Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1552U シリーズ

リファレンスマニュアル　V1.05

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

目次

図示一覧表

チャート一覧表

1.0. 紹介

ACR 1552 U シリーズ製品は ACR 1252 U シリーズの成功経験を継続し、13.56 Mhz 非接触技術に基づいて新たに開発した一連の USB NFC リーダライタです。このシリーズの非接触リーダライタは ISO 14443、ISO 15693 及び ISO 18092 規格に準拠した非接触スマートカードを読み書きでき、MIFARE® （T＝CL）、FeliCa、NFC タグ、SRI／SRIX、CTS、Innovatron、Picopas、Topaz などのカードにも対応できます。

ACR 1552 U シリーズ製品はカード読み書き、カードシミュレーション、キーボードシミュレーション、3 種類の NFC 操作モードをサポートできます。また、SAM カードスロットを内蔵しており、接触式と非接触式のアプリケーションの安全性を高めることができます。

これらのプラグアンドプレイ型 USB NFC デバイスは CCID と PC/SC 基準を満たし、多種のデバイスやアプリケーションと相互に操作でき、スマートポスターなどの非伝統型マーケティングと広告アプリケーションの理想的な選択です。

ACR 1552 シリーズ製品はキーボードシミュレーションなどの新機能を提供しており、機能強力で、コストパフォーマンスが高い「1 機で多く使う」型デバイスです。各種のスマートカードアプリケーションに極めて大きな柔軟性と利便性を提供することができます。

このリファレンスマニュアルに適応する ACR 1552 U シリーズ製品には下記のものを含めています。

- ACR1552U-M*、USB NFC カードリーダーIV
- ACM1252U-Y、アンテナ分離型の USB NFC リーダーモジュール
- ACM1252U-Z、小型 NFC リーダーモジュール

注：*SAM* カードスロットやブザーなど、製品の機能がモデルによって異なります。

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：
    - 読み書き速度が 26 kbps（ISO 15693 カード）および 848 kbps（ISO 14443 カード）
    - 内蔵アンテナは非接触タグの読み書きに使用され、スマートカードの読み取り距離は 70 mm に達する（タグの種類によって異なる）
    - ISO 15693 カードサポート
    - ISO 14443 パート 4 のタイプ A・B のカードと MIFARE シリーズサポート
    - 衝突防止機能内蔵
    - 拡張の APDU サポート（最大 64 KB）
  - SAM インターフェース：
    - 1つ SAMカードスロット
    - ISO7816クラスA のSAMカードサポート
- 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
  - CT- API サポート（ PC/SC 上のレイヤーによるパッケージ）
- 内蔵されている周辺機器：
  - 2×ユーザー制御可能な LED（青と緑）
  - ユーザー制御可能なブザー
- USB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[1]
- 以下の規格に準拠：
  - ISO 14443
  - ISO 15693
  - ISO 7816
  - PC/SC
  - CCID
  - CE
  - UKCA
  - FCC
  - RoHS
  - REACH
  - Microsoft® WHQL

[1] *ACS □ □ □ □ □ Android □ □ □ □ □ □ □*

3.0. ACR1552U の構造

3.1. リーダーの機能アーキテクチャ図

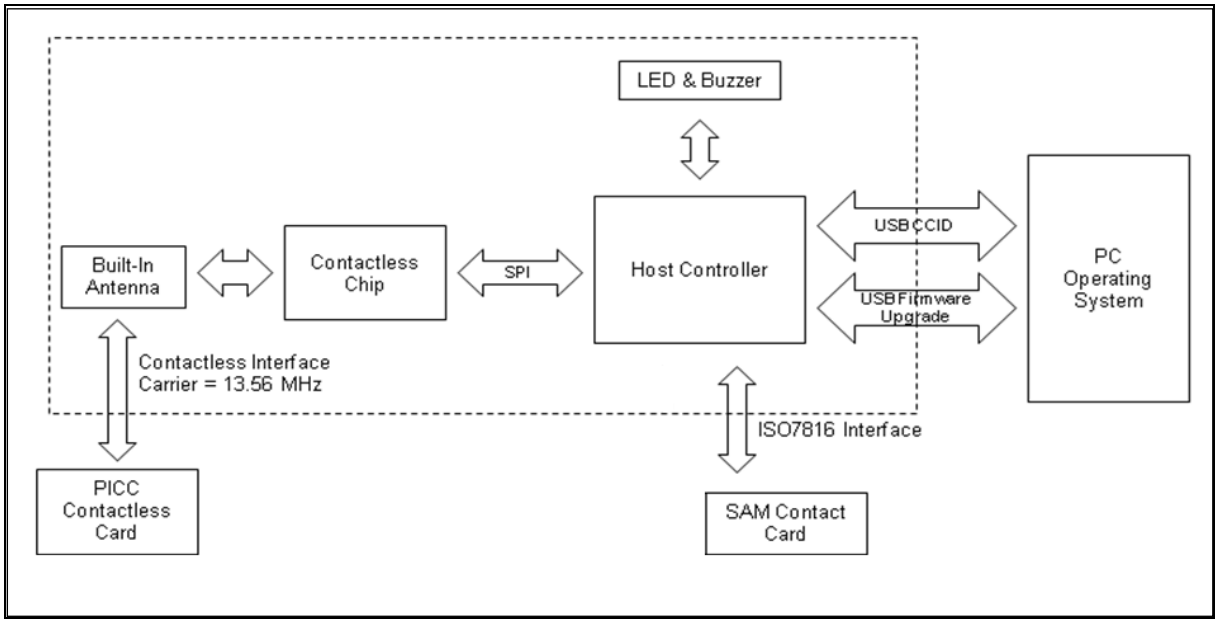


図1：ACR1552U リーダーの機能ブロック図

3.2. PC/SC ドライバと PICC および SAM 間の通信

ACR1552U が CCID プロトコルを使用して PC データ通信します。PICC と SAM 間の通信は PC/SC 規格に準拠しています。

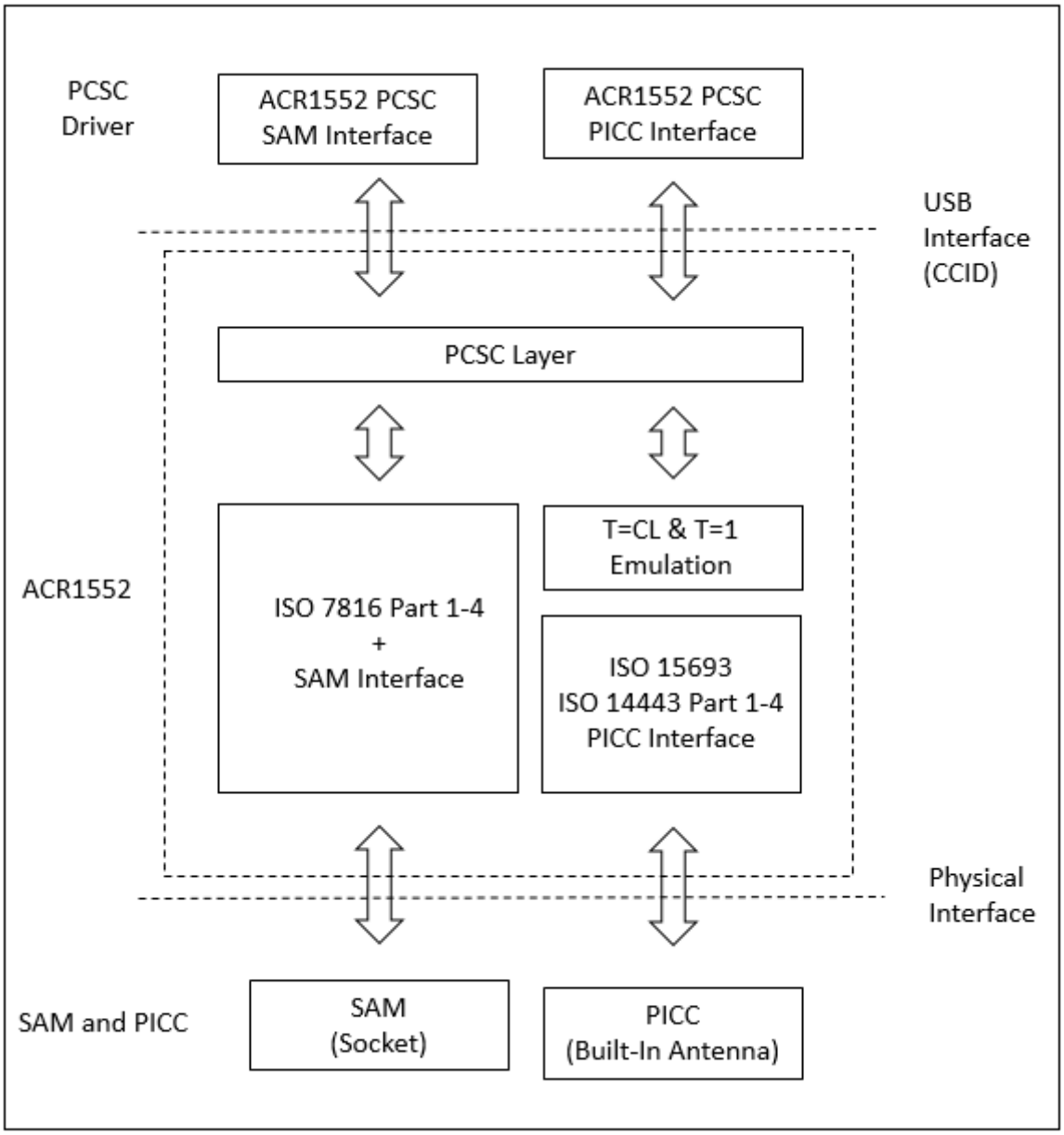


図2：ACR1552U の構造

4.0. ハードウェアのデザイン

4.1. USB

ACR1552U が USB 規格に準拠した USB インターフェースを介して PC と接続します。

4.1.1. 通信パラメ—タ—

ACR 1552 U は USB 2.0 規格の要求に従って USB インタフェースを通じてコンピュータと接続を確立します。USB 全速モードをサポートし、レートが 12 Mbps になります。

ピン	信号	機能
1	V_{BUS}	カードに+5 V の電源を供給
2	D-	ACR 1552 U と PC との間で差動信号でデータを伝送する
3	D+	ACR 1552 U と PC との間で差動信号でデータを伝送する
4	GND	電圧レベル参考

表1： USB インターフェース配線

注：*ACR1552U* は *USB* インターフェースを介して、正常に動作させるには、まずドライバプログラムをインストールする必要があります。

4.1.2. エンドポイント

ACR1552U が下記のエンドポイントを介して、ホスト PC と通信します：

Control Endpoint – 設置と制御

Bulk-OUT – ホスト PC から ACR1552U にコマンドを送信する（パケットサイズ 64 バイト）

Bulk-IN – ACR1552U からホスト PC に応答を送信する（パケットサイズ 64 バイト）

Interrupt-IN – ACR1552U からホスト PC にカード状態のメッセージを送信する（パケットサイズ 8 バイト）

4.2. 非接触スマートカードインターフェース

ACR 1552 U と非接触カードとの間のインタフェースは ISO 14443 規格に準拠しており、ACR 1552 U の実用的な機能を強化するためにいくつかの制限または向上が行われています。

4.2.1. 搬送波周波数

ACR1552U の搬送波周波数は 13.56MHz です。

4.2.2. ポーリング

ACR1552U が作業場に入る非接触カードを自動的に検出します。この機能が ISO 14443-4 の A と B タイプのカード、MIFARE カードをサポートします。

4.3. ユーザーインターフェース

4.3.1. ブザーと LED

ACR1552Uには、非接触インタフェースの状態を示すLEDインジケータと単音ブザーが搭載されています。青色LEDはPICCの状態を示すために使用されます。

リーダー状態	ブザー	青色 LED（PICC）
1. リーダー挿入	1回鳴らす	● >> ● >> ●
2. スタンバイ（非接触カードポーリング、PICCカードは存在しない）	OFF	●
3. スタンバイ（ポーリングない）	OFF	OFF
4. 非接触式カードかざす	1回鳴らす	●
5. 非接触式カード存在	OFF	●
6. 非接触式カード取り外す	OFF	●
7. 非接触式カード通信中	OFF	速く点滅する

表2：ブザーとLEDインジケータ

5.0. ハードウェアデザイン

5.1. PCSC API

このセクションでは、アプリケーションプログラミング用の PCSC API について説明します。これらの API の詳細について、Microsoft MSDN ライブラリまたは PCSC ワークグループ仕様サイト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。

参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379479%28v=vs.85%29.aspxC:\My Files\ACS\Products\ACS Card & Reader\CSCR\ACR1552\REF\quot;http:\msdn.microsoft.com\en-us\library\windows\desktop\aa379479(v=vs.85).aspx&quot

他の PCSC アクションを実行する前に、この関数を実行してください。

```
#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
   }
}
```

例：

5.1.2. SCardListReaders

SCardListReaders 関数を使用して、システム内に指定されたカードリーダグループコレクション内のカードリーダリストを取得します（重複項目排除）。

呼び出し側がカードリーダーグループのリストを提供して、関数が指定されたグループ内のカードリーダー名のリストを返す。認識できないグループの名前は無視されます。この関数が現在システムに利用できるグループ中のリーダーだけ返されます。

参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379793%28v=vs.85%29.aspx

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 関数は特定のエクスプローラコンテキストを使用して、アプリケーションと特定のカードリーダーに含まれるスマートカードとの間の接続を確立します。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。

参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379473%28v=vs.85%29.aspxC:\My Files\ACS\Products\ACS Card & Reader\CSCR\ACR1552\REF\quot;http:\msdn.microsoft.com\en-

```
#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 ACR1552 1S 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
         }
      }
   }
```

us\library\windows\desktop\aa379473(v=vs.85).aspx&quot

例：

5.1.4. SCardControl

SCardControl 関数は、SCardConnect 関数を正常に呼び出した後、SCardDisconnect 関数を正常に呼び出す前にいつでも呼び出すことができるカードリーダーの直接制御を提供します。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。

参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspxC:\My Files\ACS\Products\ACS Card & Reader\CSCR\ACR1552\REF\quot;http:\msdn.microsoft.com\en-us\library\windows\desktop\aa379474(v=vs.85).aspx&quot

注：この *API* を使用して送信する。**直接（*Escape*）命令**。

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 ACR1552 1S 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 関数はサービス請求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。

参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379804%28v=vs.85%29.aspxC:\My Files\ACS\Products\ACS Card & Reader\CSCR\ACR1552\REF\quot;http:\msdn.microsoft.com\en-us\library\windows\desktop\aa379804(v=vs.85).aspx&quot

