

ООО «ИВС-МИКОНТ»
Конструкторское бюро

КОНТРОЛЛЕРЫ
« МИКОНТ »

РУКОВОДСТВО ПО ЭКСПЛУАТАЦИИ

ПРИЛОЖЕНИЕ 3
Организация связи с контроллером.

Пермь 2002-2007

1. ОБЩИЕ СВЕДЕНИЯ

Контроллеры "МИКОНТ" обычно имеют не менее двух последовательных устройства связи RS232 (технологический порт) и RS485 (режим-полудуплекс). Для организации связи контроллера с компьютером можно использовать все устройства одновременно. Обычно используется RS485. RS485 обеспечивает возможность подключения нескольких контроллеров (до 32) к одному компьютеру, через конвертор RS232↔RS485. В этом случае необходимо присвоить каждому контроллеру свой уникальный сетевой номер 1..254 (меню "Настройка связи").

Программное обеспечение контроллера позволяет ввести следующие параметры связи (меню-Настройка связи):

- Сетевой номер контроллера, обеспечивающий индивидуальное обращение к контроллеру.
- Скорость обмена (для RS232 и RS485).
- Протокол обмена (не на всех контроллерах).

Протокол обмена данными с контроллером построен на базе известного протокола «**Modbus**» (режим ASCII) и называется **MicontBus** (ASCII) 5040h. Контроллеры могут быть связаны в сеть. Каждый контроллер имеет свой сетевой номер или адрес. Связь осуществляется по запросу. Ведущий контроллер или компьютер (master) выдает в линию связи запрос и ждет ответа от подчиненного контроллера (slave). Максимальное время ожидания ответа не превышает 125 мс. Время дано без учета скорости обмена.

Запрос и ответ имеют определенную структуру и оформляются в кадр.

Для организации связи компьютера с контроллером поставляется библиотека «**MICCLNK.DLL**» (WINDOWS 95/98/2000/XP). Библиотека включает в себя функции кодирования, декодирования, формирования кадра, а также настройки портов RS232 (COM1, COM2, ...) их тестирования. Библиотека обеспечивает обмен данными с контроллером. Планируется реализация протокола «**Modbus**» (режим **Remote Terminal Unit**).

2. ОСОБЕННОСТИ РАБОТЫ С УСТРОЙСТВОМ RS485

Устройство RS485 предназначено для организации связи на длинных линиях (до 1 000 м) и обеспечивает подключение нескольких контроллеров (до 32). RS485 контроллера работает в режиме-полудуплекс и поддерживает скорости обмена до 115200 бод.

2.1. Конвертор RS232-RS485.

Для подключения контроллера к компьютеру через устройство RS485 необходим конвертор RS232-RS485, подключаемый со стороны компьютера к RS232 (COM1..COM4). Можно использовать любой конвертор не ориентированный на конкретный протокол или особенности работы других контроллеров.

При подключении контроллера через конвертор RS232-RS485, необходимо установить два терминальных резистора 120 Ом \pm 10%, 0.5 Вт. Резисторы подключаются между линиями "А" и "В". (один на конверторе, другой на самом удаленном контроллере).

2.2. Конвертор RS232-RS485 "MICONТ-CL485".

С контроллерами "MICONТ" поставляется конвертор RS232-RS485 в двух исполнениях:

- **MICONТ-CL485-10-PC** (для подключения к компьютеру).
- **MICONТ-CL485-10-M** (для подключения к модему).

Цифры "00" вариант/версия устройства.

Рис. 2.2-1. Конвертор RS232-RS485 "MICONТ-LC485-10-PC"

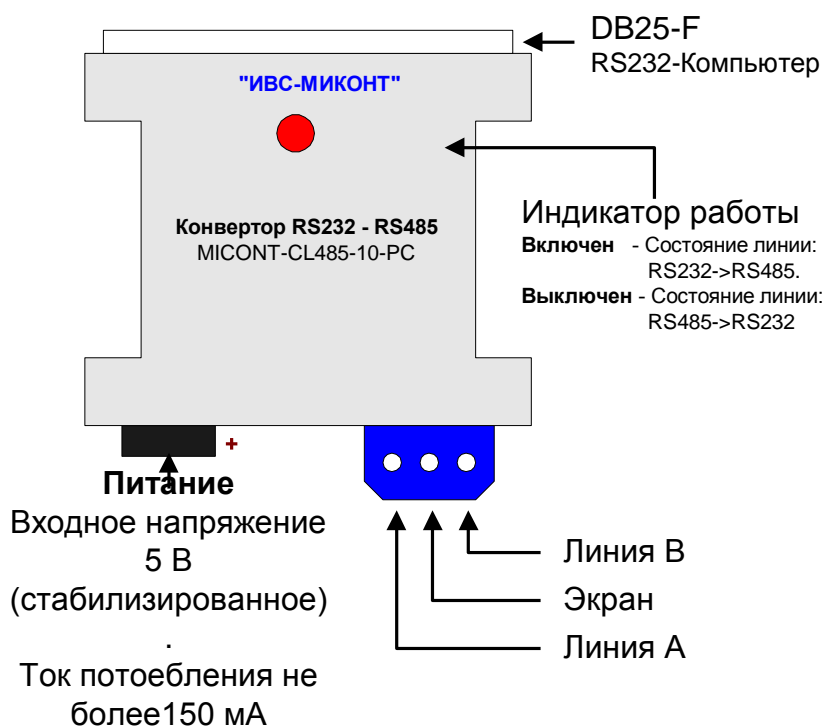
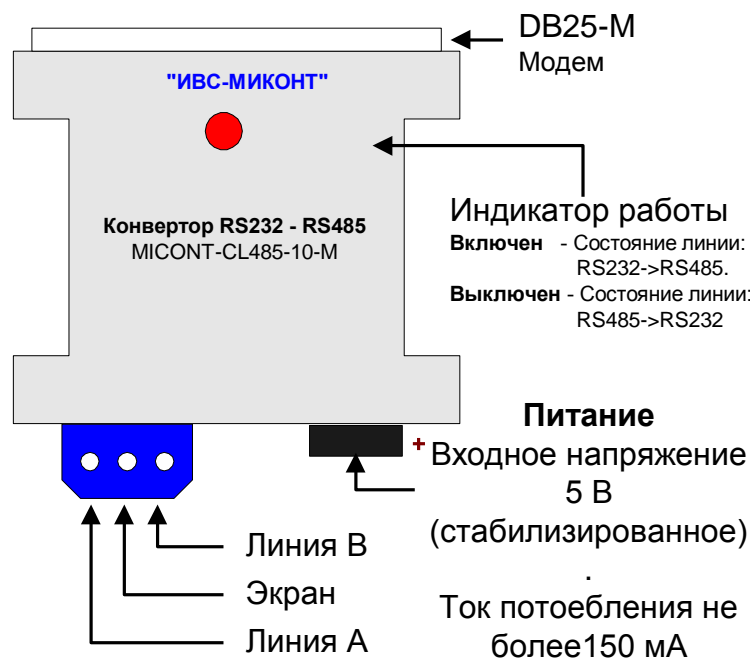
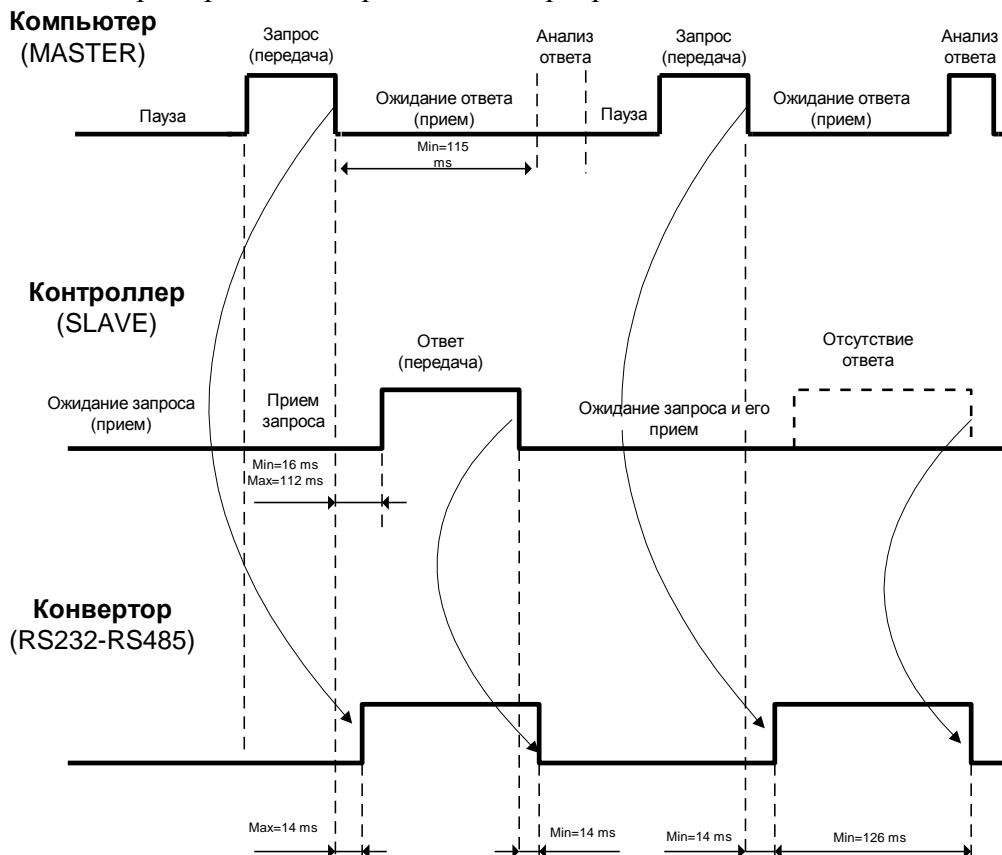


