MITSUBISHI ELECTRIC

Программирование ПЛК

Москва 2000-2001 г.
Автор Яковлев Олег Геннадьевич

Введение

Необходимость использования контроллеров назрела в начале 1960-ых. Когда промышленность начала предъявлять высокие требования к эффективному использованию производственных мощностей, а существующие решения на основе релейно-контактных схем не могли обеспечить гибкое и эффективное управление технологическими процессами, так как изменение технологических циклов требовало замены большого числа элементов управления и контроля. Громоздкость и ограниченный срок службы реле требовал создания сложных систем контроля, а поиск неисправности среди 1000 реле требовал содержания большого числа специалистов. Создание промышленных контроллеров позволило объединить сотни - тысячи реле, таймеров, счетчиков в единый и компактный модуль. Возможность перепрограммирования позволила предприятиям быстро перестраивать производство в соответствии с требованиями рынка. Требования к управлению на расстоянии начали появляться приблизительно в 1973. С момента, когда Программируемые Логические Контроллеры(ПЛК) получили возможность управлять другим ПЛК и могли находиться далеко от оборудования, которым они управляли, - вопрос о необходимости перехода на повсеместное использование контроллеров стал очевидным для всех.

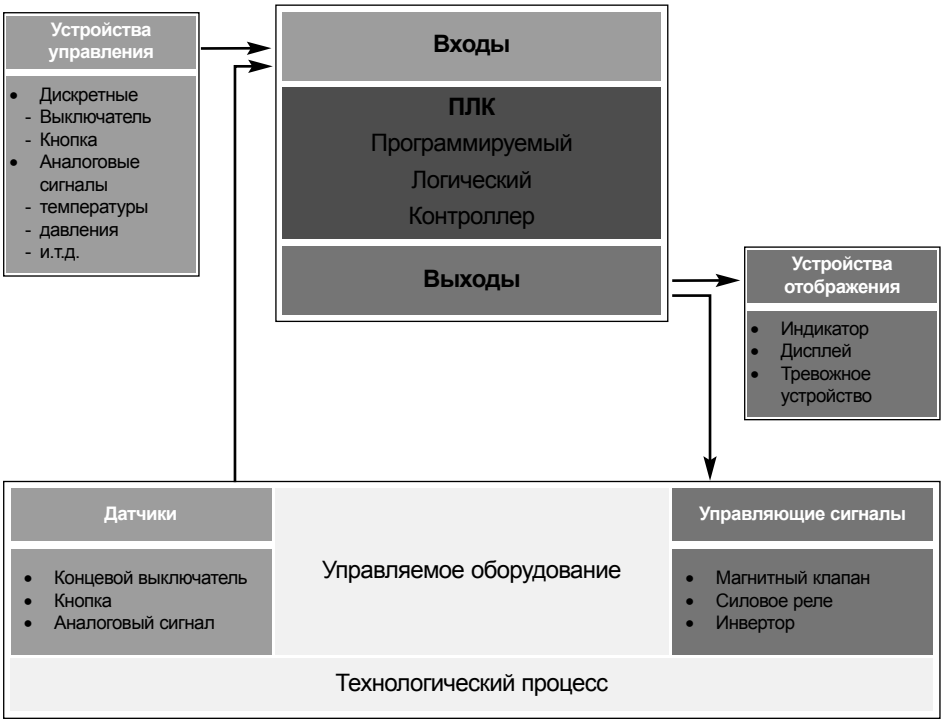
Программируемый Логический Контроллер - устройство, которое было изобретено для замены релейно-контактных схем. ПЛК опрашивает входы(выключатели, датчики и.т.д.) и в зависимости от их состояния(Включено - 1, Выключено - 0), включает-выключает выходы (исполнительные механизмы). Используя программное обеспечение, пользователь имеет возможность программировать ПЛК или вносить изменения в уже существующую программу.

Программируемый Логический Контроллер может использоваться везде там, где есть производство - любая задача, которая требует использования электрических устройств управления, имеет потребность в ПЛК. Если в Вашем производстве используются - механическая обработка, упаковка, транспортеры, конвейеры, автоматизированные линии и т.д. Вы вероятно уже используете ПЛК, если считаете свои деньги и время.

Например: предположим, что при включении выключателя нам необходимо запустить электропривод на 15 секунд, а затем выключить независимо от того, как долго выключатель включен. С помощью таймера мы можем легко решить эту задачу, но если для решения технологического процесса необходимо включить 10 выключателей и электроприводов? Нам потребуется 10 таймеров, а для расчета числа циклов включения - выключения нам понадобится такое же количество внешних счетчиков. Использование одного контроллера позволит легко решить эту простую задачу, а возможность изменения программы даст возможность максимально быстро менять технологический процесс в зависимости от текущей задачи.

Что такое ПЛК.

1.1 Назначение

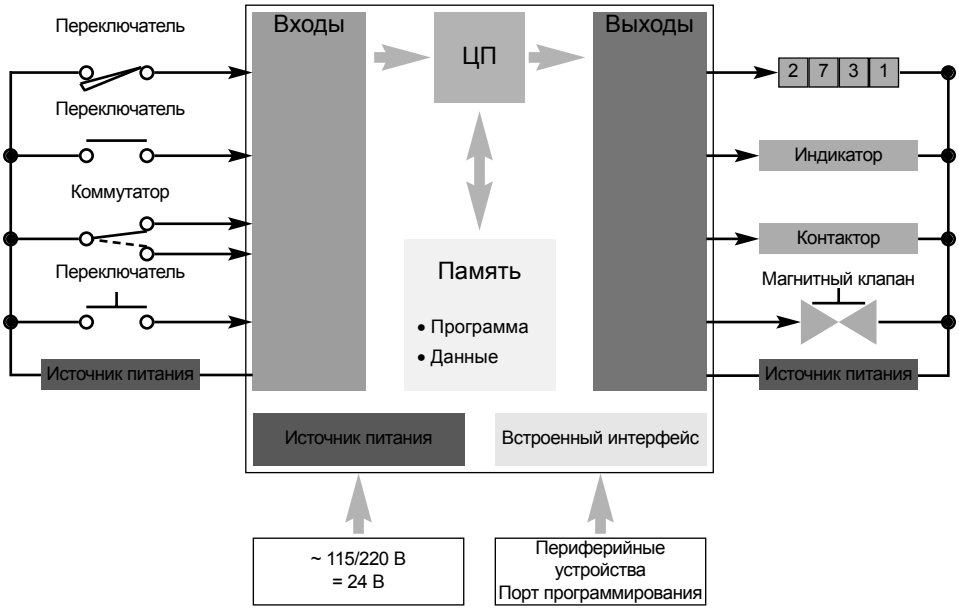


Базовые блоки контроллера могут иметь различные варианты исполнения с:
- питанием ~ 230 В или =12/24 В;
- релейными или полупроводниковыми выходами:
- количеством встроенных входов-выходов от 6 до 128;
- возможностью объединия контроллеров в сеть;
- встроенным интерфейсом RS232/RS422;
- поддержкой тригонометрических функции, встроенного сопроцессора;
- встроенным ПИД регулятором;
- часами реального времени;
- поддержкой счетных входов до 50 КГц и импульсного выхода до 20 КГц;
- аналоговых сигналов ввода/вывода, Pt100, ± 10В, до 12 Бит;
- возможностью наращивания системы.

В любом случае структура контроллера остается неизменной, и выбор модели определяется только требованиями к технологическому процесса, а широкий ряд моделей позволит вам подобрать контроллер с оптимальным соотношением цена - производительность.

1.2 Структура контроллера

Программируемый Логический Контроллер(ПЛК) - главным образом состоит из ЦП(Центрального процессора), области памяти, и функций обработки сигналов ввода - вывода. Можно считать, что ПЛК - это сотни или тысячи отдельных реле, счетчиков, таймеров и память. Все эти счетчики, таймеры, и т.д. физически не существуют, а моделируются ЦП и предназначены для обмена данными между встроенными функциями счетчиками, таймерами.



- *ВХОДЫ* - обеспечивают связь с внешним миром. Физически существуют и получают сигналы от выключателей, датчиков, и т.д.
- *ВНУТРЕННИЕ РЕЛЕ* - физически не существуют. Они моделируются ЦП - и предназначены только для обеспечения работы программы.
- *СЧЕТЧИКИ* - физически не существуют. Существует в двух вариантах:
  - Программные - скорость счета зависит от объема программного кода;
  - Аппаратные - счет идет независимо от выполнения программы.
- *ТАЙМЕРЫ* - физически не существуют. Предназначены для установки времени задержки включения - выключения.
- *ВЫХОДЫ* - обеспечивают связь с внешним миром. Физически существуют и обеспечивают включение - выключение исполнительных механизмов. Существуют два варианта исполнения реле, транзисторы.
- *ПАМЯТЬ* - предназначена для хранения программы и данных.

1.3 Работа программы

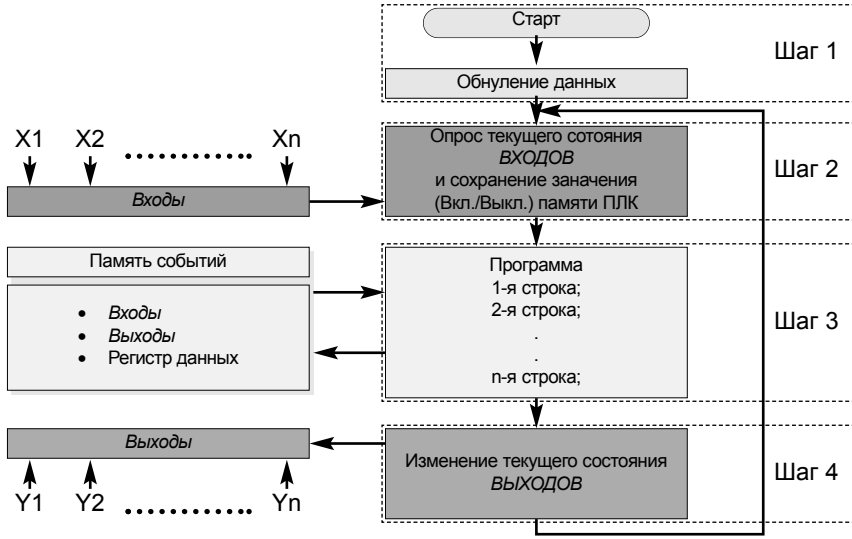
В процессе работы ПЛК непрерывно опрашивает текущее состояние входов и в соответствии с требованиями к производственному процессу изменяет состояние выходов(Вкл./Выкл). Мы можем разделить этот цикл на четыре основных шага. Все остальное может рассматриваться нами, как часть кода, необходимое для согласования между первым и четвертым шагом.

Шаг первый - инициализация системы. Необходимо помнить, что в случае сбоев по питанию или при выключении контроллера система обязана вернуться в исходное состояние. Не следует недооценивать важности этой части программного кода, так как в противном это может привести к сбоям и поломкам оборудования.

Шаг второй - проверка текущего состояния *ВХОДОВ*. ПЛК проверяет текущее состояние входов и в зависимости от значения(Вкл./Выкл) выполняет последовательные действия. Состояние любого из *ВХОДОВ* сохраняется в памяти (в области данных) и может в дальнейшем использоваться при обработке третьего шага программы.

Шаг третий - выполнение программы. Будем считать, что в ходе технологического процесса переключился *ВХОД(X1)* с выключено на включено, и в соответствии с технологическим процессом нам необходимо изменить текущее состояние *ВЫХОДА(Y1)* с выключено на включено. Так как ЦП опросил текущее состояния всех *ВХОДОВ* и хранит их текущее состояние в памяти, то выбор последующего действия обусловлен только ходом технологического процесса.

Шаг четвертый - изменение текущего состояния *ВЫХОДА*. ПЛК - изменяет текущее состояние *ВЫХОДОВ* в зависимости от того какие *ВХОДЫ* являются выключенными, а какие включенными исходов из хода вашей программы в течение третьего шага. То есть контроллер, физически переключил *ВЫХОД(Y1)* и включились исполнительные механизмы лампочка, двигатель и т.д. После этого следует возврат на Шаг второй.

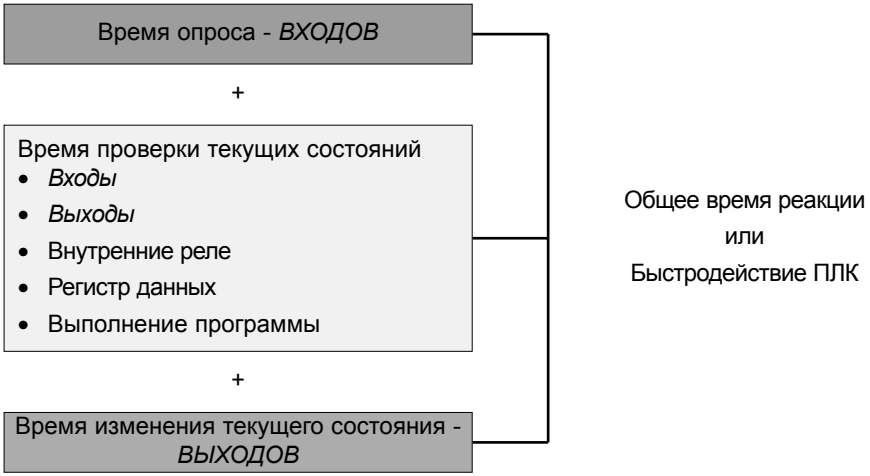


1.4 Время реакции - Быстродействие

Быстродействие Программируемого Логического Контроллера - это основополагающий фактор, влияющий на выбор ПЛК.

Примером быстродействия может служить, то время, которое требуется Вам для осмысления ответа на поставленный вопрос, подобно времени, необходимого человеку для принятия решения. Так и ПЛК - необходимо некоторое время для обработки текущего состояния. Для некоторых приложения - этот параметр может не являться критическим, и выбор того или иного значения быстродействия должен сопоставляться с реальными требованиями приложения.

Наиболее актуальным примером демонстрации времени реакции можно считать время с начала употребления алкоголя до времени прихода опьянения. Этот время зависит от многих факторов конкретного человека веса, психического состояния и т.д. Так и для контроллера время реакции зависит, как от числа опрашиваемых *ВХОДОВ/ВЫХОДОВ*, так и от производительности самого процессора. Давайте попробуем проанализировать эту банальную ситуации. С момента употребления алкоголя, до момента наступления стадии опьянения прошло определенное количество времени, которое мы можем назвать временем реакции …



1.5 Действительное - Быстродействие (ПЛК)

Теперь, когда мы получили представление о том как работает контроллер, давайте посмотрим, что происходит в действительности при обработке Программируемым Логическим Контроллером текущих состояний *Входов* и как - это влияет на время срабатывания *Выходов*.

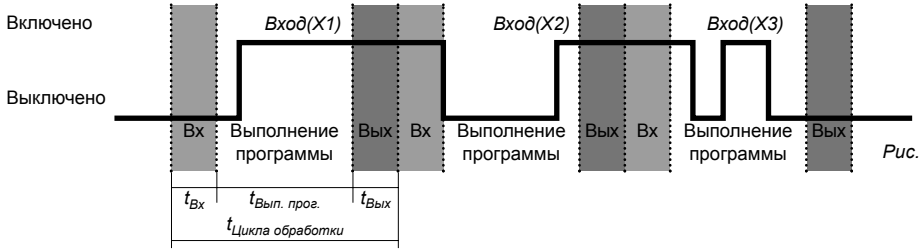


Рис. 1

$t_{Вх}$ = время опроса входов;

$t_{Вып. прог}$ = время выполнения программы;

$t_{Вых}$ = время переключения выходов;

$t_{Цикла\ обработки}$ = время цикла обработки ($t_{Вх}$+$t_{Вып. прог}$+$t_{Вых}$).

На рисунке 1 Вы видите:

1 ПЛК не зафиксировал переключение *Входа(X1)*. Это произошло потому, что переключение произошло после обработки контроллером состояний *Входов*. А значит, контроллер сможет выполнить операцию, соответствующую *Входу(X1)* только на следущим цикле опроса.

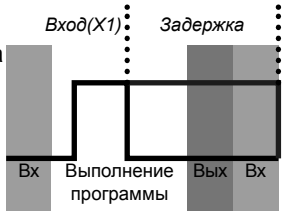
2 ПЛК не зафиксировал переключение *Входа(X2)*. Следовательно, если котроллер должен был выключить *Выход(Y1)* при одновременном включении *Входа(X1)* и *Входа(X2)*, то это событие не было обработано контроллером и операция не выполненна.

3 ПЛК не зафиксировал переключение *Входа(X3)*. И, возможно, не сможет обработать изменение состояния данного *Входа(X3)* никогда.

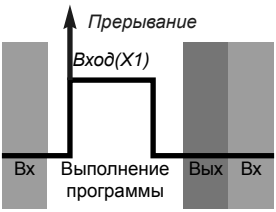
Есть три варианта решения этой проблемы:

1 Использовать контроллер с более высоким быстродействием;

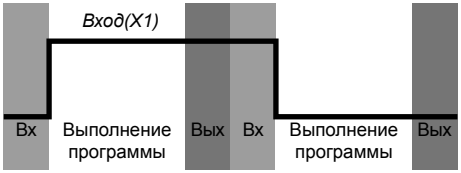
2 Воспользоваться функцией задержки времени Выкл. Эта функция увеличит длительность входного сигнала;



3 Использовать функцию обработки прерываний. То есть, как только *Вход* включен, то независимо от того этапа, на котором в настоящий момент находится программа ПЛК, немедленно останавливает выполнение основной программы и выполняет подпрограмму прерывания.



Таким образом, можно считать, что оптимальным временем продолжительности импульса будет являться время прохода одного цикла программы.



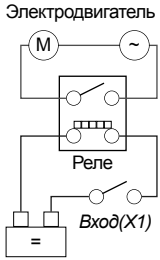
2. Основы программирование ПЛК

2.1. Реле назначение

Теперь, когда мы имеем представление об основных принципах работы Программируемого Логического Контроллера, как ПЛК обрабатывает *Входы*, *Выходы*, и как выполняется программа, мы готовы начать писать программу. Но для начала давайте вспомним, о том как работает реле, и что реле из себя представляет. Так как целью использования контролера и является физическое замещение реле.

В принципе реле - это выключатель, подобно тем выключателям, которыми мы пользуемся повседневно, включая-выключая свет в наших комнатах. Только, если дома мы используем физическую силу для включения-выключения, то промышленное реле использует электромагнитное поле. Подайте напряжение на катушку, и получите магнитное поле - это магнитное поле всасывает контакты реле, следствием чего является соединение двух контактов. В сущности основное назначение реле дать возможность току протекать между двумя контактами.

Давайте рассмотрим простой пример, в котором нам необходимо включить электродвигатель, при нажатии на выключатель *Вход(X1)*.



В результате мы получим устройство, которое можно условно разделить на три основные части:

- выключатель *Вход(X1)*;
- реле;
- электродвигатель.

Всякий раз, когда выключатель *Вход(X1)* соединяет контакты (Включается), мы закрываем цепь и ток, создавая электромагнитное поле в реле, переключает контакты реле, т.е. пускает электродвигатель.

В данном примере мы использовали типичное индустриальное реле постоянного тока для управления включением-выключением устройств. Пока *Вход(X1)* открыт, ток не может течь сквозь катушку реле и электродвигатель не работает. Но как только *Вход(X1)* закрыт, ток создает электромагнитное поле, которое заставляет контакты реле соединятся. В результате ток, протекающий сквозь реле, заставляет вращаться электродвигатель ...

2.2. Реле и контроллер

Давайте попробуем заменить реле контроллером. Конечно, данный пример не является показателем эффективности использования ПЛК в категории цена-производительность. Но он позволит Вам понять, в каких случаях и для чего Вы можете эффективно использовать контроллер. Первое, что нам необходимо осознать, каким образом мы можем объяснить ПЛК, то какую задачу он должен выполнить.

Базовым языком программирования ПЛК по сегодняшний день остается язык *Релейно-контактных схем*.

Шаг первый: мы должны переопределить все составляющие оборудования, которые мы используем в символы понятные для контроллера. Так как ПЛК ничего не знает о существовании таких вещей, как выключатель, реле, электродвигатель и т.д. ПЛК может работать с символами *Вход*, *Выход*. Для контроллера совершенно не важно, что из себя физически представляет *Вход* или *Выход.* контроллер обрабатывает только текущее состояние *Входа*(включено-выключено). Все остальные действия выполняются последовательно и только в строгом соответствии с алгоритмом, заложенным в контроллер Вами.

