Advanced Card Systems Ltd.
Card & Reader Technologies

ACM1252U-Z2

小型 NFC リーダーモジュール

リファレンスマニュアル V1.06

info@acs.com.hk
www.acs.com.hk

改定履歴

リリス日付	改訂説明	バージョン
2017-02-08	• 初版作成	1.00
2018-06-19	• マーケティング名の更新 • セクション 2.0 更新：特性	1.01
2018-09-11	• 製品写真を更新する	1.02
2019-10-14	• MIFARE の商標および帰属の説明を追加。 • セクション 5.0 を再編成 • セクション 5.1 の追加例 PCSC API • セクション 5.4.2.1 更新：認証キーのダンロード（Load Authentication Keys） • セクション 5.4.2.7 更新：数値ブロックをコピーする（Copy Value Block） • セクション 5.3.6 更新：自動的な PICC のポーリングを設置する（Set Automatic PICC Polling） • セクション 5.3.7 更新：自動的な PICC のポーリングを読取る（Read Automatic PICC Polling） • セクション 5.5.4 更新：カードのエミュレーション時の MIFARE Ultralight の UID を設定する（Set Card Emulation of MIFARE Ultralight UID） • セクション 5.5.5 更新：カードエミュレーションの FeliCa カードの IDM を設定する（Set Card Emulation FeliCa IDm） • セクション 5.5.6 更新：NFC カードエミュレーションのデータロックを設定する（Set Card Emulation Lock Data in NFC） • エスケープコマンドの例について付録 B を追加	1.03
2019-12-04	• セクション 5.1 更新：PCSC API	1.04

リリス日付	改訂説明	バージョン
2020-01-31	• セクション 5.3.7 更新：自動的な PICC のポーリングを読取る（Read Automatic PICC Polling） • セクション 5.3.10 更新：自動的な PPS を設定する（Set Auto PPS） • セクション 5.3.11 更新：自動的な PPS を読み取る（Read Auto PPS）	1.05
2020-08-27	• セクション 5.2.2.2 の追加：PICC データ取得（Get PICC Data） • セクション 5.3.6 更新：自動的な PICC のポーリングを設置する（Set Automatic PICC Polling） • セクション 5.3.7 更新：自動的な PICC のポーリングを読取る（Read Automatic PICC Polling） • セクション 5.3.10 の追加：PICC 操作のパラメーターを設定する（拡張）（Set PICC Operating Parameter） • セクション 5.3.11 の追加：PICC 操作のパラメーターを読取る（拡張）（Read PICC Operating Parameter） • セクション 5.3.15 の追加：PICC タイプ読み取り（Read PICC Type）	1.06

カタログ

図示カタログ

チャートカタログ

1.0. 紹介

ACM1252U-Z2 NFC リーダーモジュール（USB インターフェース U）は、13.56 MHz 非接触（RFID）技術に基づいて開発された PC リンクの非接触スマートカードリーダー/ライターです。MIFARE®や ISO 14443 4 パートの A および B カードと FeliCa、4 タイプすべての NFC タグもサポートしています。

ACM1252U-Z2 は ACR1252U-M1 モジュール製品です。3 種の NFC モードをサポートしています：NFC カードリーダー、カードエミュレーションおよび P 2 P 通信。

このリファレンスマニュアルは PC/SC の APDU コマンドを実行することによって、どのように非接触インターフェースと ACM1252U-Z2 の周辺機器をサポートすることを詳しく説明します。

2.0. 特性

- USB フルスピード·インターフェース
- CCID準拠
- スマートカードリーダー：
  - 非接触インターフェース：
    - 最大 424 Kbps の書き込み速度
    - 内蔵アンテナを使って、動作可能距離は最大で 30 mm です（使用する非接触タグのタイプに依存します）。
    - ISO 144434 パート A、B タイプのカードサポートしています。MIFARE Classic®、MIFARE® DESFire®、MIFARE Ultralight®、MIFARE Plus®、FeliCa カード、Topaz カードおよび 4 タイプすべての NFC タグ（ISO/IEC 18092）もサポートしています。
    - 衝突防止機能内蔵（一つのタグはいつでもアクセス可能）
    - NFC サポート：
      - カードリーダ/ライタモード
      - ピアツーピア通信モード
      - カードエミュレーションモード
- 内蔵されている周辺機器：
  - ユーザーコントロールできる二色 LED パイロットランプ
- アプリケーション プログラミング インターフェース：
  - PC/SC サポート
  - (PC / SC の上のラッパー経由で） , CT- API をサポート
- ファームウェアのアップグレード機能
- Android™ 3.1と以降のバージョンサポートしている[1]
- 以下の規格に準拠：
  - ISO 14443
  - ISO 18092
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS
  - REACH
  - Microsoft® WHQL

[1] *ACS の Android ライブラリを使用*

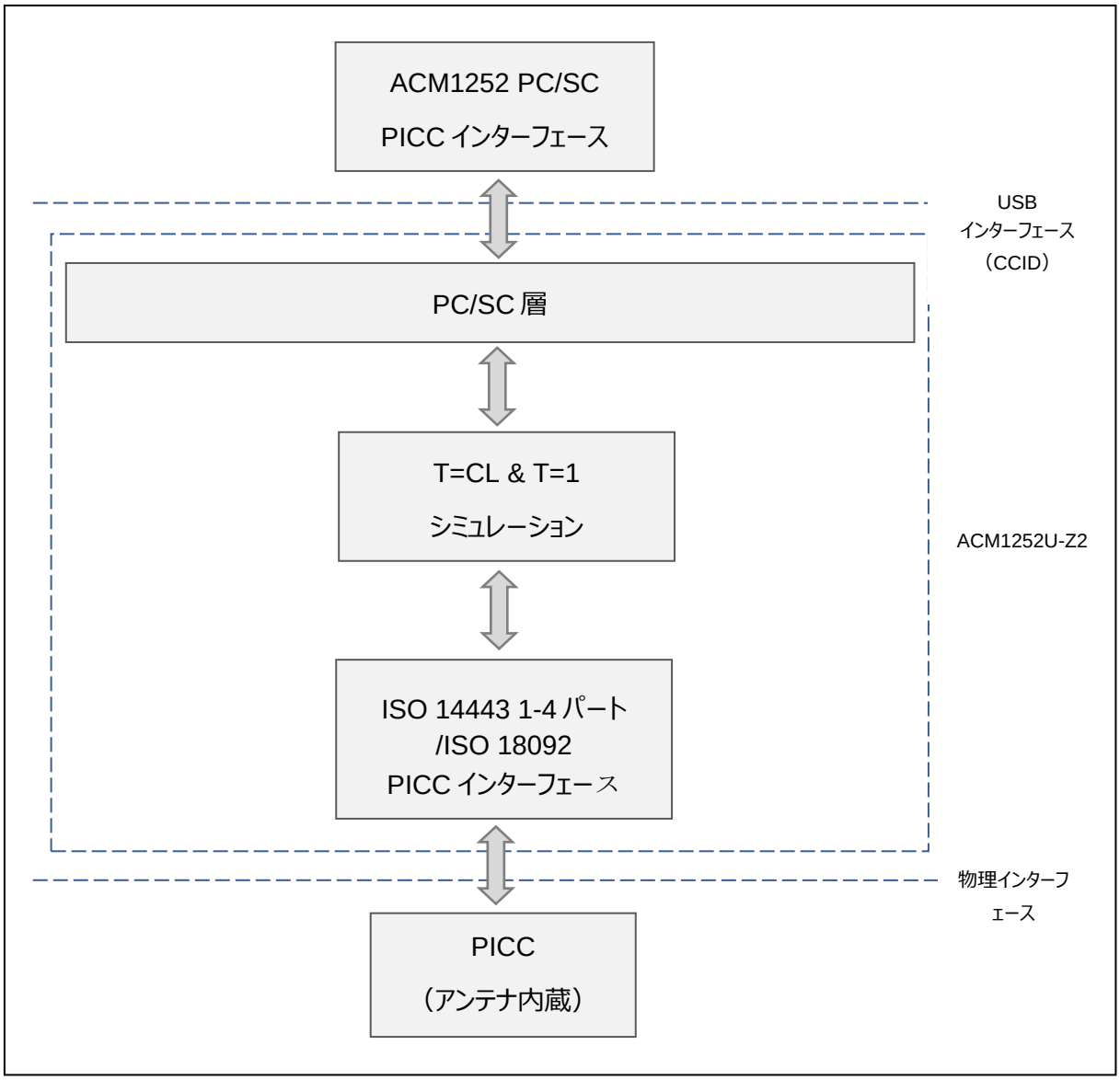
3.0. 略語

略語	説明
ATR	属性リクエストと属性応答
DEP	データ交換プロトコルのリクエストと応答
DSL	パラメーターオプションのリクエストと応答をキャンセル
PSL	パラメーターオプションのリクエストと応答
RLS	リクエストと応答をリリース
WUP	ウェイクアップリクエストとウェイクアップ応答
DID	設備 ID
BS	ビット期間を送信します
BR	ビット期間を受信します
PP	プロトコルパラメーター
Gi	イニシエータのオプションの情報フィールド
PFB	取引のための制御情報
FSL	フレーム長さの最大値
LLCP	論理リンク制御プロトコル

表1 ：略語

4.0. アーキテクチャ

ACM1252U-Z2 と PC のデータ通信は CCID プロトコルを採用しています。PICC 間の通信は PC/SC 規格に準拠しています。



図示 1：ACM1252U-Z2 アーキテクチャ

5.0. ホストプログラミング（PC リンク）API

5.1. PCSC API

このセッションでは、いくつのアプリケーションプログラミングに使用する PC/SC API コマンドを説明します。これらの API の詳しい情報について、Microsoft MSDN ライブラリまたは PC/SC ワークグループを参照してください。

5.1.1. SCardEstablishContext

SCardEstablishContext 関数はデータベース操作を実行するリソースマネージャのコンテキストを確立するのためです。

ほかの PCSC 実行する前に、この関数を実行するはずです。.

参照のウェブサイト：

https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scardestablishcontext

例：

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext;
int retCode;
void main ()
{
   // To establish the resource manager context and assign it to “hContext”
   retCode = SCardEstablishContext(SCARD_SCOPE_USER,
                  NULL,
                  NULL,
                  &hContext);
   if (retCode != SCARD_S_SUCCESS)
   {
      // Establishing resource manager context failed
   }
   else
   {
      // Establishing resource manager context successful
      // Further PCSC operation can be performed
   }
}
```

5.1.2. SCardListReaders

SCardListReaders 関数は、重複をなくして、一つのセットの名前付きリーダーグループリストを提供します。

呼び出し側はリーダーグループのリストを供給します。関数は指定しているセット中の名前付きリーダーのリストを返します。認識できないグループの名前は無視されます。この関数は現在システムに接続されて利用できるグループ中のリーダーだけに返されます。

参照のウェブサイト：

https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scardlistreadersa

例：

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT hContext; // Resource manager context
int retCode;
char readerName [256]; // List reader name

void main ()
{
   // To establish the resource manager context and assign to
"hContext"
   retCode = SCardEstablishContext(SCARD_SCOPE_USER,
                  NULL,
                  NULL,
                  &hContext);
   if (retCode != SCARD_S_SUCCESS)
   {
      // Establishing resource manager context failed
   }
   else
   {
      // Establishing resource manager context successful
      // List the available reader which can be used in the system
      retCode = SCardListReaders (hContext,
                  NULL,
                  readerName,
                  &size);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Listing reader fail
      }
      if (readerName == NULL)
      {
         // No reader available
      }
      else
      {
         // Reader listed
      }
   }
}
```

5.1.3. SCardConnect

SCardConnect 関数は（特別のリソースマネージャのコンテキスを利用して）アプリケーションと特定のリーダーを含めているスマートカードの間に接続を確立します。特定のリーダー中はカードがない場合、エラーメッセージが返されます。

参照のウェブサイト：

https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scardconnecta

例：

```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT       hContext;          // Resource manager context
SCARDHANDLE        hCard;             // Card context handle
unsigned long      dwActProtocol;     // Establish active protocol
int                retCode;
char               readerName [256]; // List reader name
char               rName [256];       // Reader name for connection

void main ()
{
   …
      if (readerName == NULL)
      {
         // No reader available
      }
      else
      {
         // Reader listed
      rName = "ACS ACR1252 CL Reader PICC 0";    // Depends on what
                                                 reader be used
                                                 // Should connect to
                                                 PICC interface

         retCode = SCardConnect(hContext,
               rName,
               SCARD_SHARE_SHARED,
               SCARD_PROTOCOL_T0,
               &hCard,
               &dwActProtocol);
         if (retCode != SCARD_S_SUCCESS)
         {
            // Connection failed (May be because of incorrect reader
         name, or no card was detected)
         }
         else
         {
            // Connection successful
         }
      }
   }
```

5.1.4. SCardControl

SCardControl 関数はユーザーにカードリーダーをダイレクトに制御する機能を提供しています。**SCardConnect** 関数が成功に呼び出されして、**SCardDisconnect** 関数を呼び出す前に、ユーザーはこの関数を自由に呼び出すことができます。リーダーの状態に対する影響は、制御コードに依存しています。

参照のウェブサイト：

https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scardcontrol

注釈：*5.3* セクションのコマンドはこの *API* で送信します。

例:

```
#define SCARD_SCOPE_USER    0

#define EscapeCommand 0x310000 + 3500*4
   SCARDCONTEXT          hContext;          // Resource manager context
   SCARDHANDLE           hCard;             // Card context handle
   unsigned long         dwActProtocol;     // Established active protocol
   int                   retCode;
   char                  readerName [256]; // Lists reader name
   char                  rName [256];       // Reader name for connection
   BYTE                  SendBuff[262],     // APDU command buffer
                         RecvBuff[262];     // APDU response buffer
   BYTE                  FWVersion [20],    // For storing firmware
                                               version message
   BYTE                  ResponseData[50]; // For storing card response
   DWORD                 SendLen,           // APDU command length
                         RecvLen;           // APDU response length
void main ()
   {
      …
      rName = "ACS ACR1252 CL Reader PICC 0";       // Depends on what
                                                    reader will be used
                                                    // Should connect to
                                                    PICC interface

         retCode = SCardConnect(hContext,
            rName,
            SCARD_SHARE_DIRECT,
            SCARD_PROTOCOL_T0| SCARD_PROTOCOL_T1,
            &hCard,
            &dwActProtocol);
         if (retCode != SCARD_S_SUCCESS)
         {
            // Connection failed (may be because of incorrect reader
         name, or no card was detected)
         }
         else
         {
            // Connection successful
            RecvLen = 262;
            // Get firmware version
            SendBuff[0] = 0xE0;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x18;
            SendBuff[4] = 0x00;
```

```
            SendLen = 5;
            retCode = SCardControl ( hCard,
                      EscapeCommand,
                      SendBuff,
                      SendLen,
                      RecvBuff,
                      RecvLen,
                      &RecvLen);
            if (retCode != SCARD_S_SUCCESS)
            {
               // APDU sending failed
               return;
            }
            else
            {
               // APDU sending successful
               // The RecvBuff stores the firmware version message.
               for (int i=0;i< RecvLen-5;i++)
               {
                  FWVersion[i] = RecvBuff [5+i];
               }
            }
            // Connection successful
            RecvLen = 262;

            // Turn Green LED on, turn Red LED off
            SendBuff[0] = 0xE0;
            SendBuff[1] = 0x00;
            SendBuff[2] = 0x00;
            SendBuff[3] = 0x29;
            SendBuff[4] = 0x01;
            SendBuff[5] = 0x02;  // Green LED On, Red LED off
            SendLen = 6;
            retCode = SCardControl ( hCard,
                      EscapeCommand,
                      SendBuff,
                      SendLen,
                      RecvBuff,
                      RecvLen,
                      &RecvLen);
            if (retCode != SCARD_S_SUCCESS)
            {
               // APDU sending failed
               return;
            }
            else
            {
               // APDU sending success
            }
```