Рис. 2.2-2. Конвертор RS232-RS485 "MICONT-LC485-10-M"



Конвертор обеспечивает скорость обмена до 115200 бод и автоматическое управление направлением передачи с учетом временных характеристик протоколов обмена контроллеров "МИКОНТ".

Рис. 2.2-3. Примерная схема работы конвертора MICONT-CL485

**ПРИМЕЧАНИЯ:**

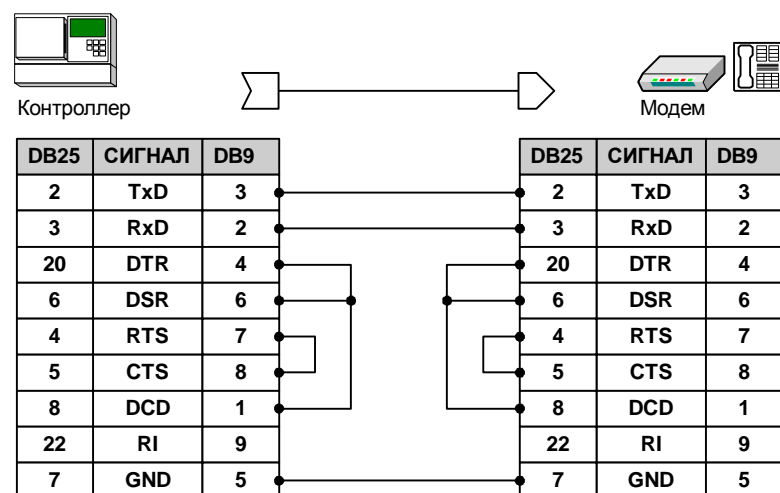
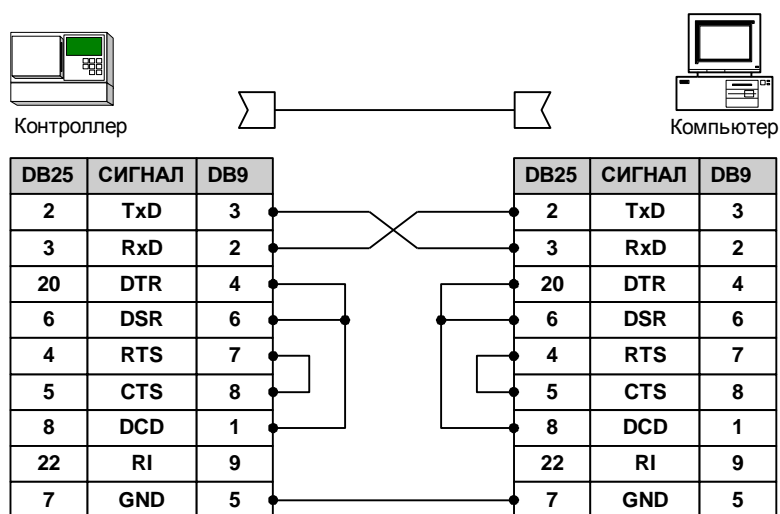
1. Начальное состояние конвертора - передача от РС к контроллеру.
2. При обнаружении сигнала со стороны РС, конвертор автоматически переводит линию в состояние передачи от РС к контроллеру.

ЮТЕРУ

3.1. Распайка кабелей связи RS232.

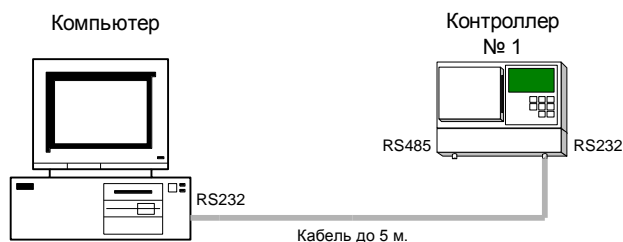
Рис. 3.1-1. Распайка кабелей связи RS232

DB25	СИГНАЛ	DB9
2	TxD-Передаваемые данные	3
3	RxD-Принимаемые данные	2
20	DTR-Готовность терминала	4
6	DSR-Готовность модема	6
4	RTS-Запрос передатчика	7
5	CTS-Сброс передатчика	8
8	DCD-Указатель несущей	1
22	RI-Указатель вызова	9
7	GND-"Земля" сигналов	5



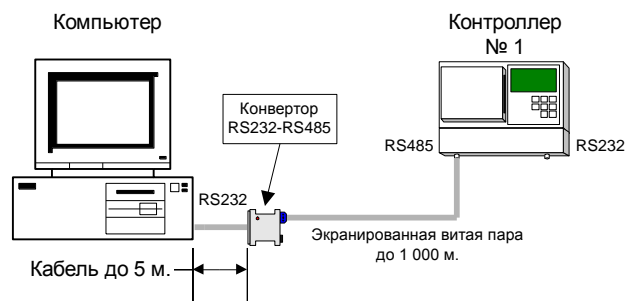
3.2. Прямое подключение одного контроллера через RS232

Рис. 3.2-1. Прямое подключение через RS232



3.3. Подключение одного контроллера через конвертор RS232-RS485.