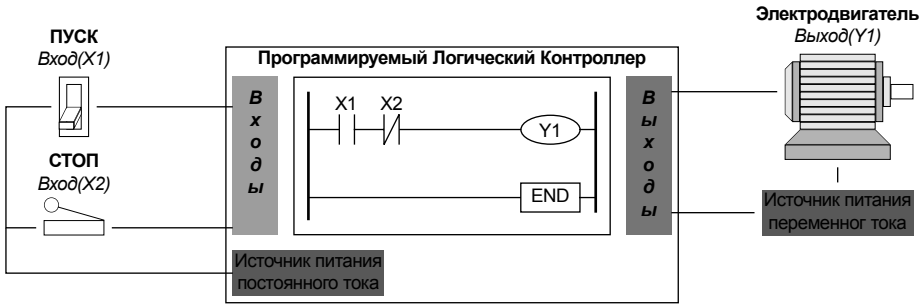
Шаг второй: сначала мы заменяем источник питания. Для языка *Релейно-контактных схем* - этим символом будет являться две паралельные прямые с левой и правой стороны диаграммы. Можно считать, что левая линия является “+”, а правая линия “-”.

Шаг третий: затем мы присваиваем символы *Входам*. В нашем примере мы имеем два входа:

- *Вход(X1)* ⊣ ⊢ - нормально открытый контакт;
- *Вход(X2)* ⊣/⊢ - нормально закрытый контакт.

Шаг четвертый: последнее мы присваиваем символ *Выходам*.

- *Выход(Y1)* —(Y1)— - символ катушки.



В результате мы получили программу, которая можент быть выполннена ПЛК.

2.3. Основные команды контроллера

Входы

Команда (LD) - нормально открытый контакт.

Прочитав этот сигнал, контроллер начинает постоянно проверять состояние *Входа(X1)*. 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.

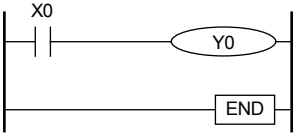
Команда (LDI) - нормально закрытый контакт.

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.

Логическое состояние	***LD***	***LDI***
0	*Ложь*	*Истина*
1	*Истина*	*Ложь*

Пример использования - *LD*

Язык релейно-контактных схем

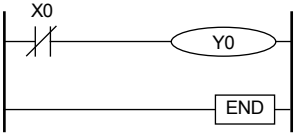


Список инструкций

```
LD      X0
OUT     Y0
END
```

Пример использования - *LDI*

Язык релейно-контактных схем



Список инструкций

```
LDI     X0
OUT     Y0
END
```

Выходы

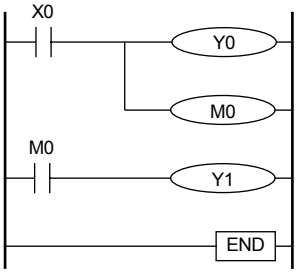
Команда (OUT) - инициализация *Выхода*.

Прочитав этот сигнал, контроллер изменит состояния *Входа(Y1)*, с выключено на включено. Также, как и в случае с Входами данный символ может относиться не только к физическим Выходам контроллера, а также и к внутренним(вспомогательным) реле. Число и ограничения по использованию внутренних реле определяется только возможностями контроллера.

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

Пример использования - *LDI*

Язык релейно-контактных схем



Список инструкций

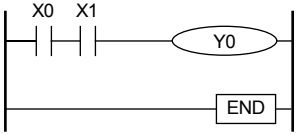
```
LD      X0
OUT     Y0
OUT     M0
LD      M0
OUT     Y1
END
```

Команда (AND) - логическое умножение

Операция логического умножения. В языках программирования и языках запросов обозначается символами AND, И, & и другими способами. Результатом операции является "истина", если оба операнда принимают значение "истина", и "ложь" - в остальных случаях.

Пример использования - *AND*

Язык релейно-контактных схем



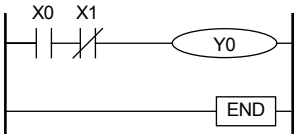
Список инструкций

```
LD      X0
AND     X1
OUT     Y0
END
```

Команда (ANI) - отрицание логического умножения

Пример использования - *ANI*

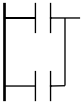
Язык релейно-контактных схем



Список инструкций

```
LD      X0
ANI     X1
OUT     Y0
END
```

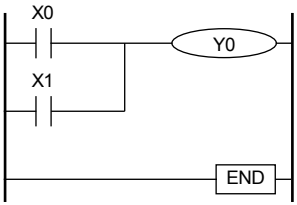
Команда (OR) - логическое сложение



Операция логического умножения. В языках программирования и языках запросов обозначается символами OR и другими способами. Результатом операции является "истина", если оба или один из операндов принимают значение "истина", и "ложь" - в остальных случаях.

Пример использования - *OR*

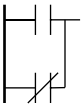
Язык релейно-контактных схем



Список инструкций

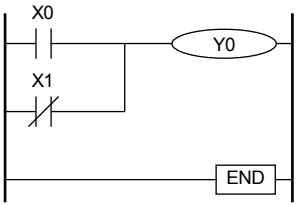
```
LD      X0
OR      X1
OUT     Y0
END
```

Команда (ORI) - отрицание логического сложения



Пример использования - *ORI*

Язык релейно-контактных схем

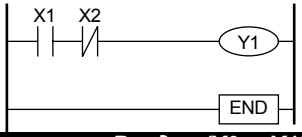


Список инструкций

```
LD      X0
ORI     X1
OUT     Y0
END
```

2.4. Регистры в контроллере

Давайте вернемся к предыдущему примеру и посмотрим, как контроллер обрабатывает состояния *Входов-Выходов*, и хранит полученные значения.



Регистры Входов (X0 ... X15)															
X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
													1	0	

Регистры Выходов (Y0 ... Y15)															
Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
														0	

В таблицах Вы можем видеть, что контрооллер хранит полученные значения в регистрах:

Входов

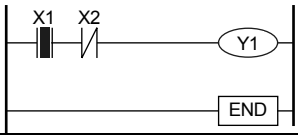
- в регистре X1 находится значение *"0"* (Бит *"0"*), то есть *Вход(X1)* - выключен;
- в регистре X2 находится значение *"1"* (Бит *"1"*) следовательно, *Вход(X2)* - включен.

Выходов

- в регистре Y1 находится значение *"0"* (Бит *"0"*), то есть *Выход(Y1)* - выключен.

В действительности во всех остальных ячейках находится значение *"0"*, но мы не отображаем данное значение, чтобы Вы могли сосредоточиться на данном примере.

Давайте посмотрим на то, как изменятся значения в регистрах после того, как мы включим выключатель *Вход(X1)*:



Входы (X0 ... X15)															
X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	X0
													1	1	

Выходы (Y0 ... Y15)															
Y15	Y14	Y13	Y12	Y11	Y10	Y9	Y8	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
														1	

Электродвигатель заработал и технологический процесс пошёл ...

Попробуем систематизировать полученную информацию и составим таблицу, в которой будут описаны все возможные состояния для двух *Входов* и одного *Выхода*.

Входы (X1/X2)		***Выход(Y1)***	***Регистры***		
LD(X1)	***LDI(X2)***	***OUT(Y1)***	***LD(X1)***	***LDI(X2)***	***OUT(Y1)***
Ложь	Истина	Ложь	0	0	0
Истина	*Истина*	*Истина*	*1*	*0*	*1*
Истина	Ложь	Ложь	1	1	0
Ложь	Ложь	Ложь	0	1	0

Как мы можем видеть, контроллер выполнит операцию включения *Выхода(Y1)* - тогда когда *Вход(X1)* и *Вход(X2)* будут в состоянии *Истина*.

Данный пример показывает нам как работает логика *AND(И)*, т.е. *Выход(Y1)* - изменит свое состояние *“0”* на *“1”* тогда и только тогда когда *Вход(X1)* и *Вход(X2)* будут в состоянии *Истина*. Во всех остальных случаях *Выход(Y1)* будет в состоянии *“0”*. Попробуем запомнить это.

2.4. Использование внутреннего реле (контроль уровня)

Теперь, когда мы получили представление о работе контроллера, давайте немного расширим наш пример, и попробуем решить более сложную задачу.

Предположим нам необходимо постоянно поддерживать определенный уровень в баке с водой. Что нам необходимо для решения этой задачи:

- 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 нашему сожалению, использование логики, для двух Входов не даст желаемого результата, так как в случае:

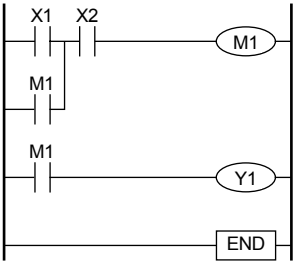
- если мы будем включать насос, когда оба входа окажутся в состоянии выключено, то насос будет поднимать уровень только до нижнего датчика или никогда уровень воды не опустится до уровня нижнего датчика. Т.е. начнет работать в очень жестком режиме постоянного включения выключения.

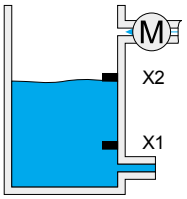
Для обеспечения работы насоса в оптимальном режиме работы нам понадобится включение в задачу дополнительного элемента, который должен обеспечить:

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

Для этого мы будем можем использовать внутренне реле контроллера *M1*.

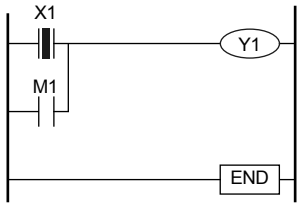
Что мы получаем в результате:



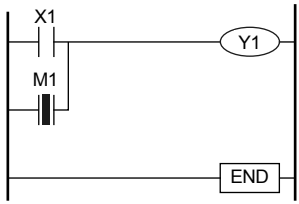


- Когда вода опустится ниже уровня второго датчика, то оба *Входа(Вход(X1)* и *Вход(X2))* изменят свое состояние с *"0"* на *"1"*, за этим последует изменение состояния внутреннего реле *M1* с *"0"* на *"1"*, и включение насоса;
- Когда вода поднимется до уровня второго датчика, то *Вход(X2)* будет в состоянии *"1"*, а *Вход(X1)* изменит свое состояние с *"1"* на *"0"*, и в результате насос будет продолжать работать, так как внутреннее реле будет в состоянии *"1"*, до тех пор пока вода не достигнет уровня верхнего датчик и не изменит состояние *Вход(X2)* с *"1"* на *"0"*, за этим последует изменение состояния регистра *M1* с *"1"* на *"0"*, и выключение насоса.

Данный пример демонстрирует нам не только необходимость и важность использования внутренних реле контроллера(количество внутренних реле зависит только от модели контроллера и позволяет Вам решать ваши задачи управления с минимальным числом внешних устройств, что и является основной задачей контроллера), а также демонстрирует работу логики *OR(ИЛИ)*.



Т.е. *Выход(Y1)* - изменит свое состояние с *"0"* на *"1"* тогда и только тогда когда один из элементов управления *Вход(X1*) или Внутреннее реле*(M1*) будут в состоянии *Истина*. Во всех остальных случаях *Выход(Y1)* будет в состоянии *"0"*. Попробуем запомнить это.

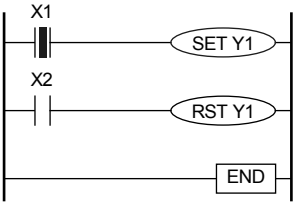


2.5. Команды SET(Установить)/RST(Сбросить)

Если вы осознали то, о чем говорилось в предыдущих главах нашего курса, и можете самостоятельно запрограммировать контроллер используя команды *AND* или *OR* Вы можете перейти к решению более сложных задач. В противном случае перечитайте предыдущие главы.

Пример использования - *SET/RST*

Язык релейно-контактных схем



Список инструкций

```
LD      X1
SET     Y1
LD      X2
RST     Y1
END
```

2.6. 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 ...),

  а 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"* * 2 . В противном случае необходимо использовать высокоскоростные - аппаратные счетчики);
- аппаратные - не зависят от быстродействия контроллера и могут работать быстрее времени обработки одного программного цикла(с частотой до 100 КГц). Мы можем считать, что данный тип счетчиков физически существует;

  но независимо от реализации они являются всего лишь счетчиками. счетчиками и счетчиками … Независимо от того - программные или аппаратные счетчики выбор должен отределяться только тем, с какой скоростью будет работать *счетный вход* и позволяет ли быстродействие контроллера обработать сигналы с датчика или нет.

Для правильного использования Счетчиков мы должны 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.**

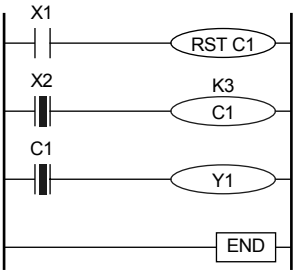
Команда (OUT Cn Km) - инициализация счетчика.
C - обозначение счетчика;
n - число от 1 до 256 номер счетчика;
K - обозначение константы;
m - число от 1 - 2.147.483.648, до которого будет вестись счет.

Вспомним наш первый пример и попробуем решить следующую задачу, пусть насос *Выход(Y1)* включится только после того как, мы три раза *включим-выключим Вход(X2)*.

То есть при *включении*:

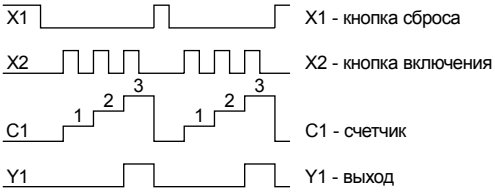
- *Входа(X1)* - *Счетчик(C1)* установится в значение *“0”*;
- Первое нажатие на *Вход(X2)* - *Счетчик(C1)* сравнит *“1”* = *К3(“3”)* нет, то запомнит *“1”*;
- Второе нажатие на *Вход(X2)* - *Счетчик(C1)* прибавит *“1”* сравнит *“2”* = *К3(“3”)* нет, то запомнит *“2”*;
- Третье нажатие на *Вход(X2)* - *Счетчик(C1)* прибавит *“1”* сравнит *“3”* = *К3(“3”)* да, то сменит состояни С1 с *“0”* на *“1”* и насос заработает;
- Повторное *включение Входа(X1)* - *Счетчик(C1)* установится в значение *“0”* и цикл начнется с начала.

РКС



Последовательность инструкций

```
LD      X1
RST     C1
LD      X2
OUT     C1      K3
LD      C1
OUT     Y1
END
```



2.7. Команда TIMER

Что такое Таймер? Ничего сложного, если Вы поняли, как работает счетчик. Для того чтобы представить для себя как работает таймер нужно только заменить число циклов *Включения-Выключения* на число секунд, после которых следует выполнить команду *Включения-Выключения Выхода*.

Когда мы видим все то разнообразие таймеров, используемых разными производителями … Стоп, пожалуйста, немного подумайте, и Вы поймете что для того чтобы понять как работает та или иная функция таймера, Вам достаточно внимательно прочесть документацию к контроллеру.

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*** - Т.е. 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 в 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:

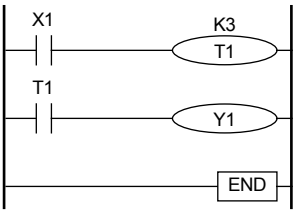
- Какой из *Входов* запустит таймер;
- Какую задержку времени мы установим. Т.е. сколько времени пройдет, прежде чем *Включим - Выключим Выход*.

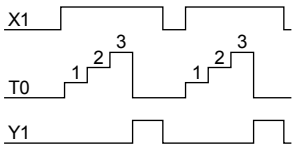
С того момента, когда все команды перед символом *Таймера* примут значение - *Истина*, *Таймер* начинает отсчёт времени и по достижении установленного значения переключит состояния *Выхода* с *Включено на Выключено* или наоборот. В любой момент времени работы *Таймера* Вы имеете возможность, отображать текущее состояние *Таймера*. Диапазон в котором может работать контролер 0 ... 32.767, -32.768 ... -32.768, 0 ... 65535, (секунд, ...) ... зависит только от конкретной модели контроллера.

Обычно различные модели контроллеров оперируют с понятием 10 или 100 ms(мсек) - миллисекунда или 1/1000 от одной секунды (10 мсек = 0.01 сек., 100 мсек = 0.1 сек.) Правила и возможность использования определяются *Документацией по Программированию* контроллера. Так же как и в случае со *Счетчиками* некоторые модели контроллеров позволяют использовать *Высокоскоростные таймеры*, но независимо от реализации они являются всего лишь *Таймерами*, *Таймерами* и *Таймерами* … И поняв принцип работы простейшего *Таймера*, Вы сможете легко программировать и все остальные.

Команда (OUT Tn Km) - инициализация таймера.
T - обозначение таймера;
n - число от 1 до 256 номер таймера;
K - обозначение константы;
m - число от 1 - 32.768, до которого будет вестись отсчет времени.

Пример 1: Таймер - с задержкой по включению



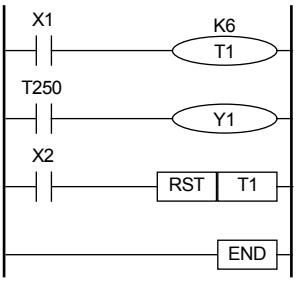


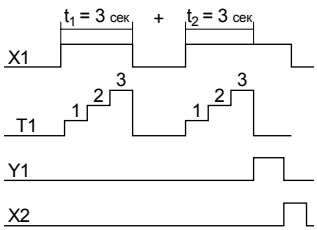
Вспомним наш любимый пример и попробуем решить следующую задачу, пусть насос *Выход(Y1)* включится только после того как, пройдет три секунды после *включения Входа(X1)*:

- *Вход(X1)-Включился*: *Таймер(T1)* начал отсчет времени;
- Прошло три секунды: *Выход(Y1)-Включился*;
- *Вход(X1)-Выключился*: *Выход(Y1)-Выключился*;
- *Вход(X1)-Включился*: *Таймер(T1)* начал отсчет времени;
- Прошло три секунды: *Выход(Y1)-Включился*;
- *Вход(X1)-Выключился*: *Выход(Y1)-Выключился*.

Как Вы понимаете шаг мог бы быть и 0,1 мсек или 0,01 мсек или ...

Пример 2: Таймер - с накоплением





Пусть насос *Выход(Y1)* включится только после того как, он проработает втечении шести секунд после *включения -выключения-включения Входа(X1)*:

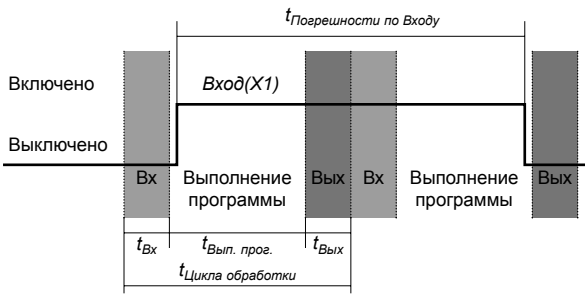
- *Вход(X1)-Включился*: *Таймер(T1)* начал отсчет времени;
- Прошло три секунды: *Выход(X1)-Выключился*;
- Таймер запомнил время работы *Входа(X1)* - t_1 = 3 сек;
- *Вход(X1)-Включился*: *Таймер(T1)* продолжил отсчет времени, как только T = t_1+t_2 = 6 сек. то *Выход(Y1)-Включился*;
- *Вход(X2)-Включился*: *Таймер(T1)* изменил свое состояния с “1” на “0” и *Выход(Y1)-Выключился*.

2.7.1 TIMER - погрешность

Теперь, когда мы научились использовать *Таймеры*, вернемся назад и вспомним, как работает контроллер - Что такое быстродействие? Конечно, если Вы используете *Таймер* с шагом 1 секунда, то Вы можете не беспокоится, но когда мы используем *Таймеры*, которые работают с шагом от 0,001 секунды, мы обязаны помнить о быстродействии контроллера.

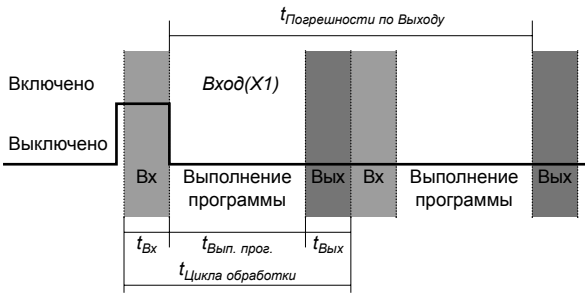
Рассмотрим два типичных случая, в которых Вы обычно не учитываете погрешность при использовании таймера:

- погрешность по *Входу*;
- погрешность по *Выходу*.



Погрешность по *Входу* означает, - что с момента когда Вы *Включите Вход(X1)* до начала работы *Таймера*, пройдет время равное:

$t_{Вып.\ прог.}*2+t_{Вх}+t_{Вх}= t_{Погрешности\ по\ Входу}$.