5.1.5. SCardTransmit

SCardTransmit 関数はサービスリクエストをスマートカードに送信するために、またはスマートカードから返されるデータを受信するために使われます。

参照のウェブサイト：

https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scardtransmit

注：*APDU* コマンド（即ち：接続を確立されたカードに送信するコマンド、***5.2.4*** セクション - *PICC* コマンドそして ***5.2.2*** セクション - 非接触インターフェースの疑似 *APDU* コマンド）はこの *API* で送信されます。

例:

```
#define SCARD_SCOPE_USER      0

SCARDCONTEXT      hContext;          // Resource manager context
SCARDHANDLE       hCard;             // Card context handle
unsigned long     dwActProtocol;     // Established active protocol
int               retCode;
char              readerName [256]; // List reader name
char              rName [256];       // Reader name for connect
BYTE              SendBuff[262],     // APDU command buffer
                  RecvBuff[262];     // APDU response buffer
BYTE              CardID [8],        // For storing the FeliCa IDM/
                                     MIFARE UID
BYTE              ResponseData[50]; // For storing card response
DWORD             SendLen,           // APDU command length
                  RecvLen;           // APDU response length
SCARD_IO_REQUEST  ioRequest;

void main ()
{
   …
   rName = "ACS ACR1252 CL Reader PICC 0";    // Depends on what reader
                                              should be used
                                              // Should connect to PICC
                                              interface
      retCode = SCardConnect(hContext,
                     rName,
                     SCARD_SHARE_SHARED,
                     SCARD_PROTOCOL_T0,
                     &hCard,
                     &dwActProtocol);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Connection failed (May be because of incorrect reader
         name, or no card was detected)
      }
      else
      {
         // Connection successful
         ioRequest.dwProtocol = dwActProtocol;
         ioRequest.cbPciLength = sizeof(SCARD_IO_REQUEST);
         RecvLen = 262;
```

```
// Get MIFARE UID/ FeliCa IDM
SendBuff[0] = 0xFF;
SendBuff[1] = 0xCA;
SendBuff[2] = 0x00;
SendBuff[3] = 0x00;
SendBuff[4] = 0x00;
SendLen = 5;
retCode = SCardTransmit( hCard,
                  &ioRequest,
                  SendBuff,
                  SendLen,
                  NULL,
                  RecvBuff,
                  &RecvLen);

if (retCode != SCARD_S_SUCCESS)
{
   // APDU sending failed
   return;
}
else
{
   // APDU sending successful
   // The RecvBuff stores the IDM for FeliCa / the UID for
   MIFARE.
   // Copy the content for further FeliCa access
   for (int i=0;i< RecvLen-2;i++)
   {
      CardID [i] = RecvBuff[i];
   }
}
```

5.1.6. SCardDisconnect

SCardDisconnect 関数は前に確立されたアプリケーションとターゲットリーダー間の接続を終了するためです。。

参照のウェブサイト：

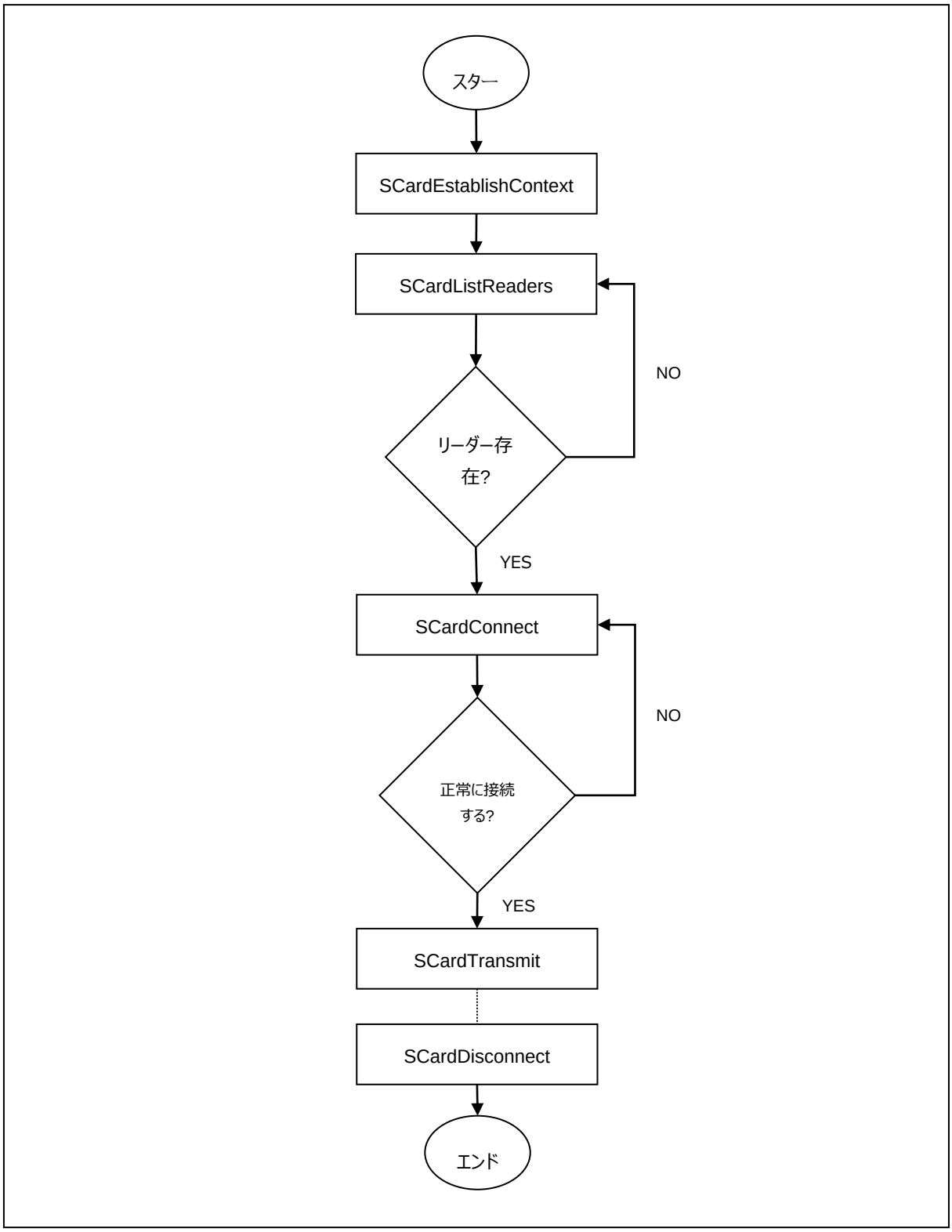
https://docs.microsoft.com/en-us/windows/win32/api/winscard/nf-winscard-scarddisconnect

この関数 PCSC 操作を終止します。.

例：

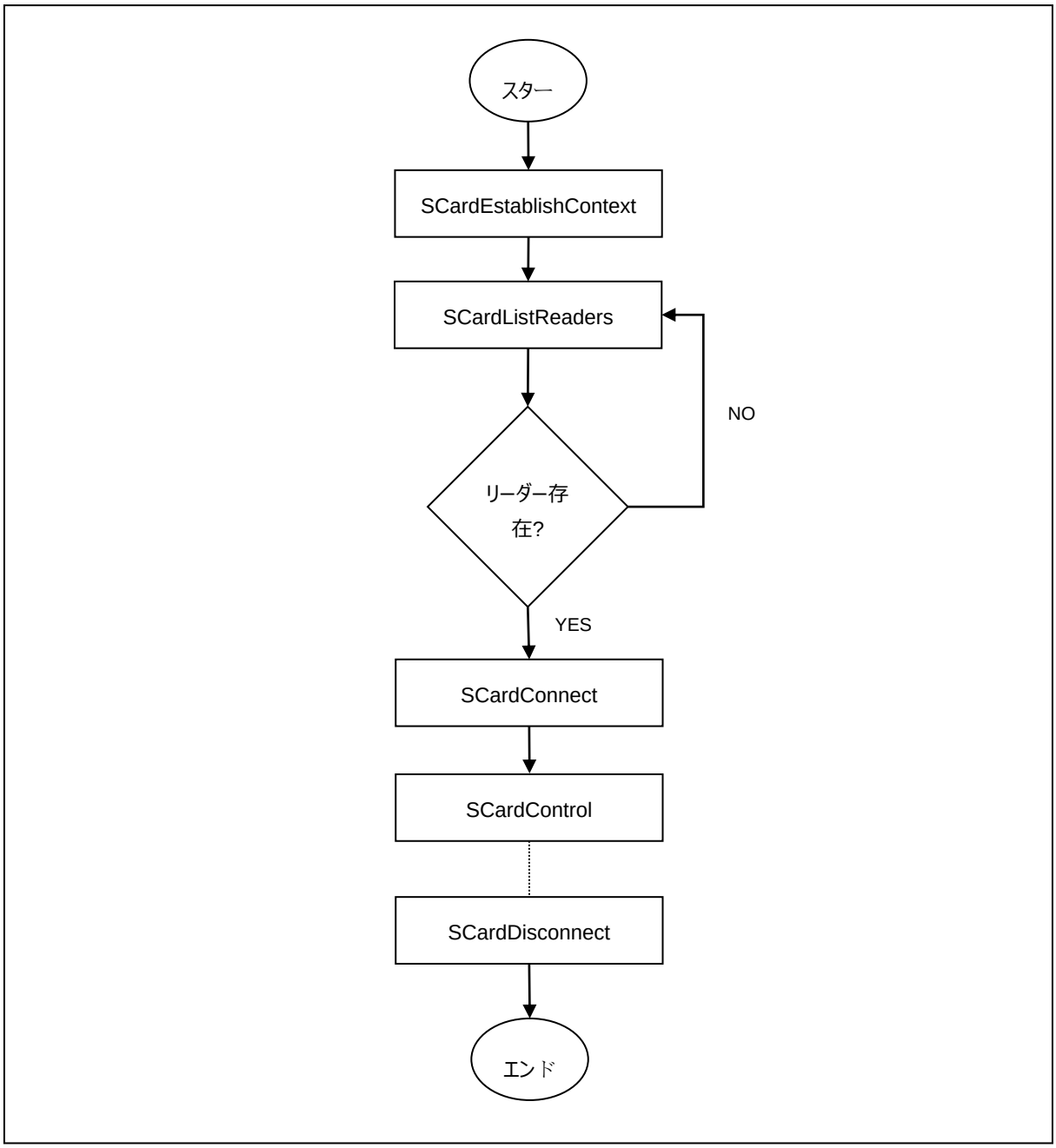
```
#define SCARD_SCOPE_USER 0

SCARDCONTEXT       hContext;          // Resource manager context
SCARDHANDLE        hCard;             // Card context handle
unsigned long      dwActProtocol;     // Established active protocol
int                retCode;

void main ()
{
   …
         // Connection successful
      …
      retCode = SCardDisconnect(hCard,SCARD_RESET_CARD);
      if (retCode != SCARD_S_SUCCESS)
      {
         // Disconnection failed
      }
      else
      {
         // Disconnection successful
      }
   }
}
```

5.1.7. APDU の流れ



図示 2:ACM1252U-Z2 の APDU の流れ

5.1.8. ダイレクトコマンド（Escape Command）の流れ



図示 3：ACM1252U-Z2 直接なコマンドの流れ

5.2. 非接触スマートカード プロトコル

5.2.1. ATR の生成

リーダーが PICC を検出すると、PICC を識別するために、ATR が PC/SC ドライバに送信されます。

5.2.1.1. ATR フォーマット（ISO 14443-3 PICC に適用）

バイト	数値	標記	説明
0	3Bh	最初のヘッダー	-
1	8Nh	T0	上位ニブル 8 の意味は：TA1、TB1 と TC1 がなくて、TD1 だけが続いている。 下位ニブル N はヒストリカルバイトの数です（HistByte 0 - HistByte N-1)
2	80h	TD1	上位ニブル 8 の意味は：TA2、TB2 と TC2 がなくて、TD2 だけが続いている。 下位ニブル 0 の意味は T=0
3	01h	TD2	上位ニブル 0 の意味は：TA3、TB3、TC3 および TD3 が全部続いていない。 下位ニブル 1 の意味は T=1
4 から 3+N	80h	T1	カテゴリインジケータバイトは、80 のステータスンジケータが任意の COMPACT-TLV データオブジェクトに存在するかもしれない意味です
	4Fh	Tk	アプリケーション識別子にはインジケータが存在している
	0Ch		長さ
	RID		登録されたアプリケーションプロバイダ識別子 （RID）# A0 00 00 03 06
	SS		基準のバイト
	C0 ..C1h		カードネームバイト
	00 00 00 00h	RFU	RFU # 00 00 00 00
4+N	UU	TCK	T0 から Tk までのすべてのバイトの排他的論理和

例：

MIFARE Classic 1K カード ATR = {3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6Ah}

その中：

長さ （YY） = 0Ch

RID = {A0 00 00 03 06h} (PC/SC ワークグループ)

標準 (SS) **=** 03h (ISO 14443A、パート 3)

カードネーム （C0 ..C1） ={00 01h} (MIFARE Classic 1K)

基準 （SS） = 03h：ISO 14443A、3 パート

= 11h：FeliCa

カードネーム (C0 ..C1)

00 01：MIFARE Classic 1K	00 38：MIFARE Plus SL2 2K
00 02：MIFARE Classic 4K	00 39：MIFARE Plus SL2 4K
00 03：MIFARE Ultralight	00 30：Topaz 和 Jewel
00 26：MIFARE Mini	00 3B：FeliCa
00 3A：MIFARE Ultralight C	FF 28：JCOP 30
00 36：MIFARE Plus SL1 2K	FF [SAK]：定義されていないタグ
00 37：MIFARE Plus SL1 4K	

5.2.1.2. ATR フォーマット（ISO 14443-4 PICC に適用）

<table>
<tr><th>バイト</th><th>数値</th><th>標記</th><th>説明</th></tr>
<tr><td>0</td><td>3Bh</td><td>最初のヘッダー</td><td>-</td></tr>
<tr><td>1</td><td>8Nh</td><td>T0</td><td>上位ニブル 8 の意味は：TA1、TB1 と TC1 がなくて、TD1 だけが続いている。
下位ニブル N はヒストリカルバイトの数です（HistByte 0 - HistByte N-1)</td></tr>
<tr><td>2</td><td>80h</td><td>TD1</td><td>上位ニブル 8 の意味は：TA2、TB2 と TC2 がなくて、TD2 だけが続いている。
下位ニブル 0 の意味は T=0</td></tr>
<tr><td>3</td><td>01h</td><td>TD2</td><td>上位ニブル 0 の意味は：TA3、TB3、TC3 および TD3 が全部続いていない。
下位ニブル 1 の意味は T=1</td></tr>
<tr><td rowspan="2">4
から
3 + N</td><td>XX</td><td>T1</td><td rowspan="2">ヒストリカルバイト

