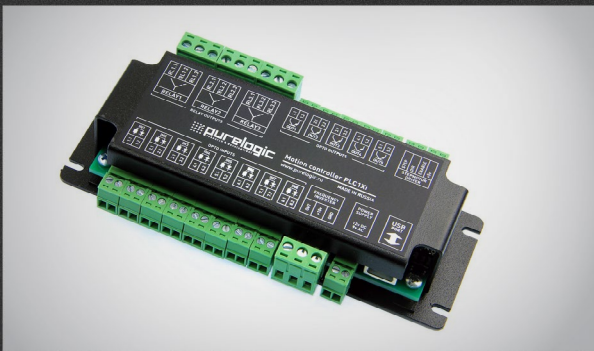


PLC1Xi

Автономный программируемый контроллер движения



РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

01. Общие сведения	2
02. Комплект поставки	2
03. Технические характеристики	3
04. Возможности контроллера	5
05. Подключение	5
06. Конвертор ШИМ > напряжение	8
07. Программирование контроллера	9
08. Описание языка программирования	10
09. Примеры программ для модуля	19
10. Гарантийные обязательства	20



Более подробную информацию по использованию и настройке нашей продукции вы найдете на www.purelogic.ru

01

Общие сведения

PLC1Xi – программируемый автономный контроллер движения (PLC). Контроллер предназначен для управления одним драйвером шагового двигателя (по протоколу STEP/DIR/ENABLE) согласно заданной программе и состоянию управляющих входов. В контроллере реализованы аппаратный разгон/торможение ШД, функции останова/запуска управляющей программы, ручное управление ШД (JOG), управление частотным преобразователем, отслеживание состояний 8 опто-входов, управление 5 оптовыходами и 3 силовыми реле.

02

Комплект поставки

- Контроллер PLC1Xi — 1 шт.
- Руководство по эксплуатации — 1 шт.

Все подключения и изменения режимов работы модуля производить только при отключенном источнике питания.

Запрещается соединение «-» источника питания с заземлением, массой, корпусом и т. д.

Технические характеристики

03

Напряжение питания	12В DC или 9В AC
Максимальный ток потребления	200мА
Интерфейс управления	USB (виртуальный COM-порт, FT232RL)
Размер управляющей программы	2 Кб, максимум
Метод управления драйвером ШД	STEP/DIR/ENABLE
Максимальная частота сигнала STEP	100 кГц
Число входов	8, оптоходы [светодиод + 1кОм, оптопара]
Число выходов	5, оптоходы [открытый коллектор, оптопара, 50В/10мА MAX]
Число силовых выходов, реле	3, перекидной, реле 6А/250В
Параметры конвертера ШИМ > напряжение Управление частотным преобразователем	Uвых=0...9.8В (при изменении скважности Q=0...1) Питание 10В от частотного преобразователя
Сопротивление изоляции	500 МОм
Рабочая температура	0...50 °С
Вес модуля без упаковки	0,3 кг
Габаритные размеры (ШхВхГ)	144 x 30 x 73 мм

