ООО «НПО Электропривод»

Техническое описание

КОНТРОЛЛЕР ШАГОВОГО ДВИГАТЕЛЯ

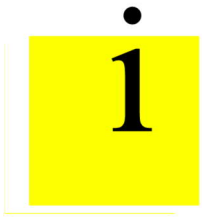
SMSD-8.0Modbus
SMSD-4.2Modbus
SMSD-1.5Modbus ver.3

2022

Настоящий документ является оригинальным руководством по эксплуатации.

Обозначение особых указаний и примечаний:

Внимание
Различные виды опасности, которые могут привести к материальному ущербу, травмам, летальному исходу.



Информация
Рекомендация, полезный совет, ссылка на другую документацию.

Знаки выделения фрагментов текста:

- отметка или символ выбора,

1) действия, которые нужно выполнять в заданной последовательности,

— общие перечисления.

Оглавление

1. Назначение

Контроллер предназначен для работы с шаговым двигателем. Может управляться от ПЛК по протоколу Modbus RTU/ASCII, а также работать автономно по заданной программе. Поддерживает режимы дробления от полношагового до 1/256, что обеспечивает низкий уровень вибрации и высокую точность позиционирования.

Контроллер поддерживает следующие режимы работы:

— программный режим позволяет работать автономно по заданной программе или напрямую управлять двигателем по протоколу Modbus;
— ручной режим позволяет задавать скорость вращения двигателя потенциометром лицевой панели;
— режим драйвера позволяет задать положение ротора двигателя логическими сигналами.

Контроллер двигателя имеет 8 логических входов, 10 логических выходов, 2 из которых являются высоковольтными. Их состояние можно считывать или задавать программой пользователя или по протоколу Modbus. Внутренняя программа может быть записана в контроллер и считана из него как через USB-порт, так и через RS-485 по протоколу Modbus. Предусмотрена защита от перегрева. Для конфигурирования контроллера, создания и редактирования внутренней программы поставляется бесплатное программное обеспечение.

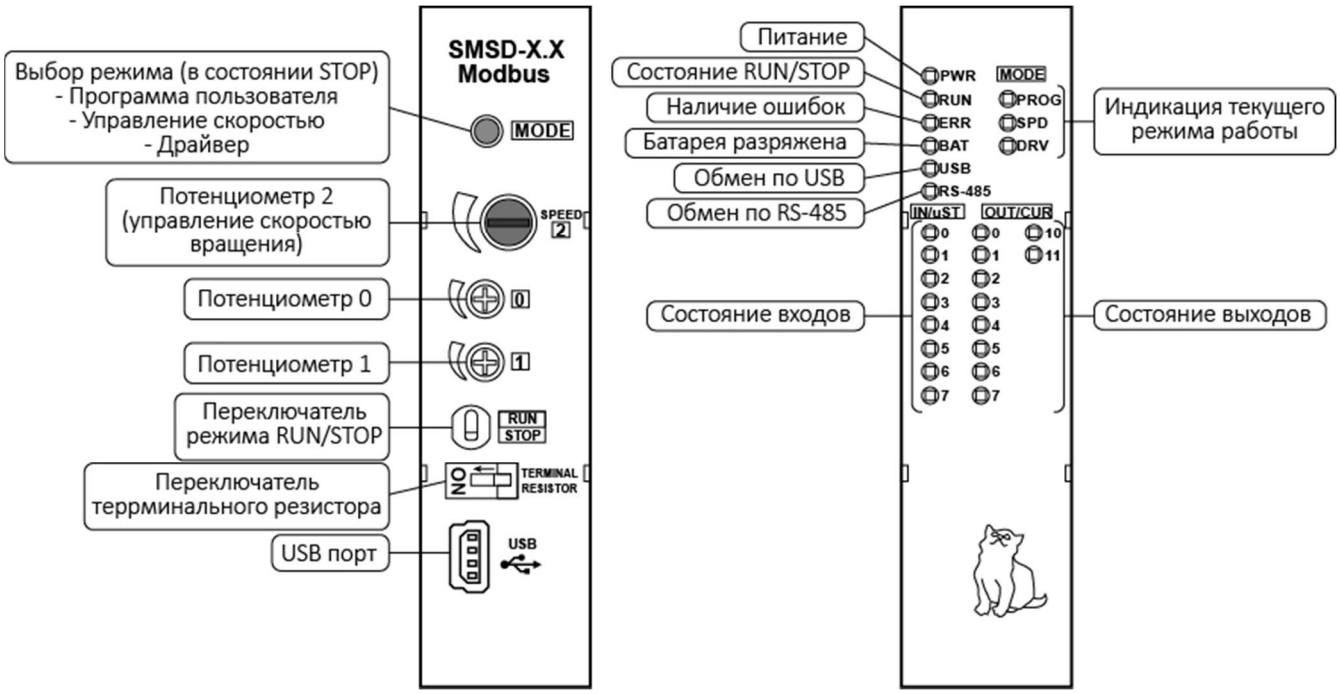


Рисунок 1 — Назначение органов управления.

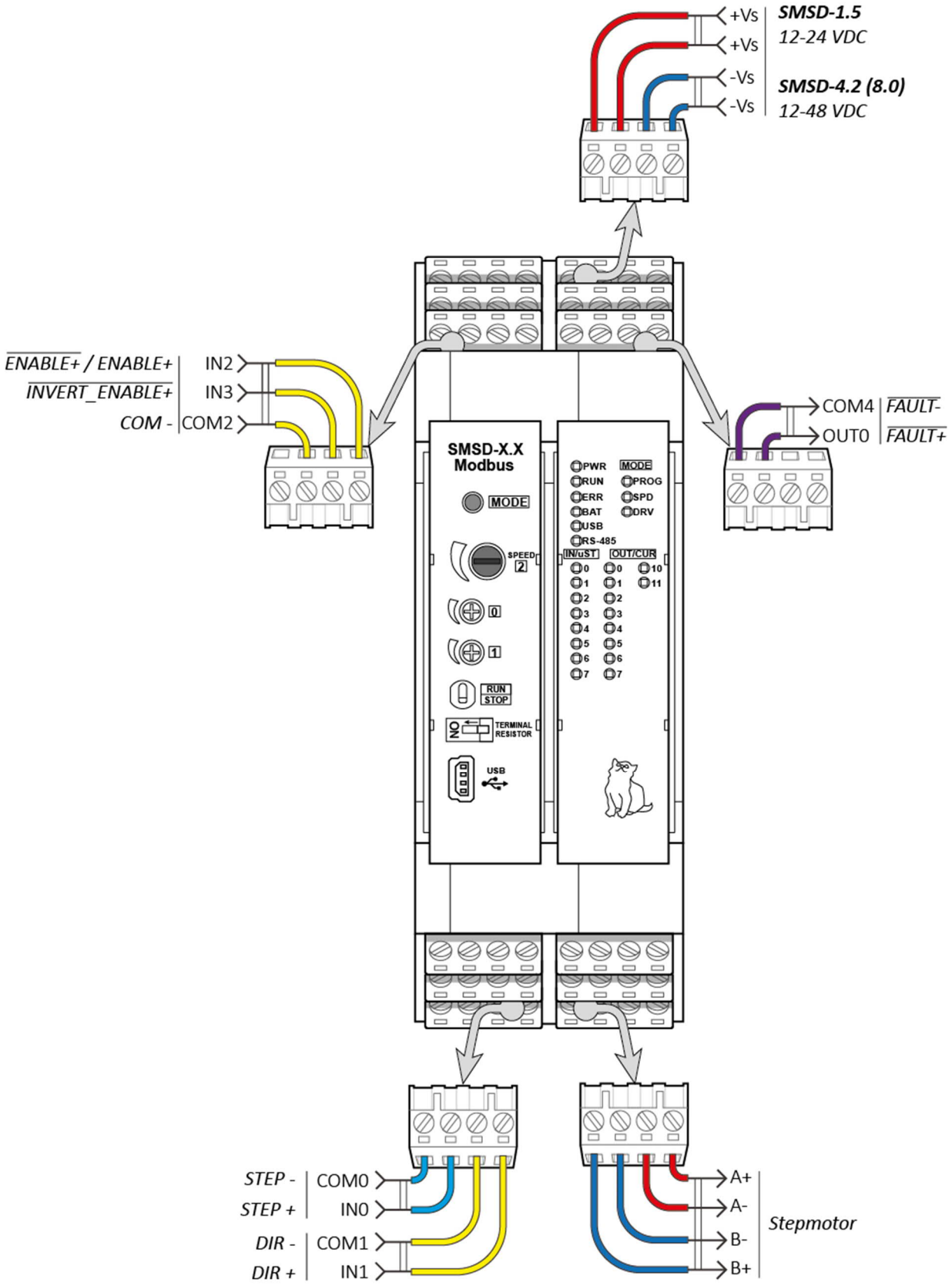


Рисунок 2а – Назначение выводов контроллера – режим драйвера.

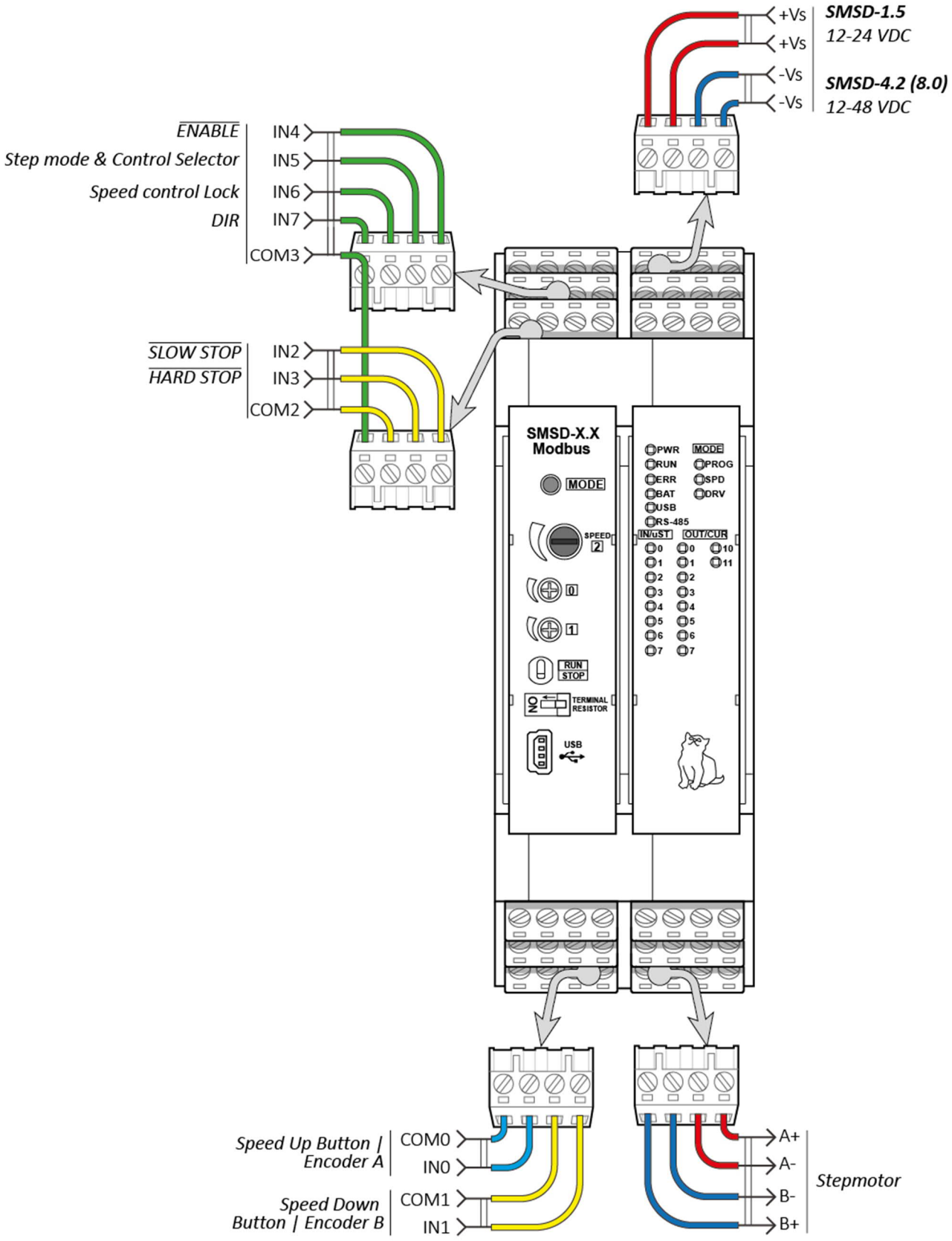


Рисунок 2б – Назначение выводов контроллера – режим управления скоростью вращения.

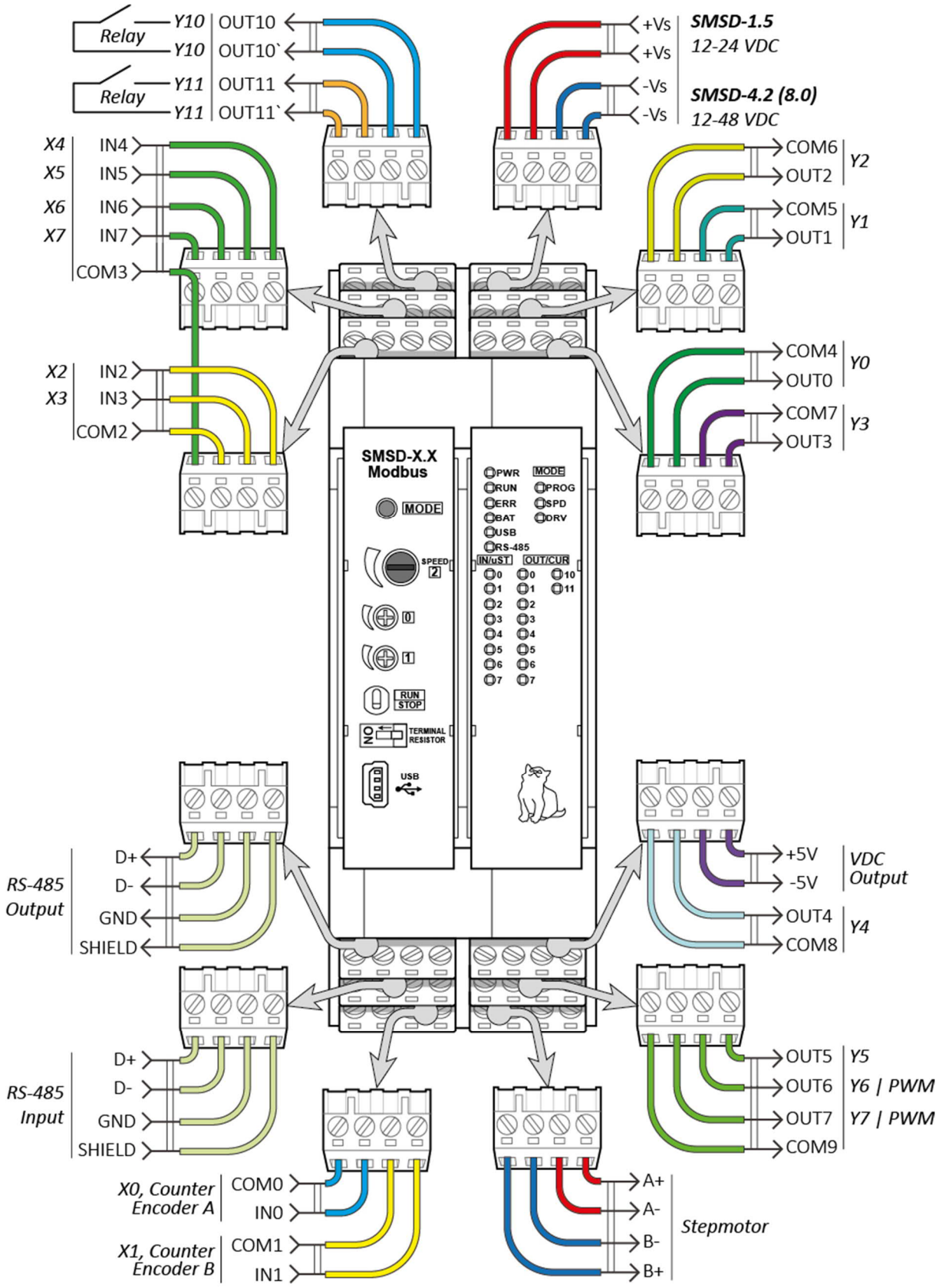


Рисунок 2в – Назначение выводов контроллера – режим пользовательской программы.

На рисунке 1 изображена лицевая панель контроллера с органами управления и индикации. Индикатор **PWR** обозначает наличие питающего напряжения на контроллере двигателя. **RUN** указывает на текущее состояние контроллера (RUN или STOP). В режиме **PROG** (выполнение программы пользователя) и в режиме **SPD** (управление скоростью вращения) индикатор **RUN** в активном состоянии свидетельствует о выполнении программы, неактивное состояние говорит об остановке выполнения програм-

мы. Для режима **DRV** (драйвер) неактивное состояние **RUN** указывает на возможность задать параметры драйвера органами управления, активное состояние свидетельствует о переходе в рабочий режим, в котором изменение параметров блокируется. Так же в состоянии **RUN** блокируется возможность перехода между режимами PROG/SPD/DRV, осуществляемого кнопкой **MODE**. Индикация **ERR** свидетельствует о наличии ошибок, **BAT** – о разряженной батарее внутри блока. **USB** и **RS485** сигнализируют об отправке Modbus кадра через USB и RS-485 соответственно. В режиме **PROG** и **SPD** светодиоды **IN0…IN7** указывают на наличие сигнала высокого логического уровня на соответствующем входе, активное состояние индикатора OUT0…OUT11 говорит об открытом состоянии транзисторного выхода (см. рис. 3 – 6). В режиме драйвера светодиоды входов **IN0…IN7** отображают дробление шага, а выходов рабочий ток **OUT0…OUT3** и ток удержания **OUT4…OUT7**.

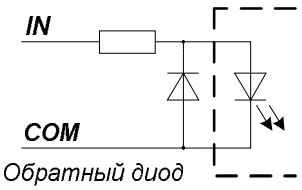


Рисунок 3 – Схемотехника входов 0 и 1.

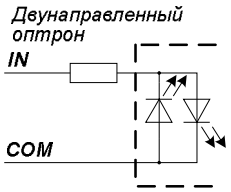


Рисунок 4 – Схемотехника входов 2 – 7.

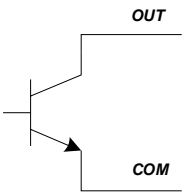


Рисунок 5 – Схемотехника выходов 0 – 7.

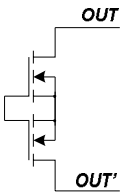


Рисунок 6 – Схемотехника выходов 10 и 11.

Положение ручек потенциометров **0, 1, 2 (SPEED)** преобразуется при помощи аналого-цифрового преобразования в числовое 12-битное значение, которое можно использовать в программе пользователя, а также считывать по протоколу Modbus. В режиме управления скоростью **SPD** потенциометром **2 (SPEED)** задаётся скорость вращения, потенциометром **0** – ускорение и торможение, **1** – рабочий ток. В режиме драйвера **DRV** потенциометр **2** задаёт дробление, **0** – ток рабочего режима, **1** – ток режима удержания.

Переключатель **RUN/STOP** запускает и останавливает выполнение программы в режимах PROG и SPD, в режиме драйвера позволяет устанавливать и применять установленные параметры.

На рисунке 2 показаны выводы контроллера и их назначение в зависимости от режима работы.

2. Технические характеристики

В таблице ниже представлены параметры контроллера.

Параметр	Значение	
	мин.	макс.
Напряжение питания, В		
SMSD-1.5Modbus ver.3	12	36
SMSD-4.2Modbus и SMSD-8.0Modbus	12	49
Ток обмоток двигателя, А		
SMSD-1.5Modbus ver.3	0,15	1,5
SMSD-4.2Modbus	1,0	4,2
SMSD-8.0Modbus	2,8	8,0
Напряжение установки логического входа, В		2,4
Напряжение сброса логического входа, В	0,7	
Напряжение логического выхода транзисторного типа, В		80
Нагрузочная способность логического выхода транзисторного типа, мА		50
Напряжение логического выхода релейного типа AC/DC, В		350
Нагрузочная способность логического выхода релейного типа DC(AC/DC), мА		250 (~120)
Нагрузочная способность выхода вспомогательного питания +5В, мА		200
Продолжительность высокого логического уровня сигнала STEP в режиме драйвера, нс	250(1)	
Продолжительность низкого логического уровня сигнала STEP в режиме драйвера, нс	250(1)	
Продолжительность высокого логического уровня для входов IN0 и IN1, нс	70(1)	
Продолжительность низкого логического уровня для входов IN0 и IN1, нс	70(1)	
Продолжительность высокого логического уровня для входов IN2…IN7, мкс	5(1)	
Продолжительность низкого логического уровня для входов IN2…IN7, мкс	5(1)	
Частота генерации ШИМ-сигнала, Гц	0,3	50000
Время обработки одной базовой инструкции, мкс(2)	20	

(1) – *при напряжении 5В на логическом входе*

(2) – *без учёта времени возврата к 0-ой строке, установки выходов и опроса входов*

Дополнительная информация.

Параметр	Значение
Поддерживаемые скорости передачи данных по RS-485, бод/с	1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000
Дробление полного шага	1/1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/128, 1/256
Поддерживаемые протоколы	Modbus RTU, Modbus ASCII
Язык программирования	IL (Instruction List), LD(1) (Ladder Diagram)

(1) – *при использовании дополнительного программного обеспечения.*

3. Описание работы контроллера

Контроллер работает следующим образом:

— чтение состояния внешних входных устройств (логические входы, Modbus Coils),
— обработка процессором предварительно заданной программы,
— установка нового состояния выходных устройств (логические выходы, Modbus Discrete Inputs, выполнение перемещения).

Программа состоит из последовательности отдельных управляющих инструкций, которые определяют функционал. Контроллер обрабатывает инструкции последовательно, т.е. одну за другой. Общий проход программы непрерывно повторяется. Время, необходимое для прохода программы называется временем цикла, а проходы программы — циклическим сканированием.

Контроллеры способны работать в реальном масштабе времени и могут быть использованы как для построения узлов локальной автоматики, так и систем распределенного ввода-вывода с организацией обмена данными по RS-485 интерфейсу с протоколом Modbus.

Для удобства отладки и написания программ разработчики предусмотрели пакет программирования, который не требует существенных ресурсов компьютера и является простым инструментом для всех категорий специалистов. Используются два языка программирования: LD (релейно-контактная логика или лестничные диаграммы), IL (список инструкций).

3.1. Принцип работы релейно-контактных схем и лестничных диаграмм в контроллере

Язык лестничных диаграмм является производной от релейно-контактной принципиальной электрической схемы в упрощенном представлении. Релейно-контактные схемы в контроллере имеют набор базовых компонентов, таких как: нормально-открытый контакт, нормально-закрытый контакт, катушка (выход), таймер, счетчик и т.д., а также прикладные инструкции: математические функции, команды управления двигателем, обработки данных и большое количество специальных функций и команд. Можно считать, что контроллер — это десятки или сотни отдельных реле, счетчиков, таймеров и память. Все эти счетчики, таймеры, и т.д. физически не существуют, а моделируются процессором и предназначены для обмена данными между встроенными функциями, счетчиками, таймерами и др.

Язык релейно-контактной логики в контроллере по используемым условно-графическим обозначениям очень похож на принципиальные релейно-контактные электрические схемы. В релейно-контактных схемах могут быть два типа логики: комбинированная, т.е. схема, состоящая из независимых друг от друга фрагментов, и последовательная логика, когда все шаги программы взаимосвязаны и схема не поддается распараллеливанию.

Комбинированная логика.

Первый сегмент схемы состоит из одного нормально-открытого контакта X0 и катушки Y0, определяющей состояние выхода Y0. При разомкнутом состоянии (логический "0") контакта X0, выход Y0 также будет разомкнут (логический "0"). При замыкании контакта X0 выход Y0 также изменит свое состояние на замкнутое (логическая "1").

Второй сегмент схемы состоит из одного нормально-закрытого контакта X1 и катушки Y1, определяющей состояние выхода Y1. В нормальном состоянии контакта X1, выход Y1 бу-

дет замкнут (логическая "1"). При изменении состояния контакта X1 на разомкнутое, выход Y1 также изменит свое состояние на разомкнутое.

На третьем сегменте схемы состояние выхода Y2 зависит от комбинации состояний трех входных контактов X2, X3 и X4. Выход Y2 будет замкнут, когда X2 выключен и X4 включен или когда X3 и X4 включены.

Общая схема является комбинацией трех сегментов, работающих независимо друг от друга.

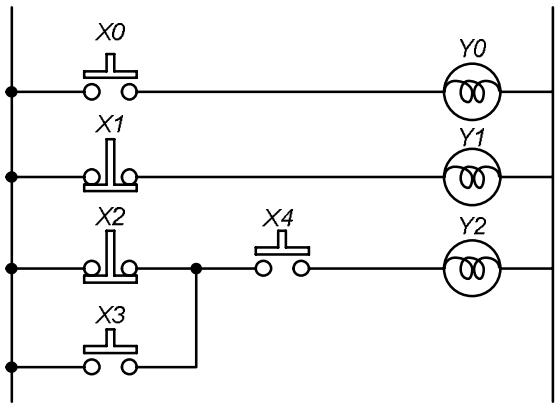


Рисунок 7 – Релейно-контактная электрическая схема.

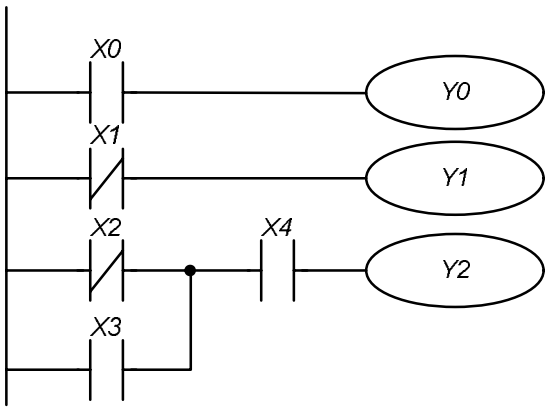


Рисунок 8 – Лестничная диаграмма в контроллере.

Последовательная логика.

В схемах с последовательной логикой результат выполнения предыдущего шага является начальным условием для последующего шага, т.е. выход в предыдущем шаге является входом в следующем шаге.

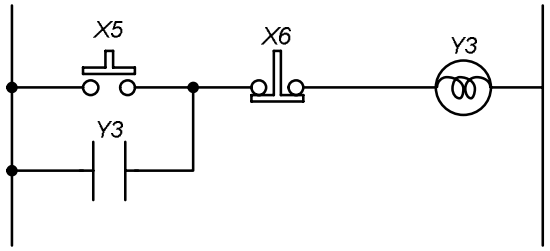


Рисунок 9 – Релейно-контактная электрическая схема.

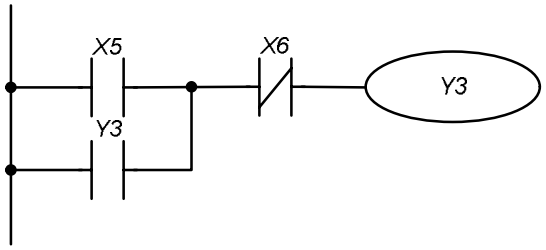


Рисунок 10 – Лестничная диаграмма в контроллере.

При замыкании контакта X5, выход Y3 изменит свое состояние на замкнутое, однако при размыкании контакта X5, выход Y3 сохранит свое замкнутое состояние до тех пор, пока не будет включен вход X6. Контакт Y3 является самоблокировочным.

3.2. Различия между логикой лестничных диаграмм в контроллере и физическими релейно-контактными электрическими схемами

В обычных релейно-контактных электрических схемах все задаваемые управляющие процессы выполняются одновременно (параллельно). Каждое изменение состояние входных сигналов сразу же действует на изменение состояния выходных сигналов.

При управлении контроллером изменение состояния входных сигналов, произошедшее во время текущего прохода программы, опознается только на следующем цикле программы. Этот недостаток контроллера сглаживается благодаря короткому времени цикла.

Время выполнения одного цикла программы зависит от количества выполняемых инструкций в программе и от типа используемых инструкций.

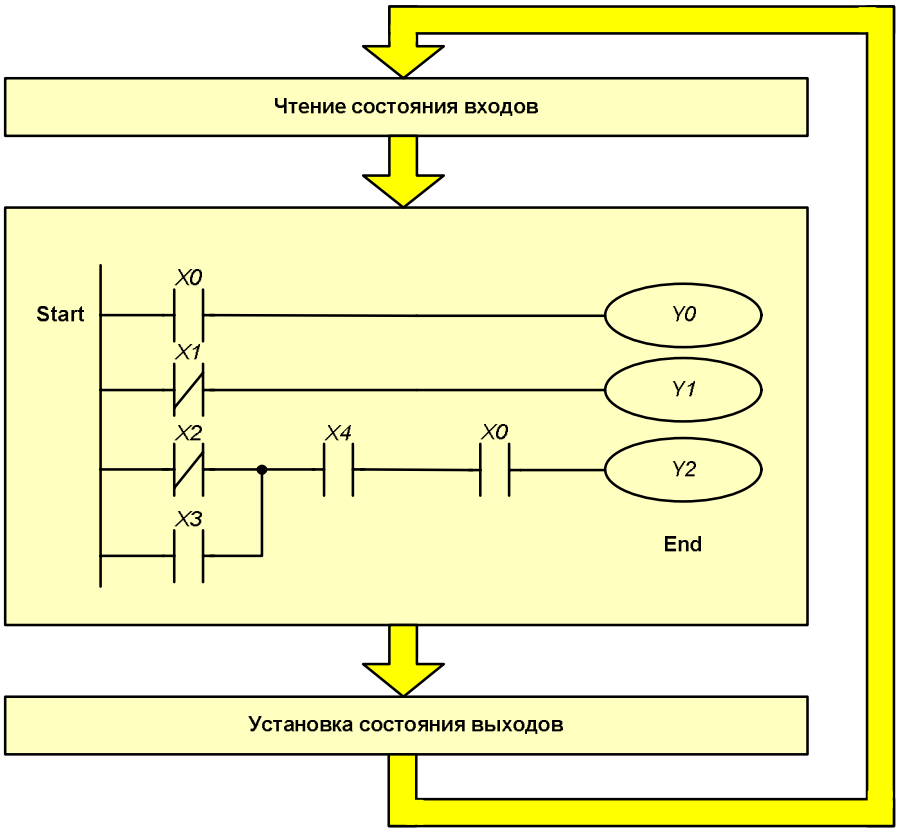


Рисунок 11 – Процесс работы контроллера.

В процессе работы контроллер непрерывно опрашивает текущее состояние входов и изменяет состояние выходов (вкл./выкл.) в зависимости от программы пользователя.

На рисунке 11 изображена структурная схема одного цикла программы.

На первом этапе происходит считывание состояния физических и виртуальных (состояние которых задаётся через Modbus Coils) входов и их буферизация во внутренней памяти контроллера.

На втором этапе происходит обработка состояния буферизированных входов и изменение состояния выходов в памяти контроллера по заданной программе пользователя. Таким образом, все модификации выходов происходят без изменения их физического состояния. В процессе выполнения данного этапа может измениться состояние физических и виртуальных входов, но следующая буферизация обновлённого состояния произойдёт на первом этапе следующего цикла программы пользователя.

На третьем этапе контроллер изменяет состояние физических и виртуальных выходов.

Еще одно отличие релейно-контактной логики контроллера от обычных релейно-контактных электрических схем заключается в том, что выполнение программ в строках идет только слева направо и сверху вниз, так, например, схема с реверсивным направлением тока (участок a-b на рис. 12) при компиляции приведёт к ошибке.

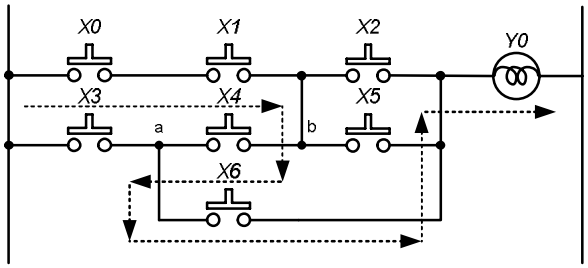
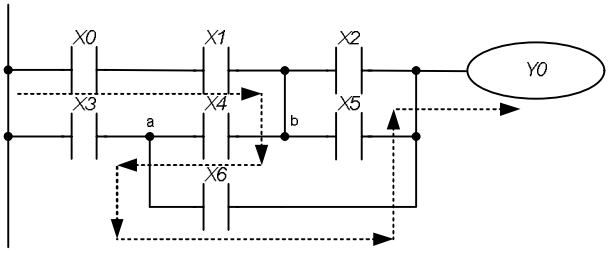


Рисунок 12 – Релейно-контактная электрическая схема.



Ошибка в третьей строке

Рисунок 13 – Релейно-контактная логика контроллера.

3.3. Операнды

Все внутренние объекты контроллера, или операнды, подразделяются на различные типы и имеют адреса. Каждый тип имеет свое обозначение и свой формат, который определяет количество занимаемого места в памяти контроллера. Так, например, входные реле обозначаются "X" и имеют однобитный формат, а регистры данных общего назначения обозначаются "D" и имеют 16-ти битный (1 слово) или 32-х битный (2 слова) формат.

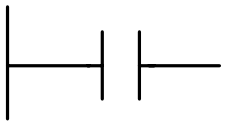
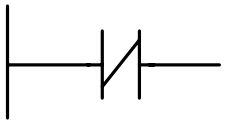
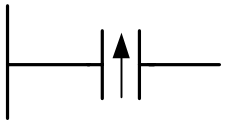
Тип и обозначение операнда		Описание
Вход	X	Входные реле. Определяют состояние внешних битовых устройств, подключенных к входным клеммам контроллера и виртуальных входов, состояние которых можно задать по протоколу Modbus. Могут принимать одно из двух состояний: 0 или 1. Адресация ведется в восьмеричной системе: X0, X1, ... X7, X10, X11, ...
Выход	Y	Выходные реле. Определяют состояние выходных клемм контроллера, к которым подключается нагрузка и виртуальных выходов, состояние которых можно опросить по протоколу Modbus. В программе могут быть как контактами, так и катушками, и принимать одно из двух состояний: 0 или 1. Адресация ведется в восьмеричной системе: Y0, Y1, ... Y7, Y10, Y11, ...
Меркер	M	Внутренние (вспомогательные) реле. Память для двоичных промежуточных результатов. В программе могут быть как контактами, так и катушками, и принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: M0, M1,... M7, M8, M9,...
Таймер	T	Реле времени. В программе могут использоваться для хранения текущего значения таймера и иметь 16-ти битный формат, а также могут быть контактами, и принимать одно из двух состояний: 0 или 1. Адресация ведётся в десятичной системе: T0, T1, ..., T64

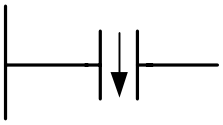
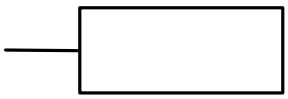
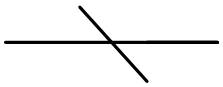
Счётчик	C	Используются для реализации счета. В программе могут использоваться для хранения текущего значения счетчика и иметь 16-ти или 32-х битный формат, а также могут быть контактами, и принимать одно из двух состояний: 0 или 1. Адресация ведется в десятичной системе: C0, C1, ..., C66
Десятичная константа	K	Определение числа в десятичной системе отсчета
Шестнадцатеричная константа	H	Определение числа в шестнадцатеричной системе отсчета
Константа с плавающей запятой	F	Определение числа в системе отсчета с плавающей запятой
Регистр данных	D	Память данных. 16-ти или 32-х битный формат. Адресация ведется в десятичной системе: D0, D1,..., D391. В 32-х битном формате один регистр занимает две ячейки, например при обращении к D10, данные будут прочитаны из ячеек D10 и D11
Индексный регистр	A	Память данных для промежуточных результатов и индексной идентификации. 16-ти битный формат. Адресация: A0 – A7, B0 – B7 в десятичной системе отсчёта
	B	
Указатель	P	Адрес для перехода к подпрограмме в десятичной системе
Указатель прерывания	I	Адрес обработки прерывания в десятичной системе отсчёта

3.4. Виды изображения управляющих инструкций

Релейно-контактная схема состоит из одной вертикальной линии, расположенной слева и горизонтальных линий, отходящих вправо. Вертикальная линия называется шиной, а горизонтальная – командной линией или ступенькой. На командной линии располагаются символы условий, ведущие к командам (инструкциям), расположенным справа. Логические комбинации этих условий определяют, когда и как выполняются правосторонние команды.

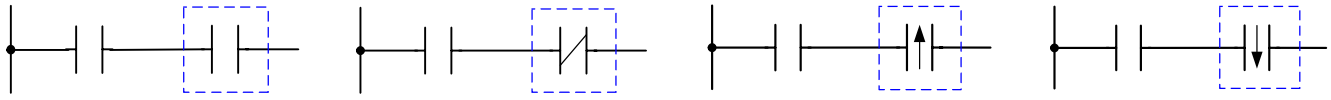
В релейно-контактных схемах в основном применяется следующая символика:

Символ	Пояснение	Команда	Операнд
	Символ для входного сигнала (нормально-открытого контакта а)	LD	X, Y, M, T, C
	Символ для входного сигнала (нормально-закрытого контакта Ь)	LDI	X, Y, M, T, C
	Символ для входного импульсного сигнала, (с опросом по переднему фронту)	LDP	X, Y, M, T, C

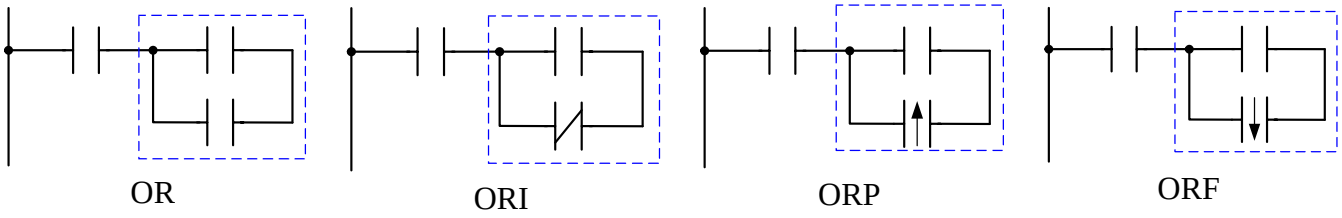
	Символ для входного импульсного сигнала, (с опросом по заднему фронту)	LDF	X, Y, M, T, C
	Символ для выходного сигнала (катушки)	OUT	Y, M
	Символ для прикладных инструкций	см. главу 7. Описание прикладных	см. главу 7. Описание прикладных
	Символ логической инверсии	INV	нет

Входные релейные контакты могут объединяться в последовательные, параллельные и комбинированные схемы:

Последовательное подключение:



Параллельное подключение:



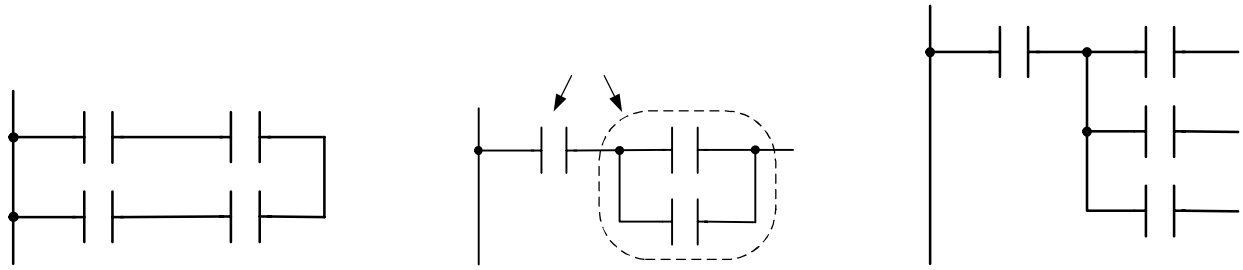


Рисунок 14 – Графическое изображение управляющих инструкций.

