ОАО «БЕЛЭЛЕКТРОМОНТАЖНАЛАДКА»

СЕРИЯ МАЛОСИГНАЛЬНЫХ ЛОГИЧЕСКИХ КОНТРОЛЛЕРОВ

«ПИКОН-МИКРО»

РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

ПШИЖ 150.00.00.00.002 РЭ

РЕСПУБЛИКА БЕЛАРУСЬ
220101, г. Минск, ул. Плеханова 105А,
т./ф. (017) 368-09-05, 367-86-56, 368-88-57
www.bemn.by, upr@bemn.by

СОДЕРЖАНИЕ

Настоящее руководство по эксплуатации предназначено для ознакомления с принципом действия и техническими характеристиками малосигнальных логических контроллеров серии «ПИКОН-МИКРО» (далее – контроллеров).

Руководство по эксплуатации (далее – РЭ) предназначено для эксплуатационного персонала и инженеров-проектировщиков автоматизированных систем управления технологическими процессами (АСУ ТП).

1 ОПИСАНИЕ И РАБОТА КОНТРОЛЛЕРА

1.1 Назначение и основные функции контроллера

1.1.1 Настоящее РЭ распространяется на малосигнальные логические контроллеры серии «ПИКОН-МИКРО» (МЛК-10, МЛК-11, МЛК-12 и МЛК-13), предназначенные для решения задач контроля и управления небольшими объектами в локальных и распределенных системах АСУ ТП, а также в качестве автономно функционирующих устройств.

1.1.2 Основные функции контроллера следующие:

- сбор информации с датчиков дискретных и аналоговых сигналов и ее первичная обработка;
- самоконтроль и диагностика всех устройств контроллера, вывод информации о техническом состоянии контроллера обслуживающему персоналу;
- выдача управляющих воздействий на исполнительные органы различных типов;
- передача по запросу предварительно обработанной информации, через канал связи в пункт управления (ПУ);
- выполнение команд ПУ;
- возможность автономной работы по запрограммированному алгоритму (логической программе).

1.1.3 Контроллер предназначен для эксплуатации в условиях воздействия:

- температуры окружающего воздуха по группе исполнения С2 ГОСТ 12997-84 (от минус 40 до +70 ºС);
- относительной влажности окружающего воздуха по группе исполнения С3 ГОСТ 12997-84 (до 95 % при 35 ºС и более низких температурах, без конденсации влаги);
- атмосферного давления, соответствующего группе исполнения Р1 по ГОСТ 12997-84 (от 84 до 106,7 кПа).

1.1.4 Контроллер может использоваться:

- как автономное устройство управления небольшими объектами;
- как удаленный терминал связи с объектом в составе распределенных систем управления;

– одновременно как локальное устройство управления и как удаленный терминал связи с объектом в составе сложных распределенных систем управления.

1.2 Технические характеристики

1.2.1 Основные технические характеристики контроллера приведены в таблице 1.

Таблица 1

Наименование характеристики	**Значение (свойства)**			
	МЛК-10	**МЛК-11**	**МЛК-12**	**МЛК-13**
Габаритные размеры (без клемм), мм	109,5×100×84,5			
Масса, кг	0,57	0,42	0,40	0,52
Характеристики электропитания: а) напряжение питания; б) частота переменного тока	~(230±23) В 50 Гц	~(230±23) В =(220±22) В 50 Гц		
Мощность, потребляемая от сети, Вт	не более 7			
Протокол обмена	МР-СЕТЬ (аналог MODBUS с режимом передачи RTU)			
Степень защиты по ГОСТ 14254-96: а) корпуса контроллера б) клеммных разъемов	 IP30 IP00			
Температура и относительная влажность воздуха рабочих условий эксплуатации	от минус 40 до + 70 ºС до 95 % при 35 ºС и более низких температурах			
Требования к надежности: а) средняя наработка на отказ; б) среднее время восстановления работоспособности; в) средний срок службы	 не менее 30000 ч; не более 0,5 ч; не менее 15 лет			

1.2.2 Технические характеристики составных частей контроллера приведены в таблицах 2 – 7.

Таблица 2 – Характеристики центрального процессора

Наименование характеристики	**Значение (свойства)**
Объем энергонезависимой памяти программ пользователя центрального процессора	не менее 8000 команд
Наличие часов реального времени	имеются (энергонезависимые, с встроенным литиевым элементом питания)
Наличие сторожевого таймера	имеется

Таблица 3 – Характеристики интерфейса связи

Наименование характеристики	**Значение (свойства)**
Тип интерфейса	RS-485 (изолированный)
Скорость передачи	от 600 до 115200 бит/с
Максимальная длина линии связи	1200 м (зависит от скорости и типа кабеля)
Тип соединения	витая пара
Максимальное количество устройств на шине	32
Протокол связи	МР-СЕТЬ

Таблица 4 – Характеристики дополнительного интерфейса связи

Наименование характеристики	**Значение (свойства)**
Тип интерфейса	USB
Гальваническая изоляция	1000 В
Скорость передачи по интерфейсу USB	USB спецификация 2.0 (12 Mbps)

Таблица 5 – Характеристики дискретных входов

Наименование характеристики	**Значение (свойства)**			
	МЛК-10	**МЛК-11**	**МЛК-12**	**МЛК-13**
Количество входов	8	8	12	12
Номинальное входное напряжение	~230 В, =220 В			
Номинальный входной ток	1 мА			
Напряжение срабатывания на постоянном токе	от 115 до 140 В			
Коэффициент возврата на постоянном токе	не менее 0,85			
Напряжение срабатывания на переменном токе	от 120 до 140 В			
Коэффициент возврата на переменном токе	не менее 0,7			

Таблица 6 – Характеристики релейных выходов

Наименование характеристики	**Значение (свойства)**			
	МЛК-10	**МЛК-11**	**МЛК-12**	**МЛК-13**
Количество релейных выходов	7	7	3	3
Коммутируемые сигналы (активная нагрузка): а) на постоянном токе; б) на переменном токе	220 В; 0,4 А 230 В; 8,0 А			
Тип контакта	нормально-замкнутый, нормально-разомкнутый[1)] или переключающий			

Наименование характеристики	Значение (свойства)			
	МЛК-10	**МЛК-11**	**МЛК-12**	**МЛК-13**
Количество циклов переключения	$16 \cdot 10^5$			
[1)] При заказе выбирается тип контакта: нормально-замкнутый или нормально-разомкнутый				

Таблица 7 – Характеристики аналоговых входов

Наименование характеристики	Значение (свойства)
Количество аналоговых входов	1
Диапазоны измерения напряжения переменного (постоянного) тока*	от 0 до 300 В
Основная приведенная погрешность	±1 %
Входное сопротивление	не менее 1 Мом
* по заказу	

1.3 Устройство и работа

1.3.1 Конструктивно контроллер выполнен в пластмассовом корпусе и устанавливается на DIN-рейку 35 мм. Внешний вид контроллера представлен на рисунках 1.1 и 1.2.

1.3.2 Изделие состоит из следующих узлов, выполненных на четырех печатных платах и устанавливаемых внутри корпуса контроллера:

- узел центрального процессора (ЦП);
- узел ввода-вывода сигналов с блоком питания.

1.3.3 Контроллер имеет выход интерфейса RS-485 для связи с модулями расширения и верхним уровнем АСУ ТП, а также выход интерфейса USB для подключения пульта оператора или ПЭВМ. При использовании интерфейса RS-485 можно организовать локальную сеть, в которую может быть подключено до 31 контроллера.

1.3.4 ЦП обеспечивает реализацию алгоритма функционирования контроллера, осуществляет программное управление системой, проводит тестирование всех устройств контроллера и обработку поступающих данных, ведет журнал системы. ЦП имеет в своем составе сторожевой таймер, часы реального времени и энергонезависимую память. Сторожевой таймер предотвращает зависание процессора и перезагружает систему в случае сбоя.

Рисунок 1.1 – Внешний вид передней панели МЛК-10

Рисунок 1.2 – Внешний вид передней панели МЛК-13

1.3.5 Контроллер имеет дискретные входы и релейные выходы в соответствии с таблицами 5 и 6 соответственно, а также 1 аналоговый вход.

1.3.6 Дискретные и аналоговые входы обеспечивают ввод, гальваническую изоляцию, определение состояния и преобразование соответственно дискретных и аналоговых сигналов контролируемого объекта.

1.3.7 Релейные выходы обеспечивают выдачу сигналов управления объектом на исполнительные устройства.

1.3.8 Блок питания, расположенный на печатной плате узла ввода-вывода сигналов, обеспечивает питание контроллера от напряжения переменного тока 230 В (частотой 50 Гц) или напряжения постоянного тока 220 В (для МЛК-11 – МЛК-13).

1.3.9 На лицевой панели контроллера имеются:

- светодиодный индикатор РАБОТА/АВАРИЯ, который отображает состояние работоспособности контроллера;
- два свободно-программируемых светодиодных индикатора, доступных для логической программы пользователя.

1.3.10 На корпусе контроллера расположены клеммные разъемы (колодки), предназначенные для подключения внешних цепей (цепей питания, интерфейса и ввода/вывода).

1.4 Маркировка

1.4.1 На контроллер нанесена маркировка, содержащая следующие данные:

- наименование изделия;
- порядковый номер изделия по системе нумерации изготовителя;
- дата изготовления.

1.5 Упаковка

1.5.1 Упаковка контроллера производится в картонные коробки в соответствии с конструкторской документацией.

2 ИСПОЛЬЗОВАНИЕ ПО НАЗНАЧЕНИЮ

2.1 Требования к месту установки

2.1.1 Помещение (сооружение), где устанавливается контроллер, должно быть закрытым взрывобезопасным и пожаробезопасным. Должны соблюдаться следующие условия:

- климатические и механические внешние воздействующие факторы в соответствии с таблицей 1 настоящего руководства;
- окружающая среда не должна содержать агрессивных паров и газов.

2.2 Подготовка контроллера к использованию

2.2.1 Перед началом работ с контроллером следует внимательно ознакомиться с данным руководством по эксплуатации и изучить назначение разъемов контроллера.

2.2.2 Монтаж, наладка и эксплуатация контроллера должны выполняться в соответствии с требованиями ГОСТ 12.2.007.0-75, ТКП 181-2009 и ТКП 339-2011.

2.2.3 При внешнем осмотре необходимо убедиться в целостности контроллера, отсутствии видимых повреждений и дефектов, наличии маркировки.

2.2.4 Контроллер размещается на объекте и подключается к внешним сигналам в соответствии с проектом АСУ ТП.

2.2.5 Контроллер должен быть жестко закреплен на базовой поверхности.

2.2.6 Габаритно-присоединительные размеры и схема подключения контроллера приведены в приложении А.

2.2.7 Присоединение цепей осуществляется с помощью клеммных винтовых разъемов диаметром 4 мм для проводов сечением до 2,5 мм^2 согласно проекту автоматизации в виде кабельных связей и жгутов вторичной коммутации. Концы провода для подключения к клеммным винтовым разъемам требуется зачистить на 10 мм. Прокладка кабелей и жгутов должна отвечать требованиям ТКП 339-2011.

2.2.8 Необходимость в экранировании входных, выходных кабельных цепей и линий связи определяется при проектировании и зависит от длины кабелей и от уровня помех в зоне прокладки кабеля.

2.2.9 Провода электропитания подключаются к контактам 1 и 2 разъема «~230В».

2.2.10 Пуско-наладочные работы по программированию конфигурации контроллера, проверке работоспособности и проверке взаимодействия с внешними устройствами осуществляются на месте установки.

ВНИМАНИЕ!!! При демонтаже корпуса запрещается касаться установленных на платах контроллера элементов, т.к. изделие содержит компоненты, чувствительные к статическому электричеству.

2.2.11 Пример установки (демонтажа) контроллера на DIN-рейку приведен на рисунке Б.1 (Приложение Б)

2.3 Использование контроллера

2.3.1 Контроллер функционирует в автоматическом режиме, не требующем вмешательства оператора.

2.4 Программирование контроллера

2.4.1 Подключение контроллера к персональному компьютеру

2.4.1.1 Подключение производится с одной стороны, к свободному COM-порту компьютера, с другой – к разъему малосигнального логического контроллера серии «ПИКОН-МИКРО».

ВНИМАНИЕ!!! Подключение должно производиться при включенном питании контроллера и персонального компьютера.

2.4.2 Настройка программы «ПЛК КОНФИГУРАТОР»

2.4.2.1 Перед началом работы необходимо настроить COM-порт компьютера. Для этого следует выбрать пункт «Порта» в меню «Настройки» (рисунок 2.1) или значок  на панели инструментов/

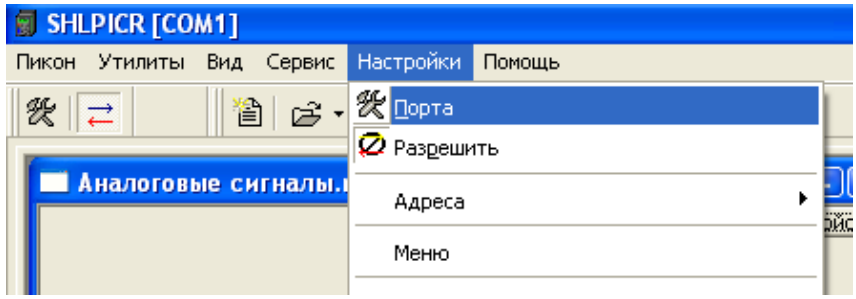


Рисунок 2.1 – Меню «Настройки»

2.4.2.2 В открывшемся окне «Настройки порта» (рисунок 2.2) выберите настройки, необходимые для установления связи модуля центрального процессора (МЦП) с компьютером. Далее нажмите «Применить» и «ОК».

