RF62X-SDK Documentation

Выпуск 2.0.0

Vladislav Kuzmin

сент. 15, 2021

Подготовка к работе


RF62X SDK
A software development kit for
work with scanners series RF62X

ГЛАВА 1

Подготовка к работе

1.1 Введение

1.1.1 Общее описание

RF62X-SDK - набор средств разработки, который позволяет специалистам создавать собственное программное обеспечение для работы с лазерными сканерами серии **RF62X** (*RF627 v20.x.x.x, RF627 v2.x.x, RF629 v2.x.x*) производства компании РИФТЭК.

1.1.2 Архитектура библиотеки

RF62X-SDK состоит из двух частей:

- **RF62X-Core** - основная библиотека («Ядро») с базовым набором функций и типов для работы с лазерными сканерами серии RF62X. Библиотека написана на языке программирования СИ в соответствии со стандартом C99 (ISO/IEC 9899:1999) и является кросс-платформенной. Для использования данной библиотеки необходима реализация платформозависимых функций (работа с памятью, работа с сетью, функции ввода/вывода).
- **RF62X-Wrappers** - библиотеки-«обёртки», в которых уже реализованы платформозависимые функции «Ядра» для конкретной платформы. Использование библиотек-«обёрток» упрощает процесс разработки приложений на следующих языках программирования: C, C++, C#, Python, LabVIEW, MatLab.

1.1.3 Принципы использования

Разработчики, которые при создании собственных приложений планируют использовать RF62X-SDK в виде **статических или динамических программных библиотек**, могут скачать готовый архив библиотек необходимой версии или собрать библиотеку RF62X-SDK самостоятельно из исходников (см.подробнее: *компиляция библиотек RF62X-SDK из исходников*).

Разработчики, которые предпочитают при создании собственных приложений использовать библиотеку RF62X-SDK в виде файлов исходного кода, должны скачать проект библиотеки RF62X-SDK (инструкции по скачиванию исходников см. *Скачивание проекта*) и включить необходимые файлы библиотеки в свой проект.

Примечание: Более подробно о использовании библиотеки RF62X-SDK в качестве файлов исходного кода или в качестве статической/динамической программной библиотеки при создании собственных приложений смотрите в разделе: *Создание проекта*

1.1.4 Основной функционал

Методы работы со канером:
- Поиск сканеров серии RF62X (*RF627 v20.x.x.x, RF627 v2.x.x, RF629 v2.x.x*).
- Получение результатов измерений.
- Получение кадров видео с матрицы сканера
- Получение/установка параметров.
- Запись/скачивание дампа профилей.
- Программный запуск/остановка измерений.
- Запись/сохранение калибровочной таблицы.
- Перезагрузка/сброс сканера/матрицы.
- Отправка/получение данных при работе с периферией.

Поддерживаемые протоколы информационного обмена со сканерами:
- RF627-Protocol (UDP). Актуально для сканеров серии RF627 v20.x.x.x
- RF62X-Protocol (UDP). Актуально для сканеров серии RF627/RF629 v2.x.x

1.1.5 Что нового

- Запись/скачивание дампа профилей.
- Программный запуск/остановка измерений.
- Перезагрузка/сброс сканера/матрицы.
- Отправка/получение данных при работе с периферией.

1.2 Целевые платформы и совместимость

1.2.1 Языки программирования

Основная программная библиотека RF62X-Core («Ядро») написана на языке СИ стандарта C99 (ISO/IEC 9899:1999) без использования сторонних программных модулей и зависимых от операционной системы или процессора функций.

1.2.2 Целевые платформы

Достигнута совместимость с любыми операционными системами семейства Windows и Linux, поддерживающими компиляторы языка СИ стандарта C99 (ISO/IEC 9899:1999). Библиотека компилируется из исходных кодов и может быть использована с любыми типами процессоров (x86, ARM, RISC-V и др.).

1.2.3 Поддерживаемые компиляторы

- GCC 5.x или новее в Linux
- XCode 8.0 или новее в OS X
- MSVC2017 или новее в Windows

1.2.4 Ссылки

Этот проект использует git для управления исходным кодом и GitHub для размещения исходного кода.

- Репозиторий проекта: https://github.com/RIFTEK-LLC/RF62X-SDK
- Документация: RF62X-SDK.ru.pdf, RF62X-SDK.en.pdf
- Форум: https://bustezeu.beget.tech/index.php

1.3 Предустановка и настройка системы

1.3.1 Установка программного обеспечения

Есть несколько вариантов построения библиотеки RF62X-SDK. Все варианты поддерживаются и должны работать одинаково корректно для:

- IDE Visual Studio 2017 и выше
- IDE Qt Creator 4.11.0 и выше
- CMake 3.13 и выше

Примечание: Если вы знакомы с CMake, то вы также можете самостоятельно создавать(открыть) проект библиотеки для любой поддерживающей CMake среды.

Если возникли сложности с установкой или настройкой сред разработки, ниже приведены более подробные инструкции:

- IDE Visual Studio 2019 (дополнительная информация доступна на оффициальном сайте docs.microsoft.com)
- IDE Qt Creator (дополнительная информация доступна на оффициальном сайте qt.io)
- CMake (дополнительная информация доступна на оффициальном сайте cmake.org)

1.4 Скачивание проекта

1.4.1 Git-клиент

Для разработчиков, которые хотят скачать библиотеку из исходников с помощью Git-клиента, следует выполнить следующие инструкции:

- **Установите git-клиент на свой локальный компьютер (если ещё не установлен)**
  - В Linux используйте команду терминала: `sudo apt install git`
  - На MacOS используйте команду терминала: `brew install git`
  - Для других платформ смотрите документацию по установке git-клиента.
- **Откройте командную строку/терминал на вашем компьютере**
  - В Linux щелкните панель запуска и найдите *terminal*
  - В OS X нажмите command-space и найдите *terminal*
  - В Windows нажмите меню «Пуск» и найдите «командную строку» - *cmd*.
- **Клонируйте репозиторий с помощью следующих команд:**

```
git clone https://github.com/RIFTEK-LLC/RF62X-SDK.git
cd RF62X-SDK
git submodule update --init --recursive
```

1.4.2 Git в Qt Creator

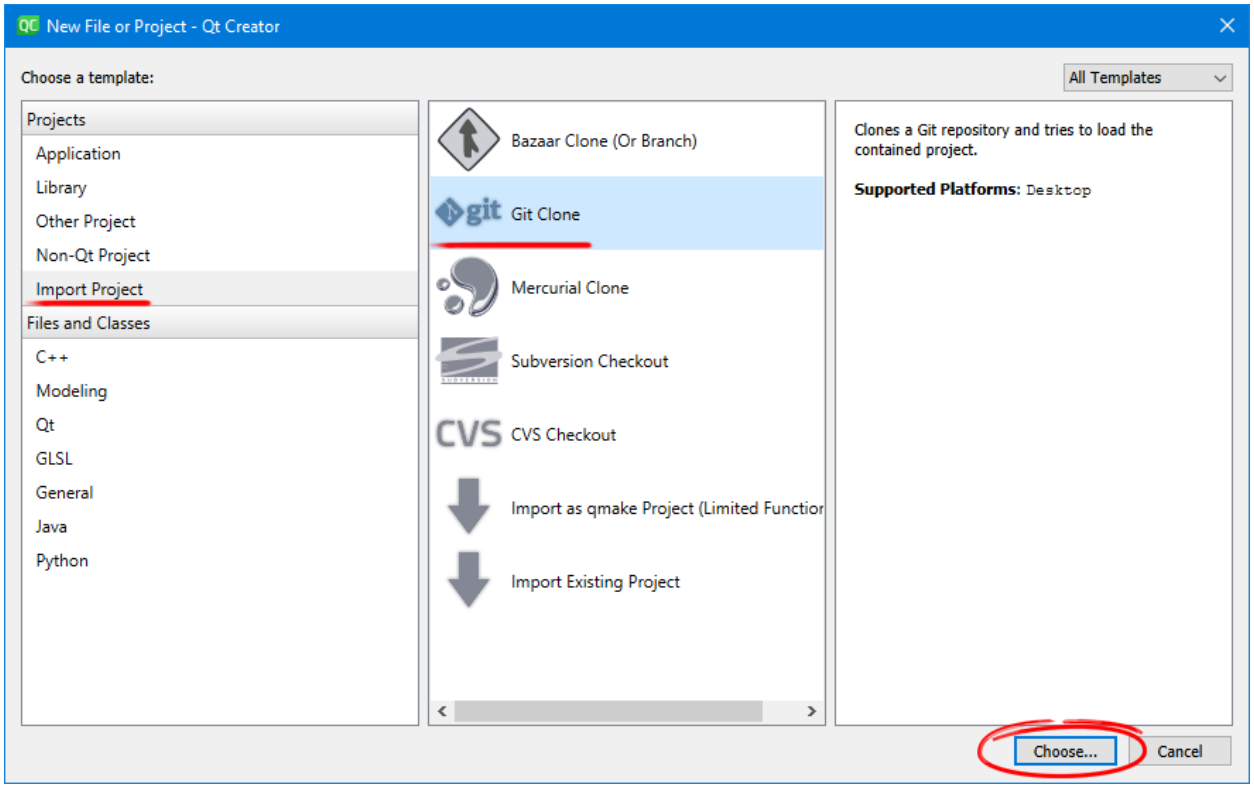
Для разработчиков, которые хотят скачать и собрать библиотеку из исходников с помощью Git, встроенного в IDE Qt Creator, следует выполнить следующие инструкции:

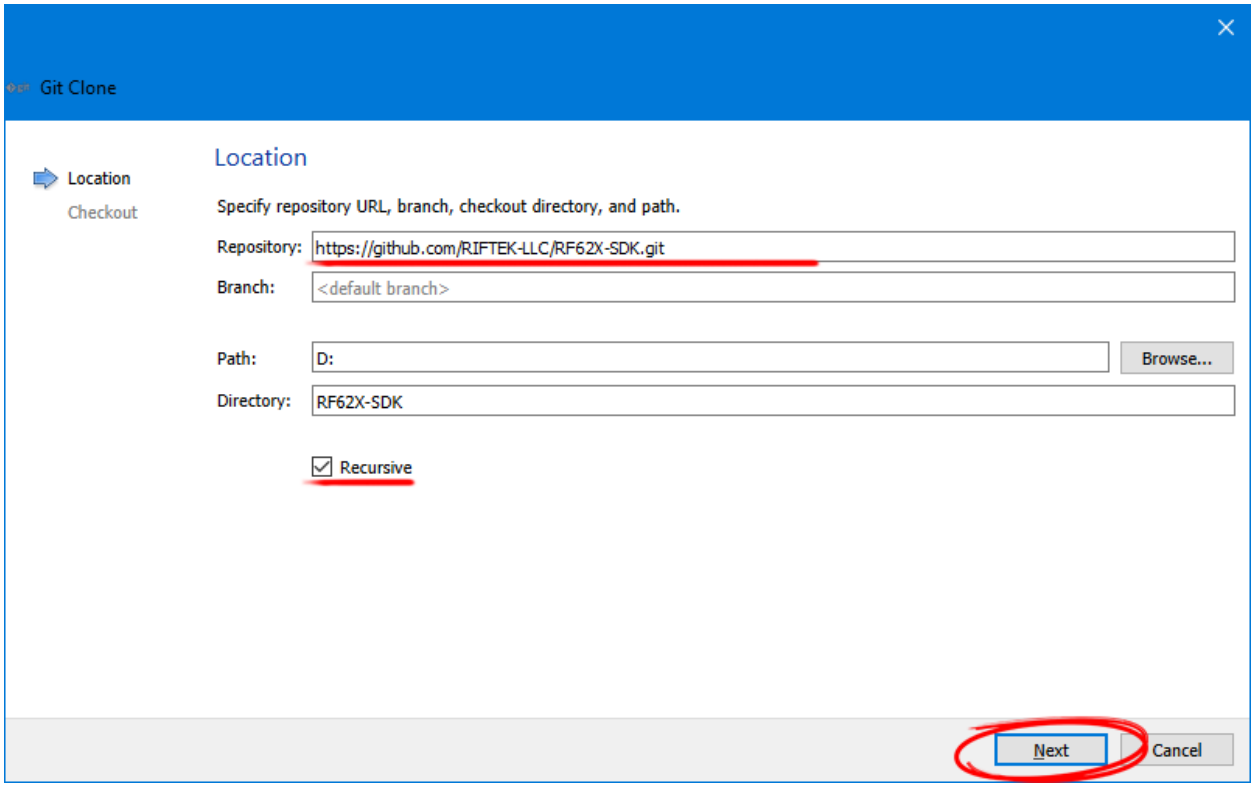
1. Нажмите **File > New File or Project**
2. Выберите опцию **Import Project > Git Clone**, как показано ниже.
3. Введите url-адрес SDK `https://github.com/RIFTEK-LLC/RF62X-SDK.git`, выберите опцию **«Recursive»**, а затем нажмите **Next**.
4. После загрузки откройте файл CMakeLists.txt необходимого вам проекта через **File > Open File or Project**, выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
5. Запустите компиляцию проекта

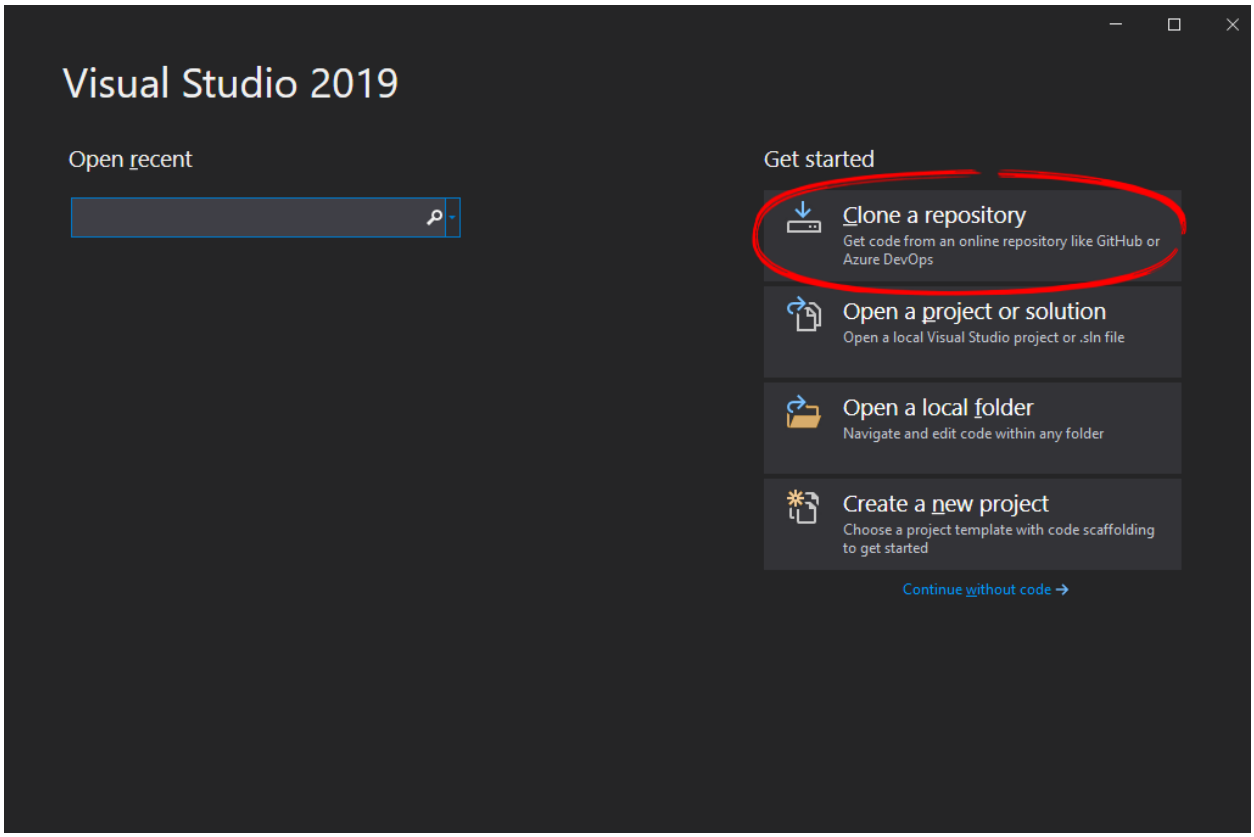
1.4.3 Git в Visual Studio

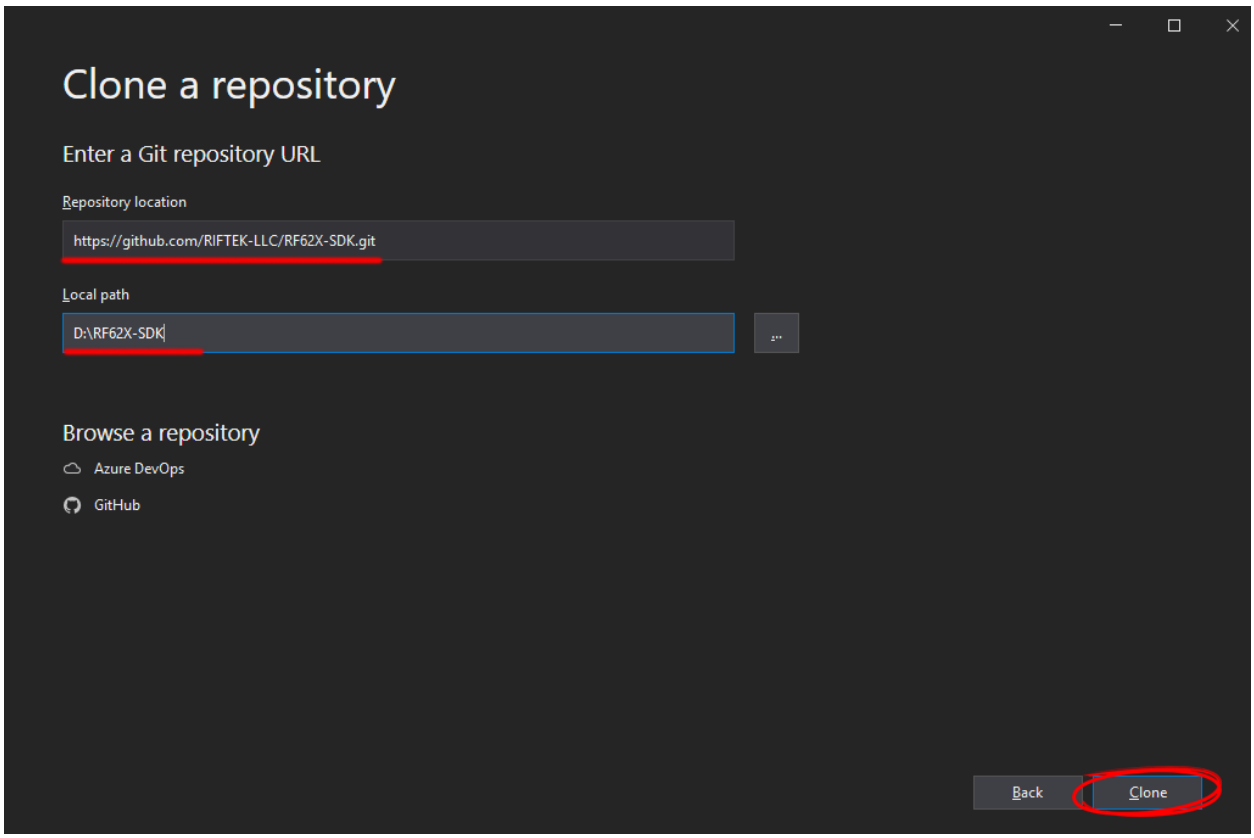
Для разработчиков, которые хотят скачать и собрать библиотеку из исходников с помощью Git, встроенного в IDE Visual Studio, следует выполнить следующие инструкции:

1. Откройте Visual Studio 2019.
2. В стартовом окне выберите **Клонирование или извлечение кода**.
3. Введите url-адрес SDK `https://github.com/RIFTEK-LLC/RF62X-SDK.git`, выберите или введите местоположение хранилища, а затем нажмите **Клонировать**.


New File or Project - Qt Creator
Choose a template:
All Templates
Projects
Application
Library
Other Project
Non-Qt Project
Import Project
Files and Classes
C++
Modeling
Qt
GLSL
General
Java
Python
Bazaar Clone (Or Branch)
Git Clone
Mercurial Clone
Subversion Checkout
CVS Checkout
Import as qmake Project (Limited Functior
Import Existing Project
Clones a Git repository and tries to load the contained project.
Supported Platforms: Desktop
Choose...
Cancel


Git Clone
Location
Checkout
Location
Specify repository URL, branch, checkout directory, and path.
Repository: https://github.com/RIFTEK-LLC/RF62X-SDK.git
Branch: <default branch>
Path: D:
Browse...
Directory: RF62X-SDK
Recursive
Next
Cancel


Visual Studio 2019
Open recent
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code


Clone a repository
Enter a Git repository URL
Repository location
https://github.com/RIFTEK-LLC/RF62X-SDK.git
Local path
D:\RF62X-SDK
Browse a repository
Azure DevOps
GitHub
Back
Clone

4. После чего Visual Studio загрузит проект из удаленного репозитория и откроет его.
5. Выберите один из необходимых вам проектов и запустите его сборку.

Примечание: Перед использованием Git-клиента в Visual Studio необходимо убедиться в установке дополнительного компонента C++ CMake tools для Windows.

1.5 Запуск примеров

Для быстрого старта, а также проверки базового функционала SDK (поиск устройств, получение профилей/кадров, установка/получение параметров, скачивание дампа и др.), разработчику предоставляется возможность открыть один из имеющихся примеров проектов на одном из доступных языков программирования (C/C++, C#, Python) и запустить его выполнение.

1.5.1 Запуск примеров на C/C++

Qt Creator

1. Откройте файл `CMakeLists.txt` из папки `RF62X-SDK/Examples/Cpp` через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
2. Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**.
3. Скомпилируйте и запустите один из примеров

Visual Studio:

1. В папке с примерами `RF62X-SDK/Examples/Cpp` откройте «командную строку» или «терминал» и введите следующую команду для создания Visual Studio проекта:

```
mkdir build && cd build
cmake ..
```

2. Откройте созданный проект `RF62XSDK-EXAMPLES.sln` в папке `build`
3. Скомпилируйте и запустите один из примеров

1.5.2 Запуск примеров на C#

1. Используя Visual Studio откройте один из имеющихся проектов (*RF627_smart*, *RF627_old*, *RF62X_WinForms*) в папке `RF62X-SDK/Examples/CSharp`.
2. Укажите целевую платформу **x64 Debug** или **x64 Release**
3. Скомпилируйте один из примеров
4. Перед запуском скачайте архив библиотек для C# (смотреть последние выпуски RF62X-SDK библиотек) и скопируйте из архива в папке `Dependencies` все файлы с именем **libRF62X-SDK** в папку к исполняемому файлу проекта (`../bin/x64/Debug/` или `../bin/x64/Release/`)

5. Запустите пример

1.5.3 Запуск примеров на Python

1. Используя Visual Studio Code (или иное IDE с поддержкой Python) откройте один из имеющихся проектов (*RF627_SMART*, *RF627_OLD*) в папке `RF62X-SDK/Examples/Python`.
2. Скачайте архив библиотек для Python (смотреть последние выпуски RF62X-SDK библиотек) и скопируйте из архива в папке `Dependencies` все файлы с именем **libRF62X-SDK** в папку к исполняемому файлу проекта.
3. Запустите пример

ГЛАВА 2

Создание проекта

В данном разделе приведены примеры создания проектов на языках C++, C# и Python с использованием проекта RF62X-SDK в качестве файлов исходного кода и в качестве статической или динамической программной библиотеки.

2.1 Создание проекта C/C++

2.1.1 Qt Creator + CMake + RF62X-SDK в качестве библиотеки

Для создания нового CMake проекта в Qt Creator с использованием динамической/статической линковки SDK необходимо выполнить следующий порядок действий:

- В Qt Creator создайте новый проект открыв в меню **File > New File or Project**, укажите **Qt Console Application** и нажмите кнопку **Choose**
- Введите имя проекта в поле **Project Name** и укажите путь к папке с проектом в поле **Project Location**, после ввода нажмите кнопку **Next**
- Выберите **CMake** в качестве системы постояние проекта и нажмите кнопку **Next** дважды
- Выберите компилятор (*MinGW*, *MSVC*, *Clang*), нажмите кнопку **Next** и завершите настройку проекта
- Скачайте архив библиотек для C/C++ (смотреть последние выпуски RF62X-SDK библиотек)
- Измените файл `CMakeLists.txt` вашего проекта в соответствии с приведенным ниже примером:

```
cmake_minimum_required(VERSION 3.14)

##############################################################################
## EXECUTABLE-PROJECT
```

(continues on next page)

(продолжение с предыдущей страницы)

```
## name and version
#############################################################################
project(RF62X_Search_Example LANGUAGES CXX)

#############################################################################
## SETTINGS
## basic project settings before use
#############################################################################
set(CMAKE_INCLUDE_CURRENT_DIR ON)
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
# creating output directory architecture in accordance with GNU guidelines
set(BINARY_DIR "${CMAKE_BINARY_DIR}")
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${BINARY_DIR}/bin")
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${BINARY_DIR}/lib")

#############################################################################
## TARGET
## create target and add include path
#############################################################################
# create glob files for *.h, *.cpp
file (GLOB H_FILES   ${CMAKE_CURRENT_SOURCE_DIR}/*.h)
file (GLOB CPP_FILES ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp)
# concatenate the results (glob files) to variable
set  (SOURCES ${CPP_FILES} ${H_FILES})
# create executable from src
add_executable(${PROJECT_NAME} ${SOURCES})

#############################################################################
## FIND PACKEGE AND LINK LIBRARIES
## linking all dependencies
#############################################################################
SET(RF62X_SDK_LIBRARY_TYPE "STATIC")
if (MSVC)
   find_package(RF62X-SDK PATHS "../RF62X-SDK_cpp/MSVC2019_64bit/CMake")
elseif(MINGW)
   find_package(RF62X-SDK PATHS "../RF62X-SDK_cpp/MinGW_64bit/CMake")
else()
   find_package(RF62X-SDK PATHS "../RF62X-SDK_cpp/GCC_64bit/CMake")
endif()
target_link_directories(${PROJECT_NAME} PUBLIC ${RF62X_SDK_LIBRARY_DIRS})
target_link_libraries(${PROJECT_NAME} PUBLIC ${RF62X_SDK_LIBRARIES})
target_include_directories(${PROJECT_NAME} PUBLIC ${RF62X_SDK_INCLUDE_DIRS})
```

Примечание: Для работы с динамической библиотекой RF62X-SDK необходимо установить параметр *RF62X_SDK_LIBRARY_TYPE* в значение *SHARED* (`SET(RF62X_SDK_LIBRARY_TYPE "SHARED")`) и скопировать библиотеку RF62X-SDK.dll(*.so) в каталог к исполняемому файлу проекта (PROJECT_BINARY_DIR)

- Измените файл `main.cpp` вашего проекта в соответствии с приведенным ниже примером:

```
#include <string>
#include <iostream>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;

int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#         Search Example v2.x.x         #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627-Smart" << std::endl;
   std::cout << "======================================="     << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<hello_info> info = list[i]->get_info();

      std::cout << "\n\nID scanner's list: " << i               << std::endl;
      std::cout << "---------------------------------------"  << std::endl;
      std::cout << "Device information: "                      << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()      << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()    << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()       << std::endl;
      std::cout << "* MAC Addr\t: " << info->mac_address()      << std::endl;

      std::cout << "\nWorking ranges: "                         << std::endl;
      std::cout << "* Zsmr, mm\t: " << info->z_smr()            << std::endl;
      std::cout << "* Zmr , mm\t: " << info->z_mr()             << std::endl;
      std::cout << "* Xsmr, mm\t: " << info->x_smr()            << std::endl;
      std::cout << "* Xemr, mm\t: " << info->x_emr()            << std::endl;

      std::cout << "\nVersions: "                               << std::endl;
      std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
      std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
      std::cout << "---------------------------------------"  << std::endl;
   }

   // Cleanup resources allocated with sdk_init()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    sdk_cleanup();
}
```

- Выберите тип сборки **Debug** или **Release** и запустите построение проекта.

