Использование EMV библиотеки (emv_ker.dll):

Использование должно начинаться с вызова функции emv_kernelInit().

В случае успеха, функция emv_kernelInit() возвращает указатель, который в дальнейшем используется как параметр при вызове функций EMV библиотеки.

В случае ошибки, emv_kernelInit() возвращает указатель на FALSE и работа EMV библиотеки не может быть продолжена.

Управление считывателем smart-card.

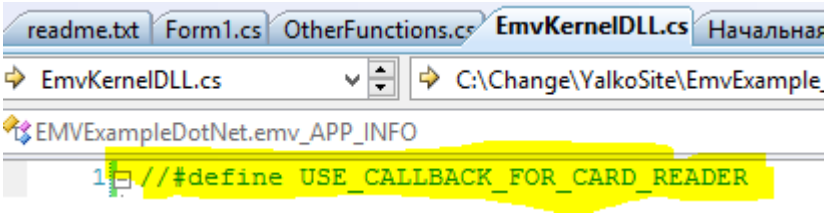
В EMV библиотеке предусмотренно, два варианта управления считывателем:

1) Посредством динамичеси присоединяемой библиотеки. (SmartCard.dll)
2) Посредством функций обратного вызова.

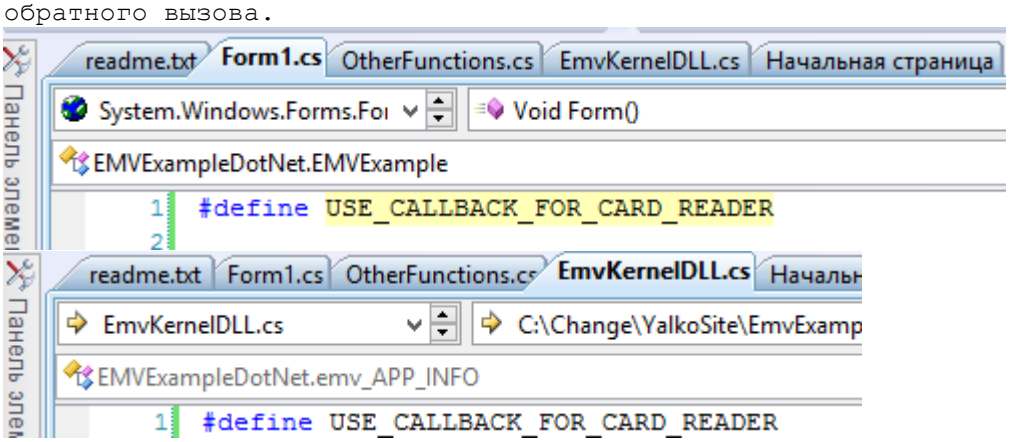
Для выбора одного из вариантов управления считывателем, предусмотренно макроопределение:

#define **USE_CALLBACK_FOR_CARD_READER**

Если макроопределение **USE_CALLBACK_FOR_CARD_READER** не установлено, то компиляция выолняется для динамичеси присоединяемой библиотеки. (SmartCard.dll)



Если макроопределение **USE_CALLBACK_FOR_CARD_READER** установлено, то компиляция выполняется для управления считывателем посредством функции обратного вызова.



1) Управление считывателем посредством динамически присоединяемой библиотеки:

В теле функции emv_kernelInit(), библиотека EMV пытается загрузить библиотеку SmartCard.dll.
Если библиотека отсутствует, или отсутствует минимальный набор функций, возвращается указатель на FALSE и работа EMV библиотеки не может быть продолжена.
Библиотека SmartCard.dll должна содержать следующий минимальный набор функций:

```
BYTE WINAPI SelectReader(LPSTR rname,DWORD rname_size);
```

```
BYTE WINAPI OpenReader(LPSTR rname);
BYTE WINAPI CloseReader(void);

/**
*****************************************************************************************
*  @fn BYTE WINAPI Atr(BYTE *atr, DWORD * max_len, BYTE * protocol)
*  @b Назначение:                                                          \n
*  @brief Получает ATR, используется EMV ядром.     \n
*  @author  Ilya Yalin
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Посылает ридеру команду подать питание на карточку.
*  2.) Получает от карточки ATR.
*  @endcode
*****************************************************************************************
*  @param Вход: \n
*  @b DWORD * max_len;- Максимальный размер буфера для приёма данных ATR.
*
*  @param Выход: \n
*  @b BYTE *atr       - Указатель на буфер для данных ATR.
*  @b DWORD * max_len;- Размер полученных данных.
*  @b BYTE * protocol;- Адрес для получения протокола. Для моторизованных ридеров всегда 0.
*****************************************************************************************
*  @return
*  @b BYTE;    - Статус выполнения операций.
*  @b 1 - операция завершилась успешно.
*  @b 0 - операция завершилась не удачей.
*****************************************************************************************
*/
BYTE WINAPI Atr (BYTE *atr, DWORD * max_len,BYTE * protocol);

/**
*****************************************************************************************
*  @fn WORD WINAPI SendApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,BYTE *answer,WORD answer_max_len)
*  @b Назначение:                                                          \n
*  @brief Посылает APDU команду с данными и получает ответ. Используется EMV ядром.      \n
*  @author
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Посылает APDU команду и получает ответ.
*  2.) Возвращает данные в буфер.
*  @endcode
*****************************************************************************************
*  @param Вход: \n
*  @b BYTE * Cmd;     - Буфер, содержащий APDU команду.
*  @b BYTE * Data;    - Буфер, содержащий данные.
*  @b WORD *Data_len; - Длина даннных.
*  @b WORD answer_max_len;    - Размер буфера для ответа.
*
*  @param Выход: \n
*  @b BYTE * answer;  - Буфер для ответа от карты на APDU команду.
*  @b WORD *Data_len; - Длина полученых от карты даннных.
*****************************************************************************************
*  @return
*  @b WORD;    - Статус, полученный от карты на APDU команду.
*****************************************************************************************
*/
WORD WINAPI SendApduIn(BYTE *cmd,BYTE * data,WORD *data_len,BYTE *answer,WORD answer_max_len);

/**
*****************************************************************************************
*  @fn WORD WINAPI SendApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,WORD answer_max_len)
*  @b Назначение:                                                          \n
*  @brief Посылает APDU команду и получает ответ. Используется EMV ядром. \n
*  @author
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Посылает APDU команду и получает ответ.
*  2.) Возвращает данные в буфер.
*  @endcode
```

```
*****************************************************************************************
*  @param Вход: \n
*  @b BYTE * Cmd;     - Буфер, содержащий APDU команду.
*  @b WORD answer_max_len;    - Размер буфера для ответа.
*
*  @param Выход: \n
*  @b WORD * Stat;    - Возвращается статус, полученный от карты на APDU команду.
*  @b BYTE * answer;  - Буфер для ответа от карты на APDU команду.
*****************************************************************************************
*  @return
*  @b WORD;    - Длинна полученного ответа от карты.
*****************************************************************************************
*/
WORD WINAPI SendApduOut(BYTE * cmd, WORD * stat, BYTE * answer, WORD answer_max_len);

/**
*****************************************************************************************
*  @fn BYTE WINAPI CheckConnect()
*  @b Назначение:                                                    \n
*  @brief Проверяет наличие карты в ридере. \n
*  @author
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Проверяет флаг наличия карты в ридере. Флаг выставляется после выполнения TakeCard(), если
была подача карты.
*  @endcode
*****************************************************************************************
*  @param Вход: \n
*  @b None
*
*  @param Выход: \n
*  @b None
*****************************************************************************************
*  @return
*  @b BYTE;    - Результат выполнения.
*  @b 1 - Карта присутствует.
*  @b 0 - Карта отсуттствует.
*****************************************************************************************
*/
BYTE WINAPI CheckConnect();
```

Фнкции:

BYTE WINAPI SelectReader(LPSTR rname,DWORD rname_size);
BYTE WINAPI OpenReader(LPSTR rname);
BYTE WINAPI CloseReader(void);

В настоящее время не используются, и оставлены для возможного использования в будущем.

В состав проекта входит SmartCard.dll, которая, обеспечивает обмен данными между EMV библиотеой и считвателем. Считыватель может быть подключён через USB или через RS232.

В комплект постави входит демонстрационный проект SmartCard.dll, который содержит минимальный набор функций и может служить для подключения собственного драйвера управления считывателем.

Демонстрационный проект SmartCard.dll, содержит фунцию, которая эмулирует ответы считывателя на запросы библиотеки EMV. (emv_ker.dll)