2.4.2.3 Для работы с контроллерами серии «ПИКОН-МИКРО» рекомендуется установить следующие настройки:

- порт – порт компьютера, к которому подключен контроллер;
- протокол – MODBUS;
- тип – без RTS;
- скорость – 115200 бит/с;
- ожидание ответа –500 мс;
- ожидание байта – 50 мс;
- включение передачи – 0 мс;
- выключение передачи – 0 мс;
- тайм-аут ввода/вывода – 0;
- тип паритета – нет;
- длина символов – 8;
- стоп бит – 1;
- прослушка – неактивно.

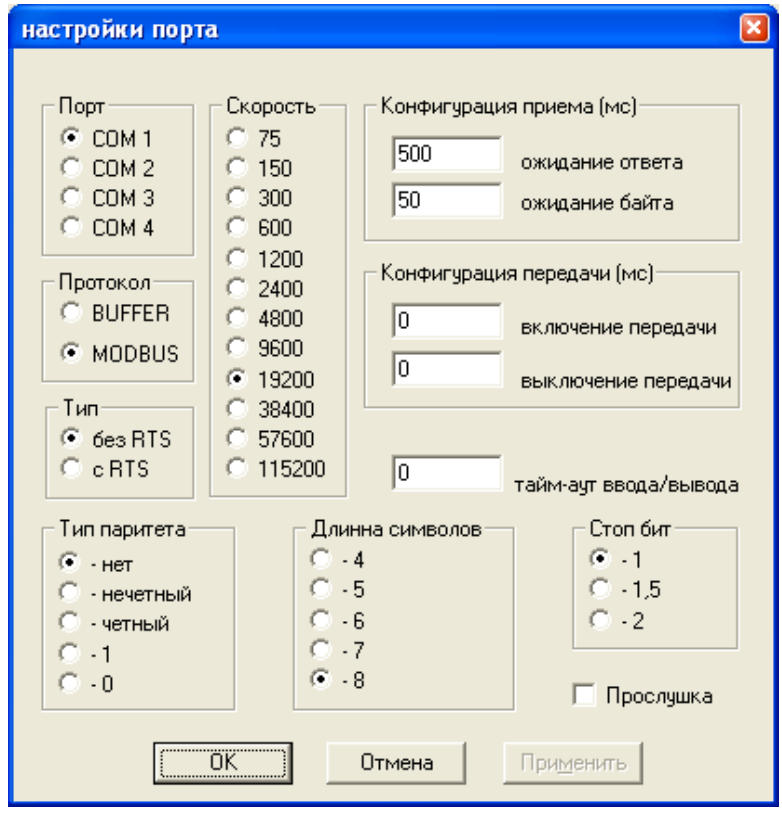


Рисунок 2.2 – Окно настройки порта

2.4.2.4 При выборе пункта «Разрешить» в меню «Настройки» разрешаются обмены данными. На панели инструментов данный пункт выглядит следующим образом:

 – разрешение обменов;

 – запрет обменов.

2.4.3 Программирование МЛК

2.4.3.1 Для того, чтобы запрограммировать контроллер, необходимо в меню «СЕРВИС» выбрать пункт «Записать/прочитать» (рисунок 2.3).

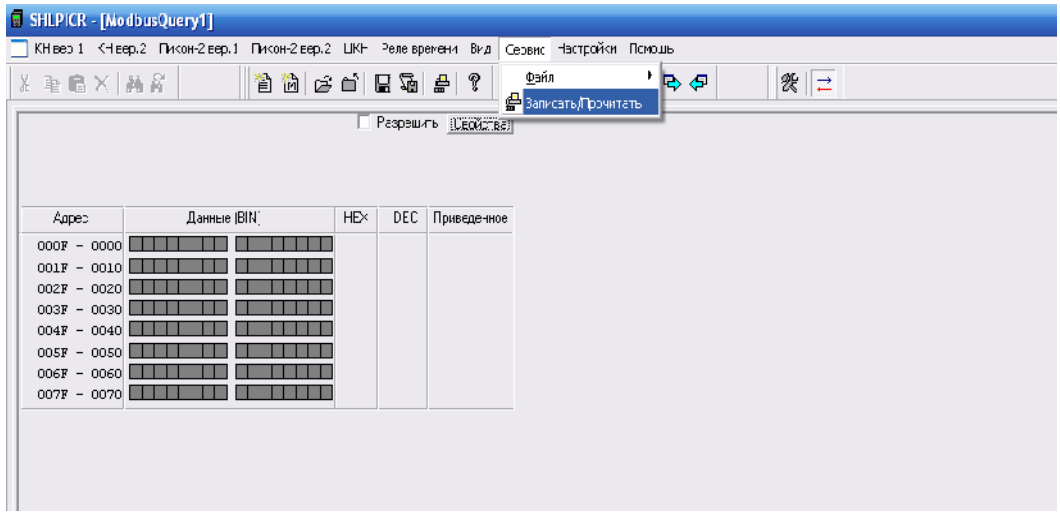


Рисунок 2.3

2.4.3.2 Далее откроется окно для программирования контроллера, которое выглядит следующим образом (рисунок 2.4).

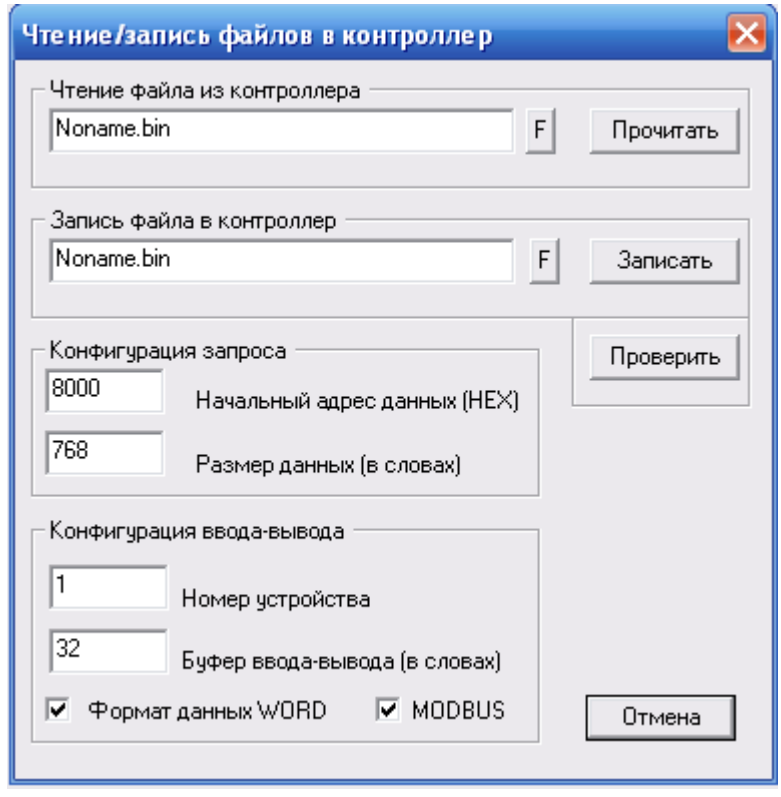


Рисунок 2.4

2.4.3.3 Для записи файла необходимо нажать на кнопку «F», и выбрать из списка файл логической программы в формате «*.bin», затем ввести начальный адрес «0×8000». В полях «Размер данных» необходимо указать примерный размер логической программы (файла *.bin) (лучше указывать больший размер, а далее программа сама скорректирует его в меньшую сторону), «Номер устройства» – 1 (или иной, если номер устройства был изменен), «Буфер ввода-вывода данных» – 32, а также выбрать «Формат данных WORD» и «MODBUS». После этого нажать на кнопку «Записать». Начнется запись и откроется прогрессбар (рисунок 2.5).

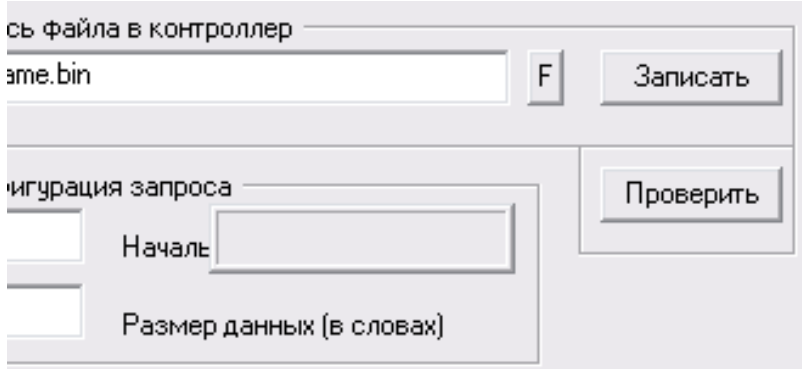


Рисунок 2.5

2.4.3.4 По мере выполнения записи программы прогрессбар будет заполнятся. По окончанию записи программы прогрессбар закроется. Для верификации данных рекомендуется так же нажать кнопку «Проверить». В этом случае также откроется прогрессбар, отображающий процесс выполнение верификации. Если ошибок нет прогрессбар закроется, если же произошли ошибки при записи программы, будет выведено окно с сообщением об ошибке – «Данные не верны» (рисунок 2.6).

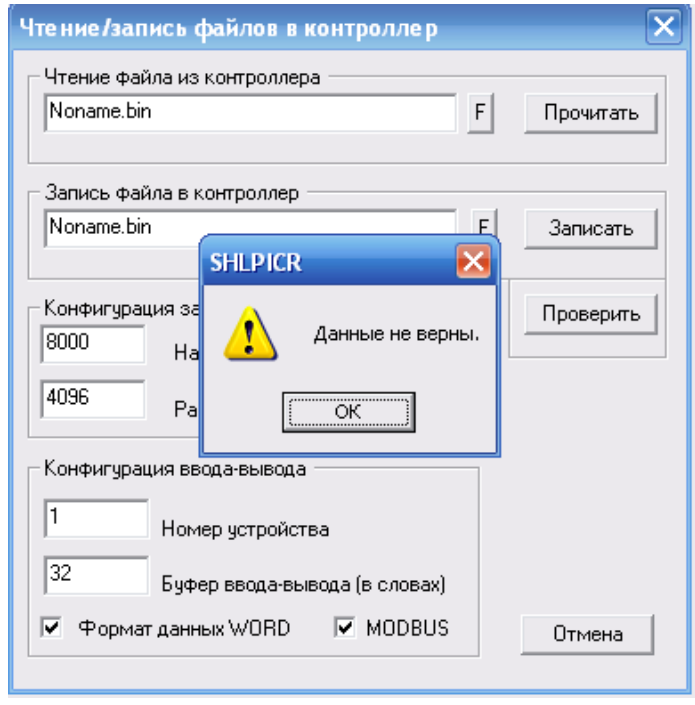


Рисунок 2.6

2.4.3.5 По выполнению предыдущих пунктов логическая программа будет записана в контроллер. Для запуска логической программы, необходимо зайти по адресу 0×840 (Word) и записать значение «209». Для этого открываем в меню «СЕРВИС» → «Файл» → «Новый Modbus».

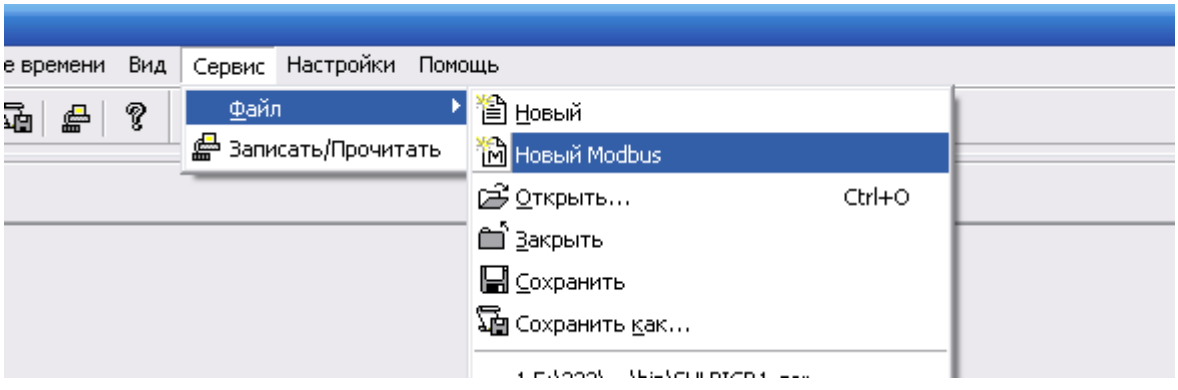


Рисунок 2.7

В появившемся окне ставим галочку «Разрешить», тем самым разрешая запросы к контроллеру.

