PLC

梯形图设计方法与应用实例

PLC TIXINGTU SHEJI FANGFA YU YINGYONG SHILI

← 张浩凤◎著 →

机械工业出版社
CHINA MACHINE PRESS

PLC 梯形图设计方法与应用实例

张浩风　著

机 械 工 业 出 版 社

梯形图是PLC的最基本、最常用的编程工具，是广大电气工程师们熟知并广泛接受和使用的。本书的设计方法突破了继电器-接触器硬件控制回路这种单一的设计方式，使用真值表法、BASIC语言程序设计方法、时序图法、状态转换图法来进行PLC梯形图设计，并且给出了使用这些方法的工程应用实例。书中的部分实例，用了不同的设计方法来实现，以便读者进行比较。

本书适用于从事PLC控制系统的梯形图编程、调试或初学编程工作的工程技术人员和电气自动化专业的在校师生。

图书在版编目(CIP)数据

PLC梯形图设计方法与应用实例/张浩风著. —北京：机械工业出版社，2008.7
ISBN 978-7-111-24320-5

Ⅰ.P… Ⅱ.张… Ⅲ.可编程序控制器 Ⅳ.TM571.6

中国版本图书馆CIP数据核字（2008）第088402号

机械工业出版社(北京市百万庄大街22号 邮政编码100037)
责任编辑：李振标 版式设计：霍永明 责任校对：张晓蓉
封面设计：陈 沛 责任印制：李 妍
北京蓝海印刷有限公司印刷
2008年8月第1版第1次印刷
184mm×260mm · 16.5印张 · 409千字
0001—4000册
标准书号:ISBN 978-7-111-24320-5
定价：38.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：(010)68326294
购书热线电话：(010)88379639 88379641 88379643
编辑热线电话：(010)88379765
封面无防伪标均为盗版

前　言

PLC（Programmable Logic Controller）过去被称作为可编程逻辑控制器，是一种工业控制用计算机控制系统，用来替代过去传统的继电器硬件控制回路。随着计算机技术的不断发展，PLC的软硬件的功能也越来越强大，现在已经不仅仅限于继电器式的开关量逻辑控制，而且引进了许多过程控制中的模拟量控制及其他许多功能，因而目前一般称之为可编程控制器，即PC（Programmable Controller）。虽然如此，但考虑到长期以来的习惯叫法，我们现在依然习惯于使用PLC一词，但它实际上代表的是可编程控制器而不是可编程逻辑控制器。现在，PLC已经广泛应用到各个行业的生产过程控制中。

梯形图是PLC的最基本、最常用的编程工具，它直接来源于传统的继电器-接触器硬件控制回路，是为广大电气工程师们熟知并且广泛接受和使用的。但作为梯形图设计方法的问题，就目前所能见到的有关书籍、资料，一般也只是沿用了继电器-接触器硬件控制回路设计方式。而在本书中，则突破了的继电器-接触器硬件控制回路这种单一的设计方式，使用了真值表法、BASIC语言程序设计法、时序图法、状态转换图法来进行PLC梯形图设计，并且给出了使用这些方法的应用实例。本书中的一些实例是直接来源于实际工程或开发项目，并且按照实际项目的规格进行设计。书中也给出了若干控制要求相同的实例，在不同章节中用不同的方法实现，以方便进行比较。

书中使用的是美国Allen-Bradley公司（简称A-B公司）用于Logix 5000系列PLC的RSLogix 5000编程软件，所有实例均在此环境下进行编程并且调试通过。书中的方法和实例均采用了PLC梯形图最为常用的指令，因此可以几乎不需要改变就可以直接用于其他品牌系列的PLC。

在第1章中，全面深入地介绍了利用真值表方法来实现组合逻辑、时序逻辑类和基于这两者的计时器类的梯形图设计方法，并且给出许多实例来进行说明。用真值表来实现各种类型的梯形图设计，直接来源于数字电路相关的理论方法，用梯形图来实现只不过是从程序执行的角度稍加改造而已，已经过实践的检验。其中，还介绍了一个通用的可将真值表直接转化为梯形图的Microsoft Office Excel 2003环境下的VBA程序。本章介绍的真值表方法是一种非程序设计方法，它可以涵盖所有开关量控制的PLC梯形图设计问题。

