Введение

Данный пакет SDK предназначен для интеграции сетевых контроллеров в разрабатываемые системы. Работа с контроллерами осуществляется с помощью конвертеров (исключение - контроллер Z-5R Web, для работы с ним отдельный конвертер не нужен).

Комплект SDK:

1. Библиотеки в нескольких вариантах сборки:
    1. ZGurard.dll – 32-битная библиотека;
    2. x64\ZGuard.dll – 64-битная библиотека;
    3. Split\ZGuard.dll, Split\ZPort.dll – 32-битные библиотеки для совместного использования с библиотеками из SDK Readers);
    4. Split\x64\ZGuard.dll, Split\x64\ZPort.dll – 64-битные библиотеки для совместного использования с библиотеками из SDK Readers);
    5. Log\ZGuard.dll – 32-битная библиотека, в которую добавлены функции лога;
    6. Log\x64\ZGuard.dll – 64-битная библиотека, в которую добавлены функции лога;
2. Демонстрационная программа - Demo.exe;
3. Примеры использования библиотек для разных языков программирования (ЯП):
    1. Borland C++ - папка "BCPP";
    2. Visual C++ - папка "VCPP";
    3. C# - папка "CSharp";
    4. Delphi - папка "Delphi";
4. Данный файл описания SDK – Help\ZGurard_rus.chm
5. Прошивки для конвертеров и контроллеров – папка "UpdateFW"
6. Утилита для работы с конвертером Z-397, Z-397 Guard по сети TCP/IP – Split\ZRetr.exe;

7. Утилита для простотра лога, создаваемого библитеками из папки "Log" – Log\Log.exe
8. Утилита для проверки обновлений Sdk – Updater\ZUpdater.exe;
9. Файл лицензии на 16 контроллеров для конвертеров Z-397 Guard, Z-397 IP, Z-397 Web и контроллера Z-5R Web (на новые конвертеры SDK устанавливает автоматически) – Lic_SDK_16.lic

Основные возможности

- **Для конвертера:**
  - Поиск конвертеров;
  - Уведомление о подключении/отключении конвертера от ПК;
  - Получение информации о конвертере (тип, с/н, версия, инф.строки);
  - Получение информации о лицензии конвертера, установка новой лицензии, удаление всех лицензий;
  - Перепрошивка конвертера;
  - Работа с конвертером на низком уровне (посылка произвольных команд);
- **Для контроллера:**
  - Поиск контроллеров, подключенных к определенному конвертеру;
  - Перепрошивка контроллера;
  - Получение информации о контроллере (тип, с/н, сетевой адрес, инф.строки);
  - Получение / установка сетевого адреса;
  - Получение / установка часов контроллера;
  - Получение / установка времен замков;
  - Получение / установка временных зон;
  - Получение / установка / удаленние ключей;
  - Получение номера последнего поднесенного ключа к считывателю контроллера;
  - Получение указателей событий контроллера;
  - Получение событий контроллера;
  - Дистанционное управление контроллером (открытие двери);
  - Уведомление о подключении/отключении контроллера;
  - Уведомление о новых событиях контроллера;
  - Уведомление о рассинхронизации часов контроллера;
  - Уведомление о изменении верхней границы ключей;

- **Для контроллера Matrix II Net с прошивкой версии 3.0:**
  - Получение / установка параметров электропитания;
  - Получение текущего состояния электропитания;
  - Включение / выключение электропитания.

Особенности SDK Guard:

- Текстовые строки передаются в UNICODE (WideChar);
- Номера ключей совместимы с номерами ключей SDK Readers .

Требования к системе

ОС: Windows® XP/Vista/Seven

Прошивки устройств и драйвера: При обнаружении неправильной работы SDK с конвертерами или с контроллерами рекомендуется обновить прошивки устройств и драйверов (для usb конвертеров). Самые последние версии прошивок доступны на сайте www.ironlogic.ru. Последние версии прошивок на момент выпуска этого SDK находятся в папке "UpdateFW" в папке установки SDK.

Поддерживаемые конвертеры:

- Z-397 (мод. USB)
- Z-397 (мод. USB Guard)
- Z-397 IP - снят с производства
- Z-397 (мод. Web)

Поддерживаемые контроллеры:

- Gate 2000
- Guard Net
- Matrix-II Net
- Matrix-III Net
- Matrix-II Wi-Fi
- Matrix-VI NFC Net 8k
- Matrix-VI EHK Net 2k
- Matrix-VI Wi-Fi
- Matrix-II EH Web
- Matrix-VI EH Web
- Z-5R Net
- Z-5R Net 8000
- Z-5R Net 16k
- Z-5R Web

- Z-5R Web BT
- Z-5R Wi-Fi
- Z-5R Web mini
- Z-9 EHT Net
- EuroLock EHT net

Рекомендуемые версии прошивок для конвертеров:

Z-397 Guard - версия 3.3 или более поздняя;

Z-397 IP - заводская версия 1.01.60 или более поздняя, обновленная 2.01.130 или более поздняя;

Z-397 Web - заводская версия 3.0.48 или более поздняя.

Рекомендуемые версии прошивок для контроллеров:

Guard-Net - заводская версия 1.0.115 или более поздняя, для прошивок "Турникет", "Шлагбаум" и "Шлюз" версии V111 или более поздняя;

для остальных контроллеров - версия прошивки для SDK не имеет значения.

Что нового в ZGuard API

v3.39.4 (24.03.2023)

! не работало с Z-5R Net 16k через конвертер в режиме Advanced

v3.39.3 (28.02.2023)

! поиск возвращал неправильные параметры Z-5R Web BT
! не компилировались примеры Borland C++
+ поддержка контроллера Z-5R Net 16k

v3.39.2 (10.08.2022)

! исправлена ошибка, из-за которой у дескриптора контроллера мог оказаться неправильный сетевой адрес
* теперь можно создавать только один дескриптор для одного контроллера, при повторном вызове ZG_Ctr_Open вернёт код ZG_E_CTRISALREADYOPEN

v3.39.1 (12.04.2021)

+ поддержка длинных серийных номеров контроллеров (32-битный номер)

v3.38.1 (25.03.2021)

+ поддержка контроллеров: Z5R-WEB BT, Z5R Wi-Fi, Matrix-II EH Web, Matrix-VI EH Web, Z5R Web mini

v3.37.1 (28.07.2020)

! исправлена ошибка, из-за которой могли меняться сетевые адреса при обрывах связи с IP конвертером
* изменены функции ZG_Ctr_WriteKeys, ZG_Ctr_ClearKeys: удален параметр fUpdateTop

+ добавлены функции ZG_Ctr_SetKeyTopIndex, ZG_Ctr_GetConfigData, ZG_Ctr_SetConfigData, ZG_GetCtrConfigParams и ZG_SetCtrConfigParams
+ Delphi: добавлен пример "CtrConfig" для чтения/записи контроллера

v3.36.9 (07.04.2020)

* изменена ZG_Cvt_GetCtrVersion: если передан параметр pWS не равный null, то функция не пытается восстановить связь с контроллером

v3.36.8 (05.03.2020)

* уменьшен размер запроса (88 -> 48) контроллеру Guard-Net для версий прошивок >= v1.59

v3.36.7 (16.01.2020)

* в функциях управления противопожарным и охранным режимами убрана проверка поддержки этих режимов контроллером

v3.36.6 (25.10.2019)

* изменен алгоритм восстановления связи с IP конвертером: если конвертер не отвечает, то Sdk автоматически переподключается к конвертеру

v3.36.3 (23.10.2019)

* изменен алгоритм восстановления связи с IP конвертером
+ добавлены дополнительные сообщения в лог для IP конвертера в режиме Сервер

v3.36.2 (22.10.2019)

! исправлена функция обновления прошивки ip конвертера
! исправлена ошибка, замедляющая получение ответа от конвертера

v3.36.1 (19.09.2019)

- удалена устаревшая функция ZG_Ctr_CloseLock
+ добавлены функции ZG_Ctr_ReadApbTime, ZG_Ctr_WriteApbTime для установки времени антипассбэк
+ добавлены подтипы контроллера: `ZG_CS_ELECTROMAGNETIC, ZG_CS_ELECTROMECHANICAL, ZG_CS_MOTORTWORELAYS, ZG_CS_MOTORONERELAY, ZG_CS_ELECTRICAL, ZG_CS_AUTOCONTROL`
+ добавлены флаги контроллера: `ZG_CTR_F_APB, ZG_CTR_F_BIGTIME, ZG_CTR_F_EXTASK, ZG_CTR_F_DUALKEY, ZG_CTR_F_BOOTMODE`

v3.35.3 (19.06.2019)

! иногда возникала ошибка ZG_E_NOANSWER при подключении к ip конвертеру

v3.35.2 (11.12.2018)

+ добавлена справка на английском языке

v3.35.1 (08.11.2017)

! поиск ip-конвертеров всегда показывал, что они заняты
+ поддержка Matrix II Wi-Fi

v3.34.3 (30.08.2017)

! при автоматической смене сетевого адреса ip-конвертером не обновлялся адрес в дескрипторе контроллера

v3.34.2 (21.06.2017)

! функция ZG_Ctr_GetKeyTopIndex возвращала неправильное значение (65344 вместо 32672) для Guard-Net в режимах X2 + JOIN
! C#: в примерах был неправильный список названий моделей контроллеров (не хватало "Matrix III Net")

v3.34.1 (19.09.2016)

+ поддержка Matrix-III Net
+ Demo: добавлен русский язык интерфейса

v3.33.14 (08.08.2016)

! исправлен поиск IP-конвертеров: sdk пыталось опрашивать порт = 0

v3.33.13 (24.02.2016)

! VC++,BC++: исправлены заголовочные файлы: неправильно работали функции LoadZGuard, LoadZPort в случае ошибки LoadLibrary
- удалено автоматическое исправление указателей событий (когда адрес не кратен 8 или превышает размер банка событий)

v3.33.12 (30.01.2016)

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

v3.33.11 (25.01.2016)

! ZPort API: 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)
! ZPort API: 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

v3.33.10 (22.01.2016)

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

v3.33.9 (18.11.2015)

! исправлены функции ZG_Cvt_UpdateCtrFirmware и ZG_Ctr_UpdateFirmware

v3.33.8 (17.11.2015)

! исправлены функции ZG_Cvt_UpdateCtrFirmware и ZG_Ctr_UpdateFirmware
! исправлено возможное зависание при закрытии порта типа ZP_PORT_COM, ZP_PORT_FT или ZP_PORT_IP

v3.33.7 (29.10.2015)

! Z-5R Web: функции ZG_Ctr_SetFireMode, ZG_Ctr_GetFireInfo, ZG_Ctr_SetFireConfig, ZG_Ctr_SetSecurMode, ZG_Ctr_GetSecurInfo, ZG_Ctr_SetSecurConfig возвращали E_NOINTERFACE

v3.33.6 (26.05.2015)

+ добавлена утилита для проверки обновлений (Updater\ZUpdater.exe)

v3.33.5 (20.05.2015)

! исправлена ошибка при закрытии порта типа ZP_PORT_COM, ZP_PORT_FT, которая могла приводить:
а) к зависанию (в том числе при поиске устройств с помощью опроса портов);
б) к ошибке E_ACCESSDENIED при открытии порта;
* дополнена документация (ZGuard_RUS.chm).

v3.33.4 (15.05.2015)

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

v3.33.3 (14.05.2015)

! исправлена ошибка закрытия порта типа ZP_PORT_IP после разрыва соединения.

v3.33.2 (12.05.2015)

! не приходило уведомление об изменении статуса подключения устройства.

v3.33.1 (12.05.2015)

* функция ZG_EnumMessages заменена на ZG_GetNextMessage;
* функция ZG_Cvt_EnumMessages заменена на ZG_Cvt_GetNextMessage;
* функция ZG_Ctr_EnumMessages заменена на ZG_Ctr_GetNextMessage;

* изменен алгоритм обработки контроллеров в фоновом потоке конвертера (используется для уведомлений);
* заменен код ошибки ZP_E_OPENACCESS на E_ACCESSDENIED;
* структуры ZP_N_EXIST_INFO и ZP_N_CHANGE_INFO объединены в одну ZP_DDN_PORT_INFO;
* структуры ZP_N_EXIST_DEVINFO и ZP_N_CHANGE_DEVINFO объединены в одну ZP_DDN_DEVICE_INFO;
* переименована структура ZP_NOTIFY_SETTINGS в ZP_DD_NOTIFY_SETTINGS;
* переименована структура ZP_DETECTOR_SETTINGS в ZP_DD_GLOBAL_SETTINGS;
* переименована функция ZP_SetNotification -> ZP_DD_SetNotification;
* переименована функция ZP_GetNextMessage в ZP_DD_GetNextMessage;
* переименована функция ZP_GetDetectorSettings в ZP_DD_GetGlobalSettings;
* переименована функция ZP_SetDetectorSettings в ZP_DD_SetGlobalSettings;
* переименована функция ZP_UpdateDetector в ZP_DD_Refresh;
* переименована константа ZP_PF_NOWAITCONNECT в ZP_POF_NO_WAIT_CONNECT;
* переименована константа ZP_PF_NOCONNECTERR в ZP_POF_NO_CONNECT_ERR;
* переименована константа ZP_PF_NOUSBDETECT в ZP_POF_NO_DETECT_USB;
+ добавлена функция ZG_Cvt_GetConnectionStatus и уведомление ZG_CVTN_CONNECTION_CHANGE.

v3.32.5 (07.05.2015)

! ZPort: исправлены ошибки детектора (ZG_SetNotification);
! ZPort: исправлена ошибка при работе с ip-конвертером в режиме "Сервер";
! исправлена работа с Z-9 EHT Net;
* изменен алгоритм Zg_Ctr_Open и Zg_CloseHandle для дескриптора контроллера;
* изменен алгоритм ZG_Cvt_EnumControllers;
+ для портов типа ZP_PORT_COM и ZP_PORT_FT добавлена возможность автоматической восстановлении связи при переподключении usb (параметр ZG_CVT_OPEN_PARAMS.pWait.nRestorePeriod).

v3.32.4 (05.05.2015)

! ZPort: исправлены ошибки детектора (ZG_SetNotification);
! исправлен инсталлятор: при удалении оставались следы в реестре;
* изменено значение по умолчанию ZP_SCAN_MAXTRIES=2.

v3.32.3 (28.04.2015)

* теперь ZGuard.dll при поиске последовательных портов игнорирует com-порты, которые не удалось идентифицировать (т.е. не удалось определить с каким устройством связан порт). Чтобы не игнорировала, нужно использовать флаг ZP_NF_UNIDCOM (для ZG_SetNotification) или ZP_SF_UNIDCOM (для ZG_SearchDevices);
+ добавлена 64-битная версия Demo (в папке "x64").

v3.32.2 (27.04.2015)

! исправлены мелкие ошибки алгоритма поиска устройств;
! исправлена функция ZG_Cvt_GetCtrInfoLine для ip-конвертеров: в конце строки обрезался один символ;

* обновлены все примеры (папка Examples);
* обновлена документация (папка Help).

v3.32.1 (23.04.2015) бета

* теперь для подключения к конвертеру в режиме клиент требуется указывать TCP-порт для прослушки, внешний IP конвертера и серийный номер конвертера;
* изменен способ уведомлений: вместо callback-функции нужно использовать Event (объект синхронизации Windows) или специальное сообщение для отправки окну с помощью `PostMessage`;
- удалены функции `ZG_EnumSerialPorts`, `ZG_EnumConverters`, `ZG_FindConverter` и `ZG_EnumIpConverters` (вместо них `ZG_GetPortInfoList`, `ZG_GetPortInfo` и `ZG_SearchDevices`, `ZG_FindNextDevice`);
* переименованы все параметры `MaxTry` в `MaxTries`;
* обновлена утилита "ZRetr.exe" для работы с конвертерами Z-397 и Z-397 Web по TCP.

v3.31.1 (21.04.2015) бета

+ добавлены новые функции для поиска устройств/портов: `ZG_SearchDevices`, `ZG_FindNextDevice`, `ZG_GetPortInfoList`, `ZG_GetPortInfo`;
+ возможность получать список конвертеров в режиме CLIENT;
* полностью переписан детектор устройств, уведомляющий о подключении/отключении портов.

v3.30.1 (07.04.2015)

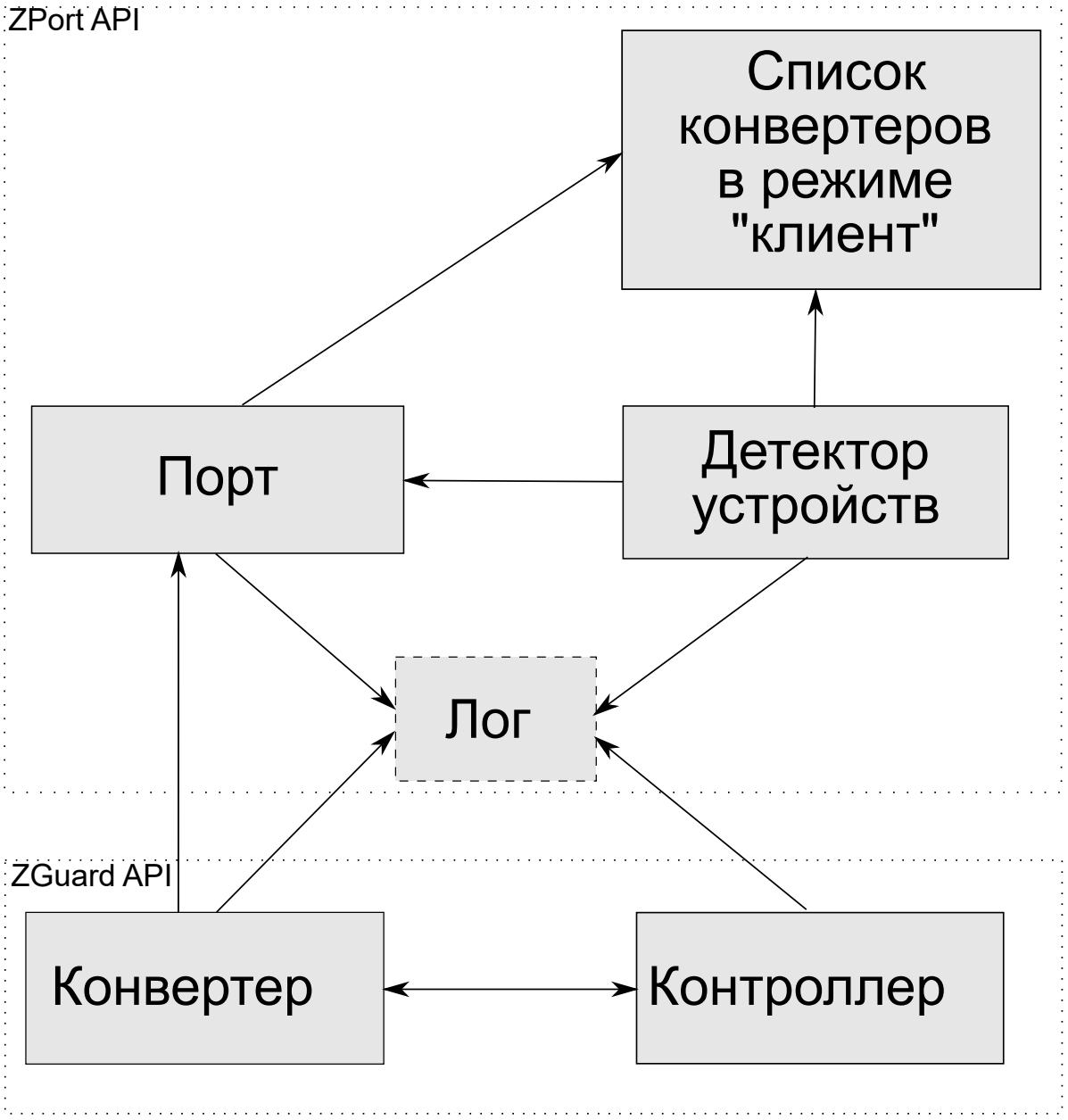
+ добавлен параметр "тайм-аут подключения к ip-конвертеру" (`ZP_WAIT_SETTINGS.nConnectTimeOut`);
* появилась возможность подключаться к ip-конвертеру в

режиме CLIENT без указания ip-адреса (нужно указать только TCP-порт, например, `PortName = ":25000"`).

v3.29.1 (03.02.2015)

+ поддержка Eurolock EHT net.

Объектная модель SDK



Конвертер

Объект "Конвертер" предназначен для работы с конвертером.

Имя функции	Кратное описание
ZG_UpdateCvtFirmware	Загружает новую прошивку в конвертер.
ZG_Cvt_UpdateFirmware	Загружает новую прошивку в конвертер.
ZG_Cvt_Open	Открывает конвертер.
ZG_Cvt_DettachPort	Отключается от конвертера, не закрывая порт, возвращает дескриптор порта, полученный функцией ZP_Open.
ZG_Cvt_GetConnectionStatus	Возвращает текущее состояние подключения к конвертеру.
ZG_Cvt_GetWaitSettings	Возвращает параметры ожидания исполнения функции.
ZG_Cvt_SetWaitSettings	Устанавливает параметры ожидания исполнения функции.
ZG_Cvt_GetInformation	Возвращает информацию о конвертере.
ZG_Cvt_SetNotification	Настраивает уведомления от конвертера.
ZG_Cvt_GetNextMessage	Возвращает следующее сообщение конвертера.
ZG_Cvt_GetScanCtrsState	Возвращает состояние фонового сканирования контроллеров.
ZG_Cvt_SetCapture	Блокирует фоновую работу SDK с конвертером.
ZG_Cvt_ReleaseCapture	Разблокирует фоновую работу SDK с конвертером.

ZG_Cvt_Clear	Очищает порт конвертера.
ZG_Cvt_Send	Отправляет запрос конвертеру.
ZG_Cvt_Receive	Возвращает ответ от конвертера.
ZG_Cvt_Exec	Отправляет запрос и возвращает ответ от конвертера.
ZG_Cvt_UpdateCtrFirmware	Загружает новую прошивку в контроллер.
ZG_Cvt_SetCtrAddrBySn	Устанавливает новый сетевой адрес контроллера по с/н.
ZG_Cvt_GetCtrInfoNorm	Возвращает информацию о контроллере по сетевому адресу.
ZG_Cvt_GetCtrInfoAdv	Возвращает информацию о контроллере по сетевому адресу (только в режиме Advanced).
ZG_Cvt_GetCtrInfoBySn	Возвращает информацию о контроллере по серийному номеру.
ZG_Cvt_GetCtrInfoLine	Возвращает информационную строку контроллера.
ZG_Cvt_GetCtrVersion	Возвращает версию контроллера.
Работа с конвертером в Advanced режиме	
ZG_Cvt_GetLicense	Возвращает информацию о лицензии конвертера (только в режиме Advanced).
ZG_Cvt_SetLicenseData	Устанавливает новую лицензию (только в режиме Advanced).
ZG_Cvt_ClearAllLicenses	Очищает все лицензии в конвертере (только в режиме Advanced).

ZG_Cvt_GetAllLicenses	Возвращает информацию о всех лицензиях, установленных в конвертер (только в режиме Advanced).
ZG_Cvt_GetShortInfo	Возвращает с/н и режим конвертера Guard (только в режиме Advanced).
ZG_Cvt_GetLongInfo	Возвращает информационные строки конвертера Guard (только в режиме Advanced).
Поиск контроллеров	
ZG_Cvt_EnumControllers	Перечисляет все подключенные контроллеры.
ZG_Cvt_FindController	Ищет контроллер по сетевому адресу и возвращает информацию о нем.
ZG_Cvt_SearchControllers	Инициализирует поиск контроллеров. Для получении информации о следующем найденном контроллере используйте функцию ZG_Cvt_FindNextController.
ZG_Cvt_FindNextController	Ищет следующий контроллер.

Контроллер

Объект "Контроллер" предназначен для работы с контроллером.

Имя функции	Кратное описание
ZG_Ctr_Open	Открывает контроллер.
ZG_Ctr_GetInformation	Возвращает информацию о контроллере.
ZG_Ctr_SetNotification	Настраивает уведомления от контроллера
ZG_Ctr_GetNextMessage	Возвращает следующее сообщение контроллера.
ZG_Cvt_SetCtrAddr	Устанавливает новый сетевой адрес контроллера.
ZG_Ctr_SetNewAddr	Устанавливает новый сетевой адрес контроллера.
ZG_Ctr_AssignAddr	Связывает дескриптор контроллера с новым сетевым адресом контроллера.
ZG_Ctr_UpdateFirmware	Загружает новую прошивку в контроллер.
ZG_Ctr_OpenLock	Открывает замок.
ZG_Ctr_ReadRegs	Читает регистры контроллера.
ZG_Ctr_ReadPorts	Читает состояние портов контроллера.
ZG_Ctr_ControlDevices	Управляет внешними устройствами.

ZG_Ctr_ReadData	Читает память контроллера (позволяет сделать образ памяти контроллера, которая распределена под времена замков, временные зоны, ключи, события и другое).
ZG_Ctr_WriteData	Пишет в память контроллера.
ZG_Ctr_ReadLockTimes	Читает времена для дверей контроллера.
ZG_Ctr_WriteLockTimes	Пишет времена для дверей контроллера.
ZG_Ctr_ReadTimeZones	Читает одну или несколько временных зон.
ZG_Ctr_WriteTimeZones	Пишет одну или несколько временных зон.
ZG_Ctr_EnumTimeZones	Перечисляет временные зоны в контроллере.
ZG_Ctr_ReadKeys	Читает один или несколько ключей.
ZG_Ctr_WriteKeys	Пишет один или несколько ключей.
ZG_Ctr_ClearKeys	Стирает один или несколько ключей.
ZG_Ctr_GetKeyTopIndex	Возвращает позицию верхней границы ключей.

ZG_Ctr_EnumKeys	Перечисляет ключи в контроллере.
ZG_Ctr_GetClock	Возвращает параметры часов контроллера.
ZG_Ctr_SetClock	Устанавливает параметры часов контроллера.
ZG_Ctr_ReadLastKeyNum	Читает номер последнего поднесенного ключа.
ZG_Ctr_ReadRTCState	Читает состояние контроллера: часы, указатели событий и номер последнего поднесенного ключа.
ZG_Ctr_ReadEventIdxs	Читает указатели событий.
ZG_Ctr_WriteEventIdxs	Устанавливает указатели событий.
ZG_Ctr_ReadEvents	Читает одно или несколько событий.
ZG_Ctr_EnumEvents	Перечисляет события контроллера.
ZG_Ctr_DecodePassEvent	Декодирует событие прохода.
ZG_Ctr_DecodeEcEvent	Декодирует событие ElectroControl (Управление элетропитанием).
ZG_Ctr_DecodeUnkKeyEvent	Декодирует событие ZG_EV_UNKNOWN_KEY (Неизвестный ключ).

ZG_Ctr_DecodeFireEvent	Декодирует событие ZG_EV_FIRE_STATE (Изменение состояния Пожара).
ZG_Ctr_DecodeSecurEvent	Декодирует событие ZG_EV_SECUR_STATE (Изменение состояния Охрана).
ZG_Ctr_EnableEmergencyUnlocking	Включает/выключает режим аварийного открывания дверей.
ZG_Ctr_IsEmergencyUnlockingEnabled	Определяет состояние режима аварийного открывания дверей.
ZG_Ctr_SetFireMode	Управление пожарным режимом по сети.
ZG_Ctr_GetFireInfo	Запрос состояния пожарного режима.
ZG_Ctr_SetFireConfig	Установка параметров пожарного режима.
ZG_Ctr_SetSecurMode	Управление режимом Охрана по сети.
ZG_Ctr_GetSecurInfo	Запрос состояния режима Охрана.
ZG_Ctr_SetSecurConfig	Установка параметров режима Охрана.
ZG_Ctr_SetCtrMode	Управление режимами контроллера.

ZG_Ctr_GetCtrModeInfo	Запрос состояния режима контроллера.
ZG_Ctr_ReadApbTime	Читает время антипассбэк.
ZG_Ctr_WriteApbTime	Пишет время антипассбэк в память контроллера.
Управление электопитанием "ElectroControl" (Matrix-II Net с прошивкой v3 и Matrix-III Net)	
ZG_Ctr_ReadElectroConfig	Читает параметры электропитания.
ZG_Ctr_WriteElectroConfig	Пишет новые параметры электропитания.
ZG_Ctr_GetElectroState	Возвращает статус электропитания.
ZG_Ctr_SetElectroPower	Включает / выключает электропитание.

Функции ZGuard API

Функции ZPort API.

Имя функции	Кратное описание
Инициализация библиотеки	
ZG_GetVersion	Возвращает номер текущей версии библиотеки.
ZG_Initialize	Инициализирует библиотеку.
ZG_Finalyze	Завершает работу библиотеки.
ZG_CloseHandle	Закрывает дескриптор.
Поиск конвертеров	
ZG_GetPortInfoList	Создает список последовательных портов. Для чтения элементов списка предназначена функция ZG_GetPortInfo.
ZG_GetPortInfo	Возвращает информацию о порте из списка, созданного функцией ZG_GetPortInfoList.
ZG_SearchDevices	Инициализирует поиск конвертеров. Для продолжения поиска предназначена функция ZG_FindNextDevice.
ZG_FindNextDevice	Ищет следующий конвертер.
ZG_GetProxyConverters	Возвращает список с/н конвертеров, подключенных к Proxy-серверу.
ZG_SetNotification	Настраивает уведомления о подключении/отключении устройств.
ZG_GetNextMessage	Возвращает следующее сообщение детектора, созданного с помощью функции ZG_SetNotification.
ZP_DD_SetGlobalSettings	Устанавливает новые настройки детектора, созданного функцией ZG_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.
Работа с конвертером	
ZG_UpdateCvtFirmware	Загружает новую прошивку в конвертер.
ZG_Cvt_UpdateFirmware	Загружает новую прошивку в конвертер.
ZG_Cvt_Open	Открывает конвертер.
ZG_Cvt_DettachPort	Отключается от конвертера, не закрывая порт, возвращает дескриптор порта, полученный функцией ZP_Open.
ZG_Cvt_GetConnectionStatus	Возвращает текущее состояние подключения к конвертеру.
ZG_Cvt_GetWaitSettings	Возвращает параметры ожидания исполнения функции.
ZG_Cvt_SetWaitSettings	Устанавливает параметры ожидания исполнения функции.
ZG_Cvt_GetInformation	Возвращает информацию о конвертере.
ZG_Cvt_SetNotification	Настраивает уведомления от конвертера.
ZG_Cvt_GetNextMessage	Возвращает следующее сообщение конвертера.
ZG_Cvt_GetScanCtrsState	Возвращает состояние фонового сканирования контроллеров.
ZG_Cvt_SetCapture	Блокирует фоновую работу SDK с конвертером.
ZG_Cvt_ReleaseCapture	Разблокирует фоновую работу SDK с конвертером.
Низкоуровневые функции для работы с конвертером	

Имя функции	Кратное описание
ZG_Cvt_Clear	Очищает порт конвертера.
ZG_Cvt_Send	Отправляет запрос конвертеру.
ZG_Cvt_Receive	Возвращает ответ от конвертера.
ZG_Cvt_Exec	Отправляет запрос и возвращает ответ от конвертера.
Работа с конвертером в Advanced режиме	
ZG_Cvt_GetLicense	Возвращает информацию о лицензии конвертера (только в режиме Advanced).
ZG_Cvt_SetLicenseData	Устанавливает новую лицензию (только в режиме Advanced).
ZG_Cvt_ClearAllLicenses	Очищает все лицензии в конвертере (только в режиме Advanced).
ZG_Cvt_GetAllLicenses	Возвращает информацию о всех лицензиях, установленных в конвертер (только в режиме Advanced).
ZG_Cvt_GetShortInfo	Возвращает с/н и режим конвертера Guard (только в режиме Advanced).
ZG_Cvt_GetLongInfo	Возвращает информационные строки конвертера Guard (только в режиме Advanced).
Поиск контроллеров	
ZG_Cvt_EnumControllers	Перечисляет все подключенные контроллеры.
ZG_Cvt_FindController	Ищет контроллер по сетевому адресу и возвращает информацию о нем.
ZG_Cvt_SearchControllers	Инициализирует поиск контроллеров. Для получении информации о следующем найденном контроллере используйте функцию ZG_Cvt_FindNextController.
ZG_Cvt_FindNextController	Ищет следующий контроллер.
Работа с контроллером	

Имя функции	Кратное описание
ZG_Cvt_UpdateCtrFirmware	Загружает новую прошивку в контроллер.
ZG_Cvt_SetCtrAddrBySn	Устанавливает новый сетевой адрес контроллера по с/н.
ZG_Cvt_GetCtrInfoNorm	Возвращает информацию о контроллере по сетевому адресу.
ZG_Cvt_GetCtrInfoAdv	Возвращает информацию о контроллере по сетевому адресу (только в режиме Advanced).
ZG_Cvt_GetCtrInfoBySn	Возвращает информацию о контроллере по серийному номеру.
ZG_Cvt_GetCtrInfoLine	Возвращает информационную строку контроллера.
ZG_Cvt_GetCtrVersion	Возвращает версию контроллера.
ZG_Ctr_Open	Открывает контроллер.
ZG_Ctr_GetInformation	Возвращает информацию о контроллере.
ZG_Ctr_SetNotification	Настраивает уведомления от контроллера
ZG_Ctr_GetNextMessage	Возвращает следующее сообщение контроллера.
ZG_Cvt_SetCtrAddr	Устанавливает новый сетевой адрес контроллера.
ZG_Ctr_SetNewAddr	Устанавливает новый сетевой адрес контроллера.
ZG_Ctr_AssignAddr	Связывает дескриптор контроллера с новым сетевым адресом контроллера.
ZG_Ctr_UpdateFirmware	Загружает новую прошивку в контроллер.
ZG_Ctr_OpenLock	Открывает замок.
ZG_Ctr_ReadRegs	Читает регистры контроллера.
ZG_Ctr_ReadPorts	Читает состояние портов контроллера.
ZG_Ctr_ControlDevices	Управляет внешними устройствами.
ZG_Ctr_ReadData	Читает память контроллера (позволяет сделать образ памяти

Имя функции	Кратное описание
	контроллера, которая распределена под времена замков, временные зоны, ключи, события и другое).
ZG_Ctr_WriteData	Пишет в память контроллера.
ZG_Ctr_ReadLockTimes	Читает времена для дверей контроллера.
ZG_Ctr_WriteLockTimes	Пишет времена для дверей контроллера.
ZG_Ctr_ReadTimeZones	Читает одну или несколько временных зон.
ZG_Ctr_WriteTimeZones	Пишет одну или несколько временных зон.
ZG_Ctr_EnumTimeZones	Перечисляет временные зоны в контроллере.
ZG_Ctr_ReadKeys	Читает один или несколько ключей.
ZG_Ctr_WriteKeys	Пишет один или несколько ключей.
ZG_Ctr_ClearKeys	Стирает один или несколько ключей.
ZG_Ctr_GetKeyTopIndex	Возвращает позицию верхней границы ключей.
ZG_Ctr_SetKeyTopIndex	Устанавливает позицию верхней границы ключей.
ZG_Ctr_EnumKeys	Перечисляет ключи в контроллере.
ZG_Ctr_GetClock	Возвращает параметры часов контроллера.
ZG_Ctr_SetClock	Устанавливает параметры часов контроллера.
ZG_Ctr_ReadLastKeyNum	Читает номер последнего поднесенного ключа.
ZG_Ctr_ReadRTCState	Читает состояние контроллера: часы, указатели событий и номер последнего поднесенного ключа.
ZG_Ctr_ReadEventIdxs	Читает указатели событий.
ZG_Ctr_WriteEventIdxs	Устанавливает указатели событий.

