技成培训网直播课教学资料

西门子 PLC 案例任务指导书

（西门子综合应用实战课）

2021-11

任务 1《基于 SCL 的传送带控制程序设计》

本节任务及目标管理				
名称	基于 SCL 的传送带控制程序设计		序号	SM-03202111-01
难易程度	基础	中级		✓ 高级
官网配套相关课程	叨叨课程《西门子 PLC 高级语言 SCL 教程【初级】》 叨叨课程《西门子 PLC 高级语言 SCL 教程【高级】》			
编制人	张志强	班级	三菱 FX 系列基础案例实操课	
上课方式	PPT+实操		考核方式	自行评价
上课时间			2021 年 11 月 19 号 20:35~21:35（叨叨直播间）	
课程准备资料			《S7-1200/1500 系列编程手册》	
学习目标	✓ 掌握 SCL 中赋值指令的用法； ✓ 掌握 SCL 中 IF 指令的用法； ✓ 掌握 SCL 中定时器指令应用； ✓ 掌握 SCL 中沿指令的用法； ✓ 掌握 SCL 编程的基本方法； ✓ 掌握结构化编程的方法。			
适用对象	● 本课程为提高课程适合有编程基础学习者； ● 适合有一定项目经验的学习者。			
课后评价	✓ 是否（能）掌握 SCL 中赋值指令的含义？ ✓ 是否（能）掌握 SCL 中 IF 指令的用法？ ✓ 是否（能）掌握 SCL 中定时器指令应用？ ✓ 是否（能）掌握 SCL 中沿指令的用法？ ✓ 是否（能）理解 SCL 编程的特点？ ✓ 是否（能）理解结构化编程的特点？			

一、案例任务控制描述

现使用 Factory IO 软件搭建了一个传送场景，传送带如图 1-1 所示，控制面板如图 1-2 所示，其控制要求如下：

1. 在自动模式下，按启动按钮，第一段传送带把货物往前输送，碰到感应器 A；启动第二段传送带，货物继续向前输送；碰到感应器 B，移栽机启动把货物往右移；碰到感应器 C，移栽机右移停止；经过短暂延时，移栽机抬升机构抬升并把货物向前移动；直到货物离开感应器 D，抬升机构下降，传送带继续输送货物，直到碰到感应器 E；在启动状态输送带一直循环输送，按停止按钮则停止。