Сканирование программы начинается от левого верхнего угла схемы и заканчивается в правом нижнем углу. Следующий пример иллюстрирует последовательность выполнения программы:

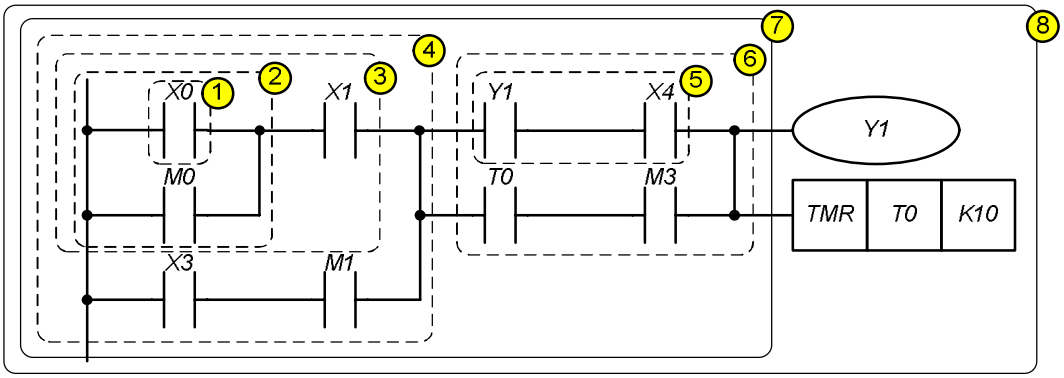
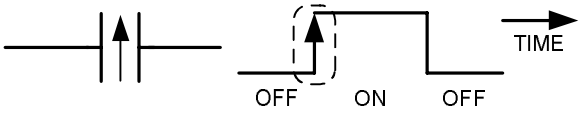


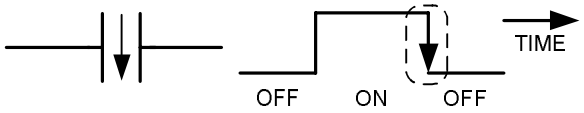
Рисунок 15 – Последовательность выполнения программы.

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

Символы входных сигналов с опросом по переднему фронту (при переходе сигнала с 0 на 1) и с опросом по заднему фронту (при переходе сигнала с 1 на 0) поясняются ниже:



По переднему фронту



По заднему фронту

Рисунок 16 – Фильтрация по фронтам.

Команды логического блока ANB и ORB не соответствуют конкретным условиям на релейно-контактной схеме, а описывают отношения между блоками. Команда ANB производит операцию ***ЛОГИЧЕСКОЕ И*** над условиями исполнения, произведенными двумя логическими блоками. Команда ORB производит операцию ***ЛОГИЧЕСКОЕ ИЛИ*** над условиями исполнения, произведенными двумя логическими блоками.

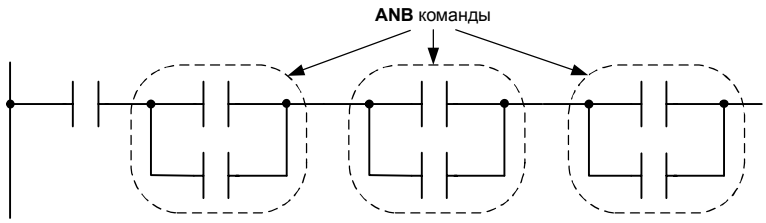


Рисунок 17 – Инструкция ANB.

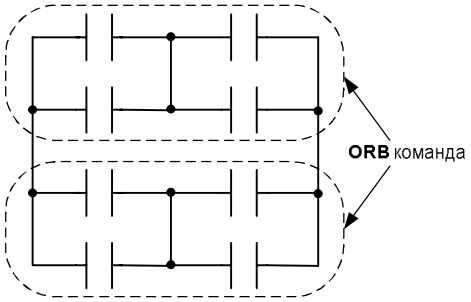


Рисунок 18 – Инструкция ORB.

3.5. Преобразование релейно-контактных схем в мнемокод

На ниже приведенном рисунке показана программа, представленная в виде релейно-контактной символики и виде списка инструкций (мнемокода). На рисунке виден порядок преобразования лестничной диаграммы в код, исполняемый контроллером.

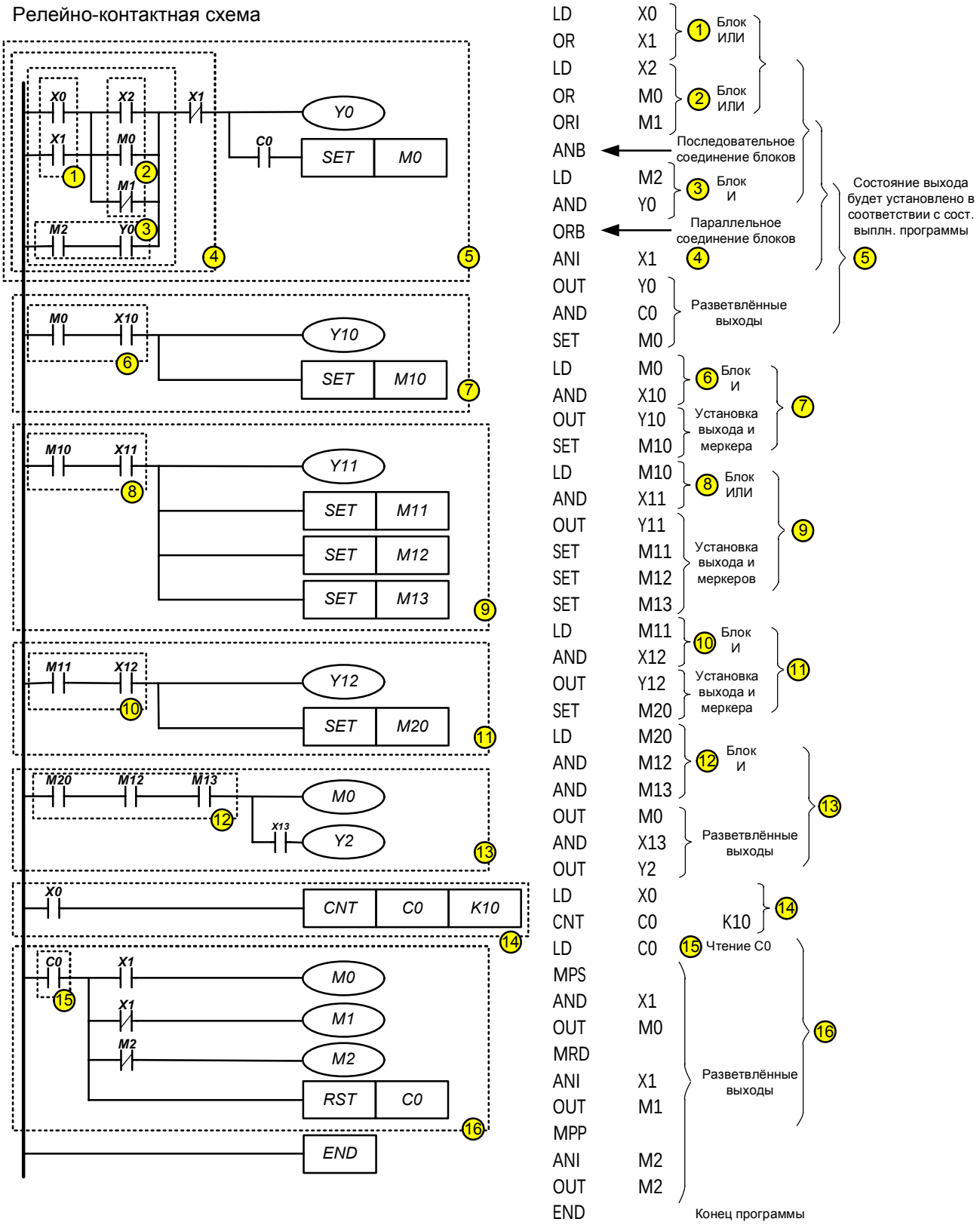
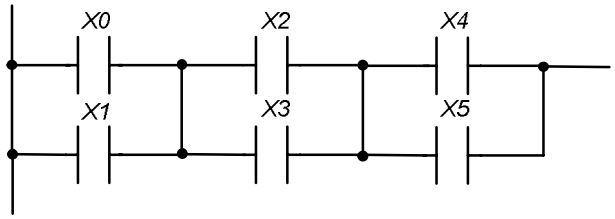


Рисунок 19 – Преобразование LD в IL.

Процесс обработки релейно-контактной схемы идет с верхнего левого угла и заканчивается в правом нижнем, однако могут быть исключения и различные варианты преобразования в мнемокод, как показано в следующих примерах:

Пример 1. Ниже приведенную схему можно кодировать двумя различными методами, однако результат будет тождественным.

Первый метод кодирования является наиболее предпочтительным, так как число логических блоков не ограничено.

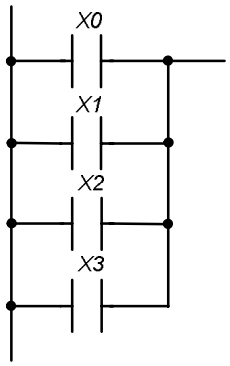


Метод 1		**Метод 2**	
LD	X0	LD	X0
OR	X1	OR	X1
LD	X2	LD	X2
OR	X3	OR	X3
ANB		LD	X4
LD	X4	OR	X5
OR	X5	ANB	
ANB		ANB	

Рисунок 20 – Варианты преобразования инструкции ANB.

Во втором методе можно объединить максимум 8 логических блоков.

Пример 2. Различное кодирование параллельно соединенных контактов.



Метод 1		**Метод 2**	
LD	X0	LD	X0
OR	X1	LD	X1
OR	X2	LD	X2
OR	X3	LD	X3
		ORB	
		ORB	
		ORB	

Рисунок 21 – Варианты преобразования инструкции ORB.

Первый метод кодирования является наиболее предпочтительным с точки зрения использования оперативной памяти контроллера.

4. Функционал контроллера

4.1. Обзор операндов

Тип	Операнд			Диапазон адресов		Назначение
Реле (1-но битная память)	**X**	Внешние входные реле	Физические входы	X0…X7	Макс. 128 точек	Входы контроллера
			Виртуальные входы (Modbus Coil)	X10…X177		
	Y	Внешние выходные реле	Физические выходы	Y0…Y7	Макс. 128 точек	Выходы контроллера
			Виртуальные выходы (Modbus Discrete Inputs)	Y10…Y177		
	M	Внутренние реле (меркеры)	Общие	M0…M99, M111…M127	Макс. 128 точек	Промежуточная двоичная память. Соответствует промежуточным реле в электрических схемах
			Специальные	M100…M110		
	T	Таймеры	Дискретность 100 мс	T0…T47 (T46, T47 – аккумулятивные)	Макс. 64 точек	Используются в качестве контактов (T), которые замыкаются при достижении соотв. таймером (команда TMR) своего заданного значения
			Дискретность 10 мс	T48…T63 (T62, T63 – аккумулятивные)		
	C	Счётчики	Инкрементный общего назначения	C0…C63	Макс. 66 точек	Используются в качестве контактов (C), которые замыкаются при достижении соотв. счетчиком (команда CNT) своего заданного значения
			Внешних импульсов	C64, C65		
Регистр (16-ти битная память)	**T**	Текущее значения таймера		64 точки (T0…T63)		Регистры для хранения тек. знач. таймеров
	C	Текущее значение счётчика		66 – 32-х битных счётчика		Регистры для хранения тек. знач. счётчиков
	D	Регистры данных	Общего назначения	D0…D319, D385…D391	Макс. 384 точки	Используются для хранения данных. Специальные регистры конфигурируют контроллер и отображают статус
			Энергонезависимые[(1)]	D320…D351		
			Специальные	D352…D384		
	A	Индексные регистры	младшие 16 бит	A0…A7	Макс. 16 точек	Индексная идентификация операндов
	B		старшие 16 бит	B0…B7		

Указатели	**P**	Указатели для инструкций CALL, CJ		32 точки (P0…P31)		Метки для операторов перехода, подпрограмм
	I	Прерывания	Коммуникац.	I0	Макс. 15 точек	Метки для программ обработки прерываний
			Временные	I1…I100 (Макс. 4 точки)		
			Внешние	I1000…I1007		
			Драйвера	I2000, I2001		
Константы	**K**	Десятичные константы		K-32768 ...K32767 (16-ти битные операции) K-2147483648 ...K2147483647 (32-х битные операции)		
	H	Шестнадцатеричные константы		H0000...HFFFF (16-ти битные операции) H00000000...HFFFFFFFF (32-х битные операции)		
	F	С плавающей запятой		F±1.175494351 E-38… 3.402823466 E+38 (только 32-х битные операции)		

(1) – сохранение данных обеспечивается внутренним источником питания CR2032.

4.2. Адресация и назначение входов [X] и выходов [Y]

Входы и выходы в программе пользователя контроллера представляются операндами. Посредством указания адреса операнда можно точно обращаться при программировании к физическим и виртуальным входам и выходам контроллера.

Адресация дискретных входов/выходов выполняется в восьмеричной системе, т.е. входы и выходы не нумеруются числами с использованием цифр 8 и 9.

Назначение входных реле X

Входные реле X считывают состояния внешних физических устройств (кнопки, переключатели, контакты реле и др.) непосредственно подключенных к входным клеммам контроллера. Каждый входной контакт X может использоваться в программе неограниченное число раз.

Назначение выходных реле Y

Выходные реле Y управляют состоянием физических выходных контактов контроллера, а следовательно и устройствами нагрузки (лампы, катушки реле и др.) непосредственно подключенными к выходным клеммам контроллера.

Каждый выходной контакт Y может использоваться в программе неограниченное число раз, но выходную катушку Y рекомендуется использовать в программе не более одного раза, т.к. при повторении катушки Y, состояние выхода будет определяться последним Y в скане.
Состояние сигналов входов/выходов может опрашиваться в программе с помощью различных инструкций.

Процесс обработки в контроллере входных/выходных сигналов:
Входы:

1. Контроллер будет считывать состояние внешних входных устройств, и помещать в память в начале каждого цикла сканирования.

2. Изменения состояния входа во время цикла не будут восприняты, если входной импульс очень короткий (меньше времени одного скана).

Программа:

3. ПЛК выполняет программу, начиная со строки 0, и сохраняет состояния всех операндов в памяти объектов.

Выходы:

4. После выполнения инструкции END состояния выходных реле Y будут переписаны в память состояния выходов и состояния выходных контактов будут изменены.

4.3. Адресация и назначение внутренних реле [M]

Для запоминания двоичных результатов логических связей (состояний сигналов "0" или "1") внутри программы применяется промежуточная память (внутреннее реле). Они соответствуют промежуточным реле в системах управления на релейной логике.

В контроллерах используется два типа внутренних реле:

1. Общие, которые не сохраняют свое состояние при отключении питания;
2. Специальные, которые предоставляют в распоряжение пользователя дополнительные функции.

Внутренние реле программируются как выходы. Они могут использоваться в программе неограниченное число раз.

Адресация внутренних реле выполняется в десятичном формате.

Назначение специальных меркеров:

Номер	Функция
M100…M107	Данные вспомогательные реле используются только совместно с прерываниями от внешних входов I1000…I1007 соответственно. Значение меркера соответствует состоянию физического входа (IN0…IN7) в момент обработки прерывания (I1000…I1007). Значения X0…X7 будут обновлены только вначале следующего сканирования пользовательской программы. Например, после попадания в обработчик прерывания I1004, запросив состояние M104 (LD M104), можно определить состояние входа IN4 (значение X4 в данном случае не актуально).
M108	Нарастающий фронт данного вспомогательного реле свидетельствует о завершении инициализации периферии контроллера. В течении последующей работы меркер сохраняет значение высокого уровня. Сброс меркера приводит к переинициализации контроллера. Например, после переопределения выходов генераторами ШИМ-сигнала и входов счётчиками импульсов, требуется переинициализация, для этого необходимо сбросить M108.
M109	Установка меркера включит индикацию «ERR» на лицевой панели контроллера, сброс – отключит.
M110	Установка меркера с последующим сбросом M108 приведёт к полной перезагрузке контроллера.

4.4. Адресация и назначение таймеров [T]

Для многих процессов управления необходимы реле времени. В релейной технике для этого применяются реле времени с задержкой на включение. В контроллере для этих целей используются внутренние элементы памяти, называемые таймеры, характеристики которых могут определяться программой.

Адресация таймеров выполняется в десятичном формате.

T	Таймеры	Дискретность 100 мс	T0…T47 (T46, T47 – аккумулятивные)	Макс. 64 точек
		Дискретность 10 мс	T48…T63 (T62, T63 – аккумулятивные)	

Требуемая уставка времени определяется с помощью десятичной константы **K**, которая указывает количество отсчитываемых шагов времени (дискрет).

Пример: Для таймера с дискретностью 100 мс, у которого уставка времени задана как K5, действительное значение уставки будет равно 5 x 100 = 500 мс.

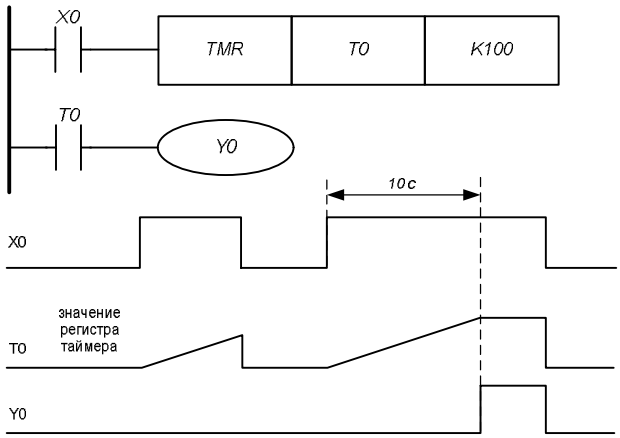
Таймер работает с задержкой на включение. Он активизируется состоянием входного контакта = 1. После отсчета установленного значения времени таймер устанавливает в состояние "1" соответствующий контакт T. Таймер возвращается в отключенное состояние и обнуляет свое текущее значение при установке своего входного контакта в "0".

Задание уставки времени может выполняться также косвенно посредством записанного ранее в регистр данных десятичного числа.

В контроллерах таймер начинает отсчет времени сразу с выполнением команды TMR.

Пояснение работы двух типов таймеров:

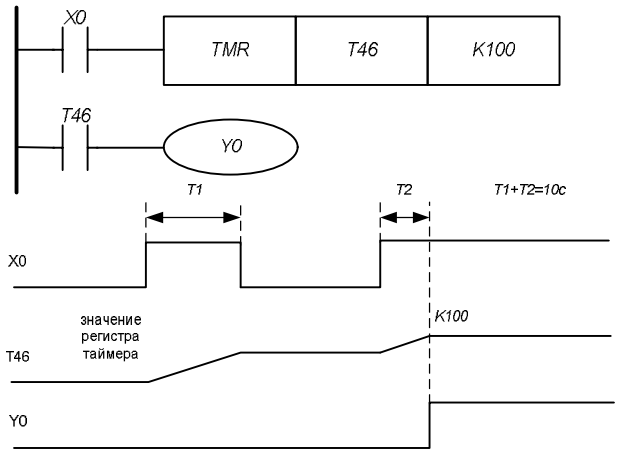
— **Таймер общего назначения.**



Если вход X0 принимает состояние "1" начинается отсчет заданного времени. После отсчета запрограммированных 10 сек выход Y0 примет состояние "1". Таймер отключится и регистр T0 обнулится, как только вход X0 примет состояние "0".

Рисунок 22 – Принцип работы таймера общего назначения.

— **Аккумулятивный таймер**



В контроллере наряду с таймерами общего назначения есть аккумулятивные таймеры, которые после отключения управляющей логической связи сохраняют накопленное значение времени.

Рисунок 23 - Принцип работы аккумулятивного таймера.

4.5. Адресация и назначение счетчиков [C]

Для многих процессов управления необходимо считать импульсы (суммировать или вычитать). В релейной технике для этого применяются счетчики импульсов. В контроллере для этих целей используются внутренние элементы памяти, называемые счетчиками, которые могут быть двух видов.

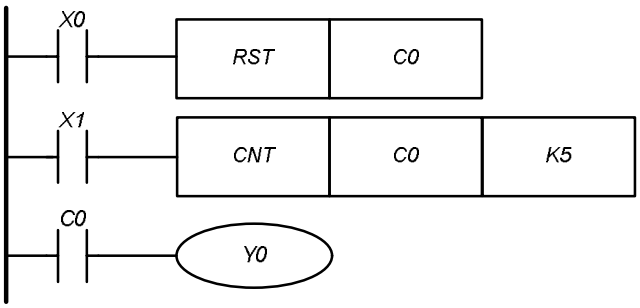
Адресация счетчиков выполняется в десятичном формате.

C	Счётчики	Инкрементный общего назначения	C0…C63	Макс. 66 точек
		Внешних импульсов (аппаратные)	C64, C65	

Работа и назначение счетчиков:

Когда входной сигнал счетчика изменяет свое состояние с 0 на 1, текущее значение счетчика С увеличится на единицу и когда оно станет равным заданному значению (уставке), рабочий контакт счетчика включится.

```
LD      X0
RST     C0
LD      X1
CNT     C0      K5
LD      C0
OUT     Y0
```



Когда X0 = 1, происходит сброс счетчика: текущее значение регистра C0 = 0, контакт C0 разомкнут.

При изменении X1 с 0 на 1, текущее значение регистра C0 будет увеличиваться на 1.

Когда C0 = 5, контакты C0 и Y0 замкнутся и последующие импульсы контакта X1 перестанут восприниматься.

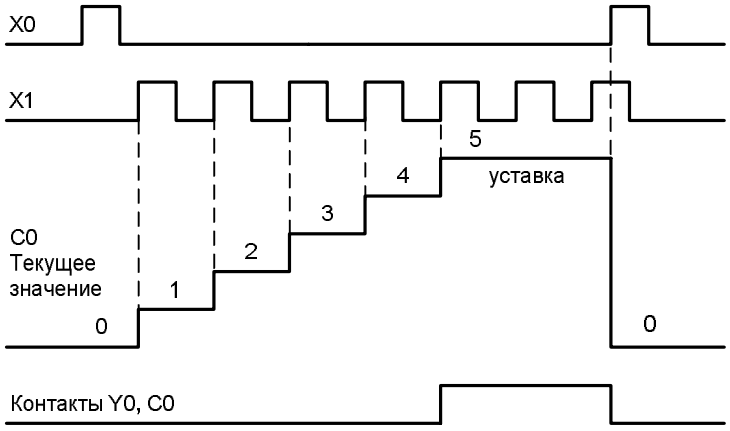
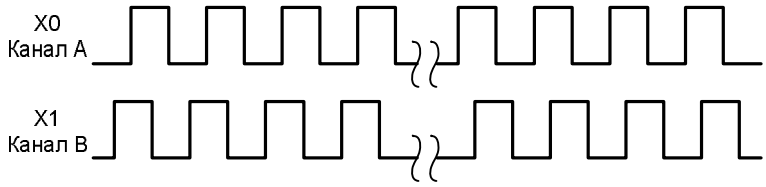


Рисунок 24 – Принцип работы счётчика.

Выше порога счётчики общего назначения не считают, в отличие от аппаратных, счёт который не зависит от входного сигнала, а лишь от физического состояния дискретного входа (входов), к которому он привязан. В зависимости от значения конфигурирующего регистра **D355** счётчики могут быть настроены следующим образом:

Значение регистра D355	**Описание конфигурации**
0	Значение по умолчанию. X0 и X1 (IN0 и IN1) работают в режиме дискретных входов.
1	Дискретный вход X0 связан со счётчиком C64 и работает в режиме счёта импульсов по переднему фронту. X1 работает в режиме дискретного входа.
2	Дискретный вход X0 связан со счётчиком C64 и работает в режиме счёта импульсов по заднему фронту. X1 работает в режиме дискретного входа.
3	Дискретный вход X0 связан со счётчиком C64 и работает в режиме счёта импульсов по обоим фронтам. X1 работает в режиме дискретного входа.
4	X0 работает в режиме дискретного входа. Дискретный вход X1 связан со счётчиком C65 и работает в режиме счёта импульсов по переднему фронту.
5	X0 работает в режиме дискретного входа. Дискретный вход X1 связан со счётчиком C65 и работает в режиме счёта импульсов по заднему фронту.
6	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 соответственно и работают в режиме счёта импульсов по переднему фронту.
7	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 и работают в режиме счёта импульсов по заднему и переднему фронту соответственно.
8	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 и работают в режиме счёта импульсов по обоим и переднему фронту соответственно.
9	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 и работают в режиме счёта импульсов по переднему и заднему фронту соответственно.
10	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 соответственно и работают в режиме счёта импульсов по заднему фронту.
11	Дискретные входы X0 и X1 связаны со счётчиками C64 и C65 и работают в режиме счёта импульсов по обоим и заднему фронту соответственно.

12	Дискретные входы X0 и X1 связаны со счётчиком C64 и работают в режиме энкодера. На входы подаётся квадратурный сигнал. 

4.6. Адресация и назначение регистров [D], [A], [B]

Регистры данных [D]

Регистры представляют память данных внутри контроллера. В регистре можно хранить числовые значения и следующую друг за другом двоичную информацию.

Данные сохраняются в 16-ти битном регистре (D0 и др.), в котором может храниться число от -32768 до +32767. Совместное включение двух 16-ти битных регистров дает 32-х битный "двойной регистр" (D0, D1 и т.д.), в котором может храниться число от -2147483648 до +2147483647.

Адресация регистров данных выполняется в десятичном формате. Для двойного регистра адресация начинается с младшего 16-ти битного регистра.

D	Регистры данных	Общего назначения	D0…D319, D385…D391	Макс. 384 точки
		Энергонезависимые	D320…D351	
		Специальные	D352…D384	

Имеются следующие типы регистров:

Регистр данных (общего назначения):
Регистр без сохранения данных при отключении напряжения питания контроллера.

Регистр данных (энергонезависимый):
Регистр с сохранением данных при отключении напряжения питания контроллера. Питание памяти осуществляется внутренним источником питания CR2032.

Индексный регистр:
Этот регистр служит для запоминания промежуточных результатов и для индицирования операндов.

Специальный регистр:

Для конфигурирования контроллера и активации дополнительных функций предусмотрен ряд специальных регистров.

Более подробные данные см. в таблице ниже:

Номер	Описание	Значение
D352	Регистр содержит данные о положении ручки потенциометра “0” лицевой панели контроллера.	0…4095
D353	Положении ручки потенциометра “1” лицевой панели контроллера.	0…4095
D354	Положении ручки потенциометра “Speed” (“2”).	0…4095

<table>
<tr><td>D355</td><td>Регистр конфигурирует типы входов IN0 и IN1, подробнее см. п.4.5.</td><td>0…12</td></tr>
<tr><td>D356</td><td>Регистр конфигурирует типы выходов OUT6 и OUT7 при использовании инструкции PWM.
<table>
<tr><th rowspan="2">Значение регистра</th><th rowspan="2">Период дискретизации, мс</th><th colspan="2">Назначение выходов</th></tr>
<tr><th>OUT6</th><th>OUT7</th></tr>
<tr><td>0</td><td>–</td><td>выход</td><td>выход</td></tr>
<tr><td>1</td><td>100</td><td>ШИМ-генератор</td><td>выход</td></tr>
<tr><td>2</td><td>10</td><td>ШИМ-генератор</td><td>выход</td></tr>
<tr><td>3</td><td>100</td><td>выход</td><td>ШИМ-генератор</td></tr>
<tr><td>4</td><td>10</td><td>выход</td><td>ШИМ-генератор</td></tr>
<tr><td>5</td><td>100</td><td>ШИМ-генератор</td><td>ШИМ-генератор</td></tr>
<tr><td>6</td><td>10</td><td>ШИМ-генератор</td><td>ШИМ-генератор</td></tr>
</table>
Более подробно о генерации ШИМ-сигнала см. раздел «7. Описание прикладных команд», PWM инструкция.</td><td>0…6</td></tr>
<tr><td>D357…D384</td><td>Режимозадающие и статусные регистры драйвера шагового двигателя. Более подробно см. раздел «8. Управление драйвером шагового двигателя»..</td><td>–</td></tr>
</table>

4.7. Индексные регистры [A], [B]

Индексные регистры применяются для индексирования адресов операндов и изменения значения константы.

Индексный регистр является 16-ти битовым регистром.

В 32-х битовых инструкциях индексные регистры A и B применяются комбинированно. A содержит 16 младших бит, B запоминает 16 старших бит. В качестве адреса назначения указывается индексный регистр A.

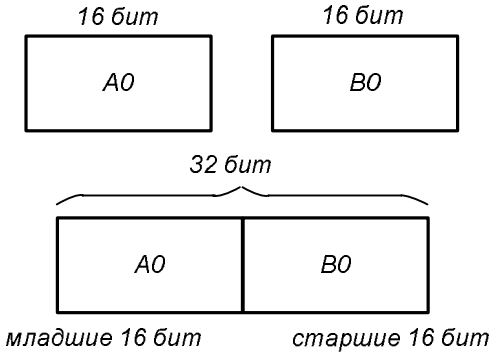


Рисунок 25 – Структура индексного регистра.

Пример передачи данных от регистра данных D5A0 регистру данных D10B0:

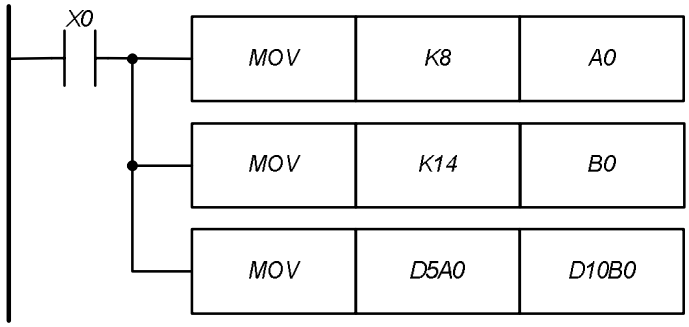


Рисунок 26 – Передача данных с использованием индексации

Когда X0 = 1: A0 = 8, B0 = 14 и значит адрес источника пересылки D5A0 = 5 + 8 = D13, а адрес пересылки D10B0 = 10 + 14 = 24.

И следовательно, имеет место передача данных от регистра D13 к регистру данных D24.

Индексные регистры могут использоваться для операций передачи и сравнения данных совместно с байтовыми операндами и битовыми операндами.

В контроллерах можно индексировать так же и константы.

При индексировании констант необходимо использовать символ @. Например: MOV K10@A0 D0B0.

4.8. Указатели [P], [I].

P	Указатели для инструкций CALL, CJ		32 точки (P0…P31)		Метки для операторов перехода, подпрограмм
I	Прерывания	Коммуникац.	I0	Макс. 15 точек	Метки для программ обработки прерываний
		Временные	I1…I100 (Макс. 4 точки)		
		Внешние	I1000…I1007		
		Драйвера	I2000, I2001		

Указатели (P) используются вместе с инструкциями CJ-перехода или CALL-вызова подпрограммы и являются адресами места перехода, в которых маркируется место перехода или подпрограмма в программе.

Пример выполнения команды перехода CJ:

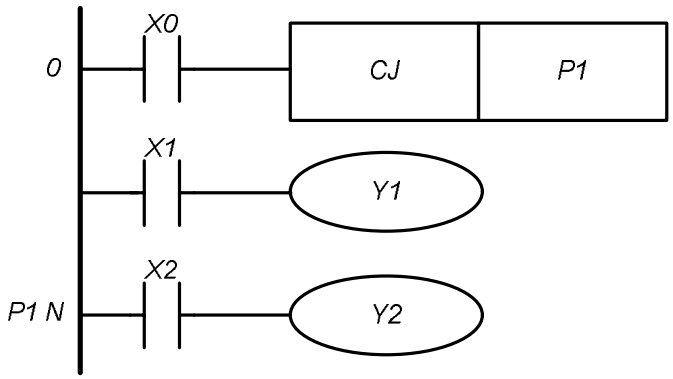
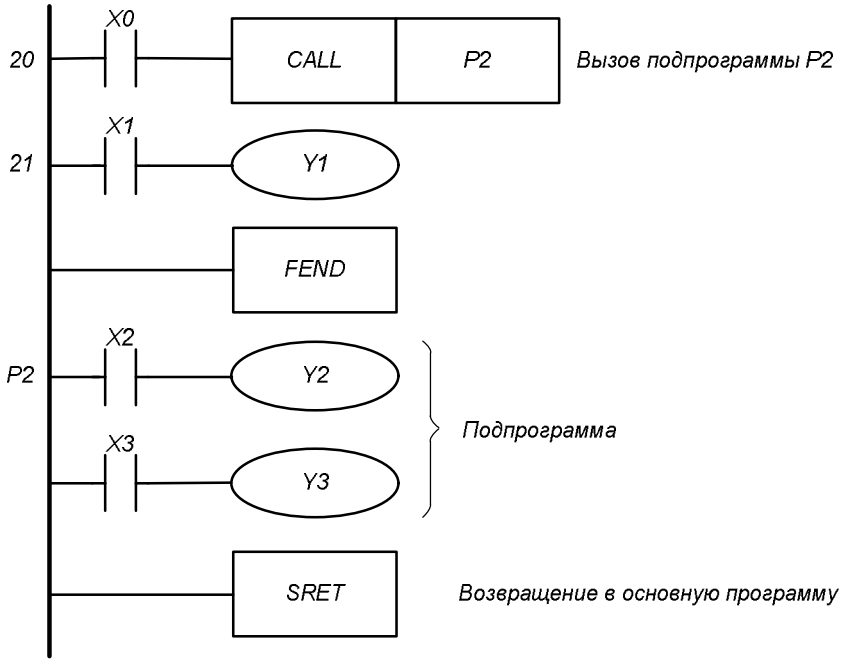


Рисунок 27 – Реализация инструкции CJ.

Когда X0 = 1, после выполнения строки 0 программа сразу переходит к строке с указателем P1 и строки расположенные между ними не выполняются.

Если X0 = 0, программа выполняется нормальным образом шаг за шагом.

Пример использования подпрограмм:



Когда X0 = 1, после выполнения строки 20 программа сразу переходит к строке с указателем P2 и выполняется подпрограмма, после инструкции SRET выполнение подпрограммы завершается и продолжается выполнение основной программы со строки 21.

Рисунок 28 – Реализация инструкции CALL.

Указатели (I) используются вместе с инструкциями EI, DI, IRET для прерывания выполнения основной программы и бывают следующих типов:

1. Коммуникационные прерывания: когда блок принимает адресованный или широковещательный кадр по Modbus протоколу, немедленно (независимо от цикла сканирования) происходит переход к выполнению подпрограммы обработки прерывания с указателем I0. Возврат в главную программу происходит после того, как будет выполнена инструкция IRET.
2. Прерывания по времени: подпрограмма обработки прерывания будет выполнятся автоматически через заданные промежутки времени от 10 до 1000 мс с шагом 10мс. Всего возможно организовать 4 временных прерывания, например, прерывания с указателями I10, I50, I80, I100 будут выполняться раз в 100мс, 500мс, 800мс и 1с соответственно. Возврат в главную программу происходит после того, как будет выполнена инструкция IRET.
3. Внешние прерывания: когда сигнал на входе IN0…IN7 переключается с 0 на 1 или с 1 на 0, немедленно происходит переход к выполнению подпрограммы обработки прерывания с соответствующим указателем I (IN0 → I1000, IN1 → I1001 и т.д.). Возврат в главную программу происходит после того, как будет выполнена инструкция IRET.
4. Прерывания от драйвера шагового двигателя: когда возникает ошибка или авария при управлении шаговым двигателем, отображаемая регистром D381 (ERROR_CODE, подробнее в п. 8), происходит переход к выполнению подпрограммы с указателем I2000. При изменении статуса драйвера шагового двигателя, регистра D371 (MOTOR_STATUS, подробнее в п. 8), происходит переход к подпрограмме с указателем I2001. Возврат в главную программу происходит после того, как будет выполнена инструкция IRET.

5. Коды ошибок

Если горит светодиод "ERR" после загрузки и запуска программы в контроллере, это означает, что в программе есть ошибка: грамматическая или неправильный операнд. Каждая возникающая ошибка фиксируется в специальном регистре (записывается номер шага и код ошибки). Эта информация может быть считана с помощью ПК или ПЛК. В таблице приведен список с описаниями кодов ошибок.

Адрес	Тип	Размер	Описание
0xE004	Input Registers	16-бит	Код ошибки при выполнении пользовательской программы.
0xE084	Input Registers	16-бит	Строка обнаружения ошибки в программе пользователя
0xE004	Coils		Флаг возникновения ошибки в процессе выполнения пользовательской программы

Код ошибки	Описание
2012h	Команда не опознана.
1007h	Внутренняя ошибка, не опознанный тип коллизии сигналов.
1005h	Внутренняя ошибка, не опознанный тип сигнала при коллизии по уровню.
1006h	Внутренняя ошибка, не опознанный тип сигнала при коллизии по инверсному уровню.
2002h	Команда LD, стек переполнен.
2001h	Тип главного сигнала не опознан.
2000h	Обработка LD-подобных команд, изменился код инструкции.
1001h	Внутренняя ошибка, неизвестный тип одиночной коллизии.
1000h	Внутренняя ошибка, одиночная коллизия по текущему значению. Тип сигнала операнда не известен.
200Bh	Обработка AND-подобной команды при снятии сигнала с выходного стека. Неизвестный тип коллизии.
1004h	Внутренняя ошибка групповой коллизии. Неизвестный тип коллизии.
1002h	Внутренняя ошибка групповой коллизии. Не определён тип сигнала операнда для коллизии по уровню.
1003h	Внутренняя ошибка групповой коллизии. Не определён тип сигнала операнда для коллизии по инверсному уровню.
200Ch	Обработка AND-подобной команды. Изменился код инструкции.
200Dh	Обработка OR-подобной команды. Изменился код инструкции.
2010h	Отсутствие записей в главном стеке при применении команды ANB.
200Fh	Ошибка применения команды ANB, нет записи в стеке выхода, а в главном стеке только одна запись.
200Eh	Неизвестный сигнал в стеке выхода при команде ANB.
2011h	Отсутствие минимум двух элементов в главном стеке для проведения операции ORB.