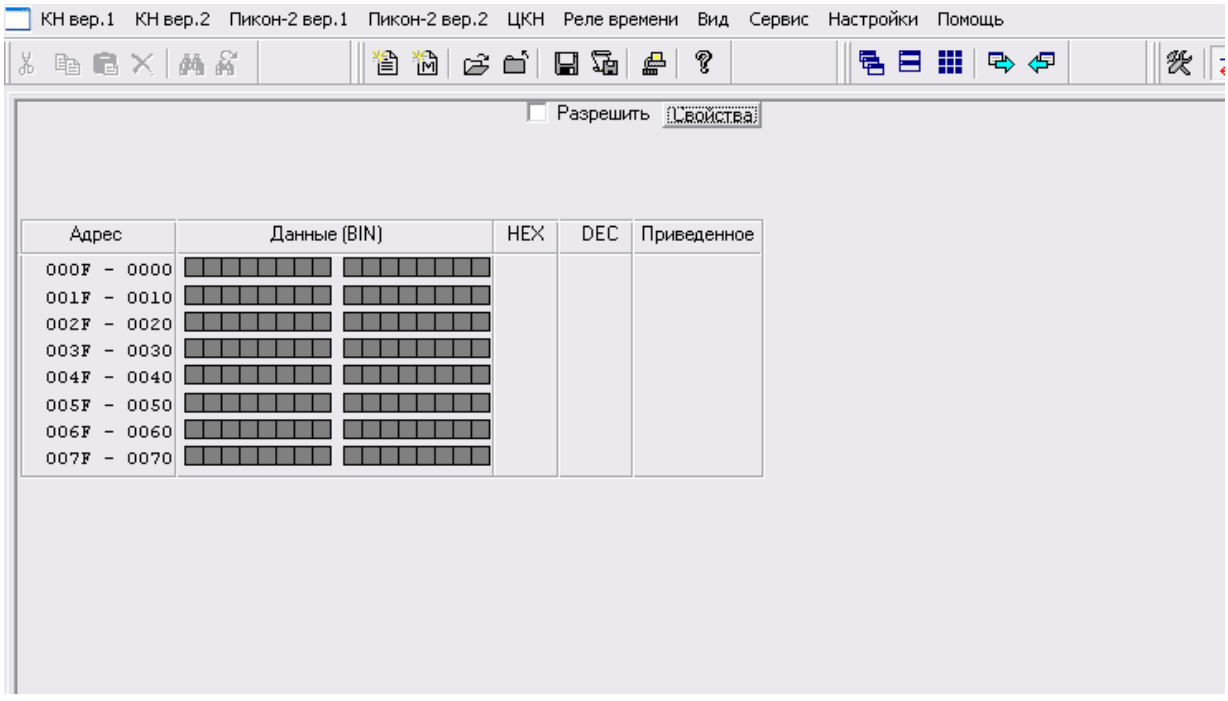


Рисунок 2.8

Далее выбираем «Свойства» и появляется окно свойств (рисунок 2.9), где: «Номер устройства» – 1 (в случае если не был изменен), «Базовый адрес» – 840, «Команды» – Словные, также выбираем «Формат данных WORD» и нажимаем «Ввод». Напротив адреса «0×840» кликаем в столбце «dec» и в появившемся окне ввода значения записываем «209», нажимаем «ENTER».

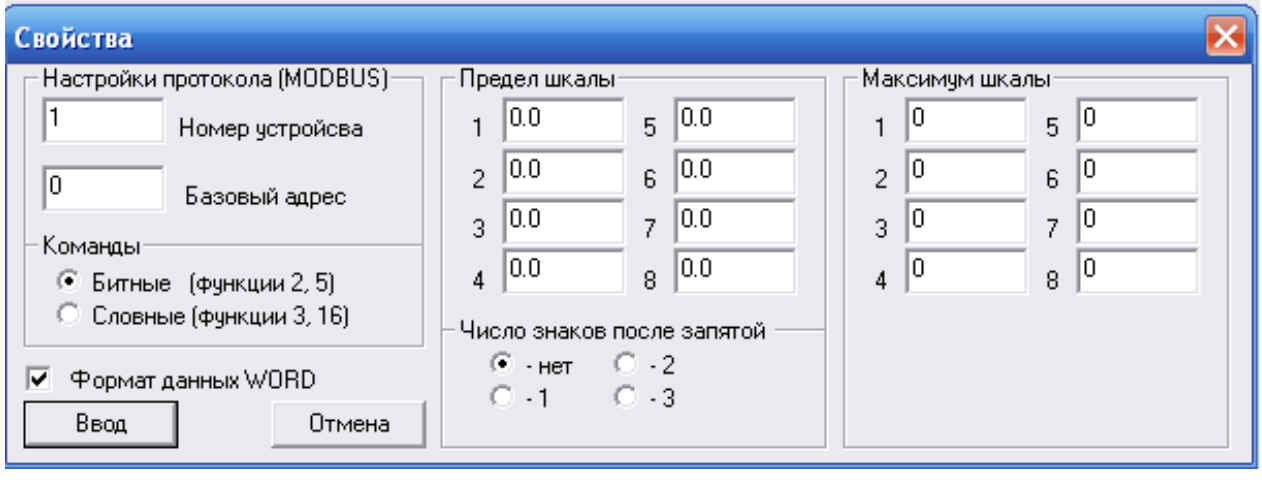


Рисунок 2.9

Затем переходим по адресу «0×1600» (Word), для этого в свойствах изменяем базовый адрес на «1600», проверяем значение по этому адресу. Если значение не равно «209», записываем его вручную.

2.4.3.6 Регистр управления логикой SCLP (0840h):

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

где:

0 – разрешение работы логики;

1 – запуск программы. При установленном бите разрешается работа, при сброшенном - выполнение программы приостанавливается;

2 – перезапуск программы. При установленном бите осуществляется перезапуск логической программы;

3 – запись логической программы. Если данный бит установлен, то производилась перезапись логической программы. Бит 1 сбрасывается автоматически;

4 – отладка логической программы. Если бит установлен, то выполнение логической программы осуществляется с учётом битов 5 и 6;

5 – пошаговая отладка. При установке бита выполняется одна команда. Бит сбрасывается автоматически;

6 – отладка до точки останова. Адрес точки останова записан в регистре DEBUG (0842h).

Для простого запуска необходимо установить биты 0 и 1.

Для перезапуска программы необходимо установить 0 и 2.

2.4.3.7 Отладка включается как до выполнения программы, так и в течение выполнения.

2.4.4 Описание языка программирования логической задачи

2.4.4.1 Объявление переменных

[имя переменной] [адрес переменной] **:** [тип переменной] [инициализационное значение]**;**

2.4.4.2 Типы данных

2.4.4.2.1 В компиляторе реализованы не все типы которые поддерживаются в стандарте IEC 61131-3:2013. В таблице 2.1 приведены типы которые поддерживаются в компиляторе.

Таблица 2.1 – Типы данных

Ключевое слово	Размер в битах	Знак	Краткое описание
BOOL	1	Нет	Битовая переменная
SINT	8	Есть	Короткое знаковое целое число
INT	16	Есть	Знаковое целое число
DINT	32	Есть	Двойное знаковое целое число
LINT	64	Есть	Длинное знаковое целое число
REAL	32	Есть	Число с плавающей запятой (дробное, знаковое)
LREAL	64	Есть	Двойное число с плавающей запятой (дробное, знаковое)
STRING	8	Нет	Строка (байтовая кодировка символа) ASCII
WSTRING	16	Нет	Двойная строка (словная кодировка символа) Unicod
BYTE	8	Нет	Битовая строка длинной 8 бит (беззнаковое, целое)
WORD	16	Нет	Битовая строка длинной 16 бит (беззнаковое, целое)
DWORD	32	Нет	Битовая строка длинной 32 бит (беззнаковое, целое)
LWORD	64	Нет	Битовая строка длинной 64 бит (беззнаковое, целое)
DATE	32	Нет	Дата (количество дней от начала века)
TIME_OF_DAY	32	Нет	Время в течении дня (количество ms от начала дня)
DATE_AND_TIME	64	Нет	Дата и время состоит из предыдущих 2-х типов
TIME	32	Есть	Длительность времени (знаковое)

2.4.4.3 Числовые литералы

2.4.4.3.1 Фора РБНФ для определения числа

Правила формирования числа:

digit :: = ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’|‘8’|‘9’
octal_digit ::= ‘0’|‘1’|‘2’|‘3’|‘4’|‘5’|‘6’|‘7’
hex_digit ::= digit|‘A’|‘B’|‘C’|‘D’|‘E’|‘F’

Числовые литералы (numeric_literals) состоят из целых и вещественных литаералов.

2.4.4.3.2 Типы знаковых целых литералов

Integer_literal ::= [integer_type_name ‘#’] (signed_integer)
Integer_literal ::= integer_type_name ‘#’ (binary_integer | octal_integer | hex_integer)
signed_integer::= [‘+’ | ‘-’] integer
integer ::= digit {[_] digit}
binary_integer ::= ‘2#’ bit{[‘_’]bit}
bit ::= ‘1’ | ‘0’
octal_integer ::= ‘8#’ octal_digit {[‘_’]octal_digit }
hex_integer ::= ‘8#’ hex_digit {[‘_’]hex_digit }
integer_type_name ::= ::= ‘SINT’|‘INT’|‘DINT’|‘LINT’
Типыданных: SINT, INT, DINT,LINT, представление знакового целого числа.
SINT [-126…127]
INT [-32766…32767]
DINT [-2 147 483 646…2 147 483 647]
LINT [-9 223 372 036 854 775 806 … 9 223 372 036 854 775 806]

В компиляторе реализовано автоматическое преобразование типа внутри подтипа. Например, компилятор автоматически не преобразует число WORD в INT, несмотря на то, что данная функция реализована во многих компиляторах высокого уровня. Однако, если число не специализированно (без префикса типа), то число преобразуется к типу переменной, с которой число используется.

Например, при присваивании переменной типа INTVar1 := 15; число преобразуется к типу INTавтоматически. Однако, если записать такое же число указав префикс основание числа: Var1 := 8#17 – будет выдана ошибка, так как число с основанием 8 или 2 или 16 автоматически преобразуется к целому беззнаковому. Будет ошибочно, если попытаться присвоить число с другим префиксом типа, например: Var1 := LINT#15;. Чтобы преобразовать число одного типа к другому нужно использовать функции преобразования. Осуществляется проверка соответствие величины числа разрешенному диапазону.

2.4.4.3.3 Типы беззнаковых целых литералов

Bit_string_literal ::= bit_string_type_name ‘#’ (unsigned_integer)
Bit_string_literal ::= [bit_string _type_name ‘#’] (binary_integer | octal_integer | hex_integer)
bit_string_type_name ::= ‘BYTE’|‘WORD’|‘DWORD’ | ‘LWORD’

Типы данных: BYTE, WORD, DWORD, LWORD, представление знакового целого числа.

BYTE [0…256] или [0…FF] hex
WORD [0…65535] или [0…FFFF] hex
DWORD [0…32767] или [0…FFFF FFFF] hex
LWORD [0 … 18 446 744 073 709 551 615] или [0…FFFF FFFFFFFF FFFF] hex

Если число не специализированно (без префикса типа), то число преобразуется к типу переменной, с которой число используется. Например, при присваивании переменной типа WORD Var1 := 8#15; число преобразуется к типу WORD автоматически. Однако, если записать такое же число с без префикса основания: Var1 := 17 – будет выдана ошибка, так как число автоматически преобразуется к знаковому. Также будет ошибкой если попытаться присвоить число с другим префиксом типа, например: Var1 := LWORD#15;.

2.4.4.3.4 Типы дробных литералов

real_literal ::= [real_type_name '#'] signed_integer '.' integer [exponent]
exponent ::= ('e' | 'E') ['+' | '-'] integer
Bit_string_literal ::= [bit_string _type_name '#'] (binary_integer | octal_integer | hex_integer)
bit_string_type_name ::= 'BYTE'|'WORD'|'DWORD' | 'LWORD'

Типы данных: REAL, LREAL, представление знакового дробного числа.

В компиляторе реализована проверка на + и – бесконечность. Если число невозможно преобразовать и оно слишком большое, то будет выдана ошибка об этом. Также реализовано автоматическое преобразование типа. Если в коде встречается число с точкой, то оно будет распознано как дробное. Таким образом разрешено присваивание переменной REAL Var2 := 12.5. Автоматическое преобразование дробного числа с разными префиксами типа не предусмотрено. ОшибкаLREAL Var1 := REAL#12.3.

2.4.4.3.5 Булевые литералы

Boolean_literal ::= (['BOOL#']('1' | '0')) | 'TRUE' | 'FALSE'

2.4.4.4 Временные литералы

time_literal ::= duration | time_of_day | date | date_and_time

Временные литералы делятся на две группы: длительности времени и времени дня и времени.

2.4.4.4.1 Длительность времени

Данный тип времени представляет собой некоторый направленный вектор времени. Он может быть отрицательным. Может быть выражен в днях, часах, минутах, секундах, миллисекундах.

Duration ::= (‘T’ | ‘Time’) ‘#’ [‘-‘] interval
Interval ::= days | hours | minute | seconds | milliseconds
Days ::= fixed_point

2.4.4.4.2 Время дня и дата

2.4.4.4.2.1 Время дня

TIME_OF_DAY принимает значения [00:00:00:000] – [23:59:59:999]. Представление времени внутри контроллера DWORD и обозначает время от начала дня в миллисекундах.

2.4.4.4.2.2 Дата

DATE принимает значения [xx00-01-01] – [xx99-12-31]. Представление времени внутри контроллера DWORD и обозначает время от начала века в днях.

2.4.4.4.2.2 Дата и время дня

DATE_AND_TIME – Есть полное время. Представление этого типа данных внутри контроллера LWORD и формируется путем конкатенации DATE(4 байта) и TIME_OF_DAY(4 байта).

2.4.4.4.3 Выражения, операторы и приоритеты

Операторы

Форма РБНФ:

Statement_list ::= statement ‘;’{statement ‘;’} - код программы – есть State-ment_list, который записывается в файле – источнике с расширением *.cst. Каждый оператор должен заканчиваться ‘;’.

Вся программа должна состоять только из операторов (statement).

Операторы (заявления) бывают: пустые, операторы присваивании, операторы управления подпрограммами, операторы выбора и итерационные операторы.