2. 在手动模式下，可以通过手动按钮，手动控制传送带，以解决遗留货物问题。

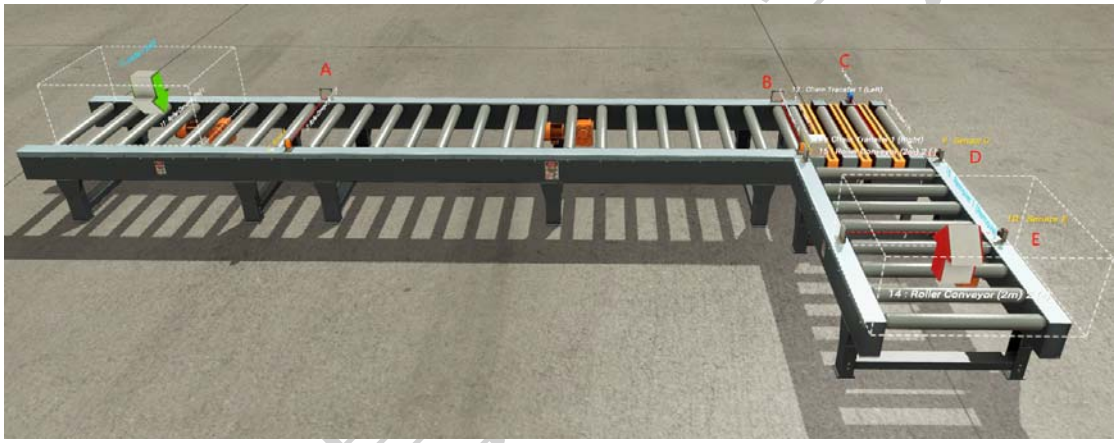

图 1-1

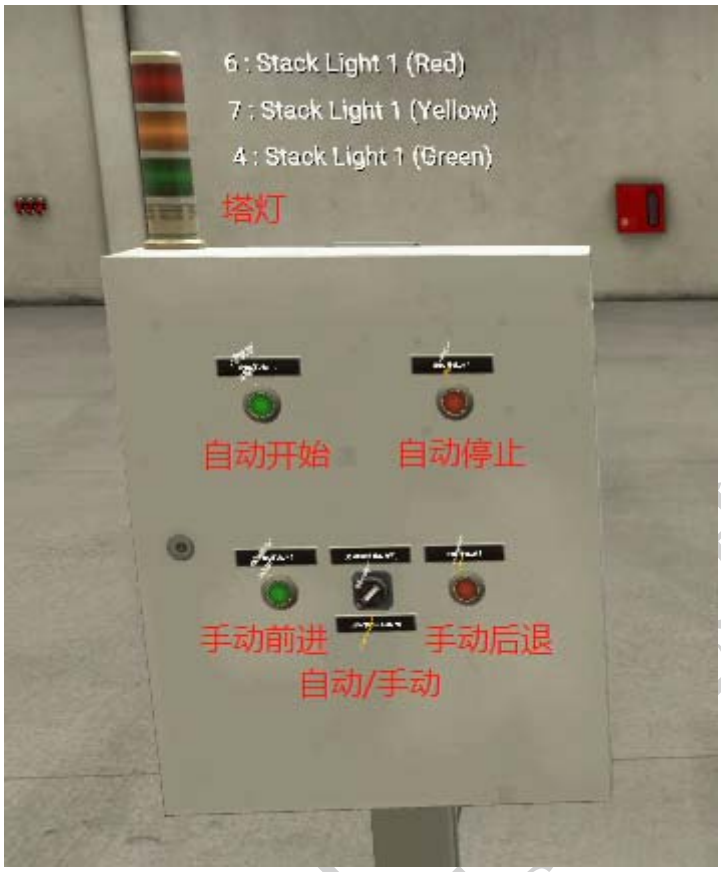

图 1-2

二、相关基础知识

任务中的案例基于 S7-1200 系列 PLC 和 FactoryIO 软件的应用案例，在学习本案例前请先自评下是否具备以下基础。

- **课程一：《西门子 S7-1200 编程应用入门》**

（1） 触点和线圈指令应用

观看地址：https://course.jcpeixun.com/5581/458363.html

观看地址：https://course.jcpeixun.com/5581/458364.html

（2） 沿指令应用

观看地址：https://course.jcpeixun.com/5581/458365.html

观看地址：https://course.jcpeixun.com/5581/458366.html

（3） 定时器的使用

观看地址：https://course.jcpeixun.com/5581/458384.html

- **课程二：《虚拟仿真入门与应用》**

（1） Factory IO 软件入门

观看地址：https://course.jcpeixun.com/7078/462510.html

（2） 通信设置

观看地址：https://course.jcpeixun.com/7078/462533.html

- **SCL 相关知识**

（1） 什么是 SCL

SCL（Structured Control Language，结构化控制语言）是一种基于 PASCAL 的高级编程语言。这种语言基于标准 DIN EN 61131-3（国际标准为 IEC 1131-3）。根据该标准，可对用于可编程逻辑控制器的编程语言进行标准化。SCL 编程语言实现了该标准中定义的 ST 语言(结构化文本) 的 PLCopen 初级水平。

SCL 除了包含 PLC 的典型元素（例如，输入、输出、定时器或存储器位）外，还包含高级编程语言，典型的就是表达式。

表达式将在程序运行期间进行运算，然后返回一个值。一个表达式由操作数（如常数、变量或函数调用）和与之搭配的操作符（如 *、/、+ 或 -）组成。

不同的运算符，分别可使用以下不同类型的表达式：

1) 算术表达式

算术表达式既可以是一个数字值，也可以是由带有算术运算符的两个值或表达式组合而成。

2) 关系表达式

关系表达式将对两个操作数的值进行比较，然后得到一个布尔值。如果比较结果为真，则结果为 TRUE，否则为 FALSE。

3) 逻辑表达式

逻辑表达式由两个操作数以及逻辑运算符（AND、OR 或 XOR）或取反操作数 (NOT) 组成。

（2） 赋值指令“:=”