注： この *API* で *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 ***PICC 的 PCSC 私有 APDU（带专有扩展）*** と ***PICC 的专属私有 APDU*** を送信します。

例：

```
#define SCARD_SCOPE_USER      0

SCARDCONTEXT         hContext;         // Resource manager context
SCARDHANDLE       hCard;            // Card context handle
unsigned long     dwActProtocol;    // Established active protocol
int               retCode;
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 ACR1552 1S 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。参照してください：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379475%28v=vs.85%29.aspxC:\My Files\ACS\Products\ACS Card & Reader\CSCR\ACR1552\REF\quot;http:\msdn.microsoft.com\en-us\library\windows\desktop\aa379475(v=vs.85).aspx&quot

この関数が PCSC 操作をエンドするために使用されます。

例：

```
#define SCARD_SCOPE_USER 0

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

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

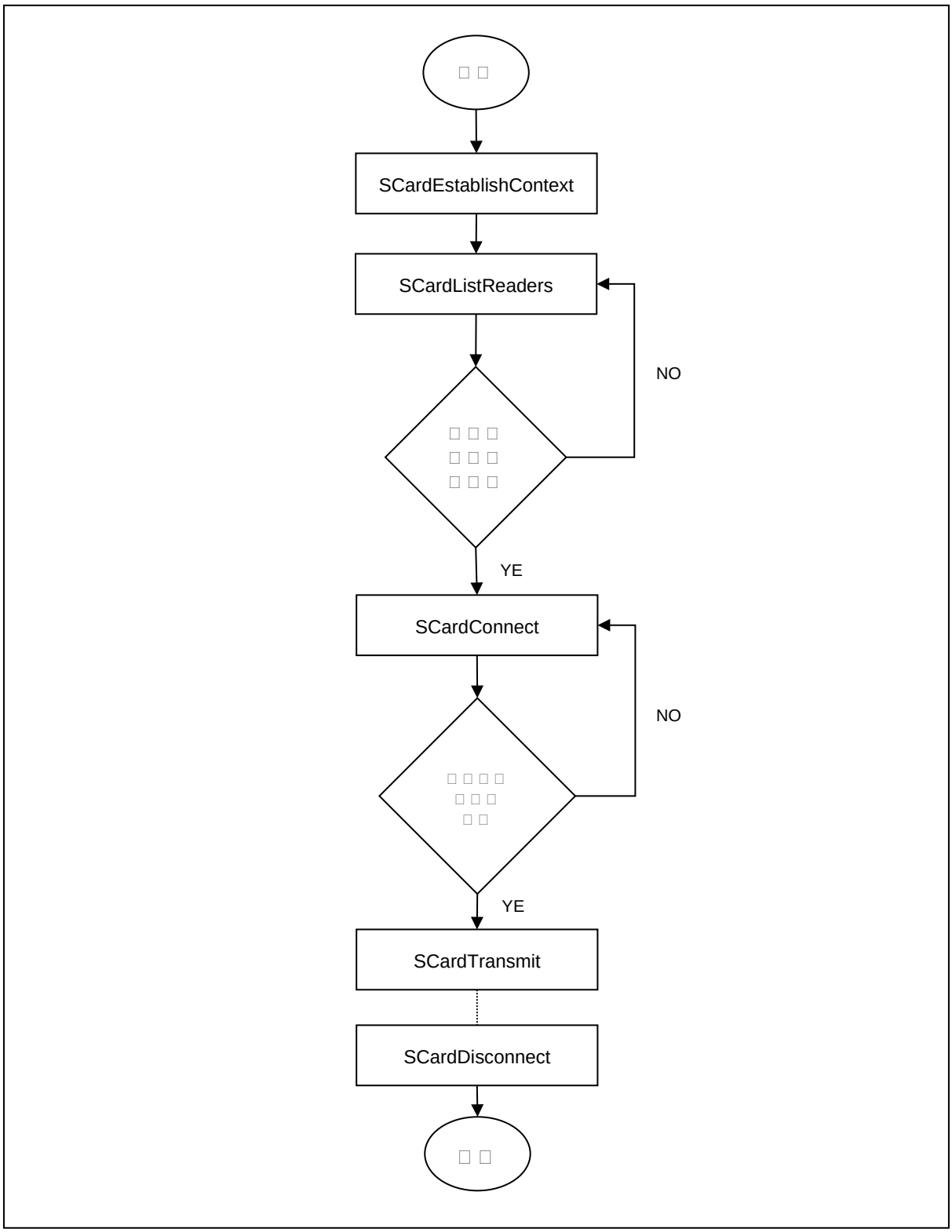


図3：ACR1552U の APDU 流れ

5.1.8. 直接なコマンドの流れ

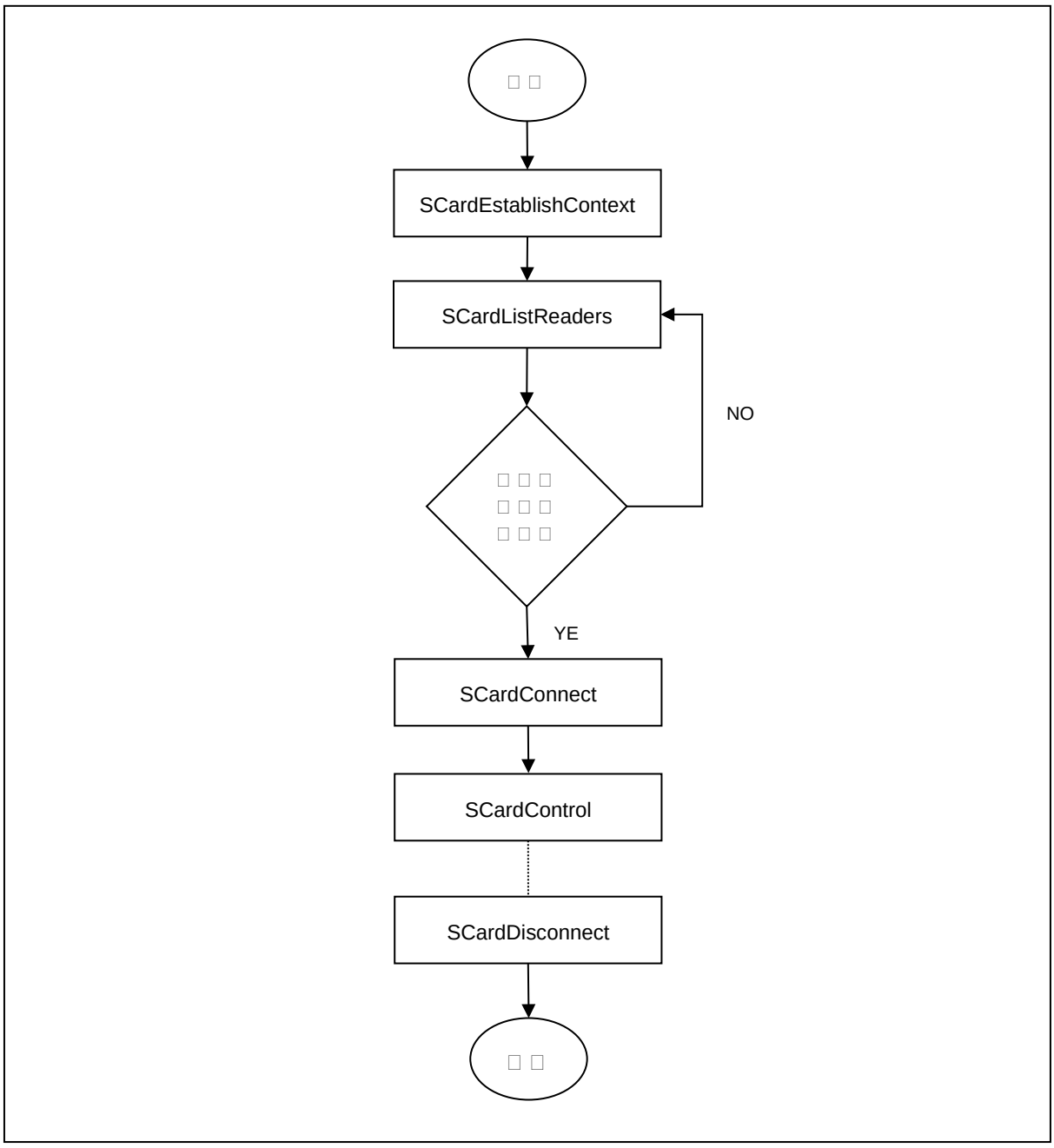


図4：ACR1552U 直接コマンドの流れ

5.2. 接触式スマートカードプロトコル

5.2.1. ACOS6-SAM カードコマンド

このセクションでは、SAM 専用のコマンドを説明します。CCID ホストが PCSC API 中の SCardTransmit（）に対応する CCID メッセージ PC _ to _ RDR _ XfrBlock を使用して、カードリーダーにカードアドミニストレーションコマンドまたは APDU を送信することができます。

注：*ACOS 6-SAM* コマンドのすべての情報とアプリケーションシーンについては、*ACS* 営業担当者に問い合わせて、*ACOS 6-SAM* リファレンスマニュアルを請求してください。

5.2.1.1. キー生成（Generate Key）

このコマンドは、顧客カードのシリアル番号などの偏差データを用いて分散鍵を生成し、ACOS 3/6 または他のカードに導入することで、顧客カード発行の目的を満たす。

APDU	説明	
CLA	80h	
INS	88h	
P1	00h 01h 02h	8 バイトのキーを生成する 16 バイトのキーを生成する 24 バイトのキーを生成する
P2	導出鍵を生成するためのマスター鍵の鍵インデックス	
P3	08h	
データ	データ入力：	

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 が正しくない、08h でなければならない
6A 83h	指定された鍵レコードが EF 2 には見つかりません
69 81h	無効な EF2（レコードサイズ、ファイルタイプなど）
6A 88h	EF2 が見つかりません
62 83h	現在の DF はロックされている; EF2 がロックされている
69 83h	使用カウンタはゼロです。

SW1 SW2	説明
69 82h	セキュリティ条件が満たされていない
6A 87h	指定されたマスターキーは 3-DES 暗号化サポートしない
61 08h	コマンドが完了、結果を得るために GET REPONSE を送信する

5.2.1.2. キーデータ分散化（またはロード）（Diversify (or Load) Key Data））

このコマンドは、鍵分散と鍵ロードによって SAM カードに暗号化操作を実行する準備をさせます。コマンドデータ入力として、シリアル番号と CBC 初期ベクトルを取ります。

APDU	説明								
CLA	80h								
INS	72h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	0	0	1	パスワード(Sc)
	-	0	0	0	0	0	1	0	アカウントキー(K_{ACCT})
	-	0	0	0	0	0	1	1	エンドキー
	-	0	0	0	0	1	0	0	カードキー
	-	0	0	0	0	1	0	1	キーを一括に暗号化する（分散しない）
	-	0	0	0	0	1	1	0	初期ベクトル
	0	-	-	-	-	-	-	-	16 バイトのキー
	1	-	-	-	-	-	-	-	24 バイトのキー
P2	マスターキーの索引： Bit7: 1 =現在 EF2 中のローカルキー； 0 =グローバルキー EF2 Bit6-Bit5: 00b - RFU Bit4-Bit0: キーインデックス								
P3	P1 = 1～4 の場合、P3 = 8/16（アルゴリズムが AES の場合、P3 = 8/16） P1 = 5 の場合、P3 = 0 P1 = 6 の場合、 P3 = 8（マスターキーの Algo は DES / 3DES / 3KDES） P3 = 16（マスターキーの Algo は AES）								
データ	P1 = 1-4 の場合、クライアントカードのシリアル番号（algo が AES の場合、データはクライアントカードのシリアル番号またはクライアントカードのシリアル番号に「0000000000000000」が付加されます） P1 = 5 の場合、コマンドデータはありません。 P1 = 6 の場合、DES / 3DES / 3KDES / AES CBC の初期ベクトル。								

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	間違った P1、P1 は 1～6 でなければなりません
67 00h	間違った P3、P3 は 8（または 0）
62 83h	現在の DF がロックされているか、または EF2 がロックされている
69 82h	セキュリティ条件が満たされていない
6A 88h	EF2 が見つかりません
6A 83h	EF2 に指定マスターキーが見つかりません
69 81h	無効な EF2（FDB、MRL など、一貫性がない）
6A 87h	指定された KEY 認証できない
69 83h	参照されたキーがロックされています
90 00h	ターゲットキーが生成され、SAM メモリに準備されている

5.2.1.3. 暗号化（Encrypt）

このコマンドは、DES または 3DES を使用して次のいずれかの方法でデータを暗号化します。

1. ACOS3 / 6、DESFire®、DESFire® EV1 または MIFARE Plus カードとの相互認証手順によって作成されたセッションキー。
2. 分散化した鍵（パスワード）。
3. キーを一括に暗号化する。
4. セッション鍵で分散化したパスワードを暗号化します。
5. 非 SM コマンドを指定して、ACOS3 セキュア・メッセージング・コマンドを作成します。

APDU	説明
CLA	80h
INS	74h

APDU	説明								
	b7	b6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	0	0	-	ECB モード
	-	0	0	0	0	0	1	-	CBC モード
	-	0	0	0	0	1	0	-	小売 MAC モード
	-	0	0	0	0	1	1	-	MAC モード
	-	0	0	0	1	0	0	-	ACOS3 SM コマンド用意する
	-	1	0	0	1	0	1	-	MIFARE DESFire 暗号化
P1	-	1	0	0	1	1	0	-	MIFARE DESFire EV1 暗号化
	-	0	0	0	1	1	1	-	CMAC
	-	0	1	0	0	0	0		MIFARE Plus コマンド
	-	0	1	0	0	0	1		MIFARE Plus 応答
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	-	-	-	-	-	-	-	-	他のすべての値 - RFU
P2	P 2 は、Load Key 機能を使用して SAM セットに分散された鍵を表します。 1–プロセスキーKs でデータを暗号化する 2–分散化キー*Sc* でデータを暗号化する 3–一括暗号化キーでデータを暗号化する 0 – ENC を返す (*Sc*, *Ks*) P1.b3 = 1 または b5 = 1 の場合、P2 は 1 でなければなりません P2 = 0h の場合、P1 は 0 または 1 のいずれかになります								
P3	P3 < 128 P1 のビット 3 が 1 に等しくなく、P1 のビット 5 が 1 に等しくない場合 - P2 = 1-3 の場合、8（DES / 3DES / 3KDES）の倍数または 16 バイト（AES）の 128 バイト - P2 = 0 の場合、0								