Форма РБНФ:

Statement ::= NIL | assignment_statement | subprogram_control_statement | selection_statment| iteration_statement

Оператор присваивания

assignment_ statement::= variable ‘:=’ expression – означает что оператор присваивания это приравнивание некоторого выражения см. нижек переменной.

Пример: a := d+c;

Операторы инкремента, декремента

Inc_ statement ::= variable {‘++’} – инкремент переменной.

Пример: a++;

Dec_ statement ::= variable {‘--’}– декремент переменной

Пример: a--;

Операторы выбора

selection_statement::= if_statement | case_ statement.

а) Оператор if.

if_ statement ::= 'IF' expression1 'THEN' statement_list1
{'ELSIF' expression2 'THEN' statement_list2}
['ELSE' statement_list3]
'END_IF'

Оператор выбора if. Выполнение: если выражение expression1 верно, то выполняется часть программы statement_list, иначе если условие expression2 верно, то выполняется – statement_list2, иначе statement_list3.

Пример:

```
If A<B+C Then
        A++;
        B:= A+C+D;
ELSIF A+C THEN
        C--;
ELSIF A-C THEN
        C :=D+A;
ELSE A++;
END_IF;
```

б) Оператор CASE

case_statement ::= 'CASE' expression 'OF'
case_element {case_element }
['ELSE' statement_list]
'END_CASE'
case_element ::= case_list ':' statement_list;
case_list ::= case_list_element {',' case_list_element }
*case_list_element ::= subrange | **signed_integer** | **bitstring** | enumerated_value.*

Операторы итерационные

iteration_statement ::= for_statement | while_statement | repeat_statement | exit_statement.

а) Оператор for.

for_statement ::= 'FOR' control_variable ':=' for_list
'DO' statement_list 'END_FOR'
control_variable :: = identifier
for_list :: = expression 'TO' expression ['BY' expression]

б) Оператор while.

while_statement ::= 'WHILE' expression
'DO' statement_list 'END_WHILE'

Итерационный оператор while. Выполнение: пока условие expression верно, выполняется часть кода statement_list.

Пример:

```
WHILE KK-C+16 DO
C++;
D := 4*A*B;
END_WHILE;
```

в) Оператор repeat.

repeat_statement ::= 'REPEAT' statement_list
'UNTIL' expression 'END_REPEAT'

Итерационный оператор repeat while. Выполнение: выполняется часть кода statement_list, пока условие expression верно.

Пример:

```
REPEAT
C++;
D := 4*A*B;
UNTIL C<B
END_REPEAT;
```

д) Оператор exit.

Оператор осуществляет выход из цикла пропуская следующие за ним строки кода.

Если оператор вызывается вне цикла – должна быть сгенерирована ошибка.

Выражения

Расширенная форма Бэкуса-Науэра для выражений:

Логические выражения используют только 2 значения 1 если число не 0 и 0 если аргумент = 0.

Например 12||0 будет вычислено 1|0.

Expression ::= xor_logic_expression{'||' xor_logic_expression} - логическоеилинавыходе 0 или 1. (Еслиодноизчиселненольнавыходе 1).

xor_logic_expression : = and_logic_expression{'^^' and_logic_expression } – логичскаясуммапомодулю2.

and_logic_expression : = or_expression{'&&' or_expression } – логическоеИ.

or_expression ::= xor_expression{'|' xor_expression} - побитовоеили, операцияпроизводитсямеждубитами 2-хчисел.

xor_expression ::=and_expression{'^' and_expression}– побитоваясуммапомодулю 2 междубитами 2-хчисел.

and_expression ::=comparison{'&' comparison} – побитовоеИмеждубитами 2-хчисел.

*Comparison ::= equ_expression{'='|'<>'equ_expression }*сравнение 2 числаравны (неравны).

equ_expression ::= add_expression{comparison_operatoradd_expression } – выражениясравнениякоторымотносятся*: comparison_operator::= '<'| '>'| '<='| '>='* – меньше, больше, меньшелиборавно, меньшелиборавно.

add_expression::=term {'+'| '-'term} – операторы сложения, вычитания.

Term ::= power_expression{multiply_operatorpower_expression}- выражения умножения деления, где*multiply_operator::= {'*'| '/'| '*/' }* –умножение , деление, остаток от деления.

*power_expression::= unary_expression{'**'unary_expression }* – возведениевстепень

unary_expression::= [unary_operator] primary_expression– унарнаяоперация.

unary_operator::= '-'|'!'- унарные операторы.

*primary_expression::=constant|enumerated_value|**variable |'(' expression ')'|number**|function* – первичноевыражение *;

- жирным шрифтом указанны те первичные выражения которые реализованы в данной версии.

Список ключевых слов применяемых при программировании.

1. Типы программных единици секции объявления переменных:
    1. PROGRAM – спецификатор программы (Данная программная единица будет использоваться как программа). Программа – это программная единица, в которой находится точка входа (может быть только одна в проекте)
    2. FUNCTION_BLOCK – спецификатор функционального блока (определяет то, что данная программная единица будет использоваться как функциональный блок). Функциональный блок – программная единица, которая запоминает предыдущие состояния и при последующих вызовах может их использовать. Память под нее выделяется статически.
    3. FUNCTION – Спецификатор функции (определяет то, что данная программная единица будет использоваться как функция) Функция - программная единица, которая не сохраняет предыдущего состояния.
    4. VAR – спецификаторданныхпрограммыдлявнутреннегоиспользования (временных переменных). Используется только в заголовочном файле.
    5. VAR_INPUT- спецификаторвходныхданныхпрограммы. Эти данные подаются на вход программной единицы “по значению”. Они не могут изменяться внутри его.
    6. VAR_OUTPUT - спецификаторвходных данныхпрограммной единицы. Эти данные передаются от программной единицы в вызывающую программную единицу по значению.
    7. VAR_IN_OUT – спецификатор входно-выходных данных программной единицы. Данные которые могут использоваться как входные и выходные в программной единице. Передаются по ссылке в вызываемую программную единицу.
    8. VAR_EXTERNAL – спецификаторданныхкоторыеобъявленывпрограммнойединице“PROGRAMM”, VAR_GLOBAL, VAR_ACCESS, RETAIN, NON_RETAIN,CONSTANT,AT,END_VAR
2. Операторы:

:=,++,--,IF,THEN,ELSIF,ELSE,END_IF,CASE,OF,ELSE,END_CASE,FOR, DO,TO,BY,END_FOR,WHILE,DO,END_WHILE,REPEAT,UNTIL,END_REPEAT,EXIT

3. Выражения:

||,|,&&,&,^^,^,!,~,+,-,<,<=,>,>=,=,<>,,/,*/,***

4. Стандартные функции:

BYTE_TO_SINT, BYTE_TO_INT, BYTE_TO_DINT, BYTE_TO_LINT, BYTE_TO_BYTE, BYTE_TO_WORD, BYTE_TO_DWORD, BYTE_TO_LWORD, BYTE_TO_REAL, BYTE_TO_LREAL,
WORD_TO_SINT, WORD_TO_INT, WORD_TO_DINT, WORD_TO_LINT, WORD_TO_BYTE, WORD_TO_WORD, WORD_TO_DWORD, WORD_TO_LWORD, WORD_TO_REAL, WORD_TO_LREAL,

DWORD_TO_SINT, DWORD_TO_INT, DWORD_TO_DINT, DWORD_TO_LINT, DWORD_TO_BYTE, DWORD_TO_WORD, DWORD_TO_WORD, DWORD_TO_LWORD, DWORD_TO_REAL, DWORD_TO_LREAL,
LWORD_TO_SINT, LWORD_TO_INT, LWORD_TO_DINT, LWORD_TO_LINT, LWORD_TO_BYTE, LWORD_TO_WORD, LWORD_TO_DWORD, LWORD_TO_LWORD, LWORD_TO_REAL, LWORD_TO_LREAL,
SINT_TO_SINT, SINT_TO_INT, SINT_TO_DINT, SINT_TO_LINT, SINT_TO_BYTE, SINT_TO_WORD, SINT_TO_DWORD, SINT_TO_LWORD, SINT_TO_REAL, SINT_TO_LREAL,
INT_TO_SINT, INT_TO_INT, INT_TO_DINT, INT_TO_LINT, INT_TO_BYTE, INT_TO_WORD, INT_TO_DWORD, INT_TO_LWORD, INT_TO_REAL, INT_TO_LREAL,
DINT_TO_SINT, DINT_TO_INT, DINT_TO_DINT, DINT_TO_LINT, DINT_TO_BYTE, DINT_TO_WORD, DINT_TO_DWORD, DINT_TO_LWORD, DINT_TO_REAL, DINT_TO_LREAL,
LINT_TO_SINT, LINT_TO_INT, LINT_TO_DINT, LINT_TO_LINT, LINT_TO_BYTE, LINT_TO_WORD, LINT_TO_DWORD, LINT_TO_LWORD, LINT_TO_REAL, LINT_TO_LREAL,
REAL_TO_SINT, REAL_TO_INT, REAL_TO_DINT, REAL_TO_LINT, REAL_TO_BYTE, REAL_TO_WORD, REAL_TO_DWORD, REAL_TO_LWORD, REAL_TO_REAL, REAL_TO_LREAL,
LREAL_TO_SINT, LREAL_TO_INT, LREAL_TO_DINT, LREAL_TO_LINT, LREAL_TO_BYTE, LREAL_TO_WORD, LREAL_TO_DWORD, LREAL_TO_LWORD, LREAL_TO_REAL, LREAL_TO_LREAL,
Эти функции приводят переменную одного типа к другому. TYPE1_TO_TYPE2 – конвертирует число из типа TYPE1 в тип TYPE2.

TRUNC
ABS, SQRT, LN, LOG, EXP, SIN, COS, TAN, ASIN, ACOS, ATAN;
ADD, MUL, SUB, DIV, MOD, EXPT, MOVE;
SHL, SHR, ROR, ROL;
AND, OR, XOR, NOT, LAND, LOR, LXOR, LNOT;
SEL, MAX, MIN, LIMIT, MUX;
GT, GE, EQ, LE, LT, NE;
LEN, LEFT, RIGHT, MID, CONCAT, INSERT, DELETE, REPLACE, FIND ;
ADD_TIME, ADD_TOD_TIME, ADD_DT_TIME, SUB_TIME, SUB_DATE_DATE, SUB_TOD_TIME, SUB_TOD_TOD, SUB_DT_TIME, SUB_DT_DT, MULTIME, DIVTIME, CONCAT_DATE_TOD, DT_TO_TOD, DT_TO_DATE;

5. Стандартные функциональные блоки:
SR, RS,R_TRIG, F_TRIG;
CTU, CTU_DINT, CTU_LINT, CTU_WORD, CTU_DWORD, CTU_LWORD,
CTD, CTD_DINT, CTD_LINT, CTD_WORD, CTD_DWORD, CTD_LWORD,
CTUD, CTUD_DINT, CTUD_LINT, CTUD_WORD, CTUD_DWORD, CTUD_LWORD,;
TON, TOF, TP, RTC;

Обработка ошибок

При возникновении ошибок компилятора, если командная строка не верна или не открывается необходимый файл. Все ошибки сохраняются в тот же каталог, что и сам исполняемый файл в файл “_default_.err” и выполнение программы останавливается.

Возможные ошибки при компиляции

Приразбореадреса:

1 PARSE_ADR_WRONG(адрес не разобран)

"Error: Wrong address: "<<[token]<<" check syntax.\n"

Возникает если символ начала адреса ‘%’ найден а за ним некоторые символы не соответствуют адресным символам. Например%12IV – когданайдетV – выдастошибку.

2 UNDEF_LOCATION_PREFIX (неизвестный префикс расположения адреса)

"Error: Undefined location prefix: "<<[lprefix]<<" in address: "<<[token]<<"."

Возникает, если префикс не найден (I,Q,M). Например: %XQ – ошибка.

3 UNDEF_SIZE_PREFIX (неизвестный префикс размера данных хранимых по адресу)

"Error: Undefined size prefix: "<<[sprefix]<<" in address: "<<[token]<<"."

Возникает, если префикс размера данных не найден (X,B,W,D,L). Например: %II – ошибка.

4 ADR_OFFSET_WRONG (смещение не разобрано)

"Error: wrong offset of address"<<[token]<<" - "<<[offset]<<". Checksyntax.\n";

Ошибка возникает если смещение в адресе не правильное. Например: %IW16.I – смещение I – не определится.

5 ADR_OFFSET_BIG(смещение адреса слишком большое)

"Error: wrong offset of address"<<[token]<<" - "<<[offset]<<". Valuetoobig.\n"

Ошибка возникает если смещение в адресе слишком велико. Например: %IW16.356 – 356 смещение.

6 ADR_WRONG(адрес не преобразован в число)

"Error: wrong address"<<[token<<" - "<<[address]<<". Checksyntax.\n";

Ошибка возникает если адресс не разобран, например %IWI.

7 ADR_BIG(адресс слишком большой (max = ffff))

"Error: wrong address"<<strTokenName<<" - "<<strAdress<<". Valuetoobig.\n";

Возникает если величина адреса больше ffff.

При разборе чисел и времен

1 PREFIX_WRONG (префикс)

"Error: "<<[Error]<<" \""<<[token]<<"\" cannot be used as prefix type.";

Ошибка возникает если префикс числа не соответствует типу или основанию системы счисления. Например, ADD#12 – ошибка или 23#- тоже ошибка.

2 SIGN_FORBID(знак в беззнаковых выражениях запрещен)

"Error: Sign: \""<<[sign]<<"\" don't use with "<<bit string <<" literals.\n";

Ошибка возникает при попытке ввода знака в беззнаковом числе. Например, INT#-12 – правильно, WORD#-12 – не правильно.

