Спецификация

Контроллер	
Процессор	RISC архитектура, 32-х разрядный ARM7, 48MHz
ОС	ОСРВ
Система программирования	Insyte конфигуратор v 2.x
Объем ОЗУ для хранения переменных программ	Переменные: 400 32 разрядных целых. 100 32 разрядных плавающая точка 100 таймеров с квантами 10 мсек
Аппаратные часы реального времени	есть, питание от встроенной батареи
Конструктивное исполнение	Моноблок, на ДИН – рейку
Тип, объем памяти хранения программ	64 KB
Сеть Modbus	До 100 узлов, 1000 переменных (Holding Registers, Input Registers, Coils, Discrete Inputs)
Резервное питание	Аккумулятор 1800mAh (опционально)
Питающее напряжение	Постоянное 6-36в
Человеко-машинный интерфейс	Светодиодная индикация состояния контроллера, кнопки старт / стоп, сброс

Периферия	
Дискретные входы на борту, тип, развязка	4, развязка групповая, 1500В. Max 48 В DC
Дискретные выходы на борту,	4, по 1 кВ 5А/240В AC 10А/28В DC
Аналоговые входы на борту	2, гальваническая развязка 1500В
Интерфейс датчиков температуры, ключей IButton	1 датчик температуры (DS1820) Количество ключей ограничено памятью программ.

Интерфейсы связи	
Управление сетью	RS485, гальваническая развязка 1500В (2400-115200, полностью настраиваемый)
Управление сторонней периферией	RS485/RS232, гальваническая развязка 1500В (только передача, полностью настраиваемый)
Управление контроллером	USB, Ethernet (ModbusTCP)
Удалённое управление	GSM Modem

Режимы.

Контроллер может находиться в одном из трёх состояний:

	«Стоп»	**«Старт»**	**«Пауза»**
Опрос сети	Нет	Выполняется согласно настройкам, узлов	Нет, однако контроллер сохраняет у себя всё состояние сети на момент подачи команды «Пауза»
Выполнение логики	Нет	Да, с квантами в 10 мс.	Нет

Описание адресного пространства контроллера.

Modbus пространство контроллера

прим. адрес контроллера всегда дожжен быть равен 1.

Регистры, функции (чтение – 3, запись – 6, 16), Holding Registers		
Адрес	**Доступ**	**Описание**
1	чтение	АЦП 1, 16 бит
2	чтение	АЦП 2, 16 бит
3	чтение	Внешнее питание, 16 бит
4	чтение	Температура, 16 бит
5	чтение/запись	32 разрядный счётчик входа 1, старшие 16 бит
6	чтение/запись	32 разрядный счётчик входа 1, младшие 16 бит
7	чтение/запись	32 разрядный счётчик входа 2, старшие 16 бит
8	чтение/запись	32 разрядный счётчик входа 2, младшие 16 бит
9	чтение/запись	32 разрядный счётчик входа 3, старшие 16 бит
10	чтение/запись	32 разрядный счётчик входа 3, младшие 16 бит
11	чтение/запись	32 разрядный счётчик входа 4, старшие 16 бит
12	чтение/запись	32 разрядный счётчик входа 4, младшие 16 бит
100	чтение/запись	Состояние: 0 – Стоп 1 – Старт 2 – Пауза
с 1001 по 2000	чтение/запись	32 разрядные переменные логики
с 2001 по 2200	чтение/запись	32 разрядные переменные логики (плавающая точка/float)
С 2201 по 2400	чтение/запись	32 разрядные таймеры логики (до 2.147.483.647)

Выходы, функции (чтение – 1, запись – 5), Coils		
Адрес	**Доступ**	**Описание**
1	чтение/запись	Выход 1
2	чтение/запись	Выход 2
3	чтение/запись	Выход 3
4	чтение/запись	Выход 4

Дискретные входы, функция – 2, Discrete Inputs		
Адрес	**Доступ**	**Описание**
1	чтение	Вход 1
2	чтение	Вход 2
3	чтение	Вход 3
4	чтение	Вход 4

Опрос сети:

Регистры, функции (чтение – 3, запись – 6)
Выходы, функции (чтение – 1, запись – 5)
Дискретные входы, функция – 2

Написание программ на языке высокого уровня.

