BECKHOFF

TwinCAT FBD编程入门教程

Version 1.11

毕孚自动化设备贸易（上海）有限公司

2019 年 7 月

目录

一、 FBD 编程简介 1
二、 创建 FBD 工程 2
三、 FBD 编程入门 4
1、 网络（Network） 4
2、 赋值（Insert Assignment） 5
3、 运算块（box） 6
4、 插入标签（Insert label） 8
5、 跳转（Insert Jump） 9
6、 返回（Insert Return） 9
7、 取反（Negation） 10
8、 置位/复位（Set/Reset） 11
9、 边沿检测 (Edge Detection) 11
10、 插入分支（Insert Branch） 12
四、 FBD 编程实例 13
五、 常见问题 25

一、 FBD 编程简介

PLC 的编程语言主要有以下几种：梯形图（LD）、指令表（LI）、顺序功能图（SFC）、结构化文本（ST）、功能块图（FBD）、连续功能图编辑器（CFC）。这六种编程语言都是符合 IEC61131-3 标准的编程语言。

FBD 编程像电子电路的集成芯片一样，封装数据与逻辑，用户不考虑其内部具体流程，只需要考虑接口和使用。采用 FBD 的编程类似于现代面向对象编程的结构化特点，符合代码反复使用的要求，可以广泛的使用在以 PLC 为基础的各种控制系统之中。

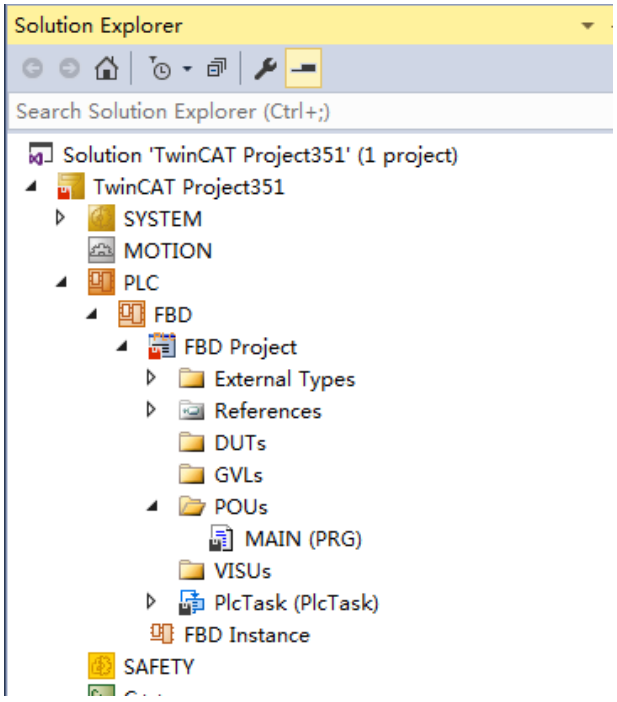
功能块图(FBD)将各种运算块进行连接，实现所需的控制功能。FBD 采用运算块和连线来实现，类似于电子线路图，图形化符号（box）代表运算块，通过图形化的 I/O 连接线段来给它分配输入输出信号。

FBD 的特点：

1. FBD 是采用图形的方法来表达系统中的功能，逻辑清晰，对于初次进行 PLC 系统软件设计的设计人员来说很好掌握
2. FBD 语言采用块的模式来实现其控制的特点，直观而且逻辑清晰，能够减少系统的设计时间
3. 很多的 PLC 联锁系统都用 FBD 语言来编程，并且 FBD 可以十分简单的表示复杂联锁系统的内部逻辑变量操作，减少 PLC 程序设计复杂度

二、 创建 FBD 工程

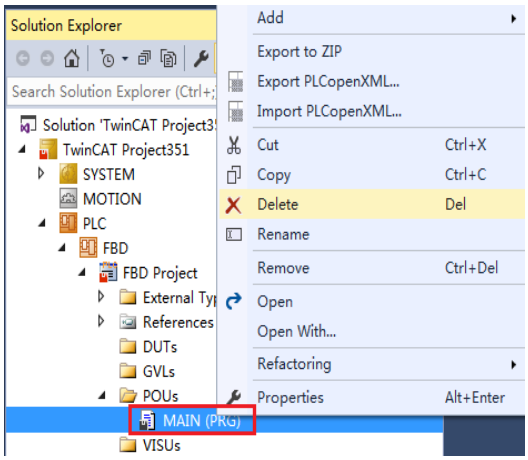
1. 打开 TwinCAT3 软件，新建工程并且在 PLC 下新建一个新 Project 项目

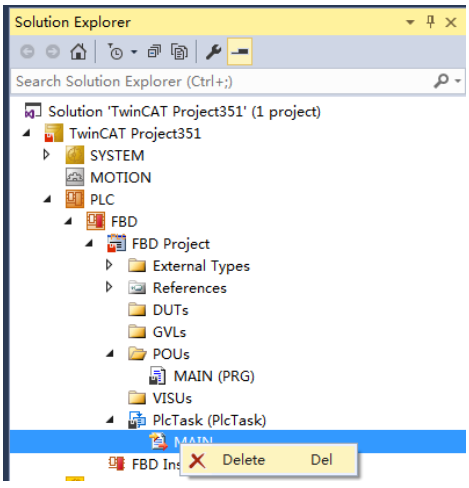


注意：

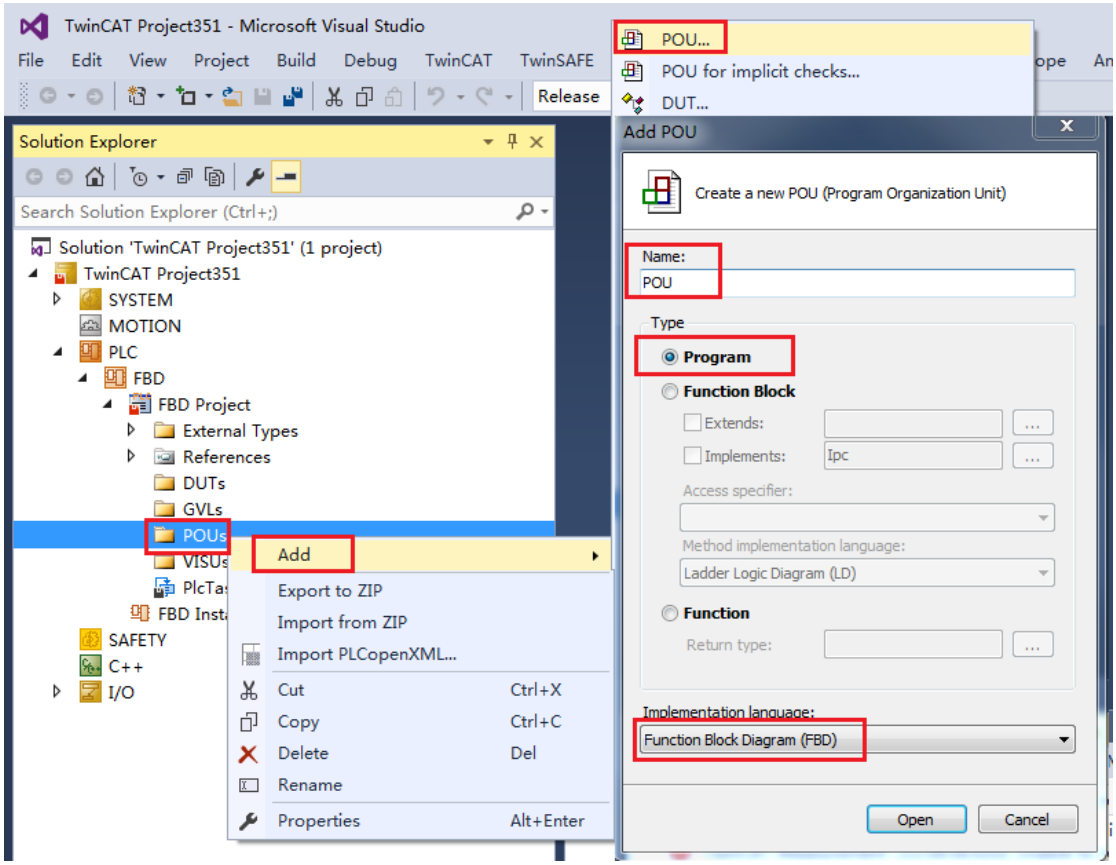
由于 TwinCAT3 软件新建的 PLC 程序是默认生成使用 ST 语言的主程序，所以需要先将原先的主程序删除，添加新的以 FBD 为编程语言的程序。