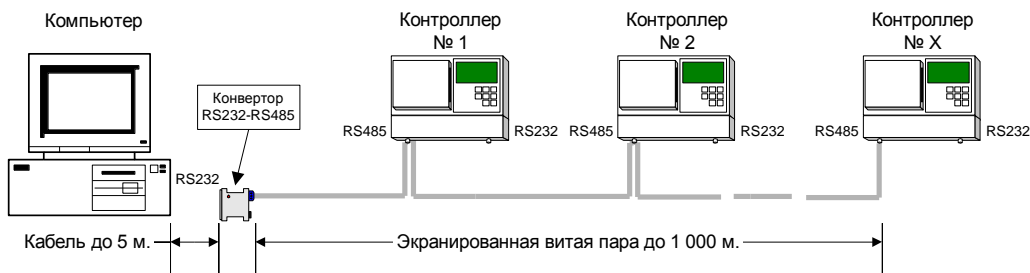
Рис. 3.3-1. Подключение через конвертор RS232-RS485



При подключении контроллера через конвертор RS232-RS485, необходимо установить два терминальных резистора $120 \text{ Ом} \pm 10\%$, 0.5 Вт. Резисторы подключаются между линиями "А" и "В". (один на конверторе, другой на контроллере).

3.4. Сетевое подключение через конвертор RS232-RS485.

Рис. 3.4-1. Сетевое подключение через конвертор RS232-RS485



При сетевом подключении контроллеров необходимо каждому контроллеру присвоить свой уникальный сетевой номер (1..254), установить для всех одинаковую скорость обмена и протокол обмена. Эти параметры устанавливаются на контроллерах в меню "Настройка связи". В сетевом варианте можно подключить до 32 контроллеров.

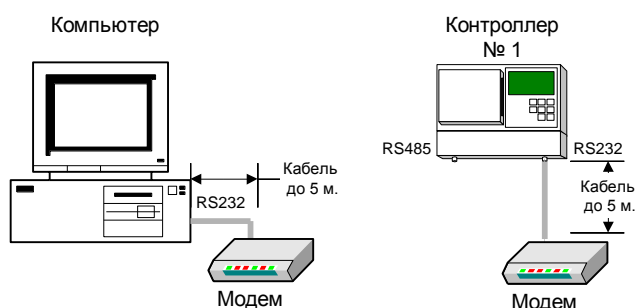
При подключении контроллера через конвертор RS232-RS485, необходимо установить два терминальных резистора $120 \text{ Ом} \pm 10\%$, 0.5 Вт. Резисторы подключаются между линиями "А" и "В". (один на конверторе, другой на самом удаленном контроллере).

3.5. Подключение с использованием модемов.

При организации связи с контроллером/контроллерами по телефонной линии используются два модема, один со стороны компьютера другой со стороны контроллера.

3.5.1. Подключение одного контроллера через модем.

Рис. 3.5.1-1. Подключение одного контроллера через модем



3.5.2. Подключение одного контроллера через модем и конвертор MICONT-CL485.

Такое подключение используется если контроллер удален от модема на расстояние более 5 метров. В этом случае модем со стороны контроллера подключается через конвертор RS232-RS485 к устройству RS485 на контроллере. Конвертор должен обеспечивать автоматическое переключение направления передачи.

Рекомендуется использовать конвертор "MICONT-CL485-10-M" или "MICONT-CL485-10-PC", но со специальным переходником для модема.

Правила подключения через конвертор см. в п.п.2.1, п.п.3.3.

3.5.4. Программирование/настройка модемов.

Организация связи с контроллерами через модем требует определенной настройки модемов как со стороны компьютера так и со стороны контроллера.

Настройку со стороны компьютера должна обеспечивать программа связи. Настройка модема со стороны контроллера осуществляется на компьютере любыми подручными средствами. После настройки модема со стороны контроллера, модем подключается к контроллеру.

Система команд модема может отличаться от модели к модели модема. Для определенности будем использовать систему команд модема **ACORP (56000)**.

Полезные команды:

- | | |
|-------|-------------------------------------------------------------|
| AT&Wn | - Сохранить текущие настройки в конфигурации №n (n=0,1, ..) |
| AT&Fn | - Загрузить стандартную конфигурацию №n |
| AT&V | - Отчет о конфигурациях. |

Установка скорости МОДЕМ ⇔ RS232:

Модем сам определяет скорость обмена с RS232 (если нет запрета), по принимаемым данным от RS232 (автоподстройка). Если ничего не было принято от RS232, а модему надо передать в него данные, то передача осуществляется на скорости по умолчанию или последней используемой.

За параметры связи с RS232 отвечает регистр S23 – конфигурационная битовая карта.

ATS23=n (n-десятичное значение байта).

Биты:

- | | |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>0</u> | - Реакция на запрос на дист.цифровой петлевой тест. |
| <u>3-1</u> | - Скорость последовательного порта:
321
000 - 0-300 бит/сек.
001 -300, 600 бит/сек.
010 -1200, 2400 бит/сек.
011 -2400 бит/сек.
100 -4800 бит/сек
101 -9600 бит/сек
110 -19200 бит/сек
111 -38400 бит/сек |
| <u>5-4</u> | - Контроль четности для последовательного порта.
54
00 Четность (EVEN)
01 Константный SPACE-контроль
10 Не четность (ODD)
11 Константный MARK-контроль |
| <u>7-6</u> | - Защитные тональные сигналы.
76
00 Запрещены (AT&G0)
01 Защитный тональный сигнал 550 Гц (AT&G1)
10 Защитный тональный сигнал 1800 Гц (AT&G1)
11 Резерв |

Четность=EVEN,

Скорость 4800 (n=8), 9600 (n=10), 9200 (n=12), 8400 (n=14).

Для установки скорости необходимо:

- Открыть COM-порт с устанавливаемой скоростью;
- Проверить регистр S23 (ATS23?);
- Сохранить настройки (AT&Wn).

Пример программирования модема со стороны контроллера.

Предварительно выполнены команды:

- | | |
|-------|------------------------------------------------|
| ATE0 | - Выключить ЭХО |
| ATL3 | - Уровень громкости динамика |
| AT&D0 | - Опция сигнала готовности (DTR), связана с &Q |
| AT&C1 | - Опция сигнала "Указатель несущей" (DCD) |

ATS0=2	- Число звонков до ответа модема
AT&K0	- Запрет управления потоком данных

Для программирования модема можно использовать программу MC_RSTST.EXE.

Пример использования для COM2:

```
mc_rstst -o2 -b19200 -pE -d8 -s1 _prg.cmd
```

где: _prg.cmd –специальный файл команд для модема, содержащий следующие команды:

```
>ATZ0
w3
:ком. ATE0
>ATE0
w3
:ком. ATL3
>ATL3
w3
:ком. ATS0
>ATS0=2
W2
:ком. AT&D0
>AT&D0
W2
:ком. AT&C1
>AT&C1
W2
:ком AT&W0
>AT&W0
```

ВНИМАНИЕ

В программах верхнего уровня (SPOON, ...), для данной настройки, необходимо установить размер порции в 300 байтов (меню настроек) и флаг использования модема в настройках порта.

4. ПРОТОКОЛ "MicontBus (ASCII) 5040h".

4.1. ФОРМАТ КАДРА В РЕЖИМЕ ASCII

В этом режиме каждый байт послышки/сообщения, за исключением трех байт, кодируется двумя символами/байтами. На пример значение байта представляется в HEX-формате (10='0A'h, 32='20'h и т.д). Идея заключается в раздельной передаче старшей и младшей тетрады байта с известным дополнением ограничивающим нижнюю и верхнюю границы полезных данных. Это позволяет оставшиеся значения (0..X...Y...255) использовать в качестве контрольных или управляющих символов. На пример можно к младшей тетраде добавлять 40h, а к старшей 50h. Это дополнительный контроль ошибок.

Пример 4.1-1. Кодирование байта

```
A=12;
B=A >> 4;
A=A & 0x0F;
B=B | 0x50;           //Ст.тетрада = 50h ='P'
A=A | 0x40;           //Мл.тетрада = 4Ah ='J'
```

Формат кадра приведен в Таб.4.1-1, где символом '*' отмечены поля не подлежащие кодировке (управляющие символы).

