SIEMENS

SIMATIC PLC 计时器的使用

SIMATIC PLC Timer usage

User Guide **Edition (2009 年一1 月)**

摘 要 本文档着重讲述西门子 PLC 中计时器的使用， 包含计时器与循环程序的关系，计时器的运行特性等，举例，并进行分析。

关键词 计时器 循环程序 异步

Key Words Timer Cyclic program Asynchronism

目　　录

一、分析 Timer

1，提出问题

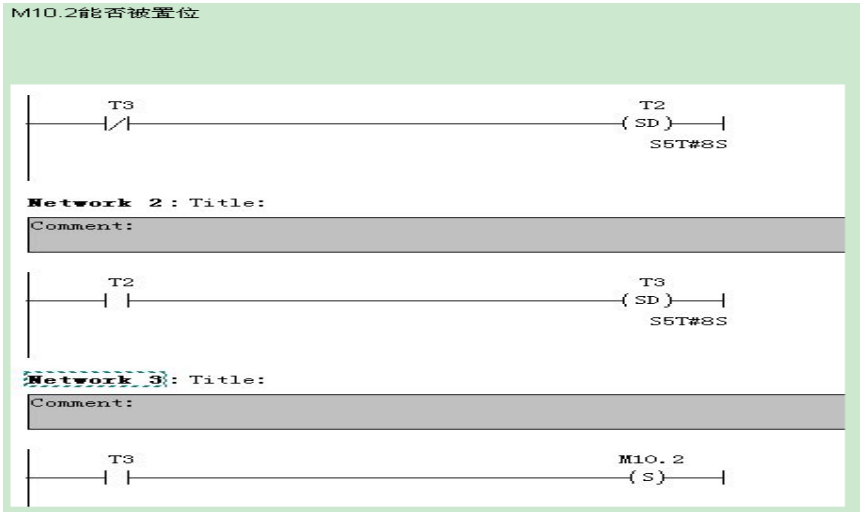

图 1

问题 1：M10.2 能否被置位？

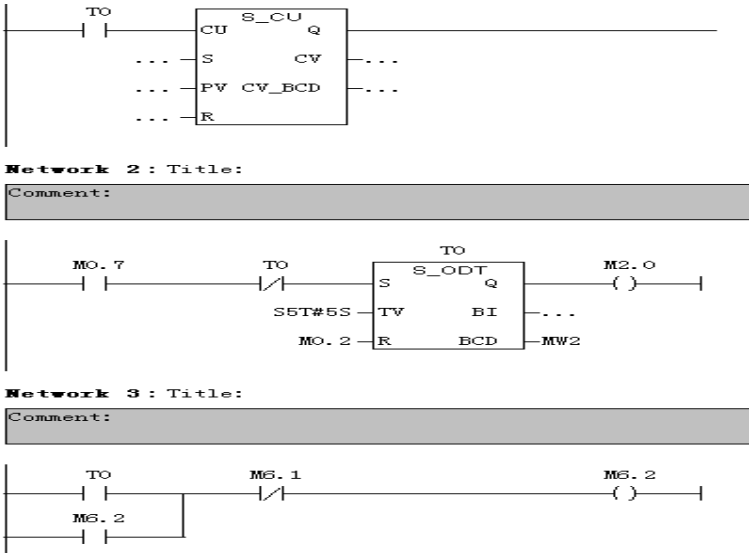

图 2

问题 2：S_CU 计数有无问题，M6.2 能否被置位?