2013h	Стек ветвлений переполнен, команда MPS.
2016h	Стек ветвлений пуст, команды MRD, MPP.
2015h	Главный стек переполнен, команды MRD, MPP.
2014h	Тип сигнала не опознан при назначении селектора. Для команд MRD, MPP.
2017h	Переполнен стек для команды NEXT.
3012h	Предварительное сканирование. Индекс P вне диапазона.
3014h	Предварительное сканирование. Индекс I вне диапазона.
3013h	Предварительное сканирование. Невозможно создать новое временное прерывание, превышен лимит по количеству.
201Dh	Неприемлемый тип операнда для инструкции CJ/CJP.
201Ch	Вне диапазона операнд команды CJ/CJP.
2024h	Неприемлемый тип операнда для инструкции CALL/CALLP.
2023h	Вне диапазона операнд команды CALL/CALLP.
2025h	Отсутствуют точки возврата в стеке при приходе от команды SRET.
2004h	Во время обработки прерывания поступила команда END/FEND.
202Ah	В основной программе поступила команда IRET.
2056h	Команда END, стек главный не пуст.
2057h	Команда END, стек ветвлений не пуст.
2058h	Команда END, стек циклов не пуст.
2059h	Команда END, стек вызовов подпрограмм не пуст.
2026h	Команда IRET, стек главный не пуст.
2027h	Команда IRET, стек ветвлений не пуст.
2028h	Команда IRET, стек циклов не пуст.
2029h	Команда IRET, стек вызовов подпрограмм не пуст.
2020h	Команда CALL/CALLP, индексный операнд не опознан.
201Eh	Команда CALL/CALLP, индексный операнд A вне диапазона.
201Fh	Команда CALL/CALLP, индексный операнд B вне диапазона.
201Ah	Команда CJ/CJP, индексный операнд не опознан.
2018h	Команда CJ/CJP, индексный операнд A вне диапазона.
2019h	Команда CJ/CJP, индексный операнд B вне диапазона.
201Bh	Команда CJ/CJP, запрашиваемого указателя на переход не существует.
2022h	Команда CALL/CALLP, запрашиваемого маркера на переход не существует.
2021h	Команда CALL/CALLP, переполнен стек.
2003h	Неприемлемый операнд, команда OUT.
200Ah	Неприемлемый операнд, команда SET/RST.
2005h	Команда SET не применима к операнду C.

2006h	Команда SET не применима к операнду T.
2007h	Команда SET не применима к операнду D.
2008h	Команда SET не применима к операнду A.
2009h	Команда SET не применима к операнду B.
202Dh	Команда INV, неизвестный тип сигнала.
202Bh	Команда TMR, первый аргумент не типовой.
202Ch	Команда CNT, первый аргумент не типовой.
202Eh	Команда INC/DEC, тип операнда не приемлем.
2037h	Команда ADD/SUB/MUL/DIV/WAND/WOR/WXOR, тип третьего операнда не приемлем.
2038h	Команда NEG/ABS, тип операнда не приемлем.
2030h	Команда CMP, тип третьего операнда не приемлем.
2031h	Команда ZCP, тип третьего операнда не приемлем.
202Fh	Команда MOV/BMOV/FMOV, неприемлемый тип операнда места назначения.
2039h	Команда XCH, неприемлемый тип данных первого операнда.
203Ah	Команда XCH, неприемлемый тип данных второго операнда.
203Bh	Команда ROR/ROL, неприемлемый тип данных первого операнда.
2033h	Команда ZRST, неоднотипные операнды.
2032h	Команда ZRST, неприемлемый тип операнда.
2036h	Команда DIV, деление на ноль целого числа.
2046h	Команда DECO, неприемлемый тип второго операнда.
2047h	Команда ENCO, неприемлемый тип второго операнда.
2048h	Команда SUM, неприемлемый тип второго операнда.
2049h	Команда BON, неприемлемый тип второго операнда.
204Bh	Команда SQR, не приемлемый тип второго операнда.
204Ah	Команда SQR, отрицательное значение под корнем.
204Ch	Команда POW, неприемлемый тип третьего операнда.
203Ch	Команда FLT, неприемлемый тип второго операнда.
203Dh	Команда INT, неприемлемый тип второго операнда.
203Eh	Команда PWM, третий операнд не предназначен для выхода ШИМ-сигнала.
203Fh	Команда PWM, неприемлемый тип третьего операнда.
2041h	Команда DECMP, неприемлемый тип первого операнда.
2040h	Команда DECMP, неприемлемый тип второго операнда.
2042h	Команда DECMP, неприемлемый тип третьего операнда.
2045h	Команда DEZCP, неприемлемый тип третьего операнда.
2044h	Команда DEZCP, неприемлемый тип первого операнда.

2043h	Команда DEZCP, неприемлемый тип второго операнда.
2050h	Команда DEADD/DESUB/DEMUL/DEDIV/DEPOW, неприемлемый тип третьего операнда.
204Fh	Команда DEADD/DESUB/DEMUL/DEDIV/DEPOW, неприемлемый тип первого операнда.
204Eh	Команда DEADD/DESUB/DEMUL/DEDIV/DEPOW, неприемлемый тип второго операнда.
204Dh	Команда DEDIV, деление на ноль.
3015h	Ошибка предварительного сканирования, обнаружена неизвестная команда.
2053h	Команда DESQR, не приемлемый тип первого операнда.
2052h	Команда DESQR, не приемлемый тип второго операнда.
2051h	Команда DESQR, отрицательное число под корнем.
2035h	Команда LD#, стек переполнен.
2034h	Команда LD#, тип главного сигнала не опознан.
4000h	Переполнение очереди коммутационных прерываний.
4001h	Переполнение очереди временных прерываний TIM0.
4002h	Переполнение очереди временных прерываний TIM1.
4003h	Переполнение очереди временных прерываний TIM2.
4004h	Переполнение очереди временных прерываний TIM3.
4005h	Переполнение очереди внешних прерываний IN0.
4006h	Переполнение очереди внешних прерываний IN1.
4007h	Переполнение очереди внешних прерываний IN2.
4008h	Переполнение очереди внешних прерываний IN3.
4009h	Переполнение очереди внешних прерываний IN4.
400Ah	Переполнение очереди внешних прерываний IN5.
400Bh	Переполнение очереди внешних прерываний IN6.
400Ch	Переполнение очереди внешних прерываний IN7.
400Dh	Переполнение очереди прерываний ошибок от драйвера.
400Eh	Переполнение очереди прерываний смены статуса двигателя.
2054h	Команда TRD/DRD, неприемлемый тип операнда.
2055h	Команда TWR/DWR, неприемлемый тип операнда.
3000h	Стеки пусты, отсутствие значения сигнала.
3001h	Главный стек пуст, отсутствие значения сигнала.
3003h	Значение индексного регистра вне диапазона.
3004h	Индекс операнда X вне диапазона.
3005h	Индекс операнда Y вне диапазона.
3006h	Индекс операнда M вне диапазона.

3007h	Индекс операнда C вне диапазона.
3008h	Индекс операнда T вне диапазона.
3009h	Индекс операнда A/B вне диапазона.
300Ah	Индекс операнда D вне диапазона.
300Bh	Индекс операнда P вне диапазона.
300Ch	Индекс операнда I вне диапазона.
300Dh	Тип операнда не опознан.
300Fh	С операнда не считать значение.
300Eh	Использование типа FLOAT с 16-битной инструкцией.
3010h	Операция получения значения операнда. Неприемлемый тип операнда.
3011h	Операция получения токена операнда. Неприемлемый тип операнда.
5000h	Короткое замыкание вспомогательного источника питания +5В.
205Ah	Команда TWR, некорректный формат времени.
205Bh	Команда MOD, деление на ноль.
205Ch	Команда DWR, некорректный формат даты.
205Dh	Переполнен стек инструкций контактного типа (LD#*)
205Eh	Переполнен стек инструкций контактного типа (AND#*)
205Fh	Переполнен стек инструкций контактного типа (OR#*)

6. Описание базовых команд

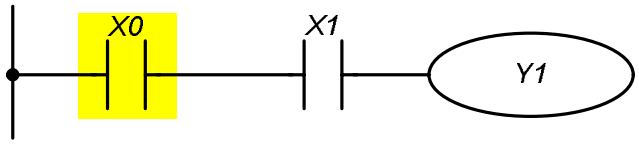
Инструкция	Функция
LD	Нормально-открытый контакт

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда LD используется в качестве нормально-открытого контакта для программирования начала логических цепочек. В контактных схемах команда всегда расположена слева и соединяется непосредственно с шиной питания.

Применение:




```
LD    X0
AND   X1
OUT   Y1
```

Команда "нормально-открытый контакт X0" открывает последовательную логическую связь. Если на входах X1 и X1 одновременно будет сигнал "1", тогда и выход Y1 установится в состояние "1".

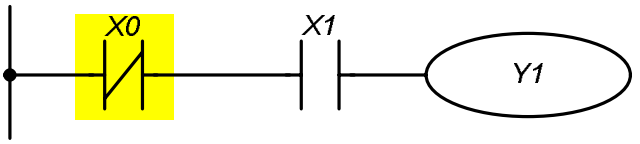
Инструкция	Функция
LDI	Нормально-закрытый контакт

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда LDI используется в качестве нормально-закрытого контакта для программирования начала логических цепочек. В контактных схемах команда всегда расположена слева и соединяется непосредственно с шиной питания.

Применение:




```
LDI   X0
AND   X1
OUT   Y1
```

Команда "нормально-закрытый контакт X0" открывает последовательную логическую связь. Если на входе X0 будет "0", а на X1 будет сигнал "1", тогда выход Y1 установится в состояние "1".

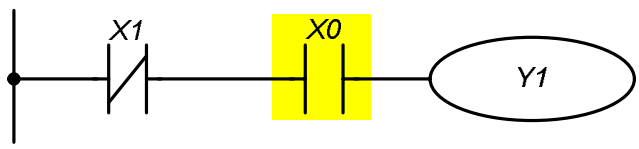
Инструкция	Функция
AND	Последовательный нормально-открытый контакт (логическое И)

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда AND используется в качестве последовательного нормально-открытого контакта для программирования операции логического умножения (И). Команда представляет логическую операцию и поэтому не может программироваться в начале цепи. В начале логического выражения программируются инструкции LD или LDI.

Применение:




```
LDI    X1
AND    X0
OU     Y1
```

Команда "последовательный нормально-открытый контакт X0" создает последовательную логическую связь с контактом X1 и служит для выполнения операции логического умножения. Если на входе X1 будет "0" и на X0 будет сигнал "1", тогда выход Y1 установится в состояние "1".

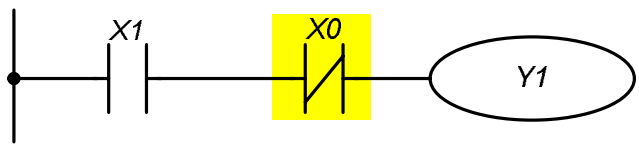
Инструкция	Функция
ANI	Последовательный нормально-закрытый контакт (логическое И-НЕ)

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда ANI используется в качестве последовательного нормально-закрытого контакта для программирования операции И-НЕ. Команда представляет логическую операцию и поэтому не может программироваться в начале цепи. В начале логического выражения программируются инструкции LD или LDI.

Применение:




```
LD     X1
ANI    X0
OUT    Y1
```

Команда "последовательный нормально-закрытый контакт X0" создает последовательную логическую связь с контактом X1 и служит для выполнения логической операции И-НЕ. Если на входе X1 будет "1" и на X0 не будет сигнала "1", тогда выход Y1 установится в состояние "1".

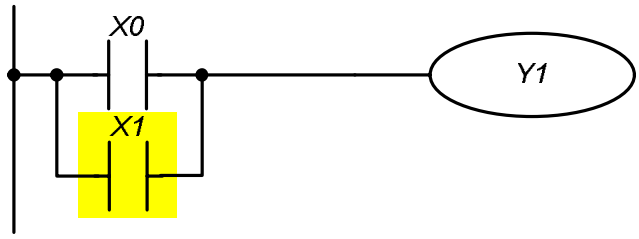
Инструкция	Функция
OR	Параллельный нормально-открытый контакт (логическое ИЛИ)

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда OR используется в качестве параллельного нормально-открытого контакта для программирования операции логического сложения (ИЛИ). Команда представляет логическую операцию и поэтому не может программироваться в начале цепи. В начале логического выражения программируются инструкции LD или LDI.

Применение:




```
LD    X0
OR    X1
OUT   Y1
```

Команда "параллельный нормально-открытый контакт X1" создает параллельную логическую связь с контактом X0 и служит для выполнения операции логического сложения. Если хотя бы на одном из входов X0 или X1 будет "1", тогда и на выходе Y1 будет состояние "1".

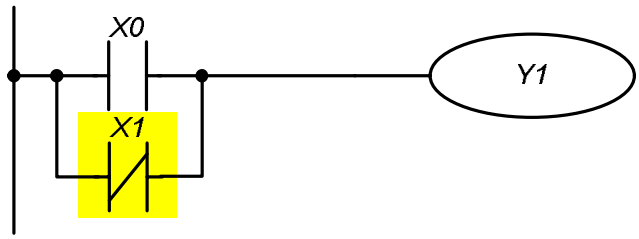
Инструкция	Функция
ORI	Параллельный нормально-закрытый контакт (логическое ИЛИ-НЕ)

Операнд	X	Y	M	T	C	A	B	D
	•	•	•	•	•			

Описание:

Команда ORI используется в качестве параллельного нормально-закрытого контакта для программирования логической операции ИЛИ-НЕ. Команда представляет логическую операцию и поэтому не может программироваться в начале цепи. В начале логического выражения программируются инструкции LD или LDI.

Применение:




```
LD    X0
ORI   X1
OUT   Y1
```

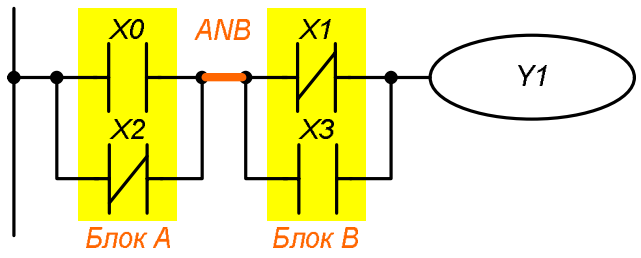
Команда "параллельный нормально-закрытый контакт X1" создает параллельную логическую связь с контактом X0 и служит для выполнения логической операции ИЛИ- НЕ. Если на входе X0 будет "1" или на входе X1 будет "0" (одно или оба условия одновременно), тогда на выходе Y1 будет состояние "1".

Инструкция	Функция
ANB	«И» блок: последовательное включение параллельных блоков

Описание:

- Команда ANB используется для последовательного соединения цепочек из двух параллельных контактов. Отдельные блоки, параллельно включенных элементов, заносятся в программу раздельно. Чтобы эти блоки соединить последовательно, после каждого блока программируется ANB инструкция.
- Начало разветвления программируется с помощью инструкций LD или LDI.
- ANB-инструкция является независимой и не требует ввода никаких операндов.
- ANB-инструкция внутри всей программы может программироваться многократно.
- В контактной схеме ANB-инструкция изображается как последовательное соединение. ANB-инструкция, имеющаяся в списке инструкций языка IL, при конвертировании в контактную схему появляется автоматически и изображается как перемычка.
- Если программируется несколько отдельных блоков, непосредственно один за другим, то нужно ограничить число LD- и LDI-инструкций, а также количество ANB-инструкций до 8.

Применение:




```
LD    X0
ORI   X2
LDI   X1
OR    X3
ANB
OUT   Y1
```

Команда ANB создает последовательную логическую связь между двумя логическими блоками (блоком А и блоком В).

Инструкция	Функция
ORB	«ИЛИ» блок: параллельное включение последовательных блоков

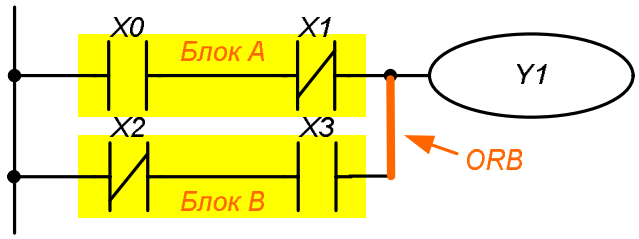
Описание:

- Команда ORB используется для параллельного соединения цепочек из двух последовательных контактов. Если несколько последовательных блоков включаются параллельно, необходимо после программирования каждого отдельного блока вводить ORB-инструкцию.
- Начало разветвления программируется с помощью инструкций LD или LDI.
- ORB-инструкция является независимой и не требует ввода никаких операндов.
- ORB-инструкция внутри всей программы может программироваться многократно.
- В контактной схеме ORB-инструкция изображается как параллельное соединение. ORB-инструкция, имеющаяся в списке инструкций языка IL, при конвертирова-

нии в контактную схему появляется автоматически и изображается как перемычка.

— Если программируется несколько отдельных блоков непосредственно один за другим, то нужно ограничить число LD и LDI инструкций и, соответственно, также число ORB-инструкций до 8.

Применение:




```
LD    X0
ANI   X1
LDI   X2
AND   X3
OR
OUT   Y1
```

Команда ORB создает параллельную логическую связь между двумя логическими блоками (блоком A и блоком B).

Инструкция	Функция
MPS	Смещение вниз по стеку

Инструкция	Функция
MRD	Считать значение из стека

Инструкция	Функция
MPP	Выход из стека

Описание:

— Инструкции MPS, MRD, MPP служат для того, чтобы создавать уровни логических связей (например, после одного начального логического выражения создать несколько логических выражений на выходе, т.е. включать несколько выходов-катушек).

— С помощью инструкции MPS запоминается предыдущий результат логических связей (обработки логического выражения).

— С помощью инструкции MRD возможно прочтение нескольких частных разветвлений между началом (MPS) и концом (MPP) разветвления, учитывающих на каждом разветвлении результат обработки логического выражения до MPS.

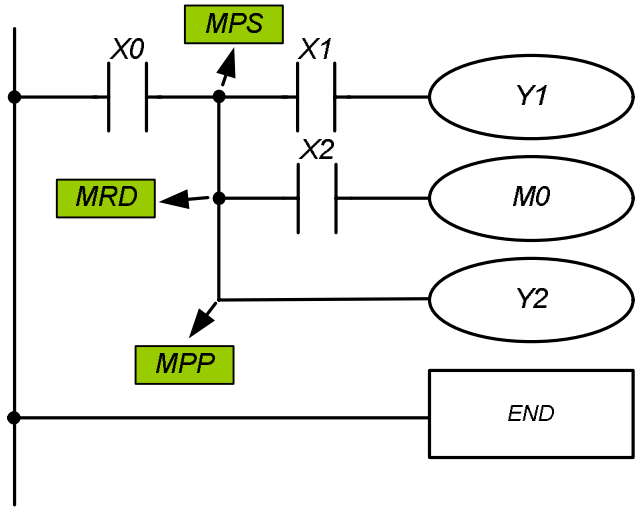
— Последнее частное разветвление создается MPP инструкцией.

— Открывшееся с помощью MPS инструкции разветвление всегда должно быть закрыто MPP инструкцией.

— Все три инструкции не требуют никаких операндов.

— В контактной схеме эти инструкции не изображаются. Если программирование выполняется в контактной схеме, разветвления используются как обычно. MPS-, MRD- и MPP-инструкции на языке списка инструкций (IL) появляются автоматически, после того как программа конвертируется в контактную схему.

Применение:




```
LD    X0
MPS
AND   X1
OUT   Y1
MPD
AND   X2
OUT   M0
MPP
OUT   Y2
END
```

MPS

Промежуточный результат (здесь X0) на 1-ом уровне логических связей занесен на 1-ое место в стековую память промежуточных связей. Выполняется логическое умножение X1 с X0 и устанавливается выход Y1.

MRD

Перед выполнением следующей инструкции опрашивается промежуточный результат на 1-ом месте памяти логических связей. Выполняется логическое умножение X2 с X0 и устанавливается выход M0.

MPP

Перед выполнением следующей инструкции опрашивается промежуточный результат на 1-ом месте памяти логических связей. Устанавливается выход M0. Операция на 1-ом уровне промежуточных результатов завершена, и память логических связей стирается.

Инструкция	**Функция**
OUT	Выход

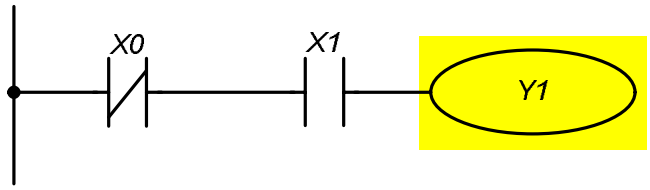
Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
		•	•					

Описание:

— Команда OUT служит для присвоения состояния сигнала (включения или отключения выхода) в зависимости от результата логических связей (результата обработки центральным процессором логического выражения).
— С помощью инструкции OUT можно завершить программирование строки (логического выражения).
— Программирование нескольких инструкций OUT как результат обработки логического выражения также возможно.

— Результат логических связей, представленный посредством инструкции OUT, может применяться в следующих шагах программы как состояние входного сигнала, т.е. может многократно опрашиваться во многих логических выражениях.
— Результат логических связей, представленный OUT инструкцией, активен (включен) до тех пор, пока действуют условия его включения.
— При программировании двойной записи одинаковых выходов (их адресов) могут возникнуть проблемы при отработке программы. Избегайте двойной записи выходов, так как это может привести к помехам при отработке программы.

Применение:




```
LDI    X0
AND    X1
OUT    Y1
```

При условии: X0 = 0 и X1 = 1 - команда OUT Y1 установит выход контроллера Y1 в состояние "1".

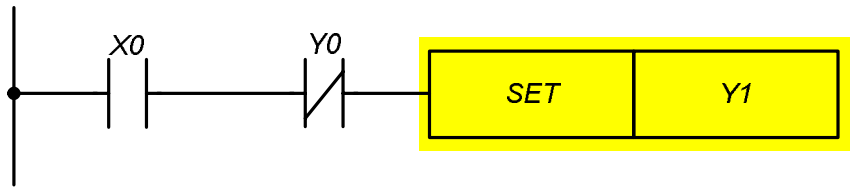
Инструкция	**Функция**
SET	Включение выхода с фиксацией

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
		•	•					

Описание:

— Состояние сигнала операнда с помощью SET инструкции (включение) может устанавливаться непосредственно.
— С помощью SET могут устанавливаться в "1" (включаться) операнды Y, M.
— Как только входное условие установится для SET инструкции (сигнал "1"), включится соответствующий операнд.
— Если входные условия для SET инструкции больше не будут выполняться, соответствующий операнд останется включенным.

Применение:




```
LD     X0
ANI    Y0
SET    Y1
```

Выход Y1 включится при выполнении условий X0, Y0 и больше от этих условий зависеть не будет. Выключить выход Y1 можно будет только командой RST Y1 или снятием питания с контроллера.

Инструкция	Функция
RST	Сброс состояния операнда

Операнд	X	Y	M	T	C	A	B	D
		●	●	●	●	●	●	●

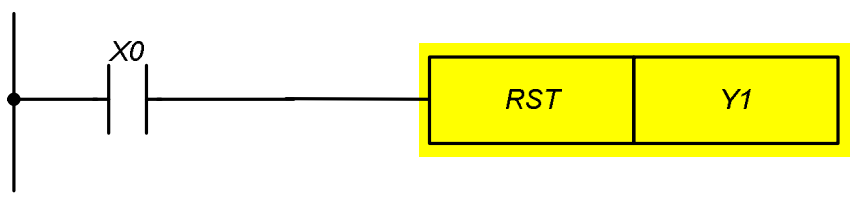
Описание:

Состояние сигнала операнда с помощью RST инструкции (сброс) может устанавливаться непосредственно.

RST-инструкция отключает соответствующие операнды. Это означает:

— Выходы Y, контакты M выключаются (состояние сигнала "0").
— Действительное значение таймера и счетчика, а также содержание регистров D, A и B сбрасываются на "0".
— Как только входное условие установится для RST инструкции (сигнал "1"), выключится соответствующий операнд.
— Если входные условия для RST инструкции больше не будут выполняться, соответствующий операнд останется выключенным.

Применение:




```
LD    X0
RST   Y1
```

Выход Y1 выключится при выполнении условия X1 и останется выключенным даже когда условие X0 выполняться не будет.

Инструкция	Функция
TMR	Таймер (16-бит)

Операнды	K	H	F	X	Y	M	T	C	A	B	D
S1							●				
S2	●	●					●	●	●	●	●

Описание:

— Команда TMR служит для присвоения состояния сигнала (включения или отключения выхода) в зависимости от результата логических связей через заданный в инструкции промежуток времени.
— С помощью инструкции TMR можно завершить программирование строки (логического выражения).
— Результат логических связей, представленный посредством инструкции TMR, может применяться в следующих шагах программы как состояние входного сигнала, т.е. может многократно опрашиваться во многих логических выражениях.

— Результат логических связей, представленный TMR инструкцией, активен (включен) до тех пор, пока действуют условия его включения.

Применение:

```
LD      X0
TMR     T5      K1000
```

При условии X0 = 1 инструкция TMR T5 будет вести отчет времени, пока значение в регистре T5 не достигнет значения K1000 (100 сек). При X0 = 0 выполнение инструкции TMR прекратится и T5 сбросится на "0".

Инструкция		Функция
CNT		Счётчик (16-бит)
DCNT		Счётчик (32-бит)

Операнды	K	H	F	X	Y	M	T	C	A	B	D
								•			
	•	•					•	•	•	•	•

Информация

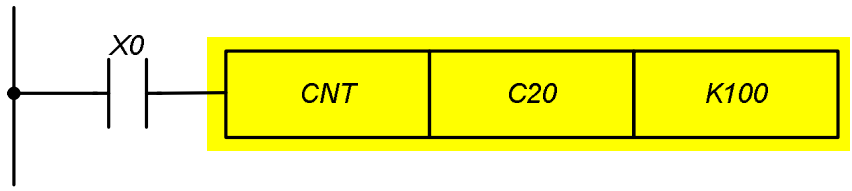
(D) Обычно для использования 32-х битных команд добавляется префикс "**D**" к названию инструкции. (D) – только 32-х битный вариант инструкции.

(P) Для импульсных инструкций со «временем жизни» в один скан добавляется постфикс "**P**". При этом понятие *один скан* следует отнести к используемому операнду. Например, операнд **M0** в строке 7 главной программы был установлен из "0" в "1". Теперь для всех ниже инструкций, находящихся до FEND или END операнд **M0** будет иметь импульсную составляющую по переднему фронту (результатом для LDP M0 будет "1"), а также для инструкций, начиная с 0-ой строки до 6-ой следующего сканирования, у операнда **M0** будет импульсная составляющая. При переходе на строку 7 (или ниже при использовании команды CJ) **M0** будет иметь высокий уровень сигнала без импульсных составляющих. Таким образом, мы совершили один круг по телу программы – один скан, смещённый до строки изменения операнда. В случае возникновения прерываний до перехода к строке 7, операнд **M0** сохраняет импульсную составляющую до перехода к основной программе. Если операнд **M0** был модифицирован в прерывании или подпрограмме, то местом изменения операнда считается строка, с которой был осуществлён переход в подпрограмму или строка основной программы, до обработки которой был вызван обработчик прерывания.

Описание:

- Команда CNT служит для суммирования количества замыканий входного контакта и присвоения состояния сигнала (включения выхода), когда текущее значение счетчика достигнет заданного значения.
- С помощью инструкции CNT можно завершить программирование строки (логического выражения).
- Результат логических связей, представленный посредством инструкции CNT, может применяться в следующих шагах программы как состояние входного сигнала, т.е. может многократно опрашиваться во многих логических выражениях.
- Для сброса текущего значения счетчика можно использовать команду RST.
- **Внимание:** аппаратные счётчики считают выше порога и работают независимо от наличия входного сигнала.

Применение:




```
LD      X0
CNT     C20     K100
```

При изменении состояния X0 с "0" на "1" значение регистра C20 будет увеличено на 1, и так пока значение в регистре C20 не достигнет значения K100 (100 импульсов). После этого счет прекратится и C20 включится. Для сброса значения регистра C20 можно использовать команду RST C20.

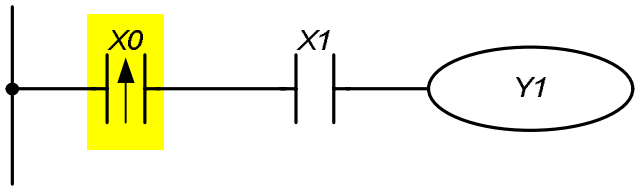
Инструкция	**Функция**
LDP	Начало логического выражения с опросом по переднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	●	●	●	●	●			

Описание:

- Команда LDP используется для программирования импульсного начала логической связи.
- Инструкция LDP должна программироваться в начале цепи.
- LDP-инструкция используется также вместе с инструкциями ANB и ORB для запуска разветвлений.
- LDP-инструкция после положительного фронта сохраняется на время цикла программы (скана).

Применение:




```
LDP   X0
AND   X1
OUT   Y1
```

Команда "LDP X0" открывает последовательную логическую связь. Если вход X0 изменит свое состояние с "0" на "1" (при этом X1 = 1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

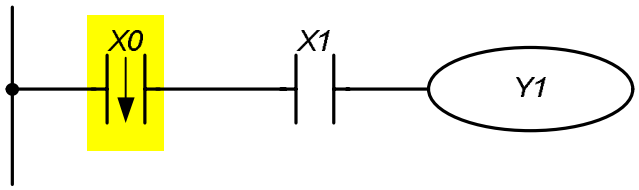
Инструкция	**Функция**
LDF	Начало логического выражения с опросом по заднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	●	●	●	●	●			

Описание:

— Команда LDF используется для программирования импульсного начала логической связи.
— Инструкция LDF должна программироваться в начале цепи.
— LDF-инструкция используется также вместе с инструкциями ANB и ORB для запуска разветвлений.
— LDF-инструкция после отрицательного фронта сохраняется на время цикла программы (скана).

Применение:




```
LDF   X0
AND   X1
OUT   Y1
```

3

Команда "LDF X0" открывает последовательную логическую связь. Если вход X0 изменит свое состояние с "1" на "0" (при этом X1 = 1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

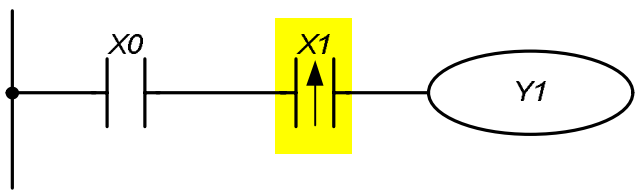
Инструкция	**Функция**
ANDP	«И» с опросом по переднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	●	●	●	●	●			

Описание:

— Команда ANDP используется для программирования последовательного соединения импульсного контакта с опросом по переднему фронту.

Применение:




```
LD    X0
ANDP  X1
OUT   Y1
```

Команда "ANDP X1" создает последовательную логическую связь. Если вход X1 изменит свое состояние с "0" на "1" (при этом X0 = 1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

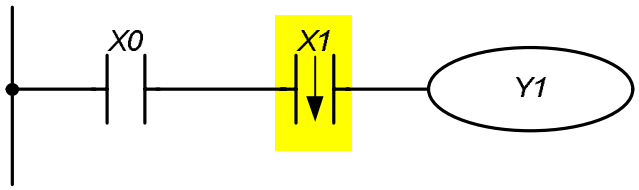
Инструкция	**Функция**
ANDF	«И» с опросом по заднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	•	•	•	•	•			

Описание:

— Команда ANDF используется для программирования последовательного соединения импульсного контакта с опросом по заднему фронту.

Применение:




```
LD    X0
ANDF  X1
OUT   Y1
```

Команда "ANDF X1" создает последовательную логическую связь. Если вход X1 изменит свое состояние с "1" на "0" (при этом X0 = 1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

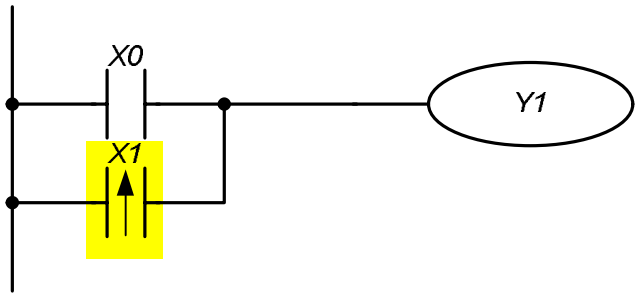
Инструкция	**Функция**
ORP	«ИЛИ» с опросом по переднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	•	•	•	•	•			

Описание:

— Команда ORP используется для программирования параллельного соединения импульсного контакта с опросом по переднему фронту.

Применение:




```
LD    X0
ORP   X1
OUT   Y1
```

Команда "ORP X1" создает параллельную логическую связь. Выход Y1 будет в состоянии "1" в течении одного скана если вход X1 изменит свое состояние с "0" на "1" или X0 = 1.

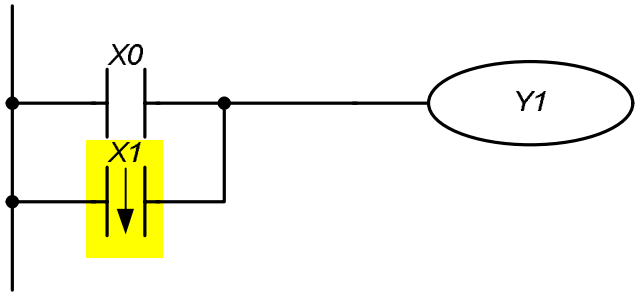
Инструкция	**Функция**
ORF	«ИЛИ» с опросом по заднему фронту (импульс)

Операнд	**X**	**Y**	**M**	**T**	**C**	**A**	**B**	**D**
	•	•	•	•	•			

Описание:

— Команда ORF используется для программирования параллельного соединения импульсного контакта с опросом по заднему фронту.

Применение:




```
LD    X0
ORF   X1
OUT   Y1
```

Команда "ORF X1" создает параллельную логическую связь. Выход Y1 будет в состоянии "1" в течении одного скана если вход X1 изменит свое состояние с "1" на "0" или X0 = 1.

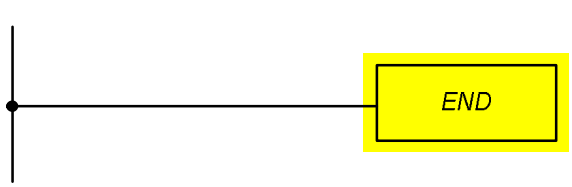
Инструкция	**Функция**
END	Конец программы

Описание:

Окончание программы контроллера и переход к началу программы (шаг 0).

— Каждая программа контроллера должна завершаться инструкцией END.
— Если программируется END-инструкция, то на этом месте оканчивается обработка программы. Последующие области программы не принимаются больше во внимание. После отработки END-инструкции выполняется установка выходов и переход к началу программы (шаг 0).

Применение:




```
END
```

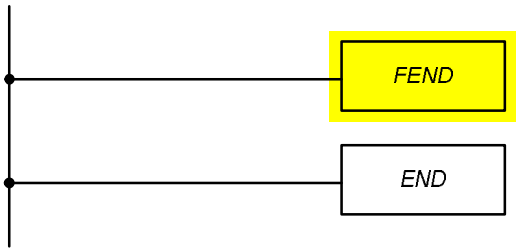
Инструкция	Функция
FEND	Конец основной программы

Описание:

Окончание основной программы контроллера и переход к началу программы (шаг 0). Основные отличия от команды END:

— Обработка не заканчивается на команде FEND. Инструкция FEND отделяет основную программу от подпрограмм и обработчиков прерываний, которые располагаются в области между инструкциями FEND и END и обрамляются **P** и **SRET**, **I** и **IRET** соответственно.
— Если в программе не используются подпрограммы и обработчики прерываний, то использование команды FEND не является обязательным.
— Допускается только однократное использование инструкции.

Применение:




```
FEND
END
```

Инструкция	Функция
NOP	Пустая строка в программе

Описание:

Можно создать пустую строку без логических функций, которая позднее может быть использована для каких-либо инструкций, например, при окончательном изготовлении программы, при отладке оборудования.

— После успешного завершения программы NOP-команды должны быть удалены, так как в противном случае они бесполезно удлиняют время цикла программы.
— Количество NOP-команд не ограничено.

Применение:

```
LD      X0
NOP
OUT     Y0
```

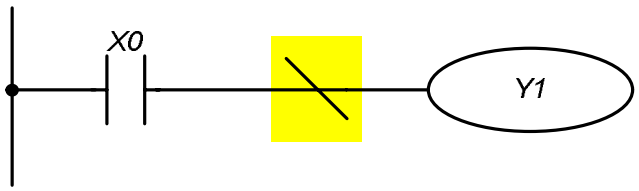
В контактных схемах инструкция NOP не отображается.

Инструкция	Функция
INV	Инверсия – замена результата логических связей на противоположный

Описание:

- INV-инструкция инвертирует состояние сигнала результата стоящей впереди инструкции.
- Полученная согласно обработки "1", после инверсии становится "0".
- Полученный согласно обработки "0", после инверсии становится "1".
- INV-инструкция может применяться, как AND и ANI инструкции.
- INV-инструкция может применяться для реверсирования сигнала результата комплексной схемы.
- INV-инструкция может применяться для реверса сигнала результата импульсных инструкций LDP, LDF, ANP и т.д.

Применение:




```
LD    X0
INV
OUT   Y1
```

Если X0 = 0, выход Y1 = 1. Если X0 = 1, выход Y1 = 0.

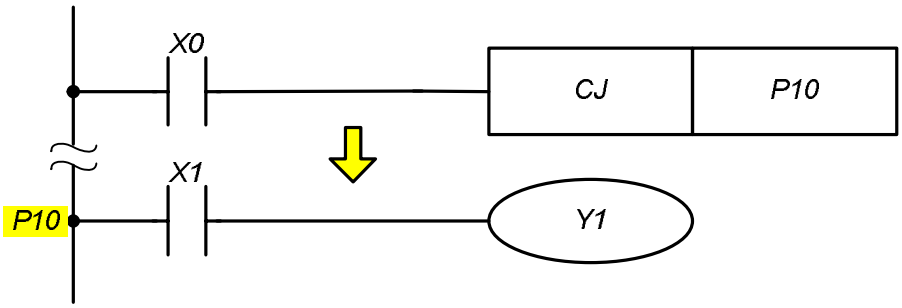
Инструкция	Функция
P	Адресация точки перехода по программе или к подпрограмме

Операнд	0…31

Описание:

- **P**-инструкция служит для указания точки перехода для команд CJ, CALL.
- Номер точки в программе не должен повторяться.

Применение:




```
LD    X0
CJ    P10
.
.
P     10
LD    X1
OUT   Y1
```

Точка P10 указывает адрес перехода программы при выполнении инструкции CJ P10.

Инструкция	Функция
I	Адресация точки прерывания

Операнд	0…100, 1000…1007, 2000, 2001

Описание:

I-инструкция служит для указания точки перехода к подпрограмме обработки прерывания. Глобально прерывания включаются командой **EN**, а выключаются – **DS**.

