PLC 基础与实际应用讲座

PLC 技术网

http://www.plcjs.com

PLC 技术网是一个专门致力于 PLC 综合应用服务的互联网运营平台，也是目前中国国内唯一一家专门从事 PLC 技术研究的可编程控制器技术门户型网站。

PLC论坛: http://bbs.plcjs.com 或 http://www.plcjs.com/bbs

博客地带: http://blog.plcjs.com 或http://www.plcjs.com/blog

下载中心: http://download.plcjs.com 或http://www.plcjs.com/download

搜索中心: http://s.plcjs.com 或http://www.plcjs.com/search

工控图文网址导航: http://dh.plcjs.com 或http://www.plcjs.com/dh

可编程控制器技术门户

2005.12.25

德力工作室全体人员

PLC 之一 什么是 PLC？, PLC 简史

PLC（即可编程逻辑控制器，Programmable Logic Controller）是用来取代用于电机控制的顺序继电器电路的一种器件。PLC 通过检查输入的状态来控制它的输出点的闭合与打开。用户输入一段程序（通常是通过软件），PLC 就给出相应的结果。

PLC 在实际中用的很多。只要有工业的地方，就有 PLC 存在的机会。如果你就在机器制造、包装、物料输送、自动装配等行业中工作，那么你可能已经在使用它了。如果没有的话，你就是正在浪费金钱和时间。几乎所有需要电气控制的地方都需要 PLC。例如，我们假设当一个开关闭合时，我们想让一个电磁阀先开启 5 秒钟，然后闭合，而不管开关处于什么状态。我们用一个简单的外部定时器就可心完成这一动作。但是，如果该过程有 10 个开关和 10 个电磁阀呢？我们需要 10 个外部定时器。如果在这一应用中也需要对开关的打开次数单独计量呢？我们需要大量的外部计数器。

可以看出，一个控制过程越复杂，我们就越需要使用 PLC。使用 PLC，我们可以很容易地对它的输入计数，并且在指定的时刻开启电磁阀。

了解一下 PLC 的发展史，或许对学习有很大的帮助。在上个世纪 60 年代末期，PLC 第一次被提出。设计这样一个器件的主要目的是为了降低当时使用复杂的继电器控制的电机控制系统的巨额成本。Bedford 联盟（Bedford，MA）向美国一家大型汽车制造商提议制作模块化数字控制器（MODICON，Modular Digital Controller）。当时，其它公司提出基于计算机的计划，其中一项是基于 PDP-8。MODICON 084 是世界上第一块成为商品的 PLC。

当生产要求变化的时候，对控制系统的要求也在变化。当变化非常频繁的时候，成本是非常高的。因为继电器是机械元件，它们的寿命都是有限的，这就需要定制一份非常严格的保养计划。当使用的继电器非常多的时候，故障诊断也是非常令人头痛的事情。现在想像一下：一块电机控制面板，上面布满了许多，或许是成百成千的独立的继电器。单其尺寸也要让人考虑半天。这么多的独立元件，其最初的接线是多么的复杂！这些继电器必须按规定用导线接在一起才能产生要求的输出结果。这样会有问题吗？你打赌：会有！

对保养人员和工厂的工程师来说，这些"新型的控制器"也必须是易于编程的。它们的寿命必须足够长，而且程序的修改也应该非常容易。它们还必须能够适应恶劣的工厂环境。要求是不是很多！答案是使用大多数人已经熟悉的可编程技术，并用固态元件（如可控硅）代替机械元件。

上世纪 70 年代中期，占支配地位的 PLC 技术是序列发生器状态机（sequencer state-machines）和基于 CPU 的位片（bit-slice)技术。AMD 2901 和 2903 在 Modicon 和 A-B PLC 中非常流行。传统的微处理器除了用于最小的 PLC

以外，缺乏快速处理 PLC 逻辑的能力。随着传统微处理器的发展，越来越大的 PLC 基于微处理器。但时至今日，有些 PLC 仍基于 2903。Modicon 已经开发出比他们的基于 2901 的 984A/B/X 速度更快的 PLC。

PLC 开始具有通信能力，大约是在 1973 年。第一个这样的系统是 Modicon 的 Modbus。从此 PLC 之间可以相互对话了，也可以离得它们控制的电机远远的。它们也可以发送和接收各种电平，从而进入模拟控制的世界。不幸的是，缺乏统一的标准和技术的不断改进，使得协议和物理网络均不兼容，从而 PLC 通信变成了一场恶梦。但对 PLC 来说，仍是伟大的 10 年。

80 年代，人们试图用通用电机（General Motor）的生产自动控制协议（manufacturing automation protocol, MAP）来将 PLC 的通信标准化。当时人们也正在努力减小 PLC 的尺寸，并使得它们可以在个人电脑上用符号编程，而不再使用专门的编程终端或手持式编程器。今天，世界上最小的 PLC 同一只控制继电器的大小差不多。

90 年代，新协议的产生和 80 年代幸存的一些较流行协议的物理层的现代化逐渐减少。最新的标准（IEC 1131-3）已经尽量将 PLC 编程语言融合为一个国际标准。现在，我们可以同时使用功能模块图（function block diagrams）、指令表（instruction lists）、C 和结构化文本（structured text）来对 PLC 编程。在许多场合，PC 已经取代了 PLC。当初被委托生产 MODICON 084 的那家公司实际已经转型生产基于 PC 的控制系统。

21 世纪将会怎样呢？让时间自己来说吧！

PLC 之二 PLC 的内部结构

PLC 主要包括一片 CPU，存储区和用来接收输入/输出数据的相应电路。实际应用中，我们可以把 PLC 看做一个装满了成百上千个分立的继电器、计数器、定时器以及数据存储区域的盒子。这些计数器、定时器真的存在吗？不，实际上它们是不存在的，它们是模拟的，可以把它们看作是软件计数器和定时器。这些内部继电器是通过寄存器内部的数位位置（bit location）来模拟的。那么各个部分是做什么的呢？

• 输入继电器（接触器）（INPUT RELAYS ）：它们与 PLC 的外围电路相连。它们是实际存在的，并从开关、传感器等外围元件接收信号。典型的输入不是用继电器，而是用晶体管。

• 内部应用继电器（接触器）（INTERNAL UTILITY RELAYS-）：它们不从外界接收信号，实际上也不存在。它们是模拟的继电器，就是它们使得 PLC 取代了外部继电器。有一些专用继电器，仅用来完成某一任务。有些是常开的，有些是常闭的。有些仅在上电期间开，典型应用是用来初始化存储的数据。