在第2章中，介绍了利用BASIC语言程序设计的方法来实现梯形图的设计。本章利用梯形图实现了BASIC语言的各种常用语句，这样我们就可以用结构化的程序设计方法来进行设计，然后转换为梯形图。利用程序设计方法可以解决任何PLC梯形图设计问题。与第1章的真值表方法相比，是完全不同的方法。

在第3章中，介绍利用时序图来进行梯形图设计。时序图可以清楚地表明控制的各种关系，对于一个正确全面地反映控制要求的时序图，其用梯形图来实现的方法几乎是固定的。

本章中，使用了置位、复位语句及计时器等来实现通过时序图到梯形图的转换。其中，对于计时器，提出了利用计时器实现时间测量和动态改变计时器计时时间的方法，并给出具体的应用实例。另外，本章还提出了“利用计时器实现多任务调度”的方法，其中一个重要的应用场合是实现了 DCS（集散系统控制）策略的“组态”功能。

在第 4 章中，介绍了用状态转换图来进行梯形图的设计。这也是一种非程序的设计方法，只需根据控制要求划分工作状态，画出各个状态之间的各种关系，就可以用梯形图来实现，其方法是固定的。本章的实例均为前面章节中的实例，主要是为了方便进行比较。

在第 5 章中，介绍了三个实际项目的实例。这些实例综合应用了前面章节介绍的各种方法，它们可以直接用于生产过程控制，也可以使用其中的方法来完成同样类型问题的控制。其中，实例“电梯控制”和“反应塔排浆泵控制”主要利用了第 2 章中介绍的 BASIC 语言程序设计方法。实例“双交叉限幅燃烧控制”，则主要利用了第 3 章介绍的“利用计时器实现多任务调度”的方法，当然也使用了第 2 章中的 BASIC 语言程序设计方法，这个实例可直接用于实际项目。

虽然笔者对书中方法实例均进行了全面测试，但难免会有疏忽和不足，欢迎广大读者提出宝贵意见，谢谢！

作　者

2007 年 10 月

目　录

第 1 章

使用真值表进行 PLC 梯形图设计

真值表是在数字电路设计中经常使用的一种方法，它用“0”，“1”表示输入输出的状态，以表格的形式列出输入与输出之间的关系，是描述控制任务的最直接的一种方法。以实现“与”的逻辑关系为例，可列出如下“与”逻辑真值表 1-1：

表 1-1 “与”逻辑真值表

输　入		输出	输　入		输出
A	B	C	A	B	C
0	0	0	1	0	0
0	1	0	1	1	1

显然，“与”关系所可能出现的各种关系都通过真值表表现出来。它表达了 $C = A \cdot B$ 的逻辑关系。可见，由各种“与”“或”“非”等形成的逻辑关系，总是可以用真值表描述的。

在数字电路的基于门电路的设计中，首先是根据要实现的控制功能，列写出真值表，根据真值表就可以写出其逻辑表达式，由于数字电路设计一般以使用门电路最少为原则，因此，要将逻辑表达式化简成最简形式，之后就可以用“与”“或”“非”等门电路来实现。

PLC 所使用的梯形图来源于常用的继电器控制回路，它体现了非常明确的“与”“或”“非”等逻辑关系，当然也就可以用真值表来描述，例如下面的梯形图 1-1。

与数字电路不同的是，梯形图是以软件形式体现出来的，不是硬件实现的。这就意味着，梯形图作为软件程序的执行是需要时间的（当然数字电路的执行也是需要时间的，只不过是时间非常非常之短），我们称梯形图从上到下运行一次为一个扫描周期。也正是程序执行的特点，使得我们可以非常方便地实现时序逻辑类的控制，而不像数字电路那样对于时