Таб.4.1-1 Формат кадра

* Начало кадра	Сетевой адрес (IDnet)	Функция (CMD)	Данные (DATA)	Контрольная сумма	* EOF	* Готовность приема
:					CR	LF

Как видно из таблицы минимальная посылка занимает 9 байт (поле *Данные* может отсутствовать).

Начало кадра извещает приемник о начале посылки (синхронизация приема). Размер = 1-байт..

EOF+Готовность приема указывают на конец кадра. *LF*-сообщает о готовности передающей стороны к приему.

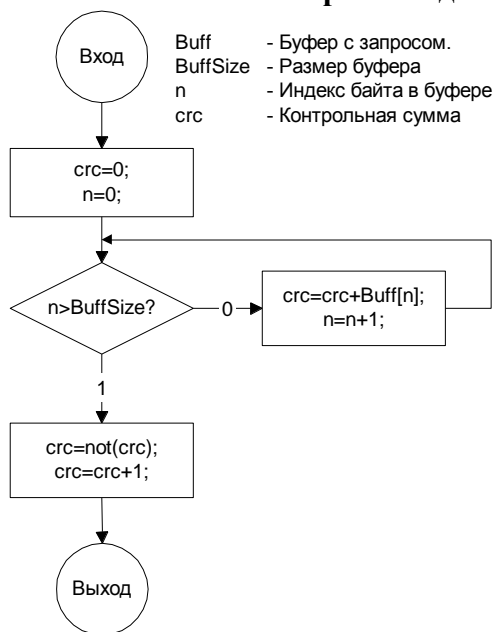
Сетевой адрес или адрес подчиненного устройства (0-255). Кодировается двумя символами. Адрес 0 интерпретируется как широковещательная команда и не требует ответа.

Функция или команда которую необходимо выполнить подчиненному устройству. Байт делится на поле команды (младшая тетрада) и поле результата (старшая тетрада). Источник команд при передаче всегда сбрасывает старшую тетраду, а исполнитель устанавливает ее значение в зависимости от результата выполнения. Если команда выполнена, то это поле (старшая тетрада) не равна 0, младшая тетрада остается неизменной. Код функции/команды кодируется двумя символами.

Данные содержат информацию необходимую для выполнения команды. Каждый байт данных кодируется двумя символами.

Контрольная сумма (Longitudinal Redundancy Check) - средство контроля ошибок. Для подсчета используются три поля - *Сетевой адрес*, *Функция*, *Данные*. Число образуется путем сложения байтов без переноса и вычисления дополнительного кода результата.

Рис. 4.1-1. Алгоритм подсчета контрольной суммы (LRC)



4.1.1. Обработка кадра (режим ASCII)

Для организации связи компьютера с контроллером поставляется библиотека «MICCLNK.DLL» (WINDOWS95/98) и рекомендуется для использования в программном обеспечении сторонних организаций. Для организаций у которых существует необходимость в создании собственных функций предлагаются алгоритмы приведенные ниже.

Пример 4.1.1-1. Формирование кадра

```

#define NET_ADD_HI  0x50
#define NET_ADD_LO  0x40
#define NET_START   ':'
#define NET_END_DATA 0x0D
#define NET_END     0x0A

#define NET_SZMAX_DATA  1024                //Размер данных
#define NET_SZMAX_REQST (1+2+2+2+4+(NET_SZMAX_DATA)+2) //Размер запроса
#define NET_SZMAX_OUT   (1+(NET_SZMAX_REQST*2)+2+1+1+1+2) //Размер Вых.буф
#define NET_SZMAX_INP   (NET_SZMAX_OUT*2+1024)           //Размер Вх.буф.
                                                         // с учетом ":\XA"

//Тип трансляции (протокола)
#define TPTRANS_ADD  0 //ASCII (add_Hi, add_Lo)
#define mbSZ_BufOut NET_SZMAX_OUT

WORD __stdcall mbNetFrmBufOut(
    BYTE *BufReqst,
    BYTE *BufOut,
    WORD szBufR,
    int type_trans
)
/*
    Формирует буфер для вывода (начало кадра, ..., стс, CR, LF)
    Вход:

```

SzBufR = Размер запроса
 BufReqst = данные для запроса
 BufOut = буфер для формирования выходных данных
 type_trans=Тип трансляции:
 0 = MODBUS ASCII (add_HI,add_Lo)

Выход:

Размер выходного буфера, если 0 то ошибка иначе

BufOut = NET_START, DATA, CRC, NET_ENDDATA, NET_END, 0

```

*/
{
    WORD i,sz;
    BYTE *BfR, *BfO, b, hi,lo,crc;

    if ((BufReqst==NULL)|| (BufOut==NULL)) return(0);
    *BufOut=0;
    if (szBufR>=NET_SZMAX_REQST) return(0);
    BfR=BufReqst;
    BfO=BufOut;
    sz=0;
    *BfO++=NET_START;
    sz++;
    crc=0;
    for(i=0;i<szBufR;i++)
    {
        b=*BfR++;
        crc+=b;
        hi=b;
        hi>>=4;
        hi|=NET_ADD_HI;
        lo=b;
        lo&=0x0F;
        lo|=NET_ADD_LO;
        *BfO++=hi;
        *BfO++=lo;
        sz+=(WORD)2;
    }
    crc=(BYTE)~crc;
    crc++;
    b=crc;
    hi=b;
    hi>>=4;
    hi|=NET_ADD_HI;
    lo=b;
    lo&=0x0F;
    lo|=NET_ADD_LO;
    *BfO++=hi;
    *BfO++=lo;
    *BfO++=NET_END_DATA;
    *BfO++=NET_END;
    sz+=(WORD)4;
    *BfO=0;
    return(sz);
}/* mbNetFrmBufOut */

```

Пример 4.1.1-2. Декодирование кадра

```

int __stdcall      mbNetExtractInpData(BYTE *BufInp, //Полученный буфер
                                     BYTE *BufRes, //Буфер результата
                                     WORD szBufInp, //Размер INP буфера
                                     WORD *szRes,  //Размер результата
                                     int type_trans //Тип трансляции
                                     )
/*
Извлечение и декодирование последнего сообщения
Выход:
    Результат разбора: 0=Ok и SzRes=Размер результата,
                      BufRes=результат без обрамления
    ИНАЧЕ КОД ОШИБКИ:
    1=Ошибка входных данных;
    2=Нет стартового символа;
    3=Ошибка кодировки данных;
    4=Нет конца данных;
    5=Нет конца посылки;
    6=Ошибка CRC;
    7=Ошибка размера результата;
*/
{
    int i,r;
    WORD sz,k;
    BYTE *bf,*bfs,b,hi,lo,crc;

    if ((BufInp==NULL)|| (BufRes==NULL)|| (szRes==NULL)|| (szBufInp==0))
    {
        return(1);
    }
    r=0;
    //Поиск последнего сообщения
    bf=&BufInp[szBufInp-1];
    sz=szBufInp;
    i=szBufInp-1;
    k=0;
    while(i>=0)
    {
        k++;
        b=*bf;
        if (b==NET_START) break;
        if (((b==NET_END)||(b==NET_END_DATA))&&(k>6)) return(2);
        bf--;
        i--;
    }
    if (*bf!=NET_START) return(2);
    bfs=BufRes;
    sz=0;
    crc=0;
    bf++;
    i=0;
    r=0;
    k>>=1;
    while((WORD)i<=k)
    {
        b=*bf++;
        if (b==NET_END_DATA)
        {
            if (*bf==NET_END)
            {
                if (crc==0)
                {

```

```
    r=1;
    break;
}
// Ошибка CRC
return(6);
}
else
{
    //Нет конца посылки
    return(5);
}
}
if ((b<NET_ADD_HI)||(b>(NET_ADD_HI | 0x0F)))
{ //Ошибка кодировки
    return(3);
}
hi=b;
hi&=0x0F;
hi<=4;
b=*bf++;
if ((b<NET_ADD_LO)||(b>=NET_ADD_HI))
{ // Ошибка кодировки
    return(3);
}
lo=b;
lo&=0x0F;
b=hi | lo;
*bfs+=b;
crc+=b;
i++;
sz++;
}
*szRes=sz;
if (sz<2)
{
    //Ошибка размера результата;
    return(7);
}
if (r==0)
{ //Ошибка конца данных
    return(4);
}
sz--;
*szRes=sz;
*bfs=0;
return(0);
}/* mbNetExtractInpData */
```

4.2. ДОСТУП К ДАННЫМ КОНТРОЛЛЕРА

Данными контроллера являются значения переменных расчета, записи журнала и различные области памяти (ОЗУ, ПЗУ). Данные разбиты на группы. Каждая группа имеет свой номер или идентификатор (см. Таб.2.1-1). В различных проектах число идентификаторов может отличаться. При попытке доступа к несуществующей группе данных в ответе контроллера будет содержаться код ошибки.

