UHF SDK Specification

2nd Revision

Version :

15.Mar.2012 16:15

Confidentiality Note

This document may only be circulated to those people involved in the project.
The document may not be passed on to third parties without permission of iDTRONIC.

Table of contents

1 Abstract

This document is intended to keep all participating parties on the project at the same level of information and to summarize all kind of ideas, wishes, recommendations and must have features from customers, employees, support team and other kind of sources.

This preliminary specification describes the product's requirements for UHF EVO SDK development.

When all preliminary specifications are clear and accepted by all participating parties, the next step should be to push this document into final specification documentation where all issues are described.

2 General Product Philosophy

RFID readers/tags represent a next step in technology today. Combination between reader and tag offers a very stable solution and high distance reading possibility make a much requested solution for industry areas.

This reader is very new for our users and integration should be done through SDK. During time several feature was added to reader and no other SDK was developed. Our users request us a new version from SDK which can provide also new feature and future development. This new SDK provides the following features:

- No need of knowledge about low-level protocol of RFID device:
  SDK library represents a transparent solution for communication with the RFID device. Because the native RFID device communication protocol is not so easy to be implemented, this SDK library assists integration of the RFID reader in an end-customer solution.
- Fast integration in end user application:
  Communication library offers also high scalability for integration in end user product. Because this library contains only a few functions, this will offer the possibility of fast integration in end user software and avoids problems created by interchanging data using low-level protocols.
- Compatibility with early version between SDK versions:
  This SDK offers high level of compatibility with early versions of the library. So updating to a newer version should be complete transparent and no modifications are needed in developed software.

3 Interoperability

UHF-EVO reader will be integrated in our SDK library to be able to offer high scalability and represents fast and common way to be accessed from end user developers from other programming languages. To demonstrate interoperability options we will deliver examples for C++ and C# programming languages.

4 Limitations

UHF-EVO reader is able to detect and to exchange data only with specific UHF cards. This protocol is 18000-C.

5 SDK Library

To help end user developers we decided to provide an SDK which emulates and optimizes reader functionality for an easy integration. The library exports these functions:

1. OpenReader
2. GetReaderCaps
3. SetReaderCaps
4. ReadData
5. WriteData
6. CloseReader

These functions will be described in detail in the following chapters.

5.1 OpenReader

Main proposal of this function is to initialize and check if UHF-EVO reader is already connected to serial host. Input parameters are next:

- nPort – port where the reader is connected

After calling this function the following information is returned:

- SDK handle to reader if successfully
- Null in case of error

5.2 GetReaderCaps

This function gets information about reader:
Input parameters are next:

- SDKHANDLE – Handle to reader from previous “OpenReader”
- EREADER_FEARURE – One of following reader features:
  - ERD_SERIAL_NUMBER – return reader serial number
  - ERD_READER_TYPE – return current reader type
  - ERD_HARDWARE_REVISION – return hardware revision
  - ERD_SOFTWARE_REVISION – return current software revision
  - ERD_BOOTLOADER_REVISION – return bootloader revision
  - ERD_CURRENT_STATE – return reader status
  - ERD_ATTENUATION –get reader attenuation
  - ERD_SENSIVITY –get reader sensitivity
  - ERD_FREQUENCY –get reader frequency
  - ERD_CARD – scan for present cards
- Pointer to byte array for returning data
- Reference to length of data

Data format request by each feature will be described in next session:

- ERD_SERIAL_NUMBER
  - Buffer to byte array where reader serial number will be returned
  - Data length – size of data buffer

- ERD_READER_TYPE
  - Buffer to byte array with reader type will be returned
  - Data length – size of data buffer

- ERD_HARDWARE_REVISION
  - Buffer to byte array with reader hardware revision will be returned
  - Data length – size of data buffer

- ERD_SOFTWARE_REVISION
  - Buffer to byte array with reader software revision will be returned
  - Data length – size of data buffer

- ERD_BOOTLOADER_REVISION
  - Buffer to byte array with reader bootloader revision will be returned
  - Data length – size of data buffer

- ERD_CURRENT_STATE
  - Return 1 byte with current status (0 – idle, > 1 busy)
  - Data length – size of byte

- ERD_ATTENUATION
  - Return data structure with next format
    - Short max attenuation supported
    - Short current attenuation
  - Data length – 2 × size of short

- ERD_SENSIVITY
  - Return data structure with next format
    - Short max sensivity supported
    - Short min sensivity
    - Short current sensivity
  - Data length – 3 × size of short

- ERD_FREQUENCY
  - Return data structure with next format
    - Byte number of frequency
    - Array with unsigned integer frequency – max 16
  - Data length – size of structure

