

# Приложение D Программируемый логический контроллер

## D.1 Порядок работы с ПЛК

Ниже описаны основные шаги для работы со встроенным ПЛК.

1. Загрузка программы осуществляется в режиме PLC2:
  - A. С помощью клавиши MODE войдите в режим индикации "PLC0".
  - B. Измените его на режим "PLC2" с помощью клавиши "UP" и затем подтвердите клавишей "ENTER".
  - C. При успешном выполнении на 1...2 сек появится сообщение "END", и снова "PLC2".



### Примечание

Не обращайте внимание на предупреждения, такие как PLod, PLSv и PIdA, перед загрузкой программы в VFD-E.



Выкл.



Работа  
PLC



Режим загрузки  
программы в ПЛК

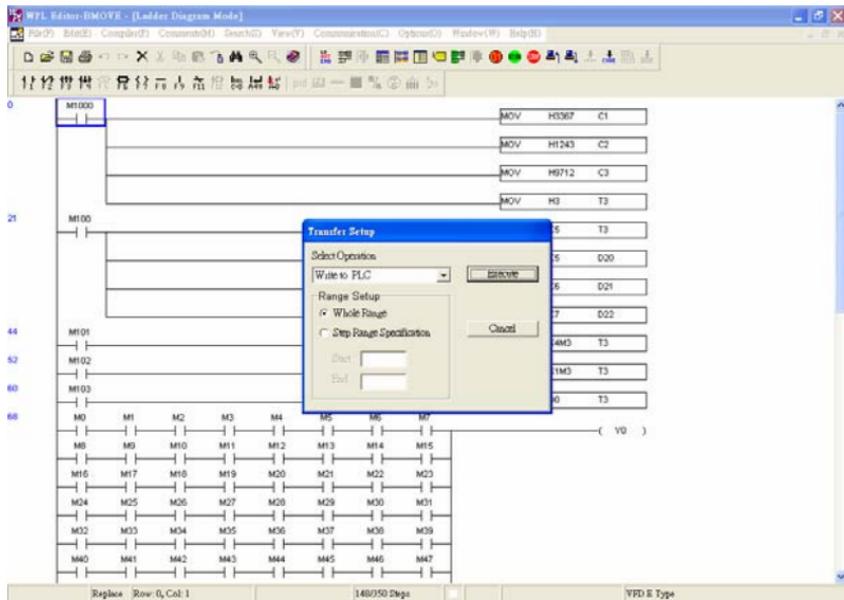
2. Соединение: подключите порт ПЧ (RJ-45) к компьютеру через конвертер RS485/RS232.



RS485



3. Загрузка программы в ПЧ: см. D.2 - D.7 для инструкции по написанию и загрузке программы в пакете программирования WPLSoft V2.09.



4. Работа программы. ПЛК должен быть в состоянии PLC2, даже если ПЧ выключен. Возможны три метода запуска программы ПЛК:
- A. В "PLC1" программа будет выполняться всегда.
  - B. В "PLC2" возможен пуск и остановка выполнения программы с помощью WPL software.
  - C. После установки дискретного входа (MI3 - MI9) = 23 (RUN/STOP PLC), на дисплее будет индикация "PLC1", когда соответствующий вход будет включен. На дисплее будет "PLC0" и программа остановится, когда соответствующий вход будет выключен.



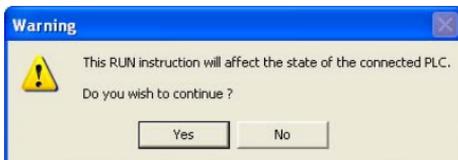
### Примечание

Когда дискретный вход, запрограммированный на функцию 23, будет включен, изменение режима PLC с пульта будет невозможно. Кроме того, когда выбран режим PLC2, вы не сможете запускать/останавливать выполнение программы с внешних терминалов.



## Примечание

При подаче питания на ПЧ, ПЛК будет находиться в состоянии "PLC1".



5. Если модификация программы производится в режиме "PLC2", рекомендуется переключится в режим "PLC1" после завершения изменения и отладки программы.



## Примечание

Когда входы и выходы (MI1–MI9, Relay1–Relay 4, MO1–MO4) используются программой ПЛК, не используйте их для других целей. Например, если Y0 используется для передачи состояния выходу (RA/RB/RC), то в момент его активизации выполнение функции параметра 03.00 будет не возможно, т.к. программа ПЛК имеет более высокий приоритет по использованию входов/выходов ПЧ.



## Примечание

Дискретные входы ПЧ MI1 - MI6 в ПЛК маркируются, как X0 - X5. Когда установлена плата расширения, её ходы будут нумероваться начиная с X06, а выходы с Y2.

## D.2 Основные правила при работе с ПЛК

1. Коммуникационный протокол ПЛК: 7,E,1
2. Всегда останавливайте привод и программу PLC перед началом загрузки новой программы.
3. Приоритет команд WPR и FREQ: FREQ > WPR.
4. Когда Pr 00.04 = 2, на дисплее будет отображаться значение регистра ПЛК D1043.
5. Индикация 0 ~ 999:



6. Индикация 1000 ~ 9999: будут отображаться только 3 старшие разряда. Светодиод справа-внизу от дисплея будет указывать, что число должно быть умножено на 10. Например, при индикации на дисплее числа 100 при светящемся светодиоде, его фактическое значение будет:  $100 \times 10 = 1000$ .



7. Индикация 10000~65535: будут отображаться только 3 старшие разряда. Светодиоды справа-внизу от дисплея и перед последней цифрой будут указывать, что число должно быть умножено на 100. Например, фактическое значение числа на ниже приведенном дисплее будет следующим:  $100 \times 100 = 10000$ .



### D.3 Принципы работы релейно-контактных схем в ПЛК

Язык релейно-контактной логики (лестничных диаграмм) в ПЛК является производной от релейно-контактной принципиальной электро-схемы в упрощенном представлении. Релейно-контактные схемы в ПЛК 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 пересылки, обработки данных и большое количество специальных функций и команд. Можно считать, что ПЛК - это сотни или

тысячи отдельных реле, счетчиков, таймеров и память. Все эти счетчики, таймеры, и т.д. физически не существуют, а моделируются процессором и предназначены для обмена данными между встроенными функциями счетчиками, таймерами.

Язык релейно-контактной логики в ПЛК по используемой символике очень похож на принципиальные релейно-контактные электро-схемы.

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

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

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

В процессе работы ПЛК непрерывно опрашивает текущее состояние входов и в соответствии с требованиями к производственному процессу изменяет состояние выходов(Вкл./Выкл).

Все внутренние объекты ПЛК, 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" имеют однобитный формат, а регистры данных общего назначения обозначаются "D" и имеют 16-ти битный (1 слово) или 32-х битный (2 слова) формат.

При указании операнда определяется, с какой операцией (инструкцией) производится работа.

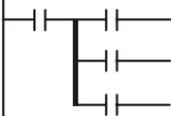
Таблица операндов в ПЛК:

Вход	Входные реле. Определяют состояние внешних битовых устройств, подключенных к входным клеммам ПЛК. Могут принимать одно из двух состояний: 0 или 1. ☞ Адресация ведется в восьмеричной системе: X0, X1, ... X7, X10, X11, ...
Выход	Выходные реле. Определяют состояние выходных клемм ПЛК, к которым подключается нагрузка. 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, ...
Внутреннее реле	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	Реле времени. В программе могут использоваться для хранения текущего значения таймера и иметь 16-ти битный формат, а также могут быть

	<p>контактами, и принимать одно из двух состояний: 0 или 1.</p> <p>☞ Адресация ведется в десятичной системе: T0, T1, ..., T255</p>
Счетчик	<p>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.</p> <p>☞ Адресация ведется в десятичной системе: C0, C1, ..., C255</p>
Регистр данных	<p>Память данных. 16-ти или 32-х битный формат.</p> <p>☞ Адресация ведется в десятичной системе: D0, D1, ..., D9999. В 32-х битном формате один регистр занимает две ячейки, например при обращении к D10, данные будут прочитаны из ячеек D10 и D11.</p>

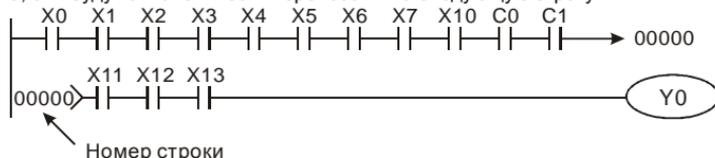
В релейно-контактных схемах в основном применяется следующая символика:

Символ	Пояснение	Команда	Операнд
	Входной нормально-открытый контакт a	LD	X, Y, M, T, C
	Входной нормально-закрытый контакт b	LDI	X, Y, M, T, C
	Последовательный нормально-открытый контакт	AND	X, Y, M, T, C
	Параллельный нормально-открытый контакт	OR	X, Y, M, T, C
	Параллельный нормально-закрытый контакт	ORI	X, Y, M, T, C
	Входной импульсный сигнал с опросом по переднему фронту	LDP	X, Y, M, T, C
	Входной импульсный сигнал с опросом по заднему фронту	LDF	X, Y, M, T, C
	Последовательный импульсный сигнал с опросом по переднему фронту	ANDP	X, Y, M, T, C
	Последовательный импульсный сигнал с опросом по заднему фронту	ANDF	X, Y, M, T, C
	Параллельный импульсный сигнал с опросом по переднему фронту	ORP	X, Y, M, T, C
	Параллельный импульсный сигнал с опросом по заднему фронту	ORF	X, Y, M, T, C

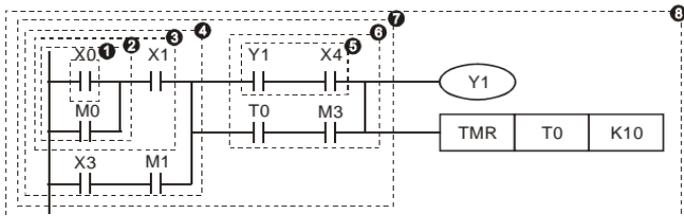
Символ	Пояснение	Команда	Операнд
	Последовательный блок	ANB	нет
	Параллельный блок	ORB	нет
	Разветвление выходов	MPS MRD MPP	нет
	Выходной сигнал (катушка)	OUT	Y, M, S
	Базовая или прикладная инструкция	Прикладная инструкция	См. описание базовых и прикладных инструкций
	Логическая инверсия	INV	none

#### D.4 Правила изображения релейно-контактных схем в ПЛК

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. Командные линии могут разветвляться и снова соединяться. Максимальное количество последовательных контактов в строке – 11. При необходимости использования большего количества, они будут автоматически перенесены на следующую строку:



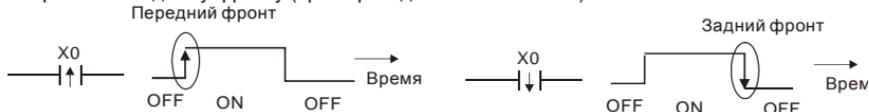
Сканирование программы начинается от левого верхнего угла схемы и заканчивается в правом нижнем углу. Следующий пример иллюстрирует последовательность выполнения программы:



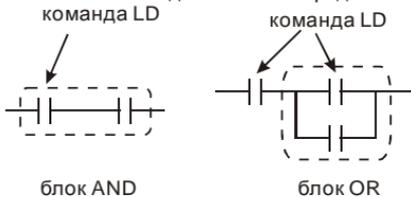
Список инструкций:

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:

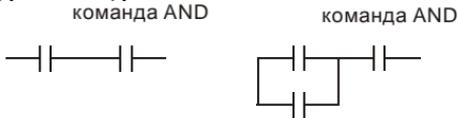


1. Команда LD (LDI): открывает логическую связь. В контактных схемах команда всегда расположена слева и соединяется непосредственно с шиной питания.

