# Динамическая библиотека PR-x08.dll

Версия 1.1

ноябрь 2007 г.

# Оглавление

# История документа

| Версия | Дата | Изменения |
|---|---|---|
| 1.0 | 28.11.2007 | Первая редакция документа. Версия библиотеки 1.0.1.x |
| | | |
| | | |
| | | |

# Введение

Данный документ описывает функционал динамической библиотеки PRx08.dll, предназначенной для работы с настольными считывателями серии PR-x08, имеющими USB интерфейс. Библиотека PRx08.dll предназначена для замены более старой PR08.dll, имеющей ограниченный функционал по сравнению с новой библиотекой.

## Что нового в этом документе

Новая библиотека отличается от ранее использовавшейся PR08.dll следующими дополнительными функциями:

- Возможность одновременной работы с произвольным числом считывателей, подключенных к ПК
- Возможность внешнего управления индикацией считывателей
- Дополнительными сервисными функциями (например, получение версии считыателя)

## Назначение

Динамическая библиотека PRx08.dll l предназначена для поддержки настольных считывателей PR-P08, PR-A08, PR-T08, PR-H08 и инкапсулирует протокол обмена со считывателями нижнего уровня, предоставляя программисту более простой прикладной интерфейс (API).

## Демонстрационный пример

Для упрощения освоения функционала библиотеки в комплекте поставляется демонстрационный пример, иллюстрирующий использование большинства функций библиотеки.

Пример написан с использованием среды программирования Delphi 5 (Object Pascal), однако легко может быть переведен на другие языки программирования (например, C или C++).

## Соглашения о вызовах

Библиотека скомпилирована с использованием соглашения о вызовах типа **stdcall**, что обеспечивает совместимость с большинством сред разработки программного обеспечения.

В качестве параметров функций используются типы short, word (unsigned short), указатели на массивы (*char или PByteArray являются синонимами).

Структуры, используемые совместно библиотекой и приложением, должны быть упакованы.

## Совместимость

Библиотека совместима со считывателями, имеющими типа PR-P08, PR-A08, PR-H08, PR-T08, имеющими интерфейс USB. Поддержка более старых считывателей с интерфейсом RS-232 не обеспечивается.

## Типы карт

Proximity считыватель **PR-A08** используется с картами:

- StandProx (Ангстрем)
- SlimProx (EM Marin и аналогичные тонкие карты под прямую печать)
- MiniTag

Proximity считыватель **PR-H08** используется с картами и брелками HID Corporation:

- ProxCard II
- PhotoProx
- ISOProx
- TagProx

Proximity считыватель **PR-P08** предназначен для работы с интерактивными (read/write) картами на частоте 13,56 МГц и может полностью поддерживать карты следующих форматов:

- ISO 14443A
- Mifare Standard 1K и 4K
- Mifare UltraLight
- Mifare Prox

Считыватель **PR-T08** используется с ключами типа Touch Memory DS1990A.

## Поддержка

По всем проблемам, связанным с использованием данной библиотеки, следует обращаться в службу технической поддержки по адресу:

support@parsec.ru

# Перечень функций библиотеки

В таблице ниже приведен перечень всех функций библиотеки PRx08.dll. Подробное описание функций дано в последующих разделах документа.

| Имя функции | Назначение |
|---|---|
| Enumerate() | Обновление внутреннего списка считывателей, подключенных к ПК |
| GetFirstReader() | Получение описателя первого считывателя |
| GetNextReader() | Получение описателей последующих считывателей |
| OpenReader() | Открытие интерфейса считывателя для операций с ним |
| CloseReader() | Закрытие интерфейса считывателя |
| GetReaderInfo() | Получение информации о считывателе |
| SetBeepBlinkMode() | Установка режима индикации на карту |
| BeepBlink() | Программное включение индикации |
| | |
| ReadCardNumberRaw() | Получение полного кода карты |
| ReadCardNumber() | Получение 4-х байтового кода карты |
| SetWiegand26() | Включение режима wiegand 26 bit (3-х байтовый код) |

# Коды ошибок

Все функции библиотеки возвращают результат операции типа short. Нулевой результат соответствует успешному выполнению функции, результат, отличный от нуля характеризует ошибку, возникшую во время выполнения функции.