API для работы с контроллером:

Прототип выполняемой функции (запускается с квантами в 10 мс).
Пользователю дополнительно доступно 1 кб. ОЗУ.

```
#define NULL  ((void*)0x00000000)
void main(  int *pTmpInt,
            float *pTmpFlt,
            int (*pInts), /* Массив челочисленных переменных 32 бит */
            float (*pFloats), /* Массив переменных с плавающей точкой 32 бит */
            int (pKeyH), /* 32 битный ключ 1-wire */
            int (pKeyL), /* 32 битный ключ 1-wire */
            int (*pGetNodeVar)( unsigned short dev_addr, unsigned short reg_addr ),
            int (*pSetNodeVar)( unsigned short dev_addr, unsigned short reg_addr,
            unsigned short val ),
            int (*pGetTimer)( unsigned short TimerNum ),
            int (*pSetTimer)( unsigned short TimerNum, int TimerVal ),
            int (*pReceiveSMS)( const char *smsnum, const char *smstext, int *parInt,
            float *parFlt ),
            int (*pSendSMS)( const char *smsnum, const char *smstext, int *parInt,
            float *parFlt ),
            int (*pSendCom2)( const char *TxData, unsigned short TxLen, const char
            *RxData, unsigned short RxLen),
            int (*pTimeCmp)( unsigned char date, unsigned char month, unsigned char
            year, unsigned char hour, unsigned char minute, unsigned char second,
            unsigned char day, unsigned char act ),
            float (*pSinf)(float _X),
            float (*pCosf)(float _X),
            float (*pLogf)(float _X),
            float (*pExpf)(float _X),
            float (*pPowf)(float _X1, float _X2),
            float (*pSqrtf)(float _X)
)
{
/* Подпрограмма логики */
}
```

Переменные, используемые конфигуратором для временных вычислений:
int *pTmpInt – указатель на массив 32 разрядных целочисленных знаковых переменных.
float *pTmpFlt – указатель на массив 32 разрядных переменных с плавающей точкой.

Переменные, доступные пользователю.
int *pInts – указатель на массив 32 разрядных целочисленных знаковых переменных.
float *pFloats – указатель на массив 32 разрядных переменных с плавающей точкой.
прим. для пользователя дополнительно доступно 512 байт **«не инициализированного»** ОЗУ, однако доступ к этим переменным может осуществляться только из «логики»

```
Пример обращения
pInts[0] = 1; // Присвоение единицы целочисленной переменной слот 0
pFloats[0] = 3.2; // Присвоение 3.2 вещественной переменной слот 0
```

Описание API:

int pGetNodeVar(unsigned short dev_addr, unsigned short reg_addr)
Возвращает значение переменной узла, в случае ошибки связи возвращает (-1)

unsigned short dev_addr – адрес устройства в сети
unsigned short reg_addr – адрес переменной

Пример использования

```
/* Проверка состояния входа 1 ПЛК */
if( pGetNodeVar(1, 10001) == 1 )
{
    // Действие
}

/* Проверка состояния связи с диммером (вход 1) */
if( pGetNodeVar(10, 10001) == (-1) )
{
    // Действие, если связи нет
}
```

int pSetNodeVar(unsigned short dev_addr, unsigned short reg_addr, unsigned short val)
Записывает значение в регистр устройства, в случае ошибки связи возвращает (-1)

unsigned short dev_addr – адрес устройства в сети
unsigned short reg_addr – адрес переменной
unsigned short val – записываемое значение

Пример использования

```
/* Проверка состояния входа 1 ПЛК */
if( pGetNodeVar(1, 10001) == 1 )
{
    // Устанавливаем 1 на выход 2 ПЛК
    pSetNodeVar(1, 2, 1);
}

if( pGetNodeVar(1, 10001) == (-1) )
{
    // Устанавливаем яркость 100% для диммера по адресу 10
    pSetNodeVar(10, 40002, 100);
}
```

int pGetTimer(unsigned short TimerNum)
Возвращает значение таймера кратное 10 мс:
(0) – таймер закончил счёт.
(-1) – таймер выключен.