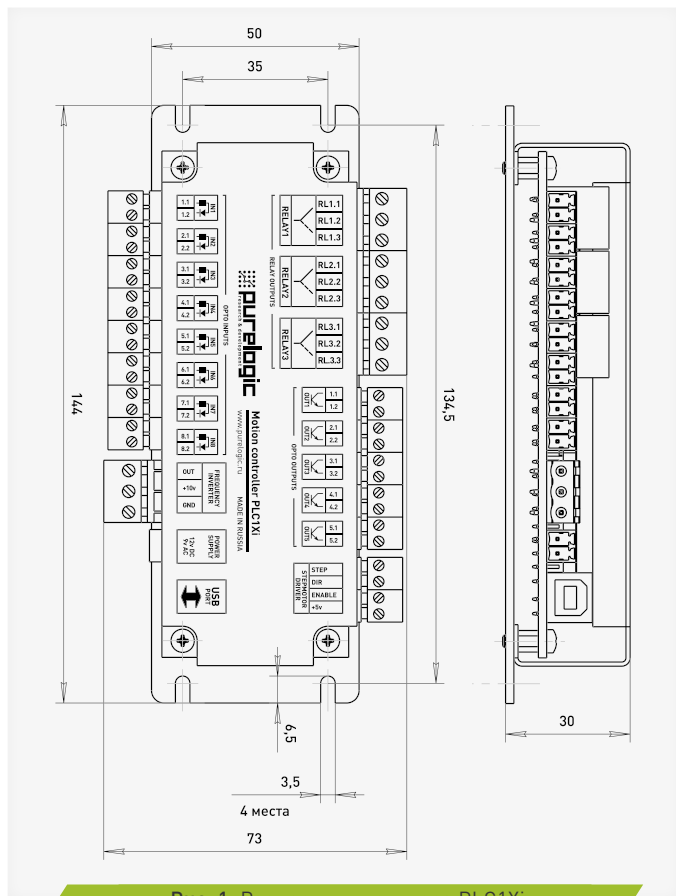


Рис. 1. Размеры контроллера PLC1Xi

Возможности контроллера

04

Контроллер предназначен для управления ШД согласно заданной программе и состоянию управляющих входов. В контроллере реализованы аппаратный разгон/торможение ШД, функции останова/запуска управляющей программы, ручное управление ШД (JOG), управление частотным преобразователем, отслеживание состояний 8 опто-входов, управление 5 оптовыходами.

Контроллер имеет собственный язык программирования (схожий с языком С, поддерживаются операторы циклов/условных переходов/безусловных переходов/работа с переменными и пр.) – это позволяет гибко построить управляющую программу и делает контроллер максимально функциональным.

Программирование/настройка/управление контроллером осуществляется через специальную программу разработки Purelogic R&D или в терминальном режиме через любую стандартную терминальную программу, в качестве интерфейса используется USB (виртуальный COM-порт).

Исполнительная программа хранится в энергонезависимой памяти контроллера. После программирования контроллера ПК не нужен, контроллер может работать автономно.

Контроллер поддерживает удаленное обновление встроенного ПО для последующего расширения функционала.

Подключение

05

Подключение входных сигналов

Контроллер поддерживает работу с 8 независимыми входными сигналами (IN1...IN8). Физически каждый вход представляет собой вход оптопары (светодиод + резистор 1кОм).

При подаче напряжения +5В на оптовход происходит запрограммированное событие. В качестве источника сигнала для оптовхода может служить любой концевой датчик, источник TTL сигнала и пр.

Подключение внешних устройств

Контроллер поддерживает управление 5 независимыми выходными сигналами (OUT1...OUT5). Физически каждый выход представляет собой выход оптопары (открытый коллектор). Пример подключения реле с напряжением срабатывания 12В к выходу представлен на рис. 2.

Контроллер поддерживает управление 3 независимыми силовыми реле для коммутации сильноточной нагрузки (RELAY1...RELAY3).

Управление выходами и реле происходит согласно управляющей программе.

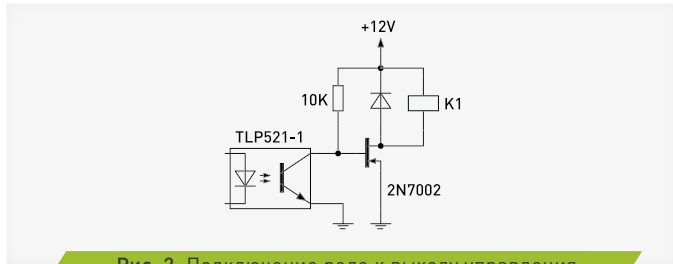


Рис. 2. Подключение реле к выходу управления

Подключение модуля к ПК

Контроллер подключается к любому свободному порту USB ПК. После подключения в системе появляется виртуальный COM-порт (мост FT232RL). При первом подключении возможно понадобится установка драйверов (<http://ftdichip.com>).

На ПК должен быть установлен пакет Microsoft .NET framework версии не ниже 2.0. Он уже присутствует в ОС Windows XP SP3 и в Windows 7 (<http://www.microsoft.com>).

Для управления модулем необходимо скачать программу с сайта Purelogic R&D (www.purelogic.ru).