2. 删除 POUs 下方的 MAIN（PRG）主程序，和 PlcTask 下方的 MAIN 任务

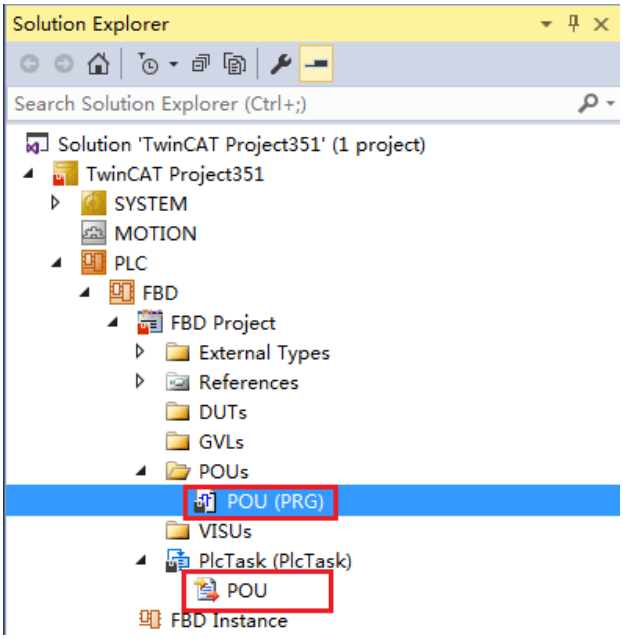




3. 在 POUs 下方添加新程序，并将编程语言手动选为 FBD 功能块图



4. 将创建好的 FBD 语言程序 POU 选中通过“拖动”，添加到 Plc Task 中



5. 这样就创建了一个以 FBD 为编程语言的 PLC 程序

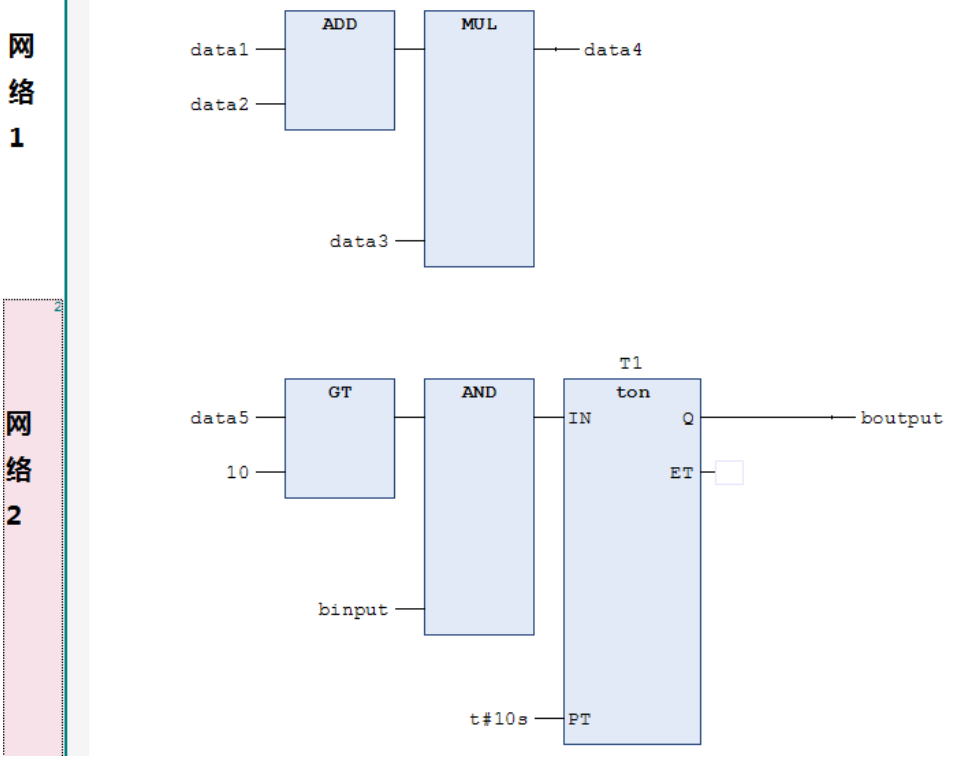
三、 FBD 编程入门

FBD 编程是由一些“网络”组成，每个“网络”有许多运算块组成，每个网络完成一段相对独立的运算，这些运算包括逻辑，算术，功能块，输入，输出，连线，跳转和返回等。

1、 网络（Network）

FBD 由一些“网络”组成，每个“网络”有许多运算块组成。
如图所示：

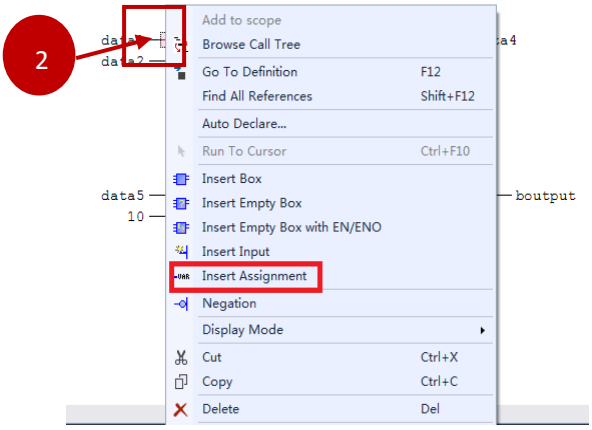
```
PROGRAM POU
VAR
    data1:INT;
    data2:INT;
    data3:INT;
    data4:INT;
    data5:INT;
    binput:BOOL;
    T1:TON;
    boutput: BOOL;
END_VAR
```



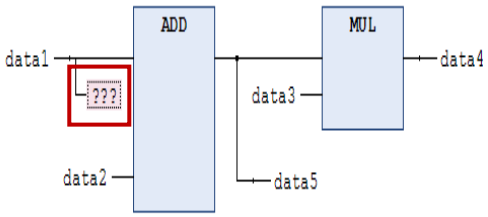
2、 赋值（Insert Assignment）

Insert Assignment 可插入一个赋值，根据插入的位置，一共分为 4 种插入。

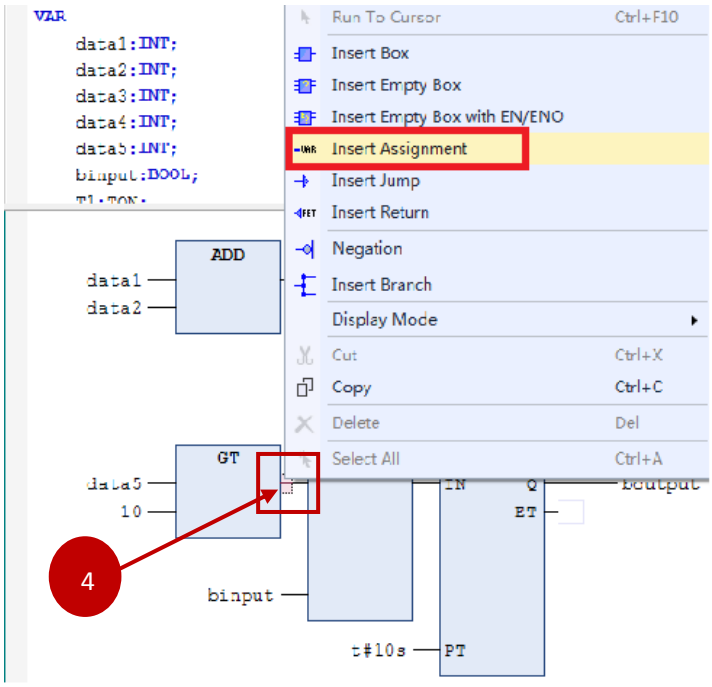
第一种插入方式：在选中的输入端（光标位置 2）右键，选择 Insert Assignment 插入赋值



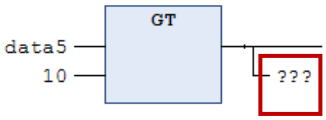
插入结果显示图



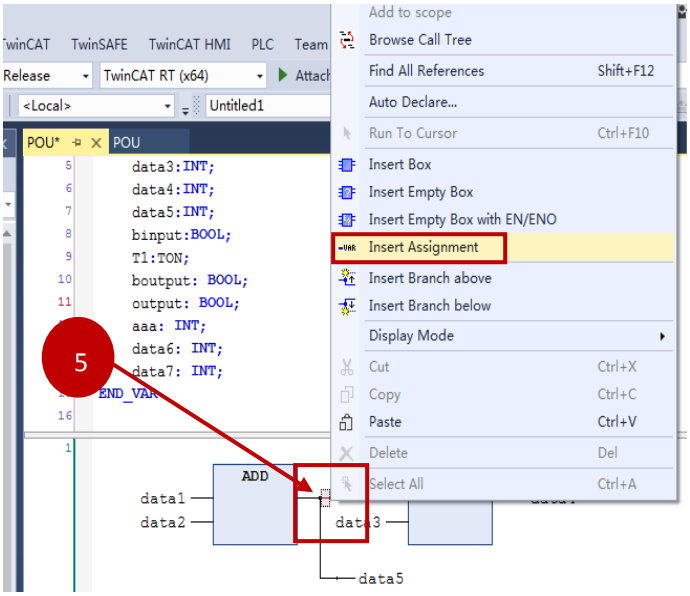
第二种插入方式：在选中的输出端（光标位置 4）右键，选择 Insert Assignment 插入赋值



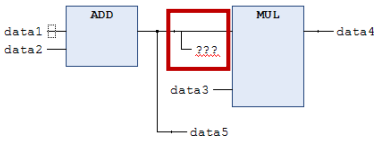
插入结果显示图



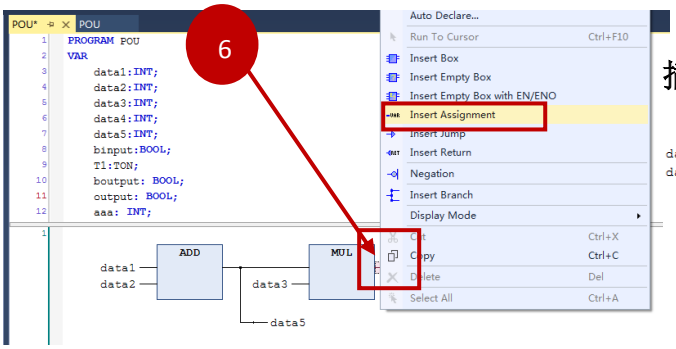
第三种插入方式：在选中的交叉线前面（光标位置 5）右键，选择 Insert Assignment 插入赋值



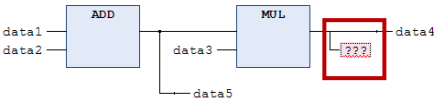
插入结果显示图



第四种插入方式：在网络的结束处（光标位置 6）右键，选择 Insert Assignment 插入赋值



插入结果显示图



插入赋值时，出现“???”文本，把”???”用控制器中的变量来替换。选中”???”，按住 F2 键，就可以实现变量的替换。

3、 运算块（box）

在网络中通过插入运算块，可以实现调用程序中的功能块，功能和接口。其中插入运算块有四种，分别是 Insert box，Insert Empty box，Insert box with EN/ENO，Insert Empty box with EN/ENO

Insert box:当选择 Insert box 时，输入助手窗口同时也被打开，只需要选择相应的运算块即可

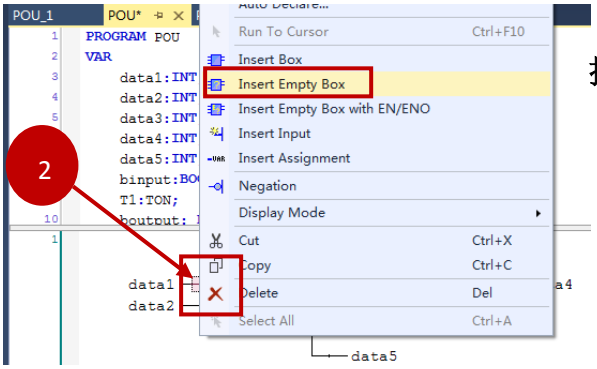
Insert Empty box：当选择 Insert Empty box 时，输入助手窗口不会被打开，需要自己手动输入相应的运算块名字或者手动进入输入窗口界面

Insert box with EN/ENO：当插入 Insert box with EN/ENO 时，运算块上输入管脚会多出 EN（使能）管脚，输出管脚会多出 ENO（使能输出）管脚，其余都和 Insert box 一样的当 EN 的输入管脚为 TRUE 时，该运算块才能执行，并且 ENO 会有相应的输出，反之亦然

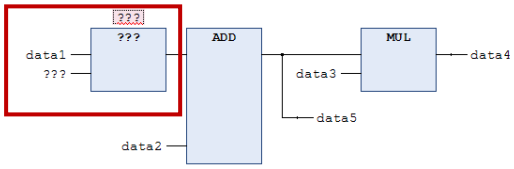
Insert Empty box with EN/ENO：当插入 Insert Empty box with EN/ENO 时，运算块上输入管脚会多出 EN（使能）管脚，输出管脚会多出 ENO（使能输出）管脚，其余都和 Insert Empty box 一样的。当 EN 的输入管脚为 TRUE 时，该运算块才能执行，并且 ENO 会有相应的输出，反之亦然

在网络中插入运算块，根据插入的位置，一共分为 3 种插入。

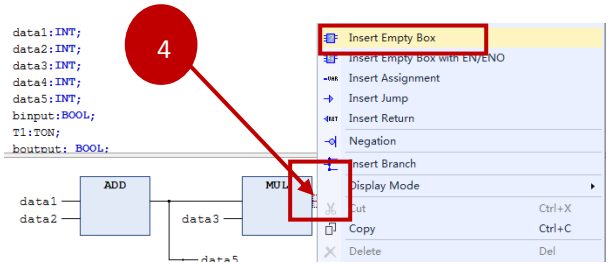
第一种插入方式：在选中的输入端（光标位置 2）右键，选择 Insert Empty Box 插入运算块，运算块会插入到该变量之前，运算块的第一个输入变量和新增的运算块的第一个输出变量会在现有的分支中连接起来。



