Введение

Данный пакет SDK предназначен для облегчения работы с usb- и ip- устройствами.. Используется в Sdk Guard и в Sdk Readers.

Комплект SDK:

1. Библиотеки в нескольких вариантах сборки:
    1. ZPort.dll – 32-битная библиотека;
    2. x64\ZPort.dll – 64-битная библиотека;
    3. Log\ZPort.dll – 32-битная библиотека, в которую добавлены функции лога;
    4. Log\x64\ZPort.dll – 64-битная библиотека, в которую добавлены функции лога;
2. Демонстрационная программа - Demo.exe;
3. Данный файл описания SDK – Help\ZPort_rus.chm
4. Утилита для работы с конвертером Z-397, Z-397 Guard по сети TCP/IP – Retr\ZRetr.exe;
5. Утилита для простотра лога, создаваемого библитеками из папки "Log" - Log\Log.exe

Основные возможности

Детектор устройств

- Поиск usb- и ip- устройств;
- Уведомление о подключении/отключении устройств от ПК;
- Получение информации о устройстве (модель, с/н, версия прошивки, и т.п.).

Работа с устройством

- Чтение/запись данных в порт устройства;
- Уведомление о появлении новых данных в порте;
- Установка скорости для порта;
- Автоматическое восстановление связи при переподключении устройства.

Особенности ZPort API:

- Текстовые строки передаются в UNICODE (WideChar);
- Для уведомлений используется либо Event (событие - объект синхронизации Windows), либо сообщение для отправки определенному окну с помощью PostMessage.

Требования к системе

ОС: Windows® XP/Vista/Seven

Прошивки устройств и драйвера: При обнаружении неправильной работы SDK с usb устройствами рекомендуется обновить их драйвера. Самые последние версии драйверов доступны на сайте www.ironlogic.ru.

Что нового в ZPort API

v1.18.9 (30.01.2016)

! исправлено записание при завершении работы с конвертером в режиме "Клиент"
* Delphi, VC++, BC++: в заголовочные файлы добавлена возможность динамической загрузки dll

v1.18.8 (25.01.2016)

! при открытии com-порта с флагом ZP_POF_NO_WAIT_CONNECT функция ZP_Port_Write не работала до момента подключения (из-за этого некорректно работало сканирование устройств детектором и ZP_SearchDevices)
! изменено действие флага ZP_POF_NO_DETECT_USB для функции ZP_Port_Open, если флаг снят: раньше объект порта ждал уведомления об подключении/отключении порта usb-устройств от системы (RegisterDeviceNotification), теперь ждет от объекта детектора

v1.18.7 (22.01.2016)

! при инициализации обоих Sdk (Readers & Guard) могла возникать ошибка "класс уже существует"

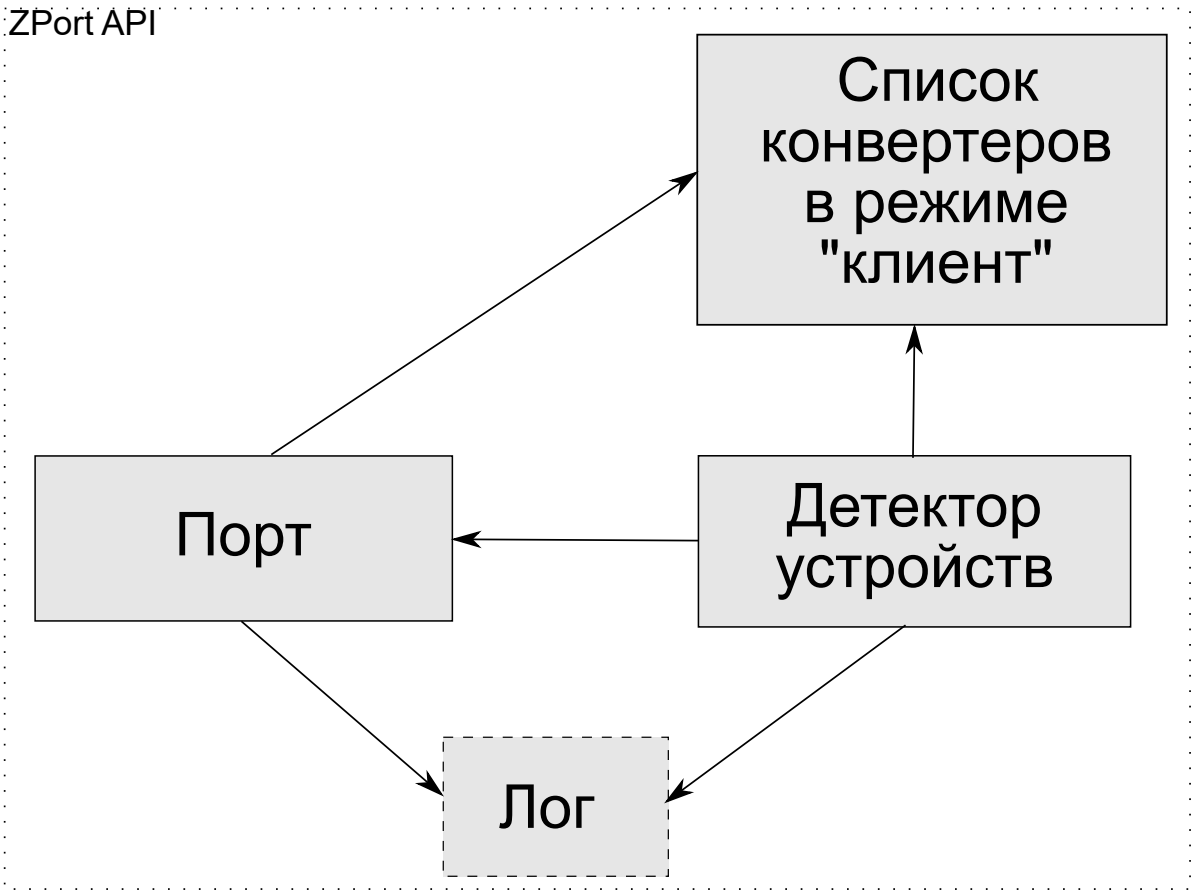
v1.18.4 (15.05.2015)

! детектор устройств уведомлял о подключении/отключении usb-устройств даже если соотвествующие уведомления не были настроены функцией ZP_DD_SetNotification

v1.18.3 (14.05.2015)

! исправлена ошибка закрытия порта типа `ZP_PORT_IP` после разрыва соединения

Объектная модель SDK



Порт

Объект "Порт" предназначен для обмена данными с устройством.

Имя функции	Кратное описание
ZP_Port_Open	Открывает порт.
ZP_Port_SetBaudAndEvChar	Устанавливает скорость порта и сигнальный символ.
ZP_Port_GetBaudAndEvChar	Возвращает скорость порта и сигнальный символ.
ZP_Port_GetConnectionStatus	Возвращает текущее состояние подключения к устройству.
ZP_Port_SetNotification	Настраивает уведомления порта.
ZP_Port_EnumMessages	Возвращает уведомления порта.
ZP_Port_Clear	Очищает входящий и исходящий буфер данных порта.
ZP_Port_Write	Отправляет данные устройству.
ZP_Port_Read	Возвращает данные, полученные от устройства.
ZP_Port_GetInCount	Возвращает количество байт, полученных от устройства.
ZP_Port_SetDtr	Настраивает Data Terminal Ready (DTR) сигнал управления.
ZP_Port_SetRts	Настраивает Request To Send (RTS) сигнал

Имя функции	Кратное описание
	управления.

Детектор устройств

Объект "Детектор устройств" предназначен для уведомления об подключении / отключении устройств.

Имя функции	Кратное описание
ZP_DD_SetNotification	Настраивает уведомления о подключении/отключении устройств.
ZP_DD_GetNextMessage	Возвращает следующее сообщение детектора, созданного с помощью функции ZP_DD_SetNotification.
ZP_DD_SetGlobalSettings	Устанавливает новые настройки детектора, созданного функцией ZP_DD_SetNotification.
ZP_DD_GetGlobalSettings	Возвращает настройки детектора.
ZP_DD_Refresh	Вызывает обновление списка устройств детектора.
Специальные функции для настройки уведомлений в службе Windows	
ZP_SetServiceCtrlHandle	Устанавливает дескриптор службы Windows для настройки уведомлений (только для приложений, которые являются службой Windows).
ZP_DeviceEventNotify	Обработчик уведомления SERVICE_CONTROL_DEVICEEVENT для приложений-служб Windows.