• 计数器（COUNTERS）：它们也不是实际存在的。它们是模拟的计数器，编程后可以用来对脉冲计数。典型的计数器可以做加计数、减计数和双向计数。因为它们是模拟的，所以限制了它们的计数速度。有些厂家也在 PLC 中加入基于硬件的高速计数器。我们可以认为它们是实际存在的。大多数情况下，这些计数器可以做加计数、减计数和双向

计数。

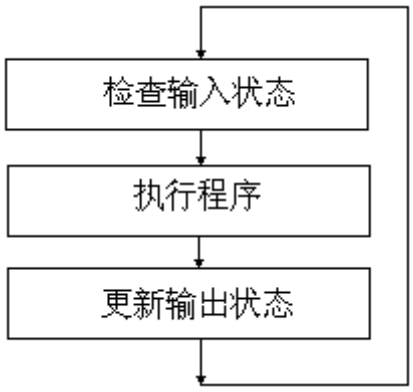
• **定时器（TIMERS）：**它们也并非实际存在。它们可以有多种变量和增量。最常见的类型是闭合延时型。另外还有打开延时型和保持、非保持型。定时器增量从 1ms 到 1s。

• **输出继电器（线圈）（OUTPUT RELAYS ）：**它们连接到 PLC 的外围电路。它们是实际存在的，并向电磁线圈、灯等发出开/关信号。它们可以是晶体管、继电器或者三端双向可控硅开关元件，这取决于 PLC 类型的选择。

• **数据存储（DATA STORAGE）：**典型应用中，可使用寄存器来存储数据。它们通常作为数据处理的临时存储器。当 PLC 断电时，它们也可用来存储数据。当上电时，它们仍保持与断电前相同的内容。非常方便，也非常有必要！

PLC 之三 PLC 的运行

PLC 的工作就是对一段程序连续扫描。我们可以把这种扫描看成包括三个重要的步骤。虽然肯定不止三个步骤，但是我们集中精力考虑重要的部分，而不用担心其余的部分。其余部分的典型作用是作系统检查和刷新当前内部计数器和定时器的值。



步骤 1－检查输入状态：首先，PLC 检查一下每个输入点，看它们是闭合还是打开。换句话说，连接到第一个输入点的传感器闭合吗？第二个呢？第三个呢？.....它将这些数据存入内存，以备在下一步使用。

步骤 2－执行程序：然后，PLC 执行你的程序，每次执行一步。你的程序或许是这样的：如果第一个输入点闭合，那么闭合第一个输出点。因为程序已从上一步中知道输入的开关状态，所以它能够根据第一个输入点的状态，决定第一个输出点是否应该闭合。PLC 将执行结果存起来以备下一步使用。

步骤 3－刷新输出结果：最后 PLC 刷新输出点的状态。它刷新的根据是第一步中读取的输入点状态和第二步中程序执行的结果。还是举第 2 步中的例子，此时 PLC 应将第一输出点闭合。原因是第一个输入点是闭合的，而且你的程序要求在这种状态下闭合第一个输出点。

步骤 3 执行完毕，PLC 返回到步骤 1，连续重复以上步骤。一次扫描时间（one scan time）就是 PLC 执行以上所列的 3 个步骤所需的时间。

PLC 之四 响应时间（Response Time）

当我们购买 PLC 的时候，其总响应时间是我们必须考虑的一个因素。就像我们的大脑一样，PLC 在对某种变化做出响应前也要花一定的时间。有些场合下速度并不重要，而有些场合则不然.....

如果你在学习本课程的时候抬一下头，你可能发现墙上有幅画。肯定是你的眼睛先看到那幅画，然后你的大脑反应出"墙上有一幅画"。在这个例子中可以把你的眼睛看作传感器。眼睛连接到你的大脑输入电路中。你的大脑输入电路会花一些时间来认知你看到的东西。（如果你喝了酒，那么这种输入响应时间会变长！）最后，你的大脑认识到眼睛看到了什么东西，并开始处理这些信息。然后，大脑向你的嘴发送输出信号。你的嘴收到该信号，然后响应它。终于你的嘴里说出这样的话，"哼，那幅画真难看！"。

输入：大脑要接收到来自眼睛的输入信号需要花费一定的时间。

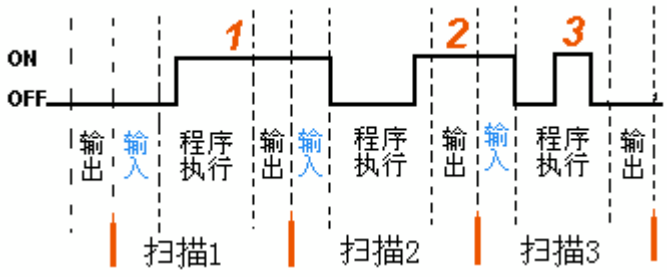
执行：大脑处理来自眼睛的信息需要花费一定的时间。把程序看作：如果眼睛看到难看的图画，然后输出相应的话到嘴巴。

输出：嘴巴收到来自大脑的信号，最后说出："哼，那幅画真难看！"

〔输入响应时间〕＋〔程序执行时间〕＋〔输出响应时间〕＝〔总响应时间〕

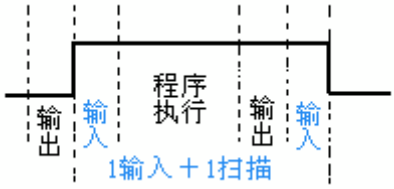
为何要关心响应时间？

前面我们已经了解了一下什么是响应时间，现在我们来看一下它在实际应用中到底意味着什么。只有当 PLC 扫描其输入的时候，它才能看到它们的开关状态。换句话说，在扫描输入状态期间它才能看到它的输入状态。



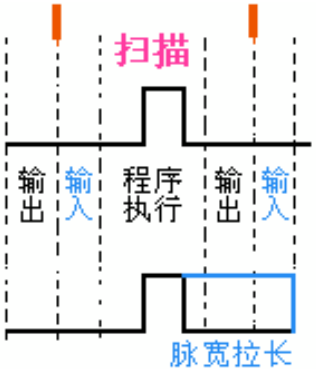
在上图中，输入 1 直到扫描 2 才会被检查到。这是因为当输入 1 变为"ON"时，扫描 1 已经完成了对输入状态的检查。同样，输入 2 直到扫描 3 才会被检查到。这也是因为当在输入 2 变为"ON"时，扫描 2 已经完成了对输入状态的检查。而输入 3 不会被检查到。这是因为当扫描 3 检查输入状态时，信号 3 仍为变为"ON"；而在扫描 4 检查输入状态时，它已经变为"OFF"了。所以信号 3 不会被 PLC 检查到。