先来了解一下都有哪几个计时器，以及它们的特性如何

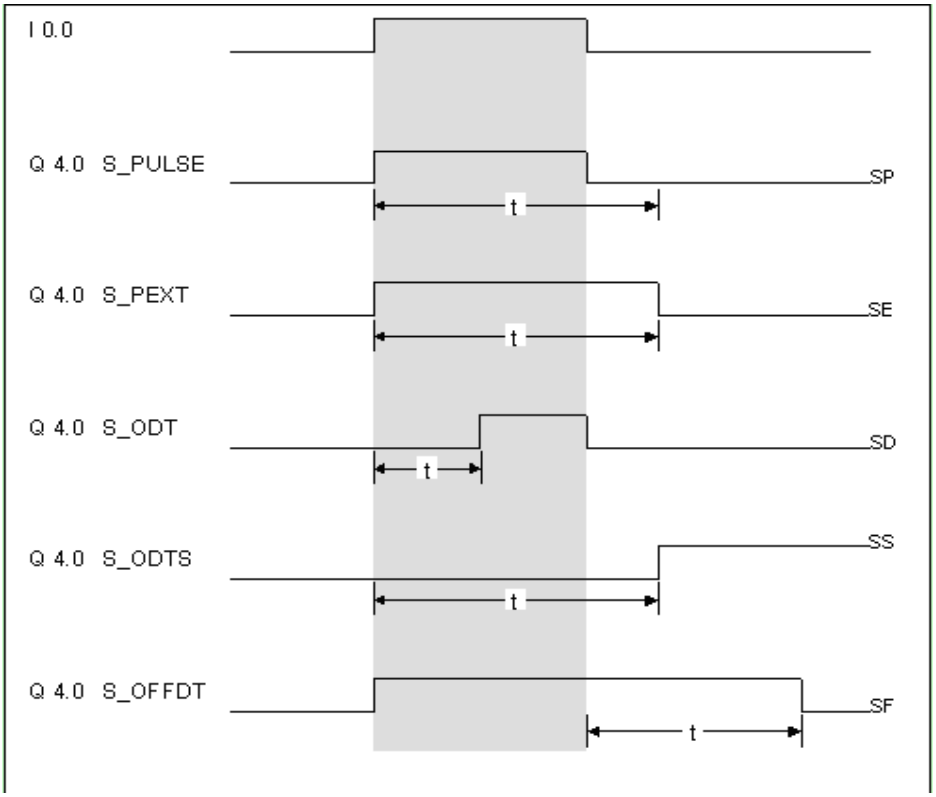

图 3

2，计时器描述

从以图 3 可以看出 5 个计时器的基本特性，可以简单的从中挑选与控制工艺相符合的计时器使用，如果想了解计时器的详细信息，可以选择计时器，并按 F1 看帮助信息中的具体逻辑图。

以计时器 SD 为例，参见图 4

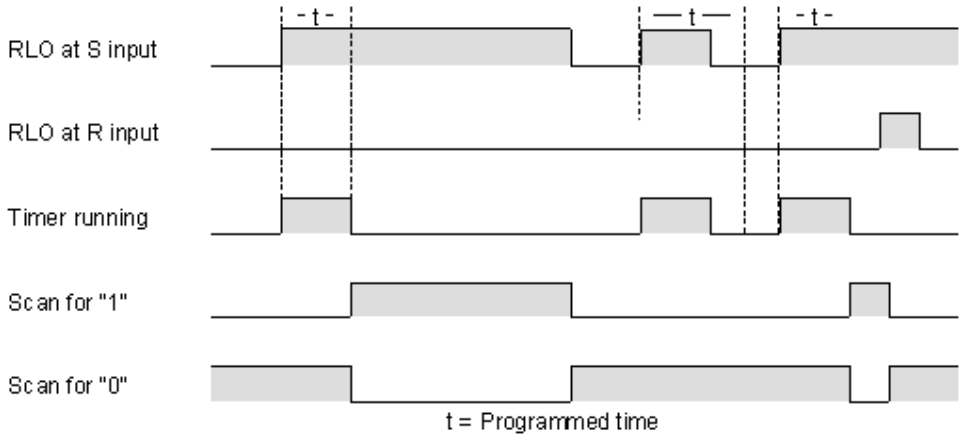

图 4

我们从中可以知道，当触发端 S 的信号为上升沿时，触发计时器开始运行，时间结束后计时器输出端为 1，S 信号为下降沿时，计时器输出端为 0
那么根据此情况，以图 1 为例，咱们可以把刚才的梯形图程序通过时序图表示如下图 5
其中 a,b 之间是在扫描此段程序两个周期之间的间隙。