Лог

Объект "Лог" предназначен для отладки работы с SDK.

Имя функции	Кратное описание
ZP_SetLog	Устанавливает параметры для отладки (ip-адрес ZLog.exe и путь к лог-файлу).
ZP_GetLog	Возвращает параметры для отладки.
ZP_AddLog	Записывает новое лог-сообщение.

Функции ZPort API

Имя функции	Кратное описание
Инициализация библиотеки	
ZP_GetVersion	Возвращает номер текущей версии библиотеки.
ZP_Initialize	Инициализирует библиотеку.
ZP_Finalyze	Завершает работу библиотеки.
ZP_CloseHandle	Закрывает дескриптор.
Поиск устройств	
ZP_GetPortInfoList	Создает список последовательных портов. Для чтения элементов списка предназначена функция ZP_GetPortInfo.
ZP_GetPortInfo	Возвращает информацию о порте из списка, созданного функцией ZP_GetPortInfoList.
ZP_SearchDevices	Инициализирует поиск устройств. Для продолжения поиска предназначена функция ZP_FindNextDevice.
ZP_FindNextDevice	Ищет следующее устройтство.
ZP_DD_SetNotification	Настраивает уведомления о подключении/отключении устройств.
ZP_DD_GetNextMessage	Возвращает следующее сообщение детектора, созданного с помощью функции ZP_DD_SetNotification.
ZP_DD_SetGlobalSettings	Устанавливает новые настройки детектора, созданного функцией ZP_DD_SetNotification.
ZP_DD_GetGlobalSettings	Возвращает настройки детектора.

Имя функции	Кратное описание
ZP_DD_Refresh	Вызывает обновление списка устройств детектора.
ZP_SetServiceCtrlHandle	Устанавливает дескриптор службы Windows для настройки уведомлений (только для приложений, которые являются службой Windows).
ZP_DeviceEventNotify	Обработчик уведомления SERVICE_CONTROL_DEVICEEVENT для приложений-служб Windows.
ZP_RegisterPortRequest	Настраивает запрос для порта.
ZP_RegisterUdpRequest	Настраивает Udp-запрос для ip-устройства.
Работа с устройством	
ZP_Port_Open	Открывает порт.
ZP_Port_SetBaudAndEvChar	Устанавливает скорость порта и сигнальный символ.
ZP_Port_GetBaudAndEvChar	Возвращает скорость порта и сигнальный символ.
ZP_Port_GetConnectionStatus	Возвращает текущее состояние подключения к устройству.
ZP_Port_SetNotification	Настраивает уведомления порта.
ZP_Port_EnumMessages	Возвращает уведомления порта.
ZP_Port_Clear	Очищает входящий и исходящий буфер данных порта.
ZP_Port_Write	Отправляет данные устройству.
ZP_Port_Read	Возвращает данные, полученные от устройства.
ZP_Port_GetInCount	Возвращает количество байт, полученных от устройства.
ZP_Port_SetDtr	Настраивает Data Terminal Ready (DTR) сигнал управления.
ZP_Port_SetRts	Настраивает Request To Send (RTS) сигнал управления.

Имя функции	Кратное описание
Функции для отладки	
ZP_SetLog	Устанавливает параметры для отладки (ip-адрес ZLog.exe и путь к лог-файлу).
ZP_GetLog	Возвращает параметры для отладки.
ZP_AddLog	Записывает новое лог-сообщение.

ZP_GetVersion

Возвращает номер текущей версии библиотеки ZPort.dll.

C++

```
DWORD ZP_GetVersion();
```

Delphi

```
function ZP_GetVersion(): Cardinal;
```

Параметры

Эта функция не имеет параметров.

Возвращаемое значение

Если функция выполнена успешно, то возвращает DWORD значение, включающее в себя основной (Major) и дополнительный (Minor) номер версии SDK в младшем слове.

ZP_Initialize

Инициализирует библиотеку.

C++

```
HRESULT ZP_Initialize(
        UINT nFlags
        );
```

Delphi

```
function ZP_Initialize(
    AFlags: Cardinal
    ): HRESULT;
```

Параметры

Flags

[in] Флаги.

Флаг	Описание
ZP_IF_NO_MSG_LOOP	Приложение не имеет очередь сообщений. Вместо получения уведомлений о подключении / отключении USB-устройства будет использоваться периодическая проверка подключенных устройств в потоке (thread).
ZP_IF_LOG	Только для версии "Log". Записывать лог ZPort.dll. По умолчанию данные передаются в программу ZLog.exe, для записи в лог-файл используйте функцию ZP_SetLog.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Используйте функцию `ZP_Finalyze`, чтобы правильно завершить работу потоков (thread) библиотеки до ее выгрузке с помощью функции `FreeLibrary`.

ZP_Finalize

Завершает работу библиотеки. При этом автоматически закрываются все дескрипторы, открытые с помощью функций ZPort API.

C++

```
HRESULT ZP_Finalize();
```

Delphi

```
function ZP_Finalize(): HRESULT;
```

Параметры

Эта функция не имеет параметров.

Возвращаемое значение

Функция всегда возвращает `S_OK`.

ZP_CloseHandle

Закрывает дескриптор, который был открыт функцией, возвращающей дескриптор (ZP_GetPortInfoList, ZP_SearchDevices, ZP_DD_SetNotification и другие).

C++

```
HRESULT ZP_CloseHandle(
        HANDLE hHandle
        );
```

Delphi

```
function ZP_CloseHandle(
    AHandle: THandle
    ): HRESULT;
```

Параметры

Handle

[in, out] Дескриптор.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_GetPortInfoList

Создает список последовательных портов устройств. Для доступа к элементам списка используйте функцию ZP_GetPortInfo.

C++

```
HRESULT ZP_GetPortInfoList(
        PHANDLE pHandle,
        PINT pCount
        );
```

Delphi

```
function ZP_GetPortInfoList(
    var VHandle: THandle;
    var VCount: Integer
    ): HRESULT;
```

Параметры

Handle
[out] Дескриптор списка.
Count
[out] Количество элементов в списке.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_GetPortInfo

Возвращает информацию о порте из списка портов.

C++

```
HRESULT ZP_GetPortInfo(
        HANDLE hHandle,
        INT nIdx,
        PZP_PORT_INFO pInfo
        );
```

Delphi

```
function ZP_GetPortInfo(
    AHandle: THandle;
    AIdx: Integer;
    var VInfo: TZP_PORT_INFO
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор списка, полученный с помощью функции ZP_GetPortInfoList.

Idx
: [in] Позиция в списке.

Info
: [out] Информация о порте.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_SearchDevices

Ищет устройства, опрашивая порты. В отличае от функции ZP_GetPortInfoList возвращает расширенную информацию о устройстве (модель, с/н, версия прошивки).

C++

```
HRESULT ZP_SearchDevices(
        PHANDLE pHandle,
        PZP_SEARCH_PARAMS pParams
        );
```

Delphi

```
function ZP_SearchDevices(
    var VHandle: THandle;
    var AParams: TZP_SEARCH_PARAMS
    ): HRESULT;
```

Параметры

Handle
[out] Дескриптор поиска.
Params
[in] Параметры поиска.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Функция `ZP_SearchDevices` только инициализирует поиск устройств, сам поиск осуществляется функцией `ZP_FindNextDevice`.

ZP_FindNextDevice

Ищет следующее устройство. Для завершения поиска устройств используйте функцию ZP_CloseHandle.

C++

```
HRESULT ZP_FindNextDevice(
        HANDLE hHandle,
        PZP_DEVICE_INFO pInfo,
        PZP_PORT_INFO pPortArr,
        INT nArrLen,
        PINT pPortCount,
        UINT nTimeout
        );
```

Delphi

```
function ZP_FindNextDevice(
    AHandle: THandle;
    var VInfo: TZP_DEVICE_INFO;
    VPortArr: PZP_PORT_INFO;
    AArrLen: Integer;
    var VPortCount: Integer;
    ATimeout: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор списка, полученный с помощью функции ZP_SearchDevices.