ISO 14443-A：
ATS 応答のヒストリカルバイト。ISO 14443-4 基準を参照してください。

ISO 14443-B：
<table>
<tr><th>バイト 1-4</th><th>バイト 5-7</th><th>バイト 8</th></tr>
<tr><td>ATQB のアプリケーションデータ</td><td>ATQB からのプロトコル情報バイト</td><td>上位ニブル =ATTRIB コマンドの MBLI；
下位ニブル（RFU）=0</td></tr>
</table></td></tr>
<tr><td>XX
XX
XX</td><td>Tk</td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>T0 から Tk までのすべてのバイトの排他的論理和</td></tr>
</table>

例 1：

MIFARE DESFire の ATR = {3B 81 80 01 80 80h} // 6 バイトの ATR

注：*APDU"FF CA 01 00 00h"*を使用して、*ISO 14443A-4* の *PICC* に準拠しているまたは *ISO 14443B-4* の *PICC* に準拠しているを区別します。可能な場合、完全な *ATS* を取得します。ISO 14443A-3 または ISO 14443B-3/4 の PICC に準拠する場合、ATS が返される。APDU コマンド = FF CA 01 00 00h

APDU 応答 = 06 75 77 81 02 80 90 00h

ATS = {06 75 77 81 02 80h}

例 2：

EZ-Link カードの ATR = {3B 88 80 01 1C 2D 94 11 F7 71 85 00 BEh}

ATQB の応答データ = 1C 2D 94 11h

ATQB からのプロトコル情報 = F7 71 85h

ATTRIB の MBLI =00h

5.2.2. 非接触インターフェースの疑似 APDU コマンド

5.2.2.1. データを取得する（Get Data）

GET DATA コマンドは“接続された PICC”のシリアルナンバーもしくは ATS を取得します。

GET UIDAPDU フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Get Data	FFh	CAh	00h 01h	00h	00h （最大長さ）

P1 = 00h の場合、Get UID の応答フォーマット（UID + 2 バイト）

応答	データ出力					
結果	UID (LSB)	…	…	UID (MSB)	SW1	SW2

例え **P1 = 01h**、ISO14443 A タイプのカードの ATS を入手する（ATS + 2 バイト）

応答	データ出力		
結果	ATS	SW1	SW2

応答コード

結果	SW1	SW2	意味
成功	90h	00h	操作が成功に完了しました。
警告	62h	82h	UID/ATS の終わりが Le バイトの前に達しました（Le は UID の長さより大きいです）
エラー	6Ch	**XX**h	間違った長さ（間違ったナンバーLe：‘XX’は精確な数字を表す）、Le は、利用可能な UID の長さ未満である場合
エラー	63h	00h	操作が失敗しました。
エラー	6Ah	81h	この機能をサポートできません。

例：

“接続された PICC”のシリアルナンバーを取得します

```
UINT8 GET_UID[5] = {FF, CA, 00, 00, 00};
```

“接続された ISO 14443-A PICC”の ATS を取得します

```
UINT8 GET_ATS[5] = {FF, CA, 01, 00, 00};
```

5.2.2.2. PICC データ取得（Get PICC Data）

GET DATA コマンドは“接続された PICC”のシリアルナンバーを取得することに使われます。

注： 208.0 以降のバジョンのみに適用します。

Get PICC Data APDU フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Get PICC Data	FFh	CAh	00h	02h	00h

A タイプのカートの場合、ATQA + UID + SAK 応答フォーマットを取得（2 バイト + 4/7/10 バイト + 1 バイト + 2 バイト）

応答	データ出力								
結果	ATQA	ATQA	UID (LSB)	…	…	UID (MSB)	SAK	SW1	SW2

B タイプのカードの場合、ATQB を取得（12 バイト+2 バイト）

応答	データ出力		
結果	ATQB	SW1	SW2

応答コード

結果	SW1	SW2	意味
成功	90h	00h	操作が成功に完了しました。
エラー	63h	00h	操作が失敗しました。
エラー	6Ah	81h	この機能をサポートできません。

5.2.3. PCSC 2.0 パート 3 の APDU コマンド（2.02 もしくは最新バージョン）

これらのコマンドが透過的にアプリケーションからデータを非接触タグへ渡し、アプリケーションとプロトコルに透過的に受信したデータを返し、同時にプロトコルを切り替えるために使用されています。

5.2.3.1. コマンドと応答の APDU フォーマット

コマンドのフォーマット

CLA	INS	P1	P2	Lc	データイン
FFh	C2h	00h	機能	DataLen	Data[DataLen]

その中：

機能　1 バイト。

00h = セッション管理

01h = トランスペアレントエクスチェンジ

02h = プロトコルの切り替え

他 = RFU

応答フォーマット

データ出力	SW1	SW2
符号化されました BER-TLV データフィールド		

すべてのコマンドは、レスポンスデータフィールド（利用可能な場合）と一緒に SW1 と SW2 を返します。SW1 と SW2 は ISO7816 に基づいて、以下の C0 データオブジェクトの SW1 SW2 も使用する必要があります。

C0 データ要素のフォーマット

タグ	速度（1 バイト）	SW2
C0h	03h	エラーステータス

エラーステータスの説明

エラーステータス	説明
XX SW1 SW2	XX = APDU 内の不正なデータオブジェクトの数量 00 = APDU の一般的なエラー 01 = 一番目のデータオブジェクト内のエラー 02 = 二番目のデータオブジェクト内のエラー
00 90 00h	エラーは発生していません
XX 62 82h	データオブジェクトの XX 警告、要求された情報は存在していません
XX 63 00h	情報なし
XX 63 01h	実行は他のデータオブジェクトの障害のため停止しました
XX 6A 81h	データオブジェクトはサポートされていません XX
XX 67 00h	予期しない長さのデータオブジェクト XX
XX 6A 80h	予期しない値のデータオブジェクト XX
XX 64 00h	データオブジェクト XX の実行エラー（IFD からの応答がありません）
XX 64 01h	データオブジェクト XX の実行エラー（ICC からの応答がありません）
XX 6F 00h	データオブジェクト XX は正確な診断なしで失敗しました

最後の 2 バイトは、エラーの説明を示しながら、一番目のバイトの数値は、誤ったデータオブジェクトの XX の数を示します。ISO7816 に基づいて、SW1 SW2 の値が許可されています。

C-APDU データフィールドには複数のデータオブジェクトがあって、1 つのデータオブジェクトが失敗した場合、他のデータオブジェクトが失敗したデータオブジェクトに依存しない場合、IFD は次のデータオブジェクトを処理することができます。

5.2.3.2. セッションを管理するコマンド（Manage Session Command）

このコマンドは、トランスペアレントなセッションを管理するために使用されます。起動と透明セッションの終了が含まれています。このコマンドを使用して、ユーザーは動作環境やトランスペアレントセッション内の IFD の機能を管理することができます。

セッションを管理するコマンド

コマンド	CLA	INS	P1	P2	Lc	データイン
Manage Session	FFh	C2h	00h	00h	DataLen	データオブジェクト (N バイト)

その中：

データオブジェクト（1 バイト)

タグ	データオブジェクト
80h	バージョンのデータオブジェクト
81h	トランスペアレントセッションを開始する
82h	トランスペアレントセッションを終了する
83h	RF フィールドをオフにする
84h	RF フィールドをオンにする
5F 46h	タイマー
FF 6Dh	パラメーターを取得する
FF 6Eh	パラメーターを設定する

セッション管理の応答データオブジェクト

タグ	データオブジェクト
C0h	一般的なエラーステータス
80h	バージョンのデータオブジェクト
FF 6Dh	IFD パラメーターデータオブジェクト

5.2.3.2.1. セッションデータオブジェクトを開始する（Start Session Data Object）

このコマンドは、透過的なセッションを開始するために使用されています。セッションが開始されると、セッションが終了されるまで、自動ポーリングが無効になります。

セッションデータオブジェクトを開始する

タグ	速度（1 バイト）	数値
81h	00h	-

5.2.3.2.2. セッションデータオブジェクトを終了する（End Session Data Object）

このコマンドは、透過的なセッションを終了するために使用されています。セッションが開始される前に自動ポーリング状態にリセットされます。

セッションデータオブジェクトを終了する

タグ	速度（1 バイト）	数値
82h	00h	-

5.2.3.2.3. バージョンのデータオブジェクト

このコマンドは、IFD Handler のバージョン番号を返すために使用されます。

バージョンのデータオブジェクト

タグ	速度（1 バイト）	数値		
80h	03h	メジャーのバージョン	マイナーのバージョン	内部のバージョン

5.2.3.2.4. RF データオブジェクトをオフにする（Turn Off the RF Data Object）

このコマンドはアンテナフィールドをオフにする時に使われます。

RF データオブジェクトをオフにする

タグ	速度（1 バイト）	数値
83h	00h	-

5.2.3.2.5. RF データオブジェクトをオンにする（Turn On the RF Data Object）

このコマンドはアンテナフィールドをオンにする時に使われます。

RF データオブジェクトをオンにする

タグ	速度（1 バイト）	数値
84h	00h	-

5.2.3.2.6. タイマーデータオブジェクト（Timer Data Object）

このコマンドは、1 µs の単位で 32 ビットのタイマーデータオブジェクトを作成するために使用されます。

例：RF をオフにするデータオブジェクトと RF をオンにするデータオブジェクト間には 5000µs のタイマデータオブジェクトがある場合、RF がオンになっている前に、りーだーは 5000µs 程度の RF フィールドをオフにします。

タイマーデータオブジェクト

タグ	速度（1 バイト）	数値
5F 46h	04h	タイマー（4 バイト）

5.2.3.2.7. パラメータデータオブジェクトを取得する（Get Parameter Data Object）

このコマンドは、IFD から異なるパラメータを取得するために使用されます。

パラメータデータオブジェクトを取得する

タグ	速度（1 バイト）	数値		
		タグ	長さ	数値
FF 6Dh	バール	TLV_Objects		

TLV_Objects

要求のパラメータ	タグ	長さ
IFD フレームサイズの整数（FSDI）	01h	00h
ICC フレームサイズの整数（FSCI）	02h	00h
フレーム待ち時間の整数（FWTI）	03h	00h
IFD でサポートされている最大な通信速度	04h	00h
ICC の通信速度	05h	00h

要求のパラメータ	タグ	長さ
指数変調	06h	00h
ISO/IEC14443 基準の PCB	07h	00h
ISO/IEC14443 基準の CID	08h	00h
ISO/IEC14443 基準の NAD	09h	00h
ISO/IEC14443 B タイプのパラメ—タ—1 - 4	0Ah	00h

5.2.3.2.8. パラメータデータオブジェクトを設定する（Set Parameter Data Object）

このコマンドは、IFD とは異なるパラメータを設定するために使用されます。

パラメータデータオブジェクトを設定する

タグ	速度（1 バイト）	数値		
		タグ	長さ	数値
FF 6Eh	バール	TLV_Objects		

TLV_Objects

要求のパラメータ	タグ	長さ
IFD フレームサイズの整数（FSDI）	01h	01h
ICC フレームサイズの整数（FSCI）	02h	01h
フレーム待ち時間の整数（FWTI）	03h	01h
IFD でサポートされている最大な通信速度	04h	01h
ICC の通信速度	05h	01h
指数変調	06h	01h
ISO/IEC14443 基準の PCB	07h	01h
ISO/IEC14443 基準の CID	08h	01h
ISO/IEC14443 基準の NAD	09h	01h
ISO/IEC14443 B タイプのパラメ—タ—1 - 4	0Ah	04h

5.2.3.3. トランスペアレントエクスチェンジコマンド（Transparent Exchange Command）

このコマンドは、送信および ICC から任意のビットまたはバイトを受信するために使用されます。

トランスペアレントエクスチェンジコマンド

コマンド	CLA	INS	P1	P2	Lc	データイン
TranspEx	FFh	C2h	00h	01h	DataLen	データオブジェクト (N バイト)

その中：

データオブジェクト（1 バイト)

タグ	データオブジェクト
90h	送受信フラグ
91h	伝送ビットフレーミング
92h	受信ビットフレーミング
93h	送信
94h	受信
95h	送受信ー送信と受信
FF 6Dh	パラメ―タ―を取得する
FF 6Eh	パラメ―タ―を設定する

トランスペアレントエクスチェンジセッションの応答データオブジェクト

タグ	データオブジェクト
C0h	一般的なエラーステータス
92h	受信したデータの最後のバイトでの有効なビットの数量
96h	応答ステータス
97h	ICC 応答
FF 6Dh	IFD パラメ―タ―データオブジェクト

5.2.3.3.1. 送受信のフラグデータオブジェクト（Transmission and Reception Flag Data Object）

このコマンドは、次の送信のためのフレーミングおよび RF パラメータを定義するために使用されます。

送受信のフラグデータオブジェクト

タグ	速度（1 バイト）	数値	
		ビット	説明
90h	02h	0	0 – 送信したデータに CRC を追加します 1 – 送信したデータに CRC を追加しません
		1	0 – 受信したデータから CRC を破棄します 1 – 受信したデータから CRC を破棄しません（CRC チェックなし）
		2	0 – 送信したデータにパリティを挿入します 1 – 送信したデータにパリティを挿入しません
		3	0 – 受信したデータのパリティを期待します 1 – パリティを期待していません（パリティチェックなし）
		4	0 – 送信したデータのプロトコルプロローグを追加したり、応答から捨てます 1 - 追加またはプロトコルのプロローグを破棄することはありません（ある場合）（例えば、PCB、CID、NAD）
		5-15	RFU

5.2.3.3.2. ビットフレーミングデータオブジェクトを送信する（Transmission Bit Framing Data Object）

このコマンドは、送受信されていないデータの最後のバイトの有効ビット数を定義するために使用されます。

ビットフレーミングデータオブジェクトを送信する

タグ	速度（1 バイト）	数値	
		ビット	説明
91h	01h	0-2	最後のバイトの有効ビット数（0 はすべてのビットが有効であることを意味します）
		3-7	RFU

伝送ビットフレーミングデータオブジェクトは、「送信」または「送受信」のみのデータオブジェクトと一緒でなければなりません。このデータオブジェクトが存在しない場合、それはすべてのビットが有効であることを意味します。

5.2.3.3.3. ビットフレーミングデータオブジェクトを受信する（Reception Bit Framing Data Object）

コマンド APDU の場合、このデータオブジェクトは、受信されたデータの最後のバイトの予期な有効ビット数を定義します。

コマンド APDU の場合、このデータオブジェクトは、受信されたデータの最後のバイトの予期な有効ビット数を通知します。

ビットフレーミングデータオブジェクトを受信する

タグ	速度（1 バイト）	数値	
		ビット	説明
92h	01h	0-2	最後のバイトの有効ビット数（0 はすべてのビットが有効であることを意味します）
		3-7	RFU