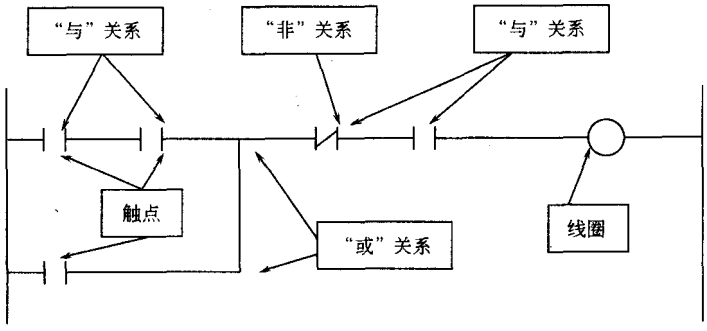


图 1-1 PLC 梯形图中的“与”“或”“非”的逻辑关系

序电路只能使用触发器。另外，逻辑表达式在梯形图中化简也不是必需的，因为梯形图设计不存在使用“门电路”最少的原则，作为一种程序它只有运算速度要求和内存限制。

因此，梯形图完全可以沿用数字电路使用真值表的方法来进行设计，根据控制要求可以用真值表来描述输入输出关系，然后转化为梯形图。

在本章中，根据输入和输出的关系，将使用真值表方法设计的梯形图类型分为以下几种：

- 如果输出仅与输入有关，我们称之为基于组合逻辑类的梯形图，与数字电路中的组合逻辑电路相对应；
- 如果输出与输入及输出的上一次输出状态有关，我们称之为基于时序逻辑类的梯形图，与数字电路中的时序逻辑电路相对应。
- 由于在 PLC 梯形图设计中经常会遇到计时问题，因此，便出现了建立在上述两种类型之上的基于计时器类的梯形图设计。

1.1 基于组合逻辑类的梯形图设计方法

组合逻辑是数字电路中的概念，它的特点是输出只与输入有关，输入输出都可有多个。例如，数字电路中的两位数值比较器，是一个典型的组合逻辑电路，下面以它为例说明真值表转化为梯形图的方法。

设输入为 A、B、C、D，输出为 Y，若 AB 大于 CD 则 Y 为 1，否则为 0。列出真值表 1-2 如下：

表 1-2 两位数值比较器真值表

序号	输入				输出	序号	输入				输出
	A	B	C	D	Y		A	B	C	D	Y
1	0	0	0	0	0	9	0	0	0	1	
2	0	0	0	1	0	10	1	0	0	1	1
3	0	0	1	0	0	11	1	0	1	0	0
4	0	0	1	1	0	12	1	0	1	1	0
5	0	1	0	0	1	13	1	1	0	0	1
6	0	1	0	1	0	14	1	1	0	1	1
7	0	1	1	0	0	15	1	1	1	0	1
8	0	1	1	1	0	16	1	1	1	1	0

按照输出为 1 的行中，输入为 1 的写原变量，输入为 0 写反变量的原则写出逻辑表达式如下：

第 6 行：$Y = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D}$

第 10 行：$Y = A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D}$

第 11 行：$Y = A \cdot \overline{B} \cdot \overline{C} \cdot D$

第 14 行：$Y = A \cdot B \cdot \overline{C} \cdot \overline{D}$

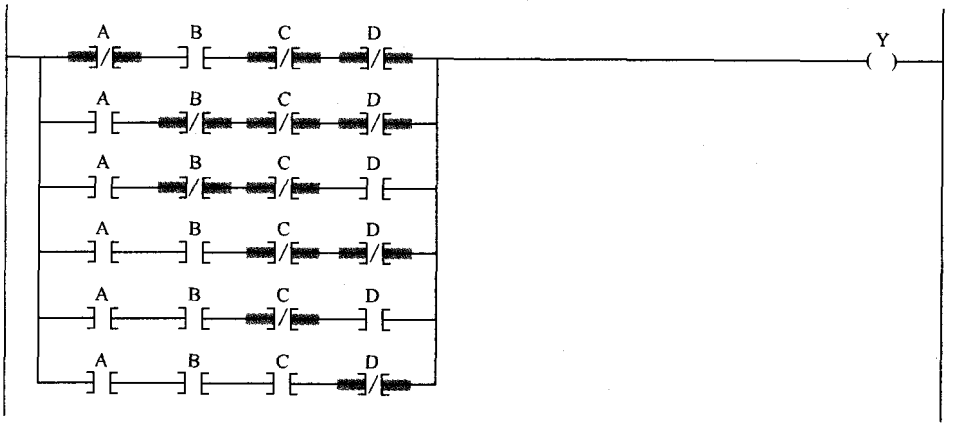
第 15 行：$Y = A \cdot B \cdot \overline{C} \cdot D$