Info

[in,out] Буфер для информации о найденном устройстве.

PortArr

[in,out] Буфер для информации о портах, ассоциированных с устройством.

ArrLen

[in] Размер массива `PortArr`.

PortCount

[out] Количество портов, ассоциированных с устройством.

Timeout

[in] Тайм-аут для этой функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция завершается по тайм-ауту (параметр *Timeout*) и конвертер не найден, то возвращает `ZP_S_TIMEOUT`.

Если конвертер не найден, то возвращает `ZP_S_NOTFOUND`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DD_SetNotification

Настраивает уведомления. Если приложение является службой Windows, то предварительно нужно вызвать функцию ZP_SetServiceCtrlHandle и в обработчике события SERVICE_CONTROL_DEVICEEVENT вызывать ZP_DeviceEventNotify.

C++

```
HRESULT ZP_DD_SetNotification(
        PHANDLE pHandle,
        PZP_DD_NOTIFY_SETTINGS pSettings
        );
```

Delphi

```
function ZP_DD_SetNotification(
    var VHandle: THandle;
    const ASettings: TZP_DD_NOTIFY_SETTINGS
    ): HRESULT;
```

Параметры

Handle
[out] Возвращаемый дескриптор уведомителя.
Settings
[in] Параметры уведомлений.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DD_GetNextMessage

Возвращает следующее сообщение детектора, созданного с помощью функции ZP_DD_SetNotification.

C++

```
HRESULT ZP_DD_GetNextMessage(
        HANDLE hHandle,
        LPUINT pMsg,
        LPARAM* pMsgParam
        );
```

Delphi

```
function ZP_DD_GetNextMessage(
    AHandle: THandle;
    var VMsg: Cardinal;
    var VMsgParam: NativeInt
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор уведомителя.
Msg
[out] Тип сообщения.
MsgParam
[out] Параметры сообщения.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK. Если сообщений больше нет, возвращает - ZP_S_NOTFOUND.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DD_SetGlobalSettings

Устанавливает новые настройки детектора, созданного функцией ZP_DD_SetNotification.

C++

```
HRESULT ZP_DD_SetGlobalSettings(
        PZP_DD_GLOBAL_SETTINGS pSettings
        );
```

Delphi

```
function ZP_DD_SetGlobalSettings(
    ASettings: PZP_DD_GLOBAL_SETTINGS
    ): HRESULT;
```

Параметры

Settings

[in] Настройки детектора.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DD_GetGlobalSettings

Возвращает настройки детектора, созданного функцией ZP_DD_SetNotification.

C++

```
HRESULT ZP_DD_GetGlobalSettings(
        PZP_DD_GLOBAL_SETTINGS pSettings
        );
```

Delphi

```
function ZP_DD_GetGlobalSettings(
    var VSettings: TZP_DD_GLOBAL_SETTINGS
    ): HRESULT;
```

Параметры

Settings

[in,out] Буфер для настроек детектора.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DD_Refresh

Вызывает обновление списка устройств детектора, созданного функцией ZP_DD_SetNotification.

C++

```
HRESULT ZP_DD_Refresh(
        UINT nWaitMs
        );
```

Delphi

```
function ZP_DD_Refresh(
    AWaitMs: Cardinal
    ): HRESULT;
```

Параметры

WaitMs
[in] Тайм-аут обновления списка устройств.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_SetServiceCtrlHandle

Устанавливает дескриптор службы Windows. Если библиотека SDK используется службой, то перед запуском детектора с помощью ZP_DD_SetNotification нужно вызывать ZP_SetServiceCtrlHandle, чтобы библиотека настроила уведомления о подключении/отключении usb-конвертера. При этом когда службе будет приходить уведомление SERVICE_CONTROL_DEVICEEVENT нужно вызывать ZP_DeviceEventNotify.

C++

```
HRESULT ZP_SetServiceCtrlHandle(
        SERVICE_STATUS_HANDLE hSvc
        );
```

Delphi

```
function ZP_SetServiceCtrlHandle(
    ASvc: THandle
    ): HRESULT;
```

Параметры

Svc

[in] Дескриптор службы Windows, полученный функцией RegisterServiceCtrlHandlerEx.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_DeviceEventNotify

Обработчик уведомления `SERVICE_CONTROL_DEVICEEVENT` для службы Windows. Используется совместно с функциями ZP_SetServiceCtrlHandle и ZP_DD_SetNotification.

C++

```
VOID ZP_DeviceEventNotify(
        DWORD nEvType,
   PVOID pEvData
        );
```

Delphi

```
procedure ZP_DeviceEventNotify(
    AEvType: Cardinal;
    AEvData: Pointer
    );
```

Параметры

EvType
[in] Код события.
EvData
[in] Данные события.

Возвращаемое значение

нет.

ZP_RegisterPortRequest

Настраивает запрос для порта.

C++

```
HRESULT ZP_RegisterPortRequest(
        PZP_PORT_REQUEST pParams
        );
```

Delphi

```
function ZP_RegisterPortRequest(
    Const AParams: TZP_PORT_REQUEST
    ): HRESULT;
```

Параметры

Params

[in] Параметры запроса.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_RegisterUdpRequest

Настраивает Udp-запрос для ip-устройства.

C++

```
HRESULT ZP_RegisterUdpRequest(
        PZP_UDP_REQUEST pParams
        );
```

Delphi

```
function ZP_RegisterUdpRequest(
    Const AParams: TZP_UDP_REQUEST
    ): HRESULT;
```

Параметры

Params

[in] Параметры запроса.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_SetLog

Устанавливает ip-адрес ZLog.exe и путь к лог-файлу. Эта функция работает только в версии библиотеки в папке "Log".

C++

```
HRESULT ZP_SetLog(
        LPCWSTR pszSvrAddr,
        LPCWSTR pszFileName,
        UINT nFileTypeMask
        );
```

Delphi

```
function ZP_SetLog(
    ASvrAddr: PWideChar;
    AFileName: PWideChar;
    AFileTypeMask: Cardinal
    ): HRESULT;
```

Параметры

SvrAddr
: [in] IP-адрес ZLog.exe. Например "127.0.0.1:1818". Если порт не указан, то используется порт 1818. Если равен **null**, то не используется.

FileName
: [in] Путь к лог-файлу. Если равен **null**, то не используется. Папка в пути к файлу должна существовать.

FileTypeMask
: [in] Маска типов сообщений для лог-файла. Если равно 0, то без сообщений, если равно FFFFFFFFh, то все сообщения.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_GetLog

Возвращает настройки лога: ip-адрес ZLog.exe и путь к лог-файлу. Эта функция поддерживается только версией "Log" библиотеки.

C++

```
HRESULT ZP_GetLog(
        LPWSTR pszSvrAddrBuf,
        INT nSABufSize,
    LPWSTR pszFileNameBuf,
    INT nFNBufSize,
    PUINT pFileTypeMask
        );
```

Delphi

```
function ZP_GetLog(
    VSvrAddrBuf: PWideChar;
    ASABufSize: Integer;
    VFileNameBuf: PWideChar;
    AFNBufSize: Integer;
    var VFileTypeMask: Cardinal
    ): HRESULT;
```

Параметры

SvrAddrBuf
: [in,out] Буфер для строки с ip-адресом ZLog.exe. Может быть равен **null**.

SABufSize
: [in] Размер буфера для ip-адреса ZLog.exe.

FileNameBuf
: [in,out] Буфер для пути к лог-файлу. Может быть равен **null**.

FNBufSize
: [in] Размер буфера для пути к лог-файлу.

FileTypeMask
: [out] Маска типов сообщений для лог-файла.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_AddLog

Записывает новое сообщение в лог. Эта функция поддерживается только версией "Log" библиотеки.

C++

```
HRESULT ZP_AddLog(
        WCHAR chSrc,
        UINT nType,
        UINT nMsgId,
        LPCWSTR pszText
        );
```

Delphi