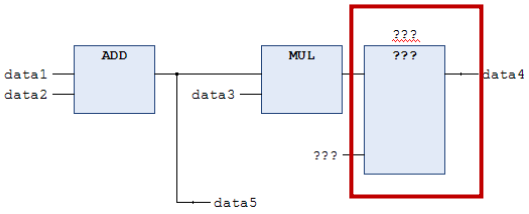
插入结果显示图



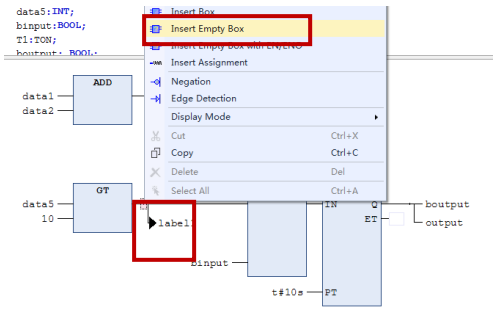
第二种插入方式：在选中的输出端（光标位置 4）右键，选择 Insert Empty Box 插入运算块，运算块会添加到该输出变量之后。新添加的运算块的第一个输入变量和原来运算块的第一个输出变量会在分支中连接起来。



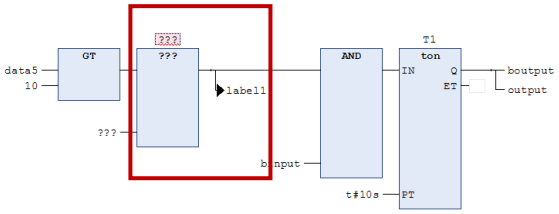
插入结果显示图



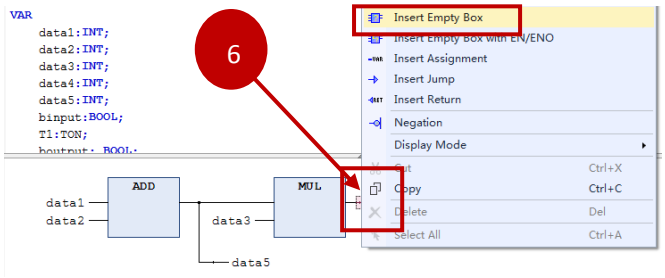
第三种插入方式：在选中的跳转或返回右键，选择 Insert Empty Box 插入运算块，新的运算块会插入到跳转块或返回块之前。新添加的运算块的第一个输入变量和原来运算块的第一个输出变量会在现有的分支中连接起来。



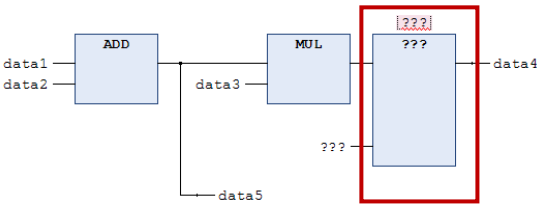
插入结果显示图



第四种插入方式：在网络末尾输出端（光标位置 6）右键，选择 Insert Empty Box 插入，运算块会被插入到网络的最后面，他的输入和原来的功能块将被连接起来。

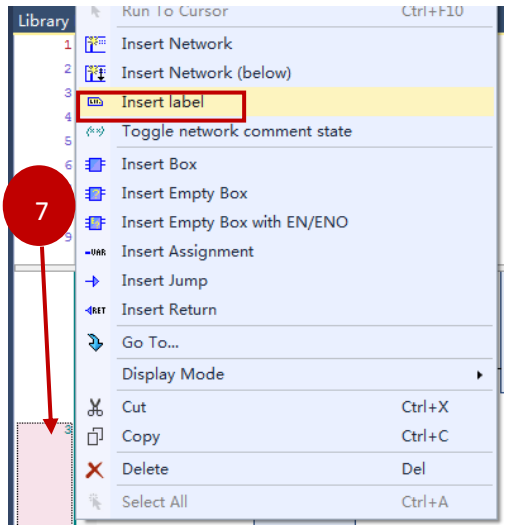


插入结果显示图

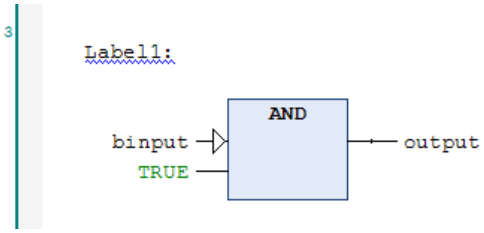


4、 插入标签（Insert label）

标签是网络标识符，标签是和跳转一起结合使用的，用于实现在不同的条件下执行不同的程序段。在网络的最左边位置（光标位置 7）右键，选择 Insert label 插入标签



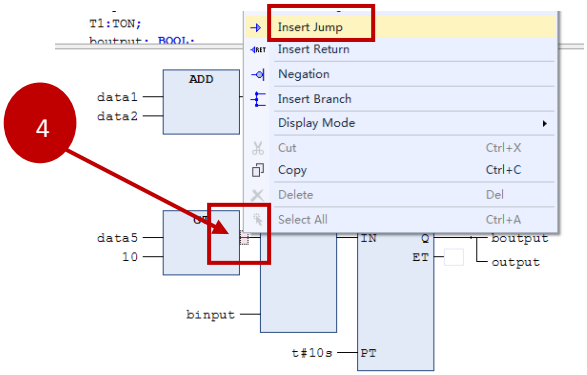
插入结果显示图



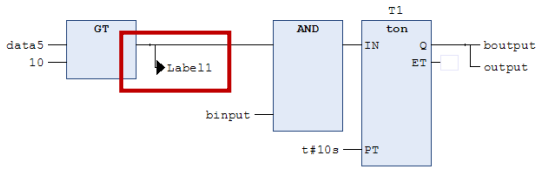
5、 跳转（Insert Jump）

在程序中使用跳转指令后，系统可以根据不同条件执行不同的程序段。在网络中插入一个跳转，根据插入的位置，一共分为 2 种插入。

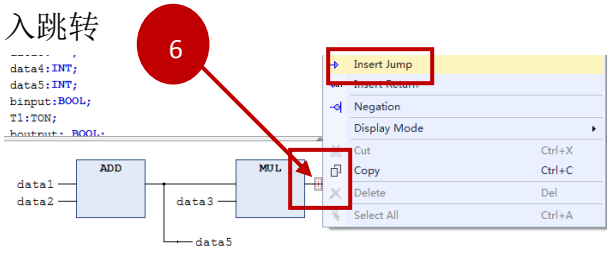
第一种插入方式：在选中的输出端（光标位置 4）右键，选择 Insert Jump 插入跳转



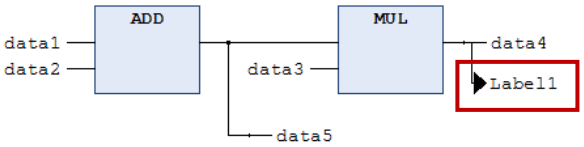
插入结果显示图



第二种插入方式：在网络末尾输出端（光标位置 6）右键，选择 Insert Jump 插入跳转



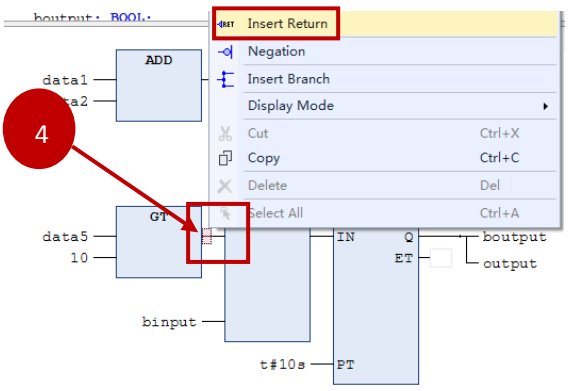
插入结果显示图



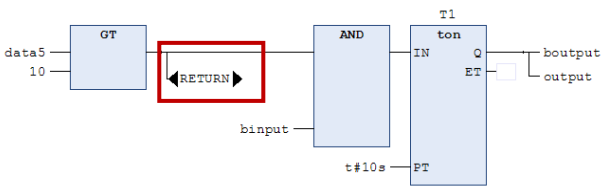
6、 返回（Insert Return）

在程序中使用返回指令后，当返回条件为 TRUE，系统就会把下面的程序段终止执行。在网络中插入一个返回，根据插入的位置，一共分为 2 种插入。

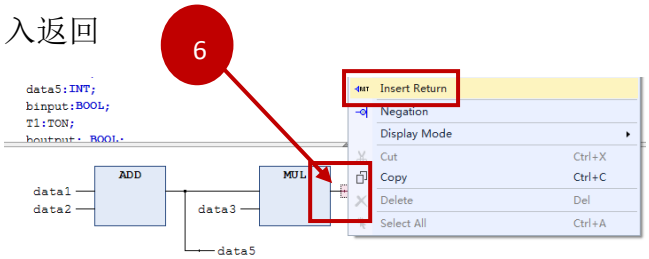
第一种插入方式：在选中的输出端（光标位置 4）右键，选择 Insert Return 插入返回



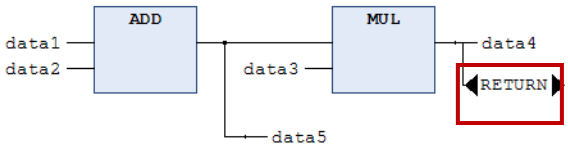
插入结果显示图



第二种插入方式：在网络末尾输出端（光标位置 6）右键，选择 Insert Return 插入返回



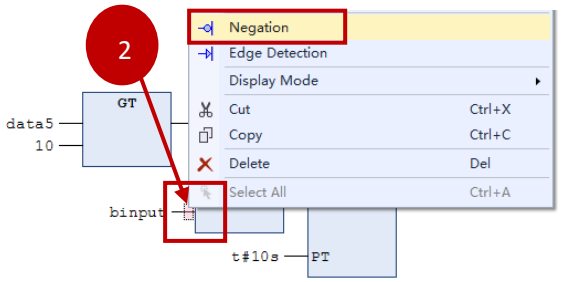
插入结果显示图



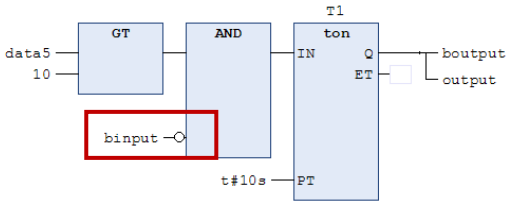
7、 取反（Negation）

用这个命令可以对输入、输出、跳转或返回指令进行取反操作，取反的符号在连接处多了一个小圆圈。在网络中插入一个取反，根据插入的位置，一共分为 2 种插入。

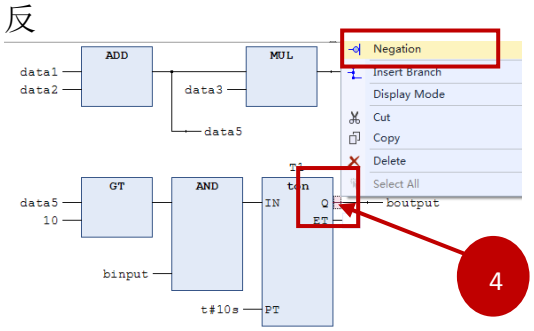
第一种插入方式：在选中的输入端（光标位置 2）右键，选择 Negation 插入取反，这个输入将会被取反