Имя функции	Кратное описание
ZG_Ctr_ReadEvents	Читает одно или несколько событий.
ZG_Ctr_EnumEvents	Перечисляет события контроллера.
ZG_Ctr_DecodePassEvent	Декодирует событие прохода.
ZG_Ctr_DecodeEcEvent	Декодирует событие ElectroControl (Управление элетропитанием).
ZG_Ctr_DecodeUnkKeyEvent	Декодирует событие ZG_EV_UNKNOWN_KEY (Неизвестный ключ).
ZG_Ctr_DecodeFireEvent	Декодирует событие ZG_EV_FIRE_STATE (Изменение состояния Пожара).
ZG_Ctr_DecodeSecurEvent	Декодирует событие ZG_EV_SECUR_STATE (Изменение состояния Охрана).
ZG_Ctr_EnableEmergencyUnlocking	Включает/выключает режим аварийного открывания дверей.
ZG_Ctr_IsEmergencyUnlockingEnabled	Определяет состояние режима аварийного открывания дверей.
ZG_Ctr_SetFireMode	Управление пожарным режимом по сети.
ZG_Ctr_GetFireInfo	Запрос состояния пожарного режима.
ZG_Ctr_SetFireConfig	Установка параметров пожарного режима.
ZG_Ctr_SetSecurMode	Управление режимом Охрана по сети.
ZG_Ctr_GetSecurInfo	Запрос состояния режима Охрана.
ZG_Ctr_SetSecurConfig	Установка параметров режима Охрана.
ZG_Ctr_SetCtrMode	Управление режимами контроллера.
ZG_Ctr_GetCtrModeInfo	Запрос состояния режима контроллера.
ZG_Ctr_ReadApbTime	Читает время антипассбэк.
ZG_Ctr_WriteApbTime	Пишет время антипассбэк в память контроллера.

Имя функции	Кратное описание
ZG_Ctr_GetConfigData	Возвращает двоичные данные конфигурации контроллера.
ZG_Ctr_SetConfigData	Устанавливает двоичные данные конфигурации контроллера.
ZG_GetCtrConfigParams	Извлекает значения параметров из двоичных данных конфигурации контроллера.
ZG_SetCtrConfigParams	Устанавливает значения параметров в двоичных данных конфигурации контроллера.
Прошивка "ElectroControl" (только Matrix II Net с FW v3.X)	
ZG_Ctr_ReadElectroConfig	Читает параметры электропитания.
ZG_Ctr_WriteElectroConfig	Пишет новые параметры электропитания.
ZG_Ctr_GetElectroState	Возвращает статус электропитания.
ZG_Ctr_SetElectroPower	Включает / выключает электропитание.
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.

ZG_GetVersion

Возвращает номер текущей версии библиотеки ZGuard.dll.

C++

```
DWORD ZG_GetVersion();
```

Delphi

```
function ZG_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.

ZG_Initialize

Инициализирует библиотеку.

C++

```
HRESULT ZG_Initialize(
        UINT nFlags
        );
```

Delphi

```
function ZG_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.
ZG_IF_LOG	Только для версии "Log". Записывать лог ZGuard.dll. По

Флаг	Описание
	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 `ZG_Finalyze`, чтобы правильно завершить работу потоков (thread) библиотеки до ее выгрузке с помощью функции `FreeLibrary`.

ZG_Finalyze

Завершает работу библиотеки. При этом автоматически закрываются все незакрытые дескрипторы.

C++

```
HRESULT ZG_Finalyze();
```

Delphi

```
function ZG_Finalyze(): HRESULT;
```

Параметры

Эта функция не имеет параметров.

Возвращаемое значение

Функция всегда возвращает `S_OK`.

ZG_CloseHandle

Закрывает дескриптор, который был открыт одной из функций: ZG_GetPortInfoList, ZG_SearchDevices, ZG_SetNotification, ZG_Cvt_Open, ZG_Ctr_Open.

C++

```
HRESULT ZG_CloseHandle(
        HANDLE hHandle
        );
```

Delphi

```
function ZG_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.

ZG_GetPortInfoList

Создает список последовательных портов конвертеров. Для доступа к элементам списка используйте функцию ZG_GetPortInfo.

C++

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

Delphi

```
function ZG_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.

ZG_GetPortInfo

Возвращает информацию о порте из списка портов.

C++

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

Delphi

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

Параметры

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

ZG_SearchDevices

Ищет конвертеры, опрашивая порты. В отличае от функции ZG_GetPortInfoList возвращает расширенную информацию о конвертерах (модель, с/н, версия прошивки).

C++

```
HRESULT ZG_SearchDevices(
        PHANDLE pHandle,
        PZP_SEARCH_PARAMS pParams,
        BOOL fSerial,
        BOOL fIP
        );
```

Delphi

```
function ZG_SearchDevices(
    var VHandle: THandle;
    var AParams: TZP_SEARCH_PARAMS;
    ASerial: Boolean;
    AIP: Boolean
    ): HRESULT;
```

Параметры

Handle
: [out] Дескриптор поиска.

Params
: [in] Параметры поиска.

Serial
: [in] True, если нужно опросить последовательные порты.

IP
: [in] True, если нужно опросить IP-конвертера по UDP.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Функция `ZG_SearchDevices` только инициализирует поиск конвертеров, сам поиск осуществляется функцией `ZG_FindNextDevice`.

ZG_FindNextDevice

Ищет следующий конвертер. Для завершения поиска конвертеров используйте функцию ZG_CloseHandle.

C++

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

Delphi

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

Параметры

Handle

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

Info

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

PortArr

[in,out] Буфер для информации о портах, ассоциированных с конвертером (при опросе по UDP у одного IP-конвертера будет 2 порта).

ArrLen

[in] Размер массива `PortArr`.

PortCount

[out] Количество портов, ассоциированных с конвертером.

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.

ZG_GetProxyConverters

Возвращает список с/н конвертеров, подключенных к Proxy-серверу.

C++

```
HRESULT ZG_GetProxyConverters(
        INT *pSnBuf,
        int nBufSize,
        LPINT pRCount,
        LPCWSTR pIpAddr,
        LPCSTR pActCode,
        PZP_WAIT_SETTINGS pWait
        );
```

Delphi

```
function ZG_GetProxyConverters(
    VSnBuf: PInteger;
    ABufSize: Integer;
    var VRCount: Integer;
    AIpAddr: PWideChar;
    AActCode: PAnsiChar;
    AWait: PZP_WAIT_SETTINGS
    ): HRESULT;
```

Параметры

SnBuf
[in,out] Буфер.
BufSize
[in] Размер буфера (максивальное количество элементов).
RCount
[out] Количество элементов, скопированных в буфер.
IpAddr

[in] ip-адрес или имя хоста proxy-сервера.

ActCode

[in] Код активации для proxy-сервера.

Wait

[in] Параметры ожидания ответа от конвертера. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_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 ZG_SetNotification(
        PHANDLE pHandle,
        PZP_DD_NOTIFY_SETTINGS pSettings,
        BOOL fSerial,
        BOOL fIP
        );
```

Delphi

```
function ZG_SetNotification(
    var VHandle: THandle;
    const ASettings: TZP_DD_NOTIFY_SETTINGS;
    ASerial, AIp: Boolean
    ): HRESULT;
```

Параметры

Handle
: [out] Возвращаемый дескриптор уведомителя.

Settings
: [in] Параметры уведомлений.

Serial
: [in] True, чтобы сканировать устройства, подключенные к последовательным портам.

Ip
: [in] True, чтобы сканировать IP-устройства.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_GetNextMessage

Возвращает следующее сообщение детектора, созданного с помощью функции ZG_SetNotification.

C++

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

Delphi

```
function ZG_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 ZG_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 ZG_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 ZG_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 ZG_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 ZG_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.

ZG_UpdateCvtFirmware

Устанавливает новую прошивку в конвертер.

C++

```
HRESULT ZG_UpdateCvtFirmware(
        PZG_CVT_OPEN_PARAMS pParams,
        LPCVOID pData,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_UpdateCvtFirmware(
    AParams: PZG_CVT_OPEN_PARAMS;
    AData: Pointer;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Params
: [in] Параметры открытия конвертера.

Data
: [in] Данные прошивки.

Count
: [in] Количество байт данных прошивки.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData

[in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_Open

Подключается к конвертеру.

C++

```
HRESULT ZG_Cvt_Open(
        PHANDLE pHandle,
        PZG_CVT_OPEN_PARAMS pParams,
        PZG_CVT_INFO pInfo,
        );
```

Delphi

```
function ZG_Cvt_Open(
    var VHandle: THandle;
    AParams: PZG_CVT_OPEN_PARAMS;
    VInfo: PZG_CVT_INFO;
    ): HRESULT;
```

Параметры

Handle
: [out] Дескриптор конвертера.

Params
: [in] Параметры открытия конвертера.

Info
: [out] Информация о конвертере. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Возвращает `ZP_E_OPENNOTEXIST` когда указано имя несуществующего порта.

Возвращает `E_ACCESSDENIED` когда порт занят другой программой.

Возвращает `ZG_E_NOCONVERTER` при подключении в режиме Proxy когда в сети нет конвертера с указанным с/н.

Возвращает `ZG_E_CVTBUSY` при подключении в режиме Proxy когда конвертер занят другим подключением.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Чтобы открыть IP-конвертер в режиме "Сервер" нужно установить `pParams.nType = ZP_PORT_IP`.

Чтобы открыть IP-конвертер в режиме "Клиент" нужно установить `pParams.nType = ZP_PORT_IPS`.

Чтобы открыть IP-конвертер в режиме "Прокси" нужно установить:

- `pParams.nType = ZP_PORT_IP`,
- `pParams.pActCode` = код активации,
- `pParams.nSn` = серийный номер конвертера.

Чтобы узнать список действующих конвертеров в режиме "Прокси" нужно использовать функцию `ZG_GetProxyConverters`.

Для завершения работы с конвертером нужно закрыть дескриптор `Handle` с помощью функции `ZG_CloseHandle`.

ZG_Cvt_DettachPort

Закрывает дескриптор конвертера, который был возвращен функцией ZG_Cvt_Open.

C++

```
HRESULT ZG_Cvt_DettachPort(
        HANDLE hHandle,
        PHANDLE pPortHandle
        );
```

Delphi

```
function ZG_Cvt_DettachPort(
    AHandle: THandle;
    var VPortHandle: THandle
    ): HRESULT;
```

Параметры

Handle
[in, out] Дескриптор конвертера.
PortHandle
[out] Дескриптор порта.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetConnectionStatus

Возвращает текущее состояние подключения к конвертеру.

C++

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

Delphi

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

Параметры

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

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetWaitSettings

Возвращает параметры ожидания исполнения функции.

C++

```
HRESULT ZG_Cvt_GetWaitSettings(
        HANDLE hHandle,
        PZP_WAIT_SETTINGS pSetting
        );
```

Delphi

```
function ZG_Cvt_GetWaitSettings(
    AHandle: THandle;
    var VSetting: TZP_WAIT_SETTING
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Setting
[out] Параметры выполнения функций.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_SetWaitSettings

Устанавливает параметры ожидания исполнения функции.

C++

```
HRESULT ZG_Cvt_SetWaitSettings(
        HANDLE hHandle,
        PZP_WAIT_SETTINGS pSetting
        );
```

Delphi

```
function ZG_Cvt_SetWaitSettings(
    AHandle: THandle;
    Const ASetting: TZP_WAIT_SETTING
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор считывателя.
Setting
[in] Параметры выполнения функций.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetInformation

Возвращает информацию о конвертере.

C++

```
HRESULT ZG_Cvt_GetInformation(
        HANDLE hHandle,
        PZG_CVT_INFO pInfo
        );
```

Delphi

```
function ZG_Cvt_GetInformation(
    AHandle: THandle;
    var VInfo: TZG_CVT_INFO
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Info
[out] Информация о конвертере.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_SetNotification

Настраивает уведомления от конвертера (о подключении/отключении контроллера, о изменении адреса контроллера). Функция создает поток (thread), который периодически опрашивает конвертер. Для временной приостановки/возобновления этого потока (thread) используйте функции ZG_Cvt_SetCapture и ZG_Cvt_ReleaseCapture.

C++

```
ZGUARD_API(HRESULT)
ZG_Cvt_SetNotification(HANDLE hHandle,
PZG_CVT_NOTIFY_SETTINGS pSettings);
```

Delphi

```
function ZG_Cvt_SetNotification(AHandle:
THandle; ASettings: PZG_CVT_NOTIFY_SETTINGS):
HRESULT; stdcall;
```

Параметры

Handle
[in] Дескриптор конвертера.
Settings
[in] Параметры уведомлений. Если равно **null**, то уведомления отключаются.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetNextMessage

Возвращает следующее сообщение конвертера.

C++

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

Delphi

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

Параметры

Handle
[in] Дескриптор конвертера.
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.

ZG_Cvt_GetScanCtrsState

Возвращает следующий адрес контроллера, сканируемого в потоке (thread) для отслеживания событий подключения/ отключения контроллеров (см. функцию ZG_Cvt_SetNotification).

C++

```
HRESULT ZG_Cvt_GetScanCtrsState(
        HANDLE hHandle,
        LPINT pNextAddr
        );
```

Delphi

```
function ZG_Cvt_GetScanCtrsState(
    AHandle: THandle;
    var VNextAddr: Integer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
NextAddr
[out] Адрес контроллера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_UpdateFirmware

Загружает новую прошивку в конвертер.

C++

```
HRESULT ZG_Cvt_UpdateFirmware(
        HANDLE hHandle,
        LPCVOID pData,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Cvt_UpdateFirmware(
    AHandle: THandle;
    AData: Pointer;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Type
[in] Данные прошивки.
Count
[in] Количество байт данных прошивки.
Callback
[in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.
UserData

[in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_SetCapture

Блокирует фоновую работу библиотеки с конвертером. Для возобновления фоновой работы используйте ZG_Cvt_ReleaseCapture. Фоновая работа инициируется функцией ZG_Cvt_SetNotification, при этом создается поток (thread), который периодически опрашивает конвертер.

C++

```
HRESULT ZG_Cvt_SetCapture(
        HANDLE hHandle
        );
```

Delphi

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

Параметры

Handle

[in] Дескриптор конвертера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_ReleaseCapture

Разблокирует фоновую работу библиотеки с конвертером, которая была блокирована функцией ZG_Cvt_SetCapture.

C++

```
HRESULT ZG_Cvt_ReleaseCapture(
        HANDLE hHandle
        );
```

Delphi

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

Параметры

Handle

[in] Дескриптор конвертера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_Clear

Очищает порт конвертера (RX).

C++

```
HRESULT ZG_Cvt_Clear(
        HANDLE hHandle
        );
```

Delphi

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

Параметры

Handle
[in] Дескриптор конвертера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_Send

Отправляет запрос конвертеру.

C++

```
HRESULT ZG_Cvt_Send(
        HANDLE hHandle,
        LPCVOID pData,
        INT nCount
        );
```

Delphi

```
function ZG_Cvt_Send(
    AHandle: THandle;
    Const AData;
    ACount: Integer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Data
[in] Данные запроса.
Count
[in] Количество байт в запросе.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_Receive

Возвращает ответ от конвертера.

C++

```
HRESULT ZG_Cvt_Receive(
        HANDLE hHandle,
        LPVOID pBuf,
        INT nBufSize,
        LPINT pCount
        );
```

Delphi

```
function ZG_Cvt_Receive(
    AHandle: THandle;
    var VBuf;
    ABufSize: Integer;
    var VCount: Integer
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор конвертера.

Buf
: [out] Буфер для ответа.

BufSize
: [in] Размер буфера.

Count
: [out] Количество байт ответа, скопированных в буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_Exec

Отправляет запрос и возвращает ответ от конвертера.

C++

```
HRESULT ZG_Cvt_Exec(
        HANDLE hHandle,
        LPCVOID pData,
        INT nCount,
        LPVOID pBuf,
        INT nBufSize,
        LPINT pRCount
        );
```

Delphi

```
function ZG_Cvt_Exec(
    AHandle: THandle;
    Const AData;
    ACount: Integer;
    var VBuf;
    ABufSize: Integer;
    var VRCount: Integer
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор конвертера.

Data
: [in] Данные запроса.

Count
: [in] Количество байт в запросе.

Buf
: [out] Буфер для ответа.

BufSize
: [in] Размер буфера.

RCount
: [out] Количество байт ответа, скопированных в буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetLicense

Возвращает информацию о лицензии конвертера Guard (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_GetLicense(
        HANDLE hHandle,
        BYTE nLicN,
        PZG_CVT_LIC_INFO pInfo
        );
```

Delphi

```
function ZG_Cvt_GetLicense(
    AHandle: THandle;
    ALicN: Byte;
    var VInfo: TZG_CVT_LIC_INFO
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор конвертера.

LicN
: [in] Номер лицензии. SDK использует лицензию №5.

Info
: [out] Информация о лицензии.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_SetLicenseData

Устанавливает новую лицензию (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_SetLicenseData(
        HANDLE hHandle,
        BYTE nLicN,
        LPCVOID pData,
        INT nCount,
        LPWORD pLicStatus=NULL
        );
```

Delphi

```
function ZG_Cvt_SetLicenseData(
    AHandle: THandle;
    ALicN: Byte;
    Const AData;
    ACount: Integer;
    VLicStatus: PWord=nil
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
LicN
[in] Номер лицензии. SDK использует лицензию №5.
Data
[in] Данные новой лицензии.
Count
[in] Количество байт лицензии.
LicStatus

[out] Статус лицензии. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_ClearAllLicenses

Очищает все лицензии в конвертере (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_ClearAllLicenses(
        HANDLE hHandle
        );
```

Delphi

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

Параметры

Handle

[in] Дескриптор конвертера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetAllLicenses

Возвращает информацию о всех лицензиях, установленных в конвертер (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_GetAllLicenses(
        HANDLE hHandle,
        PZG_CVT_LIC_SINFO pBuf,
        INT nBufSize,
        LPINT pCount
        );
```

Delphi

```
function ZG_Cvt_GetAllLicenses(
    AHandle: THandle;
    VBuf: PZG_CVT_LIC_SINFO;
    VBufSize: Integer;
    var VCount: Integer
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор конвертера.

pBuf
: [in,out] Буфер для информации о лицензиях. Может быть равен **null**.

BufSize
: [in] Количество элементов в буфере.

Count
: [out] Количество установленных лицензий.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetShortInfo

Возвращает с/н и режим конвертера Guard (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_GetShortInfo(
        HANDLE hHandle,
        INT *pSn,
        ZG_GUARD_MODE* pMode
        );
```

Delphi

```
function ZG_Cvt_GetShortInfo(
    AHandle: THandle;
    var VSn: Integer;
    var VMode: TZG_GUARD_MODE
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор конвертера.

Sn
: [out] Серийный номер конвертера.

Mode
: [out] Режим Guard.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetLongInfo

Возвращает информационные строки конвертера Guard (только в режиме Guard Advanced).

C++

```
HRESULT ZG_Cvt_GetLongInfo(
        HANDLE hHandle,
        INT *pSn,
        PWORD pVersion,
        ZG_GUARD_MODE* pMode,
        LPSTR pBuf,
        INT nBufSize,
        LPINT pLen
        );
```

Delphi

```
function ZG_Cvt_GetLongInfo(
    AHandle: THandle;
    VSn: PInteger;
    VVersion: PWord;
    VMode: PZG_GUARD_MODE;
    VBuf: PWideChar;
    ABufSize: Integer;
    VLen: PInteger
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор конвертера.

Sn

[out] Серийный номер конвертера.

Version

[out] Версия прошивки конвертера.

Mode

[out] Режим Guard.

Buf

[in,out] Буфер для информационных строк конвертера (не обязательный параметр).

BufSize

[in] Размер буфера (в символах).

Len

[out] Количество скопированных в буфер символов.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_EnumControllers

Перечисляет все подключенные контроллеры.

C++

```
HRESULT ZG_Cvt_EnumControllers(
        HANDLE hHandle,
        ZG_ENUMCTRSPROC pEnumProc,
        PVOID pUserData,
        UINT nFlags
        );
```

Delphi

```
function ZG_Cvt_EnumControllers(
    AHandle: THandle;
    AEnumProc: TZG_ENUMCTRSPROC;
    AUserData: Pointer;
    AFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
EnumProc
[in] Callback-функция.
Param
[in] Параметр передаваемый в Callback-функцию.
Flags
[in] Флаги:

Флаг	Значение	Описание
ZG_F_UPDATE	1	Обновить список сейчас. Если флаг установлен, то сканирование будет

Флаг	Значение	Описание
		выполнено сразу, иначе - будут возвращены результаты фонового сканирования, которое запускается с помощью ZG_Cvt_SetNotification.
ZG_F_REASSIGN	2	Переназначить конфликтующие адреса (не работает в режиме Advanced конвертера).
ZG_F_NOGATE	4	Не искать GATE-контроллеры (не искать все контроллеры, кроме Eurolock EHT net).
ZG_F_NOEUROLOCK	8	Не искать Eurolock EHT net.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Для **Z-397** и **Z-397 Guard** **Normal**: сканирование может производиться 2 способами: 1) сканирование всей сети сразу 2) последовательная проверка каждого адреса сети. Функция ZG_Cvt_EnumControllers использует всегда первый способ. Второй способ используется в фоновом потоке (thread) для уведомлений.

Для настройки уведомлений о подключении/отключении контроллеров используйте функцию ZG_Cvt_SetNotification.

ZG_Cvt_FindController

Ищет контроллер по сетевому адресу и возвращает информацию о нем. Для поиска всех контроллеров, подключенных к конвертеру, используйте ZG_Cvt_EnumControllers.

C++

```
HRESULT ZG_Cvt_FindController(
        HANDLE hHandle,
        BYTE nAddr,
        PZG_FIND_CTR_INFO pInfo,
        UINT nFlags,
        PZP_WAIT_SETTINGS pWait
        );
```

Delphi

```
function ZG_Cvt_FindController(
    AHandle: THandle;
    AAddr: Byte;
    var VInfo: TZG_FIND_CTR_INFO;
    AFlags: Cardinal;
    AWait: PZP_WAIT_SETTINGS
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Addr
[in] Сетевой адрес контроллера.
Info
[out] Информация о контроллере.
Flags

[in] Флаги:

Флаг	Значение	Описание
ZG_F_UPDATE	1	Обновить список сейчас. Если флаг установлен, то сканирование будет выполнено сразу, иначе - будут возвращены результаты фонового сканирования, которое запускается с помощью ZG_Cvt_SetNotification.
ZG_F_NOGATE	4	Не искать GATE-контроллеры (все контроллеры, кроме Eurolock EHT net).
ZG_F_NOEUROLOCK	8	Не искать Eurolock EHT net.

Wait

[in] Параметры ожидания ответа от конвертера. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_SearchControllers

Инициализирует поиск контроллеров. Для получении информации о следующем найденном контроллере используйте функцию ZG_Cvt_FindNextController.

C++

```
HRESULT ZG_Cvt_SearchControllers(
        HANDLE hHandle,
        INT nMaxCtrs,
        UINT nFlags
        );
```

Delphi

```
function ZG_Cvt_SearchControllers(
    AHandle: THandle;
    AMaxCtrs: Integer;
    AFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
MaxCtrs
[in] Предел количества найденных контроллеров.
Flags
[in] Флаги:

Флаг	Значение	Описание
ZG_CVSF_DETECTOR	1	Использовать готовый список найденных контроллеров детектора, который создается с помощью ZG_Cvt_SetNotification.

Флаг	Значение	Описание
ZG_CVSF_NOGATE	4	Не искать GATE-контроллеры (не искать все контроллеры, кроме замков).
ZG_CVSF_NOEUROLOCK	8	Не искать замки Eurolock EHT net и Z9 EHT Net.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Для настройки уведомлений о подключении/отключении контроллеров используйте функцию ZG_Cvt_SetNotification.

ZG_Cvt_FindNextController

Ищет следующий контроллер.

C++

```
HRESULT ZG_Cvt_FindNextController(
        HANDLE hHandle,
        PZG_FIND_CTR_INFO pInfo
        );
```

Delphi

```
function ZG_Cvt_FindNextController(
    AHandle: THandle;
    VInfo: PZG_FIND_CTR_INFO
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Info
[in,out] Буфер для информации о найденном контроллере. Если параметр равен **null**, то поиск завершается.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`. Если контроллеров больше нет, то возвращает `ZP_S_NOTFOUND`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Для завершения поиска и освобождения занятых ресурсов нужно вызвать `ZG_Cvt_FindNextController` с параметром `Info=null`. Также поиск автоматически завершается в следующих случаях:

- при возвращении функцией `ZG_Cvt_FindNextController` кода `ZP_S_NOTFOUND`;
- при закрытии дескриптора конвертера.

Если не завершить поиск, то занятые поиском ресурсы будут использованы повторно при инициализации нового поиска (`ZG_Cvt_SearchControllers`).

ZG_Cvt_UpdateCtrFirmware

Загружает новую прошивку в контроллер.

C++

```
HRESULT ZG_Cvt_UpdateCtrFirmware(
        HANDLE hHandle,
        WORD nCtrSn,
        LPCVOID pData,
        INT nCount,
        LPCSTR pszInfoStr,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        ZG_CTR_TYPE nModel
        );
```

Delphi

```
function ZG_Cvt_UpdateCtrFirmware(
    AHandle: THandle;
    ASn: Word;
    Const AData;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    AModel: TZG_CTR_TYPE
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор конвертера.
Sn
[in] С/н контроллера.
Data

[in] Данные прошивки.

Count

[in] Количество байт прошивки.

Callback

[in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData

[in] Параметр пользователя для Callback-функции.

Model

[in] Модель контроллера. Для **Eurolock EHT net** нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetCtrVersion

Возвращает версию контроллера.

C++

```
HRESULT ZG_Cvt_GetCtrVersion(
        HANDLE hHandle,
        BYTE nAddr,
        LPBYTE pVerData5,
        PZP_WAIT_SETTINGS pWS
        );
```

Delphi

```
function ZG_Ctr_GetVersion(
    AHandle: THandle;
    AAddr: Byte;
    VVerData5: PByte;
    WS: PZP_WAIT_SETTINGS
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
nAddr
[in] Сетевой адрес контроллера.
VerData5
[out] Буфер, размером 5 байт, для данных версии.
WS
[in] Параметры запроса (тайм-аут ожидания ответа).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Эту команду можно использовать для проверки подключения контроллера.

ZG_Cvt_SetCtrAddrBySn

Устанавливает новый сетевой адрес контроллера по с/н.

C++

```
HRESULT ZG_Cvt_SetCtrAddrBySn(
        HANDLE hHandle,
        WORD nSn,
        BYTE nNewAddr,
        ZG_CTR_TYPE nModel
        );
```

Delphi

```
function ZG_Cvt_SetCtrAddrBySn(
    AHandle: THandle;
    ASn: Word;
    ANewAddr: Byte;
    AModel: TZG_CTR_TYPE
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Sn
: [in] Серийный номер контроллера.

NewAddr
: [in] Новый адрес (0-254).

Model
: [in] Модель контроллера. Для Eurolock EHT net нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Конвертер Guard в режиме Advanced автоматически назначает сетевые адреса контроллерам.

ZG_Cvt_GetCtrInfoNorm

Возвращает информацию о контроллере по сетевому адресу.

C++

```
HRESULT ZG_Cvt_GetCtrInfoNorm(
        HANDLE hHandle,
        BYTE nAddr,
        PBYTE pTypeCode,
        INT *pSn,
        LPWORD pVersion,
        LPINT pInfoLines,
        PUINT pFlags,
        ZG_CTR_TYPE nModel
        );
```

Delphi

```
function ZG_Cvt_GetCtrInfoNorm(
    AHandle: THandle;
    AAddr: Byte;
    VTypeCode: PByte;
    VSn: PInteger;
    VVersion: PWord;
    VInfoLines: PInteger;
    VFlags: PCardinal;
    AModel: TZG_CTR_TYPE
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
nAddr
[in] Сетевой адрес контроллера.

TypeCode
: [out] Код типа контроллера. Может быть равно **null**.

Sn
: [out] Серийный номер контроллера. Может быть равно **null**.

Version
: [out] Версия контроллера. Может быть равно **null**.

InfoLines
: [out] Количество информационных строк контроллера. Может быть равно **null**.

Flags
: [out] Флаги контроллера. Может быть равно **0**.

Model
: [in] Модель контроллера. Для Eurolock EHT net нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetCtrInfoAdv

Возвращает информацию о контроллере по сетевому адресу.

C++

```
HRESULT ZG_Cvt_GetCtrInfoAdv(
        HANDLE hHandle,
        BYTE nAddr,
        PBYTE pTypeCode,
        INT *pSn,
        LPWORD pVersion,
        PUINT pFlags,
        PINT pEvWrIdx,
        PINT pEvRdIdx
        );
```

Delphi

```
function ZG_Cvt_GetCtrInfoAdv(
    AHandle: THandle;
    AAddr: Byte;
    VTypeCode: PByte;
    VSn: PInteger;
    VVersion: PWord;
    VFlags: PCardinal;
    VEvWrIdx: PInteger;
    VEvRdIdx: PInteger
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
nAddr
[in] Сетевой адрес контроллера.

TypeCode

[out] Код типа контроллера. Может быть равно **null**.

Код	Тип контроллера
0x24	Matrix-II-Net
0x25	Z5R-Net

Sn

[out] Серийный номер контроллера. Может быть равно **null**.

Version

[out] Версия контроллера. Может быть равно **null**.

Flags

[out] Флаги контроллера. Может быть равно **null**.

Бит	Описание
0-1	размер памяти (0x00 - 2Kb, 0x01 - 4Kb, 0x02 - 8Kb)
3	карты кодируются по стандарту Wiegand (Proximity)
7	есть новые события

EvWrIdx

[out] Адрес последнего записанного события. Может быть равно **null**.

EvRdIdx

[out] Адрес последнего считанного события. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetCtrInfoBySn

Возвращает информацию о контроллере по серийному номеру.

C++

```
HRESULT ZG_Cvt_GetCtrInfoBySn(
        HANDLE hHandle,
        INT nSn,
        PBYTE pTypeCode,
        LPBYTE pAddr,
        LPWORD pVersion,
        LPINT pInfoLines,
        PUINT pFlags,
        ZG_CTR_TYPE nModel
        );
```

Delphi

```
function ZG_Cvt_GetCtrInfoBySn(
    AHandle: THandle;
    ASn: Integer;
    VTypeCode: PByte;
    VAddr: PByte;
    VVersion: PWord;
    VInfoLines: PInteger;
    VFlags: PCardinal;
    AModel: TZG_CTR_TYPE
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Sn

[in] Серийный номер контроллера.

TypeCode

[out] Код типа контроллера. Может быть равно **null**.

Код	Тип контроллера
0x24	Matrix-II-Net
0x25	Z5R-Net

Addr

[out] Сетевой адрес контроллера. Может быть равно **null**.

Version

[out] Версия контроллера. Может быть равно **null**.

InfoLines

[out] Количество информационных строк контроллера. Может быть равно **null**.

Flags

[out] Флаги контроллера. Может быть равно **null**.

Model

[in] Модель контроллера. Для **Eurolock EHT net** нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Cvt_GetCtrInfoLine

Возвращает информационную строку контроллера.

C++

```
HRESULT ZG_Cvt_GetCtrInfoLine(
        HANDLE hHandle,
        INT nSn,
        INT nLineN,
        LPSTR pBuf,
        INT nBufSize,
        LPINT pLen=NULL,
        ZG_CTR_TYPE nModel=ZG_CTR_UNDEF
        );
```

Delphi

```
function ZG_Cvt_GetCtrInfoLine(
    AHandle: THandle;
    ASn: Integer;
    ALineN: Integer;
    VBuf: PAnsiChar;
    ABufSize: Integer;
    VLen: PInteger=nil;
    AModel: TZG_CTR_TYPE=ZG_CTR_UNDEF
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.