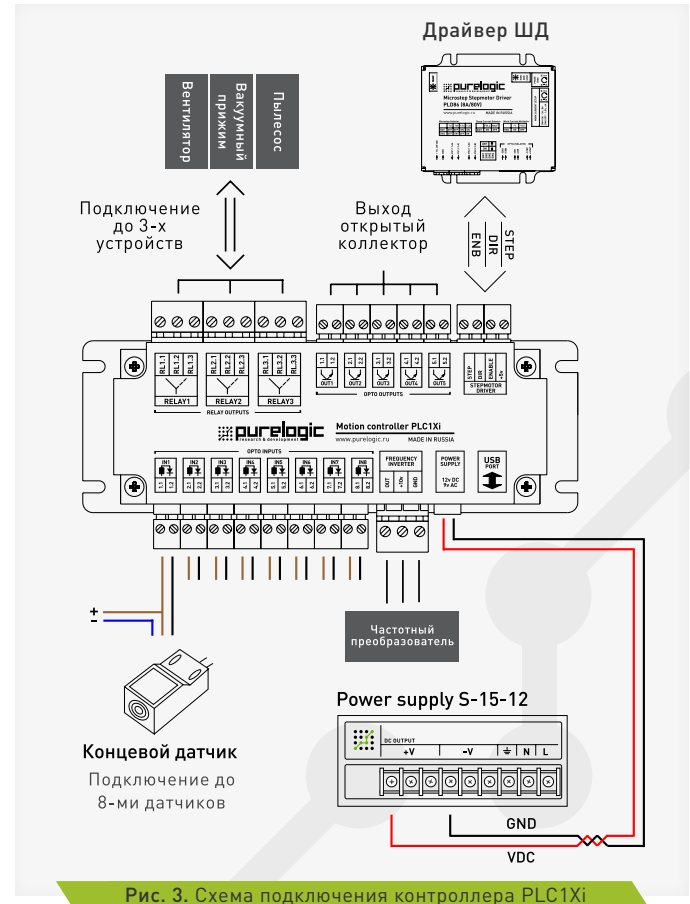


Рис. 3. Схема подключения контроллера PLC1Xi

Подключение источника питания к модулю

В качестве источника питания контроллера может выступать как порт USB (при этом не будут работать силовые реле) так и внешний источник постоянного или переменного напряжения (напряжение согласно TX). Допустимо одновременное подключение порта USB и внешнего источника питания.

Подключение драйвера шагового двигателя

Контроллер поддерживает управление одним драйвером ШД по протоколу STEP/DIR/ENABLE. Драйвер подключается к контроллеру к контактам STEP/DIR/ENABLE по схеме с общим «+», в данном случае это +5В.

На драйвере нужно соединить контакты +STEP / +DIR / +ENABLE и подключить их одним проводом к контакту +5В. Контакты -STEP / -DIR / -ENABLE подключить к STEP / DIR / ENABLE на контроллере.

06

Конвертор ШИМ > напряжение

Контроллер имеет встроенный конвертер ШИМ > напряжение. Конвертер преобразует скважность сигнала управления в напряжение – скважность $Q=0...1$ > напряжение $U=0...10В$.

Конвертер используется для управления частотным преобразователем (ЧП, инвертор), к которому подключен шпиндель (позволяет электронным способом от программы управления ЧПУ изменять обороты шпинделя).

Конвертер оптоизолирован от модуля и питается от ЧП. Стандартно, ЧП имеет 3 контакта подключения конвертера – питание 10В, земля и вход напряжения $0...10В$ (пропорционально которому меняется частота вращения шпинделя).

Управление конвертером происходит согласно управляющей программе.

Программирование контроллера

07

Контроллер построен на базе высокопроизводительного ARM-микропроцессора. Контроллер имеет собственный язык программирования (схожий с языком C, поддерживаются операторы циклов/условных переходов/безусловных переходов/работа с переменными и пр.) – это позволяет гибко построить управляющую программу и делает контроллер максимально функциональным. Используя язык программирования просто реализуются любые алгоритмы сложных движений с контролем внешних событий по датчикам – поиск «0», запуск по событию, останов по событию, смена направления по событию, движение до тех пока не произойдет событие, управление периферией по внешнему событию и пр.

Программирование/настройка/управление контроллером осуществляется через специальную программу разработки Purelogic R&D или в терминальном режиме через любую стандартную терминальную программу, в качестве интерфейса используется USB (виртуальный COM-порт).