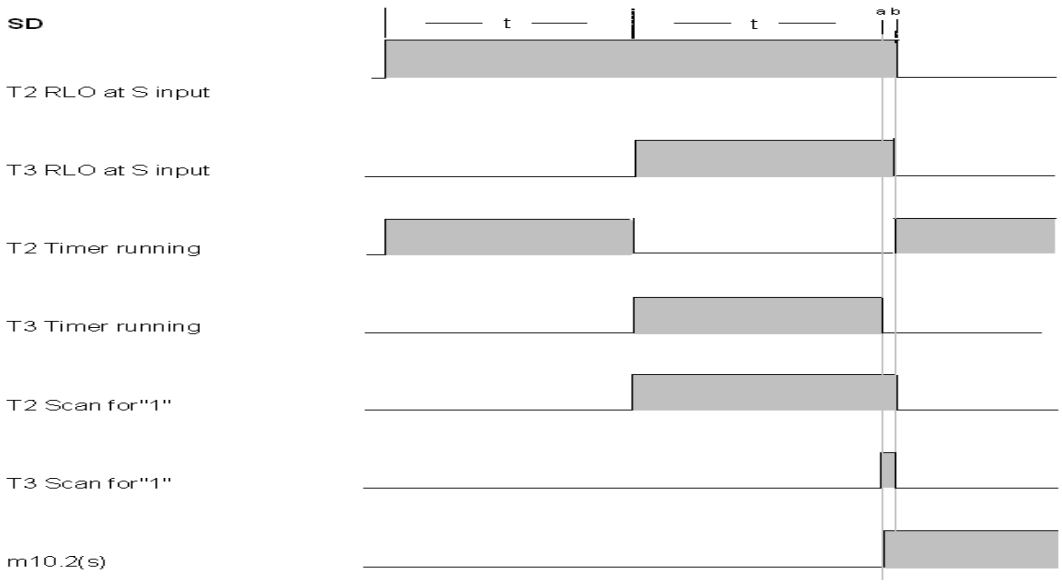

图 5

3，计时器与循环程序的关系

经过分析，可以看出，M10.2(S)是可以被置位的，那为什么没有看见其被置位呢？

大家注意，这里 t 的时间是 8s，我们知道，一个程序的扫描周期很短，可能才十几----几十毫秒，在线时候可以监控到 Scan Cycle Time。如图 6

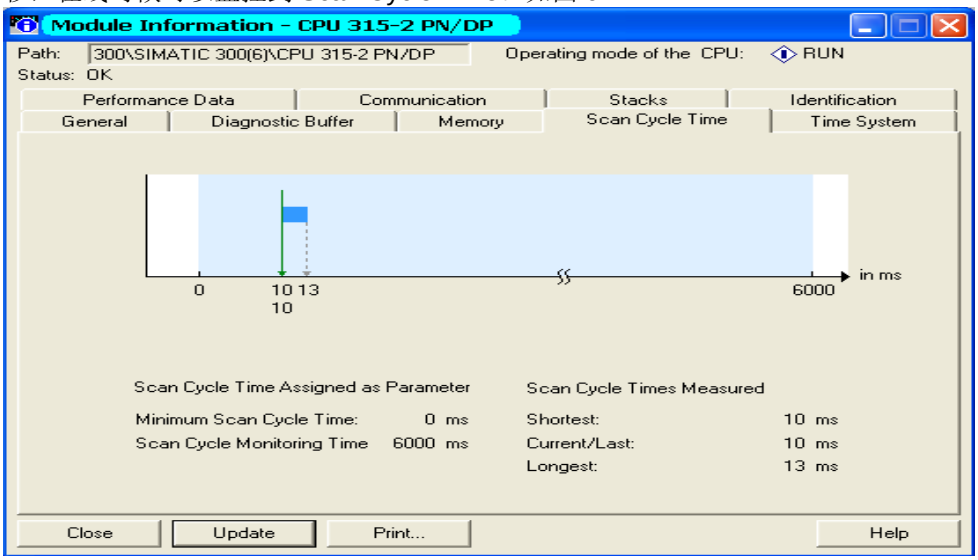

图 6

那这个时间不是远远超过了扫描周期么？

我们又知道，如果程序扫描周期大于最大扫描周期监控时间 Scan Cycle Monitoring Time，那么将会触发中断，甚至造成 CPU 进入 STOP 状态。
其实，计时器的执行是异步于 OB1 循环扫描的，只要计时器运行后，在每一周期扫描到计时器的触发端 S 信号如果为 1，那么计时器就将在此周期继续计时。因此，它对于最大周期监控时间并没有太大的影响，只是调用语句时占用了少许 us 的时间。

怎么来验证这个说法呢？就是说计时器的执行并不同步于 OB1 程序扫描周期。

1，可以在程序中加入若干 SFC47 增大程序扫描周期（保证小于 Scan Cycle Monitoring Time），通过监控计时器的时间，可以看出，计时器的时间是跳跃式的变化的，也就是说，也就是说，当程序扫描完计时器，继续往下进行时，计时器满足触发条件进行计时，此周期往后的计时是一直在进行的。
2，可以通过在中断来证明
3，通过程序死循环监视计时证明
4，通过多个计时器监视时间来证明等等各种方法

那说明了是异步的有何作用呢？

说明了刚才咱们分析程序所作的时序图有一定的问题，因为咱们的分析是按照程序一步步往下进行的，相当于是同步进行的。而实际在程序执行时，扫描周期是比较短的，所以计时器是在其中的某一个周期里计时器计时结束时输出被置位为 1，那么因为这样，所以对我们编写程序就会有一定的要求。也就有了下面一个问题

4，计时器动作的时刻

计时器的输出端是什么时候被置位呢，什么时候起作用呢，比如 T40 ┤├ **？**
是等到重新扫描到计时器块，计时器执行完毕才置位，还是不用重新扫描到计时器？程序中直接扫描的 T40 节点，它就已经被置位了呢？

1，我们可以设置 OB35 的看门狗时间为 2000ms, 如图 7

OB35 里触发计时器 T40，T40 ┤├的开点给线圈 M6.0，如图 8

OB1 里T40 ┤├的开点给线圈 M6.1，M6.0 开点给线圈 M6.2，如图 9
经过试验，观察看到，当 T40 的 Timer 运行结束后 M6.1 立刻就被置位了，而 M6.0 和 M6.2 会等到再次扫描到 OB35，才会被置位。

可得出结论，当计时器 T40 计时结束时，CPU 扫描到T40 ┤├时，它就已经为 1 了，不需要等到扫描计时器 S_ODT(SD)。

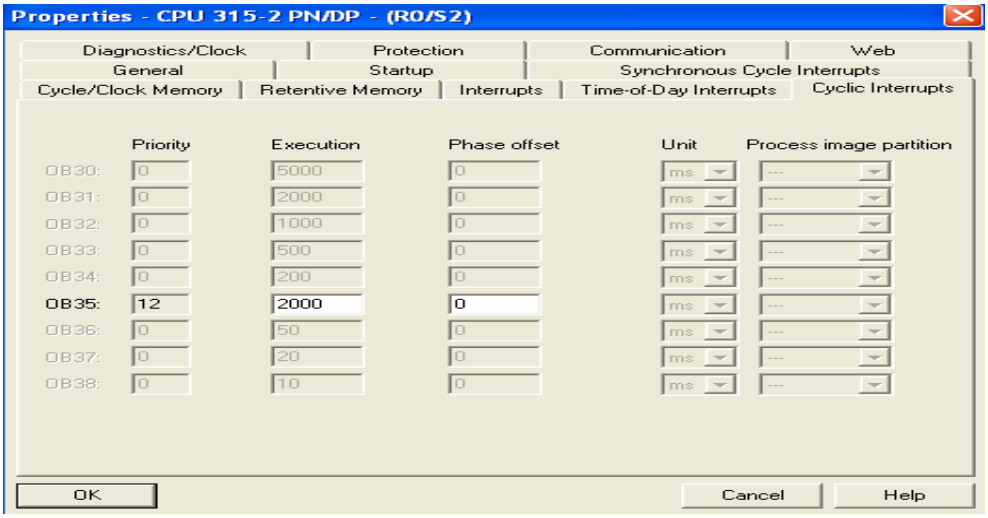
Properties - CPU 315-2 PN/DP - (R0/S2)
Diagnostics/Clock
Protection
Communication
Web
General
Startup
Synchronous Cycle Interrupts
Cycle/Clock Memory
Retentive Memory
Interrupts
Time-of-Day Interrupts
Cyclic Interrupts
Priority
Execution
Phase offset
Unit
Process image partition
OB30: 0 5000 0 ms ---
OB31: 0 2000 0 ms ---
OB32: 0 1000 0 ms ---
OB33: 0 500 0 ms ---
OB34: 0 200 0 ms ---
OB35: 12 2000 0 ms ---
OB36: 0 50 0 ms ---
OB37: 0 20 0 ms ---
OB38: 0 10 0 ms ---
OK
Cancel
Help