Sn
[in] Серийный номер контроллера.

LineN
[in] Номер информационной строки контроллера.

Buf
: [in, out] Буфер для строки (в кодировке AnsiChar).

BufSize
: [in] Размер буфера в символах.

Len
: [in] Количество скопированных символов в буфер. Может быть равно **null**.

Model
: [in] Модель контроллера. Для **Eurolock EHT net** нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_Open

Подключается к контроллеру.

C++

```
HRESULT ZG_Ctr_Open(
        PHANDLE pHandle,
        HANDLE hCvtHandle,
        BYTE nAddr,
        INT nSn,
        PZG_CTR_INFO pInfo,
        ZG_CTR_TYPE nModel
        );
```

Delphi

```
function ZG_Ctr_Open(
    var VHandle: THandle;
    ACvtHandle: THandle;
    AAddr: Byte;
    ASn: Integer;
    VInfo: PZG_CTR_INFO;
    AModel: TZG_CTR_TYPE
    ): HRESULT;
```

Параметры

Handle
: [out] Дескриптор контроллера.

CvtHandle
: [in] Дескриптор конвертера.

Addr
: [in] Сетевой адрес контроллера. Если =FFh, то используется Sn.

Sn

[in] Серийный номер контроллера.

Info

[out] Информация о контроллере (необязательный параметр).

Model

[in] Модель контроллера. Для Eurolock EHT net нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки. Возвращает `ZG_E_CTRISALREADYOPEN` (8004031Ah) если контроллер уже открыт, второй дескриптор для одного контроллера создавать нельзя.

ZG_Ctr_GetInformation

Возвращает информацию о контроллере.

C++

```
HRESULT ZG_Ctr_GetInformation(
        HANDLE hHandle,
        PZG_CTR_INFO pInfo
        );
```

Delphi

```
function ZG_Ctr_GetInformation(
    AHandle: THandle;
    var VInfo: TZG_CTR_INFO
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Info
[out] Информация о контроллере.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetNotification

Настраивает уведомления от контроллера (о новых событиях, о новых ключах, о рассинхронизации часов и др.). Функция создает (если еще не создан) для конвертера 1 поток (thread), который периодически опрашивает контроллер. Подробности смотрите в описании ZG_CTR_NOTIFY_SETTINGS.

C++

```
ZGUARD_API(HRESULT)
ZG_Ctr_SetNotification(HANDLE hHandle,
PZG_CTR_NOTIFY_SETTINGS pSettings);
```

Delphi

```
function ZG_Ctr_SetNotification(AHandle:
THandle; ASettings: PZG_CTR_NOTIFY_SETTINGS):
HRESULT; stdcall;
```

Параметры

Handle

[in] Дескриптор контроллера.

Settings

[in] Параметры уведомлений. Если равно **null**, то уведомления отключаются.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetNextMessage

Возвращает следующее сообщение контроллера.

C++

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

Delphi

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

Параметры

Handle
[in] Дескриптор контроллера.
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.

ZG_Ctr_SetNewAddr

Устанавливает новый сетевой адрес контроллера.

C++

```
HRESULT ZG_Ctr_SetNewAddr(
        HANDLE hHandle,
        BYTE nNewAddr
        );
```

Delphi

```
function ZG_Ctr_SetNewAddr(
    AHandle: THandle;
    ANewAddr: Byte
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
NewAddr
[in] Новый адрес (0-254).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Конвертер в режиме Advanced автоматически назначает сетевые адреса контроллерам.

ZG_Ctr_AssignAddr

Связывает дескриптор контроллера с новым сетевым адресом контроллера.

C++

```
HRESULT ZG_Ctr_AssignAddr(
        HANDLE hHandle,
        BYTE nAddr
        );
```

Delphi

```
function ZG_Ctr_AssignAddr(
    AHandle: THandle;
    AAddr: Byte
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Addr
[in] Адрес (0-254).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_UpdateFirmware

Загружает новую прошивку в контроллер.

C++

```
HRESULT ZG_Ctr_UpdateFirmware(
        HANDLE hHandle,
        LPCVOID pData,
        INT nCount,
        LPCSTR pszInfoStr,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Ctr_UpdateFirmware(
    AHandle: THandle;
    Const AData;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Data
[in] Данные прошивки.
Count
[in] Количество байт прошивки.
Callback
[in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
[in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadRegs

Читает регистры контроллера.

C++

```
HRESULT ZG_Ctr_ReadRegs(
        HANDLE hHandle,
        DWORD nAddr,
        INT nCount,
        LPVOID pBuf
        );
```

Delphi

```
function ZG_Ctr_ReadRegs(
    AHandle: THandle;
    AAddr: Cardinal;
    ACount: Integer;
    var VBuf
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Addr
[in] Адрес регистра.
Count
[in] Количество запрашиваемых байт.
Buf
[in,out] Буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadPorts

Читает состояние портов контроллера.

C++

```
HRESULT ZG_Ctr_ReadPorts(
        HANDLE hHandle,
        LPDWORD pData
        );
```

Delphi

```
function ZG_Ctr_ReadPorts(
    AHandle: THandle;
    var VData: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Data
[out] Состояние портов.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ControlDevices

Управляет внешними устройствами.

C++

```
HRESULT ZG_Ctr_ControlDevices(
        HANDLE hHandle,
        DWORD nDevType,
        BOOL fActive,
        DWORD nTimeMs
        );
```

Delphi

```
function ZG_Ctr_ControlDevices(
    AHandle: THandle;
    ADevType: Cardinal;
    AActive: LongBool;
    ATimeMs: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

DevType

[in] Тип устройства.

Тип устройства	Описание
ZG_DEV_RELE1	реле номер 1
ZG_DEV_RELE2	реле номер 2
ZG_DEV_SW3	силовой ключ SW3 (ОС) Конт.5 колодки К5
ZG_DEV_SW4	силовой ключ SW4 (ОС) Конт.5 колодки К6

Тип устройства	Описание
ZG_DEV_SW0	силовой ключ SW0 (ОС) Конт.1 колодки К4
ZG_DEV_SW1	силовой ключ SW1 (ОС) Конт.3 колодки К4
ZG_DEV_K65	слаботочный ключ (ОК) Конт.6 колодки К5
ZG_DEV_K66	слаботочный ключ (ОК) Конт.6 колодки К6

Active

[in] True, перевод в активное состояние, иначе - перевод в пассивное состояние.

TimeMs

[in] Время, на которое устройство переводится в новое состояние (от 1 до 255 000 мс). Если =0, то не используется. Если >255, то время округляется до целых секунд, если получается меньше секунды, то устанавливает 1 секунда.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadData

Читает память контроллера (позволяет сделать образ памяти контроллера, которая распределена под времена замков, временные зоны, ключи, события и другое).

C++

```
HRESULT ZG_Ctr_ReadData(
        HANDLE hHandle,
        INT nBankN,
        DWORD nAddr,
        INT nCount,
        LPVOID pBuf,
        LPINT pReaded,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Ctr_ReadData(
    AHandle: THandle;
    ABankN: Integer;
    AAddr: Cardinal;
    ACount: Integer;
    var VBuf;
    var VReaded: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

BankN
: [in] Номер банка (0-1).

Addr
: [in] Адрес памяти контроллера.

Count
: [in] Количество считываемых байт.

Buf
: [in,out] Буфер.

Readed
: [out] Количество прочитанных байт.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteData

Пишет в память контроллера.

C++

```
HRESULT ZG_Ctr_WriteData(
        HANDLE hHandle,
        INT nBankN,
        DWORD nAddr,
        LPCVOID pData,
        INT nCount,
        LPINT pWritten,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Ctr_WriteData(
    AHandle: THandle;
    ABankN: Integer;
    AAddr: Cardinal;
    Const AData;
    ACount: Integer;
    var VWritten: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
BankN
[in] Номер банка (0-1).

Addr
: [in] Адрес памяти контроллера.

Data
: [in] Данные для записи в память контроллера.

Count
: [in] Количество байт данных.

Written
: [out] Количество записанных байт.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadRTCState

Читает указатели событий и номер последнего поднесенного ключа.

C++

```
HRESULT ZG_Ctr_ReadRTCState(
        HANDLE hHandle,
        PZG_CTR_CLOCK pClock,
        LPINT pWrIdx,
        LPINT pRdIdx,
        Z_KEYNUM* pNum
        );
```

Delphi

```
function ZG_Ctr_ReadRTCState(
    AHandle: THandle;
    VClock: PZG_CTR_CLOCK;
    VWrIdx: PInteger;
    VRdIdx: PInteger;
    VNum: PZ_KEYNUM
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
VClock
[out] Часы контроллера. Может быть равно **null**.
WrIdx
[out] Указатель записи событий. Может быть равно **null**.
RdIdx
[out] Указатель чтения событий. Может быть равно **null**.
Num

[out] Номер ключа. Может быть равно **null**.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetConfigData

Возвращает двоичные данные конфигурации контроллера (Z-5R Net, Matrix II Wi-Fi, Z-5R Web).

C++

```
HRESULT ZG_Ctr_GetConfigData(
        HANDLE hHandle,
        LPVOID pBuf,
        UINT nBufSize,
        PUINT pReaded
        );
```

Delphi

```
function ZG_Ctr_GetConfigData(
    AHandle: THandle;
    var VBuf;
    BufSize_: Cardinal;
    var VReaded: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Buf
: [in,out] Буфер для данных конфигурации. Нужно 80 байт.

BufSize
: [in] Размер буфера в байтах.

Reaided
: [out] Количество байт конфигурации, записанных в буфер.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetConfigData

Устанавливает двоичные данные конфигурации контроллера (Z-5R Net, Matrix II Wi-Fi, Z-5R Web).

C++

```
HRESULT ZG_Ctr_SetConfigData(
        HANDLE hHandle,
        LPCVOID pData,
        UINT nDataSize
        );
```

Delphi

```
function ZG_Ctr_SetConfigData(
    AHandle: THandle;
    const AData;
    DataSize_: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Data
: [in] Данные конфигурации.

DataSize
: [in] Количество байт данных конфигурации.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_GetCtrConfigParams

Извлекает значения параметров из двоичных данных конфигурации контроллера.

C++

```
HRESULT ZG_GetCtrConfigParams(
        LPCVOID pCfgData,
        UINT nCfgDataSize,
        _ZG_CTR_CONFIGURATION *pParams
        );
```

Delphi

```
function ZG_GetCtrConfigParams(
    const CfgData_;
    CfgDataSize_: Cardinal;
    var VParams: TZG_CTR_CONFIGURATION
    ): HRESULT;
```

Параметры

CfgData
[in] Двоичные данные контроллера.
CfgDataSize
[in] Размер данных.
Params
[out] Извлечённые параметры конфигурации.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки. Возвращает *ZG_E_WRONGCRC* если контрольная

сумма *CfgData* не корректная и флаг *ZG_CTRCFGM_NOCHECKCRC* не установлен в *Params*.*Mask*.

ZG_SetCtrConfigParams

Устанавливает значения параметров в двоичных данных конфигурации контроллера.

C++

```
HRESULT ZG_SetCtrConfigParams(
        LPVOID pCfgData,
        UINT nCfgDataSize,
        _ZG_CTR_CONFIGURATION *pParams
        );
```

Delphi

```
function ZG_SetCtrConfigParams(
    var CfgData_;
    CfgDataSize_: Cardinal;
    const Params_: TZG_CTR_CONFIGURATION
    ): HRESULT;
```

Параметры

CfgData
[in,out] Двоичные данные контроллера.
CfgDataSize
[in] Размер данных.
Params
[in] Параметры конфигурации.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки. Возвращает *ZG_E_WRONGCRC* если контрольная

сумма *CfgData* не корректная и флаг *ZG_CTRCFGM_NOCHECKCRC* не установлен в *Params*.*Mask*.

Замок

Времена замков

Время открытия двери – время, на которое подается отпирающее напряжение в исполнительный механизм замка (*для электромагнитных это длительность обесточивания, для электромеханических - длительность импульса напряжения. Тип замка задается перемычкой на плате контроллера*). Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. Особым случаем является время, равное 0 – в этом случае замок блокируется и не может быть открыт ни ключом, ни кнопкой, ни командой извне до тех пор, пока время не будет изменено на любое ненулевое. При попытке открыть заблокированную дверь формируется событие «дверь заблокирована» (ZG_EV_KEY_DOOR_BLOCK) (если при этом использовался ключ из числа имеющихся в соответствующем банке ключей, его индекс помещается в соответствующее поле описателя события; если дверь пытались открыть кнопкой, поле индекса ключа в описателе события заполняется нулями).

Время контроля закрытой двери – определяет промежуток после открывания замка, в течение которого человек должен открыть дверь. Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. В момент открывания двери формируется соответствующее событие – «открыто ключом» (ZG_EV_OPEN_KEY) или «открыто кнопкой» (ZG_EV_BUT_OPEN). Если за это время дверь не была открыта, никаких событий не формируется, но при последующей попытке открыть дверь без ключа/кнопки (т.к. электромеханический замок остается открытым и после снятия напряжения, если не была открыта дверь) фиксируется событие «дверь взломана» (ZG_EV_NO_OPEN). Особым случаем является время, равное 0 – в этом случае контроль не производится и попытки взлома двери не обнаруживаются, а

события «открыто ключом», «открыто кнопкой» фиксируются в момент подачи напряжения на исполнительный механизм. Эта установка используется при отсутствии дверного геркона.

Время контроля открытой двери – определяет промежуток после открывания двери (0.1 – 25.5с), в течение которого дверь должна быть закрыта. Если по истечении этого времени дверь не была закрыта (что определяется по дверному геркону), фиксируется событие «дверь оставлена открытой» (ZG_EV_NO_CLOSE). Особым случаем является время, равное 0 – в этом случае контроль не производится. Эта установка используется при отсутствии дверного геркона.

Для чтения времен замков предназначена функция ZG_Ctr_ReadLockTimes, для записи – ZG_Ctr_WriteLockTimes.

Режим "Аварийное открывание дверей"

Этот режим используется при аварийном открывании дверей контроллера в случае пожара или иных аварийных ситуаций.

Для получения состояния предназначена функция `ZG_Ctr_IsEmergencyUnlockingEnabled`, для установки – `ZG_Ctr_EnableEmergencyUnlocking`.

ZG_Ctr_ReadLockTimes

Читает времена для дверей контроллера.

C++

```
HRESULT ZG_Ctr_ReadLockTimes(
        HANDLE hHandle,
        LPDWORD pOpenMs,
        LPDWORD pLetMs,
        LPDWORD pMaxMs,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_ReadLockTimes(
    AHandle: THandle;
    VOpenMs: PCardinal;
    VLetMs: PCardinal;
    VMaxMs: PCardinal;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

OpenMs
: [out] Время открытия двери (в миллисекундах).
Необязательный параметр.

LetMs
: [out] Время контроля закрытой двери (в миллисекундах).
Необязательный параметр.

MaxMs

[out] Время контроля открытой двери (в миллисекундах). Необязательный параметр.

BankN

[in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteLockTimes

Пишет времена для дверей контроллера.

C++

```
HRESULT ZG_Ctr_WriteLockTimes(
        HANDLE hHandle,
        DWORD AMask,
        DWORD nOpenMs,
        DWORD nLetMs,
        DWORD nMaxMs,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_WriteLockTimes(
    AHandle: THandle;
    AMask: Cardinal;
    AOpenMs: Cardinal;
    ALetMs: Cardinal;
    AMaxMs: Cardinal;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Mask

[in] Маска изменяемых параметров.

Бит 0 - Время открытия двери (`OpenMs`);

Бит 1 - Время контроля закрытой двери (`LetMs`);

Бит 2 - Время контроля открытой двери (`MaxMs`).

OpenMs

[in] Время открытия двери (в миллисекундах).

Время открывания замка, т.е. время, на которое подается отпирающее напряжение в исполнительный механизм замка. Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. При установке значения времени не кратного 0.1с SDK округляет его по слудующему правилу:
< 0.1 - пишет 0.1
0.1 >= x <= 25.5 - округляет значение (x + 0.05) до меньшего кратного 0.1 (f(0.15)=0.2, f(0.14)=0.1).
> 25.5 - пишет 25.5.

Особым случаем является время, равное 0 – в этом случае замок блокируется и не может быть открыт ни ключом, ни кнопкой, ни командой извне до тех пор, пока время не будет изменено на любое ненулевое. При попытке открыть заблокированную дверь формируется событие «дверь заблокирована» (ZG_EV_KEY_DOOR_BLOCK) (если при этом использовался ключ из числа имеющихся в соответствующем банке ключей, его индекс помещается в соответствующее поле описателя события; если дверь пытались открыть кнопкой, поле индекса ключа в описателе события заполняется нулями).

LetMs

[in] Время контроля закрытой двери (в миллисекундах).

Определяет промежуток после открывания замка, в течение которого человек должен открыть дверь. Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. При установке значения времени не кратного 0.1с **SDK** округляет его аналогично ранее описанному времени открывания замка.

В момент открывания двери формируется соответствующее событие – «открыто ключом» (ZG_EV_OPEN_KEY) или «открыто кнопкой» (ZG_EV_BUT_OPEN). Если за это время дверь не была открыта, никаких событий не формируется, но при последующей попытке открыть дверь без ключа/кнопки (т.к. электромеханический замок остается открытым и после снятия напряжения, если не была открыта дверь) фиксируется событие «дверь взломана» (ZG_EV_NO_OPEN).

Особым случаем является время, равное 0 – в этом случае контроль не производится и попытки взлома двери не обнаруживаются, а события «открыто ключом», «открыто кнопкой» фиксируются в момент подачи напряжения на исполнительный механизм. Эта установка используется при отсутствии дверного геркона.

MaxMs

[in] Время контроля открытой двери (в миллисекундах).

Определяет промежуток после открывания двери (0.1 – 25.5с), в течение которого дверь должна быть закрыта. При установке значения времени не кратного 0.1с **SDK** округляет его аналогично ранее описанному времени открывания замка.

Если по истечении этого времени дверь не была закрыта (что определяется по дверному геркону), фиксируется событие «дверь оставлена открытой» (ZG_EV_NO_CLOSE).

Особым случаем является время, равное 0 – в этом случае контроль не производится. Эта установка используется при отсутствии дверного геркона.

BankN

[in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_OpenLock

Открывает замок. Команда подает напряжение, открывающее замок, на время открытия, установленное функцией ZG_Ctr_WriteLockTimes.

C++

```
HRESULT ZG_Ctr_OpenLock(
        HANDLE hHandle,
        INT nLockN=0
        );
```

Delphi

```
function ZG_Ctr_OpenLock(
    AHandle: THandle;
    ALockN: Integer=0
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
LockN
[in] Номер замка.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_IsEmergencyUnlockingEnabled

Определяет состояние режима аварийного открывания дверей. Эта функция поддерживается только в контроллерах Matrix II Net, Z-5R Net и Z-5R Net 8000 с версией FW начиная с 2.5.

C++

```
HRESULT ZG_Ctr_IsEmergencyUnlockingEnabled(
        HANDLE hHandle,
        PBOOL pEnabled
        );
```

Delphi

```
function ZG_Ctr_IsEmergencyUnlockingEnabled(
    AHandle: THandle;
    var VEnable: LongBool
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
VEnable
[out] Состояние режима (True, включен, иначе - выключен).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_EnableEmergencyUnlocking

Включает/выключает режим аварийного открывания дверей. Эта функция поддерживается только в контроллерах Matrix II Net, Z-5R Net и Z-5R Net 8000 с версией FW начиная с 2.5.

C++

```
HRESULT ZG_Ctr_EnableEmergencyUnlocking(
        HANDLE hHandle,
        BOOL fEnable
        );
```

Delphi

```
function ZG_Ctr_EnableEmergencyUnlocking(
    AHandle: THandle;
    AEnable: LongBool
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
AEnable
[in] True, включить режим, иначе - выключить.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadApbTime

Читает время антипассбэк. Актуально, когда установлен флаг ZG_CTR_F_APB. Для записи времени антипассбэк используйте ZG_Ctr_WriteApbTime.

C++

```
HRESULT ZG_Ctr_ReadApbTime(
        HANDLE hHandle,
        UINT* pMinutes
        );
```

Delphi

```
function ZG_Ctr_ReadApbTime(
    AHandle: THandle;
    var Minutes: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Minutes
: [out] Время антипассбэк в минутах. Если =-1, то время не используется.

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteApbTime

Пишет время антипассбэк в память контроллера. Актуально, когда установлен флаг ZG_CTR_F_APB. Для чтения времени антипассбэк используйте ZG_Ctr_ReadApbTime.

C++

```
HRESULT ZG_Ctr_WriteApbTime(
        HANDLE hHandle,
        UINT nMinutes
        );
```

Delphi

```
function ZG_Ctr_WriteApbTime(
    AHandle: THandle;
    Minutes: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Minutes
: [in] Время антипассбэк в минутах. Если =-1, то время не используется..

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Расписание

Временные зоны предназначены для организации пропуска по ключам только в заданные промежутки времени и в определенные дни недели. Контроллер имеет 7 временных зон (константа `ZG_MAX_TIMEZONES`), исключением является прошивка ElectroControl - 6 временных зон.

Для каждой зоны можно задать период времени когда разрешен проход:

- список дней недели (понедельник - воскресенье),
- время начала и конца прохода.

Каждому ключу можно назначить одну или несколько временных зон.

Временная зона ограничивает доступ в течение недели. Чтобы ограничивать по дням, месяцам, и т.д., нужно программно переписывать временную зону и (или) параметр `Access` ключа.

Для чтения временных зон предназначены функции `ZG_Ctr_ReadTimeZones`, `ZG_Ctr_EnumTimeZones`, для записи – `ZG_Ctr_WriteTimeZones`.

Временная зона для прошивки "ElectroControl" (для Matrix II Net FW v3.X)

Проверить поддерживается ли ElectroControl можно с помощью флага `ZG_CTR_F_ELECTRO` (`ZG_Cvt_FindController`, `ZG_Cvt_EnumControllers`, `ZG_Ctr_Open`, `ZG_Ctr_GetInformation`).

Для хранения параметров временной зоны резервируется зона с индексом 6 из основного списка.

Для работы с временной зоной предназначены функции `ZG_Ctr_ReadElectroConfig` и `ZG_Ctr_WriteElectroConfig`.

Второй набор временных зон

Проверить поддерживается ли второй набор зон можно с помощью флага `ZG_CTR_F_DUAL_ZONE` (`ZG_Cvt_FindController`, `ZG_Cvt_EnumControllers`, `ZG_Ctr_Open`, `ZG_Ctr_GetInformation`).

Для работы с временными зонами предназначены функции: `ZG_Ctr_ReadTimeZones` и `ZG_Ctr_WriteTimeZones`, индекс первой зоны равен -9 (константа `ZG_DUAL_ZONE_TZ0`), всего зон 7 (константа `ZG_MAX_TIMEZONES`).

Временные зоны для режимов контроллера

Проверить поддерживается ли режимы контроллера можно с помощью флага `ZG_CTR_F_MODES` (`ZG_Cvt_FindController`, `ZG_Cvt_EnumControllers`, `ZG_Ctr_Open`, `ZG_Ctr_GetInformation`).

Для работы с временными зонами предназначены функции: `ZG_Ctr_ReadTimeZones` и `ZG_Ctr_WriteTimeZones`, индекс первой зоны равен -2 (константа `ZG_MODES_TZ0`), всего зон 2.

ZG_Ctr_ReadTimeZones

Читает одну или несколько временных зон.

C++

```
HRESULT ZG_Ctr_ReadTimeZones(
        HANDLE hHandle,
        INT nIdx,
        PZG_CTR_TIMEZONE pBuf,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_ReadTimeZones(
    AHandle: THandle;
    AIdx: Integer;
    VBuf: PZG_CTR_TIMEZONE;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Idx

[in] Номер первой временной зоны, считываемой из контроллера. Максимальное количество временных зон равно 7 (константа ZG_MAX_TIMEZONES).

Buf
: [in,out] Буфер для временных зон.

Count
: [in] Количество временных зон, которые нужно получить. Должно быть не больше размера буфера.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Временная зона ограничивает доступ в течение недели (контроллер имеет 7 временных зон, для каждой из которых можно задать: список дней недели, для которых разрешен проход и время начала и конца прохода), а если нужно ограничивать по дням, месяцам, и т.д., то нужно программно переписывать временную зону и(или) параметр ZG_CTR_KEY.nTZMask ключа.

ZG_Ctr_WriteTimeZones

Пишет одну или несколько временных зон.

C++

```
HRESULT ZG_Ctr_WriteTimeZones(
        HANDLE hHandle,
        INT nIdx,
        PZG_CTR_TIMEZONE pTzs,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_WriteTimeZones(
    AHandle: THandle;
    AIdx: Integer;
    ATzs: PZG_CTR_TIMEZONE;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Idx

[in] Номер первой временной зоны, записываемой в контроллер.

Tzs

[in] Массив временных зон.

Count

[in] Количество временных зон. Максимальное количество временных зон равно 7 (константа ZG_MAX_TIMEZONES)

Callback

[in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData

[in] Параметр пользователя для Callback-функции.

BankN

[in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_EnumTimeZones

Перечисляет временные зоны в контроллере.

C++

```
HRESULT ZG_Ctr_EnumTimeZones(
        HANDLE hHandle,
        INT nStart,
        ZG_ENUMCTRTIMEZONESPROC fnEnumProc,
        PVOID pUserData,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_EnumTimeZones(
    AHandle: THandle;
    AStart: Integer;
    AEnumProc: TZG_ENUMCTRTIMEZONESPROC;
    AUserData: Pointer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.

Start
[in] Номер первой временной зоны, считываемой из контроллера.

EnumProc
[in] Callback-функция, возвращающая параметры для каждой временной зоны.

UserData
[in] Параметр пользователя для Callback-функции.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Часы контроллера

Текущие дата и время контроллера. Используются для проверки расписаний и при записи новых событий.

Для получения часов предназначены функции `ZG_Ctr_GetClock`, `ZG_Ctr_ReadRTCState`, для установки – `ZG_Ctr_SetClock`.

ZG_Ctr_GetClock

Возвращает параметры часов контроллера.

C++

```
HRESULT ZG_Ctr_GetClock(
        HANDLE hHandle,
        PZG_CTR_CLOCK pClock
        );
```

Delphi

```
function ZG_Ctr_GetClock(
    AHandle: THandle;
    var VClock: TZG_CTR_CLOCK
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Clock
[out] Параметры часов.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetClock

Устанавливает параметры часов контроллера.

C++

```
HRESULT ZG_Ctr_SetClock(
        HANDLE hHandle,
        PZG_CTR_CLOCK pClock
        );
```

Delphi

```
function ZG_Ctr_SetClock(
    AHandle: THandle;
    Const AClock: TZG_CTR_CLOCK
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Clock
[in] Параметры часов.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Ключи

Список ключей предназначен для хранения разрешенных для прохода ключей, а также для сервисных ключей (блокирующих- и мастер-ключей). Список представляет собой массив фиксированной длины. Каждый элемент массива может быть занят ключом или может быть свободным (вакантная пустая ячейка). При удалении ключа вначале массива, остальные ключи не смещаются. Максимальное количество ключей можно узнать с помощью функции `ZG_Ctr_GetInformation`.

Для каждого ключа можно задать:

- RFID-номер карты;
- тип (обычный, мастер или блокирующий);
- доступ (всегда разрешен, всегда запрещен или по расписанию – набор временных зон).

Для чтения ключей предназначены функции `ZG_Ctr_ReadKeys`, `ZG_Ctr_EnumKeys`, для записи – `ZG_Ctr_WriteKeys`.

Верхняя граница ключей

Верхняя граница ключей – это переменная, которая хранится в контроллере, она определяет позицию, начиная с которой идут только свободные ячейки ключей. Контроллер обновляет эту переменную автоматически при записи или стирании ключей. При поднесении ключа к считывателю контроллер ищет его номер в диапазоне от 0 до "верхней границы ключей" (не включительно).

Для чтения верхней границы ключей предназначена функция `ZG_Ctr_GetKeyTopIndex`.

Номер последнего поднесенного ключа

Контроллер хранит в своей памяти номер ключа, который был поднесен к его считывателю последним.

Для чтения номера предназначены функции ZG_Ctr_ReadLastKeyNum, ZG_Ctr_ReadRTCState.

ZG_Ctr_ReadKeys

Читает один или несколько ключей.

C++

```
HRESULT ZG_Ctr_ReadKeys(
        HANDLE hHandle,
        INT nIdx,
        PZG_CTR_KEY pBuf,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_ReadKeys(
    AHandle: THandle;
    AIdx: Integer;
    VBuf: PZG_CTR_KEY;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Idx

[in] Номер первой карты, считываемой из контроллера. Максимальное количество карт можно определить с помощью функции ZG_Ctr_GetInformation.

Buf
: [in,out] Буфер для карт.

Count
: [in] Количество карт, которые нужно получить. Должно быть не больше размера буфера.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteKeys

Пишет одну или несколько временных зон.

C++

```
HRESULT ZG_Ctr_WriteKeys(
        HANDLE hHandle,
        INT nIdx,
        PZG_CTR_KEY pKeys,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        INT nBankN = 0
        );
```

Delphi

```
function ZG_Ctr_WriteKeys(
    AHandle: THandle;
    AIdx: Integer;
    AKeys: PZG_CTR_KEY;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    ABankN: Integer = 0
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Idx

[in] Номер первой карты, записываемой в контроллер. Если =-1, то используется значение верхней границы ключей (ZG_Ctr_GetKeyTopIndex).

Keys
: [in] Массив карт.

Count
: [in] Количество карт.

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Верхняя граница ключей автоматически обновляется контроллером после завершения процесса записи. Чтобы вручную установить границу используйте функцию `ZG_Ctr_SetKeyTopIndex`, но эту функцию поддерживают не все контроллеры, и контроллер всё равно автоматически переустановит границу. Для получения значения верхней границы ключей используйте функцию `ZG_Ctr_GetKeyTopIndex`.

ZG_Ctr_ClearKeys

Стирает один или несколько ключей.

C++

```
HRESULT ZG_Ctr_ClearKeys(
        HANDLE hHandle,
        INT nIdx,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData,
        INT nBankN = 0
        );
```

Delphi

```
function ZG_Ctr_ClearKeys(
    AHandle: THandle;
    AIdx: Integer;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer;
    ABankN: Integer = 0
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Idx
: [in] Номер первой стираемой карты.

Count
: [in] Количество стираемых карт. Если равен `7FFFFFFFh (MAXINT)`, то используется значение, равное (*Верхняя граница ключей (ZG_Ctr_GetKeyTopIndex) - Idx*).

Callback
: [in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData
: [in] Параметр пользователя для Callback-функции.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Для стирания ключа также может использоваться `ZG_Ctr_WriteKeys`, для этого нужно в структуре ZG_CTR_KEY установить флаг fErased = True .

ZG_Ctr_GetKeyTopIndex

Возвращает позицию верхней границы ключей. В этой позиции и далее находятся стертые ключи.

C++

```
HRESULT ZG_Ctr_GetKeyTopIndex(
        HANDLE hHandle,
        LPINT pIdx,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_GetKeyTopIndex(
    AHandle: THandle;
    var VIdx: Integer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Idx
: [out] Значение индекса.

BankN
: [in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Позиция верхней границы ключей указывает на позицию за последним записанным ключем, кроме случая когда верхняя граница равна 0, тогда это означает, что все ключи стерты.

До верхней границы также могут встречаться стертые ключи, поэтому верхняя граница не всегда является лучшим местом вставки.

При стирании ключей верхняя граница обновляется (уменьшается) не сразу, контроллер автоматически обновляет ее через некоторое время.

ZG_Ctr_SetKeyTopIndex

Устанавливает позицию верхней границы ключей. С этой позиции должны находиться стертые ключи.

C++

```
HRESULT ZG_Ctr_SetKeyTopIndex(
        HANDLE hHandle,
        INT nIdx,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_SetKeyTopIndex(
    AHandle: THandle;
    Idx_: Integer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Idx
[in] Значение индекса.
BankN
[in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Позиция верхней границы ключей указывает на позицию за последним записанным ключем, кроме случая когда верхняя граница равна 0, тогда это означает, что все ключи стерты.

До верхней границы также могут встречаться стертые ключи, поэтому верхняя граница не всегда является лучшим местом вставки.

При стирании ключей верхняя граница обновляется (уменьшается) не сразу, контроллер автоматически обновляет ее через некоторое время.

Не все контроллеры поддерживают эту команду.

ZG_Ctr_EnumKeys

Перечисляет ключи в контроллере.

C++

```
HRESULT ZG_Ctr_EnumKeys(
        HANDLE hHandle,
        INT nStart,
        ZG_ENUMCTRKEYSPROC fnEnumProc,
        PVOID pUserData,
        INT nBankN=0
        );
```

Delphi

```
function ZG_Ctr_EnumKeys(
    AHandle: THandle;
    AStart: Integer;
    AEnumProc: TZG_ENUMCTRKEYSPROC;
    AUserData: Pointer;
    ABankN: Integer=0
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Start
[in] Номер первого ключа, считываемого из контроллера.
EnumProc
[in] Callback-функция, возвращающая параметры для каждого ключа.
UserData
[in] Параметр пользователя для Callback-функции.
BankN

[in] Номер банка (0-1).

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadLastKeyNum

Читает номер последнего поднесенного ключа.

