WAKE Serial Protocol
specification

Протокол WAKE является логическим уровнем интерфейса управления оборудованием с помощью асинхронного последовательного канала. Физический уровень интерфейса протоколом не определяется, может использоваться, например, RS-232 или RS-485. Протокол позволяет производить обмен пакетами данных (data frames) длиной до 255 байт с адресуемыми устройствами, которых может быть до 127. Последовательный канал должен быть сконфигурирован следующим образом:

- число бит в посылке – 8
- количество стоп-бит – 1
- бит четности – нет
- скорость обмена – 300…115200 бод
- использование линий управления модемом – произвольное

Основой протокола WAKE является протокол SLIP (UNIX™ Serial Link Interface Protocol). Передача данных осуществляется в двоичном виде, т.е. используются все возможные значения байта (00h…FFh). Для передачи служебной информации зарезервированы два кода: FEND = C0h (Frame End) и FESC = DBh (Frame Escape). Управляющий код FEND служит для обозначения начала посылки, а код FESC служит для передачи ESC-последовательностей. Если в потоке данных встречаются байты, значения которых совпадают с управляющими кодами, производится подмена этих байт ESC-последовательностями. Такой механизм называют байт-стаффингом (byte stuffing). Код FEND заменяется последовательностью <FESC>, <TFEND>, а код FESC – последовательностью <FESC>, <TFESC>, где TFEND = DCh (Transposed FEND), TFESC = DDh (Transposed FESC). Коды TFEND и TFESC являются управляющими только в ESC-последовательностях, поэтому при передаче данных они в подмене не нуждаются.

Таблица 1. Управляющие коды протокола WAKE

Обозначение	Пояснение	HEX-значение
FEND	Frame End	C0h
FESC	Frame Escape	DBh
TFEND	Transposed Frame End	DCh
TFESC	Transposed Frame Escape	DDh

Таблица 2. Подмена байт данных ESC-последовательностями

Байт данных	Передаваемая последовательность
C0h	DBh, DCh
DBh	DBh, DDh

Структура пакета WAKE следующая: пакет всегда начинается управляющим кодом FEND (C0h). Затем следует необязательный байт адреса, после которого идет байт команды. За ним следует байт количества данных и собственно байты данных. Завершает пакет необязательный байт контрольной суммы CRC-8.

Таблица 3. Структура пакета WAKE

FEND	ADDR	CMD	N	Data1	…	DataN	CRC

FEND: Управляющий код FEND (C0h) является признаком начала пакета. Благодаря стаффингу, этот код больше нигде в потоке данных не встречается, что позволяет в любой ситуации однозначно определять начало пакета.

ADDR: Байт адреса используется для адресации отдельных устройств. На практике распространена ситуация, когда управление осуществляется только одним устройством. В таком случае байт адреса не требуется, и его можно не передавать. Вместо него сразу за кодом FEND передается байт команды CMD. Для того, чтобы можно было однозначно установить, адресом или командой является второй байт пакета, введены некоторые ограничения. Для адресации используется 7 бит, а старший бит, передаваемый вместе с адресом, должен всегда быть установлен:

	D7	D6	D5	D4	D3	D2	D1	D0
ADDR =	1	A6	A5	A4	A3	A2	A1	A0

Иногда возникает необходимость передать какую-то команду или данные сразу всем устройствам. Для этого предусмотрен коллективный вызов (broadcast), который осуществляется путем передачи нулевого адреса (учитывая единичный старший бит, в этом случае передаваемый байт равен 80h). Нужно отметить, что передача в пакете нулевого адреса полностью аналогична передаче пакета без адреса. Поэтому при реализации протокола можно автоматически исключать нулевой адрес из пакета. Учитывая разрядность адреса и один зарезервированный адрес для коллективного вызова, максимальное количество адресуемых устройств составляет 127.

