

前言、PLC 的发展背景及其功能概述

PLC (Programmable Logic Controller) , 乃是一种电子装置 , 早期称为顺序控制器 “ Sequence Controller ,” 1978 NEMA(National Electrical Manufacture Association) 美国国家电气协会正式命名为 Programmable Logic Controller , PLC) , 其定义为一种电子装置 , 主要将外部的输入装置如 : 按键、感应器、开关及脉冲等的状态读取后 , 依据这些输入信号的状态或数值并根据内部储存预先编写的程序 , 以微处理机执行逻辑、顺序、定时、计数及算式运算 , 产生相对应的输出信号到输出装置如 : 继电器 (Relay) 的开关、电磁阀及电机驱动器 , 控制机械或程序的操作 , 达到机械控制自动化或加工程序的目的。并藉由其外围的装置 (个人计算机 / 程序书写器) 轻易地编辑 / 修改程序及监控装置状态 , 进行现场程序的维护及试机调整。而普遍使用于 PLC 程序设计的语言 , 即是梯形图 (Ladder Diagram) 程序语言。

而随着电子科技的发展及产业应用的需要 , PLC 的功能也日益强大 , 例如位置控制及网络功能等 , 输出/入信号也包含了 DI (Digital Input)、AI (Analog Input)、PI (Pulse Input) 及 NI (Numerical Input) , DO (Digital Output) 、AO (Analog Output) 、PO (Pulse Output) 及 NO (Numerical Output) , 因此 PLC 在未来的工业控制中 , 仍将扮演举足轻重的角色。

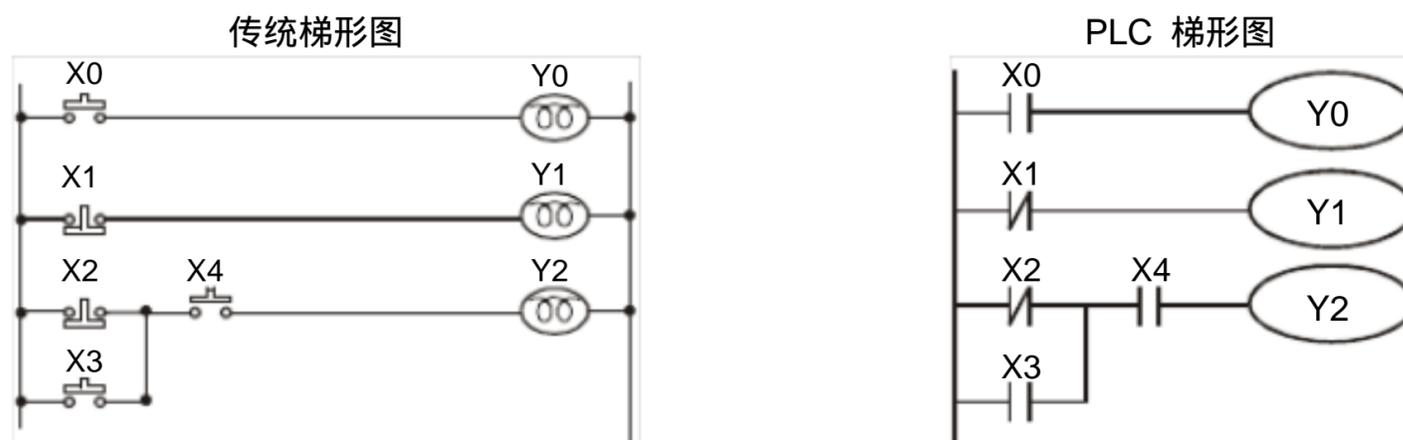
1.1 梯形图工作原理

梯形图为二次世界大战期间所发展出来的自动控制图形语言 , 是历史最久、使用最广的自动控制语言 , 最初只有 A (常开) 接点、B (常闭) 接点、输出线圈、定时器、计数器等基本机构装置 (今日仍在使用的配电盘即是) , 直到可编程器 PLC 出现后 , 梯形图之中可表示的装置 , 除上述外 , 另增加了诸如微分接点、保持线圈等装置以及传统配电盘无法达成的应用指令 , 如加、减、乘及除等数值运算功能。

无论传统梯形图或 PLC 梯形图其工作原理均相同 , 只是在符号表示上传统梯形图比较接近实体的符号表示 , 而 PLC 则采用较简明且易于计算机或报表上表示的符号表示。在梯形图逻辑方面可分为组合逻辑和顺序逻辑两种 , 分述如下 :

1. 组合逻辑 :

分别以传统梯形图及 PLC 梯形图表示组合逻辑的范例。



行 1 : 使用一常开开关 X0(NO : Normally Open)亦即一般所谓的 “ A” 开关或接点。其特性是在平常 (未按下) 时 , 其接点为开路 (Off) 状态 , 故 Y0 不导通 , 而在开关动作 (按下按钮) 时 , 其接点变为导通 (On) , 故 Y0 导通。

行 2 : 使用一常闭开关 X1 (NC : Normally Close) 亦即一般所称的 “ B” 开关或接点 , 其特性是在平常时 , 其接点为导通 , 故 Y1 导通 , 而在开关动作时 , 其接点反而变成开路 , 故 Y1 不导通。

行 3：为一个以上输入装置的组合逻辑输出的应用，其输出 Y2 只有在 X2 不动作或 X3 动作且 X4 为动作时才会导通。

2. 顺序逻辑：

顺序逻辑为具有反馈结构的回路，亦即将回路输出结果送回当输入条件，如此在相同输入条件下，会因前次状态或动作顺序的不同，而得到不同的输出结果。

分别以传统梯形图及 PLC 梯形图表示顺序逻辑的范例。



在此回路刚接上电源时，虽 X6 开关为 On，但 X5 开关为 Off，故 Y3 不动作。在启动开关 X5 按下后，Y3 动作，一旦 Y3 动作后，即使放开启动开关（X5 变成 Off）Y3 因为自身的接点反馈而仍可继续保持动作（此即为自我保持回路），其动作可以下表表示：

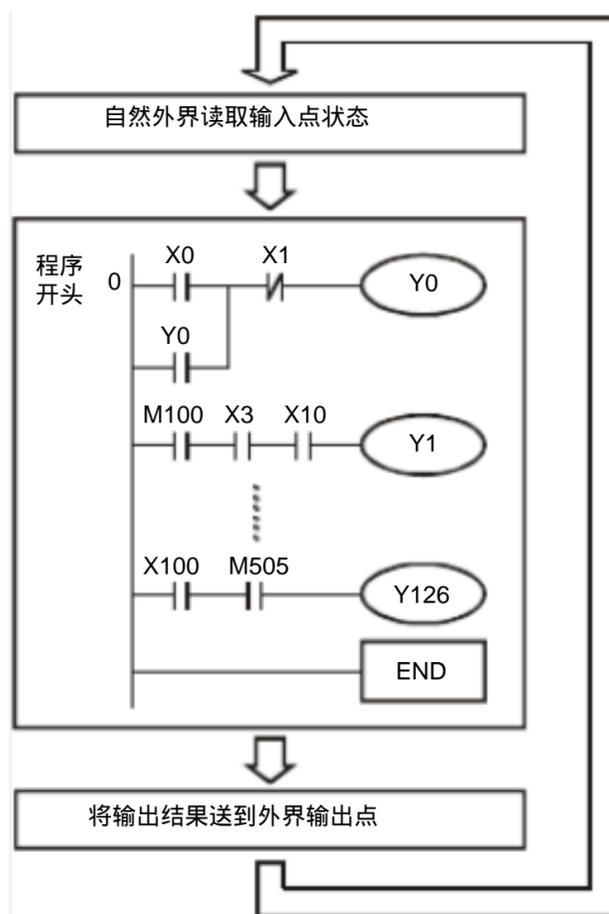
动作顺序	装置状态	X5 开关	X6 开关	Y3 状态
1		不动作	不动作	Off
2		动作	不动作	On
3		不动作	不动作	On
4		不动作	动作	Off
5		不动作	不动作	Off

由上表可知在不同顺序下，虽然输入状态完全一致，其输出结果也可能不一样，如表中的动作顺序 1 和 3 其 X5 和 X6 开关均为不动作，在状态 1 的条件下 Y3 为 Off，但状态 3 时 Y3 却为 On，此种 Y3 输出状态送回当输入（即所谓的反馈）而使回路具有顺序控制效果是梯形图回路的主要特性。在本节范例中仅列举 A、B 接点和输出线圈作说明，其它装置的用法和此相同，请参考第 3 章“基本指令”。

1.2 传统梯形图及 PLC 梯形图的差异

虽然传统梯形图和 PLC 梯形图的工作原理是完全一致的，但实际上 PLC 仅是利用微电脑（Microcomputer），来仿真传统梯形图的动作，亦即利用扫描的方式逐一地查看所有输入装置及输出线圈的状态，再将此等状态依梯形图的组态逻辑作演算和传统梯形图一样的输出结果，但因 Microcomputer 只有一个，只能逐一地查看梯形图程序，并依该程序及输入/出状态演算输出结果，再将结果送到输出接口，然后又重新读取输入状态演算输出，如此周而复始地循环执行上述动作，此一完整的循环动作所费的时间称之为扫描周期，其时间会随着程序的增大而加长，此扫描周期将造成 PLC 从输入检测到输出反应的延迟，延迟时间愈长对控制所造成的误差愈大，甚至造成无法胜任控制要求的情况，此时就必须选用扫描速度更快的 PLC，因此 PLC 的扫描速度是 PLC 的重要规格，随着微电脑及 ASIC（特定用途 IC）技术的发展，现今的 PLC 在扫描速度上均有极大的改善，下图为 PLC 的梯形图程序扫描的示意图。

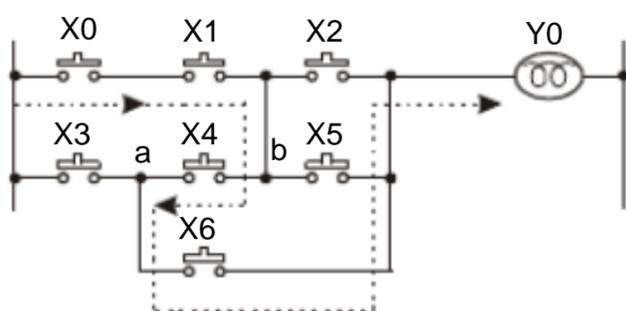
依梯形图组态演算出输出结果
(尚未送到外界输出点, 但内部装置会实时输出)



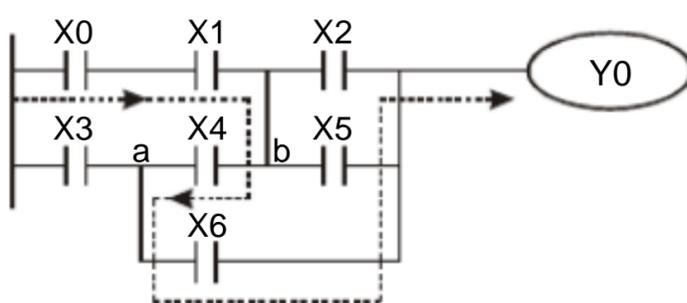
周而复始的执行

除上述扫描周期差异外, PLC 梯形图与传统梯形图尚有如下的“逆向回流”的差异, 如下图传统梯形图所示图中, 若 X0, X1, X4, X6 为导通, 其它为不导通, 在传统的梯形图回路上输出 Y0 会如虚线所示形成回路而为 On。但在 PLC 梯形图中, 因演算梯形图程序系由上而下, 由左而右地扫描。在同样输入条件下, 以梯形图编辑工具 (WPLSoft) 会检查出梯形图错误。

传统梯形图的逆向回流:



PLC 梯形图的逆向回流:



检查出梯形图形第三行错误

1.3 梯形图编辑说明

梯形图为广泛应用在自动控制的一种图形语言, 这是沿用电气控制电路的符号所组合而成的一种图形, 透过梯形图编辑器画好梯形图形后, PLC 的程序设计也就完成, 以图形表示控制的流程较为直观, 易为熟悉电气控制电路的技术人员所接受。在梯形图形很多基本符号及动作都是根据在传统自动控制配电盘中常见的机电装置如按钮、开关、继电器 (Relay)、定时器 (Timer) 及计数器 (Counter) 等等。