Всего в контроллере может быть 15 прерываний:

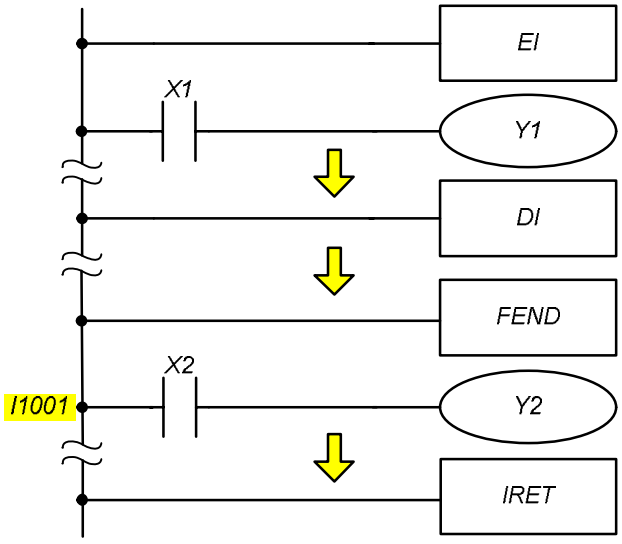
Прерывание, которое возникает при приёме Modbus кадра, широковещательного или адресованного контроллеру, через RS-485 маркируется как **I0**.

В контроллере можно реализовать четыре временных прерывания. **I***n*, где *n* – период вызова обработчика прерываний в 10мс и может иметь значение от 1 до 100. Таким образом $n = \frac{T}{10}$, мс, где T – желаемый период вызова обработчика в мс.

Восемь точек внешних прерываний **I1000**…**I1007**, каждому из которых соответствует дискретный вход с 0 по 7 соответственно. Прерывание возникает при изменении уровня входного сигнала.

Две точки прерываний от драйвера шагового двигателя: **I2000** – при возникновении ошибки и **I2001** – изменение статуса двигателя (подробнее см. п. 8).

Применение:




```
EI              Разрешение прерывания
LD    X1        Норм. откр. контакт X1
OUT   Y1        Выход Y1
…
DI              Запрещение прерывания
…
FEND            Конец основной программы
I     1001      Точка входа в обработчик прерывания
LD    X2        Норм. откр. контакт X2
OUT   Y2        Выход Y2

IRET            Конец обработчика прерываний
```

7. Описание прикладных команд

Инструкция	Операнды	Варианты исполнения	Функция
CJ	S	P	Условный переход

S	В качестве операнда могут использоваться указатели **P**, которые могут индексироваться (A, B)

Описание:

С помощью CJ-инструкции может пропускаться часть программы. При применении этой инструкции время выполнения программы может уменьшаться, например, пропуск участка программы, отведённого под инициализацию периферии контроллера, включение прерываний и т.д. (подробнее см. п. 4.8).

CALL	S	P	Вызов подпрограммы

S	В качестве операнда могут использоваться указатели **P**, которые могут индексироваться (A, B)

Описание:

С помощью CALL-инструкции вызывается подпрограмма.

- Подпрограмма маркируется с помощью точек P и вызывается CALL- инструкцией.
- В конце подпрограммы должна находиться SRET-инструкция.
- Подпрограмма программируется после FEND-инструкции и перед END- инструкцией.
- Если активируется CALL-инструкция, то выполняется переход к указанной точке маркировки. После обработки SRET-инструкции выполняется обратный переход в основную программу к инструкции, следующей за вызовом CALL.
- Так же точки могут использоваться с любым числом CALL-инструкций
- Внутри подпрограммы могут вызываться другие подпрограммы. Возможно максимум 8 уровней вложенности.

SRET			Конец подпрограммы

Описание:

С помощью SRET-инструкции определяется конец подпрограммы (подробнее см. п. 4.8).

- В конце подпрограммы должна стоять SRET-инструкция.
- После обработки SRET-инструкции выполняется обратный переход в основную программу к инструкции, следующей за вызовом CALL.
- SRET-инструкция может программироваться только вместе с CALL-инструкцией.

Примечание: для выполнения инструкции контакт не требуется.

IRET			Конец подпрограммы обработки прерывания

Описание:
IRET-инструкция завершает процесс обработки прерывания (подробнее см. п. 4.8).

Примечание: для выполнения инструкции контакт не требуется.

EI			Глобальное разрешение обработки прерываний

Описание:

EI-инструкция разрешает обработку прерываний (подробнее см. п. 4.8).

Примечание: для выполнения инструкции контакт не требуется.

DI			Глобальное запрещение обработки прерываний

Описание:
DI-инструкция запрещает обработку прерываний (подробнее см. п. 4.8).

Примечание: для выполнения инструкции контакт не требуется.

Вызов подпрограммы обработки прерывания

- При обработке прерывания выполняется переход от основной программы к подпрограмме прерывания.
- После окончания обработки подпрограммы прерывания выполняется возврат к основной программе контроллера.
- Начало программы прерывания определяется установкой маркировки (точки прерывания).
- Конец программы прерывания определяется IRET-инструкцией.
- Программа прерывания должна программироваться в конце программы контроллера после FEND-инструкцией и перед END-инструкцией.

Указание: Если ни одна из двух инструкций EI или DI не программируется, режим прерывания не активизируется, т.е. не будет обработан ни один сигнал прерывания.

Отработка программы прерывания

Несколько, следующих одна за другой, программ прерывания обрабатываются в последовательности их вызова.

Если одновременно вызываются несколько программ прерывания, то вначале обрабатывается программа прерывания с более низким адресом точки.

FOR	S		Начало цикла FOR-NEXT

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•

Примечание: для выполнения инструкции контакт не требуется.

NEXT			Конец цикла FOR-NEXT

Примечание: для выполнения инструкции контакт не требуется.

Циклы

Инструкции FOR/NEXT используются для программирования циклических повторений частей программы (петля программы).

Описание:

- Часть программы между FOR- и NEXT-инструкциями повторяется "n" раз. После завершения FOR-инструкции выполняется переход к шагу программы после NEXT-инструкции.
- Значение "n" может находиться внутри следующей области: "n": от +1 до +32 767. Если для "n" указано значение между 0 и -32 768, то петля FOR-NEXT отрабатывается только один раз.
- Можно программировать до восьми FOR-NEXT-уровней вложенности.
- FOR- и NEXT-инструкции могут программироваться только попарно. К каждой инструкции FOR должна программироваться соответственно NEXT-инструкция.

Источники ошибок

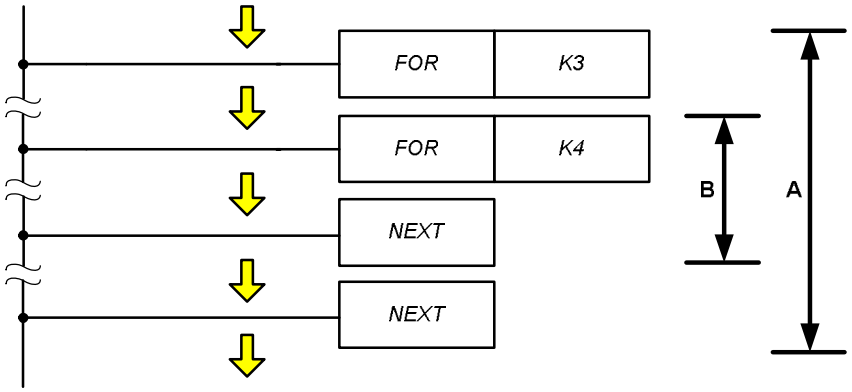
- В следующих случаях появляются ошибки в работе программы:
- NEXT-инструкция запрограммирована перед FOR-инструкцией.
- Количество NEXT-инструкций не соответствует количеству FOR-инструкций.
- Большое число повторений "n" может значительно увеличить время выполнения программы.

Пример программирования при использовании FOR- и NEXT-инструкций.

В примере запрограммированы два входящие друг в друга FOR-NEXT-цикла.

Отрезок программы A обрабатывается три раза (здесь K3 константа 3).

При каждом исполнении отрезка A отрезок программы B обрабатывается четыре раза (здесь K4 константа 4).



CMP	S1 S2 D	D P	Сравнение числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•	•					

Примечание: операнд D занимает три адреса.

Описание:

Сравнение двух числовых значений данных (больше, меньше, равно).

— Данные в обоих источниках (S1) и (S2) сравниваются друг с другом.

— Результат сравнения (больше, меньше, равно) отображается (индицируется) благодаря задействованию реле M, или выхода Y. Определение, какой из этих операндов должен задействоваться, выполняется по адресу результата (D).

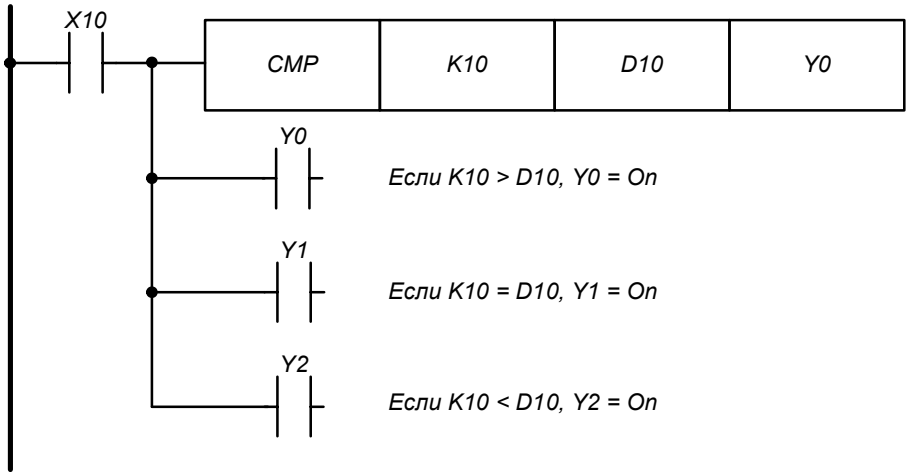
(S1) > (S2) → (D)

(S1) = (S2) → (D+1)

(S1) < (S2) → (D+2)

— Данные в S1 и S2 обрабатываются как знаковые целочисленные данные.

Пример:



Y0: включен, если K10 > значения регистра D10, Y1 и Y2 выключены.

Y1: включен, если K10 = значению регистра D10, Y0 и Y2 выключены.

Y2: включен, если K10 < значения регистра D10, Y0 и Y1 выключены.

Y0, Y1, Y2 не изменяются, если входное условие X10 выключено. Для сброса результатов сравнения используйте команды RST, ZRST.

ZCP	S1 S2 S D	D P	Зонное сравнение числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●					●	●	●	●	●
S2	●	●					●	●	●	●	●
S	●	●					●	●	●	●	●
D					●	●					

Примечание:

— Операнд D занимает три адреса.

— Операнд S1 должен быть меньше чем S2.

Описание:

— Сравнение числовых значений данных с числовыми областями данных (больше, меньше, равно)

— Данные в источнике (S) сравниваются с данными обоих источниках (S1) и (S2).

— Результат сравнения (больше, меньше, равно) отображается (индицируется) благодаря задействованию реле M или выхода Y. Определение, какой из этих операндов должен задействоваться, выполняется в регистре данных (по адресу назначения) – (D).

(S) < (S1) → (D)

(S1) ≤ (S) ≤ (S2) → (D + 1)

(S) > (S2) → (D + 2)

— Если (S1) больше (S2), то все операнды (D) будут сброшены.

Для сброса результатов сравнения используйте команды RST, ZRST.

MOV	S D	D P	Передача данных

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●	●	●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●

Описание:

- Инструкция служит для передачи данных от источника данных (S) к данным цели (D). Содержимое источника (S) при этом не изменяется.
- Данные в источнике данных (S) при выполнении MOV-инструкции автоматически интерпретируются как двоичные значения.
- Битовые операнды будут занимать соответствующее типу инструкции количество адресов, 16 либо 32 адреса. При этом можно комбинировать типы операндов для

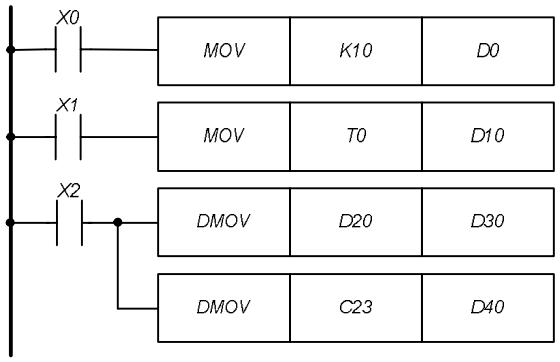
источника и цели. Например, в результате выполнения команды MOV D3 M0 реле M0…M15 будет представлено значение регистра D3 в бинарном виде.

Пример:

Если входное условие X0 включено, то значение регистра D0 будет равно 10. Если X0 выключен, значение D0 не изменится.

Если входное условие X1 включено, то регистру D10 будет передаваться текущее значение таймера T0. Если X1 выключен, значение D10 не изменится.

Если входное условие X2 включено, то регистрам (D30, D31) будет передаваться значение регистров (D20, D21) и регистрам (D40, D41) будет передаваться текущее значение счетчика C23.



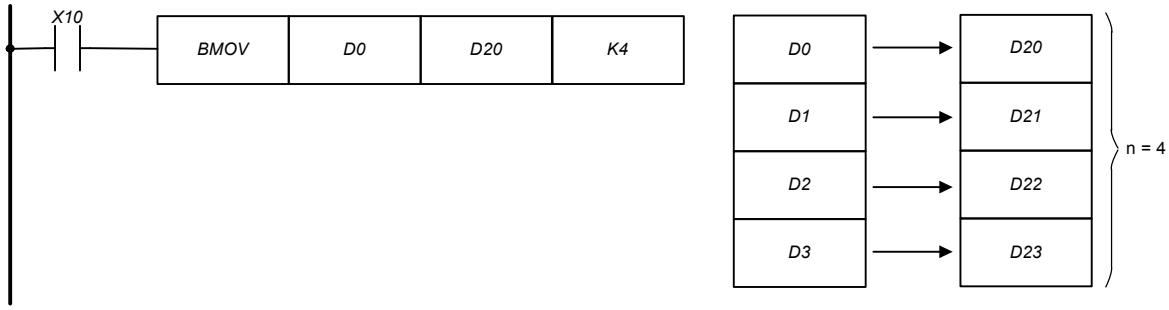
BMOV	S D n	D P	Передача блока данных

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●	●	●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Описание:

Копирование пакета данных. Сдвиг при операции осуществляется как по операнду источника (S), так и по операнду назначения (D) на (n) элементов блока в зависимости от разрядности инструкции.

Пример:



Когда X10 включено, содержимое регистров D0 – D3 будет копироваться в регистры D20 – D23.

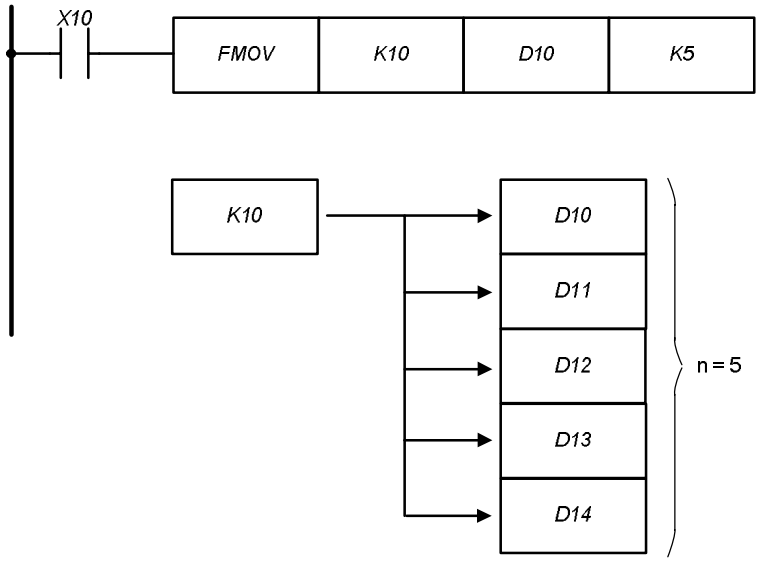
FMOV	S D n	D P	Передача данных в несколько адресов

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•	•	•	•	•	•	•	•	•
D					•	•	•	•	•	•	•
n	•	•					•	•	•	•	•

Описание:

Данные операнда источника (S) копируются в (n) однотипных операндов цели (D).

Пример:



FMOV-инструкция, копирование значения "10" в регистры данных D10...D14.

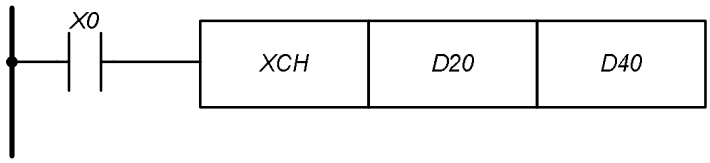
XCH	D1 D2	D P	Обмен данными

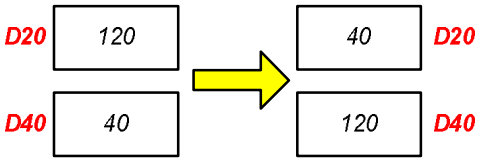
	K	H	F	X	Y	M	T	C	A	B	D
D1							•	•	•	•	•
D2							•	•	•	•	•

Описание:

Обмениваются данные (D1) и (D2).

Пример:





Когда X0 = 1, происходит следующий обмен данными:

ADD	S1 S2 D	D P	Сложение числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Описание:

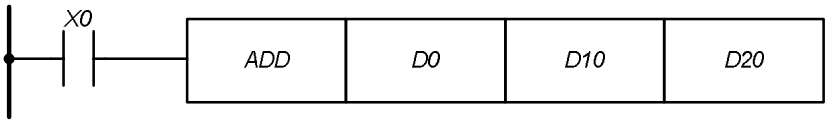
— Двоичные данные в адресах источников (S1) и (S2) суммируются. Результат суммирования запоминается в адресе цели (D). Операция выполняется над знаковыми целочисленными типами данных.

(S1) + (S2) = (D)

— В старшем бите запоминается знак числа суммирования: 0 – знак положительного числа, 1 – знак отрицательного числа.

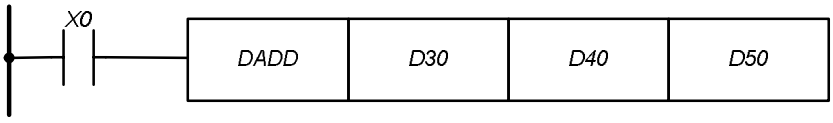
— При выполнении 32-х битной инструкции в ней указывается операнд слова младших 16 бит. Следующий за ней операнд является операндом слова старших 16 бит.

Примеры:



(D0) + (D10) = (D20)

Если включен X0, то суммируются значения данных в регистрах DO и D10. Результат суммирования запоминается в регистре данных D20.



(D31, D30) + (D41, D40) = (D51, D50)

Если включен X0, то суммируются значения данных в регистрах (D31, D30) и (D41, D40). Результат суммирования запоминается в регистрах данных (D51, D50).

SUB	S1 S2 D	D P	Вычитание числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

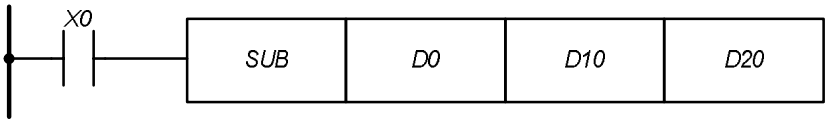
Описание:

— Значение данных в (S2) вычитается из значения данных (S1). Результат вычитания запоминается в адресе цели (D). Операция выполняется над знаковыми целочисленными типами данных.
(S1) – (S2) = (D)

— В старшем бите запоминается знак числа вычитания: 0 – знак положительного числа, 1 – знак отрицательного числа.

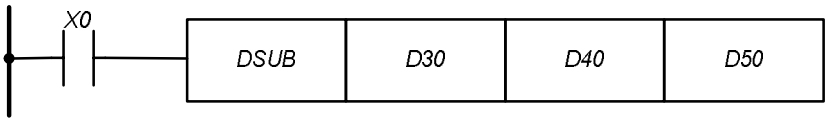
— При выполнении 32-х битной инструкции в ней указывается операнд слова младших 16 бит. Следующий за ней операнд является операндом слова старших 16 бит.

Примеры:



(D0) - (D10) = (D20)

Если включен X0, то находится разность значений данных в регистрах D0 и D10. Результат вычитания запоминается в регистре данных D20.



(D31, D30) - (D41, D40) = (D51, D50)

Если включен X0, то находится разность значений данных в регистрах (D31, D30) и (D41, D40). Результат вычитания запоминается в регистрах данных (D51, D50).

MUL	S1 S2 D	D P	Умножение числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

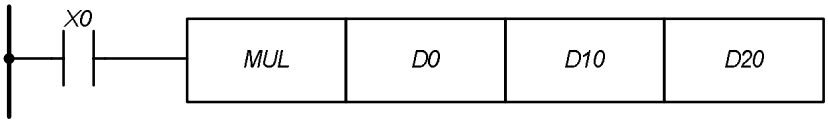
Описание:

— Данные в (S1) и (S2) перемножаются между собой. Результат умножения запоминается по адресу операнда указанного в (D). Операция выполняется над знаковыми целочисленными типами данных.
(S1) x (S2) = (D)

— В старшем бите запоминается знак результата перемножаемых чисел: 0 – знак положительного числа, 1 – знак отрицательного числа.

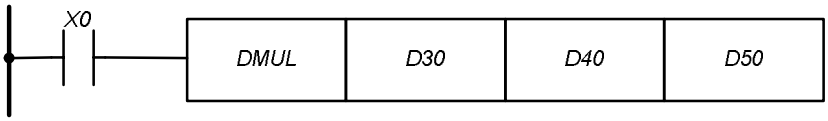
— При выполнении 32-х битной инструкции в ней указывается операнд слова младших 16 бит. Следующий за ней операнд является операндом слова старших 16 бит.

Примеры:



(D0) * (D10) = (D20)

Если включен X0, то находится произведение значений данных в регистрах D0 и D10. Результат умножения запоминается в регистре данных D20.



(D31, D30) * (D41, D40) = (D51, D50)

Если включен X0, то находится произведение значений данных в регистрах (D31, D30) и (D41, D40). Результат умножения запоминается в регистрах данных (D51, D50).

DIV	S1 S2 D	D P	Деление числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Описание:

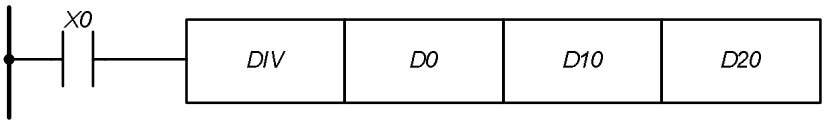
— Значение данных (S1) делится на значения данных из (S2). Целая часть результата деления запоминается в адресе цели (D). Операция выполняется над знаковыми целочисленными типами данных.
(S1) / (S2) = (D)

— В старшем бите запоминается знак числа вычитания: 0 – знак положительного числа, 1 – знак отрицательного числа.

— При выполнении 32-х битной инструкции в ней указывается операнд слова младших 16 бит. Следующий за ней операнд является операндом слова старших 16 бит.

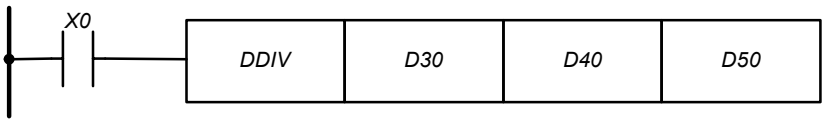
— Деление на ноль приводит к ошибке.

Примеры:



(D0) / (D10) = (D20)

Если включен X0, то выполняется деление значений данных в регистрах D0 и D10. Результат операции запоминается в регистре данных D20.



(D31, D30) / (D41, D40) = (D51, D50)

Если включен X0, то выполняется деление значений данных в регистрах (D31, D30) и (D41, D40). Результат выполнения операции запоминается в регистрах данных (D51, D50).

MOD	S1 S2 D	D P	Вычисление остатка от деления

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Описание:

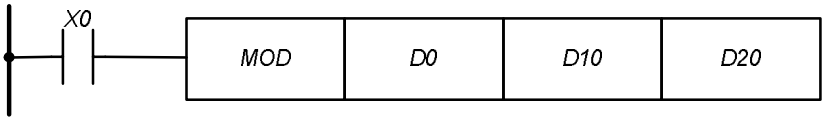
— Значение данных (S1) делится на значения данных из (S2). Остаток от деления запоминается в адресе цели (D). Операция выполняется над знаковыми целочисленными типами данных.
(S1) % (S2) = (D)

— В старшем бите запоминается знак числа вычитания: 0 – знак положительного числа, 1 – знак отрицательного числа.

— При выполнении 32-х битной инструкции в ней указывается операнд слова младших 16 бит. Следующий за ней операнд является операндом слова старших 16 бит.

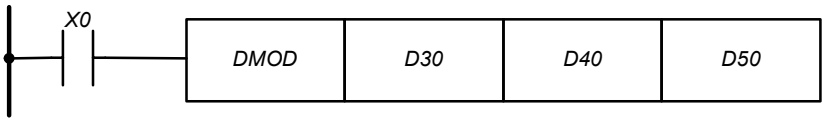
Деление на ноль приводит к ошибке.

Примеры:



(D0) % (D10) = (D20)

Если включен X0, то выполняется деление значений данных в регистрах D0 и D10. Остаток от деления запоминается в регистре данных D20.



(D31, D30) % (D41, D40) = (D51, D50)

Если включен X0, то выполняется деление значений данных в регистрах (D31, D30) и (D41, D40). Остаток от деления запоминается в регистрах данных (D51, D50).

INC	D	D P	Инкрементирование числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Описание:

К значению числа, имеющемуся в (D), прибавляется число 1, как только выполнится входное условие.

Пример:

Значение данных в регистре (D1, D0) при наличии входного сигнала X0 увеличится на 1.

Инструкция активизируется благодаря подключенной функции импульса для того, чтобы процесс суммирования не выполнялся в каждом цикле программы.

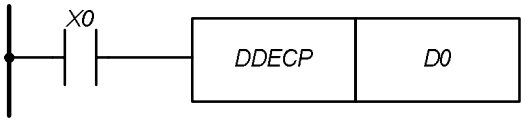
DEC	D	D P	Инкрементирование числовых данных

	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Описание:

От значения числа, имеющегося в (D), отнимается число 1, как только выполнится входное условие.

Пример:



Значение данных в регистре (D1, D0) при наличии входного сигнала X0 уменьшается на 1.

Инструкция активизируется благодаря подключенной функции импульса для того, чтобы процесс вычитания не выполнялся в каждом цикле программы.

WAND	S1 S2 D	D P	Логическое умножение числовых данных (операция «И»)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Примечание:

WAND – 16 битная инструкция, **DAND** – 32 битная инструкция.

Описание:

— Операция "логическое И" для числовых данных выполняется по отдельным битам.

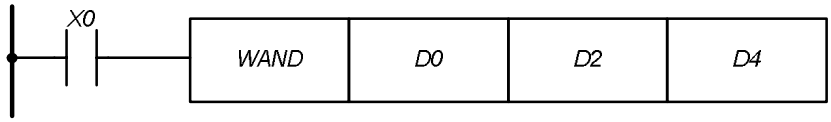
— Данные в (S1) и (S2) побитно логически связываются друг с другом. Результат связи сохраняется в (D).

— Таблица истинности логического умножения:

(S1)	(S2)	(D)
1	1	1
1	0	0
0	1	0
0	0	0

Пример:

Когда X0 = 1, производится логическое умножение данных в регистрах D0 и D2. Результат операции сохраняется в регистре D4.



		b15															b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WAND								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
									⇩								
(D)	D4	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0

WOR	S1 S2 D	D P	Логическое сложение числовых данных (операция «ИЛИ»)

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Примечание:

WOR – 16 битная инструкция, **DOR** – 32 битная инструкция.

Описание:

— Операция "логическое ИЛИ" для числовых данных выполняется по отдельным битам.

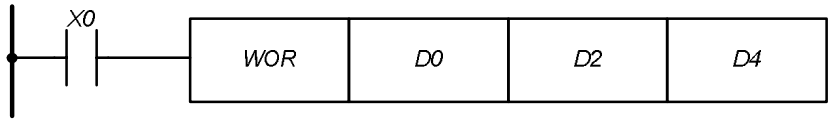
— Данные в (S1) и (S2) побитно логически связываются друг с другом. Результат связи сохраняется в (D).

— Таблица истинности логического сложения:

(S1)	(S2)	(D)
1	1	1
1	0	1
0	1	1
0	0	0

Пример:

Когда X0 = 1, производится логическое сложение данных в регистрах D0 и D2. Результат операции сохраняется в регистре D4.



		b15															b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WAND								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩															
(D)	D4	1	1	1	1	1	1	1	0	0	1	1	1	1	1	0	1

WXOR	S1 S2 D	D P	Логическое операция «исключающее ИЛИ»

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D							•	•	•	•	•

Примечание:

WXOR – 16 битная инструкция, **DXOR** – 32 битная инструкция.

Описание:

— Операция "исключающее ИЛИ" для числовых данных выполняется по отдельным битам.

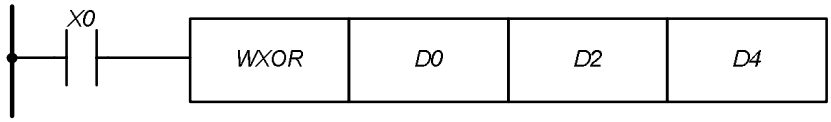
— Данные в (S1) и (S2) побитно логически связываются друг с другом. Результат связи сохраняется в (D).

— Таблица истинности операции "исключающего ИЛИ":

(S1)	(S2)	(D)
1	1	0
1	0	1
0	1	1
0	0	0

Пример:

Когда X0 = 1, производится операция "исключающего ИЛИ" данных в регистрах D0 и D2. Результат операции сохраняется в регистре D4.



		b15															b00
(S1)	D0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
									WAND								
(S2)	D2	0	0	1	1	0	1	1	0	0	1	1	0	1	1	0	0
		⇩															
(D)	D4	1	1	0	0	1	1	1	0	0	1	0	1	1	1	0	1

NEG	D	D P	Логическое отрицание

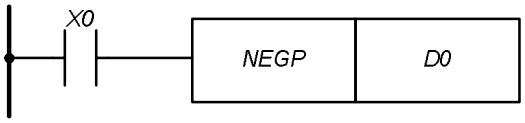
	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Описание:

Операция логического отрицания (инверсия всех битов в бинарном виде и сложение с 1) для числовых данных.

Пример:

Когда X0 = 1, производится операция логического отрицания регистра D0 с последующей его модификацией.



13931 → -13931 + 1 = -13931

		b15														b00
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1

-13931 → 13930 + 1 = 13931

		b15														b00
(D)	D0	1	1	0	0	1	0	0	1	1	0	0	1	0	1	0	1
(D)	D0	0	0	1	1	0	1	1	0	0	1	1	0	1	0	1	1

ROL	D n	D P	Кольцевой сдвиг влево

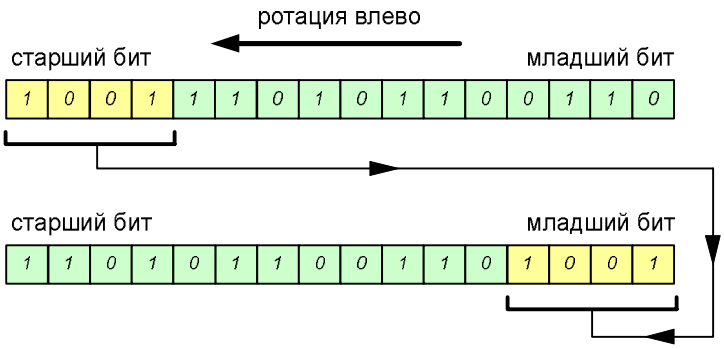
	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

Описание:

Ротация бит на (n) мест влево.

Пример:

Когда X0 = 1, производится операция кольцевого сдвига регистра D10 влево на 4 бита с последующей его модификацией.



ROR	D n	P	Кольцевой сдвиг вправо

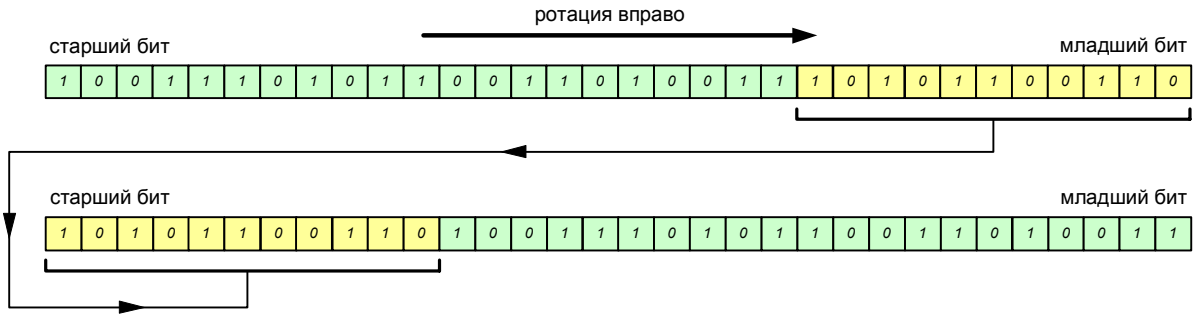
	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•
n	•	•					•	•	•	•	•

Описание:

Ротация бит на (n) мест вправо.

Пример:

Когда X0 = 1, производится операция кольцевого сдвига регистра D10 вправо на 11 бит с последующей его модификацией.



ZRST	D1 D2	D P	Групповой сброс состояний операндов

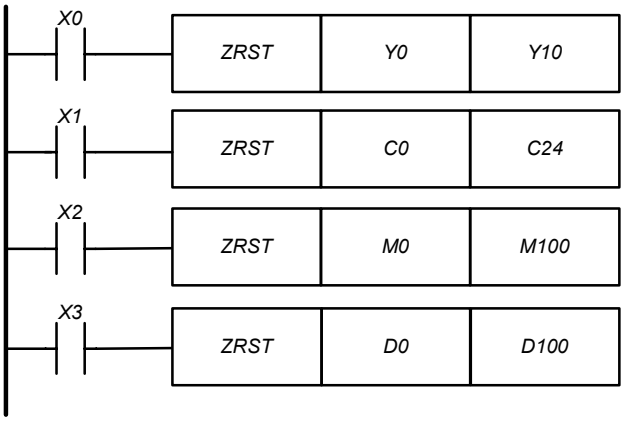
	K	H	F	X	Y	M	T	C	A	B	D
D1					●	●	●	●	●	●	●
D2					●	●	●	●	●	●	●

Описание:

Значения нескольких, следующих друг за другом, операндов (область операндов) могут быть сброшены только одной ZRST-инструкцией, т.е. битовые устройства отключены, а регистры установлены на действительное значение "0".

— В (D1) и (D2) определяется область операндов, которые могут быть сброшены.
— В (D1) и (D2) нужно указать одинаковые типы операндов.
— (D1) – адрес первого операнда, (D2) – адрес последнего операнда.
— Должно соблюдаться: (D1) < (D2).

Пример:



При выполнении соответствующих входных условий битовые операнды Y0…Y10, M0…M100 отключаются до состояния сигнала "0". Словные операнды C0…C24, D0…D100 отключаются до состояния действительного значения "0". Выключаются соответствующие катушки и контакты.

DECO	S D n	P	Дешифратор 8 → 256 бит

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●		●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Примечание:

Когда (D) битовый операнд: (n) = 1…8. Когда (D) словный операнд: (n) = 1…4. Если имеет место быть превышение, то инструкция выполняется с максимально возможным (n) в зависимости от (D).

Описание:

Декодирование данных. Данные в (n) операндов, начиная со стартового адреса, указанного в (S), декодируются. В (D) определяется стартовый адрес операнда цели, куда записывается результат дешифрования.

(n) – число операндов, данные которых должны декодироваться. При указании битового операнда в (D) должно соблюдаться: $1 \leq (n) \leq 8$. При указании словного операнда в (D) должно соблюдаться: $1 \leq (n) \leq 4$.

(S) – стартовый адрес операндов, данные которых должны декодироваться.

2^n – количество операндов цели.

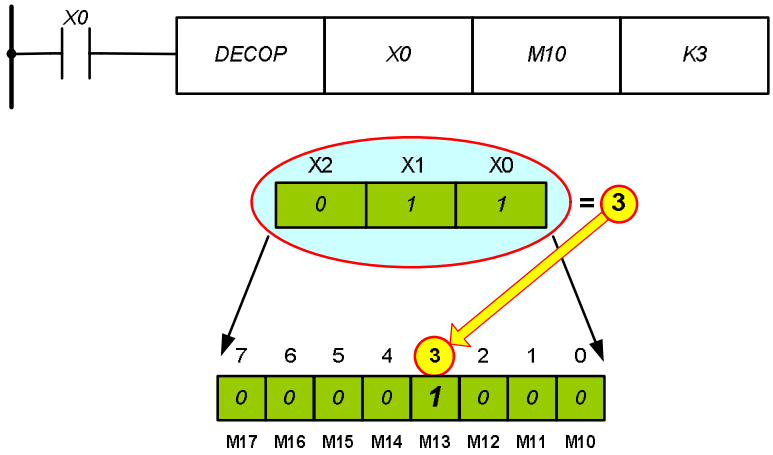
(D) – стартовый адрес операнда цели.

Внимание! Инструкция не выполняется, если (n) = 0.

Инструкция выполняется только в том случае, если включены начальные условия. Соответственно выход остается активным, если входные условия в конце действия снова отключаются.

Пример:

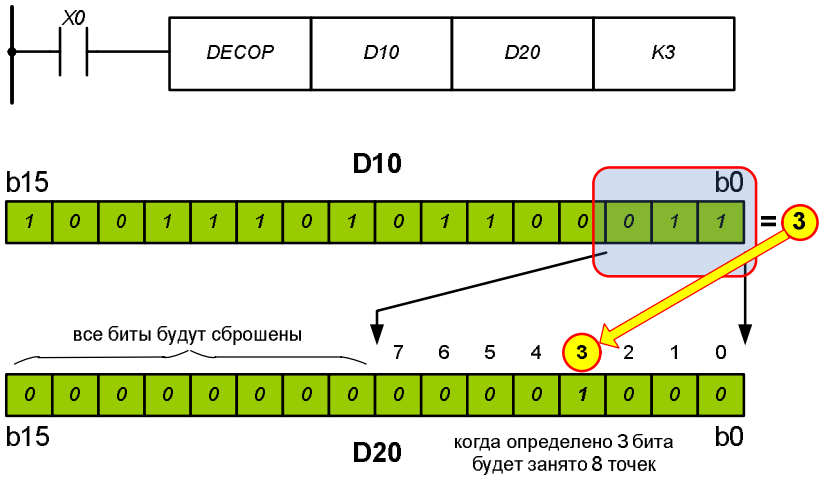
Применение DECO-инструкции с указанием битовых операндов в (D).



Если (n) = 3, обрабатываются входные операнды X0, X1 и X2. Потому что $2^n = 2^3 = 8$ представлены в качестве адресов цели реле M10…M17.

Значения входных операндов равны 1 + 2 = 3. Соответственно третий бит адреса цели, т.е. реле M13, включается. Если обрабатывается значение входного операнда "0", то включается реле M10.

Применение DECO-инструкции с указанием словных операндов в (D).



Младшие 3 бита из регистра данных D10 декодируются. Результат декодирования 1 + 2 = 3 передается в регистр данных D10. В этом регистре данных включается 3-й бит.

Если значение для (n) < 3, то все ненужные биты более высокого номера в адресах цели устанавливаются на ноль.