このデータオブジェクトが存在しない場合、それはすべてのビットが有効であることを意味します。

5.2.3.3.4. データオブジェクトを送信する（Transmit Data Object）

このコマンドは、IFD から ICC にデータを送信するために使用されます。送信が完了した後、ICC からの応答が予想されていません。

データオブジェクトを送信する

タグ	速度（1 バイト）	数値
93h	DataLen	データ（N バイト）

5.2.3.3.5. データオブジェクトを受信する（Receive Data Object）

このコマンドは、次のタイマーオブジェクトに与えられた時間内に受信モードに入るために、リーダーを強制する時に使用されます。

データオブジェクトを受信する

タグ	速度（1 バイト）	数値
94h	00h	-

5.2.3.3.6. データオブジェクトを送受信する（Transceive Data Object）

このコマンドは、ICC からのデータを送受信するために使用されます。送信が完了すると、リーダーは、タイマーデータオブジェクトに指定された時間まで待機します。

何のタイマデータオブジェクトは、データフィールドで定義されていない場合、リーダーは Set Parameter FWTI データオブジェクトに指定された期間を待っています。FWTI が設定されていない場合、リーダーは、約 302 µs を待ちます。

データオブジェクトを送受信する

タグ	速度（1 バイト）	数値
95h	DataLen	データ（N バイト）

5.2.3.3.7. ステータスデータオブジェクトを応答する（Response Status Data Object）

応答内では、このコマンドが受信されたデータの状態を通知するために使用されます。

ステータスデータオブジェクトを応答する

タグ	速度（1 バイト）	数値		
		バイト 0		バイト 1
		ビット	説明	
96h	02h	0	0 - CRC が OK、若しくはチェックしていません 1 - CRC チェックが失敗しました	衝突が検出された場合、これらのバイトは、衝突位置を教えてくれます。じゃないと“00h”を表示します。
		1	0 – 衝突なし 1 – 衝突が検出されました	
		2	0 – パリティエラーなし 1 – パリティエラーが検出されました	
		3	0 – フレームエラーなし 1 – フレームエラーが検出されました	
		4 - 7	RFU	

5.2.3.3.8. データフォーマットを応答する

応答内では、このコマンドが受信されたデータの状態を通知するために使用されます。

データフォーマットを応答する

タグ	速度（1 バイト）	数値
97h	DataLen	応答データ（N バイト）

5.2.3.4. プロトコルスイッチコマンド（Switch Protocol Command）

このコマンドは、プロトコルとトランスペアレントセッション内の標準の異なる層を指定するために使用されます。

Switch Protocol Command

コマンド	CLA	INS	P1	P2	Lc	データイン
SwProtocol	FFh	C2h	00h	02h	DataLen	データオブジェクト (N バイト)

その中：

データオブジェクト（1 バイト)

タグ	データオブジェクト
8Fh	プロトコルデータオブジェクトを切り替える
FF 6Dh	パラメ―タ―を取得する
FF 6Eh	パラメ―タ―を設定する

プロトコルの応答データオブジェクトを切り替える

タグ	データオブジェクト
C0h	一般的なエラーステータス
FF 6Dh	IFD パラメ―タ―データオブジェクト

5.2.3.4.1. プロトコルデータオブジェクトを切り替える（Switch Protocol Data Object）

このコマンドは、プロトコルおよび規格の異なる層を指定するために使用されます。

プロトコルデータオブジェクトを切り替える

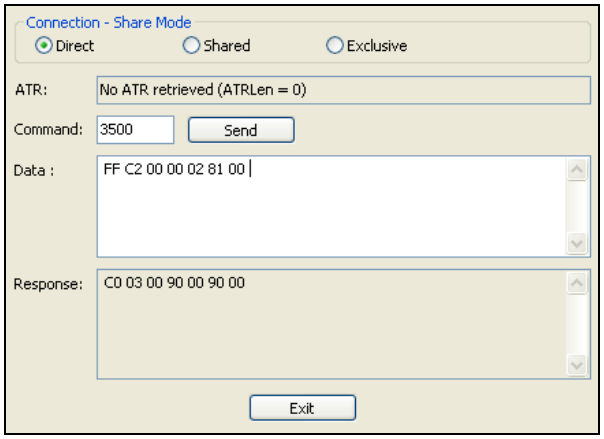
タグ	速度（1 バイト）	数値	
		バイト 0	バイト 1
8Fh	02h	00h – ISO/IEC 14443 A タイプ 01h – ISO/IEC 14443 B タイプ 03h – FeliCa 他 – RFU	00h – 層分離がない場合 02h – 2 層に切り替える 03h – 3 層に切り替えるまたは活性化する 04h – 4 層に活性化する 他 – RFU

5.2.3.5. PCSC 2.0 パート 3 の例

1. トランスペアレントセッションを開始する

   コマンド：**FF C2 00 00 02 81 00**

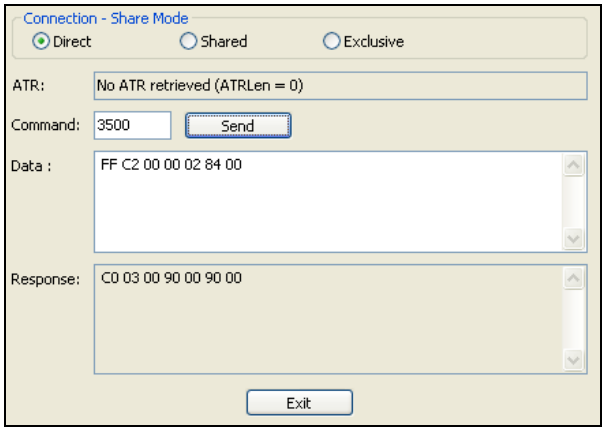
   応答：**C0 03 00 90 00 90 00**



2. アンテナフィールドをオンにする

   コマンド：**FF C2 00 00 02 84 00**

   応答：**C0 03 00 90 00 90 00**

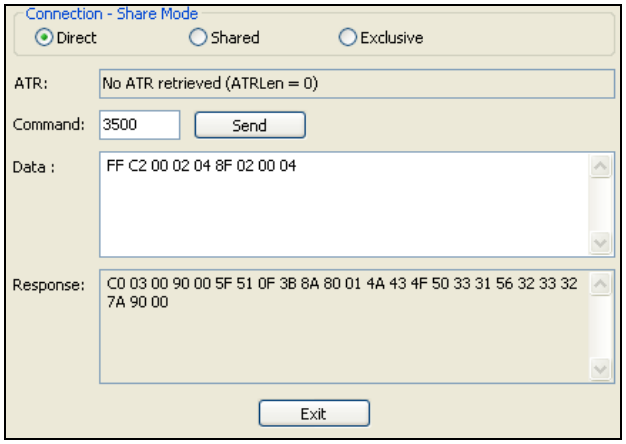


3. ISO14443-4A 有効。

   コマンド：**FF C2 00 02 04 8F 02 00 04**

   応答： **C0 03 01 64 01 90 00** (カードがない場合)

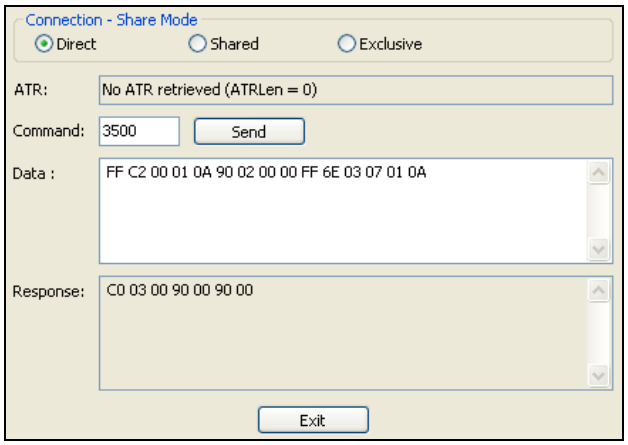
   C0 03 00 90 00 5F 51 [ATR] 90 00



4. 0AH に PCB を設定し、送信データで CRC、パリティ、プロトコルプロローグを有効にします。

   コマンド：**FF C2 00 01 0A 90 02 00 00 FF 6E 03 07 01 0A**

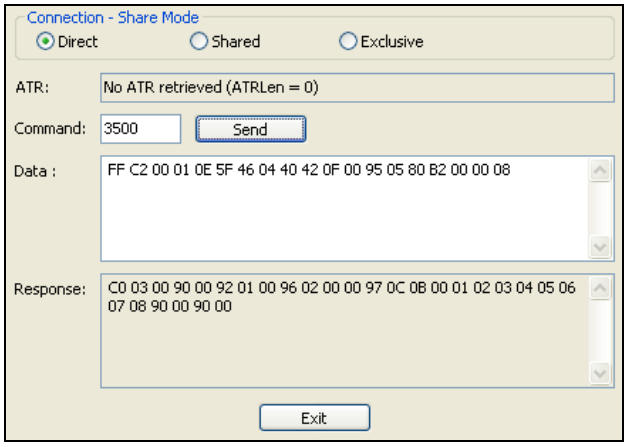
   応答：**C0 03 00 90 00 90 00**



5. カードに APDU「80B2000008」を送信し、応答を取得します。

   コマンド：**FF C2 00 01 0E 5F 46 04 40 42 0F 00 95 05 80 B2 00 00 08**

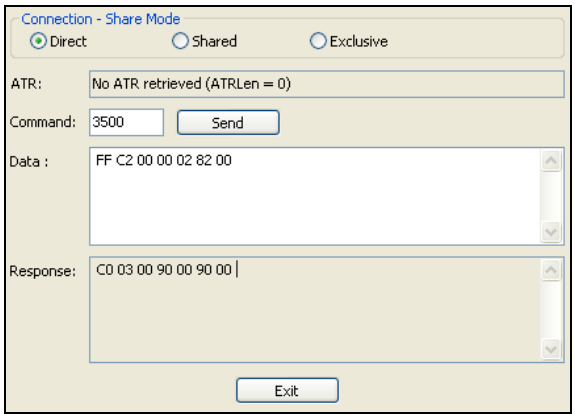
   応答：**C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [カードの応答] 90 00**



6. トランスペアレントセッションを終了する。

   コマンド：**FF C2 00 00 02 82 00**

   応答：**C0 03 00 90 00 90 00**



5.2.4. MIFARE Classic （1K/4K）メモリカードの PICC コマンド

5.2.4.1. 認証キーのダンロード（Load Authentication Keys）

このコマンドはリーダーにキーをロードする時に使われる。このキーは MIFARE Classic 1K/4K メモリカードの特定なセクターを認証するために使用される。

Load Authentication Keys APDU フォーマット（11 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Load Authentication Keys	FFh	82h	キー構造	キーナンバー	06h	キー（6 バイト）

その中：

キー構造 1 バイト。

00h = キーが揮発性キーのメモリにロードされる。

その他＝予約済み

キーの番号 1 バイト。

00h – 01h = 臨時のキーを保存するための揮発性キーのメモリ。リーダーが PC から切断された時、キーが消えます。揮発性キーは 2 つが設けられるので、異なるセッションのセッション鍵として使用することができます。デフォルト値 = {FF FF FF FF FF FFh}

キー 6 バイト。

リーダーのキーの数値をロードします、例：{FF FF FF FF FF FFh}

Load Authentication Keys 応答フォーマット（2 バイト）

応答	データ出力	
結果	SW1	SW2

Load Authentication Keys の応答コード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

例：

// 揮発性キーのメモリに 00h キーをロードする {FF FF FF FF FF FFh}。

APDU = {FF 82 00 00 06 FF FF FF FF FF FFh}

5.2.4.2. MIFARE Classic (1K/4K)カードに対しての認証（Authentication for MIFARE Classic (1K/4K）

このコマンドは、MIFARE Classic 1K/4K カード（PICC）との認証を行うためにリーダーに格納された鍵を使用しています。認証キーの二種類が用いられています：TYPE_A と TYPE_B。

Load Authentication Keys APDU フォーマット（6 バイト）「廃止された」

コマンド	CLA	INS	P1	P2	P3	データイン
Authentication	FFh	88h	00h	ブロック番号	キーのタイプ	キーナンバー

Load Authentication Keys APDU フォーマット（10 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Authentication	FFh	86h	00h	00h	05h	データバイト認証

データバイト認証（5 バイト）

バイト 1	バイト 2	バイト 3	バイト 4	バイト 5
バージョン番号 01h	00h	ブロック番号	キーのタイプ	キーナンバー

その中：

ブロック番号 1 バイト。認証されていないメモリブロック。

一枚の MIFARE Classic 1K カードが 16 個と分けて、各セクターには 4 個の連続的なブロックが含めています。例：セクター00h が含めているブロック{00h、01h、02h および 03h}；セクター01h が含めているブロック{04h、05h、06h および 07h}；ラストセクター0Fh が含めているブロック{3Ch、3Dh、3Eh および 3Fh}。当ブロックが成功に認証されると、同じセクターの全てのブロックをアクセスできる。詳しい情報は MIFARE Classic 1K/4 k 基準を参照してください。

***注釈：**ブロックが正常に認証されると、同セクターに所属する全てのブロックがアクセス可能である。

キーのタイプ 1 バイト。

60h = TYPE A キーとして、認証用に使われる。

61h = TYPE B キーとして、認証用に使われます。

キーの番号 1 バイト。

00h – 01h = キーを保存するための揮発性キーのメモリ。リーダーが PC から切断さ

れた時、キーが消えます。揮発性キーは 2 つが設けられるので、異なるセッションのセッション鍵として使用することができます。

Load Authentication Keys 応答フォーマット（2 バイト）

応答	データ出力	
結果	SW1	SW2

Load Authentication Keys の応答コード

結果	SW1	SW2	意味
成功	90h	00h	操作が成功に完了しました。
エラー	63h	00h	操作が失敗しました。

セクター (16 個のセクター，各セクターには 4 個の連続的なブロックが含めている)	データブロック (3 個のブロック，各には 16 バイト)	トレーラーブロック (1 つのブロック，16 バイト)
セクター0	00h – 02h	03h
セクター1	04h – 06h	07h
..	..	..
..	..	..
セクター14	38h – 0Ah	3Bh
セクター15	3Ch – 3Eh	3Fh

1 KB

表2 ：MIFARE Classic 1K カードのメモリマップ

セクター (32 個のセクター，各セクターには 4 個の連続的なブロックが含めている)	データブロック (3 個のブロック，各には 16 バイト)	トレーラーブロック (1 つのブロック，16 バイト)
セクター0	00h – 02h	03h
セクター1	04h – 06h	07h
..	..	..
..	..	..
セクター30	78h – 7Ah	7Bh

2 KB

セクター (32 個のセクター，各セクターには 4 個の連続的なブロックが含めている)	データブロック (3 個のブロック，各には 16 バイト)	トレーラーブロック (1 つのブロック，16 バイト)
セクター31	7Ch – 7Eh	7Fh

セクター (8 個のセクター，各セクターには 16 個の連続的なブロックが含めている)	データブロック (15 個のブロック，各には 16 バイト)	トレーラーブロック (1 つのブロック，16 バイト)
セクター32	80h – 8Eh	8Fh
セクター33	90h – 9Eh	9Fh
..	..	..
..	..	..
セクター38	E0h – EEh	EFh
セクター39	F0h – FEh	FFh

