TwinCAT PLC Control

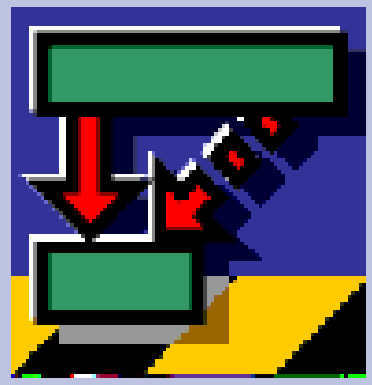
IEC 61131-3 编程

TwinCAT

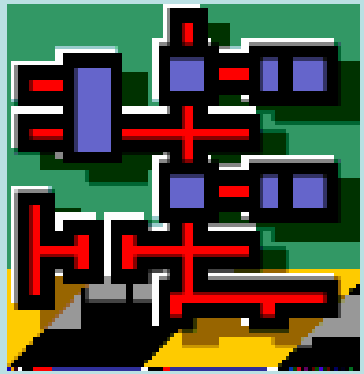
Total Windows Control and Automation Technology

TwinCAT Real Time

TwinCAT System Manager



TwinCAT PLC



TwinCAT NC/CNC

IEC 61131-3的优势

IEC（International Electrotechnical Commission）61131-3是 IEC 61131国际标准的第三部分,是第一个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。

- 国际上承认的标准
  - 逐步的在不远的将来所有供应商将采用它
  - 统一的结构，语言和操作处理方式将来自所有供应商
- 它节省你的时间
  - 统一的软件模式和数据类型概念
  - 对来自不同的**PLC**类型你只需学习一次
  - 减少了误解和错误
  - 标准的函数和功能块
  - 测试软件的可重复使用性

IEC 61131-3的优势

- 支持安全和高质量编程设计
  - 轻松和舒适的结构
  - 数据类型避免了编程错误
- 对每个问题提供了最佳编程语言
  - 一致的 **5** 种编程语言规范
  - 文本和图形语言
  - 高级语言的可用性
  - 不同语言混合编程

PLCopen 组织

PLCopen国际组织是一个独立于制造商和产品的国际组织，总部位于荷兰。致力于IEC 61131标准的推广并取得了很大成功。

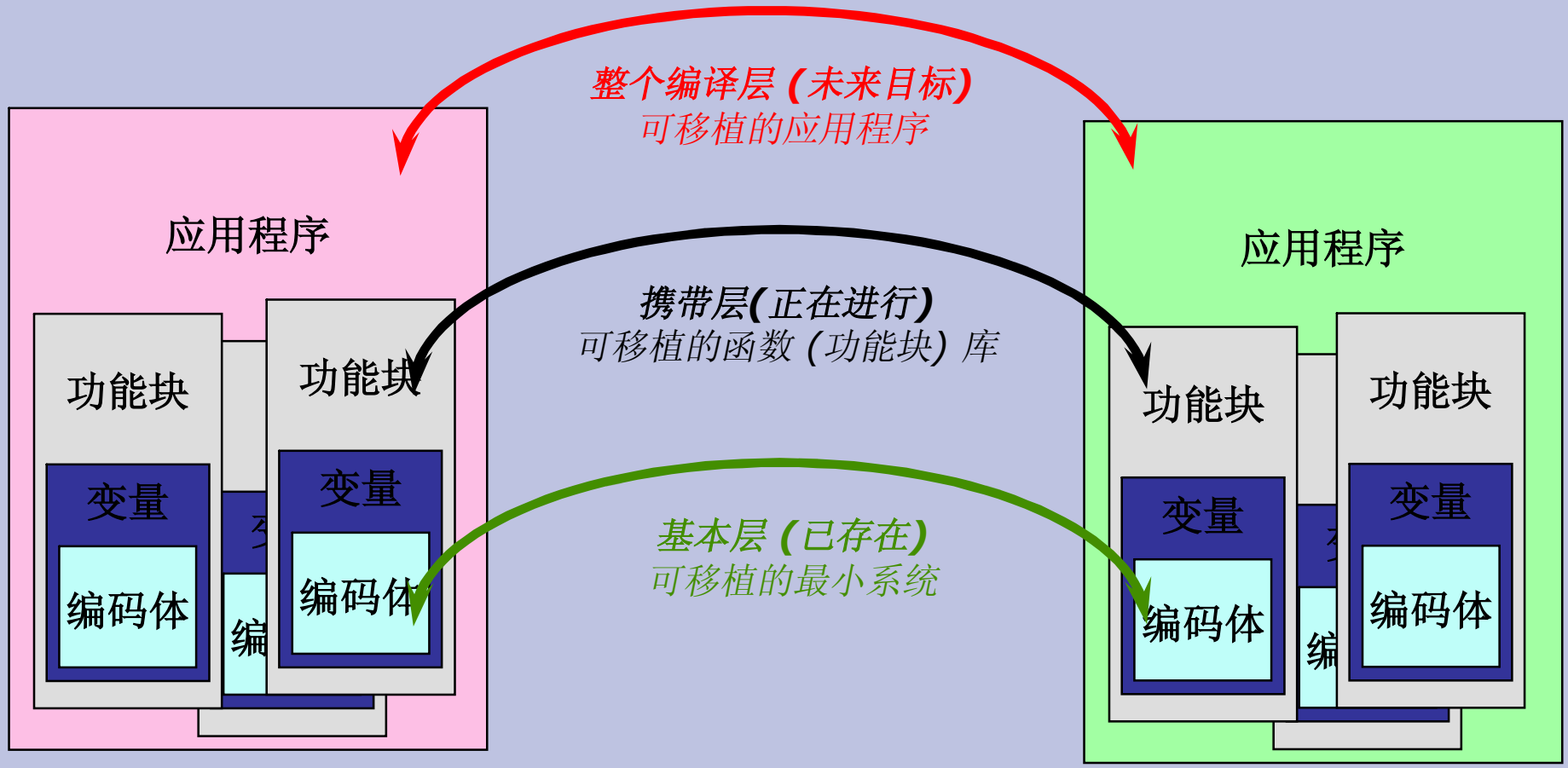
- **PLCopen是使PLC软件不依靠于供应商和独立于产品的世界组织. 它通过发布和强化IEC 61131-3 软件开发标准, 给工业控制系统的用户带来很大的价值.**
- **IEC 61131 标准给出了可依据的准则**
- **资格证书给用户提供了通向真正 IEC 61131-3 编程系统的引导 (例如 PLC开放资格表列出符合该标准的产品)**

PLCopen 组织

- 为消除混乱, **PLCopen....**

.... 已规定了**3** 层具有特性建立的编译
.... 已规定了鉴定资格程序
.... 有了资格测试协会
.... 开发好的测试软件, 在成员中共享
.... 已规定了证书程序
.... 并有了已被鉴定产品的成员

PLCopen 组织


整个编译层 (未来目标)
可移植的应用程序
携带层(正在进行)
可移植的函数 (功能块) 库
基本层 (已存在)
可移植的最小系统
应用程序
功能块
变量
编码体
功能块
变量
编码体
编
应用程序
功能块
变量
编码体
功能块
变量
编码体
编

IEC 61131 概况

- IEC61131-1 通用信息(1992)
- IEC61131-2 装置要求与测试(1992)
- IEC61131-3 编程语言(1993)
- IEC61131-4 用户导则(1995)
- IEC61131-5 通信服务规范(2000)
- IEC61131-7 模糊控制编程软件工具实施(2000)
- IEC61131-8 IEC61131-3语言实现导则(2001)
- 2000年8月由PLCopen组织向IEC提出增加IEC61131-X 功能安全性，目前尚处在草案阶段。

