Основные команды PLC

- Нормально-открытый контак

Прочитав этот сигнал, контроллер начинает постоянно проверять состояние входа, норме которого указан программистом.

И как только контролер определяет изменение состояния входа (X1), с выключено на включено, следует включение выхода (Y1). Данный символ может относиться не только к физическим входам контроллера, а также и к внутренним (вспомогательным) реле. Число и ограничения по использованию внутренних реле определяется только возможностями контроллера.

Команда (LDI) - нормально закрытый контакт

Прочитав этот сигнал, контроллер начинает постоянно проверять состояние входа (X2). И как только контроллер определяет изменение состояния входа (X2), с включено на выключено, следует включение выхода (Y1). Данный символ может относиться не только к физическим входам контроллера, а также и к внутренним (вспомогательным) реле. Логика работы соответствует таблице

Таблица логического состояния входов

Логическое состояние	*LD* - нормально открытый контакт	*LDI* – нормально закрытый контакт
0	Ложь	Истина
1	Истина	Ложь

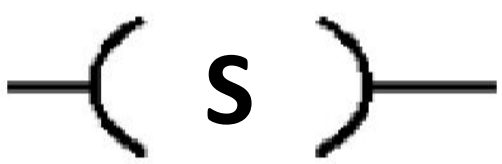
Команда (OUT) - инициализация ***Выхода***

Прочитав эту команду, контроллер изменит состояние выхода с выключено на включено. Так же, как и в случае с входами данный символ может относиться не только к физическим выходам контроллера, но и к внутренним (вспомогательным) реле. Число и ограничения по использованию внутренних реле определяется только возможностями контроллера. Логика работы соответствует таблице 11.2.

Таблица 11.2 – Таблица логического состояния выходов

Логическое состояние	*OUT* – Выход
0	**Ложь**
1	**Истина**

Команда SET – инициализация выхода



- Если S (Set [Включить]) активизирован, то значение данных на адресе OUT устанавливается в 1. Если S не активизирован, то OUT не изменяется.

Команда RESET– инициализация выхода

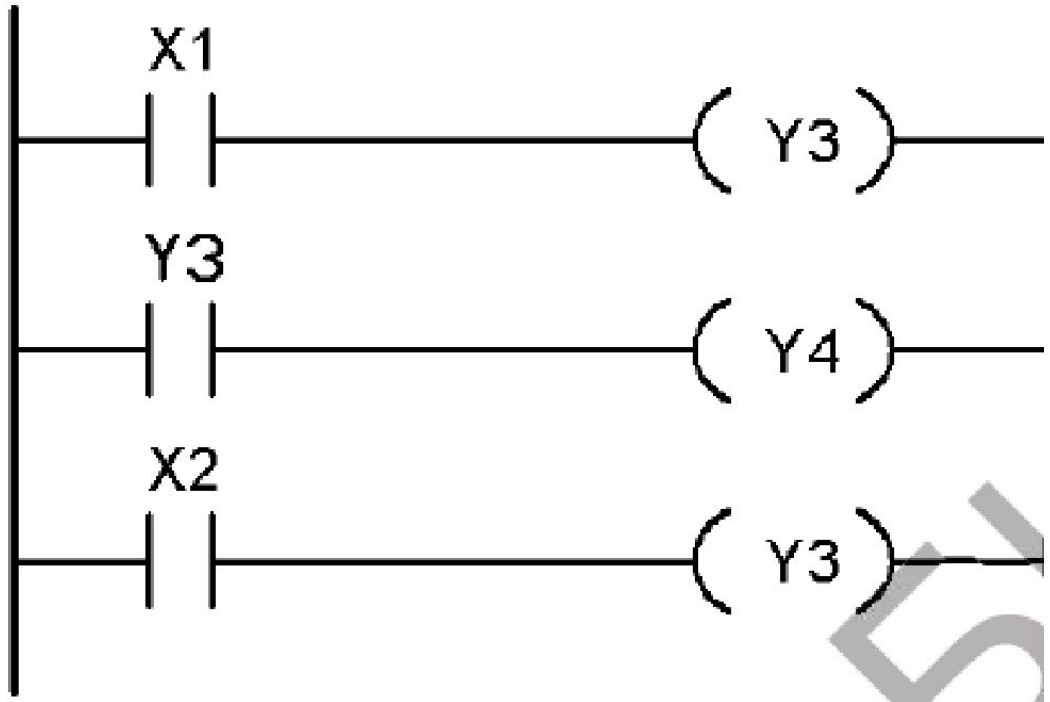
- Если R (Reset [Выключить]) активизирован, то значение данных на адресе OUT устанавливается в 0. Если R не активизирован, то OUT не изменяется.