C++

```
HRESULT ZG_Ctr_ReadLastKeyNum(
        HANDLE hHandle,
        Z_KEYNUM* pNum
        );
```

Delphi

```
function ZG_Ctr_ReadLastKeyNum(
    AHandle: THandle;
    var VNum: TZ_KEYNUM
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Num
[out] Номер ключа.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

События

Список событий предназначен для хранения информации о проходах и о других событиях. Список представляет собой массив фиксированной длины. Когда массив заполняется, то запись продолжается сначала, переписывая самые старые события. Максимальное количество событий можно узнать с помощью функции ZG_Ctr_GetInformation. Возможные типы событий можно посмотреть здесь.

Для чтения событий предназначены функции ZG_Ctr_ReadEvents, ZG_Ctr_EnumEvents.

Указатель записи и указатель чтения

Указатель записи – позиция, в которую контроллер пишет следующее событие. После записи очередного события, контроллер автоматически меняет этот указатель.

Указатель чтения – позиция следующего непрочитанного события. Если при записи нового события указатель записи становиться равным указателю чтения, то контроллер автоматически меняет указатель чтения (=указатель записи+1).

Количество новых событий можно вычислять по формуле: если *ReadEvIdx* < *WriteEvIdx*, то *NewCount* = *(WriteEvIdx - ReadEvIdx*), иначе *NewCount* = (*MaxEvents - ReadEvIdx + WriteEvIdx)*, где *NewCount – количество новых событий, WriteEvIdx – указатель записи, ReadEvIdx* – указатель чтения, *MaxEvents* – максимальное количество событий (можно узнать с помощью функции `ZG_Ctr_GetInformation`).

Для чтения указателей предназначены функции `ZG_Ctr_ReadEventIdxs`, `ZG_Ctr_ReadRTCState`, для записи – `ZG_Ctr_WriteEventIdxs`.

ZG_Ctr_ReadEventIdxs

Читает указатели событий.

C++

```
HRESULT ZG_Ctr_ReadEventIdxs(
        HANDLE hHandle,
        LPINT pWrIdx,
        LPINT pRdIdx
        );
```

Delphi

```
function ZG_Ctr_ReadEventIdxs(
    AHandle: THandle;
    var VWrIdx: Integer;
    var VRdIdx: Integer
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

WrIdx
: [out] Указатель записи событий.

RdIdx
: [out] Указатель чтения событий.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteEventIdxs

Устанавливает указатели событий.

C++

```
HRESULT ZG_Ctr_WriteEventIdxs(
        HANDLE hHandle,
        UINT nMask,
        INT nWrIdx,
        INT nRdIdx
        );
```

Delphi

```
function ZG_Ctr_WriteEventIdxs(
    AHandle: THandle;
    AMask: Cardinal;
    AWrIdx, ARdIdx: Integer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Mask
[in] Маска изменяемых параметров.
Бит 0 - индекс записи;
Бит 1 - индекс чтения.
WrIdx
[in] Указатель записи событий.
RdIdx
[in] Указатель чтения событий.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_ReadEvents

Читает одно или несколько событий.

C++

```
HRESULT ZG_Ctr_ReadEvents(
        HANDLE hHandle,
        INT nIdx,
        PZG_CTR_EVENT pBuf,
        INT nCount,
        ZG_PROCESSCALLBACK pfnCallback,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Ctr_ReadEvents(
    AHandle: THandle;
    AIdx: Integer;
    VBuf: PZG_CTR_EVENT;
    ACount: Integer;
    ACallback: TZG_PROCESSCALLBACK;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Idx
[in] Позиция первого считываемого события.
Buf
[in,out] Буфер для событий.
Count

[in] Количество событий, которые нужно получить. Должно быть не больше размера буфера.

Callback

[in] Callback-функция для возможности показа статуса выполнения и возможности прерывания процесса.

UserData

[in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Буфер событий контроллера имеет фиксированный размер `ZG_CTR_INFO.nMaxEvents`. Он организован циклически, индекс следующего записываемого события и индекс следующего читаемого события храняться в памяти контроллера. При инициализации контроллера оба индекса устанавливаются в 0. При очередном событии индекс записи увеличивается на 1 и блок информации о событии заносится в буфер, а индекс чтения не изменяется до тех пор, пока буфер не заполняется до конца и индекс записи установиться в 0 - в этом случае индекс чтения также увеличивается на 1; далее при записи нового события будут увеличиваться на 1 и индекс записи, и индекс чтения; таким образом в буфере всегда останутся `ZG_CTR_INFO.nMaxEvents` последних событий.

Для определения указателя записи и указателя чтения предназначены функции: `ZG_Ctr_ReadEventIdxs`, `ZG_Ctr_ReadRTCState`. Функция `ZG_Ctr_ReadEvents` не проверяет существование записанных событий в буфере и не

смещает индекс чтения (это можно сделать вручную с помощью функции ZG_Ctr_WriteEventIdxs).

В событиях ZG_EV_KEY_OPEN, ZG_EV_KEY_ACCESS и ZG_EV_KEY_DOOR_BLOCK хранится индекс ключа в банке ключей, номер банка соответствует направлению прохода (0 - вход, 1 - выход).

ZG_Ctr_EnumEvents

Перечисляет события контроллера.

C++

```
HRESULT ZG_Ctr_EnumEvents(
        HANDLE hHandle,
        INT nStart,
        INT nCount,
        ZG_ENUMCTREVENTSPROC fnEnumProc,
        PVOID pUserData
        );
```

Delphi

```
function ZG_Ctr_EnumEvents(
    AHandle: THandle;
    AStart: Integer;
    ACount: Integer;
    AEnumProc: TZG_ENUMCTREVENTSPROC;
    AUserData: Pointer
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

Start

[in] Позиция первого считываемого события. Если равно -1, то используется указатель чтения.

Count

[in] Количество перечисляемых событий. Если равно -1, то используется количество событий с первого считываемого события до указателя записи. Если равно 7FFFFFFFh (MAXINT), то перечисляются все события контроллера.

EnumProc
: [in] Callback-функция, возвращающая параметры для каждого события.

UserData
: [in] Параметр пользователя для Callback-функции.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodePassEvent

Декодирование событий прохода.

C++

```
HRESULT ZG_Ctr_DecodePassEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_CTR_DIRECT* pDirect,
        PINT pKeyIdx,
        PINT pKeyBank
        );
```

Delphi

```
function ZG_Ctr_DecodePassEvent(
    AHandle: THandle;
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VDirect: TZG_CTR_DIRECT;
    var VKeyIdx: Integer;
    var VKeyBank: Integer
    ): HRESULT;
```

Параметры

Handle
[in, out] Дескриптор контроллера.
Data8
[in] Данные события (8 байт).
Time
[out] Дата и время события.
Direct
[out] Направление прохода.

KeyIdx
: [out] Позиция ключа в банке ключей (номер банка в VKeyBank). Если равно -1, то ключ не известен.

KeyBank
: [out] Номер банка ключей.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodeEcEvent

Декодирование событий ElectroControl (Управление элетропитанием).

C++

```
HRESULT ZG_Ctr_DecodeEcEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_EC_SUB_EV* pSubEvent,
        PDWORD pPowerFlags
        );
```

Delphi

```
function ZG_Ctr_DecodeEcEvent(
    AHandle: THandle;
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VSubEvent: TZG_EC_SUB_EV;
    var VPowerFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in, out] Дескриптор контроллера.
pData8
[in] Данные события (8 байт).
Time
[out] Время события.
SubEvent
[out] Условие, вызвавшее события.
PowerFlags

[out] Флаги состояния электропитания.

Флаг	Описание
ZG_EC_SF_ENABLED	Состояние питания – 1 вкл/0 выкл
ZG_EC_SF_SCHEDULE	Активно включение по временной зоне
ZG_EC_SF_REMOTE	Включено по команде по сети
ZG_EC_SF_DELAY	Идет отработка задержки
ZG_EC_SF_CARD	Карта в поле контрольного считывателя

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodeUnkKeyEvent

Декодирование события `ZG_EV_UNKNOWN_KEY` (Неизвестный ключ).

C++

```
HRESULT ZG_Ctr_DecodeUnkKeyEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZ_KEYNUM pKeyNum
        );
```

Delphi

```
function ZG_Ctr_DecodeUnkKeyEvent(
    AHandle: THandle;
    AData8: PByte;
    var VKeyNum: TZ_KEYNUM
    ): HRESULT;
```

Параметры

Handle
: [in, out] Дескриптор контроллера.

Data8
: [in] Данные события (8 байт).

KeyNum
: [out] Номер ключа.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodeFireEvent

Декодирование события `ZG_EV_FIRE_STATE` (Изменение состояния Пожара).

C++

```
HRESULT ZG_Ctr_DecodeFireEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_FIRE_SUB_EV* pSubEvent,
        PDWORD pFireFlags
        );
```

Delphi

```
function ZG_Ctr_DecodeFireEvent(
    AHandle: THandle;
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VSubEvent: TZG_FIRE_SUB_EV;
    var VFireFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
: [in, out] Дескриптор контроллера.

Data8
: [in] Данные события (8 байт).

Time
: [out] Время события.

SubEvent
: [out] Условие, вызвавшее события.

FireFlags

[out] Флаги состояния режима Пожар.

Флаг	Описание
ZG_FR_F_ENABLED	Состояние пожарного режима – 1 вкл/0 выкл
ZG_FR_F_INPUT_F	Активен пожарный режим по входу FIRE
ZG_FR_F_TEMP	Активен пожарный режим по превышению температуры
ZG_FR_F_NET	Активен пожарный режим по внешней команде

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodeSecurEvent

Декодирование события `ZG_EV_SECUR_STATE` (Изменение состояния Охрана).

C++

```
HRESULT ZG_Ctr_DecodeSecurEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_SECUR_SUB_EV* pSubEvent,
        PDWORD pSecurFlags
        );
```

Delphi

```
function ZG_Ctr_DecodeSecurEvent(
    AHandle: THandle;
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VSubEvent: TZG_SECUR_SUB_EV;
    var VSecurFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in, out] Дескриптор контроллера.
Data8
[in] Данные события (8 байт).
Time
[out] Время события.
SubEvent
[out] Условие, вызвавшее события.
SecurFlags

[out] Флаги состояния режима Охрана.

Флаг	Описание
ZG_SR_F_ENABLED	Состояние охранного режима – 1 вкл/0 выкл
ZG_SR_F_ALARM	Состояние тревоги
ZG_SR_F_INPUT_A	Тревога по входу ALARM
ZG_SR_F_TAMPERE	Тревога по тамперу
ZG_SR_F_DOOR	Тревога по датчику двери
ZG_SR_F_NET	Тревога включена по сети

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_DecodeModeEvent

Декодирование события `ZG_EV_MODE_STATE` (Изменение состояния Режим).

C++

```
HRESULT ZG_Ctr_DecodeModeEvent(
        HANDLE hHandle,
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_CTR_MODE* pCurrMode,
        ZG_MODE_SUB_EV* pSubEvent
        );
```

Delphi

```
function ZG_Ctr_DecodeModeEvent(
    AHandle: THandle;
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VCurrMode: TZG_CTR_MODE;
    var VSubEvent: TZG_MODE_SUB_EV
    ): HRESULT;
```

Параметры

Handle
[in, out] Дескриптор контроллера.
Data8
[in] Данные события (8 байт).
Time
[out] Время события.
CurrMode
[out] Текущий режим.
SubEvent

[out] Условие, вызвавшее событие.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_DecodeHotelEvent

Декодирование событий ZG_EV_HOTEL40 и ZG_EV_HOTEL41.

C++

```
HRESULT ZG_DecodeHotelEvent(
        PBYTE pData8,
        PZG_EV_TIME pTime,
        ZG_HOTEL_MODE* pCurrMode,
        ZG_HOTEL_SUB_EV* pSubEvent,
        PDWORD pFlags
        );
```

Delphi

```
function ZG_DecodeHotelEvent(
    AData8: PByte;
    var VTime: TZG_EV_TIME;
    var VCurrMode: TZG_HOTEL_MODE;
    var VSubEvent: TZG_HOTEL_SUB_EV;
    var VFlags: Cardinal
    ): HRESULT;
```

Параметры

Data8
: [in] Данные события (8 байт).

Time
: [out] Время события.

CurrMode
: [out] Текущий режим.

SubEvent
: [out] Условие, вызвавшее событие.

Flags
: [out] Флаги состояния.

Флаг	Описание
ZG_HF_LATCH	Защёлка
ZG_HF_LATCH2	Задвижка
ZG_HF_KEY	Ключ
ZG_HF_CARD	Карта

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Режим Пожар

Функционирование

Включение пожарного режима осуществляется при:

- Получении команды включения по сети
- Появлении активного уровня на входе FIRE
- Превышении порога температуры, измеряемой датчиком на плате.

Выключение питания осуществляется при:

- Получении команды выключения по сети
- Пропадании активного уровня на входе FIRE
- Уменьшении температуры на 5 градусов ниже порога срабатывания
- Дистанционном изменении параметров.

Дополнительные команды

- ZG_Ctr_SetFireMode - управление пожарным режимом по сети
- ZG_Ctr_GetFireInfo - запрос состояния
- ZG_Ctr_SetFireConfig - установка параметров.

Дополнительное событие

ZG_EV_FIRE_STATE - генерируется при изменении состояния. Для раскодирования параметров события используйте функцию ZG_Ctr_DecodeFireEvent.

ZG_Ctr_SetFireMode

Управление пожарным режимом по сети.

C++

```
HRESULT ZG_Ctr_SetFireMode(
        HANDLE hHandle,
        BOOL fOn
        );
```

Delphi

```
function ZG_Ctr_SetFireMode(
    AHandle: THandle;
    AOn: LongBool
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

On
: [in] True, включить пожарный режим, иначе - выключить.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetFireInfo

Запрос состояния пожарного режима.

C++

```
HRESULT ZG_Ctr_GetFireInfo(
        HANDLE hHandle,
        PDWORD pFireFlags,
        PDWORD pCurrTemp,
        PDWORD pSrcMask,
        PDWORD pLimitTemp
        );
```

Delphi

```
function ZG_Ctr_GetFireInfo(
    AHandle: THandle;
    var VFireFlags: Cardinal;
    var VCurrTemp: Cardinal;
    var VSrcMask: Cardinal;
    var VLimitTemp: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

FireFlags

[out] Флаги состояния режима Пожар.

Флаг	Описание
ZG_FR_F_ENABLED	Состояние пожарного режима – 1 вкл/0 выкл
ZG_FR_F_INPUT_F	Активен пожарный режим по входу FIRE

Флаг	Описание
ZG_FR_F_TEMP	Активен пожарный режим по превышению температуры
ZG_FR_F_NET	Активен пожарный режим по внешней команде

CurrTemp

[out] Текущая температура (в градусах).

SrcMask

[out] Маска разрешенных источников.

Флаг	Описание
ZG_FR_SRCF_INPUT_F	Разрешен пожарный режим по входу FIRE
ZG_FR_SRCF_TEMP	Разрешен пожарный режим по превышению температуры

LimitTemp

[out] Пороговая температура (в градусах).

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetFireConfig

Установка параметров пожарного режима.

C++

```
HRESULT ZG_Ctr_SetFireConfig(
        HANDLE hHandle,
        DWORD nSrcMask,
        DWORD nLimitTemp,
        PDWORD pFireFlags,
        PDWORD pCurrTemp
        );
```

Delphi

```
function ZG_Ctr_SetFireConfig(
    AHandle: THandle;
    ASrcMask: Cardinal;
    ALimitTemp: Cardinal;
    VFireFlags: PCardinal;
    VCurrTemp: PCardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
SrcMask
[in] Маска разрешенных источников.
LimitTemp
[in] Пороговая температура (в градусах).
FireFlags
[out] Флаги состояния режима Пожар.

Флаг	Описание
ZG_FR_F_ENABLED	Состояние пожарного режима – 1 вкл/0 выкл
ZG_FR_F_INPUT_F	Активен пожарный режим по входу FIRE
ZG_FR_F_TEMP	Активен пожарный режим по превышению температуры
ZG_FR_F_NET	Активен пожарный режим по внешней команде

CurrTemp

[out] Текущая температура (в градусах).

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Режим Охрана

Функционирование

Включение тревожного режима осуществляется при:

- Получении команды включения по сети.

Выключение питания осуществляется при:

- Получении команды выключения по сети.

Дополнительные команды

- ZG_Ctr_SetSecurMode - управление режимом по сети
- ZG_Ctr_GetSecurInfo - запрос состояния
- ZG_Ctr_SetSecurConfig - установка параметров.

Дополнительное событие

ZG_EV_SECUR_STATE - генерируется при изменении состояния. Для раскодирования параметров события используйте функцию ZG_Ctr_DecodeSecurEvent.

ZG_Ctr_SetSecurMode

Управление режимом Охрана по сети.

C++

```
HRESULT ZG_Ctr_SetSecurMode(
        HANDLE hHandle,
        ZG_SECUR_MODE nMode
        );
```

Delphi

```
function ZG_Ctr_SetSecurMode(
    AHandle: THandle;
    AMode: TZG_SECUR_MODE
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Mode
[in] Режим Охрана.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetSecurInfo

Запрос состояния режима Охрана.

C++

```
HRESULT ZG_Ctr_GetSecurInfo(
        HANDLE hHandle,
        PDWORD pSecurFlags,
        PDWORD pSrcMask,
        PDWORD pAlarmTime
        );
```

Delphi

```
function ZG_Ctr_GetSecurInfo(
    AHandle: THandle;
    var VSecurFlags: Cardinal;
    var VSrcMask: Cardinal;
    var VAlarmTime: Cardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

SecurFlags

[out] Флаги состояния режима Охрана.

Флаг	Описание
ZG_SR_F_ENABLED	Состояние охранного режима – 1 вкл/0 выкл
ZG_SR_F_ALARM	Состояние тревоги
ZG_SR_F_INPUT_A	Тревога по входу ALARM
ZG_SR_F_TAMPERE	Тревога по тамперу
ZG_SR_F_DOOR	Тревога по датчику двери

Флаг	Описание
ZG_SR_F_NET	Тревога включена по сети

SrcMask

[out] Маска разрешенных источников.

Флаг	Описание
ZG_SR_SRCF_INPUT_F	Разрешена тревога по входу FIRE
ZG_SR_SRCF_TAMPERE	Разрешена тревога по тамперу
ZG_SR_SRCF_DOOR	Разрешена тревога по датчику двери

AlarmTime

[out] Время звучания сирены (в секундах).

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetSecurConfig

Установка параметров режима Охрана.

C++

```
HRESULT ZG_Ctr_SetSecurConfig(
        HANDLE hHandle,
        DWORD nSrcMask,
        DWORD nAlarmTime,
        PDWORD pSecurFlags
        );
```

Delphi

```
function ZG_Ctr_SetSecurConfig(
    AHandle: THandle;
    ASrcMask: Cardinal;
    AAlarmTime: Cardinal;
    VSecurFlags: PCardinal
    ): HRESULT;
```

Параметры

Handle

[in] Дескриптор контроллера.

SrcMask

[in] Маска разрешенных источников.

Флаг	Описание
ZG_SR_SRCF_INPUT_F	Разрешена тревога по входу FIRE
ZG_SR_SRCF_TAMPERE	Разрешена тревога по тамперу
ZG_SR_SRCF_DOOR	Разрешена тревога по датчику двери

AlarmTime

[in] Время звучания сирены (в секундах).

SecurFlags

[out] Флаги состояния режима Охрана.

Флаг	Описание
ZG_SR_F_ENABLED	Состояние охранного режима – 1 вкл/0 выкл
ZG_SR_F_ALARM	Состояние тревоги
ZG_SR_F_INPUT_A	Тревога по входу ALARM
ZG_SR_F_TAMPERE	Тревога по тамперу
ZG_SR_F_DOOR	Тревога по датчику двери
ZG_SR_F_NET	Тревога включена по сети

Возвращаемое значение

Если функция выполнена успешно, то возвращает S_OK.

Если функция выполнена с ошибками, то возвращает код ошибки.

Режимы "Блокировка" и "Свободный проход"

Контроллеры – только Z5R-Net и Matrix-II Net с новой прошивкой "xxx_2703_blk.rom".

Проверка поддержки управления режимами осуществляется с помощью флага `ZG_CTR_F_MODES`.

В прошивке реализован механизм контроля состояния двери. Всего 4 состояния:

1. Норма (обычный режим)
2. Блокировано (аналог режима «блокировка» для Z5R)
3. Свободно (замок обесточен)
4. Ожидание (переход в режим «Свободно» при проходе по карточке)

<table>
<tr><th>Действие</th><th>Норма</th><th>Блокировано</th><th>Свободно</th><th>Ожидание</th></tr>
<tr><td>Обычная карта</td><td rowspan="3">Доступ разрешен</td><td>Доступ запрещен</td><td rowspan="3">Замок обесточен</td><td rowspan="2">Доступ разрешен, переход замка в режим "Свободно"</td></tr>
<tr><td>Блокирующая карта</td><td rowspan="2">Доступ разрешен</td></tr>
<tr><td>Кнопка</td><td>Доступ разрешен, остаётся режим "Ожидание"</td></tr>
</table>

Функционирование

Включение/выключение режимов осуществляется при:

- Срабатывании временной зоны (высший приоритет)
- Получении команды по сети
- Блокирующей картой (удержание более 3 сек)
    - Если какой-либо режим включен, то он выключается
    - При открытой двери включается “Свободный проход”
    - При закрытой двери включается “Блокировка”
- Режим «Ожидание» включается только временной зоной или командой по сети и означает переход в режим «Свободный проход», при проходе любой разрешенной картой (кнопка НЕ включает «Свободный проход»)

Карточкой – удержание блокирующей карточки на считывателе более 3 сек переводит замок в состояние «Блокировано» при закрытой на момент удержания двери, или «Открыто» в случае если дверь открыта. Состояние двери определяется по датчику двери. Переход в режим «Норма» так же осуществляется удержанием карты в любом состоянии двери. В состояние «Ожидание» карточкой не перевести.

Сигналы при переходе в режим «Блокировано» – один длинный, «Свободно» – два длинных, «Норма» – серия коротких.

Командой по сети RS-485 переключение можно осуществить в любое время.

Переключение по временной зоне. Дополнительно добавлены две зоны с помощью которых можно указывать периоды состояний «Блокировано», «Свободно» и «Ожидание». **Временная зона имеет наивысший приоритет – во время её действия ни карточкой, ни командой режим**

не изменить! Если зоны перекрывают друг друга, то работает первая.

Дополнительные команды

- ZG_Ctr_SetCtrMode - управление режимом по сети
- ZG_Ctr_GetCtrModeInfo - запрос состояния

Для работы с 2 временными зонами режима предназначены функции ZG_Ctr_ReadTimeZones, ZG_Ctr_WriteTimeZones. Индекс первой зоны равен **-2** (константа ZG_MODES_TZ0).

Дополнительное событие

ZG_EV_MODE_STATE - генерируется при изменении состояния. Для раскодирования параметров события используйте функцию ZG_Ctr_DecodeModeEvent.

ZG_Ctr_SetCtrMode

Управление режимами контроллера.

C++

```
HRESULT ZG_Ctr_SetCtrMode(
        HANDLE hHandle,
        ZG_CTR_MODE nMode
        );
```

Delphi

```
function ZG_Ctr_SetCtrMode(
    AHandle: THandle;
    AMode: TZG_CTR_MODE
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
Mode
[in] Режим контроллера.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetCtrModeInfo

Запрос состояния режима контроллера.

C++

```
HRESULT ZG_Ctr_GetCtrModeInfo(
        HANDLE hHandle,
        ZG_CTR_MODE* pCurrMode,
        PDWORD pFlags
        );
```

Delphi

```
function ZG_Ctr_GetCtrModeInfo(
    AHandle: THandle;
    var VMode: TZG_CTR_MODE;
    var VFlags: Cardinal
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
CurrMode
[out] Текущий режим.
Flags
[out] Флаги.

Бит	Описание
0	Включен по временной зоне
1	Включен командой по сети
2	Включен картой

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Управление электопитанием "ElectroControl" (Matrix-II Net, Matrix-III Net)

Доработка

Выход на управление внешним светодиодом используется как управление внешним транзистором. На этот выход подается напряжение 5 вольт через резистор 360 Ом.

Внешний транзистор должен коммутировать реле управляющее подачей питания в номер.

Функционирование

Включение питания осуществляется при:

- Поднесении допущенной карты к любому считывателю (включается на заданный таймаут), повторное поднесение не сбрасывает отработку таймаута
- Подаче команды по сети на включение (ZG_Ctr_SetElectroPower)
- Нахождении допущенной карты в поле управляющего считывателя
- При активном заданном диапазоне времени работы и попадании текущего времени в его границы

Выключение питания осуществляется при:

- Отсутствии карты в поле считывателей в течение заданного периода времени
- Подаче команды по сети на выключение (ZG_Ctr_SetElectroPower)
- При активном заданном диапазоне времени работы (флаг ZG_CTR_ECF_SCHEDULE) и выходе текущего времени за его границы
- При задействовании датчика двери (флаг ZG_CTR_ECF_EXIT_OFF) и закрывании двери после снятия с контрольного считывателя (при выходе из номера)

Дополнительные команды

- ZG_Ctr_ReadElectroConfig - чтение параметров электропитания
- ZG_Ctr_WriteElectroConfig - запись новых параметров электропитания
- ZG_Ctr_GetElectroState - запрос статуса электропитания
- ZG_Ctr_SetElectroPower - включение/выключение электропитания

Дополнительное событие

ZG_EV_ELECTRO_ON - включение электропитания. Для раскодирования параметров события используйте функцию ZG_Ctr_DecodeEcEvent.

ZG_EV_ELECTRO_OFF - выключение электропитания. Для раскодирования - ZG_Ctr_DecodeEcEvent.

ZG_Ctr_ReadElectroConfig

Читает параметры электропитания. Функция ElectroControl поддерживается контроллерами: Matrix II Net (с прошивкой v3) и Matrix-III Net.

C++

```
HRESULT ZG_Ctr_ReadElectroConfig(
        HANDLE hHandle,
        PZG_CTR_ELECTRO_CONFIG pConfig
        );
```

Delphi

```
function ZG_Ctr_ReadElectroConfig(
    AHandle: THandle;
    var VConfig: TZG_CTR_ELECTRO_CONFIG
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Config
: [out] Параметры электропитания.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_WriteElectroConfig

Пишет новые параметры электропитания. Функция ElectroControl поддерживается контроллерами: Matrix II Net (с прошивкой v3) и Matrix-III Net.

C++

```
HRESULT ZG_Ctr_WriteElectroConfig(
        HANDLE hHandle,
        PZG_CTR_ELECTRO_CONFIG pConfig,
        BOOL fSetTz=TRUE
        );
```

Delphi

```
function ZG_Ctr_WriteElectroConfig(
    AHandle: THandle;
    Const AConfig: TZG_CTR_ELECTRO_CONFIG;
    ASetTz: LongBool=True
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

Config
: [in] Параметры электропитания.

SetTz
: [in] True, установить параметры временной зоны №6 банка №0.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_GetElectroState

Возвращает статус электропитания. Функция ElectroControl поддерживается контроллерами: Matrix II Net (с прошивкой v3) и Matrix-III Net.

C++

```
HRESULT ZG_Ctr_GetElectroState(
        HANDLE hHandle,
        PZG_CTR_ELECTRO_STATE pState
        );
```

Delphi

```
function ZG_Ctr_GetElectroState(
    AHandle: THandle;
    var VState: TZG_CTR_ELECTRO_STATE
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
State
[out] Состояние электропитания.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

ZG_Ctr_SetElectroPower

Включает / выключает электропитание. Функция ElectroControl поддерживается контроллерами: Matrix II Net (с прошивкой v3) и Matrix-III Net.

C++

```
HRESULT ZG_Ctr_SetElectroPower(
        HANDLE hHandle,
        BOOL fOn
        );
```

Delphi

```
function ZG_Ctr_SetElectroPower(
    AHandle: THandle;
    AOn: LongBool
    ): HRESULT;
```

Параметры

Handle
[in] Дескриптор контроллера.
AOn
[in] True, включить питание, False - выключить.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Типы ZGuard API

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_NOTIFYPROC	Функция обратного вызова для уведомления о событии.
ZG_PROCESSCALLBACK	Функция обратного вызова для показа статуса выполнения функции и возможности ее прерывания.
ZG_ENUMCTRSPROC	Функция обратного вызова, возвращающая информацию о каждом найденном контроллере.
ZG_ENUMCTRTIMEZONESPROC	Функция обратного вызова, возвращающая параметры для каждой временной зоны.
ZG_ENUMCTRKEYSPROC	Функция обратного вызова, возвращающая параметры для каждого ключа.
ZG_ENUMCTREVENTSPROC	Функция обратного вызова, возвращающая параметры для каждого события.
Типы для работы с конвертером	
ZP_PORT_TYPE	Тип порта.

Имя типа	Кратное описание
ZP_PORT_NAME	Имя порта.
ZP_PORT_ADDR	Адрес порта: тип + имя.
ZP_DEVICE_INFO	Базовая информация об устройстве.
ZG_CVT_TYPE	Тип конвертера.
ZG_GUARD_MODE	Режим работы конвертера Guard.
ZG_CVT_SPEED	Скорость конвертера Z-397 и Z-397_Guard в режиме Normal.
ZG_ENUM_IPCVT_INFO	Информация об IP-конвертере, возвращаемая функцией ZG_FindNextDevice.
ZG_ENUM_CVT_INFO	Информация о конвертере, возвращаемая функцией ZG_FindNextDevice.
ZG_CVT_INFO	Информация о конвертере, возвращаемая функциями: ZG_Cvt_Open и ZG_Cvt_GetInformation.
ZG_CVT_NOTIFY_SETTINGS	Параметры для уведомлений от конвертера.
ZG_CVT_LIC_INFO	Информация о лицензии конвертера Guard.
Типы для работы с контроллером	
Z_KEYNUM	Номер ключа.
ZG_CTR_TYPE	Тип контроллера.
ZG_FIND_CTR_INFO	Информация о найденном контроллере, возвращаемая функциями: ZG_Cvt_EnumControllers и ZG_Cvt_FindController.

Имя типа	Кратное описание
ZG_CTR_INFO	Информация о контроллере, возвращаемая функциями: ZG_Ctr_Open и ZG_Ctr_GetInformation.
ZG_CTR_NOTIFY_SETTINGS	Параметры для уведомлений от контроллера.
ZG_N_NEW_EVENT_INFO	Информация для уведомления ZG_N_CTR_NEW_EVENT.
ZG_N_KEY_TOP_INFO	Информация для уведомления ZG_N_CTR_KEY_TOP.
ZG_CTR_TIMEZONE	Временная зона контроллера.
ZG_CTR_KEY_TYPE	Тип ключа.
ZG_CTR_KEY	Ключ контроллера.
ZG_CTR_CLOCK	Часы контроллера.
ZG_CTR_EV_TYPE	Тип события контроллера.
ZG_CTR_DIRECT	Направление прохода.
ZG_CTR_EVENT	Событие контроллера.
Типы для функции ElectroControl	
ZG_CTR_ELECTRO_CONFIG	Параметры электопитания контроллера.
ZG_CTR_ELECTRO_STATE	Состояние электопитания контроллера.

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-конвертеров)
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 ZG_SearchDevices.

Friendly

Дружественное имя порта.

DevTypes

Маска типов устройств.

Owner

IP-адрес компьютера, занявшего линию конвертера.

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 ZG_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 ZG_SetNotification, и извлекаются с помощью ZG_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 и если `(((1 <<`

aDevs[i].nTypeId) & ZG_IPDEVTYPE_CVTS) != 0), т.е. если бит aDevs[i].nTypeId установлен в маске ZG_IPDEVTYPE_CVTS, то указатель можно привести к типу ZG_ENUM_IPCVT_INFO.

Если (aDevs[i].nTypeId < ZP_MAX_REG_DEV), то устройство найдено опросом портов и если (((1 << aDevs[i].nTypeId) & ZG_DEVTYPE_CVTS) != 0), то указатель можно привести к типу ZG_ENUM_CVT_INFO.

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 ZG_SetNotification, и извлекаются с помощью ZG_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 и если `(((1 << pInfo.nTypeId) & ZG_IPDEVTYPE_CVTS) != 0)`, т.е. если бит `pInfo.nTypeId` установлен в маске `ZG_IPDEVTYPE_CVTS`, то указатель можно привести к типу

ZG_ENUM_IPCVT_INFO.

Если (pInfo.nTypeId < ZP_MAX_REG_DEV), то устройство найдено опросом портов и если (((1 << pInfo.nTypeId) & ZG_DEVTYPE_CVTS) != 0), то указатель можно привести к типу ZG_ENUM_CVT_INFO.

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 ZG_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;
} *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;
  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.

ZP_NOTIFYPROC

Функция обратного вызова для уведомления о событии.

C++

```
typedef BOOL (CALLBACK* ZP_NOTIFYPROC) (UINT
nMsg, LPARAM lMsgParam, PVOID pParam);
```

Delphi

```
TZP_NOTIFYPROC = function(AMsg: Cardinal;
AMsgParam: LPARAM; AParam: Pointer): LongBool;
stdcall;
```

Параметры

Msg
: Тип сообщения.

MsgParam
: Параметр сообщения.

UserData
: Параметр пользователя для Callback-функции.

ZG_PROCESSCALLBACK

Функция обратного вызова для показа статуса выполнения функции и возможности ее прерывания.

C++

```
typedef BOOL (CALLBACK* ZG_PROCESSCALLBACK) (INT nPos, INT nMax, PVOID pUserData);
```

Delphi

```
TZG_PROCESSCALLBACK = function(APos: Integer; AMax: Integer; AUserData: Pointer): LongBool; stdcall;
```

Параметры

Pos
: Текущий шаг выполнения функции.