4.2.1. Группы данных.

Идентификаторы переменных интерпретатора/расчетов имеют номера от 0 до 254.

Структуры групп приведены в руководстве по эксплуатации контроллера «Структуры и форматы данных контроллера».

Таб.4.2.1-1 Идентификаторы групп данных

Имя	Код (HEX)	Пояснения
NETVAR_MEM	0800	Начало ОЗУ
NETVAR_FLASH	0900	Начало FLASH
NETVAR_SZB_BUF	0A00	Размер декодированного буфера обмена (в байтах)
NETVAR_F_REC	0AFF	Переменная для поиска записей в журнале по дате и времени
NETVAR_HEAD	0B00	Заголовок журнала
NETVAR_DATA	0B01	Данные журнала
NETVAR_VERSION	0B02	Версия (строка символов с завершающим нулем)
NETVAR_VERLINK	0B03	Версия обработчика запросов по связи
NETVAR_SCR	0C00	Буфер экрана
NETVAR_KBD	0C01	Буфер клавиатуры
NETVAR_DTTM	0C02	Системная дата и время контроллера
NETVAR_POWER	0C04	Усети, Fсети, температура внутри корпуса
NETVAR_CALC0	0D00	Расчет 0
NETVAR_CALC1	0D01	Расчет 1
NETVAR_CALC2	0D02	Расчет 2
NETVAR_CALC3	0D03	Расчет 3
NETVAR_TVNAME	0D04	Таблица описания переменных
NETVAR_TCNAME	0D05	Таблица описания констант
NETVAR_CONST	0D06	Значения констант
NETVAR_CALC_ON	0D07	Управление включением/выключением обработки расчетов (работа интерпретатора)
NETVAR_CVIDEO	0D08	Описание видеок кадров
NETVAR_CMSG	0D09	Описание сообщений
NETVAR_CUSER	0D0A	Список пользователей
NETVAR_CTEXT	0D0B	Паспорт прибора, информация сервис центра, описание расчетной части
NETVAR_RESET	0E00	Рестарт контроллера (только запись, ответа нет)

Как сказано выше, состав переменных зависит от операционной системы или от проекта (варианта контроллера).

Под проектом будем понимать совокупность программно-аппаратных средств. С течением времени эти средства (проект) изменяются. Каждый проект при существенных изменениях получает свое, внутреннее имя. Причем имя может быть изменено при модификации ОС, протокола обмена или аппаратной части.

Таб.4.2.1-2 Список проектов

Проект	Аппаратная часть	BIOS/OS	Протокол	Примечания
MC186	Развитие 1-го проекта (Курганский вариант)	MICONT86	Подобный "P-NET"	Проект снят с поддержки
MC186X	5-и платный вариант	MIC86X-TS	Подобный "P-NET"	Проект снят с поддержки
MC186_1	Развитие 1-го проекта (Курганский вариант)	MIC86-1TS	"MicontBus"	Развитие MC186
MC186X_1	5-платный вариан	MIC86X1TS	"MicontBus"	Развитие MC186X
MC186B	BREEZE	BR-000-NC	"MicontBus"	Новая идеология аппаратной части
BARS	BREEZE	BR-000-SV	нет	Сварка полиэтиленовых труб
SPUTNIK	BREEZE с доп.каналами	BR-000-ZU	"MicontBus"	Переходный вариант к ОС MICOS
MICOS	BREEZE расширенный	MICOS	"MicontBus"	Новая идеология ОС

Таб.4.2.1-3 Список проектов поддерживающих запросы к группам данных контроллера
Состояние на 7 декабря 2000г

Имя группы	Проект		
	MC186B	SPUTNIK	MICOS
NETVAR_MEM	+	+	+
NETVAR_FLASH	+	+	+
NETVAR_SZB_BUF	+	+	+
NETVAR_HEAD	+	+	+
NETVAR_DATA	+	+	+
NETVAR_VERSION	+	+	+
NETVAR_SCR	+	+	+
NETVAR_KBD	+	+	+
NETVAR_CALC0	+	+	+
NETVAR_CALC1	+	+	+
NETVAR_CALC2	+	+	+
NETVAR_CALC3	+	+	+
NETVAR_TVNAME	+	+	+
NETVAR_TCNAME	+	+	+
NETVAR_CONST	+	+	+
NETVAR_CALC_ON	+	+	+
NETVAR_CVIDEO	+	+	+
NETVAR_CMSG	+	+	+
NETVAR_CUSER	+	+	+
NETVAR_CTEXT	+	+	+
NETVAR_RESET	+	+	+

Состояние на 13 февраля 2002г
(NETVAR_VERLINK=0-0-0-1, от 13.02.2002)

Имя группы	Проект				
	MC186_1	MC186X_1	MC186B	SPUTNIK	MICOS
NETVAR_MEM	+	+	+	+	+
NETVAR_FLASH	+	+	+	+	+
NETVAR_SZB_BUF	+	+	+	+	+
NETVAR_F_REC	+	+	+	+	+
NETVAR_HEAD	+	+	+	+	+
NETVAR_DATA	+	+	+	+	+
NETVAR_VERSION	+	+	+	+	+
NETVAR_VERLINK	+	+	+		
NETVAR_SCR	+	+	+	+	+
NETVAR_KBD	+	+	+	+	+
NETVAR_DTTM	+	+	+	+	
NETVAR_POWER	+	+	+	+	+
NETVAR_CALC0	+	+	+	+	+
NETVAR_CALC1	+	+	+	+	+
NETVAR_CALC2	+	+	+	+	+
NETVAR_CALC3	+	+	+	+	+
NETVAR_TVNAME	+	+	+	+	+
NETVAR_TCNAME	+	+	+	+	+
NETVAR_CONST	+	+	+	+	+
NETVAR_CALC_ON	+	+	+	+	+
NETVAR_CVIDEO	---	---	+	+	+
NETVAR_CMSG	+	+	+	+	+
NETVAR_CUSER	---	---	+	+	+
NETVAR_CTEXT	+	+	+	+	+
NETVAR_RESET	+	+	+	+	+

ВНИМАНИЕ: введена переменная *NETVAR_VERLINK* - версия обработчика запросов по связи. Дальнейшие изменения будут сопровождаться указанием значения версии. Структура значения описана в главе "ПРИМЕРЫ ДОСТУПА К ДАННЫМ".

4.2.2. Формат запроса.

Запрос к контроллеру кодируется как показано выше (см. п.п.4.1.1). Запрос состоит из полей:

«IDnet» - Сетевой адрес контроллера (1 байт).

«CMD» - Код команды/функции (1 байт).

«DATA» - Данные запроса.

Эти поля подлежат кодированию и участвуют в подсчете контрольной суммы. При подсчете контрольной суммы используются не закодированные данные. Сама контрольная сумма кодируется.

