**Общая информация**

Настройки порта для RS-485, RS-232: 57600, 8, n, 1.

Также возможен обмен данными через сокет TCP/IP или по протоколу UDP.

Настройки IP адресов и номера портов задаются программой конфигурации. Структура данных одинакова для всех каналов связи.

Используемые в структурах типы данных (младший байт располагается первым):

typedef unsigned char byte; //8 bit unsigned

typedef unsigned short word; //16 bit unsigned

typedef unsigned int dword; //32 bit unsigned

typedef signed int int32; //32 bit signed

**Общая структура команд.**

Обмен данными производится по инициативе внешнего управляющего контроллера (компьютера, хоста и т.п.).

Все команды, передаваемые от управляющего контроллера (хоста) к контроллеру, а также ответы от табло к хосту упакованы в байтовые последовательности следующего формата (пакеты верхнего уровня). Ниже показана побайтовая структура пакета верхнего уровня:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Маркер начала пакета | Длина пакета (младшие 7 бит) | Длина пакета (старшие 7 бит) | Закодированный пакет второго уровня | Контрольная сумма (младшие 7 бит) | Контрольная сумма (старшие 7 бит) | Маркер конца пакета |
| 0x02 | (Len & 0x7F) | 0x80 | ((Len >>7) & 0x7F) | 0x80 | byte Data2[xLen] | (CS & 0x7F) | 0x80 | ((CS >>7) & 0x7F) | 0x80 | 0x03 |

Пакет верхнего уровня содержит маркеры начала и конца, информацию о длине данных внутри пакета, непосредственно сами данные (пакет второго уровня) и контрольную сумму.

Все байты в пакете верхнего уровня закодированы таким образом, чтобы в нем не встречались байты значением менее 0x20, за исключением маркеров начала или конца пакета.

Кодировка пакета второго уровня осуществляется по следующим правилам:

1. Считается контрольная сумма пакета.

2. У каждого байта b инвертируется старший бит (операция b=b^0x80).

3. Если получившийся байт b больше или равен 0x20 и не равен 0x7F, то он остается без изменений (b).

4. Если получившийся байт меньше, чем 0x20 или равен 0x7F, то он заменяется на 2 байта: 0x7F и (b | 0x80).

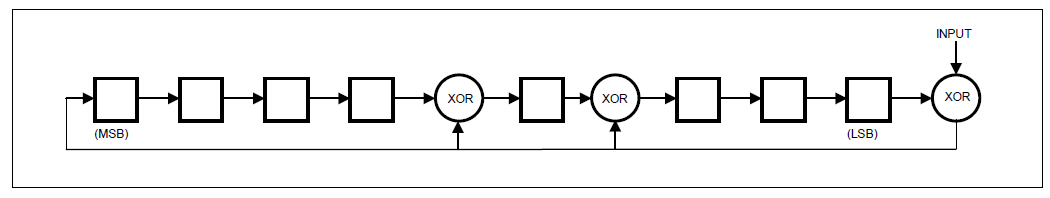
Параметр Len в пакете верхнего уровня равен длине **незакодированного** пакета второго уровня.

Параметр Len не может иметь значение более 1024.

Длина закодированного пакета в структуре пакета верхнего уровня условно обозначена как xLen.

После приема пакета, ограниченного маркерами начала и конца, устройство производит раскодировку вложенного пакета второго уровня, сверяет длину, контрольную сумму, и, в случае успешной проверки, обрабатывает пакет.

Пакеты, содержащие неверные символы (0x00 .. 0x1F), неверную длину или неверную контрольную сумму никак не обрабатываются!

Контрольная сумма имеет разрядность 14 бит. Младшие 8 бит подсчитываются в виде полиноминальной суммы, эквивалентный полином: CRC = X8 + X5 + X4 + 1

Старшие 6 бит контрольной суммы считаются так: суммируются инвертированные байты и берутся младшие 6 бит суммы.

В сумму входят **все байты пакета второго уровня**. Сумма считается на **незакодированном** пакете второго уровня!

Ниже будут описываться только структуры незакодированного пакета второго уровня для команд разного типа.