Разбор базовых чисел

1 BASE_PREFIX_WRONG (Префикс типа основания счисления (базы) ошибочен)

"Error: \""<<[PrefixStr]<<"\"- wrong prefix of base in based number.";
Ошибка возникает, если не найден тип базы (основания счисления), INT#11#27 основания счисления 11 не существует в данном компиляторе.
2 BASE_NUMB_SIGN (знак с базовыми значениями не используется)
"Error: Sign: \""<<[strSign]<<"\" don't use with based literals.";
Если при вводе базового числа был использован знак – возникает ошибка. Например: INT#-16#27
3 BASE_NUMBER_FORBIT (базовые числа используются только в целых и битовых числах)
"Error: Based Number : \""<<[token]<<"\" don't allow "<<[type]<<" type prefix.";
Ошибка возникает, если например число с плавающей точкой пытаться записать в данной форме.
4 BASE_NUMB_NOTRECOGNIZE (базовое число распознанно не до конца)
c_err()<<"Error: "<<[token]<<"- wrong based "<<(octal, binary, hexadecimal)<<" number. Recognizeonly: \""<<[recognize_token] <<"\'.";
Ошибка возникает если базове число не разобранно до конца, например 16#22.13 или 8#189.
5 BASE_NUMB_WRONG (Базовое число ошибочно)
"Error: "<<[token]<<"- wrong based "<<strPrefixBase<<" number.";
Обычно возникает ошибка, если знаки подчеркивания введены неправильно.

Разбор временных литералов.
Разбор длительности времени:
1 TIME_DUR_NOTRECOGNIZE (интервал не распознан)
"Error: "<<[token] <<"- wrong time duration literal.";
Ошибка возникает если префикс длительности времени распознан, а до ближайшего разделителя есть символы которые не соответствуют разрешенным символам времени например TIME#125ms – верно, А 125mks-ошибочно.
2 STR_PARSE_NOT_COMPLET (разбор строки неокончен)
"Error: "<<[token]<<" substring cannot be disassembled according to a rule of analysis.";
Ошибка возникает, если разбор по идее был быть закончен (ms или ’.’) однако запись до конца не разобрана. Например, TIME#13ms25s (после ms следует 25s) или 5.6h20m (5.6h – конечная запись).
3 STR_VALUE_EMPTY (строка пуста)
"Error: "<<[token]<<" - expected value of unit time, but value empty or not confirm number."; Возникает, если одно из значений времени пустое напримерTIME#12sms.
4 STR_TIME_DUR_EMPTY (единица измерения интервала пуста)
"Error: "<<[expected unit]<<" "- expected, but time unit literal empty.";
Возникает, если обозначение времени не введено нету ms, s, например TIME#13

5 ORDER_NOT_CONFIRM (единицы измерения времени следуют не по порядку)
"Error: "<<[curent duration unit]<< - unexpected duration unit. ")<<[expected unit] <<" expected. Time units should follow one another in the following order D,H,M,S,MS.");
Возникает, если единицы измерения длительности времени введены не в соответствующем порядке (D,H,M,S,MS).Например, TIME#5h12d
6 UNDERLINE_AT_VALUE_NOT_PERMETE (знаки подчеркивания запрещены)
"Error: "<<[token]<<" wrong. Only units of duration literals can be separated by underline character.";
Возникает, если в значении введен знак подчеркивания TIME#11_5ms
7 FLOAT_VALUE_NOT_CONVERTED(значение флоат не преобразовано)
"Error: "<<[value]<<" - float value "<<[token]<<" unit (time duration) not converted from string.";
Возникает при невозможности преобразования из строки в число, дробной величины.
8 INT_VALUE_NOT_CONVERTED (значение целое не преобразовано)
"Error: "<<[value]<<"- integer value <<[token]<< " unit (time duration) not converted from string.";
Возникает при невозможности преобразования из строки в число, целой величины.
9 STR_UNIT_TIME_NOTFIND (единица измерения в таблице символов не найдена)
"Error: "<<[unit]<<" - not confirm units of time duration literals.");
Так как вначале, строка может быть распознана как Time#12sm, то данная ошибка исключает возможность использования примерной записи.
10 UNIT_TOO_BIG (значение единицы измерения времени слишком велико)
"Error: "<<[unit]<<" - unit of "<<[token]<< " (time duration) too big. ItMustbelessthen ") <<[MaxValue]<<;
Возникает в 2-х случаях: если число было введено первым и оно больше 10000. Например: Time#11000s, или если интервал стоит не первым и превышает свои границы например, 24 часа, 33дня , 60 минут и секунд.
11 DURATION_TOO_BIG(длительность времени слишком велика)
Error: "<<[token]<< " - (time duration) too big. It Must be less then<<[MaxValue]<<.
Величина длительности не может быть больше 7fff_ffff_ffff-ffff (4байта/2).

Разбордаты и времени дня
1 DT_NOTRECOGNIZE (интервал не распознан)
"Error: "<<[token] <<"- wrong date and time literal."

Ошибка возникает если префикс даты или времени дня распознан, а до ближайшего разделителя есть символы которые не соответствуют разрешенным символам времени например D#2008-11k-12s.

Разбор Даты

1 D_SEPARAT_NOTFOUND(разделитель не найден)

"Error: separator \""<<[separat] <<"\" for "<<(date, date_and_time)<<" not found. ";

Ошибка возникает если ниодного разделителя(-) не найдено. А именно, D#123.

2 D_SEPARAT2_NOTFOUND (разделитель не найден)

"Error: second separator \""<<[separat] <<"\" for "<<(date, date_and_time)<<" not found. "; Ошибка возникает если 2-го разделителя(-) не найдено. А именно, D#1980-2345.

3 D_ NOTCONVERT (дата не преобразованна из строки в дату)

"Error: "<<[token]<<"\" for""<<(date, date_and_time)<<" literal not converted.");

Ошибка возникает если дата выходит за диапозон от 1970-01-01 до 3000-12-31.

4 D_NOTEXIST(дата не существует)

"Error: date: "<<[token]<<" not exist. Check how many days at this month in this year.";

Ошибка возникает при попытке ввода 30 февраля 31 апреля и так далее. Например, D#2007-02-29.

5 FLOAT_VALUE_NOT_PERMET (нельзя ввести вещественное число в качестве значения)

"Error: "<<[value]<<" - fixed point value don't use with "<<[dateUnit]<<" unit.";

При попытке ввода D#1983-12-31.2 – ошибка.

6 VALUE_TIME_EMPTY (пустоезначение)

"Error: empty value of "<<[dateUnit]<<" unit.";

Ошибка возникает если одно из значений времени пустое. Например, D#1983—01.

7 INT_VALUE_NOT_CONVERTED(значения не преобразованно)

"Error: "<<[value]<<"- integer value of "<<[dateUnit]<<" unit not converted.";

Возникает при невозможности преобразовать из строки в число. Например если слишком большое число. Например, D#10000000000000000000000-10-2

8 UNIT_VALUE_TOO_BIG(значениеслишкомбольшое)

"Error: "<<[value]<<"- value of "<<[dateUnit]<<" unit too big.";

При попытке ввода 13 месяца или 32 дня возникает ошибка. Например, D#1980-13-32;

9 VALUE_EMPTY (значениеравно 0)

"Error: "<<[value]<<"- value of "<<[dateUnit]<<" unit not available 0 value.");

Ошибка возникает если дата или месяц введена нулевая (в формате С). Например, чтобы ввести дату как в языке с, 1 января 2000 года - нужно

записать 2000-00-01. А в нашем случае D#2000-01-01, а нулевого месяца не существует.

Разборвременидня

1 TOF_SEPARAT_NOTFOUND(разделитель не найден)

"Error: separator \""<<[separat] <<"\" for "<<(time_of_date, date_and_time)<<" not found. ";

Ошибка возникает если ниодного разделителя(:) не найдено. А именно, TOF#123.

2 TOF_SEPARAT2_NOTFOUND (разделитель не найден)

"Error: second separator \""<<[separat] <<"\" for "<<(date, date_and_time)<<" not found. "; Ошибка возникает если 2-го разделителя(:) не найдено. А именно, TOF#19:2345.

3 FLOAT_VALUE_NOT_PERMET (нельзя ввести вещественное число в качестве значения)

"Error: "<<[value]<<" - fixed point value don't use with "<<[TOFunit]<<" unit.";

При попытке ввода tof#11:12:31.2 – правильно так как секунды могут быть вещественным числом., но tof#11:12.2:31 – должно вызвать ошибку.

4 VALUE_TIME_EMPTY (пустоезначение)

"Error: empty value of "<<[tofUnit]<<" unit.";

Ошибка возникает если одно из значений времени пустое. Например, TOF#19::1. или TOF#19:1:.

5 FLOAT_VALUE_NOT_CONVERTED (вещественное значение не преобразованно)

"Error: "<<[value]<<"- fixed point value of "<<[tofUnit]<<" unit not converted.";

Возникает при невозможности преобразовать из строки в число. Например если слишком большое число? Или несколько '.' . Например, TOF#11:12:2.3.4

6 INT_VALUE_NOT_CONVERTED(целое значение не преобразованно)

"Error: "<<[value]<<"- integer value of "<<[tofUnit]<<" unit not converted.";

Возникает при невозможности преобразовать из строки в число. Например если слишком большое число. Например, TOF#1000000000000000000000:10:2

7 UNIT_VALUE_TOO_BIG (значениеслишкомбольшое)

"Error: "<<[value]<<"- value of "<<[tofUnit]<<" unit too big.";

При попытке ввода 24 часов или 60 минут(секунд)возникает ошибка. Например, tof#24-61-63.7;

Разбордатыивремени.

1 (разделительненайден)

"Error: separator \""<<[separat] <<"\" for "<<(date_and_time)<<" not found. ";

Ошибка возникает если ниодного разделителя(-) не найдено. Аименно,DT#123.

2 (неверныйформат)

"Error: error format of \""<<[token]<<". Check format: year-month-day-hour:minute:second. ";
Ошибка возникает если человек перепутал и вначале ввел время а затем дату например
DT#12:22:11-1983-04-07.

Разбор целых чисел
1 NUMB_EXPECT (ожидалось число)
"Error: Expected "<<[type]<<" number.";
Ошибка возникает если был введен префикс числа (тип), а само число не введено. НапримерINT#f17.
2 NUMB_REALEXPECT (ожидалосьчисло)
"Error: Expected real literal. ";
Ошибка возникает если был введен префикс вещественного числа ,а оно не распознанно. Например, REAL#13.
3 NUMB_EXTRUCT(число до конца не распознанно)
"Error: expectednumber. Some literal not conform decimal digit.";
Если ввести число а вконце его символы, то будет выданна ошибка. Например, 12aaa.

Разбордиапазона
1 SUBR_EXPECT_UP_LIMIT (ожидалсяверхнийпределдиапазона)
"Error: Expected upper limit of subrange: "<<[token]<<".";
Если нет верхнего предела возникает ошибка. Например, 12.. – ошибкаили 112..d.
2 SUBR_PREFIX_FORBID (диапазон не может быть указан с префиксом типа)
"Error: subrange:"<<[token]<<" don't permit prefix type.";
ВозникаетесливвестиINT#16..12.
3 SUBR_PARSENOT_COMPLEAT(верхняя граница диапазона распознанна не до конца)
"Error:parse number of upper border of a subrange not compleat. Some literal not conform decimal digit.";
Ошибка возникает если в числе обозначающем верхнюю границу диапазона ошибка, например 12..33hkl12.

Разбор вещественного числа.
1 RNUMB_EXPECT(ожидалось вещественное число)
"Error: Expected real literal. ";
Ошибка возникает если спецификатор вещественного числа введен ‘.’, но продолжения числа нету. Например, 12.abcили 12. +15. .
2 RNUMB_EXPON_EXPECT(ожидалась экспонента вещественного числа)
"Error: Expected exponent integer value.";

Ошибка возникает если спецификатор экспоненты eE введен однако за ней не следует целое число. Например 12.11e – ошибка, 12.11e-1 – правильно.

Разборстроковогозначения.

1 (спецификатор окончания строкового значения не найден)

"Error: "<<(single,double)<<" byte character string:\""<<[token]<<"\" not closed";

Ошибка возникает если спецификатор окончания строкового значения не найден (для однобайтных строк спецификатор – ‘, для двубайтных “). Например, ‘sadsadsadили “sssssssssss .

При разборе имен.

1 NAME_TOO_MACH_CHARS (слишкоммногосимволов)

"Error: Too much character in name:"<<[token]<<" . Use name, whose length less then 40 character.\n";

Ошибка возникает если в имени переменной больше 40 символов (40 может быть измененно однако для версии компилятора оно одинаково). Например aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

2 Ошибка с использованием знаков подчеркиваний (см. ниже).

"Checksomenamesyntax:"<<[token]<<".\n";

3 NAME_TRAILING_UNDERLINES (2 знакаподчеркивания)

"Error: 2 or more underlines in name. Check some name syntax:"<<[token]<<".\n";

Если 2 знака подчеркивания в имени – выдается ошибка. Например, BB_aa_CC.

Использованиезнаковподчеркивания.

1 **ENumbNotConverted**(знак подчеркивания в начале числа (слова))

"Error: Underline at begin position don't permit.";

Возникает если число или слово начинается со знака подчеркивания. Например _12.

2 **E_UnderlineAtEnd**(знакподчеркиваниявконцечисла (слова))

"Error: Underline at end position don't permit.";

Возникает если число или слово заканчивается знаком подчеркивания. Например 12_.