PLC 的内部装置: PLC 内部装置的种类及数量随各厂牌产品而不同。内部装置虽然沿用了传统电气控制电路中的继电器、线圈及接点等名称, 但 PLC 内部并不存在这些实际物理装置, 它对应的只是 PLC 内部存储器的一个基本单元 (一个位, bit), 若该位为 1 表示该线圈得电, 该位为 0 表示线圈不得电, 使用常开接点 (Normal Open, NO 或 A 接点) 即直接读取该对应位的值, 若使用常闭接点 (Normal Close, NC 或 B

接点)则取该对应位值的反相。多个继电器将占有多个位 (bit), 8 个位, 组成一个字节 (或称为一个字节, byte), 二个字节, 称为一个字 (word), 两个字, 组合成双字 (double word)。当多个继电器一并处理时 (如加 / 减法、移位等) 则可使用字节、字或双字, 且 PLC 内部的另两种装置: 定时器及计数器, 不仅有线圈, 而且还有计时值及计数值, 因此还要进行一些数值的处理, 这些数值多属于字节、字或双字的形式。

由以上所述, 各种内部装置, 在 PLC 内部的数值储存区, 各自占有一定数量的储存单元, 当使用这些装置, 实际上就是对相应的储存内容以位或字节或字的形式进行读取。

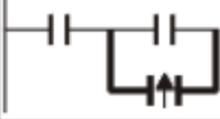
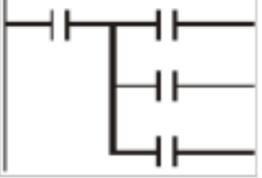
PLC 的基本内部装置介绍: (详细说明请参考第 2 章 DVP- PLC 各种装置功能)

装置种类	功能说明
输入继电器 (Input Relay)	<p>输入继电器是 PLC 与外部输入点 (用来与外部输入开关连接并接受外部输入信号的端子) 对应的内部存储器储存基本单元。它由外部送来的输入信号驱动, 使它为 0 或 1。用程序设计的方法不能改变输入继电器的状态, 即不能对输入继电器对应的基本单元改写, 亦无法由 HPP/WPLSoft 作强行 On / Off 动作 (SA/SX/SC/EH/EH2/SV 系列主机可仿真输入继电器 X 作强行 On/Off 的动作, 但此时外部输入点状态更新动作关闭, 亦即外部输入信号的状态不会被读入至 PLC 内部相对的装置内存, 只限主机的输入点, 扩展的输入点仍依正常模式动作)。它的接点 (A、B 接点) 可无限制地多次使用。无输入信号对应的输入继电器只能空着, 不能移作它用。</p> <p>装置表示: X0, X1, ... X7, X10, X11, ... 装置符号以 X 表示, 顺序以 8 进制编号。在主机及扩展上均有输入点编号的标示。</p>
输出继电器 (Output Relay)	<p>输出继电器是 PLC 与外部输出点 (用来与外部负载作连接) 对应的内部存储器储存基本单元。它可以由输入继电器接点、内部其它装置的接点以及它自身的接点驱动。它使用一个常开接点接通外部负载, 其接点也像输入接点一样可无限制地多次使用。无输出对应的输出继电器, 它是空着的, 如果需要, 它可以当作内部继电器使用。</p> <p>装置表示: Y0, Y1, ... Y7, Y10, Y11, ... 装置符号以 Y 表示, 顺序以 8 进制编号。在主机及扩展上均有输出点编号的标示。</p>
内部辅助继电器 (Internal Relay)	<p>内部辅助继电器与外部没有直接联系, 它是 PLC 内部的一种辅助继电器, 其功能与电气控制电路中的辅助 (中间) 继电器一样, 每个辅助继电器也对应着内存的一基本单元, 它可由输入继电器接点、输出继电器接点以及其它内部装置的接点驱动, 它自己的接点也可以无限制地多次使用。内部辅助继电器无对外输出, 要输出时请通过输出点。</p> <p>装置表示: M0, M1, ..., M4095, 装置符号以 M 表示, 顺序以 10 进制编号。</p>
步进点 (Step)	<p>DVP PLC 提供一种属于步进动作的控制程序输入方式, 利用指令 STL 控制步进点 S 的转移, 便可很容易写出控制程序。如果程序中完全没有使用到步进程序时, 步进点 S 亦可被当成内部辅助继电器 M 来使用, 也可当成警报点使用。</p> <p>装置表示: S0, S1, ... S1023, 装置符号以 S 表示, 顺序以 10 进制编号。</p>

装置种类	功能说明
定时器 (Timer)	<p>定时器用来完成定时的控制。定时器含有线圈、接点及定时值寄存器，当线圈受电，等到达预定时间，它的接点便动作（A 接点闭合，B 接点开路），定时器的定时值由设定值给定。每种定时器都有规定的时钟周期（定时单位：1ms/10ms/100ms）。一旦线圈断电，则接点不动作（A 接点开路，B 接点闭合），原定时值归零。</p> <p>装置表示：T0, T1, ..., T255，装置符号以 T 表示，顺序以 10 进制编号。不同的编号范围，对应不同的时钟周期。</p>
计数器 (Counter)	<p>计数器用来实现计数操作。使用计数器要事先给定计数的设定值（即要计数的脉冲数）。计数器含有线圈、接点及计数储存器，当线圈由 Off On，即视为该计数器有一脉冲输入，其计数值加一，有 16 位及 32 位及高速用计数器可供使用者选用。</p> <p>装置表示：C0, C1, ..., C255，装置符号以 C 表示，顺序以 10 进制编号。</p>
数据寄存器 (Data register)	<p>PLC 在进行各类顺序控制及定时值及计数值有关控制时，常常要作数据处理和数值运算，而数据寄存器就是专门用于储存数据或各类参数。每个数据寄存器内有 16 位二进制数值，即存有一个字，处理双字用相邻编号的两个数据寄存器。</p> <p>装置表示：D0, D1, ..., D9999，装置符号以 D 表示，顺序以 10 进制编号。</p>
文件寄存器 (File register)	<p>PLC 数据处理和数值运算所需的数据寄存器不足时，可利用文件寄存器来储存数据或各类参数。每个文件寄存器内为 16 位，即存有一个字，处理双字用相邻编号的两个文件寄存器。文件寄存器 SA/SX/SC 系列机种一共有 1,600 个，EH/EH2/SV 系列机种一共有 10,000 个，文件寄存器并没有实际的装置编号，因此需透过指令 API 148 MEMR、API 149 MEMW 或是透过周边装置 HPP02 及 WPLSoft 来执行文件寄存器的读写功能。</p> <p>装置表示：K0~K9,999，无装置符号，顺序以 10 进制编号。</p>
变址寄存器 (Index register)	<p>E、F 与一般的数据寄存器一样的都是 16 位的数据寄存器，它可以自由的被写入及读出，可用于字装置、位装置及常量来作间接寻址功能。</p> <p>装置表示：E0~E7、F0~F7，装置符号以 E、F 表示，顺序以 10 进制编号。</p>

梯形图组成图形及说明：

梯形图形结构	指令解说	指令	使用装置
--------	------	----	------

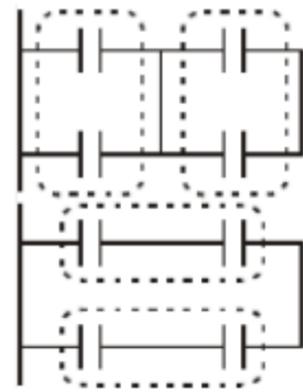
	常开开关，A 接点	LD	X、Y、M、S、T、C
	常闭开关，B 接点	LDI	X、Y、M、S、T、C
	串接常开	AND	X、Y、M、S、T、C
梯形图形结构	指令解说	指令	使用装置
	并联常开	OR	X、Y、M、S、T、C
	并联常闭	ORI	X、Y、M、S、T、C
	上升沿触发开关	LDP	X、Y、M、S、T、C
	下降沿触发开关	LDF	X、Y、M、S、T、C
	上升沿触发串接	ANDP	X、Y、M、S、T、C
	下降沿触发串接	ANDF	X、Y、M、S、T、C
	上升沿触发并联	ORP	X、Y、M、S、T、C
	下降沿触发并联	ORF	X、Y、M、S、T、C
	区块串接	ANB	无
	区块并联	ORB	无
	多重输出	MPS MRD MPP	无
	线圈驱动输出指令	OUT	Y、M、S
	步进梯形	STL	S
	基本指令、应用指令	应用指令	请参考第 3 章的基本指令 (RST/SET 及 CNT/TMR) 说明及第 5~10 章应用指令
	反向逻辑	INV	无

区块：所谓的区块是指两个以上的装置做串接或并联的运算组合而形成的梯形图形，依其运算性质可产生并联区块及串联区块。

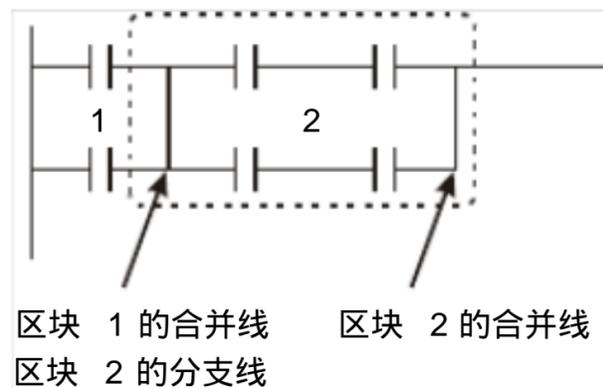
串联区块：



并联区块：

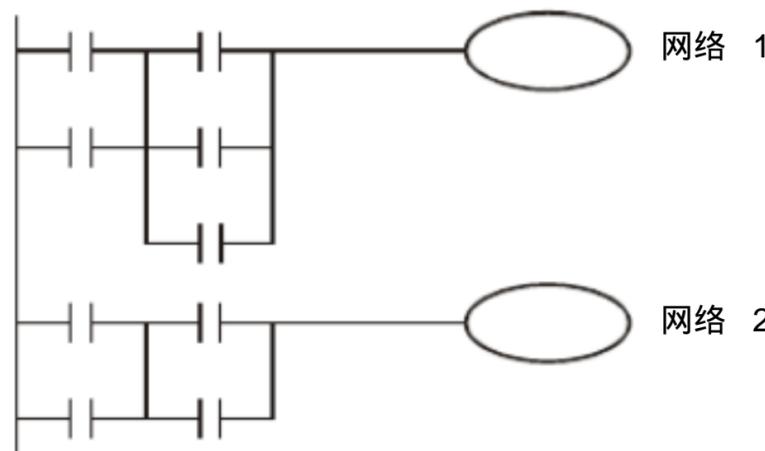


分支线及合并线：往下的一般来说是对装置来区分，对于左边的装置来说是合并线（表示左边至少有两行以上的回路与此垂直线相连接），对于右边的装置及区块来是分支线（表示此垂直线的右边至少有两行以上的回路相连接）。

