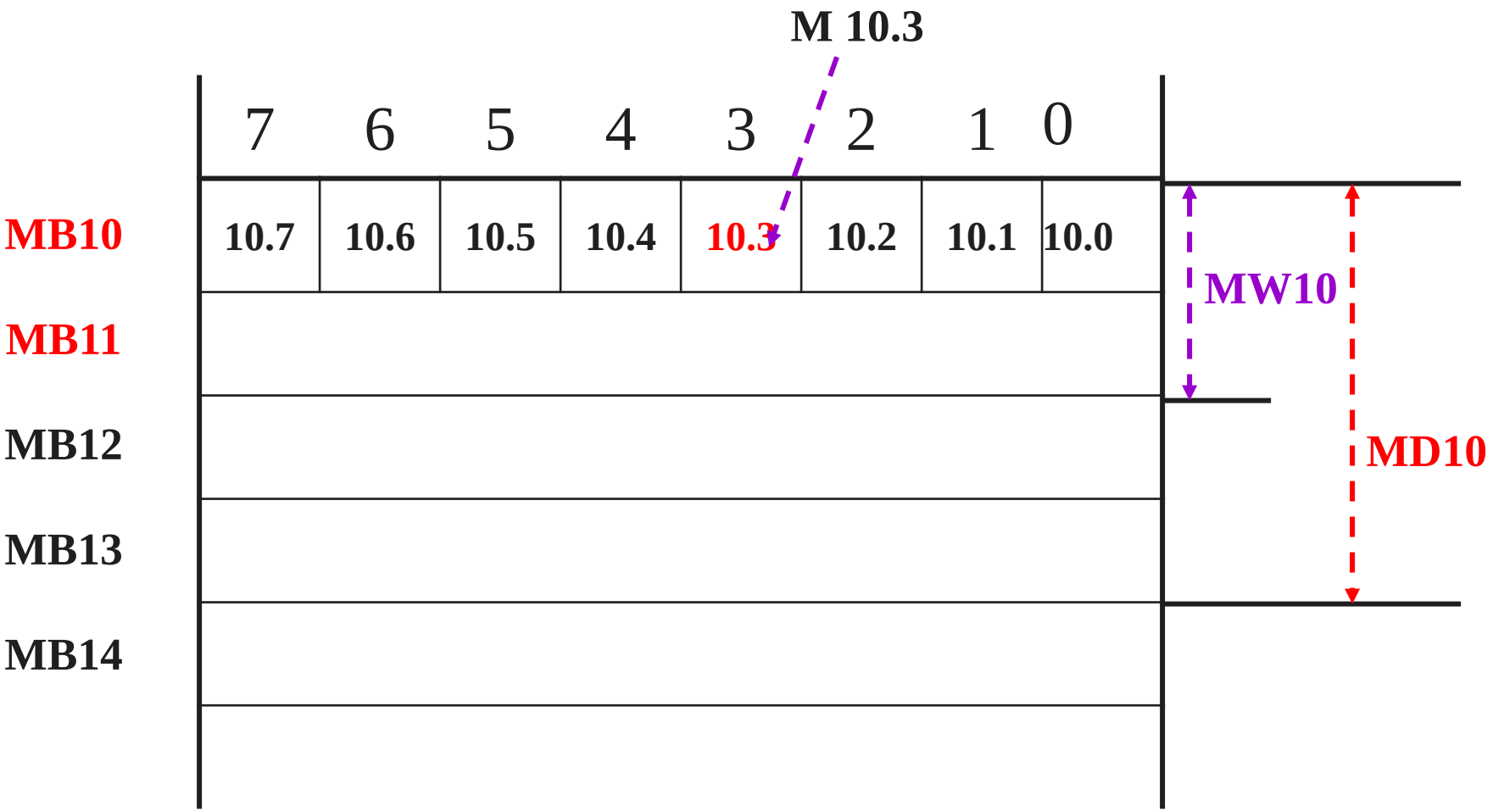
STL指令

语句指令：　操作码　操作数（≤1个）♠有些语句指令不带操作数，它们操作的对象是唯一的。

```
A    %I0.1     //对输入继电器 %I0.1（绝对地址） 进行与操作
L    %MW10     //将字%MW10装入累加器1
```

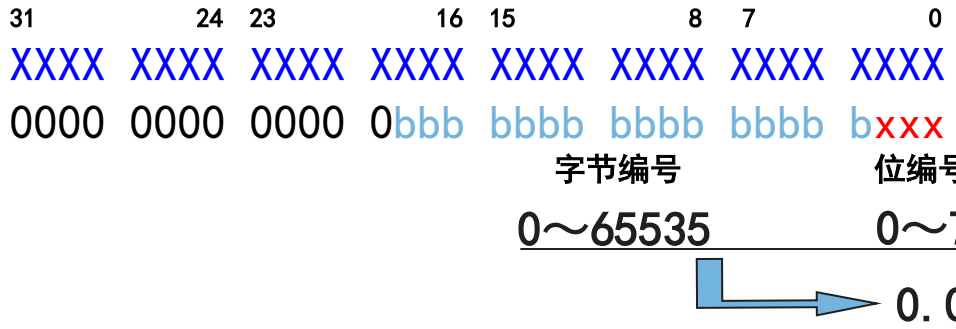
♠尽量使用符号地址！

♠有人认为指令语言会被淘汰！


M 10.3
7 6 5 4 3 2 1 0
MB10
10.7 10.6 10.5 10.4 10.3 10.2 10.1 10.0
MB11
MB12
MB13
MB14
MW10
MD10

位、字节、字、双字的关系

2、寻址方式



地址指针 P#x.y

- 双字地址指针访问字节、字、双字存储器时必须保证位编号为0
- 还有其他类型的地址指针

```
L    27
L    C#0100   // BCD码
```

立即寻址

操作数包含在指令中

```
A   %I1.2
L   %MB10
L   %MD12
```

直接寻址

在指令中直接给出操作数的存储地址

建议用符号地址

```
A   I [ DB1.DBD0 ] 如DB1.DBD0=P#1.2
L   MB [ MD16 ]    如MD16=P#10.0
L   MD [ MW20 ]    如MW20=12
```