图 7

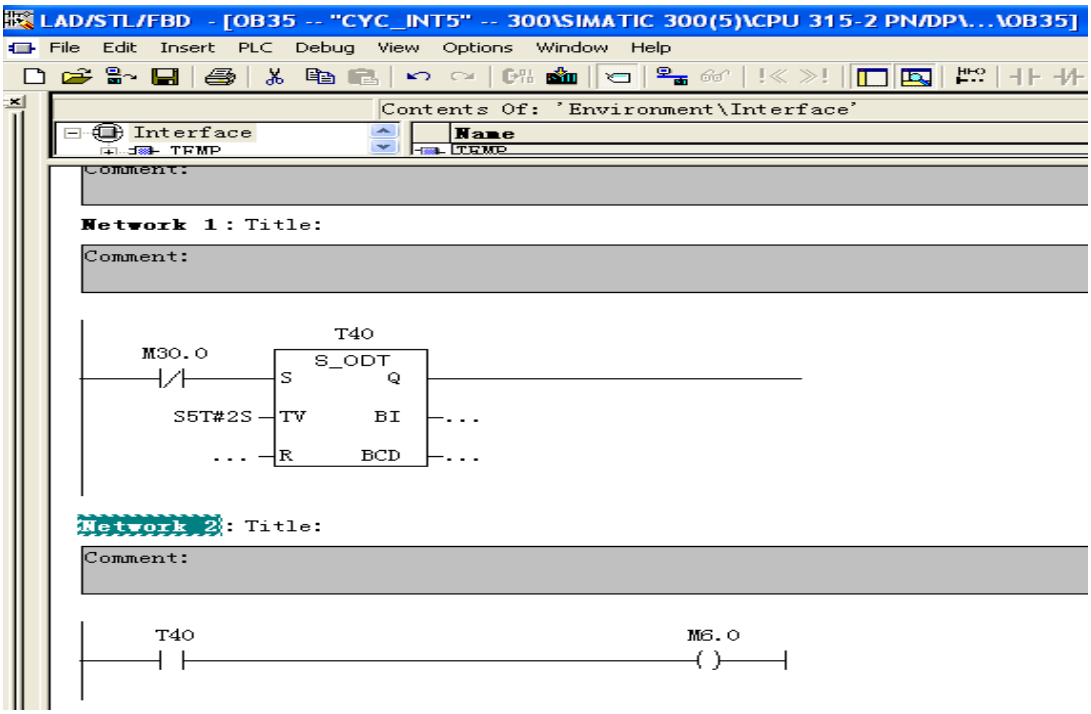
LAD/STL/FBD - [OB35 -- "CYC_INT5" -- 300\SIMATIC 300(5)\CPU 315-2 PN/DP\...\OB35]
File Edit Insert PLC Debug View Options Window Help
Contents Of: 'Environment\Interface'
Interface
TEMP
Name
TEMP
Comment:
Network 1: Title:
Comment:
T40
M30.0
S_ODT
S
Q
S5T#2S
TV
BI
...
R
BCD
Network 2: Title:
Comment:
T40
M6.0

