Утверждаю

Главный конструктор

ООО Фирма «Информтест»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Стороженко Ю.Л.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

ДРАЙВЕР МЕЗОНИНА ММКО1

Руководство системного программиста

ФТКС.75080-01 32 01

Листов 52

Разработал

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мисюк А.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Инв. № 15306 12.11.19

Проверил

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Волошановский В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

2020

АННОТАЦИЯ

Данный документ предназначен для использования в качестве руководства системного программиста. Он описывает каждую функцию из определенных в драйвере, а также коды ошибок.

СОДЕРЖАНИЕ

[1. Общие сведения 6](#_Toc51915740)

[2. Примеры 7](#_Toc51915741)

[2.1. Самоконтроль и тестирование 7](#_Toc51915742)

[2.2. Мониторинг сообщений 8](#_Toc51915743)

[2.3. Фильтрация сообщений 9](#_Toc51915744)

[2.4. Отправка отдельных сообщений 10](#_Toc51915745)

[2.5. Создание и отправка расписания сообщений 11](#_Toc51915746)

[2.6. Циклическая отправка расписания сообщений 12](#_Toc51915747)

[2.7. Мониторинг сообщений в режиме контроллера шины 13](#_Toc51915748)

[2.8. Создание ответных сегментов сообщений 14](#_Toc51915749)

[2.9. Создание ответных сегментов сообщений в ручном режиме 15](#_Toc51915750)

[2.10. Создание ответных сегментов в режиме очереди данных 16](#_Toc51915751)

[2.11. Циклический возврат данных 18](#_Toc51915752)

[2.12. Мониторинг сообщений в режиме оконечного устройства 19](#_Toc51915753)

[3. Функции библиотеки 20](#_Toc51915754)

[3.1. Константы и перечисления 20](#_Toc51915755)

[3.1.1. Константы 20](#_Toc51915756)

[3.1.2. unmmko1\_bus 21](#_Toc51915757)

[3.1.3. unmmko1\_message\_activities 21](#_Toc51915758)

[3.1.4. unmmko1\_message\_errors 22](#_Toc51915759)

[3.1.5. unmmko1\_error\_bits 22](#_Toc51915760)

[3.1.6. unmmko1\_address 23](#_Toc51915761)

[3.1.7. unmmko1\_subaddress 23](#_Toc51915762)

[3.1.8. unmmko1\_bc\_options 24](#_Toc51915763)

[3.1.9. unmmko1\_rt\_options 24](#_Toc51915764)

[3.1.10. unmmko1\_rt\_subaddress\_options 24](#_Toc51915765)

[3.1.11. unmmko1\_mon\_options 25](#_Toc51915766)

[3.2. Определения типов 26](#_Toc51915767)

[3.2.1. unmmko1\_command 26](#_Toc51915768)

[3.2.2. unmmko1\_response 26](#_Toc51915769)

[3.2.3. unmmko1\_message 27](#_Toc51915770)

[3.3. Функции управления мезонином 27](#_Toc51915771)

[3.3.1. unmmko1\_init 27](#_Toc51915772)

[3.3.2. unmmko1\_connect 28](#_Toc51915773)

[3.3.3. unmmko1\_self\_test 28](#_Toc51915774)

[3.3.4. unmmko1\_test\_exchange 29](#_Toc51915775)

[3.3.5. unmmko1\_test\_memory 29](#_Toc51915776)

[3.3.6. unmmko1\_reset 29](#_Toc51915777)

[3.3.7. unmmko1\_install\_interrupt\_handler 30](#_Toc51915778)

[3.3.8. unmmko1\_error\_query 30](#_Toc51915779)

[3.3.9. unmmko1\_revision\_query 31](#_Toc51915780)

[3.3.10. unmmko1\_error\_message 31](#_Toc51915781)

[3.3.11. unmmko1\_close 31](#_Toc51915782)

[3.4. Функции управления контроллером шины 32](#_Toc51915783)

[3.4.1. unmmko1\_bc\_rt 32](#_Toc51915784)

[3.4.2. unmmko1\_rt\_rt 32](#_Toc51915785)

[3.4.3. unmmko1\_f1 33](#_Toc51915786)

[3.4.4. unmmko1\_f2 33](#_Toc51915787)

[3.4.5. unmmko1\_f3 34](#_Toc51915788)

[3.4.6. unmmko1\_f4 34](#_Toc51915789)

[3.4.7. unmmko1\_f5 35](#_Toc51915790)

[3.4.8. unmmko1\_f6 35](#_Toc51915791)

[3.4.9. unmmko1\_f7 35](#_Toc51915792)

[3.4.10. unmmko1\_f8 36](#_Toc51915793)

[3.4.11. unmmko1\_f9 36](#_Toc51915794)

[3.4.12. unmmko1\_f10 37](#_Toc51915795)

[3.4.13. unmmko1\_pack\_cw 37](#_Toc51915796)

[3.4.14. unmmko1\_unpack\_cw 38](#_Toc51915797)

[3.4.15. unmmko1\_pack\_sw 38](#_Toc51915798)

[3.4.16. unmmko1\_unpack\_sw 38](#_Toc51915799)

[3.4.17. unmmko1\_bc\_configure 39](#_Toc51915800)

[3.4.18. unmmko1\_bc\_reset 39](#_Toc51915801)

[3.4.19. unmmko1\_bc\_schedule\_command 39](#_Toc51915802)

[3.4.20. unmmko1\_bc\_set\_schedule\_repeat\_count 40](#_Toc51915803)

[3.4.21. unmmko1\_bc\_start 40](#_Toc51915804)

[3.4.22. unmmko1\_bc\_status 40](#_Toc51915805)

[3.4.23. unmmko1\_bc\_stop 41](#_Toc51915806)

[3.4.24. unmmko1\_bc\_transmit\_command 41](#_Toc51915807)

[3.5. Функции управления оконечным устройством 42](#_Toc51915808)

[3.5.1. unmmko1\_rt\_configure 42](#_Toc51915809)

[3.5.2. unmmko1\_rt\_reset 42](#_Toc51915810)

[3.5.3. unmmko1\_rt\_set\_status\_word 43](#_Toc51915811)

[3.5.4. unmmko1\_rt\_set\_subaddress\_options 43](#_Toc51915812)

[3.5.5. unmmko1\_rt\_set\_subaddress\_data 44](#_Toc51915813)

[3.5.6. unmmko1\_rt\_set\_command\_data 45](#_Toc51915814)

[3.5.7. unmmko1\_rt\_start 45](#_Toc51915815)

[3.5.8. unmmko1\_rt\_status 46](#_Toc51915816)

[3.5.9. unmmko1\_rt\_stop 46](#_Toc51915817)

[3.6. Функции управления монитором шины 46](#_Toc51915818)

[3.6.1. unmmko1\_mon\_configure 46](#_Toc51915819)

[3.6.2. unmmko1\_mon\_reset 47](#_Toc51915820)

[3.6.3. unmmko1\_mon\_set\_timeout 47](#_Toc51915821)

[3.6.4. unmmko1\_mon\_filter\_address 47](#_Toc51915822)

[3.6.5. unmmko1\_mon\_filter\_subaddress 48](#_Toc51915823)

[3.6.6. unmmko1\_mon\_start 48](#_Toc51915824)

[3.6.7. unmmko1\_mon\_status 49](#_Toc51915825)

[3.6.8. unmmko1\_mon\_stop 49](#_Toc51915826)

[3.6.9. unmmko1\_mon\_messages\_count 49](#_Toc51915827)

[3.6.10. unmmko1\_mon\_messages\_read 50](#_Toc51915828)

[3.7. Коды ошибок и предупреждений 51](#_Toc51915829)

Введение

Данный драйвер предназначен для управления работой мезонина ММКО1. Он содержит набор функций для открытия и закрытия сеанса связи с инструментом, конфигурации и исполнения его режимов, а также ряд служебных функций. Функции драйвера разделены на классы, обеспечивающие как законченное выполнение режимов инструмента, так и управление инструментом на более низком уровне, включающее установку отдельных параметров, запуск процессов и т. п.

В документе перечислены все функции в порядке следования их в классах с описанием назначения функции, ее синтаксиса, а также с описанием каждого из параметров.

Драйвер написан на языке C и скомпонован как динамическая библиотека, предназначенная для использования в среде Windows или Linux.

В комплект файлов драйвера входит файл исходного текста C, файл заголовков H, файл динамической библиотеки DLL/SO, предназначенные для использования с компиляторами LabWindows/CVI, Borland C++, Visual C, GCC.

В комплект файлов драйвера для ОС Windows дополнительно входят файлы панели функций FP и статическая библиотека LIB для связи с динамической библиотекой.

Краткая информация об инструменте:

Мезонинный модуль ММКО1 предназначен для обмена по мультиплексному каналу передачи данных ГОСТ Р 52070-2003. Модуль имеет два канала: основной и резервный. Поддерживает работу в режиме контроллера шины, оконечного устройства и монитора шины.

# Общие сведения

Для успешного использования данного драйвера должны быть соблюдены следующие условия:

* мезонин должен быть установлен в носитель мезонинов;
* носитель мезонинов должен быть установлен в крейт VXI;
* для исполнения драйвера должен использоваться либо встроенный компьютер, либо компьютер, соединенный с крейтом VXI через интерфейс с поддержкой библиотеки VISA;
* для компиляции программы, использующей данный инструмент, должен быть использован один из компиляторов LabWindows/CVI, Borland C++, Visual C или GCC с включением файла заголовков драйвера инструмента и соответствующей статической библиотеки в состав проекта компилятора;
* в ОС Windows на компьютере должна быть установлена динамическая библиотека LabWindows/CVI RTE;
* на компьютере должно быть установлено программное обеспечение VISA;
* носитель мезонинов должен быть сконфигурирован менеджером ресурсов VXI.