为了避免这种情况的发生，输入信号至少应保持〔1 输入延时时间〕＋〔1 扫描时间〕。

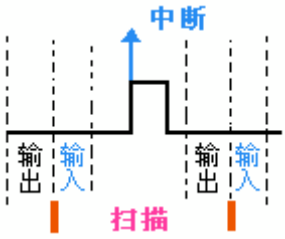


但是，如果输入不能保持这么长时间呢？那么 PLC 就不能检查到输入状态的变化。所以只好拿它作镇纸用了。当然不能这样，我们必须找到一个办法来解决这一问题。实际上有两种办法。

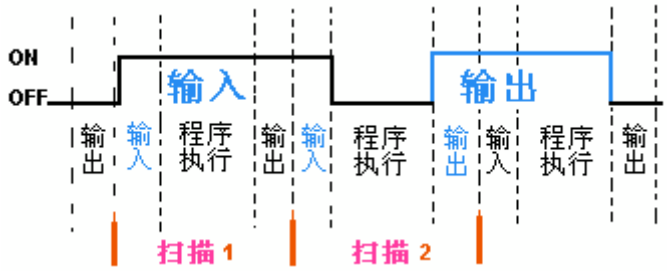
1．**脉冲拉长功能。**该功能将输入信号拉长，直到在下一个扫描中检查到该输入信号。



2．**中断功能。**该功能中断扫描去处理一个指定的子程序（你已经写好的）。即输入一变为"ON"，不管扫描到什么地方，PLC 马上停下来去执行中断子程序。（可以认为子程序是主程序外的一小段程序。）当 PLC 执行完中断子程序后，它返回到中断点继续执行常规扫描。

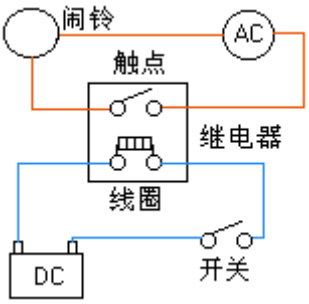


现在让我们来看一下输出响应输入的最长时间。我们假设当一个开关闭合时，我们要闭合一个连接到 PLC 输出端子的一个负载。下图所示为从输入闭合到输出闭合的最大延时（因为直到扫描 2 才检查到输入，所以是最坏的情况）。最大延时＝〔2 扫描周期－1 输入延时时间〕。



PLC 之五 继电器

现在我们明白了 PLC 是怎样处理输入、输出和实际程序的，下面我们马上就要开始编写程序了。但首先让我们看一下实际的继电器是怎样工作的。毕竟，PLC 的主要用途是替代"真实世界"中的继电器的。我们可以把继电器看作是电磁开关。给线圈加一个电压，产生一个磁场。该磁场使继电器的触点闭合，使它们连接在一起。可以把这些触点看作是开关。它们允许电流流过，从而将电路闭合。来看一下下面的例子。无论开关什么时间闭合，我们都让闹铃响（午餐时间到！）。我们使用 3 个真实的元件：一个开关、一个继电器和一个闹铃。无论开关什么时间闭合，我们都给闹铃一个电流，使它响起。



注意：在上图中我们使用了两个分立的电路。下面的电路（蓝色）为直流（DC）部分。上面的电路（红色）为交流（AC）部分。

在这个例子中我们使用一个直流（DC）继电器来控制一个交流（AC）电路。这正是继电器有趣的地方！当开关打开的时候，没有电流流过继电器线圈。但当开关一闭合，电流马上流过线圈，建立起一个电磁场。该电磁场使得继电器触点闭合。现在交流电流过闹铃，使它发出响声。一只典型的工业继电器：

如果您以前对继电器电路了如指掌，那么您学起来就容易多了.

PLC 之六 替代继电器

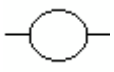
下面，让我们用 PLC 替代继电器。（在这个例子使用 PLC 不会为我们节约多少成本，我们只是用它来展示一下 PLC 的基本应用。）首先必须要做的就是一个叫做梯形图的东西。看过一些这种图后，大家就会明白它们为什么叫做梯形图。我们也必须画一张梯形图，因为 PLC 不理解示意图。它只能识别代码。幸运的是，大多数的 PLC 都有特定的软件可以将梯形图转换为代码。所以实际上我们不用学习 PLC 的代码。

第一步：我们必须将示意图中所有的元件翻译成 PLC 能够理解的符号。PLC 不能理解像开关、继电器、闹铃等这样的元件，它只能理解输入、输出、线圈、触点等等。它不关心实际的输入或输出元件到底是什么，它只关心它们是输入还是输出。

首先我们用一个符号来替代电池。这个符号对所有的梯形图都是通用的。我们画出母线，它们就是两根竖线，分别位于图的两侧。左边一根作为正电压，右边一根作为地。电流（逻辑）就是从左边流向右边。然后我们用一个符号来代表输入。这在个基本的例子里面，我们有一个真实的输入（即开关）。我们将开关连接的地方作为输入，符号如下。这个符号也用来表示继电器的触点。

一个触点符号： 

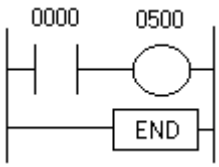
下面我们给输出一个符号。在这个例子中我们有一个输出（即闹铃）。我们用符号取代闹铃，放在它实际连接的位置。这个符号用来表示一个继电器的线圈。

一个线圈符号： 

交流电源是外部供给的，我们不把它画进梯形图。PLC 只关心要将哪一个输出变为"ON"，而不关心连接到它的到底是什么。

第二步：我们必须告诉 PLC 每个元件的位置。换句话说，我们必须给每个元件一个地址。实际中，开关接在 PLC 的什么地方呢？闹铃呢？将 PLC 比作一个城镇，如果你不知道你的朋友们的地址的话，你能找到你的朋友们吗？你知道他们生活在同一个城镇，但他们在哪一幢房子里呢？PLC 城镇有大量的房子（输入和输出），但我们必须规划好谁应该住在哪里（器件接到哪里）。我们将在后面学习 PLC 的地址规划。每个 PLC 生产商的地址分配都不一样。现在我们暂且把我们的输入叫作"0000"。输出叫作"500"。

最后一步：我们必须将示意图转换为事件逻辑序列。其实这做起来比听起来要容易得多。我们要将要画的图告诉 PLC 当某件事情发生后它该做什么。在我们的例子中，我们必须告诉 PLC，当操作者闭合开关时它应该怎么做。显然，我们想让闹铃闹响，但 PLC 不知道那是什么。它是个愚蠢的设备，不是吗！

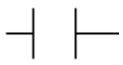


上图是最后转换好的梯形图。注意，我们用符号替代了真实的继电器。PLC 确实可以按这张图来执行。不要担心，我们后面会举更多的例子来说明。

PLC 之七 基本指令

Load

LoaD（LD）是一个常开触点。它有时也叫做 XIO（examine if on，检查是否为"ON"）。（例如用于检查输入是否为"ON"）加载指令的符号如下。

LoaD（触点）的符号： 

它用于代表需要变为"ON"的输入信号。当物理输入为"ON"时，我们说该指令为真（True）。我们检查输入有没有"ON"信号。如果输入为"ON"，那么该符号为"ON"。"ON"状态也指逻辑 1 状态。

