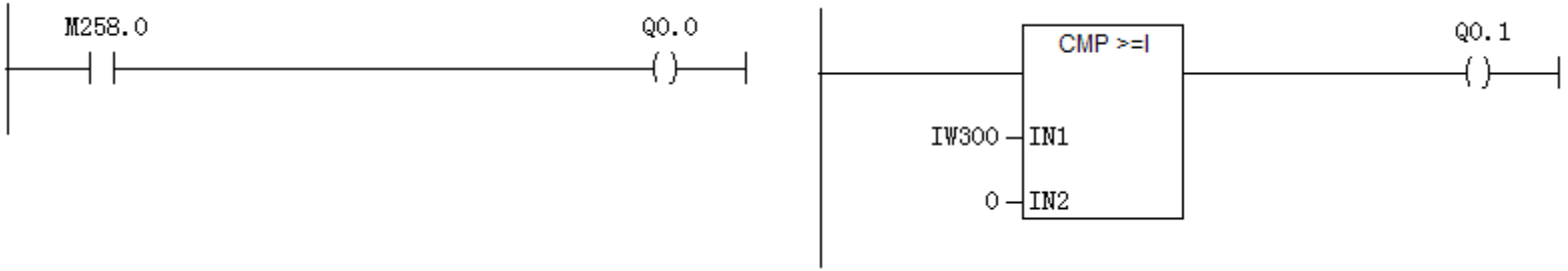
PLC编程常见问题(S7-300)

简单错误：

1. 地址超范围



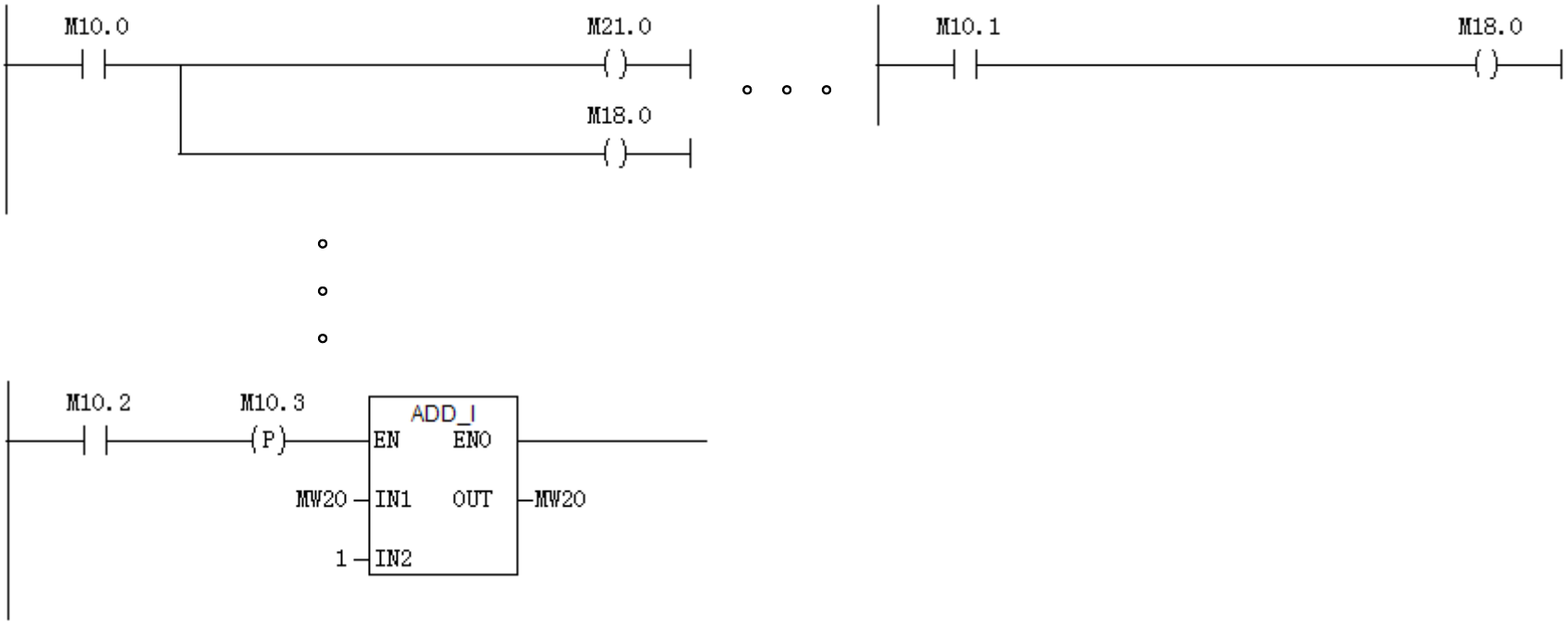
上面这两条语句语法没任何错误，但是如果我们下载到CPU314当中，那将无法完成。因为CPU314最大M区地址只支持到MB255,CPU315-2DP的模拟量通道集中式输入/输出最大支持256。