Погрешность по *Выходу* означает - что с момента когда Вы Выключите Вход(X1) до начала работы *Таймера*, пройдет время равное:

$t_{Вып.\ прог.}*2+t_{Вх}+t_{Вх}=t_{Погрешности\ по\ Выходу}$.

По этому если контроллер имеет $t_{Цикла\ обработки}$ = 5 мсек. и поддерживает работу *Таймера* с шагом 1 мсек., то минимальный шаг, с которым может корректно работать *Таймер* контроллера, должен быть больше чем 10 мсек.

В действительности, кроме "программной погрешности", мы должны принимать во внимание и существование "аппаратной погрешности". Т.е. времени, необходимого контроллеру на проверку действительности срабатывания Входа. Так как в реальных условиях возможен шум или скачок, который контроллер может принять за *Включение Входа*, хотя этого и не произошло. Обычно производители предусматривают настройку данного параметра в диапазоне от 0 - 10 мсек., в зависимости от "чистоты" линии.

Так как серия - FX2N является самой мощной и объединяет в себе все преимущества компактных контроллерных систем с производительностью модульных ПЛК, а также поддерживает все возможные инструкции для контролеров серии MELSEC FX, то в данном курсе мы в первую очередь остановимся на особенностях использования и программирования именно этой серии.

3 Контроллеры Серии MELSEC FX

3.1 Контроллеры Серии MELSEC FX - Введение

Серия контроллеров - MELSEC FX объединяет в себе широкий спектр базовых и наращиваемых модулей, позволяющих Вам сконфигурировать оптимальную систему непосредственно под Вашу задачу. В зависимости от требований Вашего приложения Вы можете выбрать как небольшие и недорогие не наращиваемые контроллеры серии - FXOS, так и масштабируемые решения на базе серий - FXON/ FX2N.

Невозможно сказать какая из серий контроллеров лучше а какая хуже, широкий ряд контроллеров позволяет Вам решать задачи с оптимальным соотношением цена - производительность, и выбор конкретной модели определяется только требованиями приложения(память, быстродействие, возможность расширения, сети …).

Мы можем говорить о том, что использование контроллеров.

Серия - FXOS

В основном используется при решении несложных задач управления, в которых дальнейшее наращивание системы не является целесообразным, а в основе лежит минимизация себестоимости оборудования:

- Упаковочное оборудование;
- Транспортеры и конвейера, подъёмные платформы, лифты;
- Смешивающее оборудование и разрыхлители;
- Системы вентиляции;
- Водоснабжение;
- Системы управления доступом.

Серии - FXON/ FX2N

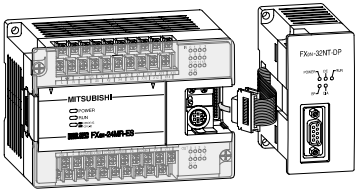
Решения на основе FXON и FX2N серий являются наиболее эффективными при установке в заводских условиях, т.к. становится возможным наращивание возможностей системы по мере роста требований к технологическому процессу. А поддержка сетевой интеграции, позволит гармонично вписать контроллеры MITSUBISHI в уже существующие решения на базе контроллеров других производителей и с панелями управления. Возможна интеграция контроллеров в качестве локальной станции в сетях (MELSECNET/MINI), ведомой станции в открытой сети (PROFIBUS/DP), поддержка интерфейсов ASI и RS-485.

- Деревообработка;
- Элеваторы, системы управления подъём ками и транспортерами;
- Системы водоснабжения и очистки;
- Системы кондиционирования воздуха;
- Текстильное оборудование;
- Пекарное оборудование;
- Автомобильная промышленность;
- Упаковочное оборудование.

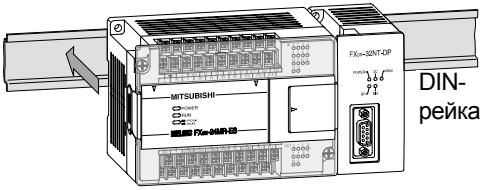
3.2 Контроллеры Серии MELSEC FX - Использование

3.2.1 Установка

Все модули имеют встроенный крепеж для установки на DIN-рейку. Если Вы желаете, модули могут быть так же установлены на плоскую поверхность с помощью винтов.



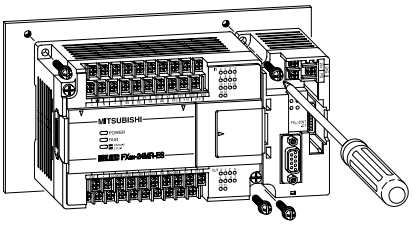
В модулях серий FX0N и FX2N все соединения между системной шиной процессора и другими модулями выполняется с помощью стандартного плоского кабеля. Никаких других кабелей не требуется.



DIN-рейка

3.2.2 Разводка кабелей

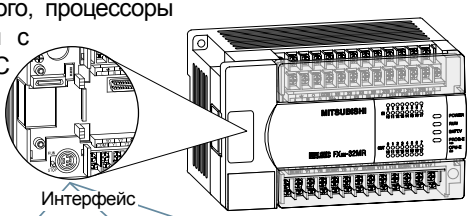
Все входящие кабели присоединяются к специальному разъему "под винт", установленному на модуле. Кабельный ввод имеет крышку для защиты от коротких замыканий и нарушения контакта.



3.2.3 Программирование

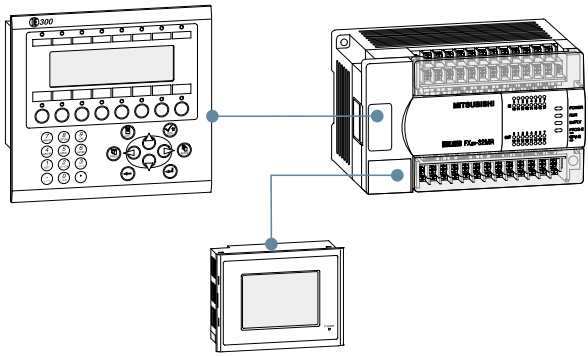
Все процессоры имеют стандартизированный программный интерфейс для присоединения к программатору или к персональному компьютеру.

Простые проекты могут быть запрограммированы непосредственно с ручного программатора, модуля управления серии FX-DU или программного средства MELSEC MEDOC FX/WIN. Кроме того, процессоры могут программироваться с помощью пакета MELSEC MEDOC plus, который запускается на обычном ПК. Это мощная оболочка для создания больших проектов удовлетворяющих стандарту IEC 1131.3.



3.2.4 Визуализация процесса

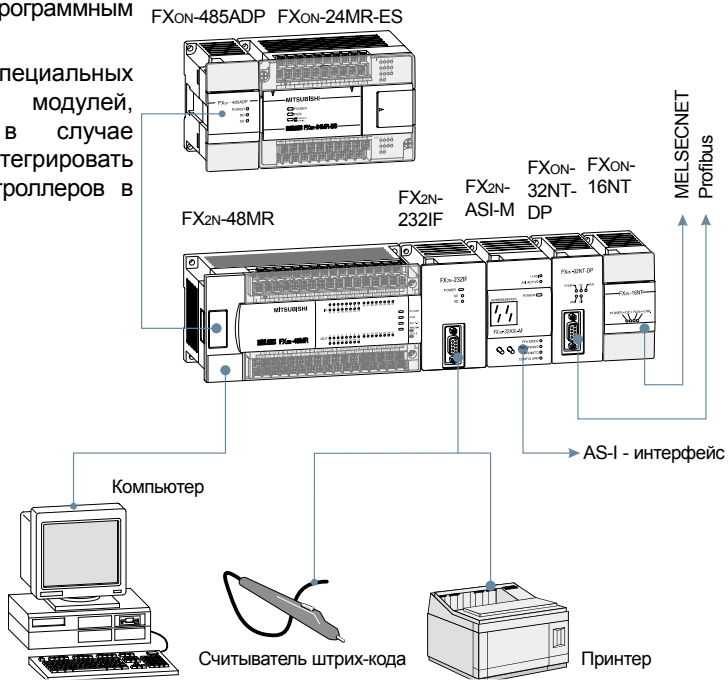
Для более эффективного наблюдения за процессом Вы можете создать систему визуализации процесса как аппаратное или программное решение с частичной или полной графической поддержкой. Продукты для визуализации от MITSUBISHI включают в себя различные операторские терминалы, а также мощный программный пакет визуализации MX-SCADA.



3.2.5 Периферия

Некоторые интерфейсные модули могут быть присоединены к устройствам вывода (например, к принтеру) или же устройству ввода (например, к считывателю штрих-кода). В качестве опции, встроенные интерфейсы могут быть присоединены к другим операторским или программным интерфейсам.

Широкий ряд специальных коммуникационных модулей, позволит Вам в случае необходимости интегрировать Вашу систему контроллеров в различные сети.



3.3 Контроллеры Серии MELSEC FX - Сети MELSEC

3.3.1 TCP/IP Ethernet

Позволит обеспечить соединение со всемирно распространенным стандартным TCP/IP протоколом. Компьютер, присоединенный к Ethernet получает полный доступ ко всем контроллерам в сети MELSECNET, и далее ко всем устройствам ввода/вывода на уровне производства.

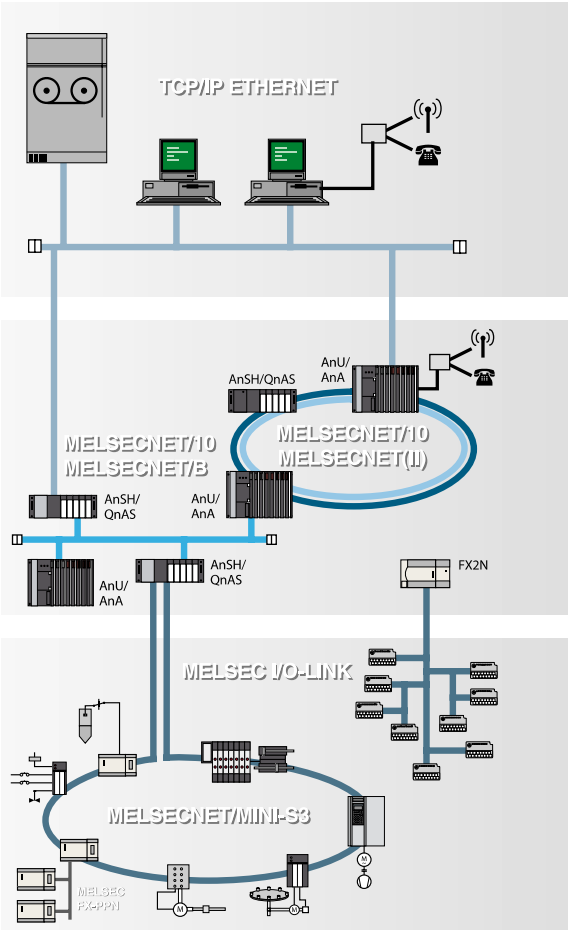
3.3.2 MELSECNET/10 и NET(II)

Недорогое, доступное и простое в установке решение (благодаря избыточности и организации сети с главным узлом). Максимальная зона покрытия - до 30 км.

УРОВЕНЬ КОМАНД

УРОВЕНЬ УПРАВЛЕНИЯ

УРОВЕНЬ ПРОИЗВОДСТВА



3.4 Контроллеры Серии MELSEC FX - *Открытые сети*

3.4.1 MAP 3.0 ETHERNET

Протокол обмена данными между уровнем управления и уровнем производства с использованием конкурентного доступа к среде передачи и малыми временами задержки.

3.4.2 PROFIBUS FMS

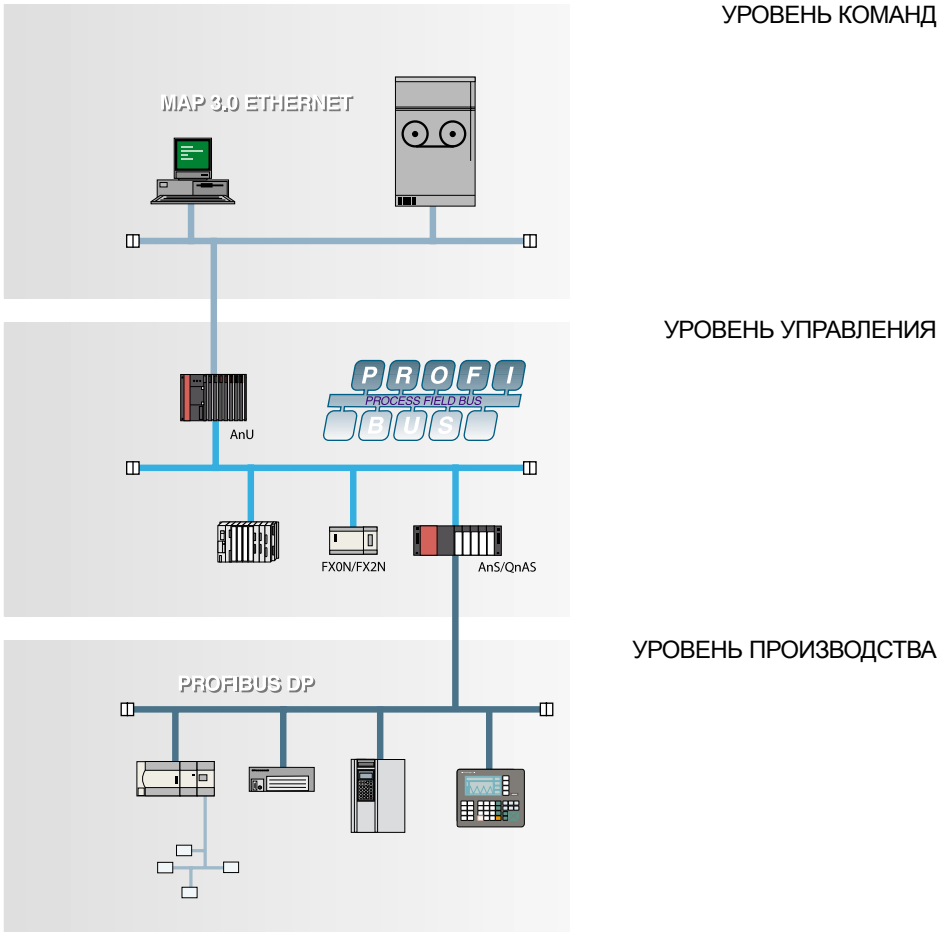
Связь между оборудованием различных производителей внутри одного завода. Автоматический обмен данными с сетью MELSEC.

3.4.3 PROFIBUS DP

Позволяет быстро и просто связать датчики и исполнительные механизмы от различных производителей с контроллерами MELSEC. Скорость передачи данных - до 12МБод.

3.4.3 Интерфейс AS

Международный стандарт для нижнего уровня сети. Соединяет датчики и исполнительные механизмы посредством кабеля "витая пара".



4 Контроллеры Серии MELSEC FX2N

4.1 Описание

Серия MELSEC FX2N обладает самыми мощными процессорами в семействе MELSEC FX. Это серия объединяет в себе преимущества небольших контроллерных систем с производительностью продвинутых модульных ПЛК-систем.

- Одни из самых производительных ПЛК-систем, с периодом обработки одной инструкции 0,08 мксек.
- Мощный базовый набор команд из 125-ти инструкций для быстрого и эффективного программирования различных задач
- Надежность в эксплуатации
- Встроенные часы реального времени
- Встроенный PID-контроллер с автоматической настройкой
- Поддерживает векторные операции и извлечение квадратного корня
- Программная память: до 16 000 шагов.

4.2 Структура системы

- Базовый блок с полнофункциональным контроллером
- Встроенный блок питания
- Процессор
- Встроенные цифровые входы/выходы
- Дополнительные платы для адаптации системы к требованиям приложений по вводу/выводу и функциональности.
- Наращиваемые модули для реализации требований по вводу/выводу данных и функциональности.
- Может быть сконфигурирована, как подчиненная или мастер-станция в сети peer-to-peer (соединение равноправных узлов) или как подчиненная станция в сети 1-n.
- Наличие мастер-функции по распределению сетевых линий ввода/вывода и ASI-интерфейса.
- Дружественный интерфейс программирующей системы, которая включает портативный программатор, ПО удовлетворяющее IEC 1131.3 и интерфейс человек-машина.
- Аксессуары.

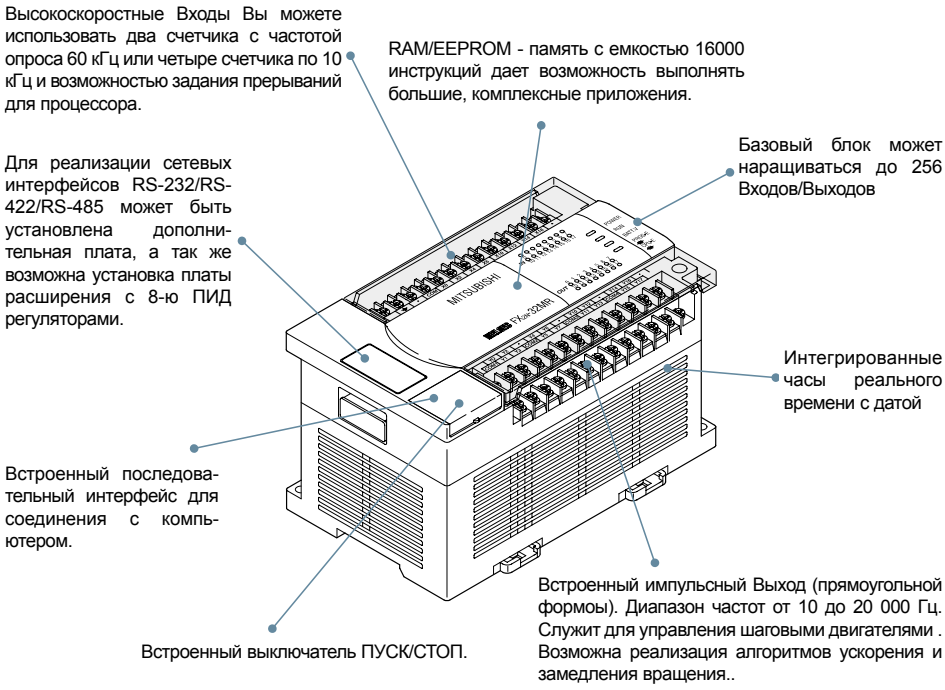
4.3 Особенности

Основой системы MELSEC FX2N является отдельно стоящий базовый блок. Точно так же, как и базовые блоки других FX-серий, он содержит такие компоненты программируемого логического контроллера, как: процессор, память, цепи Входа/Выходов. Все версии базовых блоков имеют один и тот же процессор и одинаковые характеристики по производительности. Базовые блоки выпускаются в 21-ой версии, с количеством входов/выходов от 16 до 128.

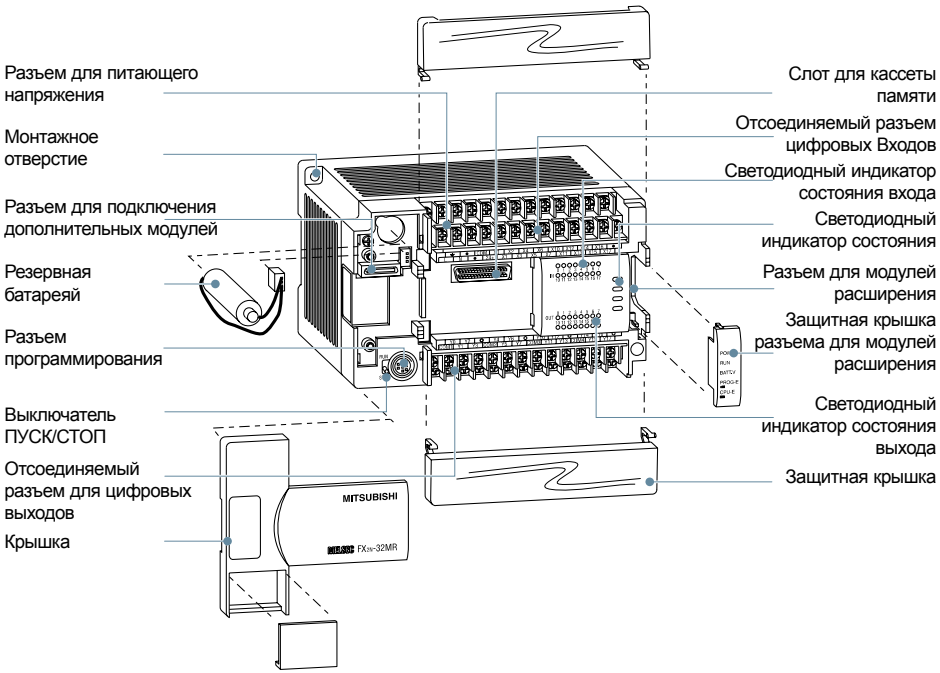
Вы можете выбирать между модулями с напряжением питания 230 В переменного напряжения, и 24 В питанием постоянным напряжением, а так же между релейными или транзисторными Выходами. Цифровые Входы запитываются через встроенный блок питания. Отсоединяемые разъемы делают перенастройку под новые задачи быстрой и легкой.