ENCO	S D n	P	Шифратор 256 → 8 бит

	K	H	F	X	Y	M	T	C	A	B	D
S	●	●		●	●	●	●	●	●	●	●
D					●	●	●	●	●	●	●
n	●	●					●	●	●	●	●

Примечание:

Когда (D) битовый операнд: (n) = 1…8. Когда (D) словный операнд: (n) = 1…4. Если имеет место быть превышение, то инструкция выполняется с максимально возможным (n) в зависимости от (D).

Описание:

Кодирование данных. Данные в 2^n операндов, начиная со стартового адреса, указанного в (S), кодируются. В (D) определяется операнд цели, куда записывается результат кодировки.

2^n – количество операндов, данные которых должны кодироваться,

n – число операндов цели.

При указании битового операнда в (S) должно соблюдаться: $1 \leq (n) \leq 8$. При указании словного операнда в (S) должно соблюдаться: $1 < (n) < 4$.

(S) – стартовый адрес операндов, данные которых должны кодироваться.

(D) – операнд цели.

Если несколько операндов, указанных в (S), имеют значение 1, то обрабатывается только младший бит.

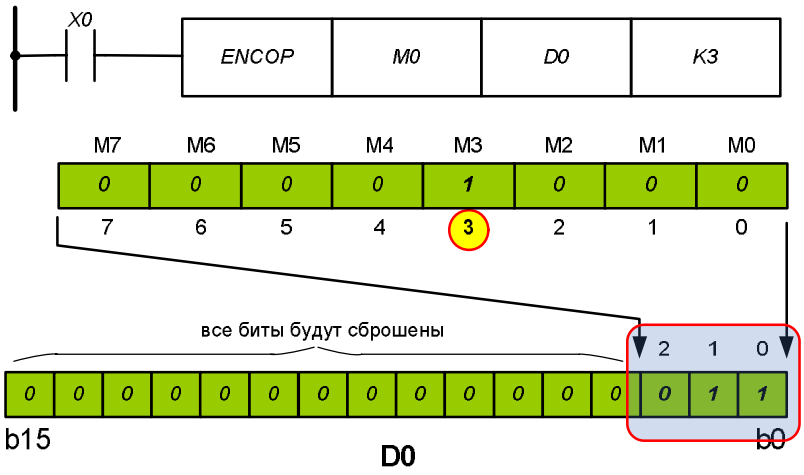
Внимание! Инструкция не выполняется, если (n) = 0.

Инструкция выполняется только в том случае, если включены начальные условия. Соответственно выход остается активным, если входные условия в конце действия снова отключаются.

Пример:

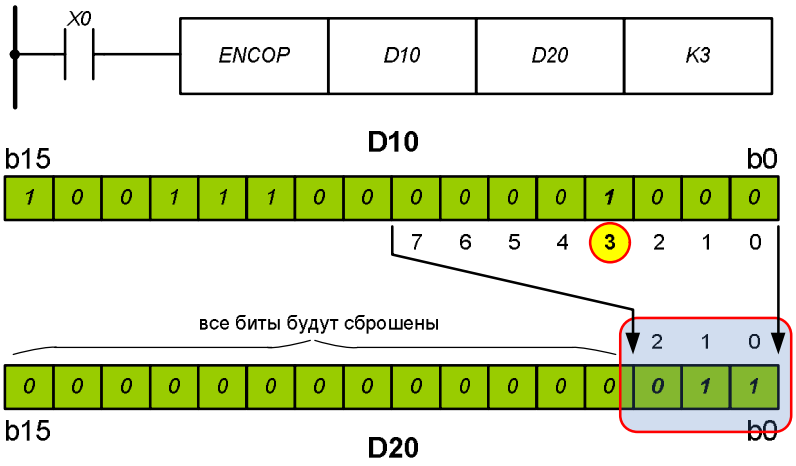
Программирование ENCO-инструкции с указанием битовых операндов в (S)

Если $2^n = 2^3 = 8$, то в качестве адресов выходов имеются реле M0...M7. Поскольку у операндов выхода 3-ий операнд, т.е. реле M3 включается, в регистр данных D0 записывается значение 3.



Программирование ENCO-инструкции с указанием словных операндов в (S)

В регистре данных D10 включается 3-ий бит. Тем самым значение числа 3 кодируется и сохраняется в регистре данных D10.



SUM	S D	D P	Сумма единичных битов

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D							•	•	•	•	•

Описание:

Определение количества активных битов в слове данных.

— Определяется количество включенных битов в (S).

— Определяемое значение заносится в (D).

Если обрабатывается 32-х битная операция, то старшие 16 битов (D + 1) операндов цели (D) устанавливаются в ноль, так как максимальное число включенных битов в (S) составляет 32.

BON	S D n	D P	Проверка состояния битов

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D					•	•					
n	•	•					•	•	•	•	•

Примечание:

Необходимое условие: (n) = 0…15 (16 бит), (n) = 0…31 (32 бит).

Описание:

Проверяется отдельный бит внутри слова данных. Если бит по номеру (n) включен внутри (S), то включается соответствующий бит в (D).

SQR	S D	D P	Вычисление квадратного корня

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Описание:

Вычисление корня квадратного, $(D) = \sqrt{(S)}$

Выполняется вычисление корня квадратного числа в (S) и с округлением до целого числа результат заносится в (D). Операция выполняется над знаковыми целочисленными типами данных.

Внимание! Корень квадратный из отрицательного числа всегда приводит к ошибке.

FLT	S D	D P	Преобразование целого числа в число с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•					•	•	•	•	•
D											•

Описание:

Преобразование целочисленного знакового числа в формат числа с плавающей запятой.

— Целое число в (S) преобразовывается в число с плавающей запятой и заносится в (D).

— Результат преобразования всегда будет записываться в 32-х битный регистр данных.

PWM	S1 S2 D		Импульсный выход с ШИМ

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D					•						

Примечание:

Значение операнда (S1) должно быть меньше или равно значения операнда (S2).

Описание:

(S1) – длительность импульса, t.

(S2) – продолжительность периода, T.

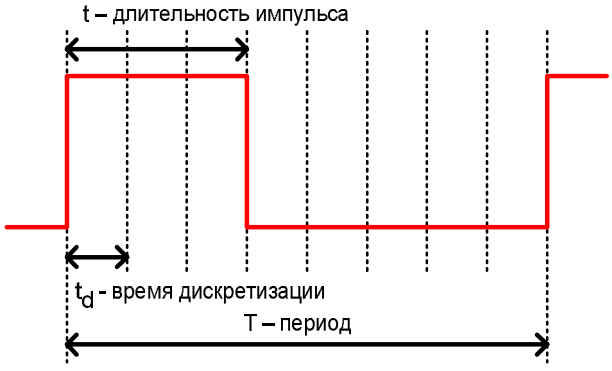
(D) – адрес выхода, Y6 или Y7.

Области допустимых значений для (S1) и (S2) от 1 до 32767 – количество интервалов дискретизации. Период дискретизации задаётся одновременно для обоих каналов 100мкс или 1мс служебным регистром D356 см. п. 4.6 Параметр S2 при использовании двух каналов должен быть одинаковым.

ШИМ-сигнал присутствует на выходе до тех пор пока активен сигнал на входе инструкции PWM.

Пример:

Пусть период дискретизации составляет 100мкс, тогда при выполнении входного условия X0 = 1 на выходе Y6 появится ШИМ-сигнал с периодом *8 × 100мкс = 0.8мс* и длительностью импульса *3 × 100мкс = 0.3мс*.



ABS	D	D P	Абсолютное значение

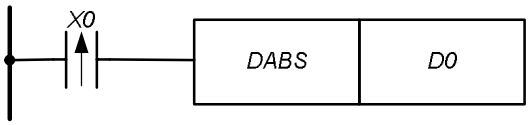
	K	H	F	X	Y	M	T	C	A	B	D
D							•	•	•	•	•

Описание:

Нахождение модуля числа и запись его абсолютного значения. Если значение в (D) отрицательное, то после выполнения ABS-инструкции (D) знак "-" отбрасывается и число становится положительным. Если (D) имело положительное значение, то никаких изменений не происходит.

Пример:

При выполнении входного условия определяется модуль числа в регистре (D1, D0).



POW	S1 S2 D	D P	Возведение в степень

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•					•	•	•	•	•
S2	•	•					•	•	•	•	•
D											•

Описание:

Возведение в степень, $(D) = (S1)^{(S2)}$.

Выполняется возведение в степень (S2) числа (S1), результат заносится в (D). Операция выполняется над знаковыми целочисленными типами данных.

DECMP	S1 S2 D	D P	Сравнение чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D					•	•					

Примечание:

— Только 32-х битная инструкция.

— Операнд (D) занимает 3 непрерывных адреса.

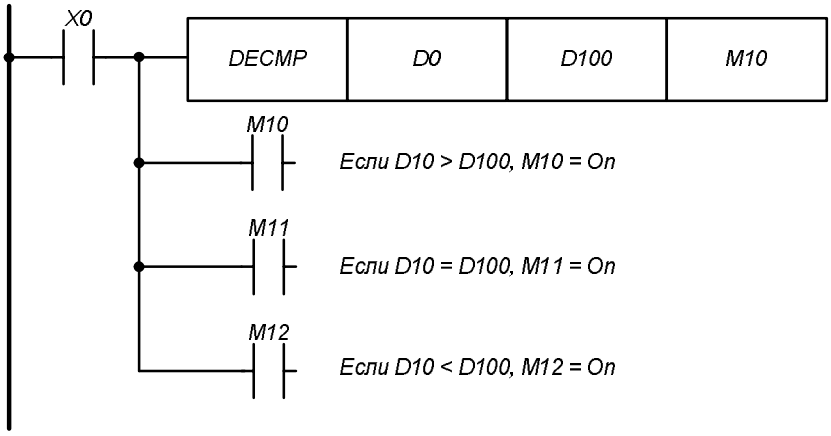
— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT.**

Описание:

Сравнение двух двоичных чисел с плавающей запятой с выдачей результата сравнения.

— DECMP-инструкция сравнивает число с плавающей запятой в (S1) с числом с плавающей запятой в (S2).
— Результат сравнения сохраняется в 3-х следующих друг за другом операндах.
— Если число в (S2) меньше числа (S1), то включается битовый операнд (D).
— Если число в (S2) равно числу (S1), то включается битовый операнд ((D)+1).
— Если число в (S2) больше числа (S1), то включается битовый операнд ((D)+2).
— Опрошенные операнды выходов остаются включенными после отключения условий выполнения DECMP-инструкции.

Пример:



При включении контакта X0 сравнивается число с плавающей запятой, указанное в D100 (S2), с числом с плавающей запятой, указанным в D0 (S1). Если число в D100 меньше числа D0, то включается реле M10. Если число в D100 равно числу D0, то включается реле M11. Если число в D100 больше числа D0, то включается реле M12.

Для получения результатов сравнения в виде: $\leq \geq \neq$ можно использовать параллельные комбинации контактов M10 – M12. Для сброса результата можно использовать команды RST, ZRST.

DEZCP	S1 S2 S D	D P	Зонное сравнение чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
S	•	•	•								•
D					•	•					

Примечание:

— Только 32-х битная инструкция.

— Операнд (D) занимает 3 непрерывных адреса.

— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используются инструкция **FLT.**

Описание:

Сравнение числа с плавающей запятой с выделенной (указанной) областью и выдачей результата сравнения.

— DEZCP-инструкция сравнивает число с плавающей запятой в (S) с областью между (S1) и (S2).

— Результат сравнения сохраняется в 3-х, следующих один за другим, операндах.

— Если число в (S) меньше чисел (S1) и (S2), то включается битовый операнд (D).

— Если число в (S) равно числу между (S1) и (S2), то включается битовый операнд ((D)+1).

— Если число в (S) больше чисел между (S1) и (S2), то включается битовый операнд ((D)+2).

— Опрошенные операнды выходов остаются включенными после отключения условий выполнения DEZCP -инструкции.

— Если (S1) больше (S2), то все операнды (D) будут сброшены.

DEADD	S1 S2 D	D P	Сложение чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.

— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT**.

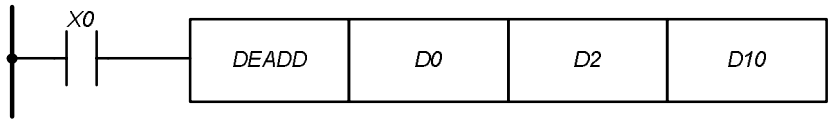
Описание:

Вычисление суммы двух чисел в двоичном формате с плавающей запятой.

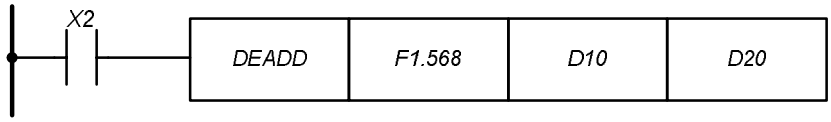
— Число с плавающей запятой, заданное в (S1), суммируется с числом с плавающей запятой в (S2). Результат запоминается в (D).
— Для каждого операнда используется по два следующих друг за другом регистра.
— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.

Примеры:

При включении входа X0, к числу с плавающей запятой, записанному в (D1, D0) будет прибавлено число с плавающей запятой, записанное в (D3, D2). Результат сохранится в (D11, D10).



При включении входа X2, к константе F1.568 прибавится число с плавающей запятой, записанное в (D11, D10). Результат сохранится в (D21, D20).



DESUB	S1 S2 D	D P	Вычитание чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.
— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используются инструкция **FLT**.

Описание:

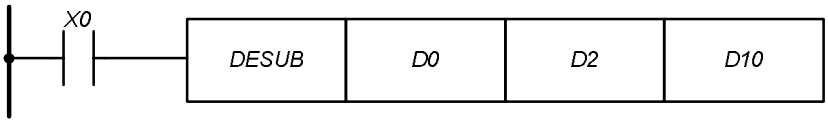
Вычисление разности двух чисел в двоичном формате с плавающей запятой.

— Число с плавающей запятой, заданное в (S2), вычитается из числа с плавающей запятой в (S1). Результат запоминается в (D).

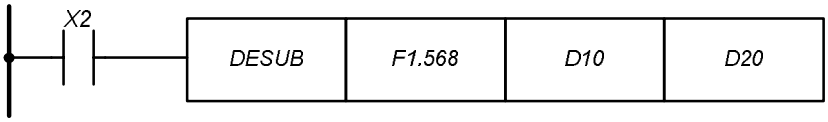
— Для каждого операнда используется по два следующих друг за другом регистра.
— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.

Примеры:

При включении входа X0, из числа с плавающей запятой, записанного в (D1, D0) будет вычтено число с плавающей запятой, записанное в (D3, D2). Результат сохранится в (D11, D10).



При включении входа X2, из константы F1.568 будет вычтено число с плавающей запятой, записанное в (D11, D10). Результат сохранится в (D21, D20).



DEMUL	S1 S2 D	D P	Умножение чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.
— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT**.

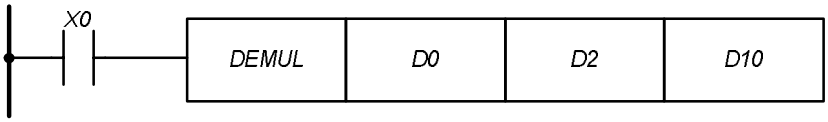
Описание:

Вычисление произведения двух чисел в двоичном формате с плавающей запятой.

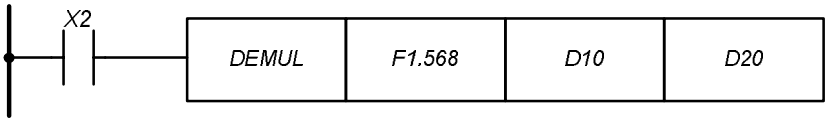
— Число с плавающей запятой, заданное в (S1), умножается на число с плавающей запятой в (S2). Результат запоминается в (D).
— Для каждого операнда используется по два следующих друг за другом регистра.
— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.

Примеры:

При включении входа X0, число с плавающей запятой, записанное в (D1, D0) будет умножено на число с плавающей запятой, записанное в (D3, D2). Результат сохранится в (D11, D10).



При включении входа X2, константа F1.568 будет умножена на число с плавающей запятой, записанное в (D11, D10). Результат сохранится в (D21, D20).



DEDIV	S1 S2 D	D P	Деление чисел с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.
— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT.**

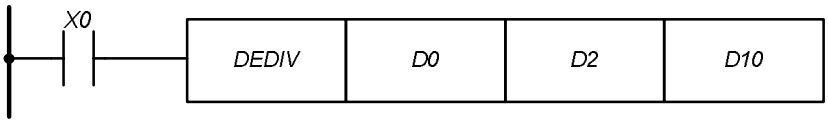
Описание:

Вычисление частного от деления двух чисел в двоичном формате с плавающей запятой.

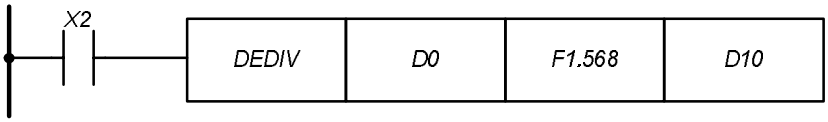
— Число с плавающей запятой, заданное в (S1), делится на число с плавающей запятой в (S2). Результат запоминается в (D).
— Для каждого операнда используется по два следующих друг за другом регистра.
— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.
— Операнд (S2) не может быть равен нулю т.к. деление на нуль не допустимо.

Примеры:

При включении входа X0, число с плавающей запятой, записанное в (D1, D0) будет разделено на число с плавающей запятой, записанное в (D3, D2). Результат сохранится в (D11, D10).



При включении входа X2, число с плавающей запятой, записанное в (D1, D0) будет разделено на константу F1.568. Результат сохранится в (D11, D10).



DESQR	S D	D P	Корень квадратный в формате с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.
— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT**.
— Необходимое условие: (S) ≥ 0

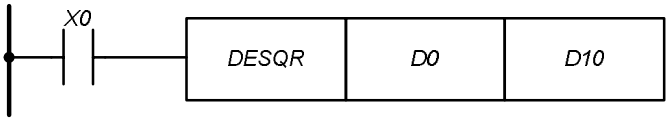
Описание:

Вычисление квадратного корня числа в двоичном формате с плавающей запятой.

— Из числа с плавающей запятой, заданного в (S), вычисляется корень квадратный. Результат запоминается в (D).
— Для каждого операнда используется по два следующих друг за другом регистра.
— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.

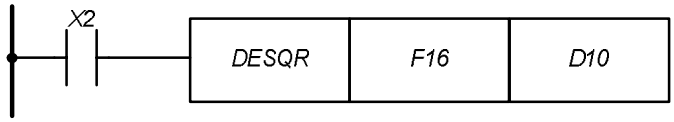
Примеры:

При включении входа X0 вычисляется корень квадратный из числа с плавающей запятой в (D1, D0). Результат сохраняется в (D11, D10).



$\sqrt{(D1, D0)} \rightarrow (D11, D10)$

При включении входа X0 вычисляется корень квадратный из константы F16. Результат сохраняется в (D11, D10).



DEPOW	S1 S2 D	D P	Степень числа в формате с плавающей запятой

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•	•								•
S2	•	•	•								•
D											•

Примечание:

— Только 32-х битная инструкция.

— Типы K и H не преобразуются в F, а проецируются в область памяти. Для преобразования целочисленных типов данных в данные с плавающей запятой используется инструкция **FLT.**

Описание:

Возведение числа в степень в двоичном формате с плавающей запятой

— Число (S1) возводится в степень (S2). Результат запоминается в (D).
$(S1)^{(S2)} = (D)$.

— Для каждого операнда используется по два следующих друг за другом регистра.

— Может применяться один и тот же операнд для источника и для цели. В этом случае, рассчитанный результат снова запоминается в операнде-источнике и может использоваться для следующего расчета. Этот процесс повторяется в каждом цикле программы.

INT	S D	D P	Преобразование числа с плавающей запятой в целое

	K	H	F	X	Y	M	T	C	A	B	D
S	.										•
D							•	•	•	•	•

Описание:

Преобразование числа с плавающей запятой в целое с округлением до ближайшего.

— Число с плавающей запятой, заданное в (S), округляется до ближайшего целого значения и запоминается в (D).

— Операнд-источник всегда является операндом двойного слова.

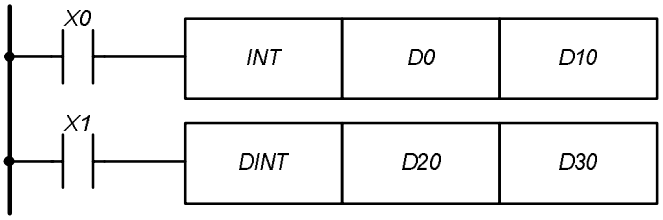
— При применении INT-инструкции словный операнд является операндом цели.

— При применении DINT-инструкции операнд цели является операндом двойного слова.
— INT-инструкция является обратной функцией FLT-инструкции.

Пример:

При включении входа X0 число с плавающей запятой в (D0, D1) округляется до ближайшего меньшего целого значения. Результат сохранится в D10.

При включении входа X1 число с плавающей запятой в (D20, D21) округляется до ближайшего меньшего целого значения. Результат сохранится в (D30, D31).



TRD	D	P	Чтение данных времени

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Примечание:

— Операнд D занимает 3 последовательных адреса.

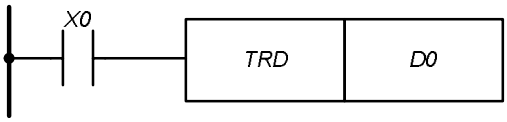
Описание:

Чтение текущего значения часов реального времени.

— С помощью TRD-инструкции выполняется чтение данных реального времени (часы, минуты, секунды).
— Эти данные хранятся в 3-х следующих друг за другом адресах операнда (D).

Пример:

С включением входа X0 считываются данные реального времени и запоминаются в регистрах D0…D2.



Регистр	Назначение	Значение	Пример	
D0	Секунды	0…59	20	12:36:20
D1	Минуты	0…59	36	
D2	Часы	0…23	12	

TWR	S	P	Запись данных времени

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Примечание:

— Операнд D занимает 3 последовательных адреса.

Описание:

Изменение текущего значения часов реального времени.

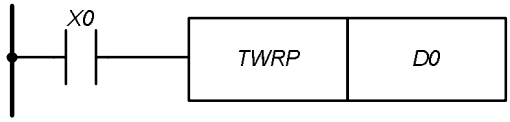
С помощью TWR-инструкции выполняется запись данных реального времени (часы, минуты, секунды).

Эти данные хранятся в 3-х следующих друг за другом адресах операнда в (S).

Если значения в (S) превышают диапазон значений, то будет зафиксирована ошибка.

Пример:

При выполнении входного условия происходит переустановка часов реального времени контроллера на указанное в регистрах D0…D2.



Регистр	Назначение	Значение	Пример	
D0	Секунды	0…59	42	03:11:42
D1	Минуты	0…59	11	
D2	Часы	0…23	3	

DRD	D	P	Чтение текущей даты

	K	H	F	X	Y	M	T	C	A	B	D
D											•

Примечание:

— Операнд D занимает 3 последовательных адреса.

Описание:

Чтение текущего значения даты.

— С помощью DRD-инструкции выполняется чтение текущей даты (день, месяц, год).

— Эти данные хранятся в 3-х следующих друг за другом адресах операнда (D).

Пример:

С включением входа X0 считываются данные о текущей дате и запоминаются в регистрах D0…D2.

Регистр	Назначение	Значение	Пример	
D0	День	1…31	26	26.01:22
D1	Месяц	1…12	1	
D2	Год	21…99	22	

DWR	S	P	Установка текущей даты

	K	H	F	X	Y	M	T	C	A	B	D
S											•

Примечание:

— Операнд S занимает 3 последовательных адреса.

Описание:

Изменение текущего значения часов реального времени.

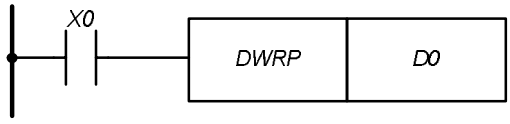
С помощью DWR-инструкции выполняется запись текущей даты (день, месяц, год).

Эти данные хранятся в 3-х следующих друг за другом адресах операнда в (S).

Если значения в (S) превышают диапазон значений, то будет зафиксирована ошибка.

Пример:

При выполнении входного условия происходит переустановка текущей даты контроллера на указанную в регистрах D0…D2.



Регистр	Назначение	Значение	Пример	
D0	День	1…31	4	04.02.22
D1	Месяц	1…12	2	
D2	Год	21…99	22	

LD#	S1 S2	D	Логические операции контактного типа

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Примечание:

— Символ #: это &, |, ^.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Выполнение логической операции "И", "ИЛИ", "Исключающее ИЛИ" над операндами (S1) и (S2), и включение LD-контакта в зависимости от результата операции.

Инструкции LD# в программе располагаются крайними слева и открывают логическую связь или являются условиями выполнения правосторонних команд.

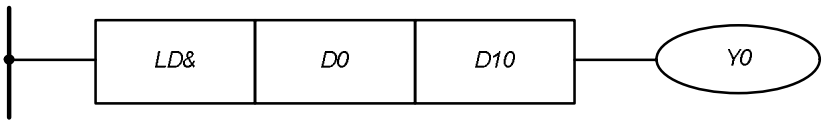
16-бит команда	32-бит команда	Контакт замкнут, если:	Контакт разомкнут, если:
LD&	DLD&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
LD|	DLD|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
LD^	DLD^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: логическое умножение (И)

|: логическое сложение (ИЛИ)

^: исключающее ИЛИ (XOR)

Пример:



AND#	S1 S2	D	Логические операции контактного типа Последовательный контакт

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Примечание:

— Символ #: это &, |, ^.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Выполнение логической операции "И", "ИЛИ", "Исключающее ИЛИ" над операндами (S1) и (S2), и включение AND-контакта в зависимости от результата операции.

Инструкции AND# в программе располагаются после команд LD и образуют с ними логическую связь "И".

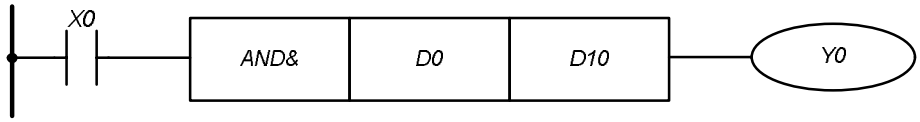
16-бит команда	32-бит команда	Контакт замкнут, если:	Контакт разомкнут, если:
AND&	DAND&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
AND|	DAND|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
AND^	DAND^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: логическое умножение (И)

|: логическое сложение (ИЛИ)

^: исключающее ИЛИ (XOR)

Пример:



OR#	S1 S2	D	Логические операции контактного типа Параллельный контакт

	K	H	F	X	Y	M	T	C	A	B	D
S1	●	●		●	●	●	●	●	●	●	●
S2	●	●		●	●	●	●	●	●	●	●

Примечание:

— Символ #: это &, |, ^.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Выполнение логической операции "И", "ИЛИ", "Исключающее ИЛИ" над операндами (S1) и (S2), и включение OR-контакта в зависимости от результата операции.

Инструкция OR# в программе располагается слева, параллельно команде LD и образует с ней логическую связь "ИЛИ".

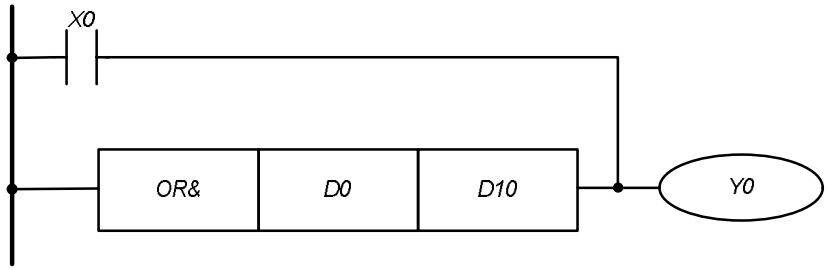
16-бит команда	32-бит команда	Контакт замкнут, если:	Контакт разомкнут, если:
OR&	DOR&	(S1) & (S2) ≠ 0	(S1) & (S2) = 0
OR|	DOR|	(S1) | (S2) ≠ 0	(S1) | (S2) = 0
OR^	DOR^	(S1) ^ (S2) ≠ 0	(S1) ^ (S2) = 0

&: логическое умножение (И)

|: логическое сложение (ИЛИ)

^: исключающее ИЛИ (XOR)

Пример:



LD*	S1 S2	D	Операции сравнения контактного типа

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Примечание:

— Символ *****: это =, >, <, <>, ≤, ≥.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Сравнение значений операндов (S1) и (S2), и включение LD-контакта в зависимости от результата операции.

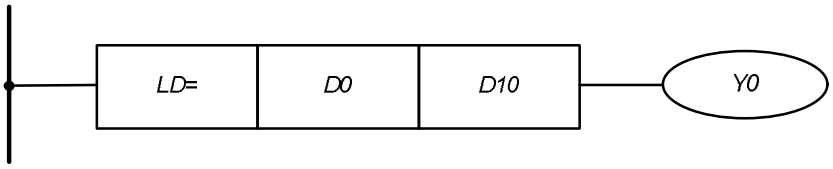
— Инструкции LD* в программе располагаются крайними слева и начинают логическую связь или являются условиями выполнения правосторонних команд.

— Если результат сравнения истинен, включается LD-контакт.

— Если результат сравнения ложен, LD-контакт не включается.

16-бит команда	32-бит команда	Контакт замкнут, если:	Контакт разомкнут, если:
LD=	DLD=	(S1) = (S2)	(S1) ≠ (S2)
LD>	DLD>	(S1) > (S2)	(S1) ≤ (S2)
LD<	DLD<	(S1) < (S2)	(S1) ≥ (S2)
LD<>	DLD<>	(S1) ≠ (S2)	(S1) = (S2)
LD<=	DLD<=	(S1) ≤ (S2)	(S1) > (S2)
LD>=	DLD>=	(S1) ≥ (S2)	(S1) < (S2)

Пример:



AND*	S1 S2	D	Операции сравнения контактного типа Последовательное соединение

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Примечание:

— Символ *****: это =, >, <, <>, ≤, ≥.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Сравнение значений операндов S1 и S2, и включение AND-контакта в зависимости от результата операции.

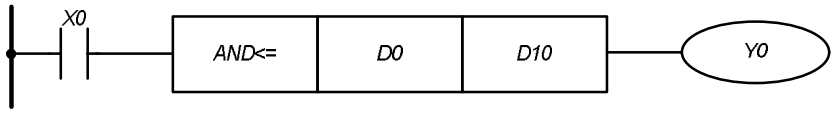
— Инструкции AND* в программе располагаются после LD-инструкций и образуют с ними логическую связь "И".

— Если результат сравнения истинен, включается AND-контакт.

— Если результат сравнения ложен, AND-контакт не включается.

16-бит команда	32-бит команда	Контакт замкнут, если:	Контакт разомкнут, если:
AND=	DAND=	(S1) = (S2)	(S1) ≠ (S2)
AND>	DAND>	(S1) > (S2)	(S1) ≤ (S2)
AND<	DAND<	(S1) < (S2)	(S1) ≥ (S2)
AND<>	DAND<>	(S1) ≠ (S2)	(S1) = (S2)
AND<=	DAND<=	(S1) ≤ (S2)	(S1) > (S2)
AND>=	DAND>=	(S1) ≥ (S2)	(S1) < (S2)

Пример:



OR*	S1 S2	D	Операции сравнения контактного типа Параллельное соединение

	K	H	F	X	Y	M	T	C	A	B	D
S1	•	•		•	•	•	•	•	•	•	•
S2	•	•		•	•	•	•	•	•	•	•

Примечание:

— Символ *****: это =, >, <, <>, ≤, ≥.

— Битовые операнды берутся по 16 или по 32, в зависимости от типа инструкции, и переводятся в целочисленный тип данных для дальнейшей обработки.

Описание:

Сравнение значений операндов (S1) и (S2), и включение OR-контакта в зависимости от результата операции.

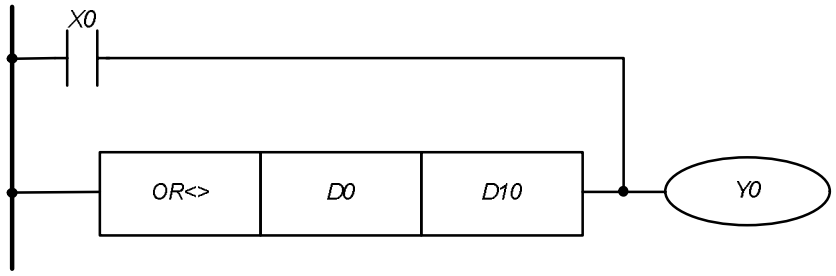
— Инструкция OR* в программе располагается параллельно LD-инструкции и образует с ними логическую связь "ИЛИ".

— Если результат сравнения истинен, включается OR-контакт.

— Если результат сравнения ложен, OR-контакт не включается.

16-бит команда	32-бит команда	Контакт за-мкнут, если:	Контакт разо-мкнут, если:
OR=	DOR=	(S1) = (S2)	(S1) ≠ (S2)
OR>	DOR>	(S1) > (S2)	(S1) ≤ (S2)
OR<	DOR<	(S1) < (S2)	(S1) ≥ (S2)
OR<>	DOR<>	(S1) ≠ (S2)	(S1) = (S2)
OR<=	DOR<=	(S1) ≤ (S2)	(S1) > (S2)
OR>=	DOR>=	(S1) ≥ (S2)	(S1) < (S2)

Пример:



8. Управление драйвером шагового двигателя

Управление драйвером шагового двигателя осуществляется при помощи команд, которые задают особенности вращения или перемещения. Все команды подразделяются на две группы **RUN** и **MOVE**. Группа RUN предназначена для контроля текущей скорости привода, а MOVE – для контроля перемещения. Для начала перемещения, после выбора команды и задания параметров драйвера, вызывается инструкция **SPIN**.

SPIN		P	Выполнить заданное перемещение
Адрес	**Тип объекта**	**Описание Modbus объекта**	
5100h	**Coils**	Запись единицы в объект (даже если он не сброшен) запускает параметризированное перемещение, сброс – игнорируется.	

Описание:

Инструкция запускает параметризированное перемещение. Параметры перемещения задаются в служебных регистрах, которые, также как и инструкции драйвера шагового двигателя, доступны через Modbus протокол в режиме RUN. Инструкция имеет приоритет выполнения ниже чем у xSTOP и xHIZ. Во избежание возникновения ошибок рекомендуется перед вызовом SPIN проверять флаги BUSY_MOVE и BUSY_RUN. Ниже представлено подробное описание служебных регистров драйвера.

Адрес	**Тип объекта**	**Служебный регистр**		**Бит**	**Описание**
		Номер	**Название**		
5000h	Holding Registers	D357	SPEED	32	Задаваемая (целевая) скорость вращения двигателя в импульсах в секунду, далее pps, для команд группы RUN и максимальная скорость для команд группы MOVE. Нижний порог 8pps. Верхний предел 120000pps при сброшенном FS_SW_EN, при установленном $SPEED_{max}=120000*2^{U\_STEP}$.
5002h	Holding Registers	D359	MIN_SPEED	32	Минимальная скорость вращения для команд группы RUN. Для группы MOVE, так же является минимальной скоростью, если не установлен флаг CMIN_SPD_EN. В случае установленного CMIN_SPD_EN оптимальная минимальная скорость будет рассчитываться автоматически.
5004h	Holding Registers	D361	ACC	16	Ускорение в pps^2.
5005h	Holding Registers	D362	DEC	16	Торможение в pps^2.
5006h	Holding Registers	D363	ABS	32	Текущая позиция. Единица значения равна перемещению на величину дробления шага.

<table>
<tr><th rowspan="2">Адрес</th><th rowspan="2">Тип объекта</th><th colspan="2">Служебный регистр</th><th rowspan="2">Бит</th><th rowspan="2">Описание</th></tr>
<tr><th>Номер</th><th>Название</th></tr>
<tr><td>5008h</td><td>Inputs Registers</td><td>D365</td><td>U_POS</td><td>16</td><td>Текущее положение микрошага в четырёх полных шагах. Указывает на положение ротора двигателя относительно полюсов статора. Регистр доступен только для чтения.</td></tr>
<tr><td>5009h</td><td>Holding Registers</td><td>D366</td><td>U_STEP</td><td>16</td><td>Величина дробления полного шага.<table><tr><th>Значение регистра</th><th>Дробление</th></tr><tr><td>0</td><td>1/1</td></tr><tr><td>1</td><td>1/2</td></tr><tr><td>2</td><td>1/4</td></tr><tr><td>3</td><td>1/8</td></tr><tr><td>4</td><td>1/16</td></tr><tr><td>5</td><td>1/32</td></tr><tr><td>7</td><td>1/128</td></tr><tr><td>8</td><td>1/256</td></tr></table>Значение 6 для контроллера недопустимо.</td></tr>
<tr><td>500Ah</td><td>Holding Registers</td><td>D374</td><td>DIR</td><td>16</td><td rowspan="2">Направление:
"1" – прямое,
"0" – обратное.</td></tr>
<tr><td>500Ah</td><td>Coils</td><td colspan="3">DIR</td></tr>
<tr><td>500Bh</td><td>Holding Registers</td><td>D377</td><td>FS_SPD_THR</td><td>32</td><td>Пороговое значение скорости перехода с микрошага на полный шаг, измеряется в полных шагах</td></tr>
<tr><td>500Dh</td><td>Holding Registers</td><td>D379</td><td>FS_SW_EN</td><td>16</td><td rowspan="2">Установка объекта включает режим перехода на полный шаг с микро. Данная опция увеличивает крутящий момент на высоких скоростях. Сброс отключит данную опцию (морфинг/torque boost).</td></tr>
<tr><td>500Dh</td><td>Coils</td><td colspan="3">FS_SW_EN</td></tr>
<tr><td>500Eh</td><td>Holding Registers</td><td>D372</td><td>TARGET_POS</td><td>32</td><td>Целевая позиция, которой необходимо достичь. Единица значения равна перемещению на величину дробления шага.</td></tr>
<tr><td>5010h</td><td>Holding Registers</td><td>D376</td><td>CMD</td><td>16</td><td>Команда драйверу (см. след. таблицу).</td></tr>
</table>

Адрес	Тип объекта	Служебный регистр		Бит	Описание
		Номер	Название		
5011h	Holding Registers	D375	SW_INPUT	16	Физический вход, к которому подключен концевой датчик для работы команд GOUNTIL_... и …RELEASE. Значения 0…7. Внимание: В основном теле программы требуется объявление прерываний по используемым входам. Обработчик прерывания можно оставить пустым. Пример: Датчик подключен к X3 (IN3), тогда программа будет выглядеть так: *FEND* *I 1003* *IRET* *END*
5012h	Holding Registers	D367	ACC_CUR	16	Ток ускорения, мА. Область допустимых значений: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…5000 SMSD-8.0Modbus – 2800…10000
5013h	Holding Registers	D368	DEC_CUR	16	Ток торможения, мА. Область допустимых значений: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…5000 SMSD-8.0Modbus – 2800…10000
5014h	Holding Registers	D369	STEADY_CUR	16	Ток движения с постоянной скоростью, мА. Область допустимых значений: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…4200 SMSD-8.0Modbus – 2800…8000
5015h	Holding Registers	D370	HOLD_CUR	16	Ток удержания, мА. Область допустимых значений: SMSD-1.5Modbus ver.3 - 150…1500; SMSD-4.2Modbus – 1000…4200 SMSD-8.0Modbus – 2800…8000

Адрес	Тип объекта	Служебный регистр		Бит	Описание
		Номер	Название		
5016h	Holding Registers	D382	CMIN_SPD_EN	16	"1" - использовать автоматический расчёт начальной и конечной скорости движения для команд группы MOVE. "0" - использовать в качества начальной и конечной скорости MIN_SPEED.
5017h	Holding Registers	D380	ERROR_SET_HIZ	16	Биты регистра определяют, при каких ошибках драйвера двигатель будет полностью обесточен (вал свободно вращается), далее HiZ-состояние.
5017h	Coils	TERMAL_OVER_CURRENT_ERROR_SET_HIZ			0-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки TERMAL_ERROR – перегрев микросхемы драйвера (регистр D381).
5018h	Coils	SOFTWARE_ERROR_SET_HIZ			1-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки SOFTWARE_ERROR – внутренняя ошибка контроллера (регистр D381).
5019h	Coils	CMD_ERROR_SET_HIZ			2-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки CMD_ERROR - невозможность обработки поступившей команды (регистр D381).
501Ah	Coils	DATA_ERROR_SET_HIZ			3-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки DATA_ERROR (регистр D381) – неверный ввод данных в регистры ACC, DEC, U_STEP.
501Bh	Coils	OUT_OF_LIM_MIN_SPD_ERROR_SET_HIZ			4-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки OUT_OF_ LIM_MIN_SPD_ERROR_SET_HIZ – заданная скорость меньше минимальной (регистр D381).
501Ch	Coils	OUT_OF_LIM_MAX_SPD_ERROR_SET_HIZ			5-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки OUT_OF_ LIM_MAX_SPD_ERROR_SET_HIZ – превышение максимально возможной скорости (регистр D381).