IEC 61131 标准 第1部

通用信息

- 在这个标准中的定义和术语
- 相关 / 参照 **IEC** 标准的列表
- 可编程控制器系统的主要功能特性

IEC 61131 标准 第2部

装置要求与测试

- 对可编程控制器和关联外设的电气,机械和功能要求
- 服务, 储存和运输条件
- 厂商提供的资料
- 为确认可编程控制器和关联外设资格的测试方法和程序

IEC 61131 标准 第3部

编程语言

- 软件-, 通讯- 和 编程-模式
- **5** 种内部连接编程语言的定义
- 两种文本和两种图形语言的语法和语句:
  指令表 **(IL)**, 结构化文本 **(ST)**,
  梯形图 **(LD)** 和功能块图**(FBD)**
- 顺序功能图 **(SFC)** -为组建程序结构

IEC 61131 标准 第4部

用户导则

帮助用户在:

- 利用可编程控制器标准的其它部分
- 为应用程序详细说明需求
- 选择和实现系统

IEC 61131 标准 第5部

通信服务规范

- 基于 **MMS (制造商信息规范)**

IEC 61131 主要特性

良好的结构

- 从上至下或从底向上的编程
- 以程序组织单元为单位 **(POUs)**
  Program Organization Unit
- 逐级构建程序

IEC 61131 主要特性

强大的数据类型测定

- 编译器探测不同数据类型的分配
- 减少编程错误

IEC 61131 主要特性

全执行控制

- 不同的任务具有不同的优先级和不同的**PLC**循环时间

IEC 61131 主要特性

复杂的流程控制

- 功能流程图
- 具有步骤，动作和转移的流序
- 可选分枝和同步分枝流程

IEC 61131 主要特性

数据结构

- 用户定义数据元素
- 包含不同的基本的或用户定义的数据元素
- 通过自己的**POU**传递数据结构

IEC 61131 主要特性

灵活的语言选择

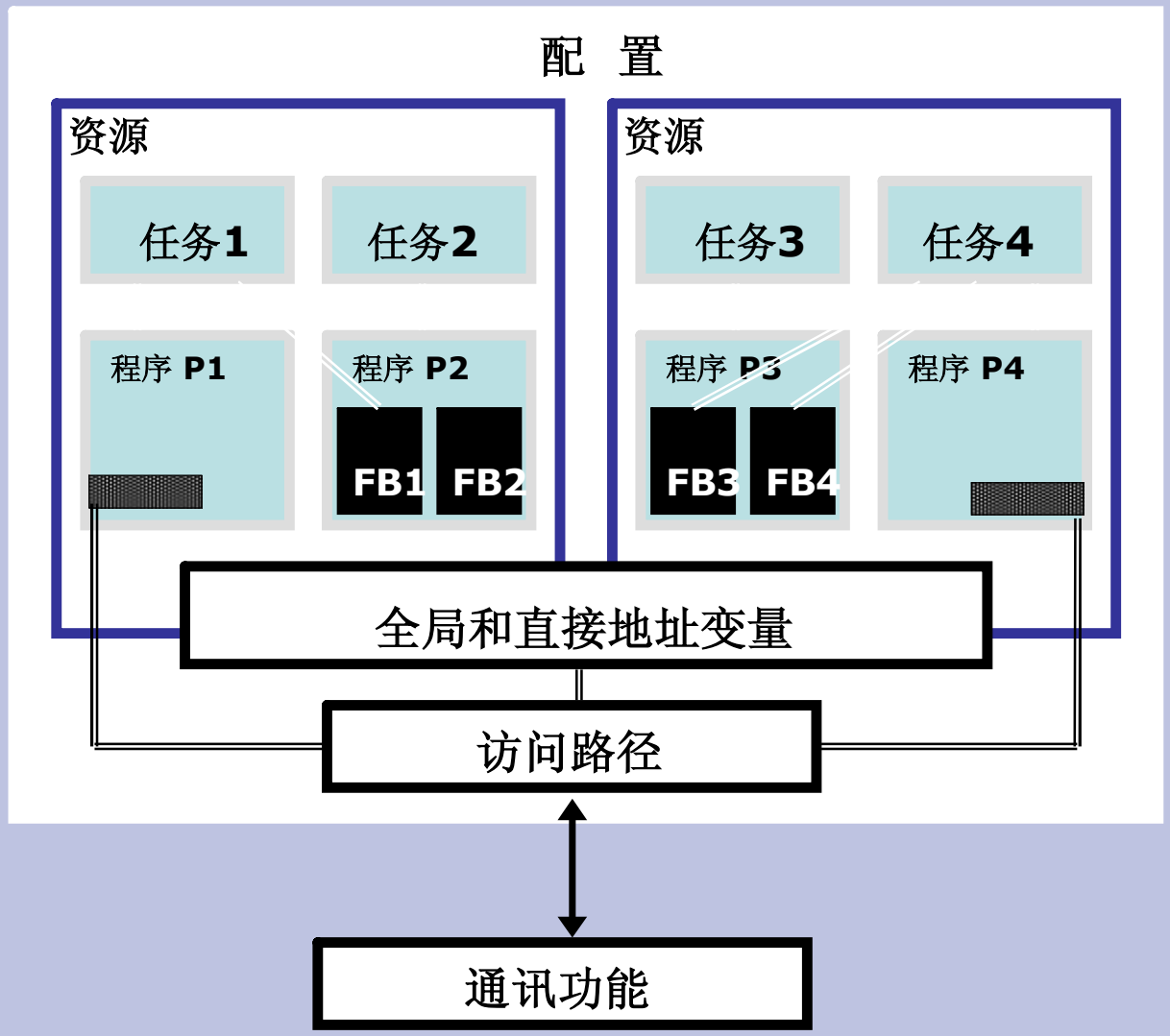
- 两种文本语言（**IL,ST**）
- 三种图形语言 **(FBD，LD，SFC)**
- 针对问题选择适当的语言

IEC 61131 主要特性

独立于供应商的软件

- 服从 **IEC 61131-3**标准
- 基本层鉴定 **(PLCopen)**
- 输入/输出接口

IEC 61131 软件模型



IEC 61131 软件模型

配置 Configuration

- 最高层，描述了整个控制系统的架构。
- 一个配置可被比作一个可编程控制器系统。
- 在一个配置里可以定义一个或几个资源。
- 在**TwinCAT** 中 一个或多个**PLC**

IEC 61131 软件模型

资源 Resource

- 一个配置有一个或多个资源
- **“实质” PLC:** 自己的全局变量, **POU,** 任务等.
- 在 **TwinCAT:** 就是一个 **PLC** 运行核 **(Runtime)**

IEC 61131 软件模型

任务 **Task**

- 对一个相关程序的周期的执行,实施控制单元
- 一个资源有一个或多个任务
- 有优先级时序排列（**0~3**，共四个级别）
- 优先权和循环时间
- 任务调用一个或多个程序
- 任务决定了所关联程序的时间调度。

IEC 61131 软件模型

程序 **Program**

·程序，是根据控制器过程的需要，包含了函数和功能块的一个逻辑组合的**POU**。

·任务调用程序

·程序调用功能块和函数

·**TwinCAT:** 程序调用其它程序

IEC 61131 软件模型

功能块 (FB) Function Block

- 程序调用功能块
- 功能块可调用功能块或函数
- **FB** 有输入, 输出变量
- **FB** 有运算法则:每次**FB**被执行,就是运行一段程序编码

IEC 61131 软件模型

函数 Function

- 程序或功能块可调用函数
- 函数有输入变量, 和一个输出变量
- 函数有运算法则:每次函数被执行,就是运行一段程序编码
- 函数可以调用另外的函数，但不能调用功能块

IEC 61131 软件模型

功能块和函数之间的区别

- **FB:** 例程, 全部数据分配内存地址
- 函数: 没有指定的内存分配地址
- **FB:** 多个输出变量或没有输出变量
- 函数: 一个输出变量
- **FB:** 可调用功能块或函数
- 函数:可调用函数，但不能调用功能块

IEC 61131 软件模型

局部变量

- 变量在一个**POU (程序, 功能块或函数)**中定义说明.
- 只能在这个**POU**中访问

IEC 61131 软件模型

全局变量

- 变量在一个资源(**PLC** 运行核**-runtime**)中定义说明
- 每个**POU**均可访问

IEC 61131 软件模型

直接描述变量

- 具有固定地址的变量
- 地址：输入 **(I)**, 输出 **(Q)** , 标记 **(M)**
- 类型：位 **(X)**, 字节 **(B)**, 字 **(W)** , 双字 **(D)**
- 例如： **%IW12, %QX1.1, %MB5**

