Legic SDK Specification

2nd Revision

Version :

15.Mar.2012 10:15

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 Legic 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

Because world is changing day by day to secure communication, our company try to keep up with new wave bringing on market a new reader from Legic. Legic readers will represent a fully trustable and safe solution for end-user. Because the Legic cards offers high level of security we should use Legic reader solutions to communicate with these cards. One of possibilities for fast integration represents the Legic SM-4200 card reader processor. This processor offer high scalability and fast development for end user solution.

Because our EVO product line was a great success we decided that one of first implementations of Legic 4200 card reader processor to be made on EVO line (R-DT-EVO-LEG42-USB). To be able to reach this target we decided to provide a SDK solution which can communicate with Legic-4200 module. SDK solution for EVO line (R-DT-EVO-LEG42-USB) offers us several advantages:

- No need of knowledge about low level protocol from Legic reader
  SDK library represent a transparent solution for communication with Legic readers. Because Legic communication protocol is secret the SDK library assists us to integrate Legic reader in our end solution.
- Fast integration in end user solution
  Communication library offers also high scalability for integration in end user product. Because this library contains only a few functions, this will offer us possibility of fast integration in end user software avoid big problems created by low level protocols interchanging data.
- Compatibility with early version between SDK versions.
  SDK offers high level of compatibility with early versions of library. So updating to a newer version should be complete transparent and should not generate any modification in developed software.

3 Interoperability

Legic readers will be integrated in our SDK library to be able to offer high scalability and represent a 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

Legic reader is able to detect and to exchange data only with specific Legic cards. For rest of cards Legic reader can detect these cards (except for Mifare cards, where only the family can be detected, but not the specific chip type) but can’t read and write. Transparent mode can’t be used. As a result our SDK will handle only Legic cards.

5 SDK Library

Legic protocol is secret so practically we can’t communicate with cards directly but we should use functionality provided by Legic SM-4200. To help end user developers we decided to provide an SDK which emulate and optimize reader functionality for an easy integration. The library exports these functions:

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

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

5.1 GetAvailableCom

This function scans after available serial Com-Ports in the system. We implement this function because in several programming language accessing is a little difficult to access Windows SDK functions. This function checks for available serial Com-Port but does not detect any Legic device connected to these ports. GetAvailableCom function request the following parameters:

- Port position – which port is detected at position 0..n

After calling this function we receive a serial Com-Port ID or -1 if no serial Com-Port available.

5.2 OpenReader

Main proposal of this function is to initialize and check if the Legic reader is already connected to the serial host. Input parameters are as follows:

- 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

5.3 GetReaderCaps

This function gets information about reader:
Input parameters are as follows:

- SDKHANDLE – Handle to reader from previous “OpenReader”
- EREADER_FEARURE - One of next reader feature:
  - ERD_READER_TYPE – return reader type
  - ERD_HARDWARE_REV – return hardware revision
  - ERD_POWER – return power type for reader
  - ERD_BOOT_LOADER – return boot loader version
  - ERD_FIRMWARE – return firmware of reader
  - ERD_CRYPT_MODE – return status of communication – crypt or not
  - ERD_CARDS – return card detected by reader
- Pointer to byte array for returning data
- Reference to length of data

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

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

- ERD_HARDWARE_REV
  - Buffer to char array where hardware rev. will be returned
  - Data length – size of data buffer

- ERD_POWER
  - Buffer to char array where power type will be returned
  - Data length – size of data buffer

- ERD_BOOT_LOADER
  - Buffer to char array where boot loader will be returned
  - Data length – size of data buffer

- ERD_FIRMWARE
  - Buffer to char array where firmware rev will be returned
  - Data length – size of data buffer

- ERD_CRYPT_MODE
  - Buffer to char array where crypt mode status will be returned
  - Data length – size of data buffer

- ERD_CARDS
  - Input Buffer
    - Byte – card format
  - Output buffer with next format:
    - UInt32 – nr cards
    - Byte – card data length
    - Buffer 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

5.4 SetReaderCaps

This function set 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 next reader feature:
  - ERD_LEDS – turn on/off module led
  - ERD_BUZZER – turn on/off module buzzer
  - ERD_CARD – select specific card
  - ERD_SELECT_FILE – select specific file from a card
- Pointer to byte array with feature information
- Length of data byte array

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

- ERD_LEDS
  - Buffer contain byte with led status (0 = off, 1=on)
  - Data length – size of data buffer – 1 byte

- ERD_BUZZER
  - Buffer contain 1 byte with buzzer status (0 = off, 1=on)
  - Data length – size of data buffer – 1 byte

- ERD_CARD
  - Buffer contain 3 byte about card
    - Byte 0 – card type
    - Byte 1 – card position in list from GetDevCaps (ERD_CARD - feature)
    - Byte 2 – communication speed (0xFF)
  - Data length – size of data buffer – 3 byte

- ERD_SELECT_FILE
  - Buffer contain info about file
    - Byte 0 – segment position
    - Byte 1 – file type
    - Byte 2 – file info length
    - Byte 3..n – file info data
  - 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

5.5 ReadData

This function read data from previous file selected using SetDevCaps feature ERD_SELECT_FILE.
Input parameters are as follows:

- SDKHANDLE – Handle to reader from previous “OpenReader”
- UInt32 – Start address
- 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

5.6 WriteData

This function writes data to a previous file selected using SetDevCaps feature ERD_SELECT_FILE.
Input parameters are as follows:

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

In return we get the following status codes:

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

5.7 CloseReader

This function ends communication with Legic reader and turn off power.
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 it’s null

6 Operation Schedule

	Duration	**Dead-line**
1st Implementing new SDK for Legic reader		16.01.2012

7 Document History

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

8 Notes