严格意义上来讲，赋值算不上是指令，它的涵义是把赋值符号右边变量或者表达式的状态赋给左边变量，可以简单理解成“等号”，但是它和“=”是不一样。在 SCL 中“=”是一个判断表达式，判断是否相等而返回一个“TRUE”（相等）或者“FALSE”（不相等）。

赋值运算的数据类型取决于左边变量的数据类型。右边表达式的数据类型必须与该数据类型一致。可通过以下方式编程赋值运算：

1) 单赋值运算

执行单赋值运算时，仅将一个表达式或变量分配给单个变量：

示例：a := b;

2) 多赋值运算

执行多赋值运算时，一个指令中可执行多个赋值运算。

示例：a := b := c;

此时，将执行以下操作：

b := c;

a := b;

3) 组合赋值运算

执行组合赋值运算时，可在赋值运算中组合使用操作符“+”、“-”、“*”和“/”：

示例：a += b;

此时，将执行以下操作：

a := a + b;

也可多次组合赋值运算：

a += b += c *= d;

此时，将按以下顺序执行赋值运算：

c := c * d;

b := b + c;

a := a + b;

（3） 条件执行指令 IF

使用“条件执行”指令，可以根据条件控制程序流的分支。该条件是结果为布尔值（TRUE 或 FALSE）的表达式。可以将逻辑表达式或比较表达式作为条件。

执行该指令时，将对指定的表达式进行运算。如果表达式的值为 TRUE，则表示满足该条件；如果其值为 FALSE，则表示不满足该条件。

根据分支的类型，可以对以下形式的指令进行编程：

IF 分支：

IF <condition> THEN <instructions>

END_IF;

如果满足该条件，则将执行 THEN 后编写的指令。如果不满足该条件，则程序将从 END_IF 后的下一条指令开始继续执行。

IF 和 ELSE 分支：

IF <condition> THEN <instructions1>

ELSE <Instructions0>

END_IF;

如果满足该条件，则将执行 THEN 后编写的指令。如果不满足该条件，则将执行 ELSE 后编写的指令。程序将从 END_IF 后的下一条指令开始继续执行。

IF、ELSIF 和 ELSE 分支：

IF <condition1> THEN <instructions1>

ELSIF <condition2> THEN <instruction2>

ELSE <Instructions0>

END_IF;

如果满足第一个条件（<条件 1>），则将执行 THEN 后的指令（<指令 1>）。

执行这些指令后，程序将从 END_IF 后继续执行。

如果不满足第一个条件，则将检查第二个条件（<条件 2>）。如果满足第二个条件（<条件 2>），则将执行 THEN 后的指令（<指令 2>）。执行这些指令后，程序将从 END_IF 后继续执行。

如果不满足任何条件，则先执行 ELSE 后的指令（<指令 0>），再执行 END_IF 后的程序部分。在 IF 指令内可以嵌套任意多个 ELSIF 和 THEN 组合。可以选择对 ELSE 分支进行编程。

（4） 定时器、沿指令

定时器在 SCL 中的用法和在 LAD 中的基本用法是一样的，唯一的区别是在 LAD 中一般用 EN 管脚控制其是否启用的逻辑，而在 SCL 中一般是直接使能定时器（即不加调用条件），控制端一般放在输入“IN”管脚。定时器如果处于条件调用（IF 或者 CASE）里而条件并未满足的情况下，保持现有状态不变。这种情况可能会使控制逻辑变得复杂，所以 SCL 中定时器一般不加调用条件。同理，沿指令也是类似的情况。

三、任务的实施

1. I/O 地址分配

Factory IO 软件上可以分配 PLC 的 IO 地址，在博途软件中，通过它提供的函数（FC9000）会自动关联 PLC 的 IO 地址，如图 3-1 所示：