2.1.2 Qt Creator + CMake + RF62X-SDK в качестве файлов исходного кода

Для создания нового CMake проекта в Qt Creator с использованием файлов исходного кода SDK необходимо выполнить следующий порядок действий:

- В Qt Creator создайте новый проект открыв в меню **File > New File or Project**, укажите **Qt Console Application** и нажмите кнопку **Choose**
- Введите имя проекта в поле **Project Name** и укажите путь к папке с проектом в поле **Project Location**, после ввода нажмите кнопку **Next**
- Выберите **CMake** в качестве системы постояние проекта и нажмите кнопку **Next** дважды
- Выберите компилятор (*MinGW*, *MSVC*, *Clang*), нажмите кнопку **Next** и завершите настройку проекта
- Скачайте **RF62X-SDK** проект (для получения дополнительной информации о шагах загрузки проекта см. *Скачивание проекта*)
- Измените файл `CMakeLists.txt` вашего проекта в соответствии с приведенным ниже примером и установите в параметре `RF62XSDK_DIR` на путь к папке `RF62X-Wrappers/Cpp` в соответствии с расположением скачанного проекта **RF62X-SDK**

```
cmake_minimum_required(VERSION 3.14)

################################################################################
## EXECUTABLE-PROJECT
## name and version
################################################################################
project(RF62X_Search_Example LANGUAGES CXX)

################################################################################
## SETTINGS
## basic project settings before use
################################################################################
set(CMAKE_INCLUDE_CURRENT_DIR ON)
set(CMAKE_CXX_STANDARD 11)
set(CMAKE_CXX_STANDARD_REQUIRED ON)
# creating output directory architecture in accordance with GNU guidelines
set(BINARY_DIR "${CMAKE_BINARY_DIR}")
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${BINARY_DIR}/bin")
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${BINARY_DIR}/lib")

################################################################################
## TARGET
## create target and add include path
################################################################################
# create glob files for *.h, *.cpp
file (GLOB H_FILES   ${CMAKE_CURRENT_SOURCE_DIR}/*.h)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
file (GLOB CPP_FILES ${CMAKE_CURRENT_SOURCE_DIR}/*.cpp)
# concatenate the results (glob files) to variable
set  (SOURCES ${CPP_FILES} ${H_FILES})
# create executable from src
add_executable(${PROJECT_NAME} ${SOURCES})

#############################################################################
## INCLUDING SUBDIRECTORIES AND LINK LIBRARIES
## linking all dependencies
#############################################################################
# set RF62XSDK path variable
set(RF62XSDK_DIR "../RF62X-Wrappers/Cpp")
# add subdirectory of RF62X-SDK lib
add_subdirectory(${RF62XSDK_DIR} RF62X-SDK)
target_link_libraries(${PROJECT_NAME} RF62X-SDK)
```

- Измените файл `main.cpp` вашего проекта в соответствии с приведенным ниже примером:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;

int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#         Search Example v2.x.x         #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627-Smart" << std::endl;
   std::cout << "======================================="     << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<hello_info> info = list[i]->get_info();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        std::cout << "\n\nID scanner's list: " << i                << std::endl;
        std::cout << "----------------------------------------"  << std::endl;
        std::cout << "Device information: "                       << std::endl;
        std::cout << "* Name  \t: "   << info->device_name()      << std::endl;
        std::cout << "* Serial\t: "   << info->serial_number()    << std::endl;
        std::cout << "* IP Addr\t: "  << info->ip_address()       << std::endl;
        std::cout << "* MAC Addr\t: " << info->mac_address()      << std::endl;

        std::cout << "\nWorking ranges: "                          << std::endl;
        std::cout << "* Zsmr, mm\t: " << info->z_smr()            << std::endl;
        std::cout << "* Zmr , mm\t: " << info->z_mr()             << std::endl;
        std::cout << "* Xsmr, mm\t: " << info->x_smr()            << std::endl;
        std::cout << "* Xemr, mm\t: " << info->x_emr()            << std::endl;

        std::cout << "\nVersions: "                                << std::endl;
        std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
        std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
        std::cout << "----------------------------------------"  << std::endl;
    }

    // Cleanup resources allocated with sdk_init()
    sdk_cleanup();
}
```

- Выберите тип сборки **Debug** или **Release** и запустите построение проекта.

2.1.3 Visual Studio + RF62X-SDK в качестве библиотеки

Для создания нового проекта в Visual Studio с использованием динамической/статической линковки SDK необходимо выполнить следующий порядок действий:

- Откройте Visual Studio и выберите **Create a new project**, затем выберите **Empty Project** и нажмите кнопку **Next**
- Введите имя проекта в поле **Project Name** и укажите путь к папке с проектом в поле **Project Location**, после ввода нажмите кнопку **Next**
- Скачайте архив библиотек для C/C++ (смотреть последние выпуски RF62X-SDK библиотек)
- Добавьте файл `main.cpp` в проект и измените его, как показано ниже:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;

int main()
{
    std::cout << "########################################"  << std::endl;
    std::cout << "#                                      #"  << std::endl;
    std::cout << "#         Search Example v2.x.x         #"  << std::endl;
    std::cout << "#                                      #"  << std::endl;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    std::cout << "########################################\n"<< std::endl;

    // Initialize sdk library
    sdk_init();

    // Print return rf62X sdk version
    std::cout << "SDK version: " << sdk_version()                << std::endl;
    std::cout << "========================================"      << std::endl;


    // Create value for scanners vector's type
    std::vector<std::shared_ptr<rf627smart>> list;
    // Search for rf627smart devices over network
    list = rf627smart::search(500);

    // Print count of discovered rf627smart in network by Service Protocol
    std::cout << "Was found\t: " << list.size()<< " RF627-Smart" << std::endl;
    std::cout << "========================================"      << std::endl;


    for (size_t i = 0; i < list.size(); i++)
    {
        std::shared_ptr<hello_info> info = list[i]->get_info();

        std::cout << "\n\nID scanner's list: " << i               << std::endl;
        std::cout << "----------------------------------------"  << std::endl;
        std::cout << "Device information: "                      << std::endl;
        std::cout << "* Name  \t: "   << info->device_name()      << std::endl;
        std::cout << "* Serial\t: "   << info->serial_number()    << std::endl;
        std::cout << "* IP Addr\t: "  << info->ip_address()       << std::endl;
        std::cout << "* MAC Addr\t: " << info->mac_address()      << std::endl;

        std::cout << "\nWorking ranges: "                         << std::endl;
        std::cout << "* Zsmr, mm\t: " << info->z_smr()            << std::endl;
        std::cout << "* Zmr , mm\t: " << info->z_mr()             << std::endl;
        std::cout << "* Xsmr, mm\t: " << info->x_smr()            << std::endl;
        std::cout << "* Xemr, mm\t: " << info->x_emr()            << std::endl;

        std::cout << "\nVersions: "                               << std::endl;
        std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
        std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
        std::cout << "----------------------------------------"  << std::endl;
    }

    // Cleanup resources allocated with sdk_init()
    sdk_cleanup();
}
```

- Выберите тип (*Debug* или *Release*) и разрядность (*x64* или *x86*) целевой платформы.
- Скопируйте файлы из скаченного архива в папке `include` в каталог проекта.
- Откройте **Project > Properties**, выберите **Configuration Properties > VC++ Directories** и добавьте пути к загруженным файлам заголовков и библиотекам в **Include Directories** и **Library Directories** соответственно.
- Скомпилируйте проект.
- Скопируйте библиотеки из скаченного архива в каталог к исполняемому файлу проекта

(`../bin/x64/Debug/` или `../bin/x64/Release/`).

- Запустите проект.

2.2 Создание проекта C#

2.2.1 Visual Studio + RF62X-SDK в качестве библиотеки

Для создания нового проекта в Visual Studio с использованием динамической библиотеки SDK необходимо выполнить следующий порядок действий:

- Откройте Visual Studio и выберите **Create a new project**, затем выберите **Empty Project** и нажмите кнопку **Next**
- Введите имя проекта в поле **Project Name** (например RF627_search) и укажите путь к папке с проектом в поле **Project Location**, после ввода нажмите кнопку **Next**
- Измените файл `Program.cs` проекта, как показано ниже:

```
using System;
using System.Collections.Generic;
using SDK.SCANNERS;

namespace RF627_search
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("########################################");
         Console.WriteLine("#                                      #");
         Console.WriteLine("#         Search Example v2.x.x         #");
         Console.WriteLine("#                                      #");
         Console.WriteLine("########################################");

         // Initialize sdk library
         RF62X.SdkInit();

         // Print return rf62X sdk version
         Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
         Console.WriteLine("========================================");

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         // Print count of discovered rf627smart in network
         Console.WriteLine("Was found\t: {0} RF627-Smart", list.Count);
         Console.WriteLine("========================================");

         for (int i = 0; i < list.Count; i++)
         {
             RF62X.HelloInfo info = list[i].GetInfo();

             Console.WriteLine("\n\nID scanner's list: {0}", i);
             Console.WriteLine("----------------------------------------");
             Console.WriteLine("Device information: ");
             Console.WriteLine("* Name\t: {0}", info.device_name);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            Console.WriteLine("* Serial\t: {0}", info.serial_number);
            Console.WriteLine("* IP Addr\t: {0}", info.ip_address);
            Console.WriteLine("* MAC Addr\t: {0}", info.mac_address);

            Console.WriteLine("\nWorking ranges: ");
            Console.WriteLine("* Zsmr, mm\t: {0}", info.z_smr);
            Console.WriteLine("* Zmr , mm\t: {0}", info.z_mr);
            Console.WriteLine("* Xsmr, mm\t: {0}", info.x_smr);
            Console.WriteLine("* Xemr, mm\t: {0}", info.x_emr);

            Console.WriteLine("\nVersions: ");
            Console.WriteLine("* Firmware\t: {0}", info.firmware_version);
            Console.WriteLine("* Hardware\t: {0}", info.hardware_version);
            Console.WriteLine("----------------------------------------");
        }

        // Cleanup resources allocated with SdkInit()
        RF62X.SdkCleanup();
    }
  }
}
```

- Выберите тип (*Debug* или *Release*) и разрядность (*x64* или *x86*) целевой платформы.
- Скачайте архив библиотек для C# (смотреть последние выпуски RF62X-SDK библиотек)
- Откройте **Project > Add References**, нажмите кнопку **Browse...** и подключите библиотеку **RF62X-SDK.dll** из скачанного архива в проект.
- Скомпилируйте проект.
- Скопируйте библиотеки из скаченного архива в папке `Dependencies` в каталог к исполняемому файлу проекта (`../bin/x64/Debug/` или `../bin/x64/Release/`).
- Запустите проект.

2.3 Создание проекта Python

2.3.1 Visual Studio Code + RF62X-SDK в качестве библиотеки

Для создания нового Python проекта в Visual Studio Code с использованием динамической библиотеки СДК необходимо выполнить следующий порядок действий:

- Создайте каталог проекта и откройте его в Visual Studio Code
- Добавьте новый py-файл (например, demo.py) в этот каталог проекта
- Скачайте архив библиотек для Python (смотреть последние выпуски RF62X-SDK библиотек)
- Разархивируйте файлы скачанного архива в каталог с проектом
- Измените py-файл проекта (например, demo.py) в соответствии с примером ниже:

```
from PYSDK_SMART import *


if __name__ == '__main__':

print("#########################################")
print("#                                       #")
print("#          Search Example v2.x.x        #")
print("#                                       #")
print("#########################################")

# Initialize sdk library
sdk_init()

# Print return rf627 sdk version
print('SDK version', sdk_version())
print("=========================================")

# Create value for scanners vector's type
list_scanners=search(500)
print("Was found\t:", len(list_scanners), "RF627-Smart")
print("=========================================")


# Iterate over all available network adapters in the current operating
# system to send "Hello" requests.
i=0
for scanner in list_scanners:
   info = get_info(scanner, kSERVICE)
   i+=1
   #  Print short information about the scanner
   print("\n\nID scanner's list:", i)
   print("-----------------------------------------")
   print("Device information:")
   print("* Name\t\t: ",   info['user_general_deviceName'])
   print("* Serial\t: ",   info['fact_general_serial'])
   print("* IP Addr\t: ",  info['user_network_ip'])
   print("* MAC Addr\t: ",  info['fact_network_macAddr'])

   print("\nWorking ranges: ")
   print("* Zsmr, mm\t: ", info["fact_general_smr"])
   print("* Zmr, mm\t: ", info["fact_general_mr"])
   print("* Xsmr, mm\t: ", info["fact_general_xsmr"])
   print("* Xemr, mm\t: ", info["fact_general_xemr"])

   print("\nVersions: " )
   print("* Firmware\t: ", info["firmware_version"])
   print("* Hardware\t: ", info["hardware_version"])
   print("-----------------------------------------")

# Cleanup resources allocated with sdk_init()
sdk_cleanup()
```

- Запустите проект.

ГЛАВА 3

Компиляция SDK

3.1 Компиляция «ядра»

3.1.1 Как скомпилировать

RF62X-Core может быть скомпилирован при помощи консоли или среды разработки (Visual Studio, Qt Creator).

Во-первых, вы должны загрузить проект (если не сделали этого ранее).

Примечание: Для получения дополнительной информации о шагах загрузки проекта см. *Скачивание проекта*

CMake

Находясь в папке проекта **RF62X-SDK** для построения RF62X-Core введите следующую команду в консоль (терминал):

```
cd RF62X-Core
mkdir build && cd build
cmake ..
cmake --build .
```

Qt Creator

Для построения **RF62X-Core** с использованием **IDE Qt Creator**:

- Загрузите файл `CMakeLists.txt` из папки **RF62X-Core** через **File > Open File or Project** (выберите файл `CMakeLists.txt`)
- Выберите компилятор (*MinGW*, *MSVC*, *Clang*) и нажмите **Configure Project**

- Скомпилируйте проект

Visual Studio

Находясь в папке проекта **RF62X-SDK** для построения **RF62X-Core** введите следующую команду в консоль (терминал):

```
cd RF62X-Core
mkdir build && cd build
cmake ..
```

- Откройте полученное решение RF62X-Core.sln в Visual Studio
- Скомпилируйте проект

3.1.2 Как использовать

Для использования библиотеки **RF62X-Core** вместо имеющихся библиотек-«обёрток» разработчику необходимо будет самостоятельно реализовать платформозависимую часть «ядра».

Обзор платформозависимых функций

В **RF62X-Core** платформозависимые функции (работа с памятью, работа с сетью, функции ввода/вывода) представлены в виде указателей на функции.

Указатели на платформозависимые функции объявлены в файлах, `memory_platform.h`, `network_platform.h` и `iostream_platform.h`.

calloc_t

Прототип: *typedef void* (*calloc_t)(rfSize num, rfSize size);*

Описание: *Указатель на функцию* `calloc_t` *выделяет блок памяти для массива размером* `num` *элементов, каждый из которых занимает* `size` *байт. В результате выделяется блок памяти размером* `number * size` *байт, причём весь блок заполнен нулями.*

Параметры:

- `num` - *Количество элементов массива, под который выделяется память.*
- `size` - *Размер одного элемента в байтах.*

Возвращаемое значение: *Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда* `void*` *, поэтому это тип данных может быть приведен к желаемому типу. Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief Allocates an array in memory with elements initialized to 0.
 *
 * @param num Number of elements to allocate.
 * @param size Size of each element.
 *
 * @return
 * - On success: returns a pointer to the allocated space.
 * - On error: NULL
 */
typedef void* (*calloc_t)(rfSize num, rfSize size);
```

malloc_t

Прототип: *typedef void* (*malloc_t)(rfSize size);*

Описание: *Указатель на функцию* `malloc_t` *выделяет блок памяти размером* `size` *байт и возвращает указатель на начало блока. Содержание выделенного блока памяти не инициализируется, оно остается с неопределенными значениями.*

Параметры:

- `size` *- Размер выделяемого блока памяти в байтах.*

Возвращаемое значение: *Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда* `void*` *, поэтому это тип данных может быть приведен к желаемому типу. Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief malloc_t - ptr to function whish allocates memory block
 * Allocates a block of size bytes of memory, returning a pointer
 * to the beginning of the block.
 *
 * @param size Size of the memory block, in bytes.
 *
 * @return
 * - On success: returns a pointer to the allocated space.
 * - On error: NULL.
 */
typedef void* (*malloc_t)(rfSize size);
```

realloc_t

Прототип: *typedef void* (*realloc_t)(void *ptr, rfSize newsize);*

Описание: *Указатель на функцию* `realloc_t` *выполняет перераспределение блоков памяти. Размер блока памяти, на который ссылается параметр* `ptr` *изменяется на* `newsize` *байтов. Блок памяти может уменьшаться или увеличиваться в размере.*

Эта функция может перемещать блок памяти на новое место, в этом случае функция возвращает указатель на новое место в памяти. Содержание блока памяти сохраняется даже если новый блок имеет меньший размер, чем старый. Отбрасываются только те данные, которые не вместились в новый блок. Если новое значение `newsize` *больше старого, то содержимое вновь выделенной памяти будет неопределенным.*

В случае, если `ptr` *равен* `NULL` *, функция ведет себя именно так, как функция* `malloc_t` *, т. е. выделяет память и возвращает указатель на этот участок памяти.*

В случае, если `newsize` *равен 0, ранее выделенная память будет освобождена, как если бы была вызвана функция* `free_t` *, и возвращается нулевой указатель.*

Параметры:

- `ptr` - *Указатель на блок ранее выделенной памяти функциями* `malloc_t` *,* `calloc_t` *или* `realloc_t` *для перемещения в новое место. Если этот параметр —* `NULL` *, просто выделяется новый блок, и функция возвращает на него указатель.*
- `newsize` - *Новый размер, в байтах, выделяемого блока памяти. Если* `newsize` *равно 0, ранее выделенная память освобождается и функция возвращает нулевой указатель,* `ptr` *устанавливается в 0.*

Возвращаемое значение: *Указатель на перераспределенный блок памяти, который может быть либо таким же, как аргумент* `ptr` *или ссылаться на новое место.

Тип данных возвращаемого значения всегда `void*` *, который может быть приведен к любому другому.*

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель, и блок памяти, на который указывает аргумент `ptr` *остается неизменным.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief realloc_t - ptr to function whish reallocates memory block
 * Changes the size of the memory block pointed to by ptr. The function
 * may move the memory block to a new location (whose address is
 * returned by the function).
 *
 * @param ptr Pointer to a memory block previously allocated.
 * @param newsize New size for the memory block, in bytes.
 *
 * @return A pointer to the reallocated memory block, which may be either
 * the same as ptr or a new location.
 */
typedef void* (*realloc_t)(void *ptr, rfSize newsize);
```

free_t

Прототип: *typedef void (*free_t)(void* data);*

Описание: *Указатель на функцию* `free_t` *освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова* `malloc_t` , `calloc_t` *или* `realloc_t` *освобождается.*

Обратите внимание, что эта функция оставляет значение `data` *неизменным, следовательно, он по-прежнему указывает на тот же блок памяти, а не на нулевой указатель.*

Параметры:

- `data` - *Указатель на блок памяти, ранее выделенный функциями* `malloc_t` , `calloc_t` *или* `realloc_t` , *которую необходимо высвободить. Если в качестве аргумента передается нулевой указатель, никаких действий не происходит.*

Возвращаемое значение: *Функция не имеет возвращаемое значение.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief Deallocates or frees a memory block.
 *
 * @param data Previously allocated memory block to be freed.
 */
typedef void (*free_t)(void* data);
```

memset_t

Прототип: *typedef void* (*memset_t)(void* memptr, rfInt val, rfSize num);*

Описание: *Указатель на функцию* `memset_t` *заполняет* `num` *байтов блока памяти, через указатель* `memptr` . *Код заполняемого символа передаётся в функцию через параметр* `val` .

Параметры:

- `memptr` - *Указатель на блок памяти для заполнения.*
- `val` - *Передается целое число, но функция заполняет блок памяти символом, преобразуя это число в символ*
- `num` - *Количество байт, которые необходимо заполнить указанным символом.*

Возвращаемое значение: *Функция возвращает указатель на блок памяти.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief memset_t - ptr to function whish fills block of memory
 * Sets the first num bytes of the block of memory pointed by ptr to the
 * specified value (interpreted as an unsigned rfChar).
 *
 * @param memptr Pointer to the block of memory to fill.
 * @param val Value to be set.
 * @param num Number of bytes to be set to the value.
 * rfSize is an unsigned rfIntegral type.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 *
 * @return ptr is returned.
 */
typedef void* (*memset_t)(void* memptr, rfInt val, rfSize num);
```

memcpy_t

Прототип: *typedef void* (*memcpy_t)(void* destination, const void* source, rfSize num);*

Описание: *Указатель на функцию* `memset_t` *копирует* `num` *байтов первого блока памяти, на который ссылается указатель* `source` *, во второй блок памяти, на который ссылается указатель* `destination` *.*

Параметры:

- `destination` *- Указатель на блок памяти назначения (куда будут копироваться байты данных).*
- `source` *- Указатель на блок памяти источник (т. е., откуда будут копироваться байты данных).*
- `num` *- Количество копируемых байтов.*

Возвращаемое значение: *Указатель на блок памяти назначения.*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief memcpy_t - ptr to function whish copies block of memory
 * Copies the values of num bytes from the location pointed to by source
 * directly to the memory block pointed to by destination.
 *
 * @param destination Pointer to the destination array where the content is to
 * be copied, type-casted to a pointer of type void*.
 * @param source Pointer to the source of data to be copied, type-casted to a
 * pointer of type const void*.
 * @param num Number of bytes to copy. rfSize is an unsigned rfIntegral type.
 *
 * @return destination is returned.
 */
typedef void* (*memcpy_t)(void* destination, const void* source, rfSize num);
```

memcmp_t

Прототип: *typedef rfInt (*memcmp_t)(const void* ptr1, const void* ptr2, rfSize num);*

Описание: *Указатель на функцию* `memcmp_t` *сравнивает первые* `num` *байтов блока памяти указателя* `ptr1` *с первыми* `num` *байтами блока памяти* `ptr2` *. Возвращаемое значение 0 если блоки равны, и значение отличное от 0, если блоки не равны.*

Параметры:

- `ptr1` *- Указатель на первый блок памяти.*

- `ptr2` - *Указатель на второй блок памяти.*
- `num` - *Количество байтов для сравнения.*

Возвращаемое значение: *Возвращает значение, информирующее о результате сравнения содержимого блоков памяти. Нулевое значение указывает, что содержимое обоих блоков памяти равны. Значение больше нуля говорит о том, что первый блок памяти —* `ptr1` *больше, чем блок памяти —* `ptr2` *, и значение меньше нуля свидетельствует об обратном*

Пример в коде:

```
/** @file memory_platform.h */

/**
 * @brief memcmp_t - ptr to function whish compare two blocks of memory
 * Compares the first num bytes of the block of memory pointed by ptr1 to the
 * first num bytes pointed by ptr2, returning zero if they all match or a
 * value different from zero representing which is greater if they do not.
 *
 * @param ptr1 Pointer to block of memory.
 * @param ptr2 Pointer to block of memory.
 * @param num Number of bytes to compare.
 *
 * @return
 * 0 - if the contents of both memory blocks are equal,
 * <0 - if the first byte that does not match in both memory blocks has a
 * lower value in ptr1 than in ptr2.
 * >0 - if the first byte that does not match in both memory blocks has a
 * greater value in ptr1 than in ptr2.
 */
typedef rfInt (*memcmp_t)(const void* ptr1, const void* ptr2, rfSize num);
```

hton_long_t, ntoh_long_t, hton_short_t, ntoh_short_t

Прототип:

- *typedef rfUint32 (*hton_long_t) (rfUint32 hostlong);*
- *typedef rfUint32 (*ntoh_long_t) (rfUint32 netlong);*
- *typedef rfUint16 (*hton_short_t)(rfUint16 hostshort);*
- *typedef rfUint16 (*ntoh_short_t)(rfUint16 netshort);*

Описание: *Указатели на функции* `hton_long_t` , `ntoh_long_t` , `hton_short_t` , `ntoh_short_t` *необходимы для преобразования многобайтовых целочисленных типов из байтового порядка хоста в сетевой порядок байтов и наоборот.*

Параметры:

- `hostlong/hostshort` - *32/16-битное число в байтовом порядке хоста.*
- `netlong/netshort` - *32/16-битное число в сетевом порядке байтов.*

Возвращаемое значение: *Функция возвращает значение в сетевом/обратном порядке байтов.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief The hton_long_t function converts a u_long from host to network byte
 * order (which is big-endian).
 *
 * @param hostlong A 32-bit number in host byte order.
 *
 * @return The function returns the value in network byte order.
 */
typedef rfUint32 (*hton_long_t) (rfUint32 hostlong);

/**
 * @brief The ntoh_long_t function converts a u_long from network order to
 * host byte order (which is little-endian on rfIntel processors).
 *
 * @param netlong A 32-bit number in network byte order.
 *
 * @return: The function returns the value supplied in the netlong parameter
 * with the byte order reversed.
 */
typedef rfUint32 (*ntoh_long_t) (rfUint32 netlong);

/**
 * @brief The hton_short_t function converts a u_short from host to network
 * byte order (which is big-endian).
 *
 * @param hostlong A 16-bit number in host byte order.
 *
 * @return The modbusHtoN_short_t function returns the value in network
 * byte order.
 */
typedef rfUint16 (*hton_short_t)(rfUint16 hostshort);

/**
 * @brief The ntoh_short_t function converts a u_short from network byte order
 * to host byte order
 *
 * @param netshort A 16-bit number in network byte order.
 *
 * @return The function returns the value in host byte order.
 */
typedef rfUint16 (*ntoh_short_t)(rfUint16 netshort);
```

create_udp_socket_t

Прототип: *typedef void* (*create_udp_socket_t)();*

Описание: *Указатель на функцию* `create_udp_socket_t` *создает несвязанный UDP сокет*

Возвращаемое значение: *После успешного завершения* `create_udp_socket_t` *должен вернуть указатель на дескриптор сокета. В противном случае должно быть возвращено значение* `NULL` *и вызвана ошибка создания сокета.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to UDP socket creation function
 *
 * @param af The address family specification.
 * @param type The type specification for the new socket.
 * @param protocol The protocol to be used.
 *
 * @return
 * - On success: A descriptor referencing the new socket
 * - On error: NULL
 */
typedef void* (*create_udp_socket_t)();
```

set_broadcast_socket_option_t, set_reuseaddr_socket_option_t, set_socket_recv_timeout_t

Прототип:

- *typedef rfInt8 (*set_broadcast_socket_option_t)(void* socket);*
- *typedef rfInt8 (*set_reuseaddr_socket_option_t)(void* socket);*
- *typedef rfInt8 (*set_socket_recv_timeout_t)(void* socket, rfInt32 msec);*

Описание: *Указатели на функции* `set_broadcast_socket_option_t` , `set_reuseaddr_socket_option_t` , `set_socket_recv_timeout_t` , *необходимы для включение в UDP сокетах таких сетевых настроек как: broadcast (позволяет отправлять или получать широковещательные пакеты), reuseaddr (позволяет сокету принудительно связываться с портом, используемым другим сокетом), recv_timeout (время, в течение которого сокет ожидает, пока данные станут доступными для чтения).*

Параметры:

- `socket` *- Указатель дескриптора сокета*
- `msec` *(только для set_socket_recv_timeout_t) - Время ожидания в миллисекундах.*

Возвращаемое значение: *После успешного завершения везвращается 0. В противном случае должно быть возвращено значение -1.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that sets a broadcast socket option.
 *
 * @param socket A descriptor that identifies a socket.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*set_broadcast_socket_option_t)(void* socket);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/**
 * @brief Pointer to the function that sets a reuseaddr socket option.
 *
 * @param socket A descriptor that identifies a socket.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*set_reuseaddr_socket_option_t)(void* socket);

/**
 * @brief Pointer to the function that sets a timeout for socket receive.
 *
 * @param socket A descriptor that identifies a socket.
 * @param msec The timeout in millisec.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*set_socket_recv_timeout_t)(void* socket, rfInt32 msec);
```

set_socket_option_t

Прототип: *typedef rfInt8 (*set_socket_option_t)(void* socket, rfInt32 level, rfInt32 optname, const rfChar* optval, rfInt32 optlen);*

Описание: *Указатель на функцию* `set_socket_option_t` *устанавливает параметр сокета.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `level` - *Уровень, на котором определена опция (например, SOL_SOCKET).*
- `optname` - *Параметр сокета, для которого должно быть установлено значение (например, SO_BROADCAST)*
- `optval` - *Указатель на буфер, в котором указано значение запрошенной опции.*
- `optlen` - *Размер в байтах буфера, на который указывает параметр* `optval`

Возвращаемое значение: *Если ошибок не происходит,* `set_socket_option_t` *возвращает ноль. В противном случае возвращается значение* `RF_SOCKET_ERROR`

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that sets a socket option.
 *
 * @param socket A descriptor that identifies a socket.
 * @param level The level at which the option is defined.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @param optname The socket option for which the value is to be set.
 * @param optval A pointer to the buffer in which the value for the requested
 * option is specified.
 * @param optlen The size, in bytes, of the buffer pointed to by the optval
 * parameter.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*set_socket_option_t)(
        void* socket, rfInt32 level, rfInt32 optname,
        const rfChar* optval, rfInt32 optlen);
```

socket_connect_t

Прототип: *typedef rfInt8 (*socket_connect_t)(void* socket, rfUint32 dst_ip_addr, rfUint16 dst_port);*