# Примеры

В разделе «Примеры» приведены листинги прикладных программ на языке C для основных сценариев использования функций библиотеки. Исходные коды примеров распространяются через установочный комплект драйвера мезонина.

Во всех примерах используется общий модуль – файл common.h. В нём содержатся функции поиска и идентификации мезонина, реализована обработка и печать ошибок.

Отдельно стоит отметить функцию search\_unmmko1, которая отвечает за поиск мезонина и возвращает адрес ресурса и позицию на нём мезонина, а также два макроса – unmbasecheck и unmkocheck – для проверки возвращаемых функциями кодов ошибки или успешного выполнения.

Все примеры собраны в виде консольных приложений, без графического интерфейса пользователя.

## Самоконтроль и тестирование

Пример показывает, как установить связь с носителем мезонинов (unmbase), мезонином (unmmko1) и выполнить функции самоконтроля, тестирования памяти и обмена.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViChar software\_version[256];

ViChar hardware\_version[256];

ViInt16 result\_code;

ViChar message[256];

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Информация о мезонине

unmkocheck(unmmko1\_revision\_query(session, software\_version, hardware\_version));

printf("Software version: %s\n", software\_version);

printf("Hardware version: %s\n", hardware\_version);

// Проводим самоконтроль мезонина

unmkocheck(unmmko1\_self\_test(session, &result\_code, message));

printf("Selftest result: %s (%d)\n", message, result\_code);

// Проводим тест обмена

unmkocheck(unmmko1\_test\_exchange(session, &result\_code, message));

printf("Exchange test result: %s (%d)\n", message, result\_code);

// Проводим тест памяти

unmkocheck(unmmko1\_test\_memory(session, &result\_code, message));

printf("Memory test result: %s (%d)\n", message, result\_code);

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return 0;

}

## Мониторинг сообщений

В этом примере демонстрируется работа мезонина в режиме монитора шины. Мониторинг сообщений производится между вызовами функций unmmko1\_mon\_start и unmmko1\_mon\_stop. Для запроса количества готовых сообщений и самих сообщений используются функции unmmko1\_mon\_messages\_count и unmmko1\_mon\_messages\_read. В приведённом примере вызовы функций происходят после остановки монитора шины, но допустимо использовать их и в процессе мониторинга сообщений.

Установление и последующий разрыв связи с устройством выполняется по тому же сценарию, что и в предыдущем примере.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt32 messages\_count = 0;

unmmko1\_message\* messages = NULL;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_mon\_configure(session, UNMMKO1\_MON\_TRANSFORM | UNMMKO1\_MON\_BUS\_A\_AND\_B));

unmkocheck(unmmko1\_mon\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_mon\_stop(session));

unmkocheck(unmmko1\_mon\_messages\_count(session, &messages\_count));

messages = (unmmko1\_message\*)calloc(messages\_count, sizeof(unmmko1\_message));

unmkocheck(unmmko1\_mon\_messages\_read(session, messages\_count, messages, &messages\_count));

print\_messages(messages\_count, messages);

free(messages);

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Фильтрация сообщений

Использование фильтров в мониторе шины показано в следующем примере. Для определения сообщений, которые пройдут через фильтр, необходимо воспользоваться функциями unmmko1\_mon\_filter\_address и unmmko1\_mon\_filter\_subaddress.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt32 messages\_count = 0;

unmmko1\_message\* messages = NULL;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_mon\_configure(session, UNMMKO1\_MON\_TRANSFORM | UNMMKO1\_MON\_BUS\_A\_AND\_B));

// Фильтрация по адресам: адреса ОУ 0, 1, 2 и 3

unmkocheck(unmmko1\_mon\_filter\_address(session, 0x0000000f));

// Фильтрация для ОУ 0: отбрасываются все команды управления

unmkocheck(unmmko1\_mon\_filter\_subaddress(session, RT\_0, 0xffffffff, 0xffffffff, 0x00000000, 0x00000000));

// Фильтрация для ОУ 1: отбрасываются обращения к подадресам приёма/передачи данных от 0 до 15

unmkocheck(unmmko1\_mon\_filter\_subaddress(session, RT\_1, 0xffff0000, 0xffff0000, 0xffffffff, 0xffffffff));

unmkocheck(unmmko1\_mon\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_mon\_stop(session));

unmkocheck(unmmko1\_mon\_messages\_count(session, &messages\_count));

messages = (unmmko1\_message\*)calloc(messages\_count, sizeof(unmmko1\_message));

unmkocheck(unmmko1\_mon\_messages\_read(session, messages\_count, messages, &messages\_count));

print\_messages(messages\_count, messages);

free(messages);

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Отправка отдельных сообщений

Пример демонстрирует использование контроллера шины для отправки командных сегментов сообщений различных форматов. Для создания командного сегмента следует использовать функции unmmko1\_fN, где N – номер формата от 1 до 10. Также можно воспользоваться функциями unmmko1\_bc\_rt и unmmko1\_rt\_rt.

Для отправки отдельного командного сегмента используется функция unmmko1\_bc\_transmit\_command.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt16 data\_words[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32};

size\_t command\_index = 0;

unmmko1\_command commands[10];

commands[0] = unmmko1\_f1 (UNMMKO1\_BUS\_A, RT\_3, SA\_2, 32, data\_words);

commands[1] = unmmko1\_f2 (UNMMKO1\_BUS\_B, RT\_3, SA\_2, 2);

commands[2] = unmmko1\_f3 (UNMMKO1\_BUS\_A, RT\_1, SA\_2, RT\_3, SA\_2, 5);

commands[3] = unmmko1\_f4 (UNMMKO1\_BUS\_B, RT\_1, SA\_MC0, 2);

commands[4] = unmmko1\_f5 (UNMMKO1\_BUS\_A, RT\_1, SA\_MC31, 19);

commands[5] = unmmko1\_f6 (UNMMKO1\_BUS\_A, RT\_1, SA\_MC0, 17, 0);

commands[6] = unmmko1\_f7 (UNMMKO1\_BUS\_A, SA\_2, 32, data\_words);

commands[7] = unmmko1\_f8 (UNMMKO1\_BUS\_A, SA\_2, RT\_1, SA\_2, 5);

commands[8] = unmmko1\_f9 (UNMMKO1\_BUS\_A, SA\_MC0, 1);

commands[9] = unmmko1\_f10(UNMMKO1\_BUS\_B, SA\_MC31, 17, 0);

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_bc\_configure(session, UNMMKO1\_BC\_DEFAULT));

unmkocheck(unmmko1\_bc\_start(session));

for (command\_index = 0; command\_index < 10; ++command\_index) {

unmkocheck(unmmko1\_bc\_transmit\_command(session, commands[command\_index]));

Sleep(1);

}

unmkocheck(unmmko1\_bc\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Создание и отправка расписания сообщений

Данный пример выполняет отправку командных сегментов в виде расписания. С помощью вызова функции unmkko1\_bc\_schedule\_command командные сегменты добавляются в расписание контроллера шины. После запуска функцией unmmko1\_bc\_start контроллер шины начинает поочерёдно отправлять команды из расписания.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt16 data\_words[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32};

size\_t command\_index = 0;

unmmko1\_command commands[10];

commands[0] = unmmko1\_f1 (UNMMKO1\_BUS\_A, RT\_3, SA\_2, 32, data\_words);

commands[1] = unmmko1\_f2 (UNMMKO1\_BUS\_B, RT\_3, SA\_2, 2);

commands[2] = unmmko1\_f3 (UNMMKO1\_BUS\_A, RT\_1, SA\_2, RT\_3, SA\_2, 5);

commands[3] = unmmko1\_f4 (UNMMKO1\_BUS\_A, RT\_1, SA\_MC0, 2);

commands[4] = unmmko1\_f5 (UNMMKO1\_BUS\_A, RT\_1, SA\_MC31, 19);

commands[5] = unmmko1\_f6 (UNMMKO1\_BUS\_A, RT\_1, SA\_MC0, 17, 0);

commands[6] = unmmko1\_f7 (UNMMKO1\_BUS\_A, SA\_2, 32, data\_words);

commands[7] = unmmko1\_f8 (UNMMKO1\_BUS\_A, SA\_2, RT\_1, SA\_2, 5);

commands[8] = unmmko1\_f9 (UNMMKO1\_BUS\_A, SA\_MC0, 1);

commands[9] = unmmko1\_f10(UNMMKO1\_BUS\_A, SA\_MC31, 17, 0);

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_bc\_configure(session, UNMMKO1\_BC\_DEFAULT));

for (command\_index = 0; command\_index < 10; ++command\_index)

unmkocheck(unmmko1\_bc\_schedule\_command(session, commands[command\_index]));

unmkocheck(unmmko1\_bc\_start(session));

Sleep(1000);

unmkocheck(unmmko1\_bc\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Циклическая отправка расписания сообщений

Листинг программы ниже настраивает расписание из трех командных сегментов. Вызовом функции unmmko1\_bc\_set\_schedule\_repeat\_count устанавливается количество запусков расписания. В приведённом примере всего будет отправлено 30 команд (10 повторений по 3 команды) – 20 раз будет отправлена команда формата 1, 10 раз – формата 2.