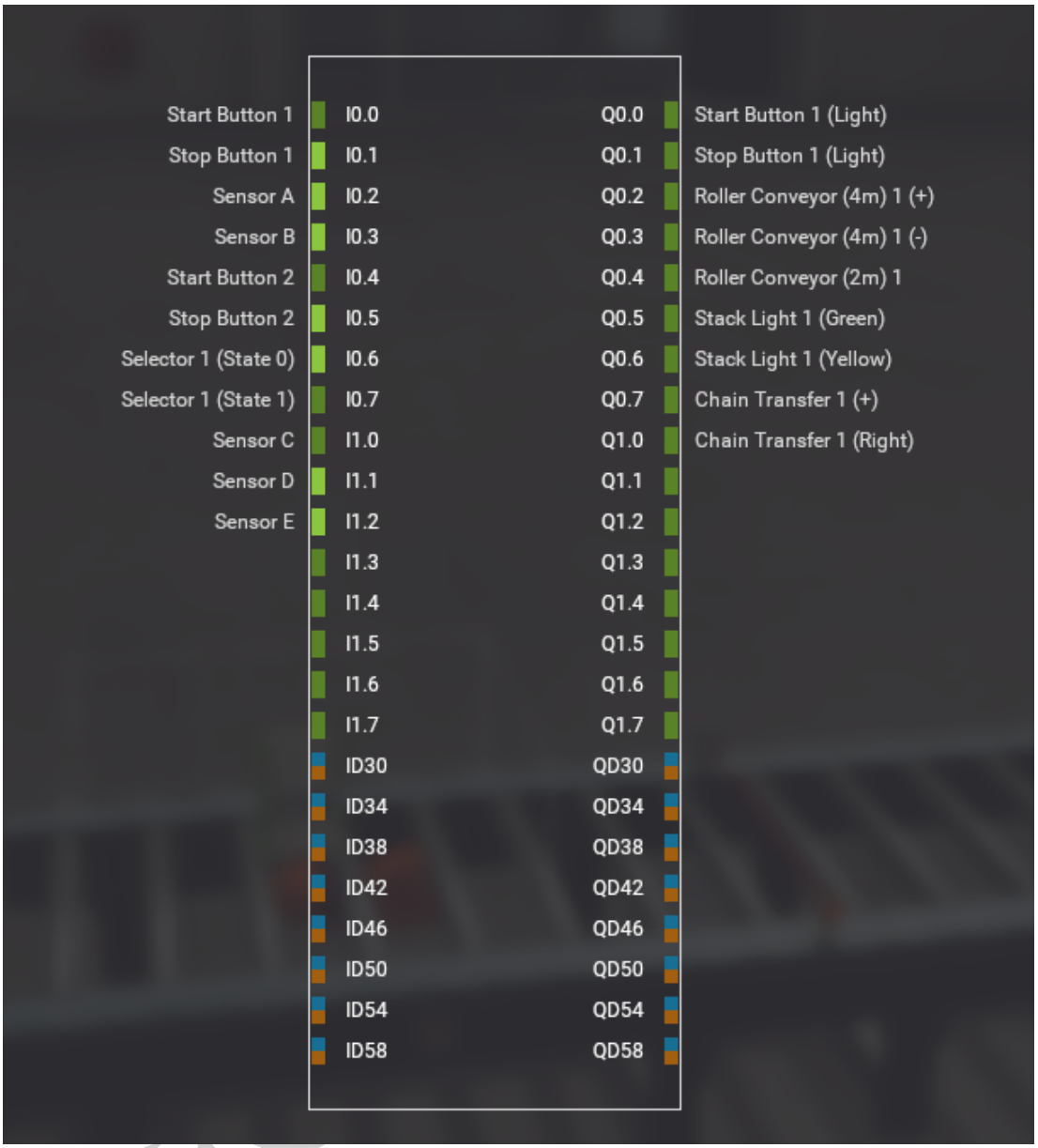

图 3-1

在程序中，为了直观表达变量的含义，建立一个 DB 块，然后通过一个 FC 块把 DB 块里的变量和 IO 相关联，之后我们只要使用 DB 块里的变量即可，如图 3-2 和 3-3 所示：