unsigned short TimerNum – номер таймера

Пример использования, см. pSetTimer

Int pSetTimer(unsigned short TimerNum, int TimerVal)
Взводит таймер

unsigned short TimerNum – номер таймера.
unsigned short TimerVal – значение таймера, кратное 10 мс.

```
Пример использования
Переключение реле 2 ПЛК с частотой один раз в секунду

/* Проверка состояния таймера 0 ПЛК */
if( pGetTimer(0) == (-1) ) // Таймер закончил счёт и выключился
{
    // Устанавливаем таймер 0 на отсчёт 1 секунды (10 мСек *100)
    pSetTimer(0, 100);

    if( pGetNodeVar(1, 2) == 1 ) //Если выход 2 ПЛК == 1
    {
        // Записываем 0
        pSetNodeVar(1, 2, 0);
    }
    else if( pGetNodeVar(1, 2) == 0 ) //Если выход 2 ПЛК == 0
    {
        // Записываем 1
        pSetNodeVar(1, 2, 1);
    }
}
```

int pReceiveSMS(const char *smsnum, const char *smstext, int *parInt, float *parFlt)
Проверяет есть ли принятая смс с подходящими параметрами, в случае успеха возвращает 0, иначе (-1)

const char *smsnum – номер с которого пришла SMS в формате UTF8
const char *smstext - текст SMS в формате UTF16
int *parInt – указатель на целочисленную переменную в которую будет записано значение принятого параметра.

float *parFlt – указатель на переменную с плавающей точкой в которую будет записано значение принятого параметра.

int pSendSMS(const char *smsnum, const char *smstext, int *parInt, float *parFlt)
Отправляет SMS на указанный телефонный номер.

const char *smsnum – номер с которого пришла SMS в формате UTF8
const char *smstext - текст SMS в формате UTF16
int *parInt – указатель на целочисленную переменную, которая будет отправлена в сообщении после текста.
float *parFlt – указатель на переменную с плавающей точкой, которая будет отправлена в сообщении после текста.

int pSendCom2(const char *TxData, unsigned short TxLen, const char *RxData, unsigned short RxLen)
Отправляет данные в последовательный порт №2

const char *TxData – указатель на буфер
unsigned short TxLen – длина отправляемых данных
const char *RxData – указатель на буфер
unsigned short RxLen – длина принимаемых данных

int pTimeCmp(unsigned char date, unsigned char month, unsigned char year, unsigned char hour, unsigned char minute, unsigned char second, unsigned char day, unsigned char act)
Сравнивает время/дату с текущей, в случае успеха возвращает (0), иначе (-1)

unsigned char date – число
unsigned char month – месяц
unsigned char year – год
unsigned char hour – часы
unsigned char minute – минуты
unsigned char second – секунды
unsigned char day – день недели, 1 – понедельник, 7 – воскресенье
unsigned char act – операция, 1 – равно, 2 – не равно, 3 – больше, 4 – меньше
Значение 255(0xFF) – игнорировать аргумент.