**Общий формат пакетов второго уровня.**

Пакет второго уровня имеет в общем случае вид следующей структуры:

typedef struct {

word SrcAddr; //адрес отправителя

word DstAddr; //адрес получателя

byte PId; //идентификатор пакета - инкрементируемое или случайное число,

//служит для идентификации ответа от табло.

//Если значение равно Pid предыдущего пакета, пакет считается повторным и игнорируется.

byte Cmd; // код команды

byte Flags; // флаги-опции команды. В нормальном режиме равен 0.

//Если установлен бит 0 (PACK\_FLAG\_NOANSWER 0x01) - табло не отправляет ответ,

//содержащий статус. Это может быть полезным при

// //отправке одной команды одновременно всем табло по RS-485 (DstAddr=0xFFFF)

//или по UDP протоколу broadcast пакетом (DstAddr=0xFFFF)

//Если установлен бит 1 (PACK\_FLAG\_NOCHECKID 0x02) - табло не проверяет поле Pid.

//Если установлен бит 2 (PACK\_FLAG\_RESEND 0x04) табло пересылает пакет по внутренней шине RS485

//в случае, если DstAddr не является адресом данного табло

byte Status; //статус выполнения команды

word DataLen; //длина данных в поле Data, от 0 до 511

byte Data[DataLen]; //данные

} TComPacket;

Структура TComPacket имеет побайтное выравнивание полей.

Поля SrcAddr, DstAddr, PId, Cmd, Flags, Status, DataLen присутствуют всегда. Длина поля Data определяется полем DataLen.

Как следует из вышеописанного, значение DataLen связано со значением Len в пакете верхнего уровня соотношением DataLen=Len-10. При приеме пакетов данное соотношение также должно проверяться.

При передаче **команды** **от хоста к табло**, поле SrcAddr должно быть равно адресу хоста (по умолчанию ноль), DstAddr - адресу табло (по умолчанию 1). Если DstAddr==0xFFFF, то на него будут откликаться все табло.

PId - может быть случайным числом или инкрементироваться от пакета к пакету. Не рекомендуется делать его каким-то постоянным числом.

Cmd - непосредственно код команды (см. ниже описание команд).

Flags - обычно равен нулю, но может содержать PACK\_FLAG\_NOANSWER и/или PACK\_FLAG\_NOCHECKID.

Status - равен 0x80, что означает, что данный пакет - команда для табло.

DataLen, Data - соответствуют описанию данной команды.

При передаче **ответа** **от табло к хосту**, поле SrcAddr должно быть равно адресу табло (по умолчанию 1), DstAddr - адресу хоста (по умолчанию 0). SrcAddr и DstAddr не могут быть равны 0xFFFF.

PId - берется из обрабатываемой команды от хоста.

Cmd - берется из обрабатываемой команды от хоста.

Flags - равен нулю.

Status - результат обработки команды (код ошибки). 0 - нет ошибок. Бит 7 (0x80) в ответе всегда сброшен, что означает, что это ответ на команду от табло.

DataLen, Data - соответствуют описанию данной команды. DataLen не нулевое (данные присутствуют), если ответ на команду подразумевает какую-либо передачу данных от табло к хосту (см. описание команд).

Возможные коды ошибок в ответе от табло (Status):

#define ecOK 0x0000 //нет ошибок

#define ecBadCommand 0x0001 //несуществующая команда

#define ecParamError 0x0002 //неверные параметры команды

#define ecExecuteError 0x0003 //невозможно выполнить команду

**Команды управления табло.**

В данном разделе будут описаны только те значения полей структуры TComPacket для команд от хоста к табло и ответов от табло к хосту, которые являются специфическими для данной команды и не указаны в общем описании, приведенном выше. Если значение поля DataLen не указано явно, то подразумевается, что оно равно нулю. Поля Cmd и PId в ответах от табло равны аналогичным в обработанной команде.

**Проверка соединения.**

Данная команда не выполняет никаких действий и может использоваться для определения наличия табло в сети.

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x01 |

Ответный пакет от табло к хосту соответствует общему описанию. Код ошибки (Status) всегда нулевой.

**Установка яркости табло.**

Данная команда устанавливает яркость табло.