Коды ошибок библиотеки приведены в таблице 1 ниже. Вместе с демонстрационным примером поставляется файл ErrCodes.pas, содержащий декларации констант кодов ошибок библиотеки.

Таблица 1

| Код | Мнемоника | Описание |
|---|---|---|
| 0 | RES_OK | Функция завершена успешно |
| 1 | RES_NO_CARD_ERROR | Нет карты в поле считывателя |
| 2 | RES_ERR_INVALID_HANDLE | Неправильный дескриптор устройства |
| 3 | RES_EMPTY | Результат выполнения операции пуст |
| 4 | RES_WRONG_PARAMS | Переданные параметры некорректны |
| 5 | RES_CRC_ERROR | Ошибка контрольной суммы |
| 6 | RES_INVALID_DATA | «Битые» данные |
| 7 | RES_COMM_TIMEOUT | Таймаут при обмене со считывателем |
| 8 | RES_READ_ERROR | Ошибка при чтении и порта |
| 9 | RES_WRITE_ERROR | Ошибка при записи в порт |
| 10 | RES_OVERFLOW_ERROR | Ошибка переполнения буфера |
| 11 | RES_JUST_OPENED | Порт уже открыт |
| 12 | RES_JUST_CLOSED | Порт уже закрыт |
| 13 | RES_NO_IMPLEMENTED | Функция не реализована |
| 14 | | |

Если функция завершена с ошибкой (код результата отличен от нуля), то возвращаемые функцией значения не содержат корректной информации.

При обмене со считывателем могут по техническим причинам возникать различные ошибки связи, к которым относятся, в частности:

RES_ERR_INVALID_HANDLE, RES_CRC_ERROR, RES_INVALID_DATA, RES_COMM_TIMEOUT, RES_READ_ERROR, RES_WRITE_ERROR.

При описании функций эти ошибки как возможный возвращаемый результат каждый раз отдельно не перечисляются.

Ошибка типа RES_OVERFLOW_ERROR может возникать как при внутренних ошибках, вызывающих переполнение выделенных буферов, так и при неправильном назначении пользовательских буферов, передаваемых функциям в случае, если информация о длине в библиотеку передается.

# Функции библиотеки

## Работа с интерфейсом

Прежде, чем связываться со считывателем, необходимо открыть USB интерфейс к которому подключен считыватель.

Общая последовательность доступа к интерфейсу следующая:

- Необходимо определить все доступные интерфейсы (функция Enumerate())
- Получить описания (номер и текстовое описание) всех доступных интерфейсов – функции GetFirstReader () и GetNextReader()
- Открыть выбранный интерфейс считывателя по его номеру функцией OpenReader()

По окончании работы программы перед ее завершением следует закрыть ранее открытый интерфейс с помощью функции CloseReader().

### Создание списка интерфейсов Enumerate

#### *Назначение*

Функция проверяет наличие подключенных на текущий момент считывателей и создает их внутренний список для дальнейшего использования.

#### *Описание*

```
function Enumerate()
```

#### *Возвращаемые результаты*

В качестве результата функция возвращает число обнаруженных (доступных) интерфейсов (считывателей).

***Примечание: это единственная функция библиотеки, которая возвращает не результат выполнения, а число доступных для использования интерфейсов.***

### Получение первого считывателя GetFirstReader

#### *Назначение*

Функция возвращает описание первого найденного считывателя из подключенных к компьютеру. Если ни один считыватель не был обнаружен при вызове функции Enumerate(), но вызывалась данная функция, то возвращается код ошибки.

Функция позиционирует внутренний указатель на первый считыватель во внутреннем списке.

#### *Описание*

```
function GetFirstReader(desc: pbytearray; maxlen: short)
```

### *Входные параметры*

Параметр maxlen задает максимальную длину описателя, которую может вернуть функция (ограничивается вызывающей программой при выделении памяти на описатель). Рекомендуемое значение – 80 символов.

### *Возвращаемые значения*

Функция возвращает описание считывателя и его серийный (заводской) номер в виде нуль – терминированной строки. Примерный вид возвращаемого описателя:

```
'Parsec Desktop Reader PR-P08 (873-00-0287)'
```

Если ни одного считывателя обнаружено не было или предварительно не вызывалась функция Enumerate(), то возвращается код ошибки RES_ERR_INVALID_HANDLE.