DEVICE

	名称	数据类型	起始值	保持	从 HMI/OPC..	从 H...	在 HMI ...	设定值
1	Static							
2	startPB	Bool	false		☑	☑	☑	
3	stopPB	Bool	false		☑	☑	☑	
4	sensorA	Bool	false		☑	☑	☑	
5	sensorB	Bool	false		☑	☑	☑	
6	sensorC	Bool	false		☑	☑	☑	
7	sensorD	Bool	false		☑	☑	☑	
8	sensorE	Bool	false		☑	☑	☑	
9	autoMode	Bool	false		☑	☑	☑	
10	manualMode	Bool	false		☑	☑	☑	
11	manualForward	Bool	false		☑	☑	☑	
12	manualBackward	Bool	false		☑	☑	☑	
13	chainTransferP	Bool	false		☑	☑	☑	
14	chainTransferR	Bool	false		☑	☑	☑	
15	conveyorA	Bool	false		☑	☑	☑	
16	conveyorB	Bool	false		☑	☑	☑	
17	entryConveyor	Bool	false		☑	☑	☑	
18	lampStart	Bool	false		☑	☑	☑	
19	lampStop	Bool	false		☑	☑	☑	
20	towerGreen	Bool	false		☑	☑	☑	
21	towerYellow	Bool	false		☑	☑	☑	

图 3-2

DeviceManager

名称	数据类型	默认值	注释
Input			
<新增>			
Output			

IF... CASE... OF... FOR... TO DO.. WHILE.. DO... (*...*) REGION

```
1  "DEVICE".startPB := "DI_00";
2  "DEVICE".stopPB  := NOT "DI_01";
3  "DEVICE".sensorA := NOT "DI_02";
4  "DEVICE".sensorB := NOT "DI_03";
5  "DEVICE".sensorC := "DI_10";
6  "DEVICE".sensorD := NOT "DI_11";
7  "DEVICE".sensorE := NOT "DI_12";
8  "DEVICE".manualForward := "DI_04";
9  "DEVICE".manualBackward := NOT "DI_05";
10 "DEVICE".autoMode := "DI_06";
11 "DEVICE".manualMode := "DI_07";
12 "DO_00" := "DEVICE".lampStart;
13 "DO_01" := "DEVICE".lampStop;
14 "DO_02" := "DEVICE".conveyorA;
15 "DO_03" := "DEVICE".conveyorB;
16 "DO_04" := "DEVICE".entryConveyor;
17 "DO_05" := "DEVICE".towerGreen;
18 "DO_06" := "DEVICE".towerYellow;
19 "DO_07" := "DEVICE".chainTransferP;
20 "DO_10" := "DEVICE".chainTransferR;
```

图 3-3

2. 程序设计

前文中我们提到过，在 SCL 中使用定时器、沿指令等控制时，一般会选择无条件调用，所以我们在开头调用所有定时器和沿指令。当然这些定时器、沿指令

并不一定是开头就规划好的，可能在是在下面需要的时候，插入进来的，只是我们习惯性的把它们放在一起而已，不放在一起大部分情况下不会对程序产影响。如图 3-4 所示：

```
#R_TRIG_sensorA(CLK:="DEVICE".sensorA);
#R_TRIG_sensorB(CLK:="DEVICE".sensorB);
#R_TRIG_sensorC(CLK := "DEVICE".sensorC);
#R_TRIG_sensorD(CLK := "DEVICE".sensorD);
#F_TRIG_sensorD(CLK := "DEVICE".sensorD);
#F_TRIG_sensorE(CLK := "DEVICE".sensorE);
#TON_conveyorA_Idle(IN:=NOT "DEVICE".sensorA AND NOT "DEVICE".sensorB,PT:=t#20s);
#TON_transferDelay(IN:=#statTransferDelay,PT:=t#0.5s);
```

图 3-4

模式切换的程序用表达式赋值指令，言简意赅，这里就要求对赋值指令的理解，它可以是一个表达式的结果赋值给一个变量，这里我们做了一个选择开关的互锁。如图 3-5 所示：

```
//模式选择
#statAutoMode := "DEVICE".autoMode AND NOT "DEVICE".manualMode;
#statManualMode := "DEVICE".manualMode AND NOT "DEVICE".autoMode;
```