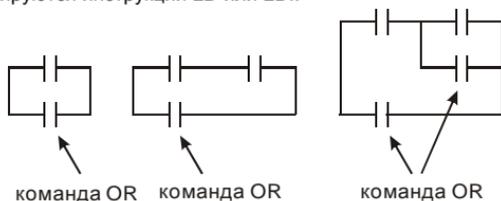


2. Команда AND (ANI): используется в качестве последовательного нормально-открытого (закрытого) контакта для программирования операции логического умножения (И). Команда представляет логическую операцию и поэтому не может

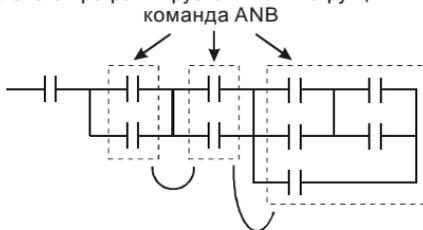
программироваться в начале цепи. В начале логического выражения программируются инструкции LD или LDI.



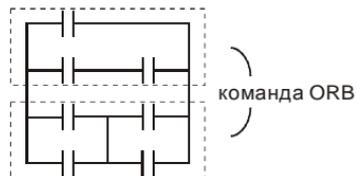
3. Команда OR (ORI): используется в качестве параллельного нормально-открытого (закрытого) контакта для программирования операции логического сложения (ИЛИ). Команда представляет логическую операцию и поэтому не может программироваться в начале цепи. В начале логического выражения программируются инструкции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



4. Команда ANB используется для последовательного соединения цепочек из двух параллельных контактов. Отдельные блоки, параллельно включенных элементов, заносятся в программу отдельно. Чтобы эти блоки соединить последовательно, после каждого блока программируется ANB инструкция.

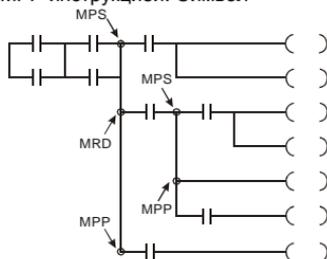


5. Команда ORB используется для параллельного соединения цепочек из двух последовательных контактов. Если несколько последовательных блоков включаются параллельно, то нужно после программирования каждого отдельного блока вводить ORB-инструкцию.



6. 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 выходо-катушек).
7. С помощью инструкции MPS запоминается предыдущий результат логических связей (обработки логического выражения). Символ "└".
8. 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. Символ "┆".
9. Последнее частное разветвление создается MPP инструкцией. Символ "└"

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



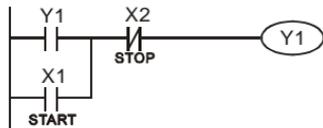
## D.5 Примеры написания программ

### ■ Старт, стоп и самоблокировка

Часто бывает необходимо использовать для старта и стопа кнопки без фиксации, но с самоблокировкой выхода. Примеры реализации таких схем представлены ниже:

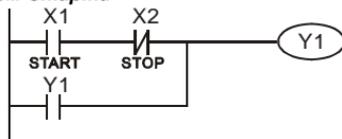
#### Пример 1: самоблокировка выхода с приоритетом Стопа

Когда X1=вкл., а X2=выкл., то выход Y1=вкл. до тех пор, пока X2 не разомкнется.



#### Пример 2: самоблокировка выхода с приоритетом Старта

Когда X1=вкл., а X2=выкл., то выход Y1=вкл. Если X2 разомкнется выход Y1 все равно останется включенным.

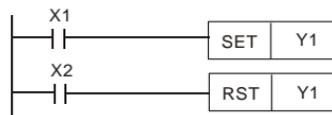


#### Пример 3: самоблокировка выхода с использованием команд SET и RESET

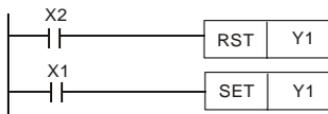
ПЛК выполняет программу сверху вниз, и следовательно приоритетом будет обладать команда расположенная ниже.

Если одновременно замкнуты оба контакта X1 и X2, то в верхней схеме выход Y1=0, а в нижней - Y1=1.

Приоритет стопа

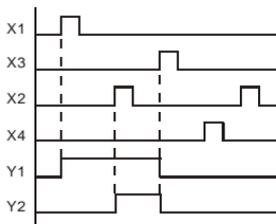
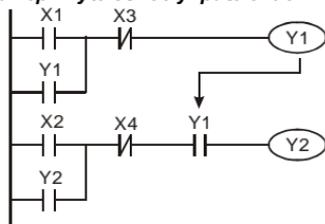


Приоритет старта



■ Схемы общего применения

**Пример 4: условное управление**



Контакты X1 и X3 включают/выключают выход Y1 автономно, а X2 и X4 могут управлять состоянием выхода Y1 только при условии, что Y1 включен, т.е. выход Y1 является последовательным контактом (логическим И) для нижней схемы.

**Пример 5: схема с взаимоблокировкой**

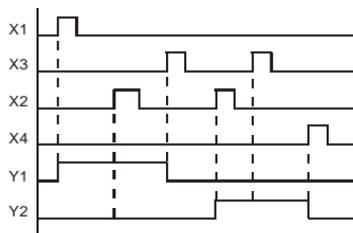
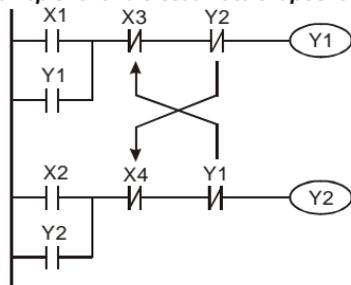
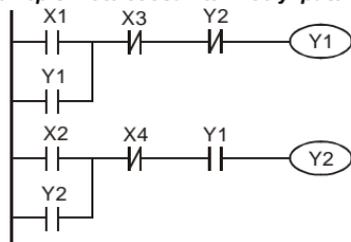


Схема исключает одновременное включение двух выходов. Когда включен один выход, второй будет заблокирован. При одновременном замыкании контактов X1 и X2 приоритет будет иметь Y1.

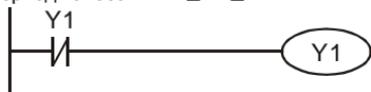
**Пример 6: последовательное управление**



Выход Y2 может быть включен, только если включен Y1, однако при включении Y2 выход Y1 будет отключен.

### Пример 7: колебательные схемы

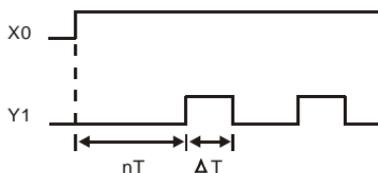
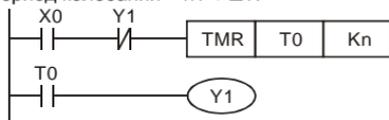
Период колебаний =  $\Delta T + \Delta T$



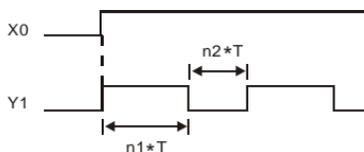
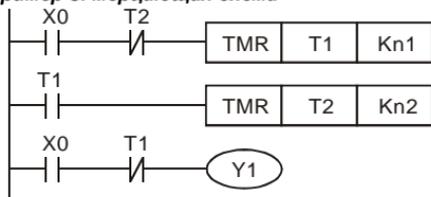
$\Delta T$  – время сканирования (время выполнения одного цикла программы)

В первом цикле сканирования выход Y1 будет включен, а во втором – выключен, и т.д.

Период колебаний =  $nT + \Delta T$ :

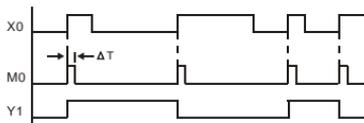
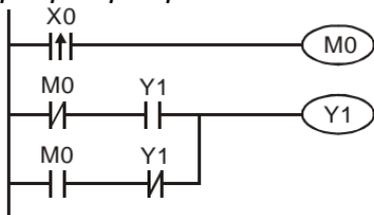


### Пример 8: мерцающая схема



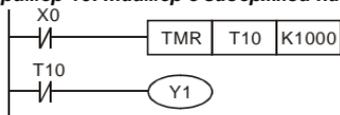
Используется для мигающей сигнализации с помощью лампы или динамика.

### Пример 9: триггерная схема

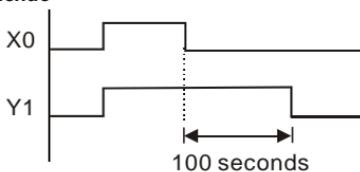


Каждое замыкание контакта X0 изменяет состояние выхода Y1 на противоположное. Эта схема еще называется импульсным реле.

**Пример 10: таймер с задержкой на выключение**

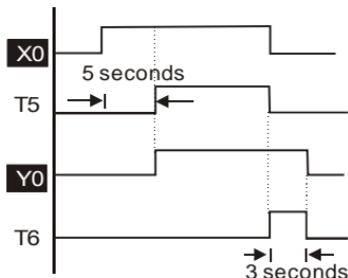
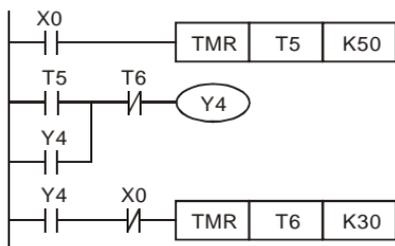


TB = 0.1 sec

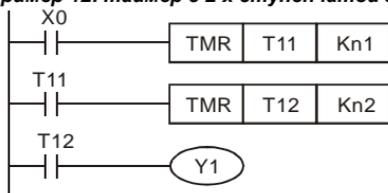


Когда X0 = 1, выход Y1 включен. При выключении X0, выход Y1 выключится через 100 секунд, т.к. операнд заданного значения таймера T0 имеет дискретность 0.1 сек, т.е. K1000 = 100 сек.

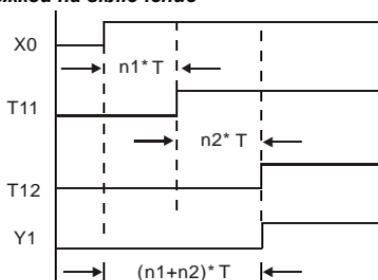
**Пример 11: таймер с задержкой на включение и выключение**



**Пример 12: таймер с 2-х ступенчатой задержкой на включение**



Выход Y1 будет включен через время  $n1+n2$  после замыкания контакта X0.