IEC 61131 软件模型

"冷" 启动

- 所有变量被初始化
- 默认初始值或用户定义的初始值被分配给所有变量
- 使能所有任务，开始执行任务

IEC 61131 软件模型

"热" 启动

- 不进行变量初始化
- 原值被使用
- 使能所有任务，开始执行任务

IEC 61131 共有特性

字符集限制

- 字母不分大小写 **(abc = Abc = ABC)**
- 语法关键字区分大小写, 总是大写字母
- 不能使用国家特有字符

IEC 61131 共有特性

标识符

- 首字符不能是数字
- 不能连续使用多于一个的下划线 (_)
- 允许的: **ab_c, AB_de, _AbC**
- 不允许的: **1abc, ___abc, a___bc**
- 头 **32** 个字符有意义

IEC 61131 共有特性

语法关键字

- 特殊字(如 **FUNCTION)** 被保留
- 语法关键字总是大写字母
- 不能使用语法关键字作为标识符

IEC 61131 共有特性

注释

- 注释语用 (* 和 *) 框住
- 可将注释语放在语句以外的任何地方
- 对语句表 (IL)有些限制
- 注释语允许多行表示
- 注释语不允许嵌套

IEC 61131 基本数据类型

数据类型

- 不同的基本的数据类型
- 数据类型有整数, 浮点数, 位和位组, 时间和日期值和字符串

IEC 61131 基本数据类型

整型

类型名	描述	占用内存（位数）	
SINT	短整型	**8**	
INT	整型	**16**	
DINT	双整型	**32**	
LINT	长整型	**64**	**(not available)**
USINT	无符号短整型	**8**	
UINT	无符号整型	**16**	
UDINT	无符号双整型	**32**	
ULINT	无符号长整型	**64**	**(not available)**

IEC 61131 基本数据类型

整型表示

- 允许十进制, 十六进制(16#), 八进制(8#) 和 二进制(2#) 表示
- 可用下划线 (_) 作单元分隔
- 对INT的十进制表示: -123, +234, 0, 1_000
- 对INT的十六进制表示: 16#F1, 16#0A_1B
- 对INT的二进制表示: 2#0001_0011_0111_1111

IEC 61131 基本数据类型

浮点数类型

类型名	描述	占用内存（位数）
REAL	实数	**32**
LREAL	长实数	**64**

IEC 61131 基本数据类型

浮点数 (实数) 表示

- 十进制小数或指数表示
- **1000.23 ; 1.23e3 ; 1.23E3** 和 **1.23E03** 是同样的

IEC 61131 基本数据类型

时间数据类型

类型名	描述	占用内存（位数）
TIME	时间	**32**

IEC 61131 基本数据类型

持续时间表示

- 在文字前加 **TIME#, t# 或 T#**
- 允许溢出 **(例如 25 小时)**
- 使用**d** 表示天, **h** 表示小时, **m** 表示分, **s** 表示秒和 **ms** 表示毫秒
- 可使用下划线 **(_)** 作单元分隔
- **T#2d_26h_4m_12s_123ms**

IEC 61131 基本数据类型

日期和时间数据类型

类型名	描述	占用内存（位数）
DATE	日期	**32**
TIME_OF_DAY **或 TOD**	一天中的时间	**32**
DATE_AND_TIME **或 DT**	日期和时间	**32**

IEC 61131 基本数据类型

日期和时间表示

- 用 **DATE#** 或 **D#** 表示日期
- 用 **TIME_OF_DAY#** 或 **TOD#** 表示一天中的时间
- 用 **DATE_AND_TIME#** 或 **DT#** 表示日期和时间
- 日期: **D#1998-12-07** 表示 **7th July 1998**
- 一天中的时间: **TOD#12:00:00.123**
- 日期和时间: **DT#1998-12-07-12:00:00.123**

IEC 61131 基本数据类型

字符串数据类型

类型名	描述	占用内存（位数）
STRING	字符串	取决于字符长度（**N+1)Byte**

默认字符数：80 个

自定义长度字符串：
如： strVar : STRING(10);
(*表示字符串strVar包含最多10个字符*)

IEC 61131 基本数据类型

字符串表示

- 用 ‘ ‘ 引括字符
- 用 **$** 插到特殊字符前 **(**换行**$L** , 制表**$T**等**)**
- 字符串表示: **‘this is a line feed character $L‘**
- 空字符串: **‘‘**

IEC 61131 基本数据类型

位和位组数据类型

类型名	描述	占用内存（位数）
BOOL	**bit**	**1**
BYTE	**8 位**	**8**
WORD	**16 位**	**16**
DWORD	**32 位**	**32**
LWORD	**64 位**	**64 (not available)**

IEC 61131 基本数据类型

位和位组表示

- **TRUE 或 1**
- **FALSE 或 0**
- 用十进制, 十六进制, 八进制 或 二进制表示
- 字 - **WORD:234, 16#ff, 2#1001_1100_0011_1111**

IEC 61131 派生数据类型

- 根据基本数据类型或其它用户定义的数据类型建立自己的数据类型

- 使用 **TYPE .. END_TYPE** 结构框架定义

```
TYPE
        myOwnReal :        REAL;
END_TYPE

TYPE
        myArray      :        ARRAY[0..1000] OF BOOL;
END_TYPE
```

IEC 61131 派生数据类型

- 根据基本数据类型或其它用户定义的数据类型建立自己的结构数据类型

- 使用 **STRUCT .. END_STRUCT** 结构框架定义

```
TYPE  myStruct:
      STRUCT
            status      :     BOOL;
            inputValue  :     REAL;
      END_STRUCT
END_TYPE
```

IEC 61131 派生数据类型

- 建立自己的枚举数据类型
- 用括弧中的变量数值定义你的枚举数据类型

```
TYPE  Modes:
    (Initialisation := 0, Running, Idle, Reset, Faulty);
END_TYPE
```

Initialisation = 0, Running = 1.

IEC 61131 派生数据类型

- 数组是一个具有同样数据类型(基本类型或用户定义类型)的数值集合
- 允许三维数组

```
TYPE matrix:
        ARRAY[1..23, 0..1] OF INT;
END_TYPE
```

IEC 61131 变量

- 具有基本类型或用户定义类型的局部和全局变量
- 用一个 **VAR..END_VAR** 结构框架声明变量
- 同样数据类型的变量用逗号列出

```
VAR
  a,b,c:        REAL;
  d,e   :       BOOL;
  f     :       ARRAY[1..12] OF BOOL;
END_VAR
```

IEC 61131 变量

- 在程序,函数或功能块(FB)中的输入变量
- 用一个 **VAR_INPUT..END_VAR** 结构框架声明变量

```
VAR_INPUT
        a,b,c  :        REAL;
END_VAR
```

IEC 61131 变量

- 在程序和功能块(FB)中的输出变量
- 用一个 **VAR_OUTPUT..END_VAR** 结构框架声明变量

```
VAR_OUTPUT
        d,e     :       INT;
END_VAR
```

IEC 61131 变量

- 在程序和功能块(FB)中的输入和输出变量
- 在 POU内部允许更改变量
- 用一个 VAR_IN_OUT..END_VAR 结构框架声明变量

```
VAR_IN_OUT
        x       :       STRING;
END_VAR
```