3 **E_UnderlineTogether** (двазнакаподчеркиванияподряд)

"Error: Underline following one after another don't permit.";

Возникает если 2 знака подчеркивания следуют друг за другом. Например, 12__3.

Стратегия распределения адресов:

Используется 2 типа доступа к памяти:

1. Прямой доступ (AT %IW) . Указывает адрес в устройстве. Служит для адресации всех переменных ввода-вывода. Иным способом данные переменные адресовать нельзя.

Ограничения: Если адресация устройства - словная, то при задании адреса нечетного должна выдаваться ошибка о выравнивании.

Компилятор работает с байтовой адресацией. Например, адрес AT %QW17 – обозначает, что в памяти вывода по адресу 17 (байтному) расположены данные размером 16 бит.

2. Косвенная адресация. Адреса автоматически назначаются компилятором. Если устройство работает со словной адресацией.

Стратегия распределения команд.

В МЛК имеется несколько программных регистров, которые используются при выполнении логической программы. AX (ACC0), BX (ACC1), CX, DX. AX и BX используются как регистры аккумуляторы с ними можно производить различные действия. CX и DX могут использоваться как индексные регистры или для хранения данных (как обычный регистр).

Также при компиляции в машинный код используется стек. SP – (stackpointer) – программный регистр указателя стека. При помещении данных в стек он увеличивается на размер данных (в байтах). Если в стек помещается бит, то он увеличивается на размер 1.SP – наращивается и уменьшается автоматически в зависимости от выполнения команд PUSH,POP.

Разберем пример:

E = a * b +c *(b * d – a*c + c *(d - b)); все переменные типа word

Программа разбивается на 3-х адресные инструкции:

№	Оператор	Аргумент 1	Аргумент 2	Результат
1	*	a	b	->CVarElement_0
2	*	b	d	->CVarElement_1
3	*	a	c	->CVarElement_2
4	-	CVarElement_1	CVarElement_2	->CVarElement_3
5	-	d	b	->CVarElement_4
6	*	c	CVarElement_4	->CVarElement_5
7	+	CVarElement_3	CVarElement_5	->CVarElement_6
8	*	c	CVarElement_6	->CVarElement_7
9	+	CVarElement_0	CVarElement_7	->CVarElement_8
10	=	->CVarElement_8		E

Преобразование 3-х адресного кода в машинные инструкции.

Дизассемблированный код машинной инструкции представляет собой вид

Команда Параметр1, Параметр2.

Где команда – общий вид команды.

Параметр1 – это одновременно и первый аргумент трехадресной инструкции и результат.

Параметр2 - это второй аргумент. Может принимать значения:

a) Регистр (указывается название регистра).

b) Значение (указывается просто число).

c) Адрес (указывается @адресс).

d) Индексный регистр().

1.Инструкция.

LDACC0,@a //загрузка значения из адреса @a в регистр ACC0
MULACC0,@b //умножаем содержимое 0 аккумулятора на значение по адресу @b
PUSHACC0//так как следующая инструкция не использует в качестве аргумента ->CVarElement_0 то значение аккумулятора сохраняем в стек (SP=2).

2.Инструкция.

LDACC0,@b //загрузка значения из адреса @b в регистр ACC0
MULACC0,@d //умножаем содержимое 0 аккумулятора на значение по адресу @d
PUSHACC0//так как следующая инструкция не использует в качестве аргумента ->CVarElement_1, то значение аккумулятора сохраняем в стек (SP=4).

3. Инструкция.

LDACC0,@a //загрузка значения из адреса @a в регистр ACC0
MULACC0,@c //умножаем содержимое 0 аккумулятора на значение по адресу @d
//так как следующая инструкция использует CVarElement_2 – результат этой инструкции оставляем в регистре ACC0.

4. Инструкция.

POPACC1//из стека помещаем в регистр ACC1 значение CVarElement_1 (оно размещалось на вершине стека SP = 2).
SUBACC1, ACC0//вычитаем из регистра ACC1 – ACC0 и результат сохраняется в ACC1.
MOVACC0, ACC1//значение из ACC1 копируем в ACC0.
PUSHACC0//так как следующая инструкция не использует в качестве аргумента ->CVarElement_3, то значение аккумулятора сохраняем в стек (SP=4).

5. Инструкция.

LDACC0,@d //загрузка значения из адреса @d в регистр ACC0
SUBACC0,@b //вычитаем из содержимого 0 аккумулятора значение по адресу @b.

6. Инструкция.

LDACC1,@c //загрузка значения из адреса @c в регистр ACC0
MULACC1, ACC0 //умножаем содержимое 1 аккумулятора на значение аккумулятора 0 Результат в аккумуляторе 1.
MOVACC0, ACC1//значение из ACC1 копируем в ACC0.

7. Инструкция.

POPACC1//из стека помещаем в регистр ACC1 значение CVarElement_3 (оно размещалось на вершине стека SP = 2).
ADDACC1, ACC0//складываем одержимое регистров ACC1 и ACC0 и результат сохраняется в ACC1.
MOVACC0, ACC1//значение из ACC1 копируем в ACC0.

8. Инструкция.

LDACC1,@c //загрузка значения из адреса @c в регистр ACC0
MULACC1, ACC0 //умножаем содержимое 1 аккумулятора на значение аккумулятора 0 Результат в аккумуляторе 1
MOVACC0, ACC1//значение из ACC1 копируем в ACC0.

9. Инструкция.

POPACC1//из стека помещаем в регистр ACC1 значение CVarElement_0 (оно размещалось на вершине стека SP = 0).
ADDACC1, ACC0//складываем одержимое регистров ACC1 и ACC0 и результат сохраняется в ACC1.
MOVACC0, ACC1//значение из ACC1 копируем в ACC0.

10. Инструкция.

ST @E, ACC0 //сохраняем значение по адресу @E.

Алгоритм

1. Считываем инструкцию.
2. Если инструкция арифметическая передаем ее калькулятору, иначе выполняем команду (она с регистрами AX, BX, CX, DX не работает поэтому нас не интересует).
3. Каждая инструкция проверяет было, ли загружено значение в аккумулятор 0. Если было загружено, то п.4.
4. Программа предполагает, что по умолчанию в аккумуляторе 0 находится значение одного из аргументов инструкции. Проверяем, является ли команда арифметической. Если не является п.3. Иначе п.4.
5. Выполняем команду она с регистрами AX, BX, CX, DX не работает.

Листинг: программа GasProgect

GasProgect.h

```
(* Файл объявления *)
PROGRAM Gasoplotnost
VAR_GLOBAL
	//**********************************************************************
********
	// Дискретные входы
	//801h слов = 4098 резерв 808h = 4112

	D01Vent		AT %MX4112.0 : BOOL;	//Вентиляция топки
	D02Start	AT %MX4112.1 : BOOL;	//Старт проверки герметичности
	D03Reset	AT %MX4112.2 : BOOL;	//Сброс проверки герметичности
	D04ZadvOtkr AT %MX4112.3 : BOOL;	//Задвижка на подводе газа к ЗСУ открыта
	D05ZadvZakr AT %MX4112.4 : BOOL;	//Задвижка на подводе газа к ЗСУ закрыта
	//**********************************************************************
********
	// Релейные выходы
	//802h слов = 4100

	R01SvechaZakr	AT %MX4100.0 : BOOL;	//Свеча безопасности закрыть
```

```
        R02Negermet AT %MX4100.1 : BOOL;    //Арматура горелки не герметична
        R03ZadvZapalOtkr    AT %MX4100.2 : BOOL;    //Задвижка на подводе газа к за-
пальнику открыть
        R04ZadvZapalZakr    AT %MX4100.3 : BOOL;    //Задвижка на подводе газа к за-
пальнику закрыть
        R05SolenoidOtkr     AT %MX4100.4 : BOOL;    //Соленоид открыть
        R06SvechaZakr       AT %MX4100.5 : BOOL;    //Задвижка на подводе газа к за-
пальнику закрыть
        R07ZapretotkrAT %MX4100.6 : BOOL;    //Соленоид открыть

        //расположены в памяти релейных выходов
        R08ERROR1           AT %MX4100.7 : BOOL;    //Ошибка 1-й проверки
        R09ERROR2           AT %MX4101.0 : BOOL;    //Ошибка 2-й проверки
        R10ERROR3           AT %MX4101.1 : BOOL;    //Ошибка 3-й проверки
        R11ERROR4           AT %MX4101.2 : BOOL;    //Ошибка 4-й проверки
        R12ArmaturaGerm             AT %MX4101.3 : BOOL;    //Арматура герметична
        //расположены
        DubERROR1           AT %MX4100.7 : BOOL;    //Ошибка 1-й проверки
        DubERROR2           AT %MX4101.0 : BOOL;    //Ошибка 2-й проверки
        DubERROR3           AT %MX4101.1 : BOOL;    //Ошибка 3-й проверки
        DubERROR4           AT %MX4101.2 : BOOL;    //Ошибка 4-й проверки
        DubArmaturaGerm             AT %MX4101.3 : BOOL;    //Арматура герметична



        tmp AT %MW8192 : WORD;


        //**********************************************************************
********
        // Аналоговые входы
        // 800h = 4096
        Pin             AT %MW4096 : WORD;    //Аналоговый вход давления 4-20мА 0-
100КПа
        //;Уставки
        Pnom            AT %MW16384 : WORD;   //Руст 7.2мА =20КПа
END_VAR

VAR
        //**********************************************************************
********
        // Временные переменные
        ErrorLast       : BOOL;//последняя ошибка
        D01VentL        : BOOL;//предыдущее значение    D01Vent
        D02StartL       : BOOL;//предыдущее значение    D02Start
        FlNextPermete           : BOOL;//запуск проверки разрешен

        temp            : WORD;//временная переменная

        //расположены в памяти релейных выходов
        ERROR1              : BOOL;       //Ошибка 1-й проверки
        ERROR2              : BOOL;       //Ошибка 2-й проверки
```

```
    ERROR3          : BOOL;     //Ошибка 3-й проверки
    ERROR4          : BOOL;     //Ошибка 4-й проверки
    ArmaturaGerm    : BOOL;     //Арматура герметична


    instDelay       : TimeOut;
    //instDelay1    : TimeOut;
END_VAR
```

GasProgect.cst

```
(* Файл реализации *)
tmp := WORD#16;
FlNextPermete := BOOL#0;
//если есть результаты проверки - запускать алгоритм не нужно
IF !(ERROR1 | ERROR2 | ERROR3 | ERROR4 | ArmaturaGerm) THEN
    IF D02Start^D02StartL&D02Start THEN //пришла команда СТАРТ!
        IF D01Vent^D01VentL&D01Vent THEN
            FlNextPermete := BOOL#1;
        END_IF;
    END_IF;
    IF FlNextPermete THEN
        D01VentL := D01Vent;//сохраняем текущие значения дискретов
        D02StartL := D02Start;//сохраняем
        R05SolenoidOtkr := BOOL#1;//открываем соленоид
        instDelay(tm := TIME#7s);//пауза 7 секунд

        R06SvechaZakr := BOOL#1;
        //импульсно включаем R01SvechaZakr
        R01SvechaZakr := BOOL#1;
        instDelay(tm := TIME#5s);//пауза 5 секунд
        R01SvechaZakr := BOOL#0;
        //продолжаем
        //instDelay(tm := TIME#1m);//пауза 1 минута
        instDelay(tm := TIME#10s);//пауза 1 минута

        IF Pin >= Pnom THEN//задвижка  на подводе газа к ЗСУ не герметична
            FlNextPermete := BOOL#0;//сохраняем состояние
            ERROR1 := BOOL#1;//сохраняем ошибку
        END_IF;
    END_IF;
    IF FlNextPermete THEN
        R05SolenoidOtkr := BOOL#0;
        instDelay(tm := TIME#3s);//пауза 3 секунд
        //импульсно включаем R03ZadvZapalOtkr
        R03ZadvZapalOtkr := BOOL#1;
        instDelay(tm := TIME#5s);//пауза 5 секунд
        R03ZadvZapalOtkr := BOOL#0;
        //продолжаем
        WHILE D05ZadvZakr| ~ D04ZadvOtkr DO //проверяем пока не установятся значения
D05ZadvZakr= 0 D04ZadvOtkr = 1
            IF D03Reset THEN//Сброс и выход из цикла
                FlNextPermete := BOOL#0;//лучше дальше не продолжать
                EXIT;
            END_IF;
        END_WHILE;
    END_IF;
    IF FlNextPermete THEN
        IF Pin >= Pnom THEN //задвижка  на подводе газа к ЗСУ не герметична
            ERROR2 := BOOL#1;
```

```
                    FlNextPermete := BOOL#0;
                END_IF;
        END_IF;
        IF FlNextPermete THEN
                R05SolenoidOtkr := BOOL#1;
                instDelay(tm := TIME#15s);//пауза 15 секунд
                IF Pin <= Pnom THEN //
                        ERROR3 := BOOL#1;
                        FlNextPermete := BOOL#0;
                END_IF;
        END_IF;
        IF FlNextPermete THEN
                //импульсно включаем R04ZadvZapalZakr
                R04ZadvZapalZakr := BOOL#1;
                instDelay(tm := TIME#5s);//пауза 5 секунд
                R04ZadvZapalZakr := BOOL#0;
                //продолжаем
                WHILE D04ZadvOtkr| ~ D05ZadvZakr DO //проверяем пока не установятся значения
D05ZadvZakr= 1 D04ZadvOtkr = 0
                        IF D03Reset THEN//Сброс и выход из цикла
                                FlNextPermete := BOOL#0;//лучше дальше не продолжать
                                EXIT;
                        END_IF;
                END_WHILE;
        END_IF;
        IF FlNextPermete THEN
                instDelay(tm := TIME#1m);//пауза 1 минута
                IF Pin <= Pnom THEN //
                        ERROR4 := BOOL#1;
                        FlNextPermete := BOOL#0;
                END_IF;
        END_IF;
END_IF;
IF FlNextPermete THEN
        IF ERROR1|ERROR2|ERROR3|ERROR4 THEN
                R02Negermet := BOOL#1;
                R06SvechaZakr := BOOL#0;
                R07Zapretotkr := BOOL#1;
                R05SolenoidOtkr := BOOL#0;
                //CALL          Impproc04
        ELSE
                R06SvechaZakr := BOOL#0;
                ArmaturaGerm := BOOL#1;
        END_IF;
END_IF;
IF D03Reset THEN
        ERROR1 := BOOL#0;
        ERROR2 := BOOL#0;
        ERROR3 := BOOL#0;
        ERROR4 := BOOL#0;
        ArmaturaGerm := BOOL#0;
        R02Negermet := BOOL#0;
        R05SolenoidOtkr := BOOL#0;
        R06SvechaZakr := BOOL#0;
        D01VentL := BOOL#0;
        D02StartL := BOOL#0;
END_IF;
R08ERROR1 := ERROR1;
R09ERROR2 := ERROR2;
R10ERROR3 := ERROR3;
R11ERROR4 := ERROR4;
R12ArmaturaGerm         := ArmaturaGerm;
```

```
DubERROR1 := ERROR1;
DubERROR2 := ERROR2;
DubERROR3 := ERROR3;
DubERROR4 := ERROR4;
DubArmaturaGerm := ArmaturaGerm;
```