存储器间接寻址

```
A    I[AR1，P#1.0]   //AR1+偏移量
=    Q[AR2，P#4.1]   //AR2+偏移量
```

寄存器间接寻址

多用于循环操作

3. 常用数值操作运算指令及间接寻址应用示例

功能	操作码	指令示例	说明
累加器数据装入	L	L 20	A1中原数据移入A2，把操作数内容（如20）装入A1
累加器数据传送	T	T MW6	把A1中内容传送到目标位置（如MW6），A1中原数据不变
地址寄存器地址装入	LAR1(LAR2)		将操作数装入AR1(AR2)，若指令中没有给出操作数，则将A1内容装入AR1(AR2)
地址寄存器地址传送	TAR1(TAR2)		将AR1(AR2)内容传给目标位置，若指令中没给出操作数，则将AR1(AR2)内容传给A1
地址寄存器地址交换	CAR		交换 AR1和 AR2的内容
比较指令	==、<> >、< >=、<=	>I >D >R	I为整型数比较(累加器中低16位) D为长整数比较 R为浮点数比较 如A2内容＞A1内容，则RLO置“1”，否则RLO置“0”。
数值运算 I：16位整数 D：32位长整数 R：32位浮点数	＋I +D +R	A2 ＋ A1	I：16位和 → A1低字；D或R：32位和 → A1
	－I －D －R	A2 － A1	I：16位差 → A1低字；D或R：32位差 → A1
	* I * D * R	A2 × A1	I：32位积 → A1；D或R：32位积 → A1
	/ I / D / R	A2 ÷ A1	I：16位商 → A1低字，余数→A1高字；D或R：32位商 → A1 （D相除无余数）
	ABS	对A1的32位实数取绝对值，32位结果 → A1	

```
L     %IW 10        //IW10的内容装载到A1的低字
L     %MW 14        //A1内容→A2，MW14的值→A1低字
+I                  //A1和A2低字的值相加，结果存在A1低字
T     %DB1.DBW25    //A1的低字中的运算结果传送到DB1的DBW25
```

寄存器间接寻址示例1

```
        L     P#0.0
        LAR1
        L     P#10.0
        LAR2
        L     64
n1:     T     #loopcounter      //把累加器A1内容放入临时变量loopcounter（首次执行值为64）
        OPN   %DB1              // 需在DB块属性中设置为不选取“优化的访问”
        CLR
        A     I [AR1,P#0.0]
        =     DBX [AR2,P#0.0]
        L     P#0.1
        +AR1
        L     P#0.1
        +AR2
        L     #loopcounter      //把loopcounter中的内容载入A1
        LOOP  n1                //A1自动减1，若A1不为0则循环到 n1，否则跳出循环
```

啥用?

把I0.0开始的64个连续位状态，通过循环方式逐个转存到从DB1.DBX10.0开始的连续64个位中

多个开关量信号的批量采样，批量存放到数据块中

注：用SCL语言编写一般会更简洁明了，效率更高

寄存器
间接寻址示例2

```
      L       P#400.0
      LAR1
      L       P#300.0
      LAR2
      L       32
n2:   T       #loopcounter
      OPN     %DB2
      L       DBD [AR1,P#0.0]
      T        #V_out
      CALL  “UNSCALE"
           IN          : =#V_out
           HI_LIM    : =1.000000e+002
           LO_LIM    : =0.000000e+000
           BIPOLAR : =FALSE
           RET_VAL : =#ret
           OUT         : =#V_dec
      L       #V_dec
      T       QW[AR2,P#0.0]
      L       P#4.0
      +AR1
      L       P#2.0
      +AR2
      L        #loopcounter
      LOOP     n2
```

作用：把从DB2.DBD400开始的连续32个0-100的模拟量，逐个通过UNSCALE函数转换为0-27648范围的某一个数值，并采用循环方式输出到地址从300.0开始的32个模拟量输出端口。

多个阀位信号的批量输出！

思考：如何用SCL语言编写？

4、位逻辑运算指令

常用的位逻辑运算指令

与 —— A

与非 —— AN

或 —— O

或非 —— ON

赋值 —— =

置位 —— S

复位 —— R

状态字中的两个关键状态位

15 … 8	7	6	5	4	3	2	1	0
……	BR	CC1	CC0	OS	OV	OR	STA	RLO	$\overline{FC}$

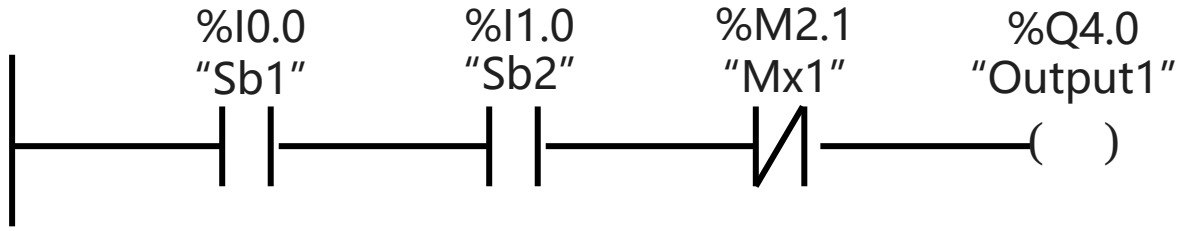
◆ **首次检测位($\overline{FC}$)**

- 若 $\overline{FC}=0$，表示新逻辑串开始，当执行位逻辑指令时，系统直接把操作数内容保存到RLO位中，并把$\overline{FC}$位置1；
- 若 $\overline{FC}=1$，操作数与RLO相运算，并把结果存于RLO
- 当执行到R、S、 = 等指令时，$\overline{FC}$清0（表示逻辑串结束）
- 通过$\overline{FC}$来区分不同的逻辑串！