注：如果使用仿真器，无法检测出该错误，PLC仍然能够运行。

PLC编程常见问题(S7-300)

简单错误：

2.地址重叠

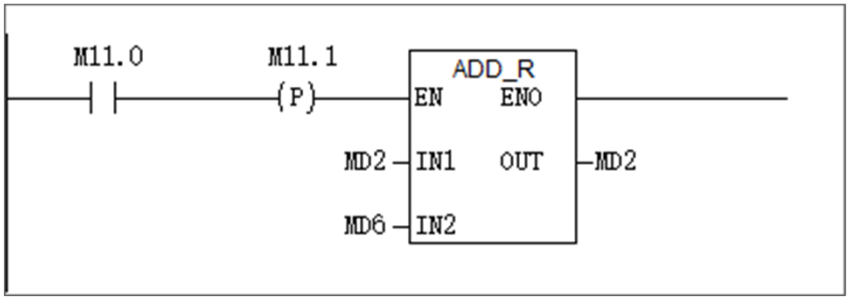


上述的地址重叠是PLC编程中比较容易犯的错误，这些语句分布在程序的不同位置，编程时较难察觉，在系统试验时也不一定能检查出来。因此我们在做变量点表时需要对变量地址进行合理分配。在写程序时养成良好的变量使用习惯。

PLC编程常见问题(S7-300)

简单错误：

3.数据类型匹配与浮点运算



瞬时流量模拟累计错误程序案例：

MD2:累计流量存储值

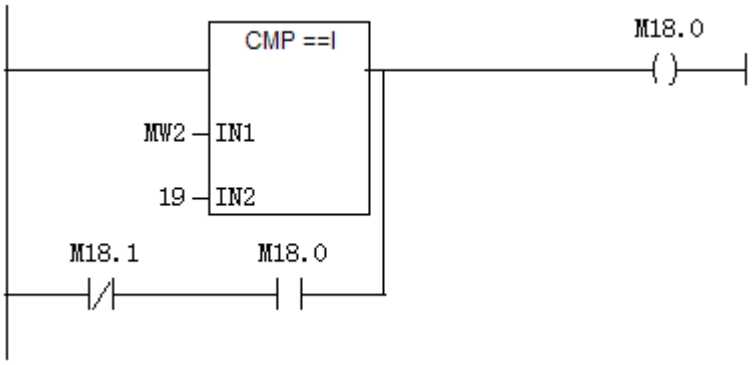
MD6:流量瞬时值

在流量累计或其他累计计算编程中常会遇到实数加法的问题。
上述程序案例是否存在问题？
在流量累计初期运行是正常的，因为流量累计初始值及瞬时值都为一个很小的浮点数，两数相加结构正确。但经过一段时间后，两数相差很大，执行加法操作后，瞬时流量的数值将被忽略掉（如9999990.0+0.2）。因此在编程时比避免数量级相差太多的浮点数之间进行运算。

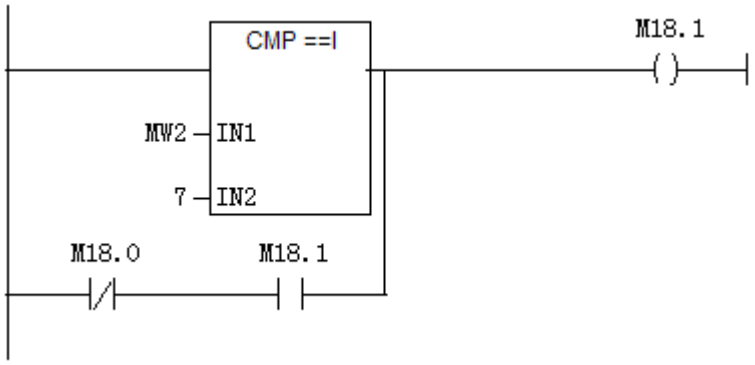
PLC编程常见问题(S7-300)