TimeOut.h

```
(* Файл объявления *)
FUNCTION_BLOCK TimeOut
VAR_INPUT
        tm : TIME;
END_VAR
VAR
        TimerForDelay :TP;
END
```

TimeOut.c

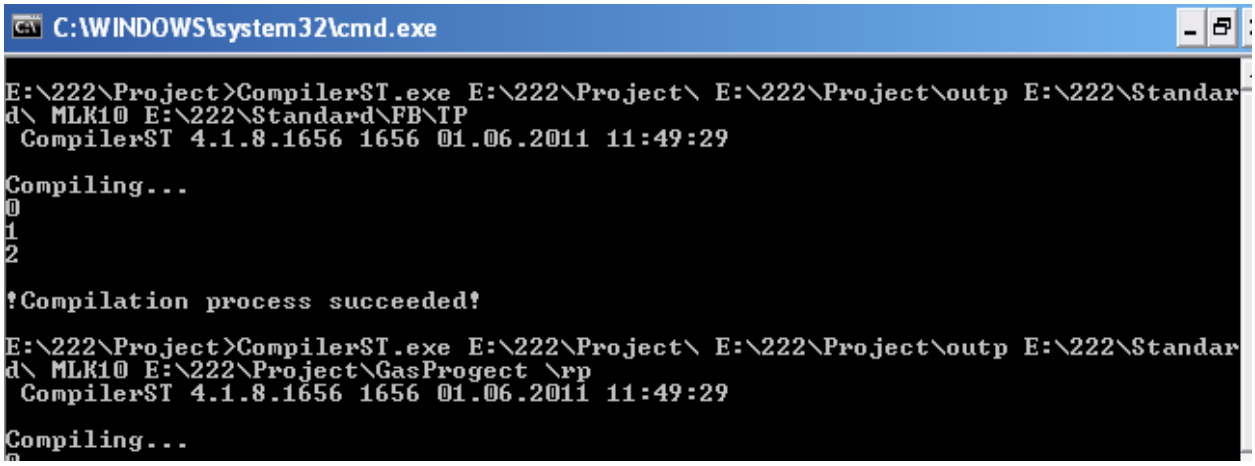
```
(* Файл реализации *)
TimerForDelay (PT :=tm,IN := BOOL#1);
WHILE TimerForDelay.Q DO
        TimerForDelay();
END_WHILE;
TimerForDelay(IN := BOOL#0);
```

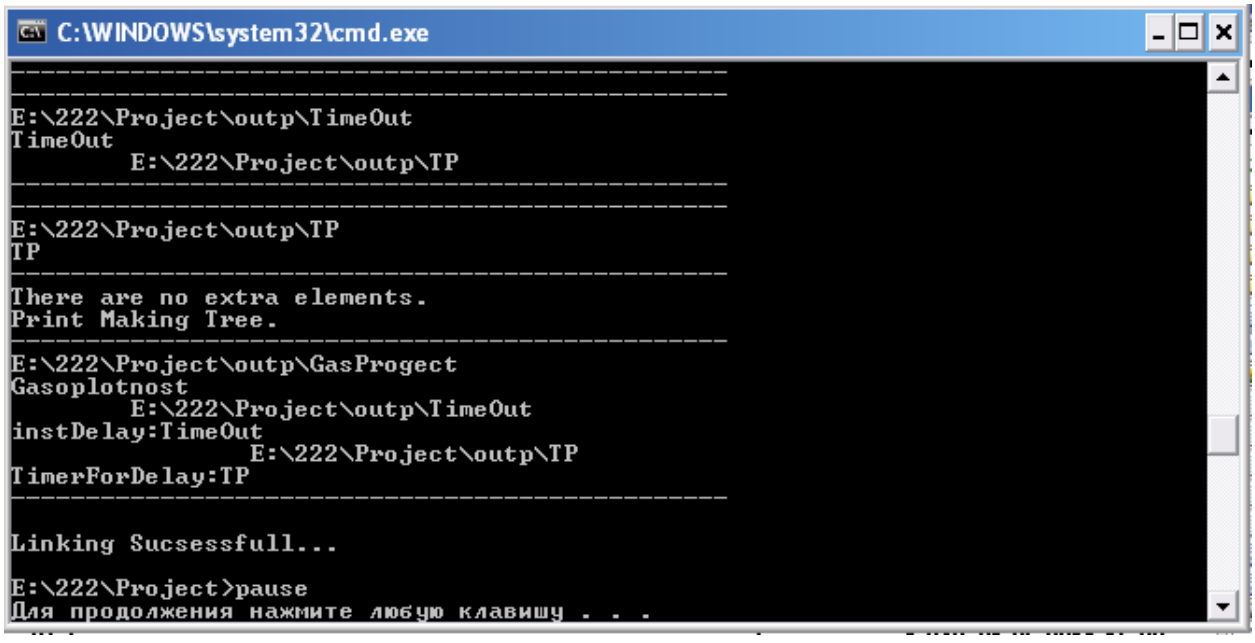
Компилирование исходного кода

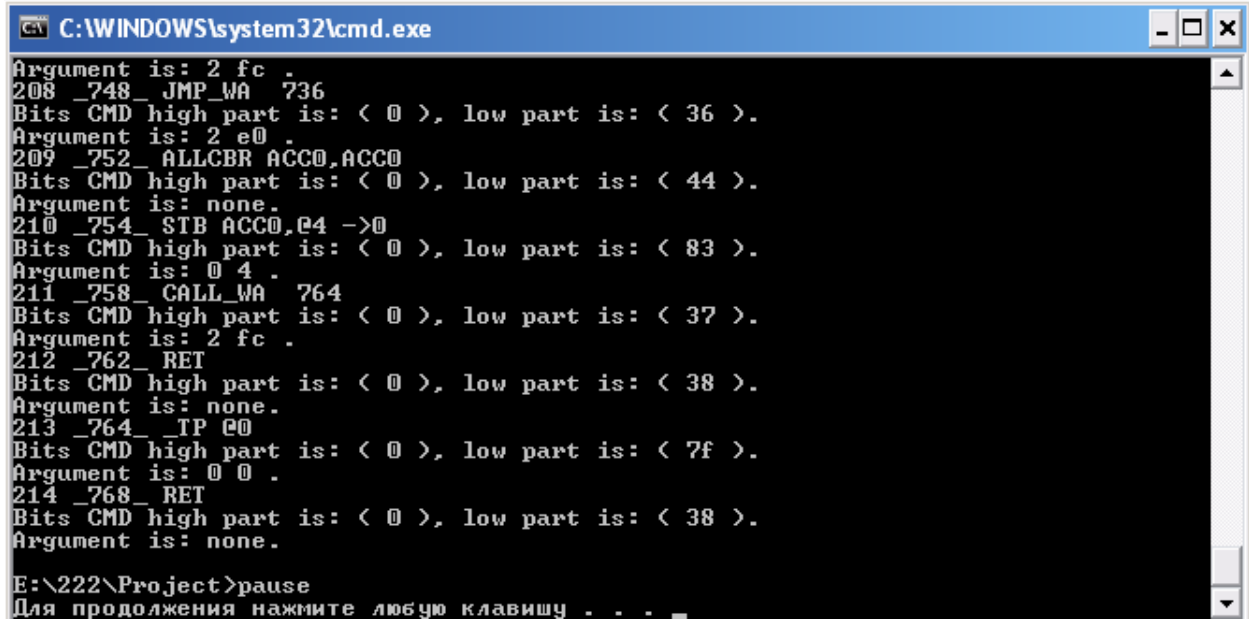
Для компилирование исходного кода необходимо файлы *.h и *.c с кодом программы скопировать в папку нахождения компилятора (CompilerST), линковщика (LinkerIec611), дизасеблера (DisassemblerMLK). Также в данной папке должна быть папка Standart – где хранятся стандартные функции которые могут быть использованы в проекте.

Далее создается *.bat файл в котором прописывается вызов компилятора с передачей ему исходных файлов кода, вызов линковщика с передачей ему объектных файлов созданных компилятором, вызов дизасемлера. Данный *.bat файл имеет вид:

```
//Передача для компилирования файлов TestDiskretRelay*.h, TestDiskretRelay.cst,TP.h,
//TP.cst, указывается выходная директория E:\Project\outp где будут хранится объектные
//файлы
CompilerST.exe E:\222\Project\ E:\222\Project\outp E:\222\Standard\ MLK10 E:\222\Standard\FB\TP
CompilerST.exe E:\222\Project\ E:\222\Project\outp E:\222\Standard\ MLK10 E:\222\Project\GasProgect \rp
CompilerST.exe E:\222\Project\ E:\222\Project\outp E:\222\Standard\ MLK10 E:\222\Project\TimeOut
Pause
//Передаем линковщику объектные файлы созданные компилятором и хранящиеся в
//директории E:\Project\outp (TestDiskretRelay.obj, TP.obj), указываем место хранения *.bin
//файла E:\Project\bin и его название test
LinkerIec611.exe E:\222\Project\outp E:\222\Project\bin E:\222\Standard\ MLK10 test2 \iw \pdo \mfs
GasProgect.obj TimeOut.obj TP.obj
pause pause
//Запускаем дизассемблер с передачей нашего исходного файла test.bin
DisassemblerMLK.exe E:\222\Project\bin\test.bin
Pause
```


C:\WINDOWS\system32\cmd.exe
E:\222\Project>CompilerST.exe E:\222\Project\ E:\222\Project\outp E:\222\Standar
d\ MLK10 E:\222\Standard\FB\TP
CompilerST 4.1.8.1656 1656 01.06.2011 11:49:29
Compiling...
0
1
2
!Compilation process succeeded!
E:\222\Project>CompilerST.exe E:\222\Project\ E:\222\Project\outp E:\222\Standar
d\ MLK10 E:\222\Project\GasProgect \rp
CompilerST 4.1.8.1656 1656 01.06.2011 11:49:29
Compiling...


C:\WINDOWS\system32\cmd.exe
E:\222\Project\outp\TimeOut
TimeOut
E:\222\Project\outp\TP
E:\222\Project\outp\TP
TP
There are no extra elements.
Print Making Tree.
E:\222\Project\outp\GasProgect
Gasoplotnost
E:\222\Project\outp\TimeOut
instDelay:TimeOut
E:\222\Project\outp\TP
TimerForDelay:TP
Linking Sucsessfull...
E:\222\Project>pause
Для продолжения нажмите любую клавишу . . .


C:\WINDOWS\system32\cmd.exe
Argument is: 2 fc .
208 _748_ JMP_WA 736
Bits CMD high part is: (0), low part is: (36).
Argument is: 2 e0 .
209 _752_ ALLCBR ACC0,ACC0
Bits CMD high part is: (0), low part is: (44).
Argument is: none.
210 _754_ STB ACC0,@4 ->0
Bits CMD high part is: (0), low part is: (83).
Argument is: 0 4 .
211 _758_ CALL_WA 764
Bits CMD high part is: (0), low part is: (37).
Argument is: 2 fc .
212 _762_ RET
Bits CMD high part is: (0), low part is: (38).
Argument is: none.
213 _764_ _TP @0
Bits CMD high part is: (0), low part is: (7f).
Argument is: 0 0 .
214 _768_ RET
Bits CMD high part is: (0), low part is: (38).
Argument is: none.
E:\222\Project>pause
Для продолжения нажмите любую клавишу . . .

2.5 Меры безопасности

2.5.1 К эксплуатации контроллеров допускается персонал, имеющий разрешение для работы на электроустановках напряжением до 1000 В и изучивший руководство по эксплуатации в полном объеме.

2.5.2 Эксплуатация контроллера разрешена при наличии инструкции по технике безопасности, утвержденной предприятием-потребителем в установленном порядке и учитывающей специфику применения контроллера на конкретном объекте.

2.5.3 Перед разборкой контроллера его необходимо обесточить.

ЗАПРЕЩЕНО!!! подключать или отключать клеммные разъемы при включенном питании. Необходимо обесточить как контроллер, так и внешние подсоединения.

3 ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ

3.1 Контроллер рассчитан на круглосуточную работу.