Если в функции unmmko1\_bc\_set\_schedule\_repeat\_count в качестве аргумента количества повторений указать значение 0, то отправка будет зациклена и завершится только после останова контроллера шины.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt16 data\_words[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

unmmko1\_command command\_f1 = unmmko1\_f1(UNMMKO1\_BUS\_A, RT\_3, SA\_2, 10, data\_words);

unmmko1\_command command\_f2 = unmmko1\_f2(UNMMKO1\_BUS\_B, RT\_3, SA\_2, 2);

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_bc\_configure(session, UNMMKO1\_BC\_DEFAULT));

unmkocheck(unmmko1\_bc\_set\_schedule\_repeat\_count(session, 10));

unmkocheck(unmmko1\_bc\_schedule\_command(session, command\_f1));

unmkocheck(unmmko1\_bc\_schedule\_command(session, command\_f2));

unmkocheck(unmmko1\_bc\_schedule\_command(session, command\_f1));

unmkocheck(unmmko1\_bc\_start(session));

Sleep(1000);

unmkocheck(unmmko1\_bc\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Мониторинг сообщений в режиме контроллера шины

Режим контроллера шины совмещён с режимом монитора шины. Для начала использования монитора шины одновременно с контроллером нужно выполнить функцию unmmko1\_mon\_start, не вызывая предварительно unmmko1\_mon\_configure.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt16 data\_words[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

unmmko1\_command command\_f1 = unmmko1\_f1(UNMMKO1\_BUS\_A, RT\_3, SA\_2, 10, data\_words);

unmmko1\_command command\_f2 = unmmko1\_f2(UNMMKO1\_BUS\_B, RT\_3, SA\_2, 2);

ViUInt32 messages\_count = 0;

unmmko1\_message\* messages = NULL;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

unmkocheck(unmmko1\_bc\_configure(session, UNMMKO1\_BC\_DEFAULT));

unmkocheck(unmmko1\_bc\_start(session));

unmkocheck(unmmko1\_mon\_start(session));

unmkocheck(unmmko1\_bc\_transmit\_command(session, command\_f1));

Sleep(1);

unmkocheck(unmmko1\_bc\_transmit\_command(session, command\_f2));

Sleep(1);

unmkocheck(unmmko1\_mon\_stop(session));

unmkocheck(unmmko1\_bc\_stop(session));

unmkocheck(unmmko1\_mon\_messages\_count(session, &messages\_count));

messages = (unmmko1\_message\*)calloc(messages\_count, sizeof(unmmko1\_message));

unmkocheck(unmmko1\_mon\_messages\_read(session, messages\_count, messages, &messages\_count));

print\_messages(messages\_count, messages);

free(messages);

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Создание ответных сегментов сообщений

В следующем примере демонстрируется использование мезонина в качестве оконечного устройства. В листинге настраивается имитация трёх оконечных устройства с адресами 0, 2 и 3. Для выбранных подадресов установлены слова данных, которые будут отправлены оконечными устройствами в ответ на команду передачи данных.

#include "common.h"

static ViUInt16 data\_words\_3[3] = {0x03, 0x03, 0x03};

static ViUInt16 data\_words\_9[9] = {0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7, 0xf8, 0xf9};

static ViUInt16 data\_words\_32[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32};

int **main**() {

ViStatus status = VI\_SUCCESS;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Настраиваем оконечные устройства с адресами 0, 2 и 3

// Оконечные устройства, настроенные с опцией UNMMKO1\_RT\_DEFAULT\_RESPONSES, инициализируют слова данных нулями для всех подадресов

unmkocheck(unmmko1\_rt\_configure(session, (1 << RT\_0) | (1 << RT\_2) | (1 << RT\_3), UNMMKO1\_RT\_TRANSFORM | UNMMKO1\_RT\_BUS\_A\_AND\_B | UNMMKO1\_RT\_DEFAULT\_RESPONSES));

// Изменяем слова данных для оконечных устройств

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_1, 3, data\_words\_3));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_2, 9, data\_words\_9));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_2, SA\_2, 9, data\_words\_9));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_3, SA\_1, 32, data\_words\_32));

unmkocheck(unmmko1\_rt\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_rt\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Создание ответных сегментов сообщений в ручном режиме

Программа, приведённая на следующем листинге, настраивает два оконечных устройства (с адресами 0 и 1) и устанавливает данные для подадресов 1 и 30. Остальные подадреса отключены, обращения к ним рассматриваются как недопустимые команды и будут проигнорированы.

#include "common.h"

static ViUInt16 data\_words[32] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32};

int **main**() {

ViStatus status = VI\_SUCCESS;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Настраиваем оконечные устройства с адресами 0 и 1

// Оконечные устройства, настроенные с опцией UNMMKO1\_RT\_CUSTOM\_RESPONSES, будут отвечать только

// на обращения к тем подадресам, для которых непосредственно установлены слова данных

unmkocheck(unmmko1\_rt\_configure(session, (1 << RT\_0) | (1 << RT\_1), UNMMKO1\_RT\_TRANSFORM | UNMMKO1\_RT\_BUS\_A\_AND\_B | UNMMKO1\_RT\_CUSTOM\_RESPONSES));

// Оконечные устройства будут отвечать только на обращения к 1 и 30 подадресам

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_1, 32, data\_words));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_30, 32, data\_words));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_1, SA\_1, 32, data\_words));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_1, SA\_30, 32, data\_words));

unmkocheck(unmmko1\_rt\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_rt\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Создание ответных сегментов в режиме очереди данных

Мезонин ММКО1 поддерживает потоковую передачу слов данных с подадреса оконечного устройства, этот режим называется режимом очереди данных. В приведённом примере настраивается оконечное устройство с адресом 0 и очередями слов данных для 1 и 3 подадресов. Для каждого подадреса устанавливается 256 слов данных, которые поочерёдно могут быть вычитаны контроллером шины за несколько вызовов команд передачи данных.

#include "common.h"

static ViUInt16 data\_words\_count = 256;

static ViUInt16\* data\_words = NULL;

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt16 index = 0;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Настраиваем оконечное устройство с адресами 0

unmkocheck(unmmko1\_rt\_configure(session, 1 << RT\_0, UNMMKO1\_RT\_TRANSFORM | UNMMKO1\_RT\_BUS\_A\_AND\_B | UNMMKO1\_RT\_DEFAULT\_RESPONSES));

// Создаём массив слов данных

data\_words = (ViUInt16\*)calloc(data\_words\_count, sizeof(ViUInt16));

for (index = 0; index < data\_words\_count; ++index)

data\_words[index] = index + 1;

// Устанавливаем режим очереди и слова данных для 1 и 3 подадресов

unmkocheck(unmmko1\_rt\_set\_subaddress\_options(session, RT\_0, SA\_1, UNMMKO1\_RT\_SUBADDRESS\_QUEUE));

unmkocheck(unmmko1\_rt\_set\_subaddress\_options(session, RT\_0, SA\_3, UNMMKO1\_RT\_SUBADDRESS\_QUEUE));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_1, data\_words\_count, data\_words));

unmkocheck(unmmko1\_rt\_set\_subaddress\_data(session, RT\_0, SA\_3, data\_words\_count, data\_words));

free(data\_words);

unmkocheck(unmmko1\_rt\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_rt\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Циклический возврат данных

Листинг показывает, как сконфигурировать 30 подадрес для приёма сообщений циклического возврата.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Настраиваем оконечное устройство с адресами 0

unmkocheck(unmmko1\_rt\_configure(session, 1 << RT\_0, UNMMKO1\_RT\_TRANSFORM | UNMMKO1\_RT\_BUS\_A\_AND\_B | UNMMKO1\_RT\_DEFAULT\_RESPONSES));

// Устанавливаем режим циклического возврата данных для 30 подадреса

unmkocheck(unmmko1\_rt\_set\_subaddress\_options(session, RT\_0, SA\_30, UNMMKO1\_RT\_SUBADDRESS\_WRAP));

unmkocheck(unmmko1\_rt\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_rt\_stop(session));

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

## Мониторинг сообщений в режиме оконечного устройства

Оконечное устройство также, как и контроллер шины, может быть совмещён с режимом монитора шины. Для начала использования монитора шины нужно выполнить функцию unmmko1\_mon\_start, не вызывая предварительно unmmko1\_mon\_configure.

#include "common.h"

int **main**() {

ViStatus status = VI\_SUCCESS;

ViUInt32 messages\_count = 0;

unmmko1\_message\* messages = NULL;

unmkocheck(search\_unmmko1(resource\_name, &position));

unmbasecheck(unmbase\_init(resource\_name, VI\_TRUE, VI\_TRUE, &carrier\_session));

unmkocheck(unmmko1\_init(resource\_name, VI\_TRUE, VI\_TRUE, &session));

unmkocheck(unmmko1\_connect(session, carrier\_session, position, VI\_TRUE, VI\_TRUE));

// Настраиваем все оконечные устройства

unmkocheck(unmmko1\_rt\_configure(session, 0xffffffffu, UNMMKO1\_RT\_TRANSFORM | UNMMKO1\_RT\_BUS\_A\_AND\_B | UNMMKO1\_RT\_DEFAULT\_RESPONSES));

unmkocheck(unmmko1\_rt\_start(session));

unmkocheck(unmmko1\_mon\_start(session));

printf("Press a key to continue\n");

while (!key\_pressed())

Sleep(1);

unmkocheck(unmmko1\_mon\_stop(session));

unmkocheck(unmmko1\_rt\_stop(session));

unmkocheck(unmmko1\_mon\_messages\_count(session, &messages\_count));

messages = (unmmko1\_message\*)calloc(messages\_count, sizeof(unmmko1\_message));

unmkocheck(unmmko1\_mon\_messages\_read(session, messages\_count, messages, &messages\_count));

print\_messages(messages\_count, messages);

free(messages);

unmkocheck(unmmko1\_close(session));

unmbasecheck(unmbase\_close(carrier\_session));

return status;

}

# Функции библиотеки

## Константы и перечисления

В этом разделе приводятся константы и перечисления, используемые в драйвере. Элементы перечислений оформлены в таблицах, в первом столбце таблицы перечислены имена элементов, во втором – их описания.