Содержимое поля «DATA» зависит от типа команды и имеет следующий формат:

Таб.4.2.2-1 Формат запроса (поле «DATA»)

Идентификатор группы или номер переменной (VAR)	Размер (SIZE)	* Смещение от начала (OFFS)	* Данные для записи (WRDATA)
2 байта (WORD)	2 байта. (WORD)	4 байта. (LONG)	Число байтов равное SIZE

Поля отмеченные «*» могут отсутствовать.

«VAR» - Идентификатор группы или номер переменной интерпретатора.

«SIZE» - Число запрашиваемых байтов (для функций/команд чтения) или число байтов для записи (для функций записи).

«OFFS» - Смещение до данных от стартового адреса группы. Под стартовым адресом понимается реальный адрес в памяти контроллера. Каждой группе данных соответствует свой стартовый адрес. Пусть адрес группы «XXXX», тогда действительный адрес данных равен сумме стартового адреса группы и смещения «XXXX»+ «OFFS».

Таб.4.2.2-2 Список команд/функций

Имя команды «CMD»	Код (HEX)	Пояснения
NETCMD_GETSIZE	01	Получить размер группы данных или размер переменной интерпретатора. Возвращает размер типа LONG.
NETCMD_GETBUF_B	02	Прочитать заданное число байтов группы данных.
NETCMD_GETBUF	03	Прочитать заданное число байтов группы данных, начиная со стартового адреса+смещение.
NETCMD_PUTBUF_B	04	Записать заданное число байтов в область группы данных. Используется для модификации значений переменных интерпретатора. Перед модификацией проверяется флаг разрешения записи (см. «Структуры и форматы данных контроллера»)
NETCMD_PUTBUF	05	Записать заданное число байтов в область данных группы, начиная со стартового адреса+смещение.

Пример 4.2.2-1. Формирование буфера запроса и вывода

Пусть необходимо прочитать с контроллера с сетевым номером 35 (23h) значение переменной интерпретатора с номером 10 (0Ah). Используем команду «*NETCMD_GETBUF_B*», код которой равен 2 (02h).

Буфер запроса «ЗАПРОС»

(вход функции mbNetFrmBufOut)

<i>IDnet</i>	<i>CMD</i>	<i>VAR</i>		<i>SIZE</i>	
		<i>Low</i>	<i>High</i>	<i>Low</i>	<i>High</i>
23h	02h	0Ah	00h	04h	00h

Буфер для вывода «КАДР»

(после обработки функцией mbNetFrmBufOut):

Буфер вывода		Пояснения	
Символ	Код		
:	3Ah	Начало кадра	
R	52h	Idne (Byte 23h)t	
C	43h		
P	50h	CMD (Byte 02h)	
B	42h		
P	50h	Low	VAR (Word 000Ah)
J	4Ah		
P	50h	High	
@	40h		
P	50h	Low	SIZE (Word 0004h)
D	44h		
P	50h	High	
@	40h		
\	5Ch	CRC	
M	4Dh		
	0Dh	CR (EOF)	
	0Ah	LF (Готовность источника)	

4.2.3. Обмен данными.

На каждый запрос контроллер отвечает, независимо от типа запроса. В поле команды «*CMD*», в старшей тетраде байта находится код результата, а в младшей тетраде код команды запроса. Контроллер отвечает только в случае совпадения сетевого адреса и поля запроса «*IDnet*». Код результата необходим для анализа сбоев и что наиболее важно для поиска ответа в буфере ввода. Поиск ответа необходим в случаях, когда на компьютере установлен преобразователь RS232⇒RS485 работающий в режиме «ЭХО». В этом случае в буфер ввода попадают как символы ответа, так и запроса. Ниже приведены коды результата при обнуленной младшей тетраде.

Таб.4.2.3-1 Коды результата

Имя команды « <i>CMD</i> »	Код (HEX)	Пояснения
NETREZ_OK	10	Результат в норме.
NETREZ_WAIT	20	Исполнение отложено (не используется)
NETREZ_BUSY	30	Занят. Формируется если за время 125 мс контроллер не может выполнить требования запроса..
NETREZ_UCMD	40	Эта команда не поддерживается. Формируется если этой команды нет в списке.
NETREZ_ERRVAR	50	Нет такой группы/переменной. Формируется если в конкретной реализации не описана группа запрашиваемых данных.
NETREZ_ERRCMD	60	Не допустимая команда. Формируется если команда и группа данных не совместимы. На пример команды чтения и идентификатор группы «NETVAR_RESET» не совместимы.
NETREZ_ERRARG	70	Не допустимый аргумент команды. Формируется если имеет не логичное значение. На пример если в командах чтения/записи поле « <i>SIZE</i> » содержит нулевое значение.
NETREZ_ERRBSIZE	80	Ошибка размера буфера. Формируется если значение поля « <i>SIZE</i> » больше max. размера буфера обмена на контроллере (1024 байта).
NETREZ_ERRBADDR	90	Ошибка адресации. Формируется если вычисленный й адрес группы данных находится за пределами адресного пространства контроллера. («Адрес группы»+« <i>OFFS</i> »+« <i>SIZE</i> »)
NETREZ_ERRPWD	A0	Доступ отвергнут. Формируется если нет прав доступа к значениям заданной группы данных. На пример при записи значений переменных интерпретатора, если переменная не описана как модифицируемая извне.

4.2.3.1. Формат ответа.

Формат ответа не отличается от формата запроса для большинства команд.

Таб.4.2.3.1-1 Формат ответа (поле «DATA»)

Идентификатор группы или номер переменной (VAR)	Размерс (SIZE)	* Смещение от начала (OFFS)	* Считанные данные (RDDATA)
2 байта (WORD)	2 байта. (WORD)	4 байта. (LONG)	Число байтов равное SIZE

Поля отмеченные «*» могут отсутствовать.

В случае ошибки (код результата не равен «NETREZ_OK») ответ обычно урезается до поля «VAR» или «SIZE» с правой стороны.

4.2.4. Функции/команды.

В описанных ниже сеансах связи «ЗАПРОС»-«ОТВЕТ» приводятся соответствующие форматы без кодировки (оформление кадра). Для кодирования/декодирования кадра используйте функции «mbNetFrmBufOut» и «mbNetExtractInpData». Для определенности установим сетевой адрес/номер терминала равный 02h в запросе и с нормальным кодом результата в старшей тетраде «CMD» равный «NETREZ_OK» (10h) в ответе. Посмотрим что получится если использовать команду «NETCMD_GETBUF_B» (код 02h):

- Поле «CMD» при запросе равно 02h.
- Поле «CMD» при ответе равно 12h.

Для выделения результата можно использовать сдвиг байта вправо на 4-е бита (тетрада – 4-е бита) или целочисленное деление на 16:

- Rez=«CMD»>>4; (Rez=01h).
- Rez=«CMD»/16; (Rez=01h).

Если не устраивает значение старшей тетрады (необходимо иметь байт кода результата совпадающий с «NETREZ_OK»), то можно выполнить обратные операции:

- Rez=«CMD»<<4; (Rez=10h).
- Rez=«CMD»*16; (Rez=10h).

Хотя тот же результата можно получить обнулением младшей тетрады (логическое умножение AND на значение F0h) - Rez=«CMD» & 0xF0; (Rez=10h).

ВНИМАНИЕ: Приведенные ниже коды команд и результата приведены в качестве примера. Рекомендуются использовать имена этих констант.

4.2.4.1. Функция: Получить размер группы данных или переменной.

Используется для определения размера группы данных или переменной интерпретатора.