Широкий спектр наращиваемых модулей позволяет точно подстраивать Вашу систему под нужды конкретного приложения.

Вы можете добавлять необходимое число Входов/Выходов, устанавливая модули расширения с 8-ю или 16-ю входами/выходами. Так же Вы можете добавлять модули расширения специального назначения, например, для обработки аналоговых сигналов, решения задач позиционирования или для реализации дополнительных интерфейсов.



4.4 Описание


Разъем для питающего напряжения
Монтажное отверстие
Разъем для подключения дополнительных модулей
Резервная батареяй
Разъем программирования
Выключатель ПУСК/СТОП
Отсоединяемый разъем для цифровых выходов
Крышка
Слот для кассеты памяти
Отсоединяемый разъем цифровых Входов
Светодиодный индикатор состояния входа
Светодиодный индикатор состояния
Разъем для модулей расширения
Защитная крышка разъема для модулей расширения
Светодиодный индикатор состояния выхода
Защитная крышка
MITSUBISHI
MELSEC FX2N-32MR

4.5 Использование модулей из различных серий

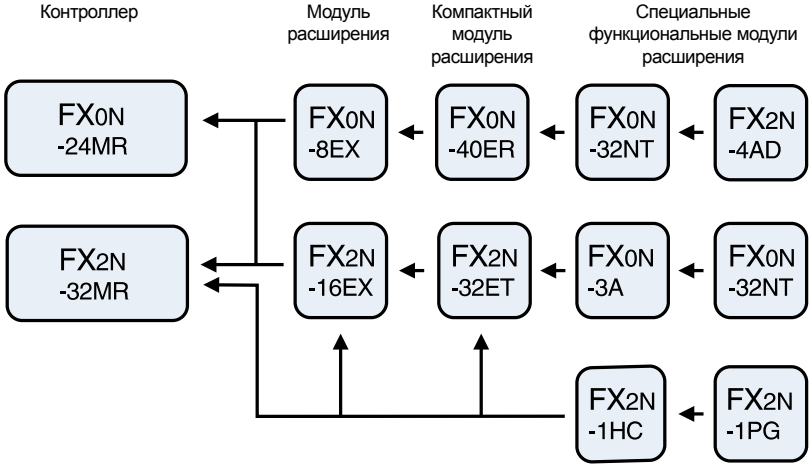
Наращиваемые модули ввода/вывода и модули специального назначения обоих серий могут быть скомбинированы, однако, следует принимать во внимание определенные ограничения, накладываемые различиями между сериями.

Например, Вы можете использовать любые модули серии FX0N вместе с базовым блоком серии FX2N. Возможно так же совместное использование модулей обоих серий.

Для соединения модулей старых FX0N серий с базовым блоком серии FX2N служит специальный конвертор FX2N-CNV-IF

Нижеприведенная таблица показывает ограничения и специальные требования при комбинировании модулей разных серий.

Серия	FX0N	FX2N
Ограничения	Все модули специального назначения применимы с процессорами версии – 2.00 и выше	
Особые требования	–	Модули FX0N-485ADP и FX0N-232ADP требуют функциональной платы FX2N-CNV-BD для работы с серией FX2N.

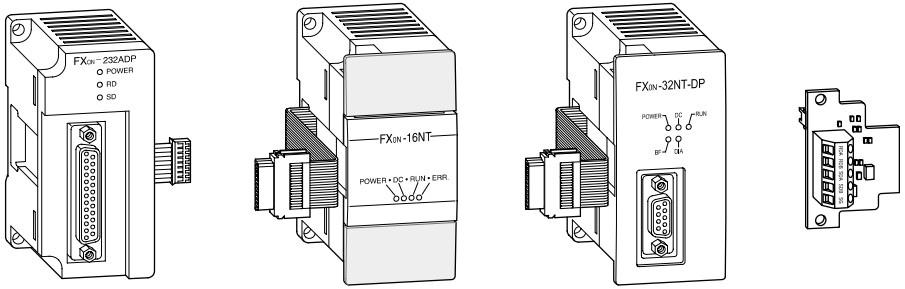


4.5 Использование модулей из различных серий

С помощью дополнительных модулей Вы можете наращивать возможности Вашей системы.

Существует три категории модулей специального назначения:

- Модули расширения Входов/Выходов (подключаются с правой стороны контроллера)
- Модули, которые не поддерживают Ввода/Вывода (подключаются с левой стороны контроллера). Такими модулями являются модули FX0N-485ADP и FX0N-232ADP.
- Внутренние платы адаптеров для FX2N серии. Такие модули устанавливаются непосредственно внутрь контроллера и не поддерживают Ввод/Вывод цифровых данных.



4.5.1 Инструкция по подключению модулей расширения к серии FX0N

Спецификации серии FX0N позволяют подключать к контроллеру блоки расширения в следующих комбинациях:

- Не более 2-ух блоков специального назначения;
- Или модули цифровых Входов/Выходов не более чем на 32 Входа/Выхода (4 x 8 Входов/Выходов или 2 x 16 Входов/Выходов);
- Или один блок специального назначения и один модуль цифрового ввода/вывода не более чем на 16 вводов/выводов (2 x 8 Входов/Выходов или 1 x 16 Входов/Выходов)

4.5.2 Инструкция по подключению модулей расширения к серии FX2N

Спецификации серии FX2N допускают присоединять к базовому блоку блоки расширения в следующих комбинациях:

- Не более 8-ми блоков специального назначения;
- Или модули цифровых Входов/Выходов не более чем на 256 Входов/Выходов.

Важно так же оценить нагрузку на 5В шину и убедиться, что она не превышает допустимую.

При использовании блоков специального назначения может потребоваться дополнительный источник питания на 24В.

Мощность нагрузки Вы можете определить, используя таблицу, приведенную на следующей странице.

4.6 Расчет энергопотребления

Расчет потребления энергии с 5В шины модулями специального назначения осуществляется с помощью таблиц, приведенных на этой странице.

Максимальный допустимый ток в 5В шине показан в нижеприведенной таблице.

Модели	Максимальный ток на 5В шине
FX2N-**M*-ES(ESS)	290 мА
FX2N-**E*-ES(ESS)	690 мА

Значение тока с 24В источника, в зависимости от количества Входов/Выходов приведено в таблице ниже.

Модули специального назначения должны быть запитаны от дополнительного источника, если их суммарное энергопотребление больше допустимой нагрузки тока контроллера.

Максимально возможное число Входов/Выходов - 256

Расчет допустимой нагрузки тока (в мА) для контроллеров FX2N-16M*-E** - FX2N-32M*-E**, а также FX2N-32E*-E** в зависимости от конфигурации.

Количество дополнительных Выходов						
	24	25				
	16	100	50	0		
	8	175	125	75	25	
	0	250	200	150	100	50
		0	**8**	**16**	**24**	**32**
		Количество дополнительных Входов				

Расчет допустимой нагрузки тока (в мА) для контроллеров FX2N-48M*-E** - FX2N-128M*-E**, а так же FX2N-48E*-E** в зависимости от конфигурации.

Количество дополнительных Выходов										
	48	10								
	40	85	35							
	32	160	110	60	10					
	24	235	185	135	85	35				
	16	310	260	210	160	110	60	10		
	8	385	335	285	235	185	135	85	35	
	0	460	410	360	310	260	210	160	110	60
		0	**8**	**16**	**24**	**32**	**40**	**48**	**56**	**60**
		Количество дополнительных Входов								

4.6.1 Пример расчета допустимой нагрузки

Таблица показывает пример расчета энергопитания ПЛК-системы.

Значения энергопотребления для модулей специального назначения приведены в спецификациях к каждому конкретному модулю.

Сравнение с максимальным значением нагрузки приведенным в таблице, показывает, что нагрузка на 5В шину лежит в допустимой области.

Однако, как показывает расчет энергопотребления с 24В шины, все блоки должны бать запитаны от внешнего источника питания напряжением 24В.

Модуль	Кол-во модулей	по шине 24 В		по шине 5 В	
		Ток	Итого	Ток	Итого
FX2N-80MR-ES	1	460 мА	+460 мА	+290 мА	+290 мА
FX2N-4AD	3	50 мА	-150 мА	30 мА	-90 мА
FX2N-4DA	2	200 мА	-400 мА	30 мА	-60 мА
FX2N-232IF	1	80 мА	-80 мА	40 мА	-40 мА
РЕЗУЛЬТАТ:			-170 мА		+100мА

Необходим дополнительный источник питания

4.7 Требования к окружающей среде

Основные характеристики	Значения
Температура окружающей среды	0 - 55°C
Рабочая температура	0 - 55°C
Температура хранения	-20 - +70°C
Напряжение питания	=24В/200 мА(FX0N), 250/460 мА (FX2N)*
Класс защиты	IP 20
Помехозащищенность	1000 Vpp от генератора шума длительность 1мксек; частота 30-100 Гц
Напряжение пробоя изоляции	~1500В в течение одной минуты
Относительная влажность	35-85% - выпадение конденсата недопустимо
Окружающая среда	Избегать установки в атмосфере коррозионных газов, устанавливать в пылезащищенном месте
Ударопрочность	10G (по 3 раза в 3 направлениях)
Вибростойкость	2G**
Сопротивление изоляции	5 МОм на 500 В постоянного напряжения
Заземление	Класс 3
Предохранитель	до FX2N-32□□ - 3.15 А от FX2N-48□□ - 5 А
Сертифицировано на соответствие	UL/CSA/CE/DNV/LL

* Коэффициент пульсации не более 5% на максимальной нагрузке;

** В течение 2-х часов в направлении 3-х координатных осей; 0,5 G для модулей монтируемых на DIN-рейке.

4.8 Основные характеристики

Системные характеристики	FX2N
ПРОГРАММНЫЕ ХАРАКТЕРИСТИКИ	
Адресов Ввода/Вывода	256
Диапазон адресов	Входы X0-X377, Выходы Y0-Y377
Память для программ	8000 шагов, RAM (внутренняя); 4000 шагов EEPROM-кассета (опция); 16 000 шагов RAM кассета (опция); 16 000 шагов EEPROM кассета (опция).
Быстродействие	0.08 мксек на одну лог. инструкцию
Количество инструкций	27 последовательных инструкций; 2 шага дополнительных инструкций; 18 изменяемых инструкций; 107 исполняемых инструкций.
Язык программирования	Последовательность инструкций, SFC
Исполнение программ	Последовательное исполнение программ, режим обновления
Защита программ Паролем	реализовано 3 уровня защиты
ОПЕРАНДЫ	
Внутренние реле	3 072
Специальные реле	256
Шаговые надстройки	1 000
Таймеры	256
Счетчики	256
Высокоскоростные счетчики	6 однофазных (4 двух-фазных) до 60КГц
Потенциометор	8 (опция)
Часы реального времени	год, месяц, день, час, минута, сек., день недели
Регистры данных	8 000
Регисты файлов	до 7 000 (настраивается)
Регистры индексов	16
Специальные регистры	256
Указатели	128
Под-операнды	8
Входы прерываний	6
Константы	16 бит: от ±32 768; 0-FFFF; 32 бит: от ±2 147 483 648; 0-FFFF-FFFF; 32 бит с плавающей запятой: 0, ±1.175 x 10^{-38} – ±3.403x10^{38}.

4.9 Базовые модели контроллеров и модули расширения

4.9.1 Контроллеры

Модель	Вх.	Вых.	~ 240 В		= 24 В	
			Реле	Транзистор	Реле	Транзистор
FX2N-16M	8	8	√	√		
FX2N-32M	16	16	√	√	√	√
FX2N-48M	24	24	√	√	√	√
FX2N-64M	32	32	√	√	√	√
FX2N-80M	40	40	√	√	√	√
FX2N-128M	64	64	√	√		

4.9.2 Компактные модули расширения

Модель	Вх.	Вых.	~ 240 В		= 24 В	
			Реле	Транзистор	Реле	Транзистор
FX2N-32E	16	16	√	√		
FX2N-48E	24	24	√	√	√	√
FXON-40E	24	16	√		√	√

4.9.3 Модульные блоки расширения

Модель	Вх.	Вых.	Реле	Транзистор
FX2N-16EX	16			
FX2N-16EY		16	√	√
FXON-8EX	8			
FXON-16EX	16			
FXON-8EY		8	√	√
FXON-16EY		16	√	√
FXON-8EY	4	4	√	

4.9.3 Модульные блоки расширения

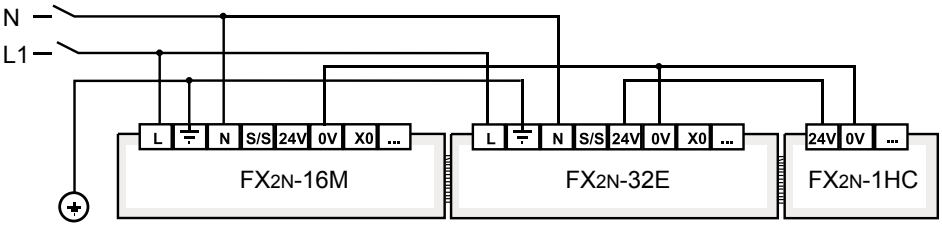
Модель	Реле
FX2N-4AD	Модуль аналоговых входов - на 4 Входа
FX2N-4DA	Модуль аналоговых выходов - на 4 Выхода
FX2N-4AD-TC	Модуль аналоговых входов для работы с термоэлементами
FX2N-4AD-PT	Модуль аналоговых входов для работы с датчиками PT100
FX2N-1HC	Высокоскоростной счетчик до 50 кГц
FX2N-1PG-E	Модуль позиционирования иппульс 100 кГц
FX2N-232IF	Интерфейсный модуль RS232C
FXON-3A	Модуль аналоговых входов/выходов - на 2 Входа и 1 Выход
FXON-16NT	Интерфейсный модуль MELSECNET/MINI
FXON-32NT-DP	Интерфейсный модуль PROFIBUS DP

4.10 Рекомендации по подключению источника питания

При использовании источника питания переменного тока:

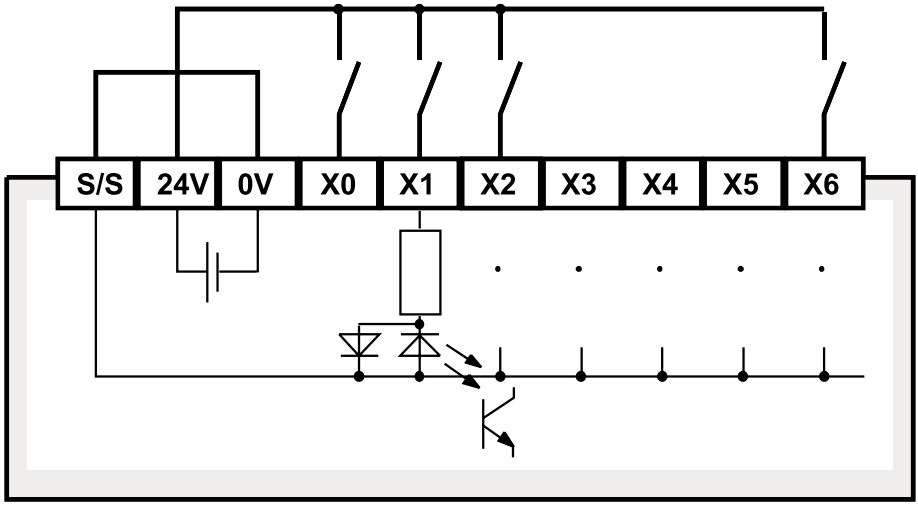
- линия *фазы* должна быть связана с клеммой ***L*** а линия ***0*** с клеммой ***N***. Во избежания поражения электротоком не соединяйте линию *фазы* ***L*** с клеммой ***N***.
- подключение заземления обязательно. Сопротивление заземления Ri < 100 Ом

При использовании источника питания постоянного тока соедините **+** клемму источника питания с **+** клеммой контроллера, а **-** клемму источника питания с **-** клеммой контролера. Ни какое другое подключение недопустимо.



Требования к источнику питания		
Номинальное напряжение	~ 115/250 В	= 24 В
Переключаемое напряжение	~ 85-264 В	= 16-32 В
Время переключения	10 мсек.	5 мсек.

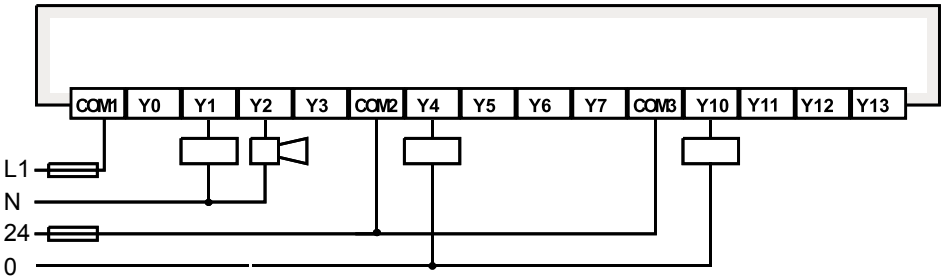
4.11 Подключение Входов



- подключения клеммы S/S к клемме 0 В, позволяет переключить Входы на использование нормально замкнутых датчиков (= 24 В = '1')

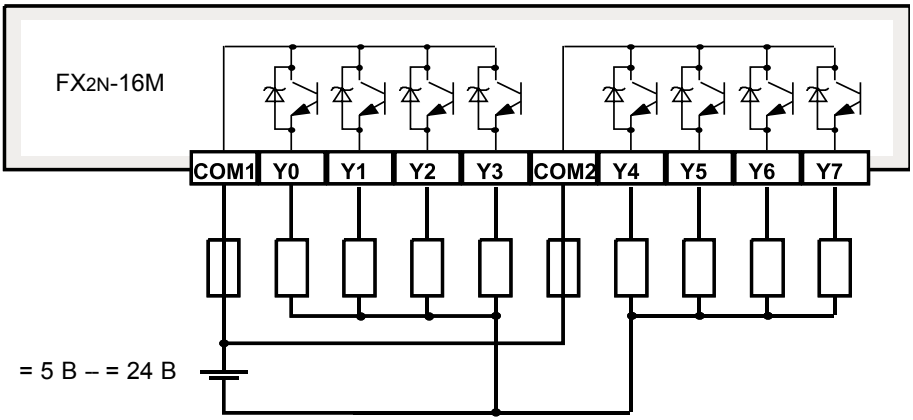
Характеристики входов	
Допустимая нагрузка	7,5 мА
Входной ток-логической единицы	4,5 мА
Входной ток-логического нуля	1,5 мА
Время опроса	10 мсек.(настраиваемое 0 - 15)
Электрическая изоляция	Оптоэлектронная

4.12 Подключение Выходов - реле



Характеристики релейных выходов	
Мак. допустимое напряжение	< ~250 В, < =24 В
Мак. допустимая нагрузка	2 А
Время переключения	10 мсек.
Электрическая изоляция	Реле

4.13 Подключение Выходов - транзистор

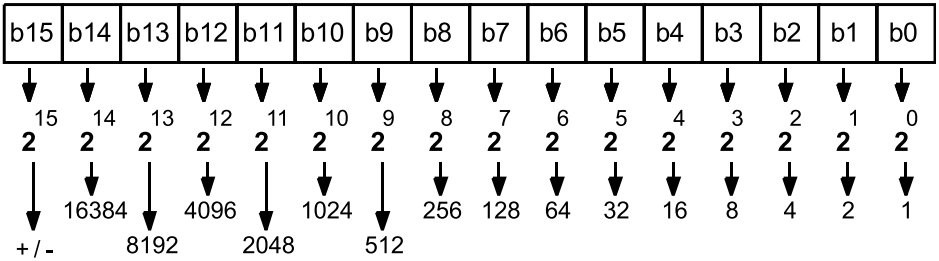


Характеристики *транзисторных* выходов	
Мак. допустимая нагрузка	0.5 А
Мак. перекл. мощьность	12 ВА(Индуктивная)/1.5 Вт(Реактивная)
Время переключения	< 0.2 мсек.
Электрическая изоляция	Оптоэлектронная

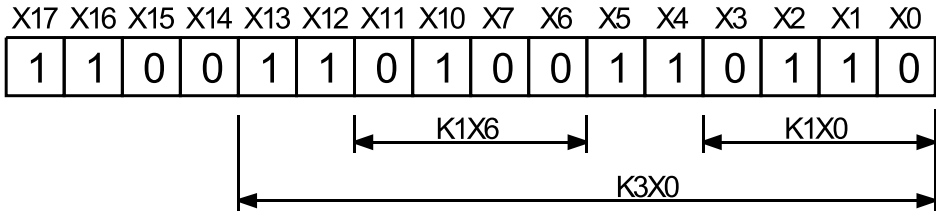
4.14 Краткий обзор обозначений устройств

Описание	Обозначение	Назначение	Адресность
Вход	X	Инициализация *Входа*	256
Выход	Y	Инициализация *Выхода*	
Внутреннии реле	M	Внутренние вспомогательные реле	3072
Таймер	T	Функции таймера	256
Счетчик	C	Функции счетчика	256, 6
Прерываний	S	Число прерываний	1000
Константы	K, H	Десятичные, шестнадцатеричные	16, 32 Бит
Регистр данных	D, R	Данные и регистры	8000
Индексные регистры	V, Z	Буфер для промежуточных значений	16
Указатели	P	Указатель для перехода	128
Прерывание	I	Завершение программы	6, 3
Вложений	N	Разветвление программы	8

4.15 Структура (16-разрядного) Регистра Данных



- Регистр данных - двоичная память для чисел;
- Десятичное число преобразовано в двоичное значение и затем сохранено как битовая комбинация в регистре данных;
- Значение знака (+, -) десятичного числа сохранено в бите b15, числовое значение в битах b0 - b14;
- Десятичные числа от -32767 до 32768.