2 KB

表3 ：MIFARE Classic 4K カードのメモリマップ

バイトナンバー	0	1	2	3	ページ
シリアルナンバー	SN0	SN1	SN2	BCC0	0
シリアルナンバー	SN3	SN4	SN5	SN6	1
内部/ロック	BCC1	Internal	Lock0	Lock1	2
OTP	OPT0	OPT1	OTP2	OTP3	3
データリーダー/ライター	Data0	Data1	Data2	Data3	4
データリーダー/ライター	Data4	Data5	Data6	Data7	5
データリーダー/ライター	Data8	Data9	Data10	Data11	6
データリーダー/ライター	Data12	Data13	Data14	Data15	7
データリーダー/ライター	Data16	Data17	Data18	Data19	8
データリーダー/ライター	Data20	Data21	Data22	Data23	9
データリーダー/ライター	Data24	Data25	Data26	Data27	10
データリーダー/ライター	Data28	Data29	Data30	Data31	11
データリーダー/ライター	Data32	Data33	Data34	Data35	12
データリーダー/ライター	Data36	Data37	Data38	Data39	13
データリーダー/ライター	Data40	Data41	Data42	Data43	14
データリーダー/ライター	Data44	Data45	Data46	Data47	15

512 ビット
または
64 バイト

表4 :MIFARE Ultralight カードのメモリマップ

例：

// {TYPE A，キーナンバーの 00h}によって、ブロック 04h を認証します。PC/SC V2.01，廃止されます

APDU = {FF 88 00 04 60 00h};

// {TYPE A，キーナンバーの 00h}によって、ブロック 04h を認証します。PC/SC V2.07

APDU = {FF 86 00 00 05 01 00 04 60 00h}

注： *MIFARE Ultralight* のメモリは自由にアクセスできる。認証はいりません。

5.2.4.3. バイナリブロックを読み取る（Read Binary Blocks）

複数のデータブロックを PICC カードから取り出すことに使われます。Read Binary Blocks コマンドを実行する前に、データブロック/トレーラーブロックを認証しなければなりません

Read Binary の APDU フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	ブロック番号	読み取りバイト数

その中：

ブロック番号 1 バイト 読み取られていないスターティングブロック。

更新バイト数 1 バイト

- MIFARE Classic 1K/4K は 16byte の倍数、MIFARE Ultralight は 4 byte 倍数
- MIFARE Ultralight は最大 16 byte
- MIFARE Classic 1K は最大 48 byte（複数のブロックモード；3 つの連続ブロック）
- MIFARE Classic 4K は最大 240 byte（複数のブロックモード；15 つの連続ブロック）

ブロックデータ 16byte 又は 4byte の倍数。データはバイナリーブロックに書き込み

例 1：10h（16 バイト）。開始ブロックだけ（単一のブロックモード）。

例 2：40h（64 バイト）。開始ブロックから開始ブロックまで+3（複数のブロックモード）。

注：安全のために、複数のブロックモードはデータブロックだけにアクセスすることに使用されます。トレーラーブロックは複数のブロックモードでアクセスされません。単一のブロックモードを使用してください。

Read Binary Block の応答フォーマット（4/16 の倍数 + 2 バイト）

応答	データ出力		
結果	データ（4/16 バイトの倍数）	SW1	SW2

Read Binary Block 応答コード

結果	SW1	SW2	意味
成功	90h	00h	操作が成功に完了しました。
エラー	63h	00h	操作が失敗しました。

例：

// バイナリブロック 04h から 16 バイトを読み取る（MIFARE Classic　1K または 4K）

APDU = FF B0 00 04 10h

// バイナリブロック 80h から 240 バイトを読み出す（MIFARE Classic 4K）

// ブロック 80 からブロック 8Eh まで（15 個ブロック）

APDU = FF B0 00 80 F0h

5.2.4.4. バイナリブロックの更新（Update Binary Blocks）

Update Binary Blocks コマンドは複数のデータブロックを PICC カードに書き入れるのに使われる。このコマンドを実行する前に、データブロック/トレーラーブロックを認証しなければなりません

Update Binary の APDU フォーマット（16 の倍数 + 5 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Update Binary Blocks	FFh	D6h	00h	ブロック番号	読み取りバイト数	データブロック（16 バイトの倍数）

その中：

ブロック番号 1 バイト。更新される開始ブロック

更新バイト数 1 バイト。

- MIFARE Classic 1K/4K は 16byte の倍数、MIFARE Ultralight は 4byte 倍数
- MIFARE Classic 1K は最大 48byte（複数のブロックモード；3 つの連続ブロック）
- MIFARE Classic 4K は最大 240byte（複数のブロックモード；15 つの連続ブロック）

ブロックデータ 16byte 又は 4byte の倍数 、バイナリブロックに書き込まれるデータ。

例 1：10h（16 バイト）。開始ブロックだけ（単一のブロックモード）。

例 2：30h（48 バイト）。開始ブロックから開始ブロックまで+2（複数のブロックモード）。

注：安全のために、複数のブロックモードはデータブロックだけにアクセスすることに使用されます。トレーラーブロックは複数のブロックモードでアクセスされません。単一のブロックモードを使用してください。

Update Binary Block 応答コード（2 バイト）

結果	SW1	SW2	意味
成功	90	00h	操作が成功に完了しました。
エラー	63	00h	操作が失敗しました。

例：

// MIFARE Classic 1K/4K カード中のバイナリブロック 04h のデータを{00 01 ..0Fh}に更新します

APDU = {FF D6 00 04 10 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0Fh}

//MIFARE Ultralight 中のバイナリブロック 04 h を{00 01 02 03}に更新する

APDU = {FF D6 00 04 04 00 01 02 03h}

5.2.4.5. 数値ブロックの操作（Value Block Operation）（INC, DEC, STORE）

このコマンドは数値を基づいてのトランザクションを実行する時に使われます（例：数値ブロックの数値を増える）。

Value Block Operation の APDU フォーマット（10 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Value Block Operation	FFh	D7h	00h	ブロック番号	05h	VB_OP	VB_Value（4 バイト）{MSB ..LSB}

その中：

ブロック番号 1 バイト。操作されていない数値のブロック

VB_OP 1 バイト。

00h = VB_Value をブロックにストアーして、このブロックは数値ブロックになります。

01h = VB_Value によって、数値ブロックの数値をインクリメントするこのコマンドは数値ブロック対しての操作のみに適用しています。

02h = VB_Value によって、数値ブロックの数値をデクリメントする。このコマンドは数値ブロック対しての操作のみに適用しています。

VB_Value 4 バイト。数値の操作に使用される符号付き長い整数です（4 バイト）。

例 1： Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB	-	-	LSB
FFh	FFh	FFh	FCh

例 2： Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB	-	-	LSB
00h	00h	00h	01h

Value Block Operation の応答フォーマット（2 バイト）

応答	データ出力	
結果	SW1	SW2

Value Block Operation 応答コード

結果	SW1	SW2	意味
成功	90	00h	操作が成功に完了しました。
エラー	63	00h	操作が失敗しました。

5.2.4.6. 数値ブロックを読み取る（Read Value Block）

このコマンドは数値ブロックの数値を取得するために使われます。数値ブロック対しての操作のみに適用しています。

Value Block Operation の APDU フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Read Value Block	FFh	B1h	00h	ブロック番号	04h

その中：

ブロック番号　1 バイト。読み書かれていない数値ブロック。

Read Value Block の応答フォーマット（4 + 2 バイト）

応答	データ出力		
結果	数値 {MSB ..LSB}	SW1	SW2

その中：

値　4 バイト。カードから返された数値で、符号付き長い整数です（4 バイト）

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

数値			
MSB	-	-	LSB
FFh	FFh	FFh	FCh

例 2：Decimal 1 = {00h, 00h, 00h, 01h}

数値			
MSB	-	-	LSB
00h	00h	00h	01h

Read Value Block コマンドの応答コード

結果	SW1	SW2	意味
成功	90	00h	操作が成功に完了しました。
エラー	63	00h	操作が失敗しました。

5.2.4.7. 数値ブロックをコピーする（Copy Value Block）

このコマンドは一つの数値ブロック中の数値を別の数値ブロックにコピーする時に使われます。

Copy Value Block の APDU フォーマット（7 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Copy Value Block	FFh	D7h	00h	**ソース ブロック番号**	02h	03h	ターゲット ブロック番号

その中：

元ブロックの番号 1 バイト。ソース値のブロックの値が目標値ブロックにコピーされる。

ターゲットブロック番号 1 バイト。復元する値ブロック。ソースとターゲット値のブロックは、必ず同じセクター内にある。

Copy Value Block の応答フォーマット（2 バイト）

応答	データ出力	
結果	SW1	SW2

Copy Value Block の応答コード

結果	SW1	SW2	意味
成功	90	00h	操作が成功に完了しました。
エラー	63	00h	操作が失敗しました。

例：

//数値"1"を数値ブロック 05h にストアーします。

APDU = {FF D7 00 05 05 00 00 00 00 01h}

// 数値ブロック 05h を読み取ります。

APDU = {FF B1 00 05 04h}

//数値をブロック 05h からブロック 06h にコピーする。

APDU = {FF D7 00 05 02 03 06h}

//ブロック 05h の値を 5 にインクリメントする。

APDU = {FF D7 00 05 05 01 00 00 00 05h}

5.2.5. PC/SC 規格に準拠しているタグにアクセスする（ISO 14443-4）

基本的に、すべての ISO14443-4 に準拠したカード（PICC カード）は、ISO7816-4 の APDU を理解できます。ACM1252U-Z2 カードリーダーは ISO 7816-4 基準の APDU および応答を交換することによって、ISO14443-4 基準のカードと通信することができます。ACM1252U-Z2 は内部で ISO14443 の 1 – 4 パートのプロトコルを処理します。

MIFARE Classic （1K/4K）、MIFARE Mini および MIFARE Ultralight タグは T=CL エミュレーションを介してサポートされます。MIFARE タグを標準な ISO 14443-4 タグとして取り扱えばいいです。詳しい情報については、5.5. MIFARE Classic （1K/4K）メモリカードの PICC コマンドを参照してください。

ISO 7816-4 仕様の APDU フォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	Le
ISO 7816 4 パートのコマンド					データの長さ		応答データの予想の長さ

ISO 7816-4 仕様の応答データフォーマット（データ+2 バイト）

応答	データ出力		
結果	応答データ	SW1	SW2

一般的な ISO 7816-4 コマンドの応答コード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

典型的なシーケンスは：

1. タグを提出して、PICC インターファースと接続します。
2. タグ中の情報を読み取り/更新する。

これを実行します：

1. タグと接続する。

   タグの ATR は 3B 88 80 01 00 00 00 00 33 81 81 00 3Ah です。

   その中、

   ATQB アプリケーションのデータ= 00 00 00 00、ATQB プロトコル 情報= 33 81 81。これは ISO 14443-4 Type B タグです。

2. APDU を送信して、乱数を入手する。

   00 84 00 00 08

   >> 1A F7 F3 1B CD 2B A9 58h [90 00h]

   注：*对于 ISO 14443-4 Type A のタグに対して、APDU"FF CA 01 00 00h"によって ATS を入手する。*

例：

// ISO 14443-4 Type B PICC（ST19XR08E）から 8 バイトを読み取ります。

APDU = {80 B2 80 00 08h}

CLA = 80h

INS = B2h

P1 = 80h

P2 = 00h

Lc = なし

データ＝なし

Le = 08h

応答：00 01 02 03 04 05 06 07h [$9000h]

5.2.6. FeliCa タグのアクセス

FeliCa タグをアクセスするためのコマンドは PC/SC 基準の MIFARE タグをアクセスするためのコマンドとは違います。

このコマンドは FeliCa 基準に準拠して、ヘッダが追加されています。

FeliCa コマンドのフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン
FeliCa コマンド	FFh	00h	00h	00h	データの長さ	FeliCa コマンド （長さバイトで始まる）

FeliCa の応答データフォーマット（データ+2 バイト）

応答	データ出力
結果	応答データ

例のメモリブロックデータの読み取り

1. FeliCa を接続する。

   The ATR = 3B 8F 80 01 80 4F 0C A0 00 00 03 06 11 00 3B 00 00 00 00 42h

   その中：11 00 3Bh = FeliCa

2. FeliCa IDM の読み取り。

   コマンド = FF CA 00 00 00h

   応答 = [IDM (8 バイト)] 90 00h

   例：FeliCa IDM = 01 01 06 01 CB 09 57 03h

3. FeliCa コマンドアクセス。

   例：メモリブロックデータの「読み取り」

   コマンド = FF 00 00 00 10 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   その中：

   Felica コマンド = 10 06 01 01 06 01 CB 09 57 03 01 09 01 01 80 00h

   IDM = 01 01 06 01 CB 09 57 03h

   応答 = メモリブロックデータ

5.3. 周辺デバイス制御

リーダーの周辺機器制御コマンドは、制御コードの **SCARD_CTL_CODE（3500）**で **SCardControl** を使用して実装されています。

5.3.1. ファームウェアのバージョンを取得する（Get Firmware Version）

このコマンドはファームウェアのバージョンを入手する時に使われます。

Get Firmware Version のコマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Get Firmware Version	E0h	00h	00h	18h	00h

Get Firmware Version の応答フォーマット（5 バイト + ファームウェアメッセージの長さ）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	受信していないバイトの数量	ファームウェアのバージョン番号

例：

応答 = E1 00 00 00 0F 41 43 52 31 32 35 32 55 5F 56 32 30 32 2E 32

ファームウェアのバージョン番号（HEX） = 41 43 52 31 32 35 32 55 5F 56 32 30 32 2E 32

ファームウェアのバージョン番号 （ASCII） = “ACR1252U_V202.2”

5.3.2. LED 制御（LED Control）

このコマンドは LCD の出力を制御するために使用されます。

LED Control コマンドフォーマット（6 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
LED Control	E0h	00h	00h	29h	01h	LED 状態

LED Control 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	LED 状態

LED 状態（1 バイト）

LED 状態	説明	説明
Bit 0	レッド	1 = ON；0 = OFF
Bit 1	グリーン	1 = ON；0 = OFF
Bit 2 - 7	RFU	RFU

5.3.3. LED 状態（LED Status）

このコマンドは LED の状態を検査するために使用されます。

LED Control コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
LED Status	E0h	00h	00h	29h	00h

LED Status 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	LED Status

LED 状態（1 バイト）

LED 状態	説明	説明
Bit 0	レッド	1 = ON；0 = OFF
Bit 1	グリーン	1 = ON；0 = OFF
Bit 2 - 7	RFU	RFU

5.3.4. PICC インターフェースの LED ステータス Indicator を設定する（Set LED Status Indicator Behavior for PICC interface）

このコマンドは LED ステータス Indicator を PICC インターフェースのステータスンジケータとして設定するために使用されます。

注：この設置は揮発性キーのメモリに保存されます。

Set LED Status Indicator Behavior コマンドフォーマット（6 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Set LED Status Indicator Behavior	E0h	00h	00h	21h	01h	操作

操作（1 バイト）

操作	モード	説明
Bit 0	カードに操作する時 LED が点滅します。	カードは PICC がアクセスされる時 LED が点滅します。
Bit 1	PICC のポーリングステータス LED	PICC のポーリングステータスを表示する 1 = 有効；0 = 無効
Bit 2	PICC 活性化状態の LED	**PICC インタフェース活性化状態表示** 1 = 有効；0 = 無効
Bit 3 - 5	RFU	RFU
Bit 6	オプションの色（緑）	緑の LED は、ステータス変更を意味します 1 = 有効；0 = 無効
Bit 7	オプションの色（赤）	赤の LED は、ステータス変更を意味します 1 = 有効；0 = 無効

注：デフォルトの操作の値= *7Fh*

Set LED Status Indicator Behavior 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	デフォルト操作

5.3.5. PICC インターフェースの LED ステータス Indicator を読み取る（Read LED Status Indicator Behavior for PICC Interface）

このコマンドは PICC インターフェースの LED ステータス Indicator を読み取る時に使われます。

Read LED Status Indicator Behavior コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read LED Status Indicator Behavior	E0h	00h	00h	21h	00h

Read LED Status Indicator Behavior 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	操作

操作（1 バイト）

操作	モード	説明
Bit 0	カードに操作する時 LED が点滅します。	カードは PICC がアクセスされる時 LED が点滅します。
Bit 1	PICC のポーリングステータス LED	PICC のポーリングステータスを表示する 1 = 有効；0 = 無効
Bit 2	PICC 活性化状態の LED	PICC インタフェース活性化状態表示 1 = 有効；0 = 無効
Bit 3 - 5	RFU	RFU
Bit 6	オプションの色（緑）	緑の LED は、ステータス変更を意味します 1 = 有効；0 = 無効
Bit 7	オプションの色（赤）	赤の LED は、ステータス変更を意味します 1 = 有効；0 = 無効

注：デフォルトの操作の値= *7Fh*

5.3.6. 自動的な PICC のポーリングを設置する（Set Automatic PICC Polling）

このコマンドはカードリーダーのポーリングモードを設置する時に使われます。

リーダーが PC に接続されるたびに、PICC ポーリング機能が自動的に PICC のスキャンを開始して、内蔵アンテナに置かれる/から削除される PICC があるかどうか確認します。

コマンドを送信して、PICC のポーリングを無効にできます。このコマンドは PC/SC Escape コマンドのインターフェースで送信されます。エネルギを節約するために、PICC が活動していない、または PICC が見つからない時、いつでもアンテナフィールドをオフにするための特別なモードが設けられている。省電力モードで、リーダーはもっと少ない電流を消費します。

注：この設置は揮発性キーのメモリに保存されます。*Bit 6* がファームウェア *208.0* 以降に適用します。

Set Automatic PICC Polling コマンドフォーマット（6 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Set Automatic PICC Polling	E0h	00h	00h	23h	01h	ポーリング設定

Set Automatic PICC Polling 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	ポーリング設定

ポーリング設定（1 バイト）

ポーリング設定	モード	説明
Bit 0	自動的に PICC ポーリング	1 = 有効；0 = 無効
Bit 1	PICC が見つかっていない場合は、アンテナフィールドをオフにします。	1 = 有効；0 = 無効
Bit 2	PICC が活動していない場合、アンテナフィールドをオフにします。	1 = 有効；0 = 無効
Bit 3	RFU	
Bit 5 ..4	PICC の PICC ポーリング間隔	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms

ポーリング設定	モード	説明
Bit 6	NXP MIFARE ISO/IEC 14443 PICC 選択	1 = 有効；0 = 無効
Bit 7	ISO 14443-A 4 パートを強制に実行します。	1 = 有効；0 = 無効

注：ポーリング設置のデフォルト値 = *8Bh*

提示：

1. 「*PICC* が活動していない場合、アンテナフィールドをオフにする」、そのオプションを有効にすることをお勧めします。そうしたら、活動していない *PICC* はずっとアンテナフィールドに公開されなくて、*PICC* の「ウォーミングアップ」を防ぎます。
2. *PICC* ポーリング間隔の長さに関わって、省エネルギがより効率になります。しかし、*PICC* ポーリングの応答時間が長くなります。省エネルギ状態で *Idle* 消費電流は *60 mA* です；非省エネルギ状態で *Idle* 消費電流は *130 mA* です。

   注釈：dle 消費電流=PICC が活性化されていない。
3. リーダーは自動的に*"ISO 14443A-4 PICC"*の *ISO 14443A-4* モードを有効にします。*B* タイプの *PICC* はこのオプションによって影響を受けることはありません。
4. *JCOP30* カードには二つのモードを持っている：*ISO 14443A-3*（*MIFARE 1K*）と *ISO 14443A-4* モード。*PICC* を有効にすると、アプリケーションは一つのモードを選択しなければなりません。
5. 「*NXP MIFARE ISO / IEC 14443 PICC* 選択」オプションを有効にすると、*SAK 28h* は *Mifare Classic 1K* カードとして認識され、*SAK 38h* は *Mifare Classic 4K* カードとして認識されます。

5.3.7. 自動的な PICC のポーリングを読取る（Read Automatic PICC Polling）

このコマンドは現在の PICC のポーリングの状態の設置を検査するために使用されます。

注釈： *Bit 6* がファームウェア *208.0* 以降に適用します。

Read Automatic PICC Polling コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read Automatic PICC Polling	E0h	00h	00h	23h	00h

Read Automatic PICC Polling 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	ポーリング設定

ポーリング設定（1 バイト）

ポーリング設定	モード	説明
Bit 0	自動的に PICC ポーリング	1 = 有効；0 = 無効
Bit 1	PICC が見つかっていない場合は、アンテナフィールドをオフにします。	1 = 有効；0 = 無効
Bit 2	PICC が活動していない場合、アンテナフィールドをオフにします。	1 = 有効；0 = 無効
Bit 3	RFU	
Bit 5 ..4	PICC の PICC ポーリング間隔	<Bit 5 – Bit 4> <0 – 0> = 250 ms <0 – 1> = 500 ms <1 – 0> = 1000 ms <1 – 1> = 2500 ms
Bit 6	NXP MIFARE ISO/IEC 14443 PICC 選択	1 = 有効；0 = 無効
Bit 7	ISO 14443-A　4 パートを強制に実行します。	1 = 有効；0 = 無効

注：ポーリング設置のデフォルト値 = *8Bh*

5.3.8. PICC 操作のパラメーターを設定する（Set PICC Operating Parameter）

このコマンドは PICC 操作のパラメーターを設定するために使われます。

注：この設置は揮発性キーのメモリに保存されます。

Set PICC Operating Parameter コマンドフォーマット（6 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Set PICC Operating Parameter	E0h	00h	00h	20h	01h	操作パラメーター

Set PICC Operating Parameter 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1	00h	00h	00h	01h	操作パラメーター

操作パラメーター（1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	ISO 14443 A タイプ	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
Bit 2	FeliCa 212 Kbps		1 = 検出 0 = スキップ
Bit 3	FeliCa 424 Kbps		1 = 検出 0 = スキップ
Bit 4	Topaz		1 = 検出 0 = スキップ
Bit 5 - 7	RFU	RFU	RFU

注：操作のデフォルト値 = *1Fh*

5.3.9. PICC 操作のパラメーターを読取る（Read PICC Operating Parameter）

このコマンドは PICC 操作のパラメーターを検査するために使用されます。

Read PICC Operating Parameter コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read PICC Operating Parameter	E0h	00h	00h	20h	00h

Read PICC Operating Parameter 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	操作パラメーター

操作パラメーター（1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	ISO 14443 A タイプ	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
Bit 2	FeliCa 212 Kbps		1 = 検出 0 = スキップ
Bit 3	FeliCa 424 Kbps		1 = 検出 0 = スキップ
Bit 4	Topaz		1 = 検出 0 = スキップ
Bit 5 - 7	RFU	RFU	RFU

注：操作のデフォルト値 = *1Fh*

5.3.10. PICC 操作のパラメーターを設定する（拡張）（Set PICC Operating Parameter）

このコマンドは PICC 操作のパラメーターを設定するために使われます。

注釈：この設置は失いやすいキーのメモリに保存されます。*208.0* 以降のバジョンのみに適用します。

Set PICC Operating Parameter コマンドフォーマット（7 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Set PICC Operating Parameter	E0h	00h	01h	20h	02h	操作パラメーター1	操作パラメーター2

Set PICC Operating Parameter 応答フォーマット（7 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1	00h	01h	00h	02h	操作パラメーター1	操作パラメーター2

操作パラメーター1（ 1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	ISO 14443 A タイプ	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
Bit 2	FeliCa 212 Kbps		1 = 検出 0 = スキップ
Bit 3	FeliCa 424 Kbps		1 = 検出 0 = スキップ
Bit 4	Topaz		1 = 検出 0 = スキップ
Bit 5	RFU		RFU
Bit 6	SRIX		1 = 検出 0 = スキップ
Bit 7	RFU	RFU	RFU

注：操作パラメーター*1* のデフォルト値 = *5Fh*

操作パラメ—ター—2（1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	Picopass	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1 - 7	RFU	RFU	RFU

注：操作パラメーター*2* のデフォルト値 = *01h*

5.3.11. PICC 操作のパラメーターを読取る（拡張）（Read PICC Operating Parameter）

このコマンドは PICC 操作のパラメーターを検査するために使用されます。

注： 208.0 以降のバジョンのみに適用します。

Read PICC Operating Parameter コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Read PICC Operating Parameter	E0h	00h	01h	20h	00h

Read PICC Operating Parameter 応答フォーマット（7 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1	00h	01h	00h	02h	操作パラメーター1	操作パラメーター2

操作パラメーター1（ 1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	ISO 14443 A タイプ	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
Bit 2	FeliCa 212 Kbps		1 = 検出 0 = スキップ
Bit 3	FeliCa 424 Kbps		1 = 検出 0 = スキップ
Bit 4	Topaz		1 = 検出 0 = スキップ
Bit 5	RFU		RFU
Bit 6	SRIX		1 = 検出 0 = スキップ
Bit 7	RFU	RFU	RFU

注： 操作パラメーター*1* のデフォルト値 = *5Fh*

操作パラメ—ター—2（1 バイト）

操作パラメーター	パラメーター	説明	オプション
Bit 0	Picopass	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
Bit 1 - 7	RFU	RFU	RFU

注：操作パラメーター*2* のデフォルト値 = *01h*

5.3.12. 自動的な PPS を設定する（Set Auto PPS）

PICC が認識されるたびに、リーダーは最大接続速度によって定義された PCD および PICC との間の通信速度を変更しようとします。カードが提案された接続速度をサポートしていない場合、リーダーはより遅い速度でとカードと接続しようとします。

Set Auto PPS コマンドフォーマット（7 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Set Auto PPS	E0h	00h	00h	24h	02h	Max Tx Speed	Max Rx Speed

Set Auto PPS 応答フォーマット（9 バイト）

応答	CLA	INS	P1	P2	Le	データ出力			
結果	E1h	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

その中：

Max Tx Speed 最大 Tx 速度（1 バイト）

Current Tx Speed 現在の Tx 速度（1 バイト）

Max Rx Speed 最大 Rx 速度（1 バイト）

Current Rx Speed 現在の Rx 速度（1 バイト）

00h = 106 Kbps；デフォルト設定、自動 PPS が設定されていないと同じです。

01h = 212 Kbps

02h = 424 Kbps

注釈：

1. 通常、アプリケーションが使用中の *PICC* の最大接続速度を知っている必要があります。環境にも達成可能な最大速度に影響します。リーダーは提案されている通信速度を使用して、*PICC* と話をします。*PICC* や環境が提案されている通信速度の要件を満たしていない場合、*PICC* はアクセスできなくなります。
2. リーダーは、送信側と受信側との間の異なる速度をサポートしています。

5.3.13. 自動的な PPS を読み取る（Read Auto PPS）

このコマンドは現在の自動的な PPS の設置を検査するために使用されます。

Read Auto PPS コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read Auto PPS	E0h	00h	00h	24h	00h

Read Auto PPS 応答フォーマット（9 バイト）

応答	CLA	INS	P1	P2	Le	データ出力			
結果	E1h	00h	00h	00h	04h	Max Tx Speed	Current Tx Speed	Max Rx Speed	Current Rx Speed

その中：

Max Tx Speed 最大 Tx 速度（1 バイト）

Current Tx Speed 現在の Tx 速度（1 バイト）

Max Rx Speed 最大 Rx 速度（1 バイト）

Current Rx Speed 現在の Rx 速度（1 バイト）

00h = 106 Kbps；デフォルト設定、自動 PPS が設定されていないと同じです。

01h = 212 Kbps

02h = 424 Kbps

5.3.14. シリアルナンバーを読み取る（Read Serial Number）

シリアルナンバーを読み取る時にこのコマンドを使用します。

Read Serial Number のコマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read Serial Number	E0h	00h	00h	33h	00h

Read Serial Number 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1	00h	00h	00h	長さ	シリアルナンバー (N バイト)

5.3.15. PICC タイプ読み取り（Read PICC Type）

このコマンドは現在の PICC タイプを確認するために使用されます。

注：*208.0* 以降のバジョンのみに適用します。

Read PICC Type フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read PICC Type	E0h	00h	00h	35h	00h

Read PICC Type 応答フォーマット（7 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	カードタイプ	カードステータス

その中：

カードのタイプ　1 バイト

CCh = ない

04h = Topaz

10h = Mifare

11h = Felica 212 Kbps

12h = Felica 424 Kbps

20h = ISO 14443-4 B タイプ

23h = ISO 14443-4 B タイプ

28h = Srix

30h = Picopass

カード状態　1 バイト

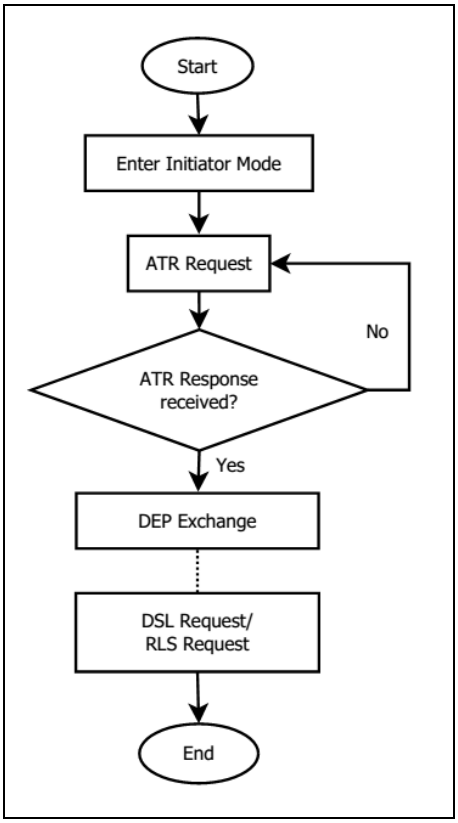
00h = PICC パワーダウン [タグが検出されていない]

そのほか = PICC 検出 [非接触式タグ検出された]

5.4. NFC　P２Pについてのコマンド

5.4.1. イニシエータモードについてのコマンド

本節はイニシエータモードで使用可能なコマンドを紹介します。次の図示はこのモードにコマンドがＰ２Ｐの流れを示します。



図示 4:イニシエータモードでの P2P 流れ

5.4.1.1. イニシエータモードタイムアウトを設定（Set Initiator Mode Timeout）

このコマンドはイニシエータモードタイムアウトを設定する時に使われます。

Set Initiator Mode Timeout コマンドフォーマット （7 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Set Initiator Mode	E0h	00h	00h	41h	02h	タイムアウト（MSB）	タイムアウト（LSB）

注：単位 = *10 ms*；イニシエータモード のタイムアウトのデフォルト値= *00 64h (100 * 10 ms = 1000 ms*）。

Set Initiator Mode Timeout 応答フォーマット （7 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	タイムアウト（MSB）	タイムアウト（LSB）

その中：

タイムアウト時間　2 バイト。イニシエータモードのタイムアウト（単位＝10 ms）。

5.4.1.2. イニシエータモードに入るモード（Enter Initiator Mode）

このコマンドは SNEP メッセージを送信するために、リーダーをイニシエータモードに入るモードに設置するときに使われます。

Enter Initiator Mode コマンドフォーマット（8 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン		
Enter Initiator Mode	E0h	00h	00h	40h	03h	NFCMode	OpMode	速率

Enter Initiator Mode フォーマット （8 バイト）

応答	CLA	INS	P1	P2	Le	データ出力		
結果	E1h	00h	00h	00h	03h	NFCMode	OpMode	速率

その中：

NFCMode 1 バイト。NFC デバイスモード

01h = MIFARE Ultralight カードエミュレーションモード

03h = FeliCa カードエミュレーションモード

08h = P2P イニシエータモード

00h = カードのリーダ/ライタモード

OpMode 1 バイト。アクティブモード/ パッシブモード

01h = アクティブモード

02h = パッシブモード

速率 1 バイト通信速度

01h = 106 Kbps

02h = 212 Kbps

03h = 424 Kbps

Enter Initiator Mode コマンドを実行した後、リーダーがターゲットモード状態の NFC デバイスを待って、予め設定された SNEP メッセージを提示して、NFC デバイスに送信します。SNEP メッセージを成功に送信するまで、リーダーは他の操作を実行しません。

5.4.1.3. ATR リクエストを送信する（Send ATR Request）

このコマンドは、フィールド内の P 2 P のターゲットモードデバイスに ATR_REQ を送信するために使用されます。

ATR Request コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン				
ATR Request	E0h	00h	00h	42h	長さ	11h	モード（1 バイト）	速度（1 バイト）	NFCID（10 バイト）	DID（1 バイト）

データイン				
BS（1 バイト）	BR（1 バイト）	PP（1 バイト）	LLCP パラメ―タ―	
			GiLen （1 バイト）	Gi（GiLen バイト）

ATR Request 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	長さ	ATR 応答（Len バイト）

その中：

モード　1 バイト動作モード
01h＝アクティブ
02h = パッシブ

速度　1 バイト。通信速度
01h = 106 Kbps
02h = 212 Kbps
03h = 424 Kbps

NFCID　10 バイト。イニシエータデバイスの NFCID

DID　1 バイト。イニシエータデバイスのデバイス識別

BS　1 バイト。イニシエータデバイスがサポートできる送信ビットレート。

BR　1 バイト。イニシエータデバイスがサポートできるビットレート。

PP　1 バイト。イニシエータデバイスのオプションパラメーター

Gi　N バイト。LLCP パラメ―タ―

5.4.1.4. DEP 交換（Exchange DEP）

このコマンドでターゲットデバイスと DEP を交換します。

DEP Exchange コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン			
DEP Exchange	E0h	00h	00h	43h	長さ	11h	PFB（1 バイト）	DepLen（1 バイト）	Dep (N バイト)

DEP Exchange 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	長さ	Dep 応答（Len バイト）

その中：

PFB 1 バイト。データ伝送とエラー回復を制御する。

DepLen 1 バイト。DEP メッセージの長さ。

Dep N バイト。DEP メッセージは P 2 P 通信に使われます。

5.4.1.5. DSL リクエストを送信する（Send DSL Request）

このコマンドは DSL リクエストをターゲットデバイスに送信するときに使われます。

DSL Request コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	
DSL request	E0h	00h	00h	44h	02h	11h	DID（1 バイト）

その中：

DID 1 バイト。デバイス識別

DSL Request 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.1.6. RLS リクエストを送信する（Send RLS Request）

このコマンドは RLS リクエストをターゲットデバイスに送信するときに使われます。

RLS Request コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	
RLS request	E0h	00h	00h	45h	02h	11h	DID（1 バイト）

その中：

DID 1 バイトのデバイスフラグ。

RLS Request コマンドフォーマット

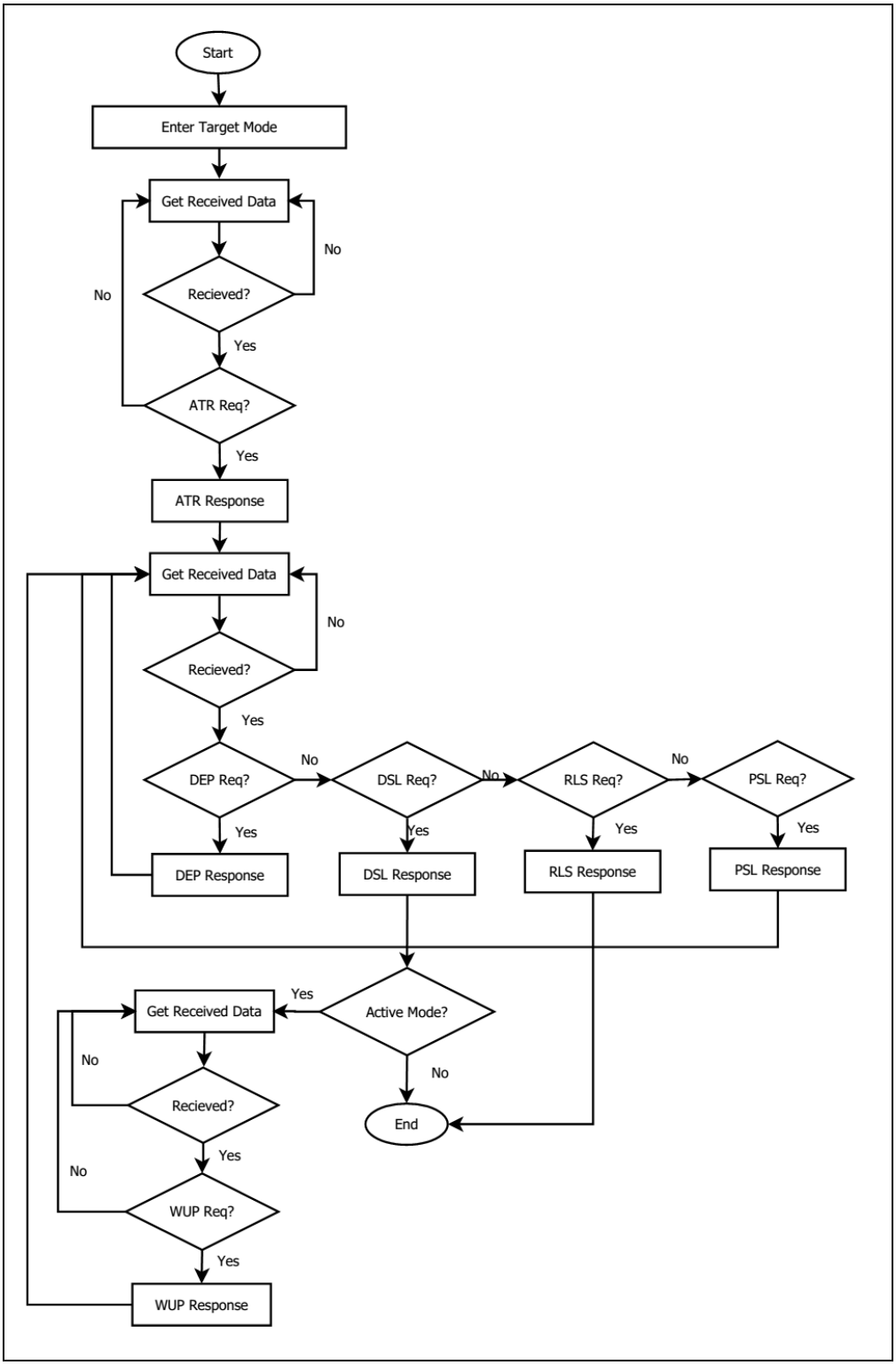
応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2. ターゲットモードについてのコマンド

本節はターゲットモードで使用可能なコマンドを紹介します。次の図示はこのモードにコマンドがＰ２Ｐの流れを示します。



図示 5：ターゲットモードでの P2P 流れ

5.4.2.1. ターゲットモードタイムアウトを設定（Set Target Mode Timeout）

このコマンドはターゲットモードタイムアウトを設定する時に使われます。

Set Target Timeout コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	
Set Target Timeout	E0h	00h	00h	59h	02h	Timeout (MSB)	Timeout (LSB)

注：単位 *100 µs*；ターゲットタイムアウトのデフォルト値 = *00 C8h (200 * 100 µs = 20 ms)*。

Set Target Timeout 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	Timeout (MSB)	Timeout (LSB)

その中：

Timeout 2 バイト。ターゲットモードのタイムアウト（単位＝100 ms）。

5.4.2.2. ターゲットモードに入る（Enter Target Mode）

このコマンドは SNEP メッセージを受信するために、リーダーをターゲットモードに入るように設置するときに使われます。

Enter Target Mode コマンドフォーマット（8 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン	
Enter Target Mode	E0h	00h	00h	51h	02h	速度	OpMode

Enter Target Mode フォーマット （8 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	速度	OpMode

Enter Target Mode 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	速度	OpMode

その中：

速度 1 バイト。通信速度

01h = 106 Kbps

02h = 212 Kbps

03h = 424 Kbps

OpMode 1 バイト。アクティブモード/ パッシブモード

01h = アクティブモード

02h = パッシブモード

Enter Target Mode コマンドを実行した後、リーダーがイニシエータモード状態の NFC デバイスを待って、SNEP メッセージを提示して、受信します。SNEP メッセージを成功に交換するまで、リーダーは他の操作を実行しません。

5.4.2.3. ATR 応答を送信する（Send ATR Response）

このコマンドでイニシエータの ATR リクエストに対しての ATR 応答を送信します。

ATR Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン
ATR Response	E0h	00h	00h	52h	長さ	LLCP パラメーター (N バイト)

その中：

LLCP パラメーター　　N バイト。ATR 応答の一般的なバイト。

ATR 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.4. DEP 応答を送信する（Send DEP Response）

このコマンドでイニシエータの DEP リクエストに対しての DEP 応答を送信します。

DEP Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	
DEP Response	E0h	00h	00h	53h	長さ	PFB（1 バイト）	DEP メッセージ (N バイト)

その中：

PFB 1 バイト。データ伝送とエラー回復を制御する。

DEP **メッセージ** N バイト。DEP 応答。

DEP 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.5. DSL 応答を送信する（Send DSL Response）

このコマンドでイニシエータの DSL リクエストに対しての DSL 応答を送信します。

DSL Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc
DSL Response	E0h	00h	00h	54h	00h

DSL 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.6. RLS 応答を送信する（Send RLS Response）

このコマンドでイニシエータの RLS リクエストに対しての RLS 応答を送信します。

RLS Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc
RLS Response	E0h	00h	00h	55h	00h

RLS 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.7. PSL 応答を送信する（Send PSL Response）

このコマンドでイニシエータの PSL リクエストに対しての PSL 応答を送信します。

PSL Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン	
PSL Response	E0h	00h	00h	56h	02h	BRS（1 バイト）	FSL（1 バイト）

その中：

BRS 1 バイト。BRS パラメ―タ―。

FSL 1 バイト。FSL パラメ―タ―。

PSL 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（2 バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.8. WUP 応答を送信する（Send WUP Response）

このコマンドでイニシエータの WUP リクエストに対しての WUP 応答を送信します。

WUP Response コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc
WUP Response	E0h	00h	00h	57h	00h

WUP 応答フォーマット

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	02h	リターンコード（２バイト）

リターンコード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。
エラー	63 00h	操作が失敗しました。

5.4.2.9. 受信したデータを入手する（Get Received Data）

このコマンドは NFC イニシエータ装置から受信したデータを取得するために使用されます。

Get Received Data フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Get Received Data	E0h	00h	00h	58h	00h

Get Received Data フォーマット（11 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	SNEP メッセージの長さ	SNEP メッセージ

その中：

SNEP メッセージの長さ　　1 バイト。受信した SNEP メッセージの長さ.

SNEP メッセージ　イニシエータデバイスから SNEP メッセージを受信しました。

5.5. NFC カードエミュレーションについてのコマンド

5.5.1. カードエミュレーションモードに入る（Enter Card Emulation Mode）

このコマンドは、MIFARE Ultralight カードや FeliCa カードをエミュレートするために、リーダーをカードエミュレーションモードに設定するために使用されます。

注： *Lock* バイトは、エミュレートされた *MIFARE Ultralight* カードでサポートされていません。*UID* は、ユーザーが設定可能です。

Enter Card Emulation Mode コマンドフォーマット（8 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン		
Enter Card Emulation Mode	E0h	00h	00h	40h	03h	NFCMode	00h	00h

Enter Card Emulation Mode 応答フォーマット（8 バイト）

応答	CLA	INS	P1	P2	Le	データ出力		
結果	E1h	00h	00h	00h	03h	NFCMode	01h	01h

その中：

NFCMode 1 バイト。NFC デバイスモード

01h = MIFARE Ultralight カードエミュレーションモード

03h = FeliCa カードエミュレーションモード

06h = P2P イニシエータモード

他 = カードのリーダ/ライタモード

バイトナンバー	0	1	2	3	USB でバイトアドレスへのアクセス
シリアルナンバー	SN0	SN1	SN2	BCC0	Nil
シリアルナンバー	SN3	SN4	SN5	SN6	Nil
内部/ロック	BCC1	内部	Lock0	Lock1	Nil
データリーダー/ライター	Data0	Data1	Data2	Data3	0-3
データリーダー/ライター	Data4	Data5	Data6	Data7	4-7
データリーダー/ライター	Data8	Data9	Data10	Data11	8-11
データリーダー/ライター	Data12	Data13	Data14	Data15	12-15
データリーダー/ライター	Data16	Data17	Data18	Data19	16-19
データリーダー/ライター	Data20	Data21	Data22	Data23	20-23
データリーダー/ライター	Data24	Data25	Data26	Data27	24-27
データリーダー/ライター	Data28	Data29	Data30	Data31	28-31
データリーダー/ライター	Data32	Data33	Data34	Data35	32-35
データリーダー/ライター	Data36	Data37	Data38	Data39	36-39
データリーダー/ライター	Data40	Data41	Data42	Data43	40-43
データリーダー/ライター	Data44	Data45	Data46	Data47	44-47
データリーダー/ライター	Data48	Data49	Data50	Data51	48-51