## D.6 Операнды в PLC

### D.6.1 Обзор операндов и основные характеристики ПЛК

Элемент		Описание		Комментарий				
Метод выполнения программы		Циклическое сканирование						
Метод обработки вх/вых		Групповое обновление (после инструкции END) или по команде обновления I/O						
Время выполнения инструкций		Базовые команды (минимум 0.24 мкс)		Прикладные команды (10 ~ 100 мкс)				
Языки программирования		LAD (релейно-контактные схемы), IL (список инструкций), SFC						
Объем памяти программы		350 шагов		SRAM + батарея				
Набор инструкций		45 команд		28 базовых команд 17 прикладных инструкций				
Входы/выходы		Входы (X): 6, выходы (Y): 2						
Реле (1-но битная память)	X	Внешние входные реле		X0~X17, 16 точек	Макс. 32 точек	Входы ПЛК		
	Y	Внешние выходные реле		Y0~Y17, 16 точек		Выходы ПЛК		
	M	Внутренние реле	Общие		M0~M159, 160 точек	Макс. 192 точек	Промежуточная двоичная память. Соответствуют промежуточным реле в электросхемах	
			Специальные		M1000~M1031, 32 точек			
	T	Таймеры	Дискретность 100 мс		T0~T15, 16 точек	Макс. 16 точек	Используются в качестве контактов (T), которые замыкаются при достижении соотв. таймером (команда TMR) своего заданного значения	
	C	Счетчики	Инкрементный (16 бит)		C0~C7, 8 точек	Макс. 8 точек	Используются в качестве контактов (C), которые замыкаются при достижении соотв. счетчиком (команда CNT) своего заданного значения	
			Инкр./декрем. 32 бит быстросрабатывающих	1фаза, 1вход		C235, 1 точка (требуется использование платы расширения PG)		Макс. 1 точка
				1фаза, 2входа				
2фазы, 2входа								

Элемент		Описание		Комментарий
Регистр (16-бит память)	T	Текущее значение таймера	T0~T15, 16 точек	Регистры для хранения текущих значений таймеров
	C	Текущее значение счетчика	C0~C8, 8-бит счетчик, 8 точек C235, 32-бит, 1 точка	Регистры для хранения текущих значений счетчиков
	D	Регистры данных	Общие	D0~D29, 30 точек
Специальные			D1000~D1044, 45 точек	
Константа	K	Десятичные константы	K-32768 ~ K32767 (16-битные операции)	
	H	Шестнадцатеричные константы	H0000 ~ HFFFF (16-битные операции)	
Коммуникационный порт (для чтения/записи программы)			RS485 (slave)	
Аналоговые входы/выходы			2 встроенных аналоговых входа и 1 аналог. выход	
Модули расширения (опции)			Платы расширения дискретных входов/выходов (A/D, D/A)	

## D.6.2 Таблицы адресации входов и выходов

Операнд	X														
	0	1	2	3	4	5	6	7	10	11	12	13	14	15	
Терминал ПЧ	MI1	MI2	MI3	MI4	MI5	MI6	--	--	--	--	--	--	--	--	
Плата реле-2C (EME-DR2CA)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
Плата реле -3A (EME-R3AA)	--	--	--	--	--	--	--	--	--	--	--	--	--	--	
Плата 3IN/3OUT(EME-D33A)	--	--	--	--	--	--	MI7	MI8	MI9	--	--	--	--	--	

Операнд	Y														
	0	1	2	3	4	5	6	7	10	11	12	13	14	15	
Терминал ПЧ	RY	MO1	--	--	--	--	--	--	--	--	--	--	--	--	
Плата реле -2C (EME-DR2CA)	--	--	RY2	RY3	--	--	--	--	--	--	--	--	--	--	
Плата реле -3A (EME-R3AA)	--	--	RY2	RY3	RY4	--	--	--	--	--	--	--	--	--	
Плата 3IN/3OUT (EME-D33A)	--	--	MO2	MO3	MO4	--	--	--	--	--	--	--	--	--	

### D.6.3 Назначение входов/выходов

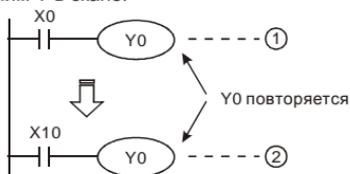
#### ■ Назначение входных реле X

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

Изменять состояние входных контактов X нельзя с помощью WPLSoft.

#### ■ Назначение выходных реле Y

Выходные реле Y управляют состоянием физических выходных контактов ПЛК (релейных или транзисторных), а следовательно и устройствами нагрузки (лампы, ТЭНы, катушки реле и др.) непосредственно подключенными к выходным клеммам ПЛК. 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 выхода Y0 будет определяться только контактом X10.

### D.6.4 Форматы чисел, константы [K] и [H]

Константа	K	Десятичная	K-32768 ~ K32767 (16-битные операции) K-2147483648 ~ K2147483647 (32-битные операции)
	H	Шестнадцатеричная	H0 ~ HFFFF (16-битные операции) H0 ~ HFFFFFFFF (32-битные операции)

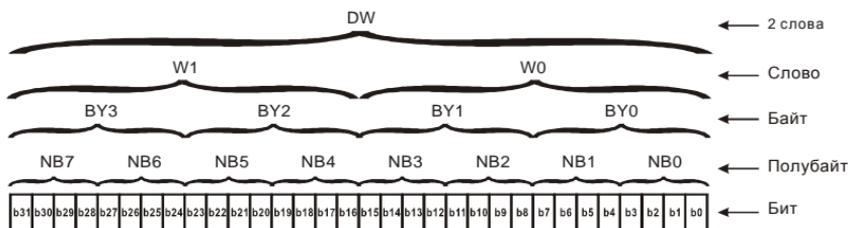
С помощью десятичных (K) и шестнадцатеричных (H) констант можно задавать числовые значения внутри программы ПЛК (например, задаваемые значения уставок времени или счета). Числовые значения кодируются внутри ПЛК в двоичном счислении.

Далее рассмотрены системы счисления используемые в DVP-PLC:

## 1. Двоичный формат чисел (BIN)

Используется для внутренних вычислений ПЛК и памяти и имеет следующее представление:

- Бит : Это основная единица измерения двоичной системы, может иметь два состояния 0 или 1
- Полубайт : Это единица измерения, состоящая из 4-х битов, b3 – b0. Может использоваться для представления чисел 0-9 (DEC) и 0-F (HEX)
- Байт : Это единица измерения, состоящая из 8-ми битов, b7 – b0. Может использоваться для представления чисел 00-FF (HEX)
- Слово : Это единица измерения, состоящая из 2-х байтов или 16-ти битов, b15 – b0. Может использоваться для представления чисел 0000-FFFF (HEX)



## 2. Восьмеричный формат чисел (OCT)

В контроллерах Delta используется для нумерации (адресации) входов и выходов:

Входы: X0, X1, X2, X3, X4, X5, X6, X7, X10, X11, ...

Выходы: Y0, Y1, Y2, Y3, Y4, Y5, Y6, Y7, Y10, Y11, ...

## 3. Десятичный формат чисел (DEC)

В контроллерах Delta используется в следующих случаях:

- для задания уставок таймеров и счетчиков, например TMR C0 K50.
- для адресации операндов M, T, C and D. Например: M10, T30.
- могут быть операндами в прикладных инструкциях, например MOV K123 D0. (K константа)

## 4. Двоично-десятичный формат чисел (BCD)

В BCD-формате каждая цифра десятичного числа представляется четырехбитным двоичным числом. В контроллерах Delta BCD-формат обычно используется для чтения входных значений от DIP-переключателей или для отображения выходных значений на 7-ми сегментном индикаторе.

## 5. Шестнадцатеричный формат чисел (HEX)

В контроллерах DVP используется для представления операнда H в прикладных инструкциях, например MOV H1A2B D0.

Константа К:

Символ "К" обычно ставится перед числом и обозначает, что число представлено в десятичном формате. Например, K100 означает 100 в десятичном формате.

Исключение: Символ "К" может использоваться для представления однобитных операндов X, Y, M в виде байтов, слов и двойных слов. Например, K2Y10 или K4M100.

Константа Н:

Символ "Н" обычно ставится перед числом и обозначает, что число представлено в шестнадцатеричном формате. Например, H100 означает 100 в шестнадцатеричном формате.

## D.6.5 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 Delta используется три типа внутренних реле:

1. Общие : не сохраняют свое состояние при отключении питания
2. Специальные : предоставляют в распоряжение пользователя специальные функции

Внутренние реле программируются как выходы. Однако отсутствует возможность присоединить к ним внешние устройства. Они могут использоваться в программе неограниченное число раз.

## D.6.6 Назначение таймеров [T]

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

Требуемая уставка времени определяется с помощью десятичной константы К, 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  $5 \times 100 = 500$  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 Т. Таймер возвращается в отключенное состояние и обнуляет свое текущее значение при установке своего входного контакта в "0".

## D.6.7 Свойства и назначение счетчиков [C]

Свойства:

Элемент	16-ти битный счетчик	32-х битный счетчик	
Тип	Общий	Общий	Высокоскоростной
Направление счета	Вверх (суммирование)	Вверх/вниз	
Диапазон счета	0–32 767	-2 147 483 648~+2 147 483 647	
Тип уставки	Константа К или регистр данных D	Константа К или регистр данных D (2 слова)	
Изменение текущего значения	Счет прекратится при достижении уставки	Счет будет продолжаться после достижения уставки	
Рабочий контакт	При достижении уставки контакт включится и зафиксируется	При текущем значении счета большем заданного контакт будет включен, при текущем значении счета меньшем заданного контакт будет выключен	
Сброс счетчика	Текущее значение счетчика будет обнулено и контакт С возвращен в исходное положение с помощью команды RST.		
Регистр текущего значения	16 бит	32 бит	
Быстродействие выхода	Выход счетчика будет обновлен в конце цикла сканирование вместе с другими	Выход счетчика будет обновлен в конце цикла сканирование вместе с другими	Выход счетчика будет обновлен немедленно при достижении уставки, не зависимо от цикла сканирования

Работа и назначение счетчиков:

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

16-ти битный счетчик C0–C7:

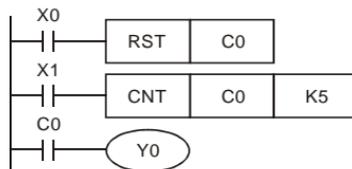
1. Диапазон заданных значений: K0 ... K32 767 (При K0 так же как и при K1, рабочий контакт будет замкнут после первого счета).
2. Общий счетчик будет обнулен при отключении питания ПЛК. Энергонезависимый счетчик сохранит свое текущее значение при отключении питания.
3. Если используется команда MOV, WPLSoft для изменения заданной уставки счетчика и будет записано значение больше, чем C0, при уже включенном контакте

C0, то контакт C0 сохранит свое состояние и текущее значение C0 будет таким же как заданное.

- Для задания уставки счетчика может использоваться десятичная константа K или регистр данных D (кроме специальных регистров D1000 – D1999) для косвенной уставки.
- Если для задания уставки используется десятичная константа K, то значения могут быть только положительными, а при использовании регистра – положительными и отрицательными в диапазоне от -32 768 до +32 767

Пример:

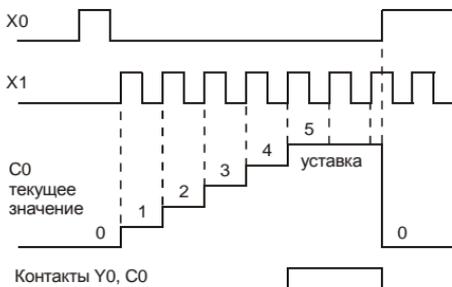
```
LD X0
RST C0
LD X1
CNT C0 K5
LD C0
OUT Y0
```



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.