```
function ZP_AddLog(
    ASrc: WideChar;
    AMsgType: Integer;
    AText: PWideChar
    ): HRESULT;
```

Параметры

Src

[in] Символ, ассоциируемый с источником сообщений. Может быть любой буквой. Например 'P' - ZPort.dll, 'G' - ZGuard.dll, 'R' - ZReader.dll.

MsgType

[in] Тип сообщения. Сообщения для лог-файла фильтруются параметром `FileTypeMask`, установленным с помощью ZP_SetLog.

Text

[in] Текст сообщения.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Типы ZPort API

Имя типа	Кратное описание
Основные типы	
ZP_DD_NOTIFY_SETTINGS	Параметры для уведомлений.
ZP_WAIT_SETTINGS	Параметры ожидания исполнения функций.
ZP_DDN_PORT_INFO	Данные уведомления о порте.
ZP_DDN_DEVICE_INFO	Данные уведомления о конвертере.
Callback-функции	
ZP_DEVICEPARSEPROC	Функция обратного вызова для разбора ответа устройства на запрос.
Типы для работы с портом	
ZP_PORT_TYPE	Тип порта.
ZP_PORT_NAME	Имя порта.
ZP_PORT_ADDR	Адрес порта: тип + имя порта.
ZP_DEVICE_INFO	Базовая информация об устройстве.

ZP_PORT_TYPE

Тип порта.

C++

```
enum ZP_PORT_TYPE
{
        ZP_PORT_UNDEF = 0,
        ZP_PORT_COM,
        ZP_PORT_FT,
        ZP_PORT_IP,
        ZP_PORT_IPS
};
```

Delphi

```
TZP_PORT_TYPE = (
    ZP_PORT_UNDEF = 0,
    ZP_PORT_COM,
    ZP_PORT_FT,
    ZP_PORT_IP,
    ZP_PORT_IPS
  );
```

Константа	Описание
ZP_PORT_UNDEF	Не известно
ZP_PORT_COM	Com-порт (виртуальный COM-порт для USB-конвертера)
ZP_PORT_FT	С/н устройства USB (для подключения через ftd2xx.dll, которая входит в состав драйвера для usb-устройств)
ZP_PORT_IP	Ip-порт конвертера в режиме SERVER или PROXY (ПК подключается к конвертеру)

Константа	Описание
ZP_PORT_IPS	Ip-порт конвертера в режиме CLIENT (конвертер подключается к ПК)

ZP_PORT_NAME

Имя порта.

C++

```
typedef WCHAR ZP_PORT_NAME[ZP_MAX_PORT_NAME +
1];
```

Delphi

```
TZP_PORT_NAME = array[0..ZP_MAX_PORT_NAME] of
WideChar;
```

Константа ZP_MAX_PORT_NAME равна 31.

Структура ZP_PORT_ADDR

Адрес порта: тип и имя.

C++

```
typedef struct _ZP_PORT_ADDR
{
        ZP_PORT_TYPE nType;
        LPCWSTR pName;
        DWORD nDevTypes;
} *PZP_PORT_ADDR;
```

Delphi

```
  TZP_PORT_ADDR = packed record
    nType             : TZP_PORT_TYPE;
    pName             : PWideChar;
    nDevTypes         : Cardinal;
  end;
  PZP_PORT_ADDR = ^TZP_PORT_ADDR;
```

Параметры

Type
: Тип порта.

Name
: Имя порта.

DevTypes
: Маска типов устройств.

Структура ZP_PORT_INFO

Информация о порте.

C++

```
typedef struct _ZP_PORT_INFO
{
        ZP_PORT_TYPE nType;
        ZP_PORT_NAME szName;
        UINT nFlags;
        ZP_PORT_NAME szFriendly;
        UINT nDevTypes;
        WCHAR szOwner[64];
} *PZP_PORT_INFO;
```

Delphi

```
  TZP_PORT_INFO = packed record
    nType             : TZP_PORT_TYPE;
    szName            : TZP_PORT_NAME;
    nFlags            : Cardinal;
    szFriendly        : TZP_PORT_NAME;
    nDevTypes         : Cardinal;
    szOwner           : array[0..63] of WideChar;
  end;
  PZP_PORT_INFO = ^TZP_PORT_INFO;
```

Параметры

Type
Тип порта.
Name
Имя порта.
Flags
Флаги порта.

Флаг	Значение	Описание
ZP_PIF_BUSY	1	Порт занят.
ZP_PIF_USER	2	Порт был указан в параметрах функции ZP_SearchDevices.

Friendly

Дружественное имя порта.

DevTypes

Маска типов устройств.

Owner

IP-адрес компьютера, занявшего линию конвертера.

Структура ZP_DEVICE_INFO

Информация об устройстве.

C++

```
typedef struct _ZP_DEVICE_INFO
{
        UINT nTypeId;
        UINT nModel;
        UINT nSn;
        UINT nVersion;
} *PZP_DEVICE_INFO;
```

Delphi

```
  TZP_DEVICE_INFO = packed record
    nTypeId          : Cardinal;
    nModel           : Cardinal;
    nSn              : Cardinal;
    nVersion         : Cardinal;
  end;
  PZP_DEVICE_INFO = ^TZP_DEVICE_INFO;
```

Параметры

TypeId
: Тип устройства.

Model
: Модель устройства.

Sn
: Серийный номер устройства.

Version
: Версия прошивки устройства.

ZP_CONNECTION_STATUS

Состояние подключения.

C++

```
enum ZP_CONNECTION_STATUS
{
        ZP_CS_DISCONNECTED = 0,
        ZP_CS_CONNECTED,
        ZP_CS_CONNECTING,
        ZP_CS_RESTORATION
};
```

Delphi

```
  TZP_CONNECTION_STATUS = (
    ZP_CS_DISCONNECTED = 0,
    ZP_CS_CONNECTED,
    ZP_CS_CONNECTING,
    ZP_CS_RESTORATION
  );
```

Константа	Описание
ZP_CS_DISCONNECTED	Отключен.
ZP_CS_CONNECTED	Подключен.
ZP_CS_CONNECTING	В процессе подключения.
ZP_CS_RESTORATION	В процессе восстанавления связи.

Структура ZP_WAIT_SETTINGS

Параметры ожидания исполнения функций.

C++

```
typedef struct _ZP_WAIT_SETTINGS
{
        UINT nReplyTimeout;
        INT nMaxTries;
        HANDLE hAbortEvent;
        UINT nReplyTimeOut0;
        UINT nCheckPeriod;
        UINT nConnectTimeOut;
        UINT nRestorePeriod;
} *PZP_WAIT_SETTINGS;
```

Delphi

```
  TZP_WAIT_SETTINGS = packed record
    nReplyTimeout   : Cardinal;
    nMaxTries       : Cardinal;
    hAbortEvent     : THandle;
    nReplyTimeOut0  : Cardinal;
    nCheckPeriod    : Cardinal;
    nConnectTimeOut : Cardinal;
    nRestorePeriod  : Cardinal;
  end;
  PZP_WAIT_SETTINGS = ^TZP_WAIT_SETTINGS;
```

Параметры

ReplyTimeout
: Тайм-аут ожидания ответа на запрос конвертеру.

MaxTries
: Количество попыток отправить запрос.

AbortEvent

Дескриптор объекта Event для аварийного прерывания функции (объект Event может быть создан функцией CreateEvent). Если объект установлен в сигнальное состояние, то функция возвращает E_ABORT.

ReplyTimeOut0

Тайм-аут ожидания первого символа ответа.

CheckPeriod

Период проверки наличия данных в порте в миллисекундах. Если =0 или =FFFFFFFFh, то никогда, т.е. по RX-событию порта.

ConnectTimeOut

Тайм-аут ожидания подключения по TCP.

RestorePeriod

Период с которым будут осуществляться попытки восстановить утерянную TCP-связь. Если =FFFFFFFFh, то связь не восстанавливается.

Структура ZP_PORT_OPEN_PARAMS

Параметры открытия порта.

C++

```
typedef struct _ZP_PORT_OPEN_PARAMS
{
        LPCWSTR szName;
        ZP_PORT_TYPE nType;
        UINT nBaud;
        CHAR nEvChar;
        BYTE nStopBits;
        UINT nConnectTimeout;
        UINT nRestorePeriod;
        UINT nFlags;
} *PZP_PORT_OPEN_PARAMS;
```

Delphi