### Константы

В следующей таблице перечислены константы, объявленные в заголовочном файле драйвера ММКО1 unmmko1.h.

| Имя константы | Описание |
| --- | --- |
| UNMMKO1\_MODEL\_CODE | Код модели мезонина (0x50) |
| UNMMKO1\_ERROR\_OFFSET | Смещение кодов ошибок мезонина |
| UNMMKO1\_WARNING\_OFFSET | Смещение кодов предупреждений мезонина |
| UNMMKO1\_WARNING\_BAD\_INITILIZATION | Код предупреждения «Ошибка во время инициализации» |
| UNMMKO1\_WARNING\_ALREADY\_STARTED | Код предупреждения «Устройство уже запущено» |
| UNMMKO1\_WARNING\_ALREADY\_STOPPED | Код предупреждения «Устройство уже остановлено» |
| UNMMKO1\_ERROR\_UNKNOWN | Код ошибки «Неизвестная ошибка» |
| UNMMKO1\_ERROR\_NULL\_POINTER | Код ошибки «Указатель NULL передан в качестве параметра» |
| UNMMKO1\_ERROR\_NOT\_ENOUGH\_MEMORY | Код ошибки «Недостаточно памяти» |
| UNMMKO1\_ERROR\_INTERNAL | Код ошибки «Внутренняя ошибка» |
| UNMMKO1\_ERROR\_SELFTEST | Код ошибки «Ошибка самоконтроля» |
| UNMMKO1\_ERROR\_WAS\_CONFIGURED\_IN\_  A\_DIFFERENT\_MODE | Код ошибки «Устройство было сконфигурировано в другом режиме» |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_  WHILE\_RUNNING | Код ошибки «Недопустимая операция при запущенном устройстве» |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_  WHILE\_STOPPED | Код ошибки «Недопустимая операция при остановленном устройстве» |
| UNMMKO1\_ERROR\_RT\_ADDRESS\_WAS\_NOT\_  CONFIGURED | Код ошибки «Оконечное устройство не настроено» |
| UNMMKO1\_ERROR\_CANNOT\_UPDATE\_DATA\_  WITH\_QUEUE\_MODE | Код ошибки «Недопустимо обновление данных в режиме очереди данных» |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_IN\_  SCHEDULE\_MODE | Код ошибки «Недопустимая операция в режиме работы по расписанию» |
| UNMMKO1\_ERROR\_EXCEEDED\_SCHEDULE\_  SIZE\_LIMIT | Код ошибки «Превышен предел размера расписания» |
| UNMMKO1\_ERROR\_FIRST\_CODE | Значение первого кода ошибки |
| UNMMKO1\_ERROR\_LAST\_CODE | Значение последнего кода ошибки |
| UNMMKO1\_WARNING\_FIRST\_CODE | Значение первого кода предупреждения |
| UNMMKO1\_WARNING\_LAST\_CODE | Значение последнего кода предупреждения |
| UNMMKO1\_BC\_SCHEDULE\_SIZE\_LIMIT | Максимальное количество командных сегментов в расписании с учётом количества повторений |
| UNMMKO1\_RT\_QUEUE\_SIZE\_LIMIT | Максимальный размер очереди слов данных для всех подадресов в режиме очереди слов данных |

### unmmko1\_bus

Магистраль передачи/приёма информации.

|  |  |
| --- | --- |
| Элемент перечисления | Описание |
| UNMMKO1\_BUS\_A | Основная магистраль |
| UNMMKO1\_BUS\_B | Резервная магистраль |

### unmmko1\_message\_activities

Флаги действий. Перечисленные действия используются в поле activity структур unmmko1\_command и unmmko1\_message.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_MSG\_ACT\_BUS\_A | Передача/приём по основной магистрали А (устанавливается в командном сегменте по умолчанию) |
| UNMMKO1\_MSG\_ACT\_BUS\_B | Передача/приём по резервной магистрали B |
| UNMMKO1\_MSG\_ACT\_CWD\_1 | В сообщении или командном сегменте присутствует командное слово 1 (устанавливается в командном сегменте по умолчанию) |
| UNMMKO1\_MSG\_ACT\_CWD\_2 | В сообщении или командном сегменте присутствует командное слово 2 |
| UNMMKO1\_MSG\_ACT\_SWD\_1 | В сообщении присутствует ответное слово 1 |
| UNMMKO1\_MSG\_ACT\_SWD\_2 | В сообщении присутствует ответное слово 2 |

### unmmko1\_message\_errors

Флаги ошибок, зафиксированных в сообщении. Перечисленные ошибки можно проверить в unmmko1\_message\_s.error, используя побитовое И.

|  |  |
| --- | --- |
| Элемент перечисления | Описание |
| UNMMKO1\_MSG\_ERR\_OK | Ошибки не зафиксированы |
| UNMMKO1\_MSG\_ERR\_NO\_RESPONSE | Состояние отсутствие ответа (таймаут ожидания ответа) |
| UNMMKO1\_MSG\_ERR\_ANY\_ERROR\_BIT | Установлен хотя бы один из признаков ошибки в ответном слове |
| UNMMKO1\_MSG\_ERR\_PROTOCOL | Протокольная ошибка |
| UNMMKO1\_MSG\_ERR\_DATA\_COUNT | Некорректное количество слов данных |
| UNMMKO1\_MSG\_ERR\_MANCHECTER | Ошибка манчестера |
| UNMMKO1\_MSG\_ERR\_SYSTEM | Внутренняя ошибка |

### unmmko1\_error\_bits

Биты ошибок.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_ERRBIT\_MESSAGE\_ERROR | [ОшС] Ошибка в сообщении  (9 бит) |
| UNMMKO1\_ERRBIT\_INSTRUMENTATION | [ПОС] Передача ответного слова (10 бит) |
| UNMMKO1\_ERRBIT\_SERVICE\_REQUEST | [ЗО] Запрос на обслуживание  (11 бит) |
| UNMMKO1\_ERRBIT\_RESERVED\_12 | Резерв (12 бит) |
| UNMMKO1\_ERRBIT\_RESERVED\_13 | Резерв (13 бит) |
| UNMMKO1\_ERRBIT\_RESERVED\_14 | Резерв (14 бит) |
| UNMMKO1\_ERRBIT\_BROADCAST\_COMMAND\_RECEIVED | [ПГК] Принята групповая команда (15 бит) |
| UNMMKO1\_ERRBIT\_BUSY | [АЗ] Абонент занят (16 бит) |
| UNMMKO1\_ERRBIT\_SYSTEM\_FLAG | [НА] Неисправность абонента  (17 бит) |
| UNMMKO1\_ERRBIT\_DYNAMIC\_BUS\_CONTROL\_ACCEPTANCE | [ПУИ] Принято управление интерфейсом (18 бит) |
| UNMMKO1\_ERRBIT\_TERMINAL\_FLAG | [НОУ] Неисправность ОУ (19 бит) |

### unmmko1\_address

Перечисление адресов оконечных устройств. Элемент перечисления RT\_BCAST определяет адрес для групповых команд. Чтобы не включать элементы данного перечисления в пространство имён, определите макрос UNMMKO1\_DISABLE\_CONSTANTS.

typedef enum {

RT\_0, RT\_1, RT\_2, RT\_3,

RT\_4, RT\_5, RT\_6, RT\_7,

RT\_8, RT\_9, RT\_10, RT\_11,

RT\_12, RT\_13, RT\_14, RT\_15,

RT\_16, RT\_17, RT\_18, RT\_19,

RT\_20, RT\_21, RT\_22, RT\_23,

RT\_24, RT\_25, RT\_26, RT\_27,

RT\_28, RT\_29, RT\_30, RT\_31,

RT\_BCAST = RT\_31

} unmmko1\_address;

### unmmko1\_subaddress

Перечисление подадресов оконечных устройств. Элементы перечисления SA\_MC0 и SA\_MC31 определяют подадрес 0 и 31 для команд управления соответственно. Чтобы не включать элементы данного перечисления в пространство имён, определите макрос UNMMKO1\_DISABLE\_CONSTANTS.

typedef enum {

SA\_0, SA\_1, SA\_2, SA\_3,

SA\_4, SA\_5, SA\_6, SA\_7,

SA\_8, SA\_9, SA\_10, SA\_11,

SA\_12, SA\_13, SA\_14, SA\_15,

SA\_16, SA\_17, SA\_18, SA\_19,

SA\_20, SA\_21, SA\_22, SA\_23,

SA\_24, SA\_25, SA\_26, SA\_27,

SA\_28, SA\_29, SA\_30, SA\_31,

SA\_MC0 = SA\_0,

SA\_MC31 = SA\_31

} unmmko1\_subaddress;

### unmmko1\_bc\_options

Флаги настроек контроллера шины. Для включения нескольких опций используйте побитовое ИЛИ. Используются в качестве аргумента bc\_options в функции unmmko1\_bc\_configure.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_BC\_DEFAULT | Настройки по умолчанию |
| UNMMKO1\_BC\_TRANSFORM | Трансформаторное подключение (по умолчанию) |
| UNMMKO1\_BC\_DIRECT | Непосредственное подключение |

### unmmko1\_rt\_options

Флаги настроек оконечного устройства. Для включения нескольких опций используйте побитовое ИЛИ. Используются в качестве аргумента rt\_options в функции unmmko1\_rt\_configure.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_RT\_DEFAULT | Настройки по умолчанию |
| UNMMKO1\_RT\_TRANSFORM | Трансформаторное подключение (по умолчанию) |
| UNMMKO1\_RT\_DIRECT | Непосредственное подключение |
| UNMMKO1\_RT\_BUS\_A\_AND\_B | Оконечное устройство отвечает на сообщения, присланные по магистрали A и B (по умолчанию) |
| UNMMKO1\_RT\_BUS\_A | Оконечное устройство отвечает только на сообщения, присланные по магистрали A |
| UNMMKO1\_RT\_BUS\_B | Оконечное устройство отвечает только на сообщения, присланные по магистрали B |
| UNMMKO1\_RT\_DEFAULT\_RESPONSES | Оконечное устройство создаёт ответы по умолчанию на все типы сообщений (по умолчанию) |
| UNMMKO1\_RT\_CUSTOM\_RESPONSES | Ответные сегменты оконечного устройства настраиваются отдельно для каждого сообщения |