32-х битный высокоскоростной счетчик C235:

- Диапазон заданных значений: K2 147 483 648–K2 147 483 647.
- Для задания уставки счетчика может использоваться десятичная константа K или два регистра данных D (кроме специальных регистров D1000 – D1999) для косвенной уставки.

Максимальная частота счетчика C235: для входов VFD-E 30 кГц, а для входов платы PG - 500 кГц.

## D.6.8 Типы и назначение регистров [D]

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

В контроллере VFD-E имеется 2 типа регистров:

1. Регистры : без сохранения данных при отключении напряжения питания данных общего назначения
2. Специальные : для определенных контрольных и проверочных функций е регистры

## D.6.9 Специальные реле

Номер	Функция	Атрибут
M1000	Нормально-открытый контакт. Контакт замкнут, когда на ПЛК подано напряжение питания, и он находится в состоянии RUN.	R
M1001	Нормально-закрытый контакт. Контакт разомкнут, когда на ПЛК подано напряжение питания, и он находится в состоянии RUN.	R
M1002	Контакт замыкается при включении ПЛК во время первого цикла выполнения программы на период, равный периоду сканирования. Все остальное время контакт разомкнут.	R
M1003	Контакт размыкается при включении ПЛК во время первого цикла выполнения программы на период, равный периоду сканирования. Все остальное время контакт замкнут.	R
M1004	Зарезервирован	--
M1005	Замыкается при возникновении ошибки, сбоя ПЧ.	R
M1006	Выходная частота = 0 Гц	R
M1007	Направление вращения привода	R
M1008	Зарезервирован	--
M1009	Зарезервирован	--
M1010	Зарезервирован	--
M1011	Датчик тактов с периодом 10мс (ON= 5 мс, OFF=5 мс)	R
M1012	Датчик тактов с периодом 100мс (ON= 50 мс, OFF=50 мс)	R
M1013	Датчик тактов с периодом 1 сек (ON= 0.5 сек, OFF=0.5 сек)	R
M1014	Датчик тактов с периодом 1 мин (ON= 30 сек, OFF = 30 сек)	R
M1015	Частота достигнута	R

Номер	Функция	Атрибут
M1016	Ошибка чтения/записи параметра	R
M1017	Запись параметра выполнена успешно	R
M1018	Разрешение функции высокоскоростного счетчика (Когда M1028=On)	R
M1019	Зарезервирован	R
M1020	Флаг нуля (Zero)	R
M1021	Флаг заимствования (Borrow)	R
M1022	Флаг переноса (Carry)	R
M1023	Делитель = 0	R
M1024	Зарезервирован	--
M1025	Пуск (ON) / Стоп (OFF) привода.	R/W
M1026	Изменение направления вращения привода (FWD: OFF, REV: ON)	R/W
M1027	Зарезервирован	--
M1028	Разрешение (ON)/ запрещение (OFF) функции высокоскоростного счетчика.	R/W
M1029	Сброс высокоскоростного счетчика.	R/W
M1030	Направление счета: вверх (OFF) / вниз (ON)	R/W
M1031	Зарезервирован	--

## D.6.10 Специальные регистры

Номер	Функция	Атрибут
D1000- D1009	Зарезервированы	--
D1010	Текущее время сканирования (ед.=0.1 мс)	R
D1011	Минимальное время сканирования (ед.= 0.1 мс)	R
D1012	Максимальное время сканирования (ед.= 0.1 мс)	R
D1013- D1019	0~32767 (ед. 0.1 мс) текущее время высокоскоростного таймера	R
D1020	Выходная частота	R
D1021	Выходной ток	R

Номер	Функция	Атрибут
D1022	Идентификационный номер (ID) платы расширения: 02 USB Card 03 12-Bit A/D (2CH) 12-Bit D/A (2CH) 04 Relay Card-2C 05 Relay Card-3A 06 3IN/3OUT Card 07 PG Card	R
D1023- D1024	Зарезервированы	--
D1025	Текущее значение высокоскоростного счетчика (младший байт)	R
D1026	Текущее значение высокоскоростного счетчика (старший байт)	R
D1027	Заданная частота ПИД-регулятора	R
D1028	Значение на аналоговом входе AV1: 0-10V соответствует 0-1023	R
D1029	Значение на аналоговом входе AC1: 4-20mA соответствует 0-1023 или значение на аналоговом входе AV12: 0-10V соответствует 0-1023	R
D1030	Значение потенциометра (V.R) пульта: 0-10V соответствует 0-1023	R
D1031- D1035	Зарезервированы	--
D1036	Код ошибки PLC	R
D1037- D1039	Зарезервированы	--
D1040	Значение на аналоговом выходе	R/W
D1041- D1042	Зарезервированы	--
D1043	Определяется пользователем (когда Pr.00.04 = 2, регистр данных будет отображаться на дисплее, как C xxx)	R/W
D1044	Режим высокоскоростного счетчика	R/W

### D.6.11 Коммуникационные адреса операндов (только для режима PLC2)

Операнд	Адрес	Операнд	Адрес	Операнд	Адрес
X0	0400H	Y0	0500H	T0-T15	0600H-060FH
X1	0401H	Y1	0501H	M0-M159	0800H-089FH
X2	0402H	Y2	0502H	M1000-M1031	0BE8H-0C07H
X3	0403H	Y3	0503H	C0-C7	0E00H-0E07H

X4	0404H	Y4	0504H	D0~D63	1000H~101DH
X5	0405H	Y5	0505H	D1000~D1044	13E8H~1414H
X6	0406H	Y6	0506H	--	--
X7	0407H	Y7	0507H	--	--
X10	0408H	Y10	0508H	--	--
X11	0409H	Y11	0509H	--	--
X12	040AH	Y12	050AH	--	--
X13	040BH	Y13	050BH	--	--
X14	040CH	Y14	050CH	--	--
X15	040DH	Y15	050DH	--	--
X16	040EH	Y16	050EH	--	--
X17	040FH	Y17	050FH	--	--

### D.6.12 Функциональный код (только для режима PLC2)

Код	Описание	Операнд
01	Чтение состояния выхода	Y, M, T, C
02	Чтение состояния входа	X, Y, M, T, C
03	Чтение данных	T, C, D
05	Изменение состояния выхода	Y, M, T, C
06	Запись данных	T, C, D
0F	Групповое изменение состояния выходов	Y, M, T, C
10	Групповая запись данных	T, C, D

## D.7 Команды

### D.7.1 Основные логические команды

Мнемоника	Функция	Операнды
LD	Нормально-открытый контакт	X, Y, M, T, C
LDI	Нормально-закрытый контакт	X, Y, M, T, C
AND	Последовательный нормально-открытый контакт (логическое И)	X, Y, M, T, C
ANI	Последовательный нормально-закрытый контакт (И-НЕ)	X, Y, M, T, C
OR	Параллельный нормально-открытый контакт (логическое ИЛИ)	X, Y, M, T, C
ORI	Параллельный нормально-закрытый контакт (ИЛИ-НЕ)	X, Y, M, T, C
ANB	«И» блок: последовательное включение параллельных связей	нет
ORB	«ИЛИ» блок: параллельное включение последовательных связей	нет
MPS	Смещение вниз по стеку	нет
MRD	Считать значение стека	нет
MPP	Выход из стека	нет
INV	Инверсия: замена результата логических связей на противоположный	нет

### D.7.2 Выходные команды

Инструкция	Функция	Операнды
OUT	ВЫХОД: присвоение выходу результата логического выражения	Y, M
SET	Включение операнда (установка лог. 1)	Y, M
RST	Сброс состояния операнда	Y, M, T, C, D

### D.7.3 Таймеры и счетчики

Инструкция	Функция	Операнды
TMR	Таймер (16 бит)	T-K или T-D
CNT	Счетчик (16 бит)	C-K или C-D

## D.7.4 Команды магистрального управления (мастер-контроля)

Инструкция	Функция	Операнды
MC	Включение условий мастер-контроля	N0~N7
MCR	Отключение условий мастер-контроля	N0~N7

## D.7.5 Входные команды с обнаружением переднего и заднего фронта

Инструкция	Функция	Операнды
LDP	Начало логического выражения с опросом по переднему фронту (импульс)	X, Y, M, T, C
LDF	Начало логического выражения с опросом по заднему фронту (импульс)	X, Y, M, T, C
ANDP	«И» с опросом по переднему фронту (импульс)	X, Y, M, T, C
ANDF	«И» с опросом по заднему фронту (импульс)	X, Y, M, T, C
ORP	«ИЛИ» с опросом по переднему фронту (импульс)	X, Y, M, T, C
ORF	«ИЛИ» с опросом по заднему фронту (импульс)	X, Y, M, T, C

## D.7.6 Выходные команды с выдачей импульса по переднему и заднему фронту

Инструкция	Функция	Операнды
PLS	Создание импульса по переднему фронту	Y, M
PLF	Создание импульса по заднему фронту	Y, M

## D.7.7 Конец программы

Инструкция	Функция	Операнды
END	Конец программы	нет

## D.7.8 Перечень прикладных инструкций

Классификация	Мнемоника	P (импульс. выполн.)	Функция
	16 бит		
Пересылка и сравнение данных	CMP	✓	Сравнение числовых данных
	ZCP	✓	Зонное сравнение числовых данных
	MOV	✓	Пересылка данных
	BMOV	✓	Пересылка блока данных
Арифметич	ADD	✓	Сложение двух чисел
	SUB	✓	Вычитание двух чисел

еские инструкции	MUL	✓	Умножение двух чисел
	DIV	✓	Деление двух чисел
	INC	✓	Инкрементирование (увеличение на 1)
	DEC	✓	Декрементирование (уменьшение на 1)
Инструкции сдвигов	ROR	✓	Кольцевой сдвиг вправо
	ROL	✓	Кольцевой сдвиг влево
Специальные инструкции	DHSCS	X	Разрешение высокоскоростного счета
	FPID	✓	Управление параметрами ПИД-регулятора ПЧ
	FREQ	✓	Управление выходной частотой ПЧ
	RPR	✓	Чтение параметров
	WPR	✓	Запись параметров

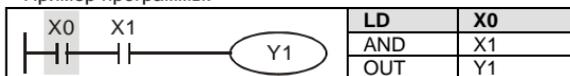
### D.7.9 Описание команд

Мнемоника	Функция						
LD	Нормально-открытый контакт						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



Команда "нормально-открытый контакт X0" открывает последовательную логическую связь.

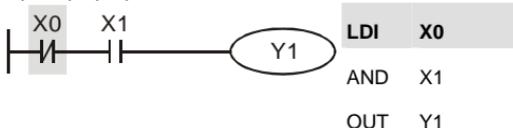
Если на входах X0 и X1 одновременно будет сигнал "1", тогда и выход Y1 установится в состояние "1".

Мнемоника	Функция						
LDI	Нормально-закрытый контакт						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



Команда "нормально-закрытый контакт X0" открывает последовательную логическую связь.

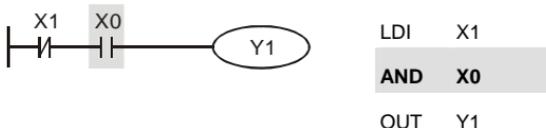
Если на входе X0 будет "0", а на X1 будет сигнал "1", тогда выход Y1 установится в состояние "1".

Мнемоника	Функция						
AND	Последовательный нормально-открытый контакт (логическое И)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



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".