IEC 61131 变量

- 全局变量
- 允许从所有 **POU** 中读写的变量
- 用一个 **VAR_GLOBAL..END_VAR** 结构框架声明变量

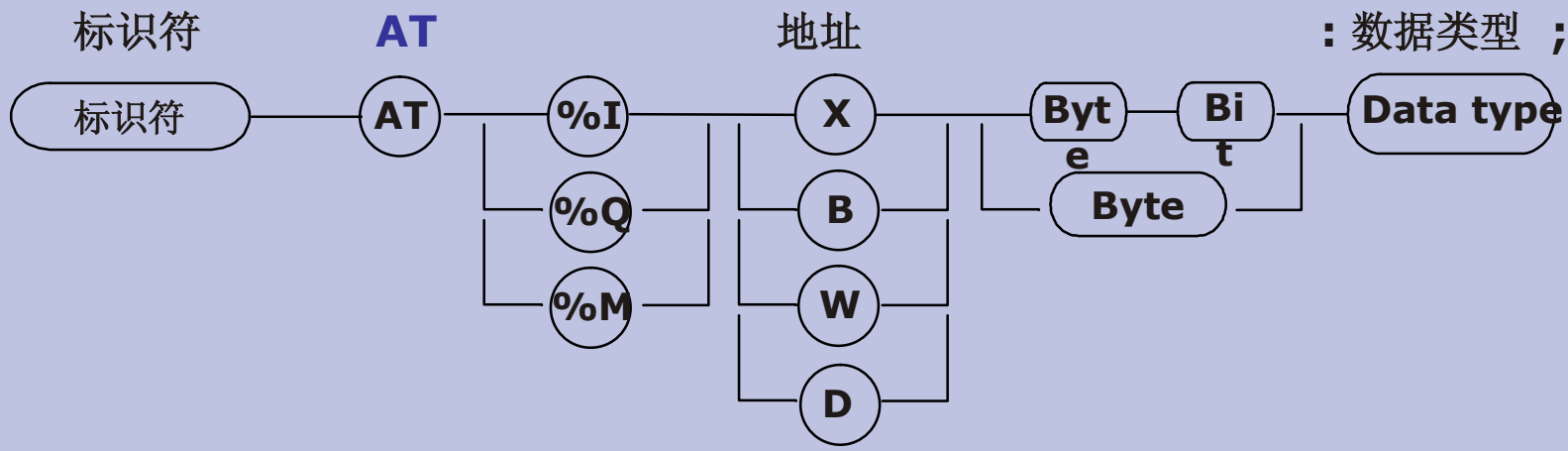
```
VAR_GLOBAL
        var    :        UDINT;
END_VAR
```

IEC 61131 变量

- 变量属性
- **RETAIN:** 在关电后,值被保存.并且**TwinCAT**启动后,值恢复.
- **CONSTANT:** 值不能被修改
- **AT:** 变量被指配存储器位置 (固定地址)

IEC 61131 变量

- 直接表示变量
- 以字符 **%** 起始
- 第二个字符为
  - **I** 对应 输入地址区**INPUT,**
  - **Q** 对应 输出地址区**OUTPUT**和
  - **M** 对应 内存地址区**MEMORY**



IEC 61131 变量

- 直接表示变量

- 第三个字符为
  - **X** 对应 位,
  - **B** 对应 字节,
  - **W** 对应 字,
  - **D** 对应 双字和
  - **L** 对应 长字 **(not available).**

- 例如:
  - **%IB24, %QX1.1, %MW12**

IEC 61131 变量

- 直接表示变量
- 地址可以交迭
- **%MB12** 是**%MW12** 的第一个字节,
  也是**%MD12** 的第一个字节
- **%MX12.0** 是 **%MB12** 的第一位

IEC 61131 变量

- 变量初值设定
- 每个变量在启动期间被赋初值
- 所有变量可改变默认初值

```
VAR
        a       :       INT := 13;
        b       :       STRING := `this is a string`;
        c       :       REAL := 1.1;
END_VAR
```

IEC 61131 变量

- 派生数据类型变量的初值设定
- 结构: 用括弧和对每个成员名赋初值

```
VAR
        a       : myStruct :=
                        (
                                status := TRUE,
                                inputValue := 2.5
                        );
END_VAR
```

IEC 61131 变量

- 派生数据类型变量的初值设定
- 数组: 使用逗号分隔设定多重初值

```
VAR
        a       : ARRAY[1..10] OF INT :=
                  1, 2, 2, 4, 5, 6, 7, 8, 9,10;
END_VAR
```

IEC 61131 函数

- 内部数据不存储
- 几个输入值
- 一个输出值
- 用户定义的函数可以用各种语言编辑代码 (除了SFC)
- 函数名必须和返回值的名称相同

IEC 61131 函数

例如

```
FUNCTION Average          :       REAL
        (* variable declaration *)
        VAR_INPUT
                IN1, IN2      :       REAL;
        END_VAR

        (* code body programmed in ST *)
        Average := (IN1 + IN2)/2;
END_FUNCTION
```

返回值的
数据类型

返回值名
= 函数名

IEC 61131 函数

- 超载函数

- 一些函数可以处理不同的数据类型 (调用操作)

```
a, b    :       REAL;
c, d    :       INT;

a       := ABS(b);   (* 用 REAL 作输入和输出 *)
c       := ABS(d);   (* 用 INT 作输入和输出 *)
```

IEC 61131 函数

- 数据类型转换函数

- 在不同数据类型之间需转换数值，你必需使用转换函数

```
a:      REAL;
b:      INT;

b:=     REAL_TO_INT(a);
```

IEC 61131 函数

- 标准 **IEC 61131-3** 函数
- 位组 **AND, OR, XOR, NOT, SHL, SHR, ROL, ROR**
- 数学 **ADD, SUB, MUL, DIV, MOD, EXPT, ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN**
- 类型转换 例如 **BOOL_TO_BYTE, REAL_TO_DINT**
- 选择 **SEL, MIN, MAX, LIMIT, MUX**
- 比较 **GT, GE, EQ, LT, LE, NE**
- 字符串 **LEN, LEFT, RIGHT, MID, CONCAT, INSERT,DELETE, REPLACE, FIND**

IEC 61131 功能块

- 设立输入, 输出和内部变量
- 运算法则建立新的输出和内部变量
- 参数被保持到下次执行 (在存储器中)
- 功能块例程是一个结构, 包含所有输入, 输出和内部变量
- 一个**FB** 允许建立多个实例

IEC 61131 功能块

- 在 **FB** 的外部只有输入, 输出和输入/输出变量传递参数
- 在其它**FB**或程序中, 以不同的编程语言使用 **FB** 调用
- 一个 **FB** 例程在调用的 **FB**/程序中是一个变量或全局变量
- **FB**例程可以是对其它**FB**/程序的输入变量

IEC 61131 功能块

```
FUNCTION_BLOCK Counter
        VAR_INPUT
                Mode   :       INT;    (* 0 = Reset, 1 =
Count *)
        END_VAR
        VAR_OUPUT
                Out    :       INT;    (* actual counter
value *)
        END_VAR
        IF Mode = 0
        THEN
                Out := 0;               (* reset *)
        ELSIF Mode = 1
        THEN
                Out := Out + 1;
        END_IF;
END_FUNCTION_BLOCK
```

老的计数值加 1
得到新的记数值

IEC 61131 功能块

标准 IEC 61131-3 功能块

- 触发器 **SR, RS, SEMA**
- 沿探测 **R_TRIG, F_TRIG**
- 记数器 **CTU, CTD, CTUD**
- 定时器 **TP, TON, TOF, RTC**

IEC 61131 程序

- 程序可以有输入, 输出, 局部变量和算法的程序代码部分
- 不同于**FB:** 程序没有例程
- 程序没有存储器
- 在程序中使用各种语言
- 程序由任务来调用 **(TwinCAT：**可由其它程序调用**)**

