0803 【万泉河】巧用程序块临时变量

上一篇文章《0728 【万泉河】谈虎色变双线圈再谈再变》有网友评论说， 文章题目应该叫做巧用临时变量，我说， 我举例的程序中用的都是全局变量，所以不能这么叫。 然而我可以以这个名字写一篇文章， 真的是巧用临时变量的。

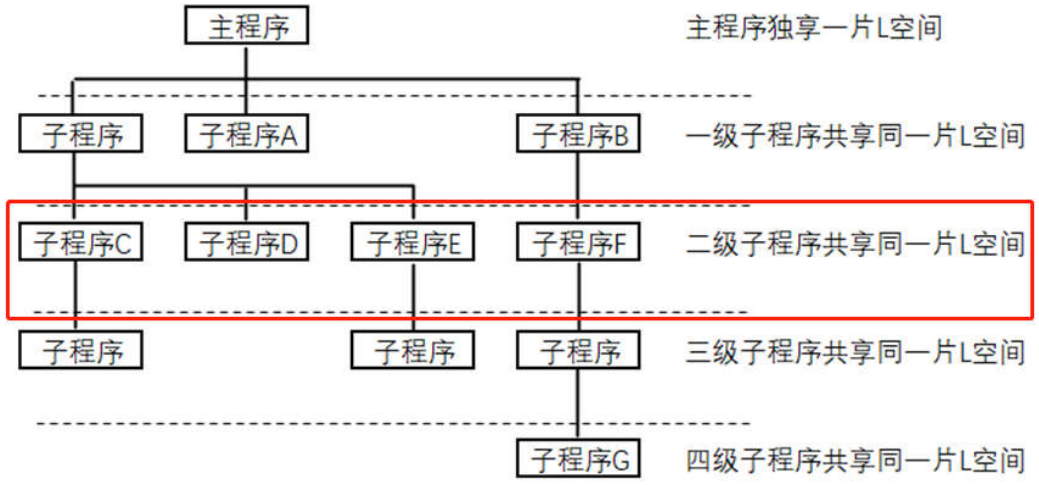
先看芳季大侠写过的系列文章：

> 【拥有属于自己的指令】提供者：芳季
> - 拥有属于自己的指令第一集
> - 拥有属于自己的指令第二集
> - 拥有属于自己的指令第三集
> - 拥有属于自己的指令第四集
>
> 本系列从用户角度一步步剖析库是如何做出的，子程序是怎么个调用原理，改如何写好写正确子程序。

入口在：
拥有属于自己的指令第一集-技术视频免费看-西门子工业 1847 会员–西门子中国(siemens.com.cn)
具体内容，里面涉及的知识点如果有不懂的，可以去仔细研读，甚至可以配合文章， 也自己实验摸索得到。

然而，我是不会从这样最基本的步骤一步步开讲，我只能把实际用过的理论方法一步到位给一个终极的解决方案。 如果芳老师不再出第五集，把我本文的内容当做其第五集，我给来个狗尾续貂也可以。

芳老师在第四集有一张附图：



本文的内容就从这张图展开。

这张图的意思是，在系统堆栈中，每一个同一级别的子程序 TEMP 变量同享的是同一片 L 空间。

这里先多讲一点 TEMP 变量的本质是什么？其实仍然是计算机的内存 RAM。原本作为 RAM，是可以被全局访问的， 只不过软件系统做了保护限制， 不允许外部访问。

多讲这一点的意义在于， 对于一些连子程序功能都没有， 或者有子程序， 然而子程序功能不如 SMART 这样强大的 PLC， 主要是西门子之外的一些其他品牌的小 PLC， 它们的系统中没有 TEMP 变量的时候，我们通过编程手段自己部署一部分内存数据区给当做 TEMP 变量使用。 这时候，操作系统本身自然不会限制， 那么我们在编程的时候就要格外小心， 要对这篇数据保护好。这是题外话。

回到上面的图片，图中我用红色方框圈出来，CDEF 四个子程序，它们是同一级别的。 那么除了说它们共享 L 变量空间之外，其实还有一个含义， 就是当程序执行完了子程序 C，又进入子程序 D 的时候， 在子程序 D 开始时， TEMP 区的内容，除了 INPUT 对应的部分被输入管脚的值刷新了之外， 其余的所有数值其实就是离开子程序 C 时的内容。