Max
: Всего количество шагов.

UserData
: Параметр пользователя для Callback-функции.

ZG_ENUMCTRSPROC

Функция обратного вызова, возвращающая информацию о каждом найденном контроллере.

C++

```
typedef BOOL (CALLBACK* ZG_ENUMCTRSPROC)
(PZG_FIND_CTR_INFO pInfo, INT nPos, INT nMax,
PVOID pUserData);
```

Delphi

```
TZG_ENUMCTRSPROC = function(AInfo:
PZG_FIND_CTR_INFO; APos: Integer; AMax: Integer;
AUserData: Pointer): LongBool; stdcall;
```

Параметры

Info
: Информация о контроллере.

Pos
: Текущий шаг выполнения функции.

Max
: Всего количество шагов.

UserData
: Параметр пользователя для Callback-функции.

ZG_ENUMCTRKEYSPROC

Функция обратного вызова, возвращающая параметры для каждого ключа.

C++

```
typedef BOOL (CALLBACK* ZG_ENUMCTRKEYSPROC) (INT nIdx, PZG_CTR_KEY pKey, INT nPos, INT nMax, PVOID pUserData);
```

Delphi

```
TZG_ENUMCTRKEYSPROC = function(AIdx: Integer; AKey: PZG_CTR_KEY; APos: Integer; AMax: Integer; AUserData: Pointer): LongBool; stdcall;
```

Параметры

Idx
: Позиция текущего ключа.

Key
: Параметры ключа.

Pos
: Текущий шаг выполнения функции.

Max
: Всего количество шагов.

UserData
: Параметр пользователя для Callback-функции.

ZG_ENUMCTRTIMEZONESPROC

Функция обратного вызова, возвращающая параметры для каждой временной зоны.

C++

```
typedef BOOL (CALLBACK* ZG_ENUMCTRTIMEZONESPROC)
(INT nIdx, PZG_CTR_TIMEZONE pTz, PVOID
pUserData);
```

Delphi

```
TZG_ENUMCTRTIMEZONESPROC = function(AIdx:
Integer; ATz: PZG_CTR_TIMEZONE; AUserData:
Pointer): LongBool; stdcall;
```

Параметры

Idx
: Позиция текущей временной зоны.

Tz
: Параметры временной зоны.

UserData
: Параметр пользователя для Callback-функции.

ZG_ENUMCTREVENTSPROC

Функция обратного вызова, возвращающая параметры для каждого события.

C++

```
typedef BOOL (CALLBACK* ZG_ENUMCTREVENTSPROC)
(INT nIdx, PZG_CTR_EVENT pEvent, INT nPos, INT
nMax, PVOID pUserData);
```

Delphi

```
TZG_ENUMCTREVENTSPROC = function(AIdx: Integer;
AEvent: PZG_CTR_EVENT; APos, AMax: Integer;
AUserData: Pointer): LongBool; stdcall;
```

Параметры

Idx
: Позиция текущего события.

Event
: Параметры события.

Pos
: Текущий шаг выполнения функции.

Max
: Всего количество шагов.

UserData
: Параметр пользователя для Callback-функции.

ZG_CVT_TYPE

Модель конвертера.

C++

```
enum ZG_CVT_TYPE : INT
{
        ZG_CVT_UNDEF = 0,
        ZG_CVT_Z397,
        ZG_CVT_Z397_GUARD,
        ZG_CVT_Z397_IP,
        ZG_CVT_Z397_WEB,
        ZG_CVT_Z5R_WEB,
        ZG_CVT_MATRIX2WIFI
};
```

Delphi

```
  TZG_CVT_TYPE = (
    ZG_CVT_UNDEF = 0,
    ZG_CVT_Z397,
    ZG_CVT_Z397_GUARD,
    ZG_CVT_Z397_IP,
    ZG_CVT_Z397_WEB,
    ZG_CVT_Z5R_WEB,
    ZG_CVT_MATRIX2WIFI
  );
```

Константа	Описание
ZG_CVT_UNDEF	Не известно
ZG_CVT_Z397	Z-397 (простой)
ZG_CVT_Z397_GUARD	Z-397 Guard
ZG_CVT_Z397_IP	Z-397 IP
ZG_CVT_Z397_WEB	Z-397 Web

Константа	Описание
ZG_CVT_Z5R_WEB	Z5R Web - контроллер, совмещенный с конвертером.
ZG_CVT_MATRIX2WIFI	Matrix-II Wi-Fi - IP-контроллер, совмещенный со считывателем EM-Marine.

ZG_CVT_SPEED

Скорость конвертера Z-397 и Z-397_Guard в режиме Normal.

C++

```
enum ZG_CVT_SPEED : UINT
{
        ZG_SPEED_19200  = 19200,
        ZG_SPEED_57600  = 57600
};
```

Delphi

```
  TZG_CVT_SPEED = (
          ZG_SPEED_19200          = 19200,
          ZG_SPEED_57600          = 57600
  );
```

Константа	Описание
ZG_SPEED_19200	19200 бит/с
ZG_SPEED_57600	57600 бит/с

ZG_GUARD_MODE

Режим работы конвертера Guard.

C++

```
enum ZG_GUARD_MODE : INT
{
        ZG_GUARD_UNDEF = 0,
        ZG_GUARD_NORMAL,
        ZG_GUARD_ADVANCED,
        ZG_GUARD_TEST,
        ZG_GUARD_ACCEPT
};
```

Delphi

```
  TZG_GUARD_MODE = (
          ZG_GUARD_UNDEF = 0,
          ZG_GUARD_NORMAL,
          ZG_GUARD_ADVANCED,
          ZG_GUARD_TEST,
          ZG_GUARD_ACCEPT
  );
  PZG_GUARD_MODE = ^TZG_GUARD_MODE;
```

Константа	Описание
ZG_GUARD_UNDEF	Не известно
ZG_GUARD_NORMAL	Обычный. Имитация обычного конвертера Z-397.
ZG_GUARD_ADVANCED	Расширенный. С дополнительными функциями Guard.
ZG_GUARD_TEST	Тестовый. Для служебного использования.
ZG_GUARD_ACCEPT	Акцепт. Для служебного использования.

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.

Структура ZG_ENUM_CVT_INFO

Информация о конвертере, возвращаемая функцией ZG_FindNextDevice.

C++

```
typedef struct _ZG_ENUM_CVT_INFO :
_ZP_DEVICE_INFO
{
        ZG_CVT_TYPE nType;
        ZG_GUARD_MODE nMode;
} *PZG_ENUM_CVT_INFO;
```

Delphi

```
  TZG_ENUM_CVT_INFO = packed record
    rBase             : TZP_DEVICE_INFO;
    nType             : TZG_CVT_TYPE;
    nMode             : TZG_GUARD_MODE;
  end;
  PZG_ENUM_CVT_INFO = ^TZG_ENUM_CVT_INFO;
```

Параметры

Base
Базовая информация об устройстве.
Type
Тип конвертера.
Mode
Режим работы конвертера Guard.

Структура ZG_ENUM_IPCVT_INFO

Информация о конвертере, возвращаемая функцией ZG_FindNextDevice.

C++

```
typedef struct _ZG_ENUM_IPCVT_INFO :
_ZP_DEVICE_INFO
{
        ZG_CVT_TYPE nType;
        ZG_GUARD_MODE nMode;
        DWORD nFlags;
} *PZG_ENUM_IPCVT_INFO;
```

Delphi

```
  TZG_ENUM_IPCVT_INFO = packed record
    rBase             : TZP_DEVICE_INFO;
    nType             : TZG_CVT_TYPE;
    nMode             : TZG_GUARD_MODE;
    nFlags            : Cardinal;
  end;
  PZG_ENUM_IPCVT_INFO = ^TZG_ENUM_IPCVT_INFO;
```

Параметры

Base
Базовая информация об устройстве.
Type
Тип конвертера.
Mode
Режим работы конвертера Guard.
Flags
Флаги: бит 0 - "VCP", бит 1 - "WEB", 0xFF - "All".

Структура ZG_CVT_INFO

Информация о конвертере, возвращаемая функциями: ZG_Cvt_Open и ZG_Cvt_GetInformation.

C++

```
typedef struct _ZG_CVT_INFO
{
        ZG_CVT_TYPE nType;
        ZG_CVT_SPEED nSpeed;
        WORD nSn;
        WORD nVersion;
        ZG_GUARD_MODE nMode;
        LPWSTR pszLinesBuf;
        INT nLinesBufMax;
} *PZG_CVT_INFO;
```

Delphi

```
  TZG_CVT_INFO = packed record
    nType             : TZG_CVT_TYPE;
    nSpeed            : TZG_CVT_SPEED;
    nSn               : Word;
    nVersion          : Word;
    nMode             : TZG_GUARD_MODE;
    pszLinesBuf       : PWideChar;
    nLinesBufMax      : Integer;
  end;
  PZG_CVT_INFO = ^TZG_CVT_INFO;
```

Параметры

Type

Тип конвертера.

Speed

Скорость конвертера.

Sn

С/н конвертера.

Version

Версия конвертера.

Mode

Режим работы конвертера Guard.

pszLinesBuf

Буфер для информационных строк. Может быть равен **null**.

nLinesBufMax

Размер буфера в символах, включая завершающий нулевой.

Структура ZG_CVT_NOTIFY_SETTINGS

Параметры для уведомлений конвертера, используются для функции ZG_Cvt_SetNotification.

C++

```
typedef struct _ZG_CVT_NOTIFY_SETTINGS
{
        UINT nNMask;
        HANDLE hEvent;
        HWND hWindow;
        UINT nWndMsgId;
        DWORD nScanCtrsPeriod;
        INT nScanCtrsLastAddr;
} *PZG_CVT_NOTIFY_SETTINGS;
```

Delphi

```
  TZG_CVT_NOTIFY_SETTINGS = packed record
    nNMask           : Cardinal;
    hEvent           : THandle;
    hWindow          : HWnd;
    nWndMsgId        : Cardinal;
    nScanCtrsPeriod : Cardinal;
    nScanCtrsLastAddr: Integer;
  end;
  PZG_CVT_NOTIFY_SETTINGS = ^TZG_CVT_NOTIFY_SETTINGS;
```

Параметры

NMask

Маска типов уведомлений:

Флаг	Значение	Описание
ZG_NF_CVT_CTR_EXIST	1	Уведомления о подключении/ отключении контроллеров (ZG_N_CVT_CTR_INSERT, ZG_N_CVT_CTR_REMOVE). Для Z-397 и Z-397 Guard **Normal**: сканирование может производиться 2 способами: 1) сканирование всей сети сразу 2) последовательная проверка каждого адреса сети. Если

Флаг	Значение	Описание
		первым способом найти ничего не удалось, то задействуется второй способ, после которого снова задействуется первый способ.
ZG_NF_CVT_CTR_CHANGE	2	Уведомление об изменении параметров контроллера (ZG_N_CVT_CTR_CHANGE).
ZG_NF_CVT_ERROR	4	Уведомление о ошибках в потоке (thread) (ZG_N_CVT_ERROR)
ZG_CVTNF_CONNECTION_CHANGE	8	Уведомление об изменении состояния подключения к конвертеру (ZG_CVTN_CONNECTION_CHANGE).
ZG_NF_CVT_CTR_DBL_CHECK	1000h	Дважды проверять отключение контроллеров. При потере связи с найденным контроллером сканирование повторяется. (работает только вместе с ZG_NF_CVT_CTR_EXIST).
ZG_NF_CVT_REASSIGN_ADDRS	2000h	Автоматическое переназначение адресов контроллеров (для **Z-397** и **Z-397 Guard** **Normal**) (работает только вместе с ZG_NF_CVT_CTR_EXIST).
ZG_NF_CVT_RESCAN_CTRS	10000h	Начать заново сканирование контроллеров (для **Z-397** и **Z-397 Guard** **Normal**) (работает только вместе с ZG_NF_CVT_CTR_EXIST).
ZG_NF_CVT_ALT_SCAN	20000h	Использовать альтернативный метод скарирования (перебором адресов). В режиме Advanced не работает.
ZG_NF_CVT_NOGATE	40000h	Не сканировать GATE-контроллеры (не сканировать всё, кроме Eurolock EHT net)
ZG_NF_CVT_NOEUROLOCK	80000h	Не сканировать Eurolock EHT net.

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.

ScanCtrsPeriod

Период сканирования списка контроллеров в миллисекундах. Если равно 0, то используется значение по умолчанию (5000).

ScanCtrsLastAddr

Последней сканируемый адрес контроллера (для Z-397 и Z-397 Guard Normal). Если равно 0, то используется значение по умолчанию (31).

Структура ZG_N_CTR_CHANGE_INFO

Информация для уведомления ZG_N_CVT_CTR_CHANGE.

C++

```
typedef struct _ZG_N_CTR_CHANGE_INFO
{
        UINT nChangeMask;
        _ZG_FIND_CTR_INFO rCtrInfo;
        WORD nOldVersion;
        BYTE nOldAddr;
        BYTE Reserved;
} *PZG_N_CTR_CHANGE_INFO;
```

Delphi

```
  TZG_N_CTR_CHANGE_INFO = packed record
    nChangeMask     : Cardinal;
    rCtrInfo        : TZG_FIND_CTR_INFO;
    nOldVersion     : Word;
    nOldAddr        : Byte;
    Reserved        : Byte;
  end;
  PZG_N_CTR_CHANGE_INFO =
^TZG_N_CTR_CHANGE_INFO;
```

Параметры

nChangeMask
Маска изменений (бит0 addr, бит1 version, бит2 proximity).
rCtrInfo
Информация о контроллере.
nOldVersion
Старая версия контроллера.
nOldAddr

Старый адрес контроллера.

Reserved

Зарезервировано (для выравнивания структуры).

Структура ZG_CVT_LIC_INFO

Информация о лицензии конвертера Guard.

C++

```
typedef struct _ZG_CVT_LIC_INFO
{
        WORD nStatus;
        WORD Reserved;
        INT nMaxCtrs;
        INT nMaxKeys;
        WORD nMaxYear;
        WORD nMaxMon;
        WORD nMaxDay;
        WORD nDownCountTime;
} *PZG_CVT_LIC_INFO;
```

Delphi

```
  TZG_CVT_LIC_INFO = packed record
    nStatus          : Word;
    Reserved         : Word;
    nMaxCtrs         : Integer;
    nMaxKeys         : Integer;
    nMaxYear         : Word;
    nMaxMon          : Word;
    nMaxDay          : Word;
    nDownCountTime   : Word;
  end;
  PZG_CVT_LIC_INFO = ^TZG_CVT_LIC_INFO;
```

Параметры

Status

Статус лицензии.

Reserved
: Зарезервировано (для выравнивания структуры).

MaxCtrs
: Максимальное количество контроллеров.

MaxKeys
: Максимальное количество ключей.

MaxYear
: Дата: год (= 0xFFFF дата неограничена).

MaxMon
: Дата: месяц.

MaxDay
: Дата: день.

DownCountTime
: Оставшееся время жизни лицензии в минутах.

Структура ZG_CVT_LIC_SINFO

Краткая информация о лицензии конвертера Guard.

C++

```
typedef struct _ZG_CVT_LIC_SINFO
{
        int nLicN;
        INT nMaxCtrs;
        INT nMaxKeys;
} *PZG_CVT_LIC_SINFO;
```

Delphi

```
  TZG_CVT_LIC_SINFO = packed record
    nLicN            : Integer;
    nMaxCtrs         : Integer;
    nMaxKeys         : Integer;
  end;
  PZG_CVT_LIC_SINFO = ^TZG_CVT_LIC_SINFO;
```

Параметры

LicN
: Номер лицензии.

MaxCtrs
: Максимальное количество контроллеров.

MaxKeys
: Максимальное количество ключей.

Z_KEYNUM

Номер ключа. Номер ключа храниться в массиве из 16 байт, где первый байт (card_num[0]) определяет длину номера карты L, следующие L байтов - номер карты.

C++

```
typedef BYTE Z_KEYNUM[16];
```

Delphi

```
TZ_KEYNUM = array[0..15] of Byte;
```

Тип карты / Номер байта	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Em-Marine	6	номер		серия	F1	F2	F3	0	0	0	0	0	0	0	0	0

ZG_CTR_TYPE

Модель контроллера.

C++

```
enum ZG_CTR_TYPE : INT
{
        ZG_CTR_UNDEF    = 0,
        ZG_CTR_GATE2K,
        ZG_CTR_MATRIX2NET,
        ZG_CTR_Z5RNET,
        ZG_CTR_Z5RNET8K,
        ZG_CTR_GUARDNET,
        ZG_CTR_Z9,
        ZG_CTR_EUROLOCK,
        ZG_CTR_Z5RWEB,
        ZG_CTR_MATRIX2WIFI,
        ZG_CTR_MATRIX6NFC,
        ZG_CTR_MATRIX6EHK,
        ZG_CTR_MATRIX6WIFI,
        ZG_CTR_Z5R_WEB_BT,
        ZG_CTR_Z5R_WIFI,
        ZG_CTR_MATRIX2EHWEB,
        ZG_CTR_MATRIX6EHWEB,
        ZG_CTR_Z5R_WEB_MINI,
        ZG_CTR_Z5RNET16K
};
```

Delphi

```
  TZG_CTR_TYPE = (
    ZG_CTR_UNDEF = 0,
    ZG_CTR_GATE2K,
    ZG_CTR_MATRIX2NET,
    ZG_CTR_Z5RNET,
    ZG_CTR_Z5RNET8K,
```

```
    ZG_CTR_GUARDNET,
    ZG_CTR_Z9,
    ZG_CTR_EUROLOCK,
    ZG_CTR_Z5RWEB,
    ZG_CTR_MATRIX2WIFI,
    ZG_CTR_MATRIX6NFC,
    ZG_CTR_MATRIX6EHK,
    ZG_CTR_MATRIX6WIFI,
    ZG_CTR_Z5R_WEB_BT,
    ZG_CTR_Z5R_WIFI,
    ZG_CTR_MATRIX2EHWEB,
    ZG_CTR_MATRIX6EHWEB,
    ZG_CTR_Z5R_WEB_MINI,
    ZG_CTR_Z5RNET16K
);
```

Константа	Описание
ZG_CTR_UNDEF	Не известно
ZG_CTR_GATE2K	Gate 2000
ZG_CTR_MATRIX2NET	Matrix II Net
ZG_CTR_Z5RNET	Z5R Net
ZG_CTR_Z5RNET8K	Z5R Net 8000
ZG_CTR_GUARDNET	Guard Net
ZG_CTR_Z9	Z-9 EHT Net
ZG_CTR_EUROLOCK	EuroLock EHT net
ZG_CTR_Z5RWEB	Z-5R Web
ZG_CTR_MATRIX2WIFI	Matrix-II Wi-Fi
ZG_CTR_MATRIX6NFC	Matrix-VI NFC Net 8k
ZG_CTR_MATRIX6EHK	Matrix-VI EHK Net 2k
ZG_CTR_MATRIX6WIFI	Matrix VI Wi-Fi
ZG_CTR_Z5R_WEB_BT	Z5R-WEB BT
ZG_CTR_Z5R_WIFI	Z5R Wi-Fi

Константа	Описание
ZG_CTR_MATRIX2EHWEB	Matrix-II EH Web
ZG_CTR_MATRIX6EHWEB	Matrix-VI EH Web
ZG_CTR_Z5R_WEB_MINI	Z5R Web mini
ZG_CTR_Z5RNET16K	Z-5R Net 16k

ZG_CTR_SUB_TYPE

Подтип контроллера.

C++

```
enum ZG_CTR_SUB_TYPE : INT
{
        ZG_CS_UNDEF     = 0,
        ZG_CS_DOOR,
        ZG_CS_TURNSTILE,
        ZG_CS_GATEWAY,
        ZG_CS_BARRIER,
        ZG_CS_ELECTROMAGNETIC,
        ZG_CS_ELECTROMECHANICAL,
        ZG_CS_MOTORTWORELAYS,
        ZG_CS_MOTORONERELAY,
        ZG_CS_ELECTRICAL,
        ZG_CS_AUTOCONTROL
};
```

Delphi

```
  TZG_CTR_SUB_TYPE = (
    ZG_CS_UNDEF = 0,
    ZG_CS_DOOR,
    ZG_CS_TURNSTILE,
    ZG_CS_GATEWAY,
    ZG_CS_BARRIER,
    ZG_CS_ELECTROMAGNETIC,
    ZG_CS_ELECTROMECHANICAL,
    ZG_CS_MOTORTWORELAYS,
    ZG_CS_MOTORONERELAY,
    ZG_CS_ELECTRICAL,
    ZG_CS_AUTOCONTROL
  );
```

Константа	Описание
ZG_CS_UNDEF	Не известно
ZG_CS_DOOR	Дверь. Только для Guard-Net.
ZG_CS_TURNSTILE	Турникет
ZG_CS_GATEWAY	Шлюз. Только для Guard-Net.
ZG_CS_BARRIER	Шлакбаум. Только для Guard-Net.
ZG_CS_ELECTROMAGNETIC	Электромагнитный замок
ZG_CS_ELECTROMECHANICAL	Электромеханический замок
ZG_CS_MOTORTWORELAYS	Моторный через два реле
ZG_CS_MOTORONERELAY	Моторный через одно реле
ZG_CS_ELECTRICAL	Электроконтроль
ZG_CS_AUTOCONTROL	Автоконтроль

Структура ZG_FIND_CTR_INFO

Информация о найденном контроллере, возвращаемая функцией ZG_Cvt_EnumControllers.

C++

```
typedef struct _ZG_FIND_CTR_INFO
{
        ZG_CTR_TYPE nType;
        BYTE nTypeCode;
        BYTE nAddr;
        INT nSn;
        WORD nVersion;
        INT nMaxKeys;
        INT nMaxEvents;
        UINT nFlags;
        ZG_CTR_SUB_TYPE nSubType;
} *PZG_FIND_CTR_INFO;
```

Delphi

```
  TZG_FIND_CTR_INFO = packed record
    nType             : TZG_CTR_TYPE;
    nTypeCode         : Byte;
    nAddr             : Byte;
    nSn               : Integer;
    nVersion          : Word;
    nMaxKeys          : Integer;
    nMaxEvents        : Integer;
    nFlags            : Cardinal;
    nSubType          : TZG_CTR_SUB_TYPE;
  end;
  PZG_FIND_CTR_INFO = ^TZG_FIND_CTR_INFO;
```

Параметры

Type
: Модель контроллера.

TypeCode
: Код модели контроллера.

Addr
: Сетевой адрес.

Sn
: Серийный номер.

Version
: Версия прошивки.

MaxKeys
: Максимум ключей.

MaxEvents
: Максимум событий.

Flags
: Флаги контроллера.

Флаг	Значение	Описание
`ZG_CTR_F_2BANKS`	0x0001	2 банка / 1 банк
`ZG_CTR_F_PROXIMITY`	0x0002	Proximity (Wiegand) / TouchMemory (Dallas) (параметр `READER` в Guard-Net)
`ZG_CTR_F_JOIN`	0x0004	Объединение двух банков (параметр `JOIN_BANK` в Guard-Net)
`ZG_CTR_F_X2`	0x0008	Удвоение ключей (параметр `X2_MODE` в Guard-Net)
`ZG_CTR_F_ELECTRO`	0x0010	Функция ElectroControl (для Matrix II Net с версией прошивки 3.X)
`ZG_CTR_F_MODES`	0x0020	Поддержка режимов прохода контроллера

Флаг	Значение	Описание
`ZG_CTR_F_DUAL_ZONE`	0x0040	Поддержка двух наборов временных зон
`ZG_CTR_F_APB`	0x0080	Поддержка AntiPassBack (APB)
`ZG_CTR_F_BIGTIME`	0x0100	Поддержка больших времен замка
`ZG_CTR_F_EXTASK`	0x0200	Поддержка запроса ExtAsk
`ZG_CTR_F_DUALKEY`	0x0400	Поддержка двойных карт (ключей с флагом `ZG_KF_DUAL`)
`ZG_CTR_F_BOOTMODE`	0x8000	Устройство в режиме "boot"

SubType

Подтип контроллера.

Структура ZG_CTR_INFO

Информация о контроллере, возвращаемая функциями: ZG_Ctr_Open и ZG_Ctr_GetInformation.

C++

```
typedef struct _ZG_CTR_INFO
{
        ZG_CTR_TYPE nType;
        BYTE nTypeCode;
        BYTE nAddr;
        INT nSn;
        WORD nVersion;
        INT nInfoLineCount;
        INT nMaxKeys;
        INT nMaxEvents;
        UINT nFlags;
        LPWSTR pszLinesBuf;
        INT nLinesBufMax;
        ZG_CTR_SUB_TYPE nSubType;
        INT nOptReadItems;
        INT nOptWriteItems;
} *PZG_CTR_INFO;
```

Delphi

```
  TZG_CTR_INFO = packed record
    nType            : TZG_CTR_TYPE;
    nTypeCode        : Byte;
    nAddr            : Byte;
    nSn              : Integer;
    nVersion         : Word;
    nInfoLineCount   : Integer;
    nMaxKeys         : Integer;
    nMaxEvents       : Integer;
    nFlags           : Cardinal;
```

```
    pszLinesBuf      : PWideChar;
    nLinesBufMax     : Integer;
    nSubType         : TZG_CTR_SUB_TYPE;
    nOptReadItems    : Integer;
    nOptWriteItems   : Integer;
  end;
  PZG_CTR_INFO = ^TZG_CTR_INFO;
```

Параметры

Type
: Модель контроллера.

TypeCode
: Код модели контроллера.

Addr
: Сетевой адрес.

Sn
: Серийный номер.

Version
: Версия прошивки.

InfoLineCount
: Количество строк с информацией.

MaxKeys
: Максимум ключей.

MaxEvents
: Максимум событий.

Flags
: Флаги контроллера.

Флаг	Значение	Описание
ZG_CTR_F_2BANKS	0x0001	2 банка / 1 банк
ZG_CTR_F_PROXIMITY	0x0002	Proximity (Wiegand) / TouchMemory (Dallas) (параметр READER в Guard-Net)

Флаг	Значение	Описание
ZG_CTR_F_JOIN	0x0004	Объединение двух банков (параметр JOIN_BANK в Guard-Net)
ZG_CTR_F_X2	0x0008	Удвоение ключей (параметр X2_MODE в Guard-Net)
ZG_CTR_F_ELECTRO	0x0010	Функция ElectroControl (для Matrix II Net с версией прошивки 3.X)
ZG_CTR_F_MODES	0x0020	Поддержка режимов прохода контроллера
ZG_CTR_F_DUAL_ZONE	0x0040	Поддержка двух наборов временных зон
ZG_CTR_F_APB	0x0080	Поддержка AntiPassBack (APB)
ZG_CTR_F_BIGTIME	0x0100	Поддержка больших времен замка
ZG_CTR_F_EXTASK	0x0200	Поддержка запроса ExtAsk
ZG_CTR_F_DUALKEY	0x0400	Поддержка двойных карт (ключей с флагом ZG_KF_DUAL)
ZG_CTR_F_BOOTMODE	0x8000	Устройство в режиме "boot"

LinesBuf

Буфер для информационных строк. Может быть равен **null**.

LinesBufMax

Размер буфера в символах, включая завершающий нулевой.

SubType

Подтип контроллера.

OptReadItems

Количество элементов, которое может быть считано одним запросом контроллеру. Если активен флаг `ZG_CTR_F_X2`, то для ключей этот параметр нужно удвоить.

OptWriteItems

Количество элементов, которое может быть записано одним запросом контроллеру. Если активен флаг `ZG_CTR_F_X2`, то для ключей этот параметр нужно удвоить.

Структура ZG_CTR_NOTIFY_SETTINGS

Параметры для уведомлений контроллера, используется для функции ZG_Ctr_SetNotification.

C++

```
typedef struct _ZG_CTR_NOTIFY_SETTINGS
{
        UINT nNMask;
        HANDLE hEvent;
        HWND hWindow;
        UINT nWndMsgId;
        INT nReadEvIdx;
        DWORD nCheckStatePeriod;
        DWORD nClockOffs;
} *PZG_CTR_NOTIFY_SETTINGS;
```

Delphi

```
  TZG_CTR_NOTIFY_SETTINGS = packed record
    nNMask           : Cardinal;
    hEvent           : THandle;
    hWindow          : HWnd;
    nWndMsgId        : Cardinal;
    nReadEvIdx       : Integer;
    nCheckStatePeriod: Cardinal;
    nClockOffs       : Cardinal;
  end;
  PZG_CTR_NOTIFY_SETTINGS =
^TZG_CTR_NOTIFY_SETTINGS;
```

Параметры

NMask

Маска типов уведомлений:

Флаг	Значение	Описание
ZG_NF_CTR_NEW_EVENT	1	Уведомления о новых событиях контроллера

Флаг	Значение	Описание
		(ZG_N_CTR_NEW_EVENT).
ZG_NF_CTR_CLOCK	2	Уведомления о рассинхронизации часов контроллера с часами ПК (ZG_N_CTR_CLOCK).
ZG_NF_CTR_KEY_TOP	4	Уведомления об изменении верхней границы ключей (ZG_N_CTR_KEY_TOP).
ZG_NF_CTR_ADDR_CHANGE	8	Уведомления об изменении сетевого адреса (ZG_N_CTR_ADDR_CHANGE).
ZG_NF_CTR_ERROR	10h	Уведомление об ошибках в потоке (thread) (ZG_N_CTR_ERROR)

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**.

nWndMsgId

Сообщение, которое будет отправляется окну когда появятся новые уведомления.

ReadEvIdx

Указатель чтения событий.

CheckStatePeriod

Период проверки состояния контроллера в миллисекундах (часы, указателей событий, верхней границы ключей). Если равно 0, то используется значение по умолчанию (1000).

ClockOffs

Допустимое расхождение часов контроллера с часами ПК в секундах.

Структура ZG_N_NEW_EVENT_INFO

Информация для уведомления ZG_N_CTR_NEW_EVENT.

C++

```
typedef struct _ZG_N_NEW_EVENT_INFO
{
        INT nNewCount;
        INT nWriteIdx;
        INT nReadIdx;
        Z_KEYNUM rLastNum;
} *PZG_N_NEW_EVENT_INFO;
```

Delphi

```
  TZG_N_NEW_EVENT_INFO = packed record
    nNewCount        : Integer;
    nWriteIdx        : Integer;
    nReadIdx         : Integer;
    rLastNum         : TZ_KEYNUM;
  end;
  PZG_N_NEW_EVENT_INFO = ^TZG_N_NEW_EVENT_INFO;
```

Параметры

nNewCount
: Количество новых событий.

nWriteIdx
: Указатель записи.

nReadIdx
: Указатель чтения.

rLastNum
: Номер последнего поднесенного ключа.

Структура ZG_N_KEY_TOP_INFO

Информация для уведомления ZG_N_CTR_KEY_TOP.

C++

```
typedef struct _ZG_N_KEY_TOP_INFO
{
        INT nBankN;
        INT nNewTopIdx;
        INT nOldTopIdx;
} *PZG_N_KEY_TOP_INFO;
```

Delphi

```
  TZG_N_KEY_TOP_INFO = packed record
    nBankN           : Integer;
    nNewTopIdx       : Integer;
    nOldTopIdx       : Integer;
  end;
  PZG_N_KEY_TOP_INFO = ^TZG_N_KEY_TOP_INFO;
```

Параметры

nBankN
: Номер банка ключей.

nNewTopIdx
: Новое значение верхней границы ключей.

nOldTopIdx
: Старое значение верхней границы ключей.

Структура ZG_CTR_TIMEZONE

Временная зона контроллера.

C++

```
typedef struct _ZG_CTR_TIMEZONE
{
        BYTE nDayOfWeeks;
        BYTE nBegHour;
        BYTE nBegMinute;
        BYTE nEndHour;
        BYTE nEndMinute;
        BYTE Reserved[3];
        ZG_CTR_MODE nMode;
} *PZG_CTR_TIMEZONE;
```

Delphi

```
  TZG_CTR_TIMEZONE = packed record
    nDayOfWeeks     : Byte;
    nBegHour        : Byte;
    nBegMinute      : Byte;
    nEndHour        : Byte;
    nEndMinute      : Byte;
    Reserved        : array[0..2] of Byte;
    nMode           : TZG_CTR_MODE;
  end;
  PZG_CTR_TIMEZONE = ^TZG_CTR_TIMEZONE;
```

Параметры

DayOfWeeks
Дни недели.
BegHour
Начало: час.

BegMinute
: Начало: минута.

EndHour
: Конец: час.

EndMinute
: Конец: минута.

Reserved
: Зарезервировано (для выравнивания размера структуры).

Mode
: Режим контроллера. Используется только для 2 временных зон, начинающихся с индекса `ZG_MODES_TZ0`.

ZG_CTR_KEY_TYPE

Тип ключа.

C++

```
enum ZG_CTR_KEY_TYPE : INT
{
        ZG_KEY_UNDEF = 0,
        ZG_KEY_NORMAL,
        ZG_KEY_BLOCKING,
        ZG_KEY_MASTER
};
```

Delphi

```
  TZG_CTR_KEY_TYPE = (
          ZG_KEY_UNDEF = 0,
          ZG_KEY_NORMAL,
          ZG_KEY_BLOCKING,
          ZG_KEY_MASTER
  );
```

Константа	Описание
ZG_KEY_UNDEF	Не известно
ZG_KEY_NORMAL	Обычный
ZG_KEY_BLOCKING	Блокирующий
ZG_KEY_MASTER	Мастер

Структура ZG_CTR_KEY

Ключ контроллера.