APDU	説明
データ	プレーンテキスト P2 b6 = 1 の場合、データのフォーマットが次のようになります： • プレインテキストデータの長さ • DESFire カードのコマンドとヘッダーの長さ • DESFire カードのコマンドとヘッダー • プレーンテキスト P1 = A1h、の場合、暗号化は Plus コマンド用です • MFP コマンドが値操作コマンドの場合、データのフォーマットは Command Code（1 バイト）＋BlockNum（2/4 バイト）＋Value（4 バイト）でなければなりません。 • MFP Command が Proximity Check の場合、データフォーマットは Command Code（1 バイト）+ PPS1（1 バイト）でなければなりません。 • MFP コマンドが読み取りの場合、データフォーマットは Command code(1 バイト）+BlockNum（2 バイト） • MFP コマンドが書き込みである場合、データフォーマットは Command code（1 バイト）+ BlockNum（2 バイト）+plaintext P1 = A3h、 • ICC によって返されたデータ（SC コードは含まず、RMAC が存在する場合は RMAC を含まない）

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 正しくない
6A 83h	ACOS ターゲットキーが準備されていません（キーを生成するために Diversify を使用してください）
61 XX	暗号化が行われ、結果を得るために GET RESPONSE を使用する

5.2.1.4. 復号化（Decrypt）

このコマンドは、DES または 3DES または AES を使用して次のいずれかを使用してデータを復号化するために使用されます。

1. ACOS3/6、MIFARE DESFire、MIFARE DESFire EV1またはMIFARE Plusカードとの相互認証手順によって作成されたセッションキー。
2. 多様化した鍵（パスワード）。
3. Aバルク暗号化キー。
4. セッション鍵で多様化したパスワードを解読化します。
5. ACOS3の安全なメッセージングの応答を確認し、解読する。

ACOS3 の安全なメッセージングの応答を確認し、解読する。

APDU	説明								
CLA	80h								
INS	76h								
P1	b7	B6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	0	0	-	ECB モード
	-	0	0	0	0	0	1	-	CBC モード
	-	0	0	0	1	0	0	-	ACOS3 の安全なメッセージングの応答を確認し、解読する。
	-	1	0	0	1	0	1	-	MIFARE DESFire で復号化
	-	1	0	0	1	1	0	-	MIFARE DESFire EV1 で復号化
	-	0	1	0	0	1	0	-	MIFARE Plus 復号化
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES
	1	-	-	-	-	-	-	1	AES
	0	0	0	0	-	-	-	-	他のすべての値 - RFU
P2	P2 は、ロードキー機能を使用して SAM セットのキーとして導出されます。 1–プロセスキーKs でデータを復号化する 2–分散化キー*Sc* でデータを復号化する 3–一括復号化キーでデータを復号化する 0 – DEC を返す (*Sc*, *Ks*)								

APDU	説明
P3	P3 < 128 P1 = A5h の場合、P3 = 16/32/48 P1 のビット 3 が 1 に等しくない場合 - P2 = 1-3 の場合、8（DES / 3DES / 3KDES）の倍数または 16 バイト（AES）の 128 バイト - P2 = 0 の場合、0
データ	暗号文 P1 = A5h の場合、データが暗号化されたテキスト P2 b6 = 1 の場合、データのフォーマットが次のようになります： • Plain テキストデータの長さ（不明の場合は 00 を使用） • DESFire カードのコマンドとヘッダーの長さ • DESFire カードのコマンドとヘッダー • 暗号化されたテキスト

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 正しくない
6A 83h	ACOS ターゲットキーが準備されていません（キーを生成するために Diversify を使用してください）
61 XX	解読化が行われ、結果を得るために GET RESPONSE を使用する

5.2.1.5. 認証準備（Prepare Authentication)

このコマンドは、ACOS 3/6、MIFARE Ultralight C/MIFARE DESFire/MIFARE Plus カードに対して SAM カード（端末として）を認証するために使用されます。

APDU	説明
CLA	80h
INS	78h
P1	00h – 3DES 01h – DES 02h – 3KDES (MIFARE DESFire EV1/ACOS3) 03h – AES (MIFARE DESFire EV1/MIFARE Plus/ACOS3)

APDU	説明
	80h – 3DES (MIFARE DESFire 認証のみ) 81h – DES (MIFARE DESFire 認証のみ) 他 – RFU
P2	0h - ACOS3 / 6 が認証を返すことを確認する 01h - MIFARE 超軽量 C / DESFire 認証（多様化）ターミナルキー 05h - MIFARE Ultralight C / DESFire は一括暗号鍵で認証します 02h - MIFARE Plus 認証 SL1 から SL3 の最初の認証 03h - MIFARE Plus 認証。SL1 から SL2 への認証。 04h - MIFARE Plus 認証。SL2 から SL3 への認証に続きます。
P3	8 – (P1 = 00h, 01h, 02h, 80h, 81h) 16 – (P1 = 03h)
データ	カードチャレンジデータ

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 が正しくない、必ず 08h
6A 83h	ACOS キー（KT または KC）が準備できていない（Diversify でこのキーを生成してください）
69 82h	セキュリティ条件が満たされていない
61 10h	コマンドが完了、結果を得るために GET REPONSE を送信する

5.2.1.6. 照合認証（Verify Authentication）

このコマンドは、ACOS 3/6、MIFARE Ultralight C、MIFARE DESFire/MIFARE DESFire EV1 または MIFARE Plus カードが端末に対する正当性を検証するために使用され、内部でプロセスキーKs も生成されます。

APDU	説明
CLA	80h
INS	7Ah
P1	00h – 3DES (P2 = 0) 01h – DES (P2 = 0) 02h – 3KDES (P2 = 0，ACOS3)

APDU	説明
	03h – AES (P2 = 0，ACOS3) 他 – RFU
P2	00h - ACOS3 / 6 が認証を返すことを確認する 01h - MIFARE Ultralight C®/ DESFire®/ DESFire® EV1 認証状況を確認する 02h - MIFARE Plus の認証状況を確認する
P3	08h - （P2 = 0、P2 = 1、セッションキーは DES / 3DES） 16h - （P2 = 1、、セッションキーは 3KDES / AES 使用） 16h - （P2 = 02、MIFARE Plus がリターンデータ ek（RndA '）） 32h - （P2 = 02、MIFARE Plus がリターンデータ ek（TI + PICCcap2 + PCDcap2））
データ	ACOS 3/6：DES (K_S, RND_T) MIFARE DESFire/ DESFire EV1 がリターンデー：ek(RndA') MIFARE Plus がリターンデータ ek（RndA '）または ek（TI + PICCcap2 + PCDcap2）

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 が正しくない、必ず 08h
6A 83h	ACOS-SAM セッションキーまたは RNDT は準備ができていません。PREPARE AUTHENTICATION を使用してこれらのキーを作成します。
69 82h	データ正しくない
90 00h	データ正しくて ACOS 相互認証成功

5.2.1.7. ACOS クエリアカウントチェック（Verify ACOS Inquire Account）

このコマンドは、ACOS3 / 6 カードの Inquire Account purse コマンドを確認するために使用します。それは、ACOS3 / 6 によって返された MAC チェックサムが、SAM の多様化された鍵で正しいことを検証する。

APDU	説明								
CLA	80h								
INS	7Ch								
P1	b7	b6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	-	0	-	ACOS INQ_AUT 無効
	-	0	0	0	0	-	1	-	ACOS INQ_AUT 有効
	-	0	0	0	0	0	-	-	ACOS INQ_ACC_MAC 無効
	-	0	0	0	0	1	-	-	ACOS INQ_ACC_MAC 無効
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES（ACOS3 のみ）
	1	-	-	-	-	-	-	1	AES（ACOS3 のみ）
P2	0h								
P3	1Dh								
データ	クライアントの ACOS カードの INQUIRE ACCOUNT によって返されるデータブロック。以下を参照してください。								

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 正しくない
6A 83h	ACOS キー K_S または K_{ACCT} は準備ができていません。K_{ACCT} 生成するには DIVERSIFY コマンドを使用します。該当する場合は、「Prepare Authentication」を介して K_S を生成します。
6F 00h	**データブロックの MAC が正しくありません**
90 00h	**データブロックの MAC が正しい**

5.2.1.8. ACOS アカウント取引準備（Prepare ACOS Account Transaction）

ACOS3 / 6 クレジット/デビットコマンドを作成するには、ACOS3 / 6 が検証するために MAC を計算する必要があります。

APDU	説明								
CLA	80h								
INS	7Eh								
P1	b7	b6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT 無効
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT 有効
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES（ACOS3 のみ）
	1	-	-	-	-	-	-	1	AES（ACOS3 のみ）
P2	E2h: クレジット E6h: デビット								
P3	0Dh								
データ	データブロック								

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 が正しくない、必ず 0Dh
6A 83h	ACOS キーK_S または K_{ACCT} は準備ができていません。K_{ACCT} 生成するには DIVERSIFY コマンドを使用します。該当する場合は、「Prepare Authentication」を介して K_S を生成します。
61 0Bh	コマンドが完了、結果を得るために GET REPONSE を送信する

5.2.1.9. デビット証明書認証（Verify Debit Certificate）

ACOS3 / 6 の場合、DEBIT コマンドに P1 = 1 が指定されている場合は、借方の証明書が返されます。デビット証明書は、ACOS3 の応答をこのコマンドの結果と比較することによって確認できます。

APDU	説明								
CLA	80h								
INS	70h								
P1	b7	b6	b5	b4	b3	b2	b1	b0	説明
	-	0	0	0	0	0	0	-	ACOS TRNS_AUT 無効
	-	0	0	0	0	0	1	-	ACOS TRNS_AUT 有効
	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES（ACOS3 のみ）
	1	-	-	-	-	-	-	1	AES（ACOS3 のみ）
P2	0h								
P3	14h								
データ	データブロック								

特定の応答ステータスバイト

SW1 SW2	説明
69 86h	DF 未選択
6A 86h	P1 もしくは P2 無効
67 00h	P3 が正しくない、14h でなければならない
6A 83h	ACOS キーK_S または K_{ACCT} は準備ができていません。K_{ACCT} 生成するには DIVERSIFY コマンドを使用します。該当する場合は、「Prepare Authentication」を介して K_S を生成します。
69 82h	セキュリティ条件が満たされていない
6F 00h	DEBIT CERTIFICATE 無効
90 00h	成功，DEBIT CERTIFICATE 有効

5.2.1.10. キー取得（Get Key）

鍵を取るコマンドにより、鍵を現在の SAM の鍵ファイル（SFI=02 h）から別の ACOS 6/ACOS 6-SAM カードに安全に注入することができ、このプロセスは、鍵分散を介さなくても実現できます。これを使用すると、注入される鍵が暗号化およびメッセージ認証コードによって保護されます。

またこのコマンドが分散化により、現在の SAM のキーファイル（SFI = 02h）から ACOS7/10、MIFARE DESFire、MIFARE DESFire EV1 または MIFARE Plus カードに安全に注入することができ。これを使用すると、注入される鍵が暗号化およびメッセージ認証コードによって保護されます。

カードヘッダーブロック（ACOS6-SAM リファレンスマニュアルのセクション 3.2）の特殊機能フラグ（キー注入専用フラグ）のビット 7 が設定され、キーファイルが有効になっている場合、キーのロードまたは変更には Get キーを使用する必要があります。Bit 7 が設定されると、鍵ファイルがアクティブになると、Read Record コマンドの使用はいずれの場合も無効になります。

このコマンドを実行する前に、**相互認証**の相互認証プロセス（ACOS6-SAM リファレンスマニュアルのセクション 5.3）または MIFARE Plus/MIFARE DESFire 相互認証プロセスで、ターゲットカードとのセッション鍵がすでに確立されています。

注：*GET KEY* コマンドはキーデータのみを取得できます。

APDU	説明			
CLA	80h			
INS	CAh			
P1	ACOS カードがキーを取得、リセットする用			
	00h	応答データは MSAM のキーです		
	01h	応答データは 16 バイトの Diversify キーです		
	02h	応答データは 24 バイトの Diversify キーです		
	03h	応答データは MIFARE Plus カードの Change Key コマンドです		
	DESFire カードのキー取得変更キー、DESFire / DESFire EV1 Change キーの応答データ			
		カードタイプ	キー番号の認証とキー番号の変更*	キー長さ
	80h	MIFARE DESFire	MIFARE®DESFire の中では異なる	16 バイト
	81h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードの中では異なる	16 バイト
	82h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードの中では異なる	24 バイト
	88h	MIFARE DESFire	MIFARE DESFire の中では同じ	16 バイト
	89h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードの中では同じ	16 バイト
	8Ah	MIFARE DESFire EV1	MIFARE DESFire EV1 カードの中では同じ	24 バイト
P2	SAM のキーID（変更のための新しいキー）			

APDU	説明
P3	P1 = 00h，P3 = 08h P1 = 02h，P3 = 10h P1 = 03h，P3 = 0h P1 = 80/81/82/88/89/8Ah：P3 = 0Bh
データ	P1 = 00h の場合、コマンドデータは RND_{Target} P1 = 01 / 02h の場合、コマンドデータは RND_{Target} +ターゲットカードのシリアル（またはバッチ）番号です P1 = 03h の場合 P1 = 03h - ターゲットカードのシリアル番号（8 バイト） - ライトコマンド（A0 または A1）（1 バイト） - BNr（2 バイト） P1 = 80/81/82/88/89/8Ah： - ターゲットカードのシリアル番号（8 バイト） - 元の鍵 ID（元の鍵が格納されている SAM カードの鍵、00 = DESFire のデフォルト鍵 - カード） - 鍵番号（DESFire カード鍵番号） - 鍵バージョン（DESFire カード鍵バージョン、使用しない場合、値= 00）

*このリストは、リストされているカードが違う Change Key と Authenticate Key を持っているか、または 2 つの鍵が同じ値を使用しているかを示しています。

特定の応答ステータスバイト

SW1 SW2	説明
69 85h	SAM セッションキーが準備完了していません
62 83h	現在の DF がブロックされているか、またはターゲット EF がブロックされている
69 86h	DF 未選択
69 81h	キーファイルのファイルタイプが間違っています。内部線形変数ファイル
69 82h	ターゲットファイルのヘッダブロックのチェックサムが正しくないか、セキュリティ条件が満たされていません
6A 86h	P1 もしくは P2 無効
67 00h	P3 正しくない
6A 83h	ターゲットキーが準備されていないか、キーの長さが 16 未満です
61 1Ch	コマンド成功、GET RESPONS で結果を取得