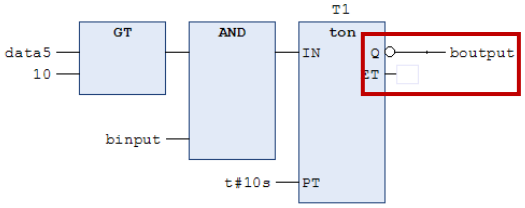
插入结果显示图



第二种插入方式：在选中的输出端（光标位置 4），右键选择 Negation 插入取反，这个输出将会被取反如果一个跳转或返回被标记，那么跳转或返回将被取反



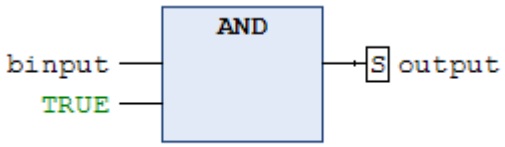
插入结果显示图

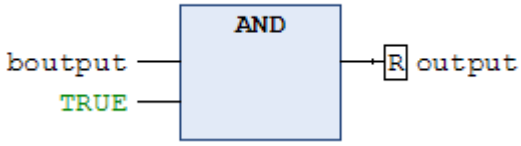


注意：取反可以通过重新取反来取消。

8、 置位/复位（Set/Reset）

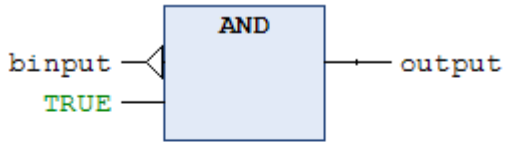
用这个命令可以实现对输出的置位和复位，置位的输出用[S]表示，复位的输出用[R]来表示，单击 Set/Reset，输出端设置为置位，双击 Set/Reset，输出设置为复位，同时输出端显示为[R]，第三次点击 Set/Reset，则会恢复到正常状态。

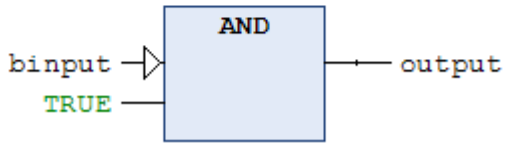




9、 边沿检测（Edge Detection）

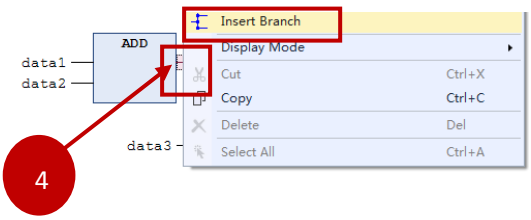
用这个命令可以实现输入输出的上升沿检测和下降沿检测，上升沿检测用  表示，下降沿检测用  表示，单击 Edge Detection，则是上升沿检测，第二次点击 Edge Detection，则表示下降沿检测，第三次点击 Edge Detection，则会恢复到正常状态。



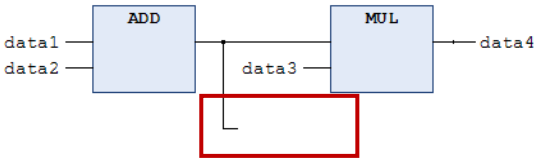


10、 插入分支（Insert Branch）

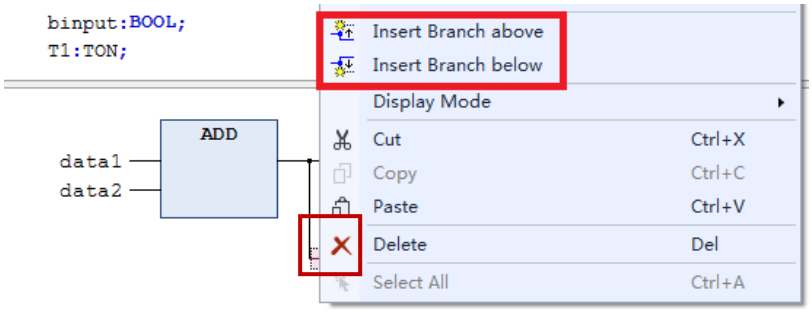
用这命令可以实现多分支的输出。在网络中插入分支，需要在选中的输出端（光标位置 4）右键，选择 Insert Branch 进行插入分支



插入结果显示图



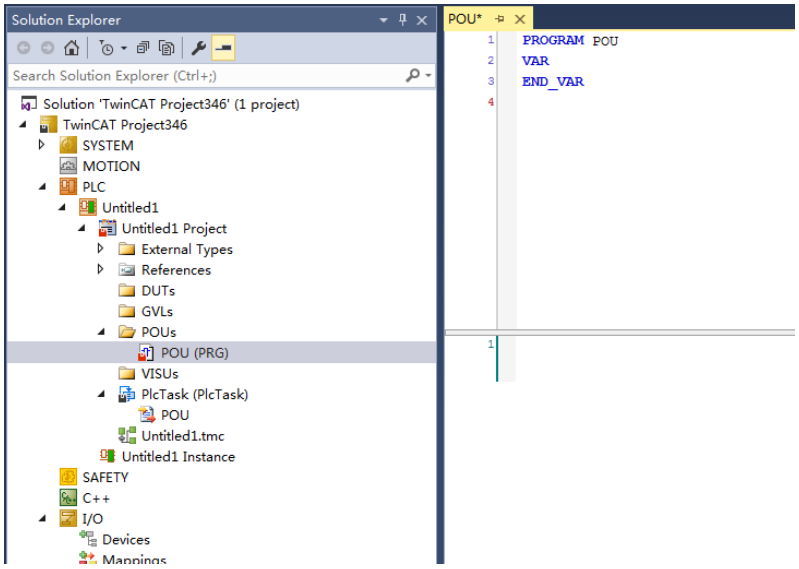
如果还想添加多个分支的话，可以通过 Insert Brach above（向上添加分支），Insert Brach below（向下添加分支）。



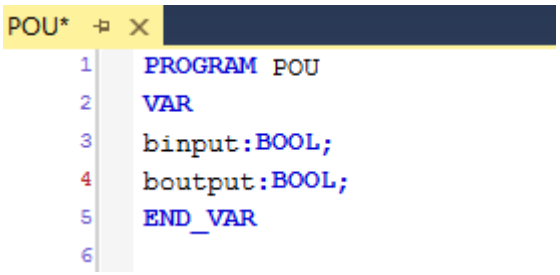
四、 FBD 编程实例

1. 编写一个简单的赋值语句，把将 binput 赋值给 boutput，即：boutput:= binput。

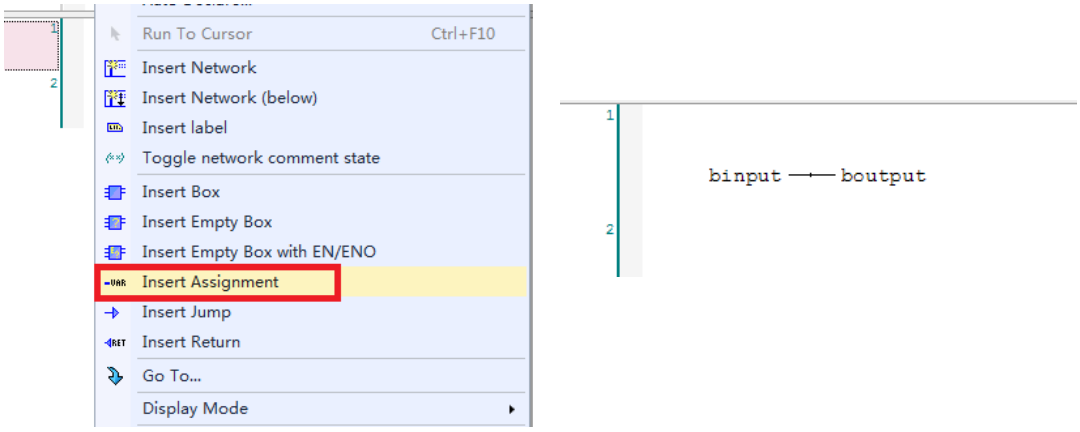
1) 新建项目工程



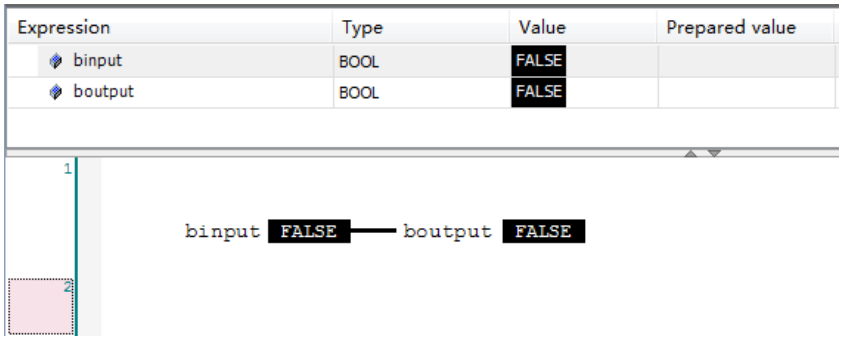
2) 在变量申明区新建两个变量，binput 和 boutput

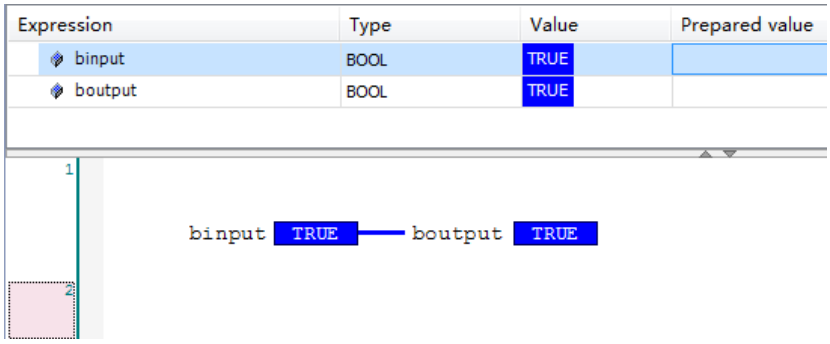


3) 编写相应的控制程序，其中简单的赋值通过 Insert Assignment 来实现，或者可以通过函数 MOVE 来实现。



4) 以上就完成了简单程序的编写，编译查看是否报错，如无报错，登入并运行程序。将 binput 置为 true 后，boutput 也立即置为 true



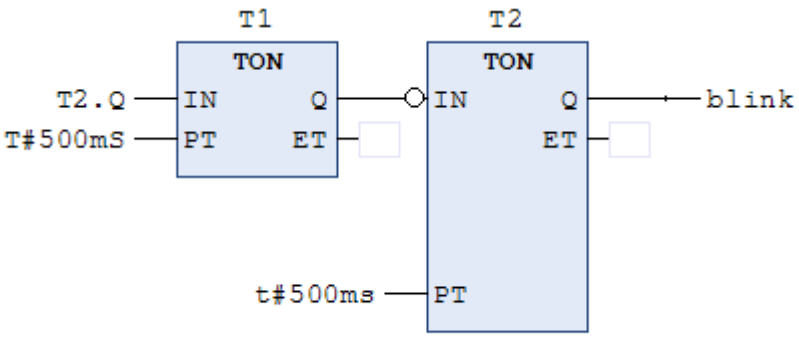


2. 实现一个周期为 1s 的 50%占空比脉冲，即：将 blink 变量设置成以周期为 1s 的 50%占空比脉冲。

1) 在变量申明区新建三个变量，T1,T2,blink。
T1 延时功能块，用于实现高电平输出时间的控制
T2 延时功能块，用于实现低电平输出的时间控制
Blink 变量就是占空比输出变量

