ДУОЖ – протокол обмена в сети RS485

1. Описание протокола

Обмен ведется пакетами, в режиме: 19600 бод, 8 бит, 1 стоп бит, без контроля четности. Пакет - последовательность символов переменной длины. MASTER (ведущее устройство: регистратор, устройство настройки и тарирования) передает пакет-запрос, адресованный одному SLAVE (ведомому устройству: датчик топлива). После передачи пакет-запроса MASTER переходит в состояние ожидания ответа. При нормальном приеме (отсутствие ошибок) пакет-ответ формируется немедленно.

2. Зарезервированные символы

Для реализации байт-ориентированного транспортного протокола выделены следующие символы:

SOH (код 0xFF), **ETX** (код 0x03), **DLE** (код 0x10).

Символы SOH и ETX являются ограничителями пакетов (начало и конец соответственно).

Символ DLE служит для того, чтобы было возможным передавать сами зарезервированные символы (SOH, ETX, DLE**)** внутри поля <Данные> и как значение контрольного байта CRC. При необходимости передать зарезервированный символ с кодом <X> он передается как последовательность DLE <~X>, где <~X> - символ с кодом 0xFF – X.

Таблица 1. Трансляция зарезервированных символов

Символ	При передаче транслируется в последовательность
0x03	0x10 0xFC
0x10	0x10 0xEF
0xFF	0x10 0x00

Преобразование выполняется прозрачным образом, т.е. при вычислении CRC будет учтен символ <X>, а не как пара DLE <~X>.

Например, вместо символа с кодом 0x10 (который совпадает с зарезервированным символом DLE) будет передана последовательность 0x10 0xEF, так как 0xEF = 0xFF – 0x10.

Соответственно, при приеме эта последовательность преобразуется в символ с кодом 0x10 = 0xFF – 0xEF

3. Формат пакета

Все пакеты имеют вид:

SOH <Адрес-КОМУ> <Адрес-ОТ КОГО> <Команда> <Данные> <CRC> ETX

где:

SOH - Start Of Header (символ с кодом 0xFF)
<Адрес-КОМУ> - адрес получателя
<Адрес-ОТ КОГО> - адрес отправителя
<Команда> - команда, которую требуется выполнить (1 байт)
<Данные> - 0..N байт, параметры <Команды>
<CRC> - байт обнаружения ошибок пакета (1 байт)

ETX - End of TeXt (символ с кодом 0x03)

Примечание:
Если значение байта CRC совпадет с зарезервированным символом, для него выполняется преобразование по Таблице 1.

4. Адреса устройств

Адреса (КОМУ) могут быть только индивидуальными.
Адрес – это байт с кодом (0x70 + Addr)
Допустимы следующие адреса:

0x70 – ДУОЖ с выставленным адресом 0;

0x71 – ДУОЖ с выставленным адресом 1.

5. Адреса в ответе

В пакет-ответе SLAVE-устройство выполняет перестановку адресов КОМУ/ОТ_КОГО.

6. Байт обнаружения ошибок CRC

Байт CRC служит для обнаружения (контроля) наличия ошибок в принятом пакете. Вычисляется для всех байтов пакета, начиная с поля SOH и кончая последним байтом поля <данные>. При расчете CRC используется полином:

$$X^8 + X^5 + X^4 + 1$$

Исходный код функций расчета контрольной суммы CRC представлен в приложении.

Преобразование зарезервированных символов выполняется прозрачным образом, т.е. при вычислении CRC будет учтен символ <X>, а не пара DLE <~X>.

Если значением CRC является один из зарезервированных символов, то <CRC> преобразуется по Таблице 1 в последовательность DLE <~CRC>.

7. Команды, реализованные в протоколе

Таблица 2. Команды

ASCII-код команды	16-тиричный код команды	Действие
‘S’	0x53	Зафиксировать текущее значение уровня как минимальное или максимальное.
‘P’	0x50	Считать значения минимального и максимального уровня.
‘F’	0x46	Заменить минимальный и максимальный уровень пользовательскими значениями
‘G’	0x47	Считать текущее значение уровня

Команды ‘S’,’P’, ‘F’ требуется для настройки аналогового выхода, после изменения длины датчика.