5.3. 非接触スマートカード プロトコル

5.3.1. 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 PCSC ドライバに送されます。

5.3.1.1. ATR フォーマット（ISO14443-3-3 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	高いニブル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 は歴史的なバイトの数です（HistByte 0 - HistByte N-1)
2	80h	TD1	高いニブル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	高いニブル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 ~ 3+N	80h	T1	カテゴリインジケータバイトは、80のステータスイ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

バイト	数値	標記	説明
	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カードの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)

基準 (SS) = 03h(ISO 14443A、3パート)

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

5.3.1.2. ATR フォーマット（ISO14443-4-3 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>高いニブル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 は歴史的なバイトの数です（HistByte 0 - HistByte N-1)</td></tr>
<tr><td>2</td><td>80h</td><td>TD1</td><td>高いニブル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>高いニブル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 ~ 3+N</td><td>XX</td><td>T1</td><td rowspan="2">歴史的なバイト：

ISO 14443-A：
ATS応答の歴史的なバイト。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>高いニブル=ATTRIBコマンドのMBLI；下位ニブル（RFU）=0</td></tr>
</table></td></tr>
<tr><td>XX</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

注釈：*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}

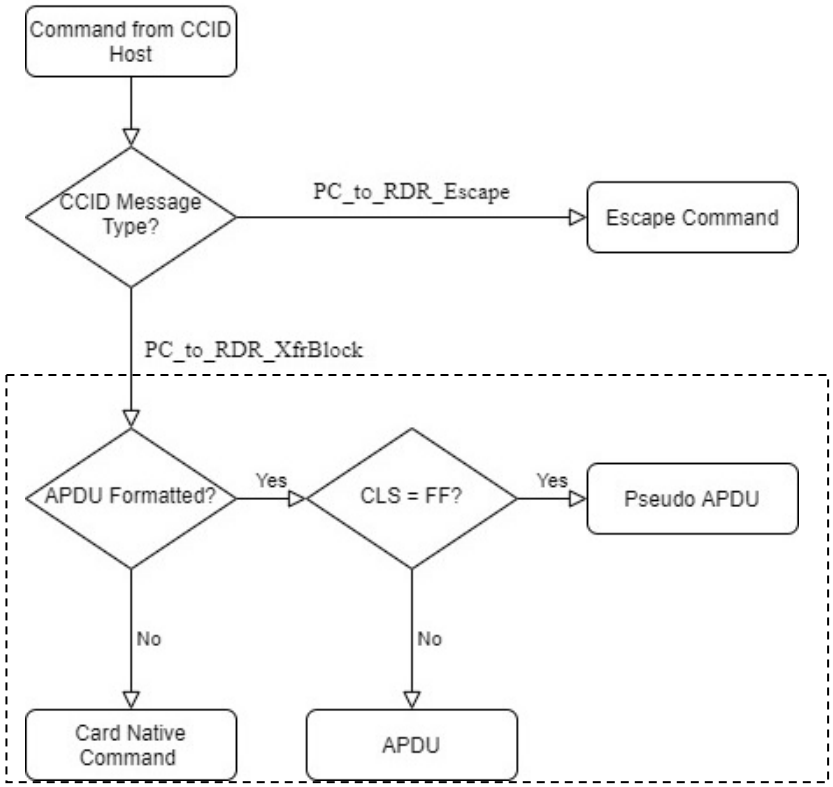
例2：

EZ-LinkのATR = {3B 88 80 01 1C 2D 94 11 F7 71 85 00 BEh}
ATQBのアプリケーションデータ = 1C 2D 94 11h
ATQBからのプロトコル情報 = F7 71 85h

ATTRIB の MBLI =00h

5.3.2. APDU、私有 APDU およびカード固有コマンド

ユーザは、PC _ to _ RDR _ XfrBlock メッセージを介して、APDU、Private APDU（Pseudo APDU）、およびカード固有コマンド（Card Native Command）をリーダライタに送信することができます。コマンドの処理が完了すると、リーダライタは RDR _ to _ PC _ DataBlock メッセージを介してレスポンスを返します。



CCID ホストが PCSC API 中の SCardTransmit（）に対応する CCID メッセージ PC _ to _ RDR _ XfrBlock を使用して、カードリーダーにカードアドミニストレーションコマンドまたは APDU を送信することができます。PICC では、カードが ISO 14443 第 4 部分プロトコルまたは Innovation プロトコルをサポートしている場合、リーダ/ライタはプロトコルフレームにコマンド/APDU をパッケージ化してカードに直接送信し、コマンド/APDU を解析しません。カードが両方のプロトコルをサポートしていない場合、CCID ホストにメッセージ「6 A 81」が返されます。

注：Microsoft Window はスマートカードプラグアンドプレイに対応しているため、Microsoft Window はカードが提示された時にカードに APDU 命令を送信することがあります。この操作により、カードをリセットしない限り、DESFire カードが ISO APDU モードになり、固有コマンドを受信できなくなります。通常、Microsoft Window はカードが無反応状態になってから 10 秒後にカードをリセットします（PC _ to _ RDR _ IccPowerOff 経由）。

5.3.3. PICC の PCSC 私有 APDU（独自の拡張機能付き）

非接触カードへの間接アクセスには、以下の私有（Pseudo）APDU を使用します。CCID ホストが PCSC API 中の SCardTransmit（）に対応する CCID メッセージ PC _ to _ RDR _ XfrBlock を使用して、カードリーダーにこれらの APDU を送信することができます。私有 APDU を受信すると、リーダライタは低レベルのコマンドを解読し、カード

に送信します。カードがこれらの低レベルコマンドを処理した後、リーダライタはカードレスポンスを収集し、レスポンスを作成して CCID ホストに返信する。

5.3.3.1. データ取得（Get Data）[FFCA…]

このコマンドは、シーケンス番号、プロトコルパラメータなど、アクティブ化中に取得したデータを読み込むために使用します。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Data	FFh	CAh	下記の表を確認ください		00h （全長）

コマンドパラメーター

P1	P2	意味
00h	00h	カードの UID/PUPI/SN を取得する
01h	00h	A タイプ 4 パートの ATS を取得
00h	02h	下記のカードタイプ関連データを取得し、転送順序： A タイプ：2 バイト ATQA/ATVA + 4/7/10 バイト UID + 1 バイト最後の SAK。 B タイプ：12 バイト ATQB
80h	00h	下記のカードタイプ関連データを取得し、転送順序： A タイプ：2 バイト ATQA/ATVA + 4/7/10 バイト UID + 1/2/3 バイト SAK。 B タイプ：12 バイト ATQB FeliCa：17 バイト ATQ (+ 6 バイト ATTR、アクティブ化されている場合) SRI：8 バイト UID + 1 バイトチップ ID。 ISO15693：1 バイト DSFID + 8 バイト UID CTS：4 バイト SN + 2 バイト ATQT Innovatron：4 バイト SN + 1 バイトタプアドレス。

応答

応答	データ出力		
結果	データ	SW1	SW2

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	6X XXh	失敗

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};

5.3.3.2. キーロード（Load Key）[FF 82 …]

このコマンドは、キーバッファ番号で指定された内部キーバッファにキーデータをロードするために使用します。鍵バッファは失いやすいストレージに属し、認証に使用されるコンテンツが含まれています。このコマンドはカードデータ転送をしません。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Load Authentication Keys	FFh	82h	00h	キーバッファ番号（0-1）	キー長さ	キー

キー長さ/データ

カードタイプ	キー長さ(Lc)	キーデータ（転送/保存順番）
MIFARE Standard MIFARE Plus SL1	06h	6 バイト Crypto1 Key A/B。
MIFARE Plus SL1 MIFARE Plus SL2	16h	6 バイト Crypto1 Key A/B + 16 バイト AES Key。
MIFARE Plus SL2	06h	6 バイト暗号化 Crypto1 Key A/B。
MIFARE UltraLightC MIFARE DESFire	10h	16 バイト 2K3DES Key。

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	6X XXh	失敗

例：

// 失いやすいキーのメモリに 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}

5.3.3.3. 認証（Authenticate）[FF 86 00 00 05 …]

このコマンドは、カードに認証プロセスを実行するために使用します。認証が成功すると、ユーザーは保護されたブロック/ページにアクセスできます。コマンドを送信する前に、ユーザーはLoad Keyコマンドを使用して正しいキーデータを指定した**密钥缓冲区号**バッファにロードする必要があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Authenticate	FFh	86h	00h	00h	05h	下記の表を確認ください

コマンドデータ

バイト 0	バイト 1	バイト 2	バイト 3	バイト 4
01h	00h (RFU)	アドレス	キーのタイプ	キーバッファ番号

アドレスとキーのタイプ

カードタイプ	アドレス	キーのタイプ
MIFARE Standard MIFARE Plus SL1 MIFARE Plus SL2	00h~FFh: ブロック 0~255	60h：Crypto1 Key A 61h：Crypto1 Key B
MIFARE UltraLightC	00h (RFU)	80h：2K3DES
MIFARE DESFire	00h~0Eh：DESFire キー番号 0~14	0Ah: 2K3DES

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	6X XXh	失敗

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 つのブロック，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
..	..	..

1 KB

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 つのブロック，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14	38h – 0Ah	3Bh
セクター15	3Ch – 3Eh	3Fh

表3：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 つのブロック，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	2 KB
セクター1	04h ~ 06h	07h	
..			
..			
セクター30	78h ~ 7Ah	7Bh	
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 つのブロック，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	2 KB
セクター33	90h ~ 9Eh	9Fh	
..			
..			
セクター38	E0h ~ EEh	EFh	
セクター39	F0h ~ FEh	FFh	

表4：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	内部	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 □
□ □
□ □ □
64 □ □

表5：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ます。認証はいりません。

5.3.3.4. バイナリブロック読み取り（Read Binary Blocks）[FF B0…]

このコマンドは、指定されたブロック/ページアドレスの場所から指定されたバイトのデータを PICC から読み出すために使用します。カードの種類によっては、このコマンドを呼び出す前に、ユーザーはまず認証を行い、これらのブロック/ページへのアクセス権を取得する必要がある場合があります。

コマンド：

コマンド	CLA	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	モードとアドレス		更新していないバイト

P1/P2（モードとアドレス）

カードタイプ	P1[7:4] モード	P1[3:0] + P2[7:0] 開始アドレス（MSB が前）
MIFARE Standard MIFARE Plus SL1 MIFARE Plus SL2	00h：テールブロックをスキップ 08h：テールブロックを含む	000h~0FFh：ブロック 0~255
MIFARE UltraLight MIFARE UltraLightC	00h（保留）	000h~02Fh：ページ 0~47
SRIX4K/SRT512	00h（保留）	000h~07Fh：ブロック 0~127 0FFh：システムゾーン
PicoPass	00h（保留）	000h~0FFh：ブロック 0~255
Topaz/NFC Type-1 タブ	00h（保留）	000h~7FFh：バイトアドレス

Le（読み取り待ちのバイト数）

タイプ	バイト 0	バイト 1	バイト 2
短い	00h：256 バイトを読み取る 01h~FFh：1~255 バイトを読み取る	--	
長い	00h	0000h：65536 バイトを読み取る 0001h~~FFFFh：1~65535 バイトを読み取る	

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	6X XXh	失敗

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

5.3.3.5. バイナリブロック更新（Update Binary Blocks）[FF D6 …]

このコマンドは、指定されたブロック/ページアドレスの場所から指定されたバイトのデータを PICC に書き込むために使用します。カードの種類によっては、このコマンドを呼び出す前に、ユーザーはまず認証を行い、これらのブロック/ページへのアクセス権を取得する必要がある場合があります。

カード内のブロック/ページへのデータの書き込みは、カードのセキュリティ設定（例えば MIFARE カードの末尾ブロック）を変更する可能性があるので、特に注意してください。誤ったデータを書き込むか、操作が失敗すると、カードがロックされる可能性があります。カードロックされるリスクを軽減するために、セキュリティブロック/ページに関わる場合には、1 つの APDU コマンドを使用して複数のブロック/ページにデータを書き込むことはお勧めしません。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Update Binary Blocks	FFh	D6h	モードとアドレス		書き込み待ちバイト	データバイト

P1/P2（モードとアドレス）と Write Size 一致（ブロック/ページ容量）

カードタイプ	P1[7:4] モード	P1[3:0] + P2[7:0] 開始アドレス（MSB が前）	ブロック/ページの容量（バイト）
MIFARE Standard MIFARE Plus SL1 MIFARE Plus SL2	0x0：テールブロックをスキップ 0x8：テールブロックを含む	000h~0FFh：ブロック 0~255	16
MIFARE UltraLight MIFARE UltraLightC	0x0（保留）	000h~02Fh：ページ 0~47	4
SRIX4K/SRT512	0x0（保留）	SRIX4K/SRT512	4
PicoPass	0x0（保留）	PicoPass	8

カードタイプ	P1[7:4] モード	P1[3:0] + P2[7:0] 開始アドレス（MSB が前）	ブロック/ページの容量 （バイト）
Topaz/NFC Type-1 タブ	0x0：消去を含む	000h~7FFh：バイトアドレス	1(アドレス 78h)また 8(その他)
	0x0：消去を含まない		

Lc（書き込み待ちバイト）

タイプ	バイト 0	バイト 1	バイト 2
短い	01h~FFh：1~255 バイトバイト書き込み待ち	--	
長い	00h	0001h~FFFFh：1~65535 バイトバイト書き込み待ち	

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	6X XXh	失敗

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}

5.3.3.6. PCSC 2.0 パート 3 サポートできる APDU コマンド（V2.02 及びその以降のバージョン）

PCSC 2.0 パート 3 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非タグへ渡し、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。

5.3.3.6.1. 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	コマンドデータイン
FFh	C2h	00h	機能	データ長さ	データ[データの長さ]

その中：機能 （1 バイト）：

00h = セッション管理
01h = 透明インタラクティブ
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 準拠以下の 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	長さ（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にエラーがある 02 = 二番目のデータオブジェクト内にエラーがある
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

エラーステータス	説明
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 実行エラー（IFD から応答ない）
XX 64 01h	データオブジェクト XX 実行エラー（ICC から応答ない）
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、エラーについての説明で、一番目のバイトの数値が誤ったデータオブジェクトの XX の番号を表します。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。

5.3.3.6.2. セッション管理（Manage Session）[FF C2 00 00 …]

このコマンドでユーザーがセッションを開始し、ポーリング機能を無効にして、後続の通信を行うことができます。通信が完了したら、ユーザーはすぐにセッションを終了する必要があります。

正しく使用されていない場合、リーダ/ライタがカードの存在を検出できず、論理的/物理的にリーダ/ライタ接続を切断しない限り、自動的にリカバリできないことに注意してください。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン	Le
Manage Session	FFh	C2h	00h	00h	Cmd データの長さ	Cmd TLV	--/00h