Считыватель, описание которого получено при вызове данной функции, всегда имеет номер 0 (номер важен для всех последующих операций со считывателем).

## Получение следующего считывателя GetNextReader

### *Назначение*

Функция возвращает описание очередного считывателя из внутреннего списка библиотеки. Если список исчерпан, то возвращается код ошибки.

Перед первым вызовом функции должны предварительно вызываться функции Enumerate() и GetFirstReader().

### *Описание*

```
function GetNextReader(desc: pbytearray; maxlen: short)
```

### *Входные параметры*

Параметр maxlen задает максимальную длину описателя, которую может вернуть функция (ограничивается вызывающей программой при выделении памяти на описатель). Рекомендуемое значение – 80 символов.

### *Возвращаемые значения*

Функция возвращает описание считывателя и его серийный (заводской) номер в виде нуль – терминированной строки. Примерный вид возвращаемого описателя:

```
'Parsec Desktop Reader PR-P08 (873-00-0287)'
```

Если список считывателей исчерпан, то возвращается код ошибки RES_ERR_INVALID_HANDLE.

Считыватели, описания которых получены при вызове данной функции, имеют последовательную румерацию от 1 до N (номер важен для всех последующих операций со считывателем).

## Открытие считывателя OpenReader

### *Назначение*

Функция открывает интерфейс считывателя для осуществления операций с ним.

### *Описание*

```
function OpenReader(num: short)
```

### *Входные параметры*

Функция имеет один входной параметр – номер считывателя в списке. Нумерация производится с нуля.

Вызывающая программа при последовательном вызове функций GetFirstReader() и GetNextReader() должна корректно поддерживать список считывателей для последующей работы с ними.

### *Возвращаемые значения*

Функция возвращает только код результата операции. Возможные коды ошибок:

- RES_JUST_OPENED – считыватель уже открыт
- INVALID_HANDLE – ошибка открытия (возможно, считыватель занят другим приложением)
- RES_EMPTY – считывателя с таким номером в списке нет

## Закрытие считывателя CloseReader

### *Назначение*

Функция закрывает ранее открытый интерфейс считывателя, освобождая все связанные с этим ресурсы системы. Должна вызываться перед окончанием работы приложения для каждого из открытых считывателей.

### *Описание*

```
function CloseReader(num: short)
```

### *Входные параметры*

Функция имеет один входной параметр – номер считывателя в списке. Нумерация производится с нуля.

### *Возвращаемые значения*

При отсутствии считывателя с таким номером во внутреннем списке библиотеки возвращается код ошибки RES_EMPTY.

## Функции – утилиты

Все функции, описываемые в данном и последующих разделах, в качестве первого параметра имеют номер считывателя, получаемый при инициализации интерфейса.

### Получение информации о версии считывателя GetReaderInfo

#### *Назначение*

Функция позволяет получить информацию о конкретном считывателе.

#### *Описание*

```
function GetReaderInfo(num: short; info: PByteArray)
```

#### *Входные параметры*

Функция имеет один входной параметр – номер считывателя в списке.

#### *Возвращаемые результаты*

Функция возвращает 16 – байтовый массив с данными считывателя. Основная информация о трактовке получаемой информации приведена в Приложении 2.

### Режим индикации SetBeepBlinkMode

#### *Назначение*

Функция определяет режим работы встроенного бипера и светодиода считывателя.

#### *Описание*

```
function SetBeepBlinkMode(num: short; useblink: short; usebeep: short)
```

#### *Входные параметры*

Кроме номера считывателя, функция получает два параметра, управляющих работой светодиода (useblink) и бипера (usebeep) при чтении карты. Нулевое значение выключает соответствующий индикатор, отличное от нуля значение (типичное – единица) наоборот, включает.

#### *Возвращаемые результаты*

При наличии связи со считывателем функция всегда возвращает результат RES_OK.

### Принудительная индикация BeepBlink

#### *Назначение*

Функция позволяет программно включить кратковременную индикацию заданного считывателя.

#### *Описание*

```
function BeepBlink(num: short)
```

### *Входные параметры*

Функция имеет один входной параметр – номер считывателя в списке.

### *Возвращаемые результаты*

Функция возвращает только код ошибки.