IEC 61131 程序

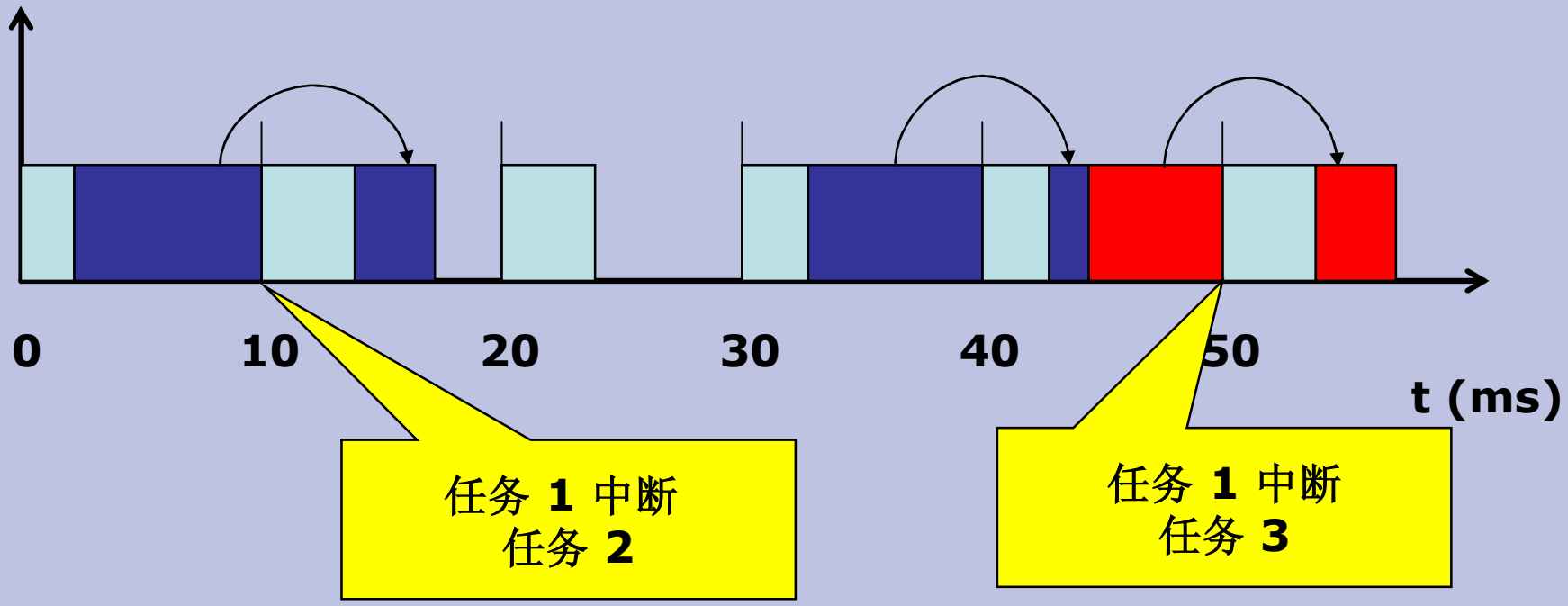
```
PROGRAM Main
        VAR
                counter_1 : Counter; (* instance of FB Counter *)
                actCount   : INT;
        END_VAR
        IF bfirstCycle
        THEN
                counter_1(Mode := 0);(* call FB with reset mode *)
        ELSE
                counter_1(Mode := 1);(* call FB with count mode*)
        END_IF
        actCount := counter_1.Out; (* use output variable of     *)
                                   (* counter_1                  *)

END_PROGRAM
```

IEC 61131 任务

- 用任务控制执行
- 执行不同循环时间的程序
- 指派任务的优先级 (要求无间断运行的程序必需有最高优先级)
- **TwinCAT:** 每个**PLC**运行核**(Runtime)**有四个任务
- 抢先调度

IEC 61131 任务

任务 1 具有优先级 0 和 循环时间 10 ms
任务 2 具有优先级 1 和 循环时间 30 ms
任务 3 具有优先级 2 和 循环时间 40 ms



IEC 61131 编程语言

- 文本语言
  - 结构文本 **(ST – Structured Text)**
  - 指令表 **(IL – Instruction List)**
- 图形语言
  - 功能块图 **(FBD – Function Block Diagram)**
  - 梯形图 **(LD – Ladder Diagram)**
  - 顺序功能图 **(SFC – Sequential Function Chart)**

IEC 61131 结构文本 (ST)

- 高级语言 (类似 **PASCAL**)
- 复杂公式 (赋值命令)
- 具有条件和反复陈述的流控
- 用制表符和注释使得编码易读

IEC 61131 结构文本 (ST)

- 分配值和表达式赋值

```
A[i] := B;

A[i+1] := SIN(SQRT(A[i+3]));

C := timer.Q; (* timer is an instance from FB TOF *)

D := E/F + COS(A[i+1]);

bFlag := X AND Y OR Z;
```

IEC 61131 结构文本 (ST)

· 条件语句

```
IF      < 布尔表达式 >      THEN
        <语句>
ELSIF  <布尔表达式>         THEN
        <语句>
ELSE
        <语句>
END_IF
```

```
(*示例：*)
IF temp<20 THEN
        heating_on := TRUE;
ELSE
        heating_on := FALSE;
END_IF;
(*温度低于20度,加热器打开，否则关闭。*)
```

IEC 61131 结构文本 (ST)

```
CASE <整数表达式> OF

        <整数选择值 1> : <语句>
        <整数选择值 2> : <语句>
        ...
        <整数选择值 n> : <语句>
        ELSE
                <语句>

END_CASE;
```

```
(*示例：*)
CASE INT1 OF
        1: BOOL1 := TRUE;
           BOOL2 := FALSE;
        2: BOOL1 := FALSE;
           BOOL2 := TRUE;
        ELSE
           BOOL1 := FALSE;
           BOOL2 := FALSE;
END_CASE;
```

IEC 61131 结构文本 (ST)

· 循环语句

```
FOR <初始变量>
        TO      <终值表达式>
        BY      <增量表达式> DO
        <语句>
END_FOR;
```

```
(*示例：*)

FOR i := 1 TO 100 BY 1 DO
        a[i] := 0;
END_FOR;
```

IEC 61131 结构文本 (ST)

```
WHILE <布尔表达式> DO
        <语句>
END_WHILE;
```

```
(*示例：*)
i := 1;
WHILE i < 100 DO
        a[i] := 0;
        i := i+1;
END_WHILE;
```

IEC 61131 结构文本 (ST)

```
REPEAT
        <语句>
UNTIL <布尔表达式>
END_REPEAT;
```

```
(*示例：*)
i := 1;
REPEAT
        a[i] := 0;
        i := i+1;
UNTIL i > 100
END_REPEAT;
```

IEC 61131 结构文本 (ST)

EXIT

在 **EXIT** 语句被执行后,在反复循环后的下个语句将被执行

RETURN

执行了**RETURN** 语句后, 当前的**POU (FB or** 函数) 执行被中断

IEC 61131 结构文本 (ST)

· 调用功能块

例子:

```
VAR
        timer  :        TOF;
        out    :        BOOL;
END_VAR
        timer(IN:= TRUE, PT:= T#1s);
        ...
        timer (IN:= FALSE);
        out := timer.Q;
```

IEC 61131 指令表 (IL)

- 低级语言（类似汇编程序）
- 面向累加器的
- 每行只允许一个操作，如存储一个值到累加寄存器
- 用跳转和标号控制流程
- 注释在每行的后面

IEC 61131 指令表 (IL)

标号	操作符	操作数	注释
	LD	TRUE	(* load TRUE *)
	ST	var1	(* store in var1 *)
	JMPC	label1	(* jump conditional *)
	LD	FALSE	(* load FALSE *)
	ST	var2	(* store var 2 *)
label1:	LD	12	(* load int literal *)
	ADD	var4	(* add
	*)		
	ST	var3	(* store var3 *)

IEC 61131 指令表 (IL)

- 调用函数和功能块
- 使用 **CAL** 操作

IEC 61131 指令表 (IL)

三种方法调用 **FB:**

· 使用输入一个列表

```
CAL FB1(in := TRUE, mode := 4)
```

· 在调用前装填输入

```
LD      TRUE
ST      FB1.in
LD          4
ST      FB1.mode
CAL     FB1
```