応答コード

応答データ	SW1 SW2	意味
--	90 00h	操作が成功に完了しました。
Rsp TLV	90 00h	Le = 0x00：Cmd TLV の一つが失敗したエラーの詳細については、Rsp TLV を参照してください。
--	6X XXh	Le = --：Cmd TLV の一つが失敗した

Cmd TLV

Cmd	意味
Start Session: 81 00h	セッションを開始し、ポーリングを無効にします。

Cmd	意味
RF Off: 83 00h	RF をオフにする
Timer: 5F 46 04h [TIME]	次の RF On/Off TLV 前のスリープ時間を設定する ［TIME］：4 バイトの値（MSB の前）、範囲は 1000～100000 us。実際のスリープ時間は、最も近い 1000 us に四捨五入されます。
RF On: 84 00h	RF をオンにする
End Session: 82 00h	セッションを終えて、ポーリングを有効にします。

Rsp TLV

Rsp	意味
TLV Error: C0 03 NN 6X XXh	NN 目のコマンド TLV エラー。

5.3.3.6.2.1. 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	長さ（1 バイト）	数値
81h	00h	-

5.3.3.6.2.2. 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	長さ（1 バイト）	数値
82h	00h	-

5.3.3.6.2.3. 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	長さ（1 バイト）	数値
83h	00h	-

5.3.3.6.2.4. RF データオブジェクトをオンにする（Turn O n 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	長さ（1 バイト）	数値
84h	00h	-

5.3.3.6.2.5. 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	長さ（1 バイト）	数値
5F 46h	04h	タイマー（4 バイト）

5.3.3.6.3. 透明交換（Transparent Exchange）[FF C2 00 01 …]

このコマンドにより、ユーザーはカードに任意のビットまたはバイトを送信/受信することができ、オプションで ISO 14443 パート 4 などの様々なリンクおよびトランスポート層、およびいくつかのリンク層冗長性（CRC およびパリティ）を構成することができます。ユーザは、任意のカード固有の生データをこのプライベート APDU に埋め込み、カードに送信することができます。

注意する必要があるのは、このコマンドがカードのサポートの内部処理プロセスに干渉する可能性があり、ドライバ/ファームウェアに通知せずにカードの状態を変更する可能性があり、ドライバ/ファームウェアを正常に戻すためにカードをリセットおよび/または削除する必要がある可能性があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン	Le
Transparent Exchange	FFh	C2h	00h	01h	Cmd データの長さ	Cmd TLV	00h

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	6X XXh	失敗

Cmd TLV

Cmd	意味
Transceive Flag: 90 02 [Flag] 00h	下記の Transceive TLV の Flag を設置する Flag[7:5]: RFU; 0 に設置する Flag[4]: ISO14443-4 を無効に設置する Flag[3]: パリティ照合処理の受信を禁止に設置する

Cmd	意味
	Flag[2]: パリティ照合処理の転送を禁止に設置する Flag[1]:CRC 処理の受信を禁止に設置する Flag[0]: CRC 処理の転送を禁止に設置する この TLV が見つからない場合は、前のコマンドで設定した Flag 値を使用します。 Flag 値を設定したことがない場合は、現在のプロトコル値を使用します。
Transmit Bit Frame: 91 01h [NumBit]	下記の Transceive TLV の Bit Fram を設置する。この TLV が見つからない場合、デフォルト値は 0 です。 NumBit[7:3]: RFU; 0 に設置する NumBit[2:0]:最後のバイトの有効ビットの数（0 はすべてのビットが有効であることを意味します）
Timer: 5F 46 04h [TIME]	下記の Transceive TLV のタイムアウト時間を設置する。 ［TIME］：4 バイトの値（MSB の前）、範囲は 1 us～1000000 us。実際のタイムアウト時間は、最も近い 302.07 x 20~15 us に四捨五入されます。 この TLV が見つからない場合は、以前設定した FWTI 値をタイムアウト時間として使用します。
Set FWTI: FF 6E 03 03 01h [FWTI]	Transceive の FWT/タイムアウトを設置する。以前に「FF C 2 h...」コマンドで FWTI を設定していない場合、デフォルト値は 0 になります。 FWTI: 0 ~ 15, FWT/タイムアウト = 302.07 x 2FWTI us
Transceive: 95h [Size] [Data]	SizeBER-TLV 長さフィールド中の符号化データのサイズ Data 転送待ちのデータ

Rsp TLV

Rsp	意味
Receive Bit framing: 92 01h [NumBit]	NumBit[7:3]: RFU; 0 に設置する。 NumBit[2:0]:最後のバイトの有効ビットの数（0 はすべてのビットが有効であることを意味します）
Response: 95h [Size] [Data]	Size: BER-TLV 長さフィールド中の符号化データのサイズ Data 受信データ。
Response Status: 96 02h [Status] 00h	Status [7:4]: RFU. Status[3]: フレーミングエラー Status[2]:パリティエラー。

Rsp	意味
	Status [1]: RFU. Status [0]: CRC エラー

5.3.3.6.3.1. 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	長さ（1 バイト）	数値		
		バイト 0		バイト 1
		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	00h
		1	0 – 受信したデータに対して CRC 検査を実施する 1 – 受信したデータに対して CRC 検査を実施しない	
		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）	
		5-7	RFU	

5.3.3.6.3.2. 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	長さ（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。

5.3.3.6.3.3. 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	長さ（1 バイト）	数値
95h	データ長さ	データ（N バイト）

5.3.3.6.3.4. 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	長さ（1 バイト）	数値
5F 46h	04h	タイマー（4 バイト）

5.3.3.6.3.5. ビットフレーミングデータオブジェクトを応答する（Response Bit Framing Data Object）

このコマンドは、応答中に受信した送信ビットフレームデータオブジェクトを提示するために使用します。

タグ	長さ（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

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。

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

このコマンドは、応答中に受信したデータの状態を提示するために使用します。

ステータスデータオブジェクトを応答する

タグ	長さ（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	RFU
		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	

5.3.3.6.3.7. データオブジェクトを応答する（Response Data Object）

このコマンドは、応答中に受信したデータの状態を提示するために使用します。

データフォーマットを応答する

タグ	長さ（1 バイト）	数値
97h	データ長さ	応答データ（N バイト）

5.3.3.6.4. プロトコル切り替え（Switch Protocol）[FF C2 00 02 …]

このコマンドを使用すると、ユーザーはプロトコルを切り替えて指定し、プロトコル・レベルとパラメータを選択できます。

注意する必要があるのは、このコマンドがカードのサポートの内部処理プロセスに干渉する可能性があり、ドライバ/ファームウェアに通知せずにカードの状態を変更する可能性があり、ドライバ/ファームウェアを正常に戻すためにカードをリセットおよび/または削除する必要がある可能性があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン	Le
Switch Protocol	FFh	C2h	00h	02h	Cmd データの長さ	Cmd TLV	00h

応答コード

応答データ	SW1 SW2	意味
Rsp TLV	90 00h	データ成功。
--	90 00h	成功
--	6X XXh	失敗

Cmd TLV

Cmd	意味
Set Baud: FF 6E 03 05 01h [Baud]	プロトコル切り替え時に使用するパート 4/レベルのボーレートを設定します。“FF C2h …”コマンドで[Baud]を設定していない場合、デフォルト値は 98h (106 kbps)になります。 Baud[7:2]: RFU,、100110b に設定する。 Baud[1:0]: 設定待ちのボーレート、00b (106 kbps), 01b (212 kbps), 10b (424 kbps), 11b (848 kbps)。
Switch Protocol: 8F 02h [RF] [Layer]	プロトコルを指定の RF と/もしくはレベルに切り替える [RF]: 00h：ISO14443A,　01h: ISO14443B 02h：ISO15693,　03h: FeliCa,　FFh: 現在の[RF]: その他 ： RFU [Layer]: 02h：レイヤ 2/セクション　03h: レイヤ 3/セクション 04h: レイヤ 4/セクション (A/B のみ適用) その他 ： RFU

Cmd	意味
	注意：レイヤ 2/セクションに切り替えた場合は、トランスペアレントセッション（ポーリング無効）状態である必要があります。

Rsp TLV

Rsp	意味
Response: 8Fh [Size] [Data]	Size: BER-TLV 長さフィールド中の符号のデータサイズ DataATR（パート 4 の場合）、最終 SAK（クラス A のパート 3 の場合）、または ATQB の PI（クラス B のパート 3 の場合）。

5.3.3.6.4.1. 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	長さ（1 バイト）	数値	
		バイト 0	バイト 1
8Fh	02h	00h – ISO/IEC14443 A タイプ 01h – ISO/IEC14443 B タイプ 02h – ISO15693 03h – FeliCa 他 – RFU	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

5.3.3.6.4.2. データオブジェクトを応答する（Response Data Object）

このコマンドは、応答中に受信したデータの状態を提示するために使用します。

データフォーマットを応答する

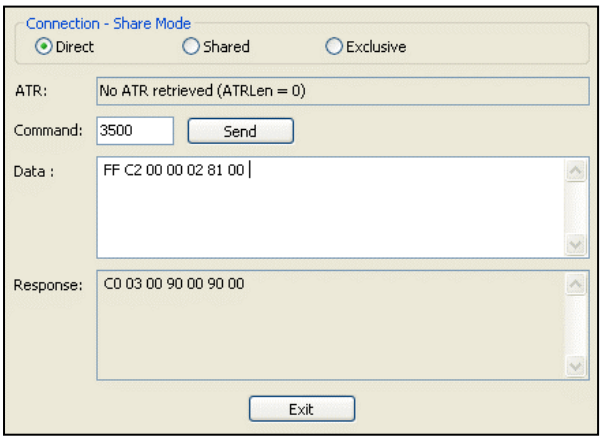
タグ	長さ（1 バイト）	数値
5F 51h	データ長さ	ATR
8Fh	データ長さ	最終 SAK（クラス A のパート 3 の場合）、または ATQB の PI（クラス B のパート 3 の場合）。

5.3.3.6.5. PCSC 2.0 パート 3 の例

1. 透明セッション開始する

   コマンド：**FF C2 00 00 02 81 00**

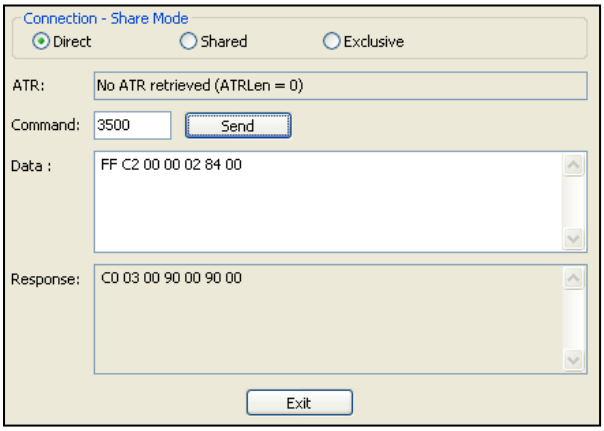
   応答：**C0 03 00 90 00 90 00**



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**

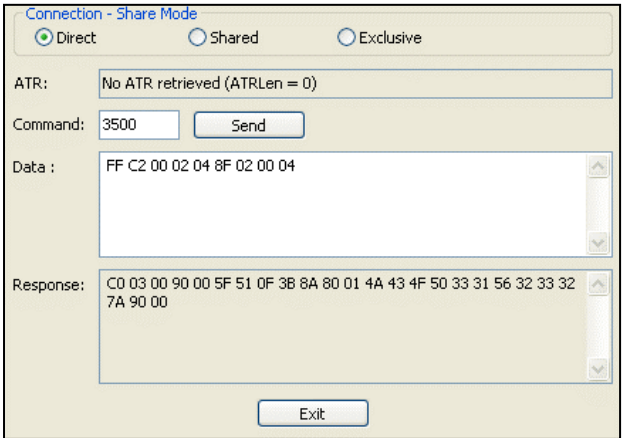


3. アクティベーISO 14443-4A 。

   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）

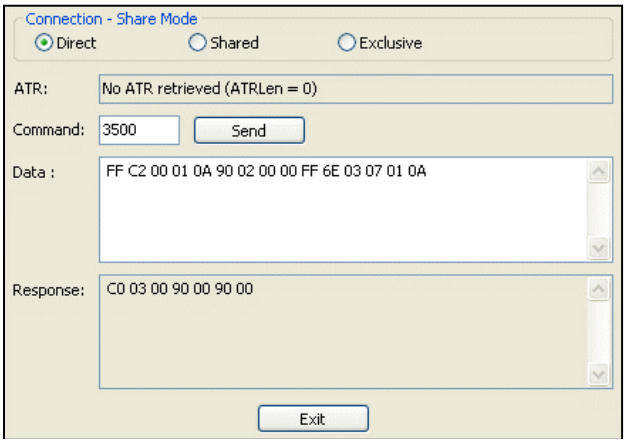
   C0 03 00 90 00 5F 51 [Len] [ATR] 90 00



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**

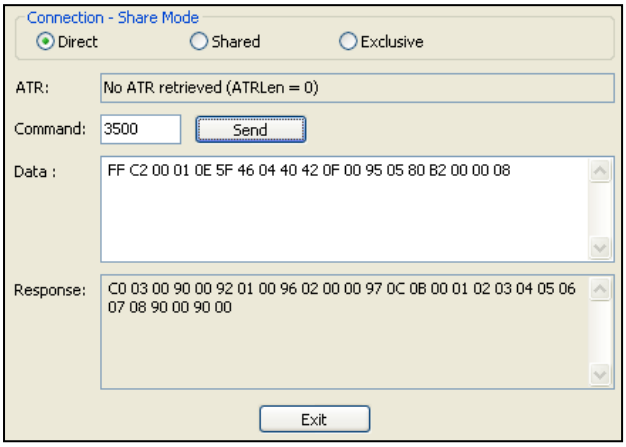
   応答：**C0 03 00 90 00 90 00**



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**

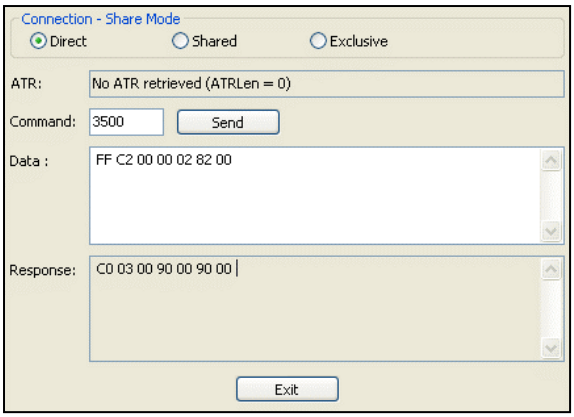
応答：**C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [カードの応答] 90 00**