Если возникает необходимость передать значение адреса 40h или 5Bh (передаваемый байт в этом случае будет равен C0h или DBh), то производится стаффинг, т.е. передача ESC-последовательности (см. таблицу 2). Поэтому следует учитывать, что устройства с такими адресами требуют большей на один байт длины пакета. Это может быть заметно в тех случаях, когда используются короткие пакеты. В таких случаях следует избегать назначения устройствам названных адресов.

CMD: Байт команды всегда должен иметь нулевой старший бит:

	D7	D6	D5	D4	D3	D2	D1	D0
CMD =	0	C6	C5	C4	C3	C2	C1	C0

Таким образом, код команды занимает 7 бит, что позволяет передавать до 128 различных команд. Коды команд выбираются произвольно в зависимости от нужд приложения. Рекомендуется использовать несколько стандартных кодов команд:

Таблица 4. Стандартные коды команд протокола WAKE

Код	Название	Описание команды
00h	C_Nop	Нет операции
01h	C_Err	Передача кода ошибки
02h	C_Echo	Запрос возврата переданного пакета
03h	C_Info	Запрос информации об устройстве

Коды остальных команд выбираются в зависимости от нужд приложения. Команды обычно имеют несколько параметров, которые передаются далее в виде пакета данных.

Поскольку код команды всегда имеет нулевой старший бит, этот код никогда не совпадает с управляющими кодами. Поэтому при передаче команды стаффинг никогда не производится.

N: Байт количества данных имеет значение, равное количеству передаваемых байт данных:

	D7	D6	D5	D4	D3	D2	D1	D0
N =	N7	N6	N5	N4	N3	N2	N1	N0

Таким образом, код количества данных занимает 8 бит, в результате один пакет может содержать до 255 байт данных. Значение N не учитывает служебные байты пакета FEND, ADDR, CMD, N и CRC. В результате стаффинга фактическая длина пакета может возрасти. Значение N **не** учитывает этот факт и отражает количество полезных байт данных (т.е. значение N всегда таково, как будто стаффинг не осуществляется). Если передаваемая команда не имеет параметров, то передается N = 00h и байты данных опускаются.

Если возникает необходимость передать значение N, равное C0h или DBh, то производится стаффинг, т.е. передача ESC-последовательности (см. таблицу 2). Однако при таких больших значениях N длина пакета столь велика, что его удлинение еще на один байт практически незаметно.

Data1…DataN: Байты данных, количество которых определяется значением N. При N = 00h байты данных отсутствуют. Байты данных могут иметь любое значение, кроме FEND (C0h) и FESC (DBh). Если возникает необходимость передать одно из этих значений, то производится стаффинг, т.е. передача ESC-последовательности (см. таблицу 2), состоящей из управляющего кода FESC и кода TFEND (TFESC).

CRC: Байт контрольной суммы CRC-8. Может отсутствовать в **некоторых реализациях** протокола. Контрольная сумма CRC-8 рассчитывается **перед** операцией стаффинга для всего пакета, начиная с байта FEND и заканчивая последним байтом данных. Если в пакете передается адрес, то при вычислении контрольной суммы используется его истинное значение, т.е. единичный старший бит не учитывается. Для расчета контрольной суммы используется полином CRC = $X^8 + X^5 + X^4 + 1$. Значение CRC перед вычислением инициализируется числом DEh. При передаче значения байта контрольной суммы C0h и DBh заменяются ESC-последовательностями (см. таблицу 2).

Таблица 5. Величина избыточности протокола WAKE

Вид пакета	Избыточность, %
FEND, CMD, 00h, CRC	75,0
FEND, CMD, 00h	66,7
FEND, ADDR, CMD, 00h, CRC	60,0
FEND, ADDR, CMD, 00h	50,0
FEND, ADDR, CMD, 0Ah, <10 bytes of data>, CRC	20,0
FEND, ADDR, CMD, 32h, <50 bytes of data>, CRC	5,5
FEND, ADDR, CMD, 7Fh, <127 bytes of data>, CRC	2,3
FEND, CMD, 7Fh, <127 bytes of data>, CRC	2,3
FEND, ADDR, CMD, 7Fh, <127 bytes of data>	1,5
FEND, CMD, 7Fh, <127 bytes of data>	1,5