アクセス可能な領域（52 バイト）

表5 ：MIFARE Ultralight カードのメモリマップ（52 バイト）

その中：

デフォルトのシリアルナンバー**[0-6]** {04h, 96h, 50h, 01h, F4h, 02h, 80h}

デフォルトのデータ**[0-3]** {E1h, 10h, 06h, 00h} //NFC Type 2 Tag

メモリ	1 データブロック（16 バイト）	USB でバイトアドレスへのアクセス
データリーダー/ライター	Block 0	0-15
データリーダー/ライター	Block 1	16-31
データリーダー/ライター	Block 2	32-47
データリーダー/ライター	Block 3	48-63
データリーダー/ライター	Block 4	64-79
データリーダー/ライター	Block 5	80-95
データリーダー/ライター	Block 6	96-111
データリーダー/ライター	Block 7	112-127
データリーダー/ライター	Block 8	128-143
データリーダー/ライター	Block 9	144-159

表6 ：FeliCa カードのメモリマップ（160 バイト）

その中：

デフォルト： Block 0 データ:{10h, 01h, 01h, 00h, 09h, 00h, 00h, 00h, 00h, 00h, 01h, 00h, 00h, 00h, 00h, 1Ch}

デフォルトブロック 0 のデータ NFC Type3 のタグ属性情報ブロック

注：

1. *FeliCa* カードエミュレーションのサポートは暗号化せずに読み取り/書き込み。
2. メーカーコードは（*0388*）に固定されています；*FeliCa* カード識別番号（*IDm*）は、ユーザが設定可能です。

5.5.2. カードエミュレーションのデータを読み取る（Read Card Emulation Data）（MIFARE Ultralight 若しくは FeliCa）

このコマンドはカードエミュレーションカードのデータを読み取るために使用されます。

Read Card Emulation Data コマンドフォーマット（9 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン			
Read Card Emulation Data	E0h	00h	00h	60h	04h	00h	NFCMode	StartOffset	長さ

Read Card Emulation Data 応答フォーマット（データ 5 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	長さ	読み取られたデータ

その中：

NFCMode 1 バイト。NFC デバイスモード

01h = MIFARE Ultralight カードエミュレーションモード

03h = FeliCa カードエミュレーションモード

StartOffset 1 バイト。読み始めるアドレス

長さ 1 バイト。読み取られていないバイト数。

読み取りデータ 読み取られたデータ

5.5.3. カードエミュレーションのデータを書き込む（Write Card Emulation Data）（MIFARE Ultralight 若しくは FeliCa）

このコマンドは、エミュレートされたカードへの書き込みに使用されています。

Write Card Emulation Data コマンドフォーマット

コマンド	CLA	INS	P1	P2	Lc	データイン				
Write Card Emulation Data	E0h	00h	00h	60h	長さ + 4	01h	NFCMode	StartOffset	長さ	書き込まれていないデータ

Write Card Emulation Data 応答フォーマット（8 バイト）

応答	CLA	INS	P1	P2	Le	データ出力		
結果	E1h	00h	00h	00h	03h	長さ	90h	00h

その中：

NFCMode 1 バイト。NFC デバイスモード

01h = MIFARE Ultralight カードエミュレーションモード

03h = FeliCa カードエミュレーションモード

StartOffset 1 バイト。読み始めるアドレス

長さ 1 バイト。書き込みするデータの長さ。

書き込みデータ 書き込まれるバイナリデータ

5.5.4. カードのエミュレーション時の MIFARE Ultralight の UID を設定する（Set Card Emulation of MIFARE Ultralight UID）

このコマンドは、エミュレートされた MIFARE Ultralight の UID を設定するために使用されています。

Set Card Emulation MIFARE Ultralight UID コマンドフォーマット（12 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Set Card Emulation Mifare Ultralight UID	E0h	00h	00h	61h	07h	7 バイトの UID

Set Card Emulation MIFARE Ultralight UID 応答フォーマット（7 バイト）

応答	CLA	INS	P1	P2	Le	データ出力	
結果	E1h	00h	00h	00h	02h	90h	00h

その中：

UID　　7 バイト。7 バイトの MIFARE カードの UID

5.5.5. カードエミュレーションの FeliCa カードの IDM を設定する（Set Card Emulation FeliCa IDm）

このコマンドはカードエミュレーションの FeliCa カード上で、6 バイトの FeliCa カードフラグを設定するために使用されます。

Set Card Emulation FeliCa Card Identification number コマンドフォーマット(11 バイト)

コマンド	CLA	INS	P1	P2	Lc	データイン
Set Card Emulation FeliCa IDm	E0h	00h	00h	64h	06h	IDm

Set Card Emulation FeliCa Card Identification number 応答フォーマット(11 バイト)

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	06h	IDm

その中：

IDm 6 バイト

5.5.6. NFC カードエミュレーションのデータロックを設定する（Set Card Emulation Lock Data in NFC）

このコマンドは NFC 通信中、カードエミュレーションのデータをロックするために使用されます。データがロックされると、NFC で書き換えることができません。

Set Card Emulation Lock Data in NFC コマンドフォーマット（6 バイト）

コマンド	CLA	INS	P1	P2	Lc	データイン
Set Card Emulation Lock Data	E0h	00h	00h	65h	01h	ロック

Set Card Emulation lock data in NFC 応答フォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	ロック

その中：

ロック　1 バイト NFC 通信で書き換えされないように、データを保護します。

ロックのパラメーター

ビット	パラメーター	説明	オプション
7-2	保留	保留	
1	Felica ロックを有効にする	データは NFC 通信で書き換えられません。USB ダイレクトコマンドでデータを変更できます。	0：ロックを無効にする 1：ロックを有効にする
0	MIFARE Ultralight ロックを有効にする		0：ロックを無効にする 1：ロックを有効にする

5.6. ACR122U 互換性のあるコマンド

5.6.1. 二色の LED 制御（Bi-color LED Control）

このコマンドは二色の LED の状態を制御するために使用されます。

Bi-color LED and Buzzer Control コマンドフォーマット（9 バイト）

コマンド	CLA	INS	P1	P2	Lc	データフィールド（4 バイト）
二色 LED 制御	FFh	00h	40h	LED 状態制御	04h	点滅周期の制御

P2　　LED 状態制御

二色の LED 制御のフォーマット（1 バイト）

コマンド	アイテム	説明
Bit 0	赤の LED の最後の状態	1 = ON；0 = OFF
Bit 1	緑の LED の最後の状態	1 = ON；0 = OFF
Bit 2	赤の LED のマスク	1 = 状態を更新する 0 = 変化なし
Bit 3	緑の LED のマスク	1 = 状態を更新する 0 = 変化なし
Bit 4	初期の赤の LED の点滅状態	1 = ON；0 = OFF
Bit 5	初期の緑の LED の点滅状態	1 = ON；0 = OFF
Bit 6	赤い LED の点滅マスク	1 = 点滅 0 = 点滅なし
Bit 7	緑の LED の点滅マスク	1 = 点滅 0 = 点滅なし

データ　　点滅周期の制御

二色 LED の点滅周期制御フォーマット（４バイト）

バイト 0	バイト 1	バイト 2	バイト 3
T1 周期 初期の LED の点滅状態 （単位 = 100 ms）	T2 周期 点滅状態を切り替える （単位 = 100 ms）	繰り返し回数	00h

データ出力　　SW1 SW2。リーダーから返された状態コード

状態コード

結果	SW1	SW2	意味
成功	90h	現在の LED 状態	操作が成功に完了しました。
エラー	63h	00h	操作が失敗しました。

LED の現在の状態（1 バイト）

状態	アイテム	説明
Bit 0	現在の赤い LED	1 = ON；0 = OFF
Bit 1	現在の緑の LED	1 = ON；0 = OFF
Bits 2 – 7	RFU	RFU

提示：

1. *LED* 状態の操作は *LED* 点滅操作の後に実行されます。
2. ***LED* 状態のマスク**が有効になっていない場合、***LED* 状態**は変更しません。
3. ***LED* 状態のマスク**が有効になっている場合、***LED*** は点滅しません。。また、繰り返し回数は *0* より大きくなければなりません。
4. *T1* および *T2* 周期のパラメータは、*LED* の点滅周期とブザーターンオン周期を制御するために使用される。例えば：もし *T1=1*，*T2=1*，デューティサイクル= *50%*。

   注：デューティサイクル= *T1/*（*T1* + *T2*）。

5.6.2. ファームウェアのバージョンを入手する（Get Firmware Version）

このコマンドはリーダーのファームウェアのバージョンを取得する時に使われます。

Get Firmware Version のコマンドフォーマット（５バイト）

コマンド	CLA	INS	P1	P2	Le
Get Firmware	FFh	00h	48h	00h	00h

Get Firmware Version の応答フォーマット（X バイト）

応答	データ出力
結果	ファームウェアのバージョン番号

例：

応答 = 41 43 52 31 32 35 32 55 5F 56 32 30 32 2E 32h = ACR1252U_V100.1 (ASCII)

5.6.3. PICC 操作のパラメーターを取得する（Read the PICC Operating Parameter）

このコマンドはリーダーの PICC 操作のパラメーターを入手する時に使われます。

Get the PICC Operating Parameter フォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Get PICC Operating Parameter	FFh	00h	50h	00h	00h

Get the PICC Operating Parameter 応答フォーマット（2 バイト）

応答	データ出力	
結果	90h	PICC 操作パラメーター

PICC 操作パラメーター

ビット	パラメーター	説明	オプション
7	自動的に PICC ポーリング	PICC ポーリングを有効する	1 = 有効にする 0 = 無効にする
6	自動に ATS 生成する	ISO 14443-4 A タイプのタグを活性化するたびに ATS リクエストを送信します。	1 = 有効にする 0 = 無効にする
5	ポーリング間隔	連続した PICC のポーリング間の時間間隔を設定します。	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
3	FeliCa 212 Kbps		1 = 検出 0 = スキップ
2	Topaz		1 = 検出 0 = スキップ
1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
0	ISO 14443 A タイプ **注：***MIFARE* タグを検査するために、*ATS* の自動生成を無効にしなければなりません。		1 = 検出 0 = スキップ

5.6.4. PICC 操作のパラメーターを設定する（Set the PICC Operating Parameter）

このコマンドはリーダーの PICC 操作のパラメーターを設定する時に使われます。

Set PICC Operating Parameter コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Le
Set PICC Operating Parameter	FFh	00h	51h	PICC 操作パラメーター	00h

Set PICC Operating Parameter 応答フォーマット（2 バイト）

応答	データ出力	
結果	90h	PICC 操作パラメーター

PICC 操作パラメーター

ビット	パラメーター	説明	オプション
7	自動的に PICC ポーリング	PICC ポーリングを有効する	1 = 有効にする 0 = 無効にする
6	自動に ATS 生成する	ISO 14443-4 A タイプのタグを活性化するたびに ATS リクエストを送信します。	1 = 有効にする 0 = 無効にする
5	ポーリング間隔	連続した PICC のポーリング間の時間間隔を設定します。	1 = 250 ms 0 = 500 ms
4	FeliCa 424 Kbps	PICC のポーリング中に検出されるタグのタイプ	1 = 検出 0 = スキップ
3	FeliCa 212 Kbps		1 = 検出 0 = スキップ
2	Topaz		1 = 検出 0 = スキップ
1	ISO 14443 B タイプ		1 = 検出 0 = スキップ
0	ISO 14443 A タイプ **注：** *MIFARE* タグを検査するために、*ATS* の自動生成を無効しなければなりません。		1 = 検出 0 = スキップ

附录A SNEP メッセージ

このコマンドのデータフォーマットを了解したい場合、“NFC Forum NFC Data Exchange Format (NDEF) Specifications 1.0”を参照してください。

例：

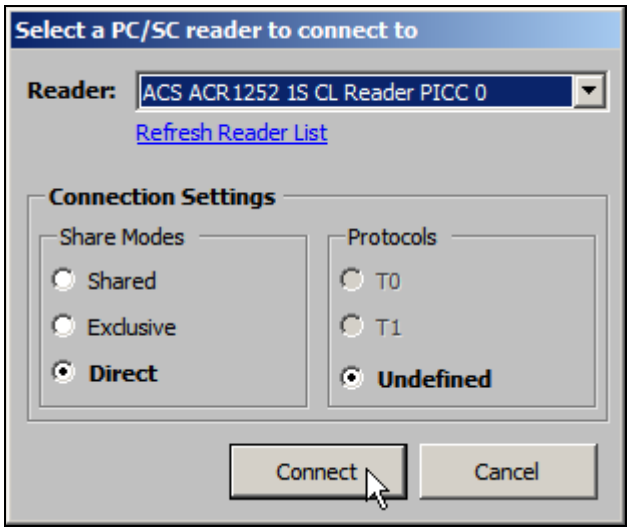
SNEP メッセージ = {D1 02 0F 53 70 D1 01 0B 55 01 61 63 73 2E 63 6F 6D 2E 68 6Bh}

オフセット	コンテンツ	長さ	説明
0	D1	1	NDEF ヘッダ。TNF = 01h、SR=1、MB=1、ME=1
1	02	1	レコード名の長さ （2 バイト）
2	0F	1	スマートポスターデータの長さ（15 バイト）
3	53 70 （“Sp”）	2	レコード名
5	D1	1	NDEF ヘッダ。TNF = 01h、SR=1、MB=1、ME=1
6	01	1	レコード名の長さ （1 バイト）
7	0B	1	URI ペイロードの長さ（11 バイト）
8	55 （“U”）	1	レコードタイプ“U”
9	01	1	略語“http://www.”
10	61 63 73 2E 63 6F 6D 2E 68 6B	10	URL 自体。“acs.com.hk”

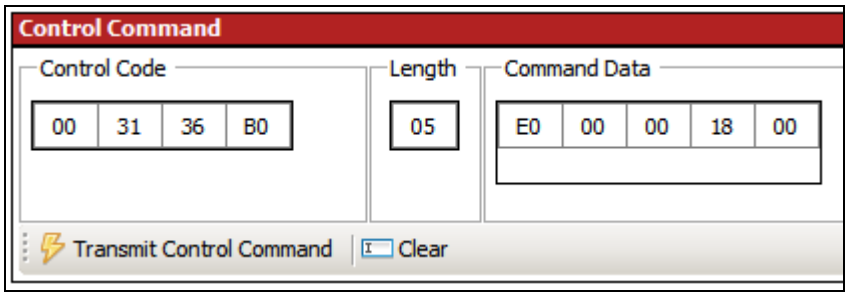
附录B 直接コマンド例

例：ACR1252U Reader Tool を使用してファームウェアのバージョンを取得します。

1. ACM1252U をパソコンに接続します。
2. ACR1252U Reader Tool 実行する。
3. ダイレクトモードで（Direct Mode）”リーダーを接続する。



4. Control Transmit タグページをクリックする。LengthData フィールドに“05”を入力する。
5. Command Data フィールドに、E0 00 00 18 00（APDU の Get Firmware Version コマンド）を入力します。



6. [Transmit Control Comman d]をクリックしてから、[応答データ]を確認します。

   例如：応答データー = E1 00 00 00 0F 41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31

   ファームウェアのバージョン番号（HEX） = 41 43 52 31 32 35 32 55 5F 56 31 30 30 2E 31

   ファームウェアのバージョン番号（ASCII) = ACR1252U_V100.1