◆ **逻辑操作结果(RLO，Result of Logic Operation)**

- 存储位逻辑指令或算术比较指令的结果
- 所有的逻辑运算结果均放在此处！！

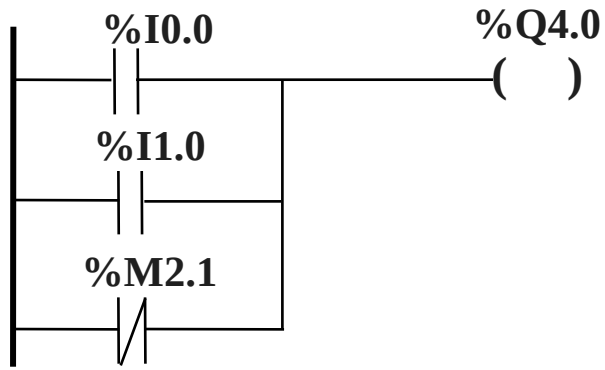
(1) 串联逻辑 A、AN



语句表		如实际状态	检测结果	RLO	$\overline{FC}$	说明
					0	下一条指令表示一新逻辑串的开始
A	"Sb1"	1	1	1	1	首次检测结果 → RLO，$\overline{FC}$置1
A	"Sb2"	1	1	1	1	检测结果与RLO "与" 运算 → RLO
AN	"Mx1"	0	1	1	1	检测结果与RLO "与" 运算 → RLO
=	"Output1"	1			0	RLO → Q4.0，$\overline{FC}$清0

取反　与　与

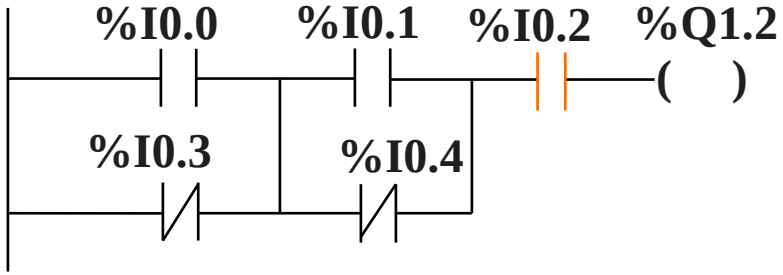
(2) 并联逻辑 O、ON



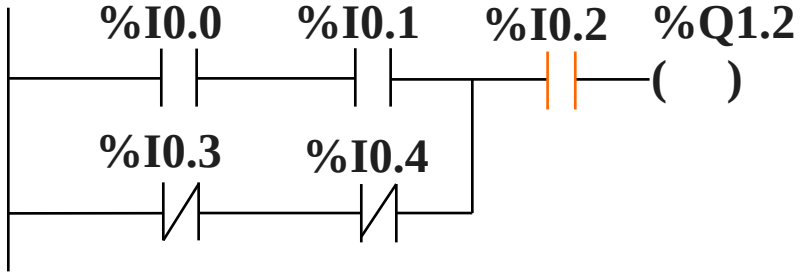

```
O     %I0.0
O     %I1.0
ON    %M2.1
=     %Q4.0
```

为可读性起见，用 "%+地址" 来表示，下同。
软件会自动转化成符号名称。

(3) 串、并联逻辑 及其 先“与”后“或”




```
A(
  O     %I0.0
  ON    %I0.3
 )
A(
  O     %I0.1
  ON    %I0.4
 )
A     %I0.2
 =    %Q1.2
```

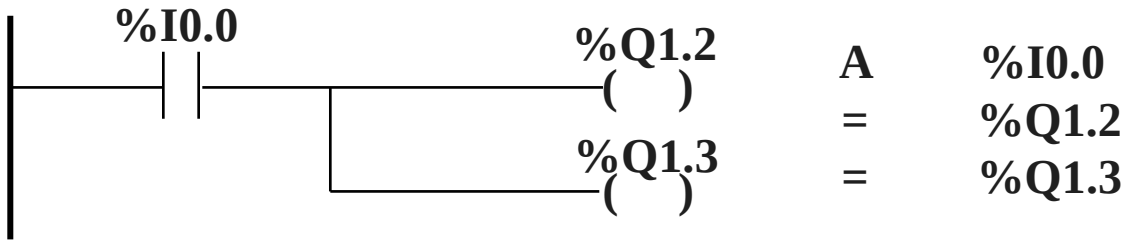



```
A(
  A     %I0.0
  A     % I0.1
 O
  AN    %I0.3
  AN    %I0.4
 )
 A      %I0.2
 =      %Q1.2
```

当逻辑串是串并联的复合组合时，CPU的扫描顺序是先“与”后“或”。

(4) 输出指令 =

- 该操作把状态字中RLO的值赋给指定的操作数（位地址）
- 把首次检测位（$\overline{FC}$位）置0，来结束一个逻辑串
- 一个RLO可以驱动多个输出元件



(5) 置位 S / 复位 R 指令

指令格式	指令示例	说明
S <位地址>	S %Q0.2	在执行该指令时，若RLO为1，则被寻址信号状态置1；若RLO为 0，输出不变。$\overline{FC}$ 清0。
R<位地址>	R %M1.2	在执行该指令时，若RLO为1，则被寻址信号状态置0；若RLO为 0，输出不变。$\overline{FC}$ 清0。

5、定时器　当满足特定条件时，完成有特定时间要求的工作

◆定时器概述（类型不少）

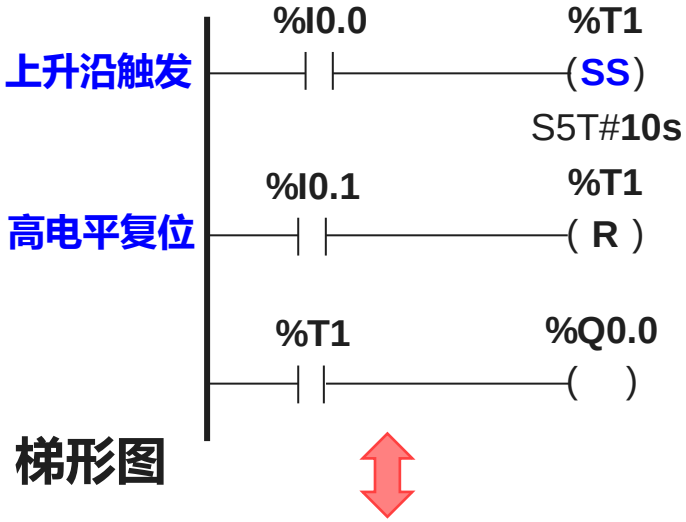
S5定时器

STL	LAD	LAD扩展形式	描述
SS	SS	S_ODTS	保持型接通延时定时器
SP	SP	S_PULSE	脉冲定时器
SE	SE	S_PEXT	扩展的脉冲定时器
SD	SD	S_ODT	接通延时定时器
SF	SF	S_OFFDT	断开延时定时器
R	R		定时器复位