· 使用输入参数

只适用于标准**FB,** 使用标准变量名 **(**例如**. counter-up** 的输入变量 **CU)**

IEC 61131 指令表 (IL)

```
0001    LD      I
0002    ADD       1
0003    ST      J
0004    LD      TRUE
0005    ST      BlinkBitsArray[J]
0006
0007    LD      I
0008    SUB     1
0009    ST      J
0010
0011    LD      FALSE
0012    ST      BlinkBitsArray[J]
0013    ST      BlinkBitsArray[I]
0014
0015
0016    LD      I
0017    ADD       1
0018    ST      I
0019
0020    LD      I
0021    LT      Len
0022    JMPC    exit_end
0023
0024    LD      FALSE
0025    ST      BlinkBitsArray[J]
0026    ST      BlinkBitsArray[I]
0027
0028    LD      1
0029    ST      i
0030
0031 exit_end:
```

操作符

操作数

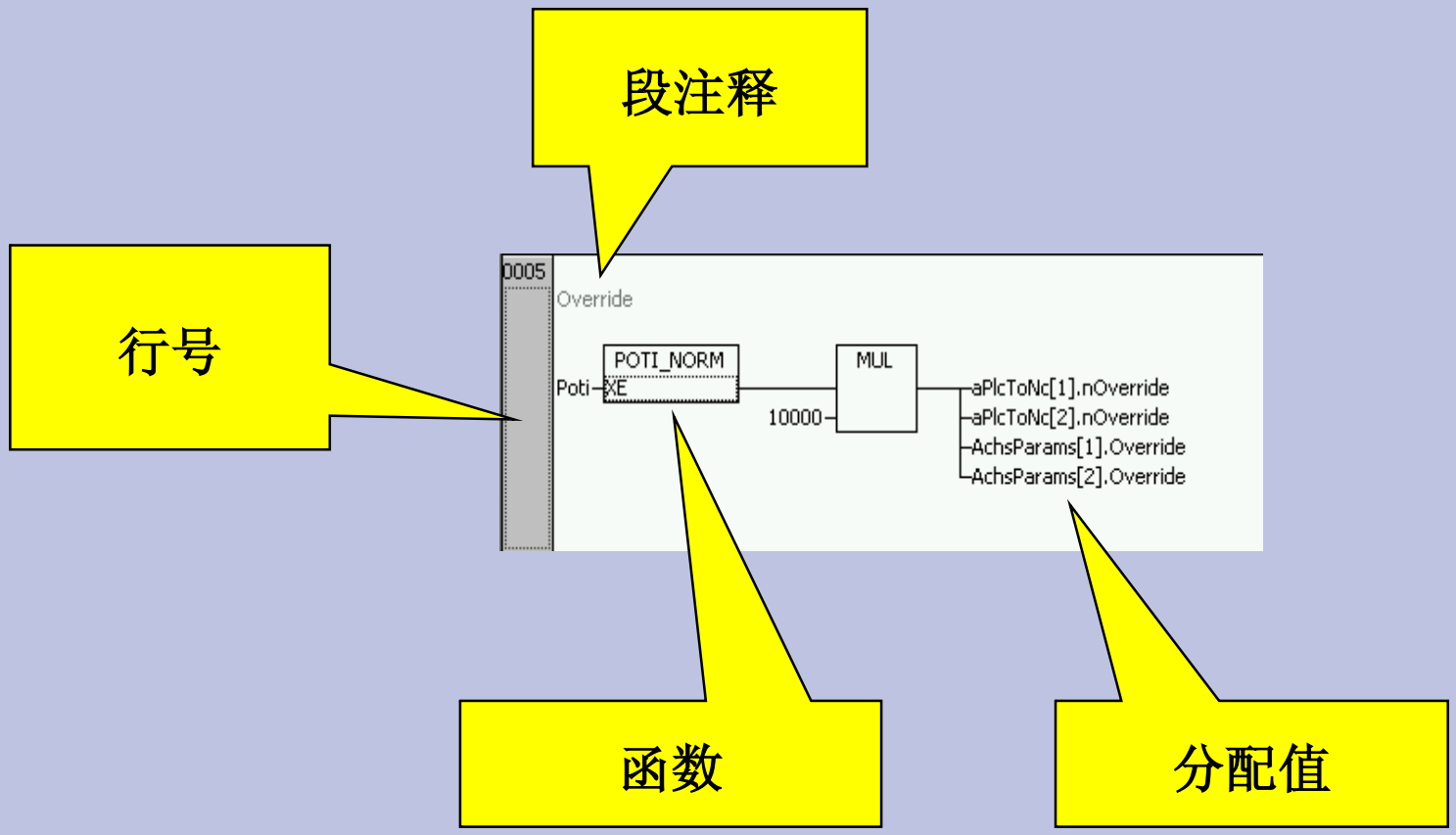
有条件跳转

标号

IEC 61131 功能图 (FBD)

- 图形表示 **FB,** 函数和程序以及它们的相互关联图
- 全图形式
- 块图 "线连" 在一起
- 允许跳转和返回

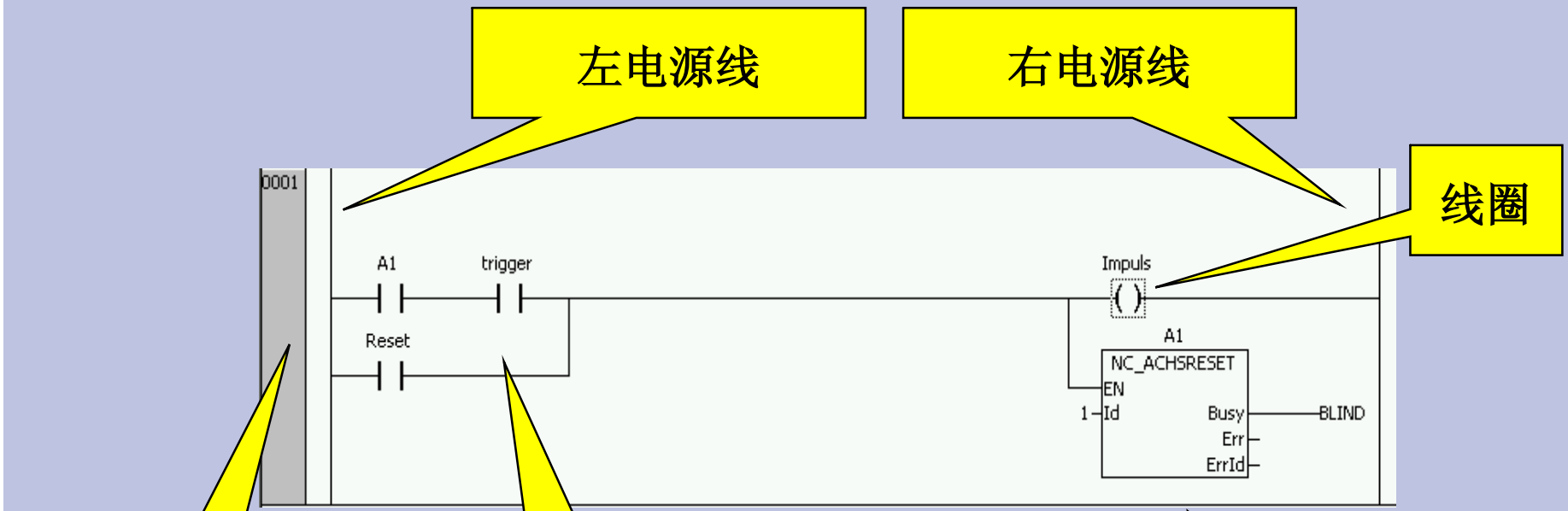
IEC 61131 功能图 (FBD)



IEC 61131 梯形图 (LD)

- '继电器梯形图', 利于复杂的 **AND** 和 **OR** 逻辑网图
- 基于 **US** 编程形式
- 左端垂直电源线连带触点和线圈
- 触点表示布尔变量
- 允许跳转返回

IEC 61131 梯形图 (LD)



IEC 61131 功能流程图 (SFC)

- 描述复杂的序列的‘语言‘
- 具有 ‘动作’ 和 ‘转移’ 的状态系统
- 通过划分成小部分来编复杂的部分
- 每个单元 (动作-action, 转移- transition) 可以用任何 IEC 语言编程

