Описание интерфейса библиотеки Prox232.dll

Введение

БиблиотекаProx232.DLL (далее просто библиотека) реализует низкоуровневый, на уровне транспортного кадра, обмен данными со считывателями. Реализация протокола высокого уровня ложится на приложение, использующее библиотеку, чем достигается независимость библиотеки от любых изменений протокола верхнего уровня, в т.ч. и будущих.

Основные особенности

- позволяет использовать любой стандартный (встроенный) либо дополнительный, в т.ч. и виртуальный COM-порт с настройкой скорости обмена данными;
- выполняет прозрачную для вызывающего приложения упаковку/распаковку данных в транспортный кадр (байтстаффинг) и вычисление и проверку контрольных сумм;
- встроенная система семафоров (mutex), позволяющая одновременное обращение к считывателю сразу нескольких программ или процессов;
- встроенный модуль трассировки вызовов, позволяющий получить протокол обмена в виде текстового файла.

История

11.05.2003 начальная версия
25.08.2003 добавлено конфигурирование параметров протокола
- варианты расчета контрольной суммы кадра
- запрет/разрешение поля адреса.

Константы, типы и структуры данных

Константы

Имя	Значение	Описание
Коды ответа оборудования		
REPLY_OK	0x00	Команда выполнена успешно, получены данные
REPLY_ACK_NACK	0x2A	Признак кадра ACK/NACK, вне библиотеки не используется и упомянуто только для полноты описания
REPLY_ACK	0x55	Получен ответ ACK
REPLY_NACK1	0x01	Получен ответ NACK1
REPLY_NACK2	0x02	Получен ответ NACK2
REPLY_NACK3	0x03	Получен ответ NACK3

REPLY_NACK4	0x04	Получен ответ NACK4
REPLY_NACK5	0x05	Получен ответ NACK5
REPLY_NACK6	0x06	Получен ответ NACK6
Коды ошибок при работе с портом		
REPLY_MISS	0xFF	Ответ не получен (отсутствует считыватель)
REPLY_ERR_OPENPORT	0xFE	Ошибка открытия порта
REPLY_ERR_WRITEPORT	0xFD	Ошибка записи в порт
REPLY_ERR_READPORT	0xFC	Ошибка чтения из порта
REPLY_ERR_LOCKPORT	0xFB	Истек таймаут при попытке захватить порт
REPLY_ERR_PORTNOTSET	0xFA	Параметры порта не установлены
REPLY_ERR_INVALIDANSWER	0xF9	Ответ не удалось правильно интерпретировать
REPLY_ERR_CRC	0xF8	Ошибка контрольной суммы
REPLY_ERR_INVALIDFRAME	0xF7	Неверный идентификатор кадра
REPLY_ERR_UNKNOWN	0xF0	Неизвестная ошибка, пока не используется.
Методы расчета контрольной суммы кадра		
CHECK_NONE	0x00	Не проверять контрольную сумму кадра
CHECK_SUM	0x01	Использовать суммирование (1 байт)
CHECK_CRC16	0x02	Использовать CRC16 (2 байта)
CHECK_FCS	0x03	Использовать FCS (2 байта)

Типы данных

Следующие типы данных являются членами структуры TLogInfo и предназначены для хранения null-terminated строк, используемых для настройки внешнего вида протокола обмена данными.

Имя	Описание
TPathStr	Массив из MAX_PATH символов
TFmtStr	Массив из 64 символов
TmarkStr	Массив из 20 символов

Структура TComInfo

Структура используется для установки параметров последовательного порта и таймаутов.

Имя элемента	Тип	Описание
RecordSize	DWORD	Размер структуры, байт
Port	BYTE	Номер порта. 1 – COM1, 2 – COM2 и т.д.
BaudRate	DWORD	Скорость обмена
WaitPortTimeout	DWORD	Время ожидания доступности порта, мс
ReadTotalTimeout	DWORD	Время ожидания ответа считывателя, мс
ReadNextByteTimeout	DWORD	Время ожидания поступления очередного байта при условии, что обнаружено начало валидного кадра, мс.

Примечание: Рекомендуется устанавливать WaitPortTimeout в 1,5..2 раза больше, чем ReadTotalTimeout. В противном случае повышается вероятность получения кода ошибки REPLY_ERR_LOCKPORT при одновременной работе с портом 2-х и более приложений.