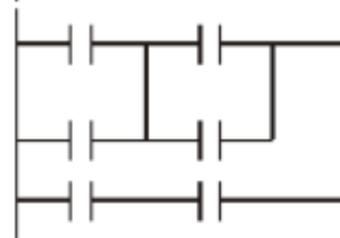


网络：由装置、各种区块所组成的完整区块网络，其垂直线或是连续线所能连接到的区块或是装置均属于同一个网络。

独立的网络：

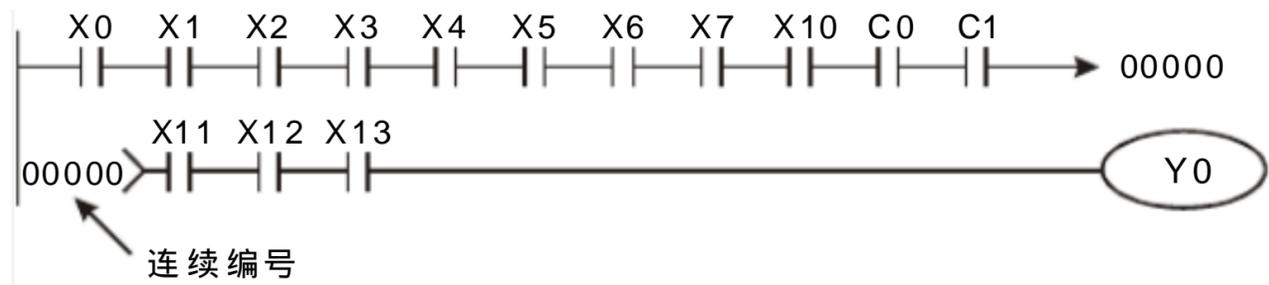


不完整的网络：

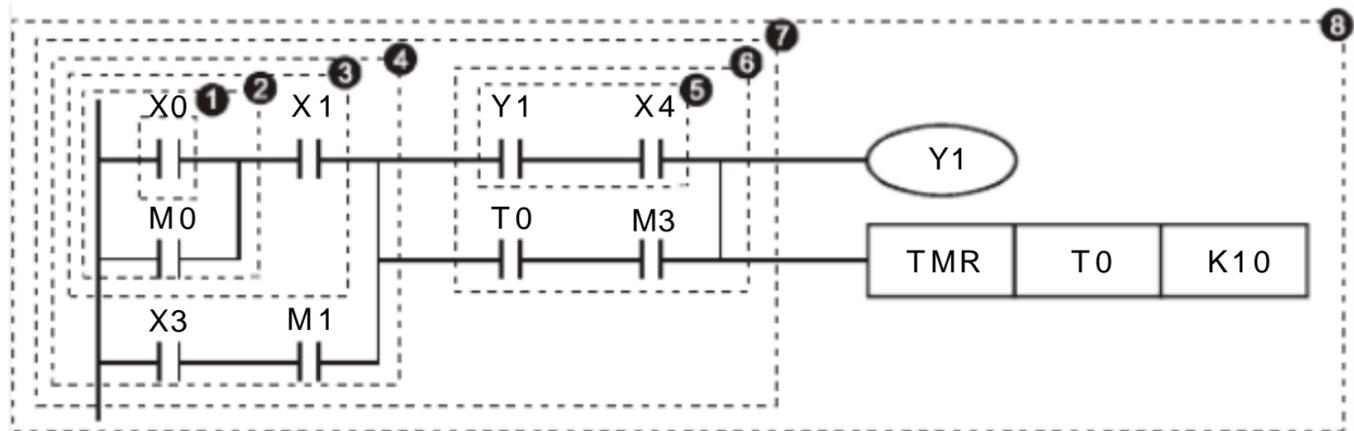


1.4 PLC 梯形图的编辑要点

程序编辑方式是由左母线开始至右母线（在 WPLSoft 编辑省略右母线的绘制）结束，一行编完再换下一行，一行的接点个数最多能有 11 个，若是还不够，会产生连续线继续连接，进而续接更多的装置，连续编号会自动产生，相同的输入点可重复使用。如下图所示：



梯形图程序的运作方式是 由左上到右下 的扫描。线圈及应用指令运算框等属于输出处理，在梯形图形中置于最右边。以下图为例，我们来逐步分析梯形图的流程顺序，右上角的编号为其顺序。

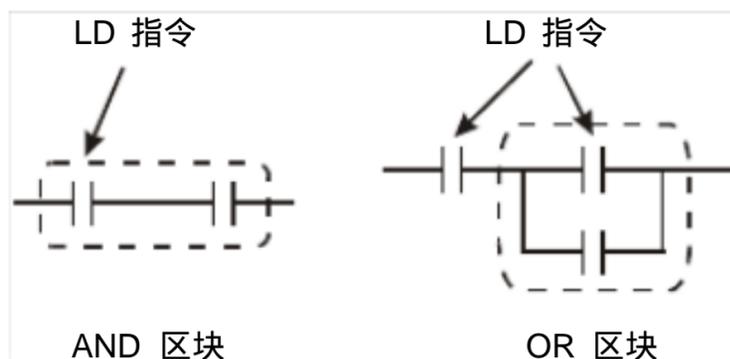


指令顺序解析：

1	LD	X0
2	OR	M0
3	AND	X1
4	LD	X3
	AND	M1
	ORB	
5	LD	Y1
	AND	X4
6	LD	T0
	AND	M3
	ORB	
7	ANB	
8	OUT	Y1
	TMR	T0 K10

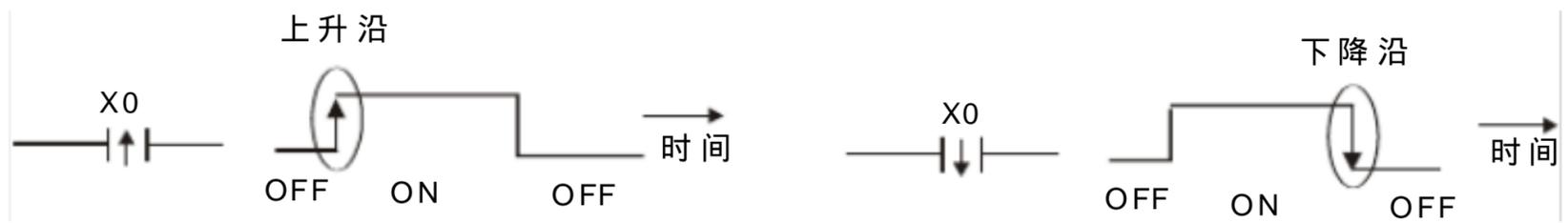
梯形图各项基本结构详述

1. LD (LDI) 指令：一区块的起始给予 LD 或 LDI 的指令。



LDP 及 LDF 的命令结构也是如此，不过其动作状态有所差别。LDP、LDF 在动作时是在接点导通的上升沿或下降沿时才有动作。如下图所示：

LDP、LDF 在动作时是在接点导通的上升沿或下降沿时才有动作。如下图所示：

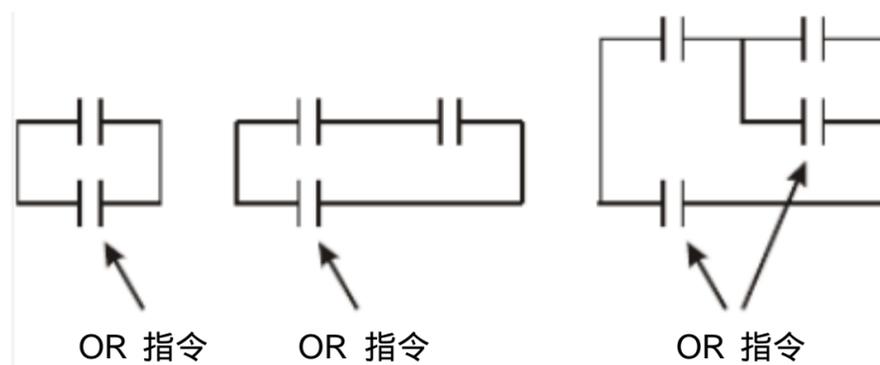


2. AND (ANI) 指令：单一装置接于一装置或一区块的串联组合。



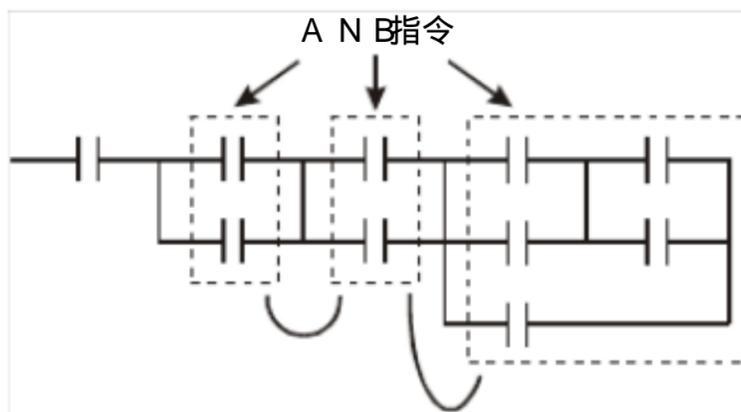
ANDP、ANDF 的结构也是如此，只是其动作发生情形是在上升及下降沿时。

3. OR (ORI) 指令：单一装置接于一装置或一区块的组合。

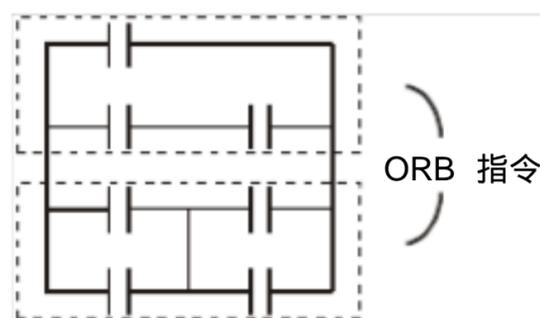


ORP、ORF 也是相同的结构，不过其动作发生时是在上升及下降沿。

4. ANB 指令：一区块与一装置或一区块的串接组合。



5. ORB 指令：一区块与一装置或与一区块并接的组合。



ANB 及 ORB 运算，如果有好几个区块结合，应该由上而下或是由左而右，依序合并成区块或是网络。

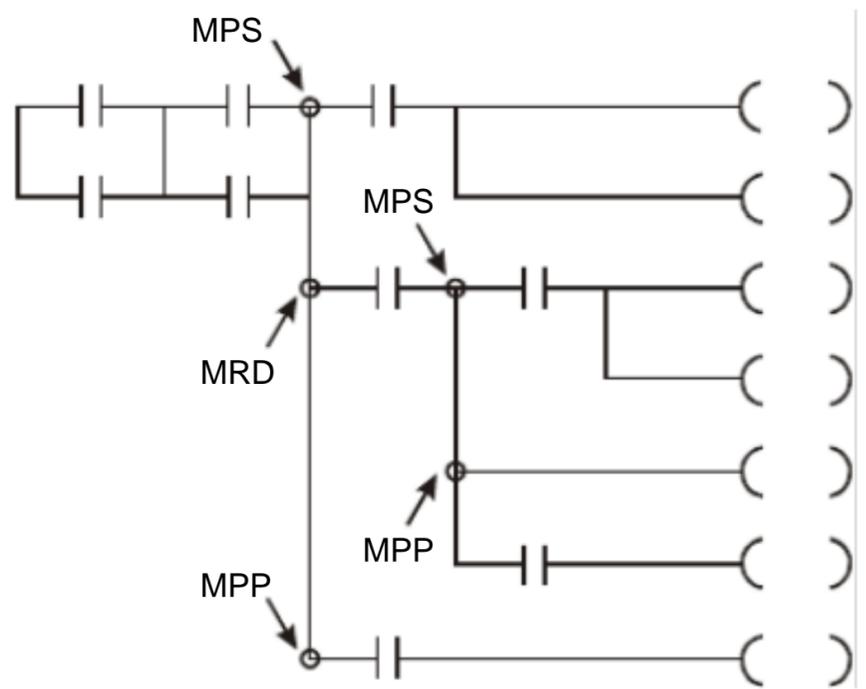
6. MPS、MRD、MPP 指令：多重输出的分支点记忆，这样可以产生多个并且具有变化的不同输出。

MPS 指令是分支点的开始，所谓分支点是指水平线及垂直线相交之处，我们必须经由同一垂直线的接点状态来判定是否应该下接点记忆指令，基本上每个接点都可以下记忆指令，但是考虑到 PLC 的运作方便性以及其容量的限制，所以有些地方在梯形图转换时就会有所省略，可以由梯形图的结构来判断是属于何种接点储存指令。