Исполнительная программа хранится в энергонезависимой памяти контроллера. После программирования контроллера ПК не нужен, контроллер может работать автономно.

Контроллер поддерживает удаленное обновление встроенного ПО для последующего расширения функционала.

08 Описание языка программирования

Для управления контроллером разработан специальный язык программирования. На этом языке пользователем может быть написана программа, выполняющая желаемую последовательность действий. Интерпретатор поддерживает работу с внутренними переменными, аппаратными входами и выходами, а также условные операторы и переходы по строкам программы.

Интерпретатор языка не чувствителен к регистру написания команд. Команда должна быть записана в одну строку, без переноса на другую строку.

Арифметические и логические операции

Язык программирования не поддерживает различные типы данных. Все переменные, значения входов/выходов и пр. имеют единый тип данных – целое число в диапазоне от -2147483648 до 2147483647. При этом, если значение должно быть интерпретировано как логическое, то число 0 принимается за «ложь», а любое ненулевое число – «истина».

Язык поддерживает следующие математические и логические операторы, перечисленные ниже (в порядке убывания приоритетов вычисления):

<->	Унарный минус. Изменения знака операнда
<!>	Логическое отрицание
<*>, </>	Арифметическое умножение и деление
<+>, <->	Арифметическое сложение и вычитание
<<>, <=>, <>>, <=>=>	Логические операции сравнения
<=>, <!=>	Проверка на равенство/неравенство
<&&>	Побитовое «И»
< >	Побитовое «ИЛИ»
<&&&>	Логическое «И»
< >	Логическое «ИЛИ»

Описанные выше операции является подмножеством операций языка программирования C и переменные имеют тип «int». Результаты вычисления выражений также полностью совпадают с аналогичными выражениями языка C.

Предопределенные переменные

Язык программирования имеет возможность работать с переменными. Переменная – это некоторая область памяти, которая может хранить одно значение (число). Каждая переменная имеет свое имя.

Контроллер оснащен несколькими аппаратными (дискретными) входами и выходами, управлять которыми можно из пользовательской программы. С точки зрения языка программирования аппаратные входы/выходы являются обычными переменными с одним ограничением: если в переменную, связанную с конкретным аппаратным входом произвести запись некоторого значения, то ничего не произойдет. При последующем чтении данных из этой переменной программа получит реальное значение входа (0 или 1), а не записанные туда данные. Интерпретатор не позволяет использовать произвольные имена для переменных.

Контроллер имеет следующие предопределенные переменные:

v1..v9 – Переменные общего назначения. При старте пользовательской программы они имеют неопределенные (случайные) значения. Их можно использовать для любых целей.
inp1..inp8 – Эти 8 переменных «связаны» с аппаратными входами. Если на соответствующем входе установлен логический «0», переменная имеет нулевое значение. В противном случае значение переменной будет неизвестным, но отличным от нуля числом.
 Запись в эту переменную игнорируется.

out1..out5 – Эти 5 переменных «связаны» с аппаратными выходами. При записи ненулевого значения в переменную, на соответствующем выходе появится логическая единица. При записи нуля выход также становится нулевым. Чтение из этой переменной возвращает текущее состояние выхода.

rel1..rel3 – Эти 3 переменных «связаны» с аппаратными релейными выходами. При записи ненулевого значения в

переменную, на соответствующем выходе появится логическая единица. При записи нуля выход также становится нулевым. Чтение из этой переменной возвращает текущее состояние выхода.

PWM1 – Переменная, определяет скважность управляющего ШИМ сигнала. Может принимать значения в диапазоне от 0 до 100 (попытка записи в переменную числа вне диапазона приведет к установке ближайшего значения из разрешенного диапазона). Изменяя значение переменной от 0...100 можно изменять напряжение на клемме OUT в диапазоне 0...10В.

Emerg – Переменная экстренной остановки программы, если одной кнопки ESTOP недостаточно. Переменной можно присвоить любое значение (выражение), когда значение переменной станет ненулевым (т.е. истинным) программа остановится.