http://rishida.net/tools/conversion/

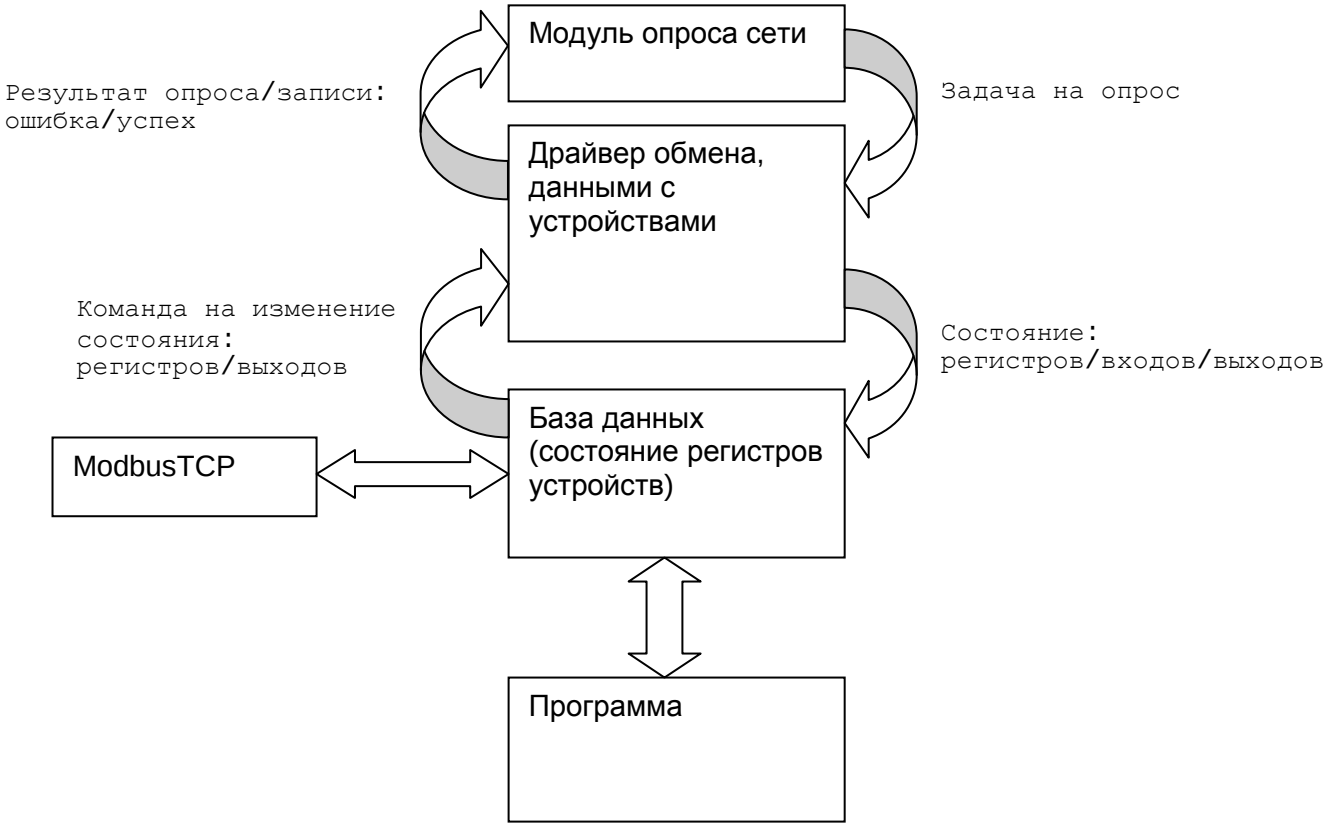
Цифры 0-9

0-0030	1-0031	2-0032	3-0033	4-0034
5-0035	6-0036	7-0037	8-0038	9-0039

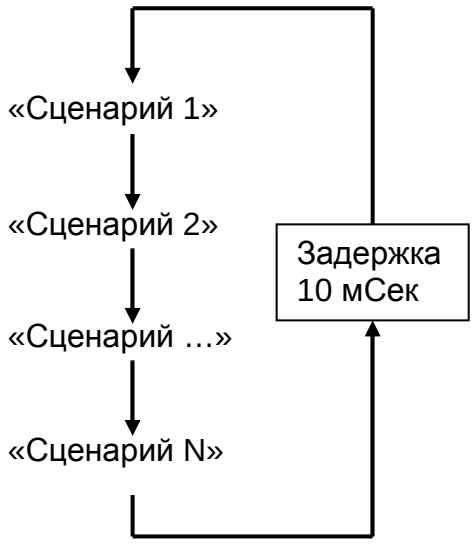
Латиница a-z, A-Z

a-0061	n-006E	A-0041	N-004E
b-0062	o-006F	B-0042	O-004F
c-0063	p-0070	C-0043	P-0050
d-0064	q-0071	D-0044	Q-0051
e-0065	r-0072	E-0045	R-0052
f-0066	s-0073	F-0046	S-0053
g-0067	t-0074	G-0047	T-0054
h-0068	u-0075	H-0048	U-0055
i-0069	v-0076	I-0049	V-0056
j-006A	w-0077	J-004A	W-0057
k-006B	x-0078	K-004B	X-0058
l-006C	y-0079	L-004C	Y-0059
m-006D	z-007A	M-004D	Z-005A