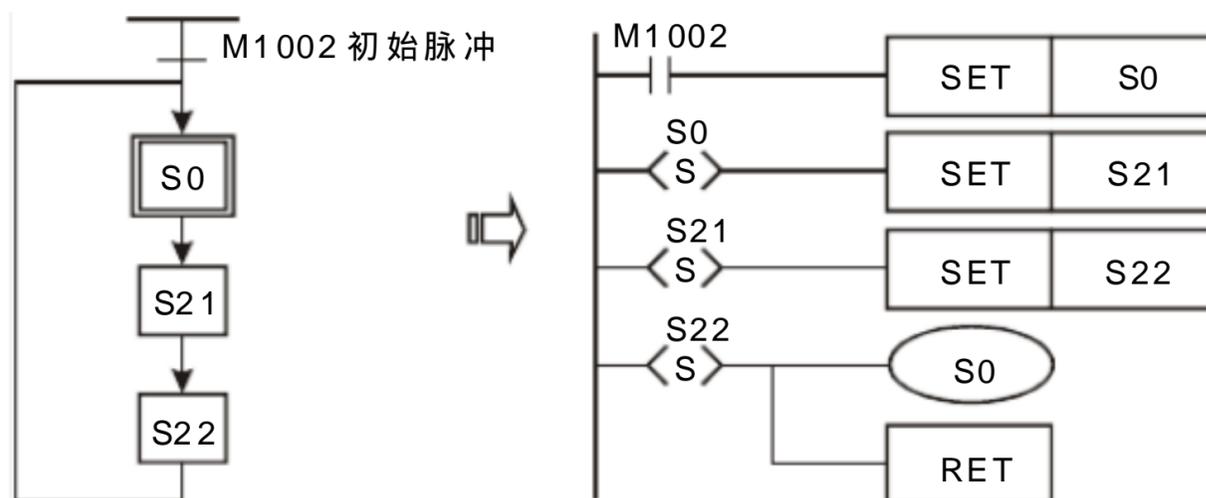
MPS 可以由 “ ” 来做分辨，一共可以连续下此指令 8 次。MRD 指令是分支点记忆读取，因为同一垂直线的逻辑状态是相同的，所以为了继续其它的梯形图的解析进行，必须要再把原接点的状态读出。

MRD 可以由 “ ” 来做分辨。MPP 指令是将最上层分支点开始的状态读出并且把它自堆栈中读出 (Pop)，因为它是同一垂直线的最后一笔，表示此垂直线的状态可以结束了。

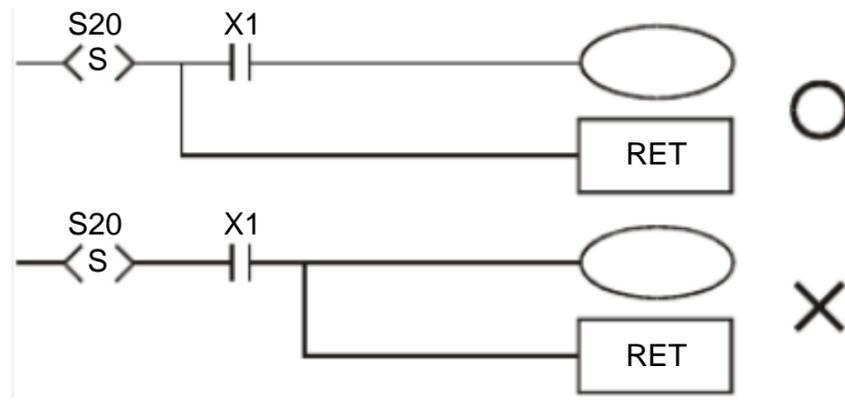
MPP 可以由 “ ” 来做判定。基本上使用上述的方式解析不会有误，但是有时相同的状态输出，编译程序会将其省略，以右图说明：



7. STL 指令：这是用来做为顺序功能图 (SFC, Sequential Function Chart) 设计语法的指令。此种指令可以让我们程序设计人员在程序规划时，能够像平时画流程图时一样，对于程序的步序更为清楚，更具可读性，如下图所示，可以很清楚地看出所要规划的流程顺序，每个步进点 S 转移至下一个步进点后，原步进点会执行 “断电” 的动作，我们可以依据这种流程转换成其右图的 PLC 梯形图型式，称之为步进梯形图。



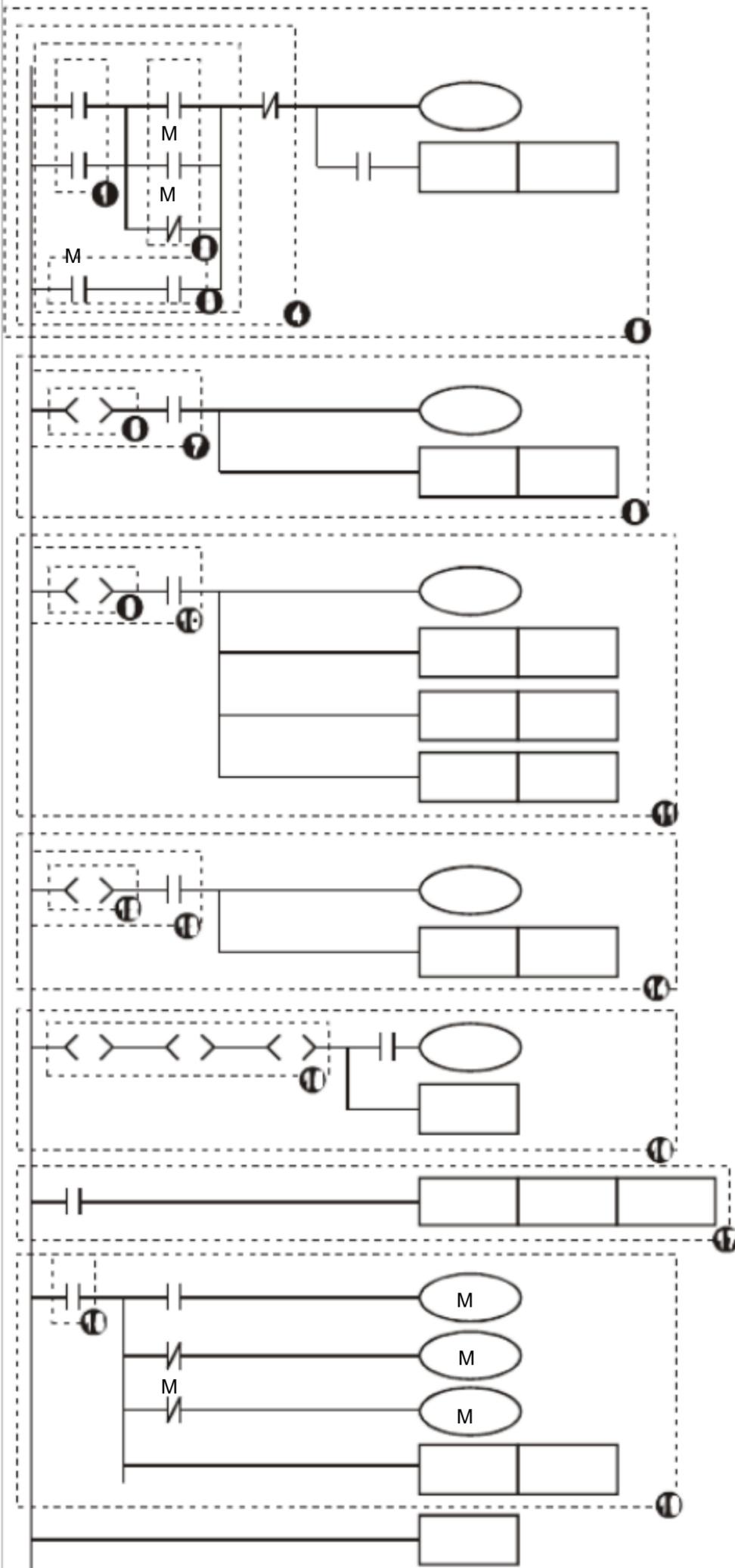
8. RET 指令在步进 梯形程序完成之后要加上 RET 指令，而 RET 也一定要加在 STL 的后面，如下图所示：



步进梯形结构请参考第 4 章步进梯形指令 [STL]、[RET]。

1.5 PLC 指令及各项图形结构的整合转换

梯形图



```

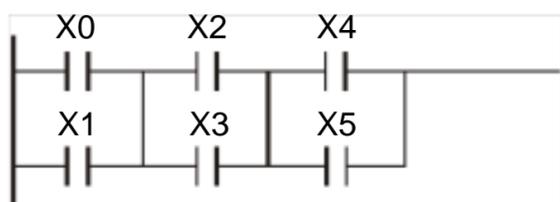
LD X0
OR X1
LD X2
OR M0
OR M1
ANB
LD M2
AND Y0
ORB
AN X1
OUT Y0
AND C0
SET S0
STL S0
LD X10
OUT Y10
SET S10
STL S10
LD X11
OUT Y11
SET S11
SET S12
SET S13
STL S11
LD X12
OUT Y12
SET S20
STL S20
STL S12
STL S13
LD X13
OUT S0
RET
LD X0
CNT C0 K10
LD C0
MPS
AND X1
OUT M0
MRD
AN X1
OUT M1
MPP
AN M2
OUT M2
RST C0
END
    
```

① OR 区块
 ② OR 区块
 串接区块
 ③ AND 区块
 ④ AN
 ⑤ 输出的状态依据④的状态继续往后处理
 ⑥ 多项输出
 ⑦ 步进梯形开始
 ⑧ 状态 S0 与 X10 运算
 ⑨ 状态工作要项及步进点转移
 ⑩ S10 状态取出
 ⑪ 取出 X11 状态
 ⑫ 状态工作要项及步进点转移
 ⑬ S11 状态取出
 ⑭ 读取 X12 状态运算
 ⑮ 状态工作要项及步进点转移
 ⑯ 分支合流
 ⑰ 步进梯形结束
 ⑱ 状态工作要项及步进点转移
 ⑲ 步进动作返回
 ⑳ 读取 C0
 ㉑ 多重输出
 程序结束

语法模糊结构

正确的梯形图解析过程应该是由左至右，由上而下解析合并，然而有些指令不按照此原则一样可以达到相同的梯形图，在此特别叙述于后：

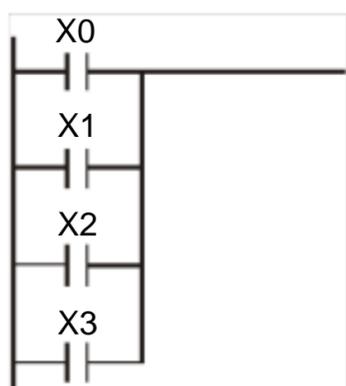
范例程序一：如下图的梯形图形，若使用指令程序表示，有两种方法表示，其动作结果相同。



理想方法		不理想方法	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

两种指令程序，转换成梯形图其图形都一样，为什么会一个较另一个好呢？问题就在主机的运算动作，第一个：是一个区块一个区块合并，第二个：则是最后才合并，虽然程序代码的最后长度都相同，但是由于在最后才合并（ANB 作合并动作，但 ANB 指令不能连续使用超过 8 次），则必须要把先前所计算出的结果储存起来，现在只有两个区块，主机可以允许，但是要是区块超过主机的限制，就会出现问题，所以最好的方式就是一区块一建立完就进行区块合并的指令，而且这样做对于程序规划者的逻辑顺序也比较不会乱。