Мнемоника	Функция						
ANI	Последовательный нормально-закрытый контакт (И-НЕ)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



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".

Мнемоника	Функция						
OR	Параллельный нормально-открытый контакт (логическое ИЛИ)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



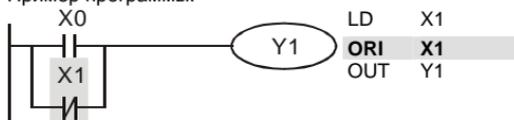
Команда "параллельный нормально-открытый контакт X1" создает параллельную логическую связь с контактом X0 и служит для выполнения операции логического сложения. Если хотя бы на одном из входов X0 или X1 будет "1", тогда и на выходе Y1 будет состояние "1".

Мнемоника	Функция						
ORI	Параллельный нормально-закрытый контакт (ИЛИ-НЕ)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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.

Пример программы:



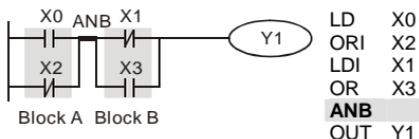
Команда "параллельный нормально-закрытый контакт X1" создает параллельную логическую связь с контактом X0 и служит для выполнения логической операции ИЛИ-НЕ. Если на входе X0 будет "1" или на входе X1 будет "0" (одно или оба условия одновременно), тогда на выходе Y1 будет состояние "1".

Мнемоника	Функция
ANB	«И» блок: последовательное включение параллельных блоков
Операнд	нет

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.

Пример программы:



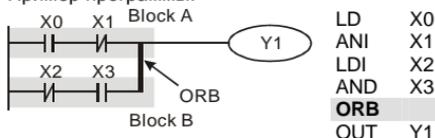
Команда ANB создает последовательную логическую связь между двумя логическими блоками (Block A и Block B).

Мнемоника	Функция
ORB	«ИЛИ» блок: параллельное включение последовательных блоков
<b>Операнд</b>	нет

Описание:

Команда ORB используется для параллельного соединения цепочек из двух последовательных контактов. Если несколько последовательных блоков включаются параллельно, то нужно после программирования каждого отдельного блока вводить ORB-инструкцию.

Пример программы:



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

Мнемоника	Функция
MPS	Смещение вниз по стеку
<b>Операнд</b>	нет

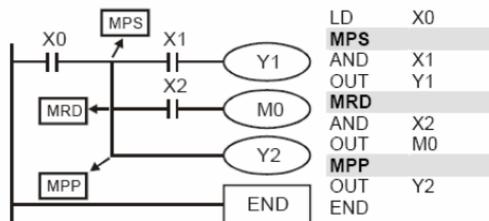
Мнемоника	Функция
MRD	Считать значение стека
<b>Операнд</b>	нет

Мнемоника	Функция
<b>MPP</b>	<b>Считать значение стека</b>
<b>Операнд</b>	<b>нет</b>

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.

Пример программы:



#### 1) 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.

## 2) 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.

## 3) MPP

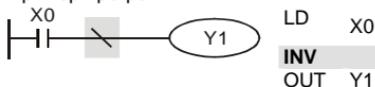
Перед выполнением следующей инструкции опрашивается промежуточный результат на 1-ом месте памяти логических связей. Устанавливается выход M0. Операция на 1-ом уровне промежуточных результатов завершена, и память логических связей стирается.

Мнемоника	Функция
INV	Инверсия - замена результата логических связей на противоположный
Операнд	нет

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".

Пример программы:



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

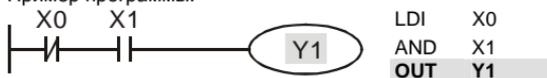
Мнемоника	Функция						
OUT	Выход						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	--	✓	✓	✓	--	--	--

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 может привести к помехам при отработке программы.

Пример программы:



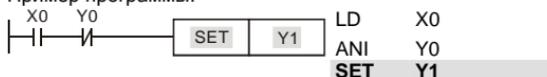
При условии: X0=0 и X1=1 – команда OUT Y1 установит выход контроллера Y1 в состояние "1".

Мнемоника	Функция						
SET	Включение выхода с фиксацией						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	--	✓	✓	✓	--	--	--

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.
- в том случае, если входные условия для SET инструкции больше не будут выполняться, соответствующий операнд останется включенным.

Пример программы:



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 ПЛК.

Мнемоника	Функция						
RST	Сброс состояния операнда						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~T15	C0~C7	D0~D29	X0~X17
	--	✓	✓	✓	✓	✓	✓

Описание:

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

- С помощью RST-инструкции могут отключаться соответствующие операнды. Это означает:
  - выходы Y, контакты M выключаются (состояние сигнала "0").
  - действительное значение таймера и счетчика, а также содержание регистров D сбрасываются на "0".
  - как только входное условие установится для RST инструкции (сигнал "1"), выключится соответствующий операнд.
  - в том случае, если входные условия для RST инструкции больше не будут выполняться, соответствующий операнд останется выключенным.

Пример программы:



Выход Y5 выключится при выполнении условия X0 и останется выключенным даже когда условие X0 выполняться не будет.

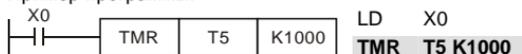
Мнемоника	Функция	
TMR	Таймер (16 бит)	
Операнд	T-K	T0~T15, K0~K32767
	T-D	T0~T15, D0~D29

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.

Пример программы:



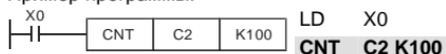
При условии X0=1 инструкция TMR T5 будет вести отчет времени, пока значение в регистре T5 не достигнет значения K1000 (100 сек). При X0=0 выполнение инструкции TMR прекратится и T5 сбросится на "0".

Мнемоника	Функция	
CNT	Счетчик (16 бит)	
Операнд	C-K	C0~C7, K0~K32767
	C-D	C0~C7, D0~D29

Описание:

1. Команда CNT служит для суммирования количества замыканий входного контакта (макс. 32767 импульсов) и присвоения состояния сигнала (включения или отключения выхода) когда текущее значение счетчика достигнет заданного значения.
2. С помощью инструкции CNT можно завершить программирование строки (логического выражения).
3. 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.
4. Для сброса текущего значения счетчика можно использовать команду RST.

Пример программы:



При изменении состояния X0 с "0" на "1" значение регистра C2 будет увеличено на 1, и так пока значение в регистре C2 не достигнет значения K100 (100 импульсов). После этого счет прекратится. Для сброса значения регистра C2 можно использовать команду RST C2.

<b>Мнемоника</b>	<b>Функция</b>
MC / MCR	Включение / выключение условий мастер-контроля
<b>Операнд</b>	N0–N7

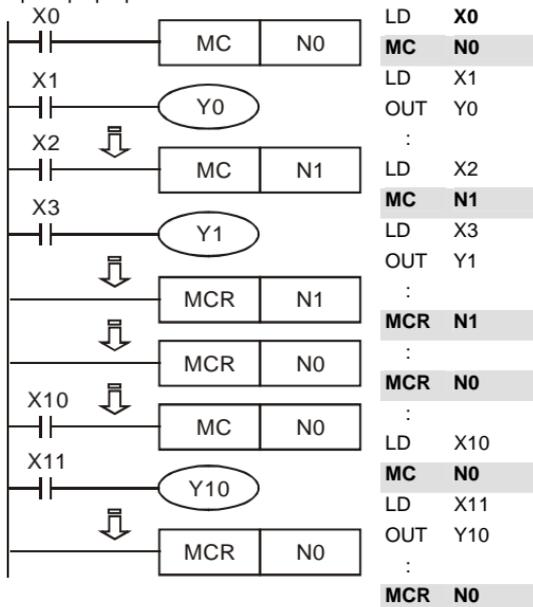
Описание:

1. Благодаря включению (MC) или отключению (MCR) условий контроля можно активизировать или деактивизировать отдельные области программ. Функция работает как главный контакт левой сборной (питающей) шины (программирование в контактной схеме).
2. С помощью MC-инструкции включаются условия контроля для активизации определенной области программы.
  - какая область программы должна активизироваться, определяется указанием адреса программирования разветвления n: определяется от N0 до N7 (адрес разветвления).
  - задание операнда Y или M определяет контакт включения. Этот контакт активизирует область программы n, как только выполняются входные условия для MC-инструкции.
3. После программирования MC-инструкции должны всегда программироваться LD или LDI инструкции.
4. MCR-инструкция отключает MC-контакт и представляет тем самым конец области программирования n.
5. Если входные условия не выполняются, состояния операндов изменяются между MC и MCR как показано ниже:

Таймеры	Текущие значения и контакты будут сброшены
Аккумулятивные таймеры	Сохранят свое текущее значение, а их контакты будут сброшены
Счетчики	Сохранят свое текущее значение, а их контакты будут сброшены
Катушки программируемые по OUT-инструкции	Все будут сброшены
Операнды, которые программируются по SET и RST инструкциям	Сохранят свое состояние
Прикладные инструкции	Все прикладные инструкции выполняться не будут

6. Внутри программы ПЛК могут быть созданы до 8 уровней вложенности. Уровень разветвления характеризуется параметром "n".
7. То, на что нужно обращать внимание при использовании нескольких MC и MCR инструкций внутри одной программы, пояснено на следующем примере.

Пример программы:



Область программы между инструкциями MC N0 и MCR N0 будет выполняться только если X0=1. Область программы между инструкциями MC N1 и MCR N1 будет выполняться только если X0=1 и X2=1

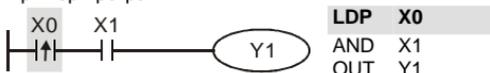
Мнемоника	Функция						
LDP	Начало логического выражения с опросом по переднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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).

Пример программы:



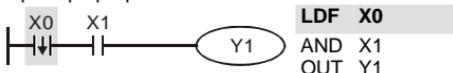
Команда "LDP X0" открывает последовательную логическую связь. Если вход X0 изменит свое состояние с "0" на "1" (при этом X1=1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

Мнемоника	Функция						
LDF	Начало логического выражения с опросом по заднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

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).

Пример программы:



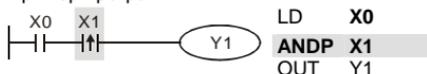
Команда "LDF X0" открывает последовательную логическую связь. Если вход X0 изменит свое состояние с "1" на "0" (при этом X1=1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

Мнемоника	Функция						
ANDP	«И» с опросом по переднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

Описание:

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

Пример программы:



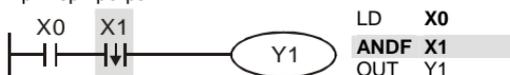
Команда "ANDP X1" создает последовательную логическую связь. Если вход X1 изменит свое состояние с "0" на "1" (при этом X0=1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

Мнемоника	Функция						
ANDF	«И» с опросом по заднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

Описание:

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

Пример программы:



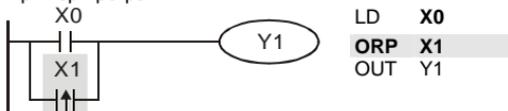
Команда "ANDF X1" создает последовательную логическую связь. Если вход X1 изменит свое состояние с "1" на "0" (при этом X0=1), тогда выход Y1 будет в состоянии "1" в течении одного скана.