Position – текущее положение, выраженное в шагах привода. Этой переменной может быть присвоено любое значение и во время перемещений значение будет автоматически корректироваться. Это позволяет, например, определить длину предыдущего перемещения, если оно было прервано сработавшим датчиком.

Помимо перечисленных переменных, в контроллере существуют также встроенные служебные переменные, которые отвечают за настройку параметров контроллера. Изменять их можно также как и любые другие переменные. Их имена начинаются с символов «P_». Значения этих переменных автоматически сохраняются в ПЗУ одновременно с сохранением программы.

В данной версии контроллера определены следующие параметры:

P_STEPS_MM – кол-во импульсов «STEP» на каждый миллиметр перемещения. Например, если двигатель имеет 200 шагов на оборот (стандартный ШД с угловым шагом 1.8°, например PL57H76), шаг винта 5мм и драйвер ШД настроен на деление шага на 8, то кол-во импульсов должно быть установлено в $(200 \cdot 8) / 5 = 320$.

P_FEED_MAX – максимально разрешенная для данной системы подача в мм/мин. Это значение будет использовано при перемещении, если в команде явно не указана другая скорость подачи. Кроме того, если в команде перемещения будет задана

скорость подачи выше указанной в данном параметре, то она автоматически будет ограничена до *P_FEED_MAX*.

P_FEED_JOG – максимально разрешенная для данной системы подача в мм/мин в режиме ручного управления JOG.

P_ACC_TIME – время в миллисекундах за которое двигатель должен разогнаться до скорости *P_FEED_MAX*. Задаёт профиль ускорения двигателя.

P_BUT_START – задает номер входа, на котором установлена кнопка «старт». 0 – кнопки старт нет.

P_BUT_STOP – задает номер входа, на котором установлена кнопка «стоп». 0 – кнопки стоп нет.

P_BUT_PLUS – задает номер входа, на котором установлена кнопка «ВПЕРЕД» в режиме ручного управления JOG. 0 – кнопки «ВПЕРЕД» нет.

P_BUT_MINUS – задает номер входа, на котором установлена кнопка «НАЗАД» в режиме ручного управления JOG. 0 – кнопки «НАЗАД» нет.

Математические выражения

Выражением называется некоторая последовательность чисел, переменных и математических операций, которую можно вычислить. Для изменения порядка вычисления в выражении могут использоваться традиционные круглые скобки.

Примеры выражений:

$V1 + (5 * V3) - 10 <= 15$ или $!inp1 \&\& \text{inp5}$

Выражения делятся на:

Однократно вычисляемые – выражение заключено в круглые скобки {}, или скобки вообще не используются.

Динамически отслеживаемые – выражение заключено в квадратные скобки [].

Пример однократно вычисляемого выражения:

SET v2, 2

SET v3, 3

SET v1, {v2 + v3}

В данном примере в переменную v1 будет помещен результат сложения v2 и v3 и теперь значение переменной v1 станет равным 5.

Пример динамически отслеживаемого выражения:

```
SET v2, 2
SET v3, 3
SET v1, [v2 + v3]
SET v4, v1
SET v2, 3
SET v5, v1
```

В данном примере в переменную v1 будет помещен не результат сложения v2 и v3, а формула «v2+v3». В четвертой строке примера происходит присваивание переменной v4 значения, хранящегося в v1. И именно в данный момент будет вычислено значение переменной v1 по формуле v2+v3 и результат (5) будет помещен в v4. В дальнейшем мы можем изменить значение переменной v2 (5-я строка) и теперь значение переменной v1 станет равным 6.

Метки

Перед любой командой может быть помещена метка, или, говоря другими словами, «название» данной строки программы. Метка может быть использована для произвольного перехода между строками программы. Метка должна состоять из латинских букв и/или цифр, причем первым символом обязательно должна быть буква или символ «_», а в конце метки должно стоять двоеточие. Примеры правильной метки: «M:», «_Label1:»,

Комментарии

Символ «#» в любом месте строки означает начало комментария. Любые символы после «#» не воспринимаются как команда.