范例程序二：如下图的梯形图形，若使用指令程序表示，亦有两种方法表示，其动作结果相.....同。

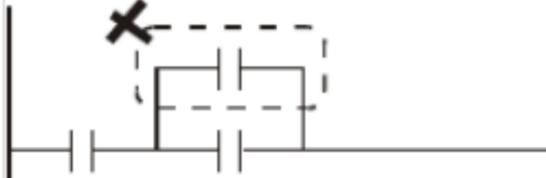
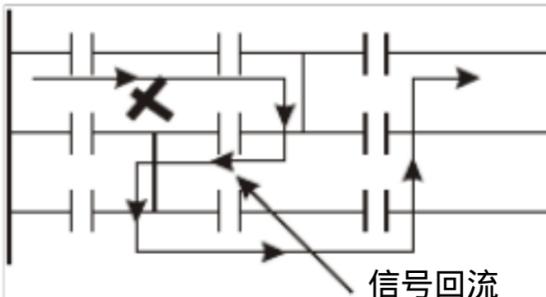
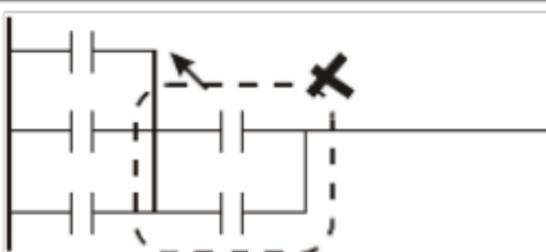
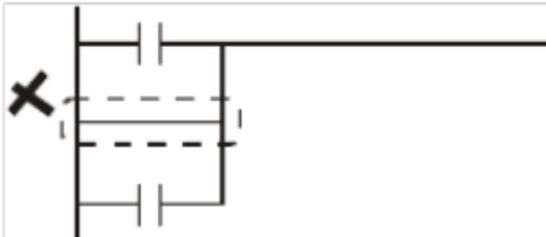
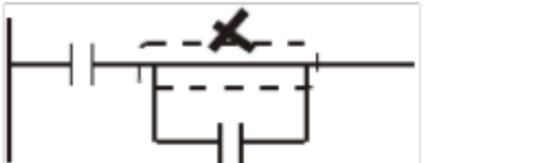
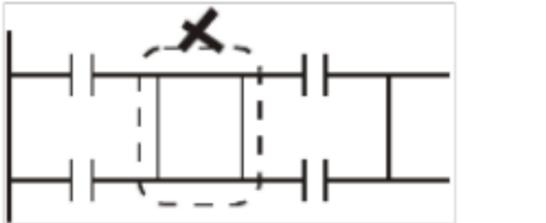
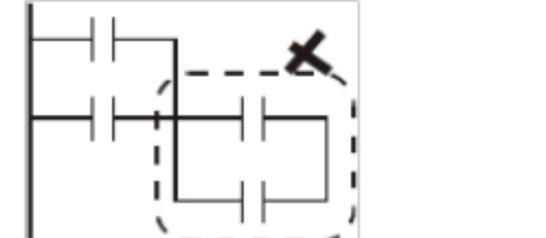
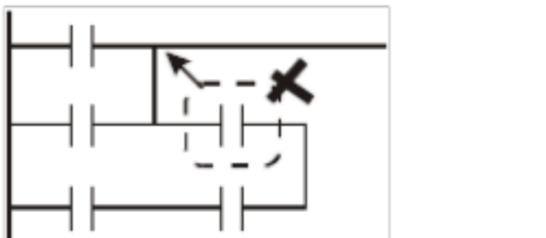


理想方法		不理想方法	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

这两个程序解析就有明显的差距，不但程序代码增加，主机的运算记忆也要增加，所以最好是能够按照所定义的顺序来撰写程序。

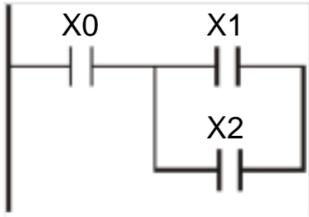
梯形图的错误图形

在编辑梯形图形时，虽然可以利用各种梯形符号组合成各种图形，由于 PLC 处理图形程序的原则是由上而下，由左至右，因此在绘制时，要以左母线为起点，右母线为终点（WPLSoft 梯形图编辑区将右母线省略），从左向右逐个横向写入。一行写完，自上而下依次再写下一行。以下为常见的各种错误图形：

	<p>不可往上做 OR 运算</p>
	<p>输入起始至输出的信号回路有 “回流” 存在</p>
	<p>应该先由右上角输出</p>
	<p>要做合并或编辑应由左上往右下，虚线括处的区块应往上移</p>
	<p>不可与空装置做并接运算</p>
	<p>空装置也不可以与别的装置做运算</p>
	<p>中间的区块没有装置</p>
	<p>串联装置要与所串联的区块水平方向接齐</p>
	<p>Label P0 的位置要在完整网络的第一行</p>
	<p>区块串接要与串并左边区块的最上段水平线接齐</p>

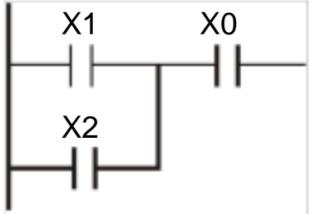
1.6 梯形图的化简 (总结：由上到下，由左到右，由大到小)

串联块与并联块串联时，将块放在前面可节省 ANB 指令



梯形图转译成指令：

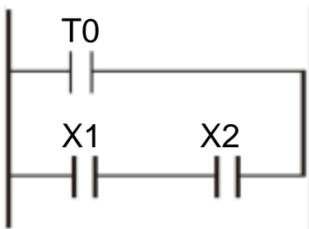
```
LD X0
LD X1
OR X2
ANB
```



梯形图转译成指令：

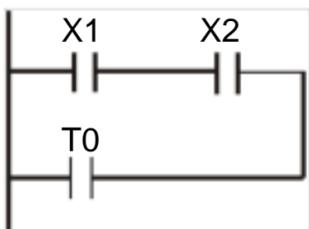
```
LD X1
OR X2
AND X0
```

单一装置与块并接，块放上面可以省 ORB 指令



梯形图转译成指令：

```
LD T0
LD X1
AND X2
ORB
```

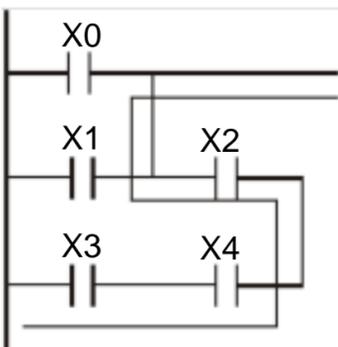


梯形图转译成指令：

```
LD X1
AND X2
OR T0
```

梯形图 (a) 中，上面的块比下面的块短，可以把上下的块调换达到同样的逻辑结果，因为图 (a) 是不合法的，因为有“信号回流”回路

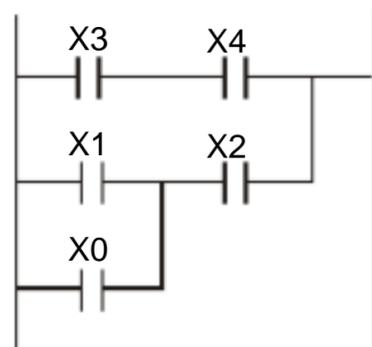
(a) 是



梯形图转译成指令：

```
LD X0
OR X1
AND X2
LD X3
AND X4
ORB
```

图(a)



梯形图转译成指令：

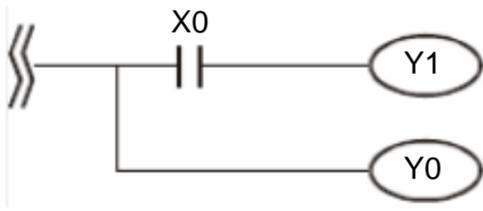
```
LD X3
AND X4
LD X1
OR X0
AND X2
```

图(b)

ORB

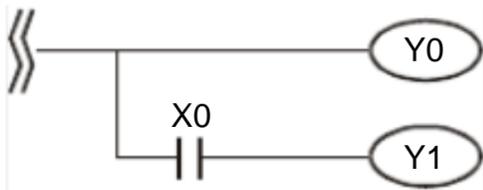
相同垂直线的多重条件输出，没有输入装置及其运算的放在上面可以省略

MPS、MPP



梯形图转译成指令：

```
MPS
AND X0
OUT Y1
MPP
OUT Y0
```



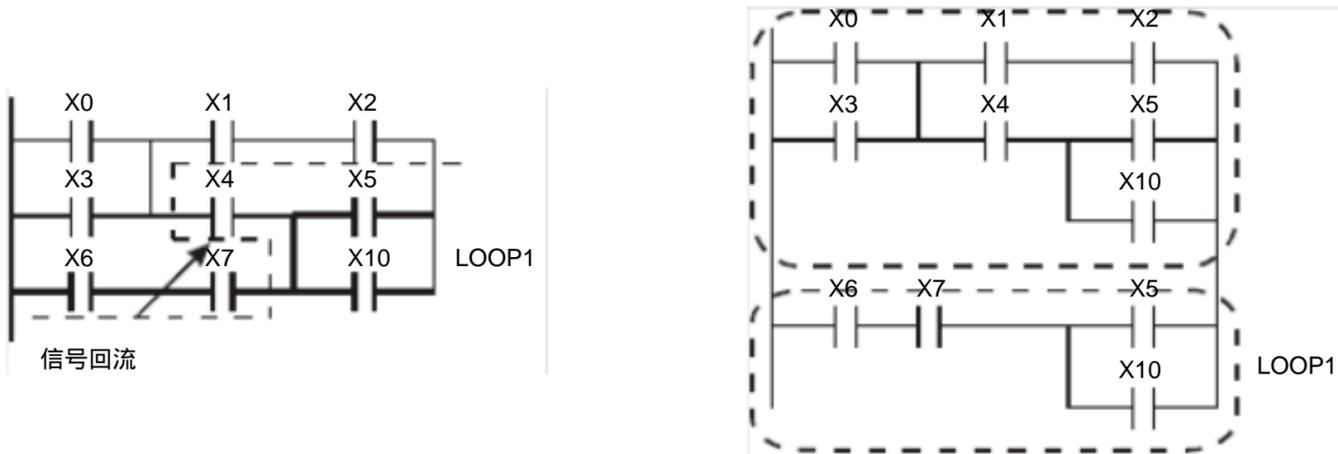
梯形图转译成指令：

```
OUT Y0
AND X0
OUT Y1
```

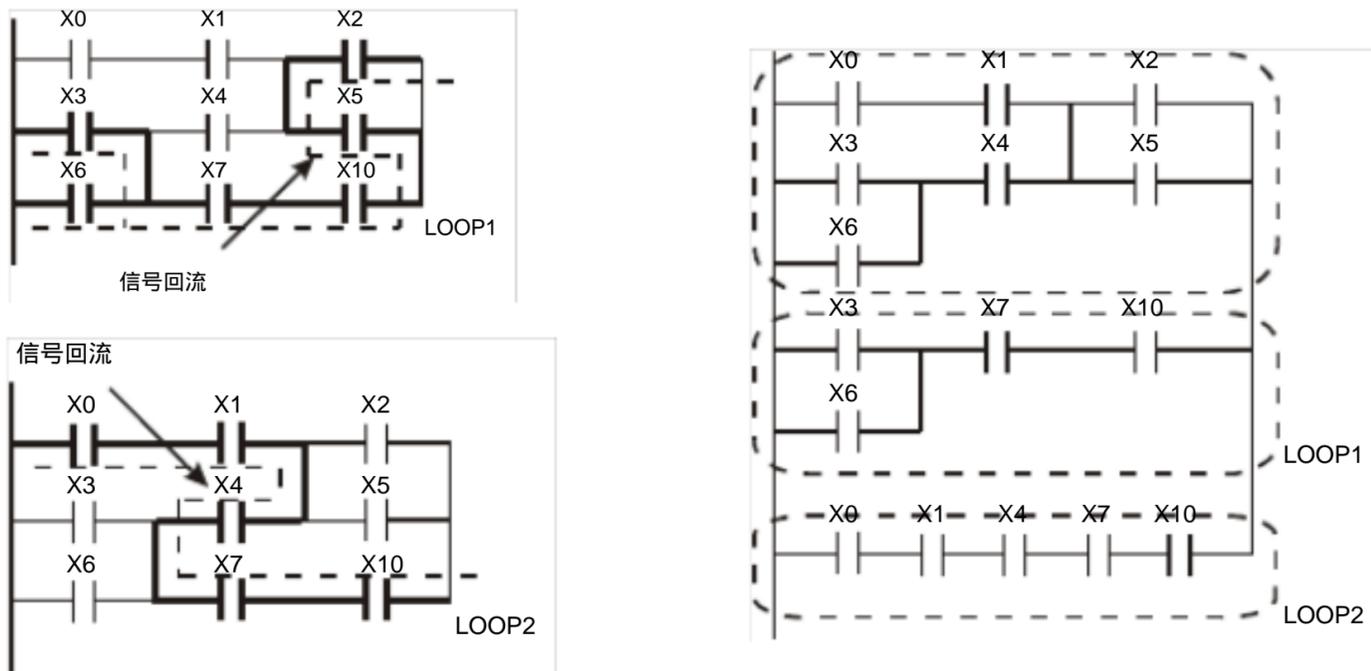
信号回流的线路修正

在以下的两个范例，左边是我们想要的图形，但是根据我们的定义，左边的图是有误的，其中存在不合法的 信号回流 路径，如图所示。并修正如右图，如此可完成使用者要的电路动作。