### unmmko1\_rt\_subaddress\_options

Флаги настроек подадресов оконечного устройства. Используются в качестве аргумента options в функции unmmko1\_rt\_set\_subaddress\_options.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_RT\_SUBADDRESS\_DEFAULT | Настройки по умолчанию |
| UNMMKO1\_RT\_SUBADDRESS\_WRAP | Для указанных адреса и подадреса используется циклический возврат данных (принятые оконечным устройством данные перезаписывают данные на передачу) |
| UNMMKO1\_RT\_SUBADDRESS\_QUEUE | Для указанных адреса и подадреса используется режим очереди данных (число слов данных может быть больше 32, при запросе данных ОУ возвращает следующие слова данные) |

### unmmko1\_mon\_options

Флаги настроек монитора. Для включения нескольких опций используйте побитовое ИЛИ. Используются в качестве аргумента mon\_options в функции unmmko1\_mon\_configure.

| Элемент перечисления | Описание |
| --- | --- |
| UNMMKO1\_MON\_DEFAULT | Настройки по умолчанию |
| UNMMKO1\_MON\_TRANSFORM | Трансформаторное подключение (по умолчанию) |
| UNMMKO1\_MON\_DIRECT | Непосредственное подключение |
| UNMMKO1\_MON\_BUS\_A\_AND\_B | Монитор шины мониторирует сообщения по магистралям A и B (по умолчанию) |
| UNMMKO1\_MON\_BUS\_A | Монитор шины мониторирует сообщения только по магистрали A |
| UNMMKO1\_MON\_BUS\_B | Монитор шины мониторирует сообщения только по магистрали B |

## Определения типов

В этом разделе перечислены типы, используемые в библиотеке драйвера. Для описания полей структур используются таблицы, в первом столбце таблицы перечислены имена полей, во втором – описания соответствующих полей.

### unmmko1\_command

**Определение:**

typedef struct {

ViUInt16 activity;

ViUInt16 command\_word\_1;

ViUInt16 command\_word\_2;

ViUInt16 data\_words\_count;

ViUInt16 data\_words[32];

} unmmko1\_command;

**Назначение:**

Командный сегмент сообщения.

**Поля:**

|  |  |
| --- | --- |
| activity | Флаги действий из перечисления unmmko1\_message\_activities |
| command\_word\_1 | Первое командное слово |
| command\_word\_2 | Второе командное слово |
| data\_count | Количество слов данных |
| data\_words[32] | Слова данных |

### unmmko1\_response

**Определение:**

typedef struct {

ViUInt16 status\_word;

ViUInt16 data\_words\_count;

ViUInt16 data\_words[32];

} unmmko1\_response;

**Назначение:**

Ответный сегмент сообщения.

**Поля:**

|  |  |
| --- | --- |
| status\_word | Ответное слово |
| data\_count | Количество слов данных |
| data\_words[32] | Слова данных |

### unmmko1\_message

**Определение:**

typedef struct {

ViUInt32 timestamp\_low;

ViUInt32 timestamp\_high;

ViUInt16 activity;

ViUInt16 error;

unmmko1\_command command;

unmmko1\_response response\_1;

unmmko1\_response response\_2;

} unmmko1\_message;

**Назначение:**

Структура сообщения.

**Поля:**

|  |  |
| --- | --- |
| timestamp\_low | Метка времени (младшие 32 бита) |
| timestamp\_high | Метка времени (старшие 32 бита) |
| activity | Флаги зафиксированных действий из перечисления unmmko1\_message\_activities |
| error | Флаги зафиксированных ошибок из перечисления unmmko1\_message\_errors |
| command | Командный сегмент сообщения |
| response\_1 | Первый ответный сегмент сообщения |
| response\_2 | Второй ответный сегмент сообщения |

## Функции управления мезонином

В разделе функций приводятся описание функций драйвера.

Для каждой функции указаны определение, назначение, аргументы функции, возвращаемое значение (в случае, если функция имеет возвращаемое значение).

Аргументы функции оформлены в таблицах: в первом столбце таблицы указывается направление передачи аргумента (in – входной аргумент, out – выходной, in,out – и входной и выходной), во втором – имя аргумента, в третьем – его описание.

### unmmko1\_init

**Определение**:

ViStatus unmmko1\_init(ViRsrc resource\_name, ViBoolean idn\_query, ViBoolean do\_reset, ViSession\* session)

**Назначение**:

Функция unmmko1\_init производит действия по инициализации мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | resource\_name | Данная переменная указывает тип интерфейса и адрес устройства для инструмента, с которым открывается сеанс |
| in | idn\_query | Данный параметр определяет необходимость запроса идентификатора мезонина в процедуре инициализации |
| in | do\_reset | Параметр определяет необходимость сброса мезонина в процедуре инициализации |
| out | session | В данной переменной функция возвращает уникальный номер сеанса связи с мезонина, который необходимо сохранить для всех последующих вызовов функций драйвера мезонина |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_connect

**Определение**:

ViStatus unmmko1\_connect(ViSession session, ViSession carrier\_session, ViUInt16 mezzanine\_number, ViBoolean idn\_query, ViBoolean do\_reset)

**Назначение**:

Функция unmmko1\_connect готовит к работе сеанс мезонина, открытый ранее с помощью функции инициализации. Данную функцию необходимо вызывать один раз сразу после инициализации мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | carrier\_session | Номер сеанса связи с носителем мезонинов |
| in | mezzanine\_  number | Позиция мезонина на носителе мезонинов |
| in | idn\_query | Данный параметр определяет необходимость запроса идентификатора мезонина |
| in | do\_reset | Данный параметр определяет необходимость сброса мезонина |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_self\_test

**Определение**:

ViStatus unmmko1\_self\_test(ViSession session, ViInt16\* result, ViChar message[])

**Назначение**:

Функция unmmko1\_self\_test производит самоконтроль мезонина и возвращает его результат.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | result | Код результата самоконтроля |
| out | message | Сообщение о результате самоконтроля |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_test\_exchange

**Определение**:

ViStatus unmmko1\_test\_exchange(ViSession session, ViInt16\* result, ViChar message[])

**Назначение**:

Функция unmmko1\_test\_exchange производит тестирование обмена в различных режимах работы мезонина. Тестирование обмена - длительная операция.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | result | Код результата тестирования |
| out | message | Сообщение о результате тестирования |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_test\_memory

**Определение**:

ViStatus unmmko1\_test\_memory(ViSession session, ViInt16\* result, ViChar message[])

**Назначение**:

Функция unmmko1\_test\_memory производит тестирование памяти мезонина. Тестирование памяти - длительная операция.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | result | Код результата тестирования |
| out | message | Сообщение о результате тестирования |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_reset

**Определение**:

ViStatus unmmko1\_reset(ViSession session)

**Назначение**:

Функция unmmko1\_reset функция производит сброс мезонина в состояние по умолчанию. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_install\_interrupt\_handler

**Определение**:

ViStatus unmmko1\_install\_interrupt\_handler (ViSession session, UNMMKO1\_EVENT\_HANDLER handler, ViAddr data)

**Назначение**:

Функция unmmko1\_install\_interrupt\_handler устанавливаем пользовательскую функцию для приёма прерываний от мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | handler | Пользовательский обработчик прерываний - указатель на функцию с семантикой, заданной в UNMMKO1\_EVENT\_HANDLER |
| In | data | Указатель на данные, передаваемые в пользовательский обработчик прерываний |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_error\_query

**Определение**:

ViStatus unmmko1\_error\_query(ViSession session, ViPInt32 error, ViChar message[])

**Назначение**:

Функция unmmko1\_error\_query возвращает текущие код и сообщение об ошибке при работе с мезонином. Для данного мезонина функция не поддерживается и возвращает статус VI\_WARN\_NSUP\_ERROR\_QUERY.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | error | Код ошибки |
| out | message | Сообщение об ошибке |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_revision\_query

**Определение**:

ViStatus unmmko1\_revision\_query(ViSession session, ViChar software\_version[], ViChar hardware\_version[])

**Назначение**:

Функция unmmko1\_revision\_query возвращает версию драйвера инструмента. Для данного мезонина функция не поддерживает возврат версии аппаратного обеспечения и возвращает статус VI\_WARN\_NSUP\_REV\_QUERY.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | software\_  version | Версия программного обеспечения мезонина |
| out | hardware\_  version | Версия аппаратного обеспечения мезонина (не поддерживается) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_error\_message

**Определение**:

ViStatus unmmko1\_error\_message (ViSession session, ViStatus status, ViChar error\_message [])

**Назначение**:

Функция unmmko1\_error\_message получает и интерпретирует код состояния, возвращенный какой-либо функцией драйвера, возвращает строку сообщения об ошибке. Поддерживается возврат сообщений на английском и русском языках, язык зависит от текущих региональных параметров (локали LC\_CTYPE).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | status | Код ошибки |
| out | error\_message | Сообщение об ошибке |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_close

**Определение**:

ViStatus unmmko1\_close (ViSession session)

**Назначение**:

Функция unmmko1\_close закрывает сеанс связи с мезонином.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

## Функции управления контроллером шины

### unmmko1\_bc\_rt

**Определение**:

unmmko1\_command unmmko1\_bc\_rt (unmmko1\_bus bus, ViUInt16 command\_word, ViUInt16 \* data\_words)

**Назначение**:

Функция unmmko1\_bc\_rt создаёт командный сегмент для сообщений обмена данными между контроллером шины и оконечным устройством. Функция предназначена для формирования сообщений с одним командным словом - форматы 1, 2, 4, 5, 6, 7, 9, 10. Требуемое количество слов данных определяется командным словом.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | command\_word | Командное слово (можно создать с помощью вызова функции unmmko1\_pack\_cw) |
| in | data\_words | Указатель на массив слов данных (может быть NULL для соответствующих форматов) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_rt\_rt

**Определение**:

unmmko1\_command unmmko1\_rt\_rt (unmmko1\_bus bus, ViUInt16 command\_word\_1, ViUInt16 command\_word\_2)