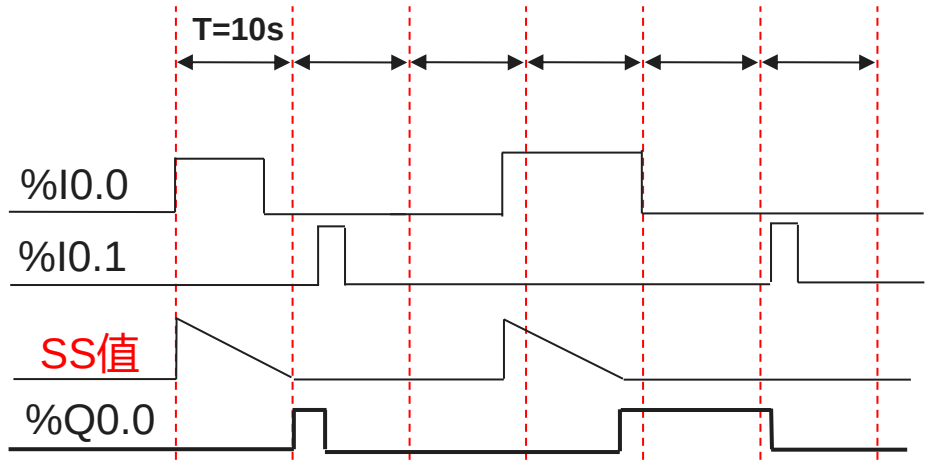
IEC定时器

LAD	描述
TP	启动脉冲定时器
TON	启动接通延时定时器
TONR	记录一个位信号为1的累计时间
RT	复位定时器
PT	加载定时时间

◆ S5定时器——SS 保持型延时接通定时器



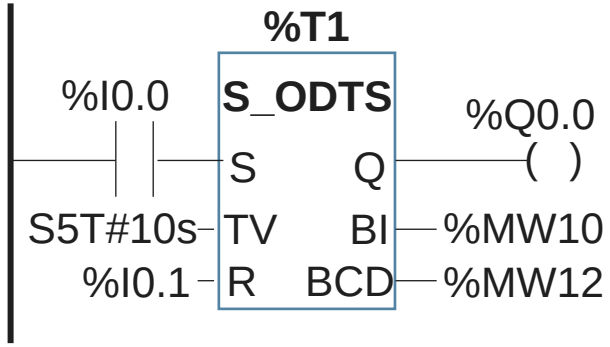
梯形图



- %I0.0由0→1，启动计时
- 计时时间＜T，输出0
- 计时时间≥T，输出1
- %I0.1高电平，定时器被复位
- **延时-接通-保持，直到被复位**

指令语言

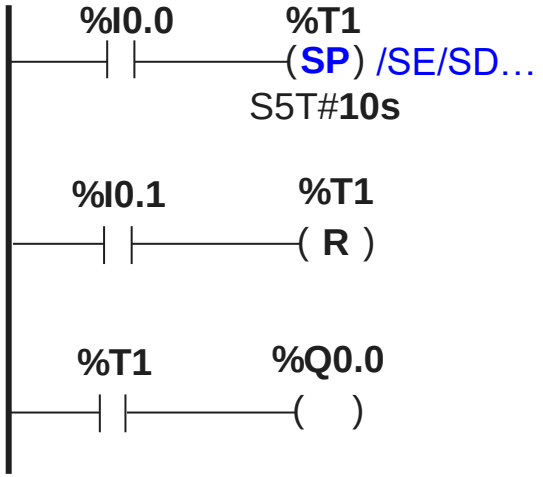
```
A     %I0.0
L     S5T#10s
SP    %T1
A     %I0.1
R     %T1
A     %T1
=     %Q0.0
```