该符号通常可以用作内部输入、外部输入和外部输出触点。记住，内部继电器在物理上是不存在的。它们是模拟（软件）继电器。

LoadBar

LoaDBar 指令是一个常闭触点。它有时也叫做 LoaDNot 或 XIC（examine if closed，检查是否闭合）。LoadBar 指令的符号如下。

LoaDNot（常闭触点）的符号： 

当输入信号变为"ON"时不需要被代表时使用。当物理输入为"OFF"时，我们说该指令为真（True）。我们用它来检查输入是否为"OFF"信号。如果输入在物理上为"OFF"，那么符号为"ON"。"OFF"状态也指逻辑 0 状态。

该符号常可用于内部输、外部输入，有时也用于外部输出触点。再提示一遍，内部继电器在物理上是存在的。它们是模拟的（软件）。它与 Load 恰好相反。

*注意：在大多数的 PLC 中，必须将指令（Load 或 Loadbar）放在梯形图左侧第一位。

逻辑状态	---Out--	--OutBar-
---0----	--False-	--True---
---1----	--True--	--False--

Out

Out 指令有时也叫做 OutputEnergize（输出激励）指令。输出指令象一个继电器线圈。它的符号如下。

Out（线圈）的符号： 

当梯形横档上它前面的指令均为真（True）时，它也为真（True）。当该指令为真（True）时，它在物理上为"ON"状态。我们可以认为该指令是一个常开输出。它既可用于内部线圈，也可用于外部输出。

OutBar

OutBar 指令有时也叫做 OutNot（输出非）指令。一些厂家的 PLC 没有这条指令。OutBar 指令像一个常闭继电器线圈。它的符号如下。

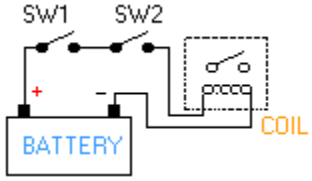
OutBar(常闭线圈)的符号： 

当该指令所在横档的前面的指令通路为假（False）时（即前面的指令不全为真），它为真（True）。当指令为真时，它实际输出"ON"。我们可以把这条指令看作是一个常闭输出。它既可用于内部线圈，也可用于外部输出。它与 OUT 指令恰恰相反。

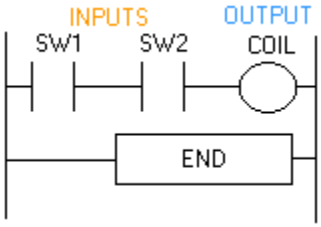
逻辑状态	---Out--	--OutBar-
---0----	--False-	--True---
---1----	--True--	--False--

PLC 之八 个简单的例子

现在让我们比较一下一个简单的梯形图和实际的继电器电路接线图，看看它们之间有什么区别。



在上面的电路图中，当电池的正负极构成闭合回路时，线圈被激励。我们可以用一个梯形图来模拟该电路。梯形图包含像真梯子一样的横档。每个横档必须包含一个或 多个输入和一个或多个输出。横档上的第一条指令通常必须是一条输入指令，最后一条指令通常应该是一条输出指令（或其等效指令）。

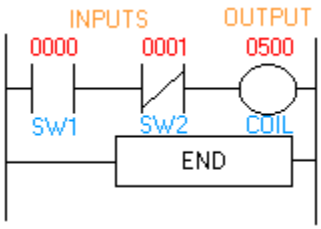


注意：在这个只有一个横档的简单的梯形图中，我们已经用一个梯形图模拟了上面提到的电路图。在这个梯形图中，我们使用了 Load 和 Out 指令。有些厂家要求 每个梯形图的最后一个横档必须为包含 END 指令的横档。有些 PLC 还需要在 END 横档后面再加一个包含 ENDH 指令的横档。

下一节我们将学习一下寄存器。什么是寄存器？下节课继续学.....

PLC 之九 PLC 的寄存器

现在我们把上一节例子中的梯形图的开关 2（SW2）换成一个常闭触点（LoadBar 指令）。最初，SW1 的实际状态为"OFF"，SW2 的实际状态为"ON"。更改后的梯形图如下：



注意:我们也给每一个符号（或指令）分配了一个地址。这些地址在 PLC 的数据文件中留出一定的存储区域，可以存储指令的状态（即真（True）/假（False）状态）。许多 PLC 使用 16 位存储结构。在上面的例子中，我们使用两个不同的存储单元或寄存器。

寄存器 00															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
														1	0
寄存器 05															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
															0

在上表中，我们可以看出在寄存器 00 中，位 00（即输入 0000）是一个逻辑 0 和位 01（即输入 0001）是一个逻辑 1。寄存器 05 所示位 00（即输出 0500）是一个逻辑 0。逻辑 0 或 1 表示指令为假（False）或真（True）。
*虽然在上表中寄存器的大多数项都是空的，但是它们每一个应该都是 0。将它们留空是为了强调我们关心的位。

指令的逻辑状态			
逻辑位	LD	LDB	OUT
Logic 0	False	True	False
Logic 1	True	False	True

只有横档上所有指令的状态都为真（True）时，PLC 才激励输出。从上表我们可以看出在前面的例子中，当且仅当 SW1 为逻辑 1，且 SW2 为逻辑 0 时，线圈 才为真（即被激励）。如果横档上输出（线圈）前面的指令有一个为假（False），那么输出（线圈）为假（False，未激励）。

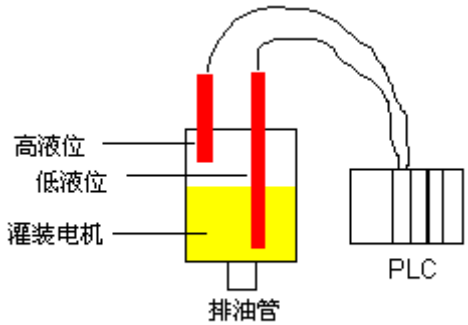
现在让我们看一下前面那个程序的真值表，通过真值表更能看出这一点。真值表列出了两个输入状态的所有可能的组合。

输入		输出	寄存器逻辑位		
SW1(LD)	SW2 (LDB)	COIL (OUT)	SW1 (LD)	SW2 (LDB)	COIL (OUT)
False	True	False	0	0	0
False	False	False	0	1	0
True	True	True	1	0	1
True	False	False	1	1	0