Пример комментирования:

```
SET v2, 2 # пусть v2=2;
```

Команды языка

Команда – MOVE C(<выражение>) L[<выражение >] F[<выражение >] D[<выражение >]

Данная команда выполняет перемещение. Команда имеет 4 аргумента:

C[<выражение>] – выполняется перемещение пока указанное выражение ложно. Выражение всегда интерпретируется как динамическое, т.е. независимо от того, задана ли команда в виде MOVE C([inp2]) или MOVE C(inp2), выражение в скобках будет перепроверяться постоянно в течение выполнения команды и в момент, когда оно станет истинно (говоря другими словами – станет ненулевым, пусть «1») перемещение будет остановлено.

D[<выражение>] – задает направление перемещения. Если выражение истинно, то выход «DIR» устанавливается в «1», если ложно, то «DIR» имеет низкий уровень. Это выражение может быть задано динамически, т.е. если в процессе движения оно изменит свое значение, то произойдет изменение направления вращения двигателя.

L[<выражение>] – выполняется перемещение на расстояние, заданное в сотых долях миллиметра, т.е. 100 единиц это 1 мм. При задании длины перемещения признак динамического выражения игнорируется. Это означает, что даже если длина задана в виде L([v1 * 100 + v4]), указанное выражение будет вычислено только один раз перед началом движения. Выражение может быть и отрицательным. В этом случае направление перемещения будет изменено на обратное.

F[<выражение>] – задает желаемую скорость линейного перемещения в мм/мин. Так же как и в случае с длиной перемещения, признак динамического выражения игнорируется, т.е. F(v1) и F([v1]) полностью идентичны. Скорость подачи будет вычислена один раз перед началом движения, и во время перемещения будет оставаться неизменной. Если задана скорость подачи большая чем параметр P_FEED_MAX, то будет установлена максимальная разрешенная подача равная P_FEED_MAX.

Любой из параметров команды MOVE может быть опущен, тогда вместо него будет подставлено значение по умолчанию. Обязательными являются хотя бы один из параметров S или L. Такое построение команды дает возможность реализовывать например бесконечные перемещения, или бесконечные перемещения пока не произойдет событие и пр.

Примеры использования команды MOVE:

MOVE L(1000) – переместиться «вперед» на 10мм с максимальной подачей;

MOVE L(-100) F(10) – переместиться «назад» на 1мм с подачей 10мм/мин;

MOVE L(-1000) D(0) – переместиться «вперед» на 10мм, максимальная подача;

MOVE L(500) D(!inp1) – переместиться на 5мм, максимальная подача, с динамическим отслеживанием направления по входу inp1;

MOVE C(inp1) D(0) – перемещаться «назад» пока на первом входе не появится «1»;

MOVE C(!inp1) D(1) F(10) L(100) – Подача 10мм/мин, движимся «вперед» пока на первом входе держится высокий уровень, но не более 1мм.

MOVE C(inp1) D(inp2) – перемещаться в направлении, заданном входом IN2 пока на входе IN1 держится низкий уровень. При изменении значения второго входа во время движения, направление перемещения также изменяется.

Команда – WAIT C(<выражение>) T(<выражение >)

Данная команда выполняет ожидание. Команда может иметь 1 или 2 аргумента:

C(<выражение>) – выполняется ожидание пока указанное выражение ложно. Выражение всегда интерпретируется как динамическое, т.е. независимо от того, задана ли команда в виде WAIT C(inp2) или WAIT C(inp2), выражение в скобках будет перепроверяться постоянно в течение выполнения команды и в момент, когда оно станет истинно (говоря другими словами – станет ненулевым) ожидание будет остановлено.

T(<выражение>) – задает желаемое время ожидания в миллисекундах.

Любой из параметров команды WAIT может быть опущен.

Примеры использования команды WAIT:

WAIT T(100) – ждать 0,1 секунды;

WAIT C(inp1) T(3000) – ожидать 3 секунды, либо пока на входе IN1 не появится логическая единица;

Команда – SET <переменная>,<выражение>

Данная команда присваивает переменной заданное выражением значение. Если выражение задано в квадратных скобках, происходит «связывание» переменной с указанной в выражении формулой и при каждом обращении к переменной, ее значение будет пересчитываться.