```
  TZP_PORT_OPEN_PARAMS = packed record
    szName           : PWideChar;
    nType            : ZP_PORT_TYPE;
    nBaud            : DWord;
    nEvChar          : AnsiChar;
    nStopBits        : Byte;
    nConnectTimeout  : Cardinal;
    nRestorePeriod   : Cardinal;
    nFlags           : Cardinal;
  end;
  PZP_PORT_OPEN_PARAMS = ^TZP_PORT_OPEN_PARAMS;
```

Параметры

Name

Имя порта.

Type
: Тип порта.

Baud
: Скорость порта.

EvChar
: Символ, определяющий конец передачи (если =0, нет символа).

StopBits
: Стоповые биты (`ONESTOPBIT=0`, `ONE5STOPBITS=1`, `TWOSTOPBITS=2`).

ConnectTimeout
: Тайм-аут подключения к ip-конвертеру. Если =0, то используется значение по умолчанию.

RestorePeriod
: Период восстановления связи. Если =0, то используется значение по умолчанию.

Flags
: Флаги.

Флаг	Значение	Описание
`ZP_POF_NO_WAIT_CONNECT`	1	Не ждать завершения процедуры подключения
`ZP_POF_NO_CONNECT_ERR`	2	Не возвращать ошибку в случае когда нет связи
`ZP_POF_NO_DETECT_USB`	4	Не использовать детектор USB-устройств (для `ZP_PORT_FT` и `ZP_PORT_COM`)

Структура ZP_DD_NOTIFY_SETTINGS

Параметры уведомлений, используются функцией ZP_DD_SetNotification. Дополнительные параметры задаются с помошью функции ZP_DD_SetGlobalSettings.

C++

```
typedef struct _ZP_DD_NOTIFY_SETTINGS
{
        UINT nNMask;

        HANDLE hEvent;
        HWND hWindow;
        UINT nWndMsgId;

        DWORD nSDevTypes;
        DWORD nIpDevTypes;

        LPWORD aIps;
        INT nIpsCount;
} *PZP_DD_NOTIFY_SETTINGS;
```

Delphi

```
  TZP_DD_NOTIFY_SETTINGS = packed record
    nNMask           : Cardinal;

    hEvent           : THandle;
    hWindow          : HWnd;
    nWndMsgId        : Cardinal;

    nSDevTypes       : Cardinal;
    nIpDevTypes      : Cardinal;

    aIps             : PCardinal;
    nIpsCount        : Integer;
```

```
  end;
  PZP_DD_NOTIFY_SETTINGS =
^TZP_DD_NOTIFY_SETTINGS;
```

Параметры

NMask

Маска типов уведомлений:

Флаг	Значение	Описание
ZP_NF_EXIST	1	Уведомлять о подключении/ отключении порта. Приходят сообщения ZP_N_INSERT и ZP_N_REMOVE.
ZP_NF_CHANGE	2	Уведомлять об изменении параметров порта. Приходит сообщение ZP_N_CHANGE.
ZP_NF_ERROR	8	Уведомлять о возникновении ошибок в потоке (thread). Приходит сообщение ZP_N_ERROR.
ZP_NF_SDEVICE	10h	Опрашивать порты типа ZP_PORT_COM и ZP_PORT_FT.
ZP_NF_IPDEVICE	20h	Опрашивать порты типа ZP_PORT_IP (конвертеры в режиме SERVER)

Флаг	Значение	Описание
ZP_NF_IPSDEVICE	80h	Опрашивать порты типа ZP_PORT_IPS (конвертеры в режиме CLIENT)
ZP_NF_COMPLETED	40h	Уведомлять о завершении сканирования. В callback-функцию приходит сообщение ZP_N_COMPLETED.
ZP_NF_DEVEXIST	4h	Уведомлять о подключении/ отключении устройств. Приходят сообщения ZP_N_DEVINSERT и ZP_N_DEVREMOVE.
ZP_NF_DEVCHANGE	100h	Уведомления о изменении параметров устройств. Приходит сообщение ZP_N_DEVCHANGE.
ZP_NF_UNIDCOM	1000h	Искать неопознанные com-порты, т.е. те com-порты, для которых не удалось определить тип устройства
ZP_NF_USECOM	2000h	Для портов типа ZP_PORT_FT использовать

Флаг	Значение	Описание
		дружественное имя (com-порт).

Event

Дескриптор события (объекта синхронизации), полученного с помощью функции `CreateEvent` из WinApi. Если равно **null**, то будут использоваться параметры `Window` и `WndMsgId`.

Window

Дескриптор окна, в которое будет отправлено сообщение `WndMsgId`. Может быть равно **null**.

WndMsgId

Сообщение, которое будет отправляется окну когда появятся новые уведомления.

SDevTypes

Маска тивов устройств, подключенных к последовательному порту.

IpDevTypes

Маска тивов Ip-устройств.

Ips

Указатель на массив двубайтовых целых чисел, в котором задаются TCP-порты для прослушивания конвертеров в режиме CLIENT. Может быть равен **null**.

IpsCount

Количество элементов в массиве *Ips*.

Структура ZP_DDN_PORT_INFO

Информация для уведомлений ZP_N_INSERT, ZP_N_REMOVE и ZP_N_CHANGE. Уведомления настраиваются с помощью функции ZP_DD_SetNotification, и извлекаются с помощью ZP_DD_GetNextMessage.

C++

```
typedef struct _ZP_DDN_PORT_INFO
{
        _ZP_PORT_INFO rPort;
        PZP_DEVICE_INFO* aDevs;
        INT nDevCount;
        UINT nChangeMask;
} *PZP_DDN_PORT_INFO;
```

Delphi

```
  TZP_DDN_PORT_INFO = packed record
    rPort            : TZP_PORT_INFO;
    aDevs            : ^PZP_DEVICE_INFO;
    nDevCount        : Integer;
    nChangeMask      : Cardinal;
  end;
  PZP_DDN_PORT_INFO = ^TZP_DDN_PORT_INFO;
```

Параметры

Port

Информация о порте.

Devs

Список устройств (массив указателей на информацию о устройствах). Может быть равно **null**.
Если `(aDevs[i].nTypeId >= ZP_MAX_REG_DEV)`, то устройство найдено опросом по UDP.

Если `(aDevs[i].nTypeId < ZP_MAX_REG_DEV)`, то устройство найдено опросом портов.

DevCount

Количество элементов в списке `Devs`.

ChangeMask

Маска изменений - набор битов, определяющих какой параметр изменился:

Константа	Значение	Описание
`ZP_CIF_BUSY`	4	Изменено состояние занятости порта.
`ZP_CIF_FRIENDLY`	8	Изменено дружественное имя порта (параметр `rPort.szFriendly`).
`ZP_CIF_OWNER`	20h	Изменен владелец порта (параметр `rPort.szOwner`) (заполняется только если устройство найдено по UDP).
`ZP_CIF_LIST`	800h	Изменен список устройств, связанных с портом.

Структура ZP_DDN_DEVICE_INFO

Информация для уведомлений ZP_N_DEVINSERT, ZP_N_DEVREMOVE и ZP_N_DEVCHANGE. Уведомления настраиваются с помощью функции ZP_DD_SetNotification, и извлекаются с помощью ZP_DD_GetNextMessage.

C++

```
typedef struct _ZP_DDN_DEVICE_INFO
{
        PZP_DEVICE_INFO pInfo;
        _ZP_PORT_INFO aPorts;
        INT nPortCount;
        UINT nChangeMask;
} *PZP_DDN_DEVICE_INFO;
```

Delphi