注意:从上表中可以看出，当输入状态改变时，输出随着改变。当同一横档上输出前面的指令全为真（True）时，输出才为真（True，被激励）。

PLC 之十 在液位控制中的应用

让我们看下面的应用：
我们要控制桶中润滑油的分配。这使用两个传感器就可以实现。我们将一个传感器放在靠近底部的地方，另一个放在靠近顶部的地方，如下图所示。

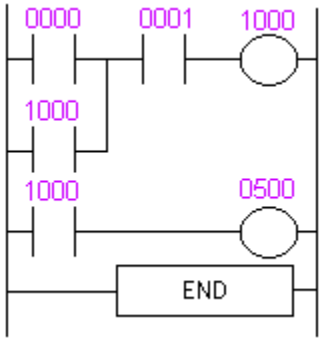


从桶中分配油

在这里，我们希望灌装电机一直将润滑油吸进桶中，直到上液位传感器动作。在那一点，关闭电机，直到液位降到低于下液位传感器时，我们再打开电机。如此重复运行。这里我们需要 3 个 I/O 口（即输入/输出口），两个作为输入（传感器），我个作为输出（灌装电机）。两个输入都是 NC（常闭）光导纤维液位传感器。当它们未浸入液体时为"ON"，当它们浸入液体时为"OFF"。我们给每个输入和输出一个地址，使得 PLC 可以知道它们的物理连接位置。地址分配如下表所示：

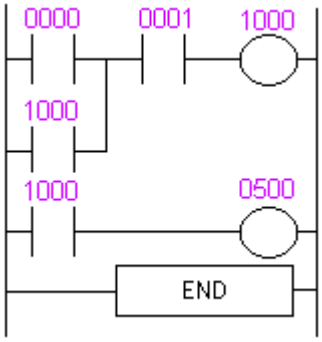
输入	地址	输出	地址	内部应用继电器
低	0000	Motor	0500	1000
高	0001			

下面是实际的梯形图。注意，在这个例子中，我们使用了一个内部继电器。这些继电器的触点可重复使用任意多次。我们使用了两次来模拟一个具有 2 组触点的继电器。记住，这些继电器并不是 PLC 中真正的继电器，它们是可用来模拟继电器的寄存器位。

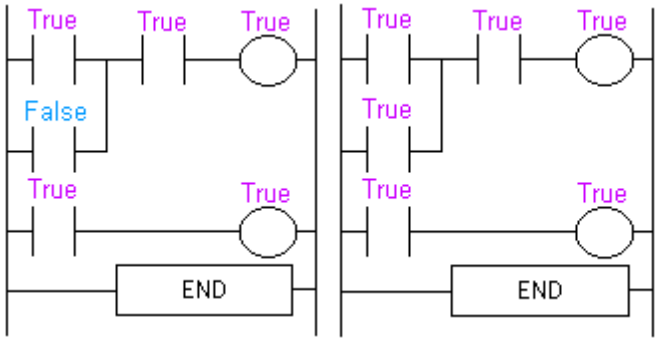


还有一点应该记住，就是我们使用 PLC 的最主要原因就是用它来替代实际的继电器。内部继电器让我们实现了这一目的。我们不可能列出每个牌子的 PLC 各包含多 少个内部继电器。有的是几百个，有的是几千个，还有的是几万个。PLC 的规格（不是物理规格，而是 I/O 规格）是决定因素。如果我们正在使用一个只有几个 I/O 的微型 PLC，我们不需要很多内部继电器。但是，如果我们使用一个大型的、带几百个或几千个 I/O 的 PLC，我们确实需要一定数量的继电器。那么厂家提供的内部继电器到底够不够用呢？可以参考生产商的使用说明书。总的来说，即便最大型的应用，厂家提供的数量应该也是足够用的。

PLC 之十一 程序扫描



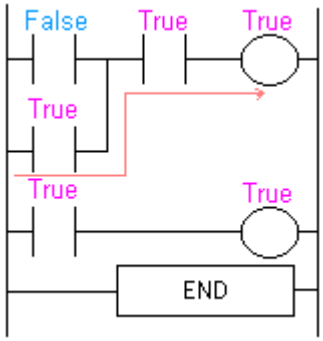
开始的时候储油桶是空的。所以，输入 0000 和输入 0001 都为 TRUE。



扫描 1　　扫描 2－100

因为 500（灌装电机）一直开着，所以储油桶逐渐地满起来。

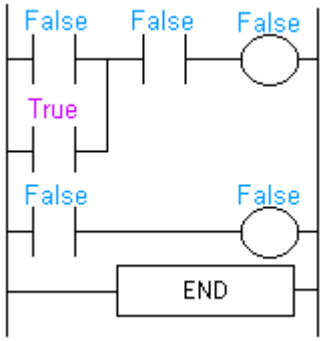
经过 100 次扫描以后，油位上升到高于上液位传感器，上液位传感器打开（即为 FALSE）。



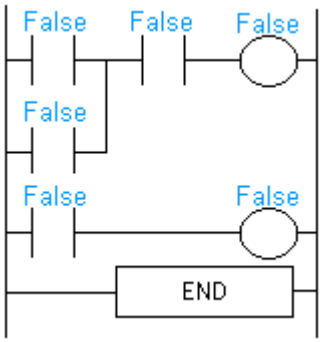
扫描 101－1000

注意：即使当上液位传感器为 FALSE，从左到右仍有一条真值的逻辑通道。这就是我们为什么使用一个中间继电器的原因。继电器 1000 将输出将输出闭锁（500）在"ON"状态。输出将保持该状态，直到没有从左到右的真值逻辑通道为止（即当 0001 变为 FALSE 时）。

经过 1000 次扫描以后，油位上升到高于上液位传感器，使上液位传感器也打开（即为 FALSE）。



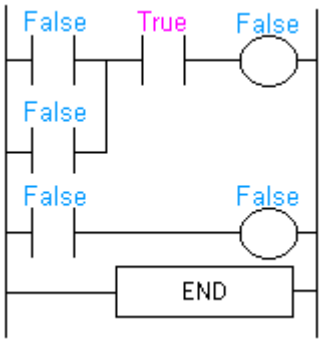
扫描 1001



扫描 1002

因为再没有真值逻辑通道，所以输出 500 不再被激励（TRUE），所以电机停止。

经过 1050 次扫描以后，油位下降到低于上液位传感器，使得它又变为 TRUE。



扫描 1050

注意：即使上液位传感器变为真，仍然没有从左到右的真值逻辑通道，所以线圈 1000 保持为 FALSE！

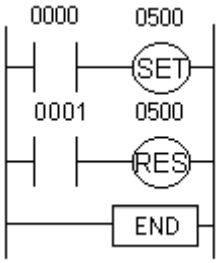
经过 2000 次扫描以后，油位降到低于下液位传感器，使得它也变为 TRUE。此时的逻辑状态变得与开始时的扫描 1 相同，程序将按上面的扫描过程重复执行。

PLC 之十二 锁存指令

想想前面的午餐闹铃例子. 如果我们找不到一只"按入按出"式(即自锁式)按钮怎么办呢? 我们在响铃的时候就不得不一直按住按钮. (一只瞬时开关) 锁存指令让我们使用瞬时开关和 PLC, 当我们按一下开关的时候铃声响起, 当我们再按一下的时候, 铃声停止.大虾您也许你会自言自语, "你在讲什么鬼东西?". (我有时也那么想!) 那么就让我们看一个真实的例子吧.