Подключение демонстрационного проекта SmartCard.dll

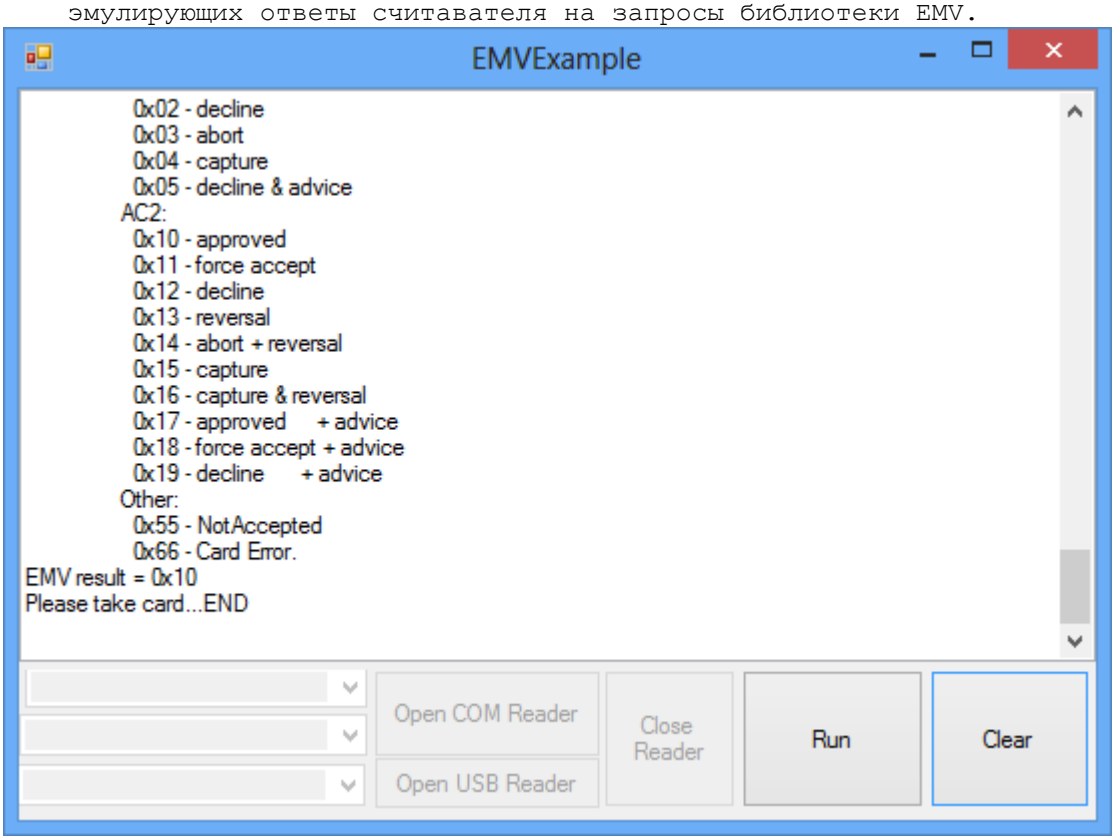
a) Переименовать \EMVExampleDotNet\bin\Release\SmartCard.dll,которая обеспечивает связь с реальным считывателем, скажем в SmartCard_full.dll.
b) Скопировать демонстрационную библиотеку «SmartCard.dll» в папку "\EMVExampleDotNet\bin\Release\".
c) Установить макроопределение **USE_CALLBACK_FOR_CARD_READER** в начале файлов Forms1.cs and EmvKerDll.cs. Макроопределение **USE_CALLBACK_FOR_CARD_READER**

"выключает" из проекта методы и функции демонстрационного приложенияуправления, которые управляют считывателем посредством SmartCard.dll.

d) Установить мароопределение **USE_DEMO_SMART_CARD_SIMULATOR_DLL** в начале файла EmvKerDll.cs. Макроопределение **USE_DEMO_SMART_CARD_SIMULATOR_DLL** - "выключает" из проекта регистрацию функций обратного вызова для управления считывателем.

e) Собрать и запустить демонстрационный проект.

При нажатии кнопки RUN,должна выполняться транзация на основе данных эмулирующих ответы считавателя на запросы библиотеки EMV.



2) Управление считывателем через функции обратного вызова:

Макроопределение **USE_CALLBACK_FOR_CARD_READER** позволяет собрать демонстрационный проект, в котором управление считывателем осуществляется через функции обратного вызова.

Макроопределение **USE_CALLBACK_FOR_CARD_READER** должно быть установленно одновременно, в начале файлов Forms1.cs and EmvKerDll.cs.

Макроопределение **USE_DEMO_SMART_CARD_SIMULATOR_DLL** должно быть "выключено".