C++

```
typedef struct _ZG_CTR_KEY
{
        BOOL fErased;
        Z_KEYNUM rNum;
        ZG_CTR_KEY_TYPE nType;
        UINT nFlags;
        UINT nAccess;
        BYTE aData1[4];
} *PZG_CTR_KEY;
```

Delphi

```
  TZG_CTR_KEY = packed record
    fErased          : LongBool;
    rNum             : TZ_KEYNUM;
    nType            : TZG_CTR_KEY_TYPE;
    nFlags           : Cardinal;
    nAccess          : Cardinal;
    aData1           : array[0..3] of Byte;
  end;
  PZG_CTR_KEY = ^TZG_CTR_KEY;
```

Параметры

Erased
: True, если ключ стерт.

Num
: Номер ключа.

Type
: Тип ключа.

Flags

Флаги.

Константа	Значение	Описание
ZG_KF_FUNCTIONAL	2	Функциональная карта (для переключения режимов)
ZG_KF_DUAL	4	Двойная карта (контроллер проверяет 2 номера карты или номер карты и цифровой код)
ZG_KF_SHORTNUM	20h	Короткий номер. Если флаг ZG_CTR_F_PROXIMITY сброшен, то контроллер будет проверять только первые 3 байта номера ключа.

Access

Доступ.

Значения:

- FFh - проход разрешен всегда, вне зависимости от расписания,
- 00h - проход запрещен,
- 01h..7Fh - маска временных зон. Номер бита в маске соответствует номеру временной зоны (0-6).

Data1

Другие данные ключа (для специалистов). Если флаг ZG_CTR_F_PROXIMITY установлен, то первые 3 байта содержат данные записи в контроллере со смещением 3. Четвертый байт - данные записи со смещением 6.

Структура ZG_CTR_CLOCK

Часы контроллера.

C++

```
typedef struct _ZG_CTR_CLOCK
{
        BOOL fStopped;
        WORD nYear;
        WORD nMonth;
        WORD nDay;
        WORD nHour;
        WORD nMinute;
        WORD nSecond;
} *PZG_CTR_CLOCK;
```

Delphi

```
  TZG_CTR_CLOCK = packed record
    fStopped         : LongBool;
    nYear            : Word;
    nMonth           : Word;
    nDay             : Word;
    nHour            : Word;
    nMinute          : Word;
    nSecond          : Word;
  end;
  PZG_CTR_CLOCK = ^TZG_CTR_CLOCK;
```

Параметры

Stopped
: True, если часы остановлены.

Year
: Год.

Month
: Месяц.
Day
: День.
Hour
: Час.
Minute
: Минута.
Second
: Секунда.

ZG_CTR_EV_TYPE

Тип события контроллера.

Константа	Функция декодирования	Описание
ZG_EV_UNKNOWN	-	Не известное
ZG_EV_BUT_OPEN	ZG_Ctr_DecodePassEvent	Открыто кнопкой изнутри
ZG_EV_KEY_NOT_FOUND	ZG_Ctr_DecodePassEvent	Ключ не найден в банке ключей
ZG_EV_KEY_OPEN	ZG_Ctr_DecodePassEvent	Ключ найден, дверь открыта
ZG_EV_KEY_ACCESS	ZG_Ctr_DecodePassEvent	Ключ найден, доступ не разрешен
ZG_EV_REMOTE_OPEN	ZG_Ctr_DecodePassEvent	Открыто оператором по сети
ZG_EV_KEY_DOOR_BLOCK	ZG_Ctr_DecodePassEvent	Ключ найден, дверь заблокирована
ZG_EV_BUT_DOOR_BLOCK	ZG_Ctr_DecodePassEvent	Попытка открыть заблокированную дверь кнопкой
ZG_EV_NO_OPEN	ZG_Ctr_DecodePassEvent	Дверь взломана
ZG_EV_NO_CLOSE	ZG_Ctr_DecodePassEvent	Дверь оставлена открытой (тайм-аут)
ZG_EV_PASSAGE	ZG_Ctr_DecodePassEvent	Проход состоялся
ZG_EV_SENSOR1	ZG_Ctr_DecodePassEvent	Сработал датчик 1 (охрана)
ZG_EV_SENSOR2	ZG_Ctr_DecodePassEvent	Сработал датчик 2 (пожар)
ZG_EV_REBOOT	ZG_Ctr_DecodePassEvent	Перезагрузка контроллера
ZG_EV_BUT_BLOCK	ZG_Ctr_DecodePassEvent	Заблокирована кнопка открывания

Константа	Функция декодирования	Описание
ZG_EV_DBL_PASSAGE	ZG_Ctr_DecodePassEvent	Попытка двойного прохода
ZG_EV_OPEN	ZG_Ctr_DecodePassEvent	Дверь открыта штатно
ZG_EV_CLOSE	ZG_Ctr_DecodePassEvent	Дверь закрыта
ZG_EV_POWEROFF	ZG_Ctr_DecodePassEvent	Пропало питание
ZG_EV_ELECTRO_ON	ZG_Ctr_DecodeEcEvent	Включение электропитания
ZG_EV_ELECTRO_OFF	ZG_Ctr_DecodeEcEvent	Выключение электропитания
ZG_EV_LOCK_CONNECT	ZG_Ctr_DecodePassEvent	Включение замка (триггер)
ZG_EV_LOCK_DISCONNECT	ZG_Ctr_DecodePassEvent	Отключение замка (триггер)
ZG_EV_MODE_STATE	ZG_Ctr_DecodeModeEvent	Переключение режимов работы контроллера
ZG_EV_FIRE_STATE	ZG_Ctr_DecodeFireEvent	Изменение состояния Пожара
ZG_EV_SECUR_STATE	ZG_Ctr_DecodeSecurEvent	Изменение состояния Охраны
ZG_EV_UNKNOWN_KEY	ZG_Ctr_DecodeUnkKeyEvent	Неизвестный ключ
ZG_EV_GATEWAY_PASS	ZG_Ctr_DecodePassEvent	Совершен вход в шлюз
ZG_EV_GATEWAY_BLOCK	ZG_Ctr_DecodePassEvent	Заблокирован вход в шлюз (занят)
ZG_EV_GATEWAY_ALLOWED	ZG_Ctr_DecodePassEvent	Разрешен вход в шлюз
ZG_EV_ANTIPASSBACK	ZG_Ctr_DecodePassEvent	Заблокирован проход (Антипассбек)
ZG_EV_HOTEL40	ZG_DecodeHotelEvent	Изменилось состояние Hotel

Константа	Функция декодирования	Описание
ZG_EV_HOTEL41	ZG_DecodeHotelEvent	Изменилось состояние Hotel

После чтения событие нужно декодировать функцией, соответствующей типу события (смотрите таблицу выше).

ZG_CTR_DIRECT

Направление прохода.

C++

```
enum ZG_CTR_DIRECT : INT
{
        ZG_DIRECT_UNDEF = 0,
        ZG_DIRECT_IN,
        ZG_DIRECT_OUT
};
```

Delphi

```
  TZG_CTR_DIRECT = (
    ZG_DIRECT_UNDEF = 0,
    ZG_DIRECT_IN,
    ZG_DIRECT_OUT
  );
```

Константа	Описание
ZG_DIRECT_UNDEF	Не известно
ZG_DIRECT_IN	Вход
ZG_DIRECT_OUT	Выход

Структура ZG_CTR_EVENT

Событие контроллера.

C++

```
typedef struct _ZG_CTR_EVENT
{
        ZG_CTR_EV_TYPE nType;
        union
        {
                struct
                {
                        BYTE nCode;
                        BYTE aParams[7];
                } ep;
                BYTE aData[8];
        };
} *PZG_CTR_EVENT;
```

Delphi

```
  TZG_CTR_EVENT = packed record
    nType            : TZG_CTR_EV_TYPE;
    case u_ed: Integer of
      0:
      (
        nCode        : Byte;
        aParams      : array[0..6] of Byte;
      );
      1: ( aData     : array[0..7] of Byte; );
  end;
  PZG_CTR_EVENT = ^TZG_CTR_EVENT;
```

Параметры

Type
: Тип события.

nCode
: Код события в контроллере.

aParams
: Параметры события.

aData
: Данные события. Используйте функцию декодирования, соответстующую типу события.

Структура ZG_EV_TIME

Время события.

C++

```
typedef struct _ZG_EV_TIME
{
        BYTE nMonth;
        BYTE nDay;
        BYTE nHour;
        BYTE nMinute;
        BYTE nSecond;
        BYTE Reserved[3];
} *PZG_EV_TIME;
```

Delphi

```
  TZG_EV_TIME = packed record
    nMonth           : Byte;
    nDay             : Byte;
    nHour            : Byte;
    nMinute          : Byte;
    nSecond          : Byte;
    Reserved         : array[0..2] of Byte;
  end;
  PZG_EV_TIME = ^TZG_EV_TIME;
```

Параметры

nMonth
: Месяц

nDay
: День.

nHour
: Час.

nMinute
: Минута.

nSecond
: Секунда.

Reserved
: Зарезервировано (для выравнивания размера структуры).

ZG_FIRE_SUB_EV

Условие, вызвавшее событие ZG_EV_FIRE_STATE (для режима Пожар).

C++

```
enum ZG_FIRE_SUB_EV : INT
{
        ZG_FR_EV_UNDEF = 0,
        ZG_FR_EV_OFF_NET,
        ZG_FR_EV_ON_NET,
        ZG_FR_EV_OFF_INPUT_F,
        ZG_FR_EV_ON_INPUT_F,
        ZG_FR_EV_OFF_TEMP,
        ZG_FR_EV_ON_TEMP
};
```

Delphi

```
  TZG_FIRE_SUB_EV = (
    ZG_FR_EV_UNDEF = 0,
    ZG_FR_EV_OFF_NET,
    ZG_FR_EV_ON_NET,
    ZG_FR_EV_OFF_INPUT_F,
    ZG_FR_EV_ON_INPUT_F,
    ZG_FR_EV_OFF_TEMP,
    ZG_FR_EV_ON_TEMP
  );
```

Константа	Описание
ZG_FR_EV_UNDEF	Не определено
ZG_FR_EV_OFF_NET	Выключено по сети
ZG_FR_EV_ON_NET	Включено по сети
ZG_FR_EV_OFF_INPUT_F	Выключено по входу FIRE
ZG_FR_EV_ON_INPUT_F	Включено по входу FIRE

Константа	Описание
ZG_FR_EV_OFF_TEMP	Выключено по датчику температуры
ZG_FR_EV_ON_TEMP	Включено по датчику температуры

ZG_SECUR_SUB_EV

Условие, вызвавшее событие ZG_EV_SECUR_STATE (режим Охрана).

C++

```
enum ZG_SECUR_SUB_EV : INT
{
        ZG_SR_EV_UNDEF = 0,
        ZG_SR_EV_OFF_NET,
        ZG_SR_EV_ON_NET,
        ZG_SR_EV_OFF_INPUT_A,
        ZG_SR_EV_ON_INPUT_A,
        ZG_FR_EV_OFF_TAMPERE,
        ZG_FR_EV_ON_TAMPERE,
        ZG_FR_EV_OFF_DOOR,
        ZG_FR_EV_ON_DOOR
};
```

Delphi

```
  TZG_SECUR_SUB_EV = (
    ZG_SR_EV_UNDEF = 0,
    ZG_SR_EV_OFF_NET,
    ZG_SR_EV_ON_NET,
    ZG_SR_EV_OFF_INPUT_A,
    ZG_SR_EV_ON_INPUT_A,
    ZG_FR_EV_OFF_TAMPERE,
    ZG_FR_EV_ON_TAMPERE,
    ZG_FR_EV_OFF_DOOR,
    ZG_FR_EV_ON_DOOR
  );
```

Константа	Описание
ZG_SR_EV_UNDEF	Не определено

Константа	Описание
ZG_SR_EV_OFF_NET	Выключено по сети
ZG_SR_EV_ON_NET	Включено по сети
ZG_SR_EV_OFF_INPUT_A	Выключено по входу ALARM
ZG_SR_EV_ON_INPUT_A	Включено по входу ALARM
ZG_FR_EV_OFF_TAMPERE	Выключено по тамперу
ZG_FR_EV_ON_TAMPERE	Включено по тамперу
ZG_FR_EV_OFF_DOOR	Выключено по датчику двери
ZG_FR_EV_ON_DOOR	Включено по датчику двери

ZG_SECUR_MODE

Режим Охрана.

C++

```
enum ZG_SECUR_MODE : INT
{
        ZG_SR_M_UNDEF = 0,
        ZG_SR_M_SECUR_OFF,
        ZG_SR_M_SECUR_ON,
        ZG_SR_M_ALARM_OFF,
        ZG_SR_M_ALARM_ON
};
```

Delphi

```
  TZG_SECUR_MODE = (
    ZG_SR_M_UNDEF = 0,
    ZG_SR_M_SECUR_OFF,
    ZG_SR_M_SECUR_ON,
    ZG_SR_M_ALARM_OFF,
    ZG_SR_M_ALARM_ON
  );
```

Константа	Описание
ZG_SR_M_UNDEF	Не определено
ZG_SR_M_SECUR_OFF	Выключить режим охраны
ZG_SR_M_SECUR_ON	Включить режим охраны
ZG_SR_M_ALARM_OFF	Выключить тревогу
ZG_SR_M_ALARM_ON	Включить тревогу

ZG_CTR_MODE

Режим прохода контроллера.

C++

```
enum ZG_CTR_MODE : INT
{
        ZG_MODE_UNDEF = 0,
        ZG_MODE_NORMAL,
        ZG_MODE_BLOCK,
        ZG_MODE_FREE,
        ZG_MODE_WAIT
};
```

Delphi

```
  TZG_CTR_MODE = (
    ZG_MODE_UNDEF = 0,
    ZG_MODE_NORMAL,
    ZG_MODE_BLOCK,
    ZG_MODE_FREE,
    ZG_MODE_WAIT
  );
```

Константа	Описание
ZG_MODE_UNDEF	Неизвестно
ZG_MODE_NORMAL	Обычный режим работы
ZG_MODE_BLOCK	Блокировка. Проходить могут только "блокирующие" карты.
ZG_MODE_FREE	Свободный. Замок обесточен, при поднесении карты регистрируются.
ZG_MODE_WAIT	Ожидание. Обычный режим работы, при поднесении допустимой карты переход в режим "Free".

ZG_MODE_SUB_EV

Условие, вызвавшее событие ZG_EV_MODE_STATE (переключение режимов работы контроллера).

C++

```
enum ZG_MODE_SUB_EV : INT
{
        ZG_MD_EV_UNDEF = 0,
        ZG_MD_EV_RS485_ALLOW,
        ZG_MD_EV_RS485_DENIED,
        ZG_MD_EV_TZ_START,
        ZG_MD_EV_TZ_FINISH,
        ZG_MD_EV_CARD_ALLOW,
        ZG_MD_EV_CARD_DENIED
};
```

Delphi

```
  TZG_MODE_SUB_EV = (
    ZG_MD_EV_UNDEF = 0,
    ZG_MD_EV_RS485_ALLOW,
    ZG_MD_EV_RS485_DENIED,
    ZG_MD_EV_TZ_START,
    ZG_MD_EV_TZ_FINISH,
    ZG_MD_EV_CARD_ALLOW,
    ZG_MD_EV_CARD_DENIED
  );
```

Константа	Описание
ZG_EC_EV_UNDEF	Неизвестно
ZG_MD_EV_RS485_ALLOW	Установка командой по сети
ZG_MD_EV_RS485_DENIED	Отказано оператору по сети
ZG_MD_EV_TZ_START	Началась временная зона
ZG_MD_EV_TZ_FINISH	Окончилась временная зона

Константа	Описание
ZG_MD_EV_CARD_ALLOW	Установка картой
ZG_MD_EV_CARD_DENIED	Отказано изменению картой

ZG_HOTEL_MODE

Режим Hotel.

C++

```
enum ZG_HOTEL_MODE : INT
{
        ZG_HMODE_UNDEF = 0,
        ZG_HMODE_NORMAL,
        ZG_HMODE_BLOCK,
        ZG_HMODE_FREE,
        ZG_HMODE_RESERVED
};
```

Delphi

```
  TZG_HOTEL_MODE = (
    ZG_HMODE_UNDEF = 0,
    ZG_HMODE_NORMAL,
    ZG_HMODE_BLOCK,
    ZG_HMODE_FREE,
    ZG_HMODE_RESERVED
  );
```

Константа	Описание
ZG_HMODE_UNDEF	Неизвестно
ZG_HMODE_NORMAL	Обычный режим работы
ZG_HMODE_BLOCK	Блокирован
ZG_HMODE_FREE	Свободный проход
ZG_HMODE_RESERVED	Зарезервировано

ZG_HOTEL_SUB_EV

Условие, вызвавшее событие ZG_EV_HOTEL40 или ZG_EV_HOTEL41.

C++

```
enum ZG_HOTEL_SUB_EV : INT
{
        ZG_H_EV_UNDEF = 0,
        ZG_H_EV_FREECARD,
        ZG_H_EV_BLOCKCARD,
        ZG_H_EV_EXFUNC,
        ZG_H_EV_NEWRCARD,
        ZG_H_EV_NETWORK,
        ZG_H_EV_TIMEZONE,
        ZG_H_EV_COUNTER,
        ZG_H_EV_CRYPTOKEY,
        ZG_H_EV_PULSEZ,
        ZG_H_EV_STATE
};
```

Delphi

```
  TZG_HOTEL_SUB_EV = (
    ZG_H_EV_UNDEF = 0,
    ZG_H_EV_FREECARD,
    ZG_H_EV_BLOCKCARD,
    ZG_H_EV_EXFUNC,
    ZG_H_EV_NEWRCARD,
    ZG_H_EV_NETWORK,
    ZG_H_EV_TIMEZONE,
    ZG_H_EV_COUNTER,
    ZG_H_EV_CRYPTOKEY,
    ZG_H_EV_PULSEZ,
```

```
    ZG_H_EV_STATE
);
```

Константа	Описание
ZG_H_EV_UNDEF	Неизвестно
ZG_H_EV_FREECARD	Карта открытия
ZG_H_EV_BLOCKCARD	Карта блокирующая
ZG_H_EV_EXFUNC	Дополнительная функция
ZG_H_EV_NEWRCARD	Создана резервная карта
ZG_H_EV_NETWORK	
ZG_H_EV_TIMEZONE	
ZG_H_EV_COUNTER	Обновлен счетчик
ZG_H_EV_CRYPTOKEY	Обновлен криптоключ
ZG_H_EV_PULSEZ	Измененение защелки в течении 2х секунд
ZG_H_EV_STATE	Состояние защелки -если нажали ручку и отпустили более чем через 2 секунды

Структура ZG_CTR_CONFIGURATION

Параметры конфигурации контроллера, используется функциями ZG_GetCtrConfigParams, ZG_SetCtrConfigParams.

C++

```
typedef struct _ZG_CTR_CONFIGURATION
{
        ZG_CTR_TYPE nModel;
        ZGCTRCFGM nMask;
        ZGCTRCFGF nFlags;
        INT nBankSizeN;
        ZG_CTR_SUB_TYPE nPassPtType;
} *PZG_CTR_CONFIGURATION;
```

Delphi

```
  TZG_CTR_CONFIGURATION = packed record
    nModel           : TZG_CTR_TYPE;
    nMask            : TZGCTRCFGM;
    nFlags           : TZGCTRCFGF;
    nBankSizeN       : Integer;
    nPassPtType      : TZG_CTR_SUB_TYPE;
  end;
  PZG_CTR_CONFIGURATION =
^TZG_CTR_CONFIGURATION;
```

Параметры

Model

[in,out] Модель контроллера.

Mask

[in] Маска параметров для получения/установки.

Константа	Значение	Описание

ZG_CTRCFGM_RDWIEGAND	0x00000001	Режим считывателей: =True Wiegand, иначе - Dallas
ZG_CTRCFGM_WIEGAND26	0x00000002	Wiegand-26
ZG_CTRCFGM_DUALBANK	0x00000004	2 банка
ZG_CTRCFGM_JOIN	0x00000008	Режим Join
ZG_CTRCFGM_APB	0x00000010	Антипассбэк
ZG_CTRCFGM_DUALZONE	0x00000020	Два набора временных зон
ZG_CTRCFGM_FLAGS	0x0000003F	nFlags. Флаги
ZG_CTRCFGM_BANKSIZEN	0x00010000	Номер размера 1 банка.
ZG_CTRCFGM_PASSPTTYPE	0x00020000	Тип точки прохода.
ZG_CTRCFGM_NOCHECKCRC	0x80000000	Не проверять контрольную сумму.

Flags

[in,out] Флаги конфигурации контроллера.

Константа	Значение	Описание
ZG_CTRCFGF_RDWIEGAND	0x00000001	Режим считывателей: =True Wiegand, иначе - Dallas
ZG_CTRCFGF_WIEGAND26	0x00000002	Wiegand-26
ZG_CTRCFGF_DUALBANK	0x00000004	2 банка
ZG_CTRCFGF_JOIN	0x00000008	Режим Join
ZG_CTRCFGF_APB	0x00000010	Антипассбэк
ZG_CTRCFGF_DUALZONE	0x00000020	Два набора временных зон

BankSizeN

[in,out] Номер размера 1 банка:

Номер	Размер
0	512
1	1024
2	2048
3	4096
4	8192

PassPtType

[in,out] Тип точки прохода.

Примечание

Параметры, поддерживаемые контроллерами:

Параметр		Z-5R Net	Matrix II WiFi	Z-5R Web
Флаг "Режим считывателей"	Flags	+	+	+
Флаг "Wiegand-26"	Flags	+	+	+
Флаг "2 банка"	Flags	-	+	+
Флаг "Режим Join"	Flags	-	+	+
Флаг "Антипассбэк"	Flags	-	+	+
Флаг "Два набора временных зон"	Flags	+	+	+
Номер размера 1 банка	BankSizeN	-	+	+
Тип точки прохода	PassPtType	+	+	+

ZG_EC_SUB_EV

Условие, вызвавшее событие ZG_EV_ELECTRO_ON или ZG_EV_ELECTRO_OFF. (для функции ElectroControl - управление электропитанием).

C++

```
enum ZG_EC_SUB_EV : INT
{
        ZG_EC_EV_UNDEF = 0,
        ZG_EC_EV_CARD_DELAY,
        ZG_EC_EV_RESERVED1,
        ZG_EC_EV_ON_NET,
        ZG_EC_EV_OFF_NET,
        ZG_EC_EV_ON_SCHED,
        ZG_EC_EV_OFF_SHED,
        ZG_EC_EV_CARD,
        ZG_EC_EV_RESERVED2,
        ZG_EC_EV_OFF_TIMEOUT,
        ZG_EC_EV_OFF_EXIT
};
```

Delphi

```
  TZG_EC_SUB_EV = (
    ZG_EC_EV_UNDEF = 0,
    ZG_EC_EV_CARD_DELAY,
    ZG_EC_EV_RESERVED1,
    ZG_EC_EV_ON_NET,
    ZG_EC_EV_OFF_NET,
    ZG_EC_EV_ON_SCHED,
    ZG_EC_EV_OFF_SHED,
    ZG_EC_EV_CARD,
    ZG_EC_EV_RESERVED2,
    ZG_EC_EV_OFF_TIMEOUT,
```

```
    ZG_EC_EV_OFF_EXIT
);
```

Константа	Описание
ZG_EC_EV_UNDEF	Не известно
ZG_EC_EV_CARD_DELAY	Поднесена валидная карта с другой стороны (для входа) запущена задержка
ZG_EC_EV_RESERVED1	(зарезервировано)
ZG_EC_EV_ON_NET	Включено командой по сети
ZG_EC_EV_ON_SCHED	Включено по временной зоне
ZG_EC_EV_OFF_SHED	Выключено по временной зоне
ZG_EC_EV_CARD	Поднесена валидная карта к контрольному устройству
ZG_EC_EV_RESERVED2	(зарезервировано)
ZG_EC_EV_OFF_TIMEOUT	Выключено после отработки таймаута
ZG_EC_EV_OFF_EXIT	Выключено по срабатыванию датчика выхода

Структура ZG_CTR_ELECTRO_CONFIG

Определяет параметры электопитания контроллера.

C++

```
typedef struct _ZG_CTR_ELECTRO_CONFIG
{
        DWORD nPowerConfig;
        DWORD nPowerDelay;
        _ZG_CTR_TIMEZONE rTz6;
} *PZG_CTR_ELECTRO_CONFIG;
```

Delphi

```
TZG_CTR_ELECTRO_CONFIG = packed record
    nPowerConfig    : Cardinal;
    nPowerDelay     : Cardinal;
    rTz6            : TZG_CTR_TIMEZONE;
end;
PZG_CTR_ELECTRO_CONFIG =
^TZG_CTR_ELECTRO_CONFIG;
```

Параметры

PowerConfig

Флаги:

Флаг	Значение	Описание
ZG_CTR_ECF_ENABLED	1	Задействовать управление питанием
ZG_CTR_ECF_SCHEDULE	2	Использовать временную зону 6 для включения питания

ZG_CTR_ECF_EXT_READER	4	Контрольный считыватель: «0» Matrix-II Net или Matrix-III Net, «1» внешний считыватель
ZG_CTR_ECF_INVERSE	8	Инвертировать управляющий выход
ZG_CTR_ECF_EXIT_OFF	10h	Задействовать датчик двери
ZG_CTR_ECF_CARD_OPEN	20h	Не блокировать функцию открывания для контрольного считывателя

PowerDelay

Время задержки в секундах.

Tz6

Параметры временной зоны №6 (считаем от 0).

Структура ZG_CTR_ELECTRO_STATE

Определяет состояние электопитания контроллера.

C++

```
typedef struct _ZG_CTR_ELECTRO_STATE
{
        DWORD nPowerFlags;
        DWORD nPowerConfig;
        DWORD nPowerDelay;
} *PZG_CTR_ELECTRO_CONFIG;
```

Delphi

```
TZG_CTR_ELECTRO_STATE = packed record
    nPowerFlags     : Cardinal;
    nPowerConfig    : Cardinal;
    nPowerDelay     : Cardinal;
end;
PZG_CTR_ELECTRO_STATE = ^TZG_CTR_ELECTRO_STATE;
```

Параметры

PowerFlags

Флаги состояния электропитания:

Флаг	Описание
ZG_CTR_ESF_ENABLED	Состояние питания – 1 вкл/0 выкл
ZG_CTR_ESF_SCHEDULE	Активно включение по временной зоне
ZG_CTR_ESF_REMOTE	Включено по команде по сети
ZG_CTR_ESF_DELAY	Идет отработка задержкид
ZG_CTR_ESF_CARD	Карта в поле контрольного считывателя

PowerConfig

Флаги настроек электропитания:

Флаг	Описание
ZG_CTR_ECF_ENABLED	Задействовать управление питанием
ZG_CTR_ECF_SCHEDULE	Использовать временную зону 6 для включения питания
ZG_CTR_ECF_EXT_READER	Контрольный считыватель: «0» Matrix-II Net, «1» внешний считыватель
ZG_CTR_ECF_INVERSE	Инвертировать управляющий выход
ZG_CTR_ECF_EXIT_OFF	Задействовать датчик двери
ZG_CTR_ECF_CARD_OPEN	Не блокировать функцию открывания для контрольного считывателя

PowerDelay

Время задержки в секундах.

Константы ZGuard API

Константы ZPort API; Значения по умолчанию, коды ошибок.

Константа	Значение	Описание
ZG_SDK_VER_MAJOR	3	Старший номер версии ZGuard API.
ZG_SDK_VER_MINOR	39	Младший номер версии ZGuard API.
ZG_DEF_CVT_LICN	5	Номер лицензии, которую использует SDK для работы с конвертером в режиме Advanced.
ZG_MAX_TIMEZONES	7	Количество временных зон в одном банке контроллера.
ZG_MODES_TZ0	-2	Позиция первой временной зоны для переключения режима контроллера, всего 2 зоны.
ZG_DUAL_ZONE_TZ0	-9	Позиция первой временной зоны во втором наборе, всего 7 зон.

Коды ошибок ZGuard API

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 (Конвертор отключен от USB?).

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 ZG_Initialize.
ZP_E_INSUFFICIENTBUFFER	8004020Ch	Размер буфера слишком мал.
ZP_E_NOCONNECT	8004020Dh	Нет связи с устройством.
ZG_E_TOOLARGEMSG	80040301h	Слишком большое сообщение для отправки.
ZG_E_NOANSWER	80040303h	Нет ответа.
ZG_E_BADANSWER	80040304h	Нераспознанный ответ.
ZG_E_WRONGZPORT	80040305h	Не правильная версия ZPort.dll
ZG_E_CVTBUSY	80040306h	Конвертер занят (при открытии конвертера в режиме "Proxy")
ZG_E_CVTERROR	80040307h	Другая ошибка конвертера.
ZG_E_LICNOTFOUND	80040308h	Ошибка конвертера: Нет такой лицензии.

Константа	Значение	Описание
ZG_E_LICEXPIRED	80040309h	Текущая лицензия истекла.
ZG_E_LICONTROLLERS	8004030Ah	Ошибка конвертера: ограничение лицензии на число.
ZG_E_LICREADKEYS	8004030Bh	Ограничение лицензии на число ключей при чтении.
ZG_E_LICWRITEKEYS	8004030Ch	Ограничение лицензии на число ключей при записи.
ZG_E_LICEXPIRED2	8004030Dh	Срок лицензии истек (определено при установке даты в контроллере).
ZG_E_NOCONVERTER	8004030Eh	Неверный адрес конвертера (конвертер не найден)
ZG_E_NOCONTROLLER	8004030Fh	Неверный адрес контроллера (контроллер не найден).
ZG_E_CTRNACK	80040310h	Контроллер отказал в выполнении команды.
ZG_E_FWBOOTLOADERNOSTART	80040311h	Загрузчик не запустился (при прошивке).
ZG_E_FWFILESIZE	80040312h	Некорректный размер файла (при прошивке).
ZG_E_FWNOSTART	80040313h	Не обнаружен старт прошивки. Попробуйте перезагрузить устройство (при прошивке).
ZG_E_FWNOCOMPATIBLE	80040314h	Не подходит для этого устройства (при

Константа	Значение	Описание
		прошивке).
ZG_E_FWINVALIDDEVNUM	80040315h	Не подходит для этого номера устройства (при прошивке).
ZG_E_FWTOOLARGE	80040316h	Слишком большой размер данных прошивки (при прошивке).
ZG_E_FWSEQUENCEDATA	80040317h	Некорректная последовательность данных (при прошивке).
ZG_E_FWDATAINTEGRITY	80040318h	Целостность данных нарушена (при прошивке).
ZG_E_WRONGCRC	80040319h	Неверная контрольная сумма (для ZG_GetCtrConfigParams / ZG_SetCtrConfigParams).
ZG_E_CTRISALREADYOPEN	8004031Ah	Контроллер уже открыт функцией ZG_Ctr_Open.

Значения по умолчанию ZGuard API

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

Константа	Значение	Описание
		(для детектора устройств).
ZP_DTC_SCANDEVPERIOD	FFFFFFFFh	Период сканирования устройств, опросом 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.

Константа	Значение	Описание
ZG_CVT_SCANCTRSPERIOD	5000	Период сканирования контроллеров.
ZG_CVT_SCANCTRSLASTADDR	31	Последней сканируемый адрес контроллера.

Конвертеры

Модель	Интерфейс подключения к ПК
Z-397	USB
Z-397 Guard	USB
Z-397 IP	Ethernet, USB (для конфигурирования и прошивки)
Z-397 Web	Ethernet, USB (для конфигурирования и прошивки)

Лицензия конвертера

Конвертеры – Z-397 Guard[Advanced], Z-397 IP, Z-397 Web.

Лицензия — это набор ограничений на количество обслуживаемых контроллеров и количество карт в каждом из контроллеров для определенной группы программ. Лицензия устанавливается в конвертер и храниться в его ПЗУ. Всего в конвертер может быть установлено до 16 лицензий для соответствущих ПО. Номер лицензии соответствует определенным ПО. SDK Guard и GuardCommander работают с лицензией №5, GuardLight - №8.

Если в конвертере нет установленной лицензии, то SDK при открытии конвертера (`ZG_Cvt_Open`) автоматически устанавливает лицензию на 16 контроллеров (остальные параметры не ограничены).

Ограничение на количество контроллеров (`MaxCtrs`) означает, что одновременно (параллельно) можно работать только с определенным количеством контроллеров (=`MaxCtrs`). Конвертер запоминает к каким контроллерам было подключение, увеличивая свой счетчик, когда становиться >`MaxCtrs`, то подключение к новому адресу будет недоступно. Чтобы сбросить счетчик нужно вызвать функцию поиска контроллеров `ZG_Cvt_EnumControllers`.

Для получения лицензии с особыми параметрами нужно запросить её у нас (фирме производителе).

Режимы работы IP-конвертера

Конвертеры – Z-397 IP, Z-397 Web.

Режим сервер

Конвертер после соединения с сетью и получения IP адреса ожидает установки соединения. Соединение устанавливает компьютер, расположенный в локальной или внешней сети.

Этот режим удобно использовать, когда известен IP адрес конвертера. Также преимуществом данного режима является возможность подключения к конвертору с разных компьютеров, как находящихся в локальной сети, так и по сети Internet.

Режим клиент

В режиме клиента, после получения IP адреса, конвертер пытается установить соединение с локальным или удалённым компьютером – сервером. При невозможности установления соединения попытка повторяется.

В данном режиме работы нет необходимости знать IP адреса всех конвертеров, входящих в систему. Все они будут соединяться с одним сервером самостоятельно. При этом нет возможности перенести сервер на другой компьютер без переконфигурирования всех конверторов.