Мнемоника	Функция						
ORP	«ИЛИ» с опросом по переднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

Описание:

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

Пример программы:



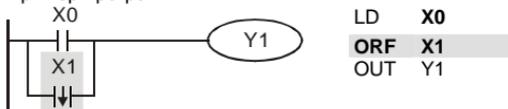
Команда "ORP X1" создает параллельную логическую связь. Выход Y1 будет в состоянии "1" в течении одного скана если вход X1 изменит свое состояние с "0" на "1" или X0=1.

Мнемоника	Функция						
ORF	«ИЛИ» с опросом по заднему фронту (импульс)						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	✓	✓	✓	✓	✓	✓	--

Описание:

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

Пример программы:



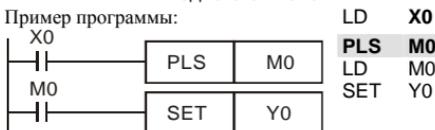
Команда "ORF X1" создает параллельную логическую связь. Выход Y1 будет в состоянии "1" в течении одного скана если вход X1 изменит свое состояние с "1" на "0" или X0=1.

Мнемоника	Функция						
PLS	Создание импульса по переднему фронту						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	--	✓	✓	--	--	--	--

Описание:

- Команда PLS служит для генерации одного импульса – опознание переднего фронта сигнала - независимо от продолжительности входного сигнала для включения соответствующего операнда. Операнд остается во включенном состоянии на протяжении одного цикла программы (скана).
- PLS-инструкция может использоваться совместно с реле М и цифровым выходом Y. Инструкции генерируют одинаковые импульсы независимо от продолжительности входного сигнала.
- После исполнения PLS, сигнал соответствующего операнда (Y или M) удерживается в "1" на протяжении одного скана.
- PLS-инструкция генерирует одиночный импульс по возрастающему фронту входного сигнала.

Пример программы:



```

LD X0
PLS M0
LD M0
SET Y0

```

Временная диаграмма:



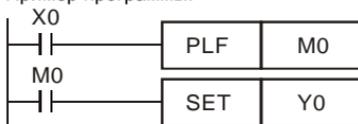
При изменении входного сигнала на входе X0 с "0" на "1" (возрастающий фронт) реле M0 благодаря PLS-инструкции получает импульс (включается на время одного скана). С помощью этого импульса по контакту реле M0 включается выход Y0.

Мнемоника	Функция						
PLF	Создание импульса по заднему фронту						
Операнд	X0~X17	Y0~Y17	M0~M159	T0~15	C0~C7	D0~D29	X0~X17
	--	✓	✓	--	--	--	--

Описание:

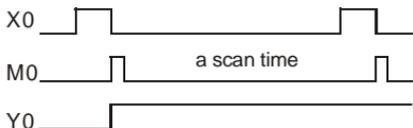
- Команда PLF служит для генерации одного импульса – опознание заднего фронта сигнала - независимо от продолжительности входного сигнала для включения соответствующего операнда. Операнд остается во включенном состоянии на протяжении одного цикла программы (скана).
- PLF-инструкция может использоваться совместно с реле М и цифровым выходом Y. Инструкции генерируют одинаковые импульсы независимо от продолжительности входного сигнала.
- После исполнения PLF, сигнал соответствующего операнда (Y или M) удерживается в "1" на протяжении одного скана.
- PLF-инструкция генерирует одиночный импульс по заднему фронту входного сигнала.

Пример программы:



```
LD X0
PLF M0
LD M0
SET Y0
```

Временная диаграмма:



При изменении входного сигнала на входе X0 с "1" на "0" (спадающий фронт) реле M0 благодаря PLS-инструкции получает импульс (включается на время одного скана). С помощью этого импульса по контакту реле M0 включается выход Y0.

Мнемоника	Функция
END	Конец программы
Операнд	нет

Описание:

Окончание программы ПЛК и переход к началу программы (шаг 0)

Каждая программа ПЛК должна завершаться инструкцией END.

API	Мнемоника		Операнды	Функция
10	CMP	P	S <sub>1</sub> , S <sub>2</sub> , D	Сравнение числовых данных

Тип Оп.	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S <sub>1</sub>				*	*	*	*	*	*	*	*	*
S <sub>2</sub>				*	*	*	*	*	*	*	*	*
D		*	*									

:

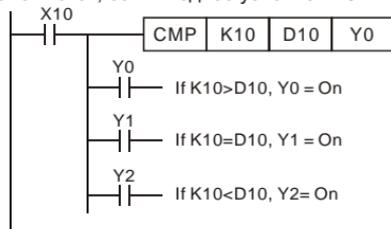
S<sub>1</sub>: Первое сравниваемое число S<sub>2</sub>: Второе сравниваемое число D: Результат сравнения

Описание:

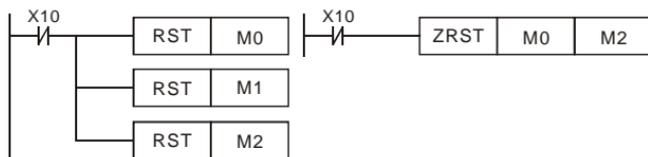
1. Операнд D занимает 3 последовательных адреса
2. Данные в обоих источниках **S<sub>1</sub>** и **S<sub>2</sub>** сравниваются друг с другом и результат сравнения (больше, меньше, равно) отображается (индицируется) благодаря задействию реле M, операнда состояния шага S или выхода Y. Определение, какой из этих операндов должен задействоваться, выполняется по адресу результата **D**.
3. Данные в S1 и S2 обрабатываются как двоичные данные.

Пример программы:

1. В адресе результата **D** в этом примере указан выход Y0. Соответствующие результаты сравнения автоматически присваиваются приращиваемым на 1 последующим адресам выходов Y0, Y1, Y2 и имеют следующие значения:
  - 1) Y0: включен, если K10 > значения регистра D10
  - 2) Y1: включен, если K10 = значению регистра D10
  - 3) Y2: включен, если K10 < значения регистра D10
3. Y0, Y1, Y2 не изменяются, если входное условие X10 выключено.



4. Для сброса результатов сравнения используйте команды RST, ZRST.



API	Мнемоника	Операнды	Функция
11	ZCP	P	S <sub>1</sub> , S <sub>2</sub> , S, D
			Зонное сравнение числовых данных

Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S <sub>1</sub>				*	*	*	*	*	*	*	*	*
S <sub>2</sub>				*	*	*	*	*	*	*	*	*
S				*	*	*	*	*	*	*	*	*
D		*	*									

:

S1: Первое сравниваемое число (Minimum) S2: Второе сравниваемое число (Maximum) S: Сравниваемое число D: Результат сравнения

Описание:

1. Операнд S<sub>1</sub> должен быть меньше операнда S<sub>2</sub>.
2. Операнд D занимает 3 последовательных адреса.
3. Данные в источнике S сравниваются с данными в обоих источниках S1 и S2. Результат сравнения (больше, меньше, равно) отображается (индицируется) благодаря задействию реле M, операнда состояния шага S или выхода Y. Определение, какой из этих операндов должен задействоваться, выполняется в регистре данных (по адресу цели) - D.
4. Данные в S1, S2 и S обрабатываются как двоичные данные.

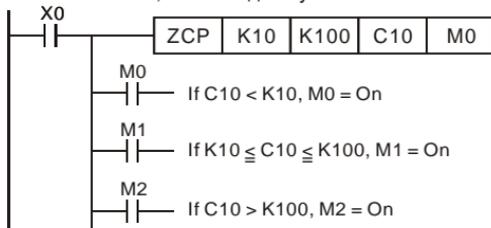
Пример программы:

1. В адресе цели (D) в этом примере указано реле M0. Соответствующие результаты сравнения автоматически присваиваются приращиваемым на 1 последующим адресам реле M0, M1, M2 и имеют следующие значения:
2. 1) M0: включен, если K10 > накопленного в счетчике C10 значения

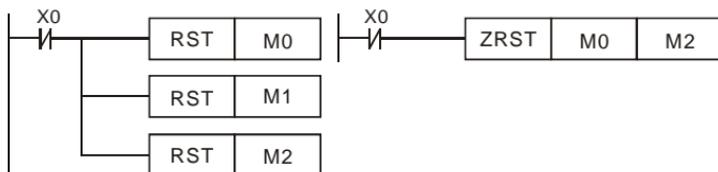
2) Если накопленное значение счетчика C10 находится в области от 10 до 100, то реле M1 включено.

3) M2: включен, если накопленное в счетчике C10 значение > K100

3. M0, M1, M2 не изменяются, если входное условие X0 выключено.



4. Для сброса результатов сравнения используйте команды RST, ZRST.



API	Мнемоника	Операнды	Функция
12	MOV P	S, D	Передача данных

Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S				*	*	*	*	*	*	*	*	*
D							*	*	*	*	*	*

:

S: Источник данных D: Данные цели

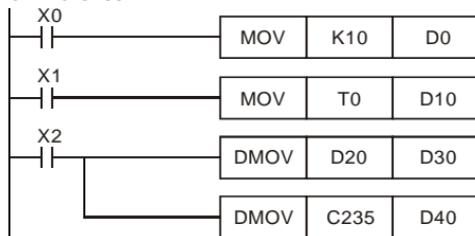
Описание:

1. Инструкция служит для передачи данных от источника данных **S** к данным цели **D**. Содержимое источника **S** при этом не изменяется.
2. Данные в источнике данных **S** при выполнении MOV-инструкции автоматически интерпретируются как двоичные значения.
3. Инструкции выполняются в каждом цикле программы. Этого можно избежать благодаря использованию вставленной впереди импульсной функции (PLS- или

PLF-инструкции или же параметра "P").

Пример программы:

1. Если входное условие X0 включено, то значение регистра D0 будет равно 10. Если X0 выключен, значение D0 не изменится.
2. Если входное условие X1 включено, то регистру D10 будет передаваться текущее значение таймера T0. Если X1 выключен, значение D10 не изменится.
3. Если входное условие X2 включено, то регистрам (D30, D31) будет передаваться значение регистров (D20, D21) и регистрам (D40, D41) будет передаваться текущее значение счетчика C235.



API	Мнемоника	Операнды	Функция
15	BMOV P	S, D, n	Передача блока данных

Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S						*	*	*	*	*	*	BMOV, BMOV P: 7 шагов
D						*	*	*	*	*	*	
n				*	*				*	*	*	

:

S: Источник D: Цель n: Число передаваемых данных

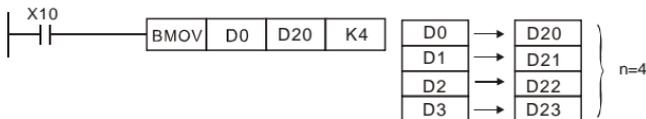
Описание:

1. Диапазон операнда n=1~ 512
2. Переносится (передается) предварительно заданное количество словных операндов.
3. Для передачи предварительно задаются стартовый адрес S, адрес цели D и число переносимых слов n.
4. Если величина пакета данных превышает величину областей цели или источника,

то передаются только слова, которые могут поместиться в области.

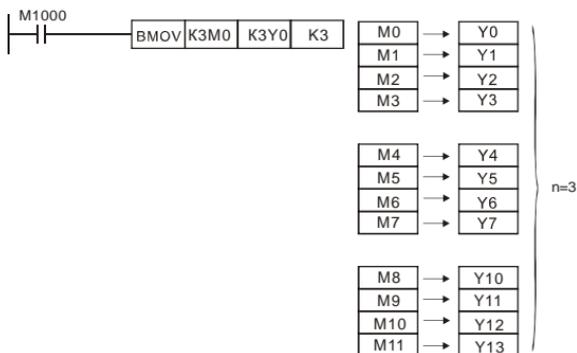
Пример программы 1:

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



Пример программы 2:

Если передаются операнды KnX, KnY, KnM, число слов в S и D должно быть одинаковым и равным n.