Адрес	Тип объекта	Служебный регистр		Бит	Описание
		Номер	Название		
501Dh	Coils	UNREACHABLE_FS_SPD_ ERROR_SET_HIZ			6-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки UNREACHABLE_FS_SPD_ERROR_ SET_HIZ – недостижимая скорость перехода на полный шаг в режиме torque boost (регистр D381).
501Eh	Coils	NOT_APP_FS_PARAM_ ERROR_SET_HIZ			7-бит регистра D380. Установка бита приведёт к HiZ-состоянию двигателя при возникновении ошибки NOT_APP_ FS_PARAM_ERROR_SET_HIZ – переход из режима torque boost невозможен пока двигатель вращается (регистр D381).
5027h	Holding Registers	D381	ERROR_CODE	16	Код ошибки. Описание битов регистра (ошибок) ниже.
5027h	Coils	TERMAL_OVER_CURRENT_ ERROR			0-бит регистра D381. TERMAL_OVER_ CURRENT_ERROR – перегрев микросхемы драйвера или превышение тока в обмотках двигателя.
5028h	Coils	SOFTWARE_ERROR			1-бит регистра D381. SOFTWARE_ERROR – внутренняя ошибка контроллера.
5029h	Coils	CMD_ERROR			2-бит регистра D381. CMD_ERROR – невозможность обработки поступившей команды.
502Ah	Coils	DATA_ERROR			3-бит регистра D381. DATA_ERROR – неверный ввод данных в регистры ACC, DEC, U_STEP
502Bh	Coils	OUT_OF_LIM_MIN_SPD_ ERROR			4-бит регистра D381. OUT_OF_LIM_ MIN_SPD_ERROR – заданная скорость меньше минимальной.
502Ch	Coils	OUT_OF_LIM_MAX_SPD_ ERROR			5-бит регистра D381. OUT_OF_LIM_ MAX_SPD_ERROR – превышение максимально возможной скорости.
502Dh	Coils	UNREACHABLE_FS_SPD_ ERROR			6-бит регистра D381. UNREACHABLE _FS_SPD_ERROR – недостижимая скорость перехода на полный шаг в режиме torque boost
502Eh	Coils	NOT_APP_FS_PARAM_ ERROR			7-бит регистра D381. NOT_APP_FS_ PARAM_ERROR – переход из режима torque boost невозможен пока двигатель вращается.

Адрес	Тип объекта	Служебный регистр		Бит	Описание
		Номер	Название		
502F	Coils	OVLO/UVLO_INTERNAL_ PROTECTION_ERROR			Ошибка – напряжение внутренних цепей питания вне штатного диапазона.
5030	Coils	VS_OUT_OF_RANGE_ERROR			Ошибка – напряжение питания вне допустимого диапазона.
5037h	Inputs Registers	D371	MOTOR_STATUS	16	Регистр, отображающий текущее состояние двигателя и системы управления. Описание битов регистра ниже.
5037h	Discrete Inputs	HIZ			0-бит регистра D371. HiZ-состояние двигателя.
5038h	Discrete Inputs	STOP			1-бит регистра D371. Режим удержания.
5039h	Discrete Inputs	ACCELERATING			2-бит регистра D371. Ускорение.
503Ah	Discrete Inputs	DECELERATING			3-бит регистра D371. Торможение.
503Bh	Discrete Inputs	STEADY			4-бит регистра D371. Движение с постоянной скоростью.
503Ch	Discrete Inputs	BUSY_MOVE			5-бит регистра D371. Флаг невозможности применения команд группы MOVE.
503Dh	Discrete Inputs	BUSY_RUN			6-бит регистра D371. Флаг невозможности применения команд группы RUN.
5047h	Inputs Registers	D383	CURRENT_SPD	32	Текущая скорость pps (в качестве события достижения заданной скорости рекомендуется использовать флаг STEADY).
5048h	Holding Registers	D385	EMERGENCY_DEC	32	Аварийное торможение в pps^2.
5100h	Coils	SPIN (APPLY_CMD)			Установка объекта или применение инструкции SPIN активирует выполнение указанной команды в CMD-регистре (D376) с заданными параметрами.
5101h	Coils	TORQUE (APPLY_CURRENT)			Установка объекта или применение инструкции TORQUE применяет значения токов ACC_CUR, DEC_CUR, RUN_CUR, HOLD_CUR для двигателя.
5102h	Coils	HSTOP (HARD_STOP)			Установка объекта или применение инструкции HSTOP немедленно останавливает двигатель с переходом в режим удержания.

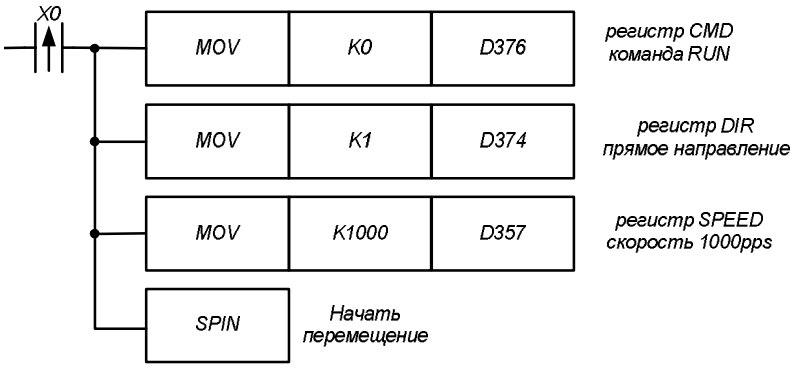
Адрес	Тип объекта	Служебный регистр		Бит	Описание
		Номер	Название		
5103h	Coils	HHIZ (HARD_HIZ)			Установка объекта или применение инструкции HHIZ немедленно переводит двигатель в HiZ-состояние.
5104h	Coils	SSTOP (SLOW_STOP)			Установка объекта или применение инструкции SSTOP останавливает двигатель согласно DEC с переходом в режим удержания.
5105h	Coils	SHIZ (SLOW_HIZ)			Установка объекта или применение инструкции SHIZ останавливает двигатель согласно DEC с переходом в HiZ-состояние.

Команды (CMD-регистр)

Значение	Группа	Название	Описание
0	RUN	RUN	Движение с заданной скоростью SPEED, ускорением ACC, торможением DEC, направлением DIR.
1	MOVE	MOVE	Переместиться на количество шагов TARGET_POS с заданными параметрами движения SPEED, ACC, DEC, DIR.
2	MOVE	GOTO	Переместиться в позицию TARGET_POS с заданными параметрами движения SPEED, ACC, DEC. DIR зависит от текущей позиции, указанное значение игнорируется.
3	MOVE	GOTO_DIR	Переместиться в позицию TARGET_POS с заданными параметрами движения SPEED, ACC, DEC и учётом DIR.
4	MOVE	GOHOME	Переместиться в позицию "0" с заданными параметрами движения SPEED, ACC, DEC. Эквивалентна GOTO "0".
5	RUN	GOUNTIL_SLOWSTOP	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по переднему фронту с первоначальной проверкой уровня входа с последующим торможением и остановкой согласно DEC

6	RUN	GOUNTIL_FRONT_SLOWSTOP	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по переднему фронту с последующим торможением и остановкой согласно DEC.
7	RUN	GOUNTIL_HARDSTOP	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по переднему фронту с первоначальной проверкой уровня входа с последующим переходом в режим удержания.
8	RUN	GOUNTIL_FRONT_HARDSTOP	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по переднему фронту с последующим переходом в режим удержания.
9	RUN	RELEASE	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по заднему фронту с первоначальной проверкой уровня входа с последующим переходом в режим удержания.
10	RUN	FRONT_RELEASE	Движение с заданной скоростью SPEED, ускорением ACC, направлением DIR до срабатывания датчика SW_INPUT по заднему фронту с последующим переходом в режим удержания.

Пример:



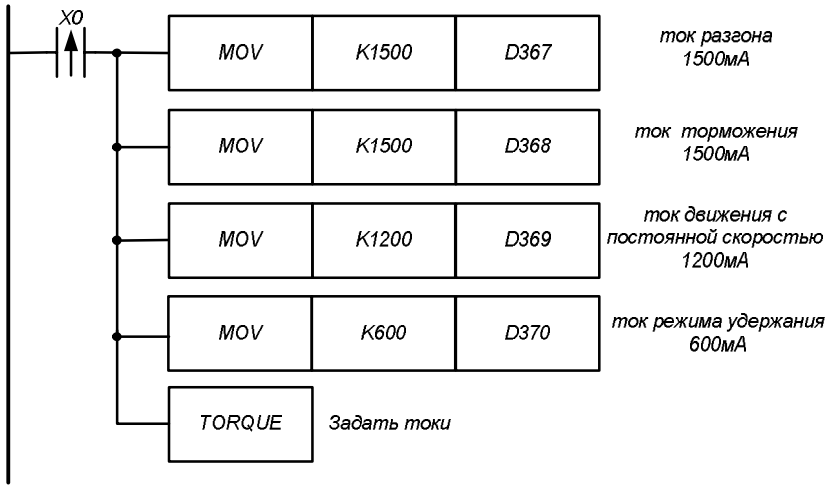
При выполнении входного условия в регистрах настройки драйвера ШД задаётся команда, направление и скорость вращения, после чего следует инструкция для начала перемещения.

TORQUE		P	Применить установленные значение токов к двигателю.
5101h	Coils	Запись единицы в объект (даже если он не сброшен) применяет записанные параметры токов к двигателю, сброс – игнорируется.	

Описание:

Применение данной инструкции устанавливает рабочие токи двигателя, указанные в регистрах ACC_CUR (D367), DEC_CUR (D368), RUN_CUR (D369), HOLD_CUR (D370).

Пример:

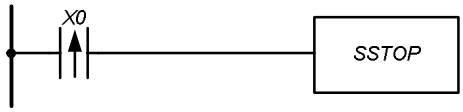


SSTOP		P	Остановка двигателя согласно параметру DEC с последующим переходом в режим удержания.
5104h	Coils	Запись единицы в объект (даже если он не сброшен) применяет торможение двигателя согласно DEC с последующим переходом в режим удержания, сброс – игнорируется.	

Описание:

Применение торможения согласно DEC с переходом в режим удержания. Данная инструкция перекрывает операцию SPIN, имеет одинаковый приоритет с SHIZ, но может быть перекрыта инструкциями HSTOP и HHIZ.

Пример:

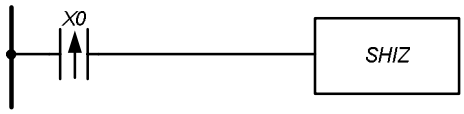


SHIZ		P	Остановка двигателя согласно параметру DEC с последующим переходом в HiZ-режим.
5105h	**Coils**	Запись единицы в объект (даже если он не сброшен) применяет торможение двигателя согласно DEC с последующим переходом в HiZ-режим, сброс – игнорируется.	

Описание:

Применение торможения согласно DEC с переходом в режим удержания. Данная инструкция перекрывает операцию SPIN, имеет одинаковый приоритет с SSTOP, но может быть перекрыта инструкциями HSTOP и HHIZ.

Пример:

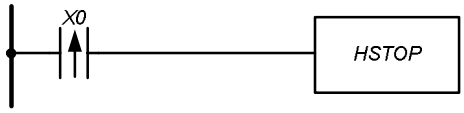


HSTOP		P	Немедленная остановка двигателя с переходом в режим удержания.
5102h	**Coils**	Запись единицы в объект (даже если он не сброшен) применяет немедленную остановку двигателя с переходом в режим удержания, сброс – игнорируется.	

Описание:

Немедленная остановка с переходом в режим удержания. Данная инструкция перекрывает операции SPIN, SSTOP, SHIZ, и имеет одинаковый приоритет с HHIZ.

Пример:

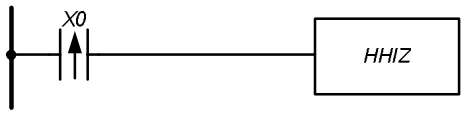


HHIZ		P	Немедленная остановка двигателя с переходом в HiZ-режим.
5103h	**Coils**	Запись единицы в объект (даже если он не сброшен) применяет немедленную остановку двигателя с переходом в HiZ-режим, сброс – игнорируется.	

Описание:

Немедленная остановка с переходом в HiZ-режим (двигатель обесточен, вал свободно вращается). Данная инструкция перекрывает операции SPIN, SSTOP, SHIZ, и имеет одинаковый приоритет с HSTOP.

Пример:



9. Параметры коммуникации

Контроллер имеет USB и RS-485 интерфейс, оба имеют одинаковый доступ к регистрам и битовым операндам. USB интерфейс является виртуальным COM портом (VCP) и преимущественно предназначен для конфигурации контроллера и записи пользовательской программы, поэтому имеет фиксированные параметры связи: Modbus ASCII, ID 1, 115200 бод/с, 7, Even, 1. Для RS-485 вариации параметров указаны в разделе Приложение А. Список регистров и полей для доступа через протокол Modbus в секции «Параметры связи интерфейса RS-485».

9.1. Изменение параметров связи для RS-485

Согласно разделу «Параметры связи интерфейса RS-485» установите требуемые параметры связи. Для вступления изменений в силу перезагрузите устройство. Это можно сделать путём выключения-включения питания или установкой объекта Coils 8101h (Reset).

Пример:

Требуется изменить параметры связи с текущих на следующие: Modbus RTU, ID 100, 128000 бод/с, 8, Odd, 1. В приложении А указаны все допустимые значения регистров и их назначение.

Последовательность действий:

1) Запись в Holding Registers 8103h значения 100d – изменение адреса устройства (ID) на 100.
2) Установка Coils 8100h – выбор протокола RTU.
3) Запись в Holding Registers 8100h значения 128000d – установка скорости 128000бод/с.
4) Запись в Holding Registers 8102h значения 1d – выбор типа проверки чётности Odd.
5) Установка Coils 8101h – выполнить перезагрузку.

9.2. Поддержка протокола Modbus

Настоятельно рекомендуется ознакомится со спецификацией протокола на сайте http://modbus.org/. Поддерживаемые функции протокола представлены в таблице ниже.

			Описание функции	**Код**
Доступ к данным	Discrete Inputs	1-бит	Чтение дискретного входа	02(02h)
	Coils		Чтение группы битов	01(01h)
			Запись одного бита	05(05h)
			Запись группы битов	15(0Fh)
	Inputs Registers	16-бит	Чтение группы регистров	04(04h)
	Holding Registers		Чтение группы регистров	03(03h)
			Запись одного регистра	06(06h)
			Запись группы регистров	16(10h)
			Чтение и запись группы регистров	23(17h)
			Маскированная запись регистра	22(16h)

Коды ошибок протокола представлены в таблице ниже.

Код ошибки	Описание
01(01h)	*ILLEGAL FUNCTION* Принятый код функции не может быть обработан.
02(02h)	*ILLEGAL DATA ADDRESS* Адрес регистра, указанного в запросе, недоступен.
03(03h)	*ILLEGAL DATA VALUE* Значение, содержащееся в поле данных запроса, является недопустимым.
04(04h)	*SERVER DEVICE FAILURE* Невосстанавливаемая ошибка возникла в процессе выполнения затребованного действия.

Коды ошибок, регистрируемые при обработке пакетов протокола, представлены в таблицах ниже.

Адрес	Тип	Размер	Описание
E003h	Input Registers	16-бит	Код ошибки при обработке Modbus кадра.
E003h	Coils		Флаг возникновения ошибки в процессе обмена по протоколу Modbus.

Код ошибки	Описание
0001h	Ошибка выделения памяти.
0002h	Ошибка контрольной суммы.
0003h	Ошибка при приёме и обработке широковещательного пакета.
0004h	Несоответствие размера кадра.
0005h	Ошибка функции (0Fh). Не все биты были перезаписаны.
0006h	Ошибка функции (10h). Не все регистры были перезаписаны.
0007h	Ошибка функции (17h). Не все регистры были перезаписаны.
0008h	Потерян кадр из-за ошибки ПДП.
0009h	Потерян кадр из-за переполненного стека обработки кадров.

Если устройство является конечным в линии связи RS-485, то подключите терминальный резистор, включив тумблер на лицевой панели блока, как показано на рисунке ниже.

Рисунок 29 – Подключение терминального резистора.

10. Установка часов реального времени

Контроллер имеет часы реального времени, которые питаются от внутреннего источника (батареи CR2032), обеспечивающего их работу в отсутствии основного питания, а также работу энергонезависимых регистров и сохранность настроек параметров связи контроллера. Светящийся индикатор **BAT** свидетельствует об отсутствии или скором времени выхода из строя внутреннего элемента питания CR2032. Установка часов реального времени может быть выполнена через программу пользователя инструкцией **TWR** или через протокол Modbus в следующем порядке:

1) Отключить автоматическую перезапись Holding Registers 8110h…8112h путём сброса Coils 8110h.
2) Запись актуального значения времени в Holding Registers 8110h, 8111h, 8112h секунды, минуты, часы соответственно (см. раздел «Настройка часов» в «Приложение А. Список регистров и полей для доступа через протокол Modbus»).
3) Задать новое значение времени путём установки Coils 8111h.
4) Включить автоматическую перезапись Holding Registers 8110h…8112h путём установки Coils 8110h.

11. Загрузка и чтение пользовательской программы

11.1. Процедура загрузки/чтения пользовательской программы

Контроллер имеет две области загрузки программ: общего назначения и специальную.

Область общего назначения предназначена для загрузки программы пользователя ёмкостью 59752 строчки (по умолчанию пуста). Специальная область имеет емкость 1926 строчек и содержит программу управления скоростью шагового двигателя от потенциометра, кнопок, энкодера. При необходимости эту область можно перезаписать.

Рассмотрим запись и чтение пользовательской программы на примере. Список регистров, задействованных в данных операциях приведён в приложении разделе «Приложение А. Список регистров и полей для доступа через протокол Modbus» в секции «Работа с ПЗУ».

Программа на языке LD:

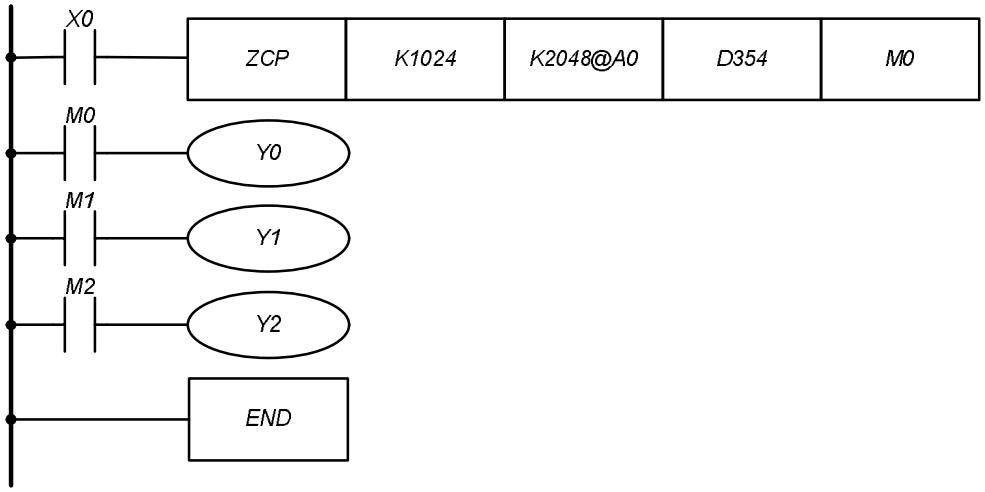


Рисунок 30 – Пользовательская программа.

Преобразованная в IL:

```
LD    X0                          ;входное условие для операции зонного сравнения
ZCP   K1024   K2048@A0   D354   M0   ;зонное сравнение, определение положения ручки
                                  ;потенциометра «SPEED» (2)
LD    M0                          ;если содержимое регистра D354 меньше 1024, то
OUT   Y0                          ;Y0 будет включен, иначе – выключен
LD    M1                          ;если содержимое регистра D354 больше либо
                                  ;равно 1024 и меньше либо равно сумме 2048 и
                                  ;значения A0, то
OUT   Y1                          ;Y1 будет включен, иначе – выключен
LD    M2                          ;если содержимое регистра D354 больше суммы
                                  ;2048 и значения A0, то
OUT   Y2                          ;Y2 будет включен, иначе – выключен
END                               ;конец программы
```

В разделе «

Приложение Б. Список инструкций» приложении Б «Список команд» представлены все коды поддерживаемых контроллером инструкций. Используйте его при ручном вводе программы пользователя в контроллер или воспользуйтесь поставляемым пакетом программ для ПК по программированию контроллера.

1) Убедитесь, что контроллер находится в состоянии STOP, так как в режиме RUN изменение программы пользователя недопустимо. Для этого нужно считать значение Discrete Inputs F001h, в состоянии STOP он сброшен, в RUN – установлен.
2) Следует проверить установлена ли защита от чтения программы. Так как программ в контроллер может быть записано две (пользовательская и служебная), то и объектов защиты от чтения два: Coils F001h и F002h – пользовательский и специальный соответственно. Сброшенный объект свидетельствует об установленной защите от чтения. Прочитать программу в этом случае нельзя.
   Если требуется операция чтения, то перейдите к пункту 5.
3) Если требуется записать новую программу, то необходимо выполнить операцию стирания, для этого установите Coils F003h или F004h для пользовательской и служебной программы соответственно. *В нашем случае запись будет идти на место программы пользователя, поэтому устанавливаем F003h.*
4) Дождитесь сброса Discrete Inputs F000h, это будет свидетельствовать о завершении процедуры стирания и готовности ПЗУ к дальнейшей работе.
5) Выберите тип операции чтение или запись. Для записи установите Coils F005h, для чтения – сбросьте.
6) Далее необходимо выбрать тип программы. Для программы пользователя Coils F006h должен быть сброшен, а для служебной – установлен.
7) Теперь необходимо задать номер строки для записи/чтения программы при помощи Holding Registers F100h. Нумерация начинается с нуля. Для операции чтения перейдите к пункту 10. *В нашем случае запишите 0.*
8) Заполните сектор загрузки F300h…F314 по следующему примеру:

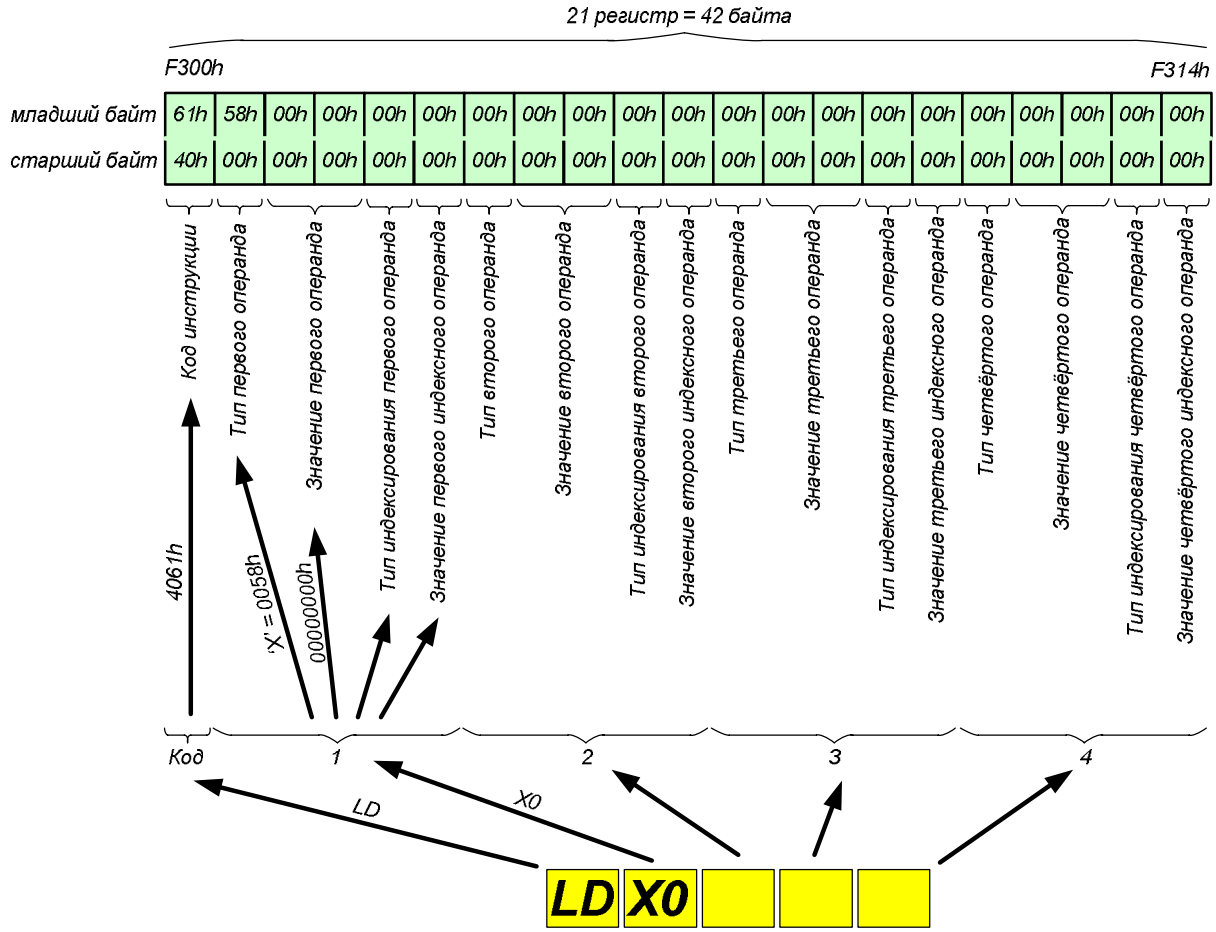


Рисунок 31 – Проецирование инструкции с операндами в адресное пространство протокола Modbus.

9) Установка Coils F000h запустит операцию, параметризированную в Coils F005h и F006h. *В нашем случае запись первой строки в ПЗУ контроллера.* Таким образом повторяя пункты 7 – 9, смещаясь по программе вниз до конца, инкрементируя Holding Registers F100h, происходит запись программы в контроллер.

   В качестве примера рассмотрим формирование сектора загрузки из второй строки программы.

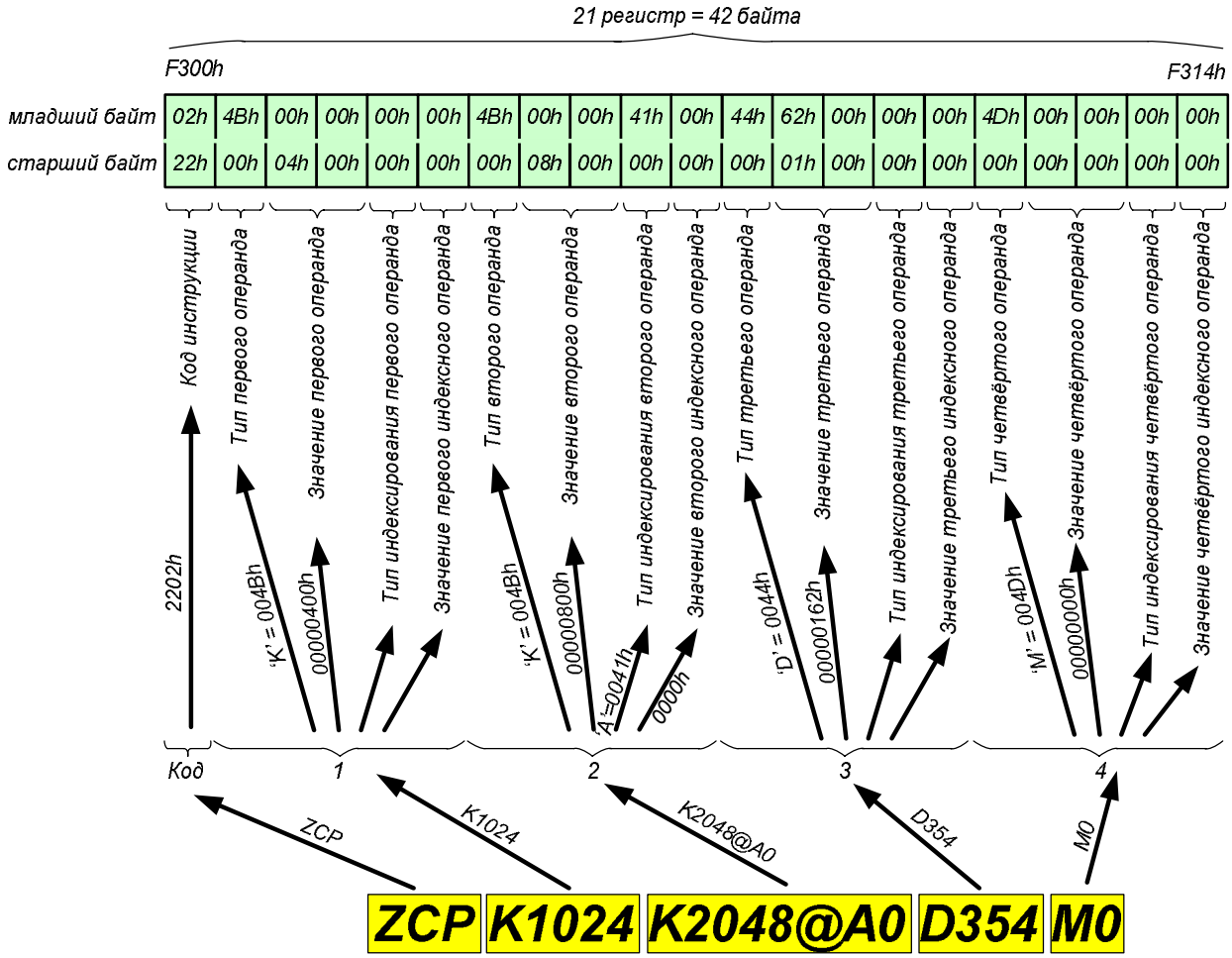


Рисунок 32 – Проецирование инструкции с операндами в адресное пространство протокола Modbus.

10) Для считывания программы выполняется обратная операция записи с той разницей, что сектор выгрузки имеет адрес Inputs Registers F200h…F214h и сперва выполняется параметризированная операция установкой Coils F000h, а затем чтение сектора с последующим инкрементированием номера строки до поступления инструкции **END**.

Ниже представлена таблица кодировки операндов.

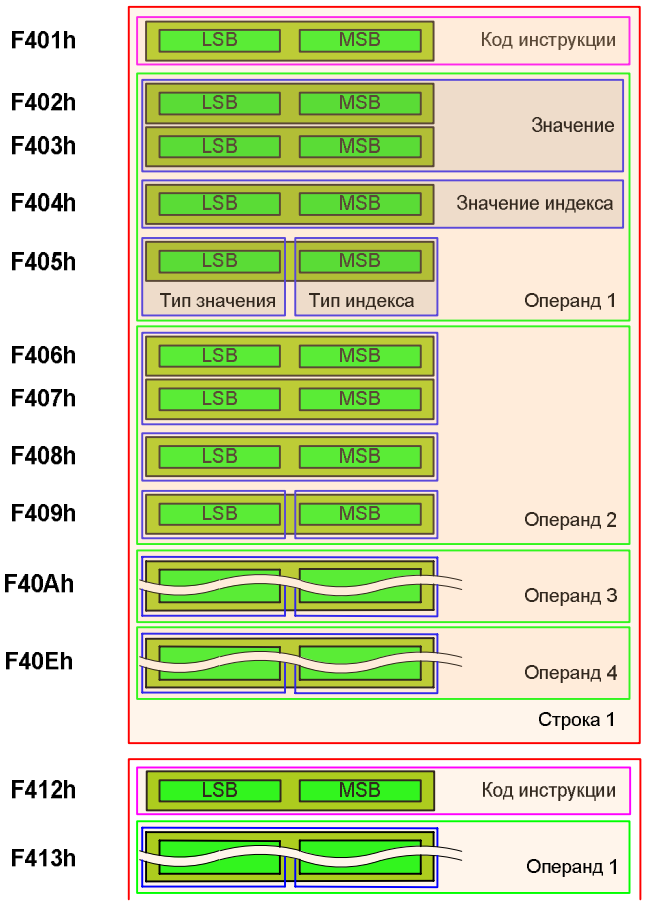
Операнд	Код
K	4Bh
H	48h
F	46h
X	58h
Y	59h
M	4Dh
T	54h
C	43h
A	41h
B	42h
D	44h
P	50h
I	49h

11.2. Блочная загрузка/выгрузка программы пользователя.

Блочные загрузка и выгрузка используются для ускорения процедуры чтения или записи пользовательской программы в память контроллера. При блочной загрузке программы используется алгоритм, аналогичный описанному в разделе «11.1 Процедура загрузки/чтения пользовательской программы», с отличиями:

При записи	
Начало операции	Вместо Coils F000h используется F007h.
Сектор загрузки	Сектор имеет адреса Holding Registers F401h…F4FFh, может содержать от 1 до 15 строк. Количество записываемых строк указывается в Holding Registers F400h.
При чтении	
Начало операции	Вместо Coils F000h используется F007h.
Сектор загрузки	Сектор имеет адреса Holding Registers F401h…F4FFh, может содержать от 1 до 15 строк. Количество считываемых строк указывается в Holding Registers F400h. По завершению в F400h будет указано реальное количество считанных строк.
Строки	Под одну строку в блоке отведено 34 байта.

Структура блока сектора загрузки/выгрузки.



11.3. Коды ошибок, возникающие при работе с ПЗУ

Адрес	Тип	Размер	Описание
E002h	Input Registers	16-бит	Код ошибки при работе с ПЗУ.
E002h	Coils		Флаг возникновения ошибки в процессе работы с ПЗУ.

Код ошибки	Описание
0001h	Не была установлена защита от чтения основной программы.
0002h	Не была установлена защита от чтения специальной программы.
0003h	Не удалось стереть сектор главной программы.
0004h	Не удалось стереть сектор специальной программы.
0005h	Сбой записи инструкции основной программы.
0006h	Сбой записи инструкции специальной программы.

12. Режим управления скоростью вращения

Данный режим предназначен для управления скоростью вращения шагового двигателя при помощи встроенного потенциометра «SPEED» (2), подключаемых кнопок или энкодера.

Рассмотрим каждый из способов регулировки подробнее. Для перехода в режим управления скоростью переведите контроллер в состояние **STOP**, затем кнопкой выбора режима установите **SPD** режим. Соберите и подключите к контроллеру схему, изображённую на рисунке ниже.

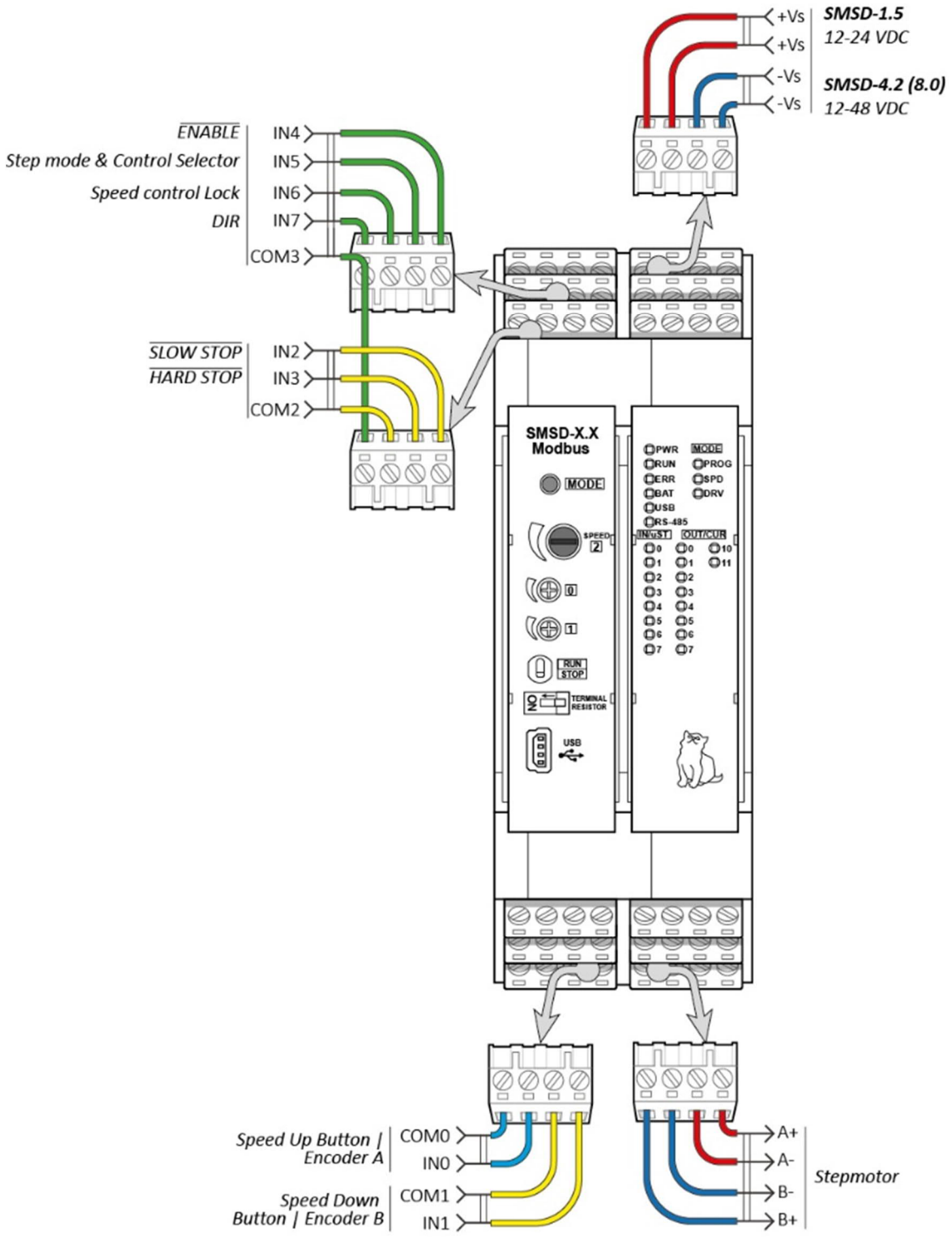


Рисунок 33 – Схема подключения органов управления.