Структура TLogInfo

Структура используется для задания параметров журнала – протокола обмена данными со считывателем. Все строки должны заканчиваться нуль-символом. Журнал представляет собой текстовый файл со строками следующего формата:
[дата/время] [усл.обозначение запроса или ответа] [Hex-дамп данных]

Имя элемента	Тип	Описание	Значение по умолчанию
RecordSize	DWORD	Размер структуры, байт	
FileName	TPathStr	Имя файла	Prox232.log
DateFormat	TFmtStr	Формат даты/времени	dd/mm/yyyy hh:nn:ss"."zzz
ReqMark	TMarkStr	Условное обозначение запроса	"> "
AnswMark	TMarkStr	Условное обозначение ответа	"< "
Flags	WORD	Флаги	0
WrapLimit	BYTE	Ширина листа, символов	0

Формат даты/времени (используются стандартные обозначения из Windows API):
d День (1-31).
dd День (01-31).
ddd День недели коротко (Sun-Sat)
dddd День недели полностью (Sunday-Saturday)
ddddd Краткий формат даты (из Панели управления)
dddddd Длинный формат даты (из Панели управления)

m Месяц (1-12)
mm Месяц (01-12)
mmm Месяц коротко (Jan-Dec)
mmmm Месяу полностью (January-December)
yy Год – два знака (00-99).
yyyy Год – четыре знака (0000-9999).
h Часы (0-23).
hh Часы (00-23).
n Минуты (0-59).
nn Минуты (00-59).
s Секунды (0-59).
ss Секунды (00-59).
z Миллисекунды (0-999).
zzz Миллисекунды (000-999).
/ Разделитель компонентов даты (из Панели Управления)
: Разделитель компонентов времени (из Панели Управления)
'xx'/"xx" Символы, заключенные в одинарные или двойные кавычки вставляются в результирующую строку «как есть», без преобразования.

Флаги

Бит	Описание
0	Записывать запросы
1	Записывать ответы
2	Подавление эха в ответах
3	Записывать дату/время
4	Записывать маркеры запроса/ответа
5	Формат даты/времени по умолчанию (из Панели Управления)
6	Форматировать данные по 16 символов на строку, 2 группы по 8 символов
7..15	Резерв

Структура TProtocolInfo

Структура используется для управления различными параметрами протокола обмена данными при работе с различными моделями устройств.

Имя элемента	Тип	Описание
DisableAddress	BOOLEAN	Размер структуры, байт
CheckMode	INTEGER	Режим проверки контрольной суммы

Описание функций

Функция W_SetPortParams: установить параметры порта

procedure W_SetPortParams(PortCfgPtr: PComInfo);
void W_SetPortParams(TComInfo *PortCfgPtr);
Функция устанавливает параметры коммуникационного порта и переключает библиотеку в режим работы с локальным COM-портом. Должна быть вызвана первой, в противном случае функция W_IO всегда будет возвращать результат REPLY_ERR_PORTNOTSET. Параметр – указатель на структуру данных TComInfo. Возвращаемого результата не имеет.

Функция W_GetPortParams: прочитать параметры порта

procedure W_GetPortParams(PortCfgPtr: PComInfo);
void W_GetPortParams(TComInfo *PortCfgPtr);
Функция обратна функции W_SetPortParams. Заполняет информацией об установках порта структуру, переданную в качестве параметра. Параметр – указатель на структуру данных TComInfo. При вызове элемент RecordSize должен содержать фактический размер переданной структуры данных.
Возвращаемого результата не имеет.

Функция W_SetIPParams: установить параметры сетевого считывателя

procedure W_SetIPParams(IPProtocol: integer; Addr: TSockAddr);
void W_SetIPParams(int IPProtocol, sockaddr_in * Addr);
Функция устанавливает параметры считывателя, подключенного к локальной сети и переключает библиотеку в режим работы с сетью. Должна быть вызвана первой, в противном случае функция W_IO всегда будет возвращать результат REPLY_ERR_PORTNOTSET. Параметры:
IPProtocol - используемый сетевой протокол. SOCK_STREAM (1) для протокола TCP, либо SOCK_DGRAM (2) для UDP.

Addr – IP-адрес устройства и используемый порт. Используется стандартная структура, подробности в Microsoft Windows Sockets 2 Reference. Не забывайте про сетевой порядок байт.
Возвращаемого результата не имеет.

Функция W_GetIPParams: прочитать параметры сетевого считывателя

procedure W_GetIPParams(var IPProtocol: integer; var Addr: TSockAddr);
void W_GetIPParams(int * IPProtocol, sockaddr_in * Addr);
Функция обратна функции W_SetIPParams. Параметры точно такие же, что и в предыдущей функции.
Возвращаемого результата не имеет.

Функция W_IO: обмен данными со считывателем

function W_IO(Address, Cmd: char; BodyLen: integer; const CmdBody; var ReplyLen: integer; var ReplyBuffer): char;
char W_IO(char Address, char Cmd, int BodyLen, void *CmdBody, int *ReplyLen, void *ReplyBuffer);
Параметры:
Address адрес считывателя (0..3F)
Cmd команда
BodyLen длина дополнительных данных для команды
CmdBody буфер, содержащий дополнительные данные
ReplyLen длина данных, полученных от считывателя
ReplyBuffer буфер, в который будут помещены полученные данные.
При вызове функции в ReplyLen должен быть записан размер области памяти, отведенной под ReplyBuffer.
На выходе в ReplyLen будет записана длина фактически полученных данных.
Возвращаемое значение – код возврата, см. раздел «Константы» REPLY_XXXXX.