Примечание:

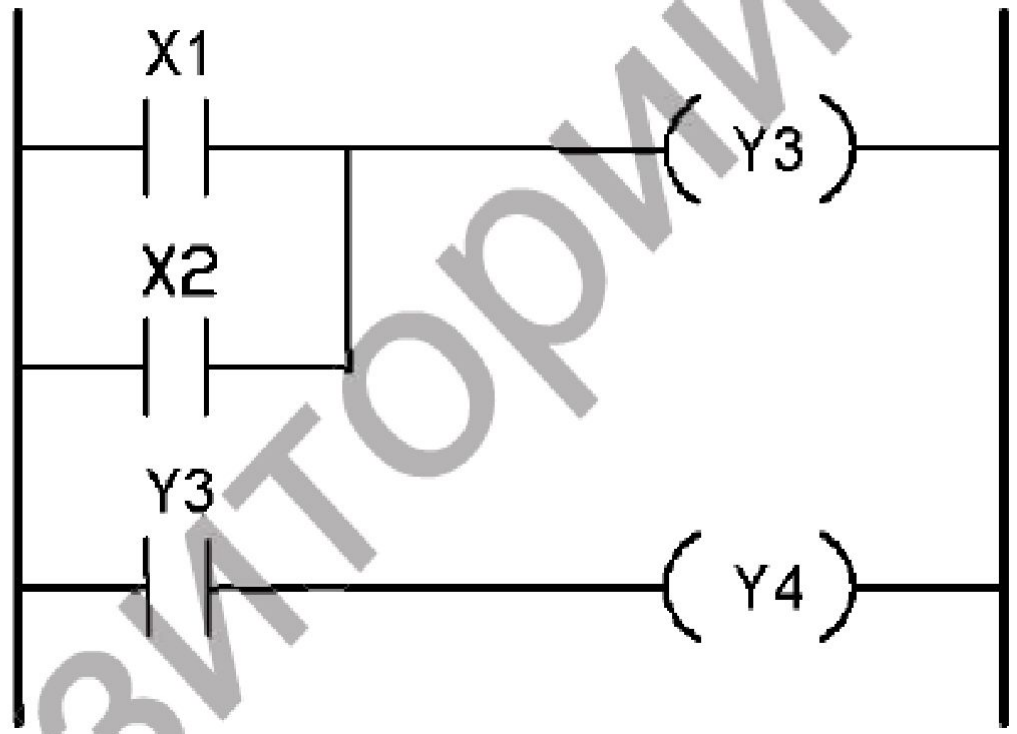
Избегайте двойной записи выходов (double coil) , так как это может привести к помехам при отработке программы.

Пример двойной записи приведен на рисунке : а – ошибочная, б – верная запись:

а)



б)



Использование внутреннего реле (меркера) – контроль уровня.

Предположим нам необходимо постоянно поддерживать определенный уровень в баке с водой (рисунок 11.13). Для решения необходимо:

- датчик верхнего уровня – вход (X2) (переходит в состояние «включено», когда уровень воды падает);
- датчик нижнего уровня – вход (X1) (переходит в состояние «включено», когда уровень воды падает);
- насос – выход (Y1).

К сожалению, использование логики для двух входов не даст желаемого результата, так как если мы будем включать насос, когда оба входа окажутся в состоянии «включено», то насос будет поднимать уровень только до нижнего датчика, или уровень воды никогда не опустится до уровня нижнего датчика, т.е. начнет работать в очень жестком режиме постоянного включения-выключения.

Для обеспечения работы насоса в оптимальном режиме нам понадобится включение в задачу дополнительного элемента, который должен обеспечить:

- *Включение* насоса при достижении нижнего уровня;
- *Выключение* насоса при достижении верхнего уровня.

Для этого мы будем использовать внутренне реле контроллера M1, также называемое меркером, в результате чего получим схему, изображенную на рисунке 11.13.