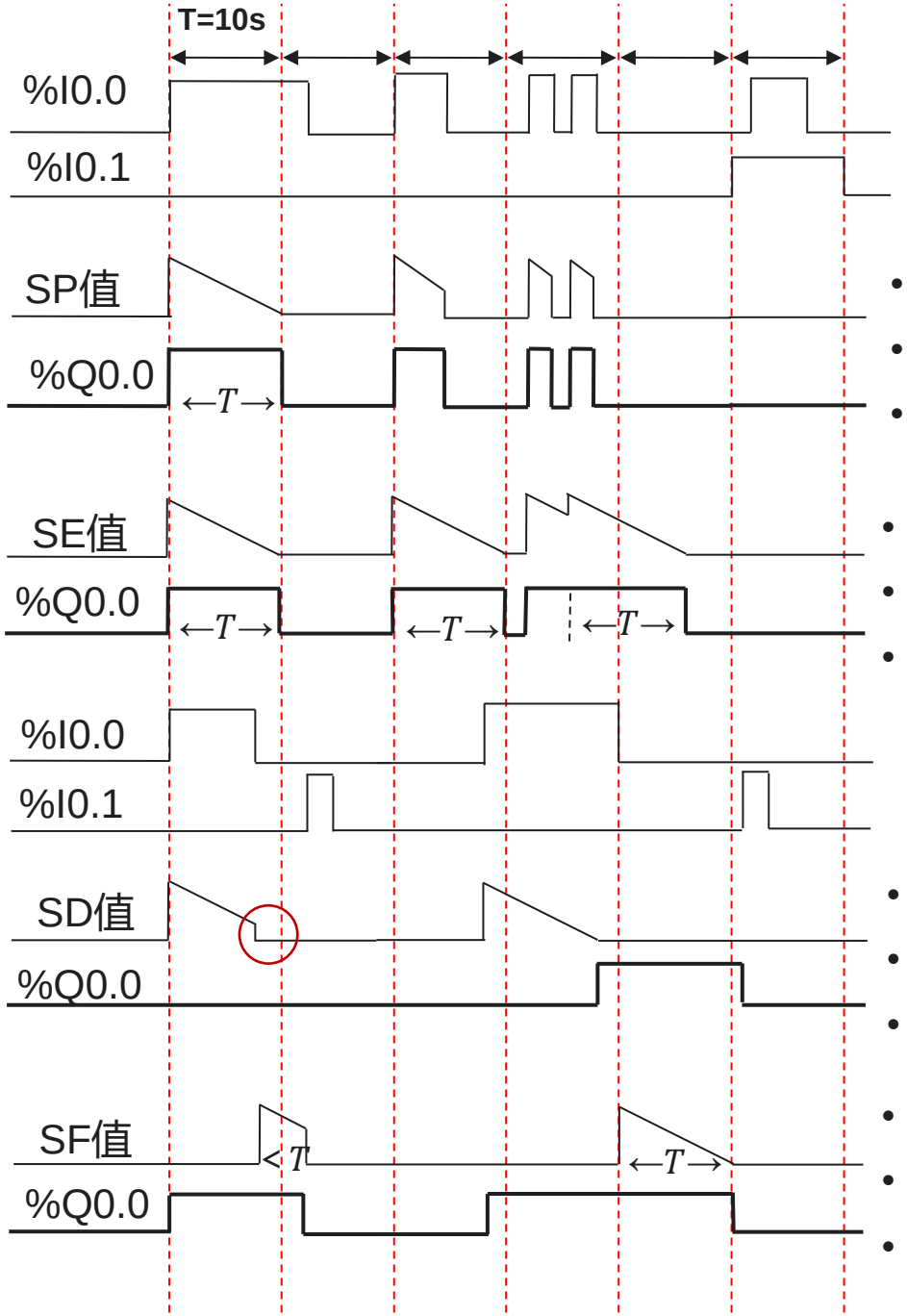
梯形图扩展形式

S: 使能输入
Q: 输出
R: 复位
TV: 预设的时间
BI: 十六进制的剩余时间
BCD: BCD格式的剩余时间

◆ S5其他定时器



S5定时器的触发和复位是分开的



- %I0.0 **1**，输出**1**
- 输出时长 ≤ T，**与%I0.0保持1的时间有关**
- %I0.1高电平复位，与%I0.0的状态无关

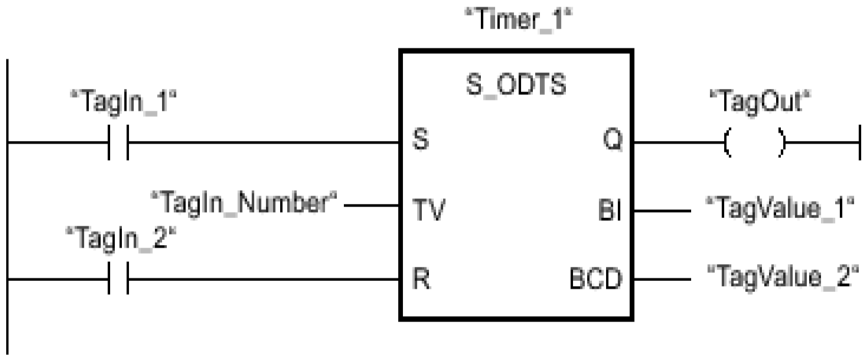
- %I0.0 **1**，输出**1**
- 输出时长≥T，**可被叠加触发**
- %I0.1高电平复位，与%I0.0的状态无关

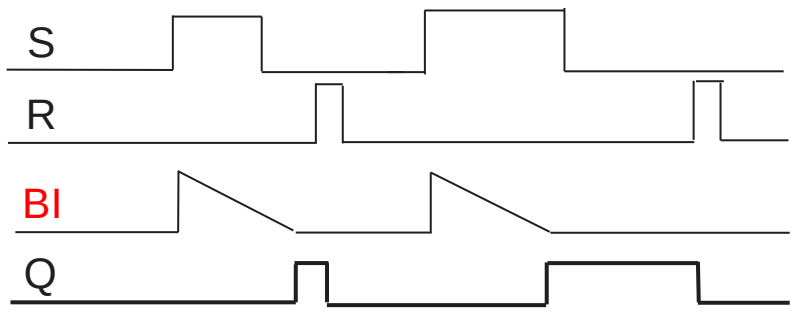
- %I0.0 **1** ＜T，输出**0**
- %I0.0 **1** ≥T，**在T后输出1**
- %I0.1高电平时复位，与%I0.0的状态无关

- %I0.0 **1**，输出**1**
- %I0.0由**1**→**0**，延时计时并输出**1**，最长延时为T
- %I0.1高电平时复位，与%I0.0的状态无关