想想你是怎样使用遥控器来控制你的电视的. 它有一个"开"按钮, 还有一个"关"按钮. (总之我用过的都是那样) 当我按一下"开"的时候, 电视机打开; 当我按一下"关"的时候, 电视机关闭. 我并不需要一直按着"开"使电视机处于打开状态. 这就是锁存指令的作用.

锁存指令通常叫作 SET 或 OTL(outputlatch, 输出锁存). 解锁指令通常叫作 RES(reset, 复位), OUT(output unlatch, 输出解锁)或者 RST(reset, 复位). 它们的使用如下图所示.



在 这个例子中，我们使用了两个瞬间按钮开关. 一个接输入 0000，另一个接输入 0001. 当操作者按下开关 0000 时，指令"set 0500"变为真(TRUE)，输出(0500)变为 ON. 即使操作者不再按这个开关，输出(0500)仍保持为 ON. 即它被锁存为 ON. 使输出变为 OFF 的唯一方法就是使输入 0001 变为 ON. 这将使得指令"res 0500"变为真(TRUE)，从而将输出 0500 解锁或复位.这样就有一个问题：如果输入 0000 和 0001 恰在同时变为 ON，将会出现什么结果呢？输出 0500 是被锁存呢，还是被解锁？ 要回答这个问题，我们不得不考虑扫描顺序. 梯形图总是从上到下，从左到右被扫描的. 扫描的第一步是输入的物理连接状态. 0000 和 0001 同时为 ON. 接下来 PLC 执行程序. 从左上开始，输入 0000 为真(TRUE)，所以"set 0500"为真. 当执行到下一横档，既然输入 0001 为真，那么"reset 0500"为真. 所以在扫描的最后部分，PLC 刷新输出的时候，0500 将保持为 OFF(即 reset 0500).
是不是对 PLC 有了更深的认识了，让我们继续学习精通它吧!

PLC 之十三 计数器

你用的是什么类型的计数器?比如，有加法计数器(它们只能正向计数 1,2,3,...). 它们在英语中被缩写为 CTU(count up，升值计数)，CNT，C，或者 CTR. 有减法计数器(它们只能逆向计数 9,8,7,...). 当它们作为一条独立的指令时，通常被叫做 CTD(count down，减值计数). 还有双向计数器(它们可双向计数 1,2,3,4,3,2,3,4,5,...). 当它们作为一条独立的指令时，通常被叫做 UDC(up-down down counter，加-减计数器).

许多厂家只有一种或两种类型的计数器，但这些计数器应能完成加计数，减计数或双向计数. 是不是有些混淆了？难道就没有一相标准吗？不要担心，计数器就是计数器，不要管生产商怎样称呼它们.

更容易引起混淆的是，大多数的生产商还加入了一定数量的高速计数器. 通常叫它们 HSC(high-speed counter),CTH(CounTer High-speed?)或者别的名称.

典型的高速计数器是一个"硬件"设备. 而上面所列的普通计数器多是"软件"计数器. 换句话说，它们并不是真正存在于 PLC 中，它们只是用软件模拟的计数器. 而硬件计数器却是真正存在于 PLC 中的，它们不依赖 PLC 的扫描时间.

按照拇指理论(rule of thumb)，一般情况下多使用普通(软件)计数器，除非所要计数的脉冲比 2 倍的扫描时间还要快. (例如扫描时间为 2ms，而所计脉冲每 4ms 或更长时间才来一次，那么此时我们使用软件计数器. 如果脉冲间隔小于 4ms(例如 3ms)，那么使用硬件(高速)计数器. (2*扫描时间 = 2*2ms = 4ms)

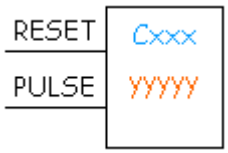
要使用计数器，我们必须知道以下三件事情:

1. 我们要计数的脉冲来自哪里. 典型情况下，它来自一个输入端子. (例如将一个传感器接到输入端 0000)
2. 在作出响应前，我们要计多少次. 例如计数 5 个玩具装入后开始打包.
3. 何时/怎样复位计数器，以便让它重新计数. 例如，我们计数 5 个玩具后，将计数器复位.

当程序在 PLC 上运行时，程序通常会显示当前或"累计"值，以便于我们观察当前的计数值.

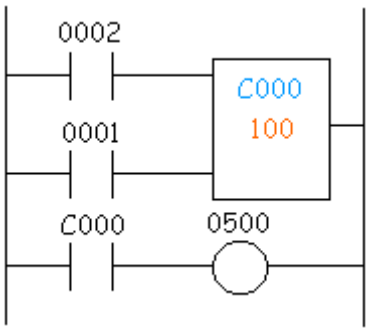
典型计数器的计数范围为 0 到 9999，-32768 到+32767，或 0 至 65535．为什么都是些这么古怪的数字呢？因为大多数 PLC 都是用的 16 位计数器．0-9999 是 16 位 BCD(binary coded decimal，二进制编码的十进制)码，-32768 到 32767 和 0 到 65535 是 16 位二进制码，我们在以后的章节会解释这是什么意思．

下面介绍一些我们将会碰到的指令符号(不同的厂家会有所不同)，并说明它们的用法．记住，它们虽然看起来不同，它用法基本都是相同的．如果我们会设置一个计数器，我们就会设置任意的计数了．

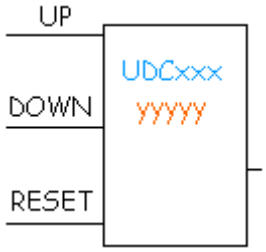


在这个计数器中，我们需要 2 个输入．一个接复位线．当该输入端为 ON 时，当前(累积)计数值将被清零．第二个输入接的是我们要计数的脉冲．例如，我们要对经过传感器前面的玩具计数，我们将传感器接到输入端 0001，然后将地址为 0001 的常开触点接在脉冲线的前面．Cxxx 是计数器的名称．如果我们想叫它计数器 000，那么在这里我们叫它"C000"．yyyyy 是我们在要求 PLC 做出响应前所要计的脉冲数．如果我们在将玩具打包前要计 5 个玩具，那么我们要该值改为 5．如果我们要计 100 个玩具，那么就将该值改为 100，等等．当计数器计数完毕(例如，我们计数了 yyyyy 个玩具)，它将一组独立的触点变为 ON，我们也将它标为 Cxxx．