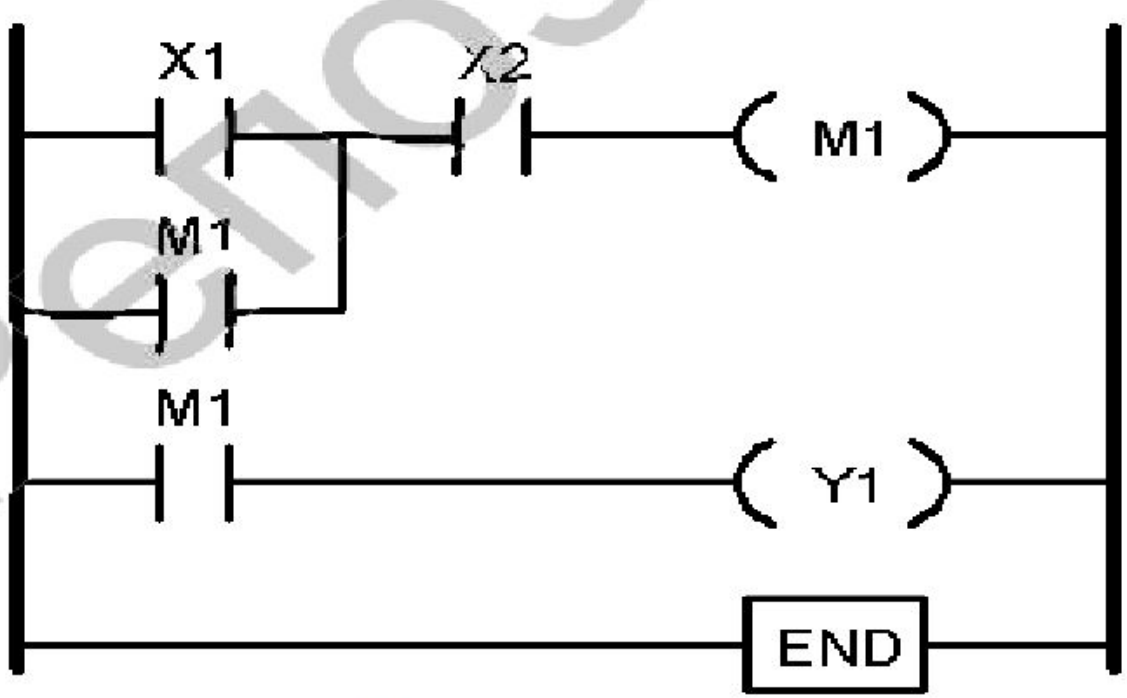


Рисунок 11.13 – Схема программы для включения насоса

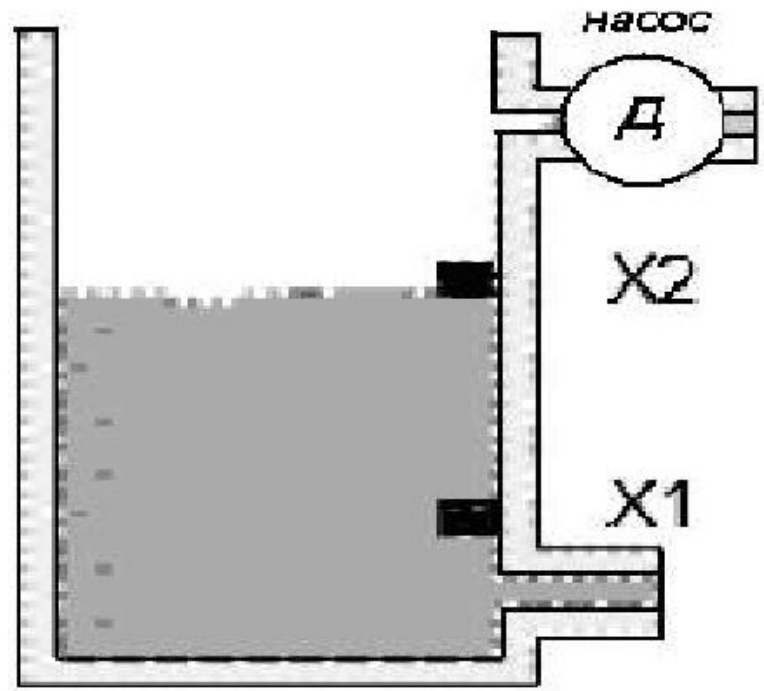


Рисунок 11.14 – Бак с водой

11.3 Программирование счетчика. Команда COUNTER

Счетчик – самое простое устройство в контроллере, так как предназначен только для одного – считать. Конечно, в реальности в зависимости от модели контроллер может поддерживать различные варианты использования данной функции:

- суммирующие счётчики (прямой счет 1, 2, 3 ...);
- обратные счётчики (счет вниз 3, 2, 1 ...);
- счет вверх-вниз (счет вниз 1, 2, 3, 4, 5, 4, 3, 2, 3, 4, 5 ...).

Возможность использования того или другого типа счетчика относится только к возможностям конкретной модели контроллера.

