ПОДКЛЮЧЕНИЕ УСТРОЙСТВА МАРКИ «ТЕРМОДАТ» К ПОСЛЕДОВАТЕЛЬНОМУ ПОРТУ ЭВМ

Для удобства работы с настоящим прибором предусмотрено его сопряжение с персональным компьютером через COM-порт (последовательный порт). Подключение производится по двухпроводной линии RS-485. Для этого необходим специальный адаптер (конвертер) RS-232 / RS-485. Конвертер подключается к ЭВМ, а прибор подключается через клеммы RS-485 (A и B) к соответствующим клеммам конвертера (см. руководство по эксплуатации конвертера). Есть возможность подключать к одному конвертеру несколько приборов Термодат одновременно. В этом случае приборы подключаются параллельно по линии RS-485, как показано на рисунке 1 (объединение приборов в сеть).

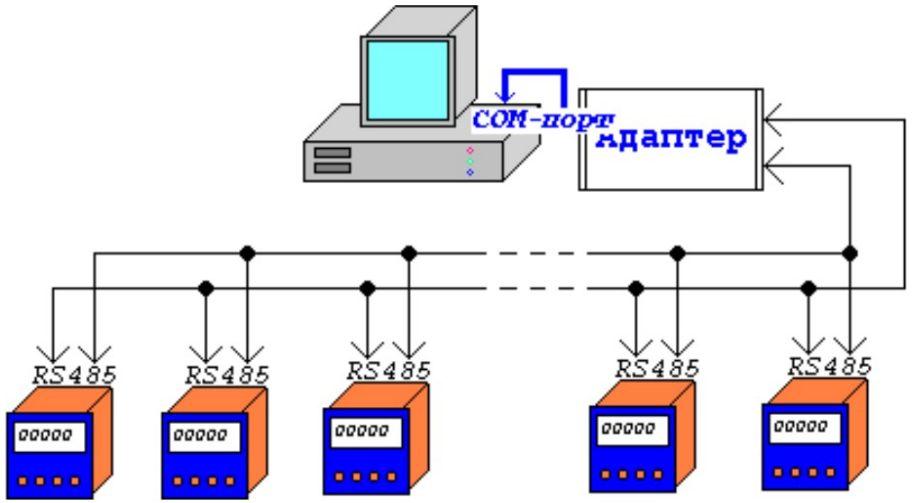


Рис.1 Подключение сети приборов «Термодат» к ЭВМ

После правильного подключения прибора можно приступить к организации его взаимодействия с ЭВМ, как ведомого по отношению к ЭВМ устройства. Это взаимодействие происходит так:

1) В последовательный порт посылается некоторая последовательность байтов (*команда-запрос*), в которой закодирована информация о запрашиваемом действии прибора (запрос значения какого-либо параметра прибора, либо на установку какого-либо параметра, либо на выдачу архивных записей прибора).
2) Далее следует дождаться ответа от используемого в данный момент прибора (время между окончанием запроса и началом ответа составляет, как правило, менее одной секунды). Ответ прибывает в тот же последовательный порт тоже в виде некоторой последовательности байтов и имеет сходный с запросом формат.

Формат запросов и ответов зависит от выбранного протокола обмена. В данной модификации прибора Термодат реализованы два различных протокола обмена: «MODBUS-ASCII» и «Термодат». В некоторых устройствах помимо упомянутых реализован еще один протокол обмена «MODBUS-RTU».

Некоторые приборы Термодат снабжены клеммами для подключения к последовательному порту ЭВМ непосредственно (подключение RS-232). Такие приборы невозможно объединить в сеть, но и конвертер не нужен.