简单错误：

4. 比较指令



路灯开启



路灯关闭

路灯控制比较指令错误程序案例：

MW2:小时时间

在编程过程中经常需要使用到比较指令，包括整数比较和浮点数比较等。
上述程序原意是时间到了晚上19点开灯，早上7点关灯。但就没考虑到如果
时间跨过19点那么该段程序就不会执行了。如果刚好18点停电，20点供电恢复
那么此时的路灯也就不会亮了。此类问题在数值范围变动比较大的模拟量数值
比较和浮点数值比较中更容易出问题。
很多网友反馈此程序案例只是不严谨，对实际使用影响不大。其实我们很多程序就是
存在类似的不够严谨的地方，正常运行是没问题的，如果极端偶然的情况下就可能
产生不合理的动作。

PLC编程常见问题(S7-300)

简单错误：

5.语句执行先后顺序1

PLC扫描程序时由头至尾依次执行的，所以编程人员必须重视程序语句执行顺序对逻辑结果的影响。简单的逻辑程序如下面程序的例子比较容易看出缘由，但在复杂的程序当中涉及其他的干扰因素出现，对于语句执行先后顺序引起的错误容易被编程人员所忽视。

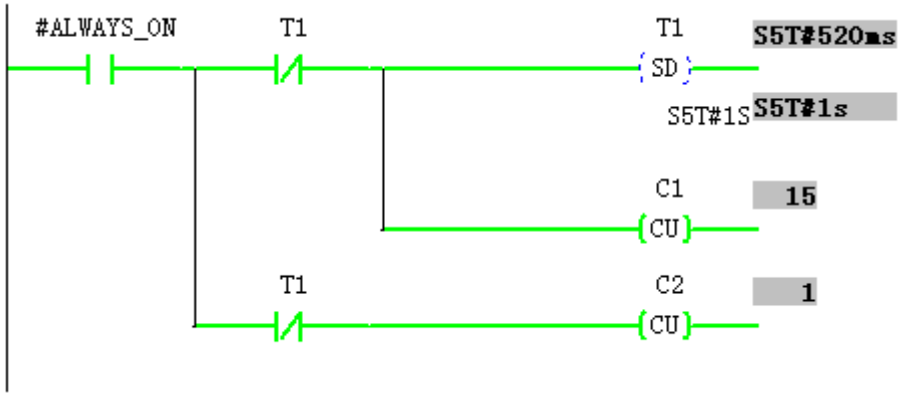
```
L     0
T     MW0
……  (其它省略的程序段，此时 MW0 的数值为 0)
L     10
T     MW0
      ……  (其它省略的程序段，此时 MW0 的数值为 10)
L     20
T     MW0
      ……  (其它省略的程序段，此时 MW0 的数值为 20)
```

PLC编程常见问题(S7-300)

简单错误：

5.语句执行先后顺序2

上个例子的还是比较容易看出缘由的，但对于下面例子大多数人会认为是正确的，但实际检测结果却是不正确的。



程序原目的：

T1定时器每秒导通一次，C1及C2会每隔一秒进行一次加1操作。

故障现象：

实际监控结果：C1工作正常，C2并未继续计数

为何会出现不一样的结果？有些有心的朋友也做了实验，也有Zane版对此故障的解说：

http://www.ad.siemens.com.cn/club/bbs/post.aspx?a_id=1621672&b_id=4&s_id=0&pno=#1621861

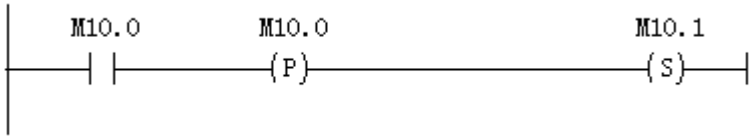
对于新写的程序是否能够完全按照编程人员的本意来执行单看程序有时是很难把握的，因此必须要做好厂内程序试验。

清楚了解语句执行的先后顺序并巧妙的加以利用往往可减少程序的编写量。

PLC编程常见问题(S7-300)