◆两个定时器的比较：

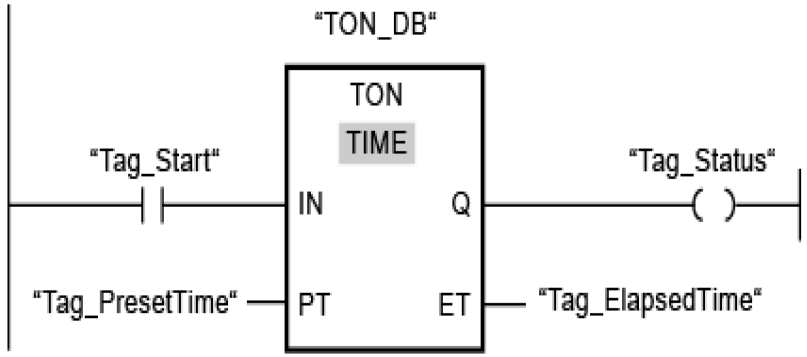
S5 定时器： SS(S_ODTS)保持型接通延时定时器

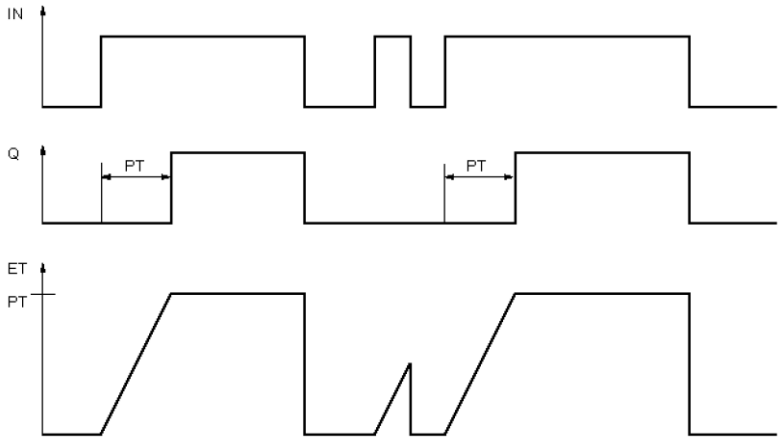




- S由0→1，启动计时
- 计时时间＜T，输出0
- 计时时间≥T，输出1
- R高电平，定时器被复位
- **延时-接通-保持，直到被复位**

IEC 定时器： TON接通延时定时器





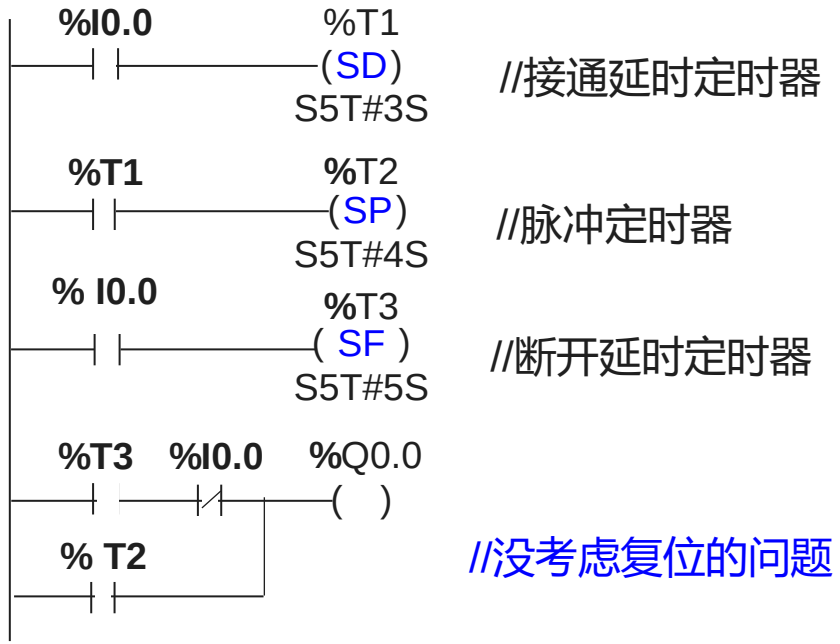
- IN由0→1，定时器开始计时
- IN 1 ≥PT，在PT后Q输出1
- IN由1→0，Q复位输出为0
- IN 1 ≤PT，Q输出不能从0变为1

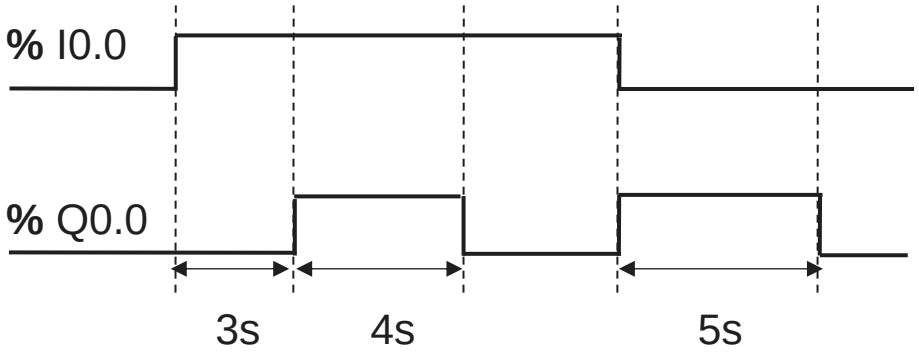
两个定时器功能相似，根据需要选用，要求掌握

◆课堂训练题：定时器的组合应用

定时器的种类较多，如果能巧妙地应用各种定时器，可以简化电路，方便地实现较复杂的时序控制功能。

控制要求：某冲水控制系统，当检测到有使用者（%I0.0）时，触发定时器T1，延时3秒钟后打开电磁阀（%Q0.0）冲水4秒钟；当使用者离开时，冲水电磁阀冲水5秒钟，然后停止冲水。