6. 透明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**



5.3.4. PICC の専属の私有 APDU

以下の私有（Pseudo）APDU は、PCSC Pseudo APDU への追加として、非接触カードへの間接アクセスに使用されます。これらの APDU の内部処理プロセスは PCSC Pseudo APDU と同様です。

5.3.4.1. 値ブロック書き込み（Write Value Block）[FF D7 …]

このコマンドは、MIFARE 規格に準拠したカードのブロックに 4 バイトの値を書き込むために使用します。このコマンドを呼び出す前に、ユーザーはまず認証を行い、このブロックにへのアクセス権を取得する必要があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Write Value Block	FFh	D7h	00h	ブロック番号	05h	下記の表を確認ください

コマンドデータ

バイト 0	バイト 1	バイト 2	バイト 3	バイト 4
00h	4 バイト値（MSB の前）			

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

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	6X XXh	失敗

5.3.4.2. 値ブロック読み取り（Read Value Block） [FF B1 …]

このコマンドは、MIFARE 規格に準拠したカードの有効ブロックから 4 バイトの値を読み取るために使用します。このコマンドを呼び出す前に、ユーザーはまず認証を行い、このブロックにへのアクセス権を取得する必要があります。

コマンド

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

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

応答

応答データ	SW1 SW2	意味
4 バイト値（MSB の前）	90 00h	データ成功。
--	6X XXh	失敗

5.3.4.3. 値の減少/増加（Decrement/Increment Value）[FF D7 …]

このコマンドは、ソースブロックから 4 バイトの値を減少/増加させ、結果をターゲットブロックに格納するために使用します（カードは MIFARE 規格に準拠している必要があります）。結果を同じソースブロックに格納する場合は、ターゲットブロックの番号を 0 またはソースブロック番号に設定できます。このコマンドを呼び出す前に、ユーザーはまず認証を行い、ソースブロックとターゲットブロックにへのアクセス権を取得する必要があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Decrement/Increment Value	FFh	D7h	ターゲットブロック#	ソースブロック#	05h	下記の表を確認ください

コマンドデータ

バイト 0	バイト 1	バイト 2	バイト 3	バイト 4
01h	4 バイト追加値（MSB の前）			
02h	4 バイト減少値（MSB の前）			

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	6X XXh	失敗

5.3.4.4. 値ブロックコピー（Copy Value Block）[FF D7 …]

このコマンドは、ソースブロックから値をターゲットブロックにコピーために使用します（カードは MIFARE 規格に準拠している必要があります）。このコマンドを呼び出す前に、ユーザーはまず認証を行い、ソースブロックとターゲットブロックにへのアクセス権を取得する必要があります。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Copy Value Block	FFh	D7h	00	ソースブロック#	02h	下記の表を確認ください

コマンドデータ

バイト 0	バイト 1
03h	ターゲットブロック#

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	6X XXh	失敗

5.3.5. PCSC 準拠のタグをアクセスする（ISO 14443-4）

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。ACR1552U 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。ACR1552U が内部で 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。詳しい情報が **PICC 的 PCSC 私有 APDU（带专有扩展）**を参照してください。

ISO 7816-4 APDU フォーマット

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン	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：*ISO 14443-4 Type A* のタグに対して、*APDU"FF CA 01 00 00h"*によって *ATS* を入手できます。

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]

5.3.6. FeliCa タグのアクセス

FeliCa タグをアクセスするコマンドは、PCSC タグと MIFARE タグをアクセスするコマンドと違います。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	コマンドデータイン
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。

   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

5.3.8. ISO15693 タグのアクセス

この節では、ISO 15693 のオプションコマンドについて説明します。ファームウェア要件の説明は次のとおりです。

- ACR1552U-M　FW 1.03.00 及びその以降
- ACM1552U-Y　FW 2.03.00 及びその以降
- ACM1552U-Z　FW 2.03.00 及びその以降

5.3.8.1. 単一ブロックを読み取る（Read Single Block）

１つのデータブロックを ISO15693 から取り出すことに使われます。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ		Le
Read Single Block	FFh	FBh	00h	00h	02h	20h	ブロック番号	--/00h

その中：

ブロック番号　　1 バイト。

データブロック番号

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

//NXP ICODE SLI カードの 10 番目のブロックのデータを読み取る

コマンド：　= { FF FB 00 00 02 20 10 }

応答：　= { XX XX XX XX 90 00 }

5.3.8.2. 単一ブロックを書き込み（Write Single Block）

このコマンドは ISO15693 タブに 1 つのデータブロックを書き込むために使われます。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ			Le
Write Single Block	FFh	FBh	00h	00h	N+2h	21h	ブロック番号	データブロック	--/00h

その中：

ブロック番号 1 バイト。

データブロック番号

ブロックデータ N バイト。

データブロックに書き込み待ちのデータ

LC 1 バイト。

データブロック長さ + 2

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

//NXP ICODE SLI カードの 10 番目のブロックにデータを書き込む

コマンド： = { FF FB 00 00 06 21 10 11 12 13 14}

応答： = { 90 00 }

5.3.8.3. 複数のブロックを読み取る（Read Multiple Blocks）

複数のデータブロック複数 ISO15693 から取り出すことに使われます。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ			Le
Read Multiple Blocks	FFh	FBh	00h	00h	03h	23h	1 つ目のブロック番号	ブロックの数	--/00h

その中：

1 つ目のブロック番号　　1 バイト。

開始データブロック番号

ブロック数　　1 バイト。

読み取り待ちのデータ・ブロックのセキュリティ・ステータスの数。
要求中のブロック数は応答中にラベルが返されるブロック・セキュリティ・ステータス数より 1 つ少ない

ブロック数　= 要求中のブロック数 - 1

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

//0x10 から 0x12 まで、複数ブロックのセキュリティ状態を読み取ります。0x03 NXP ICODE SLI カードの連続なブロック。

コマンド：　= { FF FB 00 00 03 23 10 02 }

応答：　= { XX XX XX XX XX XX XX XX XX XX XX XX 90 00 }

5.3.8.4.　複数のブロックを書き込み（Write Multiple Blocks）

このコマンドは ISO15693 タブに複数のデータブロックを書き込むために使われます。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ				Le
Write Multiple Blocks	FFh	FBh	00h	00h	N+3h	24h	1 つ目のブロック番号	ブロックの数	データブロック	--/00h

その中：

1 つ目のブロック番号　　1 バイト。

開始データブロック番号

ブロック数 1 バイト。

読み取り待ちのデータ・ブロックのセキュリティ・ステータスの数。
要求中のブロック数は応答中にラベルが返されるブロック・セキュリティ・ステータス数より 1 つ少ない

ブロック数 = 要求中のブロック数 – 1

ブロックデータ N バイト。

データブロックに書き込み待ちのデータ

LC 1 バイト。

ブロックデータの長さ + 3

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	64 XXh	失敗 XX はタブが返すエラーコードです

5.3.8.5. ロックブロック（Lock Block）

LOCK BLOCK コマンドは、要求されたブロックを永続的にロックし、応答中に動作の成功状態を報告するために使用される。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ		Le
Lock Blocks	FFh	FBh	00h	00h	02h	22h	ブロック番号	--/00h

その中：

ブロック番号 1 バイト。

データブロック番号

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

コマンド： = { FF FB 00 00 02 22 10 }

応答： = { 90 00 }

5.3.8.6. システム情報読み取り（Get System Information）

タブシステムの情報を取得するために GET SYSTEM INFORMATION コマンドを使用します。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ	Le
Get System Information	FFh	FBh	00h	00h	01h	2Bh	--/00h

Get System Information の応答フォーマット

応答	データ出力							
結果	メッセージ表示	UID	DSFID	AFI	ストレージ容量	ICNO.	SW1	SW 2

その中：

メッセージ表示 – 1 バイト

ビット	数値	説明
Bit 0	0	DSFID ない
	1	DSFID ある
Bit 1	0	AFI しない
	1	AFI ある
Bit 2	0	ストレージ容量ない
	1	ストレージ容量ある
Bit 3	0	ICNO.ない
	1	ICNO.ある
Bit 4 ~7	0	RFU

UID -8 バイト

DSFID -1 バイト

AFI -1 バイト

メモリ容量-2 バイト

バイト	説明
MSB	ブロックの大きさ（バイトを単位で）-1 （実際のブロックの大きさ=ブロックの大きさ（バイトを単位で）+-1）
LSB	ブロックの数—1 （実際のブロック数　= ブロック数 + 1

IC -1 バイト

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

コマンド： = { FF FB 00 00 01 2B }

応答： = { XX XX XX XX XX XX XX XX XX XX XX XX XX 90 00}

5.3.8.7. 複数のブロックのセキュリティ状態を取得（Get Multiple Blocks Security Status）

ブロックのセキュリティ状態を取得するために GET MULTIPLE BLOCKS SECURITY STATUS コマンドを使用します。

コマンド：

コマンド	CLA	INS	P1	P2	LC	データ			Le
Get Multiple Blocks Security Status	FFh	FBh	00h	00h	03h	2Ch	1 つ目のブロック番号	ブロックの数	--/00h

その中：

1 つ目のブロック番号 1 バイト。

開始データブロック番号。

ブロック数 1 バイト。

読み取り待ちのデータ・ブロックのセキュリティ・ステータスの数。
要求中のブロック数は応答中にラベルが返されるブロック・セキュリティ・ステータス数より 1 つ少ない

ブロック数 = 要求中のブロック数 - 1

Get System Information の応答フォーマット

応答	データ出力		
結果	ブロックセキュリティ状態	SW1	SW2

その中：

ブロックセキュリティ状態 ブロックごとに 1 バイト

00h：ロックされていない

01h：ロックされた

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	64 XXh	失敗 XX はタブが返すエラーコードです

例：

//0x10 から 0x12 まで、複数ブロックのセキュリティ状態を読み取ります。0X03 連続なブロック。

コマンド：　= { FF FB 00 00 03 2C 10 02 }

応答：　= { XX XX XX 90 00 }

5.3.9. サポート PICC ATR

デフォルトでは、次の PICC タイプ/技術がサポートされています。リーダライタにカードをタッチすると、PC _ to _ RDR _ IccPowerOn コマンドは次の ATR を CCID ホストにに返します。

カードタイプ/技術	ATR
MIFARE Std 1k2	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 01 00 00 00 00 6A
MIFARE Std 4k2	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 02 00 00 00 00 69
MIFARE UltraLight2	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 03 00 00 00 00 68
MIFARE Plus SL1 2k2	デフォルト：MIFARE Std 1k と同じ 予備：3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 36 00 00 00 00 5D
MIFARE Plus SL1 4k2	デフォルト：MIFARE Std 4k と同じ 予備：3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 37 00 00 00 00 5C
MIFARE Plus SL2 2k	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 38 00 00 00 00 53
MIFARE Plus SL2 4k	3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 39 00 00 00 00 52
MIFARE UltraLight C2	デフォルト：3B 8F 80 01 80 4F 0C A0 00 00 03 06 03 00 3A 00 00 00 00 51 予備：MIFARE UltraLight と同じ
SmartMX、模擬 MIFARE Std 1k2	デフォルト：MIFARE Std 1k と同じ 予備：ISO14443 -4、Type A 同じ
SmartMX、模擬 MIFARE Std 4k[2]	デフォルト：MIFARE Std 4k と同じ 予備：ISO14443 -4、Type A 同じ
ISO14443 -4、Type A	3B 8n 80 01 T1 ..Tn Tck n = ATS 内履歴バイト数 T1 ..Tn = ATS 内履歴バイト数 Tck = xor 8n 80 01 T1 ..Tn

[2] □ □ □ ATR □ Set Operation Mode□ □ □ Param 2□ □ □ □ □ □ □ □ □ □ □

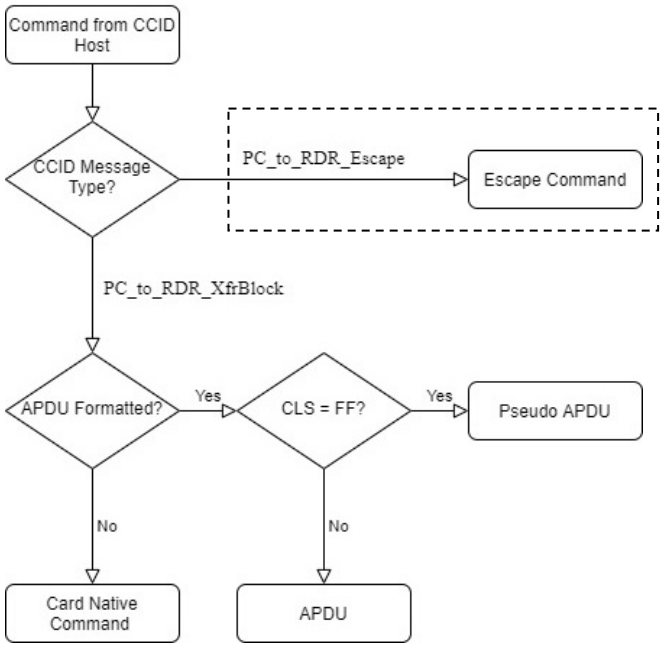
カードタイプ/技術	ATR
ISO14443 -4、Type B	3B 88 80 01 T1 ..T8 Tck T1 ..T4 = ATQB 中のアプリケーションデータ T5 ..T7 =ATQB 中のプロトコル情報 T8 = ATA 中の MBLI Tck = xor 88 80 01 T1 ..T8
FeliCa	3B 8F 80 01 80 4F 0C A0 00 00 03 06 11 00 3B 00 00 00 00 42
ISO15693-3 Generic	3B 8F 80 01 80 4F 0C A0 00 00 03 06 0B 00 00 00 00 00 00 63
Infineon My-D Vicinity (SRF55Vxxx)	3B 8F 80 01 80 4F 0C A0 00 00 03 06 0B 00 0E 00 00 00 00 6D
ST LRI	3B 8F 80 01 80 4F 0C A0 00 00 03 06 0B 00 13 00 00 00 00 70
NXP I-Code SLI	3B 8F 80 01 80 4F 0C A0 00 00 03 06 0B 00 14 00 00 00 00 77
NXP I-Code SLIX/SLIX2	3B 8F 80 01 80 4F 0C A0 00 00 03 06 0B 00 35 00 00 00 00 56
PicoPass 2K	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 17 00 00 00 00 79
PicoPass 2KS	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 18 00 00 00 00 76
PicoPass 16K	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 19 00 00 00 00 77
PicoPass 16KS	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1A 00 00 00 00 74
PicoPass 16K (8 x 2)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1B 00 00 00 00 75
PicoPass 16KS (8 x 2)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1C 00 00 00 00 72
PicoPass 32KS (16 + 16)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1D 00 00 00 00 73
PicoPass 32KS (16 + 8x2)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1E 00 00 00 00 70
PicoPass 32KS (8x2 + 16)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 1F 00 00 00 00 71
PicoPass 32KS (8x2 + 8x2)	ISO14443B: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 20 00 00 00 00 4E