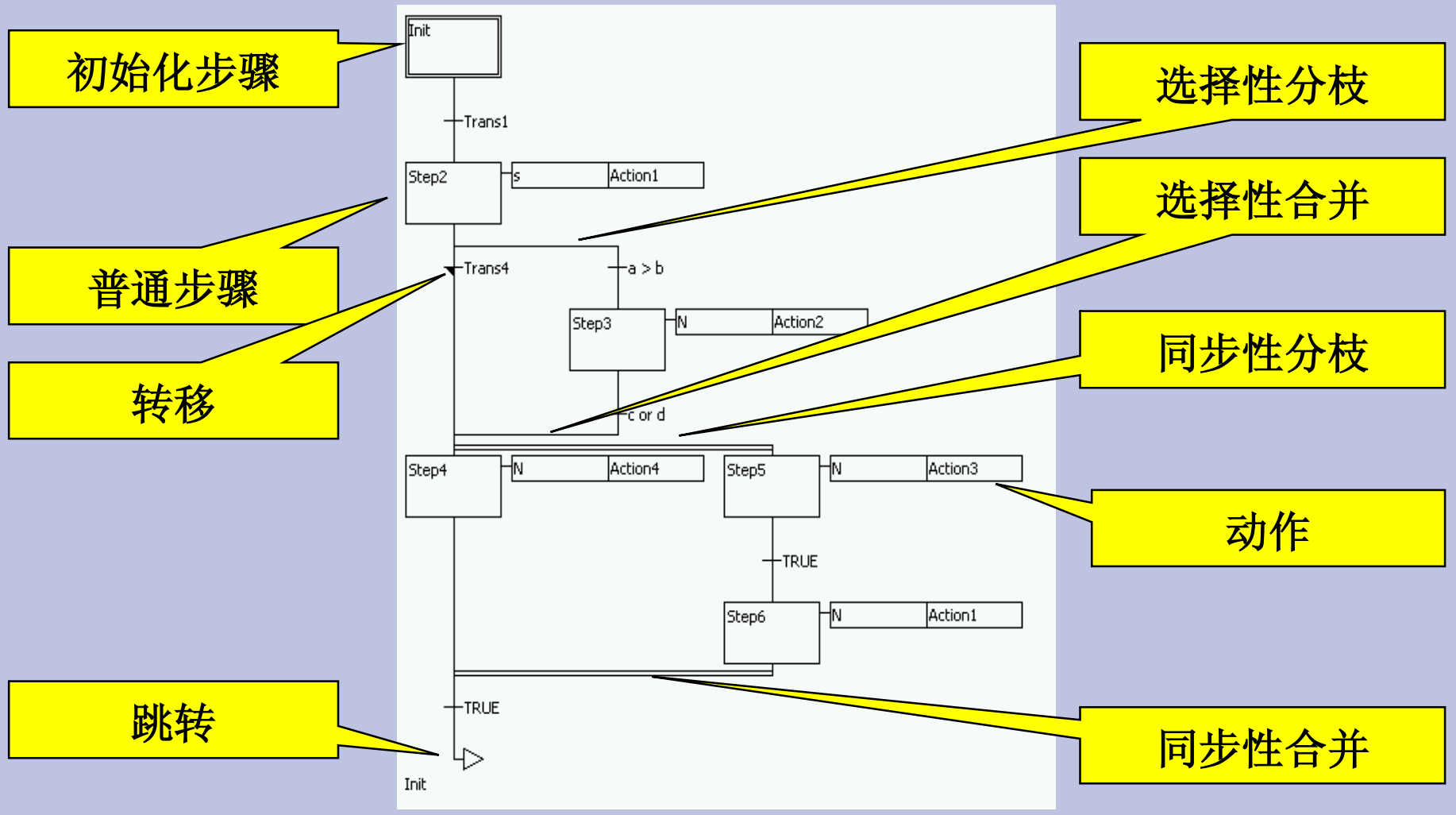
IEC 61131 功能流程图 (SFC)

- 步骤
  - 表示流程的一个状态
  - 特殊步骤: 初始化步骤
- 转移
  - 条件, 当为 **TRUE** 时, 下一步骤激活

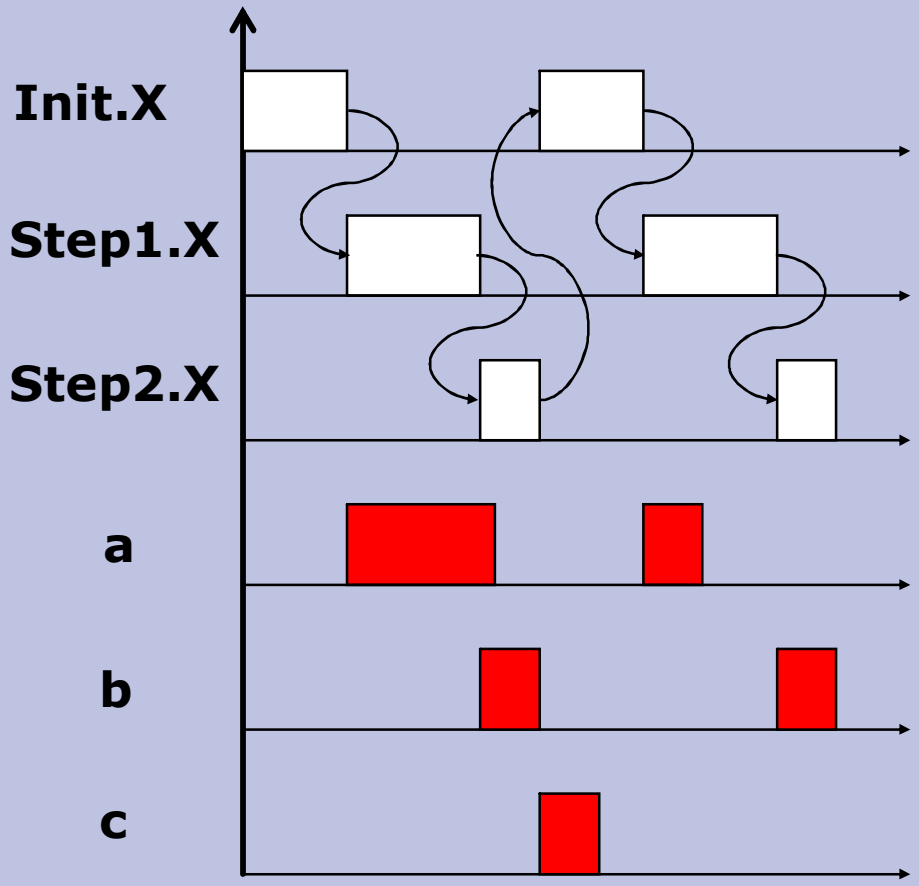
IEC 61131 功能流程图 (SFC)

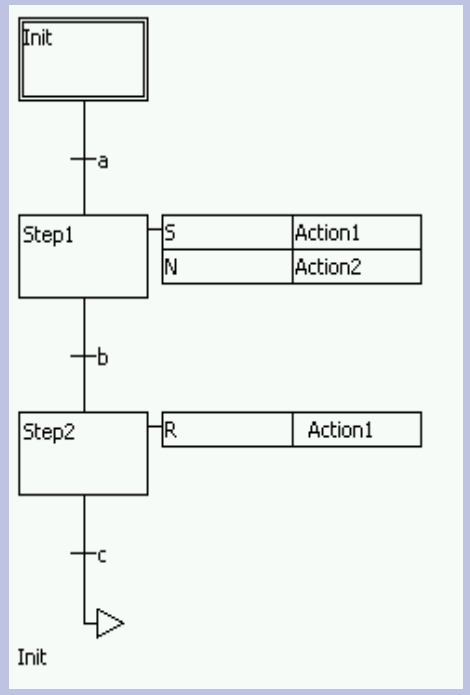
- 选择性: 分枝和合并
  - 在所有选择性路径的转移指定是否这个路径被选择
  - 转移在选择性路径的末端
  - 从左到右优先
- 同步性: 分枝和合并
  - 在所有同步路径上的所有步骤同时起动
  - 在末端有一个转移