Внимание

Перевод контроллера в режим **RUN** с замкнутыми выключателями SLOW STOP и HARD STOP и разомкнутым ENABLE **приведёт к вращению двигателя.** Во избежание бесконтрольного вращения переведите ручку потенциометра «SPEED» в минимальное положение или измените положение любого из выше указанных выключателей на противоположное указанному в схеме.

В состоянии **RUN** выберите требуемое дробление путём нажатия соответствующей кнопки, так же входом IN5 выбирается источник управления скоростью, подробнее в таблице ниже.

Индикация светодиода OUT0…7	**Дробление**
OUT0	1/1
OUT1	1/2
OUT2	1/4
OUT3	1/8
OUT4	1/16
OUT5	1/32
OUT6	1/128
OUT7	1/256

Индикация светодиода OUT10…11	**Источник управления скоростью**
Оба выключены	Скорость регулируется потенциометром «SPEED».
OUT10	Скорость регулируется кнопками «Увеличить» и «Уменьшить» (IN0 и IN1) шаг изменения задаётся потенциометром «SPEED».
OUT11	Скорость регулируется энкодером, подключенным к IN0 и IN1. Шаг изменения скорости на одно событие энкодера задаётся потенциометром «SPEED».

При включении блокировки изменения скорости контроллер перестаёт реагировать на органы управления скоростью вращения. Данная опция предназначена для предотвращения случайного механического воздействия на потенциометр, энкодер, кнопки.

DIR – изменяет направление вращения двигателя.

ENABLE – управляет наличием напряжения на обмотках шагового двигателя.

$\overline{\text{HARD STOP}}$ – размыкание цепи приведёт к немедленной остановке с переходом в режим удержания. Ток в режиме удержания составляет 50% от рабочего (150…750мА). Рабочий ток устанавливается потенциометром (**1**) от 150 до 1500мА.

$\overline{\text{SLOW STOP}}$ – размыкание цепи приведёт к остановке согласно заданному потенциометром (**0**) торможению (также потенциометром (**0**) задаётся величина ускорения).

Код служебной программы управления скоростью вращения двигателя приведён в разделе «Приложение Г. Программа управления скоростью вращения шагового двигателя». Код может быть модифицирован под определённые требования.

13. Режим драйвера

Режим управления шаговым двигателем при помощи сигналов шаг (Step) и направление (Dir), называется режимом драйвера, входы STEP+, STEP- и DIR+, DIR- соответственно. Вход ENABLE+ управляет наличием управляющего напряжения на обмотках двигателя, а INVERT_ENABLE+ инвертирует сигнал ENABLE. Выход FAULT информирует о превышении тока и перегреве, и пропуске шагов в следствии выше указанных двух причин (см. рис. 2). Для перехода в него переведите контроллер в состояние **STOP** и затем при помощи кнопки выбора режима перейдите в режим **DRV** (драйвер). В данном состоянии двигатель обесточен, и вы можете выбрать дробление полного шага, ток рабочего режима и режима удержания (переход в режим осуществляется по истечении односекундного интервала после детектирования последнего сигнала Step по переднему фронту импульса).

Дробление задаётся потенциометром «SPEED», ток рабочего режима – (**0**), ток режима удержания (**1**). Параметры отображаются на панели светодиодной индикации, подробнее в таблицах ниже.

OUT0	**OUT1**	**OUT2**	**OUT3**	**Ток рабочего режима**		
OUT4	**OUT5**	**OUT6**	**OUT7**	**Ток режима удержания**		
				SMSD-1.5Modbus ver.3	SMSD-4.2Modbus	SMSD-8.0Modbus
●				150 мА	1000 мА	2800 мА
	●			245 мА	1285 мА	3310 мА
●	●			340 мА	1570 мА	3830 мА
		●		440 мА	1860 мА	4340 мА
●		●		535 мА	2140 мА	4860 мА
	●	●		630 мА	2430 мА	5370 мА
●	●	●		730 мА	2710 мА	5885 мА
			●	825 мА	3000 мА	6400 мА
●			●	920 мА	3285 мА	6910 мА
	●		●	1020 мА	3570 мА	7430 мА
●	●		●	1115 мА	3860 мА	7940 мА
		●	●	1210 мА	4140 мА	8460 мА
●		●	●	1310 мА	4430 мА	8970 мА
	●	●	●	1400 мА	4710 мА	9485 мА
●	●	●	●	1500 мА	5000 мА	10000 мА

Индикация светодиода IN0…7	Дробление
IN0	1/1
IN1	1/2
IN2	1/4
IN3	1/8
IN4	1/16
IN5	1/32
IN6	1/128
IN7	1/256

При переводе контроллера в состояние **RUN** выше указанные параметры фиксируются и сохраняются после снятия питания. Используйте входы и выходы контроллера согласно таблице назначения выводов (см. рис. 2).

14. Режим пользовательской программы

Режим работы контроллера, при котором его состояние определяется загруженной программой и управлением по протоколу Modbus, называется режимом пользовательской программы. Нахождение контроллера в данном режиме сопровождается индикацией светодиода **PROG**, и при переходе в состояние **RUN** начинает исполнятся загруженный в режиме **STOP** код пользователя. Так же управлять состоянием контроллера, пользовательской программой, физическими выходами, драйвером шагового двигателя и контролировать состояние физических входов можно через интерфейс RS-485 по протоколу Modbus.

Примеры пользовательских программ, демонстрирующие базовый функционал контроллера, разобраны в разделе «Приложение B. Примеры программ.».

Приложение А. Список регистров и полей для доступа через протокол Modbus

Адрес	Тип	Размер	Описание
Параметры связи интерфейса RS-485			
0x8100	Coils	-	Выбор протокола обмена. Сброшен — Modbus ASCII. Установлен — Modbus RTU. Изменения вступают в силу после перезагрузки.
0x8100	Holding Registers	32-бит	Скорость передачи данных. Допустимые значения: 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, 115200, 128000, 256000. Изменения вступают в силу после перезагрузки.
0x8102	Holding Registers	16-бит	Проверка чётности. 0 – NONE 1 – ODD 2 – EVEN Изменения вступают в силу после перезагрузки.
			*Доступны следующие режимы передачи данных:* *Для режима RTU:* ■ *8-бит / EVEN / 1 STOP* ■ *8-бит / ODD / 1 STOP* ■ *8-бит / NONE / 2 STOP* *Для режима ASCII:* ■ *7бит / EVEN / 1 STOP* ■ *7бит / ODD / 1 STOP* *Параметры стопового бита устанавливаются автоматически.*
0x8103	Holding Registers	16-бит	ID устройства (Адрес устройства). Допустимые значения: 1...247. Изменения вступают в силу после перезагрузки.
0x8104	Holding Registers	16-бит	Количество интервалов принудительной временной задержки отклика, где один интервал составляет 15,26мкс. Диапазон допустимых значений от 0 до 6553. Изменения вступают в силу после перезагрузки.
Настройка часов			
0x8110	Holding Registers	16-бит	Секунды. Допустимые значения: 0...59.
0x8111	Holding Registers	16-бит	Минуты. Допустимые значения: 0...59.
0x8112	Holding Registers	16-бит	Часы. Допустимые значения: 0...23.

Адрес	Тип	Размер	Описание
0x8110	Coils	-	Автоматическое обновление регистров. Установлен — в регистрах 0x8110 - 0x8112 актуальное значение времени. Сброшен — автоматическая актуализация данных отключена, разрешена запись пользовательских значений.
0x8111	Coils	-	Установка объекта задаёт время, записанное в регистры 0x8110 — 0x8112. После чего разрешается вновь включить автоматическое обновление.
Настройка даты			
0x8113	Holding Registers	16-бит	День. Допустимые значения: 1...31.
0x8114	Holding Registers	16-бит	Месяц. Допустимые значения: 1...12.
0x8115	Holding Registers	16-бит	Год. Допустимые значения: 21...99.
			*Установка даты происходит аналогично установке времени с предварительным сбросом Coils 8110h (см. п. 10).*
0x8112	Coils	-	Установка объекта задаёт дату, записанную в регистры 0x8113 — 0x8115. После чего разрешается вновь включить автоматическое обновление.
Дополнительно			
0x8101	Coils	-	Установка объекта вызовет перезагрузку контроллера.
0xF001	Holding Registers	16-бит	Режим работы контроллера: пользовательская программа (PROG), служебная программа (SPD), режим драйвера(DRV). Изменение режима работы контроллера возможно только в состоянии STOP. 0 – Программа пользователя. 1 – Служебная программа. 2 – Режим драйвера.
Работа с ПЗУ			
0xF001	Discrete Inputs	-	Состояние тумблера RUN/STOP. В состоянии RUN работа с ПЗУ невозможна. Сброшен — состояние STOP. Установлен — состояние RUN.
0xF000	Discrete Inputs	-	Отображение статуса ПЗУ. Сброшен — ПЗУ готово к работе. Установлен — ПЗУ занято.

Адрес	Тип	Размер	Описание
0xF000	Coils	-	Управляющий объект для построчной записи/чтения пользовательской программы. Установка запустит операцию, параметризованную в объектах 0xF005 и 0xF006.
0xF001	Coils	-	Защита от чтения основной программы. Установлен — не защищено. Сброшен — установлена защита от чтения. Попытка задать объект — запустит операцию стирания основной программы.
0xF002	Coils	-	Защита от чтения служебной программы. Установлен — не защищено. Сброшен — установлена защита от чтения. Попытка задать объект — запустит операцию стирания служебной программы.
0xF003	Coils	-	Стирание основной программы. Установка объекта запускает процедуру стирания основной программы. Сброс объекта игнорируется.
0xF004	Coils	-	Стирание служебной программы. Установка объекта запускает процедуру стирания служебной программы. Сброс объекта игнорируется.
0xF005	Coils	-	Выбор типа операции. Сброшен — чтение. Установлен — запись.
0xF006	Coils	-	Выбор программы (основная/служебная) Сброшен — основная. Установлен — служебная.
0xF007	Coils	-	Управляющий объект для блочной записи/чтения пользовательской программы (раздел). Установка запустит операцию, параметризованную в объектах 0xF005 и 0xF006.
0xF100	Holding Registers	16-бит	Номер строки в программе, которая будет прочитана или перезаписана (0...59752 — для основной программы, 0...1926 — для служебной).
Сектор построчной выгрузки из ПЗУ			
0xF200	Inputs Registers	16-бит	Код инструкции.
0xF201	Inputs Registers	16-бит	Тип первого операнда
0xF202	Inputs Registers	32-бит	Значение первого операнда
0xF204	Inputs Registers	16-бит	Тип индексирования первого операнда
0xF205	Inputs Registers	16-бит	Значение первого индексного операнда
0xF206	Inputs Registers	16-бит	Тип второго операнда

Адрес	Тип	Размер	Описание
0xF207	Inputs Registers	32-бит	Значение второго операнда
0xF209	Inputs Registers	16-бит	Тип индексирования второго операнда
0xF20A	Inputs Registers	16-бит	Значение второго индексного операнда
0xF20B	Inputs Registers	16-бит	Тип третьего операнда
0xF20C	Inputs Registers	32-бит	Значение третьего операнда
0xF20E	Inputs Registers	16-бит	Тип индексирования третьего операнда
0xF20F	Inputs Registers	16-бит	Значение третьего индексного операнда
0xF210	Inputs Registers	16-бит	Тип четвёртого операнда
0xF211	Inputs Registers	32-бит	Значение четвёртого операнда
0xF213	Inputs Registers	16-бит	Тип индексирования четвёртого операнда
0xF214	Inputs Registers	16-бит	Значение четвёртого индексного операнда
Сектор построчной загрузки из ПЗУ			
0xF300	Holding Registers	16-бит	Код инструкции.
0xF301	Holding Registers	16-бит	Тип первого операнда
0xF302	Holding Registers	32-бит	Значение первого операнда
0xF304	Holding Registers	16-бит	Тип индексирования первого операнда
0xF305	Holding Registers	16-бит	Значение первого индексного операнда
0xF306	Holding Registers	16-бит	Тип второго операнда
0xF307	Holding Registers	32-бит	Значение второго операнда
0xF309	Holding Registers	16-бит	Тип индексирования второго операнда
0xF30A	Holding Registers	16-бит	Значение второго индексного операнда

Адрес	Тип	Размер	Описание
0xF30B	Holding Registers	16-бит	Тип третьего операнда
0xF30C	Holding Registers	32-бит	Значение третьего операнда
0xF30E	Holding Registers	16-бит	Тип индексирования третьего операнда
0xF30F	Holding Registers	16-бит	Значение третьего индексного операнда
0xF310	Holding Registers	16-бит	Тип четвёртого операнда
0xF311	Holding Registers	32-бит	Значение четвёртого операнда
0xF313	Holding Registers	16-бит	Тип индексирования четвёртого операнда
0xF314	Holding Registers	16-бит	Значение четвёртого индексного операнда
Сектор блочной загрузки/выгрузки программы пользователя			
Строка 1 сектора загрузки			
0xF401	Holding Registers	16-бит	Код инструкции строки 1 сектора загрузки
Операнд 1 строки 1 сектора загрузки			
0xF402	Holding Registers	32-бит	Значение операнда 1 строки 1 сектора загрузки
0xF404	Holding Registers	16-бит	Значение индекса операнда 1 строки 1 сектора загрузки
0xF405	Holding Registers	16-бит	8 бит LSB – тип операнда 1 строки 1 сектора загрузки 8 бит MSB – тип индекса операнда 1 строки 1 сектора загрузки
. . .	. . .	. . .	. . .
Операнд 4 строки 1 сектора загрузки			
0xF40E	Holding Registers	32-бит	Значение операнда 4 строки 1 сектора загрузки
0xF404	Holding Registers	16-бит	Значение индекса операнда 4 строки 1 сектора загрузки

Адрес	Тип	Размер	Описание
0xF405	Holding Registers	16-бит	8 бит LSB – тип операнда 4 строки 1 сектора загрузки 8 бит MSB – тип индекса операнда 4 строки 1 сектора загрузки
. . .	. . .	. . .	. . .
Строка 15 сектора загрузки			
0xF4EF	Holding Registers	16-бит	Код инструкции строки 15 сектора загрузки
Операнд 1 строки 15 сектора загрузки			
0xF4F0	Holding Registers	32-бит	Значение операнда 1 строки 15 сектора загрузки
0xF4F2	Holding Registers	16-бит	Значение индекса операнда 1 строки 15 сектора загрузки
0xF4F3	Holding Registers	16-бит	8 бит LSB – тип операнда 1 строки 15 сектора загрузки 8 бит MSB – тип индекса операнда 1 строки 15 сектора загрузки
. . .	. . .	. . .	. . .
Операнд 4 строки 15 сектора загрузки			
0xF4FC	Holding Registers	32-бит	Значение операнда 4 строки 15 сектора загрузки
0xF4FE	Holding Registers	16-бит	Значение индекса операнда 4 строки 15 сектора загрузки
0xF4FF	Holding Registers	16-бит	8 бит LSB – тип операнда 4 строки 15 сектора загрузки 8 бит MSB – тип индекса операнда 4 строки 15 сектора загрузки
Ошибки			
0xE000	Coils	-	Запись в объект сбрасывает все допустимые для текущего состояния контроллера типы ошибок.
0xE000	Discrete Inputs	-	Общая ошибка. Устанавливается всегда, когда появляется хотя бы один из типов ошибок с 0xE001 по 0xE004.
0xE001	Discrete Inputs	-	Установленное состояние объекта свидетельствует о разряженной батарее CR2032 внутри контроллера. Требуется замена.
0xE002	Discrete Inputs	-	Установленное состояние объекта свидетельствует о возникновении ошибки при работе с ПЗУ, код ошибки указан в Inputs Registers 0xE002.
0xE003	Discrete Inputs	-	Установленное состояние объекта свидетельствует о возникновении ошибки в процессе обмена по протоколу Modbus, код ошибки указан в Inputs Registers 0xE003.

Адрес	Тип	Размер	Описание
0xE004	Discrete Inputs	-	Установленное состояние объекта свидетельствует о возникновении ошибки в процессе выполнения пользовательской программы, код ошибки указан в Inputs Registers 0xE004. Строка возникновения в 0xE084.
0xE002	Inputs Registers	16-бит	Код ошибки при работе с ПЗУ.
0xE003	Inputs Registers	16-бит	Код ошибки Modbus протокола.
0xE004	Inputs Registers	16-бит	Код ошибки при выполнении пользовательской программы.
0xE084	Inputs Registers	16-бит	Строка обнаружения ошибки в программе пользователя (нумерация начинается с нуля, см. описание Coils 0xF100 выше).
Доступ к операндам программы			
Дискретные выходы			
0x1000	Discrete Inputs	-	Дискретный выход Y0
0x1001	Discrete Inputs	-	Дискретный выход Y1
. . .	. . .	. . .	. . .
0x107F	Discrete Inputs	-	Дискретный выход Y177
Состояние дискретных физических входов			
0x2000	Discrete Inputs	-	Дискретный вход X0
. . .	. . .	. . .	. . .
0x2007	Discrete Inputs	-	Дискретный вход X7
Дискретные входы			
0x2008	Coils	-	Дискретный вход X10
0x2009	Coils	-	Дискретный вход X11
. . .	. . .	. . .	. . .
0x207F	Coils	-	Дискретный выход X177

Адрес	Тип	Размер	Описание
Регистры общего назначения D192...D255			
0x3000	Inputs Registers	16-бит	Регистр D192
0x3001	Inputs Registers	16-бит	Регистр D193
. . .	. . .	. . .	. . .
0x303F	Inputs Registers	16-бит	Регистр D255
Регистры общего назначения D256...D319			
0x4000	Holding Registers	16-бит	Регистр D256
0x4001	Holding Registers	16-бит	Регистр D257
. . .	. . .	. . .	. . .
0x403F	Holding Registers	16-бит	Регистр D319
Энергонезависимые регистры D320...D327			
0x3100	Inputs Registers	16-бит	Регистр D320
. . .	. . .	. . .	. . .
0x3107	Inputs Registers	16-бит	Регистр D327
Энергонезависимые регистры D328...335			
0x4100	Holding Registers	16-бит	Регистр D328
. . .	. . .	. . .	. . .
0x4107	Holding Registers	16-бит	Регистр D335
Аппаратные счётчики			
0x4200	Holding Registers	32-бит	Счётчик C64
0x4202	Holding Registers	32-бит	Счётчик C65
Аналого-цифровые преобразователи			
0x3200	Inputs Registers	16-бит	Регистр D352, показания с потенциометра «0»

Адрес	Тип	Размер	Описание
0x3201	Inputs Registers	16-бит	Регистр D353, показания с потенциометра «1»
0x3202	Inputs Registers	16-бит	Регистр D354, показания с потенциометра «SPEED»
Версии аппаратной и программной части			
0x8001	Inputs Registers	16-бит	Major версия аппаратной части
0x8002	Inputs Registers	16-бит	Minor версия аппаратной части
0x8003	Inputs Registers	16-бит	Major версия программного обеспечения
0x8004	Inputs Registers	16-бит	Minor версия программного обеспечения
0x8005	Inputs Registers	16-бит	Major версия загрузчика
0x8006	Inputs Registers	16-бит	Minor версия загрузчика
Управление шаговым двигателем			
0x5000	Holding Registers	32-бит	Регистр D357 – SPEED.
0x5002	Holding Registers	32-бит	Регистр D359 – MIN_SPEED.
0x5004	Holding Registers	16-бит	Регистр D361 – ACC.
0x5005	Holding Registers	16-бит	Регистр D362 – DEC.
0x5006	Holding Registers	32-бит	Регистр D363 – ABS.
0x5008	Inputs Registers	16-бит	Регистр D365 – U_POS.
0x5009	Holding Registers	16-бит	Регистр D366 – U_STEP.
0x500A	Holding Registers	16-бит	Регистр D374 – DIR.
0x500A	Coils	-	Регистр D374 – DIR.
0x500B	Holding Registers	32-бит	Регистр D377 – FS_SPD_THR.
0x500D	Holding Registers	16-бит	Регистр D379 – FS_SW_EN.
0x500D	Coils	-	Регистр D379 – FS_SW_EN.

Адрес	Тип	Размер	Описание
0x500E	Holding Registers	32-бит	Регистр D372 – TARGET_POS.
0x5010	Holding Registers	16-бит	Регистр D376 – CMD.
0x5011	Holding Registers	16-бит	Регистр D375 – SW_INPUT.
0x5012	Holding Registers	16-бит	Регистр D367 – ACC_CUR.
0x5013	Holding Registers	16-бит	Регистр D368 – DEC_CUR.
0x5014	Holding Registers	16-бит	Регистр D369 – RUN_CUR.
0x5015	Holding Registers	16-бит	Регистр D370 – HOLD_CUR.
0x5016	Holding Registers	16-бит	Регистр D382 – CMIN_SPD_EN.
0x5017	Holding Registers	16-бит	Регистр D380 – ERROR_SET_HIZ.
0x5017	Coils	-	TERMAL_ERROR_SET_HIZ
0x5018	Coils	-	SOFTWARE_ERROR_SET_HIZ
0x5019	Coils	-	CMD_ERROR_SET_HIZ
0x501A	Coils	-	DATA_ERROR_SET_HIZ
0x5027	Holding Registers	16-бит	Регистр D381 – ERROR_CODE.
0x5027	Coils	-	TERMAL_ERROR_OVER_CURRENT
0x5028	Coils	-	SOFT_ERROR
0x5029	Coils	-	CMD_ERROR
0x502A	Coils	-	DATA_ERROR
0x502F	Coils	-	OVLO/UVLO_INTERNAL_PROTECTION_ERROR (кроме SMSD-1.5Modbus ver.3)
0x5030	Coils	-	VS_OUT_OF_RANGE_ERROR
0x5037	Inputs Registers	16-бит	Регистр D371 – MOTOR_STATUS.
0x5037	Discrete Inputs	-	HIZ
0x5038	Discrete Inputs	-	STOP
0x5039	Discrete Inputs	-	ACCELERATING

Адрес	Тип	Размер	Описание
0x503A	Discrete Inputs	-	DECELERATING
0x503B	Discrete Inputs	-	STEADY
0x503C	Discrete Inputs	-	BUSY_MOVE
0x503D	Discrete Inputs	-	BUSY_RUN
0x5048	Holding Registers	32-бит	Регистр D385 - EMERGENCY_DEC
0x5100	Coils	-	Команда SPIN – APPLY_CMD.
0x5101	Coils	-	Команда TORQUE – APPLY_CURRENT.
0x5102	Coils	-	Команда HSTOP – HARD_STOP.
0x5103	Coils	-	Команда HHIZ – HARD_HIZ.
0x5104	Coils	-	Команда SSTOP – SLOW_STOP.
0x5105	Coils	-	Команда SHIZ – SLOW_HIZ.

Приложение Б. Список инструкций

Команда		Описание
API	Код	
Базовые инструкции		
LD	0x4061	Нормально-открытый контакт
LDI	0x4001	Нормально-закрытый контакт
AND	0x4065	Последовательный нормально-открытый контакт (логическое И)
ANI	0x4005	Последовательный нормально-закрытый контакт (логическое И-НЕ)
OR	0x4066	Параллельный нормально-открытый контакт (логическое ИЛИ)
ORI	0x4046	Параллельный нормально-закрытый контакт (логическое ИЛИ-НЕ)
LDP	0x4821	Начало логического выражения с опросом по переднему фронту (импульс)
LDF	0x4841	Начало логического выражения с опросом по заднему фронту (импульс)
ANDP	0x4825	«И» с опросом по переднему фронту (импульс)
ANDF	0x4845	«И» с опросом по заднему фронту (импульс)
ORP	0x4806	«ИЛИ» с опросом по переднему фронту (импульс)
ORF	0x4826	«ИЛИ» с опросом по заднему фронту (импульс)
TMR	0x2014	Таймер (16 бит)
CNT	0x2015	Счётчик (16 бит)
DCNT	0x3015	Счётчик (32 бит)
INV	0x4016	Инверсия — замена результата логических связей на противоположный
ANB	0x4007	«И» блок — последовательное включение параллельных связей
ORB	0x4008	«ИЛИ» блок — параллельное включение последовательных связей
MPS	0x4009	Смещение вниз по стеку
MRD	0x402A	Считать значение из стека
MPP	0x400A	Выход из стека
SET	0x2024	Включение операнда (установка логической единицы)
RST	0x2004	Сброс состояния операнда

Команда		Описание
API	**Код**	
OUT	0x2002	Выход — присвоение выходу результата логического выражения
FEND	0x6003	Конец главной программы
NOP	0x8011	Пустая строка
P	0x6051	Адресация точки перехода/вызова
I	0x6031	Адресация точки прерывания
END	0x6023	Конец программы
Работа с циклами, переходами, подпрограммами		
CJ	0x200D	Переход к заданной строке
CJP	0x280D	Переход к заданной строке с опросом по переднему фронту (импульс)
CALL	0x200E	Переход к подпрограмме
CALLP	0x280E	Переход к подпрограмме с опросом по переднему фронту (импульс)
SRET	0x600F	Конец подпрограммы
FOR	0x400B	Начало цикла
NEXT	0x400C	Конец цикла
Работа с прерываниями		
IRET	0x6010	Конец обработки прерывания
EI	0x8012	Разрешение прерывания
DI	0x8013	Запрещение прерывания
Пересылка и сравнение		
CMP	0x2201	Сравнение числовых данных
CMPP	0x2A01	Сравнение числовых данных с опросом по переднему фронту (импульс)
DCMP	0x3201	Сравнение числовых данных, 32-битная команда
DCMPP	0x3A01	Сравнение числовых данных, 32-битная команда с опросом по переднему фронту (импульс)
ZCP	0x2202	Зонное сравнение числовых данных
ZCPP	0x2A02	Зонное сравнение числовых данных с опросом по переднему фронту (импульс)

Команда		Описание
API	**Код**	
DZCP	0x3202	Зонное сравнение числовых данных, 32-битная команда
DZCPP	0x3A02	Зонное сравнение числовых данных, 32-битная команда с опросом по переднему фронту (импульс)
MOV	0x2018	Пересылка данных
MOVP	0x2818	Пересылка данных с опросом по переднему фронту (импульс)
DMOV	0x3018	Пересылка данных, 32-битная команда
DMOVP	0x3818	Пересылка данных, 32-битная команда с опросом по переднему фронту (импульс)
BMOV	0x2038	Пересылка блока данных
BMOVP	0x2838	Пересылка блока данных с опросом по переднему фронту (импульс)
DBMOV	0x3038	Пересылка блока данных, 32-битная команда
DBMOVP	0x3838	Пересылка блока данных, 32-битная команда с опросом по переднему фронту (импульс)
FMOV	0x2058	Пересылка в несколько адресов
FMOVP	0x2858	Пересылка в несколько адресов с опросом по переднему фронту (импульс)
DFMOV	0x3058	Пересылка в несколько адресов, 32-битная команда
DFMOVP	0x3858	Пересылка в несколько адресов, 32-битная команда с опросом по переднему фронту (импульс)
XCH	0x220A	Обмен данными
XCHP	0x2A0A	Обмен данными с опросом по переднему фронту (импульс)
DXCH	0x320A	Обмен данными, 32-битная команда
DXCHP	0x3A0A	Обмен данными, 32-битная команда с опросом по переднему фронту (импульс)
Арифметические операции		
ADD	0x2208	Сложение двух чисел
ADDP	0x2A08	Сложение двух чисел с опросом по переднему фронту (импульс)
DADD	0x3208	Сложение двух чисел, 32-битная команда
DADDP	0x3A08	Сложение двух чисел, 32-битная команда с опросом по переднему фронту (импульс)

Команда		Описание
API	Код	
SUB	0x2228	Вычитание двух чисел
SUBP	0x2A28	Вычитание двух чисел с опросом по переднему фронту (импульс)
DSUB	0x3228	Вычитание двух чисел, 32-битная команда
DSUBP	0x3A28	Вычитание двух чисел, 32-битная команда с опросом по переднему фронту (импульс)
MUL	0x2248	Умножение двух чисел
MULP	0x2A48	Умножение двух чисел с опросом по переднему фронту (импульс)
DMUL	0x3248	Умножение двух чисел, 32-битная команда
DMULP	0x3A48	Умножение двух чисел, 32-битная команда с опросом по переднему фронту (импульс)
DIV	0x2268	Деление двух чисел
DIVP	0x2A68	Деление двух чисел с опросом по переднему фронту (импульс)
DDIV	0x3268	Вычисление остатка от деления
DDIVP	0x3A68	Вычисление остатка от деления с опросом по переднему фронту (импульс)
INC	0x2037	Инкрементирование (увеличение на 1)
INCP	0x2837	Инкрементирование (увеличение на 1) с опросом по переднему фронту (импульс)
DINC	0x3037	Инкрементирование (увеличение на 1), 32-битная команда
DINCP	0x3837	Инкрементирование (увеличение на 1), 32-битная команда с опросом по переднему фронту (импульс)
MOD	0x22E8	Вычисление остатка от деления
MODP	0x2AE8	Вычисление остатка от деления с опросом по переднему фронту (импульс)
DMOD	0x32E8	Вычисление остатка от деления, 32-битная команда
DMODP	0x3AE8	Вычисление остатка от деления, 32-битная команда с опросом по переднему фронту (импульс)
DEC	0x2017	Декрементирование (уменьшение на 1)
DECP	0x2817	Декрементирование (уменьшение на 1) с опросом по переднему фронту (импульс)

Команда		Описание
API	Код	
DDEC	0x3017	Декрементирование (уменьшение на 1), 32-битная команда
DDECP	0x3817	Декрементирование (уменьшение на 1), 32-битная команда с опросом по переднему фронту (импульс)
WAND	0x2288	Логическое умножение данных (И)
WANDP	0x2A88	Логическое умножение данных (И) с опросом по переднему фронту (импульс)
DAND	0x3288	Логическое умножение данных (И), 32-битная команда
DANDP	0x3A88	Логическое умножение данных (И), 32-битная команда с опросом по переднему фронту (импульс)
WOR	0x22A8	Логическое сложение данных (ИЛИ)
WORP	0x2AA8	Логическое сложение данных (ИЛИ) с опросом по переднему фронту (импульс)
DOR	0x32A8	Логическое сложение данных (ИЛИ), 32-битная команда
DORP	0x3AA8	Логическое сложение данных (ИЛИ), 32-битная команда с опросом по переднему фронту (импульс)
WXOR	0x22C8	Исключающее «ИЛИ»
WXORP	0x2AC8	Исключающее «ИЛИ» с опросом по переднему фронту (импульс)
DXOR	0x32C8	Исключающее «ИЛИ», 32-битная команда
DXORP	0x3AC8	Исключающее «ИЛИ», 32-битная команда с опросом по переднему фронту (импульс)
NEG	0x2209	Отрицание
NEGP	0x2A09	Отрицание с опросом по переднему фронту (импульс)
DNEG	0x3209	Отрицание, 32-битная команда
DNEGP	0x3A09	Отрицание, 32-битная команда с опросом по переднему фронту (импульс)
ABS	0x2229	Абсолютное значение (модуль)
ABSP	0x2A29	Абсолютное значение (модуль) с опросом по переднему фронту (импульс)
DABS	0x3229	Абсолютное значение (модуль), 32-битная команда
DABSP	0x3A29	Абсолютное значение (модуль), 32-битная команда с опросом по переднему фронту (импульс)

Команда		Описание
API	**Код**	
SQR	0x2215	Вычисление квадратного корня
SQRP	0x2A15	Вычисление квадратного корня с опросом по переднему фронту (импульс)
DSQR	0x3215	Вычисление квадратного корня, 32-битная команда
DSQRP	0x3A15	Вычисление квадратного корня, 32-битная команда с опросом по переднему фронту (импульс)
POW	0x2216	Возведение числа в степень
POWP	0x2A16	Возведение числа в степень с опросом по переднему фронту (импульс)
DPOW	0x3216	Возведение числа в степень, 32-битная команда
DPOWP	0x3A16	Возведение числа в степень, 32-битная команда с опросом по переднему фронту (импульс)
Операции сдвига		
ROR	0x220B	Кольцевой сдвиг вправо
RORP	0x2A0B	Кольцевой сдвиг вправо с опросом по переднему фронту (импульс)
DROR	0x320B	Кольцевой сдвиг вправо, 32-битная команда
DRORP	0x3A0B	Кольцевой сдвиг вправо, 32-битная команда с опросом по переднему фронту (импульс)
ROL	0x222B	Кольцевой сдвиг влево
ROLP	0x2A2B	Кольцевой сдвиг влево с опросом по переднему фронту (импульс)
DROL	0x322B	Кольцевой сдвиг влево, 32-битная команда
DROLP	0x3A2B	Кольцевой сдвиг влево, 32-битная команда с опросом по переднему фронту (импульс)
Операции с данными		
ZRST	0x2203	Групповой сброс операндов в заданном диапазоне
ZRSTP	0x2A03	Групповой сброс операндов в заданном диапазоне с опросом по переднему фронту (импульс)
DECO	0x2211	Дешифратор 8 → 256 бит
DECOP	0x2A11	Дешифратор 8 → 256 бит с опросом по переднему фронту (импульс)
ENCO	0x2212	Шифратор 256 → 8 бит

Команда		Описание
API	**Код**	
ENCOP	0x2A12	Шифратор 256 → 8 бит с опросом по переднему фронту (импульс)
SUM	0x2213	Сумма единичных битов в регистре
SUMP	0x2A13	Сумма единичных битов в регистре с опросом по переднему фронту (импульс)
DSUM	0x3213	Сумма единичных битов в регистре, 32-битная команда
DSUMP	0x3A13	Сумма единичных битов в регистре, 32-битная команда с опросом по переднему фронту (импульс)
BON	0x2214	Опрос состояния бита регистра с установкой выхода
BONP	0x2A14	Опрос состояния бита регистра с установкой выхода с опросом по переднему фронту (импульс)
DBON	0x3214	Опрос состояния бита регистра с установкой выхода, 32-битная команда
DBONP	0x3A14	Опрос состояния бита регистра с установкой выхода, 32-битная команда с опросом по переднему фронту (импульс)
FLT	0x220C	Преобразование целого числа в число с плавающей запятой
FLTP	0x2A0C	Преобразование целого числа в число с плавающей запятой с опросом по переднему фронту (импульс)
DFLT	0x320C	Преобразование целого числа в число с плавающей запятой, 32-битная команда
DFLTP	0x3A0C	Преобразование целого числа в число с плавающей запятой, 32-битная команда с опросом по переднему фронту (импульс)
Операции чисел с плавающей запятой		
DECMP	0x220F	Сравнение двух чисел с плавающей запятой
DECMPP	0x2A0F	Сравнение двух чисел с плавающей запятой с опросом по переднему фронту (импульс)
DEZCP	0x2210	Зонное сравнение двух чисел с плавающей запятой
DEZCPP	0x2A10	Зонное сравнение двух чисел с плавающей запятой с опросом по переднему фронту (импульс)
DEADD	0x2217	Сложение чисел с плавающей запятой
DEADDP	0x2A17	Сложение чисел с плавающей запятой с опросом по переднему фронту (импульс)

Команда		Описание
API	**Код**	
DESUB	0x2237	Вычитание чисел с плавающей запятой
DESUBP	0x2A37	Вычитание чисел с плавающей запятой с опросом по переднему фронту (импульс)
DEMUL	0x2257	Умножение чисел с плавающей запятой
DEMULP	0x2A57	Умножение чисел с плавающей запятой с опросом по переднему фронту (импульс)
DEDIV	0x2277	Деление чисел с плавающей запятой
DEDIVP	0x2A77	Деление чисел с плавающей запятой с опросом по переднему фронту (импульс)
DESQR	0x2218	Вычисление корня квадратного в формате с плавающей запятой
DESQRP	0x2A18	Вычисление корня квадратного в формате с плавающей запятой с опросом по переднему фронту (импульс)
DEPOW	0x2297	Возведение числа в степень в формате с плавающей запятой
DEPOWP	0x2A97	Возведение числа в степень в формате с плавающей запятой с опросом по переднему фронту (импульс)
INT	0x220D	Преобразование числа с плавающей запятой в целое
INTP	0x2A0D	Преобразование числа с плавающей запятой в целое с опросом по переднему фронту (импульс)
DINT	0x320D	Преобразование числа с плавающей запятой в целое, 32-битная команда
DINTP	0x3A0D	Преобразование числа с плавающей запятой в целое, 32-битная команда с опросом по переднему фронту (импульс)
Часы и генератор ШИМ-сигнала		
TRD	0x2219	Чтение текущего значения часов реального времени
TRDP	0x2A19	Чтение текущего значения часов реального времени с опросом по переднему фронту (импульс)
TWR	0x221A	Изменение значения часов реального времени
TWRP	0x2A1A	Изменение значения часов реального времени с опросом по переднему фронту (импульс)
PWM	0x220E	Выдача импульсов с широтно-импульсной модуляцией (ШИМ)

Команда		Описание
API	**Код**	
Дата		
DRD	0x2239	Чтение текущего значения даты
DRDP	0x2A39	Чтение текущего значения даты с опросом по переднему фронту (импульс)
DWR	0x223A	Изменение значения даты
DWRP	0x2A3A	Изменение значения даты с опросом по переднему фронту (импульс)
Логические операции контактного типа		
LD&	0x4204	Контакт замкнут, если S1 & S2 ≠ 0
DLD&	0x5204	Контакт замкнут, если S1 & S2 ≠ 0, 32-битная команда
LD|	0x4224	Контакт замкнут, если S1 | S2 ≠ 0
DLD|	0x5224	Контакт замкнут, если S1 | S2 ≠ 0, 32-битная команда
LD^	0x4244	Контакт замкнут, если S1 ^ S2 ≠ 0
DLD^	0x5244	Контакт замкнут, если S1 ^ S2 ≠ 0, 32-битная команда
AND&	0x4205	Последовательный контакт замкнут, если S1 & S2 ≠ 0
DAND&	0x5205	Последовательный контакт замкнут, если S1 & S2 ≠ 0, 32-битная команда
AND|	0x4225	Последовательный контакт замкнут, если S1 | S2 ≠ 0
DAND|	0x5225	Последовательный контакт замкнут, если S1 | S2 ≠ 0, 32-битная команда
AND^	0x4245	Последовательный контакт замкнут, если S1 ^ S2 ≠ 0
DAND^	0x5245	Последовательный контакт замкнут, если S1 ^ S2 ≠ 0, 32-битная команда
OR&	0x4206	Параллельный контакт замкнут, если S1 & S2 ≠ 0
DOR&	0x5206	Параллельный контакт замкнут, если S1 & S2 ≠ 0, 32-битная команда
OR|	0x4226	Параллельный контакт замкнут, если S1 | S2 ≠ 0
DOR|	0x5226	Параллельный контакт замкнут, если S1 | S2 ≠ 0, 32-битная команда
OR^	0x4246	Параллельный контакт замкнут, если S1 ^ S2 ≠ 0
DOR^	0x5246	Параллельный контакт замкнут, если S1 ^ S2 ≠ 0, 32-битная команда
LD=	0x4264	Контакт замкнут, если S1 = S2

Команда		Описание
API	Код	
DLD=	0x5264	Контакт замкнут, если S1 = S2, 32-битная команда
LD>	0x4284	Контакт замкнут, если S1 > S2
DLD>	0x5284	Контакт замкнут, если S1 > S2, 32-битная команда
LD<	0x42A4	Контакт замкнут, если S1 < S2
DLD<	0x52A4	Контакт замкнут, если S1 < S2, 32-битная команда
LD<>	0x42C4	Контакт замкнут, если S1 ≠ S2
DLD<>	0x52C4	Контакт замкнут, если S1 ≠ S2, 32-битная команда
LD<=	0x42E4	Контакт замкнут, если S1 ≤ S2
DLD<=	0x52E4	Контакт замкнут, если S1 ≤ S2, 32-битная команда
LD>=	0x4304	Контакт замкнут, если S1 ≥ S2
DLD>=	0x5304	Контакт замкнут, если S1 ≥ S2, 32-битная команда
AND=	0x4265	Последовательный контакт замкнут, если S1 = S2
DAND=	0x5265	Последовательный контакт замкнут, если S1 = S2, 32-битная команда
AND>	0x4285	Последовательный контакт замкнут, если S1 > S2
DAND>	0x5285	Последовательный контакт замкнут, если S1 > S2, 32-битная команда
AND<	0x42A5	Последовательный контакт замкнут, если S1 < S2
DAND<	0x52A5	Последовательный контакт замкнут, если S1 < S2, 32-битная команда
AND<>	0x42C5	Последовательный контакт замкнут, если S1 ≠ S2
DAND<>	0x52C5	Последовательный контакт замкнут, если S1 ≠ S2, 32-битная команда
AND<=	0x42E5	Последовательный контакт замкнут, если S1 ≤ S2
DAND<=	0x52E5	Последовательный контакт замкнут, если S1 ≤ S2, 32-битная команда
AND>=	0x4305	Последовательный контакт замкнут, если S1 ≥ S2
DAND>=	0x5305	Последовательный контакт замкнут, если S1 ≥ S2, 32-битная команда
OR=	0x4266	Параллельный контакт замкнут, если S1 = S2
DOR=	0x5266	Параллельный контакт замкнут, если S1 = S2, 32-битная команда
OR>	0x4286	Параллельный контакт замкнут, если S1 > S2
DOR>	0x5286	Параллельный контакт замкнут, если S1 > S2, 32-битная команда
OR<	0x42A6	Параллельный контакт замкнут, если S1 < S2

Команда		Описание
API	**Код**	
DOR<	0x52A6	Параллельный контакт замкнут, если S1 < S2, 32-битная команда
OR<>	0x42C6	Параллельный контакт замкнут, если S1 ≠ S2
DOR<>	0x52C6	Параллельный контакт замкнут, если S1 ≠ S2, 32-битная команда
OR<=	0x42E6	Параллельный контакт замкнут, если S1 ≤ S2
DOR<=	0x52E6	Параллельный контакт замкнут, если S1 ≤ S2, 32-битная команда
OR>=	0x4306	Параллельный контакт замкнут, если S1 ≥ S2
DOR>=	0x5306	Параллельный контакт замкнут, если S1 ≥ S2, 32-битная команда
Управление шаговым двигателем		
SPIN	0x2207	Начать заданное перемещение
SPINP	0x2A07	Начать заданное перемещение с опросом по переднему фронту (импульс)
TORQUE	0x2227	Установить заданные в регистрах токи в обмотках двигателя
TORQUEP	0x2A27	Установить заданные в регистрах токи в обмотках двигателя с опросом по переднему фронту (импульс)
HSTOP	0x2247	Немедленно перейти в режим удержания
HSTOPP	0x2A47	Немедленно перейти в режим удержания с опросом по переднему фронту (импульс)
HHIZ	0x2267	Немедленно обесточить двигатель (вал свободно вращается)
HHIZP	0x2A67	Немедленно обесточить двигатель (вал свободно вращается) с опросом по переднему фронту (импульс)
SSTOP	0x2287	Выполнить остановку с заданным торможением с последующим переходом в режим удержания
SSTOPP	0x2A87	Выполнить остановку с заданным торможением с последующим переходом в режим удержания с опросом по переднему фронту (импульс)
SHIZ	0x22A7	Выполнить остановку с заданным торможением с последующим снятием напряжения с обмоток двигателя (вал свободно вращается)
SHIZP	0x2AA7	Выполнить остановку с заданным торможением с последующим снятием напряжения с обмоток двигателя (вал свободно вращается) с опросом по переднему фронту (импульс)

Приложение В. Примеры программ.