第 16 行：$Y = A \cdot B \cdot C \cdot \overline{D}$

由于以上各行是或的关系，因此得到：

$$Y = \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot D + A \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot B \cdot \overline{C} \cdot D + A \cdot B \cdot C \cdot \overline{D}$$

转化成梯形图如下：



对于通过真值表得出的逻辑表达式可以进行化简，但为了使梯形图更加清晰易懂，也可以不进行化简，这样会更易于调试（由于进行化简的工作量也是比较大的，这样可以避免化简可能带来的错误。对于一些应用而言，化简后的表达式转化为梯形图时，会显得比较零乱没有规律，转换成梯形图时可能会出现难以查找的错误。对于有运行速度要求或有内存限制的系统，化简是必要的。化简方法可参考有关数字电路方面的书籍）。

实例 1 ~6 是基于组合逻辑的几个实例。

实例 1：电机故障报警一

控制要求：3 台电机同时工作，如有 1 台电机不工作，则报警。

输入：A、B、C 分别代表 3 台电机的运行输入信号。

输出：Y 代表有 1 台电机不工作的故障输出信号。

设计方法：根据控制要求，列出真值表 1-3 如下：

表 1-3　电机故障报警一真值表

序号	输　入			输出
	A	B	C	Y
1	0	1	1	1
2	1	0	1	1
3	1	1	0	1

由于产生电机故障报警只有上面3种情况，故输出Y为0的真值表可不必列出，因为输出为0的行是不会出现在逻辑表达式中的。

按照输出为1的行中，输入为1的写原变量，输入为0写反变量的原则写出逻辑表达式如下：

第2行：$Y = \overline{A} \cdot B \cdot C$

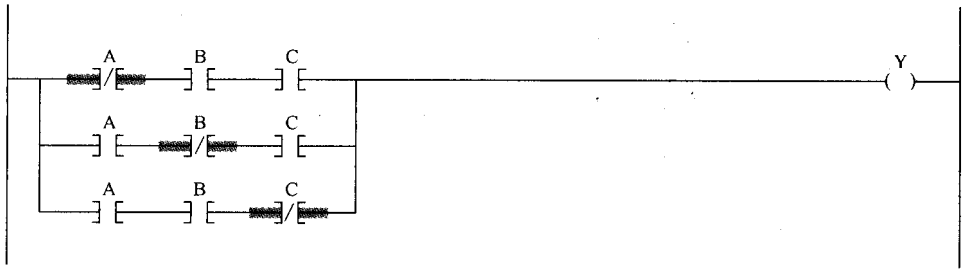
第3行：$Y = A \cdot \overline{B} \cdot C$

第4行：$Y = A \cdot B \cdot \overline{C}$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y = \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot \overline{C}$$

梯形图的实现如下：



实例2：电机故障报警二

控制要求：3台电机同时工作，如有2台电机不工作，则报警。

输入：A、B、C分别代表3台电机的运行输入信号。

输出：Y代表有2台电机不工作的故障输出信号。

设计方法：根据控制要求，列出真值表1-4：

表1-4 电机故障报警二真值表

序号	输入			输出
	A	B	C	Y
1	0	0	1	1
2	0	1	0	1
3	1	0	0	1

由于产生电机故障报警只有上面3种情况，故输出Y为0的真值表可不必列出，因为输出为0的行是不会出现在逻辑表达式中的。

按照输出为1的行中，输入为1的写原变量，输入为0写反变量的原则写出逻辑表达式如下：

第2行：$Y = \overline{A} \cdot \overline{B} \cdot C$

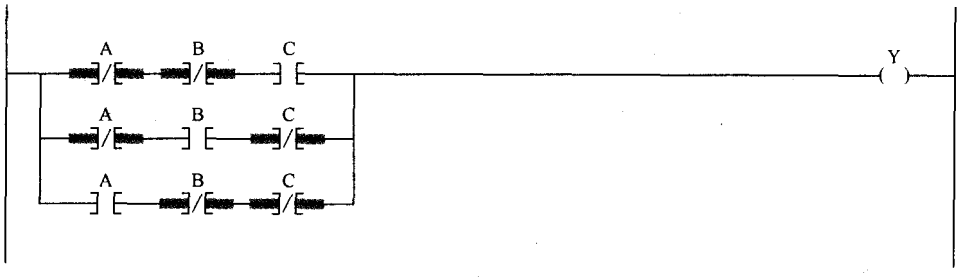
第3行：$Y = \overline{A} \cdot B \cdot \overline{C}$