Режим прокси

Конвертер активно пытается соединиться с прокси-сервером. На этот же прокси-сервер обращается компьютер. Поиск происходит по кодовому слову, заданному при конфигурировании конвертера. Этот метод используется для связи, когда конвертер и компьютер работают в разных сетях и установление прямого соединения невозможно.

В любом режиме соединение для обмена информацией устанавливается по TCP протоколу.

Примеры файлов конфигурации конверторов Z397-IP:

Сервер	Клиент	Прокси

# Configuration file for the device z387ip 	
# Version 2.01.117[Sep 6 2012 15:43:42] 	
# S/N:8 	
# Use DHCP? 	
USE_DHCP=1
 	
# Keep-Alive Settings 	
KEEP_ALIVE_TIME=10
 	
# Settings for static IP 	
LOCAL_IP=192.168.1.4
NETMASK=255.255.255.0
GATEWAY=192.168.1.1
DNS=192.168.1.1
 	
# Settings for Virtual COM port 	
RS422_MODE=0
RING_MODE=0
L1_PORT=1000
L2_PORT=1001
L1_RADDR=255.255.255.255
L1_RPORT=0
L2_RADDR=255.255.255.255
L2_RPORT=0
 	
ZPROXY_ID=
	# Configuration file for the device z387ip
	# Version 2.01.117[Sep 6 2012 15:43:42]
	# S/N:8
	# Use DHCP?
	USE_DHCP=1
	
	# Keep-Alive Settings
	KEEP_ALIVE_TIME=10
	
	# Settings for static IP
	LOCAL_IP=192.168.1.4
	NETMASK=255.255.255.0
	GATEWAY=192.168.1.1
	DNS=192.168.1.1
	
	# Settings for Virtual COM port
	RS422_MODE=0
	RING_MODE=0
	L1_PORT=1000
	L2_PORT=1001
	L1_RADDR=192.168.1.14
	L1_RPORT=25000
	L2_RADDR=255.255.255.255
	L2_RPORT=0
	
	ZPROXY_ID=
		# Configuration file for the device z387ip
		# Version 2.01.117[Sep 6 2012 15:43:42]
		# S/N:8
		# Use DHCP?
		USE_DHCP=1
		
		# Keep-Alive Settings
		KEEP_ALIVE_TIME=10
		
		# Settings for static IP
		LOCAL_IP=192.168.1.4
		NETMASK=255.255.255.0
		GATEWAY=192.168.1.1
		DNS=192.168.1.1
		
		# Settings for Virtual COM port
		RS422_MODE=0
		RING_MODE=0
		L1_PORT=1000
		L2_PORT=1001
		L1_RADDR=zproxy.con.ru
		L1_RPORT=25000
		L2_RADDR=255.255.255.255
		L2_RPORT=0
		
		ZPROXY_ID=**PASSWORD**

Сервер	Клиент	Прокси

L1_STARTCHAR=NONE
L1_ENDCHAR=NONE
L2_STARTCHAR=NONE
L2_ENDCHAR=NONE
 	
# Settings for the RS485 	
L1_BAUDRATE=19200
L2_BAUDRATE=19200
	L1_STARTCHAR=NONE
	L1_ENDCHAR=NONE
	L2_STARTCHAR=NONE
	L2_ENDCHAR=NONE
	
	# Settings for the RS485
	L1_BAUDRATE=19200
	L2_BAUDRATE=19200
		L1_STARTCHAR=NONE
		L1_ENDCHAR=NONE
		L2_STARTCHAR=NONE
		L2_ENDCHAR=NONE
		
		# Settings for the RS485
		L1_BAUDRATE=19200
		L2_BAUDRATE=19200

Конвертер Z-397

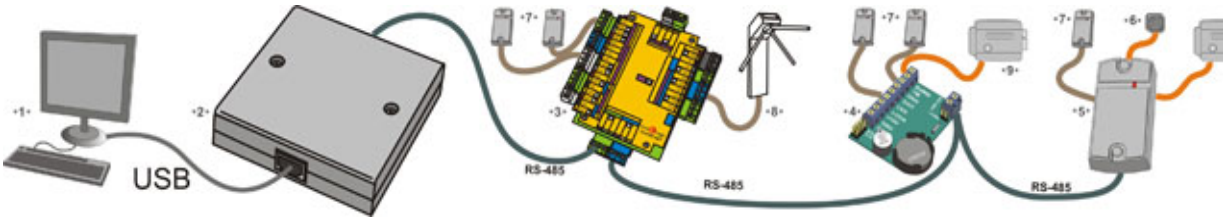
Страница на сайте www.ironlogic.ru

Спецификация

Скорость приёмо-передачи	до 200 кБит/сек
Питание	от порта USB
Материал корпуса	ABS пластик
Цвет корпуса	белый
Размер(mm)	65x65x18

Применение

Предназначен для преобразования интерфейса USB в RS485/422 и мониторинга событий, происходящих в системе через терминальную программу в реальном времени.



Конвертер Z-397 Guard

Обеспечивает преобразование коммуникационного интерфейса контроллеров (RS485) в интерфейс компьютера (USB) и далее, с помощью драйвера, в виртуальный COM-порт. Интерфейс RS485 – позволяет подключать устройства на расстояниях около одного километра. Питание конвертера осуществляется по кабелю USB от компьютера.

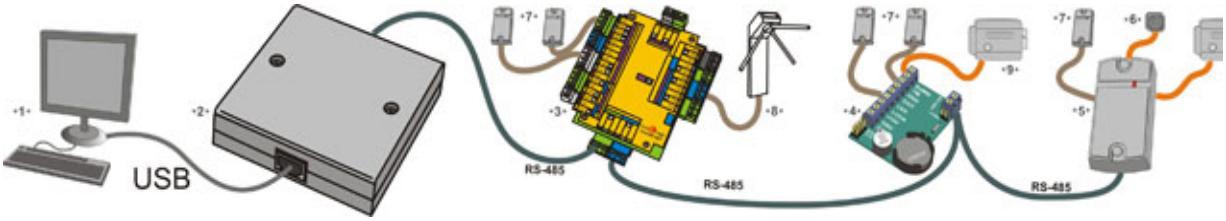
Страница на сайте www.ironlogic.ru

Спецификация

Скорость приёмо-передачи	до 115 кБит/сек
Питание	от порта USB
Материал корпуса	ABS пластик
Цвет корпуса	белый
Размер(mm)	65x65x18

Применение

Z-397 Guard предназначен для преобразования интерфейса USB в RS485.
Ускорение и облегчение диагностики и запуска сетей с сетевыми контроллерами Z-5R Net, MATRIX-II Net.



Режимы работы:

1. NORMAL - режим эмуляции работы Z-397.
2. TEST - режим проверки сети контроллеров. SDK не работает с этим режимом.
3. ACCEPT - режим для быстрого запуска установленной сети без установки ПО на компьютере. SDK не работает с этим режимом.
4. ADVANCED - режим для работы конвертера под управление специального ПО. В этом режиме конвертером реализуется ряд функций повышающих надежность ПО и обеспечивается лицензионная защита.

Нужный режим работы выбирается перестановкой перемычки №1 в одно из четырех положений. Соответствие положения и режима указано на плате конвертера.

Конвертер Z-397 IP

Обеспечивает подключение коммуникационного интерфейса контроллеров (RS485) в компьютеру по TCP/IP без создания виртуального COM-порта.

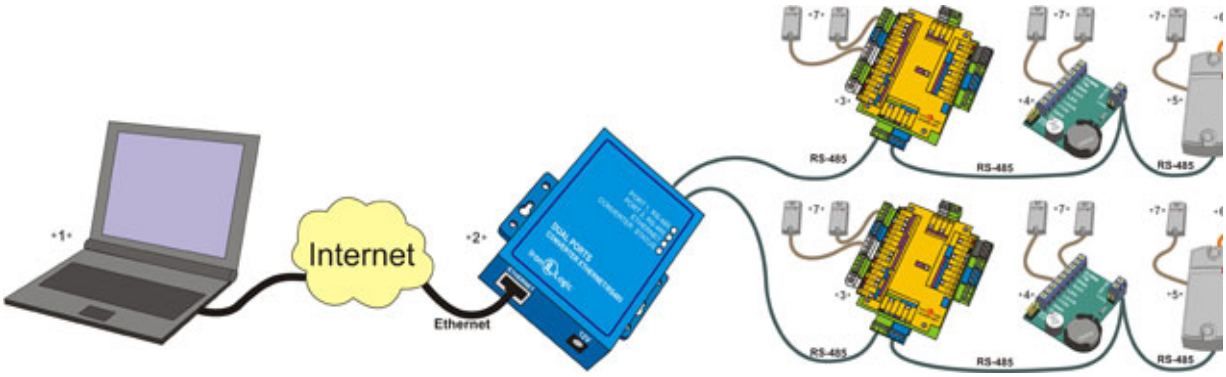
Снят с производства, заменен на Z-397 Web.

Спецификация

Ehternet	Интерфейс Ethernet RJ45 (10BASE-T, Ethernet II и IEEE 802.3); Протоколы IP, ARP, TCP, TELNET, ICMP, UDP, DHCP
RS485	Количество линий RS-485/RS-422: 2/1; Скорость: до 115200 бит/сек; Гальваническая развязка линии 1: 1 кВ; Гальваническая развязка линии 2: нет
USB	Разъём: USB Тип B; Версия: USB 2.0; Режим: Full-Speed; Класс: Mass-storage device
Напряжение питания	8-18V DC или 7-14V AC
Потребляемый ток	не более 100 мА (при 12V DC)
Температура хранения	-40°C ...+125°C
Рабочая температура	0°C ...+ 70°C
Влажность (без конденсации)	5%...95%
Материал корпуса	ABS пластик
Масса	не более 100 грамм
Размер(mm)	100x85x30

Применение

Z-397 IP предназначен для организации связи устройств, подключённых к линиям RS485(422), с удалённым компьютером через локальную сеть по протоколу TCP/IP. Ускорение и облегчение диагностики и запуска сетей с сетевыми контроллерами Z-5R Net, Z-5R Net(8000), MATRIX-II Net, Guard Net позволяет работать с контроллерами в режиме “Advanced”.
Возможно использовать бесплатные драйвера, например **HW VSP**, разработанные http://www.hw-group.com.



Режимы работы:

1. Сервер - ожидает подключения со стороны компьютера.
2. Клиент - постоянно пытается подключиться к компьютеру по IP, заданному в настройках конвертера.
3. Прокси - постоянно пытается подключиться к серверу, компьютер также подключается к серверу, сервер становиться посредником между конвертером и компьютером.

Конвертер Z-397 Web

Страница на сайте
www.ironlogic.ru

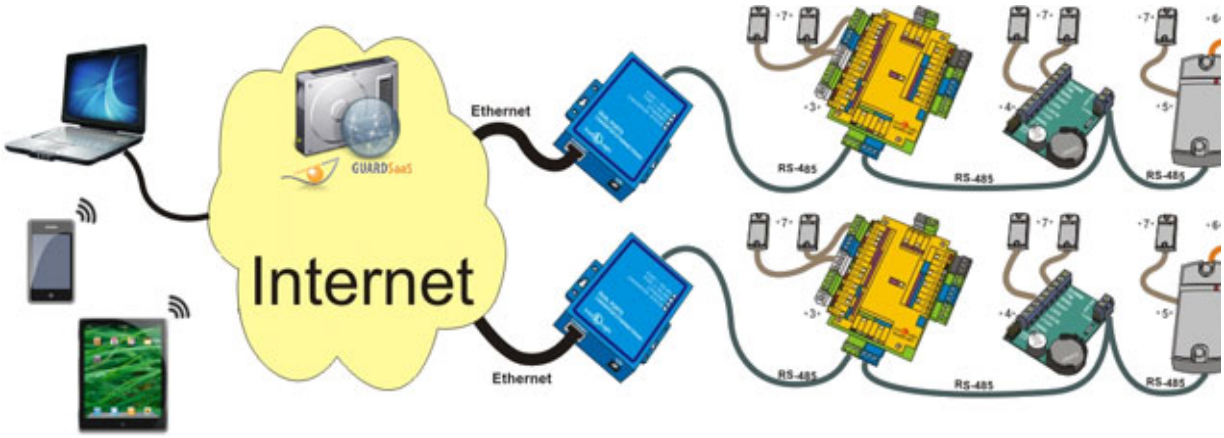
Спецификация

Ehternet	Интерфейс Ethernet RJ45 (10/100BASE-T, Ethernet II и IEEE 802.3); Протоколы IPv4 , ARP, TCP, TELNET ,ICMP, UDP, DHCP, HTTP,NVT
RS485	Количество линий RS-485 - 2 Скорость - до 115200 бит/сек Гальваническая развязка линии 1 - 1 кВ Гальваническая развязка линии 2 - нет
USB	Разъём - USB Тип B Версия - USB 2.0 Режим - Full-Speed Класс - Mass-storage device
Напряжение питания	8-18V DC или 7-14V AC
Потребляемый ток	не более 100 мА (при 12V DC)
Температура хранения	-40°C ...+125°C
Рабочая температура	0°C ...+ 70°C
Влажность (без конденсации)	5%...95%
Материал корпуса	ABS пластик
Масса	не более 100 грамм
Размер(mm)	100x85x30

Применение

Z-397 WEB предназначен для организации связи устройств, подключённых к линиям RS485, с удалённым компьютером через локальную сеть по протоколу TCP/IP. Ускорение и облегчение диагностики и запуска сетей с сетевыми контроллерами Z-5R Net, Z-5R Net(8000), MATRIX-II Net, Guard Net позволяет работать с контроллерами в режиме “Advanced”.

Пример построения сети в варианте подключения к облачному сервису **GuardSaaS**.
Страница переадресации конвертера Z397 Web на доступные сервисы здесь --> http://hw.rfenabled.com/



Сетевые контроллеры

Модель	Количество банков	Максимум ключей	Максимум событий
Gate 2000	1	2024	2048
Guard-Net	2	8168	8192
Matrix II Net	1	2024	2048
Matrix III Net	1	2024	2048
Matrix-II Wi-Fi		2024	2048
Matrix-VI NFC Net 8k		8168	8192
Matrix-VI EHK Net 2k			
Matrix VI Wi-Fi			
Matrix-II EH Web			
Matrix-VI EH Web			
Z5r-Net	1	2024	2048
Z5r-Net 8000	1	8168	8192
Z-5R Web	2	8168	8192
Z5R-WEB BT	2	8168	8192
Z5R Wi-Fi		2024	2048
Z5R Web mini			
Z-9 EHT net	1	2024	2048
Eurolock EHT net	1	2024	2048

Параметры контроллера

1. Банк*
    1. Времена замков
        1. Время открытия двери
        2. Время контроля закрытой двери
        3. Время контроля открытой двери
    2. Временные зоны
    3. Ключи
    4. Верхняя граница ключей
2. События
3. Указатель записи и указатель чтения
4. Часы
5. Номер последнего поднесенного ключа
6. Режим "Аварийное открывание дверей"
7. Параметры электропитания (только для Matrix II Net с прошивкой "ElectroControl")
8. Время антипассбэк (APB) контроллера

* Под словом "Банк" подразумевается список ключей + список временных зон + времена замков + верхняя граница ключей. Если у контроллера 2 банка, то банк №0 содержит параметры для входа, банк №1 - для выхода, иначе - параметры для входа и выхода содержатся в банке №0. У контроллера Guard-Net по-умолчанию 2 банка, у остальных контроллеров (Z5R Net, Z5r Net 8000, Matrix II Net, Gate 2000) всегда 1 банк.

Времена замков

Время открытия двери – время, на которое подается отпирающее напряжение в исполнительный механизм замка (*для электромагнитных это длительность обесточивания, для электромеханических - длительность импульса напряжения. Тип замка задается перемычкой на плате контроллера*). Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. Особым случаем является время, равное 0 – в этом случае замок блокируется и не может быть открыт ни ключом, ни кнопкой, ни командой извне до тех пор, пока время не будет изменено на любое ненулевое. При попытке открыть заблокированную дверь формируется событие «дверь заблокирована» (ZG_EV_KEY_DOOR_BLOCK) (если при этом использовался ключ из числа имеющихся в соответствующем банке ключей, его индекс помещается в соответствующее поле описателя события; если дверь пытались открыть кнопкой, поле индекса ключа в описателе события заполняется нулями).

Время контроля закрытой двери – определяет промежуток после открывания замка, в течение которого человек должен открыть дверь. Контроллер позволяет устанавливать значения от 0.1 до 25.5с с дискретностью 0.1с. В момент открывания двери формируется соответствующее событие – «открыто ключом» (ZG_EV_OPEN_KEY) или «открыто кнопкой» (ZG_EV_BUT_OPEN). Если за это время дверь не была открыта, никаких событий не формируется, но при последующей попытке открыть дверь без ключа/кнопки (т.к. электромеханический замок остается открытым и после снятия напряжения, если не была открыта дверь) фиксируется событие «дверь взломана» (ZG_EV_NO_OPEN). Особым случаем является время, равное 0 – в этом случае контроль не производится и попытки взлома двери не обнаруживаются, а

события «открыто ключом», «открыто кнопкой» фиксируются в момент подачи напряжения на исполнительный механизм. Эта установка используется при отсутствии дверного геркона.

Время контроля открытой двери – определяет промежуток после открывания двери (0.1 – 25.5с), в течение которого дверь должна быть закрыта. Если по истечении этого времени дверь не была закрыта (что определяется по дверному геркону), фиксируется событие «дверь оставлена открытой» (ZG_EV_NO_CLOSE). Особым случаем является время, равное 0 – в этом случае контроль не производится. Эта установка используется при отсутствии дверного геркона.

Для чтения времен замков предназначена функция ZG_Ctr_ReadLockTimes, для записи – ZG_Ctr_WriteLockTimes.

Временные зоны

Временные зоны предназначены для организации пропуска по ключам только в заданные промежутки времени и в определенные дни недели. Контроллер имеет 7 временных зон (константа `ZG_MAX_TIMEZONES`) , для каждой из которых можно задать: список дней недели, для которых разрешен проход и время начала и конца прохода. Каждому ключу можно назначить одну или несколько временных зон.

Временная зона ограничивает доступ в течение недели, а если нужно ограничивать по дням, месяцам, и т.д., то нужно программно переписывать временную зону и(или) параметр `Access` ключа.

Для чтения временных зон предназначены функции `ZG_Ctr_ReadTimeZones`, `ZG_Ctr_EnumTimeZones`, для записи – `ZG_Ctr_WriteTimeZones`.

Ключи

Список ключей предназначен для хранения разрешенных для прохода ключей, а также для сервисных ключей (блокирующих- и мастер-ключей). Список представляет собой массив фиксированной длины. Каждый элемент массива может быть занят ключом или может быть свободным (вакантная пустая ячейка). При удалении ключа вначале массива, остальные ключи не смещаются. Максимальное количество ключей можно узнать с помощью функции ZG_Ctr_GetInformation.

Для каждого ключа можно задать его номер (RFID-карты), тип (обычный, мастер или блокирующий) и доступ (всегда разрешен, всегда запрещен или по расписанию – набор временных зон).

Для чтения ключей предназначены функции ZG_Ctr_EnumKeys, ZG_Ctr_ReadKeys, для записи – ZG_Ctr_WriteKeys.

Верхняя граница ключей

Верхняя граница ключей – это переменная, которая хранится в контроллере, она определяет позицию, начиная с которой идут только свободные ячейки ключей. Контроллер обновляет эту переменную автоматически при записи или стирании ключей. При поднесении ключа к считывателю контроллер ищет его номер в диапазоне от 0 до "верхней границы ключей" (не включительно).

Для чтения верхней границы ключей предназначена функция `ZG_Ctr_GetKeyTopIndex`.

События

Список событий предназначен для хранения информации о проходах и о других событиях. Список представляет собой массив фиксированной длины. Когда массив заполняется, то запись продолжается сначала, переписывая самые старые события. Максимальное количество событий можно узнать с помощью функции ZG_Ctr_GetInformation. Возможные типы событий можно посмотреть здесь.

Для чтения событий предназначены функции ZG_Ctr_ReadEvents, ZG_Ctr_EnumEvents.

Указатель записи и указатель чтения

Указатель записи – позиция, в которую контроллер пишет следующее событие. После записи очередного события, контроллер автоматически меняет этот указатель.

Указатель чтения – позиция следующего непрочитанного события. Если при записи нового события указатель записи становиться равным указателю чтения, то контроллер автоматически меняет указатель чтения (=указатель записи+1).

Количество новых событий можно вычислять по формуле: если *ReadEvIdx* < *WriteEvIdx*, то *NewCount* = *(WriteEvIdx - ReadEvIdx*), иначе *NewCount* = (*MaxEvents - ReadEvIdx + WriteEvIdx)*, где *NewCount – количество новых событий, WriteEvIdx – указатель записи, ReadEvIdx* – указатель чтения, *MaxEvents* – максимальное количество событий (можно узнать с помощью функции `ZG_Ctr_GetInformation`).

Для чтения указателей предназначены функции `ZG_Ctr_ReadEventIdxs`, `ZG_Ctr_ReadRTCState`, для записи – `ZG_Ctr_WriteEventIdxs`.

Часы

Текущие дата и время контроллера. Используются для расписаний и для новых событий.

Для получения часов предназначены функции `ZG_Ctr_GetClock`, `ZG_Ctr_ReadRTCState`, для установки – `ZG_Ctr_SetClock`.

Номер последнего поднесенного ключа

Контроллер хранит в своей памяти номер ключа, который был поднесен к его считывателю последним.

Для чтения номера предназначены функции ZG_Ctr_ReadLastKeyNum, ZG_Ctr_ReadRTCState.

Режим "Аварийное открывание дверей"

Этот режим используется при аварийном открывании дверей контроллера в случае пожара или иных аварийных ситуаций.

Для получения состояния предназначена функция `ZG_Ctr_IsEmergencyUnlockingEnabled`, для установки – `ZG_Ctr_EnableEmergencyUnlocking`.

Параметры электропитания (только для Matrix II Net с прошивкой "ElectroControl")

Параметры электропитания предназначены для настройки управления электрописанием: расписание включения/ выключения питания и другие опции.

Для чтения параметров предназначена функция ZG_Ctr_ReadElectroConfig, для записи – ZG_Ctr_WriteElectroConfig.

Время антипассбэк (APB) контроллера

Время APB в минутах с 0 по 65535. Время APB поддерживается когда установлен флаг ZG_CTR_F_APB.

Карточки, у которых в параметрах доступа стоит 0xFF, не блокируются по APB.
Если карта вошла(вышла), то повторный вход(выход) блокируется на заданное время.
Блокированная карта сбрасывается просто перезаписью её обратно в память.
После сброса или истечении времени движение разрешено в любом направлении.
Блокирована карта или нет узнать никак нельзя, только по событиям или самостоятельно запустить таймер в момент прохода и вычислить.

Для чтения APB предназначена функция ZG_Ctr_ReadApbTime, для записи – ZG_Ctr_WriteApbTime.

Время антипассбэк (APB) контроллера

Время APB в минутах с 0 по 65535. Время APB поддерживается когда установлен флаг ZG_CTR_F_APB.

Карточки, у которых в параметрах доступа стоит 0xFF, не блокируются по APB.
Если карта вошла(вышла), то повторный вход(выход) блокируется на заданное время.
Блокированная карта сбрасывается просто перезаписью её обратно в память.
После сброса или истечении времени движение разрешено в любом направлении.
Блокирована карта или нет узнать никак нельзя, только по событиям или самостоятельно запустить таймер в момент прохода и вычислить.

Для чтения APB предназначена функция ZG_Ctr_ReadApbTime, для записи – ZG_Ctr_WriteApbTime.

Сетевой контроллер Z-5R Net

Страница на сайте IronLogic.ru

Спецификация

Напряжение питания	8-18V DC
Ток потребления	20 mA
Количество подключаемых считывателей	2
Тип (протокол) подключаемых считывателей	Dallas Touch Memory
Выходы МДП транзистор	1
Ток коммутации	5A
Количество ключей/карт(max)	2024
Количество запоминаемых событий(max)	2048
Протокол связи с контроллерами	RS485
Скорость связи	19200 бод/57600бод
Максимальная длина линии	1200м
Рабочая температура	-40°C до +50°C (кроме батарейки)
Размер(mm)	65x65x18

Режимы работы

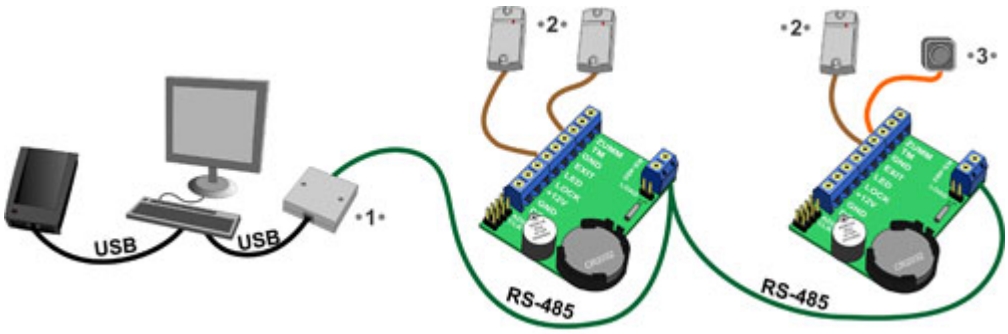
Режим ACCEPT- позволяет восстановить базу данных ключей. Активизировав режим ACCEPT контроллер разрешает доступ всем подносимым ключам и при этом заносит их ID в свою память. Тем самым, проработав несколько дней в режиме ACCEPT, контроллер формирует новую базу данных ключей.
Режим Блокировка - управление разрешением доступа. Блокирующий ключ, разрешает или запрещает открывание двери всем остальным прописанным ключам. Режим Блокировка удобен в случаях, где необходимо выполнить условие при котором нельзя входить в помещение если там нет ответственного лица (хозяин блокирующего ключа).

Применение

Предназначен для работы в составе сетевых и автономных СКУД. Простота в установке и обслуживании, идеально подойдет для управления электромагнитными и электромеханическими замками. Для управления турникетом необходимо использовать 2шт. Z5R Net. При подключении контроллера к компьютеру, рекомендуем использовать программное обеспечение .
Контроллер Z-5R Net позволяет подключить следующее оборудование:

- конвертор Z-397 Guard
- считыватель Matrix-II
- кнопка выхода

-сетевой контроллер
-электромагнитный/электромеханический замок
-внешний зуммер
-внешний светодиод
-датчик открытия двери



Сетевой контроллер Z-5R Net 8000

Страница на сайте IronLogic.ru

Спецификация

Напряжение питания	8-18V DC
Ток потребления	20 mA
Количество подключаемых считывателей	2
Тип (протокол) подключаемых считывателей	Wiegand26, Dallas Touch Memory
Выходы МДП транзистор	1
Ток коммутации	5A
Количество ключей/карт(max)	8168
Количество запоминаемых событий(max)	8192
Протокол связи с контроллерами	RS485
Скорость связи	19200 бод/57600бод
Максимальная длина линии	1200м
Рабочая температура	-40°C до +50°C (кроме батарейки)
Размер(mm)	65x65x18

Режимы работы

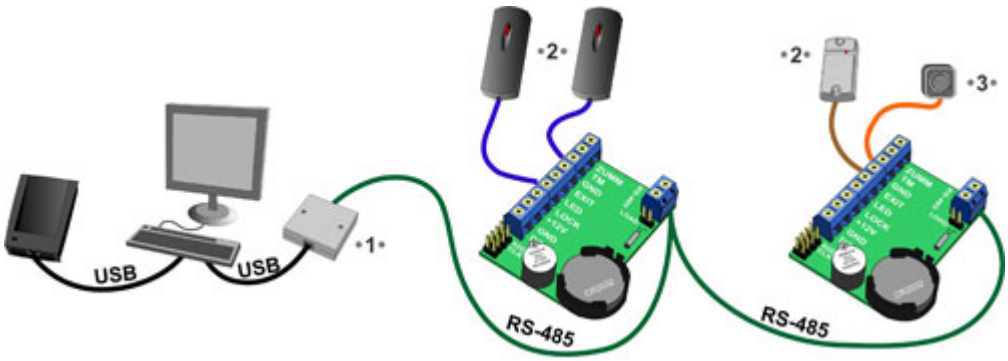
Режим ACCEPT- позволяет восстановить базу данных ключей. Активизировав режим ACCEPT контроллер разрешает доступ всем подносимым ключам и при этом заносит их ID в свою память. Тем самым, проработав несколько дней в режиме ACCEPT, контроллер формирует новую базу данных ключей.
Режим Блокировка - управление разрешением доступа. Блокирующий ключ, разрешает или запрещает открывание двери всем остальным прописанным ключам. Режим Блокировка удобен в случаях, где необходимо выполнить условие при котором нельзя входить в помещение если там нет ответственного лица (хозяин блокирующего ключа).

Применение

Предназначен для работы в составе сетевых и автономных СКУД. Простота в установке и обслуживании, идеально подойдет для управления электромагнитными и электромеханическими замками. Для управления турникетом необходимо использовать 2шт. Z5R Net 8000. При подключении контроллера к компьютеру, рекомендуем использовать программное обеспечение.
Контроллер Z-5R Net 8000 позволяет подключить следующее оборудование:

- конвертор Z-397 Guard
- считыватель Matrix-II
- кнопка выхода

-сетевой контроллер
-электромагнитный/электромеханический замок
-внешний зуммер
-внешний светодиод
-датчик открытия двери



Сетевой контроллер Matrix II Net

Страница на сайте IronLogic.ru

Спецификация

Напряжение питания	8-18V DC
Ток потребления	45 mA
Количество подключаемых считывателей	1
Тип (протокол) подключаемых считывателей	Dallas Touch Memory
Выходы МДП транзистор	1
Ток коммутации	5A
Количество ключей/карт(max)	2024
Количество запоминаемых событий(max)	2048
Протокол связи с контроллерами	RS485
Скорость связи	19200 бод/57600бод
Максимальная длина линии	1200м
Рабочая температура	-40°C до +50°C
Размер(mm)	85x44x18

Режимы работы

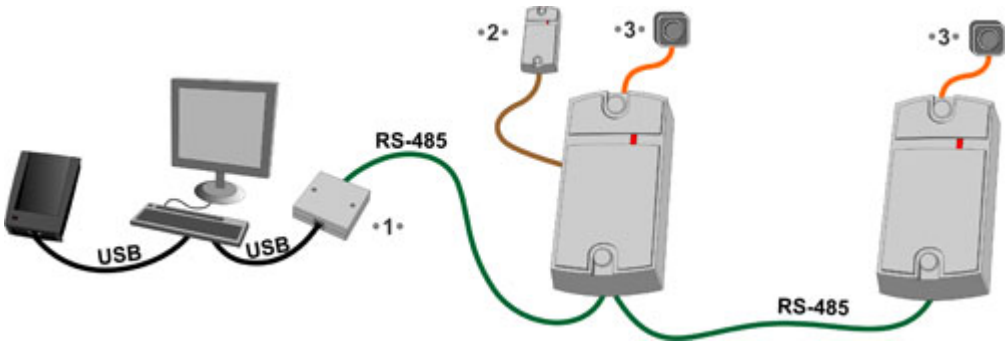
Режим ACCEPT- позволяет восстановить базу данных ключей. Активизировав режим ACCEPT контроллер разрешает доступ всем подносимым ключам и при этом заносит их ID в свою память. Тем самым, проработав несколько дней в режиме ACCEPT, контроллер формирует новую базу данных ключей.
Режим Блокировка - управление разрешением доступа. Блокирующий ключ, разрешает или запрещает открывание двери всем остальным прописанным ключам. Режим Блокировка удобен в случаях, где необходимо выполнить условие при котором нельзя входить в помещение если там нет ответственного лица (хозяин блокирующего ключа).
Режим ElectroControl - управление входной дверью и электропитанием. Подробное описание этого режима на странице Matrix II Net ElectroControl .

Применение

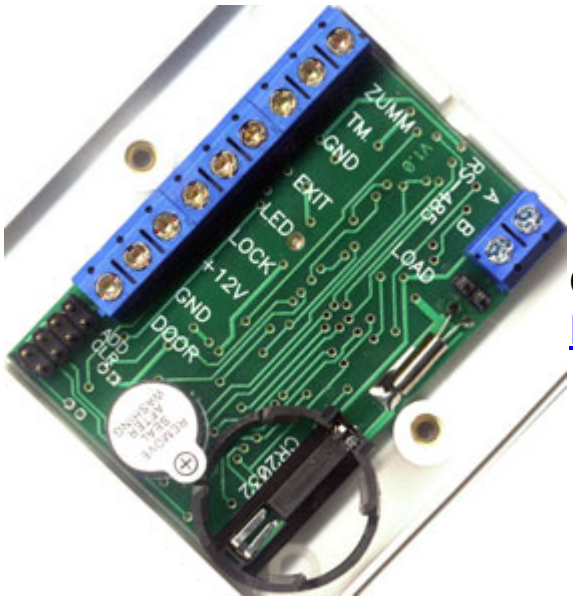
Предназначен для работы в составе сетевых и автономных СКУД. Простота в установке и обслуживании, идеально подойдет для управления электромагнитными и электромеханическими замками. Для управления турникетом необходимо использовать 2шт. Matrix II Net..При подключении контроллера к компьютеру, рекомендуем использовать программное обеспечение .
Контроллер Matrix II Net позволяет подключить следующее оборудование

- конвертор Z-397 Guard
- считыватель Matrix-II
- кнопка выхода

-сетевой контроллер
-электромагнитный/электромеханический замок
-внешний зуммер
-внешний светодиод
-датчик открытия двери



Сетевой контроллер Gate-2000



Страница на сайте
IronLogic.ru

Спецификация

Питание	12V DC, 200mA
Количество подключаемых считывателей	2
Выходы МДП транзистор	1 (электромагнитный/ электромеханический замок)
Тип (протокол) подключаемых считывателей	Dallas Touch Memory
Ток коммутации	до 5А
Количество ключей/карт(max)	2000
Количество запоминаемых событий(max)	2000
Количество расписаний	8
Сетевой режим	да
Автономный режим	да
Автономное программирование	да
Рабочая температура	-30°С до +50°С (кроме батарейки)
Размер(mm)	58x47x14

Применение

Предназначен для работы в составе сетевых и автономных СКУД. Удобен в использовании и при установке на дверь в офис , помещение.
Возможно заказать данный контроллер с модифицированным сетевым протоколом под Ваши требования.