Примеры использования команды SET:

SET v1, 5 – в переменную v1 помещаем число 5;

SET v2, v1 + 1 – в переменную v2 помещается число, равное значению V1+1;

SET v3, [v1 + 1] – в переменную v3 помещается формула «v1+1»

Команда – ALIAS <переменная>,<новое имя>

Данная команда присваивает переменной еще одно альтернативное имя. Оно не сохраняется в энергонезависимой памяти и действительно только в течение одного сеанса работы. Может быть использовано внутри пользовательской программы для упрощения понимания программы.

Примеры использования команды ALIAS:

ALIAS v1, counter

#пусть переменная v1 называется counter;

LOOP: SET counter, counter + 1

#увеличим счетчик counter на 1;

MOVE L(10)

#двигаемся на 0.1мм;

IF (counter < 10) JMP LOOP:

#переход на метку LOOP: пока counter<10;

Команда – IF (<выражение>) <команда>

Данная команда проверяет условие, заданное выражением в скобках, и, если оно истинно, т.е. не равно нулю, выполняется <команда>. В качестве команды может быть задана любая команда языка.

Примеры использования команды IF:

IF (v1 = 10 && (v2 + v3) < 0) JMP Label1: – если переменная v1 равна 10 и при этом v1 + v3 меньше нуля, перейти на строку с меткой «Label1»;

IF (inp1) IF (10 + v1 > 100) MOVE L(v1 * 100) C([!inp1]) – если на входе IN1 высокий уровень и если (10+v1)>100 то перемещаться на расстояние не более V1 мм пока на входе IN1 не появится «0».

Команда – MEM

Сохраняет текущие значения аппаратных выходов во внутренней памяти. Эти значения будут автоматически восстановлены из памяти при завершении работы программы независимо от причины остановки (кнопка ESTOP или нормальное завершение программы).

Пример использования команды MEM:

Необходимо чтобы после окончания программы выходы №1 и №2 были нулевыми, а №3 и №4 – активными.

В начале программы напишем:

```
set out1, 0
set out2, 0
set out3, 1
set out4, 1
mem
```

После выполнения программы или ее остановки (нормальное завершение программы или кнопка ESTOP) выходы вернуться к состоянию, в котором находились на момент выполнения команды MEM (в этом примере – out1 и out2 станут нулевыми, а out3 и out4 – активными).

Команда – JMP <метка>

Переход на указанную метку в программе. Строка с такой меткой должна существовать в программе иначе выполнение программы будет прервано при попытке выполнения перехода.

Примеры использования команды JMP:

JMP L1: – переход на метку L1:

Команда – ON

Включить драйвер ШД, т. е. установить выход Enable в 1.

Команда – OFF

Выключить драйвер ШД, т. е. установить выход Enable в 0.

Команда – SHOW

Выводит значение указанной переменной из памяти. Если имя переменной не задано – выводит текущую программу из памяти.

SHOW inp1 – выводит значение переменной inp1.

Примеры программ для модуля

09

Программа поиска «Нуля»

На входе IN1 установлен концевой выключатель, с активным низким сигналом.

```
MOVE D(0) C([!inp1])
#Назад с макс. подачей до
концевика, проскакиваем его
MOVE D(1) C([inp1]) F(10)
#Медленно возвращаемся пока не
«сойдем» с датчика.
```

ON

```
LOOP:
MOVE L(100) F(P_FEED_MAX / 10)
MOVE L(-100)
SET COUNTER, COUNTER - 1
IF (COUNTER > 0) JMP LOOP:
OFF
```

```
#Восстановим старое значение
подачи
SET P_FEED_MAX, OLD_FEED
```

Пример 1

```
#Назначим переменным новые
имена
ALIAS V1, OLD_FEED
ALIAS V2, COUNTER
#Сохраним текущее значение
подачи в обычную переменную
SET OLD_FEED, P_FEED_MAX
#теперь можем смело изменить
подачу для данной программы
SET P_FEED_MAX, 100
#Выполним 10 перемещений
вперед и назад
SET COUNTER, 10
```

Пример 2

```
ALIAS v1, x
SET x, 10
lab1: MOVE L(100*x) F(x)
SET x, x-1
IF (x>=0) JMP lab1:
MOVE L(10)
lab2: MOVE L(10) D(0)
```

Пример 3

```
alias v1, x
set x, 5
lab1: MOVE L(100*x) F(x) D(x & 1)
SET x, x-1
IF (x>0) JMP lab1:
```

Пример 4

```
alias v1,x
alias v2,y
set x,5
SET y,[x * 10]
lab1: MOVE L(y) F(x * 10) D(![x & 1])
SET x,[x-1]
IF [x>0] JMP lab1:
```

Пример 5

```
SET Emerg, [inp3 && [inp2 || inp4]]
Аварийное завершение программы
произойдет если станет активным
вход №3 одновременно с любым из
выходов №2 или №4.
```

ремонт.