图 8

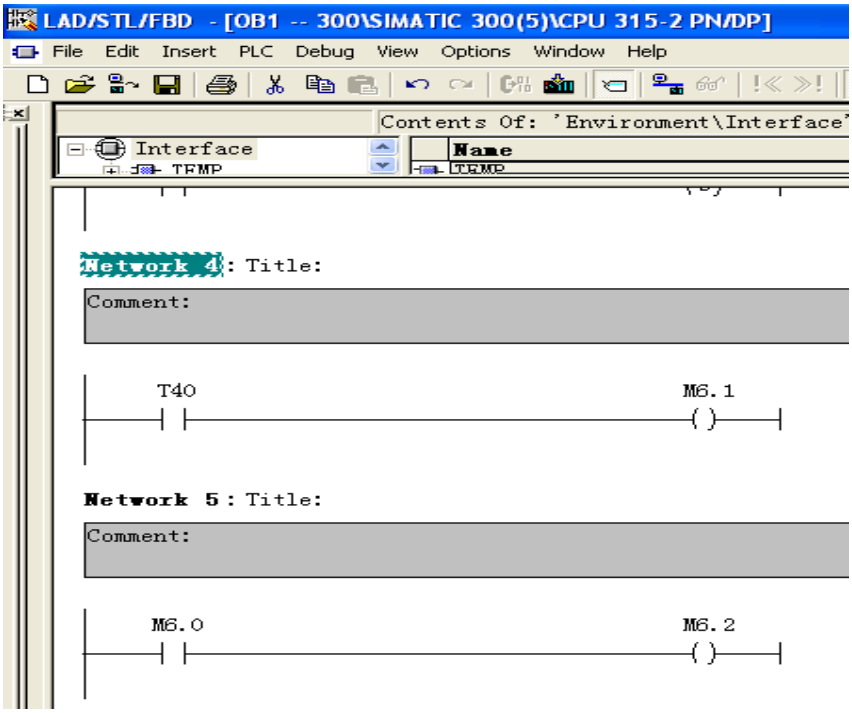

图 9

2，也可以在 OB1 里调用多个“wait”代码让 OB1 的扫描周期足够大，如 5s，先调用一个 SD T2 1s，然后调用若干“wait”，大概持续 2s，用 T2 开点触发一个线圈如 M10.0，再调用若干“wait”，大概 1s，然后再调用一个 SD T3，可以看出再 T3 还没有开始计数时，M10.0 已经被置位了。

计时器在 OB30—OB38 里呢？

是一样的。

可以在 OB35 里使用 SD 计时器，可以发现，当程序调用 OB35 时，计时器开始运行，把 OB35 执行时间和计时器时间设置大些，可以发现，只要每次在扫描的计时器触发端时，条件满足，计时器就开始运行，直到下一次扫描 OB35 时再扫描到此条件为止。

可以把计时器时间设置足够大，当计时未结束前把它的触发端变为 0，那么其计时停止，直到再次触发。

可以得出计时器的运行只与每次扫描到它的触发端有关。扫描完触发端后，计时器的运行就与触发端无关了，直到下一次再次扫描到此触发端。

5，分析程序

了解了以上的一些基本知识，咱们再来看看刚才图 1 中的程序。

一个 CPU 的扫描周期是可以计算的，根据不同的配置和数据的读取，可以计算出不同的周期，在 PLC 运行时，每个周期的大小也是不一样的，可以大致计算出范围，可以根据每条语句来计算程序的执行时间，再加上相应的循环周期检测点，周期中断，访问过程映像区，通信负载等。这些时间的长短与 CPU 型号及使用方式有关。

使用 PS307 5A，CPU315-2PN/DP (315-2EH13-0AB0 V 2.6.50)为例。以下所有时间都以此配置为标准。

我们把图 1 的梯型图换成语句表来分析指令执行的过程。

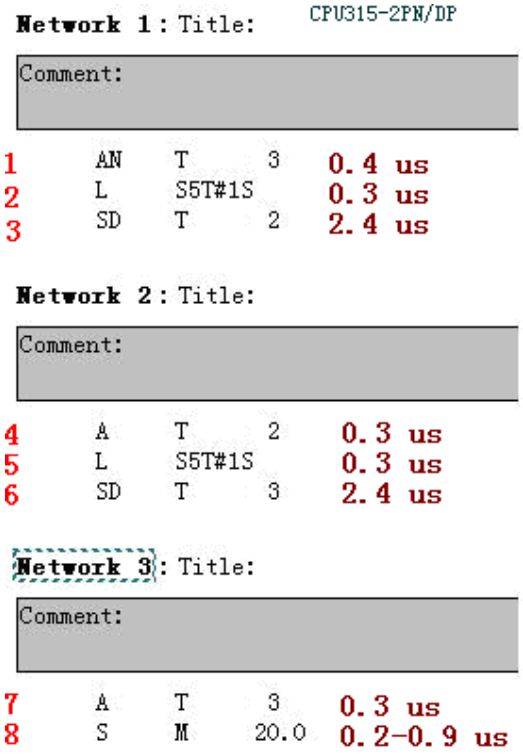

图 10

一个 CPU 的扫描周期的计算可以根据以下几个过程来进行

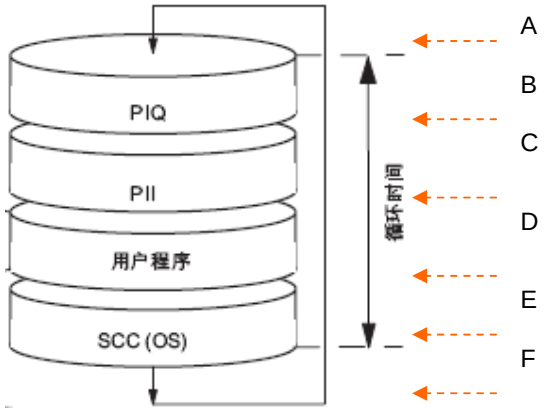

图 11

A.操作系统初始化循环时间监视
B.扫描 PIO
C.扫描 PII

D.执行用户程序，并执行程序中定义的操作
E.扫描周期检测点操作系统时间（周期结束时执行挂起的任务，如装载和删除块）
F .CPU 返回到周期开始的时间点，并重新开始循环周期监视

在以上的步骤中都是有时间的，虽然很小，但是也占用时间。可以根据不同的硬件组态，参照
CPU Specifications手册 进行计算，
为了便于计算和理解，咱们以理想状态来计算。假设 CPU 周期中的 A，B，C，E，F 的时间为固定的数值 X us。
只分析程序里的”D” --用户程序中的命令执行。