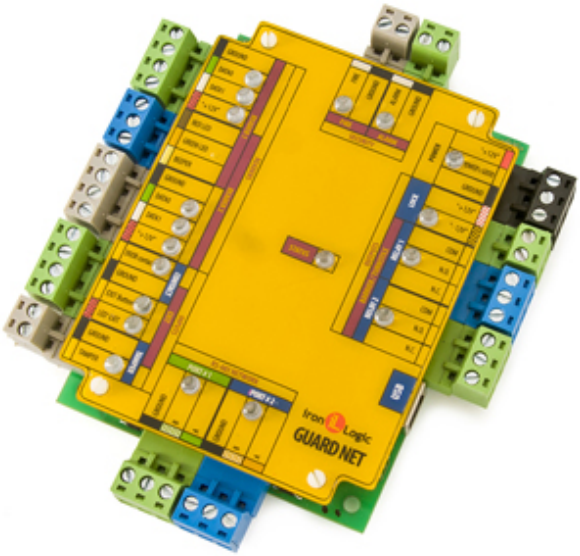
По поключению и работе в автономном режиме контролер GATE-2000 аналогичен контроллеру Z-5R.

Рекомендуется использование с:

- Считыватель "MATRIX-II"
- Конвертор USB-RS485 "Z-397"

При подключении считывателей MATRIX-II управление индикацией звука/света от контроллера происходит по проводу TM.

Сетевой контроллер GUARD Net



Страница на сайте
IronLogic.ru

Спецификация

Питание	9-16V DC 100mA
Количество подключаемых считывателей	2
Релейные выходы	2 (тип C)
Выходы МДП транзистор	1 (до 5А)
Тип (протокол) подключаемых считывателей	Wiegand, iButton (Dallas Touch Memory)
Количество ключей/карт(max)	два банка по 8168
Количество запоминаемых событий(max)	8192
Количество расписаний	8
Количество доп входов	2
Количество RS-485	2 (1 гальванически развязанный)
Дополнительный выход питания считывателей	да
Вход пожарной сигнализации	да
Сетевой режим	да
Автономный режим	да
Автономное программирование	да
Вес (г)	200

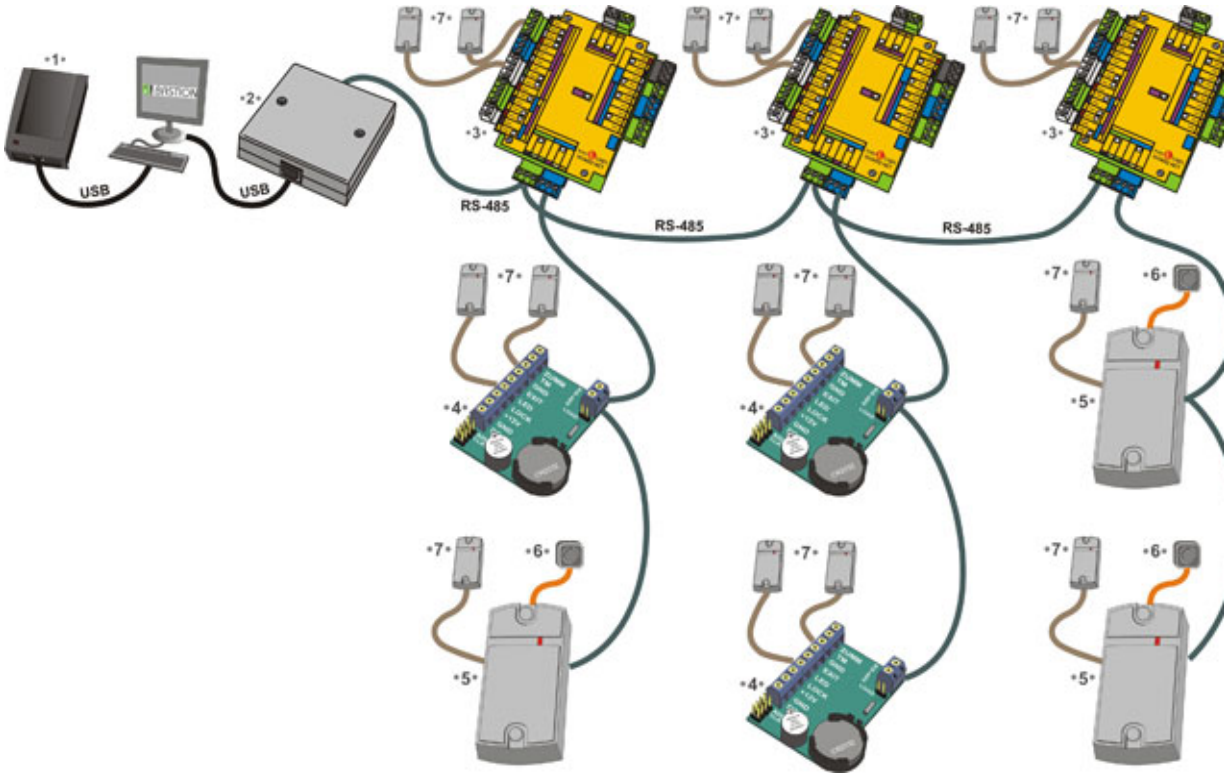
Рабочая температура	-40°C до +85°C
Размер(mm)	150x150x30

Режимы работы

Режим ACCEPT- позволяет восстановить базу данных ключей. Активизировав режим ACCEPT контроллер разрешает доступ всем подносимым ключам и при этом заносит их ID в свою память. Тем самым, проработав несколько дней в режиме ACCEPT, контроллер формирует новую базу данных ключей.
Режим Блокировка - управление разрешением доступа. Блокирующий ключ, разрешает или запрещает открывание двери всем остальным прописанным ключам. Режим Блокировка удобен в случаях, где необходимо выполнить условие при котором нельзя входить в помещение если там нет ответственного лица (хозяин блокирующего ключа).

Применение

Предназначен для работы в составе сетевых и автономных СКУД. Толковая и ясная документация, минимальное объединение проводником на клеммах обеспечивают простоту в установке и обслуживании.Наличие трех исполнительных элементов - два реле и полевой транзистор - идеально подойдет для управления электромагнитными и электромеханическими замками, турникетами и шлагбаумами. Также, в отдельных конфигурациях, обеспечивается подключение активатора, требующего подачи напряжения разной полярности для открывания и закрывания.
Наличие отдельного входа для пожарного шлейфа повышает пожарную безопасность, так как перевод его в активное состояние обеспечивает разблокировку прохода вне зависимости от наличия связи с компьютером.
USB порт позволяет конфигурировать контроллер путем простого редактирования текстового конфигурационного файла, также можно обновлять внутреннюю программу (firmware).
Наличие второго порта RS-485, реализовано в рамках контроллера, как гальванически развязанный повторитель линии RS-485. Таким образом, длинная сеть может построена, как несколько коротких гальванически развязанных участков. Это значительно повышает помехозащищенность.При этом программой все устройства воспринимаются подключенными к одной линии RS-485. Пример такой сети приведен ниже.
При подключении контроллера к компьютеру, рекомендуем использовать программное обеспечение.


•1•
•2•
•3•
•4•
•5•
•6•
•7•
USB
USB
RS-485
RS-485
RS-485

Z-9 EHT net

Электромеханический замок для двери

Основные характеристики:

- Сетевой и автономный режим работы
- Режим ОФИС / ОТЕЛЬ
- Работа в составе ПО Lock's Commander
- Подключение внешней кнопки открывания двери
- Программирование с ПК (RS-485 или адаптера RF1996)
- Не менее 25000 открываний от 4х батареек 1.5V
- Функция "Анти-паника"
- Механическое запирание ключом
- Противоотжимной механизм

Страница на сайте IronLogic.ru

Спецификация

Рабочая частота:	125 KHz
Чтение карт&брелков стандарта:	EM Marine, HID ProxCard II, Temic
Дальность чтения:	2-4 cm
Потребление тока:	30mкA (в состоянии ожидания)
Питание:	1.5V X 4 шт. стандартных батареек размера AA
Количество ключей/карт(max):	2024
Количество запоминаемых событий(max):	2048
Протокол связи с контроллерами	RS485
Скорость связи	19200 бод/57600бод
Звуковая/световая индикация:	сигнал зуммера, два светодиода
Установка длительности открывания замка:	от 0 до 220 сек.
Рабочая температура:	-30°C +60°C (кроме батареек)
Материал корпуса:	силумин
Цвет корпуса:	серебро, золото
Размер(mm)	295x65x26

Режимы работы

***Режим ОФИС**- замок могут открыть только карты занесенный в базу контроллера замка. Функции: свободный проход, блокировка, принудительное состояние, "не беспокоить". При использовании специализированного ПО, карты доступа так же будут починяться ограничениям по времени и дням недели. Так же:

- Интуитивно-понятные правила использования замка.
- Автоматическое запирание по выходу из комнаты.
- Управление разрешением доступа.
- Свободный выход изнутри и функция "Антипаника".
- Максимальная/полная автономность замка.
- Аудит открываний в памяти замка..

***Режим ОТЕЛЬ** - учитывает все особенности в безопасности и жизнедеятельности отелей/гостиниц//общежитий/студенческих кампусов и т.п. Позволяет создавать «гостевые карты», «карты персонала» и «карты специального назначения», с определенными правами доступа и защитой от копирования или утери карты. Так же:

- Интуитивно-понятные правила использования замка.
- Работа по принципу "новый гость — новый ключ".
- Автоматическое запирание по выходу из номера.
- Реализация режима "Не беспокоить".
- Свободный выход изнутри и функция "Антипаника".
- Максимальная/полная автономность замка.
- Возможность поселения гостей "без компьютера".
- Аудит открываний в памяти замка.

Применение

Замок Z-9 EHT net – это бесконтактный считыватель,сетевой контроллер и полноценная запирающая система. Для открывания двери снаружи необходимо поднести карту, а с внутренней стороны достаточно только нажать ручку двери. Замок может работать как автономно, с питанием от 4 батареек размера AA, так и в составе сети RS-485, с питанием 8-12V DC. Привлекательный внешний вид, надежная конструкция, простота в программировании и установке, “продвинутый” функционал - это все позволит использовать замок в общежитиях,студенческих кампусах и лагерях, офисных и административных помещениях любого назначения.
При совместном использовании ПО “ОТЕЛЬ” и адаптера RF1996, замки Z-9 EHT net организуют работу гостиниц, отелей, санаториев и других объектах, где необходимо вести выписку и учет гостевых карт , с защитой от утери или копирования карт злоумышленниками. Хранение событий в памяти Z-9 EHT net, позволит установить высокий уровень безопасности в номерах.

Eurolock EHT net

Электромеханическая накладка на дверной замок стандарта DIN

Основные характеристики:

- Низкая цена
- Самый простой в установке замок: монтаж 10 минут
- Установка на двери толщиной 30-70 мм
- Не требует прокладки электрической проводки
- Совместим с врезными замками стандарта DIN
- Функция "Анти-паника”
- Режим офис / отель
- Режим автономный / сетевой
- WiFi расширяемый*

Страница на сайте IronLogic.ru

Спецификация

Рабочая частота:	125 KHz
Чтение карт&брелков стандарта:	EM Marine, HID ProxCard II, Temic
Дальность чтения:	2-4 cm
Потребление тока:	30mкA (в состоянии ожидания)
Питание:	1.5V X 3 шт. стандартных батареек размера AAA
Количество ключей/карт(max):	2024
Количество запоминаемых событий(max):	2048
Линии связи:	Micro USB, RS485, WiFi* опционально
Выходной интерфейс считывателя:	Dallas Touch Memory, Wiegand 26*
Звуковая/световая индикация:	сигнал зуммера, 3х цветный светодиод
Установка длительности открывания замка:	от 0 до 220 сек.
Рабочая температура:	-30°C +60°C (кроме батареек)
Материал корпуса:	силумин, сталь
Цвет корпуса:	матовая нержавеющая сталь
Размер(mm)	280x50x20

Режимы работы

***Режим ОФИС**- замок могут открыть только карты занесенный в базу контроллера замка. Функции: свободный проход, блокировка, принудительное состояние, "не беспокоить". При использовании специализированного ПО, карты доступа так же будут починяться ограничениям по времени и дням недели. Так же:

- Интуитивно-понятные правила использования замка.
- Автоматическое запирание по выходу из комнаты.
- Управление разрешением доступа.
- Свободный выход изнутри и функция "Антипаника".
- Максимальная/полная автономность замка.
- Аудит открываний в памяти замка..

***Режим ОТЕЛЬ** - учитывает все особенности в безопасности и жизнедеятельности отелей/гостиниц//общежитий/студенческих кампусов и т.п. Позволяет создавать «гостевые карты», «карты персонала» и «карты специального назначения», с определенными правами доступа и защитой от копирования или утери карты. Так же:

- Интуитивно-понятные правила использования замка.
- Работа по принципу "новый гость — новый ключ".
- Автоматическое запирание по выходу из номера.
- Реализация режима "Не беспокоить".
- Свободный выход изнутри и функция "Антипаника".
- Максимальная/полная автономность замка.
- Возможность поселения гостей "без компьютера".
- Аудит открываний в памяти замка.

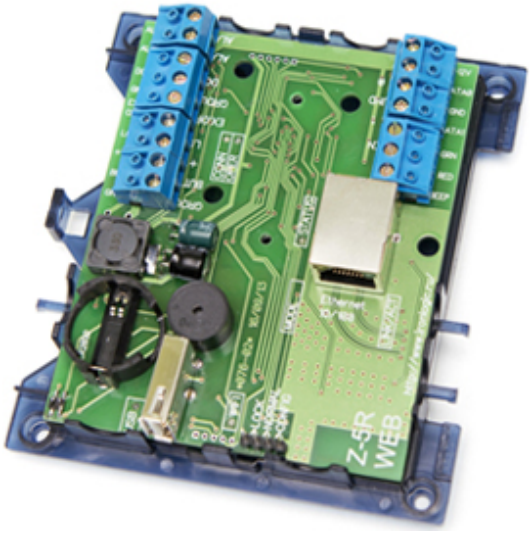
Доступные модификации:

Eurolock EHT net, серебро, dd = 0 mm
Eurolock EHT net, серебро, dd = 72 mm
Eurolock EHT net, серебро, dd = 85 mm
Eurolock EHT net, серебро, dd = 92 mm

Применение

Eurolock EHT net – это бесконтактный считыватель и контроллер с питанием от 3х батареек 1,5V размера AAA, срок службы батареи более 50 000 циклов (max. 4 года). Для открывания двери снаружи необходимо поднести карту, а изнутри достаточно только нажать ручку. Для монтажа, достаточно демонтировать старые дверные ручки и на их место установить Eurolock EHT net, сделав всего два отверстия в дверном полотне. Уникальность Eurolock EHT net в возможности оборудовать практически любую существующую дверь, потратив на монтаж и подключение всего 10минут. Система позволяет пользователям и администраторам создавать и обновлять гибкую систему контроля доступа в помещение в составе сети RS-485 или WiFi * опционально с использованием специализированного ПО, так и автономно без участия компьютера.

Сетевой контроллер Z-5R Web



Страница на сайте IronLogic.ru

Спецификация

Сетевой режим	да
Автономный режим	да
Автономное программирование	да
Количество ключей	два банка по 8168
Количество запоминаемых событий	8192
Выходы МДП транзистор	2 (до 5А/3А)
Количество расписаний	8
Количество доп входов	2
Количество сигнальных выходов	1 (оптоизолированный)
Вход пожарной сигнализации	да
Количество считывателей	2
Типы (протоколы) подключаемых считывателей	Wiegand, iButton (Dallas Touch Memory)
Количество выходов для управления индикацией считывателей	3 (red, green, sound)
Дополнительный выход питания	да (защищен от КЗ)

считывателей	
Выходы МДП транзистор	2 (до 5A/3A)
Питание	9-27V DC 300mA (при 12V)
Размер (мм)	116x104x37
Вес (г)	150
Рабочая температура	-20°C до +85°C

Применение

Предназначен для работы в составе сетевых и автономных СКУД. Контроллер со сменной функциональностью, которая определяется на этапе конфигурирования. Контроллер способен меняться в зависимости от вашей задачи. Переход в режим точки доступа для конфигурирования позволяет настраивать контроллер со смартфона. Если раньше для каждого решения вам, необходим был отдельный вид контроллера, то теперь достаточно приобрести Z5R Web и настроить под ваши нужды.

Толковая и ясная документация, минимальное объединение проводников на клеммах обеспечивают простоту в установке и обслуживании.Наличие двух исполнительных элементов - полевых транзисторов - идеально подойдет для управления электромагнитными и электромеханическими замками, турникетами и шлагбаумами. Также, в отдельных конфигурациях, обеспечивается подключение активатора, требующего подачи напряжения разной полярности для открывания и закрывания.
Наличие отдельного входа для пожарного шлейфа повышает пожарную безопасность, так как перевод его в активное состояние обеспечивает разблокировку прохода вне зависимости от наличия связи с компьютером.

Наличие возможности подключаться к локальной сети, как с помощью Ethernet, так и с помощью WiFi, позволяет экономить на монтаже сети связи.

Режимы работы:

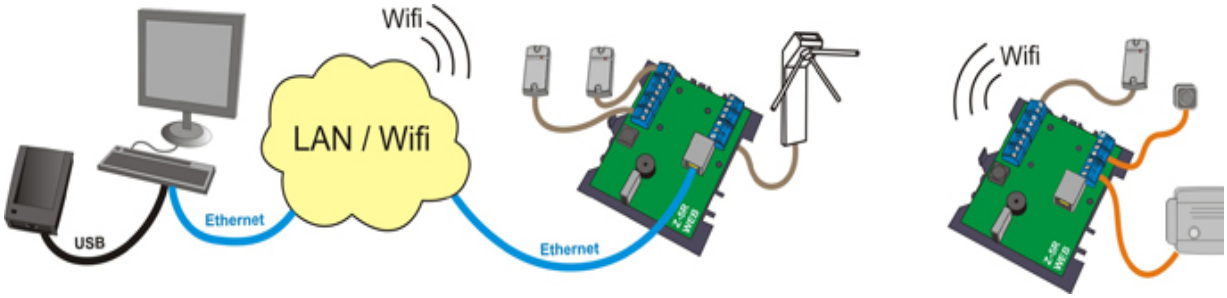
По типу точек доступа

1. Одна дверь
2. Шлюз (на каждую дверь по контроллеру)
3. Турникет

4. Активаторный замок (ворота)
5. Электроконтроль

По режиму доступа:

1. Обычный
2. Блокировано
3. Свободный проход
4. Ожидание



Сетевой контроллер Matrix III Net

Страница на сайте IronLogic.ru

Спецификация

Напряжение питания	9-16V DC
Интерфейсы	iButton (TM), Wiegand26, RS-485
Выходы МДП транзистор	1
Ток коммутации	5A
Рабочая частота	13,56 МГц
Чтение/запись идентификаторов	Mifare 1K, Mifare 4K
Дальность чтения	до 6 см
Звуковая/световая индикация	сигнал зуммера, двухцветный светодиод Исполнение: наружное
Максимальная длина линии	- Dallas Touch Memory: не более 15 м - Wiegand: не более 100 м - RS-485: не более 1200 м
Рабочая температура	-30°C +40°C
Размер(mm)	115x75x22

Matrix-II Wi-Fi

IP-контроллер СКУД со встроенным считывателем EM-Marine

Основные характеристики:

- Встроенный считыватель
- Интерфейс связи: Wi-Fi
- Влаго и пылезащищенный корпус
- Быстрый монтаж и подключение
- Автономный и сетевой режим работы
- Защита от неправильного включения
- Выбор типа замка
- Режим Обычный / TRIGGER / Блокировка / Свободный проход

Страница на сайте IronLogic.ru

Спецификация

Рабочая частота:	125 KHz
Работа с идентификаторами:	EM-Marine
Дальность чтения:	3-5 см
Количество ключей (max):	2024
Количество запоминаемых событий (max):	2048
Количество подключаемых считывателей:	2
Протокол подключаемых считывателей:	Dallas Touch Memory, Wiegand 26
Наличие перемычки для выбора типа замка:	есть
Тип исполнительного устройства:	электромеханический/электромагнитный замок
Световая и звуковая индикация:	есть
Установка длительности открывания замка:	от 0 до 25,5 с
Выход:	МДП транзистор 1шт.
Интерфейс связи:	Wi-Fi

Напряжение питания:	8-24V DC
Ток потребления:	до 100 мА
Ток коммутации:	5A
Защита от неправильного включения:	есть
Рабочая температура:	-30°C +40°C
Размер(mm)	85x44x18

Режимы работы

Обычный режим - обеспечивает проход простым ключам.

Режим TRIGGER - управление работой замка: вкл./выкл. Одно касание ключа - замок закрыт; второе касание ключа - замок открыт. Режим TRIGGER удобен в случаях, где необходимо открывать или блокировать дверь на определенный период (рабочий день, перерыв и т.д.).

Режим Блокировка - открыт проход по блокирующим ключам, простым ключам проход закрыт.

Режим **Свободный проход** - замок всегда обесточен.

Специальные режимы:
Режим **Экстренной эвакуации**
Режим **Блокировки замка по сигналу охранной сигнализации**
Режим **Охраны**

Применение

Предназначен для работы в составе сетевыхСКУД. Простота в установке и обслуживании. Работает с электромагнитными и электромеханическими замками. Для управления турникетом можно использовать один Matrix-II Wi-Fi. При подключении контроллера к компьютеру рекомендуется использовать программное обеспечение (выбор ПО).
Контроллер Matrix-II Wi-Fi работает совместно со следующим оборудованием:

1. считыватели
2. кнопка выхода
3. электромагнитный/электромеханический замок
4. внешний зуммер
5. внешний светодиод
6. датчик открытия двери

Уведомления ZGuard API

Уведомления ZPort API. Для настройки уведомлений подключения/отключения конвертера можно использовать функцию ZG_SetNotification, для извлечения сообщений - ZG_GetNextMessage.

Уведомления конвертера

Уведомления об изменении списка карт в поле считывателя (настраиваются с помощью функции ZG_Cvt_SetNotification)

Сообщение	Параметр	Описание
ZG_N_CVT_CTR_INSERT	PZG_FIND_CTR_INFO	Контроллер подключен. В параметре - указатель на информацию о контроллере.
ZG_N_CVT_CTR_REMOVE	PZG_FIND_CTR_INFO	Контроллер отключен. В параметре - указатель на информацию о контроллере.
ZG_N_CVT_CTR_CHANGE	PZG_N_CTR_CHANGE_INFO	Изменены параметры контроллера. В параметре - указатель на информацию о изменениях.
ZG_N_CVT_ERROR	PHRESULT	Произошла ошибка в потоке (thread). В параметре - указатель на код ошибки.
ZG_CVTN_CONNECTION_CHANGE	-	Изменилось состояние подключения. Получить текущее состояние можно с помощью ZG_Cvt_GetConnectionStatus.

Уведомления контроллера

Уведомления об изменении списка карт в поле считывателя (настраиваются с помощью функции ZG_Ctr_SetNotification)

Сообщение	Параметр	Описание
ZG_N_CTR_NEW_EVENT	PZG_N_NEW_EVENT_INFO	Произошло новое событие. В параметре - указатель на структуру с дополнительной информацией.
ZG_N_CTR_CLOCK	PINT64	Произошла рассинхронизации часов контроллера с часами ПК. В параметре - величина рассинхронизации в секундах.
ZG_N_CTR_KEY_TOP	PZG_N_KEY_TOP_INFO	Изменилась верхняя граница ключей. В параметре - указатель на структуру с дополнительной информацией.
ZG_N_CTR_ADDR_CHANGE	PBYTE	Изменен сетевой адрес контроллера. В параметре - новый адрес контроллера.

Сообщение	Параметр	Описание
ZG_N_CTR_ERROR	PHRESULT	Произошла ошибка в потоке (thread). Параметр является ссылкой на код ошибки.

Примеры использования SDK

Исходный код примеров находится в **<папка установки SDK>\<язык программирования>\Examples**.

Название	Кратное описание
EnumSerialPorts	Поиск портов конвертеров, уведомление о подключении/отключении конвертеров.
EnumConverters	Поиск портов конвертеров с отображеним подробной информации о конвертерах.
FindControllers	Поиск контроллеров для заданного конвертера, уведомление о подключении/отключении контроллеров.
CvtLicense	Показ параметров лицензии конвертера, установка новой лицензии (загружает из ini-файла).
CtrClock	Показ часов контроллера, синхронизация часов контроллера с часами ПК, уведомление о рассинхронизации часов.
CtrLockTimes	Показ значений времен для замков контроллера, установка новых значений, дистанционное открытие замка, установка заводских настроек.
CtrSchedule	Показ расписаний контроллера, установка нового расписания, установка заводских настроек.
CtrKeys	Показ списка ключей, поиск ключа по номеру, добавление ключа, изменение параметров ключа, удаление ключа, удаление всех ключей, уведомление об изменении верхней границы ключей.
CtrUpdateKeys	Обновление списка ключей (установка нового списка, выгрузка старого списка)

Название	Кратное описание
CtrEvents	Показ новых событий, показ всех событий, сброс указателей событий в 0 (заводские настройки), уведомление о новых событиях.
UpdateFirmware	Прошивка конвертера по имени COM-порта, прошивка контроллера по с/н.
Работа с контроллером Matrix II Net с версией прошивки 3.0	
M2nElectoControl	Управление элетропитанием (функция "ElectroControl").

Вопросы и ответы

1. **Почему при компиляции программы возникает ошибка линковки библиотеки ZPort.dll "error LNK2019: unresolved external symbol..."?** В заголовочном файле "ZPort.*" в константе ZpDllName указано неправильное имя библиотеки. Если "ZGuard.dll" взята из корневой папки установки, то ZpDllName должно быть равно "ZGuard.dll", а если из папки "Split", то ZpDllName = "ZPort.dll".
   Если в одном проекте используются SDK Readers & SDK Guard вместе, то библиотеки лучше брать из папки "Split".
   DLL лучше класть рядом с EXE-файлом Вашей программы.
   Для C++ вместе с DLL еще нужно копировать соответствующий LIB файл.

2. **Что такое адрес контроллера?**
   Адрес контроллера - уникальный адрес контроллера в сети (на новых контроллерах устанавлен адрес =1). Адрес можно изменять с помощью SDK (`ZG_Cvt_SetCtrAddr`, `ZG_Cvt_SetCtrAddrBySn` и `ZG_Ctr_SetNewAddr`), при этом конвертеры Z-297 Guard [Advanced] и Z-297 IP автоматически исправляют конфликтующие адреса, т.е. одинаковые и адреса меньше 2, конвертер Z-397 Web не позволяет командой изменять адрес в режиме Advanced.
   Все команды контроллеру посылаются по его адресу в сети.

3. **Какой функцией пользоваться для переназначения адреса -** `ZG_Ctr_SetNewAddr` **либо** `ZG_Ctr_AssignAddr`**?** Нужно использовать функцию `ZG_Ctr_SetNewAddr`. Функция `ZG_Ctr_AssignAddr` меняет только переменную "сетевой адрес" внутри объекта "контроллер" в SDK, это нужно на случай если сетевой адрес был изменен низкоуровневыми функциями.

4. **Режим блокирующих карт программно включается?**
   Нет, блокирующий режим - это режим автономной работы и программно не управляется.

5. **Зачем нужны лицензии?**
   Для защиты нашего платного ПО.
   Лицензии задают ограничения для работы с конвертером:
   - макс. количество контроллеров в сети, с которыми можно работать одновременно
   - макс. количество ключей
   - дата окончания действия лицензии
   У всех лицензий есть уникальный номер, который соответствует ПО (у GuardCommander и у SDK Guard (по-умолчанию) - №5).
   Лицензии могут быть универсальными (подходящими для любого конвертера) и специальными (для определенного конвертера, привязывается к его с/н).
   В конвертер могут быть установлены до 16 лицензий одновременно для разного ПО.

   Лицензия для SDK выдается бесплатно. Если не устраивают параметры лицензии, входящей в комплект SDK, то можно запросить её у нас.

6. **Сколько циклов перезаписи выдерживают используемая флеш память для карт/событий?**
   1 000 000

7. **Что делать если контроллеры не находятся или с ними пропадает связь?**
   Если контроллеров много, то нужно у них всех объединить "землю" (минус питания).

8. **В каких режимах (отель/офис) Eurolock EHT net работает с SDK Guard?**
   Офис. Кроме того конвертер должен быть в режиме Normal

(работа с Eurolock EHT net в Advanced режиме пока не реализована). Чтобы Z-397 Web переключился в режим Normal нужно при вызове функции `ZG_Cvt_Open` указать `ZG_CVT_OPEN_PARAMS.CvtType = ZG_CVT_Z397`.

9. **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.

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

    1. `ZG_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`.

11. **Обязательно ли подключать последовательно замки Z-9 Eht Net или можно деревом все подключить?**
    Да, нужно подключать последовательно, деревом - нельзя.

Утилиты

В составе SDK:

1. ZRetr.exe - эмулирует Ip-конвертеры с помощью usb-конвертеров (поддерживаются режимы SERVER и CLIENT) (находится в папке "Split");
2. ZLog.exe - показывает отладочные сообщения библиотеки "ZGuard.dll" из папки "Log";
3. ZUpdater.exe - проверяет обновления Sdk Guard и Sdk Readers (находится в папке "Updater").

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.

Структура ZG_CVT_OPEN_PARAMS

Параметры открытия конвертера, используемые функциями: ZG_Cvt_Open и ZG_UpdateCvtFirmware.

C++

```
typedef struct _ZG_CVT_OPEN_PARAMS
{
        ZP_PORT_TYPE nType;
        LPCWSTR pszName;
        HANDLE hPort;
        ZG_CVT_TYPE nCvtType;
        ZG_CVT_SPEED nSpeed;
        PZP_WAIT_SETTINGS pWait;
        BYTE nStopBits;
        INT nLicN;
        LPCSTR pActCode;
        int nSn;
        UINT nFlags;
} *PZG_CVT_OPEN_PARAMS;
```

Delphi

```
  TZG_CVT_OPEN_PARAMS = packed record
    nType             : TZP_PORT_TYPE;
    pszName           : PWideChar;
    hPort             : THandle;
    nCvtType          : TZG_CVT_TYPE
    nSpeed            : TZG_CVT_SPEED;
    pWait             : PZP_WAIT_SETTINGS;
    nStopBits         : Byte;
    nLicN             : Integer;
    pActCode          : PAnsiChar;
    nSn               : Integer;
    nFlags            : Cardinal;
```

```
  end;
  PZG_CVT_OPEN_PARAMS = ^TZG_CVT_OPEN_PARAMS;
```

Параметры

Type
: Тип порта.

Name
: Имя порта. Если =**null**, то используется `hPort`.

hPort
: Дескриптор порта, полученный функцией `ZP_Open`.

CvtType
: Тип конвертера. По умолчанию равно `ZG_CVT_UNDEF`.

Speed
: Скорость конвертера. Используется только для Z-397 и Z-397 Guard в режиме Normal.

Wait
: Параметры ожидания. Может быть =**null**.

StopBits
: Стоповые биты.

Константа	Значение	Описание
`ONESTOPBIT`	0	1 стоповый бит
`ONE5STOPBITS`	1	1,5 стоповых бита
`TWOSTOPBITS`	2	2 стоповых бита

LicN
: Номер лицензии. Если =0, то используется `ZG_DEF_CVT_LICN` (=5).

ActCode
: Код активации для подключения к конвертеру в режиме "Proxy".

Sn
: С/н конвертера для режима "Proxy".

Flags
: Флаги:

Константа	Значение	Описание
ZG_OF_NOCHECKLIC	1	Не проверятьи не обновлять лицензию при подключении
ZG_OF_NOSCANCTRS	2	Не сканировать контроллеры при подключении

ZG_Cvt_SetCtrAddr

Устанавливает новый сетевой адрес контроллера.

C++

```
HRESULT ZG_Cvt_SetCtrAddr(
        HANDLE hHandle,
        BYTE nOldAddr,
        BYTE nNewAddr,
        ZG_CTR_TYPE nModel=ZG_CTR_UNDEF
        );
```

Delphi

```
function ZG_Cvt_SetCtrAddr(
    AHandle: THandle;
    AOldAddr: Byte;
    ANewAddr: Byte;
    AModel: TZG_CTR_TYPE=ZG_CTR_UNDEF
    ): HRESULT;
```

Параметры

Handle
: [in] Дескриптор контроллера.

OldAddr
: [in] Текущий адрес контроллера.

NewAddr
: [in] Новый адрес (0-254).

Model
: [in] Модель контроллера. Для **Eurolock EHT net** нужно указывать обязательно.

Возвращаемое значение

Если функция выполнена успешно, то возвращает `S_OK`.

Если функция выполнена с ошибками, то возвращает код ошибки.

Примечание

Конвертер в режиме Advanced автоматически назначает сетевые адреса контроллерам.

Структура ZP_SEARCH_PARAMS

Параметры функции поиска устройств ZG_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 `ZG_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.