- K1X0: X0 – X3 ⇨ 4 Входа, начальный адрес X0;
- K1X6: X6 – X11 ⇨ 4 Входа, начальный адрес X6;
- K3X0: X0 – X13 ⇨ 12 Входов, начальный адрес X0.

5 Программирование

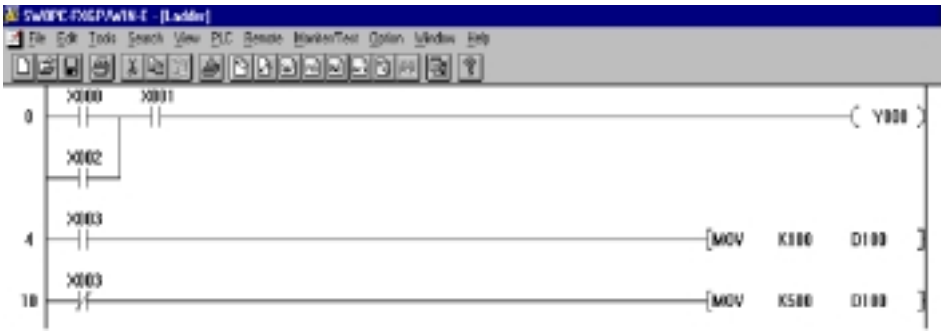
5.1 Программное обеспечение

Для программирования контроллеров серии FX2N, Вы можете выбрать одну из следующих программ:

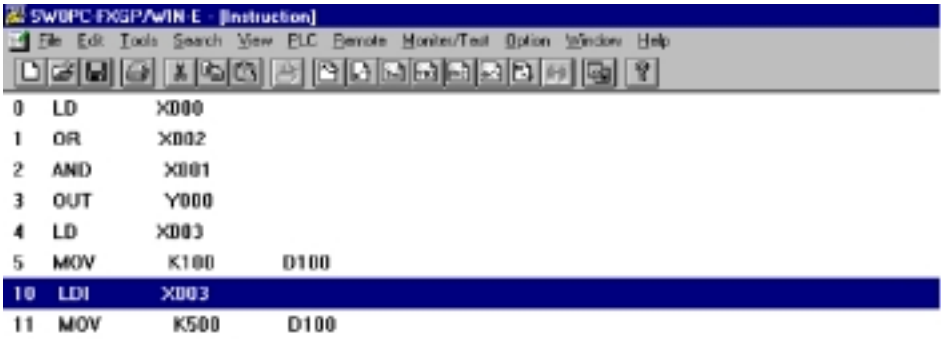
- Melsec Medoc 3.00;
- Melsec Medoc Plus 2.5;
- GPP/WIN.

5.2. Языки программирования

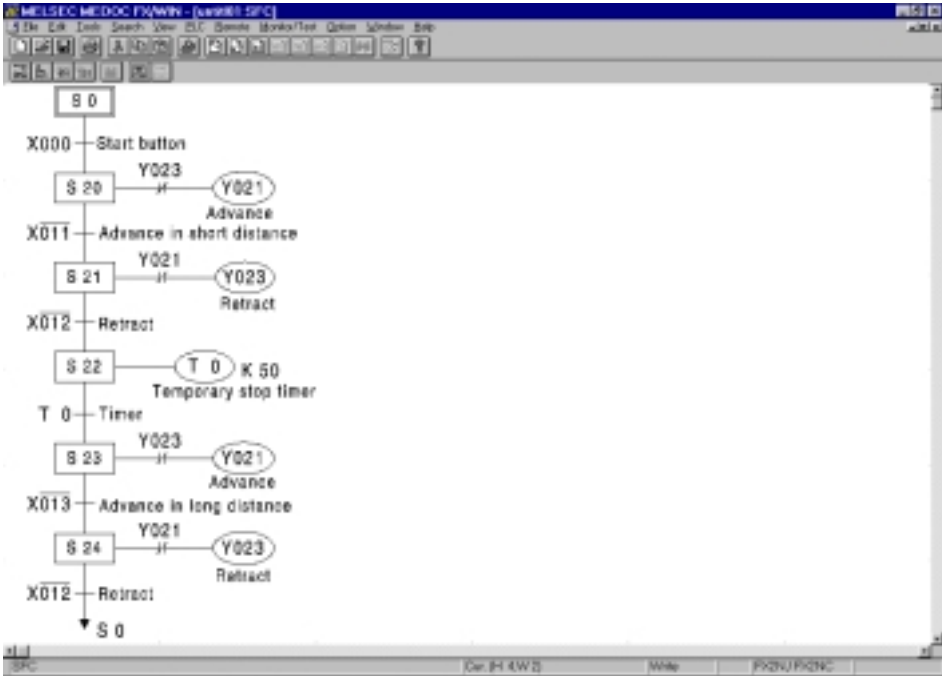
5.2.1 Язык релейно-контактных схем



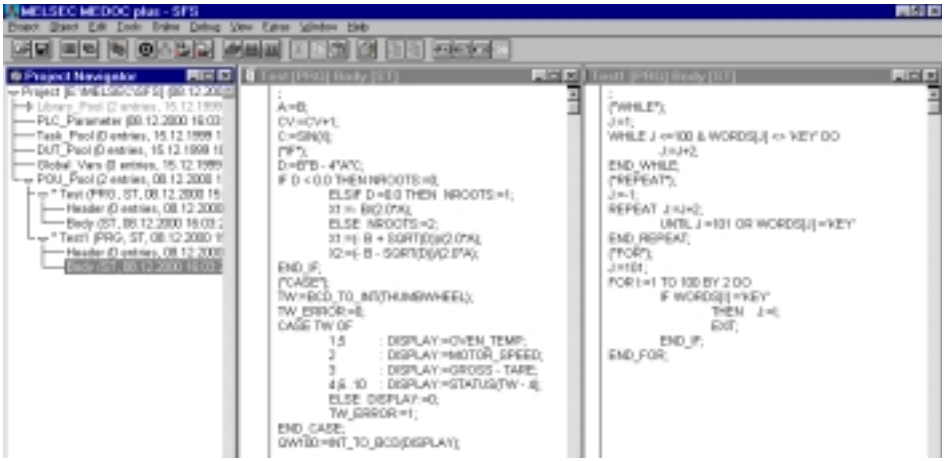
5.2.2 Список инструкций



5.2.3 Диаграмма последовательных функций



5.2.4 Структурное программирование



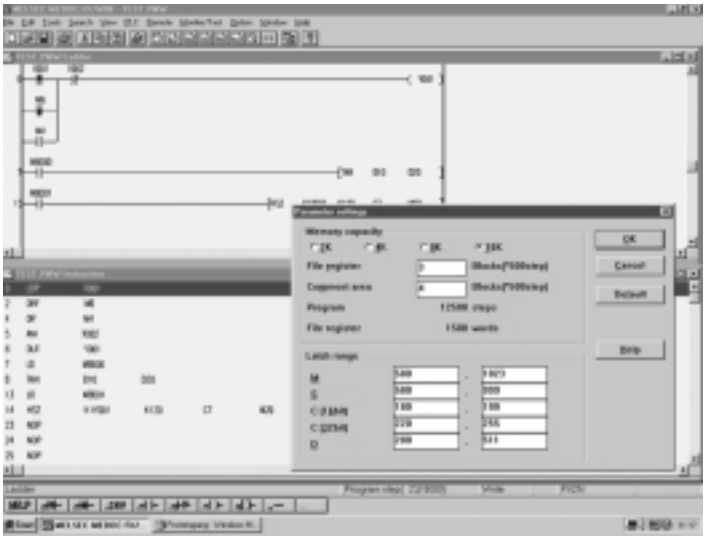
5.3.1 Melsec Medoc FX/WIN 2.20

Позволяет использовать при программировании следующие, взаимно конвертируемые языки описания приложения:

- Язык релейно-контактных схем;
- Список инструкций;
- Диаграмма последовательных функций (Вариант языка релейно-контактных схем).

Программа поддерживает работу, как клавиатуры, так и мышки. Диагностика в режиме on-line и тестирующие функции такие, как Трассировка Выборок, а также принудительная установка выходов и реле предоставляют пользователю максимум прозрачности. Контроллеры FX могут программироваться с помощью последовательных интерфейсов Com1 - Com4 или посредством модема.

Программы, написанные в MELSEC MEDOC под DOS могут быть импортированы в MELSEC MEDOC FX/WIN, что позволит Вам сохранить Ваши предыдущие приложения. В будущем также появится возможность экспортировать программы из MELSEC MEDOC FX/WIN в MELSEC MEDOC Plus.



5.3.2 MELSEC MEDOC plus 2.40

Пакет документации и программирования MELSEC MEDOC plus снижает до минимума затраты на разработку и эксплуатацию программного обеспечения при использовании ПЛК.

Передовая программная архитектура и эффективное выполнение подпрограмм предоставляют Вам принципиально новый уровень функциональности, ориентированный на оптимальное использование программно - аппаратных средств.

Полностью структурированная программная поддержка, библиотеки функций и совместимость с существующими MELSEC MEDOC программами - только часть возможностей этого программного пакета.

Структурированное программирование

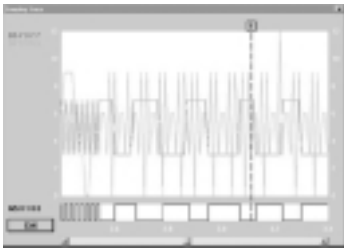
Структурированное программирование открывает новые аспекты в программной прозрачности и структурной организации.

Мощные сетевые функции

Сетевые функции полностью интегрированы, поддерживая интуитивное выполнение и простое конфигурирование параметров. Мощные инструменты, такие как, Network Monitor и Network Diagnostics значительно упрощают сетевой контроль и устранение неисправностей.

Уверенность в будущем

Программы, отвечающие стандарту IEC 1131.3, позволят использовать Ваши приложения и в будущих моделях контроллеров.



Диагностическая функция трассировки выборок отображает состояние контроллерных устройств для циклов контроллеров или временные интервалы, определенные пользователем.

Гораздо меньшее количество ошибок при программировании Ваших приложений благодаря поддержке структурированного программирования, включая язык диаграммы последовательных функций и управление задачами.

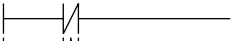
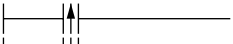
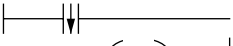
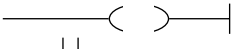
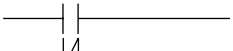
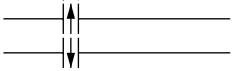
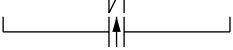
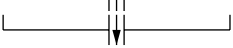
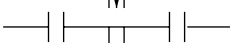
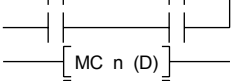
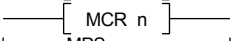
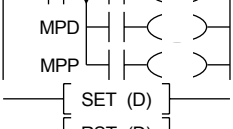
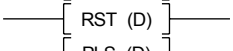
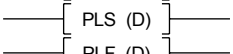
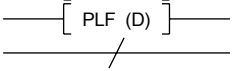
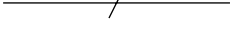
5.3.3 GPP/WIN MELSEC

GPP/WIN поддерживает все контроллеры MELSEC - от FX0S до Q4AR. Обмен списками (листингами) комментариев с пакетом Excel осуществляется при помощи стандартных функций копирования и вставки операционной системы Windows. Вы также можете копировать строки инструкций в пределах Вашей программы. Законченные программные модули могут быть скопированы из одного проекта в другой. Вы имеете возможность запустить несколько версий GPP/WIN. Project Navigator предоставляет отчетливую картину о Вашем полном проекте и может быть отключен за ненадобностью. Кроме того, он обеспечивает доступ к параметрам контроллеров, конфигурациям сети и таблицам комментариев. Все параметры контроллеров и сетевые установки проверяются на достоверность при входе.

Вы можете переключаться между списком инструкций и форматом релейно-контактной схемы, а программа предоставит Вам ряд мониторинговых функций. Широчайший набор фильтров импортирования позволяет Вам использовать существующие программы, написанные с помощью таких систем, как MELSEC MEDOC, GPPA, GPPQ или MELSEC MEDOC FX/WIN. Другим важным нововведением является интеграция программы Ladder Logic Test в GPP/WIN. Ladder Logic Test позволяет программисту производить имитацию работы программы в режиме off-line. Она вызывается из GPP/WIN, и, таким образом, обеспечивает полную работоспособность GPP/WIN в режиме on-line. Интегрированные редакторы Ladder Logic Test позволят Вам манипулировать всеми регистрами данных, вводами/выводами и реле. Программный продукт GPP/WIN - 32-разрядный пакет, работающий под управлением Windows 95 и NT.

6 Краткий обзор основных команд

6.1. Основные Инструкции

Инструкция	Символ	Назначение
LD		Инициализация *Входа*
LDI		Инициализация *Входа* инверсия
LDP		Инициализация *Входа* по включению
LDF		Инициализация *Входа* по выключению
OUT		Инициализация *Выхода*
AND		Операция логического умножения
ANDI		Операция логического умножения инверсия
ANDP		AND по включению
ANDF		AND по выключению
OR		Операция логического сложения
ORI		Операция логического сложения инверсия
ORP		OR по включению
ORF		OR по выключению
ANDB		AND блок
ORB		OR блок
MC		Мастер контроль
MCR		Мастер контроль сброс
MPS		Смещение Вниз Стека
MPD		Считать значение Стека
MPP		Выход из Стека
SET		Установить
RST		Сбросить
PLS		Генерация импульсов по включению
PLF		Генерация импульсов по выключению
INV		Инверсия
NOP		Пустая строка
END		Конец программы

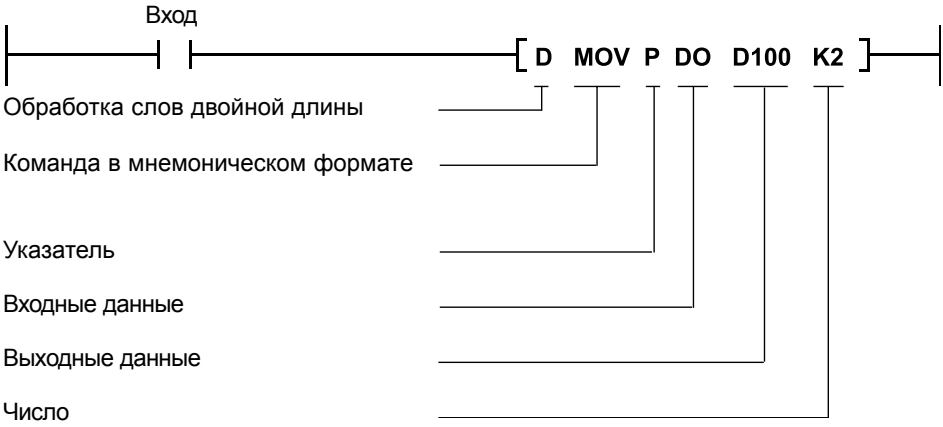
6.1.2 Дополнительные Инструкции

16 – Бит	32 – Бит	Назначение
LD=	LDD=	Инициализация, если (S1) = (S2)
LD>	LDD>	Инициализация, если (S1) > (S2)
LD<	LDD<	Инициализация, если (S1) < (S2)
LD<>	LDD<>	Инициализация, если (S1) <> (S2)
LD>=	LDD>=	Инициализация, если (S1) >= (S2)
LD<=	LDD<=	Инициализация, если (S1) <= (S2)
AND=	ANDD=	И, Вкл. если (S1) = (S2)
AND>	ANDD>	И, Вкл. если (S1) > (S2)
AND<	ANDD<	И, Вкл. если (S1) < (S2)
AND<>	ANDD<>	И, Вкл. если (S1) <> (S2)
AND>=	ANDD>=	И, Вкл. если (S1) >= (S2)
AND<=	ANDD<=	И, Вкл. если (S1) <= (S2)
OR=	ORD=	ИЛИ, Вкл. если (S1) = (S2)
OR>	ORD>	ИЛИ, Вкл. если (S1) > (S2)
OR<	ORD<	ИЛИ, Вкл. если (S1) < (S2)
OR<>	ORD<>	ИЛИ, Вкл. если (S1) <> (S2)
OR>=	ORD>=	ИЛИ, Вкл. если (S1) >= (S2)
OR<=	ORD<=	ИЛИ, Вкл. если (S1) <= (S2)

6.1.3 Специальные Инструкции

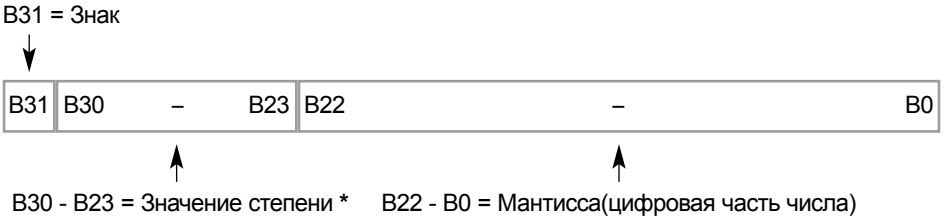
16 – Бит	32 – Бит	P	Назначение
PLSR	DPLSR		Импульсный выход до 20 кГц, (Y0, Y1 для моделей с транзисторными выходами).
	DECMP	√	Сравнение 2-х чисел с плавающей точкой (> < =)
	DEZCP	√	Сравнение числа с плавающей точкой
	DEBCD	√	Преобразование мантиссы и экспоненты в число с плавающей точкой
	DEBIN	√	Выделение мантиссы и экспоненты из числа с плавающей точкой
	DEADD	√	Сложение 2-х чисел с плавающей точкой
	DESUB	√	Вычитание, для чисел с плавающей точкой
	DEMUL	√	Умножение, для чисел с плавающей точкой
	DEDIV	√	Деление, для чисел с плавающей точкой
	DESQR	√	Вычисление квадратного корня
INT	DINT	√	Преобразование числа с плавающей точкой в целое
	DSIN	√	Вычисление синуса
	DCOS	√	Вычисление косинуса
	DTAN	√	Вычисление тангенса
SWAP	DSWAP	√	Перестановка
TCMP		√	Сравнение времени (><=)
TZCP		√	Сравнение времени по временному диапазону (><=)
TADD		√	Сложения времени
TSUB		√	Вычитание времени
TRD		√	Чтение текущего значения часов реального времени
TWR		√	Изменение значения часов реального времени
GRY	DGRY	√	Преобразование целого числа в код Грея
GBIN	DGBIN	√	Преобразование кода Грея в целое число

6.1.4 Структура Команды



6.2. Числа с плавающей точкой

Формат чисел с плавающей точкой хранится в 32-разрядном регистре данных. Под мантиссу отводится 23 Бита, значение степени(экспоненту) 8 Бит, знак 1 Бит.



* Для чисел с плавающей точкой - значение степени, в которую необходимо возвести основание системы счисления, чтобы при умножении на мантиссу получить заданное (исходное) число.