Команда ‘S’ (0x53)

Запрос:

SOH	1 байт	
<КОМУ>	1 байт	
<ОТ КОГО>	1 байт	
‘S’	1 байт	зафиксировать текущее значение уровня
параметр	1 байт	если 0 то зафиксировать текущее значение уровня как минимальное если 1 то как максимальное
CRC	1 байт	
ETX	1 байт	

Ответ:

SOH	1 байт	
<КОМУ>	1 байт	значение ОТ-КОГО пакета-запроса
<ОТ КОГО>	1 байт	значение КОМУ пакета-запроса
‘S’	1 байт	зафиксировано текущее значение уровня
параметр	1 байт	если 0 то зафиксировано текущее значение уровня как минимальное если 1 то как максимальное
CRC	1 байт	
ETX	1 байт	

Команда ‘P’ (0x50)

Запрос:

SOH	1 байт	
<КОМУ>	1 байт	
<ОТ КОГО>	1 байт	
‘P’	1 байт	считать значения минимального и максимального уровня
CRC	1 байт	
ETX	1 байт	

Ответ:

SOH	0x53	
<КОМУ>	1 байт	значение ОТ-КОГО пакета-запроса
<ОТ КОГО>	1 байт	значение КОМУ пакета-запроса
‘P’	1 байт	считаны значения минимального и максимального уровня
max	2 байт	значение максимального уровня (1 байт младший)
min	2 байт	значение минимального уровня (1 байт младший)
CRC	1 байт	
ETX	1 байт	

Команда ‘F’ (0x56)

Запрос:

SOH	1 байт	
<КОМУ>	1 байт	

<ОТ КОГО>	1 байт	
‘F’	1 байт	записать значения минимального и максимального уровня
max	2 байт	новое значение максимального уровня (1 байт младший)
min	2 байт	Новое значение минимального уровня (1 байт младший)
CRC	1 байт	
ETX	1 байт	

Ответ:

SOH	0x53	
<КОМУ>	1 байт	значение ОТ-КОГО пакета-запроса
<ОТ КОГО>	1 байт	значение КОМУ пакета-запроса
‘F’	1 байт	пользовательские значения минимального и максимального уровня установлены
CRC	1 байт	
ETX	1 байт	

Команда ‘G’ (0x47)

Запрос:

SOH	1 байт	
<КОМУ>	1 байт	
<ОТ КОГО>	1 байт	
‘G’	1 байт	считать текущее значение уровня
CRC	1 байт	
ETX	1 байт	

Ответ:

SOH	0x53	
<КОМУ>	1 байт	значение ОТ-КОГО пакета-запроса
<ОТ КОГО>	1 байт	значение КОМУ пакета-запроса
‘G’	1 байт	считано текущее значение уровня
cur	2 байт	текущее значение уровня жидкости (1 байт младший)
service	2 байт	служебные данные
CRC	1 байт	
ETX	1 байт	

Пример:

Получение одного значения измеряемой величины.

MASTER (программа «DUT NORM» с адресом 5)

FF 70 75 47 88 03

FF - SOH - символ начала пакета
70 - адрес КОМУ
75 - адрес ОТ_КОГО
47 - код команды 'G'
88 - значение CRC
03 - ETX - символ конца пакета

SLAVE (ДУОЖ с выставленным адресом 0)

FF 75 70 47 74 6D 00 00 F4 03

FF - SOH - символ начала пакета
70 - адрес КОМУ
75 - адрес ОТ_КОГО
47 - код команды 'G'
74 - поле измеряемая величина, младший байт
6D - поле измеряемая величина, старший байт
00 - резерв
00 - резерв
F4 - значение CRC
03 - ETX - символ конца пакета

Измеряемая величина **0x6D74** (28020).

ПРИЛОЖЕНИЕ

Алгоритм расчета байта обнаружения ошибок CRC

Общая функция вычисления CRC для всех байтов пакета.

```
unsigned char calc_crc(unsigned char packet_size, unsigned char *values)
{
    unsigned char crc = 0;
    while( packet_size-- )
    {
        crc = CRC8( *values++, crc );
    }
    return crc;
}++++
```