Пример 1. Использование команды RUN

```
LDP     X0                      ;ловим фронт импульса при нажатии на кнопку X0
DMOV    K8      D359            ;устанавливаем минимальную скорость 8pps
DMOV    K120000 D357            ;максимальная скорость 120000pps
FMOV    K30000  D361    K2      ;задаём ускорение и торможение 30000pps^2
MOV     K3      D366            ;дробление 1/8 (см. описание команды SPIN)
MOV     K1      D374            ;прямое направление
DMOV    K6000   D377            ;на скорости 6000 ш/с будет переход на дробление 1/1
MOV     K1      D379            ;разрешаем переход на полный шаг
MOV     K0      D376            ;команда RUN
FMOV    K1500   D367    K2      ;ток разгона и торможения 1500мА
MOV     K1200   D369            ;ток при движении с постоянной скоростью 1200мА
MOV     K600    D370            ;ток режима удержания 600мА
TORQUE                          ;применить заданные значения токов
FMOV    K0      D380    K3      ;без реакции на ошибки, сброс ошибок, использовать
                                ;MIN_SPEED
SPIN                            ;начать перемещение
LDP     X1                      ;ловим фронт импульса при нажатии на кнопку X1
SSTOP                           ;останавливаемся согласно DEC и переходим в
                                ;режим удержания
LDP     X2                      ;ловим фронт импульса при нажатии на кнопку X2
SHIZ                            ;останавливаемся согласно DEC и переходим в
                                ;HiZ состояние
LDP     X3                      ;ловим фронт импульса при нажатии на кнопку X3
HSTOP                           ;немедленно переходим в режим удержания
LDP     X4                      ;ловим фронт импульса при нажатии на кнопку X4
HHIZ                            ;немедленно переходим в состояние HiZ
END                             ;окончание программы
```

Пример 2. Использование команд MOVE, GOTO, GOHOME.

```
LD      M0                      ;для пропуска участка инициализации проверяем
CJ      P1                      ;выполнение условия и будем переходить сразу к P1
LDP     M108                    ;передний фронт у M108 только после инициализации
DMOV    K120000 D357            ;максимальная скорость 120000pps
FMOV    K30000  D361    K2      ;задаём ускорение и торможение 30000pps^2
MOV     K3      D366            ;дробление 1/8 (см. описание команды SPIN)
DMOV    K6000   D377            ;на скорости 6000 ш/с будет переход на дробление 1/1
MOV     K1      D379            ;разрешаем переход на полный шаг
FMOV    K1500   D367    K2      ;ток разгона и торможения 1500мА
MOV     K1200   D369            ;ток при движении с постоянной скоростью 1200мА
MOV     K600    D370            ;ток режима удержания 600мА
TORQUE                          ;применить заданные значения токов
FMOV    K0      D380    K2      ;без реакции на ошибки, сброс ошибок
MOV     K1      D382            ;использовать автоматический расчёт начальной
                                ;и конечной скорости
DMOV    K0      D363            ;обнуляем текущую позицию
SET     M0                      ;включаем условие обхода инициализации драйвера
P       1                       ;метка перехода
LDP     X0                      ;ловим фронт импульса при нажатии на кнопку X0
AND&    D371    K3              ;только когда двигатель в состоянии HiZ или удерж.
```

DMOV	K10000	D372		*;перемещение на 10000 микрошагов*
MOV	K1	D374		*;в прямом направлении*
MOV	K1	D376		*;выполняется командой MOVE*
SPIN				*;начать перемещение*
LDP	X1			*;ловим фронт импульса при нажатии на кнопку X1*
AND&	D371	K3		*;только когда двигатель в состоянии HiZ или удерж.*
DMOV	K100000	D372		*;перемещении к позиции с координатой 100000*
MOV	K2	D376		*;выполняется командой GOTO*
SPIN				*;начать перемещение*
LDP	X2			*;ловим фронт импульса при нажатии на кнопку X2*
AND&	D371	K3		*;только когда двигатель в состоянии HiZ или удерж.*
MOV	K0	D374		*;движение к «0» позиции в обратном направлении*
MOV	K4	D376		*;при помощи команды GOHOME*
SPIN				*;начать перемещение*
LDP	X3			*;ловим фронт импульса при нажатии на кнопку X3*
SSTOP				*;останавливаемся согласно DEC и переходим в ;режим удержания*
LDP	X4			*;ловим фронт импульса при нажатии на кнопку X4*
SHIZ				*;останавливаемся согласно DEC и переходим в ;HiZ состояние*
LDP	X5			*;ловим фронт импульса при нажатии на кнопку X5*
HSTOP				*;немедленно переходим в режим удержания*
LDP	X6			*;ловим фронт импульса при нажатии на кнопку X6*
HHIZ				*;немедленно переходим в состояние HiZ*
END				*;окончание программы*

Пример 3. Использование команд GOUNTIL_SLOWSTOP и RELEASE.

Использование команд GOUNTIL_SLOWSTOP и RELEASE на примере перемещения к началу отсчёта по положительному концевому выключателю (см. рис. 34).

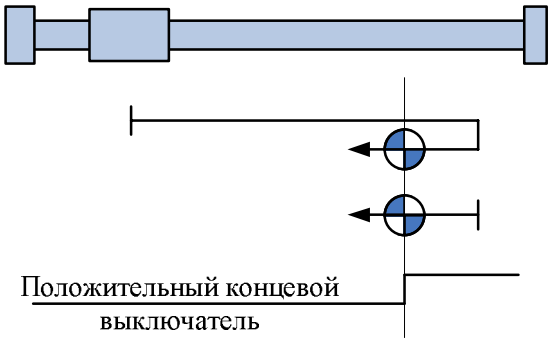


Рисунок 34 – Перемещение к началу отсчёта.

LD	M0			*;для пропуска участка инициализации проверяем*
CJ	P1			*;выполнение условия и будем переходить сразу к P1*
LDP	M108			*;передний фронт у M108 только после инициализации*
FMOV	K30000	D361	K2	*;задаём ускорение и торможение 30000pps^2*
MOV	K3	D366		*;дробление 1/8 (см. описание команды SPIN)*
DMOV	K6000	D377		*;на скорости 6000 ш/с будет переход на дробление 1/1*
MOV	K1	D379		*;разрешаем переход на полный шаг*
FMOV	K1500	D367	K2	*;ток разгона и торможения 1500мА*
MOV	K1200	D369		*;ток при движении с постоянной скоростью 1200мА*
MOV	K600	D370		*;ток режима удержания 600мА*
TORQUE				*;применить заданные значения токов*
FMOV	K0	D380	K2	*;без реакции на ошибки, сброс ошибок*

```
MOV     K1      D382    ;использовать автоматический расчёт начальной
                        ;и конечной скорости
MOV     K7      D375    ;концевой выключатель подключен к IN7
SET     M0              ;включаем условие обхода инициализации драйвера
P       1               ;метка перехода
LDP     X0              ;ловим фронт импульса при нажатии на кнопку X0
AND&    D371    K3      ;только когда двигатель в состоянии HiZ или удерж.
DMOV    K20000  D357    ;максимальная скорость 20000pps
MOV     K5      D376    ;команда GOUNTIL_SLOWSTOP
MOV     K1      D374    ;в положительном направлении (прямом)
SPIN                    ;начать перемещение
SET     M1              ;установим флаг запуска первого этапа
LD      M1              ;ждём когда на первом этапе сработает концевой
AND&    D371    K2      ;выключатель и двигатель остановится
RST     M1              ;сбросим флаг первого этапа
DMOV    K1000   D357    ;понизим скорость
MOV     K9      D376    ;будем двигаться до размыкания концевого
MOV     K0      D374    ;выключателя в обратном направлении
SPIN                    ;начнём перемещение
SET     M2              ;и перейдём ко второму этапу
LD      M2              ;на втором этапе ждём размыкания концевого
AND&    D371    K2      ;выключателя и остановки двигателя
RST     M2              ;сбросим флаг второго этапа
DMOV    K0      D363    ;и обнулим текущую позицию, она стала началом
                        ;отсчёта
LDP     X1              ;ловим фронт импульса при нажатии на кнопку X1
SSTOP                   ;останавливаемся согласно DEC и переходим в
                        ;режим удержания
LDP     X2              ;ловим фронт импульса при нажатии на кнопку X2
SHIZ                    ;останавливаемся согласно DEC и переходим в
                        ;HiZ состояние
LDP     X3              ;ловим фронт импульса при нажатии на кнопку X3
HSTOP                   ;немедленно переходим в режим удержания
LDP     X4              ;ловим фронт импульса при нажатии на кнопку X4
HHIZ                    ;немедленно переходим в состояние HiZ
FEND                    ;окончание главной программы
I       1007            ;обработчик прерывания IN7 (обязателен для команд
                        ;GOUNTIL_... и …RELEASE, может быть пустым)
IRET                    ;возвращаемся обратно в главную программу
END                     ;окончание всей программы
```

Приложение Г. Программа управления скоростью вращения шагового двигателя

LD	M0			*;условие обхода инициализации*
CJ	P1			
LDP	M108			*;блок инициализации*
MPS				
LD=	D320	K6		*;D320 сохраняет значение установленного дробления*
OR>	D320	K8		
OR<	D320	K0		
ANB				
MOV	K0	D320		
MRD				
MOV	D320	D366		
MOV	D320	D0		*;D0 – вспомогательный рег. для визуализации дробления*
MOV	K0	D2		
MRD				
AND>	D0	K6		*;так как в контроллере отсутствует дробление 1/64*
DEC	D0			*;перепрыгиваем через него*
MRD				
DECO	D0	Y0	K3	*;визуализируем дробление на шкале выходов*
MOV	K0	D379		*;первичная настройка драйвера ШД*
MOV	K0	D376		
MOV	K250	D359		
MOV	K0	D380		
MRD				
LD<	D321	K0		*;D321 хранит данные о методе управления:*
OR>	D321	K2		*;потенциометр/кнопки/энкодер*
ANB				
MOV	K0	D321		
MRD				
SET	M0			
MOV	D321	Y10		*;отображение метода управления*
MRD				
AND<>	A0	D321		
MPS				
AND=	D321	K2		*;если выбран **энкодер** то нужно соответствующим*
MOV	K12	D355		*;образом настроить периферию контроллера*
MOV	D321	A0		
RST	M108			*;и перезапустить программу*
MPP				
AND<>	D321	K2		
MOV	K0	D355		
MOV	D321	A0		
RST	M108			
MRD				
MUL	D353	K10	D4	*;показания потенциометра **1***
DIV	D4	K27	D4	
FMOV	D4	D367	K3	*;задаём ток ускорения, торможения, пост. скорости*
DIV	D4	K2	D370	*;ток удержания – 50% от рабочего*
TORQUE				
MRD				

```
AND     X7
MOV     K1      D374
MRD
ANI     X7
MOV     K0      D374
MRD
DLD<    D322    K250    ;D322, D323 скорость задаваемая кнопками
DOR>    D322    K120000
ANB
DMOV    K250    D322
MRD
DMOV    D322    D5
MRD
DLD<    D324    K250    ;D324, D325 скорость задаваемая энкодером
DOR>    D324    K120000
ANB
DMOV    K250    D324
MRD
DMOV    D324    D15
MRD
DMOV    K0      C64
DMOV    C64     D7
MPP
ANI     X4
HSTOP
LD      X2
OUT     M102
EI                      ;включаем прерывания, конец блока инициализации
P       1
LD      X4
AND&    D371    HFE
HHIZ
LDI     X4
MPS
AND     M102
AND     X3
AND&    D371    K3
CALL    P10A0
SPIN
MPP
LDI     X3
ANB
AND&    D371    HFD
HSTOP
LDI     M102
AND     X3
ANI     X4
AND&    D371    H15
SSTOP
LD      M109            ;если имела место быть ошибка и горел индикатор
TMR     T0      K10     ;ERR – запустим таймер для его отключения
AND     T0
RST     M109
```

```
LD<>    A0      K1
CJ      P19
LD=     A0      K1
MPS
ANDP    X0
DADD    D5      D354    D5
MPS
DAND>   D5      K120000
DMOV    K120000 D5
MPP
DMOV    D5      D322
MPP
ANDP    X1
DSUB    D5      D354    D5
MPS
DAND<   D5      K250
DMOV    K250    D5
MPP
DMOV    D5      D322
P       19
LD<>    A0      K2
CJ      P20
LD=     A0      K2
MPS
DSUB    C64     D7      D9
DADD    D7      D9      D7
DCMP    D9      K0      M1
MRD
AND     M1
MOV     D354    D1
DMUL    D9      D1      D11
DADD    D11     D15     D15
MPP
AND     M3
MOV     D354    D1
ABS     D9
DMUL    D9      D1      D11
DADD    D15     D11     D15
LD      M1
OR      M3
MPS
DAND<   D15     K250
DMOV    K250    D15
MRD
DAND>   D15     K120000
DMOV    K120000 D15
MPP
DMOV    D15     D324
P       20
FEND                    ;конец основной программы
P       10              ;подпрограмма установки скорости с потенциометра
LD      M108
MOV     K0      D2
```

```
MOV     D354    D1
MPS
AND=    D1      K0
MOV     K1      D1
MRD
DMUL    D1      K29     D13
MRD
DAND<   D13     K250
DMOV    K250    D13
MPP
DMOV    D13     D357
CALL    P0
SRET
P       11                      ;подпрограмма установки скорости с кнопок
LD      M108
DMOV    D322    D357
CALL    P0
SRET
P       12                      ;подпрограмма установки скорости с энкодера
LD      M108
DMOV    D324    D357
CALL    P0
SRET
P       0                       ;подпрограмма обновления ускорения и торможения
LD      M108
MOV     D352    D3
MPS
AND=    D3      K0
MOV     K1      D3
MPP
MUL     D3      K14     D361
MOV     D361    D362
SRET
I       50                      ;прерывание с периодом 500мс для обновления значений
LD      M108                    ;токов
MUL     D353    K10     D4
DIV     D4      K27     D4
FMOV    D4      D367    K3
DIV     D4      K2      D370
TORQUE
IRET
I       10                      ;прерывание с периодом 100мс для обновления значения
LDI     X6                      ;скорости
AND     X2
AND     M102
AND     X3
ANI     X4
BON     D371    M60     K6
ANI     M60
CALL    P10A0
SPIN
IRET
I       1005                    ;прерывание от входа IN5
```

```
LD      M105
INC     D320
MPS
AND=    D320    K6
INC     D320
MRD
AND=    D320    K9
MOV     K0      D320
INC     D321
MPS
AND=    D321    K3
MOV     K0      D321
MRD
MOV     D321    Y10
MOV     D321    A0
MRD
AND=    D321    K2
MOV     K12     D355
RST     M108
MPP
AND<>   D321    K2
MOV     K0      D355
RST     M108
MRD
MOV     D320    D0
MOV     D320    D366
MRD
AND>    D0      K6
DEC     D0
MPP
DECO    D0      Y0      K3
IRET
I       1004                    ;прерывание от входа IN4
LD      M104
HHIZ
LDI     M104
MPS
AND     X2
AND     X3
AND&    D371    K3
CALL    P10A0
SPIN
MPP
LDI     X2
ORI     X3
ANB
HSTOP
IRET
I       1003                    ;прерывание от входа IN3
LDI     M103
ANI     X4
HSTOP
LD      M103
```

```
AND     X2
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1002                ;прерывание от входа IN5
LDI     M102
AND     X3
ANI     X4
SSTOP
LD      M102
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       1007                ;прерывание от входа IN7
LD&     D371    H1C
MPS
AND     M107
AND=    D374    K0
SSTOP
MPP
ANI     M107
AND=    D374    K1
SSTOP
LD      M107
MOV     K1      D374
AND     X2
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
LDI     M107
MOV     K0      D374
AND     X2
AND     X3
ANI     X4
AND&    D371    K3
CALL    P10A0
SPIN
IRET
I       2000                ;прерывание при возникновении ошибки драйвера
LD&     D381    K1
MOV     K0      D381
SET     M109
MOV     K0      T0
IRET
END
```

Приложение Д. Обновление программного обеспечения контроллера.

Для обновления программного обеспечения скачайте и установите на свой компьютер пакет программ SMSD Controller Demonstrator для операционной системы Windows XP/Vista/7/8/8.1/10. Процесс установки показан на рисунках 35 – 40.

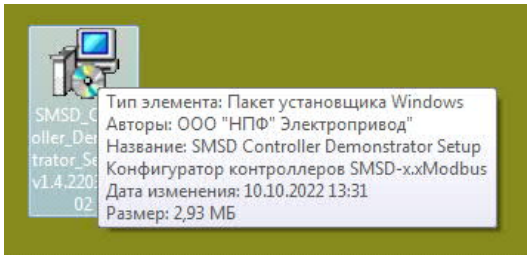


Рисунок 35 – Запустите SMSD Controller Demonstrator Setup Wizard с правами администратора.

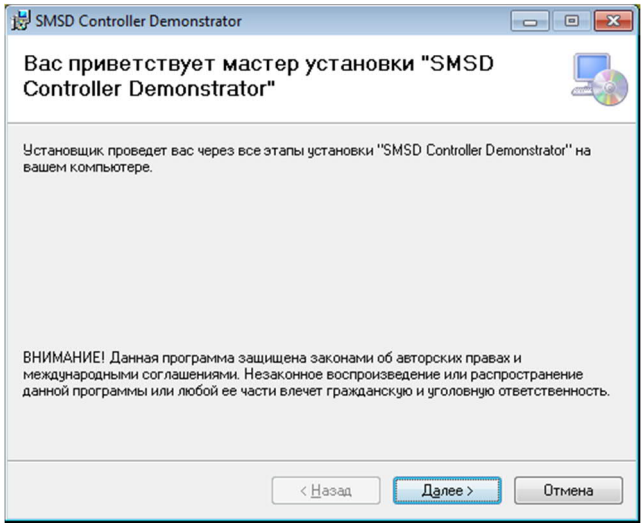


Рисунок 36 – Нажмите Далее.

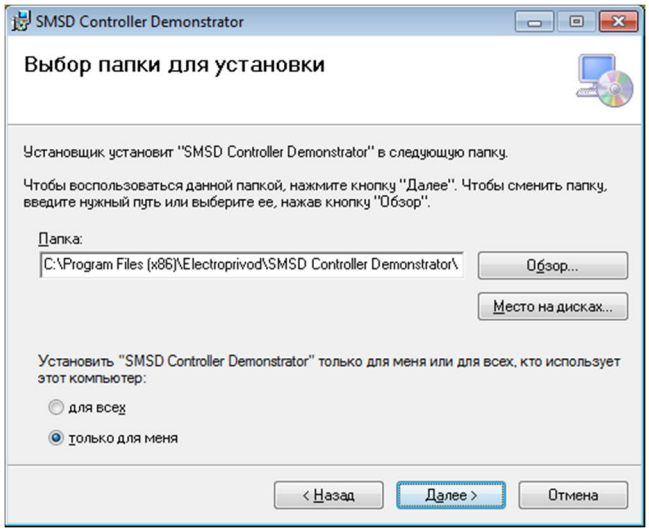


Рисунок 37 – Если требуется, то измените стандартные параметры установки.

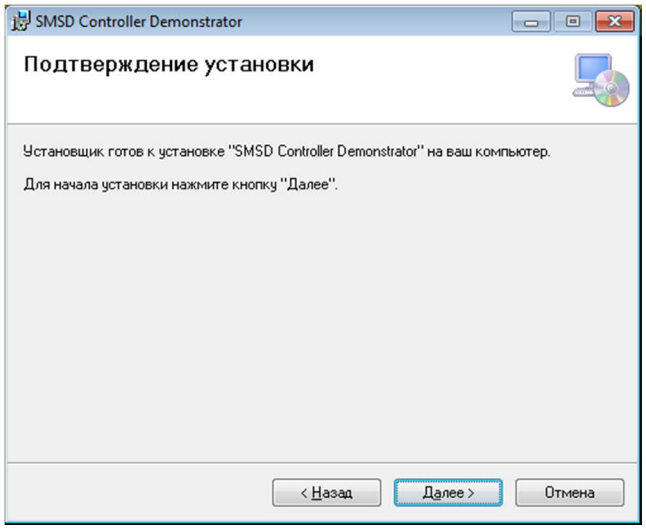


Рисунок 38 – Если требуется, то измените стандартные параметры установки.

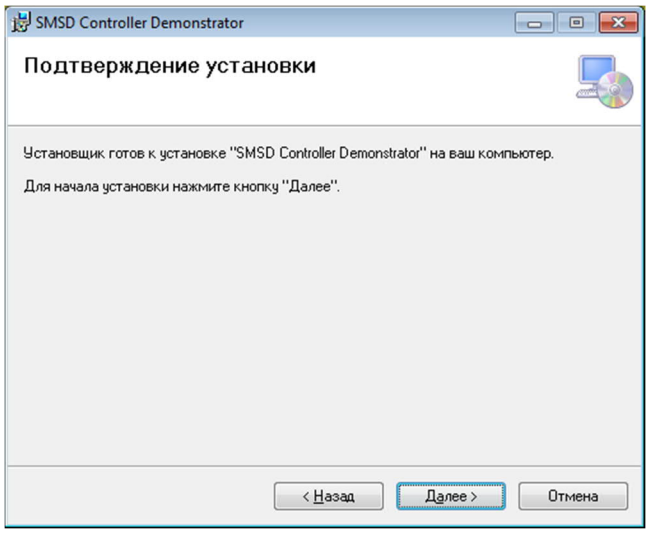


Рисунок 39 – Нажмите Далее.

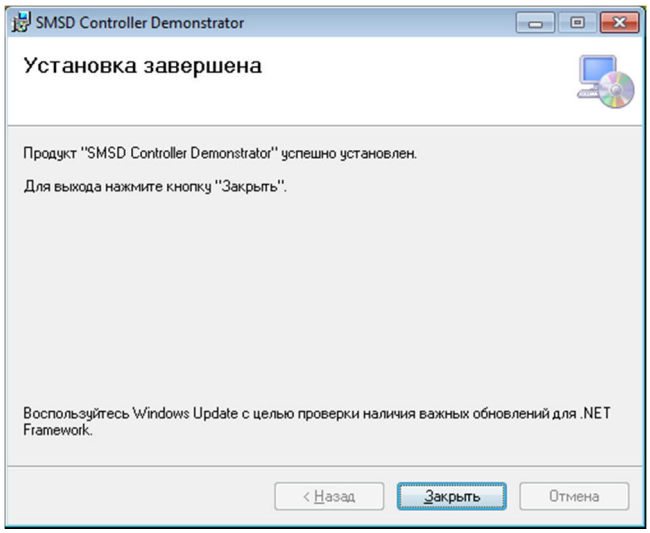


Рисунок 40 – Программное обеспечение установлено.

1) Подсоедините USB – microUSB кабель к блоку и компьютеру, дождитесь установки драйвера виртуального COM-порта. Так же можно скачать драйвер с сайта https://www.ftdichip.com/Drivers/VCP.htm и установить его вручную.
2) Включите питание блока (подробнее см. п. 2).
3) Запустите SMSD Updater (рис. 41). И выберите COM-порт контроллера (рис. 42). Если контроллер подключен по RS-485 интерфейсу – выберите соответствующий тип подключения и для него укажите параметры связи.

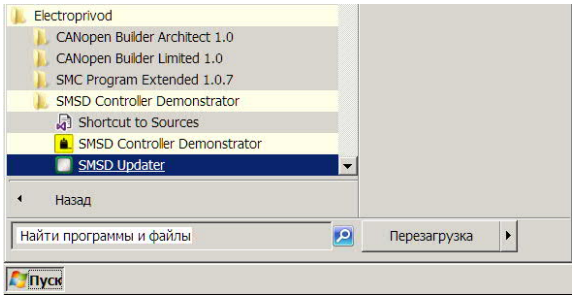


Рисунок 41.

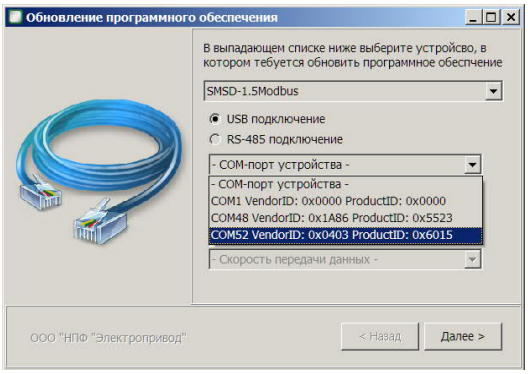


Рисунок 42.

4) Нажмите «Далее» и выберите источник программного обеспечения. Рекомендуется оставить параметры по умолчанию (рис. 43).

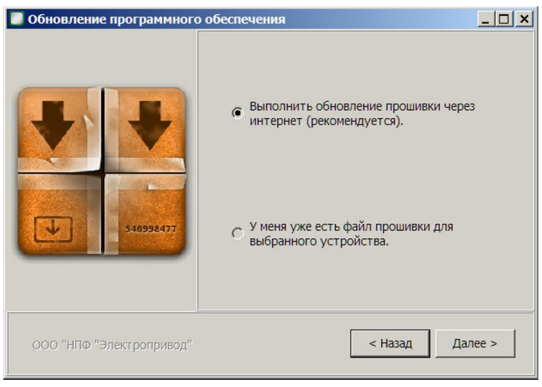


Рисунок 43.

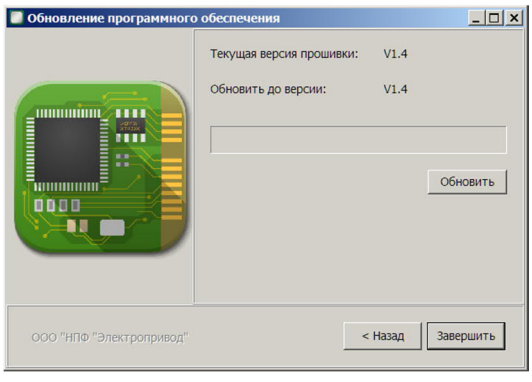


Рисунок 44.

5) Нажмите «Далее», на следующей странице будет представлена текущая версия программного обеспечения контроллера и версия прошивки до которой можно обновить. Обновление не требуется, если версии совпадают (рис. 44). Если версии различаются, то нажмите «Обновить». Внимательно прочитайте все сообщения о требованиях к процессу обновления программного обеспечения. Дождитесь его окончания, о чём будет свидетельствовать сообщение (рис. 45). Теперь SMSD Updater можно закрыть, отключить питание контроллера и отсоединить microUSB кабель. Обновление завершено.

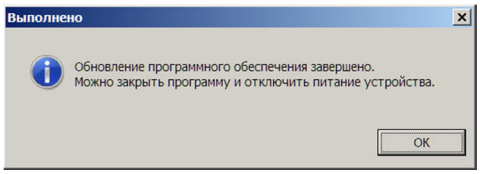


Рисунок 45.

Приложение Е. Время жизни фронтов операндов M и Y.

Данная тема для наглядности будет рассмотрена на двух примерах. Вся нижеизложенная информация справедлива для операндов M и Y, как для передних фронтов, так и для задних.

Пример 1.

Рассмотрим продолжительность жизни переднего фронта операнда M0, при гарантированном проходе точки начала жизни на следующем сканировании.

Строка	Инструкция	Аргументы, порядковый номер		Время жизни фронта M0, скан		Примечание
		1	2	1	2	
1	LDP	M108				Передний фронт M108 присутствует только на 1-ом скане программы.
2	ZRST	D0	D2			Обнуление D0…D2
3	LD	M108				
4	OUT	M0				Начало жизни переднего фронта операнда M0 в 1-ом скане и конец во 2-ом.
5	P	1				
6	LD	M0				
7	CALL	P0				Сохраняется текущее состояние M0 для каждого перехода в подпрограмму P0.
8	LDP	M0				Сработает только один раз, так как выполняется условие повторного прохождения через точку начала жизни фронта операнда M0 в следующем сканировании программы.
9	INC	D0				Результатом работы программы будет D0 = 1.
10	FEND					
11	I	0				Первое прерывание возникает после строки 2 на 1-ом сканировании, в этот момент времени передний фронт у M0 отсутствует. Второе прерывание обрабатывается после строки 6. В обработку прерывания передаётся наличие переднего фронта у M0, поэтому результатом работы будет D2 = 1.
13	LDP	M0				
14	INC	D2				
15	IRET					
16	P	0				
17	LDP	M0				В первом сканировании условие выполнится, во втором – нет.
18	INC	D1				Результатом работы программы будет D1 = 1.
19	SRET					
20	END					

- прерывание
- наличие фронта операнда

Пример 2.

Рассмотрим продолжительность жизни переднего фронта операнда M0, при отсутствии прохождения точки начала жизни на следующем сканировании.

Строка	Инструкция	Аргументы		Время жизни фронта M0, скан			Примечание
		1	2	1	2	3	
1	LDP	M0					
2	CJ	P1					Обход точки начала жизни фронта.
3	LDP	M108					Передний фронт M108 присутствует только на 1-ом скане программы.
4	ZRST	D0	D2				Обнуление D0…D2.
5	LD	M108					
6	OUT	M0					Начало жизни переднего фронта операнда M0 в 1-ом скане. Следующий проход данной точки будет только в 3-ем сканировании.
7	P	1					
8	LDP	M0					Сработает дважды, так как точка начала жизни фронта была пропущена обработчиком команд во втором сканировании, и время жизни было увеличено до поступления команды END/FEND.
9	INC	D0					Результатом работы программы будет D0 = 2.
10	END						Время жизни переднего фронта операнда M0 увеличено до конца сканирования из-за отсутствия прохождения точки начала жизни фронта M0.

- наличие фронта операнда

Если требуется однократный проход между точками 7…9, то, например, можно использовать дополнительное вспомогательное реле M1. Программа будет иметь следующий вид:

```
1   LDP     M0
2   CJ      P1
3   LDP     M108
4   ZRST    D0          D2
5   ZRST    M1                  ;Начальная инициализация M1.
6   LD      M108
7   OUT     M0
8   P       1
9   LDP     M0
10  ANI     M1                  ;Дополнительное условие.
11  INC     D0                  ;Результатом работы программы будет D0 = 1.
12  SET     M1                  ;Блокировка.
13  END
```

Приложение 3. Отладка программы пользователя.

Режим отладки позволяет:

- установить четыре точки прерывания выполнения программы пользователя (breakpoint),
- осуществлять просмотр и редактирование операндов,
- приостанавливать и возобновлять выполнение программы пользователя.

Регистры отладчика:

Адрес	Тип	Размер	Описание
	Управление программой пользователя		
0x6100	Inputs Registers	16-бит	Текущий **индекс** (номер строки) программы пользователя
0x6100	Coils	-	Установка включает режим отладки, сброс – отключает, так же из режима отладки можно выйти путём перевода переключателя «RUN/STOP» в положение STOP.
0x6100	Discrete Inputs	-	Отображение режима отладки. Установленный регистр свидетельствует о работе контроллера в режиме отладки.
0x6101	Coils	-	Управление состоянием программы пользователя. Установка регистра приостановит выполнение программы пользователя на текущем индексе. Сброс – возобновит выполнение.
0x6101	Discrete Inputs	-	Статус выполнения программы пользователя. Установленный регистр свидетельствует о приостановке выполнения, сброшенный - о выполнении.
0x6102	Coils	-	Установка регистра включит пошаговую отладку. При попытке возобновить работу пользовательской программы через сброс 6101h Coils, будет выполнено автоматическое прерывание выполнения на следующем индексе.
	Точки остановки - Breakpoints		
			Помимо пошаговых прерываний выполнения программы пользователя можно задать четыре индекса – точки остановки (breakpoints), на которых выполнение программы будет приостановленно.
	Breakpoint 1		
0x6200	Coils	-	Установка включает прерывание выполнения программы пользователя на индексе, указанном в Holding Registers 6200h.
0x6200	Holding Registers	16-бит	Индекс breakpoint 1.
	Breakpoint 2		
0x6201	Coils	-	Установка включает прерывание выполнения программы пользователя на индексе, указанном в Holding Registers 6201h.
0x6201	Holding Registers	16-бит	Индекс breakpoint 2.
	Breakpoint 3		
0x6202	Coils	-	Установка включает прерывание выполнения программы пользователя на индексе, указанном в Holding Registers 6202h.
0x6202	Holding Registers	16-бит	Индекс breakpoint 3.

Адрес	Тип	Размер	Описание
Breakpoint 4			
0x6203	Coils	-	Установка включает прерывание выполнения программы пользователя на индексе, указанном в Holding Registers 6203h.
0x6203	Holding Registers	16-бит	Индекс breakpoint 4.
Просмотр и редактирование операндов			
0x6000	Coils	-	Запрос на чтение данных путём установки регистра. Сброс происходит автоматически. Отклик на запрос свидетельствует о готовности запрошенных данных об операнде.
0x6001	Coils	-	Запрос на запись данных путём установки регистра. Сброс происходит автоматически. Отклик на запрос свидетельствует о записи данных в операнд
0x6002	Coils	-	Размер регистра для чтения/записи. Сброшен – 16-бит, установлен – 32-бит.
0x6003	Coils	-	Установка регистра отменит редактирование значения операнда при установке Coils 6001h для операндов “C” и “T”.
0x6004	Coils	-	Установка регистра отменит редактирование значения сигнала при установке Coils 6001h для операндов “C” и “T”.
0x6000	Discrete Inputs	-	Устанавливается при ошибке операций чтения или записи операнда. Сброс выполняется автоматически при запросе выполнения операций чтения или записи.
Параметризация операнда			
0x6000	Holding Registers	16-бит	Тип операнда. X (0x58), Y (0x59), M (0x4D), T (0x54), C (0x43), A (0x41), B (0x42), D (0x44).
0x6001	Holding Registers	16-бит	Индекс операнда.
Просмотр операнда			
0x6002	Inputs Registers	32-бит	Данный регистр содержит значение операнда (при наличии), параметризованного для чтения. Размер устанавливается Coils 6002h.
0x6004	Inputs Registers	16-бит	Данный регистр содержит сигнал операнда (при наличии), параметризованного для чтения. 0x00 – низкий уровень, 0x03 – высокий уровень с передним фронтом, 0x02 – высокий уровень, 0x04 – низкий уровень с задним фронтом.
Редактирование операнда			
0x6002	Holding Registers	32-бит	Данный регистр содержит значение операнда (при наличии), параметризованного для записи. Размер устанавливается Coils 6002h.
0x6004	Holding Registers	16-бит	Данный регистр содержит сигнал операнда (при наличии), параметризованного для записи. 0x00 – низкий уровень, 0x03 – высокий уровень с передним фронтом, 0x02 – высокий уровень, 0x04 – низкий уровень с задним фронтом.

Дата редактирования: 31.10.2022