Для управления считывателем, библиотека EMV вызывает седующие функции обратного вызова:

```
/**
*****************************************************************************************
*  @fn WORD WINAPI emv_ApduIn(BYTE *Cmd,BYTE * Data,WORD *Data_len,BYTE *answer,WORD answer_max_len)
*  @b Назначение:                                                                  \n
*  @brief Посылает APDU команду с данными и получает ответ. Используется EMV ядром.  \n
*  @author
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Посылает APDU команду и получает ответ.
*  2.) Возвращает данные в буфер.
*  @endcode
```

```
*****************************************************************************************
*  @param Вход: \n
*  @b BYTE * Cmd; - Буфер, содержащий APDU команду.
*  @b BYTE * Data;        - Буфер, содержащий данные.
*  @b WORD *Data_len;     - Длина даннных.
*  @b WORD answer_max_len;         - Размер буфера для ответа.
*
*  @param Выход: \n
*  @b BYTE * answer;      - Буфер для ответа от карты на APDU команду.
*  @b WORD *Data_len;     - Длина полученых от карты даннных.
*****************************************************************************************
*  @return
*  @b WORD;       - Статус, полученный от карты на APDU команду.
*****************************************************************************************
*/
public static unsafe ushort emv_ApduIn(byte* cmd, byte* data, ushort* data_len, byte* answer, ushort
answer_max_len);

/**
 *****************************************************************************************
 *       @fn WORD emv_ApduOut(BYTE *Cmd,WORD *Stat,BYTE *answer,WORD answer_max_len)
 *  @b Назначение:                                                          \n
 *       @brief Посылает APDU команду и получает ответ. Используется EMV ядром.      \n
 *  @author
 *****************************************************************************************
 *  @b Выполняет:
 *       @code
 *  1.) Посылает APDU команду и получает ответ.
 *  2.) Возвращает данные в буфер.
 *       @endcode
 *****************************************************************************************
 *  @param Вход: \n
 *  @b BYTE * Cmd;         - Буфер, содержащий APDU команду.
 *  @b WORD answer_max_len;        - Размер буфера для ответа.
 *
 *  @param Выход: \n
 *  @b WORD * Stat;        - Возвращается статус, полученный от карты на APDU команду.
 *  @b BYTE * answer;      - Буфер для ответа от карты на APDU команду.
 *****************************************************************************************
 *  @return
 *  @b WORD;      - Длинна полученного ответа от карты.
 *****************************************************************************************
*/
public static unsafe ushort emv_ApduOut(byte* cmd, ushort* stat, byte* answer, ushort answer_max_len);

/**
 *****************************************************************************************
 *       @fn BYTE emv_Atr(BYTE *atr, DWORD * max_len, BYTE * protocol)
 *  @b Назначение:                                                              \n
 *       @brief Получает ATR, используется EMV ядром.       \n
 *  @author  Ilya Yalin
 *****************************************************************************************
 *  @b Выполняет:
 *       @code
 *  1.) Посылает ридеру команду подать питание на карточку.
 *  2.) Получает от карточки ATR.
 *       @endcode
 *****************************************************************************************
 *  @param Вход: \n
 *  @b DWORD * max_len;   - Максимальный размер буфера для приёма данных ATR.
 *
 *  @param Выход: \n
 *  @b BYTE *atr - Указатель на буфер для данных ATR.
 *  @b DWORD * max_len;   - Размер полученных данных.
 *  @b BYTE * protocol;   - Адрес для получения протокола. Для моторизованных ридеров всегда 0.
 *****************************************************************************************
 *  @return
 *  @b BYTE;      - Статус выполнения операций.
 *       @b 1 - операция завершилась успешно.
 *       @b 0 - операция завершилась не удачей.
 *****************************************************************************************
*/
public static unsafe byte emv_Atr(byte* dest_atr, uint* max_len, byte* protocol);

/**
*****************************************************************************************
*  @fn BYTE emv_CheckConnect()
*  @b Назначение:                                                          \n
*  @brief Проверяет наличие карты в ридере.        \n
*  @author
*****************************************************************************************
*  @b Выполняет:
*  @code
*  1.) Проверяет флаг наличия карты в ридере. Флаг выставляется после выполнения TakeCard(),
*  если была подача карты.
```

```
*  @endcode
*****************************************************************************************
*  @param Вход: \n
*  @b None
*
*  @param Выход: \n
*  @b None
*****************************************************************************************
*  @return
*  @b BYTE;      - Результат выполнения.
*  @b 1  - Карта присутствует.
*  @b 0  - Карта отсуттствует.
*****************************************************************************************
*/
private static byte emv_CheckConnect();
```

Перед первым использованием, функции обратного вызова,должны быть зарегистрированны в EMV библиотеке:

```
emv_kernel_registry_ApduIn         (hEmvInit, _emv_ApduIn);
emv_kernel_registry_ApduOut        (hEmvInit, _emv_ApduOut);
emv_kernel_registry_Atr            (hEmvInit, _emv_Atr);
emv_kernel_registry_CheckConnect   (hEmvInit, _emv_CheckConnect);
```