IEC 61131 功能流程图 (SFC)



IEC 61131 功能流程图 (SFC)





IEC 61131 功能流程图 (SFC)

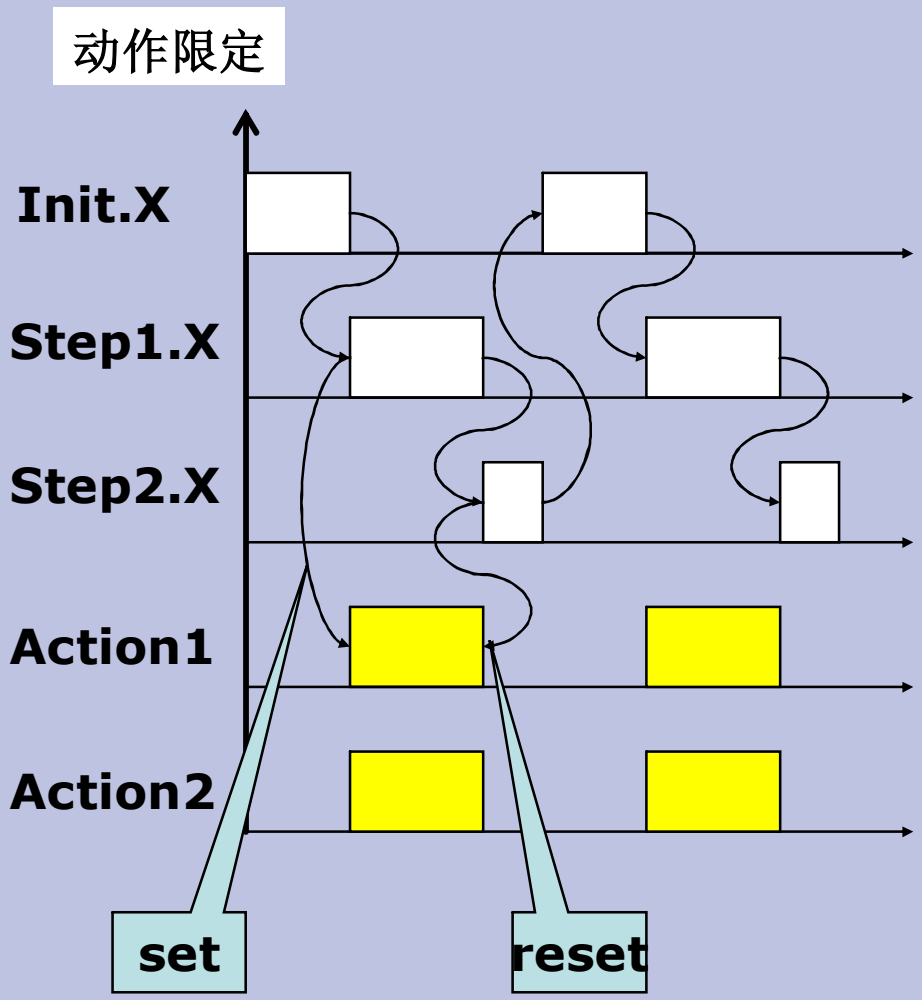
- 步骤 **(step)**
  - 普通步骤有步骤名
  - 初始化步骤, 每个图只用一个
  - 激活标记用 **<步骤名>.X**
  - 释放时间用 **<步骤名>.T**
- 转移 **(transition)**
  - 带有布尔结果的变量, 语句 或 **ST** 表达式
- 动作 **(action)**
  - 用所有**5**种语言编程
  - 使用限定控制执行

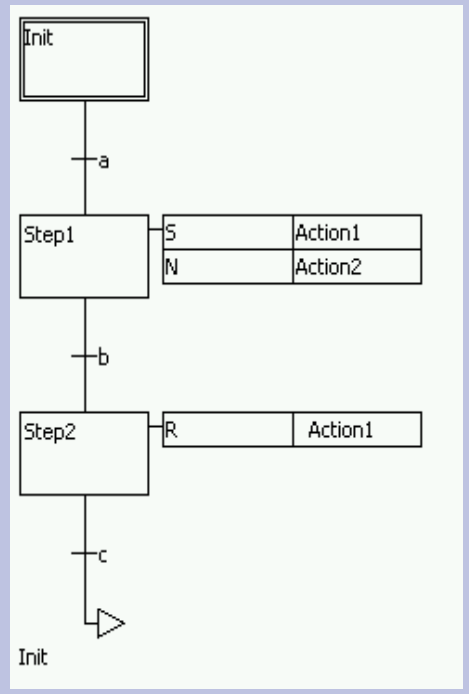
IEC 61131 功能流程图 (SFC)

动作限定

- **N, None** 不存储
- **R** 复位一个存储动作
- **S** 设定一个动作 (存储)
- **L** 时间限制动作, 在给定的周期后停止
- **D** 时间延迟动作, 在给定的周期后启动
- **P** 脉冲动作, 步骤激活时动作一次和步骤失效时动作一次
- **SD** 存储和时间延迟,在给定的周期后设定
- **DS** 动作被延迟和存储
- **SL** 存储和时间限制

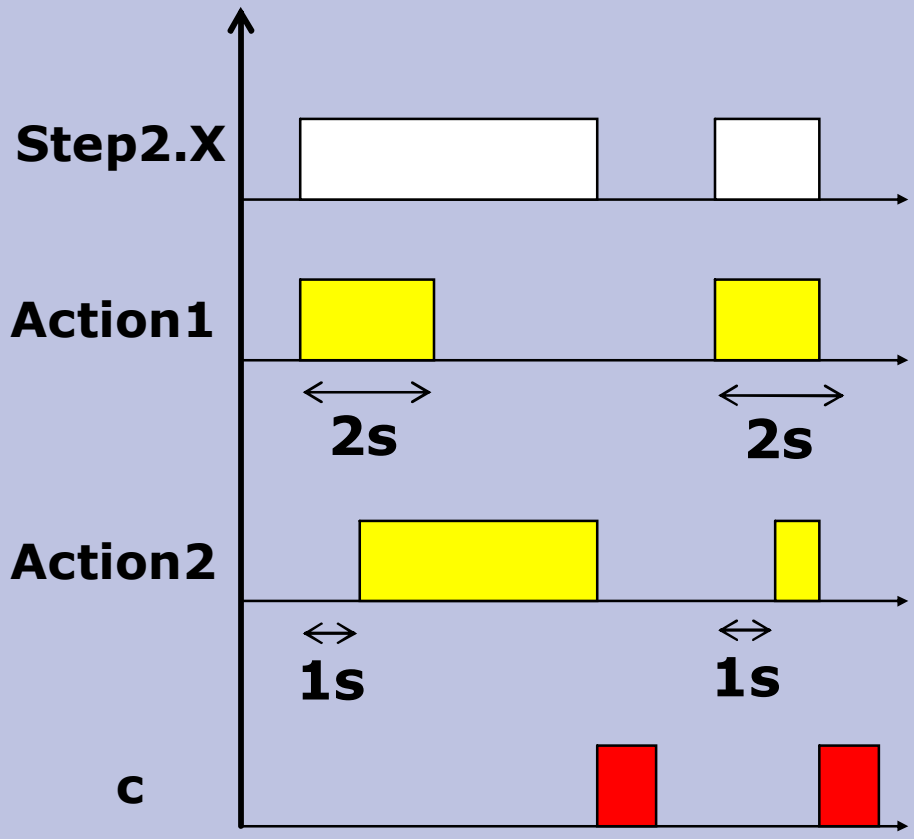
IEC 61131 功能流程图 (SFC)

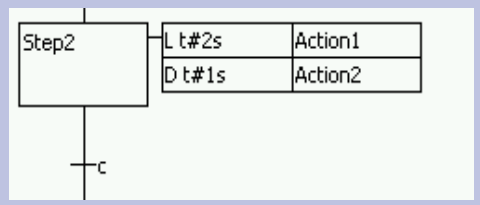




IEC 61131 功能流程图 (SFC)

动作限定





谢谢大家！！