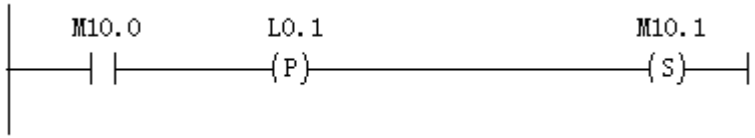
简单错误：

6.上升（下降）沿不工作



错误程序案例：

“P”指令要求使用与前面指令不同的地址



“P”指令不应使用临时变量作为存储地址
（临时变量会随着系统堆栈变化）

上升（下降）沿不工作是一种常见的错误，尽管程序中“P”或“N”指令允许的数据类型为：I、Q、M、L、DB。建议仅使用M及DB数据类型。

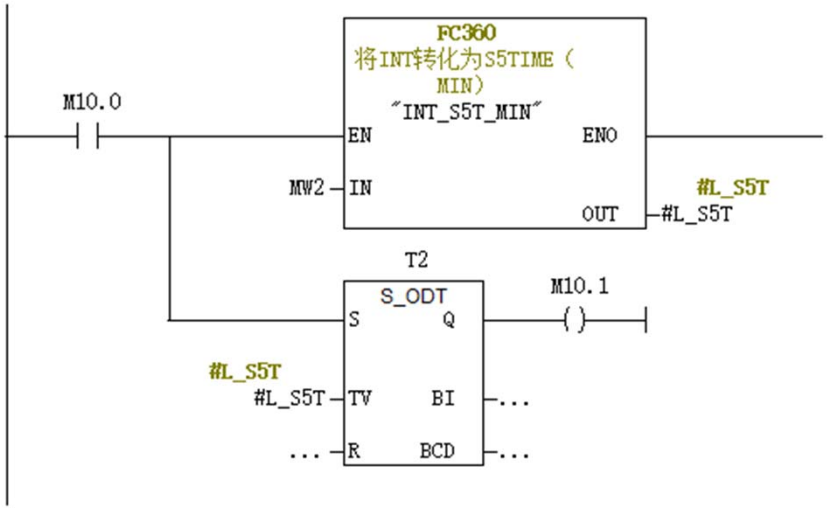
PLC编程常见问题(S7-300)

简单错误：

7. 定时器常见问题

错误程序案例：

假如MW2设置时间超过166min时，程序出错。



定时器在程序编程中存在的问题较多，主要总结有以下几点：

1）定时器时间设定超限：

一个自带定时器最长定时时间是2h46m30s，超过这个时间定时器出错。

2）分辨率对控制程序的影响：

部分程序对时间精度要求较高必须清楚定时器的精度对控制程序是否存在影响

分辨率	范围
0.01s	10 ms 到 9 s 990 ms
0.1s	100 ms 到 1 m 39 s 900 ms
1 s	1 s 到 16 m 39 s
10 s	10 s 到 2 h 46 m 30 s