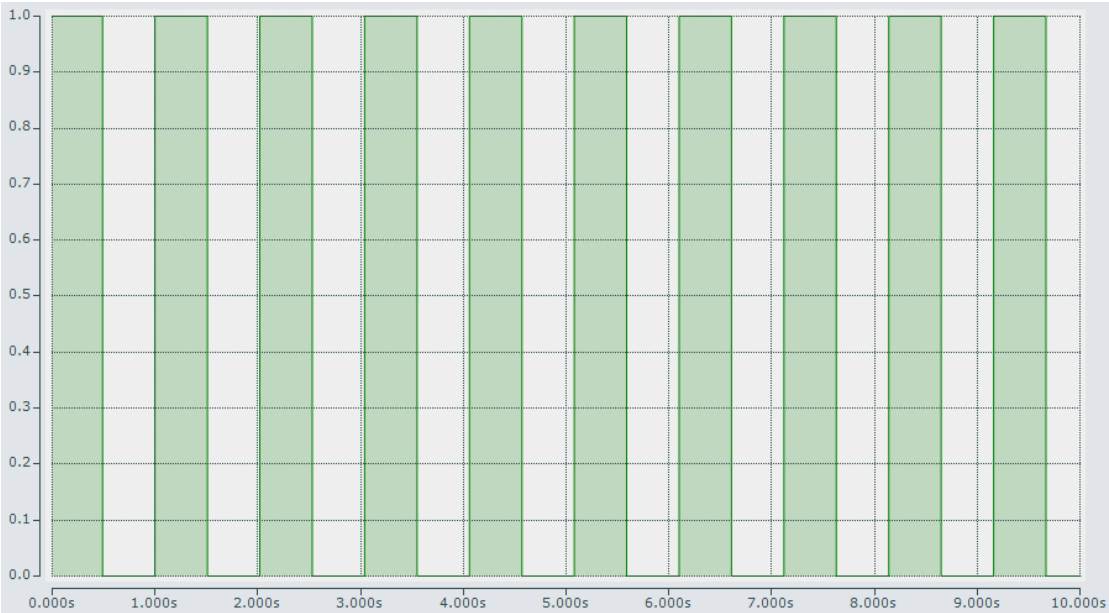
```
POU
1  PROGRAM POU
2  VAR
3  T1:TON;
4  T2:TON;
5  blink:bool;
6  END_VAR
7
```

2) 编写相应的控制程序，其中需要结合 T1,T2 这两个功能块才能实现该功能。其中先让 T2 功能块执行，实现 500ms 的延时，实现低电平 500ms 的输出；接着让 T1 和 T2 功能块同步执行，也是实现 500ms 的延时，实现高电平 500ms 的输出。



注意：如果要实现其他的占空比，只需要修改 T1,T2 的延时时间即可。

3) 完成了简单程序的编写，编译查看是否报错，如无报错，登入并运行程序。使用 scope view 来监控 blink 值的变化。



3. 用一个按钮控制两盏灯，当不按按钮的时候，两盏灯全灭，当按一下开关的时候，第一盏灯亮，再次按下开关的时候，两盏灯全亮，第三次按下开关的时候，两盏灯全灭，再次重复该循环。

   注意：本次编程建议使用到 FBD 中的跳转与返回功能。

1) 在变量申明区建立 5 个变量，分别是 SB1,CTU1,imdoe,boutput1,boutput2.

   SB1：是按钮开关

   CTU1:加计数功能块

   imdoe:中间变量，用于亮灯模式的切换

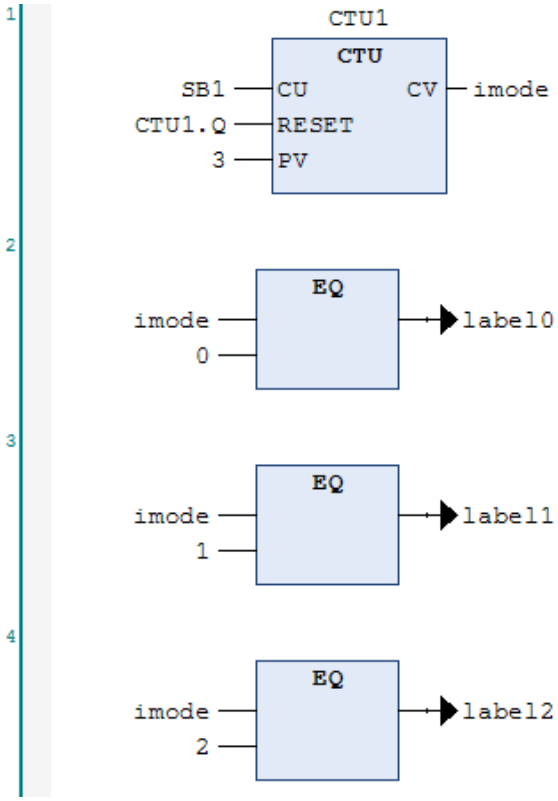
   boutput1:第一盏灯

   boutput2:第二盏灯

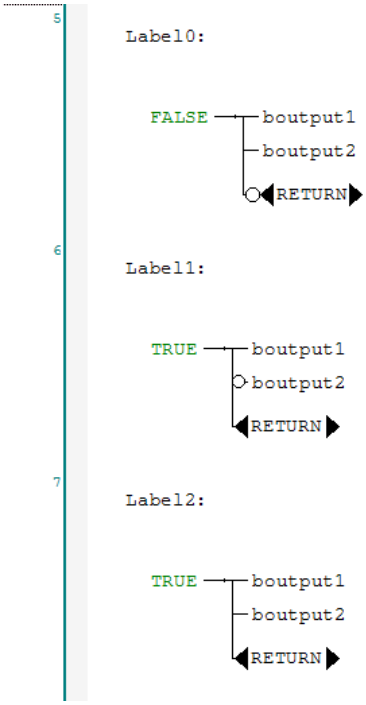
```
POU*
1  PROGRAM POU
2  VAR
3      SB1: BOOL;
4      CTU1:CTU;
5      imode: WORD;
6      boutput1: BOOL;
7      boutput2: BOOL;
8  END_VAR
```

2) 编写相应的控制程序，本次程序中亮灯模式分为三种情形，第一种是：两盏灯全部灭掉；第二种是：第一盏灯亮，第二盏灯灭；第三种是：两盏灯全部亮。接着我们通过计数功能块和比较指令，来实现模式的切换。

   程序第一部分：通过计算开关的次数，来确认是那种亮灯模式



程序第二部分：通过标签和跳转来实现不同方式的亮灯，当 imode 等于 0 时，跳转到 label0，两盏灯都灭掉；当 imode 等于 1 时，跳转到 label1，第一盏灯亮，第二盏灯灭；当 imode 等于 3 时，跳转到 label2,第一盏灯和第二盏灯全亮。



3) 编写可视化界面，用于快捷观察程序执行

其中画面中的按钮对应变量 SB1，画面中的第一盏灯对应变量 boutput1，画面中的第二盏灯对应变量 boutput2。

4) 完成了简单程序的编写，编译查看是否报错，如无报错，登入并运行程序，打开可视化界面，观看程序运行状况。

a) 当不按按钮的时候，所有灯全部灭掉

b) 只按一下按钮，第一盏灯亮，第二盏灯灭

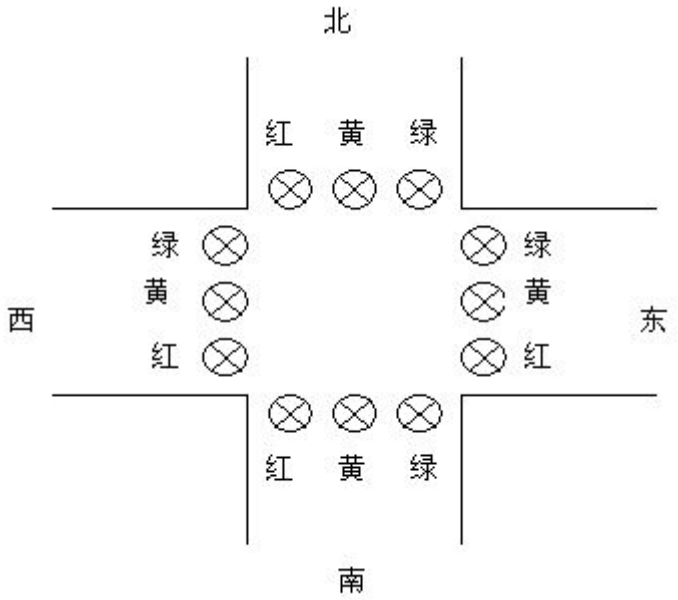
c) 再次按下按钮，两盏灯全亮

d) 又再次按下按钮，两盏灯又全灭

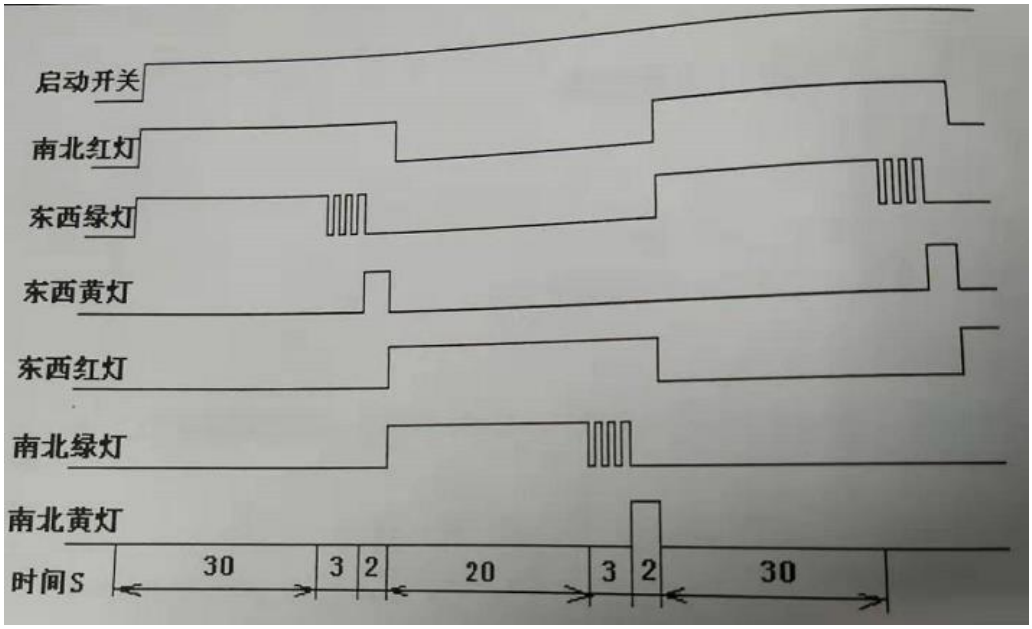
4. 十字路口交通控制

实例要求：

下图是城市十字路口交通灯示意图，交通灯的控制要求如下：



当启动开关接通时，交通灯系统开始工作。先东西绿灯亮，南北红灯亮。南北红灯亮维持 35s，在南北红灯亮的同时，东西绿灯也亮，并维持 30s，到 30s 时，东西绿灯闪亮，闪亮周期为 1s（亮 0.5S，灭 0.5s）。绿灯闪亮 3S 后熄灭，东西黄灯亮，维持 2s，到达 2s 时，东西黄灯灭，红灯亮，同时南北红灯灭，绿灯亮。东西红灯维持 25s，南北绿灯亮维持 20s，到 20s 时，南北绿灯闪亮 3s 后熄灭，南北黄灯亮，维持 2s，到 2s 时，南北黄灯灭，红灯亮，同时东西红灯灭，绿灯亮，进入第二周期的动作。此后周复始地循环。
当启动开关断开时，所有交通灯熄灭。
交通灯状态示意如下图所示：