程序是顺序扫描的，从 Network 1—3 依次进行，

以第一个周期开始时来分析，首先扫描 Network 1 中 T3 计时器为 0，因此闭点使能，T2 开始计时（0－8S），但此时扫描 T2 输出为 0，
因此扫描到 Network 2 中 T2 开点不使能，扫描到 T3 不执行，
Network 3 中 T3 开点不使能，M10.2 为 0。
到此过程[0.4+0.3+2.4+0.3+0.3+2.4+0.3+0.2(或 0.9)] us = 6.6 (或 7.3) us。
注意：T2 一直在累加时间，相当于此时 T2 计时也到达 6.6（或 7.7）us。

然后加上刚才的时间 X us，那么一个周期可以认为是 t=X+6.6 (7.7) us。X 大于 7 us，可以看出语句的执行是在很短的时刻进行，所以大家在编程时常用的每个计时器都会经过若干个程序扫描周期。

因为 Timer 是异步的，所以 T2 的时间应该在一个周期里也为 t=X+6.6 (7.7) us，那么根据上面的程序看， 因为 T2 设置为 8s，所以应该在大概 m=8s/[X+6.6 (7.7)]us 个周期时，T2 执行完毕。

T2 假设最佳情况下，T2 执行完毕的时刻是在第 m 个周期内，
A．如果发生在 Network2 的 T2 开点之前，那么扫描到此 T2 开点的语句时，T2 的输出变为 1，执行下一条语句 T2 开点就会闭合，T3 开始计时。
B．如果 T2 执行完毕的时刻是程序扫描到 T2 开点语句之后才发生的，那么因为后面的程序没有对 T2 的操作，只有在下一个 m+1 周期，才能检测到 T2 的变化。T3 开始计时。

T3 开始计时的前提条件是 T2 开点闭合，假设在第 m 个周期里，T3 开始计时，那么同样，要经过大概 m 个周期左右，T3 才能执行完毕，到此时，已经经过了 2m 个周期，因为 M10.2 线圈是由 T3 开点的闭合信号来置位的，那么现在就来分析一下什么时候可以发生此动作。

注意：在此例子程序中，在 Network1-3 中都有对 T3 的操作

T3 假设在最佳情况下，T3 执行完毕的时刻是在第 2m 个周期。在第 2m 周期内
A．如果发生在 Network1 的 T3 闭点之前，那么在程序扫描到 T3 闭点的时候，T3 的输出值已经变为 1 了，闭点变为开点，T2 输出变为 0，往下扫描到 Network2 的 T2 开点变为 0，T3 的 SD 输出也变为 0，继续扫描到 Network3，T3 开点为 0，那么 M10.2 未被置位。

B．如果发生在 Network1 的 T3 闭点之后，Network3 的 T3 开点之前，（ 则 T2 是保持为 1 的），在扫描到 T3 开点时，T3 的输出值变为 1，T3 开点变为闭点，M10.2 被置位。

C．如果发生在 Network3 的 T3 开点之后，那么在此周期内对 m10.2 不会产生置位，在下一周期（2m+1）,T3 输出值变为 1 了，所以在 Network1 里 T3 闭点变为开点，T2 输出变为 0，扫描到 Network2 里，T2 开点变为 0，导致 T3 输出值变为 0，扫描到 Network3 里，T3 开点变为 0，因此不会对 M10.2 置位。在再下一周期（2m+2），扫描到 Network1 里 T3 闭点为 0，使能 T2 重新开始计时。

从以上分析可以看出，M10.2 是可以被置位的，但是在条件符合情况下，看 T3 中情况 B 的时间大致为图 10 中的 2，3，4，5，6 操作 Y=（0.3+2.4+0.3+0.3+2.4）=5.7us，也就是图 5 中 a 时刻得在这个时间段内，这个时间极为短暂。因此我们在检测的时候很难捕捉到此信号。

我们可以在假设最佳情况下来计算一下概率，就以现在这个例子

t=X+6.6(7.7)us　Y=5.7us　m=8s/t

$$\frac{Y}{t} * \frac{1}{2m} = \frac{5.7us}{X+6.6(7.7)us} * \frac{1}{2*[8s/[X+6.6(7.7)us]}} = \frac{5.7us}{16s} = 0.00000035625$$

可以看出概率非常小，只有增大 5.7us 才能增大概率，也就是增大 Y（或 B）的时间

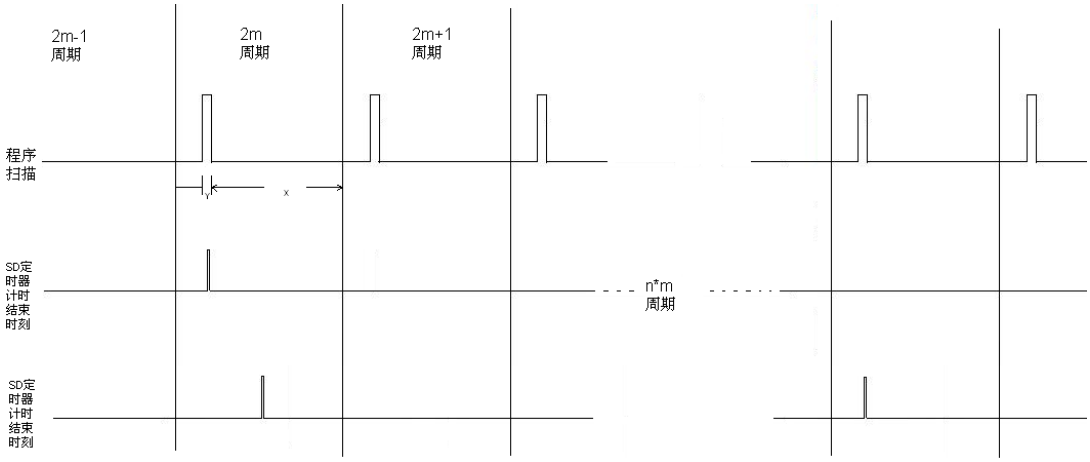

图 12

只有 SD 计时器结束时刻发生在 Y 时间段内，那么 M10.2 才能被置位，也就是
Y：Network1 的 T3 闭点之后，Network3 的 T3 开点之前这个时间段内
如何通过试验来来验证以上的理论说法呢。
可以实际通过实验来检测