**Назначение**:

Функция unmmko1\_rt\_rt создаёт командный сегмент для сообщений обмена данными между оконечными устройствами. Функция предназначена для формирования сообщений с двумя командными словами - форматы 3 и 8.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | command\_  word\_1 | Первое командное слово (можно создать с помощью вызова функции unmmko1\_pack\_cw) |
| in | command\_  word\_2 | Первое командное слово (можно создать с помощью вызова функции unmmko1\_pack\_cw) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f1

**Определение**:

unmmko1\_command unmmko1\_f1 (unmmko1\_bus bus, ViUInt16 address, ViUInt16 subaddress, ViUInt16 word\_count, ViUInt16 \* data\_words)

**Назначение**:

Функция unmmko1\_f1 создаёт командный сегмент сообщения в формате 1. Формат 1 - передача слов данных от контроллера к оконечному устройству.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес (1 .. 30) |
| in | word\_count | Количество слов данных |
| in | data\_words | Указатель на массив слов данных (массив должен содержать как минимум word\_count слов данных) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f2

**Определение**:

unmmko1\_command unmmko1\_f2 (unmmko1\_bus bus, ViUInt16 address, ViUInt16 subaddress, ViUInt16 word\_count)

**Назначение**:

Функция unmmko1\_f2 создаёт командный сегмент сообщения в формате 2. Формат 2 - передача слов данных от оконечного устройства к контроллеру.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес (1 .. 30) |
| in | word\_count | Количество слов данных |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f3

**Определение**:

unmmko1\_command unmmko1\_f3 (unmmko1\_bus bus, ViUInt16 receive\_address, ViUInt16 receive\_subaddress, ViUInt16 transmit\_address, ViUInt16 transmit\_subaddress, ViUInt16 word\_count)

**Назначение**:

Функция unmmko1\_f3 создаёт командный сегмент сообщения в формате 3. Формат 3 - передача слов данных от оконечного устройства к оконечному устройству.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | receive\_  address | Адрес оконечного устройства, которое принимает слова данных |
| in | receive\_  subaddress | Подадрес приёма слов данных (1 .. 30) |
| in | transmit\_  address | Адрес оконечного устройства, которое передаёт слова данных |
| in | transmit\_  subaddress | Подадрес передачи слов данных (1 .. 30) |
| in | word\_count | Количество слов данных |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f4

**Определение**:

unmmko1\_command unmmko1\_f4 (unmmko1\_bus bus, ViUInt16 address, ViUInt16 subaddress, ViUInt16 mode\_code)

**Назначение**:

Функция unmmko1\_f4 создаёт командный сегмент сообщения в формате 4. Формат 4 - передача команды управления без слов данных.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес/режим управления (SA\_MC0 (0) или SA\_MC31 (31)) |
| in | mode\_code | Код команды управления (0 .. 8) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f5

**Определение**:

unmmko1\_command unmmko1\_f5 (unmmko1\_bus bus, ViUInt16 address, ViUInt16 subaddress, ViUInt16 mode\_code)

**Назначение**:

Функция unmmko1\_f5 создаёт командный сегмент сообщения в формате 5. Формат 5 - передача команды управления и приём слова данных от оконечного устройства.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес/режим управления (SA\_MC0 (0) или SA\_MC31 (31)) |
| in | mode\_code | Код команды управления (16, 18, 19) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f6

**Определение**:

unmmko1\_command unmmko1\_f6 (unmmko1\_bus bus, ViUInt16 address, ViUInt16 subaddress, ViUInt16 mode\_code, ViUInt16 data\_word)

**Назначение**:

Функция unmmko1\_f6 создаёт командный сегмент сообщения в формате 6. Формат 6 - передача команды управления со словом данных.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес/режим управления (SA\_MC0 (0) или SA\_MC31 (31)) |
| in | mode\_code | Код команды управления (17, 20, 21) |
| in | data\_word | Слово данных |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f7

**Определение**:

unmmko1\_command unmmko1\_f7 (unmmko1\_bus bus, ViUInt16 subaddress, ViUInt16 word\_count, ViUInt16 \* data\_words)

**Назначение**:

Функция unmmko1\_f7 создаёт командный сегмент сообщения в формате 7. Формат 7 - передача слов данных (в групповом сообщении) от контроллера шины к оконечным устройствам.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | subaddress | Подадрес (1 .. 30) |
| in | word\_count | Количество слов данных |
| in | data\_words | Указатель на массив слов данных (массив должен содержать как минимум word\_count слов данных) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f8

**Определение**:

unmmko1\_command unmmko1\_f8 (unmmko1\_bus bus, ViUInt16 receive\_subaddress, ViUInt16 transmit\_address, ViUInt16 transmit\_subaddress, ViUInt16 word\_count)

**Назначение**:

Функция unmmko1\_f8 создаёт командный сегмент сообщения в формате 8. Формат 8 - передача слов данных (в групповом сообщении) от оконечного устройства к оконечным устройствам.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | receive\_  subaddress | Подадрес приёма слов данных |
| in | transmit\_  address | Адрес оконечного устройства, которое передаёт слова данных |
| in | transmit\_  subaddress | Подадрес передачи слов данных |
| in | word\_count | Количество слов данных |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f9

**Определение**:

unmmko1\_command unmmko1\_f9 (unmmko1\_bus bus, ViUInt16 subaddress, ViUInt16 mode\_code)

**Назначение**:

Функция unmmko1\_f9 создаёт командный сегмент сообщения в формате 9. Формат 9 - передача групповой команды управления без слов данных.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | subaddress | Подадрес/режим управления (SA\_MC0 (0) или SA\_MC31 (31)) |
| in | mode\_code | Код команды управления (1, 3, 4, 5, 6, 7, 8) |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_f10

**Определение**:

unmmko1\_command unmmko1\_f10 (unmmko1\_bus bus, ViUInt16 subaddress, ViUInt16 mode\_code, ViUInt16 data\_word)

**Назначение**:

Функция unmmko1\_f10 создаёт командный сегмент сообщения в формате 10. Формат 10 - передача групповой команды управления со словом данных.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | bus | Выбранная магистраль для передачи командного сегмента |
| in | subaddress | Подадрес/режим управления (SA\_MC0 (0) или SA\_MC31 (31)) |
| in | mode\_code | Код команды управления (17, 20, 21) |
| in | data\_word | Слово данных |

**Возвращаемое значение:**

Командный сегмент.

### unmmko1\_pack\_cw

**Определение**:

ViUInt16 unmmko1\_pack\_cw (ViUInt16 address, ViUInt16 rx\_tx, ViUInt16 subaddress, ViUInt16 word\_count)

**Назначение**:

Функция unmmko1\_pack\_cw создаёт 16-битное командное слово из составляющих его полей.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | address | Адрес оконечного устройства |
| in | rx\_tx | Разряд приёма/передачи (приём - 0, передача - 1) |
| in | subaddress | Подадрес/режим управления |
| in | word\_count | Количество слов данных/код команды управления |

**Возвращаемое значение:**

Командное слово.

### unmmko1\_unpack\_cw

**Определение**:

void unmmko1\_unpack\_cw (ViUInt16 command\_word, ViUInt16 \* address, ViUInt16 \* rx\_tx, ViUInt16 \* subaddress, ViUInt16 \* word\_count)

**Назначение**:

Функция unmmko1\_unpack\_cw извлекает поля из 16-битного командного слова. Для игнорирования любого из возвращаемых аргументов передайте значение NULL вместо указателя на аргумент.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | command\_word | Командное слово |
| out | address | Адрес оконечного устройства |
| out | rx\_tx | Разряд приёма/передачи (приём - 0, передача - 1) |
| out | subaddress | Подадрес/режим управления |
| out | word\_count | Количество слов данных/код команды управления |

### unmmko1\_pack\_sw

**Определение**:

ViUInt16 unmmko1\_pack\_sw (ViUInt16 address, ViUInt16 error\_bits)

**Назначение**:

Функция unmmko1\_pack\_sw создаёт 16-битное ответное слово из составляющих его полей.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | address | Адрес оконечного устройства |
| in | error\_bits | Признаки ответного слова (объединение констант из перечисления unmmko1\_error\_bits) |

**Возвращаемое значение:**

Ответное слово.

### unmmko1\_unpack\_sw

**Определение**:

void \_VI\_FUNC unmmko1\_unpack\_sw (ViUInt16 status\_word, ViUInt16 \* address, ViUInt16 \* error\_bits)

**Назначение**:

Функция unmmko1\_pack\_sw извлекает поля из ответного слова. Для игнорирования любого из возвращаемых аргументов передайте значение NULL вместо указателя на аргумент.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | status\_word | Ответное слово |
| out | address | Адрес оконечного устройства |
| out | error\_bits | Признаки ответного слова (объединение констант из перечисления unmmko1\_error\_bits) |

### unmmko1\_bc\_configure

**Определение**:

ViStatus unmmko1\_bc\_configure (ViSession session, int bc\_options)

**Назначение**:

Функция unmmko1\_bc\_configure осуществляет настройку мезонина в режиме контроллера шины. Вызов функции также осуществляет сброс настроек контроллера. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | bc\_options | Настройки контроллера шины - объединение флагов UNMMKO1\_BC\_\* из перечисления unmmko1\_bc\_options |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_reset

**Определение**:

ViStatus unmmko1\_bc\_reset (ViSession session)

**Назначение**:

Функция unmmko1\_bc\_reset осуществляет сброс настроек контроллера шины: удаление командных сегментов из расписания, количество повторений сбрасывается в 1. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_schedule\_command

**Определение**:

ViStatus unmmko1\_bc\_schedule\_command (ViSession session, unmmko1\_command command)

**Назначение**:

Функция unmmko1\_bc\_schedule\_command добавляет командный сегмент в конец расписания. Количество командных сегментов в расписании, умноженное на количество повторений расписания, не должно превышать UNMMKO1\_BC\_SCHEDULE\_SIZE\_LIMIT.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | command | Командный сегмент сообщения |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_set\_schedule\_repeat\_count