Описание: *Указатель на функцию* `socket_connect_t` *устанавливает соединение с указанным сокетом.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `dst_ip_addr` - *IP-адрес назначения, с которым должно быть установлено соединение.*
- `dst_port` - *Порт назначения, к которому должно быть установлено соединение.*

Возвращаемое значение: *Если ошибок не происходит,* `socket_connect_t` *возвращает ноль. В противном случае возвращается значение* `RF_SOCKET_ERROR`

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that establishes a connection to a
 * specified socket
 *
 * @param socket A descriptor identifying an unconnected socket.
 * @param dst_ip_addr Destination IP Addr to which the connection should be
 * established.
 * @param dst_port Destination port to which the connection should be
 * established.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*socket_connect_t)(
        void* socket, rfUint32 dst_ip_addr, rfUint16 dst_port);
```

socket_bind_t

Прототип: *typedef rfInt (*socket_bind_t)(void* socket, rfUint32 host_ip_addr, rfUint16 host_port);*

Описание: *Указатель на функцию* `socket_bind_t` *связывает локальный адрес с сокетом.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `dst_ip_addr` - *IP-адрес, с которым должен быть связан сокет.*
- `dst_port` - *Порт, с которым должен быть связан сокет.*

Возвращаемое значение: *Если ошибок не происходит,* `socket_bind_t` *возвращает ноль. В противном случае возвращается значение* `RF_SOCKET_ERROR`

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that associates a local address with
 * a socket.
 *
 * @param socket A descriptor identifying an unconnected socket.
 * @param host_ip_addr Host IP Addr to which the connection should be bind.
 * @param host_port Host port to which the connection should be bind.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt (*socket_bind_t)(
        void* socket, rfUint32 host_ip_addr, rfUint16 host_port);
```

socket_listen_t

Прототип: *typedef rfInt8 (*socket_listen_t)(void* socket, rfInt32 backlog);*

Описание: *Указатель на функцию* `socket_listen_t` *переводит сокет в состояние, в котором он ожидает входящее соединения.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `backlog` - *Максимальная длина очереди ожидающих подключений.*

Возвращаемое значение: *Если ошибок не происходит,* `socket_listen_t` *возвращает ноль. В противном случае возвращается значение* `RF_SOCKET_ERROR`

Пример в коде:

```
/** @file network_platform.h */

/** @brief Pointer to the function that places a socket in a state in which
 * it is listening for an incoming connection.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 *
 * @param socket A descriptor identifying a bound, unconnected socket.
 * @param backlog The maximum length of the queue of pending connections.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*socket_listen_t)(void* socket, rfInt32 backlog);
```

socket_accept_t

Прототип: *typedef void* (*socket_accept_t)(void* socket, rfUint32* srs_ip_addr, rfUint16* srs_port);*

Описание: *Указатель на функцию* `socket_accept_t` *разрешает попытку входящего подключения к сокету.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `srs_ip_addr` - *Указатель на IP-адрес входящего соединения.*
- `srs_port` - *Указатель на порт входящего соединения.*

Возвращаемое значение: *Если ошибок не происходит,* `socket_accept_t` *возвращает указатель на дескриптор принятого сокета. В противном случае возвращается нулевой указатель.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that permits an incoming connection attempt
 * on a socket.
 *
 * @param socket A descriptor that identifies a socket that has been placed in
 * a listening state with the modbusSocketListen_t function.
 * The connection is actually made with the socket that is returned by accept.
 * @param srs_ip_addr Pointer to the IP address of the incoming connection.
 * @param srs_port Pointer to the port of the incoming connection.
 *
 * @return
 * - On success: value is a handle for the socket
 * - On error : NULL
 */
typedef void* (*socket_accept_t)(
        void* socket, rfUint32* srs_ip_addr, rfUint16* srs_port);
```

close_socket_t

Прототип: *typedef rfInt8 (*close_socket_t)(void* socket);*

Описание: *Указатель на функцию* `close_socket_t` *закрывает существующий сокет.*

Параметры:

- `socket` - *Указатель дескриптора сокета*

Возвращаемое значение: *Если ошибок не происходит,* `close_socket_t` *возвращает ноль. В противном случае возвращается значение* `RF_SOCKET_ERROR`

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that closes an existing socket.
 *
 * @param socket A descriptor identifying the socket to close.
 *
 * @return
 * - On success: 0
 * - On error: -1
 */
typedef rfInt8 (*close_socket_t)(void* socket);
```

send_tcp_data_t

Прототип: *typedef rfInt (*send_tcp_data_t)(void* socket, const void *buf, rfSize len);*

Описание: *Указатель на функцию* `send_tcp_data_t` *отправляет данные в подключенный TCP сокет.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `buf` - *Указатель на буфер, содержащий данные для передачи.*
- `len` - *Длина в байтах данных в буфере, на который указывает параметр* `buf`

Возвращаемое значение: *Если ошибок не происходит,* `send_tcp_data_t` *возвращает общее количество отправленных байтов, которое может быть меньше количества, запрошенного для отправки в параметре* `len` *. В противном случае возвращается значение -1.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the send function that sends data on a TCP
 * connected socket.
 *
 * @param socket A descriptor identifying a connected socket.
 * @param buf A pointer to a buffer containing the data to be transmitted.
 * @param len The length, in bytes, of the data in buffer pointed to by the
 * buf parameter.
 *
 * @return
 * - On success: the total number of bytes sent, which can be less than the
 * number requested to be sent in the len parameter.
 * - On error: -1
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 */
typedef rfInt (*send_tcp_data_t)(void* socket, const void *buf, rfSize len);
```

send_udp_data_t

Прототип: *typedef rfInt (*send_udp_data_t)(void* socket, const void *data, rfSize len, rfUint32 dest_ip_addr, rfUint16 dest_port);*

Описание: *Указатель на функцию* `send_udp_data_t` *отправляет данные по UDP в определенное место назначения.*

Параметры:

- `socket` *- Указатель дескриптора сокета*
- `data` *- Указатель на буфер, содержащий данные для передачи.*
- `len` *- Длина в байтах данных в буфере, на который указывает параметр* `data`
- `dest_ip_addr` *- IP-адрес, на который данные должны быть отправлены.*
- `dest_port` *- Порт, на который данные должны быть отправлены.*

Возвращаемое значение: *Если ошибок не происходит,* `send_udp_data_t` *возвращает общее количество отправленных байтов, которое может быть меньше количества, запрошенного для отправки в параметре* `len` *. В противном случае возвращается значение -1.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the send function that sends data on a UDP socket
 *
 * @param socket A descriptor identifying a socket.
 * @param buf A pointer to a buffer containing the message to be sent.
 * @param len The size of the message in bytes.
 * @param dest_addr Points to a sockaddr_in structure containing the
 * destination address.
 * @param addrlen Specifies the length of the sockaddr_in structure pointed
 * to by the dest_addr argument.
 *
 * @return
 * - On success: the total number of bytes sent, which can be less than
 * the number requested to be sent in the len parameter
 * - On error: -1
 */
typedef rfInt (*send_udp_data_t)(
        void* socket, const void *data, rfSize len,
        rfUint32 dest_ip_addr, rfUint16 dest_port);
```

recv_data_from_t

Прототип: *typedef rfInt (*recv_data_from_t)(void* socket, void *buf, rfSize len, rfUint32* srs_ip_addr, rfUint16* srs_port);*

Описание: *Указатель на функцию* `recv_data_from_t` *получает данные из сокета и адрес отправителя.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `buf` - *Указатель на буфер для приема входящих данных*
- `len` - *Длина в байтах буфера, на который указывает параметр* `buf`
- `srs_ip_addr` - *Указатель на IP-адрес из которого были получены данные*
- `srs_port` - *Указатель на порт из которого были получены данные*

Возвращаемое значение: *Если ошибок не происходит,* `recv_data_from_t` *возвращает общее количество принятых байтов. В противном случае возвращается значение -1.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that receive message from socket and capture
 * address of sender.
 *
 * @param socket Specifies a socket descriptor from which data should
 * be received.
 * @param buf Specifies the buffer in which to place the message.
 * @param len Specifies the length of the buffer area.
 * @param srs_ip_addr Pointer to the IP address from which the data
 * was received.
 * @param srs_port Pointer to the port from which the data was received.
 *
 * @return
 * - On success: the number of bytes received
 * - On error: -1
 */
typedef rfInt (*recv_data_from_t)(
        void* socket, void *buf, rfSize len,
        rfUint32* srs_ip_addr, rfUint16* srs_port);
```

recv_data_t

Прототип: *typedef rfInt (*recv_data_t)(void* socket, void *buf, rfSize len);*

Описание: *Указатель на функцию* `recv_data_t` *получает данные от подключенного сокета или привязанного сокета без установления соединения.*

Параметры:

- `socket` - *Указатель дескриптора сокета*
- `buf` - *Указатель на буфер для приема входящих данных*
- `len` - *Длина в байтах буфера, на который указывает параметр* `buf`

Возвращаемое значение: *Если ошибок не происходит,* `recv_data_t` *возвращает общее количество принятых байтов. В противном случае возвращается значение -1.*

Пример в коде:

```
/** @file network_platform.h */

/**
 * @brief Pointer to the function that receive message from socket and capture
 * address of sender.
 *
 * @param sockfd Specifies a socket descriptor from which data
 * should be received.
 * @param buf Specifies the buffer in which to place the message.
 * @param len Specifies the length of the buffer area.
 *
 * @return
 * - On success: the number of bytes received
 * - On error: -1
 */
typedef rfInt (*recv_data_t)(void* socket, void *buf, rfSize len);
```

trace_info_t, trace_warning_t, trace_error_t

Прототип:

- *typedef rfInt(*trace_info_t)(const rfChar* msg, …);*
- *typedef rfInt(*trace_warning_t)(const rfChar* msg, …);*
- *typedef rfInt(*trace_error_t)(const rfChar* msg, …);*

Описание: *Указатели на функции* `trace_info_t` , `trace_warning_t` *и* `trace_error_t` *, необходимы для вывода как информационных сообщений, так и сообщений о предупреждениях и ошибках.*

Параметры:

- `msg` *- Указатель на строку, содержащую текст для вывода*
- `...` *(дополнительные аргументы) - В зависимости от формата строки, функция может ожидать последовательность дополнительных аргументов.*

Возвращаемое значение: *В случае успеха возвращается общее количество написанных символов*

Пример в коде:

```
/** @file iostream_platform.h */

/**
 * @brief Method for outputting debugging information
 *
 * @param msg Pointer to a string containing the text to be output
 * @param ...(additional arguments) Depending on the format string,
 * the function may expect a sequence of additional arguments
 *
 * @return On success, the total number of characters written is returned.
 */
typedef rfInt(*trace_info_t)(const rfChar* msg, ...);

/**
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @brief Method for outputting alert information
 *
 * @param msg Pointer to a string containing the text to be output
 * @param ...(additional arguments) Depending on the format string,
 * the function may expect a sequence of additional arguments
 *
 * @return On success, the total number of characters written is returned.
 */
typedef rfInt(*trace_warning_t)(const rfChar* msg, ...);

/**
 * @brief Method for outputting error information
 *
 * @param msg Pointer to a string containing the text to be output
 * @param ...(additional arguments) Depending on the format string,
 * the function may expect a sequence of additional arguments
 *
 * @return On success, the total number of characters written is returned.
 */
typedef rfInt(*trace_error_t)(const rfChar* msg, ...);
```

Запуск «ядра»

После реализации всех платформозависимых функций разработчику необходимо проинициализировать следующие структуры `iostream_platform_dependent_methods_t`, `memory_platform_dependent_methods_t` и `network_platform_dependent_methods_t`

Пример в коде:

```
/** @file iostream_platform.h */

/**
 * @brief Structure with user-provided iostream platform-specific methods
 */
typedef struct
{
   trace_info_t trace_info;
   trace_warning_t trace_warning;
   trace_error_t trace_error;
}iostream_platform_dependent_methods_t;
extern iostream_platform_dependent_methods_t iostream_platform;


/** @file memory_platform.h */

/**
 * @brief Structure with user-provided memory platform-specific methods
 */
typedef struct
{
   calloc_t rf_calloc;
   malloc_t rf_malloc;
   realloc_t rf_realloc;
   free_t rf_free;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
   memset_t rf_memset;
   memcpy_t rf_memcpy;
   memcmp_t rf_memcmp;

}memory_platform_dependent_methods_t;
extern memory_platform_dependent_methods_t memory_platform;


   /** @file memory_platform.h */

/**
 * @brief Structure with user-provided network platform-specific methods
 */
typedef struct
{
   hton_long_t hton_long;
   ntoh_long_t ntoh_long;
   hton_short_t hton_short;
   ntoh_short_t ntoh_short;

   create_udp_socket_t create_udp_socket;
   set_broadcast_socket_option_t set_broadcast_socket_option;
   set_reuseaddr_socket_option_t set_reuseaddr_socket_option;
   set_socket_option_t set_socket_option;
   set_socket_recv_timeout_t set_socket_recv_timeout;
   socket_connect_t socket_connect;
   socket_bind_t socket_bind;
   socket_listen_t socket_listen;
   socket_accept_t  socket_accept;
   close_socket_t close_socket;

   send_tcp_data_t send_tcp_data;
   send_udp_data_t send_udp_data;

   recv_data_from_t recv_data_from;
   recv_data_t recv_data;
}network_platform_dependent_methods_t;

typedef struct
{
   rfUint32 host_ip_addr;
   rfUint32 host_mask;
}network_platform_dependent_settings_t;

typedef struct
{
   network_platform_dependent_methods_t network_methods;
   network_platform_dependent_settings_t network_settings;
}network_platform_t;
extern network_platform_t network_platform;
```

Инициализация данных структур производится путем присваивания указателей на реализованные платформозависимые функции, а адреса проинициализированных экземпляров структур передаются в метод `init_platform_dependent_methods` для инициализации кросс-платформенной части «ядра».

Пример в коде:

```
/** @file rf62X_core.h */

/**
 * @brief Init platform dependent methods and settings
 *
 * @param memory_methods Structure with platform-specific methods
 * for work with memory
 * @param iostream_methods Structure with platform-specific methods
 * for work with iostream
 * @param network_methods Structure with platform-specific methods
 * for work with network
 * @param adapter_settings Structure with adapter settings
 */
API_EXPORT void init_platform_dependent_methods(
         memory_platform_dependent_methods_t* memory_methods,
         iostream_platform_dependent_methods_t* iostream_methods,
         network_platform_dependent_methods_t* network_methods,
         network_platform_dependent_settings_t* adapter_settings);
```

3.2 Компиляция «обёртки» на C

Данная библиотека позволяет упростить разработку приложений на языке СИ.

Для её использования в проектах СИ разработчик должен включить необходимые h-файлы библиотеки в свой проект и собрать статическую или динамическую программную библиотеку.

3.2.1 Как скомпилировать

Библиотека-«обёртка» может быть скомпилирован при помощи консоли или среды разработки (Visual Studio, Qt Creator).

Для начала необходимо скачать проект SDK.

Примечание: Для получения дополнительной информации о шагах загрузки проекта см. *Скачивание проекта*

CMake

Находясь в папке проекта **RF62X-SDK**, для построения библиотеки-«обёртки» на СИ введите следующую команду в консоль (терминал):

```
mkdir build && cd build
cmake .. -D_WRAPPER_LANGUAGE:STRING=C -D_SUBMODULE_CACHE_OVERWRITE=OFF
cmake --build .
```

Qt Creator

Для построения библиотеки-«обёртки» с использованием **IDE Qt Creator**:

- Загрузите файл CMakeLists.txt из папки **RF62X-SDK** через **File > Open File or Project** (выберите файл CMakeLists.txt)
- Выберите компилятор (MinGW, MSVC, Clang) и нажмите **Configure Project**
- Измените параметр **WRAPPER_LANGUAGE** (язык компилируемой обертки) в файле CMakeLists.txt проекта на **C** (`SET(${PARENT}_WRAPPER_LANGUAGE "C" CACHE STRING "" ${REWRITE_FORCE})`)
- Скомпилируйте проект

Visual Studio

Находясь в папке проекта **RF62X-SDK**, для построения библиотеки-«обёртки» на СИ введите следующую команду в консоль (терминал):

```
mkdir build && cd build
cmake .. -D_WRAPPER_LANGUAGE:STRING=C -D_SUBMODULE_CACHE_OVERWRITE=OFF
```

- Откройте полученное решение RF62X-SDK.sln в Visual Studio
- Скомпилируйте проект

3.2.2 Как использовать

Вы можете **создать свой проект**, включив в него статическую или динамическую библиотеку и необходимые заголовочные файлы (см. *Создание проекта C/C++*), или **открыть и скомпилировать** один из имеющихся в СДК примеров (см. *Запуск примеров на C/C++*).

Примечание: Для получения дополнительной информации по каждой функции «обёртки» СИ см. *Интерфейс «обёртки» на C*

3.3 Компиляция «обёртки» на C++

Данная библиотека позволяет упростить разработку приложений на языке C++.

Для её использования в проектах C++ разработчик должен включить необходимые h-файлы библиотеки в свой проект и собрать статическую или динамическую программную библиотеку.

3.3.1 Как скомпилировать

Библиотека-«обёртка» RF62X-SDK может быть скомпилирован при помощи консоли или среды разработки (Visual Studio, Qt Creator).

Для начала необходимо скачать проект SDK.

Примечание: Для получения дополнительной информации о шагах загрузки проекта см. *Скачивание проекта*

CMake

Находясь в папке проекта **RF62X-SDK**, для построения библиотеки-«обёртки» на C++ введите следующую команду в консоль (терминал):

```
mkdir build && cd build
cmake .. -D_WRAPPER_LANGUAGE:STRING=C++ -D_SUBMODULE_CACHE_OVERWRITE=OFF
cmake --build .
```

Qt Creator

Для построения библиотеки-«обёртки» с использованием IDE Qt Creator:

- Загрузите файл CMakeLists.txt из папки **RF62X-SDK** через **File > Open File or Project** (выберите файл CMakeLists.txt)
- Выберите компилятор (MinGW, MSVC, Clang) и нажмите **Configure Project**
- Измените параметр **WRAPPER_LANGUAGE** (язык компилируемой обертки) в файле CMakeLists.txt проекта на **C++** (`SET(${PARENT}_WRAPPER_LANGUAGE "C++" CACHE STRING "" ${REWRITE_FORCE})`)
- Скомпилируйте проект

Visual Studio

Находясь в папке проекта **RF62X-SDK**, для построения библиотеки-«обёртки» на C++ введите следующую команду в консоль (терминал):

```
mkdir build && cd build
cmake .. -D_WRAPPER_LANGUAGE:STRING=C++ -D_SUBMODULE_CACHE_OVERWRITE=OFF
```

- Откройте полученное решение RF62X-SDK.sln в Visual Studio
- Скомпилируйте проект

3.3.2 Как использовать

Вы можете **создать свой проект**, включив в него статическую или динамическую библиотеку и необходимые заголовочные файлы (см. *Создание проекта C/C++*), или **открыть и скомпилировать** один из имеющихся в СДК примеров (см. *Запуск примеров на C/C++*).

Примечание: Для получения дополнительной информации по каждому методу «обёртки» C++ см. *Интерфейс «обёртки» на C++*

3.4 Компиляция «обёртки» на C#

Данная «обёртка» представляет собой библиотеку .NET, написанную на языке C#, которая может быть использована в приложении на любом языке программирования, поддерживающем платформу **.NET Framework** (C#, Visual Basic, C++/CLI и др.)

Для её использования в проектах .NET разработчик должен собрать или скачать динамическую программную библиотеку RF62X-SDK для .NET и добавить её в проект, а также скачать дополнительные зависимости в каталог к исполняемому файлу проекта.

Примечание: Архив библиотек для C# можно скачать по ссылке: последние выпуски RF62X-SDK библиотек

3.4.1 Как скомпилировать

Библиотека-«обёртка» может быть скомпилирована при помощи среды разработки Visual Studio.

Для начала необходимо скачать проект SDK.

Примечание: Для получения дополнительной информации о шагах загрузки проекта см. *Скачивание проекта*

Visual Studio

- Откройте в Visual Studio решение RF62X-SDK.sln (папка **RF62X-SDK/RF62X-Wrappers/CSharp**)
- Скомпилируйте проект

3.4.2 Как использовать

Вы можете **создать свой проект**, включив в него динамическую библиотеку-«обёртку» (см. *Создание проекта C#*), или **открыть и скомпилировать** один из имеющихся в СДК примеров (см. *Запуск примеров на C#*).

Примечание: Для получения дополнительной информации по каждому методу «обёртки» .NET см. *Интерфейс «обёртки» на C#*

ГЛАВА 4

Описания API

4.1 Интерфейс «обёртки» на C

Данная библиотека позволяет упростить разработку приложений на языке СИ.

Для её использования в проектах СИ разработчику необходимо включить h-файлы библиотеки в свой проект, а также добавить к проекту «обёртку» в качестве статической или динамической программной библиотеки.

Для скачивания библиотеки см. Последние выпуски RF62X-SDK библиотек.
Для компиляции библиотеки из исходников см. *Компиляция «обёртки» на СИ*.
Для использования в проектах см. *Создание проекта C/C++*.

4.1.1 Инициализация SDK

Файл `rf62Xcore.h` необходим для вызова функции инициализации SDK: `core_init()`

Файл `rf62X_sdk.h` является основным файлом программного интерфейса (API) для разработки программ на языке СИ и определяет функциональность библиотеки-«обёртки».

Файл `rf62X_types.h` содержит основные структуры и типы, используемые в SDK.

core_init

Прототип: *int8_t core_init();*

Описание: *Функция инициализации SDK. Должна быть вызвана один раз перед дальнейшими вызовами любых библиотечных функций*

Возвращаемое значение: *После успешного завершения возвращается 1 (TRUE). В противном случае должен быть возвращен код ошибки.*

Пример в коде:

```
/** @file rf62Xcore.h */

/**
 * @brief core_init - Initialize sdk library
 * @details Must be called once before further calls to any
 * library functions
 *
 * @return 1 if success or error code.
 */
int8_t core_init();

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{

   // Initialize sdk library
   uint8_t is_init = core_init();

   if (is_init == 1)
      printf("SDK version: %s\n", sdk_version());
   else
   {
      printf("SDK initialization error: %s\n", is_init);
      return -1;
   }

   // some code...

}
```

core_cleanup

Прототип: *void core_cleanup();*

Описание: *Функция высвобождает ресурсы, выделенные с помощью функции* `core_init`

Пример в коде:

```
/** @file rf62Xcore.h */

/**
 * @brief core_cleanup - Cleanup resources allocated
 * with core_init() function
 */
void core_cleanup();

-------------------------------------------------------------------------------
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Initialize sdk library
   core_init();

   // some code...

   // Cleanup resources
   core_cleanup();
}
```

sdk_version

Прототип: *char* sdk_version();*

Описание: *Функция получения информации о версии SDK*

Пример в коде:

```
/** @file rf62Xcore.h */

/**
 * @brief sdk_version - Return info about SDK version
 *
 * @return SDK version
 */
char* sdk_version();

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // Cleanup resources
    core_cleanup();
}
```

4.1.2 Интерфейс работы со сканерами серии RF627 v20.x.x.x

Файлы `rf62X_sdk.h`, `rf62X_types.h` и `rf62Xcore.h` предоставляют весь необходимый интерфейс для работы со сканерами серии RF627 v20.x.x.x

search_scanners

Прототип: *rfUint8 search_scanners(vector_t *list, scanner_types_t type, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция поиска устройств RF62X v20.x.x.x в сети*

Параметры:

- `list` - *Указатель на список, который будет заполнен найденными сканерами в сети.*
- `type` - *Тип сканера для поиска (kRF627_OLD, kRF627_SMART).*
- `timeout` - *Время поиска на каждом Ethernet интерфейсе (мс).*
- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief search_scanners - Search for RF62X devices over network
 *
 * @param[out] list Ptr to list of rf627 objects. If not null list will be
 * filled with found devices
 * @param[in] type Scanner's type (RF627-old, RF627-smart)
 * @param[in] timeout Time to search
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 search_scanners(
     vector_t *list, scanner_types_t type,
     rfUint32 timeout, protocol_types_t protocol);

------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{

   // Initialize sdk library
   core_init();

   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);


   // Iterate over all available network adapters in the current operating
   // system to send "Hello" requests.
   uint32_t count = 0;
   for (int i=0; i<GetAdaptersCount(); i++)
   {
      uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
      uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
      // call the function to change adapter settings inside the library.
      set_platform_adapter_settings(host_mask, host_ip_addr);

      // Search for rf627old devices over network by Service Protocol.
      if (host_ip_addr != 0)
      {
         // Get another IP Addr and set this changes in adapter settings.
         printf("Search scanners from:\n "
                "* IP Address\t: %s\n "
                "* Netmask\t: %s\n",
                GetAdapterAddress(i), GetAdapterMasks(i));
         search_scanners(scanners, kRF627_OLD, 300, kSERVICE);

         // Print count of discovered rf627old in network
         printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
         printf("----------------------------------------\n");
         count = (int)vector_count(scanners);
      }
   }

   // Print count of discovered rf627old in network
   printf("Was found\t: %d RF627 v20.x.x.x", (int)vector_count(scanners));

   // some code...
}
```

get_info_about_scanner

Прототип: *hello_information get_info_about_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция получения информации о сканере из пакета приветствия (Hello-пакет)*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому был получен пакет приветствия (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `hello_information` *в случае успеха, иначе ошибка*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_info_about_scanner - Get information about scanner from
 * hello packet
 *
 * @param[in] device Ptr to scanner
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return hello_information on success
 */
hello_information get_info_about_scanner(
      scanner_base_t *device, protocol_types_t protocol);

------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      hello_information result =
               get_info_about_scanner(vector_get(scanners,i), kSERVICE);

      rf627_smart_hello_info_by_service_protocol* info =
               result.rf627smart.hello_info_service_protocol;

      printf("\n\nID scanner's list: %d\n", i);
      printf("---------------------------------------\n");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        printf("Device information: \n");
        printf("* Name\t: %s\n", info->user_general_deviceName);
        printf("* Serial\t: %d\n", info->fact_general_serial);
        printf("* IP Addr\t: %s\n", info->user_network_ip);
        printf("* MAC Addr\t: %s\n", info->fact_network_macAddr);

        printf("\nWorking ranges: \n");
        printf("* Zsmr, mm\t: %d\n", info->fact_general_smr);
        printf("* Zmr , mm\t: %d\n", info->fact_general_mr);
        printf("* Xsmr, mm\t: %d\n", info->fact_general_xsmr);
        printf("* Xemr, mm\t: %d\n", info->fact_general_xemr);

        printf("\nVersions: \n");
        printf("* Firmware\t: %d.%d.%d\n",
               info->fact_general_firmwareVer[0],
               info->fact_general_firmwareVer[1],
               info->fact_general_firmwareVer[2]);
        printf("* Hardware\t: %d\n", info->fact_general_hardwareVer);
        printf("-----------------------------------------\n");
    }

    // some code...
}
```

free_scanner

Прототип: *void free_scanner(scanner_base_t *device);*

Описание: *Функция очистки памяти, используемой объектом типа* `scanner_base_t`

Параметры:

- `device` - *Указатель на сканер*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief free_scanner - Cleanup resources allocated by device
 *
 * @param[in] device Prt to scanner
 */
void free_scanner(scanner_base_t *device);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   while (vector_count(scanners) > 0)
   {
      uint32_t index = vector_count(scanners)-1;
      // Get last scanner in vector for delete
      scanner_base_t* device = vector_get(scanners, index);

      // Cleanup resources allocated by device
      free_scanner(device);

      // Delete from vector
      vector_delete(scanners, index);
   }
}
```

connect_to_scanner

Прототип: *rfUint8 connect_to_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция установки соединения со сканером*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому будет выполнено подключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief connect_to_scanner - Establish connection to the RF62X device
 *
 * @param[in] device Ptr to scanner
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 connect_to_scanner(
      scanner_base_t *device, protocol_types_t protocol);