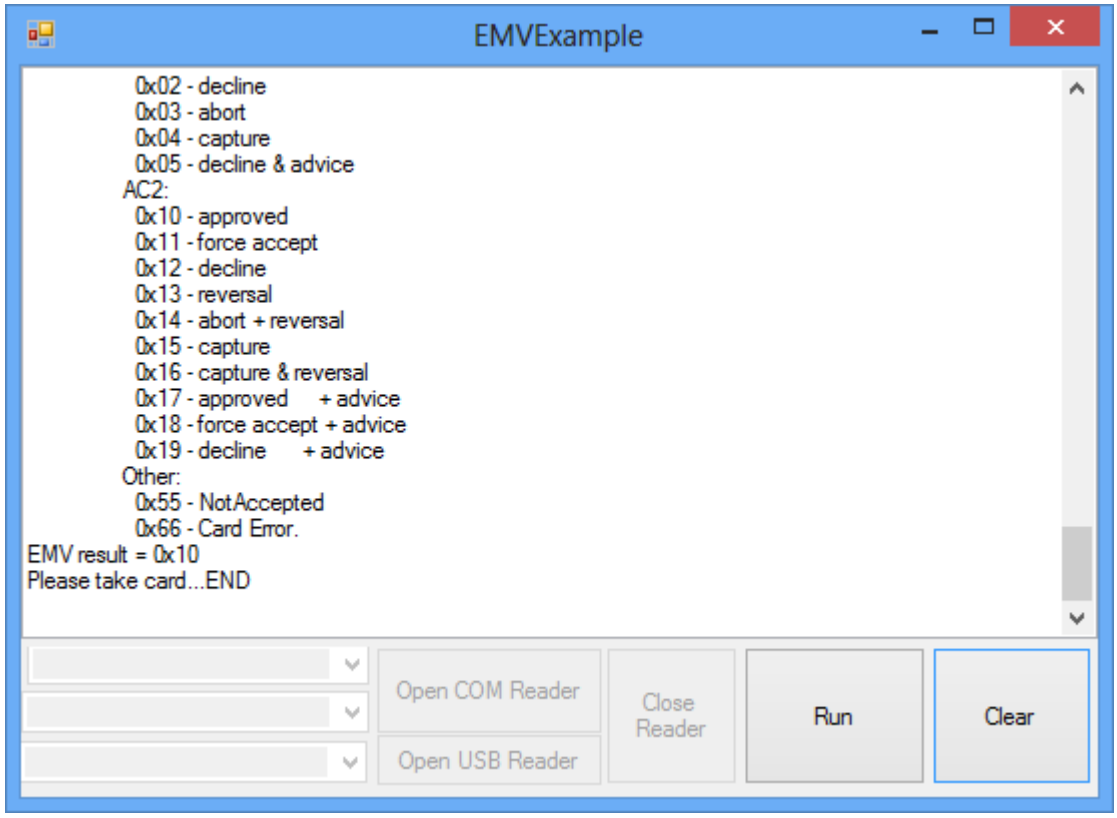
Демонстрационный пример содержит фунцию, которая эмулирует ответы считывателя на запросы библиотеки EMV.

Подключение демонстрационного проекта для управлением считывателем посредством функций обратного вызова:

a) Установить макроопределение USE_CALLBACK_FOR_CARD_READER в начале файлов Forms1.cs and EmvKerDll.cs.Макроопределение USE_CALLBACK_FOR_CARD_READER "выключает" из проекта методы и функции демонстрационного приложенияуправления, которые управляют считывателем посредством SmartCard.dll.
b) Выключить мароопределение USE_DEMO_SMART_CARD_SIMULATOR_DLL в начале файла EmvKerDll.cs. Макроопределение USE_DEMO_SMART_CARD_SIMULATOR_DLL - "выключает" из проекта регистрацию функций обратного вызова.
c) Собрать и запустить демонстрационный проект.

Демонстрационный пример содержит фунцию, которая эмулирует ответы считывателя на запросы библиотеки EMV.

При нажатии кнопки RUN,должна выполняться транзация на основе данных эмулирующих ответы считавателя на запросы библиотеки EMV.


EMVExample
0x02 - decline
0x03 - abort
0x04 - capture
0x05 - decline & advice
AC2:
0x10 - approved
0x11 - force accept
0x12 - decline
0x13 - reversal
0x14 - abort + reversal
0x15 - capture
0x16 - capture & reversal
0x17 - approved + advice
0x18 - force accept + advice
0x19 - decline + advice
Other:
0x55 - NotAccepted
0x66 - Card Error.
EMV result = 0x10
Please take card...END
Open COM Reader
Open USB Reader
Close Reader
Run
Clear