## Функции для работы с картами

Описанные в данном разделе функции предназначены для чтения кодов карт со считывателей, подключенных к ПК.

Следует отметить, что считыватель выдает код карты в ответ на вызов соответствующей функции до тех пор, пока карта находится в поле чтения. Определение, является ли карта вновь поднесенной или находится на считывателе давно, возлагается на пользовательскую программу (то есть пользовательская программа должна помнить номер прочитанной в предыдущих циклах карты и сравнивать номер вновь прочитанной карты с ранее запомненной). При этом рекомендуется единичное отсутствие данной карты на считывателе не считать за ее исчезновение, а опросить считыватель как минимум 2 – 3 раза, после чего принимать решение о том, что карта из поля считывателя исчезла.

При работе со считывателем PR-P08 для того, чтобы карта читалась при каждом обращении, между обращениями несущая выключается (в противном случае код карты читался бы только через раз в силу внутренних механизмов карт стандарта ISO-14443).

### Получение полного номера карты ReadCardNumberRaw

#### *Назначение*

Функция позволяет получить полный код карты в соответствии с ее типом.

#### *Описание*

```
function ReadCardNumberRaw(num: short; var ctype: short; cserial:
PByteArray; var clen: short)
```

#### *Входные параметры*

Функция кроме номера считывателя в списке получает максимальную длину буфера для полного кода карты clen. Рекомендуемая минимальная длина выделяемого для обмена буфера равна 10 байтам.

#### *Возвращаемые результаты*

При успешном выполнении операции функции возвращает тип карты, полный код карты и реальную длину кода карты. Если карты в поле считывателя нет, то возвращается код ошибки RES_NO_CARD_ERROR. При ошибках обмена со считывателем возвращаются другие коды ошибок, описанные ранее в данном документе.

Описание форматов полных номеров карт для различных считывателей приведено в Приложении 1 к данному документу.

### Получение номера карты ReadCardNumber

#### *Назначение*

Функция позволяет получить код карты в виде четырехбайтового слова (DWORD).

#### *Описание*

```
function ReadCardNumber(num: short; var cod: dword)
```

### *Входные параметры*

Функция кроме номера считывателя в списке получает указатель на двойное слово, в которое следует поместить код карты.

### *Возвращаемые результаты*

При успешном выполнении операции функции возвращает код карты в формате, совместимом с представлением карты в системе ***ParsecNET***. Если карты в поле считывателя нет, то возвращается код ошибки RES_NO_CARD_ERROR. При ошибках обмена со считывателем возвращаются другие коды ошибок, описанные ранее в данном документе.

При включенном режиме wiegand 26 bit код карты урезается до 3-х байтового значения (старший байт равен нулю, а для карт типа HID представление совместимо с принятым в системе ***ParsecNET***).

## Формат кода для функции ReadCardNumber - SetWiegand26

### *Назначение*

Функция позволяет переключить представление кода карты, возвращаемого функцией ReadCardNumber()

### *Описание*

```
function SetWiegand26(on: short)
```

### *Входные параметры*

Функция получает один параметр – on. При нулевом значении формат карты, возвращаемый функцией ReadCardNumber() будет 4-х байтовым, при отличном от нуля значении параметра on – трехбайтовым.

Функция переключает режим представления кода не для конкретного считывателя, а для всех считывателей, обслуживаемых в данный момент библиотекой.

При инициализации библиотеки значение on равно нулю.

### *Возвращаемые результаты*

Функция всегда возвращает код RES_OK.

# Приложение 1

## Форматы кодов карт

Ниже описаны форматы кодов карт, возвращаемые функцией ReadCardNumberRaw() для каждого из типов карт.

В качестве первого возвращаемого параметра передается код типа карты, который кодируется следующим образом:

| Мнемоника | Значение | Тип карты |
|---|---|---|
| CARD_UNKNOWN | 0 | Тип карты неизвестен |
| CARD_ISO14443A | 1 | Карты ISO-14443-A, в том числе Mifare |
| CARD_EM | 2 | Карта типа EM Marin (Ангстрем) |
| CARD_HID | 3 | Карта типа HID |
| CARD_DALLS | 4 | Код ключа I-Button (Touch Memory) |
| CARD_CHEKPOINT | 5 | В настоящее время не поддерживается |

### Карты типа Mifare (ISO-14443-A)

Для данного типа карт серийный номер может, в соответствии со стандартом, содержать 4, 7 или 10 байт. Код передается старшим байтом вперед. Параметр длины при возврате из функции принимает соответственно значение 4, 7 или 10.