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x02 |
| DataLen | 1 |
| Data[0] | От 1 до 10 - яркость |

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Установка часов табло.**

Данная команда устанавливает время во внутренних часах табло.

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x03 |
| DataLen | 6 |
| Data[0] | Секунды от 0 до 59 |
| Data[1] | Минуты от 0 до 59 |
| Data[2] | Часы от 0 до 23 |
| Data[3] | День от 1 до 31 |
| Data[4] | Месяц от 1 до 12 |
| Data[5] | Год от 0 до 99 |

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Чтение часов табло.**

Данная команда читает время из внутренних часов табло.

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x04 |

Поля ответного пакета от табло к хосту:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| DataLen | 6 |
| Data[0] | Секунды от 0 до 59 |
| Data[1] | Минуты от 0 до 59 |
| Data[2] | Часы от 0 до 23 |
| Data[3] | День от 1 до 31 |
| Data[4] | Месяц от 1 до 12 |
| Data[5] | Год от 0 до 99 |

**Вывод текста в текстовые зоны**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x05 |
| DataLen | N+9, где N - длина выводимого текста |
| Data[0] | Номер дисплея - младшие 6 бит (от 0 до 63). Старшие два бита - тип дисплея |
| Data[1] | Номер шрифта (от 0 до 7) |
| Data[2] | Выравнивание (0 влево, 1 - вправо, 2 - по центру) |
| Data[3] | Опции показа, битовые флаги. |
| Data[4] | Младшие 3 бит скорость движения (от 0 до 5) |
| Data[5] | Цвет (от 1 до 7). |
| Data[6] | Номер эффекта |
| Data[7] | Кол-во повторов эффекта |
| Data[8] | Временной интервал эффекта |
| Data[9] | Символ 1 текста |
| Data[10] | Символ 2 текста |
| ..... |  |
| Data[N+8] | Символ N текста |

Примечание:

1. Тип дисплея определяется двумя старшими битами и может принимать следующие значения:  
    0x00 - дисплей по умолчанию. Если в настройках сегментных дисплеев стоит галочка   
    "использовать по умолчанию" - вывод будет производится на сегментный дисплей,   
    иначе - на текстовый  
    0x01 - вывод будет произведен на текстовый дисплей  
    0x02 - вывод будет произведен на сегментный дисплей  
    0x03 - резерв
2. Опция показа может включать флаги, такие как:   
    0x80 - сохранить в энергонезависимой памяти  
    0x01 - пауза во время эффекта (имеет смысл только для бегущего текста)
3. Скорость движения бегущего текста может быть от 0 до 5. Бит 7 (0x80) задает адаптивную скорость - если текст целиком помещается на дисплее, то он будет неподвижен, иначе будет бежать с указанной скоростью.
4. Цвет: бит 0 - канал 1 (красный), бит 0 - канал 2 (зеленый), бит 0 - канал 3 (синий),
5. Кодировка выводимого текста Win-1251
6. Номер эффекта:  
    0 - без эффекта  
    1 - пауза (имеет смысл только для бег. текста)  
    2 - мигание
7. Кол-во повторов эффекта: для мигания это кол-во миганий, для паузы - время паузы вычисляется как произведение кол-ва повторов на временной интервал эффекта
8. Временной интервал эффекта - для паузы время рассчитывается, как указано в п. 6, для мигания - это время активной (включено) и неактивной (выключено) фазы.

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Вывод текста в текстовые поля**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x1B |
| DataLen | N+7, где N - длина выводимого текста |
| Data[0] | номер ряда поля |
| Data[1] | номер столбца поля |
| Data[2] | Номер шрифта (от 0 до 7) |
| Data[3] | Цвет (от 1 до 7). |
| Data[4] | Выравнивание (0 влево, 1 - вправо, 2 - по центру) |
| Data[5] | Опции - битовые флаги |
| Data[6] | Не используется, всегда ноль |
| Data[7] | Символ 1 текста |
| Data[8] | Символ 2 текста |
| ..... |  |
| Data[N+6] | Символ N текста |

Примечание:

1. Опция показа может включать флаги, такие как:   
 0x80 - сохранить в энергонезависимой памяти

0x01 - вывод текста будет производиться по абсолютному номеру поля (иначе по ряду и столбцу).