Число с плавающей точкой = ± Мантисса $*$ 2 Экспонента

Знак

B31	
0	"+"

B31	
1	"-"

Экспонента

E30	E29	E28	E27	E26	E25	E24	E23
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Область значений: $(E7 \times 2^7 + E6 \times 2^6 + ... + E0 \times 2^0) - 127$

Мантисса

B22	B21	B20	B19	B18	B17	B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}	2^{-15}	2^{-16}	2^{-17}	2^{-18}	2^{-19}	2^{-20}	2^{-21}	2^{-22}	2^{-23}

Область значений: $1 \times 2^0 + B22 \times 2^{-1} + B21 \times 2^{-2} + ... + B0 \times 2^{-23}$

6.2.1 Числа с плавающей точкой (Пример)

Знаковый бит (B31) = 0(**+**): Положительное значение

Экспонента = 10000001

$= (1 \times 2^7 + 0 \times 2^0 + ... + 1 \times 2^0) - 127$

$= (128 + 0 + ... + 1) - 127 = 2$

Мантисса = 10101000000000000000000

$= 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + ... + 0 \times 2^{-23}$

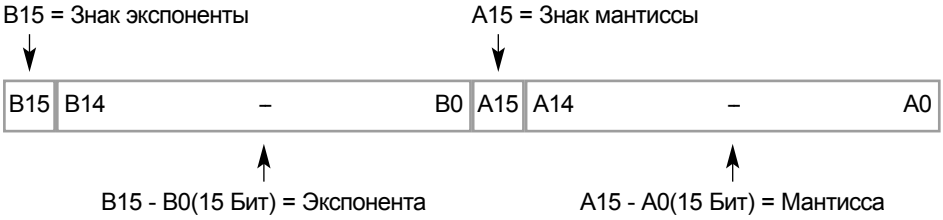
$= 1 + 0{,}5 + 0 + 0{,}125 + ... + 0$

$= 1{,}625$

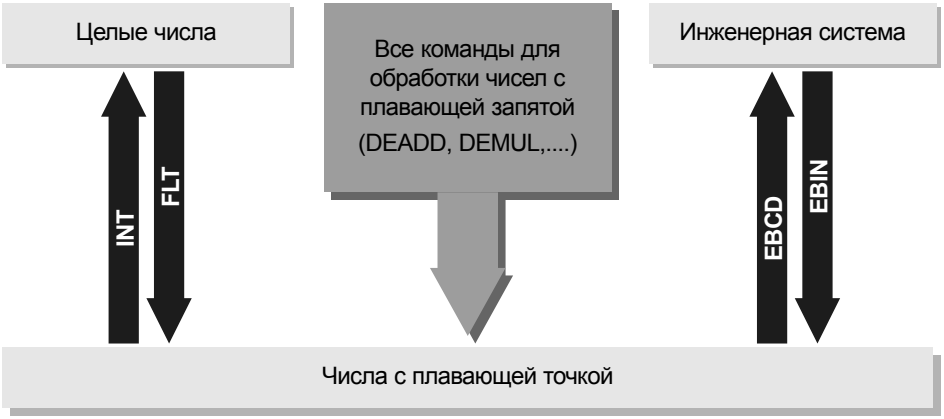
Значение $= + 1{,}625 \times 2^2 = 1{,}625 \times 4 = 6{,}5$

6.2.2 Инженерная система вычислений

Число с плавающей точкой = ± Мантисса $*$ 10 $^{\pm\text{Экспонента}}$



6.2.2 Структура чисел с плавающей точкой



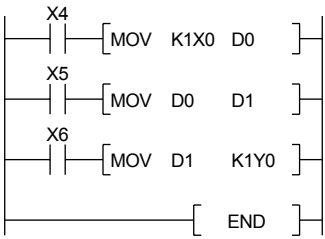
6.2.2 Числа с плавающей точкой (Пример прграммирования)

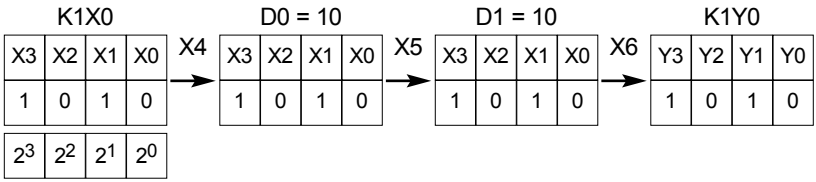
Пример прграммирования				
LD	M8000			
MOV	K123	D0		; Запись числа 123 в регистр D0
FLT	D0	D5		; Преобразование в число с плавающей запятой
DEDIV	D5	K12	D10	; Разделить D5 на 12 ⇒ D10
DEADD	D5	K100	D20	; Сложить D5 и 100 ⇒ D20
DINT	D10	D15		; Преобразовать D10 в целое D15
DEBCD	D20	D25		; Преобразовать D20 в инженерное D25
END				; Конец программы

6.2.3 Перемещение двоичных данных

Пример: Перемещение двоичных данных			
LD	X4		
MOV	K1X0*	D0	; Запись состояния регистров в X3 ... X0 регистр D0
LD	X5		
MOV	D0	D1	; Записать состояние D0 в D1
LD	X6		
MOV	D1	K1Y0*	; Записать состояние D1 в D1
END			; Конец программы

* См. главу ***4.15 Структура (16-разрядного) Регистра Данных.***



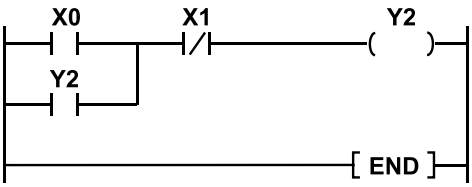


= 8+0+2+0 = 10

6.3. Примеры программирования

6.3.1 Цикл самоподдержки

Язык релейно-контактных схем



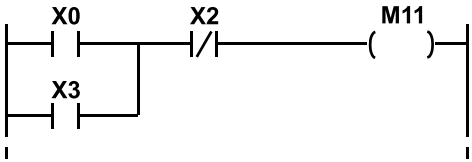
Список инструкций

```
LD      X0
OR      Y2
ANI     X1
OUT     Y2
END
```

X0: ON - Включение исполнительного устройства (Начало работы)
Y2: Исполнительный устройство (блокировка выключения)
X1: OFF- Отключение исполнительного устройства

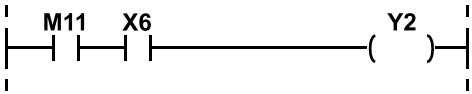
6.3.2 Функции внутреннего реле (вспомогательное реле)

Сохранение результата подключения



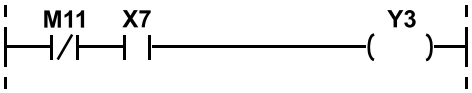

```
LD      X0
OR      X3
ANI     X2
OUT     M11
```

Нормально открытый контакт




```
LD      M11
AND     X6
OUT     Y2
```

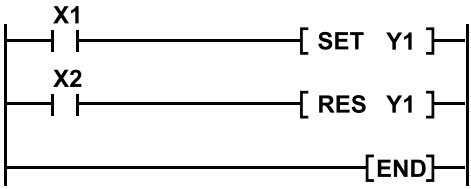
Нормально закрытый контакт




```
LDI     M11
AND     X7
OUT     Y3
```

6.3.3 Установка - Сброс

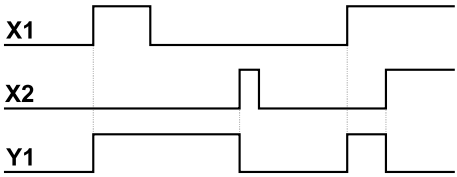
Язык релейно-контактных схем



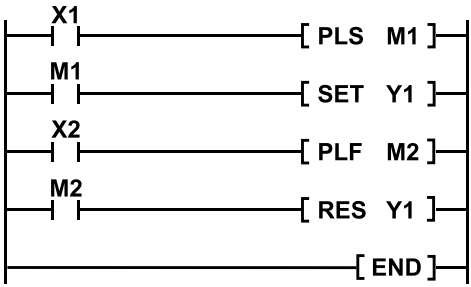
Список инструкций

```
LD      X1
SET     Y1
LD      X2
RST     Y1
END
```

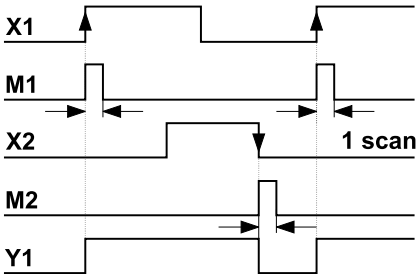
Команда сброса доминирует над командой установки



6.3.4 Импульс и вспомогательное реле



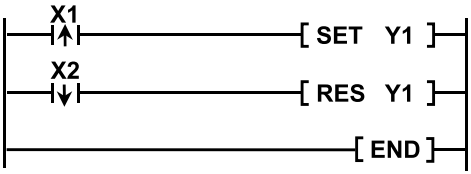

```
LD      X1
PLS     M1
LD      M1
SET     Y1
LD      X2
PLF     M2
LD      M2
RST     Y1
END
```



PLS: Импульс по *Включению* X1
PLF: Импульс по *Выключению* X2

6.3.5 Импульс - Включение по фронту Отключение по спаду в течение цикла

Язык релейно-контактных схем

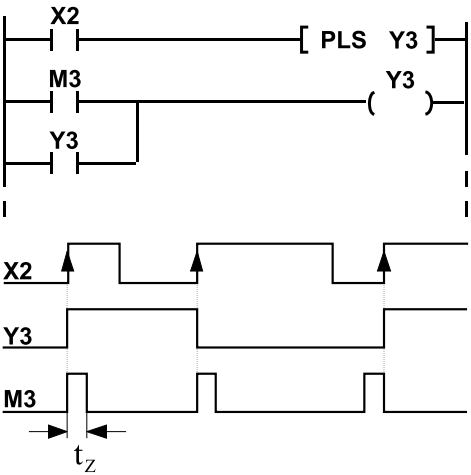


Список инструкций

```
LDP     X1
SET     Y1
LDF     X2
RST     Y1
END
```

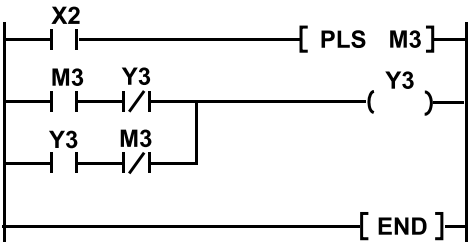
6.3.6 Реализация функции триггера

Реализация функции триггера по включению




```
LD      X2
PLS     M3
LD      M3
OR      Y3
OUT     Y3
```

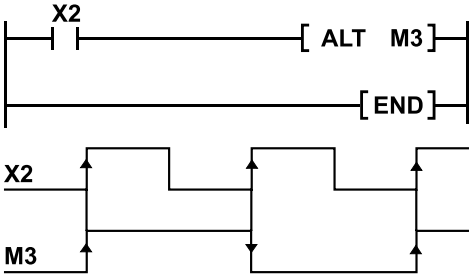
Реализация функции триггера по выключению




```
LD      X2
PLS     M3
LD      M3
ANI     Y3
LD      Y3
ANI     M3
ORB
OUT     Y3
END
```

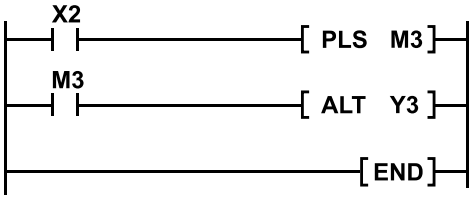
6.3.7 Реализация функции триггера с использованием функций ALT и ALTP

Функция ALT



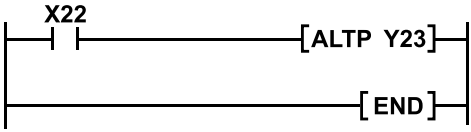

```
LD    X2
ALT   M3
END
```

Реализация функции триггера с использованием функци ALT




```
LD    X2
PLS   M3
LD    M3
ALT   Y3
END
```

Реализация функции триггера с использованием функци ALTP




```
LD    X22
ALTP  Y23
END
```

6.3.8 Реализация функции задержки

Выбор точности часов

Внутреннее реле	Назначение
M8011	0,01 сек
M8012	0,1 сек
M8013	1 сек
M8014	60 сек

Расчет времени

t = K*Число(16, 32 Бит)

Пример:

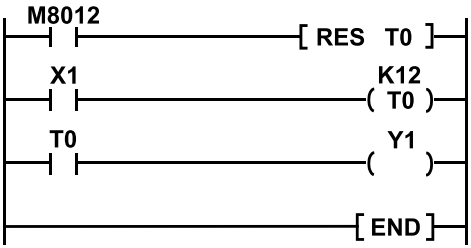
M8012

K = 0,1 сек

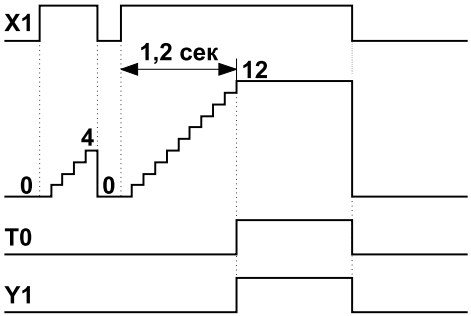
Число = 12

t = 0,1*12 = 1,2 сек.

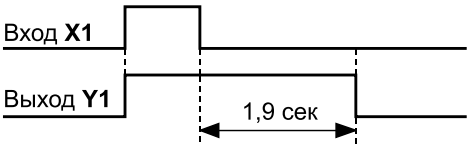
Функция - задержки *Включения*




```
LD      M8012
RES     T0
LD      X1
OUT     T0      K12
LD      T0
OUT     Y1
END
```

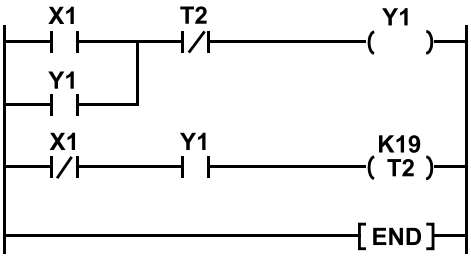


Функция - задержки *Выключения*



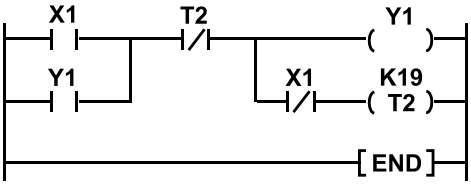
Задержка *Выключения*

Вариант 1



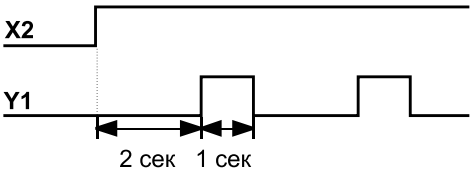

```
LD      X1
OR      Y1
ANI     T2
OUT     Y1
LDI     X1
AND     Y1
OUT     T2      K19
```

Вариант 2

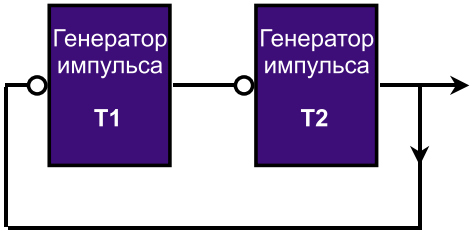



```
LD      X1
OR      Y1
ANI     T2
OUT     Y1
ANI     X1
OUT     T2      K19
```

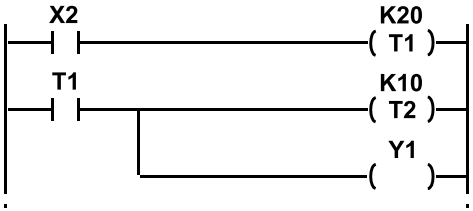
6.3.8 Реализация функции генератора тактовых импульсов



Функциональная схема

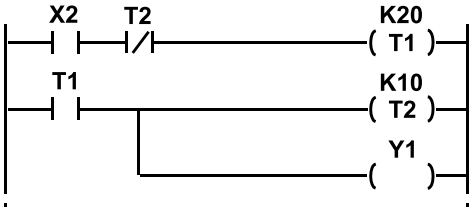


Генератор импульсов



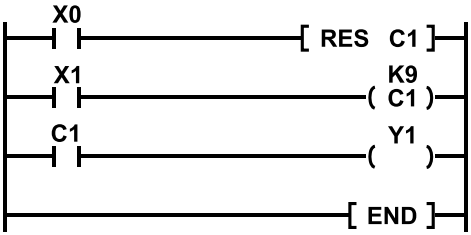

```
LD      X2
OUT     T1      K20
LD      T1
OUT     T2      K10
OUT     Y1
```

Автоматическое повторение

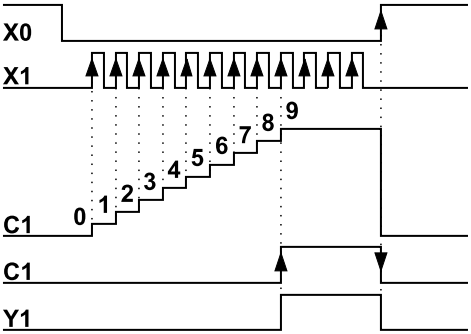



```
LD      X2
ANI     T2
OUT     T1      K20
LD      T1
OUT     T2      K10
OUT     Y1
END
```

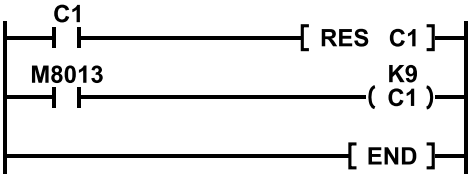
6.3.9 Реализация функции счетчика




```
LD      X0
RST     C1
LD      X1
OUT     C1      K9
LD      C1
OUT     Y1
END
```



Кольцевой счетчик



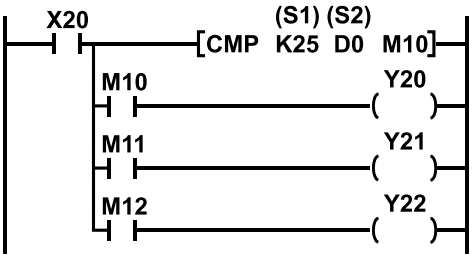
Генератор секунды

```
LD      C1
RST     C1
LD      M8013
OUT     C1      K9
END
```

Использование up/down счетчиков

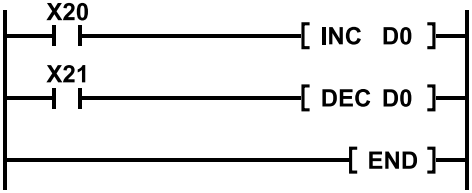
Сравнение данных

Значение S2 сравнивается со значением S1 и в соответствии с выбором M10 … M12, включает Y20 … Y22.



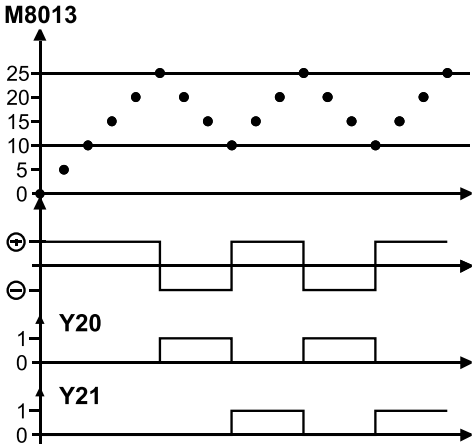
M10 - ON
Если K25 > текущего значения D0
M11 - ON
Если K25 = текущему значению D0
M12 - ON
Если K25 < текущего значения D0

Увеличение уменьшение значения



X20 - ON
Увеличение значения D0 на 1
X21 - ON
Уменьшение значения D0 на 1

Пример использования up/down счетчиков



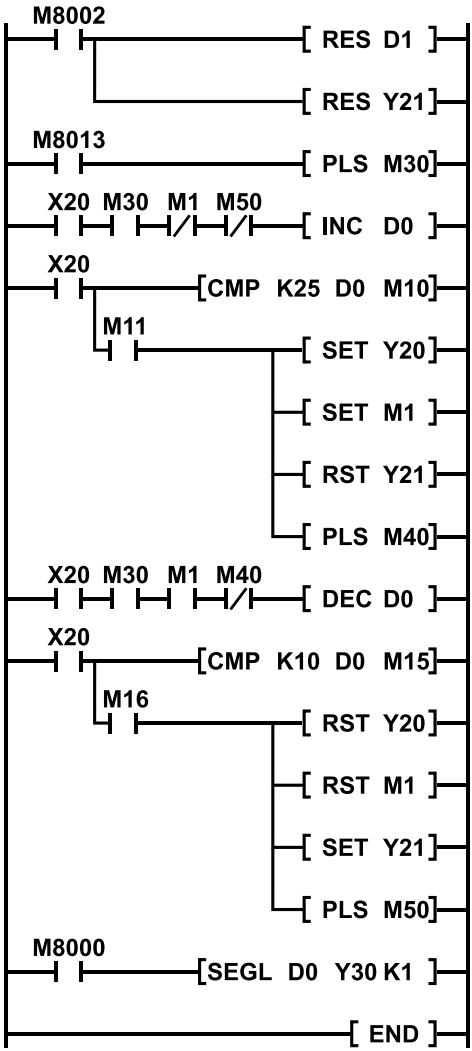
Используем функции INC и DEC, сохраняем текущее значение в регистре D0, специальное реле M8013 задает такт с шагом 1.