--------------------------------------------------------------------------------
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // some actions with scanner...
   }
}
```

disconnect_from_scanner

Прототип: *rfUint8 disconnect_from_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция закрытия ранее установленного соединения со сканером*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому будет выполнено отключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief disconnect_from_scanner - Close connection to the device
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 *
 * @param[in] device Prt to scanner
 * @param[in] protocol Protocol's type (Service, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 disconnect_from_scanner(
      scanner_base_t *device, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // some actions with scanner...

      // Disconnect from scanner.
      disconnect_from_scanner(scanner, kSERVICE)
   }
}
```

check_connection_to_scanner

Прототип: *check_connection_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция проверки доступности сканера в сети (после подключения к нему)*

Параметры:

- `device` - *Указатель на сканер*
- `timeout` - *Время проверки соединения со сканером (мс).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief check_connection_to_scanner - Check connection to the RF62X device
 *
 * @param[in] device Ptr to scanner
 * @param[in] timeout Time to check connection
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 check_connection_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // Check connection to the RF627 device.
      uint8_t is_available =
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            check_connection_to_scanner(scanner, 300, kSERVICE);
      if (!is_available){
         printf("Scanner is not available now!");
         continue;
      }

      // some actions with scanner...
   }
}
```

get_profile2D_from_scanner

Прототип: *rf627_profile2D_t* get_profile2D_from_scanner(scanner_base_t *device, rfBool zero_points, rfBool realtime, protocol_types_t protocol);*

Описание: *Функция получения результатов измерений*

Параметры:

- `device` - *Указатель на сканер*
- `zero_points` - *Включать нулевые точки в возвращаемом профиле.*
- `realtime` - *Получение профиля в реальном времени (буферизация отключена).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Указатель на* `rf627_profile2D_t` *при успехе, иначе -* `NULL`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_profile2D_from_scanner - Get measurement from scanner's
 * data stream
 *
 * @param[in] device - ptr to scanner
 * @param[in] zero_points Enable zero points in return profile2D
 * @param[in] realtime Enable getting profile in realtime (buffering disabled)
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return ptr to rf627_profile_t structure
 */
rf627_profile2D_t* get_profile2D_from_scanner(
      scanner_base_t *device, rfBool zero_points,
      rfBool realtime, protocol_types_t protocol);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      uint8_t zero_points = TRUE;
      uint8_t realtime = TRUE;
      // Get profile from scanner's data stream by Service Protocol.
      rf627_profile2D_t* result = get_profile2D_from_scanner(
            scanner, zero_points, realtime, kSERVICE);
      rf627_smart_profile2D_t* profile2D = result->rf627old_profile2D;
      if (profile2D != NULL) {
         printf("Profile was successfully received!");
         // some actions with profile...
         free_profile2D(result);
      }else
         printf("Profile was not received!");
   }
}
```

free_profile2D

Прототип: *void free_profile2D(rf627_profile2D_t* profile);*

Описание: *Функция очистки ресурсов, выделенных для rf627_profile2D_t*

Параметры:

- `profile` - *Указатель на профиль*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief free_profile2D - Cleanup resources allocated for profile2D
 *
 * @param[in] profile Ptr to rf627_profile2D_t
 */
void free_profile2D(rf627_profile2D_t* profile);

--------------------------------------------------------------------------
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      uint8_t zero_points = TRUE;
      uint8_t realtime = TRUE;
      // Get profile from scanner's data stream by Service Protocol.
      rf627_profile2D_t* result = get_profile2D_from_scanner(
            scanner, zero_points, realtime, kSERVICE);
      rf627_smart_profile2D_t* profile2D = result->rf627old_profile2D;
      if (profile2D != NULL) {
         printf("Profile was successfully received!");
         // some actions with profile...
         free_profile2D(result);
      }else
         printf("Profile was not received!");
   }
}
```

read_params_from_scanner

Прототип: *rfUint8 read_params_from_scanner(scanner_base_t *device, uint32_t timeout, protocol_types_t protocol);*

Описание: *Функция получения текущих параметров сканера. При вызове данной функции SDK вычитывает со сканера все актуальные параметры, сохраняя их ввиде «списка параметров» для дальнейшей работы во внутренней памяти SDK.*

Параметры:

- `device` - *Указатель на сканер*
- `timeout` - *Время получения списка параметров со сканера.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service*

Protocol, ENIP, Modbus-TCP)

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief read_params_from_scanner - Read parameters from device to
 * Internal structure.
 *
 * @param device Ptr to scanner
 * @param timeout Time to read parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
 rfUint8 read_params_from_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      uint8_t is_read = read_params_from_scanner(scanner, 300, kSERVICE);
      if (is_read) {
         printf("Scanner parameters were read successfully!");
         // some actions with params...
      }else
         printf("Scanner parameters were not read!");
   }
}
```

get_parameter

Прототип: *parameter_t* get_parameter(scanner_base_t *device, const rfChar* param_name);*

Описание: *Функция получения конкретного параметра по его имени (ключу). При вызове данной функции SDK осуществляет поиск нужного параметра из последних прочитанных при вызове функции read_params_from_scanner . В случае, если запрашиваемый параметр отсутствует в конкретном сканере, функция вернёт NULL.*

Параметры:

- `device` - *Указатель на сканер*
- `param_name` - *Имя (ключ) параметра.*

Возвращаемое значение: `parameter_t*` *при успехе, иначе -* `NULL`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_parameter - Search parameters by his name
 *
 * @param device - ptr to scanner
 * @param param_name - name of parameter
 *
 * @return param on success, else - null
 */
parameter_t* get_parameter(
      scanner_base_t *device, const rfChar* param_name);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Establish connection.
        connect_to_scanner(scanner, kSERVICE);

        // Read params.
        read_params_from_scanner(scanner, 300, kSERVICE);

        // Get parameter of Device Name
        parameter_t* name = get_parameter(scanner,"user_general_deviceName");
        if (name != NULL) {
            char* value = name->val_str->value;
            printf("Current Device Name\t: %s\n", value);
        }

        // Get parameter of Sensor Framerate
        parameter_t* framerate = get_parameter(scanner,"user_sensor_framerate");
        if (framerate != NULL) {
            uint32_t value = framerate->val_uint32->value;
            printf("Current FPS\t\t: %d\n", value);
        }

        // some actions with other parameters...

    }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

set_parameter

Прототип: *rfUint8 set_parameter(scanner_base_t *device, parameter_t* param)*

Описание: *Функция установки конкретного параметра. При вызове данной функции происходит* установка параметра в списке параметров во внутренней памяти SDK.* *Для отправки изменений в сканер необходимо вызвать метод write_params_to_scanner .*

Параметры:

- `device` - *Указатель на сканер.*
- `param` - *Указатель на параметр для установки.*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief set_parameter - Set parameter
 *
 * @param device Ptr to scanner
 * @param param Parameter name
 *
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @return TRUE on success
 */
rfUint8 set_parameter(
      scanner_base_t *device, parameter_t* param);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);

      //
      // Example of working with the parameter type:
      // uint32_t
      //
      // Get parameter of Laser Enabled
      parameter_t* laser = get_parameter(scanner, "user_laser_enabled");
      if (laser != NULL)
      {
         uint32_t is_enabled = laser->val_uint32->value;
         printf("Current Laser State\t: %s\n", (is_enabled? "ON":"OFF"));

         // Change the current state to the opposite
         is_enabled = !is_enabled;
         laser_enabled->val_uint32->value = is_enabled;
         printf("New Laser State\t: %s\n", (is_enabled? "ON":"OFF"));
         printf("-------------------------------------\n");

         set_parameter(scanner, laser_enabled);
      }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      // some actions with other parameters before applying changes...

   }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

write_params_to_scanner

Прототип: *rfUint8 write_params_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol)*

Описание: *Функция передачи параметров из внутренней памяти SDK в сканер. При вызове данной функции происходит отправка изменённых параметров в сканер*

Параметры:

- `device` - *Указатель на сканер.*
- `timeout` - *Время отправки изменённых параметров в сканер.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief write_params_to_scanner - Send current parameters to device
 *
 * @param device Ptr to scanner
 * @param timeout Time to send parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 write_params_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   // Print count of discovered rf627old in network by Service Protocol
   printf("Discovered: %d rf627-old\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);

      //
      // Example of working with the parameter type:
      // uint32_t
      //
      // Get parameter of Laser Enabled
      parameter_t* laser = get_parameter(scanner, "user_laser_enabled");
      if (laser != NULL)
      {
         uint32_t is_enabled = laser->val_uint32->value;
         printf("Current Laser State\t: %s\n", (is_enabled? "ON":"OFF"));

         // Change the current state to the opposite
         is_enabled = !is_enabled;
         laser_enabled->val_uint32->value = is_enabled;
         printf("New Laser State\t: %s\n", (is_enabled? "ON":"OFF"));
         printf("-------------------------------------\n");

         set_parameter(scanner, laser_enabled);
      }

      // some actions with other parameters before applying changes...

      // Apply changed parameters to the device
      uint8_t is_applied = write_params_to_scanner(scanner, 300, kSERVICE);
      if (is_applied)
         printf("Scanner parameters were applied successfully!");
      else
         printf("Scanner parameters were not applied!");

   }
}
```

save_params_to_scanner

Прототип: *rfUint8 save_params_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция сохранения параметров сканера во внутреннюю память устройства. Сохраненные параметры также будут использоваться после перезапуске устройства или после смены(обновления) прошивки.*

Параметры:

- `device` - *Указатель на сканер.*
- `timeout` - *Время ожидания результата сохранения параметров в сканере.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief save_params_to_scanner - Save changes to device's memory
 *
 * @param device Ptr to scanner
 * @param timeout Time to save parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 save_params_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-old devices over network by Service Protocol.
   search_scanners(scanners, kRF627_OLD, timeout, kSERVICE);

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // After changing some parameters...

        // Apply changed parameters to the device
        write_params_to_scanner(scanner, 300, kSERVICE);

        // Save current parameters in the device memory
        uint8_t is_saved = save_params_to_scanner(scanner, 300, kSERVICE);
        if (is_saved)
            printf("Scanner parameters saved successfully!");
        else
            printf("Scanner parameters were not saved!");
    }
}
```

4.1.3 Интерфейс работы со сканерами серии RF62X v2.x.x

Файлы `rf62X_sdk.h`, `rf62X_types.h` и `rf62Xcore.h` предоставляют весь необходимый интерфейс для работы со сканерами серии RF62X v2.x.x

search_scanners

Прототип: *rfUint8 search_scanners(vector_t *list, scanner_types_t type, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция поиска устройств RF62X v2.x.x в сети*

Параметры:

- `list` - *Указатель на список, который будет заполнен найденными сканерами в сети.*
- `type` - *Тип сканера для поиска (kRF627_OLD, kRF627_SMART).*
- `timeout` - *Время поиска на каждом Ethernet интерфейсе (мс).*
- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief search_scanners - Search for RF62X devices over network
 *
 * @param[out] list Ptr to list of rf627 objects. If not null list will be
 * filled with found devices
 * @param[in] type Scanner's type (RF627-old, RF627-smart)
 * @param[in] timeout Time to search
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 search_scanners(
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    vector_t *list, scanner_types_t type,
    rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{

   // Initialize sdk library
   core_init();

   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);


   // Iterate over all available network adapters in the current operating
   // system to send "Hello" requests.
   uint32_t count = 0;
   for (int i=0; i<GetAdaptersCount(); i++)
   {
      uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
      uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
      // call the function to change adapter settings inside the library.
      set_platform_adapter_settings(host_mask, host_ip_addr);

      // Search for rf627smart devices over network by Service Protocol.
      if (host_ip_addr != 0)
      {
         // Get another IP Addr and set this changes in adapter settings.
         printf("Search scanners from:\n "
                "* IP Address\t: %s\n "
                "* Netmask\t: %s\n",
                GetAdapterAddress(i), GetAdapterMasks(i));
         search_scanners(scanners, kRF627_SMART, 300, kSERVICE);

         // Print count of discovered rf627smart in network
         printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
         printf("---------------------------------------\n");
         count = (int)vector_count(scanners);
      }
   }

   // Print count of discovered rf627smart in network
   printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // some code...
}
```

get_info_about_scanner

Прототип: *hello_information get_info_about_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция получения информации о сканере из пакета приветствия (Hello-пакет)*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому был получен пакет приветствия (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `hello_information` *в случае успеха, иначе ошибка*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_info_about_scanner - Get information about scanner from
 * hello packet
 *
 * @param[in] device Ptr to scanner
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return hello_information on success
 */
hello_information get_info_about_scanner(
      scanner_base_t *device, protocol_types_t protocol);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    for (int i = 0; i < (int)vector_count(scanners); i++)
    {
        hello_information result =
                get_info_about_scanner(vector_get(scanners,i), kSERVICE);

        rf627_smart_hello_info_by_service_protocol* info =
                result.rf627smart.hello_info_service_protocol;

        printf("\n\nID scanner's list: %d\n", i);
        printf("----------------------------------------\n");
        printf("Device information: \n");
        printf("* Name\t: %s\n", info->user_general_deviceName);
        printf("* Serial\t: %d\n", info->fact_general_serial);
        printf("* IP Addr\t: %s\n", info->user_network_ip);
        printf("* MAC Addr\t: %s\n", info->fact_network_macAddr);

        printf("\nWorking ranges: \n");
        printf("* Zsmr, mm\t: %d\n", info->fact_general_smr);
        printf("* Zmr , mm\t: %d\n", info->fact_general_mr);
        printf("* Xsmr, mm\t: %d\n", info->fact_general_xsmr);
        printf("* Xemr, mm\t: %d\n", info->fact_general_xemr);

        printf("\nVersions: \n");
        printf("* Firmware\t: %d.%d.%d\n",
               info->fact_general_firmwareVer[0],
               info->fact_general_firmwareVer[1],
               info->fact_general_firmwareVer[2]);
        printf("* Hardware\t: %d\n", info->fact_general_hardwareVer);
        printf("----------------------------------------\n");
    }

    // some code...
}
```

free_scanner

Прототип: *void free_scanner(scanner_base_t *device);*

Описание: *Функция очистки памяти, используемой объектом типа* `scanner_base_t`

Параметры:

- `device` - *Указатель на сканер*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief free_scanner - Cleanup resources allocated by device
 *
 * @param[in] device Prt to scanner
 */
void free_scanner(scanner_base_t *device);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   while (vector_count(scanners) > 0)
   {
      uint32_t index = vector_count(scanners)-1;
      // Get last scanner in vector for delete
      scanner_base_t* device = vector_get(scanners, index);

      // Cleanup resources allocated by device
      free_scanner(device);

      // Delete from vector
      vector_delete(scanners, index);
   }
}
```

connect_to_scanner

Прототип: *rfUint8 connect_to_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция установки соединения со сканером*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому будет выполнено подключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief connect_to_scanner - Establish connection to the RF62X device
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 *
 * @param[in] device Ptr to scanner
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 connect_to_scanner(
      scanner_base_t *device, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // some actions with scanner...
   }
}
```

disconnect_from_scanner

Прототип: *rfUint8 disconnect_from_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция закрытия ранее установленного соединения со сканером*

Параметры:

- `device` - *Указатель на сканер*

- `protocol` - *Тип протокола, по которому будет выполнено отключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief disconnect_from_scanner - Close connection to the device
 *
 * @param[in] device Prt to scanner
 * @param[in] protocol Protocol's type (Service, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 disconnect_from_scanner(
      scanner_base_t *device, protocol_types_t protocol);

------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // some actions with scanner...

      // Disconnect from scanner.
      disconnect_from_scanner(scanner, kSERVICE)
   }
}
```

check_connection_to_scanner

Прототип: *check_connection_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция проверки доступности сканера в сети (после подключения к нему)*

Параметры:

- `device` - *Указатель на сканер*
- `timeout` - *Время проверки соединения со сканером (мс).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief check_connection_to_scanner - Check connection to the RF62X device
 *
 * @param[in] device Ptr to scanner
 * @param[in] timeout Time to check connection
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 check_connection_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      // Establish connection to the RF627 device by Service Protocol.
      uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
      if (!isConnected){
         printf("Failed to connect to scanner!");
         continue;
      }

      // Check connection to the RF627 device.
      uint8_t is_available =
            check_connection_to_scanner(scanner, 300, kSERVICE);
      if (!is_available){
         printf("Scanner is not available now!");
         continue;
      }

      // some actions with scanner...
   }
}
```

get_profile2D_from_scanner

Прототип: *rf627_profile2D_t* get_profile2D_from_scanner(scanner_base_t *device, rfBool zero_points, rfBool realtime, protocol_types_t protocol);*

Описание: *Функция получения результатов измерений*

Параметры:

- `device` - *Указатель на сканер*
- `zero_points` - *Включать нулевые точки в возвращаемом профиле.*
- `realtime` - *Получение профиля в реальном времени (буферизация отключена).*
- `protocol` - *Тип протокола, по которому будет получен профиль (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Указатель на* `rf627_profile2D_t` *при успехе, иначе -* `NULL`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_profile2D_from_scanner - Get measurement from scanner's
 * data stream
 *
 * @param[in] device - ptr to scanner
 * @param[in] zero_points Enable zero points in return profile2D
 * @param[in] realtime Enable getting profile in realtime (buffering disabled)
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return ptr to rf627_profile_t structure
 */
rf627_profile2D_t* get_profile2D_from_scanner(
      scanner_base_t *device, rfBool zero_points,
      rfBool realtime, protocol_types_t protocol);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      uint8_t zero_points = TRUE;
      uint8_t realtime = TRUE;
      // Get profile from scanner's data stream by Service Protocol.
      rf627_profile2D_t* result = get_profile2D_from_scanner(
            scanner, zero_points, realtime, kSERVICE);
      rf627_smart_profile2D_t* profile2D = result->rf627smart_profile2D;
      if (profile2D != NULL) {
         printf("Profile was successfully received!");
         // some actions with profile...
         free_profile2D(result);
      }else
         printf("Profile was not received!");
   }
}
```

free_profile2D

Прототип: *void free_profile2D(rf627_profile2D_t* profile);*

Описание: *Функция очистки ресурсов, выделенных для rf627_profile2D_t*

Параметры:

- `profile` - *Указатель на профиль*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief free_profile2D - Cleanup resources allocated for profile2D
 *
 * @param[in] profile Ptr to rf627_profile2D_t
 */
void free_profile2D(rf627_profile2D_t* profile);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      uint8_t zero_points = TRUE;
      uint8_t realtime = TRUE;
      // Get profile from scanner's data stream by Service Protocol.
      rf627_profile2D_t* result = get_profile2D_from_scanner(
            scanner, zero_points, realtime, kSERVICE);
      rf627_smart_profile2D_t* profile2D = result->rf627smart_profile2D;
      if (profile2D != NULL) {
         printf("Profile was successfully received!");
         // some actions with profile...
         free_profile2D(result);
      }else
         printf("Profile was not received!");
   }
}
```

get_frame_from_scanner

Прототип: *rf627_frame_t* get_frame_from_scanner(scanner_base_t *device, protocol_types_t protocol);*

Описание: *Функция получения кадров с матрицы устройства*

Параметры:

- `device` - *Указатель на сканер*
- `protocol` - *Тип протокола, по которому будет получен кадр (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Указатель на* `rf627_frame_t` *при успехе, иначе -* `NULL`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_frame_from_scanner - Get RAW frame from scanner
 *
 * @param[in] device Ptr to scanner
 * @param[in] protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return ptr to rf627_frame_t structure
 */
rf627_frame_t* get_frame_from_scanner(
      scanner_base_t *device, protocol_types_t protocol);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      // Get frame from CMOS-sensor.
      rf627_frame_t* frame = get_frame_from_scanner(scanner, kSERVICE);
      if (frame != NULL) {
         printf("Frame was successfully received!");
         // some actions with Frame...
         free_frame(frame);
      }else
         printf("Frame was not received!");
   }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
}
```

free_frame

Прототип: *void free_frame(rf627_frame_t* profile);*

Описание: *Функция очистки ресурсов, выделенных для rf627_frame_t*

Параметры:

- `frame` - *Указатель на кадр*

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief free_frame - Cleanup resources allocated for frame
 *
 * @param[in] frame Ptr to rf627_frame_t
 */
void free_frame(rf627_frame_t* frame);

--------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);
      connect_to_scanner(scanner, kSERVICE);

      // Get frame from CMOS-sensor.
      rf627_frame_t* frame = get_frame_from_scanner(scanner, kSERVICE);
      if (frame != NULL) {
         printf("Frame was successfully received!");
         // some actions with Frame...
         free_frame(frame);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      }else
         printf("Frame was not received!");
   }
}
```

read_params_from_scanner

Прототип: *rfUint8 read_params_from_scanner(scanner_base_t *device, uint32_t timeout, protocol_types_t protocol);*

Описание: *Функция получения текущих параметров сканера. При вызове данной функции SDK вычитывает со сканера все актуальные параметры, сохраняя их ввиде «списка параметров» для дальнейшей работы во внутренней памяти SDK.*

Параметры:

- `device` - *Указатель на сканер*
- `timeout` - *Время получения списка параметров со сканера.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief read_params_from_scanner - Read parameters from device to
 * Internal structure.
 *
 * @param device Ptr to scanner
 * @param timeout Time to read parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
 rfUint8 read_params_from_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // Search for RF627-smart devices over network by Service Protocol.
    search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

    // Print count of discovered rf627smart in network by Service Protocol
    printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

    for (int i = 0; i < (int)vector_count(scanners); i++)
    {
        scanner_base_t* scanner = vector_get(scanners,i);
        connect_to_scanner(scanner, kSERVICE);

        uint8_t is_read = read_params_from_scanner(scanner, 300, kSERVICE);
        if (is_read) {
            printf("Scanner parameters were read successfully!");
            // some actions with params...
        }else
            printf("Scanner parameters were not read!");
    }
}
```

get_parameter

Прототип: *parameter_t* get_parameter(scanner_base_t *device, const rfChar* param_name);*

Описание: *Функция получения конкретного параметра по его имени (ключу). При вызове данной функции SDK осуществляет поиск нужного параметра из последних прочитанных при вызове функции read_params_from_scanner . В случае, если запрашиваемый параметр отсутствует в конкретном сканере, функция вернёт NULL.*

Параметры:

- `device` - *Указатель на сканер*
- `param_name` - *Имя (ключ) параметра.*

Возвращаемое значение: `parameter_t*` *при успехе, иначе -* `NULL`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief get_parameter - Search parameters by his name
 *
 * @param device - ptr to scanner
 * @param param_name - name of parameter
 *
 * @return param on success, else - null
 */
parameter_t* get_parameter(
      scanner_base_t *device, const rfChar* param_name);

-------------------------------------------------------------------------------

/** @file main.c */
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);

      // Get parameter of Device Name
      parameter_t* name = get_parameter(scanner,"user_general_deviceName");
      if (name != NULL) {
         char* value = name->val_str->value;
         printf("Current Device Name\t: %s\n", value);
      }

      // Get parameter of Sensor Framerate
      parameter_t* framerate = get_parameter(scanner,"user_sensor_framerate");
      if (framerate != NULL) {
         uint32_t value = framerate->val_uint32->value;
         printf("Current FPS\t\t: %d\n", value);
      }

      // some actions with other parameters...

   }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

set_parameter

Прототип: *rfUint8 set_parameter(scanner_base_t *device, parameter_t* param)*

Описание: *Функция установки конкретного параметра. При вызове данной функции происходит* установка параметра в списке параметров во внутренней памяти SDK.* *Для отправки изменений в сканер необходимо вызвать метод* *write_params_to_scanner* .

Параметры:

- `device` - *Указатель на сканер.*
- `param` - *Указатель на параметр для установки.*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief set_parameter - Set parameter
 *
 * @param device Ptr to scanner
 * @param param Parameter name
 *
 * @return TRUE on success
 */
rfUint8 set_parameter(
      scanner_base_t *device, parameter_t* param);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        //
        // Example of working with the parameter type:
        // uint32_t
        //
        // Get parameter of Laser Enabled
        parameter_t* laser = get_parameter(scanner, "user_laser_enabled");
        if (laser != NULL)
        {
            uint32_t is_enabled = laser->val_uint32->value;
            printf("Current Laser State\t: %s\n", (is_enabled? "ON":"OFF"));

            // Change the current state to the opposite
            is_enabled = !is_enabled;
            laser_enabled->val_uint32->value = is_enabled;
            printf("New Laser State\t: %s\n", (is_enabled? "ON":"OFF"));
            printf("-------------------------------------\n");

            set_parameter(scanner, laser_enabled);
        }

        // some actions with other parameters before applying changes...

    }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

write_params_to_scanner

Прототип: *rfUint8 write_params_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol)*

Описание: *Функция передачи параметров из внутренней памяти SDK в сканер. При вызове данной функции происходит отправка изменённых параметров в сканер*

Параметры:

- `device` - *Указатель на сканер.*
- `timeout` - *Время отправки изменённых параметров в сканер.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief write_params_to_scanner - Send current parameters to device
 *
 * @param device Ptr to scanner
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @param timeout Time to send parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 write_params_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   // Print count of discovered rf627smart in network by Service Protocol
   printf("Discovered: %d rf627-smart\n", (int)vector_count(scanners));

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);

      //
      // Example of working with the parameter type:
      // uint32_t
      //
      // Get parameter of Laser Enabled
      parameter_t* laser = get_parameter(scanner, "user_laser_enabled");
      if (laser != NULL)
      {
         uint32_t is_enabled = laser->val_uint32->value;
         printf("Current Laser State\t: %s\n", (is_enabled? "ON":"OFF"));

         // Change the current state to the opposite
         is_enabled = !is_enabled;
         laser_enabled->val_uint32->value = is_enabled;
         printf("New Laser State\t: %s\n", (is_enabled? "ON":"OFF"));
         printf("-------------------------------------\n");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            set_parameter(scanner, laser_enabled);
        }

        // some actions with other parameters before applying changes...

        // Apply changed parameters to the device
        uint8_t is_applied = write_params_to_scanner(scanner, 300, kSERVICE);
        if (is_applied)
            printf("Scanner parameters were applied successfully!");
        else
            printf("Scanner parameters were not applied!");

    }
}
```

save_params_to_scanner

Прототип: *rfUint8 save_params_to_scanner(scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);*

Описание: *Функция сохранения параметров сканера во внутреннюю память устройства. Сохраненные параметры также будут использоваться после перезапуске устройства или после смены(обновления) прошивки.*

Параметры:

- `device` - *Указатель на сканер.*
- `timeout` - *Время ожидания результата сохранения параметров в сканере.*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `TRUE` *при успехе, иначе -* `FALSE`

Пример в коде:

```
/** @file rf62X_sdk.h */

/**
 * @brief save_params_to_scanner - Save changes to device's memory
 *
 * @param device Ptr to scanner
 * @param timeout Time to save parameters
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return TRUE on success
 */
rfUint8 save_params_to_scanner(
      scanner_base_t *device, rfUint32 timeout, protocol_types_t protocol);

-------------------------------------------------------------------------------

/** @file main.c */

#include <stdio.h>
#include <stdlib.h>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   // Actions before search (see example of search_scanners() method)...

   // Search for RF627-smart devices over network by Service Protocol.
   search_scanners(scanners, kRF627_SMART, timeout, kSERVICE);

   for (int i = 0; i < (int)vector_count(scanners); i++)
   {
      scanner_base_t* scanner = vector_get(scanners,i);

      // Establish connection.
      connect_to_scanner(scanner, kSERVICE);

      // Read params.
      read_params_from_scanner(scanner, 300, kSERVICE);

      // After changing some parameters...

      // Apply changed parameters to the device
      write_params_to_scanner(scanner, 300, kSERVICE);

      // Save current parameters in the device memory
      uint8_t is_saved = save_params_to_scanner(scanner, 300, kSERVICE);
      if (is_saved)
         printf("Scanner parameters saved successfully!");
      else
         printf("Scanner parameters were not saved!");
   }
}
```

4.2 Интерфейс «обёртки» на C++

Данная библиотека позволяет упростить разработку приложений на языке C++

Для её использования в проектах C++ разработчику необходимо включить h-файлы библиотеки в свой проект, а также добавить к проекту «обёртку» в качестве статической или динамической программной библиотеки.