Со стороны PC протокол реализован в динамической библиотеке wsp32.dll, которая может использоваться при разработке коммуникационных программ на любом языке. Дальнейшее описание предполагает, что библиотека используется приложением, написанным на языке C++. Заголовочный файл этой библиотеки приведен ниже:

```
#ifndef WSP32_H
#define WSP32_H

#ifdef WSP32_Exports
#define WSP32_API __declspec(dllexport)
#else
#define WSP32_API __declspec(dllimport)
#endif

extern "C"
 {

  WSP32_API bool WINAPI AccessCOM(char *P);
  WSP32_API bool WINAPI OpenCOM(char *P, DWORD baud);
  WSP32_API bool WINAPI CloseCOM(void);
  WSP32_API bool WINAPI SetModLns(DWORD F);
  WSP32_API bool WINAPI GetModLns(LPDWORD lpD);
  WSP32_API bool WINAPI PurgeCOM(void);
  WSP32_API bool WINAPI FlushCOM(void);
  WSP32_API bool WINAPI GetMaskCOM(LPDWORD lpEvtMask);
  WSP32_API bool WINAPI SetMaskCOM(DWORD EvtMask);
  WSP32_API bool WINAPI WaitEventCOM(LPDWORD lpEvtMask);
  WSP32_API bool WINAPI RxFrame(DWORD To, unsigned char &ADD,
             unsigned char &CMD, unsigned char &N, unsigned char *Data);
  WSP32_API bool WINAPI TxFrame(unsigned char ADDR, unsigned char CMD,
             unsigned char N, unsigned char *Data);
 }

#endif
```

bool AccessCOM(char *P) – функция проверяет доступность порта. В качестве параметра передается имя порта (например, “COM1”). Возвращает true в случае доступности порта.

bool OpenCOM(char *P, DWORD baud) – функция открывает порт. В качестве параметров передаются имя порта (например, “COM1”) и скорость в бодах, которая может принимать одно из стандартных значений (например, 115200). Возвращает true в случае успешного выполнения. Функция устанавливает на линии DTR уровень –12В, а на линии RTS уровень +12В.

bool CloseCOM(void) – функция закрывает порт. Возвращает true в случае успешного выполнения.

bool SetModLns(DWORD F) – функция осуществляет управление линиями RTS и DTR. Возвращает true в случае успешного выполнения. Параметр такой же, как у функции API EscapeCommFunction.

bool GetModLns(LPDWORD lpD) – функция считывает линии управления модема CTS и DSR. Возвращает true в случае успешного выполнения. Параметр такой же, как у функции API GetCommModemStatus.

bool PurgeCOM(void) – функция очищает буфер COM-порта и прерывает текущие операции приема/передачи. Возвращает true в случае успешного выполнения.

bool FlushCOM(void) – функция очищает буфер COM-порта, дождавшись завершения передачи. Возвращает true в случае успешного выполнения.

bool GetMaskCOM(LPDWORD lpEvtMask) – функция считывает маску событий COM-порта. Возвращает true в случае успешного выполнения. Параметр такой же, как у функции API GetCommMask.

bool SetMaskCOM(DWORD EvtMask) – функция устанавливает маску Возвращает true в случае успешного выполнения. Параметр такой же, как у функции API SetCommMask.

bool WaitEventCOM(LPDWORD lpEvtMask) – функция служит для ожидания события COM-порта. Возвращает true в случае успешного выполнения.

bool RxFrame(DWORD To, unsigned char &ADD, unsigned char &CMD, unsigned char &N, unsigned char *Data) – функция осуществляет прием WAKE-пакета. Возвращает true в случае успешного выполнения.

bool TxFrame(unsigned char ADDR, unsigned char CMD, unsigned char N, unsigned char *Data) - функция осуществляет передачу WAKE-пакета. Возвращает true в случае успешного выполнения.