Дополнительно счетчики делятся по способу обработки импульсов на два основных типа:

- Программные – напрямую зависят от быстродействия контроллера и не могут работать быстрее скорости обработки двух программных циклов (при использовании программного счетчика допускается, что быстродействии счетчика не превышает «*Времени цикла обработки*» X2. В противном случае необходимо использовать высокоскоростные аппаратные счетчики);
- Аппаратные – не зависят от быстродействия контроллера и могут работать быстрее времени обработки одного программного

Для правильного использования счетчиков необходимо определить для себя всего 3 вещи:

- **С какой частотой мы хотели бы считать.** Так как обычно, использование *программного счетчика* допустимо для любого из входов, то при выборе *аппаратного счетчика* мы можем использовать только те входы, которые служат для высокоскоростного счета. Для определения возможности использования того или иного входа в качестве высокоскоростного необходимо сверяться с руководством пользователя или с руководством по программированию;
- **До какого значения мы собираемся считать импульсы.** Диапазон, в котором может считать контролер 0 ... 32.767, -32.768 ... -32.768, 0 ... 65535, ... зависит только от конкретной модели контроллера;
- **По какому условию мы можем остановить счет.**

11.4 Программирование таймера. Команда TIMER

Попробуем классифицировать, какими бывают таймеры:

- ***с задержкой по включению.*** Другими словами, после того, как Вы нажали на выключатель (т.е. придя поздно вечером домой и нажав на выключатель, вы с удивлением обнаруживаете, что свет загорается только через одну минуту после включения выключателя);
- ***с задержкой по выключению.*** Иначе, после того, как Вы выключили свет (т.е. уходя поздно вечером из дома и нажав на выключатель, вы с удивлением обнаруживаете, что свет в квартире погас только спустя некоторое время);
- ***Накапливающий таймер*** – Этот тип таймера позволит Вам выключить свет только после того, как суммарное время включения достигнет определенного значения. Данный тип таймера требует обязательного использования двух входов.

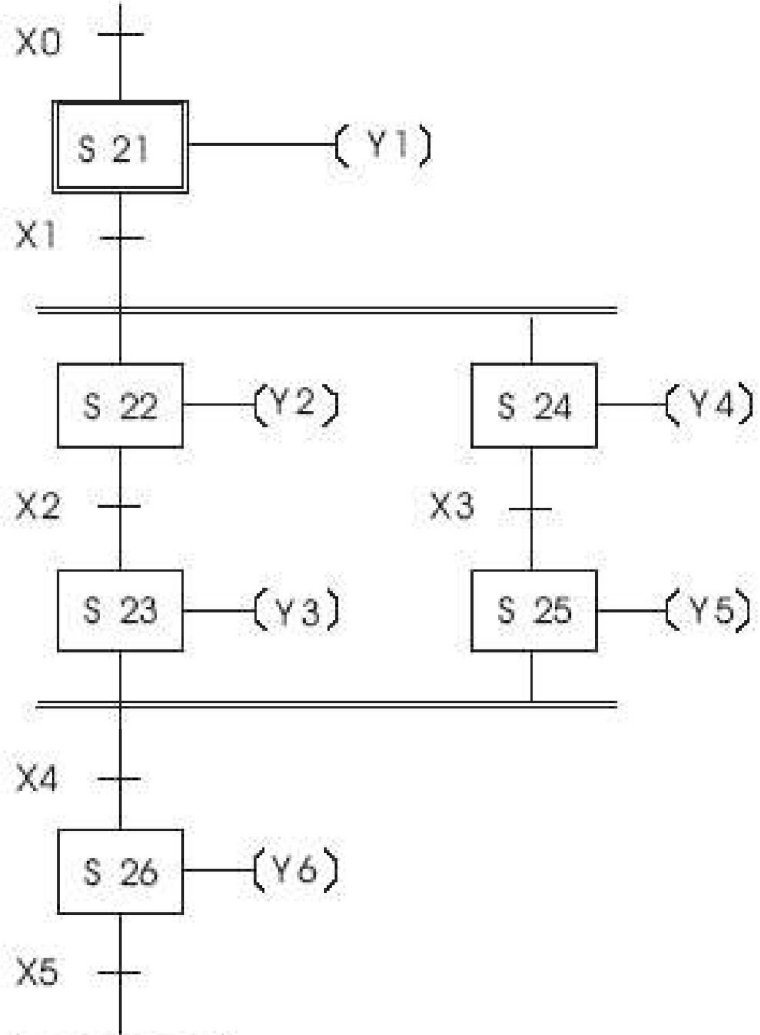
Что мы должны определить для себя – это всего лишь 2 вещи:

- Какой из входов запустит таймер;
- Какую задержку времени установить, т.е. сколько времени пройдет, прежде чем включится-выключится выход.