编程提示：
1. 需要利用两个定时器实现周期为 1s（亮 0.5S，灭 0.5s）的秒脉冲
2. 需要利用 6 个定时器来控制红绿灯的亮的时间及亮、灭顺序。

编程讲解：

1) 在变量申明区建立 10 个变量

bStart：启动按钮；

arrTON:延时功能块数组，总共包括 8 个延时功能块，前 6 个延时功能块用于控制红黄绿灯亮的时间，后面两个功能块实现秒脉冲的功能；

arrTONoutput 也是数组变量，对应 arrTON 前 6 个延时功能块的输出；

p_1s：是秒脉冲；

bEW_Red，bEW_Green，bEW_Yellow：分别是东西红灯，绿灯和黄灯；

bNS_Red，bNS_Green，bNS_Yellow：分别是南北红灯，绿灯和黄灯。

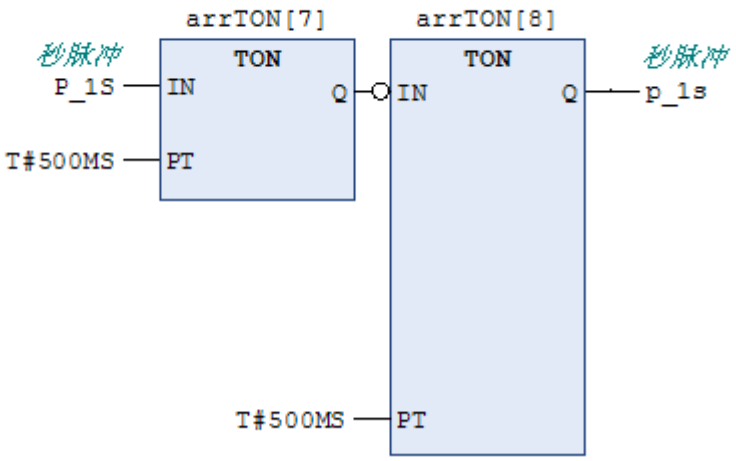
POU*

```
PROGRAM POU
VAR
    bStart: BOOL;
    arrTON: ARRAY[1..8] OF TON;
    arrTONoutput: ARRAY[1..6] OF BOOL;
    p_1s: BOOL;//秒脉冲
    bEW_Red: BOOL;//东西红灯
    bEW_Green: BOOL;//东西绿灯
    bEW_Yellow: BOOL;//东西黄灯
    bNS_Red: BOOL; //南北红灯
    bNS_Green: BOOL;//南北绿灯
    bNS_Yellow: BOOL;//南北黄灯
END_VAR
```