ЗАПРОС

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word
02h	NETCMD_GETSIZE 01h	XX

ОТВЕТ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>RDDATA</i> Размер группы (4 байта) Dword/LONG
02h	NETREZ_OK + NETCMD_GETSIZE 11h	XX	RRRR

4.2.4.2. Функция: Прочитать заданное число байтов.**ЗАПРОС**

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word
02h	NETCMD_GETBUF_B 02h	XX	YY

ОТВЕТ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число считанных байт (2 байта) Word	<i>RDDATA</i> Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	XX	YY	R...R

4.2.4.3. Функция: Прочитать заданное число байтов со смещением.

Функция используется для чтения больших непрерывных массивов. Нарастивая значения поля «OFFS» на размер предыдущей порции считанных данных (чтение журнала). За счет смещения можно обратиться к любой известной области памяти.

ЗАПРОС

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта [Word])	<i>SIZE</i> (2 байта) Word	<i>OFFS</i> (4 байта) Dword/LONG
02h	NETCMD_GETBUF 03h	XX	YY	ZZZZ

ОТВЕТ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта [Word])	<i>SIZE</i> (2 байта) Word	<i>OFFS</i> (4 байта)	<i>RDDATA</i>
					Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF 13h	XX	YY	ZZZZ	R...R

4.2.4.4. Функция: Записать заданное число байтов.

Функция используется для модификации значений группы данных или переменных интерпретатора. Особенностью функции является возможность записи в буфер клавиатуры контроллера и контроль флага записи в описании переменных/констант интерпретатора.

ЗАПРОС

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записываемых байт (2 байта) Word	<i>WRDATA</i>
				Байты для записи
02h	NETCMD_PUT_B 04h	XX	YY	W...W

ОТВЕТ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записанных байт (2 байта) Word
02h	NETREZ_OK + NETCMD_PUTBUF_B 14h	XX	YY

4.2.4.5. Функция: Записать заданное число байтов со смещением.

Функция используется для записи больших непрерывных массивов. Нарастивая значения поля «*OFFS*» на размер предыдущей порции записанных данных.

ЗАПРОС

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта [Word])	<i>SIZE</i> (2 байта) Word	<i>OFFS</i> (4 байта)	<i>WRDATA</i> Байты для записи
02h	NETCMD_PUTBUF 05h	XX	YY	ZZZZ	W...W

ОТВЕТ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта [Word])	<i>SIZE</i> (2 байта) Word	<i>OFFS</i> (4 байта) Dword/LONG
02h	NETREZ_OK + NETCMD_PUTBUF 15h	XX	YY	ZZZZ

5. ЧТЕНИЕ ЖУРНАЛА

Структура журнала приведена в руководстве по эксплуатации «*Структуры и форматы данных контроллера*» и см. п.п.6.5.1

Чтение журнала производится следующим образом:

- Получить размер заголовка журнала.
(запрос «NETCMD_GETSIZE», группа «NETVAR_HEAD»)
- Прочитать заголовок выполнив вычисленное число чтений.
(запрос «NETCMD_GETBUF» , группа «NETVAR_HEAD»)
Заголовок можно записать в файл.
- Получить размер журнала
(запрос «NETCMD_GETSIZE», группа «NETVAR_DATA»)
- Прочитать данные журнала, выполнив вычисленное число чтений.
(запрос «NETCMD_GETBUF» , группа «NETVAR_DATA»)
Данные журнала добавить к заголовку, получив тем самым файл журнала, описанный в том-же руководстве по эксплуатации.

6. ПРИМЕРЫ ДОСТУПА К ДАННЫМ КОНТРОЛЛЕРА

6.1. Значения переменных интерпретатора.

Идентификаторы значений переменных интерпретатора имеют номера от 0 до 254. Типы значений (*FLOAT/LONG*) можно узнать из таблицы описания переменных (доступ через *NETVAR_TVNAME*). Значение переменных всегда имеет размер 4-е байта. Для считывания значений переменных можно использовать как групповое чтение (за одно чтение несколько значений) так и одиночное (за одно чтение одно значений). Для модификации значений используется одиночная запись командой *NETCMD_GETBUF_B*. Модифицировать переменную можно при установленном флаге модификации (таблица описания переменных).

Пример 6.1-1. Групповое чтение 48 значений переменных

Задание: получить 48 значений переменных, начиная с десятой.

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word
02h	NETCMD_GETBUF_B 02h	10	48*4= 192

Ответ

<i>IDnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число считанных байт (2 байта) Word	<i>RDDATA</i> Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	10	192	48 значений по 4 байта

Пример 6.1-2. Модификация значения переменной

Задание: изменить значение переменной №23 на 94 (*LONG*).

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записываемых байт (2 байта) Word	<i>WRDATA</i> Байты для записи
02h	NETCMD_PUT_B 04h	23	4	94

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записанных байт (2 байта) Word
02h	NETREZ_OK + NETCMD_PUTBUF_B 14h	23	4

6.2. Версия обработчика запросов по связи.

Для доступа к версии обработчика по связи используется переменная *NETVAR_VERLINK*. Эта переменная поможет определить набор доступных данных контроллера (см. п.п.4.2).

Структура значения версии:

Смещение (HEX)	Тип	Размер	Имя поля/Назначение
00h	Byte	1	1-й байт версии
01h	Byte	1	2-й байт версии
02h	Byte	1	3-й байт версии
03h	Byte	1	4-й байт версии (младший)
04h	Word	2	Год
06h	Byte	1	Месяц
07h	Byte	1	Число

Пример 6.2-1. Чтение версии обработчика запросов

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word
02h	NETCMD_GETBUF_B 02h	NETVAR_VERLINK 0B03h	8

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число считанных байт (2 байта) Word	<i>RDDATA</i> Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	NETVAR_VERLINK 0B03h	8	8 байт версии

6.3. Системная дата и время

Для доступа к дате и времени используется переменная *NETVAR_DTTM*. Эта переменная доступна для чтения и для записи.

Структура значения переменной *NETVAR_DTTM*:

Смещение (HEX)	Тип	Размер	Имя поля/Назначение
00h	Byte	1	Секунды
01h	Byte	1	Минуты
02h	Byte	1	Часы
03h	Byte	1	Число
04h	Byte	1	Месяц
05h	Word	2	Год
07h	Byte	1	День недели (0..6, 0=Вск, 1=Пнд, ..., 6=Суб.

Пример 6.3-1. Чтение даты и времени

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word
02h	NETCMD_GETBUF_B 02h	NETVAR_DTTM 0C02h	8

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число считанных байт (2 байта) Word	<i>RDDATA</i> Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	NETVAR_DTTM 0C02h	8	дата и время

Пример 6.3-2. Установка даты и времени

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записываемых байт (2 байта) Word	<i>WRDATA</i> Байты для записи
02h	NETCMD_PUT_B 04h	NETVAR_DTTM 0C02h	8	Дата и время

Внимание: при установке, поле *день недели* игнорируется. Он вычисляется на контроллере.

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число записанных байт (2 байта) Word
02h	NETREZ_OK+NETCMD_PUTBUF_B 14h	NETVAR_DTTM 0C02h	8

6.4. Параметры питания контроллера

Для доступа к параметрам питания контроллерами используется переменная *NETVAR_POWER*. Эта переменная доступна только для чтения.

Структура значения переменной *NETVAR_POWER*:

Смещение (HEX)	Тип	Размер	Имя поля/Назначение
00h	FLOAT	4	Напряжение сети
04h	FLOAT	4	Частота сети
08h	FLOAT	4	Температура внутри корпуса

Пример 6.4-1. Чтение параметров питания

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word
02h	NETCMD_GETBUF_B 02h	NETVAR_POWER 0C04h	12

Внимание: запрашивать всегда 12 байт.

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число считанных байт (2 байта) Word	<i>RDDATA</i>
				Считанные байты
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	NETVAR_POWER 0C04h	12	параметры

6.5. Поиск записи в журнале.