**Определение**:

ViStatus unmmko1\_bc\_set\_schedule\_repeat\_count (ViSession session, ViUInt16 repeat\_count)

**Назначение**:

Функция unmmko1\_bc\_set\_schedule\_repeat\_count устанавливает количество повторений расписания. Количество командных сегментов в расписании, умноженное на количество повторений расписания, не должно превышать UNMMKO1\_BC\_SCHEDULE\_SIZE\_LIMIT.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | repeat\_count | Количество повторений расписания (значение 0 означает бесконечное повторение расписания) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_start

**Определение**:

ViStatus unmmko1\_bc\_start(ViSession session)

**Назначение**:

Функция unmmko1\_bc\_start запускает мезонин в режиме контроллера шины. В зависимости от того, было ли ранее установлено расписание, будет осуществляться запуск в режиме расписания или в режиме отправки одиночных командных сегментов. Настройки контроллера шины предварительно устанавливаются функцией unmmko1\_bc\_configure. Расписание настраивается функциями unmmko1\_bc\_schedule\_command и unmmko1\_bc\_set\_schedule\_repeat\_count.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_status

**Определение**:

ViStatus unmmko1\_bc\_status (ViSession session, ViBoolean \* status)

**Назначение**:

Функция unmmko1\_bc\_status возвращает состояние запуска мезонина в режиме контроллера шины. Функция возвращает VI\_TRUE после удачного запуска контроллера шины с помощью unmmko1\_bc\_start. Запрос состояния запуска не может использоваться как признак завершения выполнения расписания.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | status | Состояние запуска контроллера шины (VI\_TRUE - контроллер шины запущен, VI\_FALSE - остановлен) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_stop

**Определение**:

ViStatus unmmko1\_bc\_stop(ViSession session)

**Назначение**:

Функция unmmko1\_bc\_stop останавливает мезонин, запущенный ранее в режиме контроллера шины.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_bc\_transmit\_command

**Определение**:

ViStatus unmmko1\_bc\_transmit\_command (ViSession session, unmmko1\_command command)

**Назначение**:

Функция unmmko1\_bc\_transmit\_command однократно передаёт командный сегмент сообщения.

Примечание. Функция передачи командного сегмента недоступна и возвращает ошибку, если ранее было настроено расписание.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | command | Командный сегмент сообщения |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

## Функции управления оконечным устройством

### unmmko1\_rt\_configure

**Определение**:

ViStatus unmmko1\_rt\_configure (ViSession session, ViUInt32 addresses, int rt\_options)

**Назначение**:

Функция unmmko1\_rt\_configure осуществляет настройку мезонина в режиме оконечного устройства, определяет адреса имитируемых оконечных устройств. Вызов функции также осуществляет сброс настроек контроллера. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

Флаги UNMMKO1\_RT\_BUS\_A, UNMMKO1\_RT\_BUS\_B и UNMMKO1\_RT\_BUS\_A\_AND\_B (по умолчанию) определяют магистрали, по которым возможен обмен данными с оконечными устройствами. Установка флага UNMMKO1\_RT\_CUSTOM\_RESPONSES означает, что все команды приёма и передачи данных для всех подадресов оконечных устройств отмечаются как недопустимые. На недопустимые команды приёма/передачи оконечное устройство не отвечает. Для того, чтобы сделать команду допустимой (для определённых адреса и подадреса) нужно установить данные вызовом функции unmmko1\_rt\_set\_subaddress\_data. Использовать флаг UNMMKO1\_RT\_CUSTOM\_RESPONSES нужно в тех случаях, когда требуется часть команд приёма/передачи сделать недопустимыми. Установка же флага UNMMKO1\_RT\_DEFAULT\_RESPONSES (по умолчанию) означает, что все команды приёма и передачи данных для всех подадресов будут допустимыми. Также по умолчанию все слова данных будут установлены в значение 0, вызовом функции unmmko1\_rt\_set\_subaddress\_data значения слов данных можно переписать. Независимо от того, какой флаг (UNMMKO1\_RT\_CUSTOM\_RESPONSES или UNMMKO1\_RT\_DEFAULT\_RESPONSES) будет установлен, оконечное устройство будет отвечать на команды управления.

В оконечных устройствах реализовано выполнение следующих команд управления: "Передать ОС" (КУ 2), "Блокировать передатчик" (КУ 4), "Разблокировать передатчик" (КУ 5), "Установить ОУ в исходное состояние" (КУ 8), "Передать последнюю команду" (КУ 18).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | addresses | Маска адресов оконечных устройств (младший значащий бит отвечает за адрес оконечного устройства 0, второй бит - за 1 и так далее) |
| in | rt\_options | Настройки оконечного устройства - объединение флагов UNMMKO1\_RT\_\* из перечисления unmmko1\_rt\_options |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_reset

**Определение**:

ViStatus unmmko1\_rt\_reset (ViSession session)

**Назначение**:

Функция unmmko1\_rt\_reset производит сброс значений и настроек ответных сегментов. Настройка подадресов сбрасывается на значение по умолчанию - UNMMKO1\_RT\_SUBADDRESS\_DEFAULT. Значения слов данных, слова данных на команды управления обнуляются. Настройка ответного сегмента по умолчанию - UNMMKO1\_RT\_RESPONSE\_DEFAULT. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_set\_status\_word

**Определение**:

ViStatus unmmko1\_rt\_set\_status\_word (ViSession session, ViUInt16 address, ViUInt16 status\_word)

**Назначение**:

Функция unmmko1\_rt\_set\_status\_word предназначена для установки значения ответного слова для конкретного адреса оконечного устройства. Указываемый адрес оконечного устройства должен быть заранее настроен функцией unmmko1\_rt\_configure. Вызов функции возможен также в процессе работы мезонина в режиме оконечного устройства (после unmmko1\_rt\_start).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | address | Адрес оконечного устройства |
| in | status\_word | Ответное слово (можно создать с помощью вызова функции unmmko1\_pack\_sw) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_set\_subaddress\_options

**Определение**:

ViStatus unmmko1\_rt\_set\_subaddress\_options (ViSession session, ViUInt16 address, ViUInt16 subaddress, unmmko1\_rt\_subaddress\_options subaddress\_options)

**Назначение**:

Функция unmmko1\_rt\_set\_subaddress\_options предназначена для установки настроек подадреса оконечного устройства. Указываемый адрес оконечного устройства должен быть заранее настроен функцией unmmko1\_rt\_configure.

Флаг UNMMKO1\_RT\_SUBADDRESS\_DEFAULT определяет поведение по умолчанию - оконечное устройство поддерживает хранение 32 слов данных для подадреса. Установленные в этом режиме слова данных (функцией unmmko1\_rt\_set\_subaddress\_data) возвращаются контроллеру шины в ответ на команду передачи данных (формат 2). Если в команде передачи запрашивается меньше, чем 32 слова данных, то оконечное устройство возвращает слова данных, начиная с самого первого.

Флаг UNMMKO1\_RT\_SUBADDRESS\_WRAP определяет, что указанный подадрес оконечного устройства должен работать в режиме циклического возврата данных, то есть слова, принятые оконечным устройством в этот подадрес, перезаписывают слова данных на передачу. В этом режиме обеспечивается хранение 32 слов данных, устройство возвращает слова данных, начиная с самого первого.

Флаг UNMMKO1\_RT\_SUBADDRESS\_QUEUE включает режим очереди данных для указанного подадреса. В этом режиме обеспечивается хранение произвольного количества слов данных (слова данных и их количество устанавливаются функцией unmmko1\_rt\_set\_subaddress\_data до старта мезонина). Особенность режима очереди данных в том, что оконечное устройство передаёт слова данных по очереди. Первый запрос данных проводит к отправке слов, начиная с первого слова. Последующие запросы – начиная со следующего за последним переданным словом.

Вызов функции недоступен в процессе работы мезонина в режиме оконечного устройства (после unmmko1\_rt\_start).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес оконечного устройства |
| in | subaddress\_  options | Флаг настройки подадреса |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_set\_subaddress\_data

**Определение**:

ViStatus unmmko1\_rt\_set\_subaddress\_data (ViSession session, ViUInt16 address, ViUInt16 subaddress, int data\_words\_count, ViUInt16 \* data\_words)

**Назначение**:

Функция unmmko1\_rt\_set\_subaddress\_data предназначена для установки слов данных подадреса оконечного устройства. Вызов функции для подадреса в режиме по умолчанию (UNMMKO1\_RT\_SUBADDRESS\_DEFAULT) и в режиме циклического возврата данных (UNMMKO1\_RT\_SUBADDRESS\_WRAP) приводит к перезаписи указанного количества слов данных. В этих режимах вызов функции возможен также в процессе работы мезонина (после unmmko1\_rt\_start). В режиме очереди данных (UNMMKO1\_RT\_SUBADDRESS\_QUEUE) функция формирует очередь из data\_words\_count слов данных и записывает в неё слова. Длина очереди слов данных ограничена размером расширенной области памяти, которая составляет UNMMKO1\_RT\_QUEUE\_SIZE\_LIMIT слов данных на все очереди оконечных устройств. Вызов функции в режиме очереди данных недоступен в процессе работы мезонина (после unmmko1\_rt\_start).