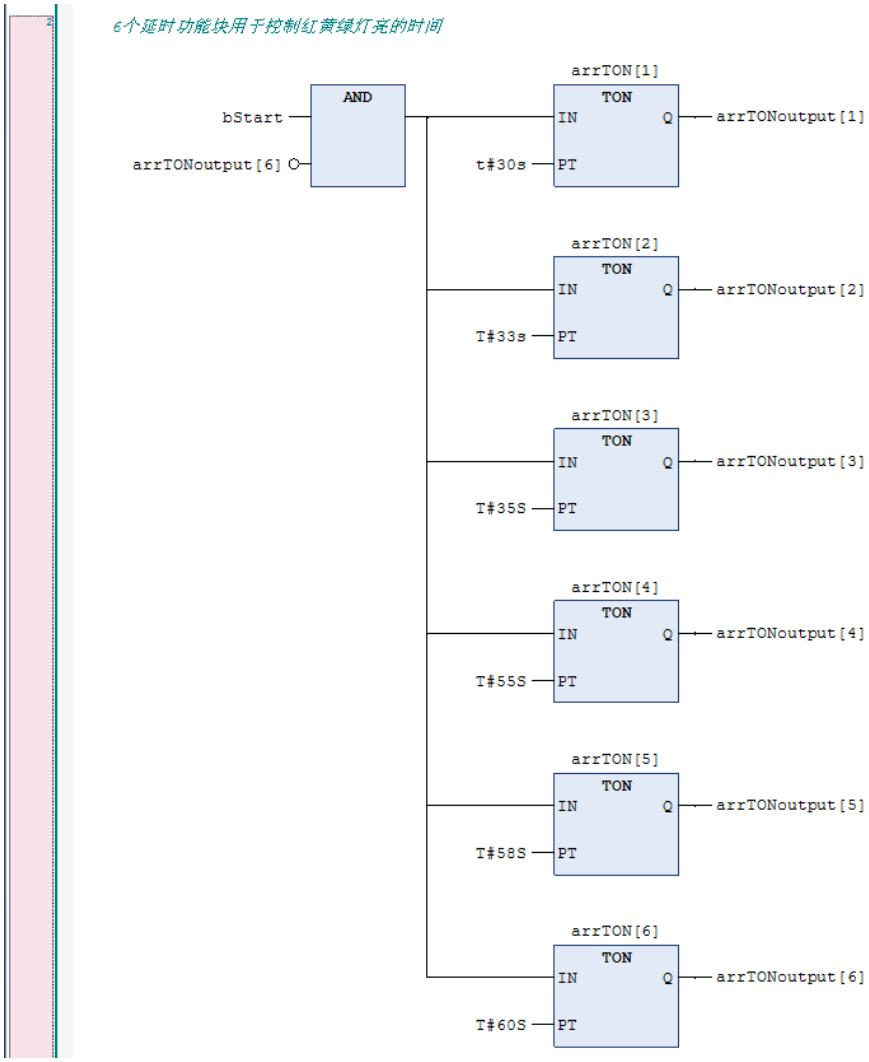
2) 编写相应的控制程序，本次程序分为三部分

第一部分：用于实现 1s 周期的秒脉冲（亮 0.5s，灭 0.5s）

```
用于实现1s周期的秒脉冲（（亮0.5S，灭0.5s）
```



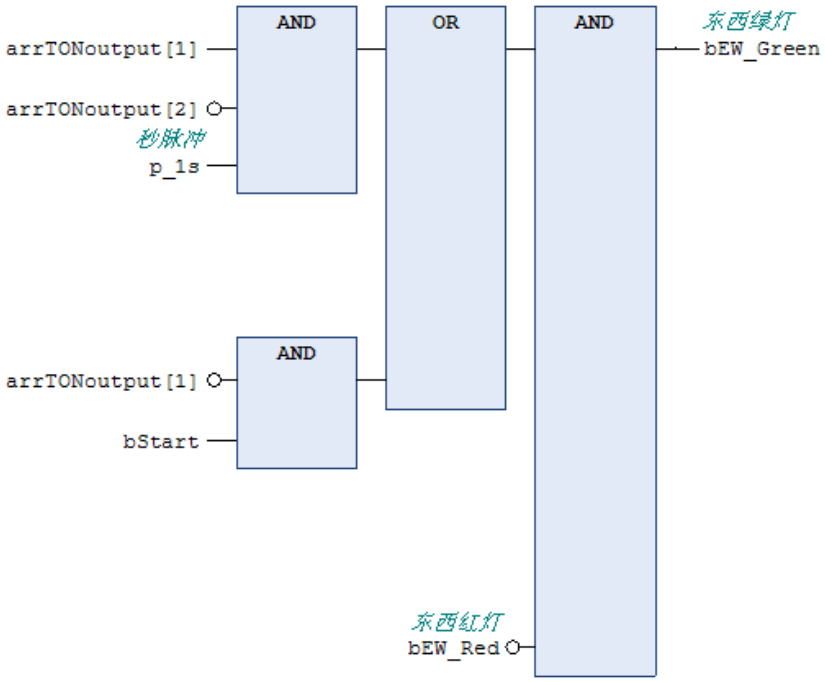
第二部分：用 6 个延时功能块用于控制红黄绿灯亮的时间



第三部分：通过与或非来实现红黄绿灯亮、灭顺序

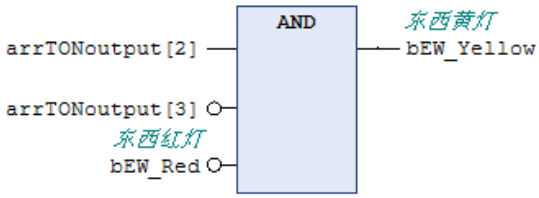
3

控制东西绿灯亮，分成两部分第一部分亮30s，接着闪3下



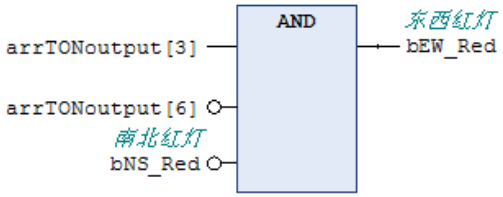
4

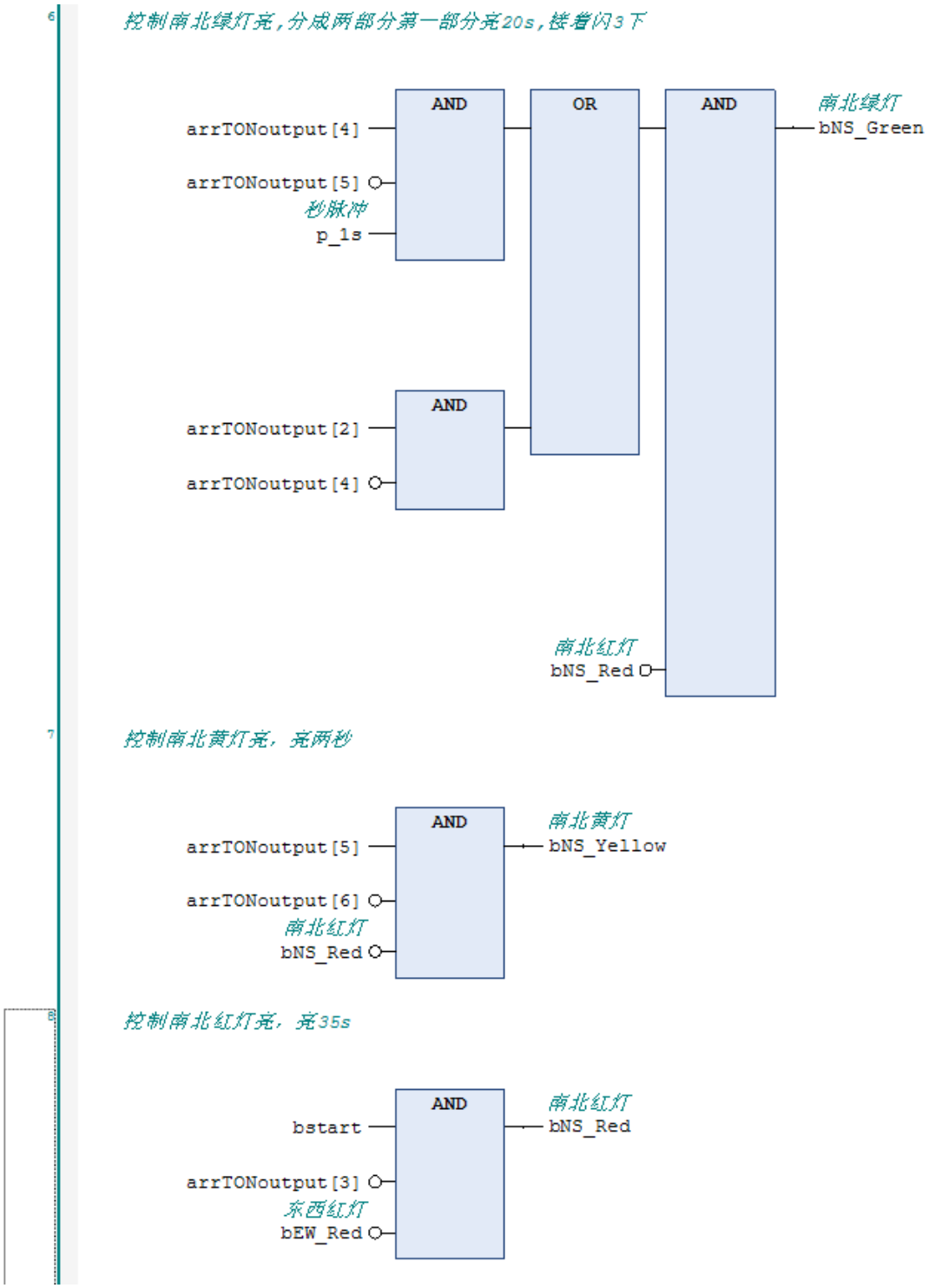
控制东西黄灯亮，亮两秒



5

控制东西红灯亮，亮25s



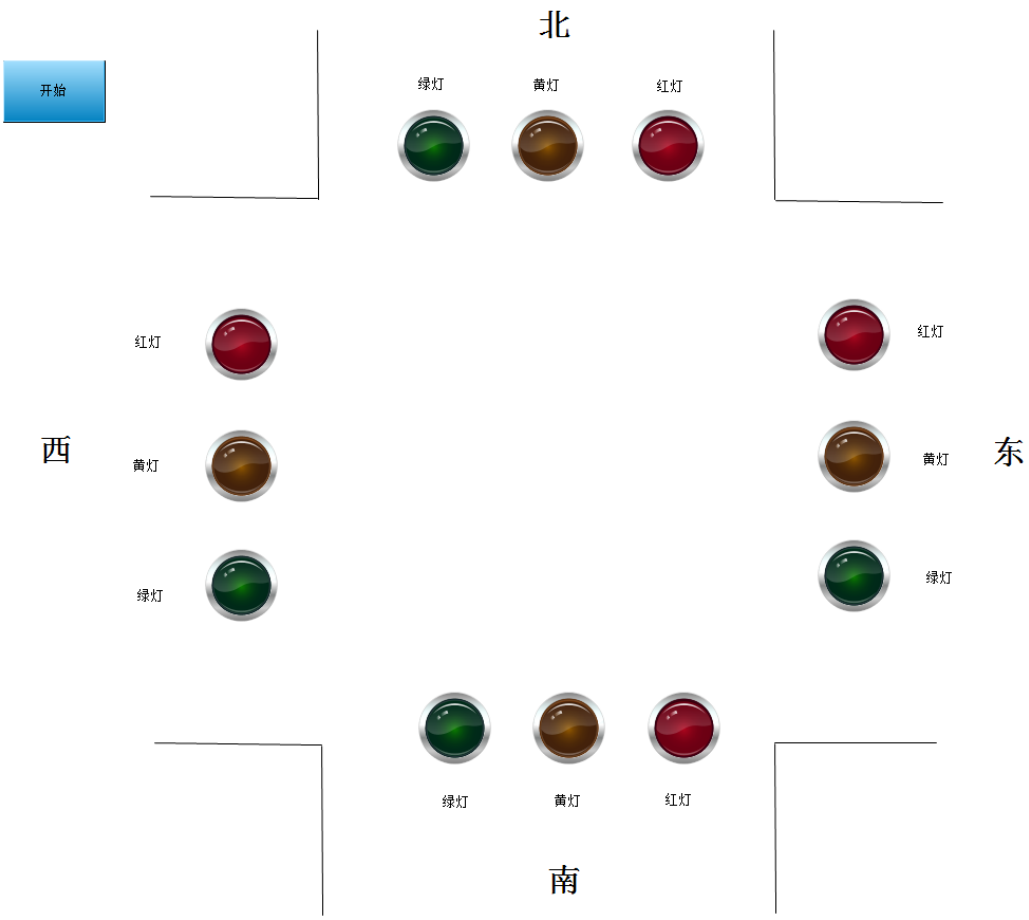


5) 编写可视化界面，用于快捷观察程序执行

开始按钮对应程序变量 bStart

南北方向的绿灯，黄灯和红灯对应的程序变量 bEW_Red，bEW_Green，bEW_Yellow

东西方向的绿灯，黄灯和红灯对应的程序变量 bNS_Red，bNS_Green，bNS_Yellow

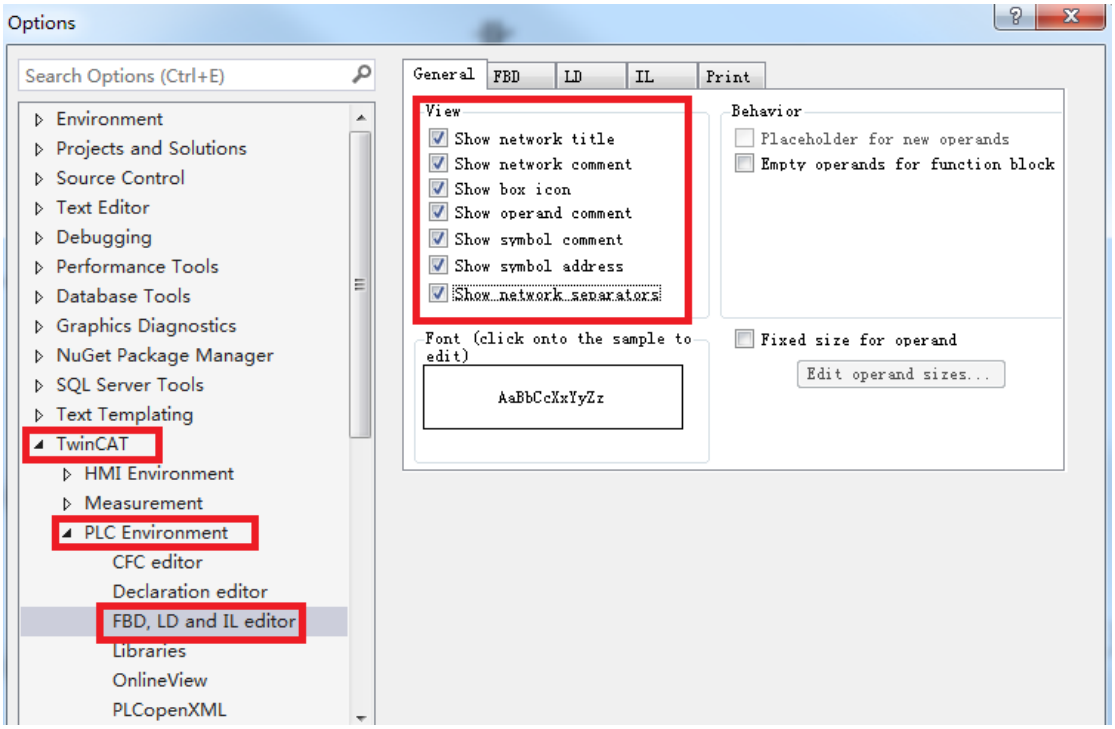

北
开始
绿灯
黄灯
红灯
西
红灯
黄灯
绿灯
东
红灯
黄灯
绿灯
绿灯
黄灯
红灯
南

五、 常见问题

1. 怎么在 FBD 中如何添加注释？

在 FBD 编程时可以对网络，操作数，符号做注释，其中这些注释可以是中文的。

在 tools-options 进行设置



Show network title:显示网络标题

Show network comment： 显示网络注释

Show box icon:显示功能块图标

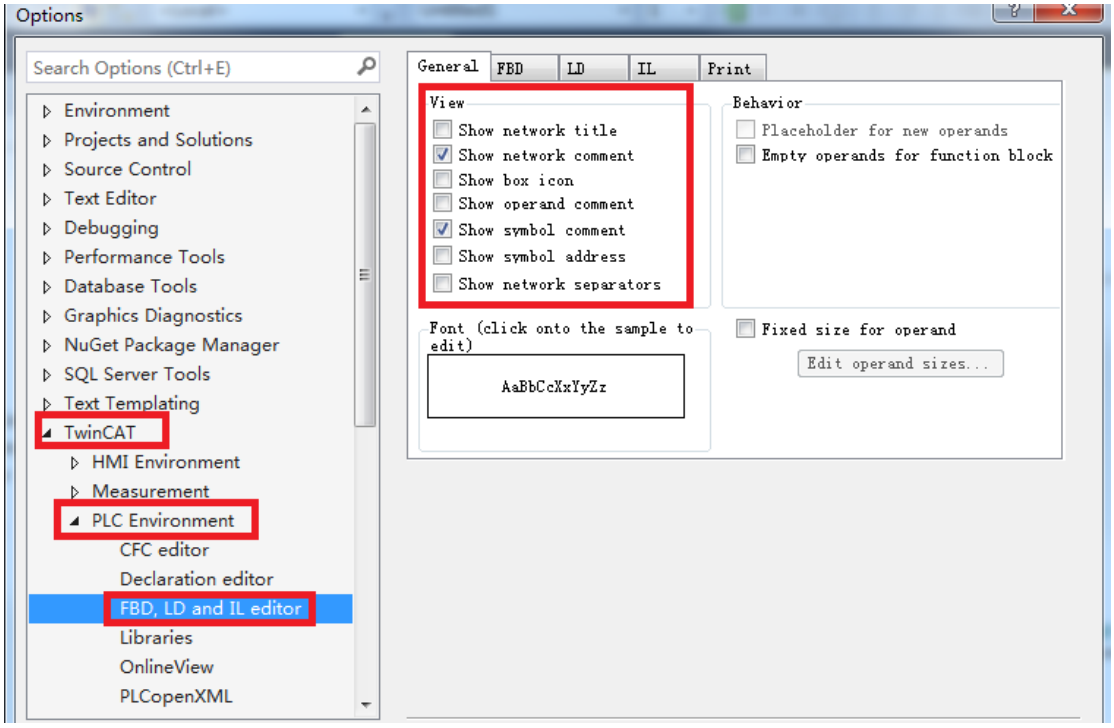
Show operand comment:显示操作数的注释

Show symbol comment:显示变量注释

Show symbol address:显示变量地址

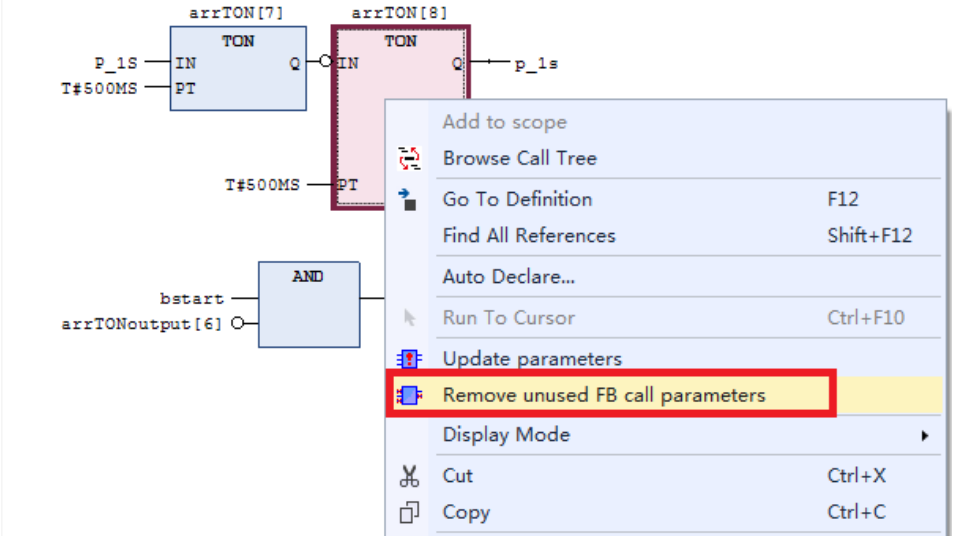
Show network separators:显示网络分隔符

在十字路口交通控制，我们设置了显示网络注释和显示变量注释，所以在十字路口交通控制程序中就会有注释显示