3）时间控制掉电重上电的影响：

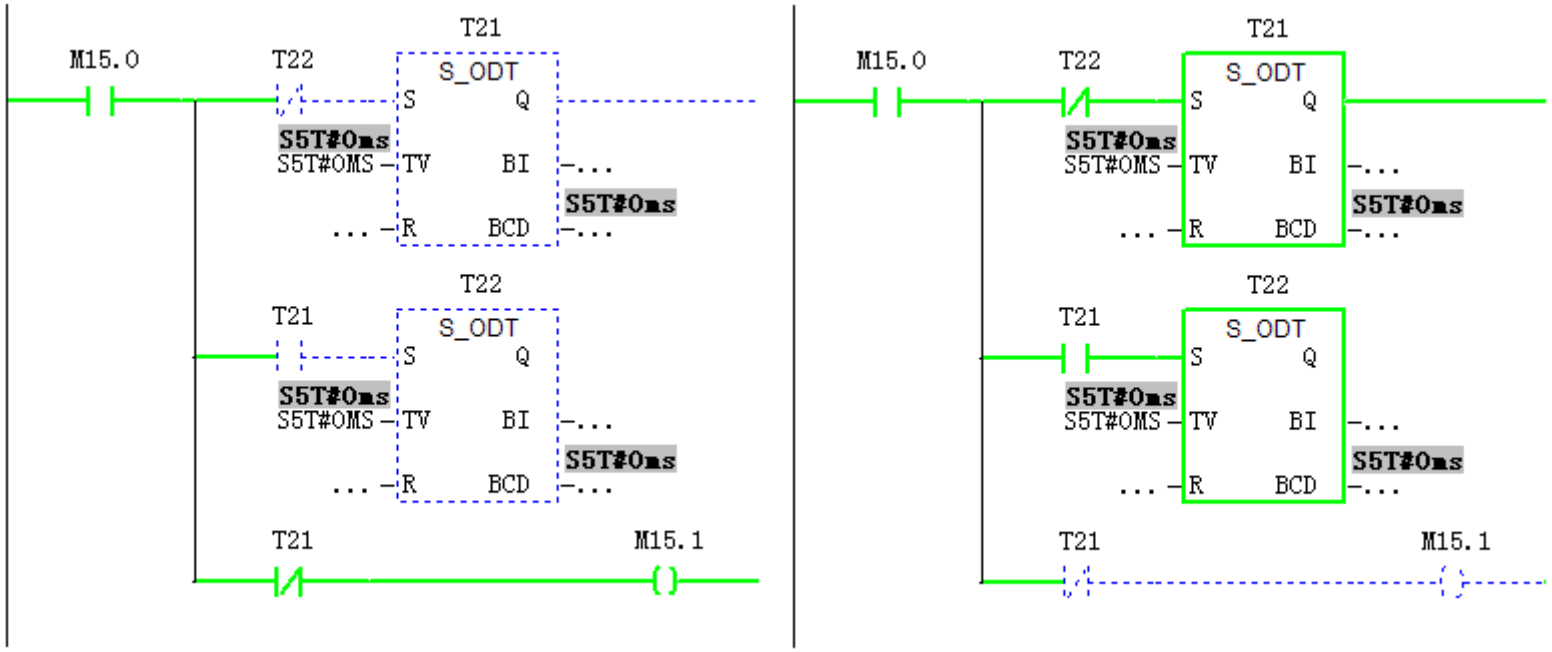
时间控制程序在工程中大量运用，在程序制作及试验时往往并未对时间控制掉电重上电后设备如何运行进行试验，而实际停电后恢复供电往往对工艺运行或设备运行产生负面影响。

PLC编程常见问题(S7-300)

简单错误：

7. 定时器常见问题

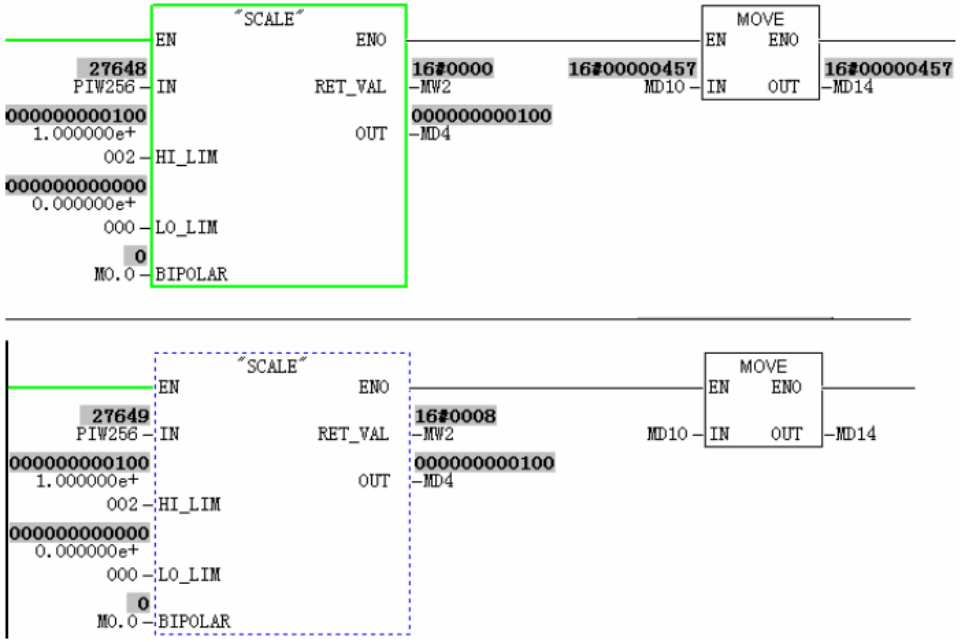
4）定时器时间设置为0：
在使用定时器编程时很多时候并未注意到定时器时间如果设置为0时会产生什么后果。如果程序编程中不考虑对定时器时间为0时设备该如何运行，那么可能会对实际设备运行造成设备严重伤害。
下面案例是某时间控制程序的错误案例简化程序：