例一：



例二：



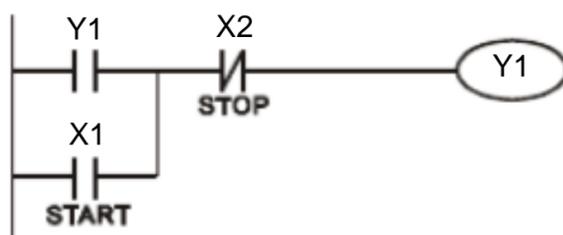
1.7 常用基本程序设计范例

起动、停止及自保

有些应用场合需要利用按钮的瞬时闭合及瞬时断开作为设备的启动及停止。因此若要维持持续动作，则必须设计自保回路，自保回路有下列几种方式：

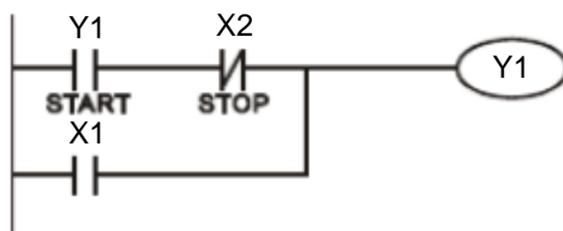
范例 1：停止优先的自保回路

当启动常开接点 $X1=On$ ，停止常闭接点 $X2 = Off$ 时， $Y1=On$ ，此时将 $X2=On$ ，则线圈 $Y1$ 停止受电，所以称为停止优先。



范例 2：启动优先的自保回路

启动常开接点 $X1=On$ ，停止常闭接点 $X2 = Off$ 时， $Y1=On$ ，线圈 $Y1$ 将受电且自保，此时将 $X2=On$ ，线圈 $Y1$ 仍因自保接点而持续受电，所以称为启动优先。

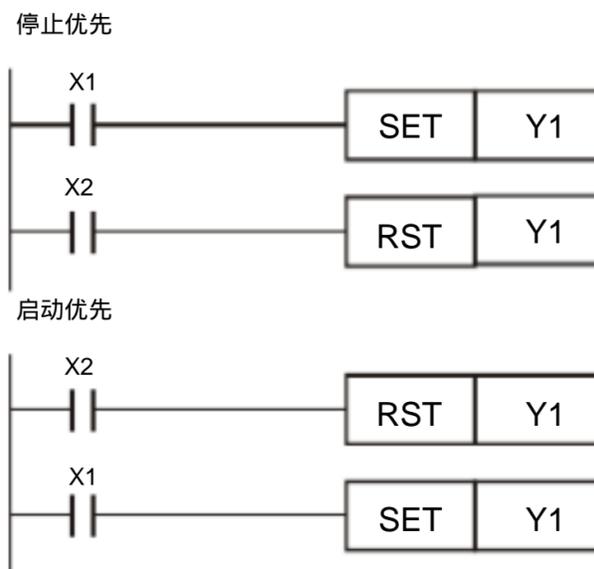


范例 3：置位 (SET)、复位 (RST) 指令的自保回路

右图是利用 RST 及 SET 指令组合成的自保电路。

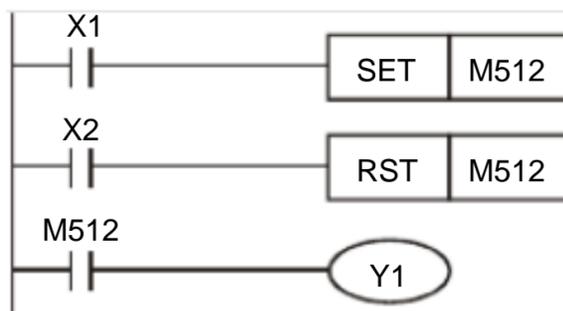
RST 指令设置在 SET 指令之后，为停止优先。由于 PLC 执行程序时，是由上而下，因此会以程序最后 $Y1$ 的状态作为 $Y1$ 的线圈是否受电。所以当 $X1$ 及 $X2$ 同时动作时， $Y1$ 将失电，因此为停止优先。

SET 指令设置在 RST 指令之后，为启动优先。当 $X1$ 及 $X2$ 同时动作时， $Y1$ 将受电，因此为启动优先。



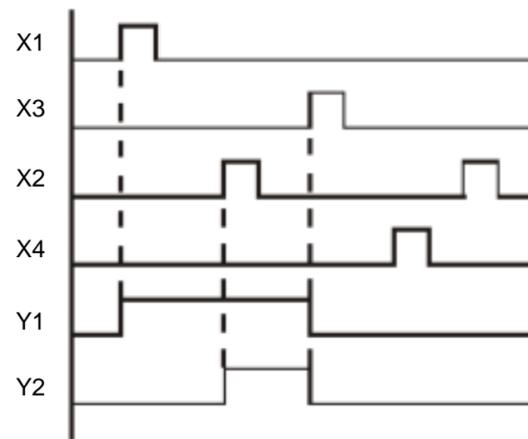
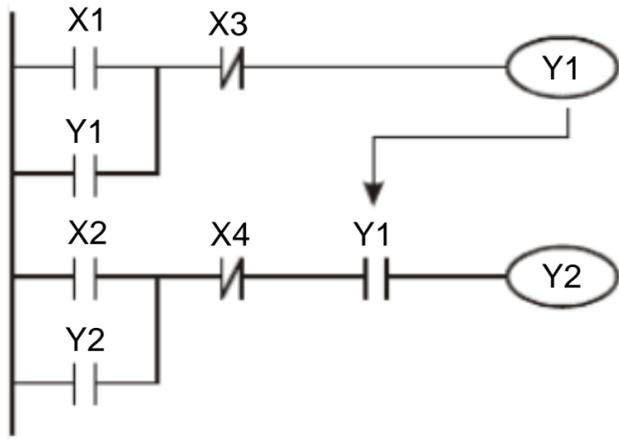
范例 4：停电保持

右图辅助继电器 $M512$ 为停电保持（请参考 PLC 主机使用手册），则如图的电路不仅在通电状态下能自保，而且一旦停电再复电，还能保持停电的自保状态，因而使原控制保持连续性。



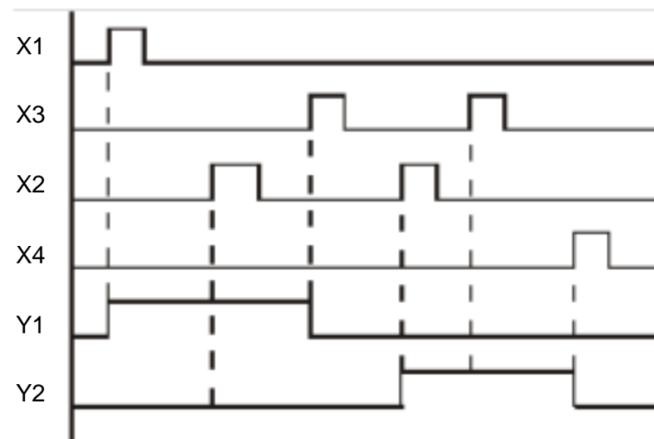
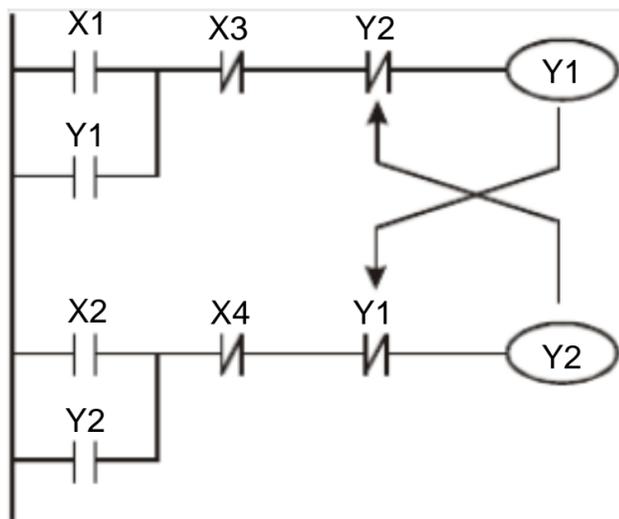
常用的控制回路

范例 5：条件控制



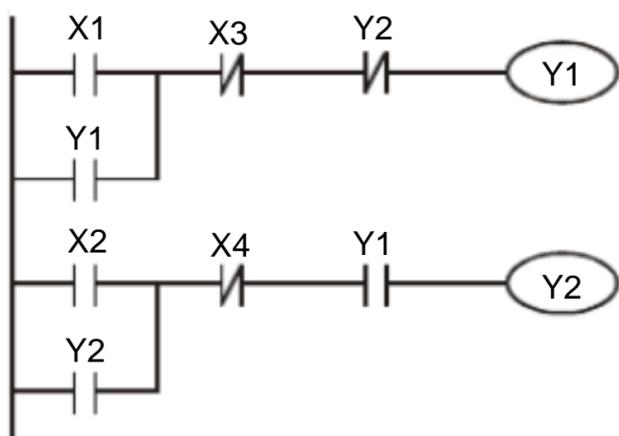
X1、X3 分别启动 /停止 Y1，X2、X4 分别启动 /停止 Y2，而且均有自保回路。由于 Y1 的常开接点串联了 Y2 的电路，成为 Y2 动作的一个 AND 的条件，所以 Y2 动作要以 Y1 动作为条件，Y1 动作中 Y2 才可能动作。

范例 6：互锁控制



上图为互锁控制回路，启动接点 X1、X2 那一个先有效，对应的输出 Y1、Y2 将先动作，而且其中一个动作了，另一个就不会动作，也就是说 Y1、Y2 不会同时动作（互锁作用）。即使 X1，X2 同时有效，由于梯形图程序是自上而下扫描，Y1、Y2 也不可能同时动作。本梯形图只有让 Y1 优先。

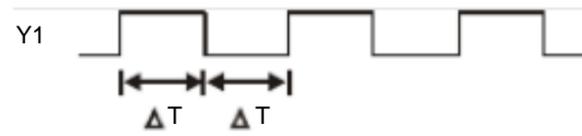
范例 7：顺序控制



若把范例 5 条件控制”中 Y2 的常闭接点串入到 Y1 的电路中，作为 Y1 动作的一个 AND 条件（如左图所示），则这个电路不仅 Y1 作为 Y2 动作的条件，而且当 Y2 动作后还能停止 Y1 的动作，这样就使 Y1 及 Y2 确实执行顺序动作的程序。

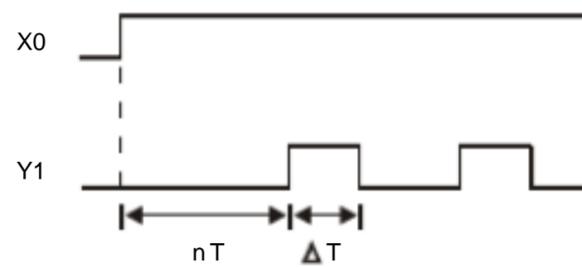
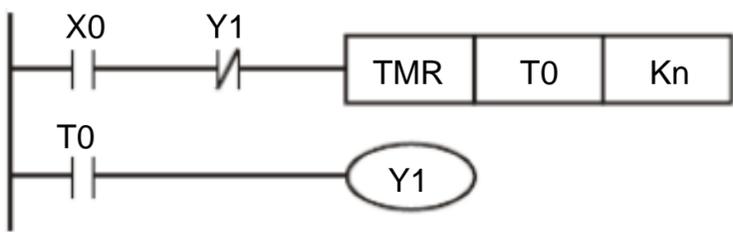
范例 8：振荡电路