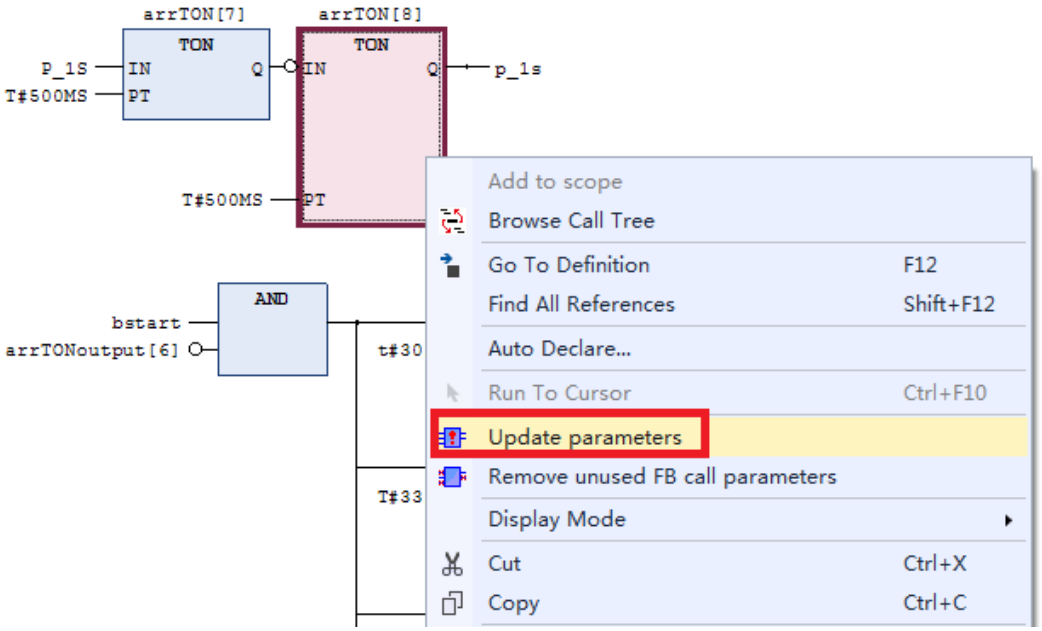
2. 在 FBD 编程的时候，需要把函数或者功能块没有用到的输入输出管脚不显示，怎么设置？

选中相应的函数或者功能块右键，单击 Remove unused FB call parameters



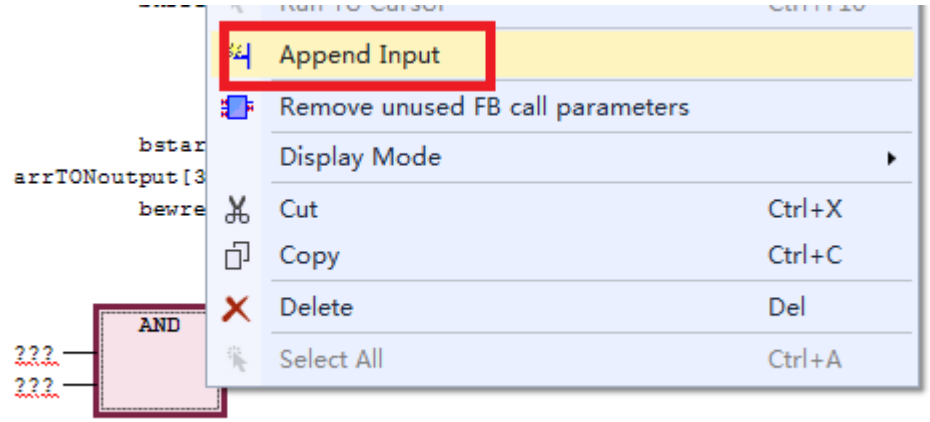
3. 在 FBD 的编程的时候，之前把函数或者功能块的没有用到的输入输出管脚不显示，现在由于需要修改程序，要把不显示的输入输出管脚再次显示，怎么实现？

选中相应的函数或者功能块右键，单击 Update parameters



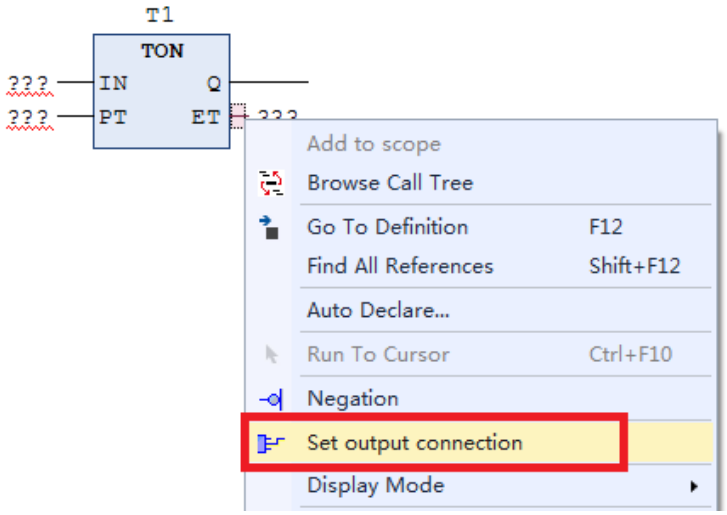
4. 当使用 AND 函数的时候，发现只有 2 个输入管脚，但是我需要 4 个变量进行与操作，怎么设置？

选中相应的函数或者功能块右键，单击 Append Input

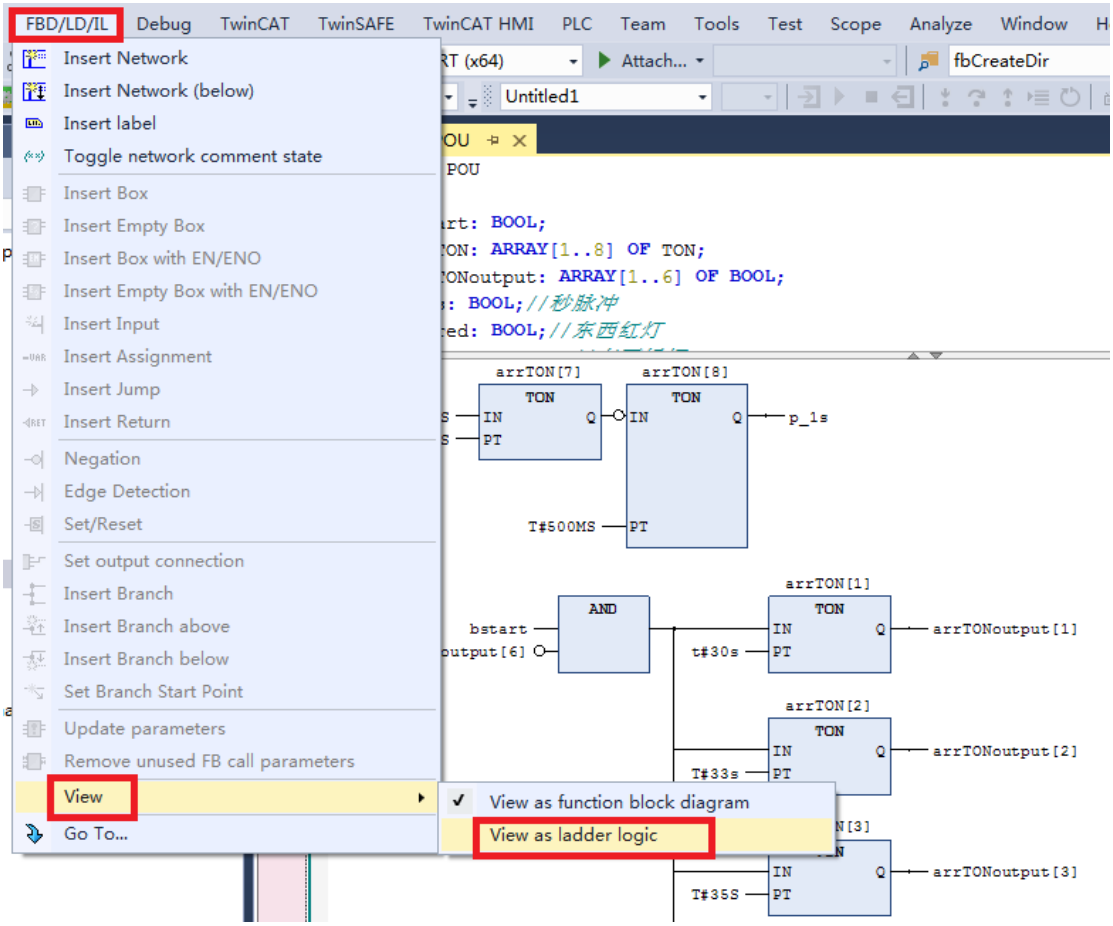


5. 在 FBD 编程的时候，怎么把函数或者功能块的第二个输出管脚用于连接下一个函数或者功能块？

选中第二个输出管脚右键，单击 Set output connection



6. 怎么把 FBD 编程语言转换成梯形图？
   FB/LD/IL---View---View as ladder logic



上海（ 中国区总部）
中国上海市静安区汶水路 299 弄 9号（市北智汇园）
电话：021-66312666　　传真：021-66315696　　邮编：200072

北京分公司
北京市西城区新街口北大街 3 号新街高和大厦 407 室
电话：010-82200036　　传真：010-82200039　　邮编：100035

广州分公司
广州市天河区珠江新城珠江东路16号高德置地G2603室
电话：020-38010300/1/2　传真：020-38010303　　邮编：510623

成都分公司
成都市锦江区东御街18号 百扬大厦2305 房
电话：028-86202581　　传真：028-86202582　　邮编：610016

请用微信扫描二维码
通过公众号与技术支持交流

倍福中文官网：
http://www.beckhoff.com.cn/
倍福虚拟学院：
http://tr.beckhoff.com.cn/

招贤纳士：job@beckhoff.com.cn
技术支持：support@beckhoff.com.cn
产品维修：service@beckhoff.com.cn
方案咨询：sales@beckhoff.com.cn