### Карты типа EM

Карты данного типа всегда имеют серийный номер длиной в пять байтов. Код передается старшим байтом вперед. Параметр длины при возврате из функции принимает соответственно значение 5. Старший (первый) байт иногда именуют групповым идентификатором или facility – кодом (данное понятие не совсем совпадает с классическим понятием falsity в форматах wiegand для систем доступа).

### Карты типа HID

Для данной карты полная внутренняя длина кода всегда равна 45 битам, именно в этом виде код возвращается функцией ReadCardNumberRaw() в виде массива длиной 6 байт. При этом выравнивание производится по левой границе, то есть младшие три бита последнего байта являются незначащими и всегда равны нулю. Параметр длины при возврате из функции принимает соответственно значение 6.

### Карты типа DALLASS

Код ключей Touch Memory всегда имеет длину 6 байтов. Код передается старшим байтом вперед. Параметр длины при возврате из функции принимает соответственно значение 6.

# Приложение 2

## Формат структуры, возвращаемой функцией GetReaderInfo

Данная функция возвращает блок параметров, содержащий полную информацию о считывателе. Размер структуры равен шестнадцати байтам. Байты ответа имеют следующее смысловое значение (байты пронумерованы по порядку индексов в возвращаемом массиве):

| Байт | Описание | Примечание |
|---|---|---|
| 0 | 5 байт ProductInfo | Только для PR-P08 |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | 4 байта серийного номера микросхемы считывателя | Только для PR-P08 |
| 6 | | |
| 7 | | |
| 8 | | |
| 9 | Версия 1 в формате ASCII | |
| 10 | Версия 2 в формате ASCII | |
| 11 | Поддерживаемые интерфейсы | Всегда USB |
| 12 | Тип процессора | |
| 13 | Режим работы считывателя | |
| 14 | Состояние несущей | |
| 15 | Адрес считывателя | Для USB не актуально |

Первые 9 байт структуры (с нулевого по 8-й) актуальны только для считывателей PR-P08, которые передают информацию о микросхеме считывателя, имеющей прошитую при производстве микросхемы соответствующую информацию.

Версия (байты 9 и 10 в структуре) представляется в формате ASCII, например, для версии 3.4 значения байтов будут равны соответственно 33h и 34h.

Поддерживаемые интерфейсы (байт номер 11 в структуре) задаются соответствующими битами байта как показано в таблице ниже:

| Бит | Значение |
|---|---|
| 7 | RFU |
| 6 | Ethernet supported |
| 5 | USB supported |
| 4 | Parsec V2 supported |
| 3 | Parsec V1 supported |
| 2 | Wiegand supported |
| 1 | RS-485 supported |
| 0 | RS-232 supported |

Байт типа процессора (байт 12 в структуре) определяет, на базе какого процессора сделан считыватель. На текущий момент определены следующие типы микропроцессоров:

| Бит | Значение |
|---|---|
| 7 – 3 | RFU |
| 2 – 0 | 000 – 001 – RFU<br>010 – Atmega16<br>011 – Atmega32<br>100 – Atmega64<br>101 – Atmega128<br>110 – Microchip<br>111 и далее – RFU |

Байт режима работы (байт номер 13 в структуре) трактуется следующим образом:

| Бит | Значение |
|---|---|
| 7 – 3 | RFU |
| 2 – 0 | 000 – ISO 14443A<br>001 – ISO 14443B<br>010 – RFU<br>011 – RFU<br>100 – Mifare® <br>101 – EM Marin (Ангстрем)<br>110 – HID<br>111 – Dallass |

Байт состояния несущей (байт номер 14 в структуре) содержит информацию о состоянии передатчика считывателя и трактуется следующим образом:

| Бит | Значение |
|---|---|
| 7 – 2 | RFU |
| 1 – 0 | 00 – RF off<br>01 – RF on, low power<br>10 – RF on, high power<br>11 – reserved |

Байт адреса считывателя (байт 15 в структуре) для считывателей с интерфейсом USB смысла не имеет – он используется для считывателей с интерфейсом RS-485.

## Для заметок