周期为 $T_{on} + T_{off}$ 的振荡电路



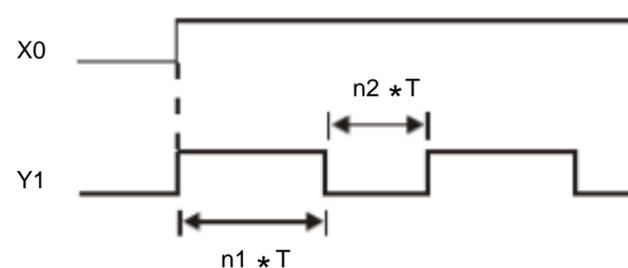
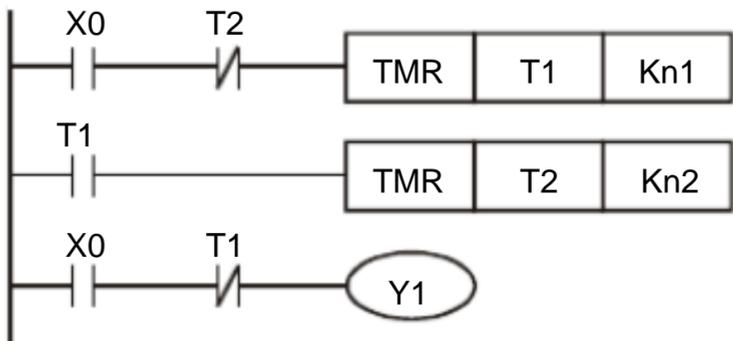
上图为一个很简单的梯形图形。当开始扫描 Y1 常闭接点时，由于 Y1 线圈为失电状态，所以 Y1 常闭接点闭合，接着扫描 Y1 线圈时，使其受电，输出为 1。下次扫描周期再扫描 Y1 常闭接点时，由于 Y1 线圈受电，所以 Y1 常闭接点打开，进而使线圈 Y1 失电，输出为 0。重复扫描的结果，Y1 线圈上输出了周期为 $T_{(On)} + T_{(Off)}$ 的振荡波形。

周期为 $nT + \Delta T$ 的振荡电路



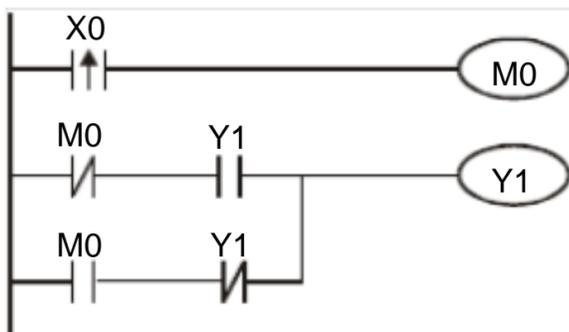
上图的梯形图程序使用定时器 T0 控制线圈 Y1 的受电时间，Y1 受电后，它在下个扫描周期又使定时器 T0 关闭，进而使 Y1 的输出成了上图中的振荡波形。其中 n 为定时器的十进制设定值，T 为该定时器时基（时钟周期）。

范例 9：闪烁电路



上图是常用的使指示灯闪烁或使蜂鸣器报警用的振荡电路。它使用了两个定时器，以控制 Y1 线圈的 On 及 Off 时间。其中 n_1 、 n_2 分别为 T1 及 T2 的计时设定值，T 为该定时器时基（时钟周期）。

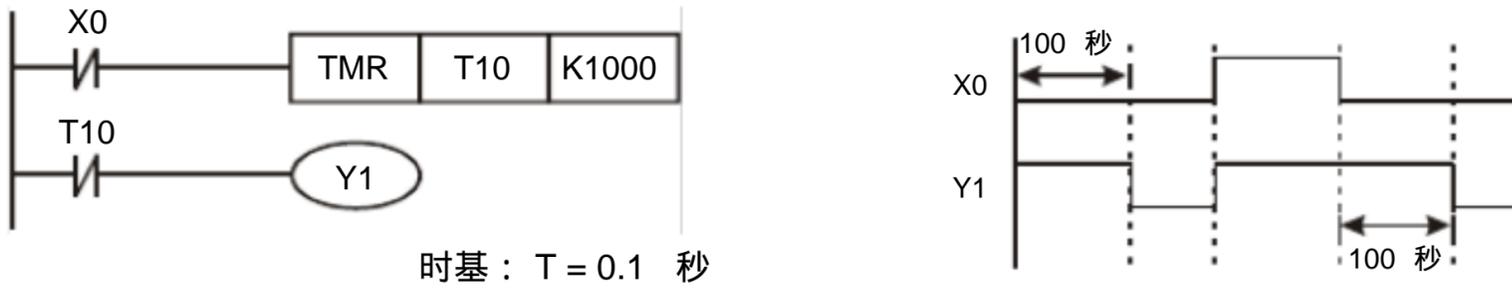
范例 10：触发电路



在上图中，X0 的上升沿微分指令使线圈 M0 产生 T（一个扫描周期时间）的单脉冲，在这个扫描周期内线圈 Y1 也受电。下个扫描周期线圈 M0 失电，其常闭接点 M0 及常闭接点 Y1 都闭合着，进而使线圈 Y1 继续保持受电状态，直到输入 X0 又来了一个上升沿，再次使线圈 M0 受电一个扫描周期，同时导致线

圈 Y1 失电 ...。其动作时序如上图。这种电路常用于靠一个输入使两个动作交替执行。另外由上时序图形可看出：当输入 X0 是一个周期为 T 的方波信号时，线圈 Y1 输出便是一个周期为 2T 的方波信号。

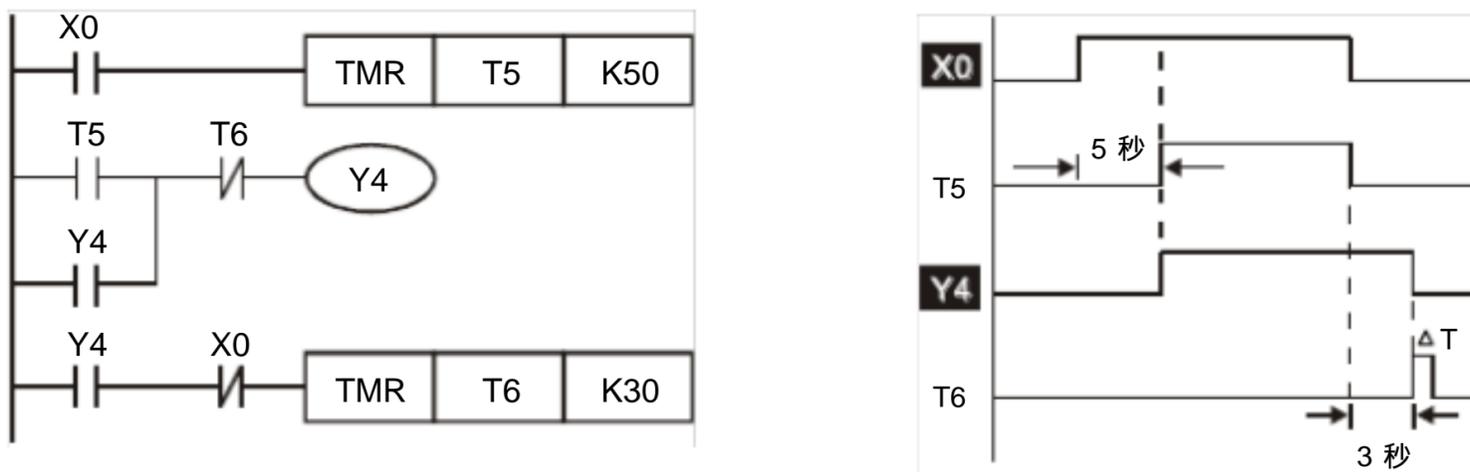
范例 11：延迟电路



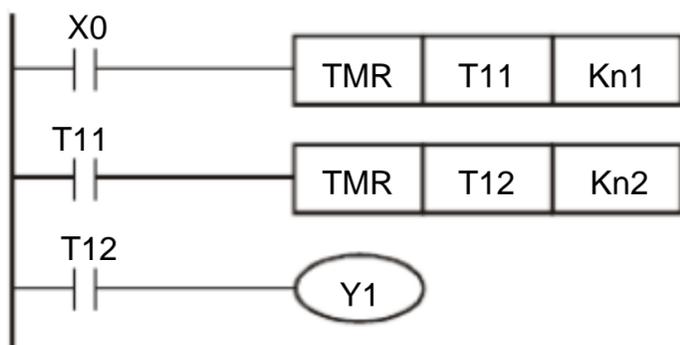
时基：T = 0.1 秒

当输入 X0 On 时，由于其对应常闭接点 Off，使定时器 T10 处于失电状态，所以输出线圈 Y1 受电，直到输入 X0 Off 时，T10 得电并开始计时，输出线圈 Y1 延时 100 秒 ($K1000 \times 0.1 \text{ 秒} = 100 \text{ 秒}$) 后失电，请参考上图的动作时序。

范例 12：通断延迟电路，使用两个定时器组成的电路，当输入 X0 On 及 Off 时，输出 Y4 都会产生延时。

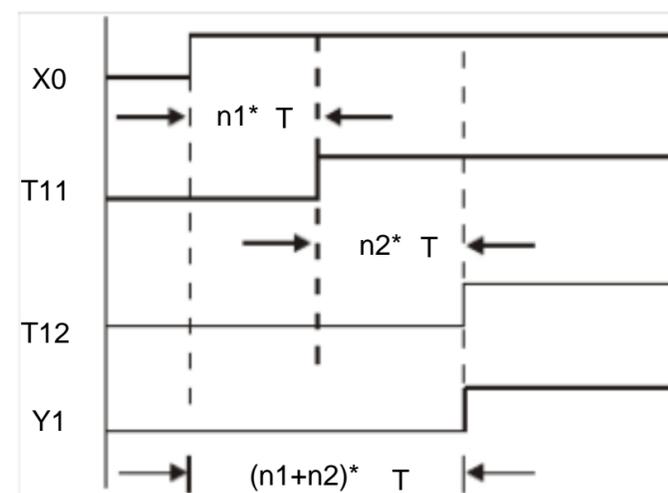


范例 13：延长计时电路

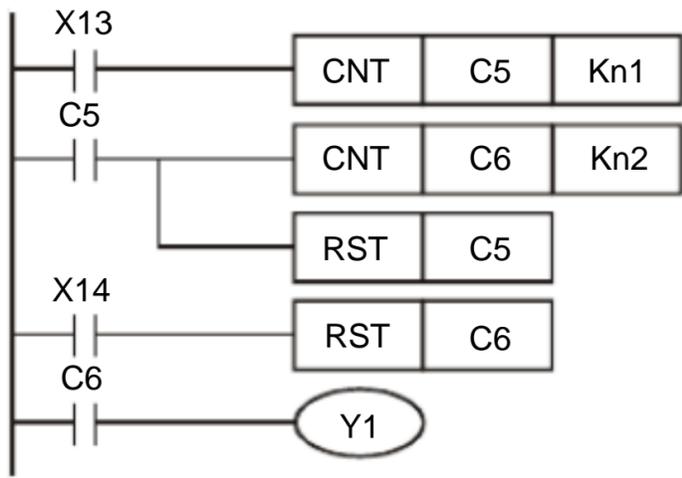


定时器 T11、T12，时钟周期：T

在左图电路中，从输入 X0 闭合到输出 Y1 得电的总延迟时间 $= (n1+n2) \times T$ ，其中 T 为时钟周期。

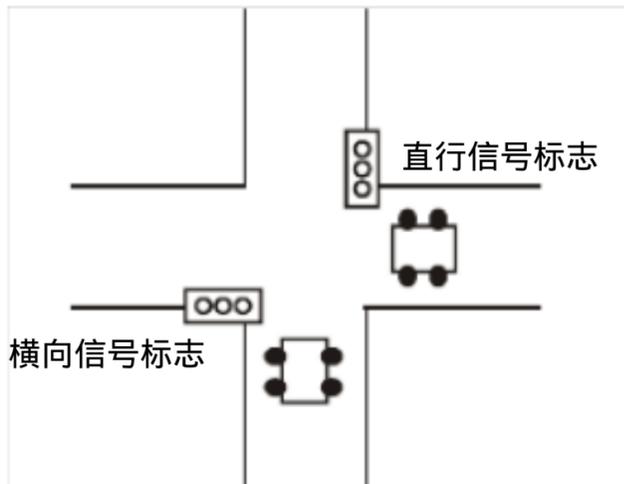


范例 14：扩大计数范围的方法



16 位的计数器，计数范围为 0~32,767，如左图电路，用两个计数器，可使计数数值扩大到 $n1 \cdot n2$ 。当计数器 C5 计数到达 $n1$ 时，将使计数器 C6 计数一次，同时将自己复位 (Reset)，以接着对来自 X13 的脉冲计数。当计数器 C6 计数到达 $n2$ 时，则自 X13 输入的脉冲正好是 $n1 \cdot n2$ 次。