即所有数据值，现场被完整保留。

这就是我们平常教育新手的时候， 总要强调， TEMP 的数据，不要先读， 不要先读，不要读操作在先。 应该先写操作，赋值以后，后面的数值才是真实可控的，逻辑才可控。

有的人会不信， 说自己明明做子程序验证过了，我就是先读数值了， 也没有问题。 那是因为你测试程序中同一级别的子程序数量太少， 可能只有它自己。比如图中的 F。 那么数据区就不会被别的子程序污染。 你做的逻辑暂时好用。然而等到了现场， 真实应用中， 子程序多了， 数据就混乱了，逻辑就乱套了。

所以， 我经常建议对 TEMP 变量属性不熟悉的新手，调试中遇到疑惑的时候， 不防在子程序的开始时候， 把用到的一片 TEMP 变量全部清零。 后面程序功能都好用，才叫好用。

这是传统做法。

今天则是颠覆传统，利用前面说的进入下一个子程序后，前一个子程序的 TEMP 数据现场被完整保留集成的这个特性，我们可以用来做点文章。才是题目宣称的巧用临时变量。

我在 3 年前写过一篇文章，做过程序编程方法征集：

《【万泉河】有偿征集 2：S7-200 函数块 BLOCK_MOVE》

https://www.ad.siemens.com.cn/club/bbs/PostStory.aspx?a_id=1574124&b_id=80&s_id=157&num=3#anch

需要把一整块的数据用 BLOKMOVE 的方法传送到一个完全封装的子程序的内部。 然而并没有如愿得到答案。 还是我自己在去年底的时候，对 TEMP 变量思考到上述的内容后，得到了解决。

即，数据传送部分下降层级，放到相邻调用的前一个子程序中，直接对 TEMP 数据 BLOKMOVE，后面的子程序中的 TEMP 就自然收到了数值。

空讲理论光说不练可能有些难懂。 下面做个例子来演示下实现方法以及实际的应用。

通常 SMART 的子程序的管脚数量有限制，可能是 16 个，数值不精确，但是没意义，总之系统给的再多，在做标准化库函数时也会不够用。所以总要有有扩张管脚的需求。