```
  TZP_DDN_DEVICE_INFO = packed record
    pInfo             : PZP_DEVICE_INFO;
    aPorts            : TZP_PORT_INFO;
    nPortCount        : Integer;
    nChangeMask       : Cardinal;
  end;
  PZP_DDN_DEVICE_INFO = ^TZP_DDN_DEVICE_INFO;
```

Параметры

Info

Информация об устройстве. Может быть равно **null**.
Если (pInfo.nTypeId >= ZP_MAX_REG_DEV), то устройство найдено опросом по UDP.
Если (pInfo.nTypeId < ZP_MAX_REG_DEV), то устройство найдено опросом портов.

Ports

Список портов, связанных с устройством (массив с информацией о портах).

PortCount

Количество элементов в списке Ports.

ChangeMask

Маска изменений.

Константа	Значение	Описание
`ZP_CIF_VERSION`	200h	Изменена версия прошивки устройства (параметр `pInfo.nVersion`).
`ZP_CIF_DEVPARAMS`	400h	Изменены расширенные параметры устройства (параметры `pInfo` за пределами структуры `ZP_DEVICE_INFO`).
`ZP_CIF_LIST`	800h	Изменен список портов, связанных с устройством.

Структура ZP_DD_GLOBAL_SETTINGS

Настройки детектора, который создается функцией ZP_DD_SetNotification. Для установки новых параметров используйте ZP_DD_SetGlobalSettings, для получения - ZP_DD_GetGlobalSettings.

C++

```
typedef struct _ZP_DD_GLOBAL_SETTINGS
{
        UINT nCheckUsbPeriod;
        UINT nCheckIpPeriod;
        UINT nScanDevPeriod;
        UINT nIpReqTimeout;
        INT nIpReqMaxTries;
        _ZP_WAIT_SETTINGS rScanWS;
        UINT nIpsReqs;
} *PZP_DD_GLOBAL_SETTINGS;
```

Delphi

```
  TZP_DD_GLOBAL_SETTINGS = packed record
    nCheckUsbPeriod : Cardinal;
    nCheckIpPeriod  : Cardinal;
    nScanDevPeriod  : Cardinal;
    nIpReqTimeout   : Cardinal;
    nIpReqMaxTries  : Integer;
    rScanWS         : TZP_WAIT_SETTINGS;
    nIpsReqs        : Cardinal;
  end;
  PZP_DD_GLOBAL_SETTINGS =
^TZP_DD_GLOBAL_SETTINGS;
```

Параметры

CheckUsbPeriod
: Период проверки списка последовательных портов (в миллисекундах).

CheckIpPeriod
: Период опроса IP-конвертеров по UDP (в миллисекундах).

nScanDevPeriod
: Период сканирования устройств, опросом портов (в миллисекундах).

nIpReqTimeout
: Тайм-аут ожидания ответа на запрос по UDP (в миллисекундах).

IpReqMaxTries
: Количество попыток опросить по UDP.

ScanWS
: Параметр ожидания при опросе последовательных портов.

IpsReqs
: Маска запросов для идентификации конвертеров в режиме "Клиент".

ZP_DEVICEPARSEPROC

Функция обратного вызова для разбора ответа устройства на запрос.

C++

```
typedef BOOL (CALLBACK* ZP_DEVICEPARSEPROC) (
        LPCVOID pReply,
        UINT nCount,
        PBOOL pPartially,
        PZP_DEVICE_INFO pInfo,
        PZP_PORT_INFO pPortArr,
        INT nArrLen,
        PINT pPortCount
);
```

Delphi

```
TZP_DEVICEPARSEPROC = function(
  AReply: Pointer;
  ACount: Cardinal;
  var VPartially: LongBool;
  VInfo: PZP_DEVICE_INFO;
  VPortArr: PZP_PORT_INFO;
  AArrLen: Integer;
  var VPortCount: Integer
): LongBool; stdcall;
```

Параметры

Reply
: Ответ устройства на запрос. Может быть не полным при опросе порта.

Count
: Количество байт в ответе.

Partially
: Флаг, указывающий частичный ответ или полный.

Info
: Информация об устройстве.

PortArr
: Буфер для списка с информацией о портах.

ArrLen
: Размер списка.

PortCount
: Количество портов, связанных с устройством. Должно быть равно количеству заполненных элементов списка `PortArr`.

Константы ZPort API

Значения по умолчанию, коды ошибок.

Константа	Значение	Описание
ZP_SDK_VER_MAJOR	1	Старший номер версии ZPort API.
ZP_SDK_VER_MINOR	18	Младший номер версии ZPort API.
ZP_MAX_PORT_NAME	31	Максимальная длина имени порта (исключая завершающий ноль).
ZP_MAX_REG_DEV	32	Максимальное количество зарегистрированных запросов.

Коды ошибок ZPort API

Константа	Значение	Описание
S_OK	0	Операция выполнена успешно.
E_FAIL	80004005h	Незвестная ошибка.
E_INVALIDARG	80070057h	Неправильные параметры.
E_NOINTERFACE	80004002h	Функция не поддерживается.
E_OUTOFMEMORY	8007000Eh	Недостаточно памяти для обработки команды.
E_HANDLE	80000006h	Неправильный дескриптор.
E_ABORT	80000007h	Функция прервана (см. ZP_WAIT_SETTINGS).
E_ACCESSDENIED	80000009h	Отказано в доступе.
ZP_S_CANCELLED	00040201h	Отменено пользователем.
ZP_S_NOTFOUND	00040202h	Не найден (для функций поиска)
ZP_S_TIMEOUT	00040203h	Операция завершилась по тайм-ауту.
ZP_E_OPENNOTEXIST	80040203h	Порт не существует.
ZP_E_OPENPORT	80040205h	Другая ошибка открытия порта.
ZP_E_PORTIO	80040206h	Ошибка порта.

Константа	Значение	Описание
ZP_E_PORTSETUP	80040207h	Ошибка настройки порта.
ZP_E_LOADFTD2XX	80040208h	Неудалось загрузить Ftd2xx.dll.
ZP_E_SOCKET	80040209h	Не удалось инициализировать сокеты.
ZP_E_SERVERCLOSE	8004020Ah	Дескриптор закрыт со стороны сервера.
ZP_E_NOTINITALIZED	8004020Bh	Не проинициализировано с помощью ZP_Initialize.
ZP_E_INSUFFICIENTBUFFER	8004020Ch	Размер буфера слишком мал.
ZP_E_NOCONNECT	8004020Dh	Нет связи с устройством.

Значения по умолчанию ZPort API

Все временные параметры измеряются в миллисекундах.

Константа	Значение	Описание
ZP_IP_CONNECTTIMEOUT	4000	Тайм-аут подключения по TCP для порта типа ZP_PORT_IP.
ZP_IP_RESTOREPERIOD	3000	Период восстановления утерянной TCP-связи (для порта типа ZP_PORT_IP).
ZP_IPS_CONNECTTIMEOUT	10000	Тайм-аут подключения по TCP для порта типа ZP_PORT_IPS.
ZP_USB_RESTOREPERIOD	3000	Период восстановления утерянной связи (для портов типов ZP_PORT_COM и ZP_PORT_FT)
ZP_DTC_FINDUSBPERIOD	5000	Период поиска USB-устройств (только поиск com-портов) (для детектора устройств).
ZP_DTC_FINDIPPERIOD	15000	Период поиска IP-устройства по UDP (для детектора устройств).
ZP_DTC_SCANDEVPERIOD	FFFFFFFFh	Период сканирования устройств, опросом

Константа	Значение	Описание
		портов (для детектора устройств).
ZP_SCAN_RCVTIMEOUT0	500	Тайм-аут ожидания первого байта ответа на запрос при сканировании устройств.
ZP_SCAN_RCVTIMEOUT	3000	Тайм-аут ожидания ответа на запрос при сканировании устройств.
ZP_SCAN_MAXTRIES	2	Максимум попыток запроса при сканировании устройств.
ZP_SCAN_CHECKPERIOD	FFFFFFFFh	Период проверки входяших данных порта при сканировании портов.
ZP_FINDIP_RCVTIMEOUT	1000	Тайм-аут поиска ip-устройств по UDP.
ZP_FINDIP_MAXTRIES	1	Максимум попыток поиска ip-устройств по UDP.

Уведомления ZPort API

Уведомления детектора устройств

Уведомления об изменении списка портов (настраиваются с помощью функции ZP_DD_SetNotification)

<table>
<tr><th>Сообщение</th><th>Параметр</th><th>Описание</th></tr>
<tr><td>ZP_N_INSERT</td><td>PZP_DDN_PORT_INFO</td><td>Подключение порта (подключение устройства к USB). В параметре - указатель на информацию о порте.</td></tr>
<tr><td>ZP_N_REMOVE</td><td>PZP_DDN_PORT_INFO</td><td>Отключение порта (отключение устройства от USB). В параметре - указатель на информацию о порте.</td></tr>
<tr><td>ZP_N_CHANGE</td><td>PZP_DDN_PORT_INFO</td><td>Изменение состояния порта. В параметре - инфо об изменениях.</td></tr>
<tr><td>ZP_N_ERROR</td><td>PHRESULT</td><td>Произошла ошибка в потоке (thread). В параметре - указатель на код ошибки.</td></tr>
<tr><td>ZP_N_COMPLETED</td><td>PINT</td><td>Сканирование завершено. В параметре - маска:<table><tr><th>Флаг</th><th>Описание</th></tr><tr><td>1</td><td>Обновлен список com-портов.</td></tr><tr><td>2</td><td>Обновлен список ip-портов.</td></tr></table></td></tr>
</table>

<table>
<tr><th>Сообщение</th><th>Параметр</th><th>Описание</th></tr>
<tr><td></td><td></td><td><table>
<tr><th>Флаг</th><th>Описание</th></tr>
<tr><td>4</td><td>Обновлена информация об устройствах.</td></tr>
<tr><td>80000000h</td><td>Ошибка при сканировании.</td></tr>
</table></td></tr>
<tr><td>ZP_N_DEVINSERT</td><td>PZP_DDN_DEVICE_INFO</td><td>Подключение устройства.</td></tr>
<tr><td>ZP_N_DEVREMOVE</td><td>PZP_DDN_DEVICE_INFO</td><td>Отключение устройства.</td></tr>
<tr><td>ZP_N_DEVCHANGE</td><td>PZP_DDN_DEVICE_INFO</td><td>Изменение параметров устройства</td></tr>
</table>

Уведомления порта

Уведомления порта настраиваются с помощью функции ZP_Port_SetNotification и извлекаются с помощью ZP_Port_EnumMessages.

Флаг	Описание
ZP_PNF_RXEVENT	Поступили данные от устройства.
ZP_PNF_STATUS	Измененилось состояние подключения к устройству. Для получения текущего состояния используйте ZP_Port_GetConnectionStatus.

Вопросы и ответы

1. **Что делать если после установки драйвера для usb-конвертера не появился com-порт?** Драйвер для usb-конвертера устанавливается в 2 этапа: сначала устанавливается драйвер для usb-устройства, затем - драйвер виртуального com-порта. Если не появился com-порт, то скорее всего не установлена вторая часть драйвера. Нужно открыть Диспетчер устройств найти в разделе "Другие устройства" элемент "USB Serial Port" и обновить ему драйвер, указав мастеру установки на папку с драйвером.

2. **Какие особенности работы библиотеки в службе?**

    1. `ZP_Initialize` нужно вызывать с флагом `ZP_IF_NO_MSG_LOOP`;
    2. До первого вызова `ZP_DD_SetNotification` нужно передать библиотеке дескриптор службы с помощью `ZP_SetServiceCtrlHandle` и в обработчике события службы `SERVICE_CONTROL_DEVICEEVENT` вызывать `ZP_DeviceEventNotify`.

Утилиты

В составе SDK:

1. ZRetr.exe - эмулирует Ip-конвертеры с помощью usb-конвертеров (поддерживаются режимы SERVER и CLIENT).
2. ZLog.exe - показывает отладочные сообщения библиотеки "ZGuard.dll" из папки "Log".

На сайте www.ironlogic.ru:

1. Com2ip - программа предназначена для создания в операционной системе Windows виртуальных COM портов и перенаправления данных из этих портов через сеть TCP/IP к конвертерам Z397 Web. Это позволяет использовать конвертеры Z397 Web с программным обеспечением, рассчитанным на работу с обычными COM портами;
2. Find397 - Программа для обновления прошивки и поиска конвертера Z397 IP в локальной сети;
3. FindIP, FindWeb - Программы для поиска и конфигурирования конвертеров Z397 IP и Z397 Web.

Контакт с автором

E-mail: marketing@ironlogic.ru

Internet: www.ironlogic.ru

RF Enabled Limited

UK, Northumberland House
Aldbridge Suite 230 High Street
BROMLEY KENT BR1 1PQ
Mobile: +44(0) 780 385 3449.

ZP_Port_Write

Отправляет данные устройству.

C++