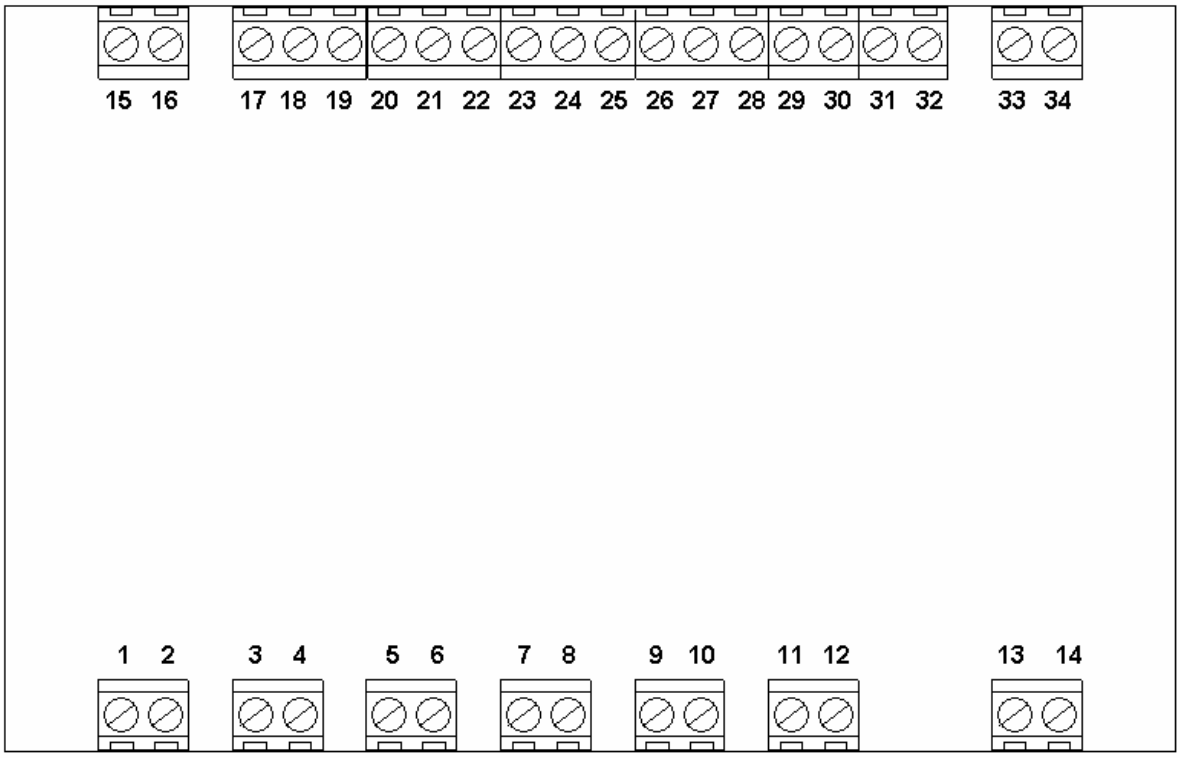
Принцип работы контроллера



Выполнение программы

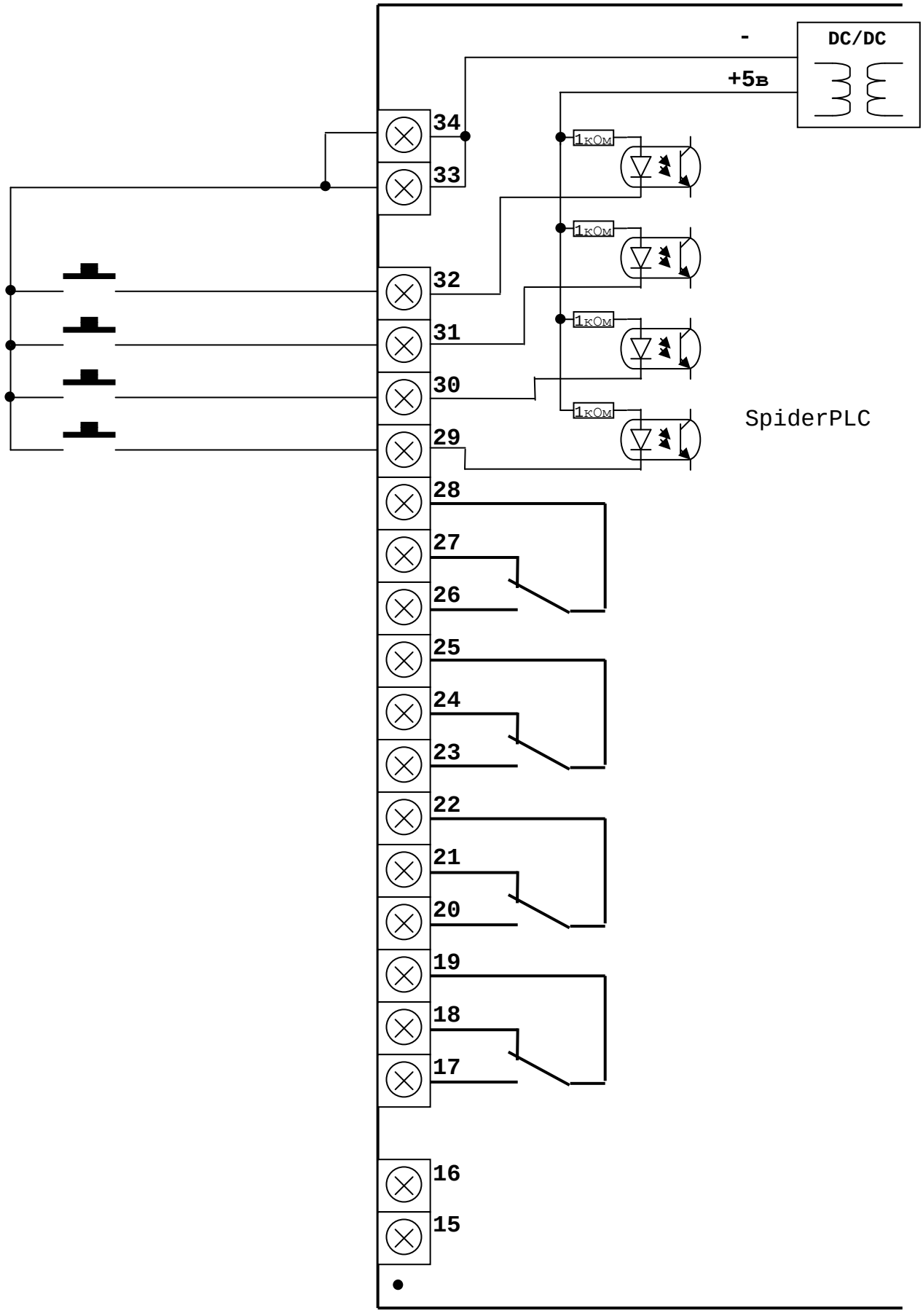


Подключение.



1	RS485 “A1”	15	Вход “-” питания контроллера
2	RS485 “B1”	16	Вход “+”питания контроллера
3	RS485 “GND1”	17	Реле 1 нормально разомкнутый контакт
4	RS485/RS232 “GND2”	18	Реле 1 нормально замкнутый контакт
5	RS485/RS232 “A2”/TXD	19	Реле 1 общий контакт
6	RS485/RS232 “B2”/RXD	20	Реле 2 нормально разомкнутый контакт
7	One Wire DATA	21	Реле 2 нормально замкнутый контакт
8	One Wire GND	22	Реле 2 общий контакт
9	АЦП 1 “+”	23	Реле 3 нормально разомкнутый контакт
10	АЦП 1 “–”	24	Реле 3 нормально замкнутый контакт
11	АЦП 2 “+”	25	Реле 3 общий контакт
12	АЦП 2 “–”	26	Реле 4 нормально разомкнутый контакт
13	Выход питания датчиков +5”/”+12”	27	Реле 4 нормально замкнутый контакт
14	Выход питания датчиков “GND”	28	Реле 4 общий контакт
		29	Вход 1
		30	Вход 2
		31	Вход 3
		32	Вход 4
		33	Общий входов
		34	Общий входов

Подключение логических входов и релейных выходов



Подключение аналоговых входов и датчика температуры