第4行：$Y = A \cdot \overline{B} \cdot \overline{C}$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C}$$

梯形图的实现如下：



实例 3：三控开关

控制要求：3 个开关位于 3 个不同的位置，要求在任意位置均可控制同一盏灯的亮灭，即若灯是打开的，任何一个开关均可将其关闭；若灯是关闭的，任何一个开关均可将其打开。

输入：A、B、C 分别代表 3 个自锁型按钮，若按下，则为 1；按下弹开，则为 0。

输出：Y 代表控制灯的开关输出。

设计方法：根据控制要求，不难看出，在输入全为 0 时，即第一次运行前，输出 Y 应为 0，灯不亮；当有任一按钮按下时，灯应该亮，由真值表第 3、7、9 行实现；当有任一按钮按下后，再按一下弹开时，则返回真值表第 2 行状态，灯关闭；若已经有 2 个按钮按下，则应该是灯关闭，由真值表第 4、6、8 行实现，若再按其中一个，按钮弹开时，则返回真值表第 3、7、9 行状态，灯应该亮；若 3 个按钮都按下，则灯应该打开，由真值表第 5 行实现，若其中任一按钮按下弹开时，则返回真值表第 4、6、8 行状态，灯关闭。真值表 1-5：

表 1-5　3 控开关的梯形图实现真值表

序号	输　入			输出	序号	输　入			输出
	A	B	C	Y		A	B	C	Y
1	0	0	0	0	5	1	0	1	0
2	1	0	0	1	6	0	1	0	1
3	1	1	0	0	7	0	1	1	0
4	1	1	1	1	8	0	0	1	1

按照输出为 1 的行中，输入为 1 的写原变量，输入为 0 写反变量的原则写出逻辑表达式如下：

第 3 行：　　$Y = A \cdot \overline{B} \cdot \overline{C}$

第 5 行：　　$Y = A \cdot B \cdot C$

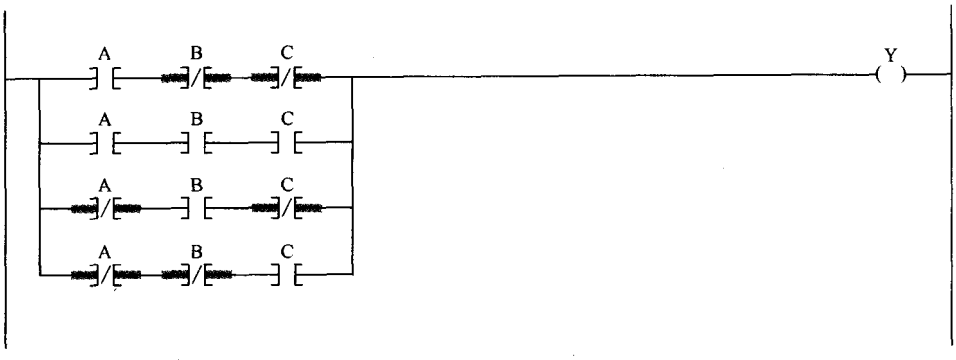
第 7 行：　　$Y = \overline{A} \cdot B \cdot \overline{C}$

第 9 行：　　$Y = \overline{A} \cdot \overline{B} \cdot C$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y = A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C$$

梯形图的实现如下：



实例 4：4 选 1 数据选择器

控制要求： 用梯形图实现 4 选 1 数据选择器功能。

输入： A、B 代表数据选择器地址，D0、D1、D2、D3 代表数据输入。

输出： Y 代表数据选择器输出。

设计方法： 根据数据选择器的特性可知，在输入 A、B 为 00 时，输出 Y = D0；A、B 为 01 时，输出 Y = D1；A、B 为 10 时，输出 Y = D2；A、B 为 11 时，输出 Y = D3。列出真值表 1-6（X 代表 1，0 任意状态，可称之为无关项）：