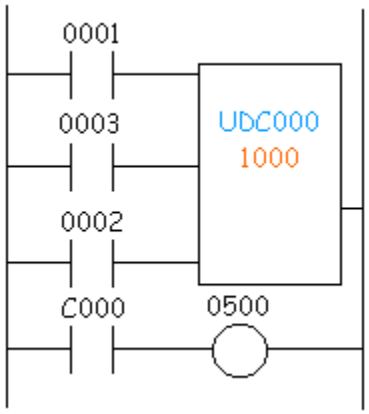
注意：计数器的累加值仅在脉冲输入的上升沿发生变化．



在上面的梯形图中，我们将计数器(叫做计数器 000)设置为从输入 0001 计数 100 个玩具，然后使输出 500 变为 ON．传感器 0002 将计数器复位．下面是我们会碰到的一个双向计数器．我们使用于上例相同的缩写(例如 UDCxxx 和 yyyyy)．



在这个双向计数器中，我们需要使用 3 个输入端．复位输入的功能与上例相同．但是，对于脉冲输入有两个．一个是加计数，一个是减计数．在这个例子中，我们把这个计数器叫做 UDC000，并且给它一个预设值 1000．(我们共要计数 1000 个脉冲) 在输入端，我们给输入端 0001 接上一个传感器，当它检测到目标时，使输入端 0001 变为 ON，给输入端 0003 也接上一个相同的传感器．当输入端 0001 变为 ON 时，PLC 正向计数，当输入端 0003 变为 ON 时，PLC 逆向计数．当计数值到达 1000 时，输出端 500 变为 ON．再次提醒注意的是，计数器的累计值仅在脉冲输入的下降沿改变．梯形图如下所示．



还有一件事要特别注意，在大多数的 PLC 中计数器和定时器的名称是不一样的．这是因为它们通常使用相同的寄存器．虽然我们还没有学到定时器，但我们必须记住这一点，因为它的确很重要.好了，上面讲的计数器可能有点难以理解，但只要我们用过一次，它们看起来就容易多了．它们的确是一种必要的工具．它们也是"非标准"基本指令之一．但是,有一点要记住，不管是哪个厂家生产的，用法都是一样的.

PLC 之十四 定时器

实际上有多种多样的定时器，这也是它们有趣的地方．通常，不同生产商都提供不同类型的定时器．下面是最常用的一些定时器:

• **延时 ON 定时器(On-Delay Timer)** -- 这种类型的定时器为"延时后变为 ON"．换句话说，当我们的传感器(输入)变为 ON 以后，等待 x 秒后，才激励一个电磁阀(输出)．这是最常见的定时器．它通常叫做 TON(timer on-delay，延时 ON 定时器)，TIM(timer，定时器)或 TMR(timer)．

• **延时 OFF 定时器(Off-Delay Timer)** -- 这种定时器与上面提到的延时 ON 定时器刚好相反．这种定时器仅仅"延时 OFF"．我们的传感器(输入)检测到目标后，激励电磁线圈(输出)．当传感器检测不到目标时，电磁线圈将保持被激励的状态 x 秒后才变为 OFF．这种定时器叫做 TOF(延时 OFF 定时器)，它较上面提到的延时 ON 定时器少见.(例如，很少生产商在其 PLC 中装入这种定时器)．

• **保持或累积定时器** -- 这种定时器需要两个输入端．一个输入端启动定时器(即时钟开始计时)，另一个输入端将定时器复位.如果输入传感器在整个定时时间内没有 ON/OFF，则上面提到的延时 ON 或 OFF 传感器将被复位.而当传感器中途断开时，这种定时器将保持当前状态，直到将其复位为止．例如，我们想知道在 1 个小时内传感器为 ON 的时间有多长．如果我们用上面提到的那些定时器，当传感器变为 OFF 或 ON 时，它们将保持复位．而保持或累积定时器，将给我们一个总的或累积的时间．我们通常把它叫做 RTO(Retentive Timer，保持定时器)或 TMRA(Accumulating Timer，累积定时器)．

下面我们来看一下怎样使用它们．典型地，我们需要知道两件事情:

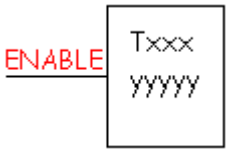
1. 用什么启动定时器．典型情况下为一输入．(例如连接到输入 0000 的一个传感器)
2. 在做出反应前，我们要延时多长时间．例如，我们在使一个电磁阀打开前要等待 5 秒钟.

当定时器符号前面的指令为真(True)时，定时器开始计数．当定时时间到达时，定时器将自动关闭它的触点．当程序在 PLC 上运行时，将显示逝去的或"累积的"时间，便于我们观察当前值．典型定时器的定时范围为 0 到 9999 或 0 到 65535 次.

为什么是这么怪异的数字呢? 这是因为大多数的 PLC 使用的是 16 位定时器. 我们将在以后学习这是什么意思, 现在我们只要知道 0-9999 是 16 位 BCD(Binary Coded Decimal, 二进制编码的十进制数), 0 到 65535 是 16 位二进制数就行. 时钟每计一次为 X 秒.

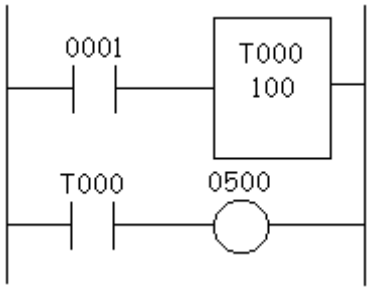
典型地, 每个生产商提供几种不同的计数单位. 大多数厂家提供 10 和 100ms 的增量. 一"ms"是一毫秒或 1/1000 秒. 一些生产商也提供 1ms 或 1 秒的增量. 这些不同增量的计数器工作起来和上面讲的一样, 只是有时为了表明它们的时基不同,它们的名称不一样. 有的叫做 TMH(High speed Timer, 高速定时器), TMS(Super high speed Timer, 超高速定时器), 或 TMRAF(Accumulating Fast Timer, 累积式快速定时器).

下面是一个典型的定时器指令符号, 我们该怎样使用它呢? 记住一点, 当它们外表看起来不同时, 它们的基本用法都是相同的. 如果我们会设置一个, 我们就会设置所有的定时器.



这个定时器为延时 ON 型的, 名称为 Txxx. 当定时器的使能输入端为 ON 时, 它开始计时. 当它计了 yyyy(预设值)次后, 它将它的触点变为 ON, 我们将在程序的后面使用该触点. 注意, 每次计数的间隔时间(增量)因生产商和所用时基不同而有所不同. (即, 计一个数可能为 1ms 或 1 秒或...).