ОПИСАНИЕ ПРОТОКОЛА «MODBUS-ASCII» ДЛЯ УСТРОЙСТВ МАРКИ ТЕРМОДАТ

Протоколы обмена «MODBUS» широко распространены. Подробную информацию об этих протоколах можно прочесть во многих изданиях (например, в сети Internet). Версия протокола «MODBUS» для настоящего устройства обладает такими свойствами. Как запрос, так и ответ представляют собой последовательности байтов, каждый из которых закодированный символ, согласно таблице символов ASCII (стандартные однобайтовые коды символов для большинства ЭВМ). Поэтому далее следует описание команд в текстовой (не двоичной) форме. Все команды-запросы и ответы имеют такой формат:

- 1 символ – заголовок команды, двоеточие (код **3A**h).
- 2 и 3 символы – сетевой идентификатор прибора (адрес), уникален для каждого прибора в сети (шестнадцатеричное число).
- 4 и 5 символы – код функции, т.е. идентификатор запрашиваемого действия (тоже шестнадцатеричное число).
- далее следуют данные, содержащие необходимую информацию – это числа в шестнадцатеричной системе счисления (цифры `**0**` .. `**9**`, латинские буквы `**A**` .. `**F**`, либо `**a**` .. `**f**`).
- После данных следуют два символа контрольной суммы LRC (тоже число в шестнадцатеричной системе счисления), в которой участвуют байты, начиная с сетевого адреса, заканчивая последним байтом данных. Алгоритм подсчета LRC представлен на языке программирования «C» ниже:

```
unsigned char digchar(unsigned char v)
{
    v-='0';
    if(v>41) return v-39; /* a .. f */
    if(v>9) return v-7;   /* A .. F */
    return v;             /* 0 .. 9 */
}

unsigned char LRC(unsigned char *str)
{
    unsigned char val=0;
    while(*str)
     {
         val+=(digchar(*str)<<4)|digchar(str[1]);
         str+=2;
```

```
        }
        return (unsigned char)(-((signed char)val));
    }
```

- Последние 2 символа имеют коды **0D**h и **0A**h.

В следующей таблице приведены формы запросов и ответов в зависимости от функции.

Функция	**Код функции**	**Форма запроса: Обозначение и количество передаваемых байтов**	**Форма ответа: Обозначение и количество получаемых байтов**
Читать несколько параметров	03h либо 04h	**:** 1 Adr 2 Fc 2 PAdr 4 PNum 4 LRC 2 CRLF 2	**:** 1 Adr 2 Fc 2 BNum 2 PVal1 4 PValN 4 LRC 2 CRLF 2
Записать один параметр	06h	**:** 1 Adr 2 Fc 2 PAdr 4 PVal1 4 LRC 2 CRLF 2	**:** 1 Adr 2 Fc 2 PAdr 4 PVal1 4 LRC 2 CRLF 2
Записать несколько параметров	10h	**:** 1 Adr 2 Fc 2 PAdr 4 PNum 4 BNum 2 PVal1 4 PValN 4 LRC 2 CRLF 2	**:** 1 Adr 2 Fc 2 PAdr 4 PNum 4 LRC 2 CRLF 2

Условные обозначения:

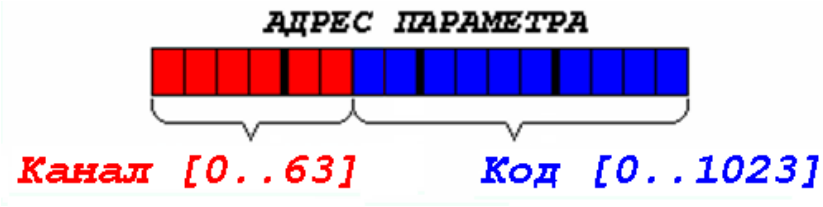
1) **Adr** – сетевой адрес устройства, 2 знака
2) **Fc** – код функции, 2 знака
3) **PAdr** – адрес параметра, 4 знака
4) **PNum** – количество запрашиваемых (передаваемых) параметров от PAdr включительно, 4 знака
5) **PVal1 .. PValN** – значения параметров с адресами PAdr .. PAdr+(PValN-1), по 4 знака на каждое значение
6) **BNum** – количество байтов данных BNum = 2·PNum, 2 знака
7) **LRC** – контрольная сумма, 2 знака
8) **CRLF** – концевые символы с кодами 0Dh и 0Ah, 2 знака

Основные особенности:

1) Если **Adr = 00h** («мастер-адрес»), то все приборы воспринимают данные, но ответа не следует
2) Все параметры передаются и принимаются как двухбайтовые шестнадцатеричные числа в текстовом формате ASCII (на каждый байт по 2 символа), в языке программирования «С» этот тип называется **unsigned int**
3) Если не существует параметра по запрашиваемому адресу (либо в случае ошибки), то вместо значения параметра посылается число **7FFFh** (либо **7FFEh**)
4) Если проверка **LRC** в запросе не увенчалась успехом, то запрос не воспринимается и ответа не следует
5) В некоторых приборах, например в Термодат-25, Термодат-29, введены специальные обозначения: если измеренное значение равно **7D00h**, то на данном канале обрыв, либо датчик отсутствует; если измеренное значение равно **7D64h**, то по каким-либо причинам нет данных от измерительного модуля.
6) Это касается устройств, содержащих *оконный* доступ к архиву. Имеется поддержка сообщений об ошибке согласно стандарту «MODBUS». Если получен неверный запрос, то ответ на него имеет следующий вид:
   :<Adr><80h+Fc><код ошибки modbus><LRC><CRLF> ,
   где **<код ошибки modbus>** принимает следующие значения:
   2 - ILLEGAL_DATA_ADDRESS - указанный адрес (PAdr) параметра не поддерживается для указанной операции
   3 - ILLEGAL_DATA_VALUE - принятое значение (PVal) не может быть записано по указанному адресу (PAdr) параметра
   4 - SLAVE_DEVICE_FAILURE - выполнить указанную операцию невозможно (например, происходит переполнение буфера посылки внутри устройства)

Адреса параметров:

Двухбайтовое число – адрес параметра – содержит два числа: *код* параметра и *канал*, для которого этот параметр определен, если прибор является многоканальным. Структура адреса параметра представлена в виде битовой диаграммы:



Там же указаны возможные диапазоны для кода и канала. Нумерация битов справа налево. Можно эту диаграмму представить формулой:

АДРЕС = КОД + (КАНАЛ * 1024),

где КАНАЛ отсчитывается от нуля.