表 1-6 4 选 1 数据选择器真值表

序号	输入						输出
	A	B	D0	D1	D2	D3	Y
1	0	0	0	X	X	X	0
2	0	0	1	X	X	X	1
3	0	1	X	0	X	0	0
4	0	1	X	1	X	X	1
5	1	0	X	X	0	X	0
6	1	0	X	X	1	X	1
7	1	1	X	X	X	0	0
8	1	1	X	X	X	1	1

显然，在地址选中的情况下，相应数据输入为 0 时，输出 Y = 0；数据输入为 1 时，输出 Y = 1。

按照输出为 1 的行中，输入为 1 的写原变量，输入为 0 写反变量，输入 X 为省略的原则写出逻辑表达式如下：

第 3 行：$Y = \overline{A} \cdot \overline{B} \cdot D0$

第 5 行：$Y = \overline{A} \cdot B \cdot D1$

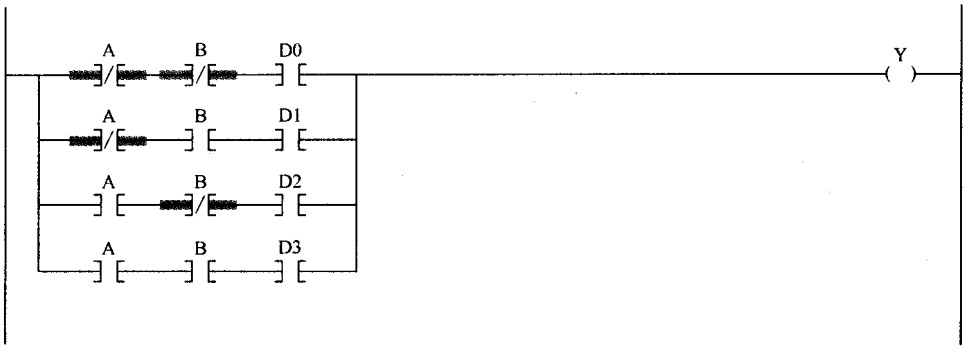
第 7 行：$Y = A \cdot \overline{B} \cdot D2$

第 9 行：$Y = A \cdot B \cdot D3$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y = \overline{A} \cdot \overline{B} \cdot D0 + \overline{A} \cdot B \cdot D1 + A \cdot \overline{B} \cdot D2 + A \cdot B \cdot D3$$

梯形图的实现如下：



实例 5：2-4 译码器

控制要求：用梯形图实现 2-4 译码器功能。

输入：A、B 代表 2-4 译码器地址输入。

输出：Y0、Y1、Y2、Y3 代表译码器输出。

设计方法：根据 2-4 译码器的特性，可知在输入 A、B 为 00 时，输出 Y0 = 0，Y1 = 1，Y2 = 1，Y3 = 1；A、B 为 01 时，输出 Y0 = 1，Y1 = 0，Y2 = 1，Y3 = 1；A、B 为 10 时，输出 Y0 = 1，Y1 = 0，Y2 = 1，Y3 = 1；A、B 为 11 时，输出 Y0 = 1，Y1 = 1，Y2 = 1，Y3 = 0。列出真值表 1-7（X 代表 1，0 任意状态，可称之为无关项）：

表 1-7　2-4 译码器真值表

序号	输　入		输　出			
	A	B	Y0	Y1	Y2	Y3
1	0	0	0	1	1	1
2	0	1	1	0	1	1
3	1	0	1	1	0	1
4	1	1	1	1	1	0

按照输出为 1 的行中，输入为 1 的写原变量，输入为 0 写反变量的原则写出逻辑表达式如下：

Y0 输出：

第 3 行：　　$Y0 = \overline{A} \cdot B$

第 4 行：　　$Y0 = A \cdot \overline{B}$

第 5 行：　　$Y0 = A \cdot B$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y0 = \overline{A} \cdot B + A \cdot \overline{B} + A \cdot B$$

化简后可得到：

$$Y0 = A + B$$

Y1 输出：

第 2 行：　　$Y1 = \overline{A} \cdot \overline{B}$

第 4 行：　　$Y1 = A \cdot \overline{B}$

第 5 行：$Y1 = A \cdot B$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y1 = \overline{A} \cdot \overline{B} + A \cdot \overline{B} + A \cdot B$$