Для скачивания библиотеки см. Последние выпуски RF62X-SDK библиотек.
Для компиляции библиотеки из исходников см. *Компиляция «обёртки» на C++*.
Для использования в проектах см. *Создание проекта C/C++*.

4.2.1 Инициализация SDK

Файл `rf62Xsdk.h` является основным файлом программного интерфейса (API) и определяет функциональность библиотеки-«обёртки». Файл `rf62Xtypes.h` содержит дополнительный набор классов, структур, типов и перечислений используемых в SDK.

sdk_init

Прототип: *bool sdk_init();*

Описание: *Функция инициализации SDK. Должна быть вызвана один раз перед дальнейшими вызовами любых библиотечных функций*

Возвращаемое значение: *После успешного завершения возвращается 1 (true). В противном случае должен быть возвращен код ошибки.*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief sdk_init - Initialize sdk library
 * @details Must be called once before further calls to any
 * library functions
 *
 * @return true if success.
 */
int8_t sdk_init();

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   bool isInit = sdk_init();

   if (isInit)
      std::cout << "SDK version: " << sdk_version() << std::endl;
   else
   {
      std::cout << "SDK initialization error!" << std::endl;
      return -1;
   }

   // some code...

}
```

sdk_cleanup

Прототип: *void sdk_cleanup();*

Описание: *Функция высвобождает ресурсы, выделенные при инициализации СДК функцией sdk_init*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief sdk_cleanup - Cleanup resources allocated with
 * sdk_init() function
 */
void sdk_cleanup();

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{
   // Initialize sdk library
   sdk_init();

   // some code...

   // Cleanup resources
   sdk_cleanup();
}
```

sdk_version

Прототип: *std::string sdk_version();*

Описание: *Функция получения информации о версии SDK*

Возвращаемое значение: *версия SDK в формате X.Y.Z (мажорная, минорная, патч)*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief sdk_version - Return info about SDK version
 *
 * @return SDK version
 */
std::string sdk_version();

-------------------------------------------------------------------------------
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{
   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version() << std::endl;

   // some code...
}
```

4.2.2 Класс rf627old

Данный класс определён в файле `rf62Xsdk.h` и предоставляет интерфейс для работы со сканерами серии RF627 v20.x.x.x

search

Прототип: *static std::vector<std::shared_ptr<rf627old>> search(uint32_t timeout = 300, bool only_available_result = true, PROTOCOLS protocol = PROTOCOLS::SERVICE);*

Описание: *Метод поиска устройств RF62X v2.x.x в сети*

Параметры:

- `timeout` - *Время поиска на каждом Ethernet интерфейсе (мс).*
- `only_available_result` - *Без сохранения истории поиска (только доступные сканеры на текущий момент).*
- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Вектор rf627old устройств*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief search - Search for rf627old devices over network
 *
 * @param timeout Search timeout[ms] for each Ethernet interface
 * @param only_available_result Without saving search history
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @return vector of rf627old devices
 */
static std::vector<std::shared_ptr<rf627old>> search(
        uint32_t timeout = 300, bool only_available_result = true,
        PROTOCOLS protocol = PROTOCOLS::SERVICE);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627old>> list;
   // Search for rf627old devices over network
   list = rf627old::search(500);

   // Print count of discovered rf627old in network
   std::cout << "Was found\t: " << list.size() << " RF627 v20.x.x.x";

   // some code...

}
```

get_info

Прототип: *std::shared_ptr<hello_info> get_info(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения информации о сканере из пакета приветствия (Hello-пакет)*

Параметры:

- `protocol` - *Тип протокола, по которому был получен пакет приветствия (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `hello_info` *в случае успеха, иначе -* `null`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_info - Get information about scanner from hello packet
 *
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return hello_info on success, else - nullptr
 */
std::shared_ptr<hello_info> get_info(
        PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627old>> list;
   // Search for rf627old devices over network
   list = rf627old::search(500);

   // Print count of discovered rf627old in network by Service Protocol
   std::cout << "Was found\t: "<<list.size()<<" RF627-Old"<< std::endl;
   std::cout << "========================================" << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<hello_info> info = list[i]->get_info();

      std::cout << "\n\nID scanner's list: " << i                << std::endl;
      std::cout << "----------------------------------------"  << std::endl;
      std::cout << "Device information: "                       << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()       << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()     << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()        << std::endl;
      std::cout << "* MAC Addr\t: " << info->mac_address()       << std::endl;

      std::cout << "\nWorking ranges: "                          << std::endl;
      std::cout << "* Zsmr, mm\t: " << info->z_smr()             << std::endl;
      std::cout << "* Zmr , mm\t: " << info->z_mr()              << std::endl;
      std::cout << "* Xsmr, mm\t: " << info->x_smr()             << std::endl;
      std::cout << "* Xemr, mm\t: " << info->x_emr()             << std::endl;

      std::cout << "\nVersions: "                                << std::endl;
      std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
      std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
      std::cout << "----------------------------------------"  << std::endl;
   }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // some code...
}
```

connect

Прототип: *bool connect(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод установки соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено подключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief connect - Establish connection to the rf627old device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
 bool connect(PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Create value for scanners vector's type
    std::vector<std::shared_ptr<rf627old>> list;
    // Search for rf627old devices over network
    list = rf627old::search(500);

    // Print count of discovered rf627old in network by Service Protocol
    std::cout << "Was found\t: "<<list.size()<<" RF627-Old"<< std::endl;
    std::cout << "========================================" << std::endl;


    for (size_t i = 0; i < list.size(); i++)
    {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        std::shared_ptr<rf627old> scanner = list[i];

        // Establish connection to the RF627 device by Service Protocol.
        bool isConnected = scanner->connect();
        if (!isConnected){
            std::cout << "Failed to connect to scanner!" << std::endl;
            continue;
        }

        // some actions with scanner...

    }
}
```

is_connected

Прототип: *bool is_connected();*

Описание: *Получение статуса подключения к сканеру методом connect*

Возвращаемое значение: `true` *, если соединение со сканером было успешно установлено, иначе - * `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief is_connected - Scanner connection status by the
 * connect() method.
 *
 * @return true, if a connection to the scanner was previously
 * established using the connect() method, else - false.
 */
bool is_connected();

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627old devices over network
    std::vector<std::shared_ptr<rf627old>> list = rf627old::search();

    for (size_t i = 0; i < list.size(); i++)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    {
        std::shared_ptr<rf627old> scanner = list[i];

        // Establish connection to the RF627 device by Service Protocol.
        scanner->connect();

        bool result = scanner->is_connected()
        if (result) {
            std::cout << "Connection has been established";
        }
    }
}
```

disconnect

Прототип: *bool disconnect(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод закрытия ранее установленного соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено отключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief disconnect - Close connection to the device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
 bool disconnect(PROTOCOLS protocol = PROTOCOLS::CURRENT);

------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627old devices over network
    std::vector<std::shared_ptr<rf627old>> list = rf627old::search();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    for (size_t i = 0; i < list.size(); i++)
    {
        std::shared_ptr<rf627old> scanner = list[i];

        // Establish connection to the RF627 device by Service Protocol.
        bool isConnected = scanner->connect();
        if (!isConnected){
            std::cout << "Failed to connect to scanner!" << std::endl;
            continue;
        }

        // some actions with scanner...

        // Disconnect from scanner.
        scanner->disconnect();
    }
}
```

check_connection

Прототип: *bool check_connection(uint32_t timeout = 500, PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод проверки доступности сканера в сети (после подключения к нему)*

Параметры:

- `timeout` - *Время проверки соединения со сканером (мс).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief check_connection - Check the connection with the
 * rf627old device
 *
 * @param timeout Connection check timeout
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool check_connection(
        uint32_t timeout = 500, PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> list = rf627old::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627old> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (!isConnected){
         std::cout << "Failed to connect to scanner!" << std::endl;
         continue;
      }

      // Check connection to the RF627 device.
      bool isAvailable = scanner->check_connection(300);
      if (!isAvailable){
         std::cout << "Scanner is not available now, "
                   << "please call back later!" << std::endl;
         continue;
      }

      // some actions with scanner...
   }
}
```

is_available

Прототип: *bool is_available();*

Описание: *Метод получения статуса доступности сканера в сети. Значение, возвращаемое методом, зависит от результатов выполнения методов search и check_connection*

Возвращаемое значение: `true` *если сканер доступен, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief is_available - Scanner availability status on the network.
 * @details The value returned by the method depends on the results
 * of the execution of the search() and check_connection() methods.
 *
 * @return true, if the scanner is available, otherwise - false.
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 */
bool is_available();

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> list = rf627old::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627old> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      scanner->connect();

      // some time after using the scanner...

      // Check network connections to scanner
      scanner->check_connection(300);
      bool isAvailable = scanner->is_available();
      if (!isAvailable){
         std::cout << "Scanner is not available!" << std::endl;
         std::cout << "Check the power supply to the scanner.";
      }

      // some code...

   }
}
```

get_profile2D

Прототип: *std::shared_ptr<profile2D> get_profile2D(bool zero_points = true, bool realtime = true, PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения результатов измерений*

Параметры:

- `zero_points` - *Включать нулевые точки в возвращаемом профиле.*
- `realtime` - *Получение профиля в реальном времени (буферизация отключена).*

- `protocol` - *Тип протокола, по которому будет получен профиль (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `std::shared_ptr<profile2D>` *при успехе, иначе -* `nullptr`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_profile2D - Get 2D measurement from scanner's data stream
 *
 * @param zero_points Enable zero points in return profile2D
 * @param realtime Enable getting profile in real time (buffering is disabled)
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return profile2D if success, else - nullptr
 */
std::shared_ptr<profile2D> get_profile2D(
        bool zero_points = true, bool realtime = true,
        PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> list = rf627old::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627old> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (isConnected) {
         bool zero_points = true;
         bool realtime = true;
         std::shared_ptr<profile2D> profile = nullptr;

         // Get profile from scanner
         profile = scanner->get_profile2D(zero_points, realtime);
         if (profile != nullptr) {
            std::cout << "Profile was successfully received!" << std::endl;
            // some actions with profile...
         }else
            std::cout << "Profile was not received!" << std::endl;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        }
        // some code...
    }
}
```

read_params

Прототип: *bool read_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения текущих параметров сканера. При вызове данного метода SDK вычитывает со сканера все актуальные параметры, сохраняя их в виде «списка параметров» для дальнейшей работы во внутренней памяти SDK.*

Параметры:

- `protocol` - *Тип протокола, по которому будут прочитаны параметры (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief read_params - Read parameters from device to
 * internal SDK memory
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool read_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> list = rf627old::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627old> scanner = list[i];
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Establish connection to the RF627 device by Service Protocol.
        bool isConnected = scanner->connect();
        if (isConnected) {
            // read params from RF627 device by Service Protocol.
            bool isRead = scanner->read_params();
            if (isRead) {
                std::cout << "Scanner parameters were read successfully!";
                // some actions with params...
            }else
                std::cout << "Scanner parameters were not read!";
        }
    }
}
```

get_param

Прототип: *std::shared_ptr<param> get_param(std::string param_name);*

Описание: *Метод получения конкретного параметра по его имени (ключу). При вызове данного метода SDK осуществляет поиск нужного параметра из последних прочитанных при вызове функции read_params . В случае, если запрашиваемый параметр отсутствует в конкретном сканере, метод вернёт nullptr.*

Параметры:

- `param_name` - *Имя (ключ) параметра.*

Возвращаемое значение: `std::shared_ptr<param>` *при успехе, иначе -* `nullptr`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_param - Get parameter by his name
 * Before using read_params() method should be called
 *
 * @param param_name Name of parameter
 *
 * @return param on success, else - null
 */
std::shared_ptr<param> get_param(std::string param_name);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // Initialize sdk library
    sdk_init();

    // Search for rf627old devices over network
    std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

    for (size_t i = 0; i < scanners.size(); i++)
    {
        // Establish connection.
        scanners[i]->connect();

        // Read params.
        scanners[i]->read_params();

        // Get parameter of Device Name
        auto name = scanners[i]->get_param("user_general_deviceName");
        if (name != nullptr) {
            std::string str_name = name->getValue<std::string>();
            std::cout << "Current Device Name \t: " << str_name << std::endl;
        }

        // Get parameter of Sensor Framerate
        auto framerate = scanner->get_param("user_sensor_framerate");
        if (framerate != nullptr) {
            uint32_t framerate_value = framerate->getValue<uint32_t>();
            std::cout<<"Current FPS\t\t: "<< framerate_value << std::endl;
        }

        // some actions with other parameters...

    }
    // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

set_param

Прототип: *bool set_param(std::string name, T value);*

Описание: *Метод установки конкретного параметра. При вызове данного метода происходит* установка параметра в списке параметров во внутренней памяти SDK.* *Для отправки изменений в сканер необходимо вызвать метод write_params .*

Параметры:

- `param_name` - *Имя (ключ) параметра.*
- `value` *- Новое значение параметра

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief set_param - Set parameter
 *
 * @param name Name of parameter
 * @param value Value to set
 *
 * @return true on success, else - false
 */
template<typename T>
bool set_param(std::string name, T value);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <vector>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // Set parameter of Device Name
      scanner->set_param("user_general_deviceName", "RF627 New Name");

      // Sen parameter of Sensor Framerate
      scanner->set_param("user_sensor_framerate", 100);

      // Set parameter of Device IP Addr
      std::vector<uint32_t> ip {192, 168, 1, 31};
      scanner->set_param("user_network_ip", ip);

      // some actions with other parameters...

   }
   // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

set_param_by_key

Прототип: *bool set_param_by_key(std::string name, std::string key);*

Описание: *Метод установки конкретного параметра по ключу. При вызове данного метода происходит установка параметра в списке параметров во внутренней памяти SDK. Для отправки изменений в сканер необходимо вызвать метод write_params .*

Параметры:

- `param_name` - *Имя (ключ) параметра.*
- `key` *- Ключ (enum) параметра

Возвращаемое значение: `true` *при успехе, иначе* - `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief set_param_by_key - Set parameter from Enum
 *
 * @param name Name of parameter
 * @param key Key to set
 *
 * @return true on success, else - false
 */
bool set_param_by_key(std::string name, std::string key);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // Set parameter of Laser Enabled (TRUE or FALSE)
      scanner->set_param_by_key("user_laser_enabled", "FALSE");

      // Sen parameter of Sensor sync source (SYNC_INTERNAL,
      // SYNC_EXTERNAL, SYNC_SOFTWARE_EXT or SYNC_SOFTWARE)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        scanner->set_param_by_key("user_sensor_syncSource", "SYNC_INTERNAL");

        // Set parameter of Streams Format (DATA_FORMAT_PROFILE or
        // DATA_FORMAT_RAW_PROFILE)
        scanner->set_param_by_key("user_streams_format", "DATA_FORMAT_PROFILE");

        // some actions with other parameters...

    }
    // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

write_params

Прототип: *bool write_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод передачи параметров из внутренней памяти SDK в сканер. При вызове данного метода происходит отправка изменённых параметров в сканер*

Параметры:

- `protocol` - *Тип протокола, по которому будут отправлена команда на установку параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief write_params - Send current parameters to device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool write_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    // Initialize sdk library
    sdk_init();

    // Search for rf627old devices over network
    std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

    for (size_t i = 0; i < scanners.size(); i++)
    {
        scanners[i]->connect();
        scanners[i]->read_params();

        // Set parameter of Device Name
        scanner->set_param("user_general_deviceName", "RF627 New Name");
        // Sen parameter of Sensor Framerate
        scanner->set_param("user_sensor_framerate", 100);

        // some actions with other parameters...

        // Apply changed parameters to the device
        bool isApplied = scanner->write_params();
        if (isApplied)
            std::cout << "Scanner parameters were applied successfully!";
        else
            std::cout << "Scanner parameters were not applied!";

    }
    // some code...
}
```

save_params

Прототип: *bool save_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод сохранения параметров сканера во внутреннюю память устройства. Сохраненные параметры также будут использоваться после перезапуске устройства или после смены(обновления) прошивки.*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда сохранения параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief save_params - Save changes to device's memory
 * @details The saved parameters will also be used if the device
 * is restarted or even if the firmware is updated.
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @return true on success, else - false
 */
bool save_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // After changing some parameters...

      // Apply changed parameters to the device
      scanner->write_params();

      // Save current parameters in the device memory
      bool isSaved = scanner->save_params();
      if (isSaved)
         std::cout << "Scanner parameters saved successfully!";
      else
         std::cout << "Scanner parameters were not saved!";

   }
   // some code...
}
```

reboot_device

Прототип: *bool reboot_device(PROTOCOLS protocol = PROTOCOLS::CURRENT)*

Описание: *Метод перезагрузки устройства*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда на перезагрузку устройства (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief reboot_device - The scanner will restart
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool reboot_device(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Start device reboot
      scanners[i]->reboot_device();

      // Waiting 10 sec
      std::this_thread::sleep_for(std::chrono::seconds(10));

      bool isAvailable = scanners[i]->check_connection();
      if (isAvailable){
         std::cout << "Scanner has been successfully restarted" << std::endl;
      }

      // some other actions with scanner...

   }
}
```

send_cmd

Прототип: *bool send_cmd(std::string command_name, std::vector<uint8_t> in, std::vector<uint8_t>& out);*

Описание: *Метод отправки команды в сканер*

Параметры:

- `command_name` - *Название команды. (CID_PERIPHERY_SEND, CID_PROFILE_SET_COUNTERS, и др.)*
- `in` - *Данные для отправки.*
- `out` - *Данные, которые были получены.*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief send_cmd - Send command to scanner
 *
 * @param command_name Name of command:
 * CID_PERIPHERY_SEND - send/receive data to/from a peripheral device
 * CID_PROFILE_SET_COUNTERS - set counters in devices
 * @param input Data to send in command payload
 * @param output Data to receive from command payload
 *
 * @return true on success, else - false
 */
bool send_cmd(std::string command_name,
              std::vector<uint8_t> in, std::vector<uint8_t>& out);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627old devices over network
   std::vector<std::shared_ptr<rf627old>> scanners = rf627old::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Send data to periphery
      std::vector<uint8_t> in { 1, 2, 3, 4, 5};
      std::vector<uint8_t> out;
      bool isSent = scanners[i]->send_cmd("CID_PERIPHERY_SEND", in, out);
      if (isSent){
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            std::cout << "The data was sent successfully." << std::endl;
            std::cout << "Size of received data: " << out.size() << std::endl;
        }

        // Reset counters
        std::vector<uint8_t> in(0);
        std::vector<uint8_t> out;
        isSent = scanners[i]->send_cmd("CID_PROFILE_SET_COUNTERS", in, out);
        if (isSent){
            std::cout << "The counters was reset successfully." << std::endl;
        }

        // some other actions with scanner...
    }
}
```

4.2.3 Класс rf627smart

Данный класс определён в файле `rf62Xsdk.h` и предоставляет интерфейс для работы со сканерами серии RF62X v2.x.x

search

Прототип: *static std::vector<std::shared_ptr<rf627smart>> search(uint32_t timeout = 300, bool only_available_result = true, PROTOCOLS protocol = PROTOCOLS::SERVICE);*

Описание: *Метод поиска устройств RF62X v2.x.x в сети*

Параметры:

- `timeout` - *Время поиска на каждом Ethernet интерфейсе (мс).*
- `only_available_result` - *Без сохранения истории поиска (только доступные сканера на текущий момент).*
- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Вектор rf627smart устройств*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief search - Search for rf627smart devices over network
 *
 * @param timeout Search timeout[ms] for each Ethernet interface
 * @param only_available_result Without saving search history
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return vector of rf627smart devices
 */
static std::vector<std::shared_ptr<rf627smart>> search(
        uint32_t timeout = 300, bool only_available_result = true,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        PROTOCOLS protocol = PROTOCOLS::SERVICE);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network
   std::cout << "Was found\t: " << list.size() << " RF62X v2.x.x";

   // some code...

}
```

get_info

Прототип: *std::shared_ptr<hello_info> get_info(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения информации о сканере из пакета приветствия (Hello-пакет)*

Параметры:

- `protocol` - *Тип протокола, по которому был получен пакет приветствия (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `hello_info` *в случае успеха, иначе -* `null`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_info - Get information about scanner from hello packet
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return hello_info on success, else - nullptr
 */
```

(continues on next page)

(продолжение с предыдущей страницы)

```
std::shared_ptr<hello_info> get_info(
        PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: "<<list.size()<<" RF627-Smart"<< std::endl;
   std::cout << "========================================" << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<hello_info> info = list[i]->get_info();

      std::cout << "\n\nID scanner's list: " << i               << std::endl;
      std::cout << "----------------------------------------"  << std::endl;
      std::cout << "Device information: "                      << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()      << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()    << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()       << std::endl;
      std::cout << "* MAC Addr\t: " << info->mac_address()      << std::endl;

      std::cout << "\nWorking ranges: "                         << std::endl;
      std::cout << "* Zsmr, mm\t: " << info->z_smr()            << std::endl;
      std::cout << "* Zmr , mm\t: " << info->z_mr()             << std::endl;
      std::cout << "* Xsmr, mm\t: " << info->x_smr()            << std::endl;
      std::cout << "* Xemr, mm\t: " << info->x_emr()            << std::endl;

      std::cout << "\nVersions: "                               << std::endl;
      std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
      std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
      std::cout << "----------------------------------------"  << std::endl;
   }

   // some code...
}
```

connect

Прототип: *bool connect(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод установки соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено подключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief connect - Establish connection to the rf627smart device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
 bool connect(PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: "<<list.size()<<" RF627-Smart"<< std::endl;
   std::cout << "========================================" << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (!isConnected){
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            std::cout << "Failed to connect to scanner!" << std::endl;
            continue;
        }

        // some actions with scanner...

    }
}
```

is_connected

Прототип: *bool is_connected();*

Описание: *Получение статуса подключения к сканеру методом connect*

Возвращаемое значение: `true` *, если соединение со сканером было успешно установлено, иначе - * `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief is_connected - Scanner connection status by the
 * connect() method.
 *
 * @return true, if a connection to the scanner was previously
 * established using the connect() method, else - false.
 */
bool is_connected();

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627smart devices over network
    std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

    for (size_t i = 0; i < list.size(); i++)
    {
        std::shared_ptr<rf627smart> scanner = list[i];

        // Establish connection to the RF627 device by Service Protocol.
        scanner->connect();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        bool result = scanner->is_connected()
        if (result) {
            std::cout << "Connection has been established";
        }
    }
}
```

disconnect

Прототип: *bool disconnect(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод закрытия ранее установленного соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено отключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief disconnect - Close connection to the device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
 bool disconnect(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627smart devices over network
    std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

    for (size_t i = 0; i < list.size(); i++)
    {
        std::shared_ptr<rf627smart> scanner = list[i];
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Establish connection to the RF627 device by Service Protocol.
        bool isConnected = scanner->connect();
        if (!isConnected){
            std::cout << "Failed to connect to scanner!" << std::endl;
            continue;
        }

        // some actions with scanner...

        // Disconnect from scanner.
        scanner->disconnect();
    }
}
```

check_connection

Прототип: *bool check_connection(uint32_t timeout = 500, PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод проверки доступности сканера в сети (после подключения к нему)*

Параметры:

- `timeout` - *Время проверки соединения со сканером (мс).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief check_connection - Check the connection with the
 * rf627smart device
 *
 * @param timeout Connection check timeout
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool check_connection(
        uint32_t timeout = 500, PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
```

(continues on next page)

(продолжение с предыдущей страницы)

```
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (!isConnected){
         std::cout << "Failed to connect to scanner!" << std::endl;
         continue;
      }

      // Check connection with device
      bool isAvailable = scanner->check_connection(300);
      if (!isAvailable){
         std::cout << "Scanner is not available now, "
                   << "please call back later!" << std::endl;
         continue;
      }

      // some actions with scanner...
   }
}
```

is_available

Прототип: *bool is_available();*

Описание: *Метод получения статуса доступности сканера в сети. Значение, возвращаемое методом, зависит от результатов выполнения методов search и check_connection*

Возвращаемое значение: `true` *если сканер доступен, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief is_available - Scanner availability status on the network.
 * @details The value returned by the method depends on the results
 * of the execution of the search() and check_connection() methods.
 *
 * @return true, if the scanner is available, otherwise - false.
 */
bool is_available();

-------------------------------------------------------------------------------
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      scanner->connect();

      // some time after using the scanner...

      // Check network connections to scanner
      scanner->check_connection(300);
      bool isAvailable = scanner->is_available();
      if (!isAvailable){
         std::cout << "Scanner is not available!" << std::endl;
         std::cout << "Check the power supply to the scanner.";
      }

      // some code...

   }
}
```

get_profile2D

Прототип: *std::shared_ptr<profile2D> get_profile2D(bool zero_points = true, bool realtime = true, PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения результатов измерений*

Параметры:

- `zero_points` - *Включать нулевые точки в возвращаемом профиле.*
- `realtime` - *Получение профиля в реальном времени (буферизация отключена).*
- `protocol` - *Тип протокола, по которому будет получен профиль (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `std::shared_ptr<profile2D>` *при успехе, иначе -* `nullptr`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_profile2D - Get 2D measurement from scanner's data stream
 *
 * @param zero_points Enable zero points in return profile2D
 * @param realtime Enable getting profile in real time (buffering is disabled)
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return profile2D if success, else - nullptr
 */
std::shared_ptr<profile2D> get_profile2D(
        bool zero_points = true, bool realtime = true,
        PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (isConnected) {
         bool zero_points = true;
         bool realtime = true;
         std::shared_ptr<profile2D> profile = nullptr;

         // Get profile from scanner
         profile = scanner->get_profile2D(zero_points, realtime);
         if (profile != nullptr) {
            std::cout << "Profile was successfully received!" << std::endl;
            // some actions with profile...
         }else
            std::cout << "Profile was not received!" << std::endl;
      }
      // some code...
   }
}
```

get_frame

Прототип: *std::shared_ptr<frame> get_frame(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения кадров видео с матрицы сканера*

Параметры:

- `protocol` - *Тип протокола, по которому будет получен кадр (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `std::shared_ptr<frame>` *при успехе, иначе -* `nullptr`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_frame - Get RAW frame from scanner
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return frame if success, else - null
 */
std::shared_ptr<frame> get_frame(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];

      // Establish connection to the RF627 device by Service Protocol.
      bool isConnected = scanner->connect();
      if (isConnected) {
         // Get Frame from scanner.
         std::shared_ptr<frame> frame = scanner->get_frame();
         if (frame != nullptr) {
            std::cout << "Frame was successfully received!" << std::endl;
            // some actions with profile...
         }else
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            std::cout << "Frame was not received!" << std::endl;
        }
        // some code...
    }
}
```

read_params

Прототип: *bool read_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод получения текущих параметров сканера. При вызове данного метода SDK вычитывает со сканера все актуальные параметры, сохраняя их ввиде «списка параметров» для дальнейшей работы во внутренней памяти SDK.*

Параметры:

- `protocol` - *Тип протокола, по которому будут прочитаны параметры (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief read_params - Read parameters from device to
 * internal SDK memory
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool read_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627smart devices over network
    std::vector<std::shared_ptr<rf627smart>> list = rf627smart::search();

    for (size_t i = 0; i < list.size(); i++)
    {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        std::shared_ptr<rf627smart> scanner = list[i];

        // Establish connection to the RF627 device.
        bool isConnected = scanner->connect();
        if (isConnected) {
            // read params from RF627 device.
            bool isRead = scanner->read_params();
            if (isRead) {
                std::cout << "Scanner parameters were read successfully!";
                // some actions with params...
            }else
                std::cout << "Scanner parameters were not read!";
        }
    }
    // some code...
}
```

get_param

Прототип: *std::shared_ptr<param> get_param(std::string param_name);*

Описание: *Метод получения конкретного параметра по его имени (ключу). При вызове данного метода SDK осуществляет поиск нужного параметра из последних прочитанных при вызове метода read_params . В случае, если запрашиваемый параметр отсутствует в конкретном сканере, функция вернёт nullptr.*

Параметры:

- `param_name` - *Имя (ключ) параметра.*

Возвращаемое значение: `std::shared_ptr<param>` *при успехе, иначе -* `nullptr`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_param - Get parameter by his name
 * Before using read_params() method should be called
 *
 * @param param_name Name of parameter
 *
 * @return param on success, else - null
 */
std::shared_ptr<param> get_param(std::string param_name);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"
```

(continues on next page)

(продолжение с предыдущей страницы)

```
int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      // Establish connection.
      scanners[i]->connect();

      // Read params.
      scanners[i]->read_params();

      // Get parameter of Device Name
      auto name = scanners[i]->get_param("user_general_deviceName");
      if (name != nullptr) {
         std::string str_name = name->getValue<std::string>();
         std::cout << "Current Device Name \t: " << str_name << std::endl;
      }

      // Get parameter of Sensor Framerate
      auto framerate = scanner->get_param("user_sensor_framerate");
      if (framerate != nullptr) {
         uint32_t framerate_value = framerate->getValue<uint32_t>();
         std::cout<<"Current FPS\t\t: "<< framerate_value << std::endl;
      }

      // some actions with other parameters...