Старт программы по *Включению* X20.

Прибавляем 1 к D0(D0 = D0 + 1), при достижении значения 25 (D0 = 25) *Выход* Y20 переходит в состояние *Включено* и счетчик начинает вычитать 1 из D0 (D0 = D0 - 1), при достижении значения D0 = 10, Выход У20 будет Выключен и Включен Выход Y21.

Используемые специальные внутренние реле

Внутреннее реле	Назначение
M8000	ПЛК включен
M8002	Начальный импульс



При инициализации внутреннего реле M8002 данные в регистре D0 примут значение 0, а *Выход(Y21)* будет *Выключен.*

При *Вкл. Входа(X20)* следует начало цикла с увеличением значения D0 на 1.

Содержание регистра D0 сравнивается с K25(K = 25). По достижению значения K=25 *Выход(Y20)* - *Включается* и *Выход(Y21)* - *Отключается.*

При *Вкл. M1* следует начало цикла с уменьшением значения D0 на 1.

Содержание регистра D0 сравнивается с K10(K = 10). По достижению значения K=10 *Выход(Y20)* - *Выключается* и *Выход(Y21)* - *Включается.*

Текущее значение регистра D0 выводится на семисегментный дисплей.

6.3.10 Использование высоко-скоростных счетчиков (ВСС)

	1 - фазный счетчик без функции Старт/Сброс						1 - фазный счетчик с функцией Старт/Сброс						2 - фазный счетчик двунаправленный						Счетчик с контролем фаз - AB					
	C235	C236	C237	C238	C239	C240	C241	C242	C243	C244	C245	C246	C247	C248	C249	C250	C251	C252	C253	C254	C255			
X0	U/D						U/D			U/D		U	U		U		A	A		A				
X1		U/D					R			R		D	D		D		B	B		B				
X2			U/D					U/D			U/D		R			R		R		R				
X3				U/D				R			R			U					A		A			
X4					U/D				U/D					D					B		B			
X5						U/D			R					R					R		R			
X6										S					S					S				
X7											S					S					S			

U - счет вверх;

D - счет вниз;

A - фаза;

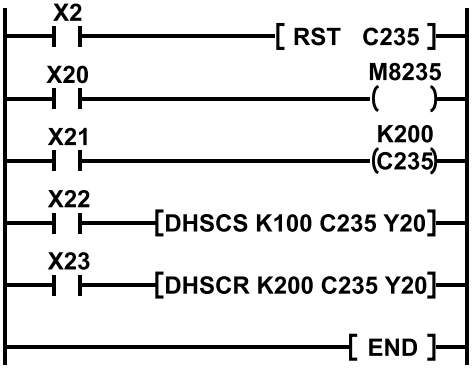
B - фаза;

R - сброс;

S - старт.

Примеры использования высоко-скоростного счетчика

Подобная запись недопустима, так как старт *ВСС* не может осуществляться *Входом*, по которому осуществляется счет.



Сброс *ВСС* C235 в состояние 0.

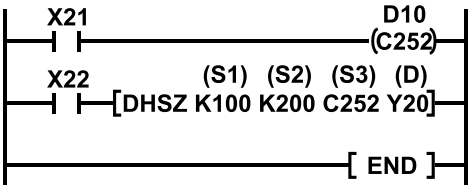
Определение направления:

- M8235 - *Выкл.*(счет вверх);
- M8235 - *Вкл.*(счет вниз).

Y20 - *Вкл.*
Если C235 = K100

Y20 - *Выкл.*
Если C235 = K200

Высоко-скоростной счетчик с контролем фаз AB

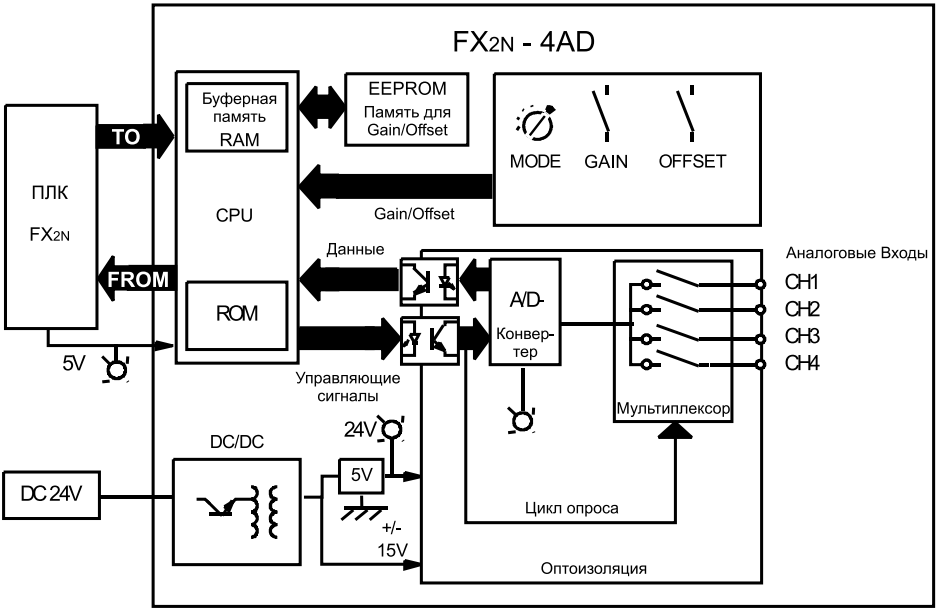


Y20 (D) - C252 < S1, K100 (S3< S1)

Y21 (D+1) - C252 > S1, K100 < S2, K200 (S3> S1, S3< S2)

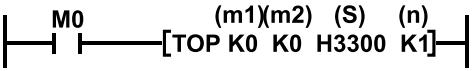
Y22 (D+2) - C252 > S2, K200 (S3> S2)

6.3.11 Использование модуля аналоговых Входов FX2N-4AD



Инициализация канала

Формат команды *TO*



m1	- адрес модуля (0 - 7);
m2	- адрес ячейки БМ (0 - 32167);
S	- значение;
n	- число слов.

Формат *H0000*

$H0^{CH4}0^{CH3}0^{CH2}0^{CH1}$

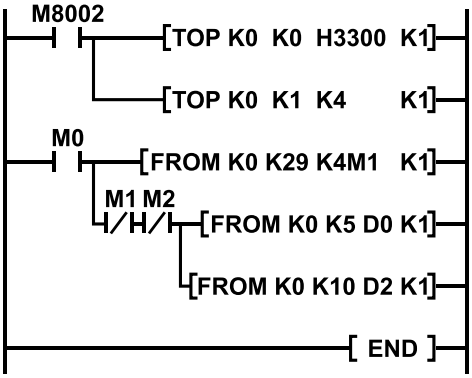
0 = 0 - установка диапазона (от - 10В до +10В);
0 = 1 - установка диапазона (от +4mA до +20mA);
0 = 2 - установка диапазона (от -20mA до +20mA);
0 = 3 - канал *Выкл.*
Пример:
H3310
0^{CH1} = 0 - Канал 1 измеряет значение в диапазоне (от - 10В до +10В);
0^{CH2} = 1 - Канал 2 измеряет значение в диапазоне (от +4mA до +20mA);
0^{CH3} = 3 - Канал 3 *Выключен*;
0^{CH4} = 3 - Канал 4 *Выключен*.

Буферная память FX2N-4AD			
0	*	Инициализация	
1	*	Канал № 1	Число выборок для усреднения
2	*	Канал № 2	
3	*	Канал № 3	
4	*	Канал № 4	
5	**	Канал № 1	Среднее значение
6	**	Канал № 2	
7	**	Канал № 3	
8	**	Канал № 4	
9	**	Канал № 1	Текущее значение
10	**	Канал № 2	
11	**	Канал № 3	
12	**	Канал № 4	
13-14		Не используется	
15	**	Время преобразования	
16-19		Не используется	
20	*	Сброс значений	
21	*	Включить настройки Offset/Gain	
22	*	Настройка Offset/Gain	
23	*	Значение Offset в мВ или μA	
24	*	Значение Gain в мВ или μA	
25-28		Не используется	
29	**	Код ошибки	
30	**	Идентифицирующий код	
31		Не используется	

* Данные значения могут быть изменены с помощью инструкции *TO*;

** Данные значения могут быть считаны с помощью инструкции *FROM*.

Пример использования



Каналы 1 и 2 измеряют напряжение;
Каналы 3 и 4 *Выключены*.

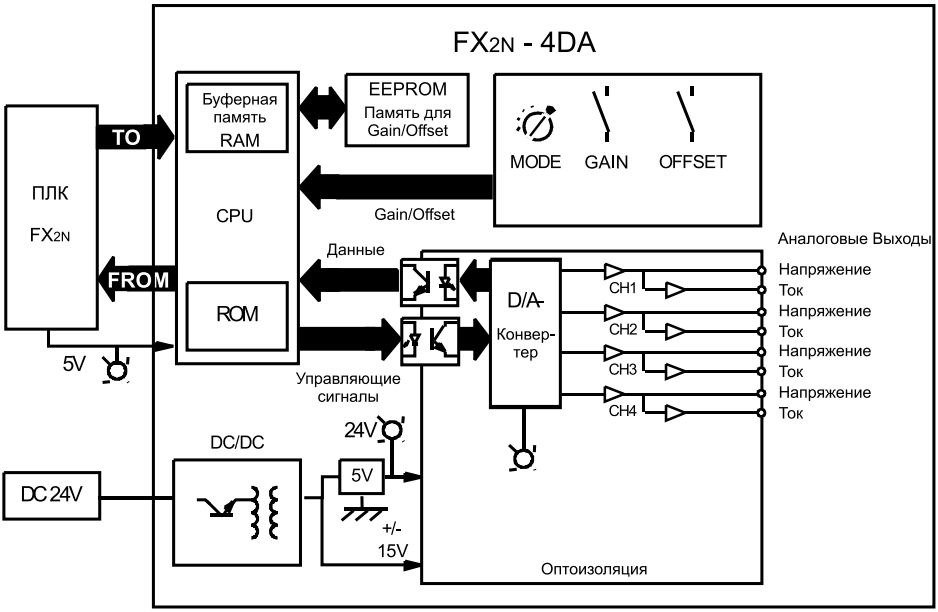
Усреднение по 4-м значениям.

Запись значения *Ошибки* в регистры M1-M4.

Поместить усредненное значение (Канал 1) в регистр D0.

Поместить текущее значение (Канал 1) в регистр D2.

6.3.12 Использование модуля аналоговых Выходов FX2N-4DA



Формат *H0000*

$H0^{CH4}0^{CH3}0^{CH2}0^{CH1}$

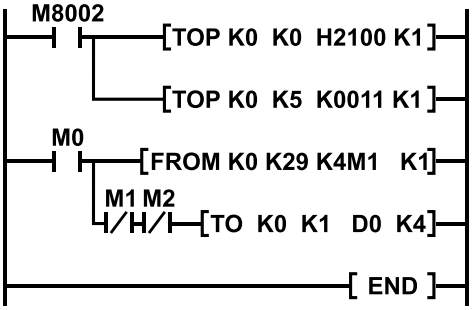
0 = 0 - установка диапазона (от - 10В до +10В);
0 = 1 - установка диапазона (от +4mA до +20mA);
0 = 2 - установка диапазона (от 0mA до +20mA).

Пример:

H2110

0^{CH1} = 0 - Канал 1 изменяет значение в диапазоне (от - 10В до +10В);
0^{CH2} = 1 - Канал 2 изменяет значение в диапазоне (от +4mA до +20mA);
0^{CH3} = 1 - Канал 3 изменяет значение в диапазоне (от +4mA до +20mA);
0^{CH4} = 2 - Канал 4 изменяет значение в диапазоне (от 0mA до +20mA).

Пример использования



Каналы 1 и 2 изменяют нанряжение;
Каналы 3 и 4 изменяют ток.

Каналы 1 и 2 сброс значения;
Каналы 3 и 4 сохранение значения.

Запись значения *Ошибки* в регистры M1-M4.

Поместить значение в регистры D0-D3.

Буферная память FX2N-4DA			
0	*	Инициализация	
1	*	Канал № 1	Числовое значение *Выхода*
2	*	Канал № 2	
3	*	Канал № 3	
4	*	Канал № 4	
5	*	Сохранение данных	
6-7		Не используется	
8	*	Установка Offset/Gain (K1,K2)	
9	*	Установка Offset/Gain (K3,K4)	
10	*	Канал № 1 - установка Offset (мВ, μA)	
11	*	Канал № 1 - установка Gain (мВ, μA)	
12	*	Канал № 2 - установка Offset (мВ, μA)	
13	*	Канал № 2 - установка Gain (мВ, μA)	
14	*	Канал № 3 - установка Offset (мВ, μA)	
15	*	Канал № 3 - установка Gain (мВ, μA)	
16	*	Канал № 4 - установка Offset (мВ, μA)	
17	*	Канал № 4 - установка Gain (мВ, μA)	
18-19		Не используется	
20	*	Сохранение данных	
21	*	Сохранение данных	
22-28	*	Не используется	
29	**	Код ошибки	
30	**	Идентифицирующий код	
31		Не используется	

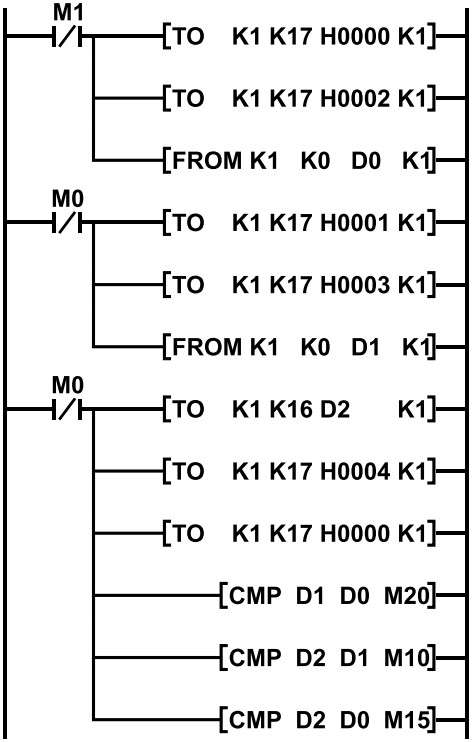
* Данные значения могут быть изменены с помощью инструкции *TO*;

** Данные значения могут быть считаны с помощью инструкции *FROM*.

*6.3.13 Использование модуля аналоговых Входов/Выходов FX*ON*-3A*

Буферная память FXON-3A									
	b15-b8	b7	b6	b5	b4	b3	b2	b1	b0
0	*	Текущее значение (*Входа*) для первого или второго Канала							
16		Текущее значение (*Выхода*)							
17	*						D/A старт	A/D старт	A/D канал
1-15	*								
18-31									

Пример использования



Канал 1 - *Вкл.*

Канал 1 - Старт A/D преобразования

Запись текущего значения Канала 1 в регистр D0.

Канал 2 - *Вкл.*

Канал 2 - Старт A/D преобразования

Запись текущего значения Канала 2 в регистр D1.

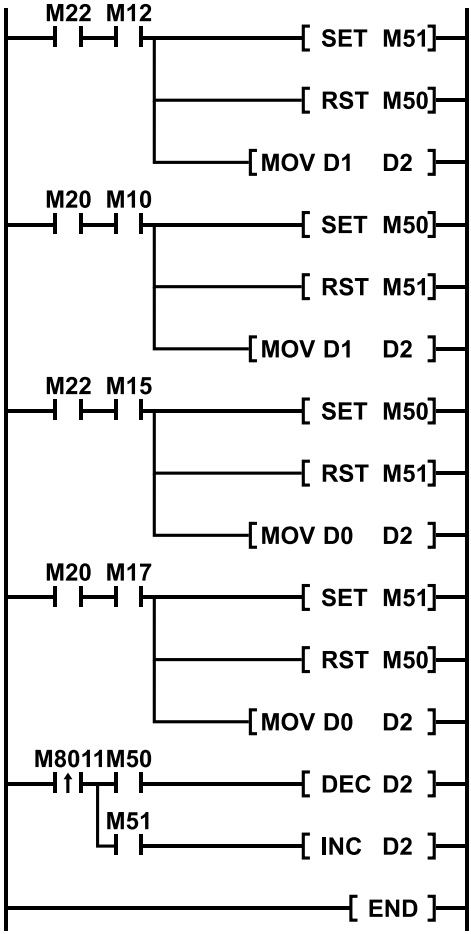
Канал 3 - Поместить значение из регистра D2 в БМ17.

Канал 3 - Преобразование цифрового значения D2 в аналоговое.

Сравнить D1 с D0

Сравнить D2 с D1

Сравнить D2 с D0



Если D0 > D1 и D1 > D2, то

M51 - У*становить*;
M50 - *Сбросить*;
Записать значение из D1 в D2.

Если D0 < D1 и D1 < D2, то

M50 - У*становить*;
M51 - *Сбросить*;
Записать значение из D1 в D2.

Если D0 > D1 и D0 > D2, то

M50 - У*становить*;
M51 - *Сбросить*;
Записать значение из D0 в D2.

Если D0 < D1 и D0 > D2, то

M51 - У*становить*;
M50 - *Сбросить*;
Записать значение из D0 в D2.

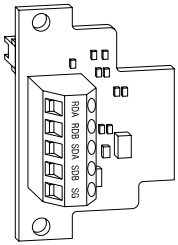
M8011 - Включить таймер с частотой 10мсек;

Если
M50 - Включено;
то
Увеличить значение D2.

Если

M51 - Включено;
то
Уменьшить значение D2.

6.3.13 Использование модуля FX2N-485BD



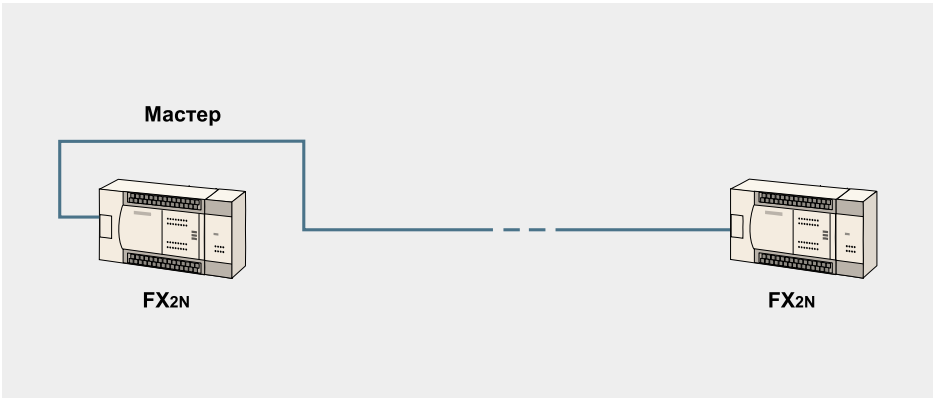
Адаптер интерфейса FX2N-485BD обеспечивает поддержку RS-485 интерфейса для управления внешними устройствами (инвертор, серводвигатели и т.д.).

А так же FX2N-485BD предназначен для организации сетей:

- Соединение равноправных узлов (до 8 контроллеров FXON/FX2N);
- Многоточечной сети 1:n (до 16 контроллеров);
- Параллельная работа.

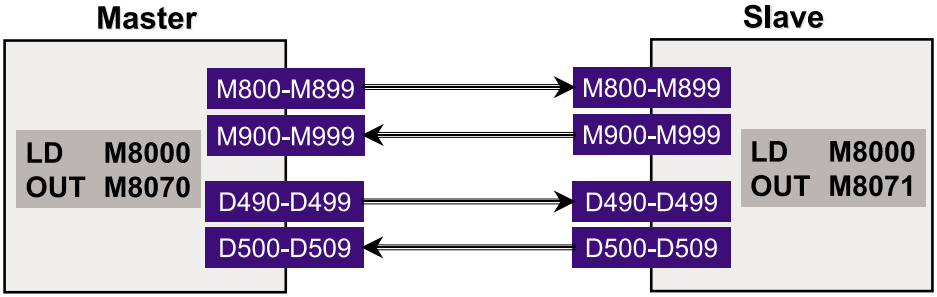
Адаптер устанавливается в слот расширения в базовом блоке FX2N.