PLC编程常见问题(S7-300)

FC/FB使用中常见问题：

1.ENO的问题



程序原目的：

进行模拟量转换，并无条件地将MD10的数值传送至MD14。

故障现象：

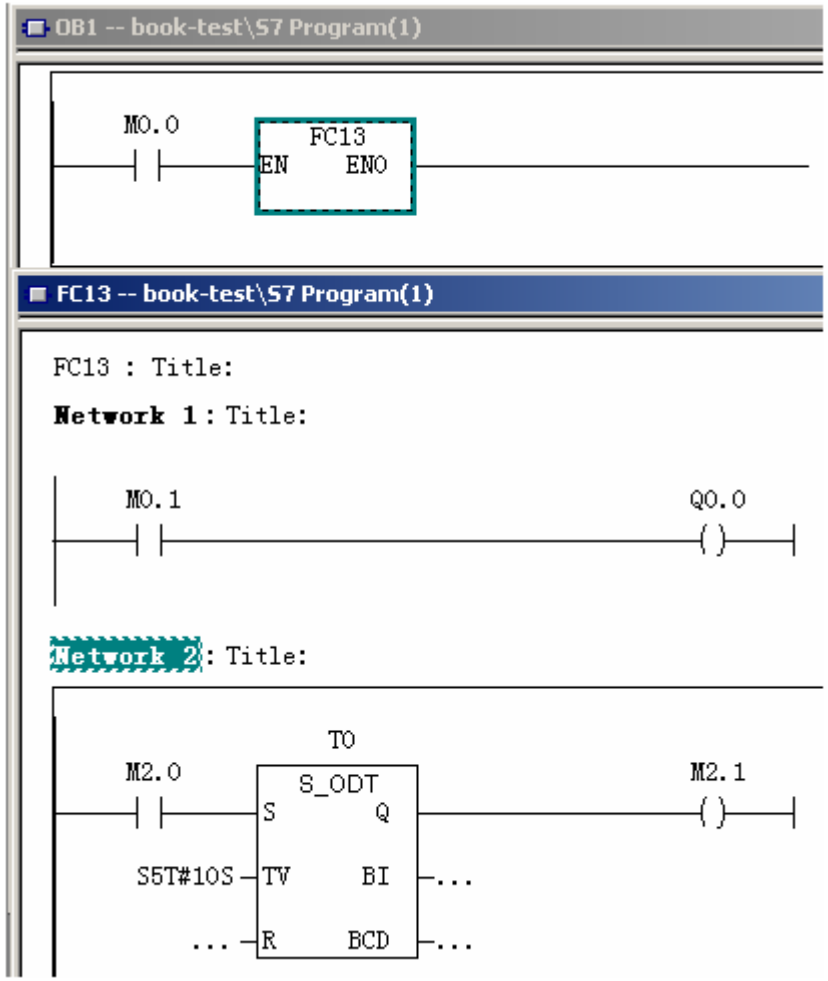
当IN值超出了上限，ENO不导通MOVE指令不执行。

对初学者来说，容易在ENO的使用上出错。很多时候将FC/FB串起来使用时并未确保ENO是否保持导通。要想ENO保持导通，在FC/FB程序结尾进行如下处理：

PLC编程常见问题(S7-300)

FC/FB使用中常见问题：

2.停止对FC/FB调用引起的问题



程序原目的：

在OB1中当M0.0为1时，调用FC13
在OB1中当M0.0为0时，不调用FC13
FC13包含简单逻辑及定时器的使用

故障现象：

当M0.0，M2.0都为1的时间超过10秒，M2.1为1 M0.0变为0后，再次变为1，由于定时器保持着计时到达的状态，M2.1会立刻变为1。在实际应用中，如此逻辑为某设备的启动信号，那么这设备可能会跳过延时执行逻辑被立刻执行，可能会产生不良后果。

PLC编程常见问题(S7-300)