Если оконечное устройство было настроено с флагом UNMMKO1\_RT\_CUSTOM\_RESPONSES, то unmmko1\_rt\_set\_subaddress\_data делает команды приёма/передачи к этому подадресу допустимыми.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | address | Адрес оконечного устройства |
| in | subaddress | Подадрес оконечного устройства |
| in | data\_words\_  count | Количество слов данных |
| in | data\_words | Массив слов данных |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_set\_command\_data

**Определение**:

ViStatus unmmko1\_rt\_set\_command\_data (ViSession session, ViUInt16 address, ViUInt16 command\_code, ViUInt16 data\_word)

**Назначение**:

Функция unmmko1\_rt\_set\_command\_data предназначена для установки слов данных в ответ на команду управления в формате сообщения 5. Возвращаемые слова данных возможно установить в ответ на команды управления "Передать векторное слово" (КУ 16) и "Передать слово ВСК ОУ" (КУ 19). Слово данных в ответе оконечного устройства на команду управления "Передать последнюю команду" (КУ 18) определяется только последней командой. Допустимо использовать функцию в процессе работы мезонина в режиме оконечного устройства (после unmmko1\_rt\_start).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | address | Адрес оконечного устройства |
| in | command\_code | Код команды управления |
| in | data\_word | Слово данных |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_start

**Определение**:

ViStatus unmmko1\_rt\_start (ViSession session)

**Назначение**:

Функция unmmko1\_rt\_start запускает мезонин в режиме оконечного устройства. Настройки режима предварительно устанавливаются функцией unmmko1\_rt\_configure.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_status

**Определение**:

ViStatus unmmko1\_rt\_status (ViSession session, ViBoolean \* status)

**Назначение**:

Функция unmmko1\_rt\_status возвращает состояние запуска мезонина в режиме оконечного устройства.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | status | Состояние запуска оконечного устройства (VI\_TRUE - режим оконечного устройства запущен, VI\_FALSE - остановлен) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_rt\_stop

**Определение**:

ViStatus unmmko1\_rt\_stop (ViSession session)

**Назначение**:

Функция unmmko1\_rt\_stop останавливает мезонин, запущенный ранее в режиме оконечного устройства.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

## Функции управления монитором шины

### unmmko1\_mon\_configure

**Определение**:

ViStatus unmmko1\_mon\_configure (ViSession session, int mon\_options)

**Назначение**:

Функция unmmko1\_mon\_configure осуществляет настройку мезонина в режиме монитора шины. Вызов функции также осуществляет сброс настроек монитора. Если мезонин был запущен в каком-либо режиме работы, то также будет произведена остановка мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | mon\_options | Настройки монитора - объединение флагов UNMMKO1\_MON\_\* из перечисления unmmko1\_mon\_options |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_reset

**Определение**:

ViStatus unmmko1\_mon\_reset (ViSession session)

**Назначение**:

Функция unmmko1\_mon\_reset производит сброс настроек времени отсутствия ответа и фильтрации, удаление сообщений монитора. Вызов функции не приводит к останову мезонина.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_set\_timeout

**Определение**:

ViStatus unmmko1\_mon\_set\_timeout (ViSession session, ViUInt16 timeout\_in\_us)

**Назначение**:

Функция unmmko1\_mon\_set\_timeout устанавливает значение времени отсутствия ответа, используемое монитором для разбора сообщений.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | timeout\_in\_us | Значение времени отсутствия ответа в микросекундах |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_filter\_address

**Определение**:

ViStatus unmmko1\_mon\_filter\_address (ViSession session, ViUInt32 addresses\_mask)

**Назначение**:

Функция unmmko1\_mon\_filter\_address позволяет установить фильтр по адресам оконечных устройств. Фильтр определяется маской, заданный в аргументе addresses\_mask. Младший значащий бит аргумента отвечает за адрес оконечного устройства 0. Старший значащий бит - за адрес 31. Если функция не используется, то мониторироваться будут все оконечные устройства.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | addresses\_mask | Маска адресов оконечных устройств |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_filter\_subaddress

**Определение**:

ViStatus unmmko1\_mon\_filter\_subaddress (ViSession session, ViUInt16 address, ViUInt32 receive\_subaddresses, ViUInt32 transmit\_subaddresses, ViUInt32 receive\_modecodes, ViUInt32 transmit\_modecodes)

**Назначение**:

Функция unmmko1\_mon\_filter\_subaddress позволяет установить фильтр по подадресам для конкретного оконечного устройства. Фильтр определяется масками подадресов приёма и передачи данных, масками кодов команд приёма и передачи. В каждой маске младший значащий бит отвечает за подадрес 0 или код команды 0. Старший значащий бит - за подадрес 31 или код команды 31. Если функция не используется, то мониторироваться будут все сообщения оконечного устройства.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | address | Адрес оконечного устройства |
| in | receive\_  subaddresses | Битовая маска подадресов приёма |
| in | transmit\_  subaddresses | Битовая маска подадресов передачи |
| in | receive\_  modecodes | Битовая маска команд управления приёма |
| in | transmit\_  modecodes | Битовая маска команд управления передачи |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_start

**Определение**:

ViStatus unmmko1\_mon\_start(ViSession session)

**Назначение**:

Функция unmmko1\_mon\_start запускает мезонин в режиме монитора шины. Настройки монитора шины предварительно устанавливаются функцией unmmko1\_mon\_configure. Монитор шины допускается использовать совместно с режимами контроллера шины или оконечного устройства, и в этом случае вызывать unmmko1\_mon\_configure не надо.

Примечание. При запуске монитора шины удаляются ранее записанные сообщения.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_status

**Определение**:

ViStatus unmmko1\_mon\_status (ViSession session, ViBoolean \* status)

**Назначение**:

Функция unmmko1\_mon\_status возвращает состояние запуска мезонина в режиме монитора шины.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | status | Состояние запуска монитора шины (VI\_TRUE - монитора шины запущен, VI\_FALSE - остановлен) |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_stop

**Определение**:

ViStatus unmmko1\_mon\_stop(ViSession session)

**Назначение**:

Функция unmmko1\_mon\_stop останавливает мезонин, запущенный ранее в режиме монитора шины. Если монитор шины используется совместно с режимами контроллера шины или оконечного устройства, то вызов функции приведёт к остановке только монитора шины. Считанные монитором шины сообщения можно запросить с помощью функций unmmko1\_mon\_messages\_count и unmmko1\_mon\_messages\_read.

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_messages\_count

**Определение**:

ViStatus unmmko1\_mon\_messages\_count (ViSession session, ViUInt32 \* messages\_count)

**Назначение**:

Функция unmmko1\_mon\_messages\_count служит для запроса количества сообщений, считанными монитором шины. Допустимо использовать функцию как в процессе работы мезонина в режиме монитора шины (после unmmko1\_mon\_start), так и после останова монитора (функцией unmmko1\_mon\_stop).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| out | records\_count | Количество считанных сообщений |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

### unmmko1\_mon\_messages\_read

**Определение**:

ViStatus unmmko1\_mon\_messages\_read (ViSession session, ViUInt32 messages\_count, unmmko1\_message \* messages, ViUInt32 \* read\_messages\_count)

**Назначение**:

Функция unmmko1\_mon\_messages\_read обеспечивает чтение сообщений в память компьютера. Допустимо использовать функцию как в процессе работы мезонина в режиме монитора шины (после unmmko1\_mon\_start), так и после останова монитора (функцией unmmko1\_mon\_stop).

**Аргументы**:

|  |  |  |
| --- | --- | --- |
| in | session | Номер сеанса связи с мезонином |
| in | messages\_count | Количество сообщений, которое необходимо считать |
| in/out | messages | Массив, в который считываются готовые сообщения. Массив должен состоять как минимум из messages\_count элементов типа unmmko1\_message |
| out | read\_messages\_count | Количество считанных сообщений в messages |

**Возвращаемое значение:**

Код ошибки или код успешного выполнения функции.

## Коды ошибок и предупреждений

В этом разделе приводятся ошибки и предупреждения, которые специфичны для данного мезонина. Ошибки, возвращаемые драйвером носителя и ретранслируемые драйвером мезонина, приведены в руководстве системного программиста на драйвер носителя мезонинов. Ошибки, возвращаемые библиотекой VISA и ретранслируемые драйвером, приведены в описании библиотеки VISA.

| Элемент перечисления | Описание |
| --- | --- |
| VI\_ERROR\_PARAMETER<N> | Ошибка параметра N |
| UNMMKO1\_WARNING\_BAD\_INITILIZATION | Ошибка во время инициализации |
| UNMMKO1\_WARNING\_ALREADY\_STARTED | Устройство уже запущено |
| UNMMKO1\_WARNING\_ALREADY\_STOPPED | Устройство уже остановлено |
| UNMMKO1\_ERROR\_UNKNOWN | Неизвестная ошибка |
| UNMMKO1\_ERROR\_NULL\_POINTER | Указатель NULL передан в качестве параметра |
| UNMMKO1\_ERROR\_NOT\_ENOUGH\_MEMORY | Недостаточно памяти |
| UNMMKO1\_ERROR\_INTERNAL | Внутренняя ошибка |
| UNMMKO1\_ERROR\_SELFTEST | Ошибка самоконтроля |
| UNMMKO1\_ERROR\_WAS\_CONFIGURED\_IN\_  A\_DIFFERENT\_MODE | Устройство было сконфигурировано в другом режиме |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_  WHILE\_RUNNING | Недопустимая операция при запущенном устройстве |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_  WHILE\_STOPPED | Недопустимая операция при остановленном устройстве |
| UNMMKO1\_ERROR\_RT\_ADDRESS\_WAS\_NOT\_  CONFIGURED | Оконечное устройство не настроено |
| UNMMKO1\_ERROR\_CANNOT\_UPDATE\_DATA\_  WITH\_QUEUE\_MODE | Недопустимо обновление данных в режиме очереди данных |
| UNMMKO1\_ERROR\_ILLEGAL\_OPERATION\_IN\_  SCHEDULE\_MODE | Недопустимая операция в режиме работы по расписанию |
| UNMMKO1\_ERROR\_EXCEEDED\_SCHEDULE\_  SIZE\_LIMIT | Превышен предел размера расписания |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в докум. | № докум. | Входящий № сопроводи-тельного докум. и дата | Подп. | Дата |
| изменен-ных | заменен-ных | новых | аннулиро-ванных |
| 1 | - | Все | 24-52 | - | 52 | ФТКС.282-2020 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |