Работа со считывателями

Устройства, требования к устройствам, принципиальная схема подключения. Настройка декодирования карт, логическая схема работы со считывателями

- Устройства и технические требования
  - Рекомендуемые модели
- Подключение считывателя к ПО Абонемент
  - Настройка декодирования карт Em-marine (кратко)
  - Настройка декодирования для Mifare (подробно)
  - Настройка считывателя для Честного знака
  - Принцип декодирования с накладыванием маски на примере использования карт Mifare и Em-marine
  - Принцип декодирования с использованием FastScript
  - Алгоритмы декодирования
  - Принципиальная схема работы со считывателем в AbonementManager через TCPCardReader
  - Подключение сканера штрих-кодов
- Конфигурационные параметры

Устройства и технические требования

Рекомендуемые модели

Наиболее часто используемая модель считтывателя IronLogic Z-2 (мод RD_ALL)

Подключение считывателя к ПО Абонемент

Настройка TCPCardReader, настройка декодирования, схема взаимодейтсвия

Настройка декодирования карт Em-marine (кратко)

Для декодирования необходимо скачать и разархивировать утилиту **TcpCardReader** из базовой поставки ПО Абонемент. Далее необходимо настроить **TcpCardReader.ini**, в моем случае настройки выглядят так :

```
[application]
AutoStart = 1
;Автоматически стартовать программу после запуска
AutoHide = 1
;Сворачивать программу после запуска
[setup]
COMPORTCOUNT=1
;Количество подключений к COM портам.
[TCP]
PORT=7760
IP=0.0.0.0
;TCP порт и адрес сетевого интерфейса, к которому будет производиться подключение клиентов
[READER1]
COMPORT=7
COMPORTPARAMS=baud=9600 data=8 parity=N stop=1
;Настройки первого подключения к ком порту.  Номер и параметры COM порта.
Prefix=;
;Строка, которая передаётся клиентам перед первым прочтенным символом.  В данном примере добавится символ: «точка с запятой»
Postfix=?
;Строка, которая передаётся клиентам после последнего прочтенного символа.  В данном примере добавится символ: «Вопросительный знак»
useDecodeCards=1
;0 - не использовать декодирование decodecards.dll,  1 - использовать декодирование decodecards.dll
StartTermChars=M
;Используется, если useDecodeCards=1 Признак начала трека для декодирования
```

```
FinishTermChars=?#13;#10;#0;
;Используется, если useDecodeCards=1 Признак конца трека для декодирования
[LOG]
level=30
debuglevel=30
showlog=1
;Уровни логирования. Используются значения от 0 до 30
[TCPCARDREADER]
Prefix=;
;Строка, которая передаётся клиентам перед первым прочтенным символом. В данном примере «точка с запятой»  появится перед треком карты (что служит указателем начала для decodecards.dll)
Postfix=?
;Строка, которая передаётся клиентам после последнего прочтенного символа. В данном примере «Вопросительный знак» появится после трека карты (что служит указателем конца для decodecards.dll)
```

Настройки для конкретного считывателя в секции с именем READER и без пробелов любая цифра, например:

```
[READER1]
COMPORT=7
COMPORTPARAMS=baud=9600 data=8 parity=N stop=1
Prefix=;
Postfix=?

useDecodeCards=1

StartTermChars=i
FinishTermChars=?#13;#10;#0;
```

COMPORT - номер com порта на котором расположен считыватель(определяется через диспетчер устройств).

Prefix и Postfix символы которые будут добавит *TcpCardReader* добавит к треку перед отправкой клиентам.

useDecodeCards определяет будет ли использоваться декодирование с использованием *decodecards.dll*

При включение декодирования используются параметры:

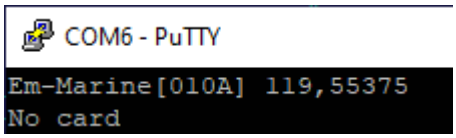
StartTermChars - определяет символ после которого начинается трек

FinishTermChars - определяет набор символов после одного из которых заканчивается трек.

В данном примере мы настраиваем использование считывателя под карты Em-Marine:

включаем декодирование *useDecodeCards=1*

и определяем в *StartTermChars* символ "i" и *FinishTermChars* символы ?#13;#10;#0;



В decodecards.ini в секции [mask]

```
[mask]
  card_em = ne*
```

В ней настраиваются маски карт в формате - ИМЯ МАСКИ=МАСКА

и в секции с именем ИМЯ МАСКИ настраиваем алгоритм декодирования

```
[cards.card_em]
ExcludedPrefix=ne[
code=2
CardDecodeType = MASK
mask=****dddddddd
```

ExcludedPrefix исключаем из работы алгоритма префикс "ne["
code=2
CardDecodeType - алгоритм декодирования MASK
mask - применяемая маска ****dddddddd

/Abonementmanger

//gkhostconnect

///Использование GKHost

//Использование TCPCardReader (GK)

//connecter

Настройка декодирования для Mifare (подробно)

1. У тебя есть читалка (допустим z-2) и читалка турникета (matrix)
2. Главная задача это привести треки получаемые со считалок в один единый вид (в 10 формате)
3. Если
3.1 Mifare
3.1.1 Скачиваешь утилиту Putty
3.1.2 Подключаешься через Putty к читалке
3.1.3 Сканируешь карту и получаешь трек **Mifare[A1B2C3D4] (какая то номерная последовательность, которая совершенно не интересна нам) <терминальные символы>**
3.1.4 Открываешь version.txt из каталогов TCPCardReader и decodercards (лежит в сборке в UTILS)
3.1.5 ~~Начинаешь читать и разбираться какие параметры нужны~~
3.1.6 Итак, сперва, чтобы настроить декодирование в TCPCardRead нам нужно отделить искомую последовательность от остальной информации. В данном случае необходимо вытащить UID из трека Mifare[**A1B2C3D4**] (какая то номерная последовательность, которая совершенно не интересна нам) <терминальные символы> **(выделено жирным, без скобок !)**.
3.1.7 Открываем TcpCardReader.ini удаляем от туда секцию [TCPCARDREADER] (если есть (мы её не используем))
3.1.8 Смотрим секции типа [READER1], [READER2], ... [READERN] - эти секции отвечаю за каждую свою подключённую читалку. Если нам нужна 1 читалка, то делаем секцию [READER1], если две, то записываем секцию [READER2] и т.д.
3.1.9 Итак, настроим, например, только 1 считыватель. Смотрим секцию [READER1]

```
[READER1]
COMPORT=9
COMPORTPARAMS=baud=4800 data=8 parity=N stop=1
;Параметр Prefix задаёт строку, которая передаётся клиентам перед первым прочтенным символом
Prefix=
;Параметр Postfix задаёт строку, которая передаётся клиентам после последнего прочтенного символа
Postfix=
```

```
;Использовать decodecards.dll
useDecodeCards=0
StartTermChars=;
FinishTermChars=?#13;#10;#0;

;Файл логирования треков
TrackLogFile=
ConvertTrackLog=0
IncludeTypeStartTermChar=0
IncludeStartTermChar=0
NeedCheckAndReopenCom=1
```

3.1.10 Смотрим параметры из version.txt (например, через NotePad++)

{
COMPORT=9 - номер порта берется из диспетчера устройств. (диспетчер устройств -> порты)
COMPORTPARAMS=baud=4800 data=8 parity=N stop=1 (параметры подключения) практически некогда не меняется (берется из свойств порта. см. диспетчер устройств)
}
;Параметр Prefix задаёт строку, которая передаётся клиентам перед первым прочтенным символом
Prefix=

> (строка №7 из version.txt) (";Параметр Prefix задаёт строку, которая передаётся клиентам перед первым прочтенным символом") Это означает, что передаваемый трек клиентам будет начинаться с установленного префикса. Например, если предположим, что декодирование настроено, то пусть наш трек выглядит так 1200. Мы хотим передавать трек с префиксом ;, то устанавливаем в параметр Prefix=; При передаче трека клиентам (abonementmanager, connecter, ...) передастся трек ;1200 . Если мы хотим передать трек с префиксом mrgreen, то ставим в параметрах Prefix=mrgreen. Соответственно при передаче трека клиентам, они получат mrgree1200

;Параметр Postfix задаёт строку, которая передаётся клиентам после последнего прочтенного символа
Postfix=

> (строка №9 из version.txt) (";Параметр Postfix задаёт строку, которая передаётся клиентам после последнего прочтенного символа") По аналогии с суффиксом. Это означает, что передаваемый трек клиентам, оканчивается на установленный суффикс.

> Например, хотим передать суффикс ?, то Postfix=?. Клиенты получат трек 1200?. Если хотим передать суффикс `?#, то ставим Postfix=`?#. Клиенты получат трек 1200@?#. Таким образом, если установлены параметры Prefix=; и Postfix=?, то клиенты получат трек ;1200?

;Использовать decodecards.dll
useDecodeCards=0

> (строка №27 из version.txt) использовать= 1 - не использовать=0 decodecards.dll для извлечения номера. По умолчанию 0 - не использовать.

StartTermChars=;

> (строка №28 из version.txt) "Стартовые символы. Используются, если useDecodeCards=1" Стартовые символы или управляющие. Это символ(ы), с которого идет наш необработанный трек, передаваемый в decodercards.

FinishTermChars=?#13;#10;#0;

> (строка №29 из version.txt) Терминальные символы. Используются, если useDecodeCards=1. Это символ(ы), с которого идет наш не обротанный трек, передаваемый в decodercards.

Т.е.
Допустим Трек **Mifare[A1B2C3D4] (какая то номерная последовательность, которая совершенно не интересна нам) <терминальные символы>**

> (**терминальный символ зависит от прошивки/настройки устройства**, он передается вместе с данными, **как правило это перевод на новую строку и возврат каретки** и - №10 и №13 символы таблицы ASCII
>
> https://ru.wikipedia.org/wiki/ASCII ; получаемая последовательность со считывателя в TPCCardReader или Putty выглядит наподобие Mifare[A1B2C3D4] (....) <LF><CR>).

Если сделаем **StartTermChars=**i и **FinishTermChars=?#13;#10;#0;** то будет следующее:
1. В треке находиться начальный символ, указанный в **StartTermChars**, это **i**. В нашем случае M **i** fare[A1B2C3D4] (....) <LF><CR>
2. Затем ищется окончание (последний символ), указанные в **FinishTermChars** (можно указывать несколько подряд).
Т.е. смотрим на Mifare[A1B2C3D4] (....) <LF><CR>,
есть ли в данном треке символ "?" - нету

есть ли символ #13; (синтаксис 13 элемента в таблице в TCPCardReader - это <CR> возврат каретки), есть Mifare[A1B2C3D4] (....) <LF> **<CR>**
Ага, **последовательность найдена** и **результат будет** : **fare[A1B2C3D4] (....) <LF>** и **он уже передается на обработку в decodercards.dll**

;Файл логирования треков
TrackLogFile=
ConvertTrackLog=0
(строка №107 из version.txt)

> Если TrackLogFile определен, то используется дополнительный параметр ConvertTrackLog.
> Он определяет, каким образом логировать невизуальные символы. По умолчанию ConvertTrackLog=0 (не конвертировать).
> Если ConvertTrackLog=1, то невизуальные символы конвертируются в их код.

IncludeTypeStartTermChar=0 - (строка №126 из version.txt)

> 1.1. IncludeTypeStartTermChar - тип обработки стартового символа
> Имеет смысл, если useDecodeCards=1, IncludeStartTermChar=1 и UseOriginalTrack=0
> Если IncludeTypeStartTermChar=0 (по умолчанию), то стартовый символ НЕ передается в decodecards, а подставляется перед истинным номером как дополнительный префикс
> Если IncludeTypeStartTermChar=1 , то стартовый символ передается в decodecards и участвует в извлечении истинного номера

Обычно используем по умолчанию

NeedCheckAndReopenCom=1 - (строка №135 из version.txt)

> Если NeedCheckAndReopenCom=1 (по умолчанию), то постоянно контролируется состояние соединения. Если связь потеряна, то происходит попытка переподключиться.
> NeedCheckAndReopenCom=0 можно устанавливать, если устройство подключается не к USB, а к COM-порту.

Оставляем 1.
3.1.11 Смотря пункт 3.1.10, составляем следующую конфигурацию в секции **[READER1]**

```
COMPORT=9
COMPORTPARAMS=baud=4800 data=8 parity=N stop=1
;Параметр Prefix задаёт строку, которая передаётся клиентам перед первым прочтенным символом
Prefix=;
;Параметр Postfix задаёт строку, которая передаётся клиентам после последнего прочтенного
символа
Postfix=?

;Использовать decodecards.dll
useDecodeCards=1
StartTermChars=i
FinishTermChars=?#13;#10;#0;
```

3.1.12 Итак, получили на входе в decodercards.dll трек на обработку **fare[A1B2C3D4] (....) <LF>**
3.1.13 Разбираемся с декодированием. Открываем ini из сборки и видим

```
[frf]
fsUnit=fsdecodecard.upas

[general]
usemask=1
TrackResultLog=

[mask]
  card = 05015005*
  card2= 778=12345678=*
  cardfile=file://cards.txt

[cards]
  CardPrefix   = 05015005
;                 05015005280507210
  CardPrefix   = 750=
  CardPrefix   = 778=999999999=
  CardPrefix   = 778=123456789=
  StaffCardPrefix        =  778=87121234=
  StaffCardPrefix        = 778=201050001
```

```
  RegularCardPrefix = 778=444444444=
  PDSCardPrefix   = 811q

[cards.cardREPLACECARDNO]
  ExcludedPrefix=
  code=2
  CardDecodeType = REPLACECARDNO
  NewCardNo = 5678956

[cards.ARRAYOFBYTES]
  CardDecodeType = MASK
  mask=*cccc*
  bitmask=$7FFFFFFF
  MaskType=ARRAYOFBYTES
  code=2

[cards.Card2]
ExcludedPrefix=778=12345678=
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=2

;  CardDecodeType = LAST8
;  CardDecodeType = LAST9

;CardDecodeType = MASK
mask=hhhhhhhh*
bitmask=$7FFFFFFF

[cards.Card]
ExcludedPrefix=05015005
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=2

;  CardDecodeType = LAST8
;  CardDecodeType = LAST9

CardDecodeType = MASK
mask=ddd
```

```
;mask=hhhhhhhh*
bitmask=$FFFFFFFF

DecodeTypeEx = 0

[cards.RegularCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=22

[cards.StaffCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=21

[cards.PDSCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=24
```

3.1.14 Разберем по секциям

```
[frf]
fsUnit=fsdecodecard.upas
```

Эта секция отвечает подключение файла со скриптом (FastScript). Можно написать свой алгоритм для обработки трека.

```
[general]
usemask=1
TrackResultLog=
```

TrackResultLog= - имя файла, в котором будет результат обработки трека. (если надо делаем, обычно убираем)

usemask=1

> (version.txt строка 41)
>
> Если usemask=1, то секция [cards] игнорируется
> Если usemask=1, тогда при декодировании проверяется, какой маске соответствует считанный трек.
> После этого происходит обращение к секции [cards.ИМЯ МАСКИ]

Ставим единичку и создаем секцию [mask] (если нет), в ней прописываем
card_mi (произвольное имя) = (маска, по которой смотрим в какую секцию отправить на обработку трек)
В наем примере сделаем так:

```
card_mi = fare* (fare* берется из трека fare[A1B2C3D4] (....) <LF>, * - любой символ)
```

Таким образом, **все треки, которые содержат fare будут определяться и направляться на обработку в секцию card_mi**

Также в этом случае необходимо создать секцию [cards.card_mi], если её нет !

В этой секции могу содержаться различные параметры (см. version.txt), но рассмотрим под наш случай.
Нужно будет для обработки трека полученного **fare[A1B2C3D4] (....) <LF>** использовать следующие параметры: **ExcludedPrefix**, **code**, **CardDecodeType**, **mask**.

ExcludedPrefix-Отрезаем от трека префикс.
Для того, чтобы корректно обработать наш трек fare[**A1B2C3D4**] (....) <LF> , необходимо **удалить fare[**.
Тогда необходимо прописать в параметре **ExcludedPrefix** следующее:
ExcludedPrefix=fare[
Таким образом, мы получим внутри decodercards следующую последовательность трека **A1B2C3D4**] (....) <LF>, где жирным указано интересующая нас последовательность в шестнадцатеричной системе.

Далее используется параметр **CardDecodeType - этот параметр определяет алгоритм декодирования**. Существуют несколько вариантов, например, LAST8, LAST9, MASK, FastScript и т.п.
В нашем случае мы используем **MASK** (накладываем маску и сравниваем с нашим треком)
CardDecodeType = MASK
далее необходимо добавить параметр mask, который собственно, и определяет её.

> Из version.txt строка 32 описание следующее
>
> mask=**hh* - взять 3-й и 4-й символ как цифры в шестнадцатеричном формате
> или
> mask=**dd* - взять 3-й и 4-й символ как цифры в десятичном формате

Поскольку нам из трека **A1B2C3D4**] (....) <LF> интересна лишь последовательность A1B2C3D4, то наша маска выглядит следующим образом: **mask = hhhhhhhh**. Т.е. берем символы с 1 по 8 в шестнадцатеричном формате и преобразуем в конечный вариант в десятичный формат (DEC).

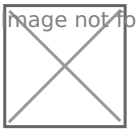
Остается для обработки добавить лишь параметр **code** - он отвечает за тип карты. code=2 (*фичи с ним на обучении только, либо сами*)

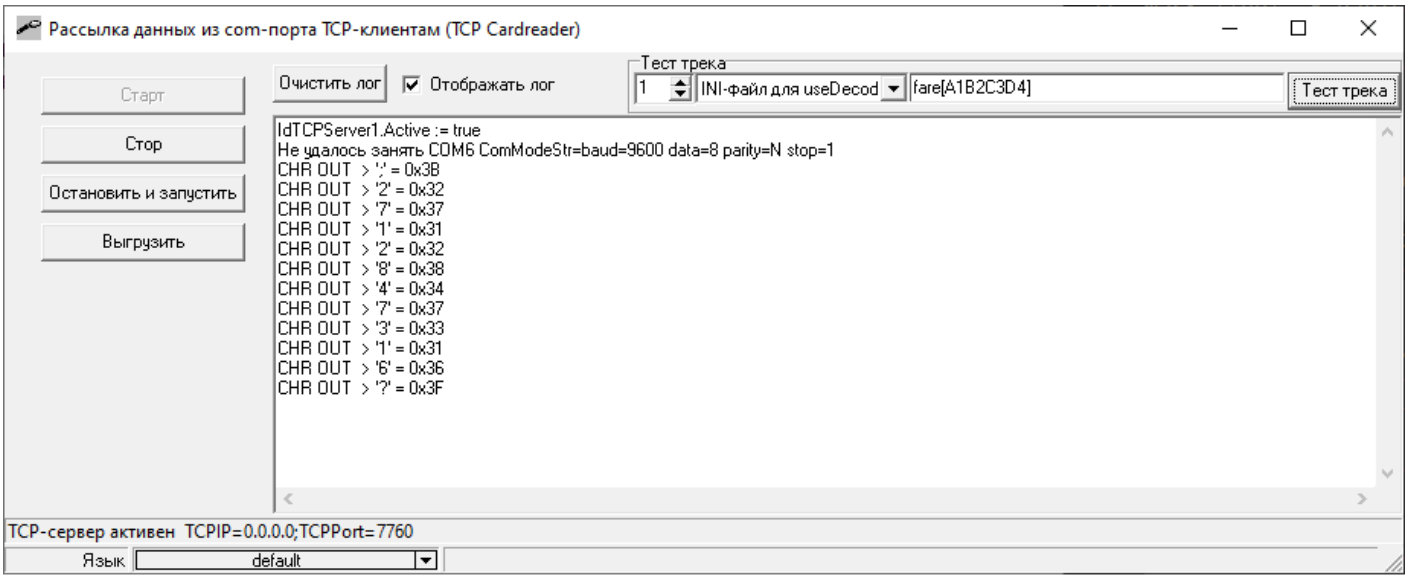
На выходе при треке **A1B2C3D4 должны получить 2712847316**

3.1.15 Таким образом **получаем следующую конфигурацию decodercards.ini при TCPCardReader**

```
[frf]
fsUnit=fsdecodecard.upas

[general]
usemask=1

[mask]
  card_mi = fare*

[cards.card_mi]
ExcludedPrefix=fare[
code=2
CardDecodeType = MASK
mask=hhhhhhhh
```

3.1.16 Можно проверить правильность настроек **с помощью теста** в **TCPCardReader**. В поле слева от кнопки тест, **вводиться трек, который должен передаться в decodercards.dll**





В отображение вы должны получить последовательность отправки символов клиентам (трек).

Если вместо этого получаете :
CHR OUT> ;
CHR OUT> 0
CHR OUT> ?
Вы неправильно настроили декодирования - повторяете изучение материала с 1 пункта

3.1.16 Можно также убедиться по логам decodercards, при использовании декодировании в логах TCPCardReader можно увидеть только информацию отправляемую клиентам.

Лог правильно настроенного TCPCardReader:

```
22.09.2021 18:05:35 581059359 2C74> CALL extractcardnoexp_int64: track=fare[F2D1BF1F]
191,53746 1K (0004,08) ; term0= , term1=
22.09.2021 18:05:35 581059359 2C74> CALL extractcardnoex: track=fare[F2D1BF1F] 191,53746 1K
(0004,08) ; term0= , term1=
22.09.2021 18:05:35 581059359 2C74> UseMask2: UseMask=1
22.09.2021 18:05:35 581059359 2C74> CALL extractcardnoex2: track=fare[F2D1BF1F] 191,53746 1K
(0004,08) ; term0= , term1=
22.09.2021 18:05:35 581059375 2C74> EXIT extractcardnoex2: result=4073832223 , cardcode=2 ,
ExtractWithoutPrefix2=F2D1BF1F] 191,53746 1K (0004,08), ci=card_mi
22.09.2021 18:05:35 581059375 2C74> EXIT extractcardnoexp_int64: result=4073832223 ,
cardcode=2
```

CALL extractcardnoexp_int64: track=fare[F2D1BF1F] 191,53746 1K (0004,08) ; term0= , term1= | **- прилетел трек**

EXIT extractcardnoex2: result=4073832223 , cardcode=2 , ExtractWithoutPrefix2=F2D1BF1F] 191,53746 1K (0004,08), ci=card_mi | **ci - имя секции, по которой трек обрабатывается;**

ExtractWithoutPrefix2=F2D1BF1F] 191,53746 1K (0004,08) - **результат после исключения из трека префикса,** | result=4073832223 -**результат декодирования** **EXIT extractcardnoexp_int64: result=4073832223 , cardcode=2** | **результат на выходе из decodercards.dll !!!**

3.1.17 **После правильного настроенного TCPCardReader**, **необходимо настроить decodercards.ini при abonementmanager или при коннекторе, или при gkhostconnect.**
Рассмотрим простой и стандартный вариант- это Abonementmanager
3.1.18 Открываем decodercards.ini и видим

```
[frf]
fsUnit=fsdecodecard.upas

[general]
usemask=1
TrackResultLog=

[mask]
  card = 05015005*
  card2= 778=12345678=*
  cardfile=file://cards.txt

[cards]
  CardPrefix   = 05015005
;                 05015005280507210
  CardPrefix   = 750=
  CardPrefix   = 778=999999999=
  CardPrefix   = 778=123456789=
  StaffCardPrefix        =  778=87121234=
  StaffCardPrefix        = 778=201050001
  RegularCardPrefix = 778=444444444=
  PDSCardPrefix   = 811q
```

```
[cards.cardREPLACECARDNO]
  ExcludedPrefix=
  code=2
  CardDecodeType = REPLACECARDNO
  NewCardNo = 5678956

[cards.ARRAYOFBYTES]
  CardDecodeType = MASK
  mask=*cccc*
  bitmask=$7FFFFFFF
  MaskType=ARRAYOFBYTES
  code=2

[cards.Card2]
ExcludedPrefix=778=12345678=
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=2

;  CardDecodeType = LAST8
;  CardDecodeType = LAST9

;CardDecodeType = MASK
mask=hhhhhhhh*
bitmask=$7FFFFFFF

[cards.Card]
ExcludedPrefix=05015005
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=2

;  CardDecodeType = LAST8
;  CardDecodeType = LAST9

CardDecodeType = MASK
mask=ddd
;mask=hhhhhhhh*
bitmask=$FFFFFFFF
```

```
DecodeTypeEx = 0

[cards.RegularCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=22

[cards.StaffCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=21

[cards.PDSCard]
;  CardDecodeType = ANGSTREMCARD
;  CardDecodeType = ANGSTREMBRASLET
code=24
```

3.1.18 Приводим к этому виду, удалив и почистив почти все

```
[general]
usemask=1

[mask]
card_all = *

[cards.card_all]
code=2

[cards.Card]
code=2

[cards.RegularCard]
code=22

[cards.StaffCard]
code=21

[cards.PDSCard]
code=24
```

Подробнее

```
[mask]
card_all = *
```

Как и по аналогии с TCPCardReader делаем настройку по маске.
card_all = * | * - все получаемые треки. Поскольку abonementmanager подключается к TCPCardRead, на стороне которого происходят все преобразования треков, получаемых со считывателей, в абонемент manager прилетает TCP **только трек в десятичном формате**. Т.е. исходя из выше обработанного к нам прилетает **2712847316**. Соответственно, мы уже получили почти готовый трек.
Создаем секцию **[cards.card_all]**
И устанавливаем "физический" тип карты **code=2.**

Также для того, чтобы корректно отрабатывалось определение физического типа рекомендуется использовать во всех клиент с decodercards.dll данные секции:

[cards.Card] - гостевая
code=2

[cards.RegularCard] - постоянная
code=22

[cards.StaffCard] - карта персонала
code=21

[cards.PDSCard] - ПДС
code=24

Соответственно, **при прикреплении карты в базу будет прикрепляться трек с физическим типом 2**. **При наличии служебных секций он определится, как гостевой**

3.1.19 Осталось настроить декодирование при gkhostconnect

3.1.20 При использовании **GKHOST в GKHOSTCONNECT** на основе выше перечисленного трека прилетают карты вида **CARD A1 B2 D3 C4** !

На конце обычно еще бывает F (*CARD A1 B2 D3 C4*FF ...)зависит от прошивки считывателя

3.1.21 Открываем decodercards.ini при gkhostconnect и удаляем все, оставляя только необходимое и приводим к следующему виду

```
[frf]
fsUnit=fsdecodecard.upas

[general]
usemask=1
TrackResultLog=

[mask]
  card = *

[cards.card]
  code=2
  CardDecodeType = MASK
  mask=hhhhhhhh
```

Поскольку к нам прилетает карта сразу HEX значении. А при передаче в decodercards элементы #### CARD откидываются автоматически, остается лишь преобразовать hex в dec. И поставить корректный тип карты в нашем случае это 2.

4. Готово. ~~(отходим и нервно курим в сторонке)~~

Дополнительные материалы

Настройка_decodecards_.txt

version (decodercards).txt

version (TCPCardReader).txt

Настройка считывателя для Честного знака

Принцип декодирования с накладыванием маски на примере использования карт Mifare и Em-marine

Декодирование на стороне TCPCardReader'а включается строкой

```
useDecodeCards=1
```

в TCPCardReader.ini в настройках конкретного считывателя:

```
[READER1]
COMPORT=6
COMPORTPARAMS=baud=9600 data=8 parity=N stop=1
Prefix=;
Postfix=?
useDecodeCards=1
StartTermChars=i
FinishTermChars=?#13;#10;#0;
```

Таким образом при включении декодирования TCPCardReader отправляет часть трека начинающегося с символа следующего за *StartTermChars* и заканчивающимся символом стоящим перед одним из *FinishTermChars в* Decodecards*.*

Итак Decodecards получает только эту часть и декодирует (преобразовывает/выделяет идентификатор) её.

В **decodecards.ini** В секции **[mask]** перечисляются маски по префиксам которых можно разделять декодирование разных карт

```
[mask]
card_em = ne*
card_mifare = fare*
card = *
```

В данном примере если пришедший от TCPCardReader'a трек содержит префикс "ne" ,то обработка такого трека будет описана в секции [cards.**card_em**]. Соответственно если трек содержит префикс "fare" то обработка такого трека описана в секции [cards.card_mifare]. Символ "*" в данном случае означает любое количество символов.

`Чтение данной секции происходит последовательно, то есть при первом совпадении пришедшего трека с маской дальнейшей просмотр секции не ведётся`

Далее в секциях с названием "cards." и левой частью до знака равенства в секции [mask] конкретного типа трека настраивается декодирование.

```
[cards.card_em]
ExcludedPrefix=ne[
code=2
CardDecodeType = MASK
mask=****dddddddd
DecodeTypeEx = 0
```

Опцией CardDecodeType = MASK выбираем алгоритм декодирования с накладыванием маски.

Сначала удаляется часть не участвующая в декодировании используя параметр ExcludedPrefix.

Далее происходит декодирование(выделение трека) применением маски **mask=.**

```
mask=****dddddddd
```

Здесь * означает любой один символ который игнорируется.

Символы "d" или "h" означают систему счисления в которой пришел в decodecards.

h - HEX 16-тиричная (Hexadecimal)

d - DEC 10-тиричная (Decimal)

В этом примере обрабатываются первые 12 символов(их может быть больше 12 в пришедшем треке) из которых первые 4 игнорируются, а следующие 8 воспринимаются как десятичное число. Если на их месте появляется символ который не возможно преобразовать к числу в десятичной системе счисления декодирование произойдет с ошибкой и Декодер вернет "0" в качестве трека.

В случае если трек приходит в 16-теричной системе (например у карт Mifare) в маске необходимо использовать уже символы h чтобы Декодер воспринимал трек как 16-ричный.

```
mask=****hhhhhhhh
```

После применения маски получается трек который всегда будет в 10-тичной системе независимо от исходной системы и он уже вернется в TcpCardReader. Далее TCPCardReader обрамит этот трек префиксом "Prefix=;" и суффиксом "Postfix=?" и отправит клиентам.

Готовый пример Dodecoder.ini для карт E-marine и Mifare:

```
[frf]
fsUnit=fsdecodecard.upas

[general]
usemask=1
TrackResultLog=

[mask]
  card_mi = fare*
  card_em = ne*

[cards.card_mi]
ExcludedPrefix=fare[
code=2
CardDecodeType = MASK
mask=hhhhhhhh
DecodeTypeEx = 0

[cards.card_em]
ExcludedPrefix=ne[
code=2
CardDecodeType = MASK
mask=****dddddddd
```

```
DecodeTypeEx = 0
```

Принцип декодирования с использованием FastScript

Также как в любом другом случае декодирование на стороне TCPCardReader'а включается строкой.

```
useDecodeCards=1
```

в TCPCardReader.ini в настройках конкретного считывателя:

```
[READER1]
COMPORT=6
COMPORTPARAMS=baud=9600 data=8 parity=N stop=1
Prefix=;
Postfix=?
useDecodeCards=1
StartTermChars=i
FinishTermChars=?#13;#10;#0;
```

Таким образом при включении декодирования TCPCardReader отправляет часть трека начинающегося с символа следующего за *StartTermChars* и заканчивающимся символом стоящим перед одним из *FinishTermChars в* Decodecards*.*

Итак Decodecards получает только эту часть и декодирует (преобразовывает/выделяет идентификатор) её.

В **decodecards.ini** В секции **[mask]** перечисляются маски по префиксам которых можно разделять декодирование разных карт

[mask]
card_em = ne*
card_mifare = fare*
card = *

В данном примере если пришедший от TCPCardReader'а трек содержит префикс "ne" ,то обработка такого трека будет описана в секции [cards.**card_em**]. Соответственно если трек содержит префикс "fare" то обработка такого трека описана в секции [cards.card_mifare].

Символ "*" в данном случае означает любое количество символов.

> `Чтение данной секции происходит последовательно, то есть при первом совпадении пришедшего трека с маской дальнейшей просмотр секции не ведётся`

Далее в секциях с названием "cards." и левой частью до знака равенства в секции [mask] конкретного типа трека настраивается декодирование.

```
[cards.Card]
code=2
CardDecodeType = FastScript
DecodeTypeEx = 0
```

Опцией CardDecodeType = FastScript выбираем алгоритм декодирования с использованием FastScript.

В таком случаем алгоритм декодирования программируется в функции decodecard во внешнем файле, задаваемым параметром fsUnit в секции [frf].

```
[frf]
fsUnit=fsdecodecard.upas
```

Пример внешнего файла fsdecodecard.upas

```
function fsUnitVersion:integer;
begin
  result := 1;
end;

function DecodeCard(Track:string; var Code:integer):string;
begin
  if ( Length(track) >= 4 ) then
    result :1= copy( track, length(track)-3, 4 )
  else
    result := track;
end;

begin
```

```
end.
```

Т.е. при считывании идентификатора, для его декодирования будет вызвана функция DecodeCard() из данного файла.

Входные параметры функции:
Track - полный трек полученный от считывателя (типа string),
Code - код типа идентификатора (типа integer, подлежит изменению в теле функции);
Выходные параметры функции:
Result - номер идентификатора представленный строкой, после получения должен безошибочно конвертироваться в Int64 (типа string). Т.е результатом работы функции DecodeCard должна быть строка которая должна однозначно приводиться к 10-тиричному числу.

В файле скрипта доступны все функции доступные в печатных формах абонемента, что позволяет облегчить написание и отладку скрипта.
Пример печатной формы для тестирования разрабатываемого скрипта в приложении.

Вся обработка происходит в функции DecodeCard в которую передаются трек без *ExcludedPrefix* и *code* указанный в соответствующей секции файла *decodecards.ini.* Чаще всего это *code=2* как в данном примере. В функции можно использовать другие функции реализованные в этом же файле.

Далее TCPCardReader обрамит этот трек префиксом "Prefix=;" и суффиксом "Postfix=?" и отправит клиентам.

Третий закон Чизхолма[1].

Даже если вы считаете, что сделали прекрасную вещь, за которую вам все будут благодарны, всё равно найдётся человек, которому это не понравится, и он даже может рассердиться на вас

Алгоритмы декодирования

Алгоритм декодирования устанавливается параметром *CardDecodeType (алгоритм извлечения истинного номера карты)* в каждой секции обработки типа карты файла **decodecards.ini**.

```
[cards.card_mi]
ExcludedPrefix=fare[
code=2
CardDecodeType = MASK
mask=hhhhhhhh
```

Возможны следующие варианты:

```
CardDecodeType = LAST8
```

В этом случае в качестве номера карты берутся последние 8 цифровых символов трека.

```
CardDecodeType = LAST9
```

В этом случае в качестве номера карты берутся последние 9 цифровых символов трека.

```
CardDecodeType = MASK
```

MASK - накладывается маска. Подобно описана в соответствующем разделе. Наиболее часто используемый вариант.

```
CardDecodeType = GAMEKEEPERCARD
```

В этом случае в качестве номера карты берутся цифровые символы между вторым и третьим знаками "=" трека.

```
CardDecodeType = REPLACECARDNO
```

Для этого алгоритма используется дополнительный параметр NewCardNo
NewCardNo - это числовая константа (integer), которая подставляется как результат декодирования.

```
CardDecodeType = MASK2
```

Это алгоритм похож на MASK, за исключением того,
что алгоритм MASK перед накладыванием маски на трек УДАЛЯЕТ дополнительно все символы, кроме 0-9 и A-F.
Все параметры алгоритма MASK2 совпадают с соответствующими параметрами алгоритма MASK

```
CardDecodeType = FastScript
```

Алгоритм декодирования программируется в функции decodecard во внешнем файле.
Подобно описана в соответствующем разделе.

Принципиальная схема работы со считывателем в AbonementManager через TCPCardReader

under construction

// Схема

//Описание логики

Подключение сканера штрих-кодов

Данный увеличения скорости обслуживания клиента при реализации товаров и услуг, уместно произвести дополнительную настройку для сканера штрих-кодов.

Если HyperTerminal выдает последовательность `9785318001000#

 9785318001000 - реальный номер

 ` и # Старт и стоп символы выдаваемые считывателем

Настройка TcpCardReader.ini

[READER1] Устройство считывания штрих кодов
COMPORT=1 Порт, к которому подключен
COMPORTPARAMS=baud=9600 data=8 parity=N stop=1 Характеристики устройства.
Prefix=;BarCode Префикс идентификатор.
Postfix=? Постфикс
useDecodeCards=0 DecodeCards не используеться

Настройка abonementmanager.ini

[TCPCardReader]
StartTermChars=;
FinishTermChars=#13;#10;#35; #35 Это символ #
BarCodePrefix=BarCode Префикс штрих кодов (Prefix в TcpCardReader.ini). Только буквы латинского алфавита и цифры
PrefixMaskForProhibitDecode = BarCode* Маска штрих кодов

Использование для идентификации клиента:

Работа со сканером(ами) аналогично подлючению считывателя.

under construction

//Использование для поиска услуг

//Использование по поиску абонемента

Конфигурационные параметры