Байт Data[0] будет проигнорирован, абсолютный номер поля будет взят из Data[1].

2. Кодировка выводимого текста Win-1251

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Вывод данных на семисегментные (цифровые) индикаторы по строкам и столбцам**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x0F |
| DataLen | N+9, где N - длина выводимого текста |
| Data[0] | Номер строки (отсчет от 0) |
| Data[1] | Номер столбца (отсчет от 0) |
| Data[2] | Выравнивание (0 влево, 1 - вправо, 2 - по центру) |
| Data[3] | Опции показа, битовые флаги. |
| Data[4] | Не используется |
| Data[5] | Не используется |
| Data[6] | Номер эффекта |
| Data[7] | Кол-во повторов эффекта |
| Data[8] | Не используется |
| Data[9] | Символ 1 текста |
| Data[10] | Символ 2 текста |
| ..... |  |
| Data[N+8] | Символ N текста |

Примечание:

1. Опция показа может включать флаги, такие как:   
    0x80 - сохранить в энергонезависимой памяти
2. Цвет: бит 0 - канал 1 (красный), бит 0 - канал 2 (зеленый), бит 0 - канал 3 (синий),
3. Кодировка выводимого текста Win-1251
4. Номер эффекта:  
    0 - без эффекта  
    2 - мигание
5. Кол-во повторов эффекта: для мигания это кол-во миганий

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Считывание данных (семисегментные индикаторы) по строкам и столбцам**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x10 |
| DataLen | 4 |
| Data[0] | Номер строки (отсчет от 0) |
| Data[1] | Номер столбца (отсчет от 0) |
| Data[2] | Не используется |
| Data[3] | Не используется |

Поля ответного пакета от табло к хосту:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |
| DataLen | N - длина текста |
| Data[0] | Символ 1 текста |
| Data[1] | Символ 2 текста |
| ..... |  |
| Data[N-1] | Символ N текста |

Поле Status принимает значение ecParamError в том случае, если поля данных с указанными номером строки или столбца не существует.

**Установка значений счетчиков**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x53 |
| DataLen | 2+N\*4, где N - кол-во счетчиков, для которых устанавливаются значения |
| Data[0] | 1 - идентификатор типа данных (всегда 0x01) |
| Data[1] | Index - индекс первого устанавливаемого счетчика (от 0 до 11) |
| Data[2..5] | Значение счетчика с индексом index (тип int32) |
| Data[5..8] | Значение счетчика с индексом index+1 (тип int32) |
| ..... | ...... |
| Data[N\*4-2 ... N\*4+1] | Значение счетчика с индексом index+N-1 (тип int32) |

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Считывание значений счетчиков**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0x52 |
| DataLen | 3 |
| Data[0] | 1 - идентификатор типа данных (всегда 0x01) |
| Data[1] | Index - индекс первого устанавливаемого счетчика (от 0 до 11) |
| Data[2] | Item\_count - кол-во одновременно считываемых счетчиков (от 1 до 12) |

Поля ответного пакета от табло к хосту:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |
| DataLen | N\*4, где N - кол-во счетчиков, для которых считываются значения |
| Data[0..3] | Значение счетчика с индексом index (тип int32) |
| Data[4..7] | Значение счетчика с индексом index+1 (тип int32) |
| ..... | ...... |
| Data[N\*4-4 ... N\*4-1] | Значение счетчика с индексом index+N-1 (тип int32) |

Поле Status принимает значение ecParamError в том случае, если максимальный индекс счетчика больше 11 (т.е., если сумма index+item\_count>12).

**Вывод сигнала на внешнюю нагрузку (гудок)**

Поля пакета от хоста к табло:

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Cmd | 0xE6 (230) |
| DataLen | 16 – размер структуры TExtFuncData |
| Data[0..15] | Содержимое структуры TExtFuncData |

Структура TExtFuncData для данной команды имеет следующий вид:

typedef struct {

dword cmd\_id; // константа 0x5C1E9A49

dword func\_id; // константа 0x50 (80)

dword line\_no; // номер линии нагрузки от 0 до 7

dword beep\_length; // длительность сигнала в миллисекундах

} TExtFuncData;

Поле line\_no определяет номер выхода контроллера, куда выдается сигнал (нумерация от нуля).