下面是梯形图中的符号.

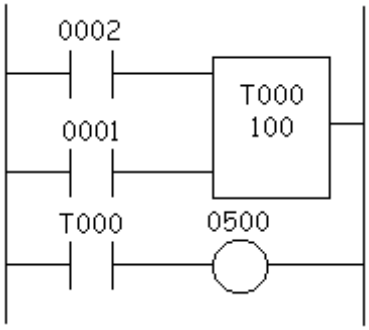


在该图中, 我们等待输入 0001 变为 ON. 当它变为 ON 时, 定时器 T000(一个 100ms 时基的定时器)开始计数. 它将计数 100 次. 每次(增量)为 100ms, 于是该定时器为一 10000ms(即 10 秒)定时器.100 次×100ms = 10,000ms. 当 10 秒钟过去后, T000 接触器闭合, 500 变为 ON. 当输入 0001 变为 Off(错误)时, 定时器 T000 将复位为 0, 同时使它的触点变为 OFF(变为 False), 所以使输出 500 变为返回 OFF 状态.

一个累积定时器看起来将如下图所示:

该定时器被命名为 Txxx. 当使能输入为 ON 时, 定时器开始计时. 当它计过 yyyy(预设值)次后, 它将使它的触点变为 ON, 以备我们在后面的程序中使用. 注意, 每计一个数的时间(增量)会因厂家和时基不同而不同(例如,1ms 或 1 秒或...). 但是, 如果使能输入在定时结束前变为 OFF, 当前值将被保留. 当输入返回到 ON 状态时, 定时器将从它停止的地方继续计时. 使定时器强制返回预设值状态的唯一办法就是重新启动.

在梯形图中，它的符号如下：



在该图中，我们等待输入 0002 变为 ON. 当它变为 ON 时，定时器 T000(一个 10ms 增量定时器)开妈计时. 它将计数 100 次. 每计一个数的时间(增量)为 10ms，于是定时器的定时时间为 1000ms(即 1 秒). 100 次×10ms=1,000ms. 当 1 少过去后，T000 的触点闭合，输出 500 变为 ON. 如果输入 0002 返回到 OFF 状态，当前计时值将被保留. 当 0002 返回到 ON 状态时，计时从停止点继续. 当输入 0001 变为 ON(True，真)时，定时器 T000 将复位到 0，同时使得它的触点变为 OFF 状态(变为 False，假)，所以输出 500 返回 OFF 状态.

重要的一点就是注意计数器和定时器不能拥有相同的名字(在大多数的 PLC 中). 这是因为它们使用相同的寄存器. 还有一点要时刻牢记，那就是虽然它们的符号看起来不一样，但是它们的工作方式都是一样的. 主要区别就是每计一个数所花的时间不一样.

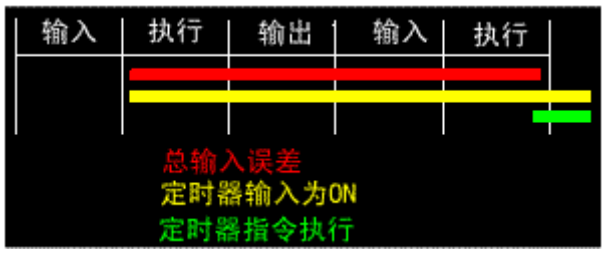
PLC 之十五 定时器的精度

在应用定时器时通常有两种误差. 第一种叫做输入误差. 另一种叫做输出误差. 总的误差是输入误差和输出误差之和.

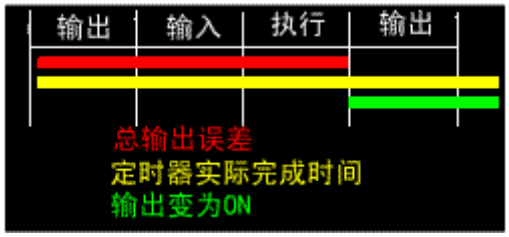
• **输入误差**-该误差的产生取决于定时器的输入在扫描周期内变为 ON 的时间. 当定时器的输入恰在 PLC 扫描完输入状态时变为 ON，输入误差最大. (即大于 1 个全扫描时间!). 这是因为，请回想一下，(见以前学过的扫描时间部分)在一个扫描周期内，输入只被扫描一次. 如果当 PLC 扫描输入的时候它没有变为 ON，而是在扫描完输入的时候变为 ON，显然有一个误差. 另外，我们还不得不在扫描周期的程序执行部分等待定时器指令的执行. 如果定时器指令是那一横档上的最后一条指令，那么又有了一个不小的误差!

• **输出误差**-另一个误差的产生取决于定时器定时完成的确切时间，PLC 完成程序执行然后更新输出的时间. 这是因为虽然定时器在程序执行期间已定时完成，但是 PLC 必须首先执行完余下部分的程序，才去更新输出.

下图所示为最坏情况下的输入误差. 可以看出最大的输入误差为 1 完整的扫描时间+1 程序执行时间. 注意，程序执行时间会因程序不同而不同.(取决于程序中的指令数.)



下图所示为最坏情况下的输出误差. 从中可以看出最大的输出误差为 1 完整的扫描时间.



基于上面的分析，我们可以得出最坏情况下的定时器总误差为：
1 扫描时间 + 1 程序执行时间 + 1 扫描时间= 2 扫描时间 + 1 程序执行时间.

这到底意味着什么呢？这意味着虽然大多数的生产商目前均提供增量为 1ms 的定时器，但是它们实际上不能用于少于几个毫秒的定时．这是假设我们的扫描时间为 1ms．如果我们的扫描时间为 5ms，那么最好不用少于 15ms 的定时器．即便如此，我们所预计的误差也会产生．既然我们能预计误差的大小，那么我们就知道我们的应用程序能否容忍此误差的存在．在大多数的应用中，误差都是可以忽略不计的，但是在一些高速或要求非常精确的应用中误差就变得不容忽视.

我们也应该注意上面的误差仅仅指的是"软件误差"．当然还有硬件输入误差和硬件输出误差.

硬件输入误差是由 PLC 扫描输入时确切认知其输入为 ON 所花的时间引起的，典型值为 10ms．这是因为许多 PLC 要求一个输入应该为 ON 几个扫描周期以后，才确认其物理状态为 ON．（这是为了减少噪声或瞬动输入的影响）

硬件输出误差的产生，是由于从 PLC 告诉它的输出变为 ON，到其确实变为 ON 要花费一定的时间．典型的晶体管要花费大约 0.5ms 的时间，而机械式的继电器要花费大约 10ms 的时间.

误差是不是越来越大了？如果对于我们的应用来说，它已经变得非常大，那么就应该考虑使用外部"硬件"定时器了.