3.2 Специального технического обслуживания контроллер не требует. Для обеспечения нормальной работы рекомендуется один раз в год выполнить следующие мероприятия:

- проверять надежность крепления контроллера в месте установки и его внешних соединений;
- проводить очистку контроллера от пыли путем протирания внешних доступных частей, а также путем воздушной продувки сухим и чистым сжатым воздухом;
- провести полную диагностику контроллера, проверить журнал событий, скорректировать часы, если требуется.

4 ТЕКУЩИЙ РЕМОНТ

4.1 Ремонт контроллера осуществляет только изготовитель по гарантийным обязательствам.

4.2 Срок и стоимость работ по **не гарантийному ремонту** определяется после осмотра изделия специалистом изготовителя.

5 ХРАНЕНИЕ

5.1 При получении контроллеров следует убедиться в полной сохранности упаковки и транспортной тары. При наличии повреждений следует составить акт в установленном порядке и обратиться с рекламацией в транспортную организацию.

5.2 Контроллеры должны храниться в сухом и вентилируемом помещении при температуре окружающего воздуха от минус 50 до +70 °С и относительной влажности до 95 %. Воздух в помещении не должен содержать пыль и примеси агрессивных паров и газов.

6 ТРАНСПОРТИРОВАНИЕ

6.1 Транспортирование контроллеров допускается только в упаковке изготовителя и может производиться любым видом крытого транспорта.

6.2 Контроллер в транспортной таре выдерживает следующие механико-динамические нагрузки, действующие в направлении, обозначенном на таре манипуляционным знаком по ГОСТ 14192-96 «Верх»:

- вибрации по группе исполнения N2 (частота от 10 до 55 Гц, амплитуда смещения 0,35 мм) в соответствии с ГОСТ 12997-84;

– - удары со значением пикового ударного ускорения 100 м/с^2, длительностью ударного импульса 16 мс (число ударов не менее 1000).

6.3 Во время погрузочно-разгрузочных работ и транспортирования упакованные контроллеры не должны подвергаться резким ударам и воздействию атмосферных осадков. Способ укладки на транспортное средство должен исключать их перемещение.

6.4 Контроллеры после транспортирования необходимо выдержать в помещении с нормальными условиями не менее 3 ч, только после этого произвести распаковку.

7 СОПРОВОЖДЕНИЕ

7.1 Контроллер разработан и изготовлен в Республике Беларусь. Вы всегда можете получить квалифицированную информацию по телефону, по электронной почте или непосредственно в ОАО «Белэлектромонтажналадка» по любым вопросам, касающимся контроллера ПИКОН-МИКРО и другой нашей продукции. Информация обо всех разработках и изделиях нашего предприятия распространяется бесплатно. Вы можете получить ее в печатном виде, в виде файлов на дискете или по электронной почте. Мы также будем благодарны за все предложения по улучшению работы и модернизации изделия.

ПРИЛОЖЕНИЕ А

(обязательное)

ГАБАРИТНЫЕ РАЗМЕРЫ КОНТРОЛЛЕРА И СХЕМА ПОДКЛЮЧЕНИЯ

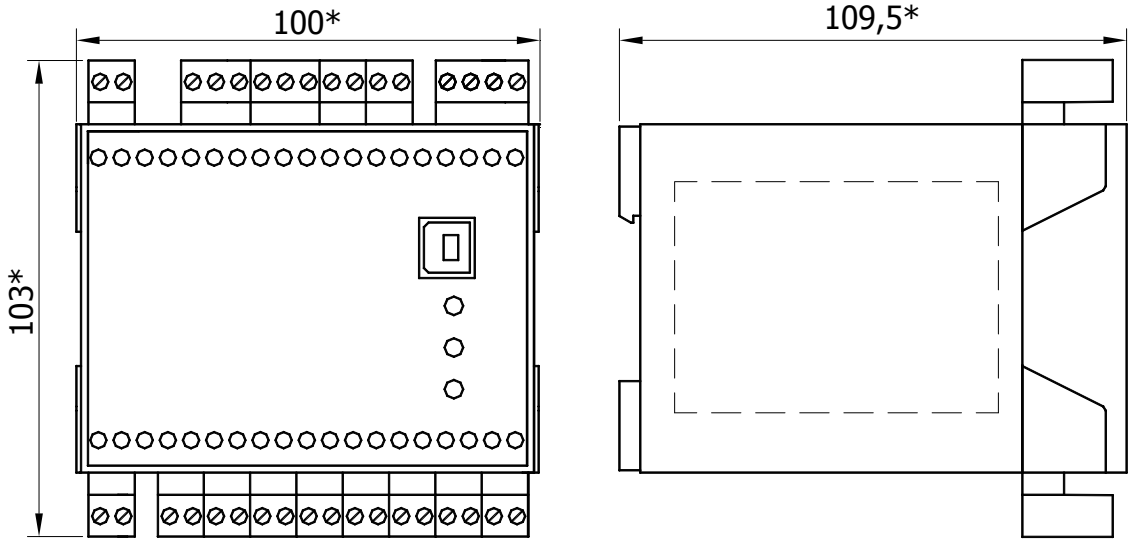


Рисунок А.1 – Габаритные размеры контроллера

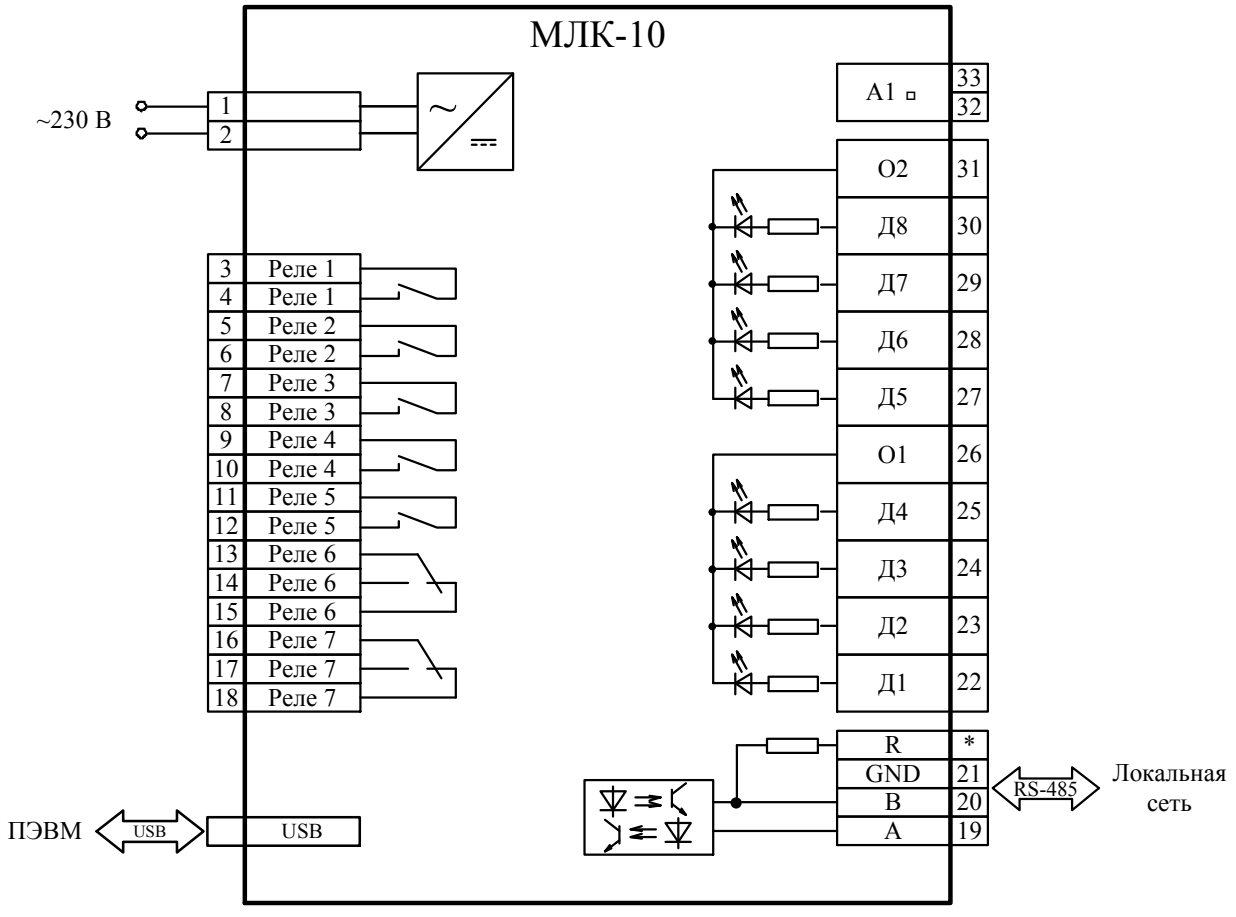


Рисунок А.2 – Схема подключения контроллера МЛК-10 (МЛК-11)

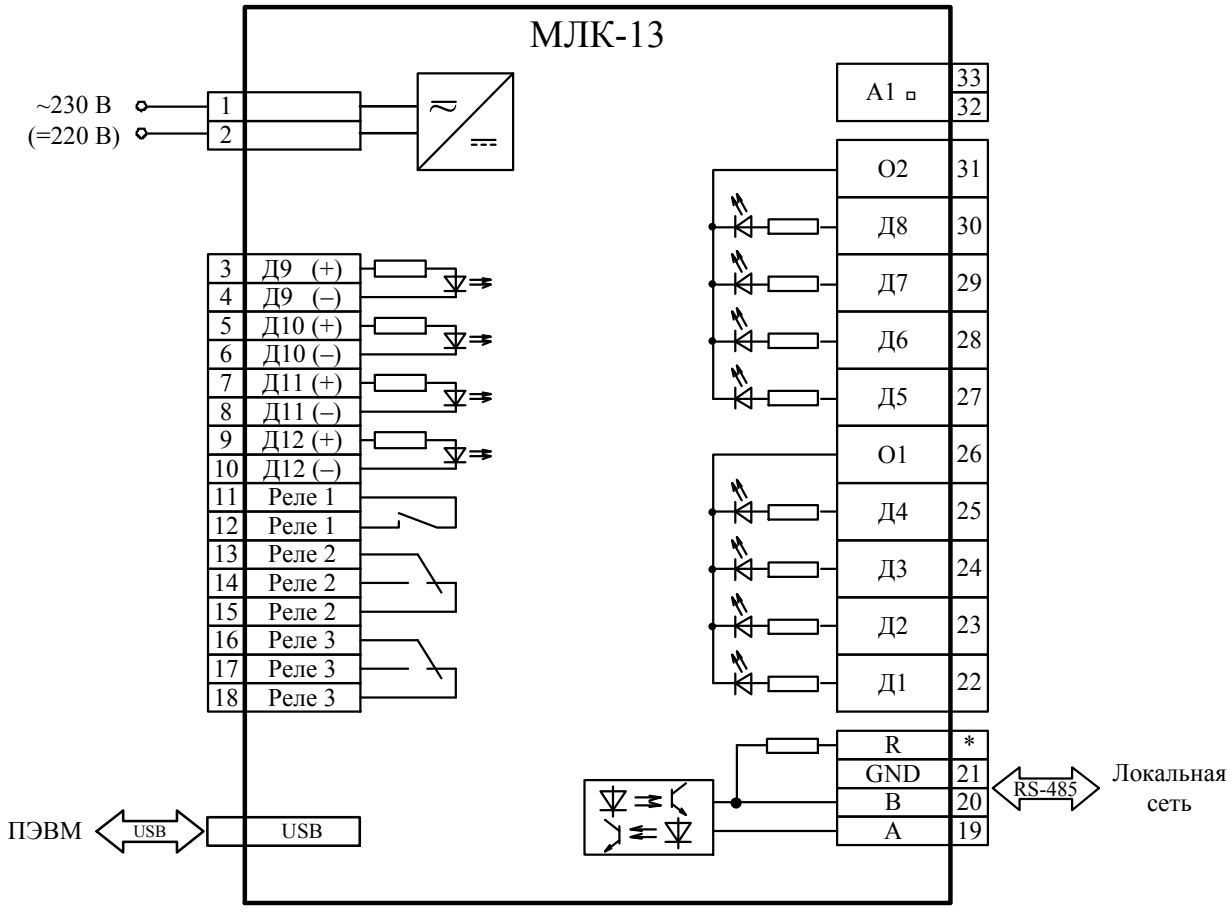


Рисунок А.3 – Схема подключения контроллера МЛК-13 (МЛК-12)

ПРИЛОЖЕНИЕ Б

(справочное)

МОНТАЖ НА DIN-РЕЙКУ

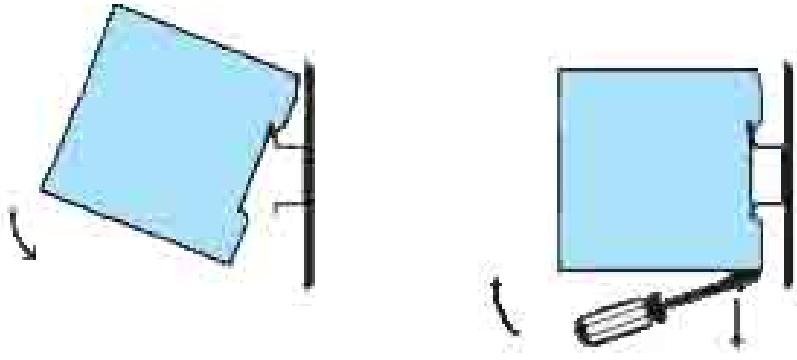


Рисунок Б.1 – Монтаж контроллера на DIN-рейку