图 3-5

如果真正用于实际生产，便捷的手动操作是必不可少的，我们是演示案例，所以主要是写自动程序，这并不代表手动程序不重要。接下来主要是自动程序，在自动模式下，按启动按钮自动启动，按停止按钮自动停止，这是一个标准的 SCL 表达的起保停程序。在自动启动的状态下，第一段传送带直接启动，然后用 A 传感器的上升沿启动第二段传送带，它的停止条件是空闲一段时间，定时器写在了图 3-4 的程序中。此段程序如图 3-6 所示：

```
IF #statAutoMode THEN
    //大家喜欢的起保停
    IF "DEVICE".startPB THEN
        #statAutoStart := TRUE;
    END_IF;
    IF "DEVICE".stopPB THEN
        #statAutoStart := FALSE;
    END_IF;

    //开始进入传送带
    IF #statAutoStart THEN
        "DEVICE".entryConveyor := TRUE;
        //传送带A
        IF #R_TRIG_sensorA.Q THEN
            "DEVICE".conveyorA := TRUE;
        END_IF;
        IF #TON_conveyorA_Idle.Q THEN
            "DEVICE".conveyorA := FALSE;
        END_IF;
```

图 3-6

后面控制其他传送带的逻辑是一样的，用感应器的沿脉冲触发启动，再编写其停止的条件，如图 3-7 所示：

```
        //链条移栽机
        IF #R_TRIG_sensorB.Q THEN
            "DEVICE".chainTransferP := TRUE;
        END_IF;
        IF #R_TRIG_sensorC.Q THEN
            "DEVICE".chainTransferP := FALSE;
            #statTransferDelay := TRUE;
        END_IF;
        IF #TON_transferDelay.Q THEN
            "DEVICE".chainTransferR := TRUE;
            #statTransferDelay := FALSE;
        END_IF;

        //传送带B
        IF #R_TRIG_sensorD.Q THEN
            "DEVICE".conveyorB := TRUE;
        END_IF;
        IF #F_TRIG_sensorD.Q THEN
            "DEVICE".chainTransferR := FALSE;
        END_IF;
        IF #F_TRIG_sensorE.Q THEN
            "DEVICE".conveyorB := FALSE;
        END_IF;
```

图 3-7

当按下停止按钮自动停止的时候，所有传送带停止，如图 3-8 所示：

```
    ELSE
        "DEVICE".entryConveyor := FALSE;
        "DEVICE".conveyorA := FALSE;
        "DEVICE".conveyorB := FALSE;
        "DEVICE".chainTransferP := FALSE;
        "DEVICE".chainTransferR := FALSE;
    END_IF;
```

图 3-8

另外，简单的做了手动程序和指示灯程序，都是用赋值指令，类似于 LAD 的触点关联线圈，这里做的都比较简单，主要还是起一个抛砖引玉的作用，如图 3-9 所示：

```
//手动模式
IF #statManualMode THEN
    #statAutoStart          := FALSE;
    "DEVICE".conveyorA      := "DEVICE".manualForward;
    "DEVICE".conveyorB      := "DEVICE".manualForward;
    "DEVICE".entryConveyor  := "DEVICE".manualForward;
    "DEVICE".chainTransferP := "DEVICE".manualForward;
    "DEVICE".chainTransferR := "DEVICE".manualForward;
END_IF;
//指示灯
"DEVICE".lampStart   := #statAutoMode AND #statAutoStart;
"DEVICE".lampStop    := #statAutoMode AND NOT #statAutoStart;
"DEVICE".towerGreen  := #statAutoMode;
"DEVICE".towerYellow := #statManualMode;
```

图 3-9

四、课后测评

1、本节案例是非常基础性质的案例，用梯形图来做也非常的简单，主要还是体现一个 SCL 编程的思路，希望大家能够理解这种思路的转变。

2、需要理解案例中定时器、沿指令的用法，可以试着把定时器放在条件里做有条件调用，然后测试定时器的状态，这样对理解会有很大的帮助。

3、理解用 DB 块映射外部 IO 的思路，这样可以方便修改以及做更多灵活的操作，后续课程会加强这块的应用，这里也是一个抛砖引玉的作用。