FC/FB使用中常见问题：

3.FC/FB中临时变量的使用

TEMP区的临时变量在编程中起着重要的作用而被大量的使用。但在编程中如果对临时变量的特性不熟悉，很容易出现一些错误。
对于FC或FB中的临时变量，不要希望将本次调用的数值可以存储在里面以供下次程序调用，因为这些临时变量所使用的L堆栈空间在FC或FB调用结束后就释放给系统了，其他后续程序可以任意使用。所以下列用法都是错误的：
1）将临时变量用于上升/下降沿指令
2）将临时变量用于自保持逻辑
3）临时变量未在所在程序段中赋值，直接使用
注：在熟悉临时变量的特性前提下，上面第1、2点还是可以使用的。
在编程过程中，临时变量有着使用方便简单，占用存储空间小等很多优点，因此在FC/FB编程中能使用临时变量的地方推荐尽量采用临时变量。

4.FC输出处理

对于FC的使用，另一个常见的错误是对输出的错误处理。相比较于FB,FC是一个没有存储空间的逻辑块。如果没有数据被写至FC的OUT参数，FC将会输出一个随值。因此OUT参数必须要在每次执行FC时赋给一个确定值，下列用法都是错误的：
1）将输出变量用于上升/下降沿指令
2）将输出变量用于自保持逻辑
3）输出变量未在所在程序段中赋值

PLC编程常见问题(S7-300)

OB使用中常见问题：

1.OB未装载

STEP7中所有的用户程序都将在组织块中被调用。而针对不同事件，CPU将会调用不同的组织块，在某事件发生时，如果CPU中没有下载相对应的组织块，CPU将进入STOP状态
（例如DP从站通信故障时，CPU中如果没有OB86，CPU将进入STOP状态）
因此一般在每个工程中都需加载OB82、85、86、87、122这几个错误处理组织块

PLC编程之个人见解

在我们的编程过程中，很多时候按着需求去编，初步看起来按着要求设定参数，正常运行时程序运行是正常的，但是在编写完程序之后是否认真审视过自己的程序是否是没有问题了,是否合理了，是否做过实验程序去验证了你的程序完全没问题了？后面这些步骤很多时候都给我们偷懒去省略了。编程难并不是难在按正常的逻辑去编好你的程序，而是编好后或者在编程的过程中需要考虑现场可能出现的各种问题，在此情况下你的程序是如何与之应对的。充分考虑了各种情况下的各种应对措施，你才能够大胆的说：“我的程序是安全的，你随便用吧，没有BUG的”

这类的考虑通常有以下几个方面：

1.错误的参数设置的对应处理：正常参数设置是设备正常运行的前提，但是操作人员并不是编程人员，水平参差不齐，在设置错误的参数时候，你的程序是怎么运行的？是否会导致现场的设备错误的动作导致安全隐患？这个错误其实在我们的编程中是很常见的。

2.断电重上电的对应处理：程序在运行中突然遇到断电了，重上电之后设备是什么状态的？是否会产生一些错误的动作？这类问题一般出现在变量是否需要断电保持以及定时器是否需要断电保持。要验证此类问题一般需要通过做一个模拟实验程序，通过实际的通断电去验证你的程序是否安全。

3.通讯中断的应对处理：现在运用各类总线，各类远程I/O，各类通讯日益频繁，在我们的编程过程中就需要认真考虑到当此类通讯出现故障的时候，你的程序该如何去应对的：包括短时瞬断的应对，长时间通讯中断的应对等等。

4.设备故障的对应处理：在控制过程中当其中某个设备故障了，程序是否做了对应处理，此类设备故障处理相信大多数的程序员都会做到，但是否全面安全就要考量了。

总的来说一个好的程序最重要的第一条就是安全，在安全的基础上才能去考量你的编程逻辑，编程技巧是否好，是否对程序进行了标准化。按正常的逻辑去编程可能你很快去编好了，在编好后你如果按安全的角度综合考虑增加了各种各样的补丁程序，你就会发现你的程序变得很臃肿了，这个臃肿的程序你觉得会是不好的程序吗？肯定不是的，你能编好臃肿而安全的程序说明了你对方方面面的了解，也说明了你对控制工艺的深入研究。可以在下一次的编程中尝试将臃肿的程序合理化，逻辑化，可读化，那你的程序就又上升了一步。之后可以对相类似的控制工艺进行程序的标准化，模块化的编程，日后这类程序就可以信手拈来。

当你可以拍着胸脯对操作人员说你使劲去“操”吧，没事的，安全的！这个时候说明你的程序已经没啥BUG了。