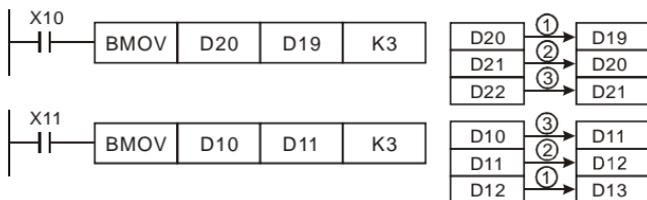


Пример программы 3:

Если адреса источников S и приемников D данных пересекаются, то порядок копирования будет следующий:

Если  $S > D$ : порядок копирования  $1 \rightarrow 2 \rightarrow 3$

Если  $S < D$ : порядок копирования  $3 \rightarrow 2 \rightarrow 1$



API	Мнемоника		Операнды	Функция								
20	ADD	P	S <sub>1</sub> , S <sub>2</sub> , D	Сложение числовых данных								
Тип OP	Биты			Слова								Шаги в программе
	X	Y	M	K	H	KnX	KnY	KnM	T	C	D	ADD, ADDP: 7 шагов DADD, DADDP: 13 шагов
S <sub>1</sub>			*	*	*	*	*	*	*	*	*	
S <sub>2</sub>				*	*	*	*	*	*	*	*	
D						*	*	*	*	*	*	

Операнды:

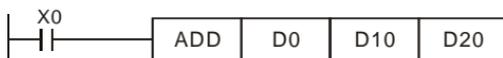
S1: Первое слагаемое S2: Второе слагаемое D: Сумма

Описание:

- S<sub>1</sub> + S<sub>2</sub> = D.** Двоичные данные в адресах источников **S1** и **S2** суммируются. Результат суммирования запоминается в адресе цели **D**.
- В старшем бите запоминается знак числа суммирования:  
0: знак положительного числа 1: знак отрицательного числа, напр. 3 + (-9) = -6.
- При определенных результатах счета после исполнения инструкции включается специальное реле (флаг).  
16-битные операции:
  - ФЛАГ НУЛЯ (Zero) M1020:** если результатом сложения является 0, включается флаг нуля
  - ФЛАГ ЗАИМСТВОВАНИЯ (Borrow) M1021:** если результатом сложения явилось число меньше -32 767, включается флаг заимствования.
  - ФЛАГ ПЕРЕНОСА (Carry) M1022:** если результатом сложения явилось число выше +32 767, включается флаг переноса.

Пример программы 1:

Если включен X0, то суммируются значения данных в регистрах D0 и D10. Результат суммирования запоминается в регистре данных D20.



API	Мнемоника		Операнды	Функция
21	SUB	P	S <sub>1</sub> , S <sub>2</sub> , D	Вычитание числовых данных

Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S <sub>1</sub>				*	*	*	*	*	*	*	*	SUB, SUBP: 7 шагов DSUB, DSUBP: 13 шагов
S <sub>2</sub>				*	*	*	*	*	*	*	*	
D							*	*	*	*	*	

Операнды:

S<sub>1</sub>: Уменьшаемое S<sub>2</sub>: Вычитаемое D: Разность

Описание:

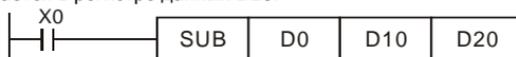
1.  $S_1 - S_2 = D$ . Значение данных в **S2** вычитается из значения данных **S1**. Результат вычитания запоминается в адресе цели **D**. Операция выполняется в BIN-формате.
2. В старшем бите запоминается знак числа вычитания:  
0: знак положительного числа 1: знак отрицательного числа
3. При определенных результатах счета после исполнения инструкции включается специальное реле (флаг).

16-битные операции:

- A. Если результат операции = "0", включится Zero flag, M1020 = ON.
- B. Если результат операции меньше -32768, включится borrow flag, M1021 = ON.
- C. Если результат операции больше 32767, включится carry flag, M1022 = ON.

Пример программы:

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



API	Мнемоника		Операнды	Функция
22	MUL	P	S <sub>1</sub> , S <sub>2</sub> , D	Умножение числовых данных

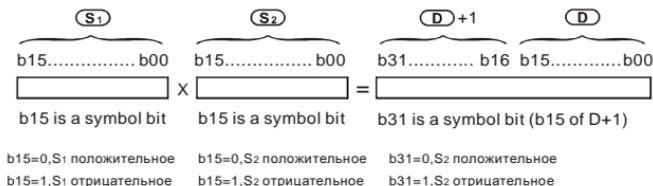
Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S <sub>1</sub>				*	*	*	*	*	*	*	*	MUL, MULP: 7 шагов
S <sub>2</sub>				*	*	*	*	*	*	*	*	
D							*	*	*	*	*	

Операнды:

S<sub>1</sub>: Множимое S<sub>2</sub>: Множитель D: Произведение

Описание:

- В 16-ти битном режиме операнд D занимает 2 адреса.
- S<sub>1</sub> × S<sub>2</sub> = D**. Данные в S<sub>1</sub> и S<sub>2</sub> перемножаются между собой. Результат умножения запоминается по адресу операнда указанного в D и в следующем за ним адресе операнда. Операция выполняется в BIN-формате.
- При выполнении 16-ти битной операции результат заносится в 32-х битное число в (D) и (D+1). Результат 16-ти битного умножения оказывается 32-х битным числом. Это число запоминается как 32-х битное значение. Младшие 16 бит записываются по адресу операнда, заданному в (D). Старшие 16 бит записываются по следующему за ним адресу операнда.

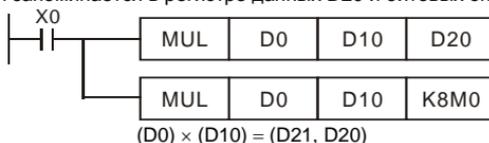


Если D - битовый операнд и размер его K1-K4 для 16-ти битной операции, то флаги ошибки M1067, M1068 включаться в регистре D1067 будет записан код "0E19"

Пример программы:

Если включен X0, то находится произведение значений данных в регистрах D0 и D10.

Результат умножения запоминается в регистре данных D20 и битовых операндах M0 – M31.



API	Мнемоника		Операнды	Функция
23	DIV	P	S <sub>1</sub> , S <sub>2</sub> , D	Деление числовых данных

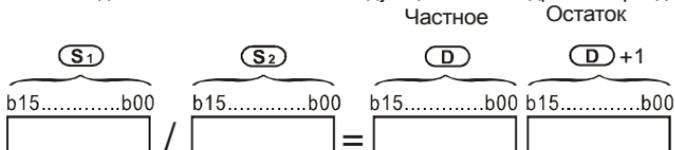
Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
S <sub>1</sub>				*	*	*	*	*	*	*	*	*
S <sub>2</sub>				*	*	*	*	*	*	*	*	*
D							*	*	*	*	*	*

Операнды:

S<sub>1</sub>: Делимое S<sub>2</sub>: Делитель D: Частное и остаток

Описание:

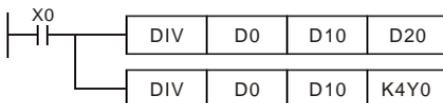
- В 16-ти битном режиме операнд D занимает 2 адреса.
- $S_1 \div S_2 = D$ . Выполняется деление данных в (S1) и (S2). Данные в (S1) соответствуют делимому, в (S2) - делителю. Результат деления запоминается по адресу операнда указанного в (D). Остаток делимого запоминается в следующем адресе операнда. Операция выполняется в BIN-формате.
- Результат 16-ти битного деления запоминается по адресу операндов, заданном в D. Остаток от деления запоминается в следующем за ним адресе операнда.



При работе программы появляется ошибка, если значение делителя равно 0. Флаги ошибки M1067, M1068 включатся в регистре D1067 будет записан код "0E19".

Пример программы:

Если включен X0, то выполняется деление значений данных в регистрах D0 и D10. Результат деления запоминается в регистре данных D20 и битовых операндах Y0 – Y15, а остаток деления запоминается в D21.



API	Мнемоника			Операнды			Функция						
24		INC	P	D			Инкрементирование числовых данных						
Тип OP	Биты			Слова						Шаги в программе			
	X	Y	M	K	H	KnX	KnY	KnM	T	C	D	INC, INCP: 3 шагов	
D							*	*	*	*	*		

Операнды:

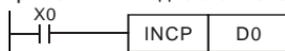
D: Цель

Описание:

1. К числовому значению данных прибавляется число 1, как только выполнится входное условие.
2. Инструкция выполняется в каждом цикле программы. Этого можно избежать благодаря введению впереди функций импульса (PLS- или PLF-инструкции) или применив командный параметр P.
3. Если при 16-ти битовом операнде значение 1 добавится к + 32 767, то запишется значение - 32 768. Не появляется никакого флага.

Пример программы:

Значение данных в регистре D0 при наличии входного сигнала X0 увеличится на число 1.



API	Мнемоника			Операнды			Функция						
25		DEC	P	D			Декрементирование числовых данных						
Тип OP	Биты			Слова						Шаги в программе			
	X	Y	M	K	H	KnX	KnY	KnM	T	C	D	DEC, DECP: 3 шагов	
D							*	*	*	*	*		

Операнды:

D: Цель

Описание:

1. Из значения числа, имеющегося в D, вычитается число 1, как только выполнится входное условие.
2. Инструкция выполняется в каждом цикле программы. Этого можно избежать благодаря введению впереди функций импульса (PLS- или PLF-инструкции) или применив командный параметр P.
3. Если при 16-ти битовом операнде значение 1 отнимется от числа -32 768, то

запишется значение +32 768. Не появляется никакого флага.

Пример программы:

Значение данных в регистре D0 при наличии входного сигнала X0 уменьшится на число 1.



API	Мнемоника		Операнды	Функция
30	ROR	P	D, n	Кольцевой сдвиг вправо

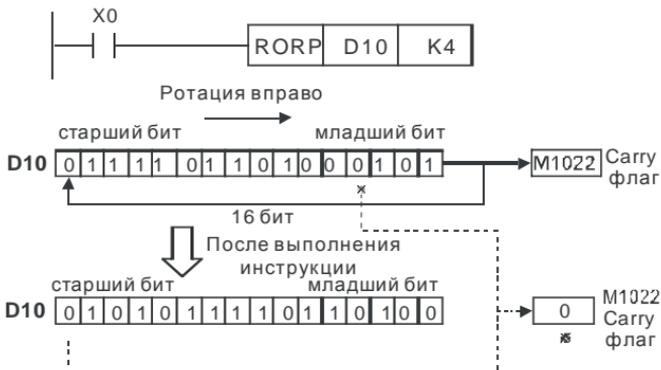
Тип OP	Биты			Слова							Шаги в программе	
	X	Y	M	K	H	KnX	KnY	KnM	T	C		D
D							*	*	*	*	*	ROR, RORP: 5 шагов
n				*	*							

Описание:

1. Битовое отображение в **D** сдвигается вправо на **n** мест при каждом исполнении ROR
2. Если операнд, D это KnY, KnM, KnS, возможно только Kn = K4 (16 бит)
3. Необходимое условие:  $1 \leq n \leq 16$  (16-бит).
4. Состояние последнего сдвигаемого бита копируются в M1022 (флаг переноса - Carry)
5. Если не программируется никакого опознания фронта, то сдвиг битового отображения повторяется в каждом цикле программы.