```
HRESULT ZP_Port_Write(
        HANDLE hHandle,
        LPCVOID pBuf,
        UINT nCount
        );
```

Delphi

```
function ZP_Port_Write(
    AHandle: THandle;
    Const ABuf;
    ACount: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

Buf
: [in] Данные для отправки устройству.

Count
: [in] Размер данных (в байтах).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_Open

Открывает порт.

C++

```
HRESULT ZP_Port_Open(
        PHANDLE pHandle,
        PZP_PORT_OPEN_PARAMS pParams
        );
```

Delphi

```
function ZP_Port_Open(
    var VHandle: THandle;
    AParams: PZP_PORT_OPEN_PARAMS
    ): HRESULT;
```

Параметры

Handle
[out] Дескриптор порта.
Params
[in] Параметры открытия порта.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_SetBaudAndEvChar

Устанавливает скорость порта и сигнальный символ.

C++

```
HRESULT ZP_Port_SetBaudAndEvChar(
        HANDLE hHandle,
        UINT nBaud,
        CHAR chEvChar
        );
```

Delphi

```
function ZP_Port_SetBaudAndEvChar(
    AHandle: THandle;
    ABaud: Cardinal;
    AEvChar: AnsiChar
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

Baud
: [in] Скорость порта.

EvChar
: [in] Символ, который используется для уведомления о поступлении данных от устройства. Если в поступивших данных встречается этот символ, то Sdk посылает уведомление (если соответствующее уведомление настроено с помощью `ZP_Port_SetNotification`).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_GetBaudAndEvChar

Возвращает скорость порта и сигнальный символ.

C++

```
HRESULT ZP_Port_GetBaudAndEvChar(
        HANDLE hHandle,
        LPUINT pBaud,
        PCHAR pEvChar
        );
```

Delphi

```
function ZP_Port_GetBaudAndEvChar(
    AHandle: THandle;
    VBaud: PCardinal;
    VEvChar: PAnsiChar
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

Baud
: [out] Скорость порта.

EvChar
: [out] Символ, который используется для уведомления о поступлении данных от устройства.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_GetConnectionStatus

Возвращает текущее состояние подключения к устройству.

C++

```
HRESULT ZP_Port_GetConnectionStatus(
        HANDLE hHandle,
        ZP_CONNECTION_STATUS* pValue
        );
```

Delphi

```
function ZP_Port_GetConnectionStatus(
    AHandle: THandle;
    var VValue: TZP_CONNECTION_STATUS
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор порта.
Value
[out] Состояние подключения.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_SetNotification

Настраивает уведомления порта. Для извлечения уведомлений используйте функцию ZP_Port_EnumMessages.

C++

```
HRESULT ZP_Port_SetNotification(
        HANDLE hHandle,
        HANDLE hEvent,
        HWND hWindow,
        UINT nWndMsgId,
        UINT nMsgMask
        );
```

Delphi

```
function ZP_Port_SetNotification(
    AHandle: THandle;
    AEvent: THandle;
    AWindow: HWnd;
    AWndMsgId: Cardinal;
    AMsgMask: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор порта.

Event

[in] Дескриптор события (объекта синхронизации), полученного с помощью функции `CreateEvent` из WinApi. Если равно **null**, то будут использоваться параметры `Window` и `WndMsgId`.

Window

[in] Дескриптор окна, в которое будет отправлено сообщение `WndMsgId`. Может быть равно **null**.

WndMsgId

[in] Сообщение, которое будет отправляется окну когда появятся новые уведомления.

MsgMask

[in] Маска типов уведомлений, которые будут приходить.

Флаг	Значение	Описание
`ZP_PNF_RXEVENT`	1	Уведомлять о поступлении данных от устройства.
`ZP_PNF_STATUS`	2	Уведомлять об изменении состояния подключения к устройству.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_EnumMessages

Возвращает уведомления порта. Уведомления настраиваются с помощью функции ZP_Port_SetNotification.