4. Гарантия не распространяется на стекло, электролампы, стартеры и расходные материалы, а также на:

- 4.1. Товар с повреждениями, вызванными ненадлежащими условиями транспортировки и хранения, неправильным подключением, эксплуатацией в штатном режиме либо в условиях, не предусмотренных производителем (в т.ч. при температуре и влажности за пределами рекомендованного диапазона), имеющий повреждения вследствие действия сторонних обстоятельств (скачков напряжения электропитания, стихийных бедствий и т.д.), а также имеющий механические и тепловые повреждения.
- 4.2. Товар со следами воздействия и (или) попадания внутрь посторонних предметов, веществ (в том числе пыли), жидкостей, насекомых, а также имеющим посторонние надписи.
- 4.3. Товар со следами несанкционированного вмешательства и (или) ремонта (следы вскрытия, кустарная пайка, следы замены элементов и т.п.).
- 4.4. Товар, имеющий средства самодиагностики, свидетельствующие о ненадлежащих условиях эксплуатации.
- 4.5. Технически сложный Товар, в отношении которого монтажно-сборочные и пуско-наладочные работы были выполнены не специалистами Продавца или рекомендованными им организациями, за исключением случаев прямо предусмотренных документацией на товар.
- 4.6. Товар, эксплуатация которого осуществлялась в условиях, когда электропитание не соответствовало требованиям производителя, а также при отсутствии устройств электрозащиты сети и оборудования.
- 4.7. Товар, который был перепродан первоначальным покупателем третьим лицам.
- 4.8. Товар, получивший дефекты, возникшие в результате использования некачественных или выработавших свой ресурс запасных частей, расходных материалов, принадлежностей, а также в случае использования не рекомендованных изготовителем запасных частей, расходных материалов, принадлежностей.

10

Гарантийные обязательства

Гарантийный срок службы составляет 12 месяцев со дня приобретения. Гарантия сохраняется только при соблюдении условий эксплуатации и регламентного обслуживания.

1. Общие положения

- 1.1. В случае приобретения товара в виде комплектующих Продавец гарантирует работоспособность каждой из комплектующих в отдельности, но не несет ответственности за качество их совместной работы (неправильный подбор комплектующих). В случае возникновения вопросов Вы можете обратиться за технической консультацией к специалистам компании).
- 1.2. Продавец не предоставляет гарантии на совместимость приобретаемого товара и товара имеющегося у Покупателя, либо приобретенного им у третьих лиц.
- 1.3. Характеристики изделия и комплектация могут изменяться производителем без предварительного уведомления в связи с постоянным техническим совершенствованием продукции.

2. Условия принятия товара на гарантийное обслуживание

- 2.1. Товар принимается на гарантийное обслуживание в той же комплектности, в которой он был приобретен.

3. Порядок осуществления гарантийного обслуживания

- 3.1. Гарантийное обслуживание осуществляется путем тестирования (проверки) заявленной неисправности товара.
- 3.2. При подтверждении неисправности проводится гарантийный

Изготовлен и принят в соответствии с обязательными требованиями действующей технической документации и признан годным для эксплуатации.

№ партии:

ОТК:





Обращаем Ваше внимание на то, что в документации возможны изменения в связи с постоянным техническим совершенствованием продукции. Последние версии Вы всегда можете скачать на нашем сайте www.purelogic.ru



www.purelogic.ru

Контакты

 +7 (495) 505-63-74 - Москва
+7 (473) 204-51-56 - Воронеж

 394033, Россия, г. Воронеж,
Ленинский пр-т, 160,
офис 135

 Пн-Чт: 8.00–17.00
Пт: 8.00–16.00
Перерыв: 12.30–13.30

 sales@purelogic.ru