Параллельная работа



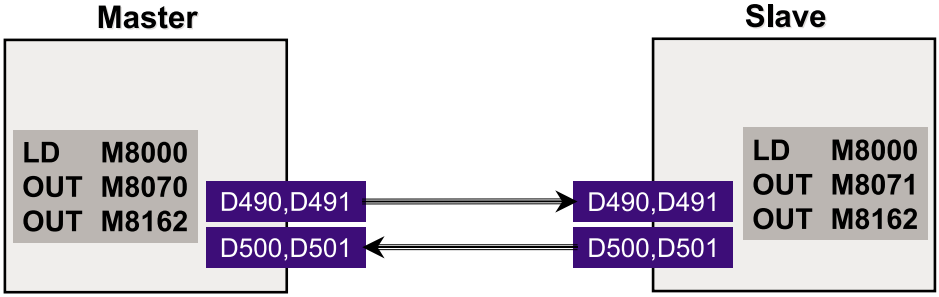
Параллельная работа - Нормальный режим

Связь с автоматической передачей данных, время обмена данными: 70 мсек.



Параллельная работа - Высокоскоростной режим

Связь с автоматической передачей данных, время обмена данными: 20 мсек.



Пример использования

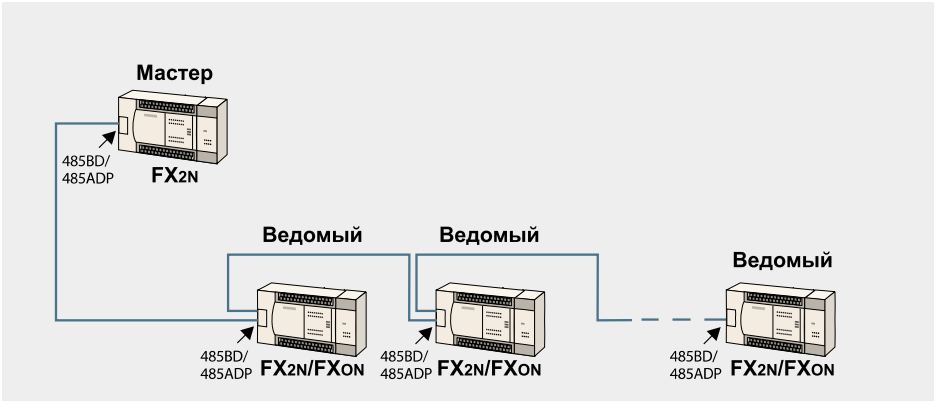
Мастер

```
LD      M8000
OUT     M8070
MOV     K2X0    K2M800
MOV     D500    K4Y0
END
```

Ведомый

```
LD      M8002
MOV     K1      D500
LD      M8000
OUT     M8071
MOV     K2M800  K2Y0
LD      M8013
ROLP    D500    K1
END
```

Соединение равноправных узлов



Время обмена данными [мсек]			
	K0	K1	K2
2	18	22	34
3	26	32	50
4	33	42	66
5	41	52	83
6	49	62	99
7	57	72	115
8	65	82	131
	10 + (n x 8)	12 + (n x 10)	18 + (n x 16)

Инициализация - *Мастера*

```
0 LD        M8038
  MOV  K0   D8176
  MOV  K2   D8177
  MOV  K1   D8178
  MOV  K3   D8179
  MOV  K6   D8180
```

Инициализация только с “0” строки
Номер станции
Число станций
Выбор области данных
Число повторов
Установка ВРЕМЕНИ задержки

Внутреннее реле D8178 - *Выбор области данных*
K0: 4 регистра данных;
K1: 4 регистра данных и 32 внутренних реле;
K2: 8 регистра данных и 32 внутренних реле.

Инициализация - *Ведомого*

```
0 LD    M8038
  MOV   K1        D8176
  LD    M8000
  MOV   K1M1000   K1Y10
  MOV   K1X01     K1M1064
  MOV   K1M1128   K1Y20
```

Инициализация только с “0” строки
Номер станции K1 - K7

Использование данных от мастера
Данных от *Ведомого* 1 ⇒ Остальные
Использование данных от 2-го *Ведомого*

- область обмена начинается с внутреннего реле M1000, блоками по 64 бита.
- область обмена данных начинается с регистра D0, блоками по 8 слов.

7 Реализация приложений

7.1. Примеры

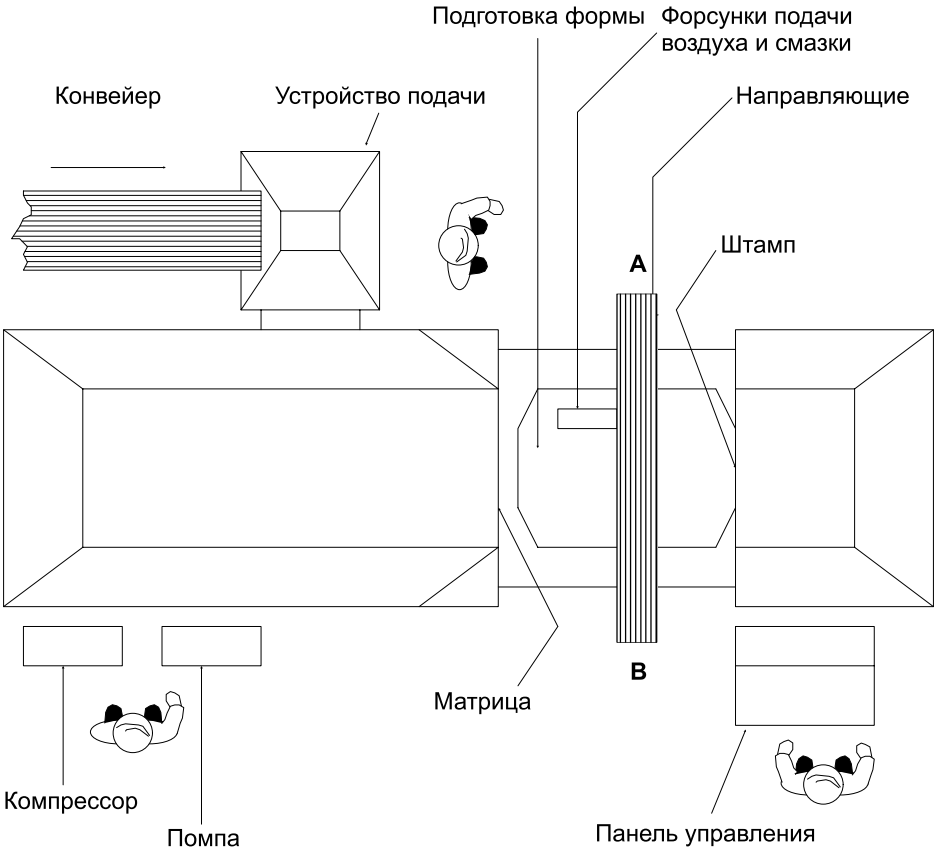
7.1.1 Штамповочная машина

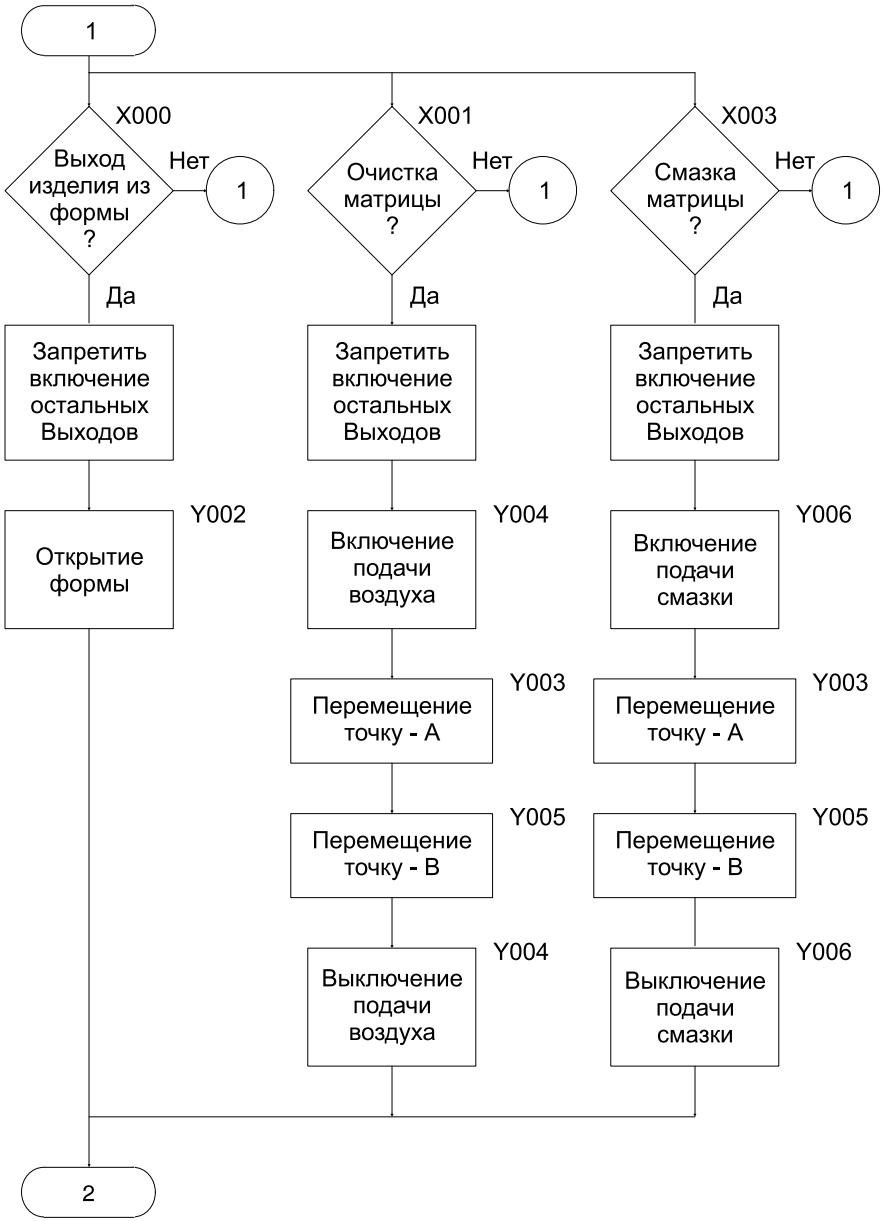
Задача - Выход готового изделия и подготовка матрицы к следующему циклу.

Данный пример демонстрирует работу только части кода программы отвечающей за:

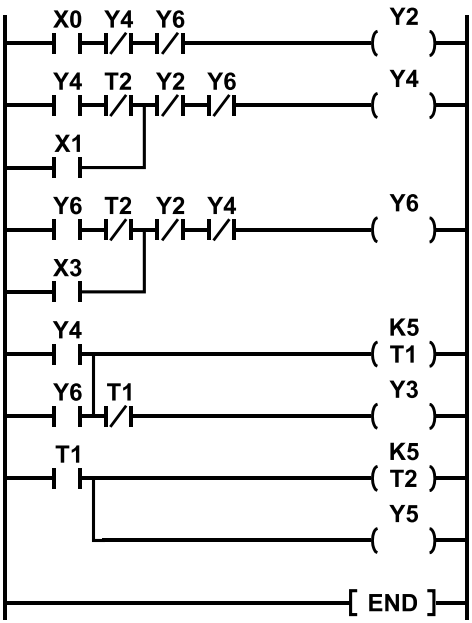
- Открытие формы(перемещение штампа вправо);
- Очистка матрицы(подача воздуха и перемещение из точки А(5 сек) в точку В(5 сек);
- Смазка матрицы(подача смазки и перемещение из точки А (5 сек) в точку В(5 сек).

Примечание: Остальной код, описывающий ход технологического процесс по штамповке изделия, в настоящем примере не представлен.




1
X000
Выход изделия из формы ?
Нет
1
Да
Запретить включение остальных Выходов
Открытие формы
Y002
X001
Очистка матрицы ?
Нет
1
Да
Запретить включение остальных Выходов
Включение подачи воздуха
Y004
Перемещение точку - A
Y003
Перемещение точку - B
Y005
Выключение подачи воздуха
Y004
X003
Смазка матрицы ?
Нет
1
Да
Запретить включение остальных Выходов
Включение подачи смазки
Y006
Перемещение точку - A
Y003
Перемещение точку - B
Y005
Выключение подачи смазки
Y006
2

Язык релейно-контактных схем



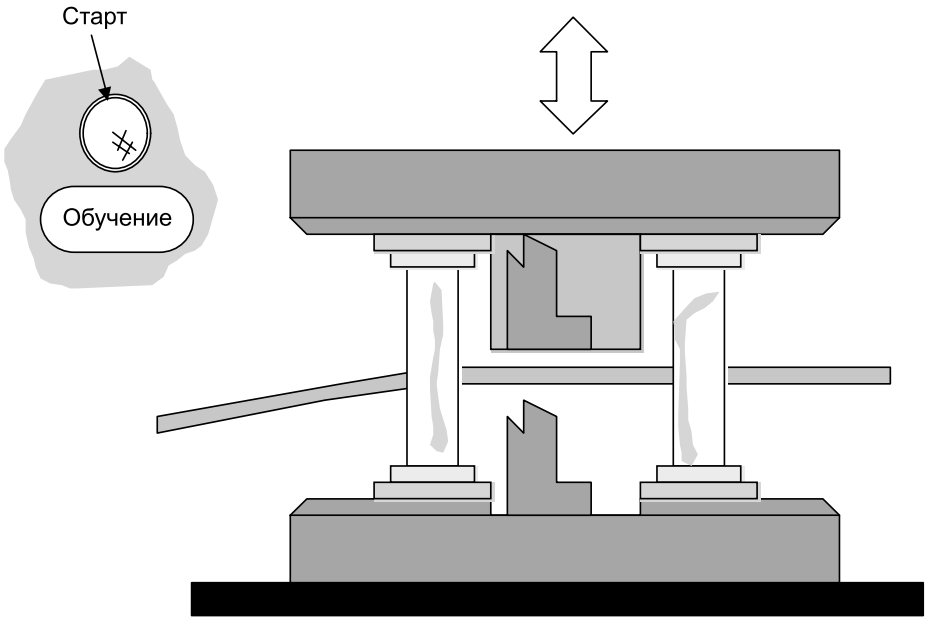
Список инструкций

```
LD      X0
ANI     Y4
ANI     Y6
OUT     Y2
LD      Y4
ANI     T2
OR      X1
ANI     Y2
ANI     Y6
OUT     Y4
LD      Y6
ANI     T2
OR      X3
ANI     Y2
ANI     Y4
OUT     Y6
LD      Y4
OR      Y6
OUT     T1      K5
ANI     T1
OUT     Y3
LD      T1
OUT     T2      K5
OUT     Y5
END
```

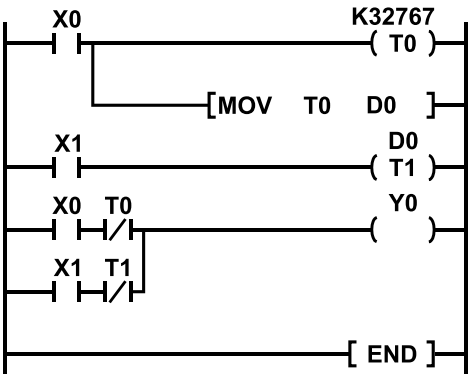
7.1.1 Штамповочная машина - Настройка времени работы штампа

Задача - Настроить время работы штампа

- по нажатию на кнопку *Обучение(ВходX0)* запомнить время работы штампа;
- по нажатию на кнопку *Старт(ВходX1)* выполнять операцию используя время полученное на этапе обучения.



Язык релейно-контактных схем



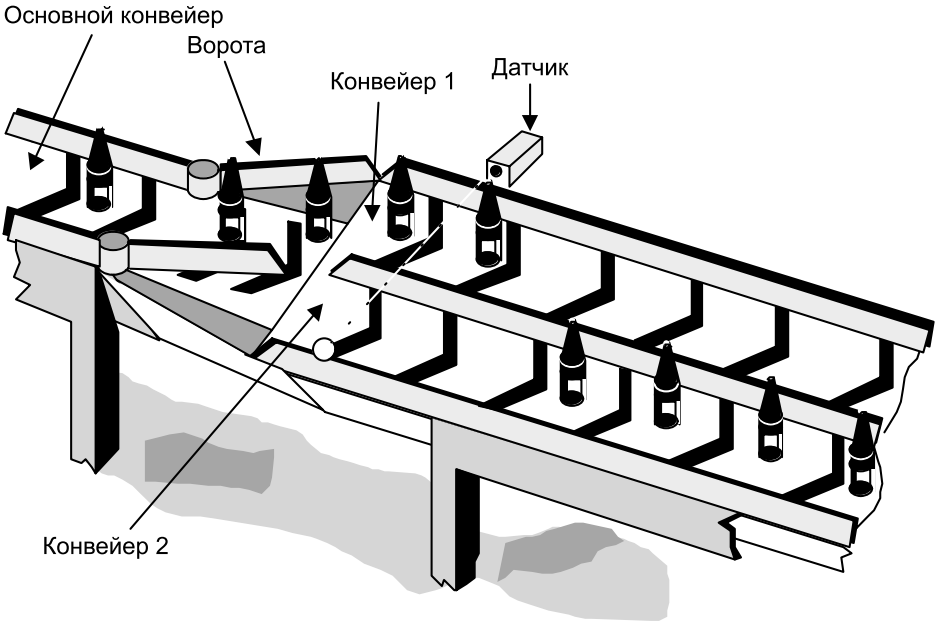
Список инструкций

```
LD    X0
OUT   T0    K32767
MOV   T0    D0
LD    X1
OUT   T1    D0
LD    X0
ANI   T0
LD    X1
ANI   T1
ORB
OUT   Y0
END
```

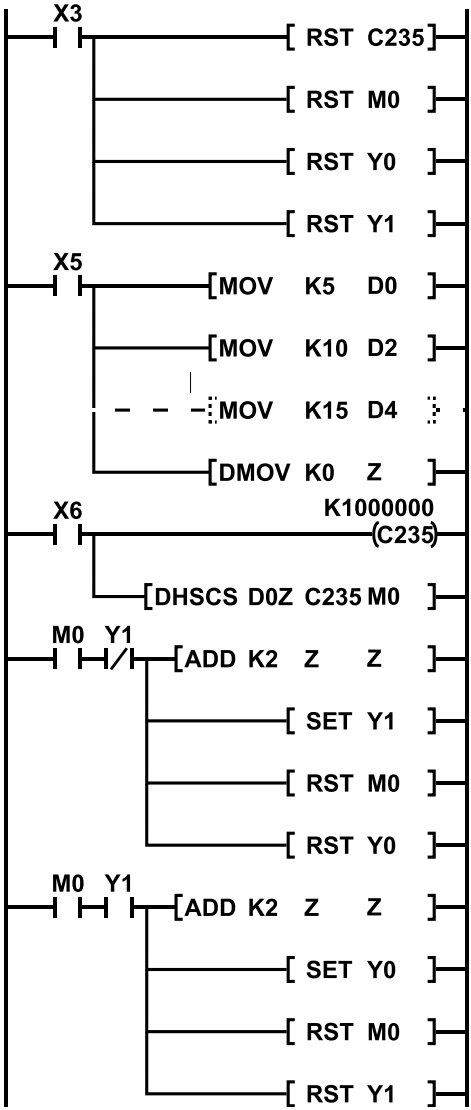
7.1.2 Конвейер - Разделение потоков

Задача - Считать число бутылок и переключать направление.

- Считать число бутылок;
- по достижению заданного значения *D0* переключить направление с *Конвейера 1* на *Конвейера 2*;
- по достижению заданного значения *D2* переключить направление с *Конвейера 2* на *Конвейера 1*;
- ...



Язык релейно-контактных схем



Список инструкций

```
LD      X3
RST     C235
RST     M0
RST     Y0
RST     Y1
LD      X5
MOV     K5      D0
MOV     K10     D2
MOV     K15     D4
...
DMOV    K0      Z
LD      X6
OUT     C235    K1000000
DHSCS   D0Z     C235    M0
LD      M0
ANI     Y1
ADD     K2      Z       Z
SET     Y1
RST     M0
RST     Y0
LD      M0
AND     Y1
ADD     K2      Z       Z
SET     Y0
RST     M0
RST     Y1
...
```