Для поиска записей в журнале используется переменная *NETVAR_F_REC*. Эта переменная доступна только для чтения. Причем используется команда *NETCMD_GETBUF_B* с дополнительными параметрами (DATA_EXT).

Задача поиска - найти запись, дата и время которой отличаются от заданных в меньшей степени. Иначе говоря - найти ближайшую запись при условии, что *ДатаВремяЗаписи* больше или равно *ДатаВремяЗаданное*.

Пример:

Найти запись от 17.01.2002, 12:30:25,0

Журнал:

Запись 1: 17.01.2002, 10:31:54,0

Запись 2: 17.01.2002, 12:29:16,0

Запись 3: 17.01.2002, 12:32:43,0

Запись 4: 17.01.2002, 12:32:43,0

Запись 5: 17.01.2002, 12:35:21,0

Результат: Запись 3 от 17.01.2002, 12:32:43,0

Структура переменной *NETVAR_F_REC* (*DATA_EXT*-доп.параметры запроса):

Смещение (HEX)	Тип	Размер	Имя поля/Назначение
00h	Word	2	Результат (№ записи)
02h	Word	2	Год
04h	Byte	1	Месяц
05h	Byte	1	Число
06h	Byte	1	Часы
07h	Byte	1	Минуты
08h	Byte	1	Секунды
09h	Word	2	Миллисекунды

В ОС *MICOS* время записи хранится с точность до миллисекунд.

При запросе на поиск формируются следующие значения полей:

- Результат = 0;
- Остальные поля = Заданное дата и время

Если поиск завершен успешно, то в поле *Результат* возвращается *№записи+1* (*№* записи отличный от нуля) и в поля даты и времени заполнены значениями найденной записи (дата и время найденной записи). Записи в журнале нумеруются с нуля. Если запись не найдена, то в поле *Результат* возвращается 0.

Есть возможность поискать последнюю запись в журнале. Последняя запись не обязательно является младшей по дате и времени (могли измениться системная дата и время). Последняя запись – запись, сохраненная последней в журнале. Для поиска последней записи необходимо заполнить поле *Год*, в запросе, значением FFFFh.

Для поиска первой записи, самой старшей, достаточно поле *Год* заполнить значением 0.

В приведенных, ниже примерах, всегда предполагается успешный поиск.

Пример 6.5-1. Поиск произвольной записи

Запрос

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word	<i>DATA_EXT</i> Дополнительные данные
02h	NETCMD_GETBUF_B 02h	NETVAR_F_REC 0AFFh	11	11 байт, данные запроса.

Внимание: запрашивать всегда 11 байт.

Ответ

<i>Idnet</i> (1 байт)	<i>CMD</i> (1 байт)	<i>VAR</i> (2 байта) Word	<i>SIZE</i> Число байт для чтения (2 байта) Word	<i>DATA_EXT</i> Дополнительные данные
02h	NETREZ_OK + NETCMD_GETBUF_B 12h	NETVAR_F_REC 0AFFh	11	11 байт, результат.

Для чтения записи с заданным номером необходимо знать *№записи* (0..n) и размер записи. Для чтения произвольной записи используется команда NETCMD_GETBUF и переменная NETVAR_DATA.

Смещение до записи вычисляется так: *№записи * Размер записи*.

Размер записи и емкость журнала (число записей) можно получить из заголовка журнала (*NETVAR_DATA*).

6.5.1. Особенности алгоритма поиска записи в журнале.

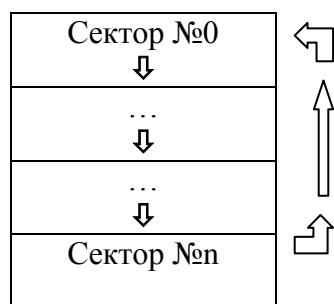
В настоящее время используется алгоритм “быстрого поиска”, основанный на особенностях ведения журнала и предположении отсутствия “глупых/аварийных ситуаций”. Для начала введем несколько понятий.

Журнал хранится во FLASH. FLASH разбит на сектора по 64Кб. Для записи байта в какую-либо ячейку FLASH требуется, чтобы эта ячейка имела значение FFh. Для заполнения ячеек FLASH значениями FFh используется операция стирания. Стирание возможно только по секторам т.е. по 64Кб.

Журнал имеет фиксированный размер на конкретное число записей. Всегда известен адрес текущей записи. Сохранение записей в журнале производится по кольцу - после сохранения последней записи происходит переход к первой. Сохранение производится с опережающим стиранием сектора:

- Если до конца сектора остается одна запись, то следующий сектор стирается;
- Если это последний сектор журнала, то стирается первый.

Порядок сохранения записей в журнале:



ПРИМЕРЫ.

Для простоты введем несколько допущений:

- будем считать дату и время сохранения записей одним целым числом;
- один сектор содержит 6 записей;
- 4 сектора в журнале.

На базе этих допущений имеем следующий вид одного сектора:

№ сект.	№ зап.	Дата и время записи
0	0	
	1	
	2	
	3	
	4	
	5	

Ниже приведены примеры заполнения журнала.

Пример №1

Первое заполнение журнала.

№ сект.	№ зап.	Дата и время записи
0	0	10
	1	20
	2	30
	3	40
	4	50
	5	60
1	6	70
	7	80
	8	90
	9	100
	10	110
	11	120
2	12	130
	13	140
	14	150
	15	160
	16	170
	17	180
3	18	190
	19	200
	20	210
	21	220
	22	230
	23	

Пример №2

Был переход на начало журнала.

Искать числа: 2;

305

№ сект.	№ зап.	Дата и время записи
0	0	250
	1	260
	2	270
	3	280
	4	290
	5	300
1	6	310
	7	
	8	
	9	
	10	
	11	
2	12	130
	13	140
	14	150
	15	160
	16	170
	17	180
3	18	190
	19	200
	20	210
	21	220
	22	230
	23	240

БЫСТРЫЙ ПОИСК:

При обычном, “линейном”, поиске результат безошибочен. Но такой поиск требует больших временных затрат, превышающих гарантированное время отклика контроллера (120ms). Большие затраты связаны с большим объемом журнала (2600-5200 записей или 325Kб-512Kб.).

Используя особенности FLASH, журнала (разрыв значений на границах сектора) можно применить следующий алгоритм:

- 1) Определение сектора с наиболее подходящими значениями (проверяется первая и последняя запись каждого сектора)
- 2) В наиболее подходящем секторе производится линейный поиск.

Подходящим сектором считается такой, в котором самой подходящей записью является первая или последняя.

Этот алгоритм основан на предположении, что записи в отдельном секторе упорядочены. Но он не является точным т.к. в середине сектора могут оказаться два перехода корректирующие ошибки. Под ошибкой будем понимать - сбой часов или ошибку оператора, которая в последствии исправляется.

Допустим, что ошибка и исправление были зафиксированы в журнале. Тогда имеем следующий вид журнала:

№ сект.	№ зап.	Дата и время записи
0	0	45
	1	55
	2	10 000-ОШИБКА ВВОДА
	3	80-ИСПРАВЛЕНИЕ
	4	85
	5	95
1	6	105
	7	96 - ПЕРЕХОД
	8	115
	9	
	10	
	11	
2	12	130
	13	2-ОШИБКА ВВОДА
	14	149-ИСПРАВЛЕНИЕ
	15	160
	16	170
	17	180
3	18	190
	19	199-ПЕРЕХОД
	20	210
	21	15-СБОЙ
	22	25
	23	35

На этом примере, с предлагаемым алгоритмом, невозможно правильно выполнить поиск с заданием **2** и **10 000**. В случае задания **2** будет найдено число **15** (запись №21). В случае задания **10 000** будет найдено число **210** (запись №19).

Описанные ситуации являются исключительными. Для их обнаружения необходимо прочитать весь журнал .