在不做任何修改的情况下，上面的程序要对 M10.2 进行置位的条件是很难捕捉到的，在 n*2m 个周期也难以捕捉到，因此经过长时间运行程序，M10.2 也难以发现被置位。

为了比较直观，加上了 Network4，用计数器来大致评估时间。如图 13

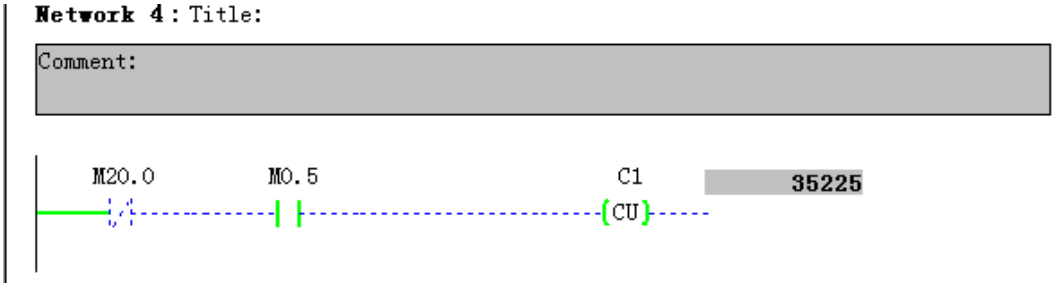

图 13

(1) 如果在 Network1 的 T3 闭点后加上 SFC47 设置 1ms 延长此段时间，这样可以大大增加 **T3** 中情况 B 的时间（也就是增大 Y 的时间），那这样也就增加了它的概率。可以看出，在 C1 计算到 14 时，M20.0（相当与前问所述 M10.2）已经被置位。如图 14

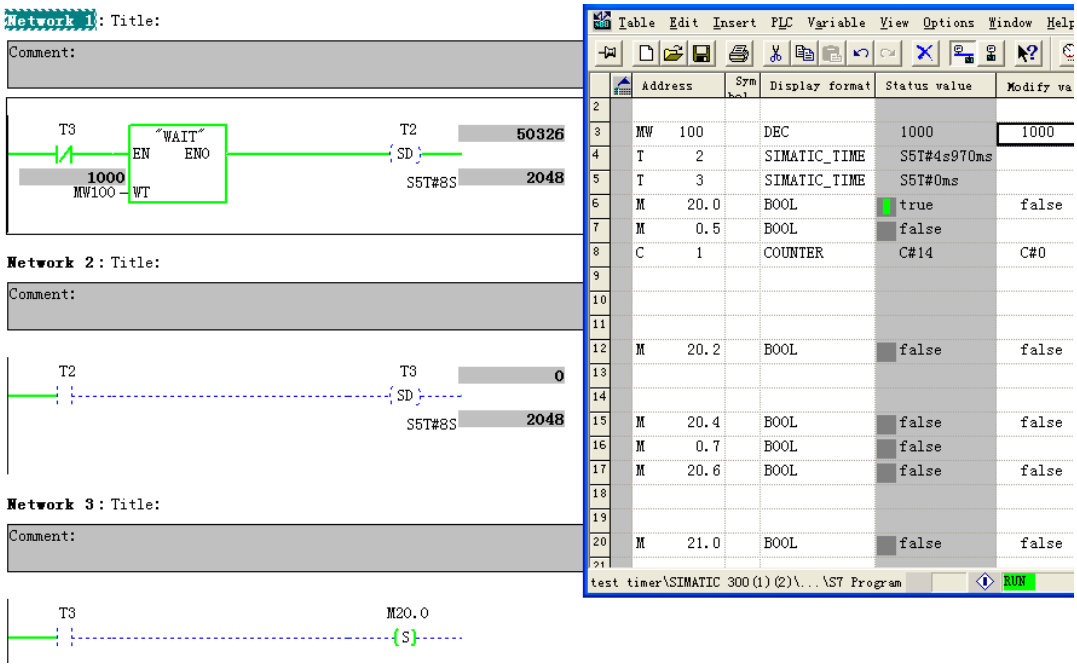

图 14

此种情况概率大约也可以计算为

t=X+6.6(7.7)us+1ms　　Y=5.7us+1ms　　m=8s/t

$$\frac{Y}{t} * \frac{1}{2m} = \frac{5.7us+1ms}{X+6.6(7.7)us+1ms} * \frac{1}{2*\{8s/[X+6.6(7.7)us+1ms]\}} = \frac{5.7us+1ms}{16s} = 0.00006285625$$