ОПИСАНИЕ ПРОТОКОЛА «MODBUS-RTU» ДЛЯ УСТРОЙСТВ МАРКИ ТЕРМОДАТ

Протокол обмена «MODBUS-RTU» отличается от «MODBUS-ASCII» тем, что байты посылаются и принимаются не в текстовом (шестнадцатеричном), а в двоичном формате. Это основное отличие. Например, если адрес прибора 25, то посылаются не символы `2` (32h) и `5` (35h), как в «MODBUS-ASCII», а число 25 (19h). Так количества передаваемых и принимаемых байтов сокращаются в двое по сравнению с «MODBUS-ASCII». Заголовок (двоеточие) в протоколе «MODBUS-RTU» уже не используется, т. к. посылка может начинаться с любого символа (работаем не с символами, а с байтами). Вместо этого используются временные задержки до и после посылаемого/принимаемого кадра, равные посылке четырех байтов со всеми стартовыми, стоповыми битами, битами четности на текущей скорости порта. Задержки могут длиться и более чем посылка четырех байтов, лишь бы не менее. Заметим, что внутри кадра (между байтами в нем) не может быть задержек более либо равных посылке четырех байтов. Рекомендуем для работы в этом протоколе сначала накапливать данные в буфере, а потом посылать целый кадр, байт за байтом, в порт без задержек на получение, запись данных и другие сопутствующие операции. Формы команд-запросов и ответов на них совпадают с формами команд в протоколе «MODBUS-ASCII». Смотрите таблицу выше. Поддерживаются также три различные функции (03h (она же 04h), 06h и 10h). Отличия заключаются в контрольной сумме: вместо одного байта LRC посылаются два байта CRC, отсутствии заголовка-двоеточия и концевых символов CRLF. Пример функции на «C» для ее подсчета смотрите ниже:

```
const unsigned char arrcrch[256]=
{ /* 1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /*  1 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /*  2 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /*  3 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /*  4 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /*  5 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /*  6 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /*  7 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /*  8 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /*  9 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /* 10 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /* 11 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /* 12 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, /* 13 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /* 14 */
  0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40, 0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, /* 15 */
  0x00, 0xc1, 0x81, 0x40, 0x01, 0xc0, 0x80, 0x41, 0x01, 0xc0, 0x80, 0x41, 0x00, 0xc1, 0x81, 0x40  /* 16 */
};

const unsigned char arrcrcl[256]=
{ /* 1     2     3     4     5     6     7     8     9    10    11    12    13    14    15    16 */
  0x00, 0xc0, 0xc1, 0x01, 0xc3, 0x03, 0x02, 0xc2, 0xc6, 0x06, 0x07, 0xc7, 0x05, 0xc5, 0xc4, 0x04, /*  1 */
  0xcc, 0x0c, 0x0d, 0xcd, 0x0f, 0xcf, 0xce, 0x0e, 0x0a, 0xca, 0xcb, 0x0b, 0xc9, 0x09, 0x08, 0xc8, /*  2 */
  0xd8, 0x18, 0x19, 0xd9, 0x1b, 0xdb, 0xda, 0x1a, 0x1e, 0xde, 0xdf, 0x1f, 0xdd, 0x1d, 0x1c, 0xdc, /*  3 */
  0x14, 0xd4, 0xd5, 0x15, 0xd7, 0x17, 0x16, 0xd6, 0xd2, 0x12, 0x13, 0xd3, 0x11, 0xd1, 0xd0, 0x10, /*  4 */
  0xf0, 0x30, 0x31, 0xf1, 0x33, 0xf3, 0xf2, 0x32, 0x36, 0xf6, 0xf7, 0x37, 0xf5, 0x35, 0x34, 0xf4, /*  5 */
  0x3c, 0xfc, 0xfd, 0x3d, 0xff, 0x3f, 0x3e, 0xfe, 0xfa, 0x3a, 0x3b, 0xfb, 0x39, 0xf9, 0xf8, 0x38, /*  6 */
  0x28, 0xe8, 0xe9, 0x29, 0xeb, 0x2b, 0x2a, 0xea, 0xee, 0x2e, 0x2f, 0xef, 0x2d, 0xed, 0xec, 0x2c, /*  7 */
  0xe4, 0x24, 0x25, 0xe5, 0x27, 0xe7, 0xe6, 0x26, 0x22, 0xe2, 0xe3, 0x23, 0xe1, 0x21, 0x20, 0xe0, /*  8 */
  0xa0, 0x60, 0x61, 0xa1, 0x63, 0xa3, 0xa2, 0x62, 0x66, 0xa6, 0xa7, 0x67, 0xa5, 0x65, 0x64, 0xa4, /*  9 */
  0x6c, 0xac, 0xad, 0x6d, 0xaf, 0x6f, 0x6e, 0xae, 0xaa, 0x6a, 0x6b, 0xab, 0x69, 0xa9, 0xa8, 0x68, /* 10 */
  0x78, 0xb8, 0xb9, 0x79, 0xbb, 0x7b, 0x7a, 0xba, 0xbe, 0x7e, 0x7f, 0xbf, 0x7d, 0xbd, 0xbc, 0x7c, /* 11 */
  0xb4, 0x74, 0x75, 0xb5, 0x77, 0xb7, 0xb6, 0x76, 0x72, 0xb2, 0xb3, 0x73, 0xb1, 0x71, 0x70, 0xb0, /* 12 */
  0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, /* 13 */
  0x9c, 0x5c, 0x5d, 0x9d, 0x5f, 0x9f, 0x9e, 0x5e, 0x5a, 0x9a, 0x9b, 0x5b, 0x99, 0x59, 0x58, 0x98, /* 14 */
  0x88, 0x48, 0x49, 0x89, 0x4b, 0x8b, 0x8a, 0x4a, 0x4e, 0x8e, 0x8f, 0x4f, 0x8d, 0x4d, 0x4c, 0x8c, /* 15 */
  0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83, 0x41, 0x81, 0x80, 0x40  /* 16 */
};


unsigned int CRC(unsigned char *buf, unsigned int num)
{
    unsigned char CRC_lo, CRC_hi, index;
    CRC_lo=255;
    CRC_hi=255;
    while(num--)
     {
           index=CRC_lo^(*buf);
           CRC_lo=CRC_hi^(arrcrch[index]);
           CRC_hi=arrcrcl[index];
           buf++;
     }
    /* сначала `lo`, потом `hi` */
    return ((unsigned int)CRC_lo<<8)|CRC_hi;
}
```

Пункты «Адреса параметров» и «Основные особенности», описанные в предыдущем разделе, актуальны и для «MODBUS-RTU» с учетом, конечно, особенностей этого протокола (вместо пар символов – байты, вместо LRC используется CRC, нет заголовка **`:`** команды, а также концевых символов **CRLF**).