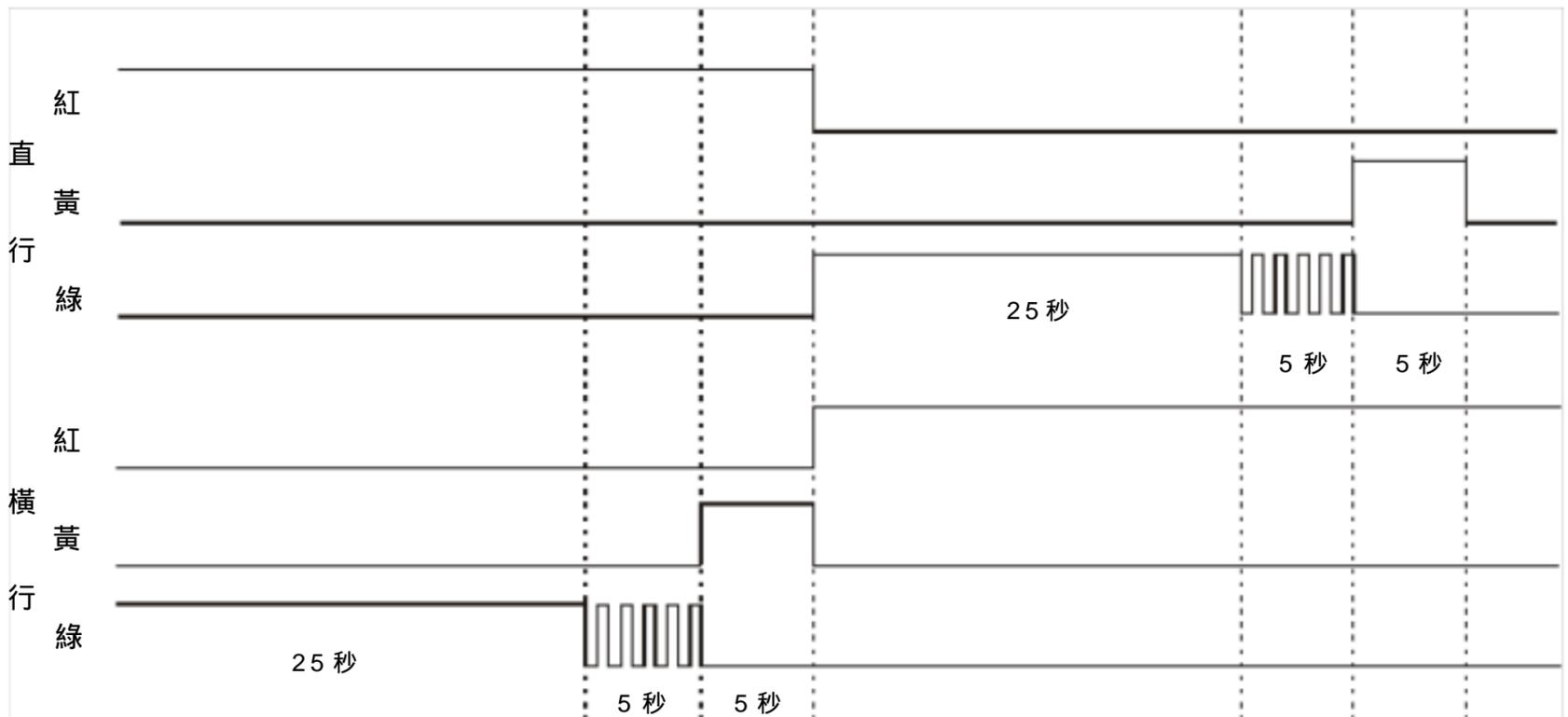
范例 15：红绿灯控制（使用步进梯形指令）



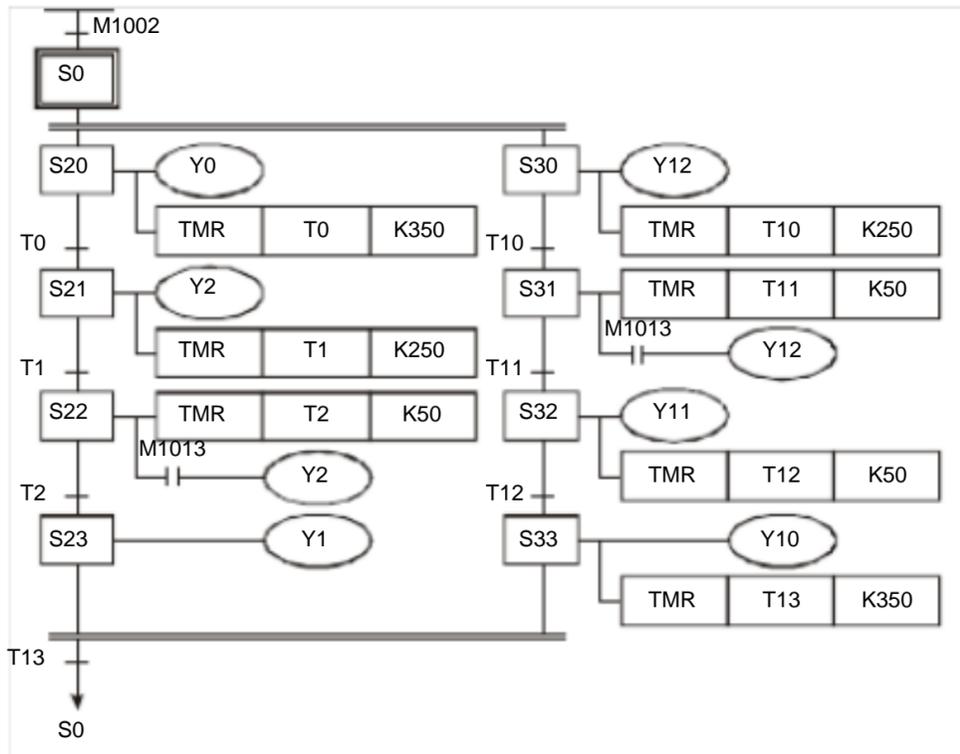
红绿灯控制：

	红灯	黄灯	绿灯	绿灯闪烁
直向信号标志	Y0	Y1	Y2	Y2
横向信号标志	Y10	Y11	Y12	Y12
灯号时间	35 秒	5 秒	25 秒	5 秒

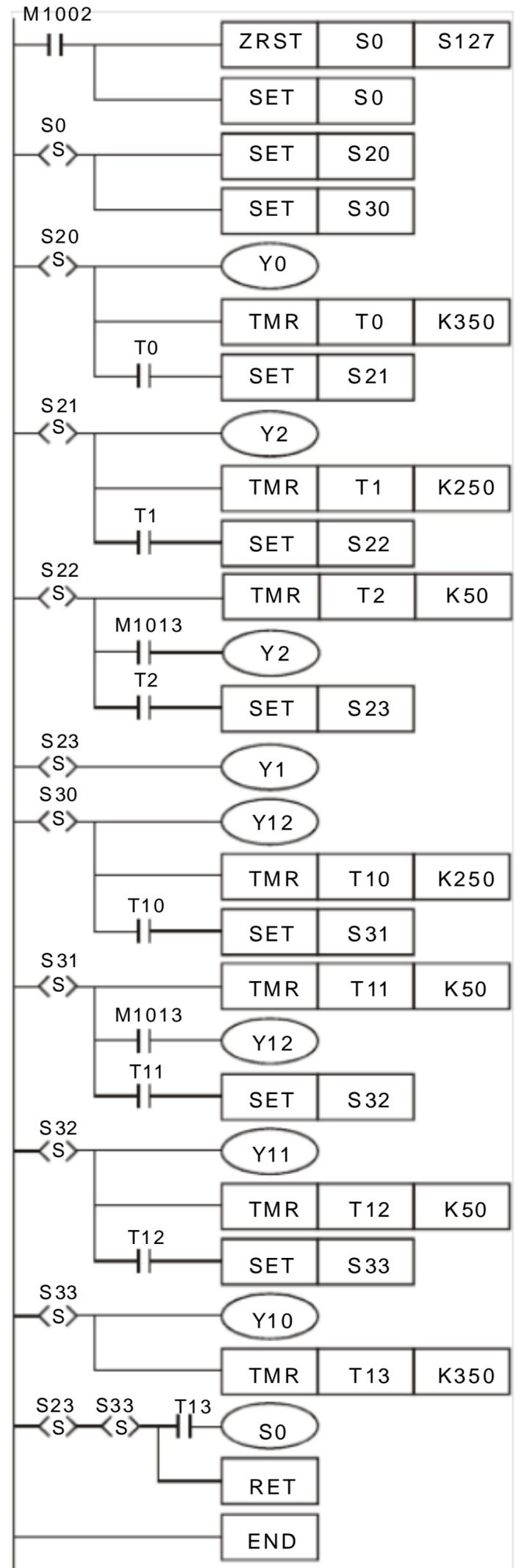
时序图：



SFC 图：



梯形图：



SFC 绘制	内部梯形图检视
<p>The SFC diagram shows a sequence of states. It starts with LAD-0, which leads to state S0. From S0, transition 0 leads to a parallel branch of states S20 and S30. From S20, transition 1 leads to state S21. From S30, transition 5 leads to state S31. From S21, transition 2 leads to state S22. From S31, transition 6 leads to state S32. From S22, transition 3 leads to state S23. From S32, transition 7 leads to state S33. Finally, transition 4 leads from S23/S33 back to state S0. States LAD-0, S22, and S32 are circled in the diagram.</p>	<p>1. LAD-0</p> <p>Ladder logic for LAD-0: A normally closed contact labeled M1002 is connected to two parallel outputs. The top output is a ZRST (Reset) instruction for S0, with S127 as a comment. The bottom output is a SET (Set) instruction for S0.</p>
	<p>2. 转移条件 1</p> <p>Ladder logic for transition 1: A normally closed contact labeled T0 is connected to a TRANS* (Transition) instruction.</p>
	<p>3. S22</p> <p>Ladder logic for S22: A normally closed contact labeled M1013 is connected to two parallel outputs. The top output is a TMR (Timer) instruction for T2 with a value of K50. The bottom output is a coil for output Y2.</p>
	<p>4. 转移条件 4</p> <p>Ladder logic for transition 4: A normally closed contact labeled T13 is connected to a TRANS* (Transition) instruction.</p>
	<p>5. 转移条件 7</p> <p>Ladder logic for transition 7: A normally closed contact labeled T12 is connected to a TRANS* (Transition) instruction.</p>

MEMO