化简后可得到：

$$Y1 = A + \overline{B}$$

Y2 输出：

第 2 行：$Y2 = \overline{A} \cdot \overline{B}$

第 3 行：$Y2 = \overline{A} \cdot B$

第 5 行：$Y2 = A \cdot B$

由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y2 = \overline{A} \cdot \overline{B} + \overline{A} \cdot B + A \cdot B$$

化简后可得到：

$$Y2 = \overline{A} + B$$

Y3 输出：

第 2 行：$Y3 = \overline{A} \cdot \overline{B}$

第 3 行：$Y3 = \overline{A} \cdot B$

第 4 行：$Y3 = A \cdot \overline{B}$

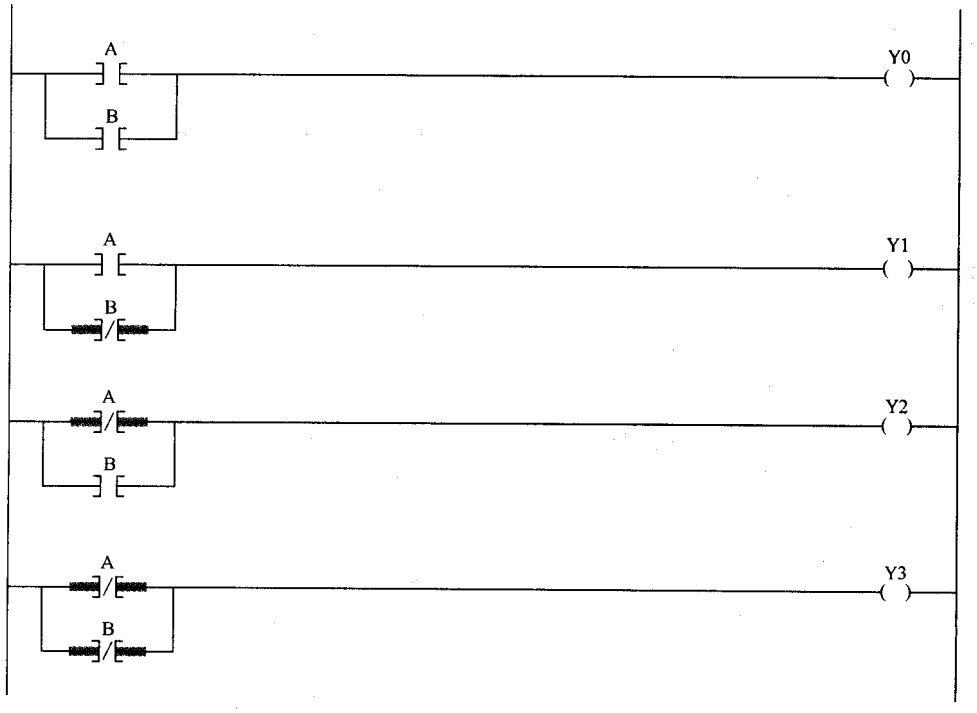
由于以上各行是“或”的关系，因此得到逻辑表达式：

$$Y3 = \overline{A} \cdot \overline{B} + \overline{A} \cdot B + A \cdot \overline{B}$$

化简后可得到：

$$Y3 = \overline{A} + \overline{B}$$

梯形图的实现如下：



1.2 基于时序逻辑类的梯形图设计方法

时序类逻辑的特点是，输出与输入及输出的上一次输出状态有关，不像组合逻辑只与输入有关。这个特点在梯形图中是很容易实现的，由于梯形图是从上往下，从左往右执行的，每运行一次称为一个扫描周期，输出以触点的形式表现出来的则一定是上一次扫描周期的运行结果（梯形图首次运行时，输出变量的初始值默认为0，除非在运行前已经赋值为其他值。RSLogix5000 可以将变量在运行前赋值），输出以线圈的形式表现出来的则一定是本次扫描将会有的输出。因此，输出是根据上一次输出即触点的结果进行下一次输出。