- ERD_CARD
  - Buffer with next format where card id will be returned
    - Number of cards detected
    - Byte card length – size of long
    - Byte array with card id
  - Data length – size of data buffer

In return we get the following status codes:

- ER_OK – no error detected
- ER_FEATURE – invalid feature
- ER_MORE_DATA – more data available
- ER_INVALID_POINTER – data buffer is null

5.3 SetReaderCaps

This function sets some reader features and should be treated very carefully:
Input parameters are as follows:

- SDKHANDLE – Handle to reader from previous “OpenReader”
- EREADER_FEARURE – One of following reader features:
  - ERD_ATTENUATION – set reader attenuation
  - ERD_SENSIVITY – set reader sensitivity
  - ERD_FREQUENCY – set reader frequency
  - ERD_REBOOT – perform reader reboot
  - ERD_RESTORE_SETTINGS – restore factory settings
  - ERD_SAVE_SETTINGS – save current settings applied
  - ERD_READ_WRITE_PARAM – set read/write parameters
  - ERD_KILL_TAG – destroy specific tag
  - ERD_LOCK – lock specific memory bank from a tag

- Pointer to byte array with feature information
- Length of data byte array

Data format requested by each feature will be described in the following section:

- ERD_ ATTENUATION
  - Buffer with next data structure
    - Short current attenuation
  - Data length – size of short

- ERD_ SENSIVITY
  - Buffer with next data structure
    - Short current sensitivity
  - Data length – size of short

- ERD_ FREQUENCY
  - Buffer with next data structure
    - Byte number of frequency
    - Array with unsigned integer frequency – max 16
  - Data length – size of structure

- ERD_REBOOT
  - Buffer contain 1 byte with reboot status (1 – true/ 0 - false)
  - Data length – size of data buffer – 1 byte

- ERD_RESTORE_SETTINGS
  - Buffer contain 1 byte with restore status (1 – true/ 0 - false)
  - Data length – size of data buffer – 1 byte

- ERD_SAVE_SETTINGS
  - Buffer contain 1 byte with save status (1 – true/ 0 - false)
  - Data length – size of data buffer – 1 byte

- ERD_READ_WRITE_PARAM
  - Buffer contains 44 bytes with info about read/write parameters
    - 1 byte – tag id length (max 32)
    - n bytes – tag id
    - 4 bytes – tag pass
    - 1 byte – memory bank
    - 4 bytes – number blocks to read/write (as unsigned long)
  - Data length – size of data buffer – 44 bytes

- ERD_KILL
  - Buffer contain 38 bytes with info about read/write parameters
    - 1 byte – tag id length (max 32)
    - n bytes – tag id
    - 4 bytes – tag pass
  - Data length – size of data buffer – 38 byte

- ERD_LOCK
  - Buffer contain 41 bytes with info about read/write parameters
    - 1 byte – tag id length (max 32)
    - n bytes – tag id
    - 4 bytes – tag pass
    - 1 byte – memory bank
    - 1 byte – lock type (0 – unlock/1 – lock/2 – permanent lock)
  - Data length – size of data buffer – 41 byte

In return we get the following status codes:

- ER_OK – no error detected
- ER_FEATURE – invalid feature
- ER_MORE_DATA – more data available
- ER_INVALID_POINTER – data buffer is null

5.4 ReadData

This function read data from previous file selected using SetDevCaps feature ERD_READ_WRITE_PARAM . Like input parameter we have next:

- SDKHANDLE - Handle to reader from previous “OpenReader”
- Pointer to byte array for return data
- UInt32 - Length of data to be read

In return we get the following status codes:

- ER_OK – no error detected
- ER_READ – invalid feature
- ER_INVALID_POINTER – data buffer is null

5.5 WriteData

This function writes data from previous file selected using SetDevCaps feature ERD_READ_WRITE_PARAM. The input parameters are as follows:

- SDKHANDLE – Handle to reader from previous “OpenReader”
- Pointer to byte array with data
- UInt32 - Length of data to be written

In return we get the following status codes:

- ER_OK – no error detected
- ER_WRITE – invalid feature
- ER_INVALID_POINTER – data buffer is null

5.6 CloseReader

This function ends the communication with the RFID device and turns the power off. The input parameters are as follows:

- SDKHANDLE - Handle to reader from previous “OpenReader”

In return we get the following status codes:

- ER_OK – no error detected
- ER_INVALID_POINTER – handle is null

6 Operation Schedule

	Duration	**Dead-line**
1st Implementing new SDK for UHF-EVO reader		09.03.2012

7 Document History

Version / Date	**Changes**
1st / March, 2012	Initial Version
2nd / March, 2012	Revised

8 Notes