C++

```
HRESULT ZP_Port_EnumMessages(
        HANDLE hHandle,
        PUINT pMsgs
        );
```

Delphi

```
function ZP_Port_EnumMessages(
    AHandle: THandle;
    var VMsgs: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор порта.

Msgs

[out] Маска уведомлений.

Флаг	Значение	Описание
ZP_PNF_RXEVENT	1	Поступили данные от устройства.
ZP_PNF_STATUS	2	Измененилось состояние подключения к устройству. Для получения текущего состояния используйте ZP_Port_GetConnectionStatus.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_Clear

Очищает входящий и исходящий буфер данных порта.

C++

```
HRESULT ZP_Port_Clear(
        HANDLE hHandle,
        BOOL fIn,
        BOOL fOut
        );
```

Delphi

```
function ZP_Port_Clear(
    AHandle: THandle;
    AIn, AOut: LongBool
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

In
: [in] True, очистить входящий буфер.

Out
: [in] True, очистить исходящий буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_Read

Возвращает данные, полученные от устройства.

C++

```
HRESULT ZP_Port_Read(
        HANDLE hHandle,
        LPVOID pBuf,
        UINT nCount,
        PUINT pRead
        );
```

Delphi

```
function ZP_Port_Read(
    AHandle: THandle;
    var VBuf;
    ACount: Cardinal;
    var VRead: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

Buf
: [in,out] Буфер для данных, полученных от устройства.

Count
: [in] Размер буфера (в байтах).

Read
: [out] Количество байт, скопированных в буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_GetInCount

Возвращает количество байт, полученных от устройства.

C++

```
HRESULT ZP_Port_GetInCount(
        HANDLE hHandle,
        PUINT pCount
        );
```

Delphi

```
function ZP_Port_GetInCount(
    AHandle: THandle;
    var VCount: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор порта.
Count
[out] Количество байт, полученных от устройства.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_SetDtr

Настраивает Data Terminal Ready (DTR) сигнал управления.

C++

```
HRESULT ZP_Port_SetDtr(
        HANDLE hHandle,
        BOOL fState
        );
```

Delphi

```
function ZP_Port_SetDtr(
    AHandle: THandle;
    AState: LongBool
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

State
: [in] True, устанавливает Data Terminal Ready (DTR) сигнал управления, иначе - сбрасывает.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZP_Port_SetRts

Настраивает Request To Send (RTS) сигнал управления.

C++

```
HRESULT ZP_Port_SetRts(
        HANDLE hHandle,
        BOOL fState
        );
```

Delphi

```
function ZP_Port_SetRts(
    AHandle: THandle;
    AState: LongBool
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор порта.

State
: [in] True, устанавливает Request To Send (RTS) сигнал управления, иначе - сбрасывает.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Структура ZP_SEARCH_PARAMS

Параметры функции поиска устройств ZP_SearchDevices.

C++

```
typedef struct _ZP_SEARCH_PARAMS
{
        UINT nDevMask;
        UINT nIpDevMask;
        PZP_PORT_ADDR pPorts;
        INT nPCount;
        UINT nFlags;
        PZP_WAIT_SETTINGS pWait;
        UINT nIpReqTimeout;
        INT nIpReqMaxTries;
} *PZP_SEARCH_PARAMS;
```

Delphi

```
  TZP_SEARCH_PARAMS = packed record
    nDevMask         : Cardinal;
    nIpDevMask       : Cardinal;
    pPorts           : PZP_PORT_ADDR;
    nPCount          : Integer;
    nFlags           : Cardinal;
    pWait            : PZP_WAIT_SETTINGS;
    nIpReqTimeout    : Cardinal;
    nIpReqMaxTries   : Integer;
  end;
  PZP_SEARCH_PARAMS = ^TZP_SEARCH_PARAMS;
```

Параметры

DevMask

Маска типов устройств (для опроса портов).

IpDevMask

Маска типов ip-устройств (для опроса по UDP).

Ports

Список портов, которых нужно опросить.

PCount

Размер списка портов `Ports`.

Flags

Опции поиска.

Константа для флага	Значение	Описание
`ZP_SF_USECOM`	1	Если установлен, то для FT-портов (`ZP_PORT_FT`) используется связанный с ним COM-порт (дружественное имя).
`ZP_SF_DETECTOR`	2	Если установлен, то поиск производиться не будет, а будет возвращет список уже найденных устройств с помощью детектора (создается функцией `ZP_DD_SetNotification`).
`ZP_SF_IPS`	4	Если установлен, то в список будут добавлены найденные IP-конвертеры в режиме CLIENT.
`ZP_SF_UNID`	8	Если установлен, то в список будут добавлены порты неопознанных конвертеров.
`ZP_SF_UNIDCOM`	10h	Включить в список неопознанные com-порты, т.е. порты, для которых не удалось определить тип устройства

Wait

Параметры ожидания при опросе портов.

IpReqTimeout

Тайм-аут ожидания ответа при опросе по UDP.

IpReqMaxTries

Количество попыток опросить по UDP.

Структура ZP_PORT_REQUEST

Параметры запроса для порта.

C++

```
typedef struct _ZP_PORT_REQUEST : _ZP_REQUEST
{
        PDWORD pVidPids;
        INT nVidPidCount;
        UINT nBaud;
        CHAR nEvChar;
        BYTE nStopBits;
        LPCWSTR pszBDesc;
} *PZP_PORT_REQUEST;
```

Delphi

```
  TZP_PORT_REQUEST = packed record
    rBase            : TZP_REQUEST;
    pVidPids         : PDWord;
    nVidPidCount     : Integer;
    nBaud            : Cardinal;
    nEvChar          : AnsiChar;
    nStopBits        : Byte;
    pszBDesc         : PWideChar;
  end;
  PZP_PORT_REQUEST = ^TZP_PORT_REQUEST;
```

Параметры

Base

Базовые параметры запроса.

VidPids

Массив пар VID-PID для фильтрации usb-устройств, которым предназначен запрос.

VidPidCount
: Количество элементов в массиве VidPids.

Baud
: Скорость порта.

EvChar
: Символ-признак конца передачи. Если равен 0, то не используется.

StopBits
: Символ, сигнализирующий о поступлении ответа от устройства.

BDesc
: Описание устройства, предоставленное usb-шиной (для usb-устройств).

Структура ZP_UDP_REQUEST

Параметры Udp-запроса для ip-устройства.

C++

```
typedef struct _ZP_UDP_REQUEST : _ZP_REQUEST
{
        WORD nReqPort;
        INT nMaxPort;
} *PZP_UDP_REQUEST;
```

Delphi

```
  TZP_UDP_REQUEST = packed record
    rBase            : TZP_REQUEST;
    nReqPort         : Word;
    nMaxPort         : Integer;
  end;
  PZP_UDP_REQUEST = ^TZP_UDP_REQUEST;
```

Параметры

Base
: Базовые параметры запроса.

ReqPort
: UDP-порт для широковещательного запроса.

MaxPort
: Максимум портов у устройства.

Структура ZP_REQUEST

Информация о запросе.

C++

```
typedef struct _ZP_REQUEST
{
        UINT nTypeId;
        LPCVOID pReqData;
        UINT nReqSize;
        ZP_DEVICEPARSEPROC pfnParse;
        UINT nDevInfoSize;
} *PZP_REQUEST;
```

Delphi

```
  TZP_REQUEST = packed record
    nTypeId          : Cardinal;
    pReqData         : Pointer;
    nReqSize         : Cardinal;
    pfnParse         : TZP_DEVICEPARSEPROC;
    nDevInfoSize     : Cardinal;
  end;
  PZP_REQUEST = ^TZP_REQUEST;
```

Параметры

TypeId
: Тип запроса.

ReqData
: Данные запроса.

ReqSize
: Размер данных запроса (в байтах).

Parse
: Функция для разбора ответа от устройства.

DevInfoSize

Размер структуры для информации об устройстве (в байтах). Должно быть больше или равно размера базовой структуры ZP_DEVICE_INFO.