可以看出概率大了很多

(2)如果把此 SFC47 放在 T3 闭点之前，那么如 **T3** 中情况 A 的分析，对其程序扫描对产生 B 的效果不会有任何增加。可以看出，在 C1 计算到 999 时，M20.0（相当与前问所述

M10.2）还没有被置位。（补充说明：最终也能被置位，在 Network3 后面放若干个延时块 SFC47 也不会增加概率即缩短被置位的时间）如图 15 所截图，可以看出

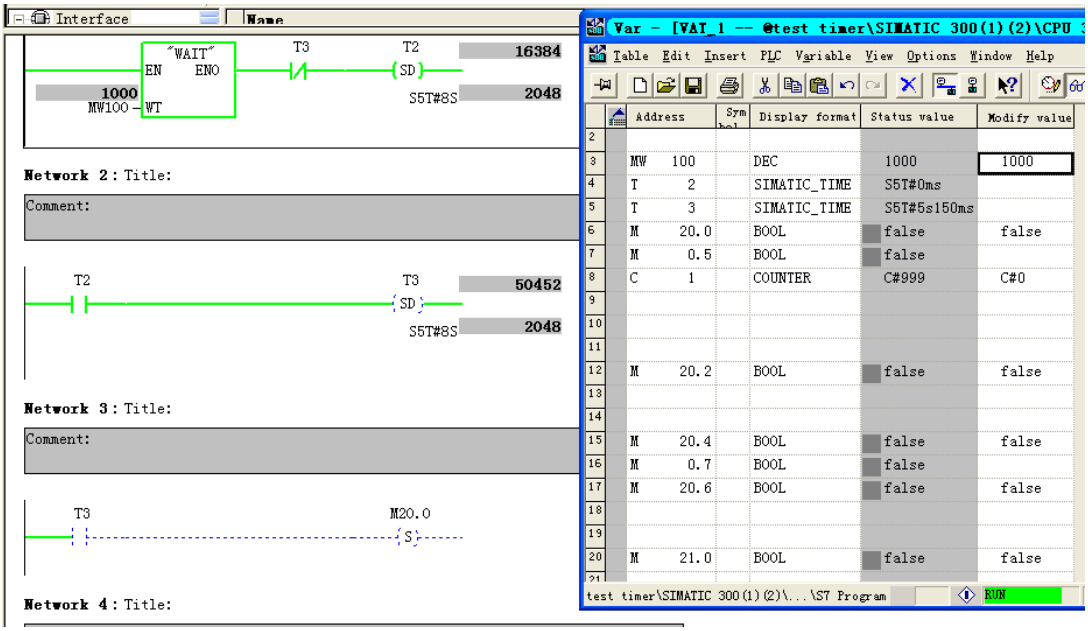

图 15

此种情况概率没有改善，因为关键参数 Y 没有变化

t=X+6.6(7.7)us Y=5.7us m=8s/t

所以概率还是为 5.7us/16s=0.00000035625

注：以上情况经过多次检测。

由此可得出结论：

1、只有增大 B（Network1 的 T3 闭点之后，Network3 的 T3 开点之前这个时间段内）的时间，才能增大置位的概率。

2、或者减少计时器时间，也可以在时间方面增大概率，但对编程逻辑无益处

注意：我们刚才的概率分析并不是绝对的，只是假定的理论上情况，而且并没有考虑中断，网络结构，计时器时基（可参考 Online Help）等等各种情况。

图 1 中的程序经过分析。

答案是：M10.2 能被置位，只是概率问题

所以对于图 2 中的程序咱们也可以同样分析它的情况。

答案是：S_CU 计数有可能会丢数（即，不是每一次都能被记录），M6.2 能被置位，只是概率问题

二、使用计时器注意

从刚才的分析来看

注意事项：

1、计时器的执行条件与触发它的条件是有关系的，每种计时器都不太一样，但原理相通。当扫描到触发端时，由触发端决定计时器的计时是否开始，停止或继续。

2、要想很好的使用计时器，使用时都得考虑计时器的特性和程序指令执行的先后顺序，也就是要注意到，计时器计时结束的时刻到下一次程序中调用到计时器的节点，一定要有足够充裕的时间，让程序来捕捉。以免造成不必要的情况。

3、编程时不仅要考虑到逻辑，还要考虑到计时器的运行方式和动作触发时机，这样才能更好的让计时器为程序服务。

一些小技巧：

1、可以在计时器后面调用一个延时块，如 SFC47 来延长计时器输出端的有效时间。用以增大概率。

2、在满足逻辑要求的前提下，可以把对计时器的节点的操作放到计时器（如 SD）前，这样也可以一定程度增大计时器输出端有效时间。

3、如果只是做一些逻辑处理而不是对时间有特别要求的程序，计时器时间尽量设置小一些，这样也可以在绝对时间方面增大概率。但对逻辑无益

4、编程时涉及到不能确认的部分地方，可以先用 PLCSIM 模拟器运行看看会不会有未知的情况发生。

附录一推荐网址

AS
西门子（中国）有限公司
工业自动化与驱动技术集团 客户服务与支持中心
网站首页：http://www.ad.siemens.com.cn/Service/
AS 下载中心：
http://www.ad.siemens.com.cn/download/DocList.aspx?TypeId=0&CatFirst=1&CatSecond=-1&CatThird=-1
专家推荐精品文档：http://www.ad.siemens.com.cn/Service/recommend.asp
“找答案”AS 版区：
http://www.ad.siemens.com.cn/service/answer/category.asp?cid=1027