   }
   // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

set_param

Прототип: *bool set_param(std::string name, T value);*

Описание: *Метод установки конкретного параметра. При вызове данного метода происходит* установка параметра в списке параметров во внутренней памяти SDK.* *Для отправки изменений в сканер необходимо вызвать метод write_params .*

Параметры:

- `param_name` - *Имя (ключ) параметра.*
- `value` - *Новое значение параметра*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief set_param - Set parameter
 *
 * @param name Name of parameter
 * @param value Value to set
 *
 * @return true on success, else - false
 */
template<typename T>
bool set_param(std::string name, T value);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <vector>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // Set parameter of Device Name
      scanner->set_param("user_general_deviceName", "RF627 New Name");

      // Set parameter of Sensor Framerate
      scanner->set_param("user_sensor_framerate", 100);

      // Set parameter of Device IP Addr
      std::vector<uint32_t> ip {192, 168, 1, 31};
      scanner->set_param("user_network_ip", ip);

      // some actions with other parameters...

   }
   // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X

Firmware Cloud

set_param_by_key

Прототип: *bool set_param_by_key(std::string name, std::string key);*

Описание: *Метод установки конкретного параметра по ключу. При вызове данного метода происходит установка параметра в списке параметров во внутренней памяти SDK. Для отправки изменений в сканер необходимо вызвать метод write_params .*

Параметры:

- `param_name` - *Имя (ключ) параметра.*
- `key` - *Ключ (enum) параметра*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief set_param_by_key - Set parameter from Enum
 *
 * @param name Name of parameter
 * @param key Key to set
 *
 * @return true on success, else - false
 */
bool set_param_by_key(std::string name, std::string key);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Set parameter of Laser Enabled (TRUE or FALSE)
        scanner->set_param_by_key("user_laser_enabled", "FALSE");

        // Sen parameter of Sensor sync source (SYNC_INTERNAL,
        // SYNC_EXTERNAL, SYNC_SOFTWARE_EXT or SYNC_SOFTWARE)
        scanner->set_param_by_key("user_sensor_syncSource", "SYNC_INTERNAL");

        // Set parameter of Streams Format (DATA_FORMAT_PROFILE or
        // DATA_FORMAT_RAW_PROFILE)
        scanner->set_param_by_key("user_streams_format", "DATA_FORMAT_PROFILE");

        // some actions with other parameters...

    }
    // some code...
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

write_params

Прототип: *bool write_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод передачи параметров из внутренней памяти SDK в сканер. При вызове данного метода происходит отправка изменённых параметров в сканер*

Параметры:

- `protocol` - *Тип протокола, по которому будут отправлена команда на установку параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief write_params - Send current parameters to device
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool write_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // Set parameter of Device Name
      scanner->set_param("user_general_deviceName", "RF627 New Name");
      // Sen parameter of Sensor Framerate
      scanner->set_param("user_sensor_framerate", 100);

      // some actions with other parameters...

      // Apply changed parameters to the device
      bool isApplied = scanner->write_params();
      if (isApplied)
         std::cout << "Scanner parameters were applied successfully!";
      else
         std::cout << "Scanner parameters were not applied!";

   }
   // some code...
}
```

save_params

Прототип: *bool save_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод сохранения параметров сканер во внутреннюю память устройства. Сохраненные параметры также будут использоваться после перезапуске устройства или после смены(обновления) прошивки.*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда сохранения параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief save_params - Save changes to device's memory
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * @details The saved parameters will also be used if the device
 * is restarted or even if the firmware is updated.
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool save_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();
      scanners[i]->read_params();

      // After changing some parameters...

      // Apply changed parameters to the device
      scanner->write_params();

      // Save current parameters in the device memory
      bool isSaved = scanner->save_params();
      if (isSaved)
         std::cout << "Scanner parameters saved successfully!";
      else
         std::cout << "Scanner parameters were not saved!";

   }
   // some code...
}
```

load_recovery_params

Прототип: *bool load_recovery_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);*

Описание: *Метод загрузки значений параметров устройства из области восстановления. Загруженные значения будут записаны в пользовательскую область*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда на восстановление параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief load_recovery_params - Loading parameters from recovery area
 * @details The device will automatically reboot.
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool load_recovery_params(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Load parameters from recovery area
      bool isLoaded = scanner->load_recovery_params();
      if (isLoaded)
         std::cout << "Recovery parameters loaded successfully!";
      else
         std::cout << "Recovery parameters were not loaded!";

   }
   // some code...
}
```

start_dump_recording

Прототип: *bool start_dump_recording(uint32_t count_of_profiles = 0);*

Описание: *Метод включения записи профилей во внутреннюю память устройства - запуск записи дампа. Запись остановится, когда количество записанных профилей превысит максимально допустимый размер дампа, или когда будет превышено количество переданного в метод параметра* `count_of_profiles`, *или когда будет вызван метод остановки записи* *stop_dump_recording* .

Параметры:

- `count_of_profiles` - *Количество профилей для записи дампа*

Возвращаемое значение: `true` *если запись началась успешно, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief start_dump_recording - enabling profile recording to the internal
 * memory of the device - generating a dump.
 * @details Recording will stop when the number of recorded profiles exceeds
 * the maximum allowed dump size, or when the count_of_profiles number is
 * exceeded, or when the stop_dump_recording method is called.
 *
 * @param count_of_profiles The number of profiles to record the dump:
 * if count_of_profiles == 0 - Recording will continue until the maximum
 * dump size is reached, or until recording is stopped by calling
 * the stop_dump_recording method;
 * if count_of_profiles > 0  - Recording will continue until the number
 * of recorded profiles exceeds the specified number.
 *
 * @return true if recording started successfully, else - false
 */
 bool start_dump_recording(uint32_t count_of_profiles = 0);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Start dump recording
      uint32_t count_of_profiles = 1000;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        bool isStarted = scanners[i]->start_dump_recording(count_of_profiles);
        if (isStarted)
            std::cout << "Dump recording started...";
        else
            std::cout << "Dump recording failed!";

        // Next steps in stop_dump_recording example...

    }
}
```

stop_dump_recording

Прототип: *bool stop_dump_recording(uint32_t& count_of_profiles);*

Описание: *Метод остановки записи профилей во внутреннюю память устройства - остановка записи дампа.*

Параметры:

- `count_of_profiles` - *Количество записанных профилей в дампе*

Возвращаемое значение: `true` *если запись остановлена успешно, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief stop_dump_recording - disabling profile recording to the internal
 * memory of the device.
 *
 * @param count_of_profiles The number of recorded profiles
 *
 * @return true if recording was stopped successfully, else - false
 */
bool stop_dump_recording(uint32_t& count_of_profiles);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627smart devices over network
    std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    for (size_t i = 0; i < scanners.size(); i++)
    {
        scanners[i]->connect();

        // Start dump recording
        scanners[i]->start_dump_recording();

        // Waiting 1 sec
        std::this_thread::sleep_for(std::chrono::seconds(1));

        // Stop dump recording
        uint32_t count_of_profiles = 0;
        bool isStopped = scanners[i]->stop_dump_recording(count_of_profiles);
        if (isStopped)
            std::cout << "Current profiles in dump: " << count_of_profiles;
        else
            std::cout << "Failed to stop dump recording!";

        // Next steps in get_dumps_profiles example...

    }
}
```

get_dumps_profiles

Прототип: *std::vector<std::shared_ptr<profile2D>> get_dumps_profiles(uint32_t index, uint32_t count, uint32_t timeout = 10000);*

Описание: *Метод скачивания дампа из внутренней памяти устройства*

Параметры:

- `index` - *Начальный номер запрашиваемого профиля из дампа*
- `count` - *Количество скачиваемых профилей относительно начального номера* `index`
- `timeout` - *Время ожидания скачивания дампа.*

Возвращаемое значение: *Вектор профилей*

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief get_dumps_profiles - getting the content of the profile dump
 *
 * @param index Start number of the requested profile from memory
 * @param count The count of requested profiles
 * @param timeout Waiting time for dump download
 *
 * @return Vector profiles
 */
std::vector<std::shared_ptr<profile2D>> get_dumps_profiles(
        uint32_t index, uint32_t count, uint32_t timeout = 10000);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Start dump recording
      uint32_t count_of_profiles = 1000;
      scanners[i]->start_dump_recording(count_of_profiles);

      // Print information about the current dump size
      uint32_t size = 0;
      do {
         scanners[i]->read_params();
         size =scanners[i]->get_param("user_dump_size")->getValue<uint32_t>();
         std::cout << "Current profiles in the dump: " << size << std::endl;
      }while(size < count_of_profiles);

      std::vector<std::shared_ptr<profile2D>> dump =
               list[i]->get_dumps_profiles(0, count_of_profiles);

      std::cout << dump.size() << " Profiles were received! " << std::endl;

      // Next steps with dumps...

   }
}
```

start_profile_capturing

Прототип: *std::vector<std::shared_ptr<profile2D>> get_dumps_profiles(uint32_t index, uint32_t count, uint32_t timeout = 10000);*

Описание: *Метод начала выполнения измерений. Используется только в режиме запуска программного измерения (параметр* `user_sensor_syncSource = "SYNC_SOFTWARE"` *или* `user_sensor_syncSource = "SYNC_SOFTWARE_EXT"` *). При получении команды устройство запускает цикл измерения, после чего выполняется расчет и отправляется стандартный пакет с профилем*

В режиме «программного измерения» метод get_profile2D должен использоваться с параметром `realtime = false` *, чтобы избежать потери запрашиваемых профилей.*

Параметры:

- `count_of_profiles` - *Количество запрашиваемых измерений.*

Возвращаемое значение: `true` *если измерения запущены успешно, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief start_profile_capturing - Command to start profiles measuring.
 * @details This command is used only in the "software measurement" mode:
 * when parameter "user_sensor_syncSource" == "SYNC_SOFTWARE"
 * or "SYNC_SOFTWARE_EXT". Device starts a measurement cycle immediately
 * after receiving this command.
 * ! In "software measurement" mode the get_profile2D method must be used
 * with the realtime == false argument to avoid loss of requested profiles.
 *
 * @param count_of_profiles The count of measurements
 *
 * @return true if measuring was started successfully, else - false
 */
bool start_profile_capturing(uint32_t count_of_profiles = 0);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Select SYNC_SOFTWARE measurement mode
      scanners[i]->set_param_by_key("user_sensor_syncSource", "SYNC_SOFTWARE
↪");

      // Start profile capturing
      uint32_t count_of_profiles = 1000;
      scanners[i]->start_profile_capturing(count_of_profiles);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // realtime must be set to false
        bool realtime = false; /// It is important!
        bool zero_points = true;
        // Get profile from scanner
        std::shared_ptr<profile2D> profile =
           scanner[i]->get_profile2D(zero_points, realtime);

        // some actions with profiles...
    }
}
```

reboot_device

Прототип: *bool reboot_device(PROTOCOLS protocol = PROTOCOLS::CURRENT)*

Описание: *Метод перезагрузки устройства*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда на перезагрузку устройства (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief reboot_device - The scanner will restart
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool reboot_device(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

    // Initialize sdk library
    sdk_init();

    // Search for rf627smart devices over network
    std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Start device reboot
      scanners[i]->reboot_device();

      // Waiting 10 sec
      std::this_thread::sleep_for(std::chrono::seconds(10));

      bool isAvailable = scanners[i]->check_connection();
      if (isAvailable){
         std::cout << "Scanner has been successfully restarted" << std::endl;
      }

      // some other actions with scanner...

   }
}
```

reboot_sensor

Прототип: *bool reboot_sensor(PROTOCOLS protocol = PROTOCOLS::CURRENT)*

Описание: *Метод переинициализация CMOS-сенсора устройства, необходимо использовать, если сенсор «завис» при воздействии внешней помехи.*

Параметры:

- `protocol` - *Тип протокола, по которому будет отправлена команда на переинициализацию CMOS-сенсора устройства (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief reboot_sensor - Reboot CMOS-sensor
 *
 * @param protocol Protocol's type (Service Protocol, ENIP, Modbus-TCP)
 *
 * @return true on success, else - false
 */
bool reboot_sensor(PROTOCOLS protocol = PROTOCOLS::CURRENT);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Reboot CMOS-sensor
      scanners[i]->reboot_sensor();

      // Waiting 100 milliseconds
      std::this_thread::sleep_for(std::chrono::milliseconds(100));

      // some other actions with scanner...

   }
}
```

send_to_periphery

Прототип: *bool send_to_periphery(std::string iface_name, std::vector<char> in, std::vector<char>& out, uint32_t timeout)*

Описание: *Метод отправки данных к периферийному устройству. Под периферией понимаются «внешние» по отношению к сканеру устройства, например: драйверы шаговых двигателей, пневматические клапаны, устройства термостатирования и т.д. Периферия подключается по стандартным интерфейсам, например USART, I2C и т.д. Команды работы с периферией обеспечивают «прозрачный» обмен, не добавляя и не убирая никаких данных.*

Параметры:

- `iface_name` - *Имя интерфейса, на который будут отправляться данные. Если интерфейс не существует, будет возвращена ошибка «RF_WRONG_ARGUMENT». Поддерживаемые значения: «Usart0» - отправка на периферию, подключенную через USART0.*
- `in` - *Данные для отправки. Если данных для отправки нет, будет возвращена ошибка «RF_NO_DATA». Ограничение длины: 1024 байта. Если длина данных превышает лимит, будет возвращена ошибка «RF_OUT_OF_BOUNDS».*
- `out` - *Данные, которые были получены.*
- `timeout` - *Таймаут ответа в мс. Если таймаут равен 0, будут возвращены данные, уже находящиеся в буфере устройства.*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief send_to_periphery - sending data to a peripheral device.
 * @details Peripherals are devices "external" to the scanner,
 * for example: stepper motors, pneumatic valves, thermostats, etc.
 * Peripherals are connected via standard interfaces such as USART,
 * I2C, etc. Peripheral commands provide transparent exchange without
 * adding or removing any data.
 *
 * @param iface_name The name of the interface where the
 * data will be sent. If the interface does not exist,
 * the error "RF_WRONG_ARGUMENT" will be returned.
 * Supported values:
 * "Usart0" - sending to the peripherals connected via the USART0.
 * @param in The data to be sent. If there is no data to send,
 * then the error "RF_NO_DATA" will be returned.
 * Length limit: 1024 bytes. If the data length is greater than
 * the limit, the error "RF_OUT_OF_BOUNDS" will be returned.
 * @param out The data that was received.
 * @param timeout Response timeout in ms. If the timeout is 0,
 * the data already in the device buffer will be returned.
 *
 * @return true on success, else - false
 */
bool send_to_periphery(
        std::string iface_name, std::vector<char> in,
        std::vector<char>& out, uint32_t timeout);

--------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Send data to periphery
      std::vector<char> in {'H', 'E', 'L', 'L', 'O'};
      std::vector<char> out;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        bool isSent = scanners[i]->send_to_periphery("Usart0", in, out, 1000);
        if (isSent){
            std::cout << "The data was sent successfully." << std::endl;
            std::cout << "Size of received data: " << out.size() << std::endl;
        }

        // some other actions with scanner...

    }
}
```

receive_from_periphery

Прототип: *bool receive_from_periphery(std::string iface_name, uint16_t count, std::vector<char>& out, uint32_t timeout)*

Описание: *Метод приема данных от периферийного устройства.*

Параметры:

- `iface_name` - *Имя интерфейса с которого будут прочитаны данные. Если интерфейс не существует, будет возвращена ошибка «RF_WRONG_ARGUMENT». Поддерживаемые значения: «Usart0» - получение с периферии, подключенную через USART0.*
- `count` - *Количество байт, которые ожидается принять. Если* `count` *равен 0, то будет возвращена ошибка «RF_NO_DATA». Ограничение длины: 1024 байта. Если длина данных для разового получения превышает лимит, будет возвращена ошибка «RF_OUT_OF_BOUNDS».*
- `out` - *Данные, которые были получены.*
- `timeout` - *Таймаут ответа в мс. Если требуемое количество байт не было принято по истечении данного времени, то будет возвращено принятое количество байт. Если таймаут равен 0, будут возвращены данные, уже находящиеся в буфере устройства.*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/** @file rf62Xsdk.h */

/**
 * @brief receive_from_periphery - receiving data from a
 * peripheral device
 *
 * @param iface_name The name of the interface from which the data
 * will be received. If the interface does not exist, the error
 * "RF_WRONG_ARGUMENT" will be returned.
 * Supported values:
 * "Usart0" - sending to the peripherals connected via the USART0.
 * @param count The number of bytes expected to be received
 * @param out The data that was received.
 * @param timeout Response timeout in ms. If the requested number of
 * bytes is not received after this time, will be returned current
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 * number of bytes.
 *
 * @return true on success, else - false
 */
bool receive_from_periphery(
        std::string iface_name, uint16_t count,
        std::vector<char>& out, uint32_t timeout);

-------------------------------------------------------------------------------

/** @file main.cpp */

#include <string>
#include <iostream>
#include <chrono>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

int main()
{

   // Initialize sdk library
   sdk_init();

   // Search for rf627smart devices over network
   std::vector<std::shared_ptr<rf627smart>> scanners = rf627smart::search();

   for (size_t i = 0; i < scanners.size(); i++)
   {
      scanners[i]->connect();

      // Receive data from periphery
      std::vector<char> out;
      scanners[i]->receive_from_periphery("Usart0", 512, out, 1000);

      std::cout << "Size of received data: " << out.size() << std::endl;

      // some other actions with scanner...

   }
}
```

4.3 Интерфейс «обёртки» на C#

Эта «обёртка» представляет собой библиотеку .NET, написанную на языке C#, которая позволяет упростить разработку приложений на языках C#, Visual Basic .NET, C++/CLI и JScript .NET

Для её использования в проектах .NET разработчику необходимо собрать или скачать динамическую программную библиотеку **RF62X-SDK.dll**, после чего подключить библиотеку в проект (**Project > Add References**), а также добавить дополнительные зависимости в каталог к исполняемому файлу проекта.

Для скачивания библиотеки см. Последние выпуски RF62X-SDK библиотек.
Для компиляции библиотеки из исходников см. *Компиляция «обёртки» на C#*.
Для использования в проектах см. *Создание проекта C#*.

4.3.1 Инициализация SDK

Файл `RF62X-SDK.cs` является основным файлом программного интерфейса (API) и определяет функциональность библиотеки-«обёртки». `RF62X-TYPES.cs` содержит дополнительный набор классов, структур, типов и перечислений используемых в SDK.

SdkInit

Прототип: *bool SdkInit();*

Описание: *Функция инициализации SDK. Должна быть вызвана один раз перед дальнейшими вызовами любых библиотечных функций*

Возвращаемое значение: *После успешного завершения возвращается true. В противном случае должен быть возвращен false.*

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// SdkInit - Initialize sdk library
/// </summary>
/// <remarks>
/// Must be called once before further calls to any library functions
/// </remarks>
/// <returns>true if success.</returns>
public static bool SdkInit();

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         bool isInit = RF62X.SdkInit();

         if (isInit)
            Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
         else
         {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            Console.WriteLine("SDK initialization error!");
        }

        // some code...
    }
  }
}
```

SdkCleanup

Прототип: *void SdkCleanup();*

Описание: *Функция высвобождает ресурсы, выделенные при инициализации СДК функцией SdkInit*

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// SdkCleanup - Cleanup resources allocated with sdk_init() function
/// </summary>
public static void SdkCleanup();

------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
  class Program
  {
    static void Main(string[] args)
    {
      // Initialize sdk library
      bool isInit = RF62X.SdkInit();

      if (isInit)
        Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
      else
      {
        Console.WriteLine("SDK initialization error!");
      }

      // some code...

      // Cleanup resources
      SdkCleanup();
    }
  }
}
```

SdkVersion

Прототип: *string SdkVersion();*

Описание: *Функция получения информации о версии SDK*

Возвращаемое значение: *версия SDK в формате X.Y.Z (мажорная, минорная, патч)*

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// SdkCleanup - Cleanup resources allocated with sdk_init() function
/// </summary>
public static void SdkCleanup();

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         bool isInit = RF62X.SdkInit();

         if (isInit)
            Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
         else
         {
            Console.WriteLine("SDK initialization error!");
         }

         // some code...

         // Cleanup resources
         RF62X.SdkCleanup();
      }
   }
}
```

4.3.2 Класс rf627smart

Данный класс определён в файле `RF62X-SDK.cs` и предоставляет интерфейс для работы со сканерами серии RF627 v2.x.x

Search

Прототип: *static List<RF627smart> Search(uint timeout = 300, PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод поиска устройств RF62X v2.x.x в сети*

Параметры:

- `timeout` - *Время поиска на каждом Ethernet интерфейсе (мс).*
- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: *Список обнаруженных в сети сканеров серии RF627 v2.x.x.

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Search for RF627smart devices over network
/// </summary>
/// <param name="timeout">Search timeout for each Ethernet interface</param>
/// <param name="protocol">Protocol's type</param>
/// <returns>List of RF627smart devices</returns>
static List<RF627smart> Search(
      uint timeout = 300, PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         // Print count of discovered rf627smart in network
         Console.WriteLine("Was found\t: {0} RF627-Smart", list.Count);
         Console.WriteLine("=======================================");

         // some code...

         // Cleanup resources allocated with SdkInit()
         RF62X.SdkCleanup();
      }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }
}
```

GetInfo

Прототип: *HelloInfo GetInfo(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод получения информации о сканере из пакета приветствия (Hello-пакет)*

Параметры:

- `protocol` - *Тип протокола, по которому будет осуществляться поиск (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `HelloInfo` *в случае успеха*

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Get information about scanner from hello packet
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>Hello_info on success</returns>
HelloInfo GetInfo(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            RF62X.HelloInfo info = list[i].GetInfo();

            Console.WriteLine("\n\nID scanner's list: {0}", i);
            Console.WriteLine("----------------------------------------");
            Console.WriteLine("Device information: ");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            Console.WriteLine("* Name\t: {0}", info.device_name);
            Console.WriteLine("* Serial\t: {0}", info.serial_number);
            Console.WriteLine("* IP Addr\t: {0}", info.ip_address);
            Console.WriteLine("* MAC Addr\t: {0}", info.mac_address);

            Console.WriteLine("\nWorking ranges: ");
            Console.WriteLine("* Zsmr, mm\t: {0}", info.z_smr);
            Console.WriteLine("* Zmr , mm\t: {0}", info.z_mr);
            Console.WriteLine("* Xsmr, mm\t: {0}", info.x_smr);
            Console.WriteLine("* Xemr, mm\t: {0}", info.x_emr);

            Console.WriteLine("\nVersions: ");
            Console.WriteLine("* Firmware\t: {0}", info.firmware_version);
            Console.WriteLine("* Hardware\t: {0}", info.hardware_version);
            Console.WriteLine("----------------------------------------");
        }
    }
  }
}
```

Connect

Прототип: *bool Connect(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод установки соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено подключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Establish connection to the RF627smart device
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool Connect(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
  class Program
  {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        static void Main(string[] args)
        {
            // Initialize sdk library
            RF62X.SdkInit();

            // Search for RF627smart devices over network
            List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

            for (int i = 0; i < list.Count; i++)
            {
                // Establish connection to the RF627 device.
                bool isConnected = list[i].Connect();
                if (!isConnected){
                    Console.WriteLine("Failed to connect to scanner!");
                    continue;
                }

                // some actions with scanner...
            }
        }
    }
}
```

Disconnect

Прототип: *bool Disconnect(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод закрытия ранее установленного соединения со сканером*

Параметры:

- `protocol` - *Тип протокола, по которому будет выполнено отключение (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Close connection to the device
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool Disconnect(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE)

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
```

(continues on next page)

(продолжение с предыдущей страницы)

```
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            // Establish connection to the RF627 device.
            bool isConnected = list[i].Connect();
            if (!isConnected){
               Console.WriteLine("Failed to connect to scanner!");
               continue;
            }

            // some actions with scanner...

            // Disconnect from scanner.
            list[i].Disconnect();
         }
      }
   }
}
```

CheckConnection

Прототип: *bool CheckConnection(uint timeout = 300, PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод проверки доступности сканера в сети (после подключения к нему)*

Параметры:

- `timeout` - *Время проверки соединения со сканером (мс).*
- `protocol` - *Тип протокола, по которому будет выполнена проверка (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Check connection with RF627smart device
/// </summary>
/// <param name="timeout">Connection check timeout</param>
/// <param name="protocol">Protocol's type</param>
/// <returns>true on success</returns>
bool CheckConnection(
      uint timeout = 300, PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            // Establish connection to the RF627 device.
            bool isConnected = list[i].Connect();
            if (!isConnected){
               Console.WriteLine("Failed to connect to scanner!");
               continue;
            }

            // Check connection with device
            bool isAvailable = list[i].CheckConnection(500);
            if (!isAvailable){
               Console.WriteLine("Scanner is not available now.");
               Console.WriteLine("Please call back later!");
               continue;
            }

            // some actions with scanner...
         }
      }
   }
}
```

GetProfile

Прототип: *Profile2D GetProfile(bool zero_points = true, bool realtime = true, PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод получения результатов измерений*

Параметры:

- `zero_points` - *Включать нулевые точки в возвращаемом профиле.*
- `realtime` - *Получение профиля в реальном времени (буферизация отключена).*

- `protocol` - *Тип протокола, по которому будет получен профиль (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `Profile2D` *при успехе, иначе -* `null`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Get 2D measurement from scanner's data stream
/// </summary>
/// <param name="zero_points">include zero points in return Profile</param>
/// <param name="realtime">Enable getting profile in real time</param>
/// <param name="protocol">protocol's type</param>
/// <returns>Profile</returns>
Profile2D GetProfile(
      bool zero_points = true, bool realtime = true,
      PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            // Establish connection to the RF627 device.
            list[i].Connect();

            // Get profile from scanner's data stream by Service Protocol.
            RF62X.Profile2D profile = null;
            bool zero_points = true;
            bool realtime = true;

            // Get profile from scanner
            profile = list[i].GetProfile(zero_points, realtime);
            if (profile != null) {
               Console.WriteLine("Profile was successfully received!");
            }else
               Console.WriteLine("Profile was not received!");
         }
      }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }
}
```

GetFrame

Прототип: *Frame GetFrame(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод получения кадров видео с матрицы сканера*

Параметры:

- `protocol` - *Тип протокола, по которому будет получен кадр (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `Frame` *при успехе, иначе -* `null`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Get RAW frame from scanner
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>Frame</returns>
Frame GetFrame(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            // Establish connection to the RF627 device.
            list[i].Connect();

            // Get Frame from scanner.
            RF62X.Frame frame = list[i].GetFrame();
            if (frame != null) {
               Console.WriteLine("Frame was successfully received!");
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            }else
               Console.WriteLine("Frame was not received!");
         }
      }
   }
}
```

ReadParams

Прототип: *bool ReadParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод получения текущих параметров сканера. При вызове данного метода SDK вычитывает со сканера все актуальные параметры, сохраняя их в виде «списка параметров» для дальнейшей работы во внутренней памяти SDK.*

Параметры:

- `protocol` - *Тип протокола, по которому будут прочитаны параметры (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Read parameters from device to internal structure.
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool ReadParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE)

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            // Establish connection to the RF627 device.
            list[i].Connect();

            // Read params from RF627 device.
            bool isRead = list[i].ReadParams();
            if (isRead) {
               Console.WriteLine("Scanner parameters were read!");
            }else
               Console.WriteLine("Scanner parameters were not read!");
         }
      }
   }
}
```

GetParam

Прототип: *dynamic GetParam(string nameKey);*

Описание: *Метод получения конкретного параметра по его имени (ключу). При вызове данного метода SDK осуществляет поиск нужного параметра из последних прочитанных при вызове метода ReadParams . В случае, если запрашиваемый параметр отсутствует в конкретном сканере, функция вернёт null.*

Параметры:

- `param_name` - *Имя (ключ) параметра.*

Возвращаемое значение: `Parameter` *при успехе, иначе -* `null`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Search parameters by his name
/// </summary>
/// <param name="nameKey">name of parameter</param>
/// <returns>param on success, else - null</returns>
dynamic GetParam(string nameKey);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            RF62X.SdkInit();

            // Search for RF627smart devices over network
            List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

            for (int i = 0; i < list.Count; i++)
            {
                // Establish connection to the RF627 device.
                list[i].Connect();

                // Read params from RF627 device.
                list[i].ReadParams();

                RF62X.Parameter<string> name =
                      list[i].GetParam("user_general_deviceName");
                if (name != null) {
                    string str_name = name.GetValue();
                    Console.WriteLine("Current Device Name \t: {0}", str_name);
                }

                RF62X.Parameter<uint> framerate =
                      list[i].GetParam("user_sensor_framerate");
                if (framerate != null) {
                    uint framerate_value = framerate.GetValue();
                    Console.WriteLine("Current FPS\t: {0}", framerate_value);
                }
            }
        }
    }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

SetParam

Прототип: *bool SetParam(string name, dynamic value);*

Описание: *Метод установки конкретного параметра. При вызове данного метода происходит* установка параметра в списке параметров во внутренней памяти SDK.* *Для отправки изменений в сканер необходимо вызвать метод WriteParams .*

Параметры:

- `param_name` - *Имя (ключ) параметра.*
- `value` *- Новое значение параметра

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
```

(continues on next page)

(продолжение с предыдущей страницы)

```
/// Set parameter by name
/// </summary>
/// <param name="param">Name of parameter</param>
/// <param name="value">Value to set</param>
/// <returns>true on success, else - false</returns>
bool SetParam(string name, dynamic value)

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            // Establish connection to the RF627 device.
            list[i].Connect();

            // Read params from RF627 device.
            list[i].ReadParams();

            // Set parameter of Device Name
            list[i].SetParam("user_general_deviceName", "RF627 New Name");

            // Set parameter of Sensor Framerate
            list[i].SetParam("user_sensor_framerate", 100);

            // some actions with other parameters...