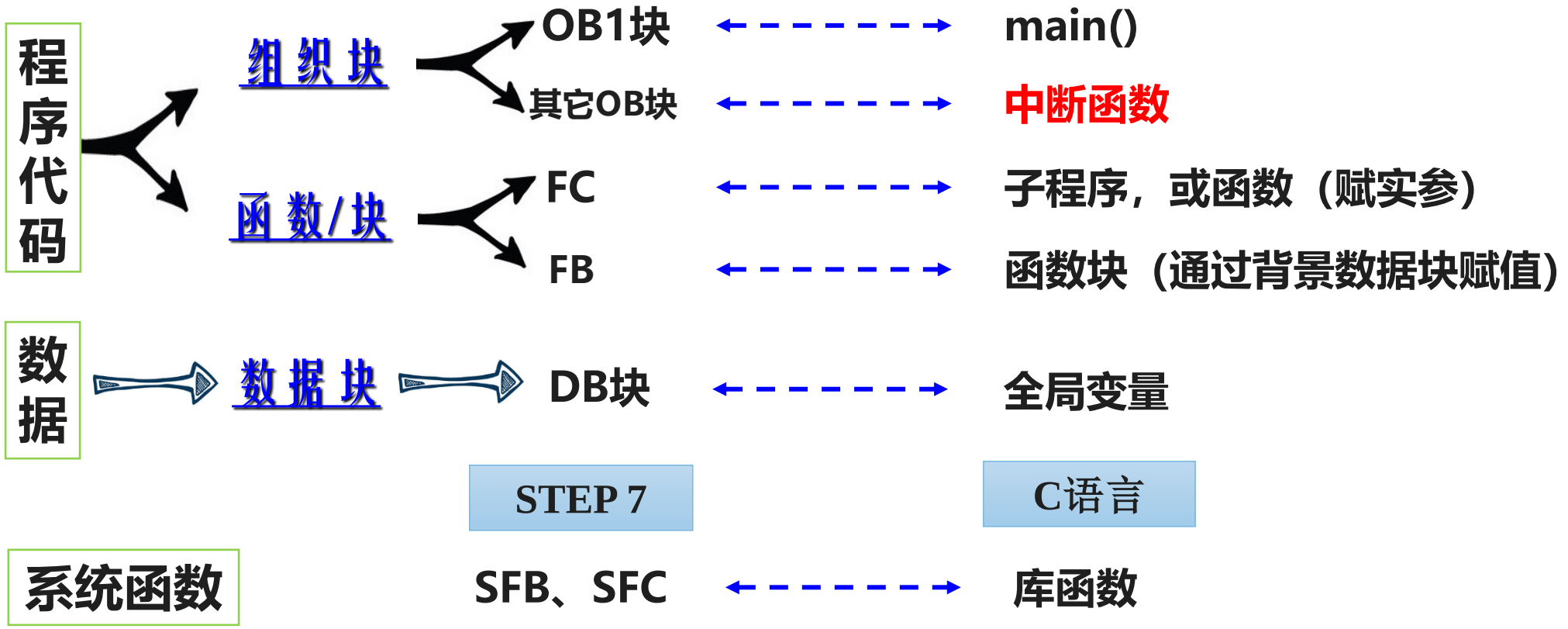




思考：如需%I0.0触发并持续3秒后%Q0.0才正式认为需要动作(防止人经过就触发），该怎么处理?

五、程序结构简介

1、程序结构概述


程序代码
组织块
OB1块
main()
其它OB块
中断函数
函数/块
FC
子程序，或函数（赋实参）
FB
函数块（通过背景数据块赋值）
数据
数据块
DB块
全局变量
STEP 7
C语言
系统函数
SFB、SFC
库函数
结构化编程的“块”

2、组织块（Organization Block）及中断优先级

- **组织块**（OB块）由事件触发，执行编写在OB块中的程序
- 设置不同的组织块，用以完成不同的功能
- 每一个OB块都对应为一种有不同优先级的中断，优先级从1（最低）到26（最高）
- 可把OB块设成优化或非优化属性，非优化属性可访问绝对地址，否则仅能访问符号地址
- 组织块可以编写的最大持续容量与CPU型号有关（一般够用）

部分常用OB块的描述：

OB块	说明	OB编号	优先级（默认）	可能的OB个数	备注
启动	系统从停止切换到运行状态时调用，进行初始化	100，≥123	1（1）	0-100	多个启动OB按编号由小到大依次执行
程序循环★	基本组织块，每循环周期执行1次	1，≥123	1（1）	0-100	同上
时间中断	根据设置的日期、时间定时启动执行	10-17，≥123	2-24（2）	0-20	
循环中断★	按照设定的时间间隔循环执行	30-38，≥123	2-24（8-17）	0-20	OB块中程序执行时间应小于时间间隔
硬件中断	具有硬件中断能力的硬件触发的中断响应，如AI模块	40-47，≥123	2-26（18）	0-50	不同的硬件中断可触发同一硬件中断OB
机架错误	当ProfiBus或ProfiNet系统分布式IO错误时执行	86	2-26（6）	0或1	
IO访问错误	执行程序指令期间直接访问IO数据出错	122	2-26（7）	0或1	

……

2、函数（Function）

分为无形参的子程序和有形参的函数，有形参函数被调用时应逐一赋实参，类似于C#中的方法

可定义的参数类型：

Input	只读	被调用时应赋实参
Output	只写	被调用时应赋实参
InOut	读写	被调用时应赋实参
Temp	内部临时变量	
Constant	常量符号	

形参接口区

Address	Declaration	Name	Type	Start value	Comment
0.0	in	a1	INT		
2.0	in	a2	INT		
4.0	out	b1	REAL		
8.0	in_out	c1	REAL		
0.0	temp	d1	INT		

应用程序

```
FC18 : 说明示例
Comment:
Network 1: Title:
Comment:
      A     I     0.0
      A     I     0.1
      =     Q     0.1
```

3、函数块（Function Block）

带很多形参的函数，被调用时通过数据块赋实参，类似于C#中的类

可定义的参数类型：

Input	只读	被调用时可赋实参	可用背景数据块整体赋值
Output	只写	被调用时可赋实参	
InOut	读写	被调用时可赋实参	
Stat	静态变量	存储中间过程值	
Temp	内部临时变量		
Constant	常量符号		

形参接口区

Address	Declaration	Name	Type	Start value	Comment
0.0	in	a1	INT	0	
2.0	in	a2	INT	0	
4.0	out	b1	REAL	0.000000e+0	
8.0	in_out	c1	REAL	0.000000e+0	
12.0	stat	e1	BOOL	FALSE	
13.0	stat	e2	BYTE	B#16#0	
0.0	temp	d1	INT		

应用程序

```
FB4 : Title:
Comment:
Network 1: Title:
Comment:
      A     I     0.1
      A     M     0.1
      =     Q     0.1
```