一般的なアプリケーションの応答時間を短縮するために、デフォルトでは次の PICC タイプ/テクノロジーのサポートが無効になっています。ユーザーは、直接コマンド“Set operation Mode”で、さまざまなタイプ/テクノロジーのサポートを有効にすることができます。対応するタイプ/テクノロジーが有効になり、リーダライタにカードをタッチ、PC _ to _ RDR _ IccPowerOn コマンドは次の ATR を CCID ホストにに返します。

カードタイプ/技術	ATR
SRI (SRIX4K/SRT512)	3B 8F 80 01 80 4F 0C A0 00 00 03 06 06 00 07 00 00 00 00 69
Topaz	3B 8F 80 01 80 4F 0C A0 00 00 03 06 02 00 30 00 00 00 00 5A
PicoPass 2K	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 17 00 00 00 00 75
PicoPass 2KS	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 18 00 00 00 00 7A
PicoPass 16K	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 19 00 00 00 00 7B
PicoPass 16KS	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1A 00 00 00 00 78
PicoPass 16K (8 x 2)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1B 00 00 00 00 79
PicoPass 16KS (8 x 2)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1C 00 00 00 00 7E
PicoPass 32KS (16 + 16)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1D 00 00 00 00 7F
PicoPass 32KS (16 + 8x2)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1E 00 00 00 00 7C
PicoPass 32KS (8x2 + 16)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 1F 00 00 00 00 7D
PicoPass 32KS (8x2 + 8x2)	ISO15693: 3B 8F 80 01 80 4F 0C A0 00 00 03 06 0A 00 20 00 00 00 00 42
Innovatron	3B 88 80 01 80 4F 05 F0 49 4E 4E 4F 35
CTS	3B 87 80 01 80 4F 04 F0 43 54 53 79

6.0. Escape コマンド

Escape コマンドは、PCSC API の SCARD__CTL__CODE（3500）に対応する PC__to__RDR__Escape（）を介して送信されます。コマンドの処理完了すると、リーダライタは RDR_to_PC_Escape メッセージを介してレスポンスを返します。



以下のコマンドは、PCD/NFC の構成、およびリーダ/ライタへのアクセスのための特別な機能に使用されます。CCID ホストが PCSC API の SCARD__CTL__CODE（3500（SCardControl））に対応する CCID メッセージ PC_to_RDR_Escape を使用して、カードリーダーにこれらのコマンドを送信することができます。Escape コマンドを受信すると、リーダライタはコマンドを解読して実行し、レスポンスを生成して CCID ホストに返信します。

注釈：
これらのコマンドは正しいインタフェースを介して送信する必要があります。例えば、*E 0 00 25 01 00*（*6.4.1.1*節）は *PICC*インタフェースを介して送信されるべきです（*6.4.1*節）。

6.1. PICCEscape コマンド

6.1.1. RF 制御（RF Control）[E0 00 00 25 01 …]

このコマンドは RF 制御を設定するために使われます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
RF Control	E0h	00h	00h	25h	01h	RF 状態

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	RF 状態

RF 状態：1 バイト

RF 状態	説明
00h	RF オフ
01h	RF オン ポーリング
02h	RF オン ポーリングしない

デフォルト設定– *01h (RF オン ポーリング)*

6.1.2. PCD/PICC 状態取得（Get PCD/PICC Status）[E0 00 00 25 00]

このコマンドは PCD/PICC の状態を取る際に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get PCD/PICC Status	E0h	00h	00h	25h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	Get PCD/PICC Status

PCD/PICC 状態：1 バイト

RF 状態	説明
00h	RF オフ
01h	PICC ない
02h	PICC 準備完了
03h	PICC 選択済み/アクティブ
FFh	エラー

6.1.3. ポーリング/ATR オプションの取得（Get Polling/ATR Option）[E0 00 00 23 00]

このコマンドはポーリングオプションを設定/取得するために使用され、他のコマンドを使用することなく設定を保存できます。このコマンドは、最初のリーダ/ライタ構成にのみ使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Polling/ATR Option	E0h	00h	00h	23h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	03h	01h	PICC のポーリング/ATR オプション

6.1.4. ポーリング/ATR オプションの設定（Set Polling/ATR Option）[E0 00 00 23 01 …]

このコマンドはポーリングオプションを設置する時に使われます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set Polling/ATR Option	E0h	00h	00h	23h	01h	PICC のポーリング/ATR オプション

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	PICC のポーリング/ATR オプション

PICC ポーリング/ATR オプション　_1 バイト

操作パラメータ	パラメータ	説明	オプション
Bit 0	ポーリングを有効する	PICCが検出待ちのタブタイプをポーリングします。	1 = 検出 0 = スキップ
Bit 1	RF　オフ		
Bit 2	RFU		
Bit 3	追加の MIFARE タイプに対する第 3 部分カード ATR の識別を有効にする	PICC が検出待ちのタブタイプをポーリングします。	1 = 検出 0 = スキップ
Bit 4 ~ 5	RF　オフ		下記の表を確認ください
Bit 6	RFU		
Bit 7	SmartMX/JCOS カードシミュレーション MIFARE 用のパート 4 ATR を有効化	PICC が検出待ちのタブタイプをポーリングします。	1 = 検出 0 = スキップ

RF オフ間隔–2 Bit　　ケース 1：RF オフを無効にする（Bit 1=0）

操作パラメータ		USB 実行(D0)	USB 保留(D2)
Bit 5	Bit 4		
0	0	RF　オフない	250 ms
0	1		500 ms
1	0		1000 ms
1	1		2500 ms

ケース 2：RF オフ有効 (Bit 1 = 1)

操作パラメーター		USB 実行(D0)	USB 保留(D2)
Bit 5	**Bit 4**		
0	0	250 ms	500 ms
0	1	500 ms	1000 ms
1	0	1000 ms	2500 ms
1	1	2500 ms	2500 ms

デフォルト設定–*8 Bh*（ポーリングを有効にし、*RF* オフを有効にし、第 *3* 部分カードが *ATR* における追加の *MIFARE* タイプの識別を有効にし、*Rf* オフ間隔［*00*］、*SmartMX/JCOS* カードシミュレーション *MIFARE* に適用する第 *4* 部分 *ATR* を有効にする）

6.1.5. PICC ポーリングタイプ取得（Get PICCPolling Type）[E0 00 01 20 00]

このコマンドは許可されるテクノロジー/ポーリングタイプを取得するために使用され、他のコマンドを使用することなく設定を保存できます。最初のリーダ/ライタ構成にのみ使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get PICC Polling Type	E0h	00h	01h	20h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
						バイト 1	バイト 0
結果	E0h	00h	00h	00h	02h	PICC ポーリングタイプ	

6.1.6. PICC ポーリングタイプ設定（Set PICC Polling Type）[E0 00 01 20 02 ...]

このコマンドは PICC ポーリングタイプを設置する時に使われます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力	
						バイト 1	バイト 0
Set PICC Polling Type	E0h	00h	01h	20h	02h	PICC ポーリングタイプ	

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
						バイト 1	バイト 0
結果	E0h	00h	00h	00h	02h	PICC ポーリングタイプ	

PICC ポーリングタイプ-2 バイト、LSB の前で、ビットマスクは次の通りです

□ 节	□ □ □ □ □ —□ —	パラメーター	説明	オプション
バイト 1	Bit 0	ISO 14443A Type A	PICC が検出待ちのタブタイプをポーリングします。	1 = 検出 0 = スキップ
	Bit 1	ISO 14443A Type B		
	Bit 2	FeliCa		
	Bit 3	RFU		
	Bit 4	Topaz		
	Bit 5	Innovatron		
	Bit 6	SRI/SRIX		
	Bit 7	RFU		
バイト 0	Bit 0	Picopass (ISO14443B)		
	Bit 1	Picopass (ISO15693)		
	Bit 2	ISO15693		
	Bit 3	CTS		
	Bit 4-7	RFU		

デフォルト設定 – バイト *1 = 07h (ISO14443 Type A, ISO14443 Type B, FeliCa)*

バイト *0 = 05h (Picopass (ISO14443B), ISO15693)*

例：

コマンド: *E0 00 01 20 02 07 05*

応答: *E1 00 00 00 02 07 05*

ポーリングタイプ: バイト *1 = 07h = 0000 0111b = ISO14443 Type A, ISO14443 Type B, FeliCa*

バイト *0 = 05h = 0000 0101b = Picopass (ISO14443B), ISO15693*

6.1.7. 自動 PPS 取得（Get Auto PPS ）[E0 00 00 24 00]

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は定義変更れた 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。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Auto PPS	E0h	00h	00h	24h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	最高速度	現在速度

6.1.8. 自動 PPS 設定（Set Auto PPS ）[E0 00 00 24 01…]

このコマンドは自動的な PPS のを設置するために使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set Auto PPS	E0h	00h	00h	24h	01h	最高速度

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	最高速度	現在速度

PPSの速率

速率	説明
00h	106 kbps;自動PPSが設定されていないと同じです。
01h	212 kbps
02h	424 kbps
03h	848 kbps

デフォルト設定– *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によるして、*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、高いレート設定がリーダライタの動作に影響を与える場合は、低いレート設定に切り替えます。

6.1.9. PICC タイプ取得（Read PICC Type）[E0 00 00 35 00]

PICC タイプを読み取る時にこのコマンドを使用します。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get PICC Type	E0h	00h	00h	35h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	Type	状態

タイプ：1 バイト

タイプ	説明
CCh	PICC ない
04h	Topaz
10h	MIFARE
11h	FeliCa
20h	Type A, Part 4
23h	Type B, Part 4
25h	Innovatron
28h	SRIX
30h	PicoPass
FFh	その他

状態：1 バイト

状態	説明
00h	RF オフ
01h	PICC ない
02h	PICC 準備完了
03h	PICC 選択済み/アクティブ
FFh	エラー

6.1.10. RF パワー設定取得（Get RF Power Setting）[E0 00 00 50 00]

このコマンドは、RF の電力設定を読み取るために使用されます。ファームウェアのバージョン要件は次のとおりです:

- ACR1552U-M FW 1.03.03 以降
- ACM1552U-Y FW 2.03.03 以降
- ACM1552U-Z FW 2.03.03 以降

コマンド

コマンド	CLA	INS	P1	P2	Le
Get RF Power Setting	E0h	00h	00h	50h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E0h	00h	00h	00h	01h	RF パワー

6.1.11. RF パワー設定（Set RF Power Setting）[E0 00 00 50 01 ...]

このコマンドは、PICC ポーリング タイプを設定するために使用されます。ファームウェアのバージョン要件は、RF 電力設定の取得と一致しています。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set RF Power Setting	E0h	00h	01h	50	01h	RF パワー

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E0h	00h	00h	00h	01h	RF パワー

パーセンテージモード

RF パワー -1 バイト

パラメータ	説明
00h	手動RFパワー設定を無効にする
01h	20%
02h	40%
03h	60%

パラメーター	説明
04h	80%
05h	100%

デフォルトに– *00h*

*ハードウェアの制約により、パーセンテージモードの RF 電力値は有効にならない可能性があります。

6.1.12. PICC– HID キーボードの Escape コマンド

6.1.12.1. 出力フォーマット取得（Get Output Format）[E0 00 00 90 00]

このコマンドは RTC の出力フォーマットを取得する時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Output Format	E0h	00h	00h	90h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	出力フォーマット	出力順番

6.1.12.2. 出力フォーマット設定（Set Output Format）[E0 00 00 90 02…]

このコマンドは出力フォーマットを設定する時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力	
Set Output Format	E0h	00h	00h	90h	02h	出力フォーマット	出力順番

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	出力フォーマット	出力順番

出力フォーマット　　1 バイト。

操作パラメ―タ―	パラメ―タ―	説明	オプション
Bit 7 ~ 4	文字の大文字と小文字	PICCが検出待ちのタブタイプをポーリングします。	1 = 検出 0 = スキップ
Bit 3 ~ 0	表示モード		

出力順番　　1 バイト。

状態	説明
00h	デフォルト順番(UIDバイト0, UIDバイト1 … UIDバイトN) 例：aa cc bb dd (原始/実際のUID順番)
01h	順番反転(UIDバイトN, UIDバイトN-1 … UIDバイト0) 例：dd bb cc aa (UID順番反転)

大文字と小文字：高 4 位(Bit 7 から Bit 4 まで)

状態(bit 7 から bit 4 まで)	説明(x bit を注目する必要がない)
1XXX	保留
00x0	小文字
00x1	大文字
000x	4バイトのUIDのみサポート
001x	4、7、8、10バイトのUIDサポート

表示モード：低 4 位(Bit 3 から Bit 0)

状態(bit 7 から bit 4 まで)	説明(x bit を注目する必要がない)
0h	Hex
1h	Dec (バイトごと)
2h	Dec
3h	6H-6H
4h	8H-8H
5h	10H-10H
6h	14H-14H
7h	20H-20H
8h	6H-8D
9h	6H-10D
Ah	8H-10D

状態(bit 7 から bit 4 まで)	説明(x bit を注目する必要がない)
Bh	10H-14D
Ch	2H4H-8D
Dh	14H-17D

6.1.12.3. UID 開始、中間、終了ビット文字の取得（Get Character at Start, Between, at End UID）[E0 00 00 91 00]

このコマンドは UID 開始、中間、終了のビットを取得する際に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Character of UID	E0h	00h	00h	91h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン		
結果	E1h	00h	00h	00h	03h	中間	終わり	終始

6.1.12.4. UID 開始、中間、終了ビット文字の設定（Set Character at Start, Between, at End UID）[E0 00 00 91 03…]

このコマンドは UID 開始、中間、終了のビットを設定する際に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力		
Set Character of UID	E0h	00h	00h	91h	03h	中間	終わり	終始

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン		
結果	E1h	00h	00h	00h	03h	中間	終わり	終始