Варианты функций расчета CRC8 для полинома $X^8 + X^5 + X^4 + 1$

Алгоритм 1.

```
//---------------------------------------------------------------------------
unsigned char tab_crc[256] =
    {
                0x00,0x5E,0xBC,0xE2,0x61,0x3F,0xDD,0x83,
                0xc2,0x9c,0x7e,0x20,0xa3,0xfd,0x1f,0x41,
                0x9d,0xc3,0x21,0x7f,0xfc,0xa2,0x40,0x1e,
                0x5f,0x01,0xe3,0xbd,0x3e,0x60,0x82,0xdc,
                0x23,0x7D,0x9F,0xC1,0x42,0x1C,0xFE,0xA0,
                0xE1,0xBF,0x5D,0x03,0x80,0xDE,0x3C,0x62,
                0xBE,0xE0,0x02,0x5C,0xDF,0x81,0x63,0x3D,
                0x7C,0x22,0xC0,0x9E,0x1D,0x43,0xA1,0xFF,
                0x46,0x18,0xFA,0xA4,0x27,0x79,0x9B,0xC5,
                0x84,0xDA,0x38,0x66,0xE5,0xBB,0x59,0x07,
                0xDB,0x85,0x67,0x39,0xBA,0xE4,0x06,0x58,
                0x19,0x47,0xA5,0xFB,0x78,0x26,0xC4,0x9A,
                0x65,0x3B,0xD9,0x87,0x04,0x5A,0xB8,0xE6,
                0xA7,0xF9,0x1B,0x45,0xC6,0x98,0x7A,0x24,
                0xF8,0xA6,0x44,0x1A,0x99,0xC7,0x25,0x7B,
                0x3A,0x64,0x86,0xD8,0x5B,0x05,0xE7,0xB9,
                0x8C,0xD2,0x30,0x6E,0xED,0xB3,0x51,0x0F,
                0x4E,0x10,0xF2,0xAC,0x2F,0x71,0x93,0xCD,
                0x11,0x4F,0xAD,0xF3,0x70,0x2E,0xCC,0x92,
                0xD3,0x8D,0x6F,0x31,0xB2,0xEC,0x0E,0x50,
                0xAF,0xF1,0x13,0x4D,0xCE,0x90,0x72,0x2C,
                0x6D,0x33,0xD1,0x8F,0x0C,0x52,0xB0,0xEE,
                0x32,0x6C,0x8E,0xD0,0x53,0x0D,0xEF,0xB1,
                0xF0,0xAE,0x4C,0x12,0x91,0xCF,0x2D,0x73,
                0xCA,0x94,0x76,0x28,0xAB,0xF5,0x17,0x49,
                0x08,0x56,0xB4,0xEA,0x69,0x37,0xD5,0x8B,
                0x57,0x09,0xEB,0xB5,0x36,0x68,0x8A,0xD4,
                0x95,0xCB,0x29,0x77,0xF4,0xAA,0x48,0x16,
                0xE9,0xB7,0x55,0x0B,0x88,0xD6,0x34,0x6A,
                0x2B,0x75,0x97,0xC9,0x4A,0x14,0xF6,0xA8,
                0x74,0x2A,0xC8,0x96,0x15,0x4B,0xA9,0xF7,
                0xB6,0xE8,0x0A,0x54,0xD7,0x89,0x6B,0x35
     };

unsigned char CRC8( unsigned char value, unsigned char crc )
{
    return tab_crc[ value ^ crc ];
}
//---------------------------------------------------------------------------
```

Алгоритм 2:

```
//---------------------------------------------------------------------------
unsigned char CRC8( unsigned char value, unsigned char crc )
{
    unsigned char i = value ^ crc;
    crc = 0;
    if(i & 0x01) crc ^= 0x5e;
    if(i & 0x02) crc ^= 0xbc;
    if(i & 0x04) crc ^= 0x61;
    if(i & 0x08) crc ^= 0xc2;
    if(i & 0x10) crc ^= 0x9d;
    if(i & 0x20) crc ^= 0x23;
    if(i & 0x40) crc ^= 0x46;
    if(i & 0x80) crc ^= 0x8c;
    return crc;
}
//---------------------------------------------------------------------------
```

Алгоритм 3.

```
//---------------------------------------------------------------------------
unsigned char CRC8( unsigned char value, unsigned char crc )
{
    unsigned char i = 8;
    do
    {
        if ( ((value ^ crc ) & 0x01) > 0 )
        {
            crc = ( (crc >> 1 ) ^ 0x0C) | 0x80;
        }
        else
        {
                crc >>= 1;
        }
        value >>= 1;
    }
    while(--i);
    return crc;
}
//---------------------------------------------------------------------------
```