什么场合可以设计FB来提高程序效率？

浙江大学控制科学与工程学院

FC、FB的调用示例

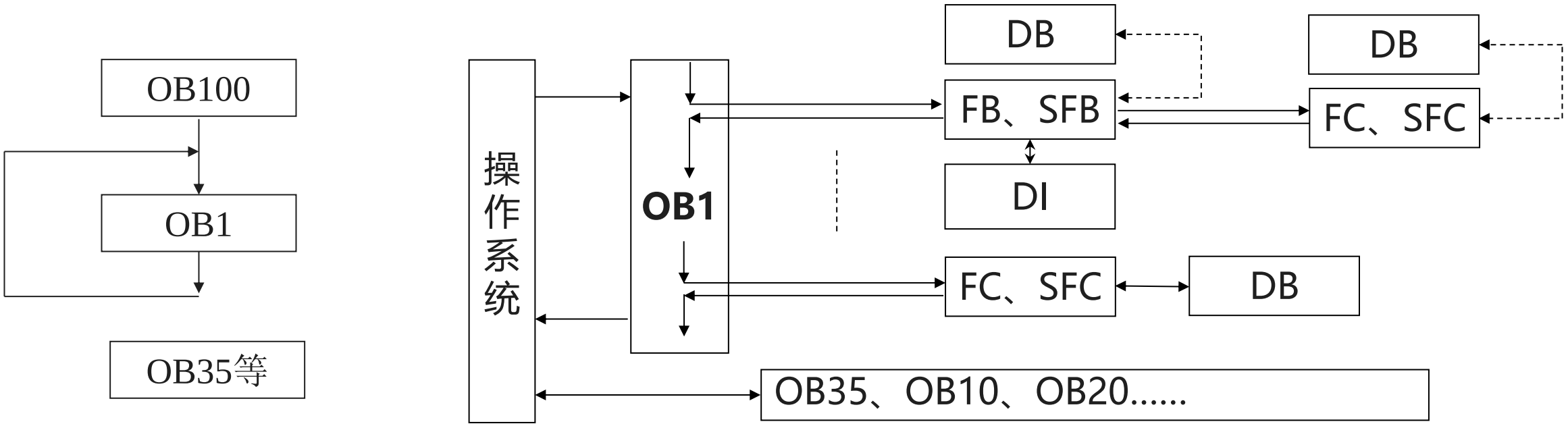
- FB块的调用： CALL %FB4 , %DB33
  a1:=
  a2:=
  b1:=
  c1:=

  • DB33应该是通过FB4派生出的数据块，除temp变量之外，块内数据结构与FB4变量声明表完全相同

  • 所有参数、变量都可通过背景数据块赋值

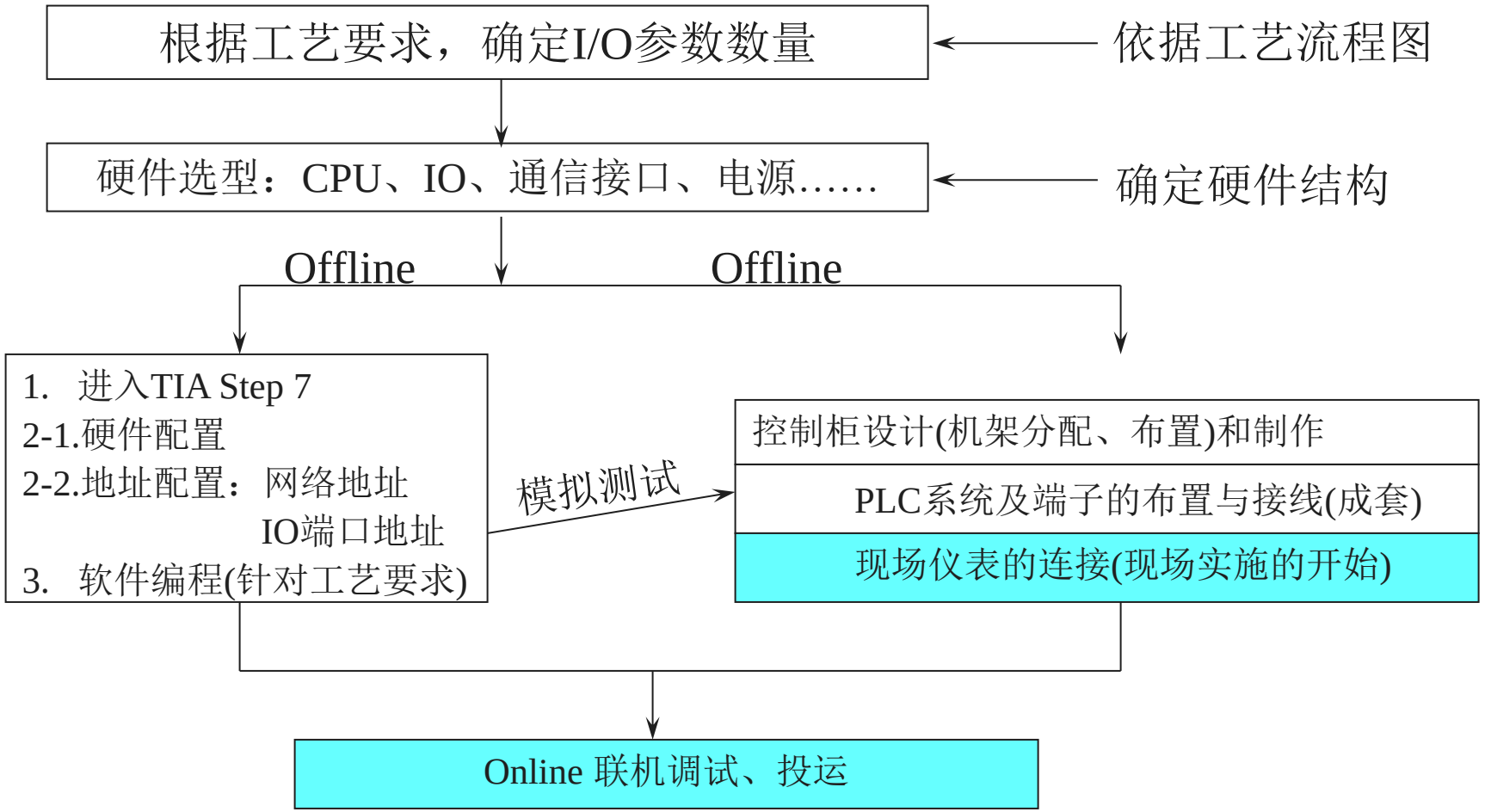
- FC块是调用： CALL %FC1　　调用FC时应逐一赋实参，实参数据类型应与形参相同
  a1:= %DB1.DBD0.0
  a2:= %DB2.DBW6.0
  b1:= %DB10.DBX5.6
  c1:= %MW12

4、逻辑块的调用关系



- OB块只能被系统调用（被事件触发），不可以被其他程序调用
- OB块可以调用FB或FC
- FB或FC可以调用其他FB、FC
- 系统为每个优先级分配64KB的临时变量（L堆栈），供组织块、程序块使用
- 嵌套越深，占用L堆栈越多，防止用“爆”

5、PLC系统开发的基本流程



课外作业：

把I300开始的32个连续模拟量信号（对应0-200kPa压力变送器输入），通过寄存器间接寻址以循环方式逐个转存到从DB1.DBD0开始的连续32个real变量中。

Thank You