Для работы в протоколе «MODBUS-RTU» следует задавать такие настройки порта, чтобы общее число передаваемых битов было **11**, то есть:

1. 1 стартовый бит;
2. 8 битов данных (меньше восьми вообще не поддерживается)
3. либо 1 бит контроля четности (нечетный, четный) и 1 стоповый бит, либо 2 стоповых бита.

Именно для одиннадцати битов рассчитывается время передачи четырех байтов в приборах Термодат.

Примеры:

1. «MODBUS-ASCII»

1.1. Получение текущих измеренных значений

Запрос: :0103000000004F8<CR><LF>

Ответ: :010308FC1CF9C6F770F51AA7<CR><LF>

Интерпретация полученных данных: на первом канале FC1Ch=-99,6°C; на втором канале F9C6h=-159,4°C; на третьем F770h=-219,2°C; на четвертом F51Ah=-279,0°C.

1.2. Установить значение типа аварийной сигнализации (параметр с кодом 019Ah) в «Максимум» (значение 2) на канале номер 4
(019Ah + (4-1) ^{x} 1024 = 0D9Ah)

Запрос: :01060D9A000250<CR><LF>

Ответ: :01060D9A000250<CR><LF>

Подсчет контрольной суммы LRC: 01h+06h+0Dh+9Ah+00h+02h=B0h; -B0h=50h; проверим: 01h+06h+0Dh+9Ah+00h+02h+50h=100h; lo_byte{100h}=00h

2. «MODBUS-RTU»

2.1. Получение текущих измеренных значений

Запрос: [01h][03h][00h][00h][00h][04h][44h][09h][время посылки 4х байтов]

Ответ: [01h][03h][08h][FDh][93h][FCh][1Fh][FAh][AAh][F9h][35h][EBh][42h][время посылки 4х байтов]

Интерпретация полученных данных: на первом канале FD93h=-62,1°C; на втором канале FC1Fh=-99,3°C; на третьем FAAAh=-136,6°C; на четвертом F935h=-173,9°C.

2.2. Запуск регулирования на канале номер 3 (0180h + (3-1) ^{x} 1024 = 0980h)

Запрос: [01h][06h][09h][80h][00h][01h][4Ah][7Eh][время посылки 4х байтов]

Ответ: [01h][06h][09h][80h][00h][01h][4Ah][7Eh][время посылки 4х байтов]

Интерпретация полученных данных: регулирование успешно запущено (возвращено значение 0001h).

2.3. Установка значения текущей даты и времени 11 декабря 2007 года, 15 часов, 30 минут, 20 секунд

Запрос:
[01h][10h][01h][40h][00h][06h][0Ch][00h][07h][00h][0Ch][00h][0Bh][00h][0Fh][00h][1Eh][00h][14h][ABh][0Bh][время посылки 4х байтов]

Ответ: [01h][10h][01h][40h][00h][06h][40h][23h][время посылки 4х байтов]

Приложение. Подсчет контрольной суммы LRC на языке программирования «Паскаль»:

```
Function d2c(v:byte):byte;
   begin
      v:=v-byte('0');         {0 .. 9}
      if v>41 then v:=v-39    {a .. f}
       else
         if v>9 then v:=v-7; {A .. F}
      Result:=v; {возвращаемое значение}
   end;

Function LRC(s: string):byte; {string – строковый тип}
   var uLRC:byte; {беззнаковое целое, 8 бит}
       i:integer; {целое, 16 бит}
   begin
      uLRC:=0;
      i:=0;
      while i<length(s) do
       {функция length(s: string) – вычисление длины строки в символах}
         begin
            uLRC:=byte( uLRC + (d2c(byte(s[i]))*16 or d2c(byte(s[i+1]))) );
             {оператор `or` - побитовое `или`}
            i:=i+2
         end;
      uLRC:=-uLRC; {равносильно (255-uLRC)+1}
      Result:=uLRC; {возвращаемое значение}
   end;
```