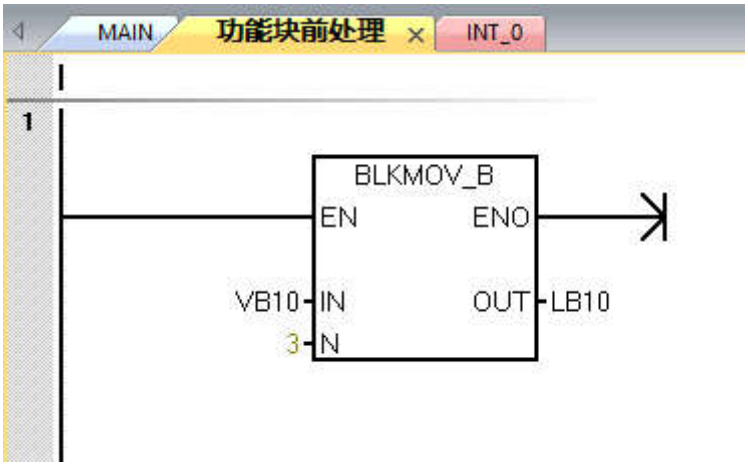
我们有建立一个子程序叫做“功能块”：

变量表

	地址	符号	变量类型	数据类型	注释
1		EN	IN	BOOL	
2	L0.0	START	IN	BOOL	
3			IN		
4			IN_OUT		
5	L0.1	QDONE	OUT	BOOL	
6			OUT		
7	LD1	DUMP1	TEMP	DWORD	
8	LD5	DUMP2	TEMP	DWORD	
9	LB9	DUMP3	TEMP	BYTE	
10	L10.0	IN11	TEMP	BOOL	
11	L10.1	IN12	TEMP	BOOL	
12	L10.2	IN13	TEMP	BOOL	
13	L10.3	IN14	TEMP	BOOL	
14	L10.4	IN15	TEMP	BOOL	
15	L10.5	IN16	TEMP	BOOL	
16	L10.6	IN17	TEMP	BOOL	
17	L10.7	IN18	TEMP	BOOL	
18	L11.0	IN19	TEMP	BOOL	
19	L11.1	IN20	TEMP	BOOL	
20	L11.2	IN21	TEMP	BOOL	
21	L11.3	IN22	TEMP	BOOL	
22	L11.4	IN23	TEMP	BOOL	
23	L11.5	IN24	TEMP	BOOL	
24	L11.6	IN25	TEMP	BOOL	
25	L11.7	IN26	TEMP	BOOL	
26	L12.0	IN27	TEMP	BOOL	
27	L12.1	IN28	TEMP	BOOL	
28	L12.2	IN29	TEMP	BOOL	
29	L12.3	IN30	TEMP	BOOL	
30	L12.4	IN31	TEMP	BOOL	
31	L12.5	IN32	TEMP	BOOL	
32	L12.6	IN33	TEMP	BOOL	
33	L12.7	IN34	TEMP	BOOL	

其原生管脚很少，然而我们把更多的管脚建立到 LB10,LB11,LB12 的 L 数据区中。

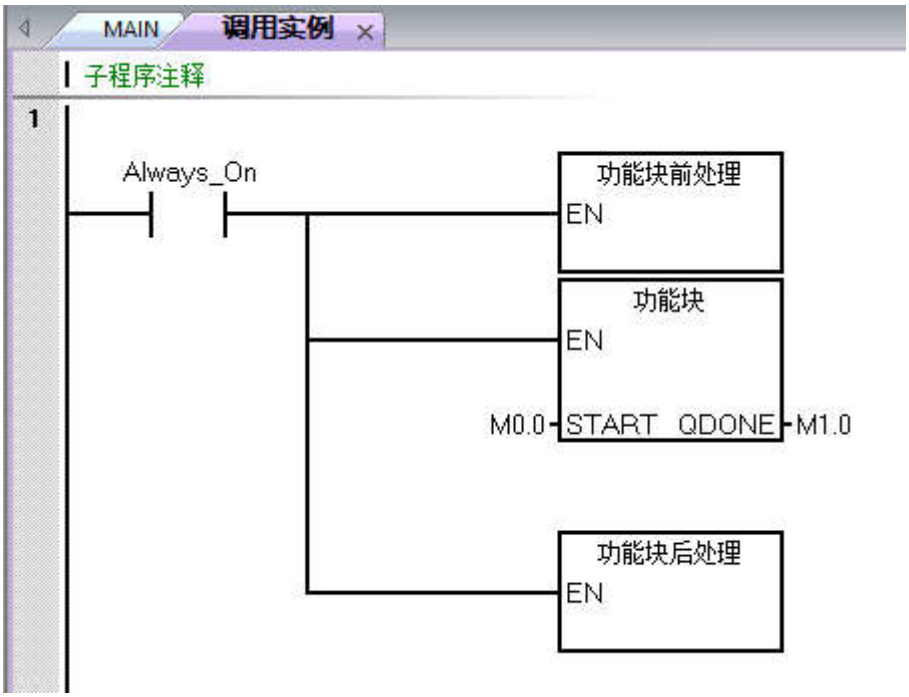
做一个“功能块前处理”的子程序 ，在其中调用 BLOCKMOVE 功能:



即实现了对内部 TEMP 变量的批量赋值，VB10 的数据可以作为功能块的 INPUT 数据使用了。
当然，我们还可以做个“功能块后处理”，数值的传送方向颠倒一下：

那么建立的 IN11-IN34 的临时变量事实上成为了功能块的 INOUT 管脚。

程序的调用过程变为：



如此实现了给功能块扩张了 24 个管脚。 如果还需要更多，则简单修改即可。

我做的标准化烟台方法的分享项目的 SMART 200 的程序， 在 2022 年春节左右，做了一次 V2.0 的升级， 其中包含了本文讲解的技术方法，使得程序更加简练，优雅， 封装更彻底。

V2.0 的升级程序不是完整的项目程序，而只是做了一部分功能的演示， 学员需要在加入学习营后，从 QQ 群文件中自行下载。学习掌握之后改进升级到所有程序功能。 在掌握之前，还是仍然可以只学习 V1.0 的方法，以及工程应用。其实只是我一个人的心结， 追求更优雅更完美的程序架构，实际功能是一样的。