中間：1 バイト（各 UID 間のキャラクター）

状態	説明
FFh	中間にはキャラクターがない
その他	汎用シリアルバス（USB）HID使用テーブルを参照

終わり 1 バイト（末尾の文字を出力）

状態	説明
FFh	中間にはキャラクターがない

状態	説明
その他	汎用シリアルバス（USB）HID使用テーブルを参照

終始 1 バイト（開始文字を出力）

状態	説明
FFh	中間にはキャラクターがない
その他	汎用シリアルバス（USB）HID使用テーブルを参照

注釈：

1. *AZERTY*キーボードレイアウトは、零*(0)*とバックスペースキー“ではなく;” “,” “,” “,” “-“ を中間の文字としてサポートされます。

6.1.12.5. キーボードレイアウト言語の取得（Get Keyboard Layout Language）[E0 00 00 92 00]

このコマンドはキーボードレイアウト言語を取得する時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Keyboard Layout Language	E0h	00h	00h	92h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	キーボードレイアウト言語

6.1.12.6. キーボードレイアウト言語の設定（Set Keyboard Layout Language）[E0 00 00 92 01 …]

このコマンドはキーボードレイアウト言語を設定する時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set Keyboard Layout Language	E0h	00h	00h	92h	01h	キーボードレイアウト言語

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	キーボードレイアウト言語

キーボードレイアウト言語：1 バイト

状態	説明
00h	英語
01h	フランス語
02h	保留
03h	リトアニア語

6.1.12.7. ホストインターフェース取得 （Get Host Interface） [E0 00 00 93 00]

このコマンドはホストインターフェースを取得する時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Host Interface	E0h	00h	00h	93h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	ホストインターフェース

6.1.12.8. ホストインターフェース設定 （Set Host Interface） [E0 00 00 93 01…]

このコマンドはホストインターフェースを設定する時に使用されます。

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set Host Interface	E0h	00h	00h	93h	01h	ホストインターフェース

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	ホストインターフェース

ホストインターフェース:バイト

状態	説明
00h	HIDキーボードのみ適応
01h	CCIDカードリーダーのみ適用
02h	HIDキーボード+CCIDカードリーダー

6.1.13. PICC-カードシミュレーションの Escape コマンド

6.1.13.1. カードエミュレーションモードに入る（Enter Card Emulation Mode）[E0 00 00 40 03 …]

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、ユーザが設定可能です。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力		
Enter Card Emulation Mode	E0h	00h	00h	40h	03h	NFC モード	00h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	03h	NFC モード

NFC 設備モード：3 バイト

状態	説明
02h	NFCフォーラムタイプ2ラベルモード
03h	FeliCa
その他	カードリーダ/ライタモード

注：異なるカードシミュレーションモードに切り替える前に、まずカードの読み取り/書き込みモードに入ってください。カードシミュレーションモードの初期完了後にレスポンスが表示されます。

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	SN3	Nil
保留	保留	保留	保留	保留	Nil
内部/ロック	保留	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

(1988

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
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
データ読み取り/書き込み	Data52	Data53	Data54	Data55	52-55
データ読み取り/書き込み		…			…
データ読み取り/書き込み	Data1984	Data1985	Data1986	Data1987	1984~1987

表6：NFC フォーラムタイプ 2 ラベルのメモリマップ（2000 バイト）

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7：FeliCa カードのメモリマップ（160 バイト）

その中：

デフォルト： ブロック 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 タイプ 3 タグ属性情報ブロック

注釈：

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. *FeliCa* カード識別番号（*IDm*）はユーザに定義可能で、メーカーコードが（*0388*）に固定されています。

6.1.13.2. カードエミュレーションのデータを読み取る（Read Card Emulation Data）（NFC フォーラムタイプ 2 ラベル）[E0 00 00 60 04 …]

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。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン			
Read Card Emulation Data	E0h	00h	00h	60h	04h	00h	NFC モード	終始オフセット	長さ

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	長さ	データ

終始オフセット：　1 バイト – **表 6** 中 Data0 からのアドレス

長さ　1 バイト –バイト数量

6.1.13.3. カードエミュレーションのデータ書き込む（Write Card Emulation Data）（NFC フォーラムタイプ 2 ラベル）[E0 00 00 60 …]

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。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン				
Write Card Emulation Data	E0h	00h	00h	60h	長さ＋04h	01h	NFC モード	終始オフセット	長さ	データ

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン		
結果	E1h	00h	00h	00h	03h	長さ	90h	00h

NFC 設備モード：1 バイト

状態	説明
02h	NFCフォーラムタイプ2ラベルモード
03h	FeliCa
その他	カードリーダ/ライタモード

終始オフセット：　1 バイト – **表 6** 中 Data0 からのアドレス

長さ　1 バイト –バイト数量

6.1.13.4. カードエミュレーションのデータを読み取る（Read Card Emulation Data）（NFC フォーラムタイプ 2 ラベル）（拡張）

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。

コマンド

コマンド	CLA	INS	P1	P2	Lc		コマンドデータイン			
Read Card Emulation Data	E0h	00h	01h	60h	05h	00h	NFC モード	終始オフセット Bit [15:8]	終始オフセット Bit [7:0]	長さ

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	長さ	データ

終始オフセット： 2 バイト – **表 6** 中 SN 0 からアドレスを読み取る

長さ： 1 バイトー読み取り待ちのバイト

6.1.13.5. カードエミュレーションのデータ書き込む（Write Card Emulation Data）（NFC フォーラムタイプ 2 ラベル）（拡張）

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。

コマンド

コマンド	CLA	INS	P1	P2	Lc		コマンドデータイン				
Write Card Emulation Data	E0h	00h	01h	60h	長さ＋ 05h	01h	NFC モード	終始オフセット Bit [15:8]	終始オフセット Bit [7:0]	長さ	データ

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン		
結果	E1h	00h	00h	00h	03h	長さ	90h	00h

NFC 設備モード：1 バイト

状態	説明
02h	NFCフォーラムタイプ2ラベルモード
その他	カードリーダ/ライタモード

終始オフセット： 2 バイト – **表 6** 中 SN 0 からアドレスを書き込む

長さ： 1 バイトー書き込み待ちのバイト

6.1.13.6. NFC フォーラムタイプ 2 ラベル模擬 ID を設定（Set Card Emulation of NFC Forum Type 2 Tag ID）[E0 00 00 61 03 …]

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。

コマンド

コマンド	CLA	INS	P1	P2	Lc	コマンドデータイン
Set Card Emulation Lock Data	E0h	00h	00h	61h	03h	3 バイトの UID

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン	
結果	E1h	00h	00h	00h	02h	90h	00h

6.1.13.7. NFC カードエミュレーションロックデータロックを設定する（Set Card Emulation Lock Data in NFC）[E0 00 00 65 01…]

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。

コマンド

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

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	ロック

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

操作パラメーター	パラメーター	説明	オプション
Bit 7 ~ 2	保留	保留	
Bit 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 直接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
Bit 0	NFC フォーラムタイプ 2 タブを有効する		

6.1.13.8. カードエミュレーションの FeliCa の IDM を設定する（Set Card Emulation FeliCa IDm）[E0 00 00 64 06 …]

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。

コマンド

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

応答コード

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

6.1.13.9. カードエミュレーション状態取得（Get Card Emulation Status）[E0 00 00 69 00]

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取得するために使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc
Get Card Emulation Status	E0h	00h	00h	69h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	状態

状態：1 バイト

操作パラメータ	モード	説明
Bit 7 ~ 6	保留	保留
Bit 5	模擬カードがアクティブ化された	1 = アクティブ化される
Bit 4	模擬カードが外れた	1 = カードが外れた
Bit 3	模擬カードが全部読み取りました	1 = すべてのデータが読み取りました
Bit 2	模擬カードが読み取りました	1 = データが読み取りました
Bit 1	模擬カード書き込まられた	1 = データ書き込まられた
Bit 0	模擬カードが検出された	1 = カード検出

6.1.13.10. シミュレーション NFC フォーラムタイプ 2 ラベルモードのコマンドサンプルセット

このコマンドセットは、ACR 1552 U の NFC フォーラムタイプ 2 ラベルモードをシミュレートし、ACS サイトをトリガします https://www.acs.com.hk。ステップ：

1. 次のコマンドを使用してカードシミュレーションモードに入ります。
    - 送信 Enter Card Emulation Mode

      E0 00 00 40 03 02 00 00
2. 下記のコマンドで NDEF データを含めています。データ書きます：
    - 送信 Write Card Emulation Data (NFC Forum Type 2 Tag)

      E0 00 00 60 1C 01 02 00 18 E1 10 F4 00 03 0F D1 01 0B 55 02 61 63 73 2E 63 6F 6D 2E 68 6B FE 00 00

注釈：

NDEF（*NFC* データ相互フォーマット）に関する詳細な情報と規定を了解する必要がある場合は、*NDEF* 仕様を参照することをお勧めします。この仕様では、*NDEF* レコードの構造と使用について包全面的なガイドラインと詳細情

報を提供し、これらの *NFC* レコードは *NFC* データのインタラクションによく使用されています。*NDEF* 仕様は、*ACR552U* デバイス環境における *NDEF* コマンドとデータの解読と利用方法を深く理解するのに役立ちます。

6.1.14. PICC-検出モードの Escape コマンド

6.1.14.1. 検出モードに入る（Enter Discovery Mode）[E0 00 00 6A 01 …]

このコマンドは検出モードに入る時に使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Enter Discovery Mode	E0h	00h	00h	6Ah	01h	検出モード

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	検出モード

検出モード：1 バイト

状態	説明
00h	カードリーダーモード
02h	NFCフォーラムタイプ2ラベルモード
03h	FeliCa

6.2. 周辺設備制御と他の Escape コマンド

6.2.1. ファームウェアバージョン取得（Get Firmware Version）[E0 00 00 18 …]

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。

コマンド

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

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	ファームウェアのバージョンの長さ	ファームウェアのバージョン

例：
コマンド：　　E0 00 00 18 00
応答コード：　　E1 00 00 00 14 41 43 52 31 35 35 32 20 52 20 46 57 20 31 2E 30 30 2E 30 30
16 進法ファームウェアバージョン：41 43 52 31 35 35 32 20 52 20 46 57 20 31 2E 30 30 2E 30 30
ASCII ファームウェアバージョン：ACR1552 R FW 1.00.00

6.2.2. シリアルナンバー取得（Get Serial Number ）[E0 00 00 33 00]

シリアルナンバーを取得する時にこのコマンドを使用します。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get Serial Number	E0h	00h	00h	33h	00h

応答コード

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	シリアルナンバーの長さ	シリアルナンバー

6.2.3. USB 記述子内の S/N を設定する（Set S/N in USB Descriptor）[E0 00 00 F0]

このコマンドは USB 記述子内の S/N を設定するために使われます。

コマンド

コマンド	CLA	INS	P1	P2	Le	コマンドデータイン	
Set S/N in USB Descriptor	E0h	00h	00h	F0h	02h	00h	USB 記述子内の SN を有効する

応答コード

応答	CLA	INS	P1	P2	Le	データ出力		
結果	E1h	00h	00h	00h	03h	USB 記述子内の SN を有効する	90h	00h

USB 記述子内の SN を有効する（1 バイト）

USB 記述子内の SN を有効する	説明
00h	USB 記述子内の SN を無効する
01h	USB 記述子内の SN を有効する

6.2.4. ブザー制御の設定-単発（Set Buzzer Control - Single Time）[E0 00 00 28 01 …]

このコマンドは単発のブザーを設定するために使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Buzzer Control	E0h	00h	00h	28h	01h	ブザー状態

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	ブザー状態

ブザーの状態(1 バイト)

ブザー状態	説明
00h	OFF
01 ~ FFh	オン、持続時間は 10 ms を単位にする

6.2.5. ブザー制御の設定-重複（Set Buzzer Control - Repeatable）[E0 00 00 28 03 …]

このコマンドはブザーの周期を設定するために使用されます周期

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Buzzer Control	E0h	00h	00h	28h	03h	ブザー状態

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	03h	ブザー状態

ブザーの状態(3 バイト)

操作パラメ―タ―	ブザー状態	説明
パラメ―タ― 1 – バイト 0	オン時間帯	01 ~ FF: オンにする持続時間は 10 ms を単位にする
パラメ―タ―2 – バイト 1	OFF 時間帯	01 ~ FF: OFF にする持続時間は 10 ms を単位にする
パラメ―タ―3 – バイト 2	重複時間	01 ~ FF: 繰り返し回数

6.2.6. LED 状態取得（Get LED Status）[E0 00 00 29 00]

このコマンドは LED の状態を取得するために使用されます。

コマンド

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

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	LED 状態

6.2.7. LED 制御設定（Set LED Control）[E0 00 00 29 01 …]

このコマンドは LED 制御を設定するために使われます

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set LED Control	E0h	00h	00h	29h	01h	LED 状態

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	LED 状態

LED 状態（1 バイト）

LED 状態	説明
Bit 0 = 青色 LED	1 = ON；0 = OFF
Bit 1：緑の LED	1 = ON；0 = OFF
Bit 2-7 ： RFU	その他

6.2.8. UI 操作取得（Get UI Behaviour）[E0 00 00 21 00]

このコマンド PCD UI の操作を取得するために使用され、他のコマンドを使用することなく設定を保存できます。最初のリーダ/ライタ構成にのみ使用されます。

コマンド

コマンド	CLA	INS	P1	P2	Le
Get PICC UI Behaviour	E0h	00h	00h	21h	00h

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	PICC UI 操作

6.2.9. UI 操作設定（Set UI Behaviour）[E0 00 00 21 01 …]

このコマンドは、PICC UI の動作を設定するために使用されます。

UI 操作の設定 bit1 および bit2 は、次のバージョンのファームウェアのみをサポートします。

- ACR1552U-M FW 1.03.05 以降
- ACM1552U-Y FW 2.03.05 以降
- ACM1552U-Z FW 2.03.05 以降

コマンド

コマンド	CLA	INS	P1	P2	Lc	データ出力
Set PICC UI Behaviour	E0h	00h	00h	21h	01h	PICCUI 操作

応答コード

応答	CLA	INS	P1	P2	Le	コマンドデータイン
結果	E1h	00h	00h	00h	01h	PICC UI 操作

UI 操作-1 バイト、ビットマスクは次の通りです

操作パラメータ	パラメータ	説明	オプション
Bit 0	動作中（LED が速く点滅する）	リーダーのUI操作	1 = 有効にする 0 = 無効にする
Bit 3	カードがアンテナ領域に入るイベント（ブザーが一時的に鳴る）		
Bit 4	カードがアンテナ領域から外すイベント（ブザーが一時的に鳴る）		

PICC デフォルト設定– *09h*

注釈：

*1.UI*操作コマンドの取得／設定に*SAM*インタフェースは含まれていません。

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”