根据数字电路时序电路的有关概念，我们称在梯形图中输出以触点的形式出现的为输出的“现态”，输出线圈则称为“次态”，这样就可以实现时序类梯形图设计。

时序类逻辑转化为梯形图必须遵守的原则：

- 对于具有单输出的时序类，从真值表转换为梯形图时，可沿用组合逻辑类的方法，而且可以进行逻辑表达式的化简。
- 对于具有多输出的时序类，则要注意一定是在“现态”有关变量运行完之后再赋值给“次态”输出线圈，为此需要引入中间变量，即将“现态”有关变量运行结果赋值给中间变量，由中间变量再赋值给“次态”输出线圈，而且每个“次态”输出线圈在梯形图中只能出现一次，只有这样才能体现出“次态”是输入与“现态”的函数。
- 如果有多个真值表对相同的输出线圈进行控制，则需以“或”的形式并联接到输出线圈（只能出现一次）的赋值回路中。
- 如果要进行逻辑表达式的化简，则只能以中间变量为输出进行，否则有可能出现“次态”以“现态”的形式参与运行，从而会导致错误。

这些原则的应用情况可参看下面实例。

数字电路时序逻辑电路具有状态保持功能，按照触发保持的方式可分为电平触发及上升沿触发（或下降沿），例如，RS 触发器可以看做是电平触发，而 D 触发器则是上升沿（或下降沿）触发。本节将介绍这两种不同方式梯形图的实现。

1.2.1 时序逻辑电平触发类的梯形图设计方法

所谓电平触发方式和梯形图本身并没什么关系，只是说明这样一种情况，在某输入为1（或0）时，输出根据相应的输入会有一个输出状态，这个输出状态在某输入变为0（或1）时，保持不变。以 RS 触发器为例来说明电平触发类的设计方法。RS 触发器在复位端 R = 1 的时候，输出为0，并且在 R = 0 时保持输出为0；在置位端 S = 1 时，输出为1，并且在 S = 0 时保持输出为1。在数字电路中要求 R、S 不能同时置位，否则输出状态不定。但在梯形图中不存在这个问题，因为我们可以定义 R、S 同时为1时的输出状态，例如，输出为0，即复位优先；输出为1，即置位优先；还可定义为保持原状态不变。下面列出复位 R 优先的 RS 触发器真值表 1-8：

表 1-8 复位 R 优先的 RS 触发器真值表

输入			输出	输入			输出
R	S	Q	Q	R	S	Q	Q
0	0	0	0	1	0	X	0
0	0	1	1	1	1	X	0
0	1	X	1				

输入中的 Q 代表“现态”，在梯形图中将以触点的形式出现；输出中的 Q 代表“次态”，以线圈的形式出现。

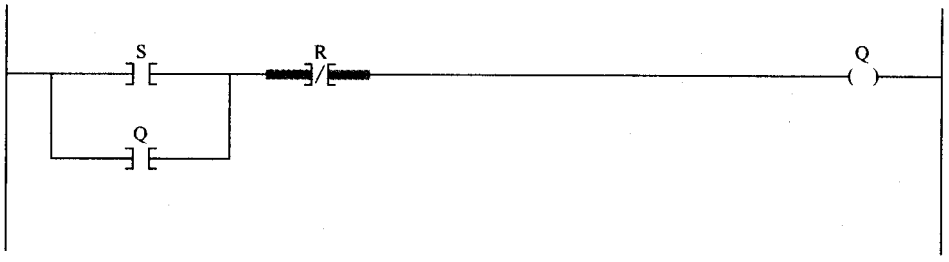
按照上一节介绍的方法写出逻辑表达式如下：

$$Q = \overline{R} \cdot \overline{S} \cdot Q + \overline{R} \cdot S$$

化简后得到：

$$Q = \overline{R} \cdot (S + Q)$$

转化成梯形图如下：



不难看出复位优先的 RS 触发器梯形图的实现和电机控制回路中常用的“启动-保持-停止”电路完全一样。

同样方法我们可以得到置位优先的 RS 触发器梯形图的实现，真值表 1-9：

表 1-9 置位 S 优先的 RS 触发器真值表

输入			输出	输入			输出
R	S	Q	Q	R	S	Q	Q
0	0	0	0	1	0	X	0
0	0	1	1	1	1	X	1
0	1	X	1				

按照上一节介绍的方法写出逻辑表达式如下：

$$Q = \overline{R} \cdot \overline{S} \cdot Q + \overline{R} \cdot S + R \cdot S$$

化简后得到：

$$Q = S + \overline{R} \cdot Q$$

转化成梯形图如下：