Функция W_LastRequest: получить последний запрос

procedure W_LastRequest(var ReqLen: integer; var Request);
void W_LastRequest(int *ReqLen, void *Request);
Копирует данные посланные в устройство при последнем вызове функции W_IO в буфер, заданный параметром Request. Используется для протоколирования обмена с устройством. Данные возвращаются в том же самом виде, в каком они посланы в устройство. Копируется максимум ReqLen байт, после вызова функции в ReqLen записывается полная длина данных, посланных устройству.
Внимание: с полученным буфером нельзя работать, как со строкой, во-первых, потому, что заключительный нуль-сиимвол не добавляется и, во-вторых, данные сами по себе могут содержать нуль-символы.

Функция W_LastAnswer: получить последний ответ

procedure W_LastAnswer(var AnswerLen: integer; var Answer);
void W_LastAnswer(int *AnswerLen, void *Answer);
Копирует данные, полученные от устройства, при последнем вызове функции W_IO в буфер, заданный параметром Answer. Используется для протоколирования обмена с устройством. Данные возвращаются в том же самом виде, в каком они получены от устройства, байтстаффинг не производится. Копируется максимум AnswerLen байт, после вызова функции в AnswerLen записывается длина данных, полученных от устройства. Если ответ от устройства не получен, то будет возвращено 0 байт.
Внимание: с полученным буфером нельзя работать, как со строкой, во-первых, потому, что заключительный нуль-сиимвол не добавляется и, во-вторых, данные сами по себе могут содержать нуль-символы.

Функция W_SetLogParams: установить параметры журнала

procedure W_SetLogParams(LogCfgPtr: PLogInfo);
void W_SetLogParams(TLogInfo *LogCfgPtr);
Функция устанавливает параметры журнала – протокола обмена со считывателем. Параметр – указатель на структуру данных TLogInfo. Вызов функции не начинает запись журнала, а только устанавливает его параметры, для начала записи необходимо вызвать функцию W_EnableLog.
Возвращаемого результата не имеет.

Функция W_GetLogParams: прочитать параметры журнала

procedure W_GetLogParams(LogCfgPtr: PLogInfo);
void W_GetLogParams(TLogInfo *LogCfgPtr);
Функция обратна функции W_SetLogParams. Заполняет информацией о параметрах записи журнала структуру, переданную в качестве параметра. Параметр – указатель на структуру данных TLogInfo. При вызове элемент RecordSize должен содержать фактический размер переданной структуры данных.
Возвращаемого результата не имеет.

Функция W_EnableLog: включить протоколирование обмена

function W_EnableLog: boolean;
BOOL W_EnableLog(void)

Включает запись протокола обмена со считывателем. В случае успеха возвращается значение "истина". Файл протокола не затирается, производится добавление в конец файла журнала.

Функция W_DisableLog: выключить протоколирование обмена

procedure W_DisableLog;
void W_DisableLog(void);
Выключает запись протокола обмена со считывателем.

Функция W_SetProtocolParams: установка параметров протокола обмена

procedure W_SetProtocolParams(ProtocolInfoPtr: PProtocolInfo);
void W_SetProtocolParams(ProtocolInfoPtr *PProtocolInfo);
Функция устанавливает параметры протокола обмена со считывателем. Параметр – указатель на структуру данных TProtocolInfo. Если элемент CheckMode переданной структуры содержит недопустимое значение, то он игнорируется.
Возвращаемого результата не имеет.

Функция W_GetProtocolParams:чтение параметров протокола обмена

procedure W_GetProtocolParams(ProtocolInfoPtr: PProtocolInfo);
void W_GetProtocolParams(ProtocolInfoPtr *PProtocolInfo);
Функция обратна функции W_GetProtocolParams. Заполняет информацией о параметрах протокола структуру, переданную в качестве параметра. Параметр – указатель на структуру данных TProtocolInfo.
Возвращаемого результата не имеет.

Пример использования: чтение журнала событий

```
procedure ReadEventLog;
var
Ret: char;
RL: integer;
Event: TEvent;
begin
// до вызова этой функции должна быть вызвана функция W_SetPortParams
// читаем считыватель с адресом 3F
repeat
Ret := W_IO(#$3F, #$10, 0, '', RL, Event);
if Ret = REPLY_OK then begin
// Успешно прочитали событие, обрабатываем его…
Ret := W_IO(#$3F, #$11, 0, '', RL, Event);
if Ret <> REPLY_ACK then begin
// Ошибка удаления последнего события – обрабатываем…
end;
end else begin
// Ошибка чтения события – обрабатываем…
end;
until Ret <> REPLY_ACK; // выход из цикла чтения при ошибке
end;
```