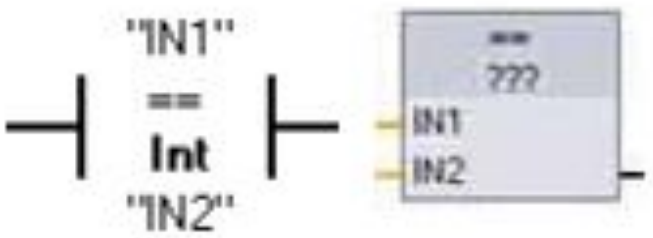
С того момента, когда все команды перед символом таймера примут значение – «истина», таймер начинает отсчёт времени и по достижении установленного значения переключит состояния выхода с «включено» на «выключено» или наоборот. В любой момент времени работы таймера есть возможность отображать его текущее состояние. Диапазон, в котором может работать контроллер (0 ... 32.767, -32.768 ... -32.768, 0 ... 65535) зависит только от конкретной модели контроллера.

12.8.4.3 Параллельное разветвление

При параллельном разветвлении два или несколько процессов состояний обрабатываются одновременно. Из одного состояния разветвление может создавать несколько (максимум 8) процессов состояний.



Сравнение



LAD FBD

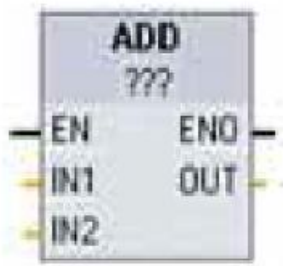
Команды сравнения используются для сравнения двух величин, относящихся к одному и тому же типу данных. Если сравнение в виде контакта в LAD имеет значение ИСТИНА, то контакт активизирован. Если сравнение в виде блока в FBD имеет значение ИСТИНА, то выход блока имеет значение ИСТИНА.

Щелкнув на команде в программном редакторе, вы можете выбрать тип сравнения и тип данных из ниспадающих меню.

Тип отношения	Сравнение истинно, если:
==	IN1 равно IN2
<>	IN1 не равно IN2
>=	IN1 больше или равноIN2
<=	IN1 меньше или равно IN2
>	IN1 больше, чем IN2
<	IN1 меньше, чем IN2

6.1.5 Арифметические команды

Команды сложения, вычитания, умножения и деления



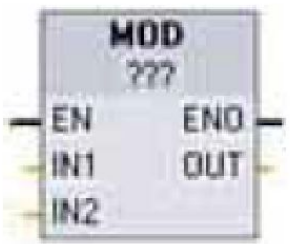
Блоковые арифметические команды используются для программирования основных арифметических операций:

- ADD: Сложение (IN1 + IN2 = OUT)
- SUB: Вычитание (IN1 - IN2 = OUT)
- MUL: Умножение (IN1 * IN2 = OUT)
- DIV: Деление (IN1 / IN2 = OUT)

При целочисленном делении дробная часть частного отбрасывается, что приводит к появлению целочисленного выходного значения.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

Команда MOD (получение остатка от деления)



Команда MOD (modulo) используется для выполнения операции IN1 modulo IN2.
Операция IN1 MOD IN2 = IN1 - (IN1 / IN2)*IN2 = параметр OUT.

Щелкните под именем блока и выберите тип данных из ниспадающего меню.

6.1.6 Команда Move

Команды передачи и блочной передачи

Команды передачи используются для копирования элементов данных в новый адрес в памяти и преобразования из одного типа данных в другой. При этом источник данных не изменяется.

- MOVE: Копирует элемент данных, хранящийся по определенному адресу в новый адрес
- MOVE_BLK: Прерываемая передача, которая копирует блок элементов данных в новый адрес
- UMOVE_BLK: Непрерываемая передача, которая копирует блок элементов данных в новый адрес

6.1.7 Преобразование

Команда преобразования

Команда CONVERT преобразует элемент данных из одного типа данных в другой. Щелкните под именем блока, а затем выберите типы данных для IN и OUT из ниспадающего списка.

После выбора типа данных источника (преобразовать из) в ниспадающем списке отображаются возможные преобразования (преобразовать в). Преобразования из и в BCD16 ограничены типом данных Int. Преобразования из и в BCD32 ограничены типом данных DInt.

Щелкните под именем блока и выберите типы данных из ниспадающих меню.