Пример программы:

Битовые данные в регистре данных D10 каждый раз сдвигаются вправо на 4 бита (K4), когда вход X0 переходит из состояния ОТКЛ. в состояние ВКЛ. Значение последнего сдвигаемого бита запоминается во флаге переноса (M1022).

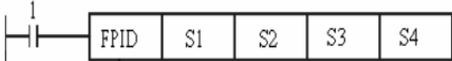


### Специальные инструкции

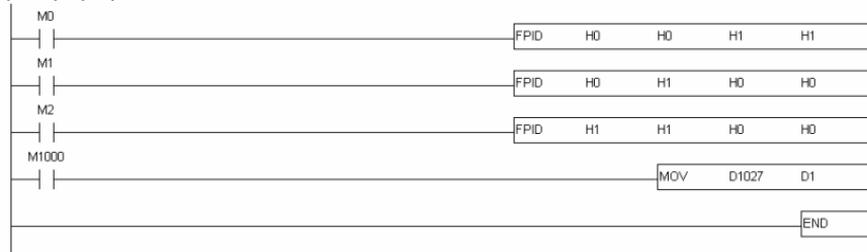
Мнемоника	Функция	Операнды
<b>DHSCS</b>	<p>Высокоскоростной счетчик</p> <p>1: START/STOP (вкл/выкл)            S1: Заданное значение счетчика            S2: Текущее значение счетчика            S3: Выходное реле</p>	<p>1: (X, Y, M, T, C)            S1: (K, H, D)            S2: (C235)            S3: (Y, M)</p>

Пример программы:



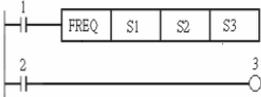
Мнемоника	Функция	Операнды
<b>FPID</b>	ПИД-регулятор  1:PID RUN/STOP (вкл/выкл) S1: Заданное значение ПИД-регулятора. S2: Kp (пропорциональный коэффициент) S3: Ki (интегральный коэффициент) S4: Kd (дифференциальный коэффициент)	1:( X, Y, M, T, C) S1: (K,H,D) S2: (K,H,D) S3: (K,H,D) S4: (K,H,D)

Пример программы:

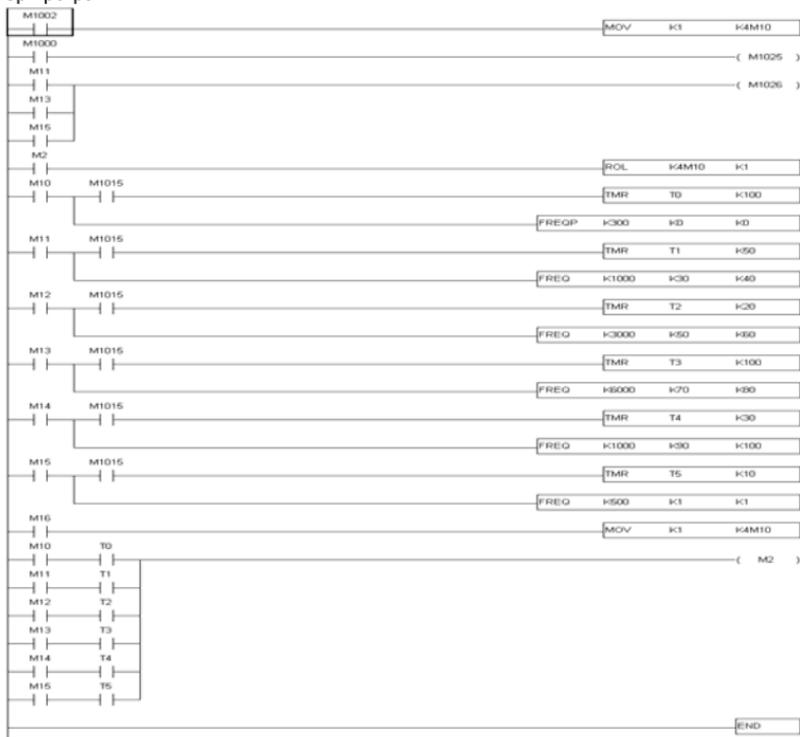


Описание:

Эта инструкция позволяет напрямую управлять параметрами ПИД-регулятора: 10.00, 10.02 и 10.03.

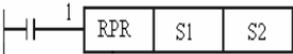
Мнемоника	Функция	Операнды
<b>FREQ</b>	<p>Управление выходной частотой ПЧ</p>  <p>1:RUN/STOP (вкл/выкл)  S1: Заданная частота  S2: Время разгона  S3: Время замедления</p>	<p>1:( X, Y, M, T, C)  S1: (K,H,D)  S2: (K,H,D)  S3: (K,H,D)  2:( X, Y, M, T, C)  3:M1028</p>

Пример программы:



Описание:

Команда **FREQ** позволяет управлять скоростью вращения привода и временем разгона/замедления. **M1025** и **M1026** позволяют запускать и останавливать привод (команды **RUN/STOP**) и изменять направление вращения.

Мнемоника	Функция	Операнды
<b>RPR</b>	<p>Чтение значений параметров ПЧ</p>  <p>1:READ/NONE (вкл/выкл) S1: Номер параметра S2: Адрес регистра, в который будет сохранено значение параметра</p>	<p>1:( X, Y, M, T, C) S1: (K, H, D) S2: (D)</p>
<b>WPR</b>	<p>Запись значений параметров в ПЧ</p>  <p>1:WRITE/NONE (вкл/выкл) S1: Адрес регистра, значение которого будет передано в параметр S2: Номер параметра</p>	<p>1:( X, Y, M, T, C) S1: (K,H,D) S2: (D,H,K)</p>

Пример программы:



Описание:

Команды RPR и WPR могут использоваться для чтения/записи параметров ПЧ и коммуникационных адресов.

Коды ошибок

Код	ID	Описание	Необходимые действия
PLod	20	Ошибка записи программы	Проверьте программу на наличие ошибок и попробуйте загрузить снова.
PLSv	21	Ошибка записи данных во время выполнения	Выключите и включите питание и попробуйте загрузить снова.
PLdA	22	Ошибка чтения программы	1. Попробуйте еще. 2. Свяжитесь с поставщиком
PLFn	23	Ошибка команды при загрузке	Проверьте программу на наличие ошибок и попробуйте загрузить снова.
PLor	30	Размер программы превышает объем памяти	Выключите и включите питание и попробуйте загрузить снова.
PLFF	31	Ошибка команды при выполнении	
PLSn	32	Ошибка контрольной суммы	
PLEd	33	В программе отсутствует инструкция "END".	
PLCr	34	Команда MC непрерывно используется более 9 раз	

## D.8 Таблица применений ПЛК

Внешний (1): Внешний аналоговый сигнал; Внешний (2): Дискретные входы

Первый источник задания частоты (Pr.02.00)				Источник команд управления приводом (Pr.02.01)				Пояснение
Внешний (1) (Pr.02.00=2/3)	PLC	COM (Pr.02.00=4/5)	KEYPAD (Pr.02.00=0/1)	Внешний (2) (Pr.02.01=1/2)	PLC	COM (Pr.02.01=3/4)	KEYPAD (Pr.02.01=0)	
X	X	X	O	X	X	X	O	FREQUENCY, M1025 and M1026 cannot be used in the PLC program.
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	
				O	X	X	X	
				O	X	X	O	
X	X	O	X	X	X	X	O	
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	
				O	X	X	X	
				O	X	X	O	
X	X	O	O	X	X	X	O	
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	
				O	X	X	X	
				O	X	X	O	
X	O	X	X	X	X	X	O	M1025 and M1026 cannot be used in the PLC program.
				X	X	O	X	
				X	X	O	O	

Первый источник задания частоты (Pr.02.00)				Источник команд управления приводом (Pr.02.01)				Пояснение
Внешний (1) (Pr.02.00=2/3)	PLC	COM (Pr.02.00=4/5)	KEYPAD (Pr.02.00=0/1)	Внешний (2) (Pr.02.01=1/2)	PLC	COM (Pr.02.01=3/4)	KEYPAD (Pr.02.01=0)	
				X	O	X	X	FREQ, RPR, M1025 и M1026 могут использоваться в программе ПЛК.
				O	X	X	X	M1025 и M1026 не могут исп-ся в программе ПЛК.
				O	X	X	O	
X	O	X	O	X	X	X	O	M1025 и M1026 не могут использоваться в программе ПЛК.
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	нет
				O	X	X	X	M1025 и M1026 не могут исп-ся в программе ПЛК.
				O	X	X	O	
X	O	O	X	--	--	--	--	нет
X	O	O	O	--	--	--	--	нет
O	X	X	X	X	X	X	O	FREQ, M1025 и M1026 не могут использоваться в программе ПЛК.
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	
				O	X	X	X	
				O	X	X	O	
O	X	X	O	X	X	X	O	
				X	X	O	X	
				X	X	O	O	
				X	O	X	X	

Первый источник задания частоты (Pr.02.00)				Источник команд управления приводом (Pr.02.01)				Пояснение
Внешний (1) (Pr.02.00=2/3)	PLC	COM (Pr.02.00=4/5)	KEYPAD (Pr.02.00=0/1)	Внешний (2) (Pr.02.01=1/2)	PLC	COM (Pr.02.01=3/4)	KEYPAD (Pr.02.01=0)	
				○	×	×	×	
				○	×	×	○	
○	×	○	×	×	×	×	○	
				×	×	○	×	
				×	×	○	○	
				×	○	×	×	
				○	×	×	×	
○	×	×	○	○	×	×	○	
○	×	○	○	--	--	--	--	нет
○	○	×	×	×	×	×	○	M1025 и M1026 не могут использоваться в программе ПЛК.
				×	×	○	×	
				×	×	○	○	
				×	○	×	×	
				○	×	×	×	
○	×	×	○	○	×	×	○	
○	○	×	○	×	×	×	○	M1025 и M1026 могут использоваться в программе ПЛК.
				×	×	○	×	
				×	×	○	○	
				×	○	×	×	
				○	×	×	×	
○	×	×	○	○	×	×	○	
○	○	○	×	--	--	--	--	нет
○	○	○	○	--	--	--	--	нет



**DELTA ELECTRONICS, INC.**

[www.delta.com.tw/industrialautomation](http://www.delta.com.tw/industrialautomation)

## **ASIA**

**Delta Electronics, Inc.**

Taoyuan1

31-1, Xingbang Road, Guishan Industrial Zone,

Taoyuan County 33370, Taiwan, R.O.C.

TEL: 886-3-362-6301 / FAX: 886-3-362-7267

**Delta Electronics (Jiang Su) Ltd.**

Wujiang Plant3

1688 Jiangxing East Road,

Wujiang Economy Development Zone,

Wujiang City, Jiang Su Province,

People's Republic of China (Post code: 215200)

TEL: 86-512-6340-3008 / FAX: 86-769-6340-7290

## **EUROPE**

**Deltronics (The Netherlands) B.V.**

Eindhoven Office

De Witbogt 15, 5652 AG Eindhoven, The Netherlands

TEL: 31-40-2592850 / FAX: 31-40-2592851