Поле beep\_length задает длительность сигнала в миллисекундах.

Если beep\_length=0xFFFFFFFF, то заданный сигнал будет находиться в активном состоянии бесконечно (до отключения питания контроллера, либо до получения команды с другим значением beep\_length).

Если beep\_length=0, то заданный сигнал переходит в неактивное состояние.

Для корректной работы, ненулевое значение beep\_length должно быть не менее 20 мс (сигнал меньшей длительности может быть не выведен). Точность длительности сигнала на выходе контроллера -3/+20 мс.

Поля ответного пакета от табло к хосту

|  |  |
| --- | --- |
| **Имя поля** | **Значение** |
| Status | ecOK или ecParamError |

**Функция подсчета контрольной суммы.**

Передаваемые параметры:

DataPtr - указатель на начало пакета второго уровня TComPacket

Cnt - длина пакета, Cnt= DataLen+10, где DataLen - значения поля DataLen структуры TComPacket.

Выходные параметры: байты cslo и cshi. Значащими являются 8 бит cslo и 6 бит cshi.

void CountCSNew(void \*DataPtr, dword Cnt, byte \*cslo, byte \*cshi){

dword cs1, cs2;

byte \*Buff;

cs1=0;

cs2=0;

Buff=DataPtr;

while (Cnt) {

cs1=CrcTable[cs1 ^ (\*Buff)];

cs2 +=~(\*Buff);

Buff++;

Cnt--;

};

\*cslo=cs1;

\*cshi=cs2;

}

Полученный результат cslo и cshi может быть перекодирован в результат типа word для непосредственной вставки в пакет первого уровня следующей функцией CS2word:

word CS2word(byte cslo, byte cshi){

return (0x8080 | cslo | ((((word)cslo)<<1) & 0x100) | (((word)cshi)<<9));

}

Этой же функцией CS2word кодируется и значение Len в пакете первого уровня:

CodedLen = CS2word(Lo(Len), Hi(Len)) (CodedLen - полученное значение, непосредственно вставляемое в пакет первого уровня).

Используется следующая таблица констант:

const byte CrcTable[256]={

0, 94, 188, 226, 97, 63, 221, 131, 194, 156, 126, 32, 163, 253, 31, 65,

157, 195, 33, 127, 252, 162, 64, 30, 95, 1, 227, 189, 62, 96, 130, 220,

35, 125, 159, 193, 66, 28, 254, 160, 225, 191, 93, 3, 128, 222, 60, 98,

190, 224, 2, 92, 223, 129, 99, 61, 124, 34, 192, 158, 29, 67, 161, 255,

70, 24, 250, 164, 39, 121, 155, 197, 132, 218, 56, 102, 229, 187, 89, 7,

219, 133, 103, 57, 186, 228, 6, 88, 25, 71, 165, 251, 120, 38, 196, 154,

101, 59, 217, 135, 4, 90, 184, 230, 167, 249, 27, 69, 198, 152, 122, 36,

248, 166, 68, 26, 153, 199, 37, 123, 58, 100, 134, 216, 91, 5, 231, 185,

140, 210, 48, 110, 237, 179, 81, 15, 78, 16, 242, 172, 47, 113, 147, 205,

17, 79, 173, 243, 112, 46, 204, 146, 211, 141, 111, 49, 178, 236, 14, 80,

175, 241, 19, 77, 206, 144, 114, 44, 109, 51, 209, 143, 12, 82, 176, 238,

50, 108, 142, 208, 83, 13, 239, 177, 240, 174, 76, 18, 145, 207, 45, 115,

202, 148, 118, 40, 171, 245, 23, 73, 8, 86, 180, 234, 105, 55, 213, 139,

87, 9, 235, 181, 54, 104, 138, 212, 149, 203, 41, 119, 244, 170, 72, 22,

233, 183, 85, 11, 136, 214, 52, 106, 43, 117, 151, 201, 74, 20, 246, 168,

116, 42, 200, 150, 21, 75, 169, 247, 182, 232, 10, 84, 215, 137, 107, 53

};