         }
      }
   }
}
```

Примечание: Для более детального описания каждого параметра и его свойств см. RF62X Firmware Cloud

WriteParams

Прототип: *bool WriteParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод передачи параметров из внутренней памяти SDK в сканер. При вызове данного метода происходит отправка изменённых параметров в сканер*

Параметры:

- `protocol` - *Тип протокола, по которому будут отправлена команда на установку параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Send current parameters to device
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool WriteParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE)

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            list[i].Connect();
            list[i].ReadParams();

            // Set parameter of Device Name
            list[i].SetParam("user_general_deviceName", "RF627 New Name");
            // Set parameter of Sensor Framerate
            list[i].SetParam("user_sensor_framerate", 100);

            // some actions with other parameters...

            // Apply changed parameters to the device
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            bool isApplied = list[i].WriteParams();
            if (isApplied)
               Console.WriteLine("Scanner parameters were applied!");
            else
               Console.WriteLine("Scanner parameters were not applied!");
         }
      }
   }
}
```

SaveParams

Прототип: *bool SaveParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод сохранения параметров сканер во внутреннюю память устройства. Сохраненные параметры также будут использоваться после перезапуске устройства или после смены(обновления) прошивки.*

Параметры:

- `protocol` - *Тип протокола, по которому будут отправлена команда на установку параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Save changes to device's memory.
/// The saved parameters will also be used if the device
/// is restarted or even if the firmware is updated.
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool SaveParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

--------------------------------------------------------------------------------

/// <file> Program.cs

using System;
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            // Search for RF627smart devices over network
            List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

            for (int i = 0; i < list.Count; i++)
            {
                list[i].Connect();
                list[i].ReadParams();

                // After changing some parameters...

                // Apply changed parameters to the device
                list[i].WriteParams();

                // Save current parameters in the device memory
                bool isSaved = list[i].SaveParams();
                if (isSaved)
                    Console.WriteLine("Scanner parameters were applied!");
                else
                    Console.WriteLine("Scanner parameters were not applied!");
            }
        }
    }
}
```

LoadRecoveryParams

Прототип: *bool SaveParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);*

Описание: *Метод загрузки значений параметров устройства из области восстановления. Загруженные значения будут записаны в пользовательскую область*

Параметры:

- `protocol` - *Тип протокола, по которому будут отправлена команда на установку параметров (Service Protocol, ENIP, Modbus-TCP)*

Возвращаемое значение: `true` *при успехе, иначе -* `false`

Пример в коде:

```
/// <file> RF62X-SDK.cs

/// <summary>
/// Loading parameters from recovery area.
/// The device will automatically reboot.
/// </summary>
/// <param name="protocol">protocol's type</param>
/// <returns>true on success</returns>
bool LoadRecoveryParams(PROTOCOL_TYPES protocol = PROTOCOL_TYPES.SERVICE);

------------------------------------------------------------------------------

/// <file> Program.cs

using System;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
using System.Collections.Generic;
using System.Threading;
using SDK.SCANNERS;

namespace EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         // Initialize sdk library
         RF62X.SdkInit();

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         for (int i = 0; i < list.Count; i++)
         {
            list[i].Connect();

            // Load parameters from recovery area
            bool isLoaded = list[i].LoadRecoveryParams();
            if (isLoaded)
               Console.WriteLine("Recovery parameters loaded successfully!");
            else
               Console.WriteLine("Recovery parameters were not loaded!");
         }
      }
   }
}
```

ГЛАВА 5

Примеры использования

5.1 Примеры для C

5.1.1 Поиск устройств RF62X

Ниже приведен пример поиска сканеров серии RF627 v2.x.x:

```
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   printf("########################################\n");
   printf("#                                      #\n");
   printf("#         Search Example v2.x.x        #\n");
   printf("#                                      #\n");
   printf("########################################\n");

   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
   printf("========================================\n");


   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
// Iterate over all available network adapters in the current operating
// system to send "Hello" requests.
uint32_t count = 0;
for (int i=0; i<GetAdaptersCount(); i++)
{
   uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
   uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
   // call the function to change adapter settings inside the library.
   set_platform_adapter_settings(host_mask, host_ip_addr);

   // Search for rf627old devices over network by Service Protocol.
   if (host_ip_addr != 0)
   {
      // Get another IP Addr and set this changes in adapter settings.
      printf("Search scanners from:\n "
             "* IP Address\t: %s\n "
             "* Netmask\t: %s\n",
             GetAdapterAddress(i), GetAdapterMasks(i));
      search_scanners(scanners, kRF627_SMART, 300, kSERVICE);

      // Print count of discovered rf627old in network
      printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
      printf("-----------------------------------------\n");
      count = (int)vector_count(scanners);
   }
}

// Print count of discovered rf627smart in network
printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));


for (int i = 0; i < (int)vector_count(scanners); i++)
{
   hello_information result =
            get_info_about_scanner(vector_get(scanners,i), kSERVICE);

   rf627_smart_hello_info_by_service_protocol* info =
            result.rf627smart.hello_info_service_protocol;

   printf("\n\nID scanner in list: %d\n", i);
   printf("-----------------------------------------\n");
   printf("Device information: \n");
   printf("* Name  \t: %s\n", info->user_general_deviceName);
   printf("* Serial\t: %d\n", info->fact_general_serial);
   printf("* IP Addr\t: %s\n", info->user_network_ip);
   printf("* MAC Addr\t: %s\n", info->fact_network_macAddr);

   printf("\nWorking ranges: \n");
   printf("* Zsmr, mm\t: %d\n", info->fact_general_smr);
   printf("* Zmr , mm\t: %d\n", info->fact_general_mr);
   printf("* Xsmr, mm\t: %d\n", info->fact_general_xsmr);
   printf("* Xemr, mm\t: %d\n", info->fact_general_xemr);

   printf("\nVersions: \n");
   printf("* Firmware\t: %d.%d.%d\n",
          info->fact_general_firmwareVer[0],
          info->fact_general_firmwareVer[1],
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            info->fact_general_firmwareVer[2]);
      printf("* Hardware\t: %d\n", info->fact_general_hardwareVer);
      printf("-----------------------------------------\n");
   }

   core_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном обнаружении сканера в сети:

```
#########################################
#                                       #
#         Search Example v2.x.x         #
#                                       #
#########################################
SDK version: 2.17.2
=========================================
Search scanners from:
* IP Address    : 192.168.1.2
* Netmask       : 255.255.255.0
Discovered      : 1 RF627
-----------------------------------------
Search scanners from:
* IP Address    : 192.168.2.104
* Netmask       : 255.255.255.0
Discovered      : 0 RF627
-----------------------------------------
Was found       : 1 RF627 v2.x.x

ID scanner in list: 0
-----------------------------------------
Device information:
* Name           : RF627 scanner
* Serial         : 190068
* IP Addr        : 192.168.1.30
* MAC Addr       : 00:0A:35:6E:07:F5

Working ranges:
* Zsmr, mm       : 70
* Zmr , mm       : 50
* Xsmr, mm       : 30
* Xemr, mm       : 42

Versions:
* Firmware       : 2.7.1
* Hardware       : 302388224
-----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/C/RF627_SMART/SEARCH_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.1.2 Получение профиля сканера

Ниже приведен пример получение профиля со сканера серии RF627 v2.x.x:

```
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   printf("########################################\n");
   printf("#                                      #\n");
   printf("#        Profile Example v2.x.x        #\n");
   printf("#                                      #\n");
   printf("########################################\n");

   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
   printf("========================================\n");


   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);

   // Iterate over all available network adapters in the current operating
   // system to send "Hello" requests.
   uint32_t count = 0;
   for (int i=0; i<GetAdaptersCount(); i++)
   {
      uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
      uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
      // call the function to change adapter settings inside the library.
      set_platform_adapter_settings(host_mask, host_ip_addr);

      // Search for rf627old devices over network by Service Protocol.
      if (host_ip_addr != 0)
      {
         // Get another IP Addr and set this changes in adapter settings.
         printf("Search scanners from:\n "
                "* IP Address\t: %s\n "
                "* Netmask\t: %s\n",
                GetAdapterAddress(i), GetAdapterMasks(i));
         search_scanners(scanners, kRF627_SMART, 300, kSERVICE);

         // Print count of discovered rf627old in network
         printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
         printf("----------------------------------------\n");
         count = (int)vector_count(scanners);
      }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
}

// Print count of discovered rf627smart in network
printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));


for (int i = 0; i < (int)vector_count(scanners); i++)
{
   scanner_base_t* scanner = vector_get(scanners,i);

   hello_information _info = get_info_about_scanner(scanner, kSERVICE);

   rf627_smart_hello_info_by_service_protocol* info =
            _info.rf627smart.hello_info_service_protocol;

   printf("\n\nID scanner in list: %d\n", i);
   printf("----------------------------------------\n");
   printf("Device information: \n");
   printf("* Name\t\t: %s\n", info->user_general_deviceName);
   printf("* Serial\t: %d\n", info->fact_general_serial);
   printf("* IP Addr\t: %s\n", info->user_network_ip);

   // Establish connection to the RF627 device
   uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
   if (!is_connected)
      continue;

   // Read params from RF627 device
   uint8_t is_read = read_params_from_scanner(scanner, 3000, kSERVICE);
   if (!is_read)
      continue;

   uint8_t zero_points = TRUE;
   uint8_t realtime = TRUE;
   // Get profile from scanner's data stream by Service Protocol.
   rf627_profile2D_t* result = get_profile2D_from_scanner(
               scanner, zero_points, realtime, kSERVICE);
   rf627_smart_profile2D_t* profile2D = result->rf627smart_profile2D;
   if (profile2D != NULL)
   {
      printf("Profile information: \n");
      switch (profile2D->header.data_type)
      {
      case (int)DTY_PixelsNormal:
      {
          printf("* DataType\t: PIXELS\n");
          uint32_t count = profile2D->pixels_format.pixels_count;
          printf("* Count\t\t: %d\n", count);
          break;
      }
      case (int)DTY_PixelsInterpolated:
      {
          printf("* DataType\t: PIXELS_INTRP\n");
          uint32_t count = profile2D->pixels_format.pixels_count;
          printf("* Count\t\t: %d\n", count);
          break;
      }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        case (int)DTY_ProfileNormal:
        {
            printf("* DataType\t: PROFILE\n");
            uint32_t count = profile2D->profile_format.points_count;
            printf("* Count\t\t: %d\n", count);
            break;
        }
        case (int)DTY_ProfileInterpolated:
        {
            printf("* DataType\t: PROFILE_INTRP\n");
            uint32_t count = profile2D->profile_format.points_count;
            printf("* Count\t\t: %d\n", count);
            break;
        }
        }
        printf("Profile was successfully received!\n");
        printf("----------------------------------------\n");
        free_profile2D(result);
      }else
      {
        printf("Profile was not received!\n");
        printf("----------------------------------------\n");
      }

      disconnect_from_scanner(scanner, kSERVICE);
   }

   // Cleanup resources allocated with core_init()
   core_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном получении профиля:

```
########################################
#                                      #
#        Profile Example v2.x.x        #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address    : 192.168.1.2
* Netmask       : 255.255.255.0
Discovered      : 1 RF627
----------------------------------------
Search scanners from:
* IP Address    : 192.168.2.104
* Netmask       : 255.255.255.0
Discovered      : 0 RF627
----------------------------------------
Was found       : 1 RF627 v2.x.x

ID scanner in list: 0
----------------------------------------
Device information:
* Name           : RF627 scanner
* Serial         : 190068
```

(continues on next page)

(продолжение с предыдущей страницы)

```
* IP Addr       : 192.168.1.30
Profile information:
* DataType      : PROFILE
* Count         : 648
Profile was successfully received!
---------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/C/RF627_SMART/PROFILE_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.1.3 Получение кадра матрицы

Ниже приведен пример получение кадра матрицы со сканера серии RF627 v2.x.x:

```
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   printf("########################################\n");
   printf("#                                      #\n");
   printf("#         Frame Example v2.x.x         #\n");
   printf("#                                      #\n");
   printf("########################################\n");

   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
   printf("========================================\n");


   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);

   // Iterate over all available network adapters in the current operating
   // system to send "Hello" requests.
   uint32_t count = 0;
   for (int i=0; i<GetAdaptersCount(); i++)
   {
      uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
        // call the function to change adapter settings inside the library.
        set_platform_adapter_settings(host_mask, host_ip_addr);

        // Search for rf627old devices over network by Service Protocol.
        if (host_ip_addr != 0)
        {
            // Get another IP Addr and set this changes in adapter settings.
            printf("Search scanners from:\n "
                   "* IP Address\t: %s\n "
                   "* Netmask\t: %s\n",
                   GetAdapterAddress(i), GetAdapterMasks(i));
            search_scanners(scanners, kRF627_SMART, 300, kSERVICE);

            // Print count of discovered rf627old in network
            printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
            printf("----------------------------------------\n");
            count = (int)vector_count(scanners);
        }
    }

    // Print count of discovered rf627smart in network
    printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));


    for (int i = 0; i < (int)vector_count(scanners); i++)
    {
        scanner_base_t* scanner = vector_get(scanners,i);

        hello_information _info = get_info_about_scanner(scanner, kSERVICE);

        rf627_smart_hello_info_by_service_protocol* info =
                _info.rf627smart.hello_info_service_protocol;

        printf("\n\nID scanner in list: %d\n", i);
        printf("----------------------------------------\n");
        printf("Device information: \n");
        printf("* Name\t\t: %s\n", info->user_general_deviceName);
        printf("* Serial\t: %d\n", info->fact_general_serial);
        printf("* IP Addr\t: %s\n", info->user_network_ip);

        // Establish connection to the RF627 device
        uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
        if (!is_connected)
            continue;

        // Read params from RF627 device
        uint8_t is_read = read_params_from_scanner(scanner, 3000, kSERVICE);
        if (!is_read)
            continue;

        rf627_frame_t* _frame = get_frame_from_scanner(scanner, kSERVICE);
        if (_frame != NULL && _frame->rf627smart_frame != NULL)
        {
            uint32_t data_size = _frame->rf627smart_frame->data_size;
            uint32_t frame_width = _frame->rf627smart_frame->fact_sensor_width;
            uint32_t frame_height = _frame->rf627smart_frame->fact_sensor_height;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        printf("Frame information: \n");
        printf("* Data Size\t: %d\n", data_size);
        printf("* Frame Width\t: %d\n", frame_width);
        printf("* Frame Height\t: %d\n", frame_height);
        printf("Frame was successfully received!\n");
        printf("----------------------------------------\n");

        free_frame(_frame);
    }

    disconnect_from_scanner(scanner, kSERVICE);
  }

  // Cleanup resources allocated with core_init()
  core_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном получении кадра:

```
########################################
#                                      #
#         Frame Example v2.x.x         #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627
----------------------------------------
Search scanners from:
* IP Address   : 192.168.2.104
* Netmask      : 255.255.255.0
Discovered     : 0 RF627
----------------------------------------
Was found      : 1 RF627 v2.x.x

ID scanner in list: 0
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
Frame information:
* Data Size     : 316224
* Frame Width   : 648
* Frame Height  : 488
Frame was successfully received!
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/C/RF627_SMART/FRAME_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)

- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.1.4 Получение и установка параметров

Ниже приведен пример получения и изменения имени сканера и смены состояния лазера (включение/выключение):

```
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   printf("########################################\n");
   printf("#                                      #\n");
   printf("#       Parameter Example v2.x.x       #\n");
   printf("#                                      #\n");
   printf("########################################\n");

   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
   printf("========================================\n");


   // Create value for scanners vector's type
   vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
   // Initialization vector
   vector_init(&scanners);

   // Iterate over all available network adapters in the current operating
   // system to send "Hello" requests.
   uint32_t count = 0;
   for (int i=0; i<GetAdaptersCount(); i++)
   {
      uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
      uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
      // call the function to change adapter settings inside the library.
      set_platform_adapter_settings(host_mask, host_ip_addr);

      // Search for rf627old devices over network by Service Protocol.
      if (host_ip_addr != 0)
      {
         // Get another IP Addr and set this changes in adapter settings.
         printf("Search scanners from:\n "
                "* IP Address\t: %s\n "
                "* Netmask\t: %s\n",
                GetAdapterAddress(i), GetAdapterMasks(i));
         search_scanners(scanners, kRF627_SMART, 300, kSERVICE);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Print count of discovered rf627old in network
        printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
        printf("----------------------------------------\n");
        count = (int)vector_count(scanners);
    }
}

// Print count of discovered rf627smart in network
printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));


for (int i = 0; i < (int)vector_count(scanners); i++)
{
    scanner_base_t* scanner = vector_get(scanners,i);
    printf("\n\nID scanner in list: %d\n", i);
    printf("----------------------------------------\n");

    // Establish connection to the RF627 device
    uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
    if (!is_connected)
        continue;

    // Read params from RF627 device
    uint8_t is_read = read_params_from_scanner(scanner, 3000, kSERVICE);
    if (!is_read)
        continue;

    //
    // Example of working with the parameter type:
    // string
    //
    // Get/Set parameter of Device Name
    parameter_t* name = get_parameter(scanner,"user_general_deviceName");
    if (name != NULL)
    {
        char* value = name->val_str->value;
        printf("Current Device Name\t: %s\n", value);
        char* new_value = "TEST NAME";
        printf("New Device Name\t\t: %s\n", new_value);

        parameter_t* temp = create_parameter_from_type(name->base.type);

        uint32_t name_size = strlen(name->base.name) + 1;
        temp->base.name = platform_calloc(name_size, sizeof (char));
        platform_memcpy(temp->base.name, name->base.name, name_size);

        uint32_t value_size = strlen(new_value) + 1;
        temp->val_str->value = platform_calloc(value_size, sizeof (char));
        platform_memcpy(temp->val_str->value, new_value, value_size);
        temp->base.size = value_size;
        printf("-------------------------------------\n");

        set_parameter(scanner, temp);
        free_parameter(temp, scanner->type);
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      //
      // Example of working with the parameter type:
      // uint32_t
      //
      // Get/Set parameter of Laser Enabled
      parameter_t* laser = get_parameter(scanner, "user_laser_enabled");
      if (laser != NULL && strcmp(
                  laser->base.type, "uint32_t")== 0)
      {
         uint32_t isEnabled = laser->val_uint32->value;
         printf("Current Laser State\t: %s\n", (isEnabled?"ON":"OFF"));

         // Change the current state to the opposite
         isEnabled = !isEnabled;
         laser->val_uint32->value = isEnabled;
         printf("New Laser State\t\t: %s\n", (isEnabled?"ON":"OFF"));
         printf("-----------------------------------\n");

         set_parameter(scanner, laser);
      }

      // Apply changed parameters to the device
      char answer = 'n';
      printf("Apply changed params to the device? (y/n): ");
      scanf("%c", &answer);
      if (answer == 'y' || answer == 'Y')
      {
         write_params_to_scanner(scanner, 3000, kSERVICE);
         // Save changes to the device's memory
         printf("\nSave changes to device's memory? (y/n): ");
         scanf("%c", &answer);
         if (answer == 'y' || answer == 'Y')
               save_params_to_scanner(scanner, 3000, kSERVICE);
      }
   }

   // Cleanup resources allocated with core_init()
   core_cleanup();
}
```

Ниже приведён результат вывода приложения при успешной установке новых параметров:

```
########################################
#                                      #
#       Parameter Example v2.x.x       #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address    : 192.168.1.2
* Netmask       : 255.255.255.0
Discovered      : 1 RF627
----------------------------------------
Search scanners from:
* IP Address    : 192.168.2.104
```

(continues on next page)

(продолжение с предыдущей страницы)

```
* Netmask      : 255.255.255.0
Discovered     : 0 RF627
----------------------------------------
Was found      : 1 RF627 v2.x.x

ID scanner's list: 0
----------------------------------------
Current Device Name     : RF627 scanner
New Device Name         : TEST NAME
----------------------------------------
Current Laser State     : ON
New Laser State         : OFF
----------------------------------------
Apply changed params to the device? (y/n): y
Save changes to device's memory? (y/n): n

Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/C/RF627_SMART/PARAMETER_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.1.5 Запись и скачивание дампа

Ниже приведен пример записи дампа профилей и его скачивание:

```c
#include <stdio.h>
#include <stdlib.h>

#include "network.h"
#include "rf62Xcore.h"
#include "rf62X_sdk.h"
#include "rf62X_types.h"

int main()
{
   printf("##########################################\n");
   printf("#                                        #\n");
   printf("#          Dump Example v2.x.x           #\n");
   printf("#                                        #\n");
   printf("##########################################\n");

   // Initialize sdk library
   core_init();

   // Print return rf627 sdk version
   printf("SDK version: %s\n", sdk_version());
   printf("========================================\n");


   // Create value for scanners vector's type
```

(continues on next page)

(продолжение с предыдущей страницы)

```
vector_t* scanners = (vector_t*)calloc(1, sizeof (vector_t));
// Initialization vector
vector_init(&scanners);

// Iterate over all available network adapters in the current operating
// system to send "Hello" requests.
uint32_t count = 0;
for (int i=0; i<GetAdaptersCount(); i++)
{
   uint32_t host_ip_addr = ntohl(inet_addr(GetAdapterAddress(i)));
   uint32_t host_mask = ntohl(inet_addr(GetAdapterMasks(i)));
   // call the function to change adapter settings inside the library.
   set_platform_adapter_settings(host_mask, host_ip_addr);

   // Search for rf627old devices over network by Service Protocol.
   if (host_ip_addr != 0)
   {
      // Get another IP Addr and set this changes in adapter settings.
      printf("Search scanners from:\n "
             "* IP Address\t: %s\n "
             "* Netmask\t: %s\n",
             GetAdapterAddress(i), GetAdapterMasks(i));
      search_scanners(scanners, kRF627_SMART, 300, kSERVICE);

      // Print count of discovered rf627old in network
      printf("Discovered\t: %d RF627\n",(int)vector_count(scanners)-count);
      printf("----------------------------------------\n");
      count = (int)vector_count(scanners);
   }
}

// Print count of discovered rf627smart in network
printf("Was found\t: %d RF627 v2.x.x", (int)vector_count(scanners));


for (int i = 0; i < (int)vector_count(scanners); i++)
{
   scanner_base_t* scanner = vector_get(scanners,i);
   printf("\n\nID scanner in list: %d\n", i);
   printf("----------------------------------------\n");

   // Establish connection to the RF627 device
   uint8_t is_connected = connect_to_scanner(scanner, kSERVICE);
   if (!is_connected)
      continue;

   // Read params from RF627 device
   uint8_t is_read = read_params_from_scanner(scanner, 3000, kSERVICE);
   if (!is_read)
      continue;

   uint8_t status = FALSE;
   rf627_profile2D_t** dumps = NULL;
   uint32_t profiles_in_dump = 0;

   uint32_t count_of_profiles = 1000;
   // Get parameter of user_dump_capacity
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    parameter_t* capacity = get_parameter(scanner,"user_dump_capacity");
    if (capacity != NULL)
    {
       capacity->val_uint32->value = count_of_profiles;
       set_parameter(scanner, capacity);
       write_params_to_scanner(scanner, 300, kSERVICE);
    }

    // Get parameter of user_dump_enabled
    parameter_t* enabled  = get_parameter(scanner,"user_dump_enabled");
    if (enabled != NULL && strcmp(enabled->base.type,"uint32_t")== 0)
    {
       enabled->val_uint32->value = TRUE;
       set_parameter(scanner, enabled);
       write_params_to_scanner(scanner, 300, kSERVICE);
    }

    printf("Start dump recording...\n");
    printf("----------------------------------------\n");
    uint32_t size = 0;
    // wait dump recording
    do {
       read_params_from_scanner(scanner, 300, kSERVICE);
       size = get_parameter(scanner,"user_dump_size")->val_uint32->value;
       printf("Current profiles in the dump: %d\n", size);
    }while(size < count_of_profiles);
    printf("----------------------------------------\n");

    printf("Start dump downloading...\n");
    // Get parameter of user_dump_enabled
    parameter_t* unit_size =get_parameter(scanner,"fact_dump_unitSize");
    if (unit_size != NULL && strcmp(unit_size->base.type,"uint32_t")== 0)
    {
       dumps = calloc(count_of_profiles, sizeof (rf627_profile2D_t*));
       uint32_t start_index = 0;
       status = get_dumps_profiles_from_scanner(
                   scanner, start_index, count_of_profiles,
                   10000, kSERVICE,
                   dumps, &profiles_in_dump,
                   unit_size->val_uint32->value);
    }

    if (status) {
       printf("%d Profiles in dump were downloaded!\n", profiles_in_dump);
       printf("----------------------------------------\n");
    }else {
       printf("Dump was not received!\n");
       printf("----------------------------------------\n");
    }

    for(uint32_t i = 0; i < profiles_in_dump; i++)
       free_profile2D(dumps[i]);
    free(dumps);
}

// Cleanup resources allocated with core_init()
core_cleanup();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
}
```

Ниже приведён результат вывода приложения при успешной записи и скачивании дампа профилей:

```
########################################
#                                      #
#       Parameter Example v2.x.x       #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627
----------------------------------------
Search scanners from:
* IP Address   : 192.168.2.104
* Netmask      : 255.255.255.0
Discovered     : 0 RF627
----------------------------------------
Was found      : 1 RF627 v2.x.x

ID scanner's list: 0
----------------------------------------
Start dump recording...
----------------------------------------
Current profiles in the dump: 0
Current profiles in the dump: 67
Current profiles in the dump: 205
Current profiles in the dump: 415
Current profiles in the dump: 702
Current profiles in the dump: 921
Current profiles in the dump: 1000
----------------------------------------
Start dump downloading...
1000 Profiles were received!
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/C/RF627_SMART/DUMP_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.2 Примеры для C++

5.2.1 Поиск устройств RF62X

Ниже приведен пример поиска сканеров серии RF627 v2.x.x:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;


int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#         Search Example v2.x.x         #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search(500);

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627 v2.x.x"<< std::endl;
   std::cout << "======================================="     << std::endl;


   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<hello_info> info = list[i]->get_info();

      std::cout << "\n\nID scanner's list: " << i               << std::endl;
      std::cout << "---------------------------------------"  << std::endl;
      std::cout << "Device information: "                      << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()      << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()    << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()       << std::endl;
      std::cout << "* MAC Addr\t: " << info->mac_address()      << std::endl;

      std::cout << "\nWorking ranges: "                         << std::endl;
      std::cout << "* Zsmr, mm\t: " << info->z_smr()            << std::endl;
      std::cout << "* Zmr , mm\t: " << info->z_mr()             << std::endl;
      std::cout << "* Xsmr, mm\t: " << info->x_smr()            << std::endl;
      std::cout << "* Xemr, mm\t: " << info->x_emr()            << std::endl;

      std::cout << "\nVersions: "                               << std::endl;
      std::cout << "* Firmware\t: " << info->firmware_version() << std::endl;
      std::cout << "* Hardware\t: " << info->hardware_version() << std::endl;
      std::cout << "---------------------------------------"  << std::endl;
   }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
   // Cleanup resources allocated with sdk_init()
   sdk_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном обнаружении сканера в сети:

```
########################################
#                                      #
#          Search Example v2.x.x       #
#                                      #
########################################

SDK version: 2.19.0
======================================
Search scanners from:
* IP Address   : 192.168.2.100
* Netmask      : 255.255.255.0
Discovered     : 0 RF627-Smart
--------------------------------------
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627-Smart
--------------------------------------
Was found      : 1 RF627 v2.x.x
======================================


ID scanner's list: 0
--------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
* MAC Addr      : 00:0A:35:6E:07:F5

Working ranges:
* Zsmr, mm      : 70
* Zmr , mm      : 50
* Xsmr, mm      : 30
* Xemr, mm      : 42

Versions:
* Firmware      : 2.7.1
* Hardware      : 18.6.20
--------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/Cpp/RF627_SMART/SEARCH_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.2.2 Получение профиля сканера

Ниже приведен пример получение профиля со сканера серии RF627 v2.x.x:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;


int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#         Profile Example v2.x.x       #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()               << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search();

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627 v2.x.x"<< std::endl;
   std::cout << "======================================="     << std::endl;


   // Iterate over all discovered scanners in network, connect to each of␣
↪them,
   // get a profile and print the main profile-info.
   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];
      std::shared_ptr<hello_info> info = scanner->get_info();

      // Print information about the scanner to which the profile belongs.
      std::cout << "\n\nID scanner's list: " << i              << std::endl;
      std::cout << "---------------------------------------" << std::endl;
      std::cout << "Device information: "                     << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()     << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()   << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()      << std::endl;

      // Establish connection to the RF627 device by Service Protocol.
      bool is_connected = scanner->connect();
      if (!is_connected)
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            continue;

        // Get profile from scanner's data stream by Service Protocol.
        std::shared_ptr<profile2D> profile = nullptr;
        bool zero_points = true;
        bool realtime = true;

        if ((profile=scanner->get_profile2D(zero_points,realtime)))
        {
            std::cout << "Profile information: "                    <<std::endl;
            switch ((PROFILE_DATA_TYPES)profile->getHeader().data_type)
            {
            case PROFILE_DATA_TYPES::PIXELS:
                std::cout<<"* DataType\t: "<<"PIXELS"               <<std::endl;
                std::cout<<"* Count\t: "<<profile->getPixels().size()<<std::endl;
                break;
            case PROFILE_DATA_TYPES::PIXELS_INTRP:
                std::cout<<"* DataType\t: "<<"PIXELS_INTRP"         <<std::endl;
                std::cout<<"* Count\t: "<<profile->getPixels().size()<<std::endl;
                break;
            case PROFILE_DATA_TYPES::PROFILE:
                std::cout<<"* DataType\t: "<<"PROFILE"              <<std::endl;
                std::cout<<"* Size\t: "<<profile->getPoints().size() <<std::endl;
                break;
            case PROFILE_DATA_TYPES::PROFILE_INTRP:
                std::cout<<"* DataType\t: "<<"PROFILE_INTRP"        <<std::endl;
                std::cout<<"* Size\t: "<<profile->getPoints().size() <<std::endl;
                break;
            }
            std::cout << "Profile was successfully received!"      <<std::endl;
            std::cout << "----------------------------------------"<<std::endl;
        }else
        {
            std::cout << "Profile was not received!"               <<std::endl;
            std::cout << "----------------------------------------"<<std::endl;
        }

        // Disconnect from scanner.
        scanner->disconnect();
    }

    // Cleanup resources allocated with sdk_init()
    sdk_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном получении профиля:

```
########################################
#                                      #
#         Profile Example v2.x.x       #
#                                      #
########################################

SDK version: 2.19.0
========================================
Search scanners from:
* IP Address    : 192.168.2.100
```

(continues on next page)

(продолжение с предыдущей страницы)

```
* Netmask      : 255.255.255.0
Discovered     : 0 RF627-Smart
----------------------------------------
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627-Smart
----------------------------------------
Was found      : 1 RF627 v2.x.x
========================================


ID scanner's list: 0
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
* MAC Addr      : 00:0A:35:6E:07:F5
Profile information:
* DataType      : PROFILE
* Size          : 648
Profile was successfully received!
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/Cpp/RF627_SMART/PROFILE_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.2.3 Получение кадра матрицы

Ниже приведен пример получение кадра матрицы со сканера серии RF627 v2.x.x:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;


int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#          Frame Example v2.x.x        #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search();

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627 v2.x.x"<< std::endl;
   std::cout << "======================================="     << std::endl;


   // Iterate over all discovered scanners in network, connect to each of␣
↪them,
   // get a profile and print the main profile-info.
   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];
      std::shared_ptr<hello_info> info = scanner->get_info();

      // Print information about the scanner to which the profile belongs.
      std::cout << "\n\nID scanner's list: " << i              << std::endl;
      std::cout << "---------------------------------------" << std::endl;
      std::cout << "Device information: "                     << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()     << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()   << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()      << std::endl;

      // Establish connection to the RF627 device by Service Protocol.
      bool is_connected = scanner->connect();
      if (!is_connected)
         continue;

      // Get profile from scanner's data stream by Service Protocol.
      std::shared_ptr<profile2D> profile = nullptr;
      bool zero_points = true;
      bool realtime = true;

      std::shared_ptr<frame> frame = nullptr;
      if ((frame = scanner->get_frame()))
      {
         std::cout << "Frame information: "                          << "\n";
         std::cout << "* Data Size\t: " << frame->getDataSize()      << "\n";
         std::cout << "* Frame Height\t: " << frame->getFrameHeight()<< "\n";
         std::cout << "* Frame Width\t: " << frame->getFrameWidth()  << "\n";
         std::cout << "Frame was successfully received!"             << "\n";
         std::cout << "---------------------------------------"     << "\n";
      }else
      {
         std::cout << "Frame was not received!"                      << "\n";
         std::cout << "---------------------------------------"     << "\n";
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        }

        // Disconnect from scanner.
        scanner->disconnect();
    }

    // Cleanup resources allocated with sdk_init()
    sdk_cleanup();
}
```

Ниже приведён результат вывода приложения при успешном получении кадра:

```
########################################
#                                      #
#          Frame Example v2.x.x        #
#                                      #
########################################

SDK version: 2.19.0
========================================
Search scanners from:
* IP Address   : 192.168.2.100
* Netmask      : 255.255.255.0
Discovered     : 0 RF627-Smart
----------------------------------------
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627-Smart
----------------------------------------
Was found      : 1 RF627 v2.x.x
========================================


ID scanner's list: 0
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
* MAC Addr      : 00:0A:35:6E:07:F5
Frame information:
* Data Size     : 316224
* Frame Height  : 488
* Frame Width   : 648
Frame was successfully received!
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/Cpp/RF627_SMART/FRAME_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.2.4 Получение и установка параметров

Ниже приведен пример получения и изменения имени сканера и смены состояния лазера (включение/выключение):

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;


int main()
{
   std::cout << "##########################################"  << std::endl;
   std::cout << "#                                        #"  << std::endl;
   std::cout << "#          Frame Example v2.x.x          #"  << std::endl;
   std::cout << "#                                        #"  << std::endl;
   std::cout << "##########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                 << std::endl;
   std::cout << "========================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search();

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627 v2.x.x"<< std::endl;
   std::cout << "========================================="     << std::endl;


   // Iterate over all discovered scanners in network, connect to each of␣
↪them,
   // get a profile and print the main profile-info.
   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];
      std::shared_ptr<hello_info> info = scanner->get_info();

      // Print information about the scanner to which the profile belongs.
      std::cout << "\n\nID scanner's list: " << i              << std::endl;
      std::cout << "-----------------------------------------" << std::endl;
      std::cout << "Device information: "                      << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()     << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()   << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()      << std::endl;

      // Establish connection to the RF627 device by Service Protocol.
      bool is_connected = scanner->connect();
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    if (!is_connected){
        std::cout << "Failed to connect to scanner!" << std::endl;
        continue;
    }

    // read params from RF627 device by Service Protocol.
    bool is_read = scanner->read_params();
    if (!is_read){
        std::cout << "Failed to read scanner parameters!" << std::endl;
        continue;
    }

    //
    // Example of working with the parameter type:
    // std::string
    //
    // Get parameter of Device Name
    auto name = scanner->get_param("user_general_deviceName");
    if (name != nullptr)
    {
        std::string str_name = name->getValue<std::string>();
        std::cout << "Current Device Name \t: " << str_name << std::endl;

        // Add "_TEST" to the ending of the current name
        str_name += "_TEST";
        scanner->set_param("user_general_deviceName", str_name);
        std::cout << "New Device Name \t: " << str_name     << std::endl;
        std::cout << "-------------------------------------"<< std::endl;
    }

    //
    // Example of working with the parameter type:
    // uint32_t
    //
    // Get parameter of Sensor Framerate
    auto fps = scanner->get_param("user_sensor_framerate");
    if (fps != nullptr)
    {
        uint32_t value = fps->getValue<uint32_t>();
        std::cout<<"Current FPS\t\t: "<< value << std::endl;

        // Change the framerate to 100
        scanner->set_param("user_sensor_framerate", 100);
        std::cout<<"New FPS  \t\t: " << 100 << std::endl;
        std::cout << "-------------------------------------"<< std::endl;
    }

    //
    // Example of working with the parameter type:
    // std::vector<uint32_t>
    //
    // Get parameter of Device IP Addr
    auto ip_addr = scanner->get_param("user_network_ip");
    if (ip_addr != nullptr)
    {
        std::vector<uint32_t> ip =ip_addr->getValue<std::vector<uint32_t>>();
        std::cout << "Current Device IP\t: ";
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        for(auto i: ip) std::cout<<std::to_string(i)<<".";std::cout<< "\n";

        // Change last digit of IP address (e.g. 192.168.1.30->192.168.1.31)
        //ip[3]++;
        scanner->set_param("user_network_ip", ip);
        std::cout << "New Device IP    \t: ";
        for(auto i: ip) std::cout<<std::to_string(i)<<".";std::cout<< "\n";
        std::cout << "-------------------------------------"       << "\n";
    }

    //
    // Example of working with using an Enum:
    // uint32_t
    //
    // Get parameter of Sync Source
    auto syncSource = scanner->get_param("user_sensor_syncSource");
    if (syncSource != nullptr)
    {
        uint32_t value = syncSource->getValue<uint32_t>();
        auto syncEnum = syncSource->getEnum<uint32_t>();
        std::cout << "Current Sync Source\t: "
                  << syncEnum.findLabel(value)
                  << std::endl;

        // Change the current state to SYNC_EXTERNAL (or SYNC_INTERNAL)
        if(value == syncEnum.getValue("SYNC_INTERNAL"))
        {
            scanner->set_param_by_key(
                        "user_sensor_syncSource", "SYNC_EXTERNAL");
            std::cout << "New Sync Source  \t: "
                      << syncEnum.getLabel("SYNC_EXTERNAL")
                      << std::endl;
        }else
        {
            scanner->set_param_by_key(
                        "user_sensor_syncSource", "SYNC_INTERNAL");
            std::cout << "New Sync Source  \t: "
                      << syncEnum.getLabel("SYNC_INTERNAL")
                      << std::endl;
        }
        std::cout << "-------------------------------------"<< std::endl;
    }

    //
    // Example of working with using an Enum:
    // uint32_t
    //
    // Get parameter of Laser Enabled
    std::shared_ptr<param> laser = scanner->get_param("user_laser_enabled");
    if (laser != nullptr)
    {
        uint32_t isEnabled = laser->getValue<uint32_t>();
        auto laserEnum = laser->getEnum<uint32_t>();
        std::cout << "Current Laser State\t: "
                  << laserEnum.findLabel(isEnabled)
                  << std::endl;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        // Change the current state to the opposite
        if(isEnabled == laser->getEnum<uint32_t>().getValue("FALSE"))
        {
           scanner->set_param_by_key("user_laser_enabled", "TRUE");
           std::cout << "New Laser State  \t: "
                     << laserEnum.getLabel("TRUE")
                     << std::endl;
        }else
        {
           scanner->set_param_by_key("user_laser_enabled", "FALSE");
           std::cout << "New Laser State  \t: "
                     << laserEnum.getLabel("FALSE")
                     << std::endl;
        }
        std::cout << "-------------------------------------"<< std::endl;
     }


     // Apply changed parameters to the device
     std::string answer = "n";
     std::cout << "Apply changed params to the device? (y/n): ";
     std::cin >> answer;
     if (answer == "y" || answer == "Y")
     {
        scanner->write_params();
        // Save changes to the device's memory
        std::cout<<std::endl<<"Save changes to device's memory? (y/n): ";
        std::cin >> answer;
        if (answer == "y" || answer == "Y")
           scanner->save_params();
     }

     // Disconnect from scanner.
     scanner->disconnect();
  }

  // Cleanup resources allocated with sdk_init()
  sdk_cleanup();
}
```

Ниже приведён результат вывода приложения при успешной установке новых параметров:

```
########################################
#                                      #
#       Parameter Example v2.x.x       #
#                                      #
########################################

SDK version: 2.19.0
=======================================
Search scanners from:
* IP Address    : 192.168.2.100
* Netmask       : 255.255.255.0
Discovered      : 0 RF627-Smart
---------------------------------------
Search scanners from:
```

(continues on next page)

(продолжение с предыдущей страницы)

```
* IP Address    : 192.168.1.2
* Netmask       : 255.255.255.0
Discovered      : 1 RF627-Smart
----------------------------------------
Was found       : 1 RF627 v2.x.x
========================================


ID scanner's list: 0
----------------------------------------
Device information:
* Name           : RF627 scanner
* Serial         : 190068
* IP Addr        : 192.168.1.30
* MAC Addr       : 00:0A:35:6E:07:F5
Current Device Name      : RF627 scanner
New Device Name          : RF627 scanner_TEST
-----------------------------------
Current FPS              : 490
New FPS                  : 100
-----------------------------------
Current Device IP        : 192.168.1.30.
New Device IP            : 192.168.1.30.
-----------------------------------
Current Sync Source      : Internal
New Sync Source          : External
-----------------------------------
Current Laser State      : true
New Laser State          : false
-----------------------------------
Apply changed params to the device? (y/n): y
Save changes to device's memory? (y/n): n
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/Cpp/RF627_SMART/PARAMETER_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.2.5 Запись и скачивание дампа

Ниже приведен пример записи дампа профилей и его скачивание:

```
#include <string>
#include <iostream>

#include "rf62Xsdk.h"
#include "rf62Xtypes.h"

using namespace SDK::SCANNERS::RF62X;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
int main()
{
   std::cout << "########################################"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "#          Frame Example v2.x.x        #"  << std::endl;
   std::cout << "#                                      #"  << std::endl;
   std::cout << "########################################\n"<< std::endl;

   // Initialize sdk library
   sdk_init();

   // Print return rf62X sdk version
   std::cout << "SDK version: " << sdk_version()                << std::endl;
   std::cout << "======================================="     << std::endl;


   // Create value for scanners vector's type
   std::vector<std::shared_ptr<rf627smart>> list;
   // Search for rf627smart devices over network
   list = rf627smart::search();

   // Print count of discovered rf627smart in network by Service Protocol
   std::cout << "Was found\t: " << list.size()<< " RF627 v2.x.x"<< std::endl;
   std::cout << "======================================="     << std::endl;


   // Iterate over all discovered scanners in network, connect to each of␣
↪them,
   // get a profile and print the main profile-info.
   for (size_t i = 0; i < list.size(); i++)
   {
      std::shared_ptr<rf627smart> scanner = list[i];
      std::shared_ptr<hello_info> info = scanner->get_info();

      // Print information about the scanner to which the profile belongs.
      std::cout << "\n\nID scanner's list: " << i              << std::endl;
      std::cout << "---------------------------------------" << std::endl;
      std::cout << "Device information: "                     << std::endl;
      std::cout << "* Name  \t: "   << info->device_name()     << std::endl;
      std::cout << "* Serial\t: "   << info->serial_number()   << std::endl;
      std::cout << "* IP Addr\t: "  << info->ip_address()      << std::endl;
      std::cout << "---------------------------------------" << std::endl;

      // Establish connection to the RF627 device by Service Protocol.
      bool is_connected = scanner->connect();
      if (!is_connected){
         std::cout << "Failed to connect to scanner!" << std::endl;
         continue;
      }

      uint32_t count_of_profiles = 1000;
      scanner->start_dump_recording(count_of_profiles);

      std::cout << "Start dump recording..."                  << std::endl;
      std::cout << "---------------------------------------" << std::endl;
      uint32_t size = 0;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      do {
         scanner->read_params();
         size = scanner->get_param("user_dump_size")->getValue<uint32_t>();
         std::cout << "Current profiles in the dump: "<< size << std::endl;
      }while(size < count_of_profiles);
      std::cout << "----------------------------------------" << std::endl;

      std::cout << "Start dump downloading..."                 << std::endl;
      std::vector<std::shared_ptr<profile2D>> dump =
               scanner->get_dumps_profiles(0, count_of_profiles);

      std::cout << dump.size() << " Profiles in dump were downloaded!\n";
      std::cout << "----------------------------------------" << std::endl;

      // Disconnect from scanner.
      scanner->disconnect();
   }

   // Cleanup resources allocated with sdk_init()
   sdk_cleanup();
}
```

Ниже приведён результат вывода приложения при успешной записи и скачивании дампа профилей:

```
########################################
#                                      #
#       Parameter Example v2.x.x       #
#                                      #
########################################

SDK version: 2.19.0
========================================
Search scanners from:
* IP Address   : 192.168.2.100
* Netmask      : 255.255.255.0
Discovered     : 0 RF627-Smart
----------------------------------------
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627-Smart
----------------------------------------
Was found      : 1 RF627 v2.x.x
========================================


ID scanner's list: 0
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
----------------------------------------
Start dump recording...
----------------------------------------
Current profiles in the dump: 0
```

(continues on next page)

(продолжение с предыдущей страницы)

```
Current profiles in the dump: 67
Current profiles in the dump: 205
Current profiles in the dump: 415
Current profiles in the dump: 702
Current profiles in the dump: 921
Current profiles in the dump: 1000
----------------------------------------
Start dump downloading...
1000 Profiles were received!
----------------------------------------
Press <RETURN> to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Qt Creator**:

- Загрузите файл *CMakeLists.txt* из папки **Examples/Cpp/RF627_SMART/DUMP_EXAMPLE** через **File > Open File or Project** (укажите файл *CMakeLists.txt*)
- Выберите компилятор (*MinGW, MSVC, Clang*) и нажмите **Configure Project**
- Запустите проект

5.3 Примеры для C#

5.3.1 Поиск устройств RF62X

Ниже приведен пример поиска сканеров серии RF627 v2.x.x:

```
using System;
using System.Collections.Generic;
using SDK.SCANNERS;

namespace SEARCH_EXAMPLE
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("########################################");
         Console.WriteLine("#                                      #");
         Console.WriteLine("#         Search Example v2.x.x         #");
         Console.WriteLine("#                                      #");
         Console.WriteLine("########################################");

         // Initialize sdk library
         RF62X.SdkInit();

         // Print return rf62X sdk version
         Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
         Console.WriteLine("========================================");

         // Search for RF627smart devices over network
         List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

         // Print count of discovered rf627smart in network
         Console.WriteLine("Was found\t: {0} RF627 v2.x.x", list.Count);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            Console.WriteLine("========================================");


            for (int i = 0; i < list.Count; i++)
            {
               RF62X.HelloInfo info = list[i].GetInfo();

               Console.WriteLine("\n\nID scanner's list: {0}", i);
               Console.WriteLine("----------------------------------------");
               Console.WriteLine("Device information: ");
               Console.WriteLine("* Name\t: {0}", info.device_name);
               Console.WriteLine("* Serial\t: {0}", info.serial_number);
               Console.WriteLine("* IP Addr\t: {0}", info.ip_address);
               Console.WriteLine("* MAC Addr\t: {0}", info.mac_address);

               Console.WriteLine("\nWorking ranges: ");
               Console.WriteLine("* Zsmr, mm\t: {0}", info.z_smr);
               Console.WriteLine("* Zmr , mm\t: {0}", info.z_mr);
               Console.WriteLine("* Xsmr, mm\t: {0}", info.x_smr);
               Console.WriteLine("* Xemr, mm\t: {0}", info.x_emr);

               Console.WriteLine("\nVersions: ");
               Console.WriteLine("* Firmware\t: {0}", info.firmware_version);
               Console.WriteLine("* Hardware\t: {0}", info.hardware_version);
               Console.WriteLine("----------------------------------------");
            }

            // Cleanup resources allocated with SdkInit()
            RF62X.SdkCleanup();

            Console.WriteLine("Press any key to close this window...");
            Console.ReadKey();
        }
    }
}
```

Ниже приведён результат вывода приложения при успешном обнаружении сканера в сети:

```
########################################
#                                      #
#          Search Example v2.x.x       #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address    : 192.168.1.2
* Netmask       : 255.255.255.0
Discovered      : 1 RF627
----------------------------------------
Search scanners from:
* IP Address    : 192.168.2.104
* Netmask       : 255.255.255.0
Discovered      : 0 RF627
----------------------------------------
Was found       : 1 RF627 v2.x.x
```

(continues on next page)

(продолжение с предыдущей страницы)

```
ID scanner in list: 0
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
* MAC Addr      : 00:0A:35:6E:07:F5

Working ranges:
* Zsmr, mm      : 70
* Zmr , mm      : 50
* Xsmr, mm      : 30
* Xemr, mm      : 42

Versions:
* Firmware      : 2.7.1
* Hardware      : 18.6.20
----------------------------------------
Press any key to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Visual Studio**:

- Используя Visual Studio откройте из папки **RF62X-SDK/Examples/CSharp/RF627_smart** проект *RF627_TESTS*.
- Укажите целевую платформу **x64 Debug** или **x64 Release**
- Скомпилируйте **SEARCH_EXAMPLE**
- Перед запуском скачайте архив библиотек для C# (смотреть последние выпуски RF62X-SDK библиотек) и скопируйте из архива в папке `Dependencies` все файлы с именем **libRF62X-SDK** в папку к исполняемому файлу проекта (`../bin/x64/Debug/` или `../bin/x64/Release/`)
- Запустите пример

5.3.2 Получение профиля сканера

Ниже приведен пример получение профиля со сканера серии RF627 v2.x.x:

```
using System;
using System.Collections.Generic;
using SDK.SCANNERS;

namespace SEARCH_EXAMPLE
{
   class Program
   {

      public static uint profileCount = 0;
      public static uint profileLost = 0;
      public static bool isReceiveRun = true;
      public static void receive_profiles(RF62X.RF627smart scanner)
      {
         // Get profile from scanner's data stream by Service Protocol.
         RF62X.Profile2D profile = null;
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        bool zero_points = true;
        bool realtime = false;

        uint last_index = 0;
        bool first_profile = true;
        while (true)
            if ((profile = scanner.GetProfile(zero_points, realtime)) != null)
            {
                if (first_profile)
                {
                    last_index = profile.header.measure_count;
                    first_profile = false;
                }
                else
                {
                    profileCount++;
                    if (profile.header.measure_count - last_index > 1)
                        profileLost+=(profile.header.measure_count - last_index);
                    last_index = profile.header.measure_count;
                }
            }
            else
            {
                Console.WriteLine("Profile was not received!");
                Console.WriteLine("----------------------------------------");
            }
    }

    static void Main(string[] args)
    {
        Console.WriteLine("##########################################");
        Console.WriteLine("#                                        #");
        Console.WriteLine("#         Profile Example v2.x.x         #");
        Console.WriteLine("#                                        #");
        Console.WriteLine("##########################################");

        // Initialize sdk library
        RF62X.SdkInit();

        // Print return rf62X sdk version
        Console.WriteLine("SDK version: {0}", RF62X.SdkVersion());
        Console.WriteLine("========================================");

        // Search for RF627smart devices over network
        List<RF62X.RF627smart> list = RF62X.RF627smart.Search(500);

        // Print count of discovered rf627smart in network
        Console.WriteLine("Was found\t: {0} RF627-Smart", list.Count);
        Console.WriteLine("========================================");

        int index = -1;
        if (list.Count > 1)
        {
            Console.WriteLine("Select scanner for test: ");
            for (int i = 0; i < list.Count; i++)
                Console.WriteLine("{0}. Serial: {1}", i,
                                  list[i].GetInfo().serial_number);
```

(continues on next page)

(продолжение с предыдущей страницы)

```
            index = Convert.ToInt32(Console.ReadLine());
        }
        else if (list.Count == 1)
            index = 0;

                    if (index == -1)
             return;

        RF62X.HelloInfo info = list[index].GetInfo();

        Console.WriteLine("----------------------------------------");
        Console.WriteLine("Device information: ");
        Console.WriteLine("* Name  \t: {0}", info.device_name);
        Console.WriteLine("* Serial\t: {0}", info.serial_number);
        Console.WriteLine("* IP Addr\t: {0}", info.ip_address);
        Console.WriteLine("----------------------------------------");

        // Establish connection to the RF627 device by Service Protocol.
        bool is_connected = list[index].Connect();

        if (is_connected)
        {
            Thread receiver = new Thread(() => receive_profiles(list[index]));
            isReceiveRun = true;
            receiver.Start();

            Console.WriteLine("Thread of receiving profiles started");
            Console.WriteLine("For interrupt receiving press \"Ctrl+C\"");

            bool isRun = true;
            Console.CancelKeyPress += delegate
                (object sender, ConsoleCancelEventArgs consoleArgs) {
                consoleArgs.Cancel = true;
                isRun = false;
                isReceiveRun = false;
            };

            while (isRun)
            {
                Thread.Sleep(1000);
                Console.WriteLine("FPS: {0}, Lost: {1}",
                                  Program.profileCount, profileLost);
                profileLost = 0;
                profileCount = 0;
            }

            receiver.Join();
            Console.WriteLine("Thread of receiving profiles interrupted");
            Console.WriteLine("----------------------------------------");
        }

        // Cleanup resources allocated with sdk_init()
        RF62X.SdkCleanup();

        Console.WriteLine("Press any key to close this window...");
        Console.ReadKey();
    }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    }
}
```

Ниже приведён результат вывода приложения при успешном получении профиля:

```
########################################
#                                      #
#         Profile Example v2.x.x       #
#                                      #
########################################
SDK version: 2.17.2
========================================
Search scanners from:
* IP Address   : 192.168.1.2
* Netmask      : 255.255.255.0
Discovered     : 1 RF627
----------------------------------------
Search scanners from:
* IP Address   : 192.168.2.104
* Netmask      : 255.255.255.0
Discovered     : 0 RF627
----------------------------------------
Was found      : 1 RF627 v2.x.x
========================================
----------------------------------------
Device information:
* Name          : RF627 scanner
* Serial        : 190068
* IP Addr       : 192.168.1.30
----------------------------------------
Thread of receiving profiles started
For interrupt receiving press "Ctrl+C"
FPS: 494, Lost: 0
FPS: 490, Lost: 0
FPS: 491, Lost: 0
Thread of receiving profiles interrupted
----------------------------------------
Press any key to close this window...
```

Вы можете открыть и скомпилировать этот пример с помощью **Visual Studio**:

- Используя Visual Studio откройте из папки **RF62X-SDK/Examples/CSharp/RF627_smart** проект *RF627_TESTS*.
- Укажите целевую платформу **x64 Debug** или **x64 Release**
- Скомпилируйте **PROFILE_EXAMPLE**
- Перед запуском скачайте архив библиотек для C# (смотреть последние выпуски RF62X-SDK библиотек) и скопируйте из архива в папке `Dependencies` все файлы с именем **libRF62X-SDK** в папку к исполняемому файлу проекта (`../bin/x64/Debug/` или `../bin/x64/Release/`)
- Запустите пример