



Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1251U

SAM 付きの NFC

カードリーダー



アプリケーションプログラミングインターフェース V1.14



改定履歴

リリース日付	改訂説明	バージョン
2013/05/31	<ul style="list-style-type: none">● 初回発布	1.00
2014/01/08	<ul style="list-style-type: none">● セクション 6.4 の以下のコマンドを削除する<ul style="list-style-type: none">○ 自動 PPS 設定○ 自動 PPS 読み取り○ アンテナフィールド制御○ アンテナフィールド状況を読み取る○ ユーザー余裕の保護時間を読み取る○ “616C” 自動操作オプション設定○ “616C” 自動操作オプション読み取る	1.01
2014/03/07	<ul style="list-style-type: none">● 各セクションを改めて並ぶ	1.02
2014/04/03	<ul style="list-style-type: none">● 5.7.6、5.7.7、5.6.8 および 5.6.9 セクションを変更します： LED インジケータとブザーのインジケータを設定/読み取り（ファームウェアバージョン 205 前、およびファームウェアバージョン 206 以降に適用）● MIFARE の商標および帰属の説明を追加。	1.03
2014/05/23	<ul style="list-style-type: none">● セクション 5.8 更新：NFC ピアツーピアモードコマンド● 書式設定更新● 付録略号を第 3.0 項に移動：略語● ACR122U 互換コマンドを 5.9 セクションに移動● 2.0 セクションの特性に KC 認証を追加	1.04
2014/10/16	<ul style="list-style-type: none">● セクション 2.0 更新：特性	1.05
2014/10/30	<ul style="list-style-type: none">● 5.3.1 セクション追加：データを取得	1.06
2015/10/06	<ul style="list-style-type: none">● ACR1251U-A1 を ACR1251U にの変更● 2.0 セクションの特性に REACH 認証を追加	1.07
2017/09/20	<ul style="list-style-type: none">● セクション 2.0 更新：特性	1.08



リリース日付	改訂説明	バージョン
2018/05/21	<ul style="list-style-type: none">• セクション 5.2 の非接触型スマートカード関連コマンドの再グループ化• セクション 5.1 の追加例 PCSC API• セクション 5.2.3 -PCSC2.0 パート 3 のための APDU コマンドの追加• セクション 5.3 スマートカードプロトコルにコンタクトを追加• セクション 5.3.1 ACOS6-SAM コマンドの追加• セクション 5.4.14 自動 PPS の設定の追加とセクション 5.4.15 自動 PPS の読み取り• 更新されたセクション 5.7.10 自動 PICC ポーリングの設定• セクション 5.2.1.1 ISO 14443 パート 3 PICC の ATR フォーマットの更新	1.09
2018/09/11	<ul style="list-style-type: none">• セクション 2.0 更新：特性	1.10
2019/08/05	<ul style="list-style-type: none">• セクション 5.4.10 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 5.4.11 更新：自動的な PICC のポーリングを読み取る (Read Automatic PICC Polling)	1.11
2019/08/30	<ul style="list-style-type: none">• MIFARE の商標および帰属の説明を追加。• セクション 5.2.4.7 更新：数値ブロックをコピーする (Copy Value Block)	1.12



リリース日付	改訂説明	バージョン
2020/03/27	<ul style="list-style-type: none">• セクション 5.4 更新 :• セクション 5.4.6 更新 : LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior)• セクション 5.4.7 更新 : LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior)• セクション 5.4.8 更新 : PICC インターフェースの LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior for PICC interface)	1.13
2020/08/27	<ul style="list-style-type: none">• セクション 5.4.9 更新 : PICC インターフェースの LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior for PICC interface)• セクション 5.4.11 更新 : 自動的な PICC のポーリングを読み取る (Read Automatic PICC Polling)• セクション 5.4.14 更新 : 自動的な PPS を設定する (Set Auto PPS)• セクション 5.4.15 更新 : 自動的な PPS を読み取る (Read Auto PPS) <hr/> <ul style="list-style-type: none">• セクション 5.4.6 更新 : LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW205 以下)• セクション 5.4.7 更新 : LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior) (FW205 以下)• セクション 5.4.8 更新 : LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW206 以降)• セクション 5.4.9 更新 : LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior) (FW206 以降)	1.14



カタログ

1.0.	紹介	7
2.0.	特性	8
3.0.	略語	9
4.0.	アーキテクチャ	10
5.0.	ホストプログラミング (PC リンク) API	11
5.1.	PC/SC API	11
5.1.1.	SCardEstablishContext	11
5.1.2.	SCardListReaders	12
5.1.3.	SCardConnect	13
5.1.4.	SCardControl	14
5.1.5.	SCardTransmit	16
5.1.6.	SCardDisconnect	18
5.1.7.	APDU の流れ	19
5.1.8.	ダイレクトコマンド (Escape Command) の流れ	20
5.2.	非接触スマートカード プロトコル	21
5.2.1.	ATR の生成	21
5.2.2.	非接触インターフェースの疑似 APDU コマンド	25
5.2.3.	PCSC 2.0 パート 3 の APDU コマンド (2.02 もしくはもっと新しいバージョン)	26
5.2.4.	MIFARE Classic 1K/4K メモリカードの PICC コマンド (T=CL エミュレーション)	42
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)	55
5.2.6.	FeliCa タグのアクセス	57
5.3.	非接触スマートカード プロトコル	58
5.3.1.	ACOS6-SAM コマンド	58
5.4.	周辺デバイス制御	76
5.4.1.	ファームウェアのバージョンを取得する (Get Firmware Version)	76
5.4.2.	LED 制御 (LED Control)	77
5.4.3.	LED 状態 (LED Status)	78
5.4.4.	ブザー制御 (Buzzer Control)	79
5.4.5.	ブザーの状態 (Buzzer Status)	80
5.4.6.	LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW205 以下)	81
5.4.7.	LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior) (FW205 以下)	82
5.4.8.	LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW206 以降)	83
5.4.9.	LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior) (FW206 以降)	84
5.4.10.	自動的な PICC のポーリングを設置する (Set Automatic PICC Polling)	85
5.4.11.	自動的な PICC のポーリングを読み取る (Read Automatic PICC Polling)	87

5.4.12.	PICC 操作のパラメータを設定する (Set PICC Operating Parameter)	88
5.4.13.	PICC 操作のパラメータを読み取る (Read PICC Operating Parameter)	89
5.4.14.	自動的な PPS を設定する (Set Auto PPS)	90
5.4.15.	自動的な PPS を読み取る (Read Auto PPS)	91
5.5.	NFC P2P についてのコマンド	92
5.5.1.	イニシエータモードについてのコマンド	92
5.5.2.	ターゲットモードについてのコマンド	100
5.6.	ACR122U 互換性のあるコマンド	110
5.6.1.	二色の LED とブザー制御 (Bi-color LED and Buzzer Control)	110
5.6.2.	ファームウェアのバージョンを取得する (Get Firmware Version)	113
5.6.3.	PICC 操作のパラメータを入手する (Read the PICC Operating Parameter)	114
5.6.4.	PICC 操作のパラメータを設定する (Set PICC Operating Parameter)	115
Appendix A. SNEP メッセージ		116

図示カタログ

図 1	: ACR1251U-A1 のアーキテクチャ	10
図 2	: ACR1251U-A1 の APDU の流れ	19
図 3	: ACR1251U-A1 直接コマンド (Escape Command) の流れ	20
図 4	: イニシエータモードでの P2P 流れ	92
図 5	: ターゲットモードでの P2P 流れ	100

チャートカタログ

表 1	: 略語	9
表 2	: MIFARE Classic 1K カードのメモリマップ	45
表 3	: MIFARE Classic 4K カードのメモリマップ	46
表 4	: MIFARE Ultralight カードのメモリマップ	47



1.0. 紹介

ACR1251U は 13.56MHz 非接触技術に基づいて開発された PC リンク非接触スマートカードリーダー/ライターです。ACS 最も成功した世界初の CCID に準拠した非接触型 NFC リーダー--ACR122U の次、もっと高度な機能を提供しています。ISO 14443 A カード、B カード、MIFARE®、FeliCa、4 テープの NFC タグとデバイスをサポートしています。

PC とカードの間の仲介デバイスとして、ACR1251U は PC からのコマンドを実行して、非接触タグ、SAM カードおよび周辺デバイス（LED またはブザー）と通信する専用デバイスです。ACR1251U-A1 は PC からのコマンドを実行して、非接触タグ、SAM カードおよび周辺デバイス（LED またはブザー）と通信する専用デバイスです。ACR1251U-A1 の 2 つのインターフェース、すなわち PICC と SAM インターフェース、両方は PC / SC 仕様に準拠している。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は ACR1251U の周辺機器をサポートする。この API ドキュメントはこれについて詳しく説明している。



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：
 - 書き込み速度= 424 kbps
 - 内蔵アンテナを使って、通信距離は最大 50 mm (タグのタイプに応じて)
 - Mifare®や ISO 14443 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。
 - 衝突防止機能保有 (どんな時でも一枚タグしかアクセスできません)
 - 拡張の APDU サポート (最大 64 KB)
 - NFC サポート
 - 読み取りモード
 - ピアツーピア通信モード
 - SAM インターフェース
 - 1 つ SAM カードスロット
 - ISO 7816 クラス A、B、C の SAM カードサポート
- 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
 - ユーザー制御可能なブザー
- 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
- 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：
 - EN 60950/IEC 60950
 - ISO 7816 (SAM スロット)
 - ISO 14443
 - ISO 18092
 - FeliCa 性能認証
 - PC/SC
 - CCID
 - CE
 - FCC
 - RoHS
 - REACH
 - J-LIS (日本)
 - VCCI (日本)
 - MIC (日本)
 - KC (韓国)
 - Microsoft® WHQL

¹ ACS の Android ライブラリを使用

3.0. 略語

略語	説明
ATR	属性請求と属性応答
DEP	データ交換プロトコルの請求と応答
DSL	請求と応答のキャンセル
PSL	パラメーターオプションの請求と応答
RLS	請求リリースと応答リリース
WUP	ウェイクアップ請求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

ACR1251U と PC のデータ通信は CCID プロトコルを採用していますが、PICC と SAM 間の通信は PC/SC 規格に準拠しています。

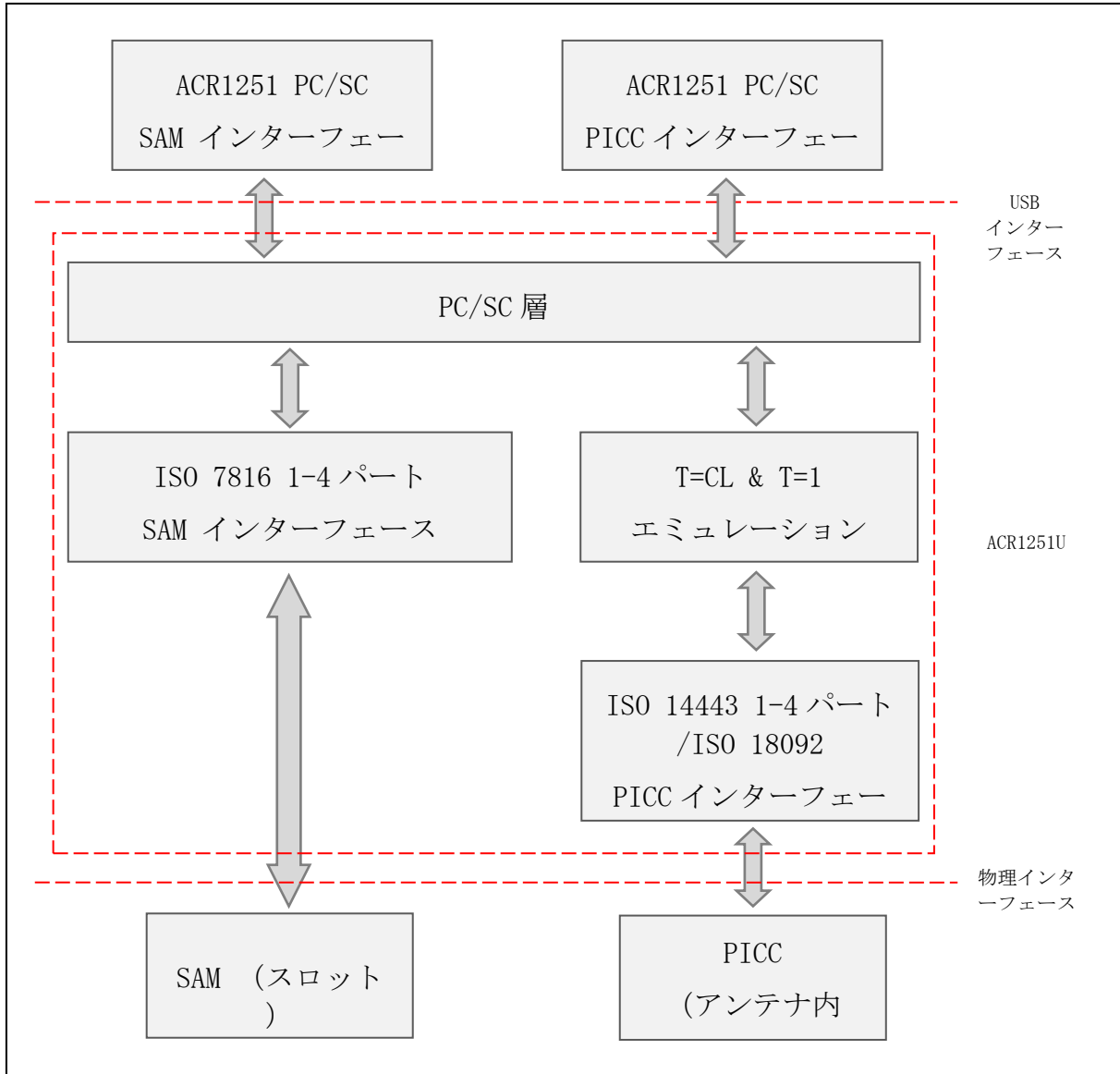


図1 : ACR1251U-A1 のアーキテクチャ



5.0. ホストプログラミング (PC リンク) API

5.1. PC/SC API

このセッションでは、いくつか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。

参照のウェブサイト :

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa379479%28v=vs.85%29.aspx>

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：

<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（特別のリソースマネージャのコンテキストを利用して）アプリケーションと特定のリーダーを含んで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：

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa379473%28v=vs.85%29.aspx>

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 ACR1251 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
        }
    }
}
```

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：

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspx>

注釈： 5.4 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 ACR1251 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 :

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa379804%28v=vs.85%29.aspx>

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、**5.2.4.2** セクション - PICC コマンドそしてセクション **5.2.2** - 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 ACR1251 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。。

参照のウェブサイト:

<http://msdn.microsoft.com/en-us/library/windows/desktop/aa379475%28v=vs.85%29.aspx>

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

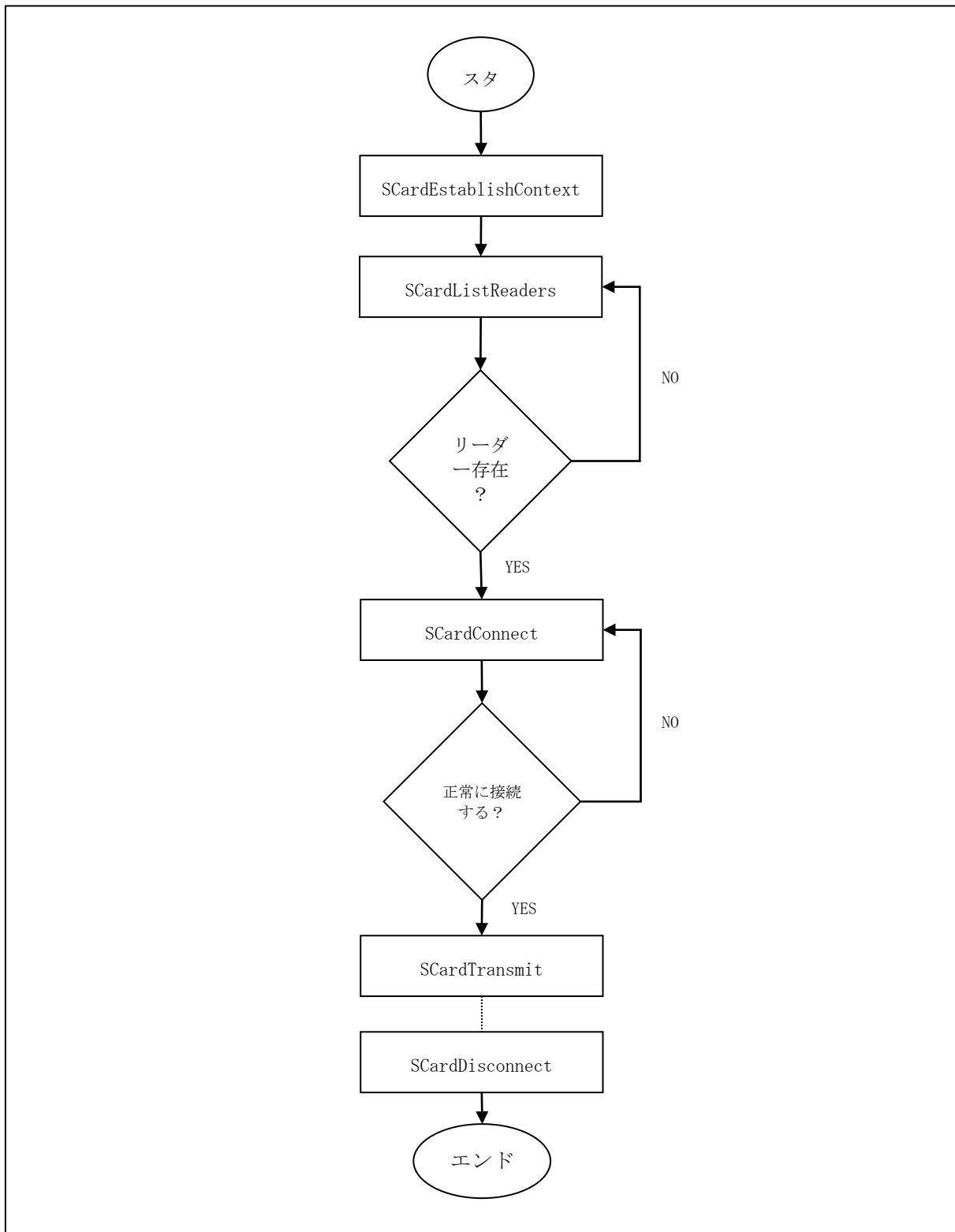


図2 :ACR1251U-A1 の APDU の流れ

5.1.8. ダイレクトコマンド（Escape Command）の流れ

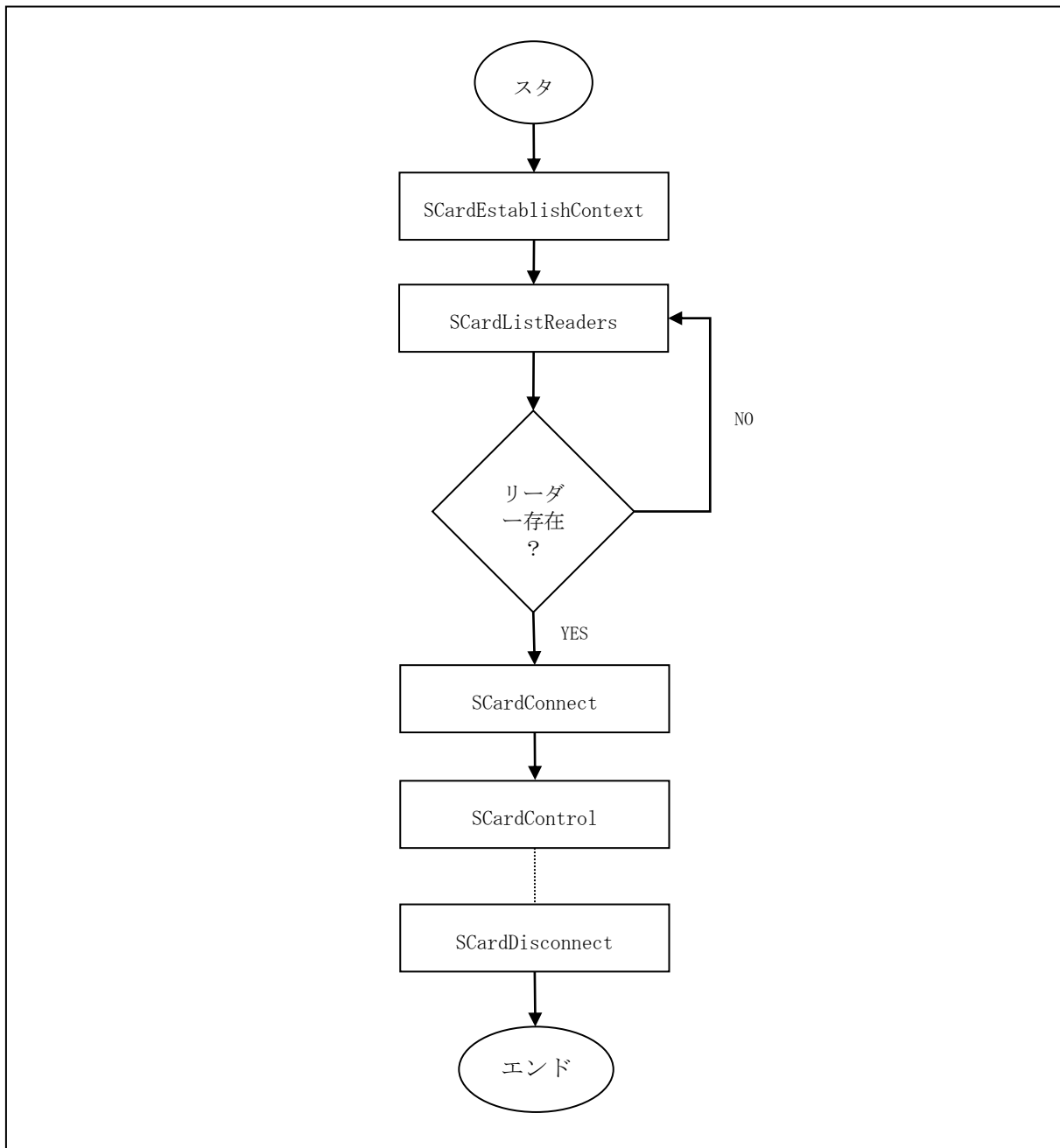


図3 :ACR1251U-A1 直接コマンド（Escape Command）の流れ

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 ドライバに送られます。

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

バイト	Value	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 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		Length
	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
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)
 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) 2K	00 01: MIFARE Classic 1K	00 38: MIFARE Plus® SL2
	00 02: MIFARE Classic 4K	00 39: MIFARE Plus® SL2 4K
	00 03: MIFARE Ultralight®	00 30: Topaz と 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]: 定義していないタグ
	00 37: MIFARE Plus® SL1 4K	

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

バイト	Value	標記	説明					
0	3Bh	最初のヘッダー	-					
1	8N	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 から 3 + N	XX	T1	歴史的なバイト					
	XX XX XX	Tk	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 border="1" data-bbox="762 1218 1370 1503"> <thead> <tr> <th>Byte1-4</th> <th>Byte5-7</th> <th>Byte8</th> </tr> </thead> <tbody> <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> </tbody> </table>	Byte1-4	Byte5-7	Byte8	ATQB のアプリケーションデータ	ATQB からのプロトコル情報バイト
Byte1-4	Byte5-7	Byte8						
ATQB のアプリケーションデータ	ATQB からのプロトコル情報バイト	高いニブル =ATTRIB コマンドの MBLI ; 下位ニブル (RFU) =0						
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 1 : MIFARE® DESFire® 的 ATR = {3B 81 80 01 80 80h} // 6 bytes of 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 に準拠していることを区別します。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 UID の APDU 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)

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

例えば 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	XXh	間違った長さ (間違ったナンバー-Le : 'XX'は正確な数字を表す) 、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};
```

5.2.3. PCSC 2.0 パート 3 の APDU コマンド (2.02 もしくはもっと新しいバージョン)

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 = 透明交換

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	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、透明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透明セッション内の 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	データ長さ	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	透明セッションを開始する
82h	透明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	長さ (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	長さ (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	長さ (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	長さ (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	長さ (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 のタイマー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、リーダーは 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)

5.2.3.2.7. 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	長さ (1 バイト)	数値		
		タグ	長さ	数値
FF 6Dh	パール	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

要求のパラメータ	タグ	長さ
ICC の通信速度	05h	00h
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 タイプのパラメータ—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	長さ (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 タイプのパラメータ—1 - 4	0Ah	04h

5.2.3.3. 透明交換のコマンド (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。

透明交換のコマンド

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-送信と受信
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

透明交換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	応答メッセージの状態コード
97h	ICC 応答
FF 6Dh	IFD パラメーター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	長さ (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	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に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	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

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	長さ (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	長さ (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。

何のタイマー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	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	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	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ださい。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	長さ (1 バイト)	数値
97h	DataLen	応答データ (N バイト)

5.2.3.4. プロトコルを切り替えるコマンド (Switch Protocol Command)

このコマンドは、プロトコルと透明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。

プロトコルを切り替えるコマンド

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	パラメーターを取得する
FF 6Eh	パラメーターを設定する

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 パラメーター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

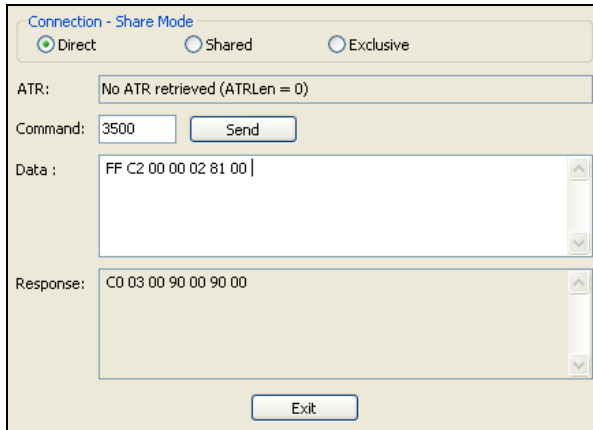
タグ	長さ (1 バイト)	数値	
		バイト0	バイト1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa 他 – RFU	00h – 層分離がない場合 02h – 2層に切り替える 03h – 3層に切り替えるまたは活性化する 04h – 4層に活性化する 他 - RFU

5.2.3.4.2. PCSC 2.0 パート 3 の例

1. 透明セッションを開始する

コマンド : **FF C2 00 00 02 81 00**

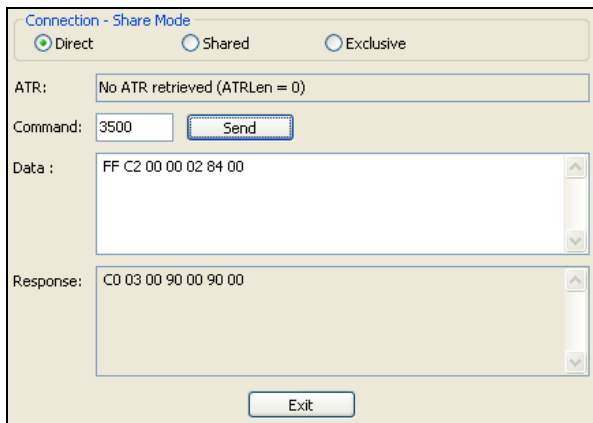
応答 : **C0 03 00 90 00 90 00**



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**

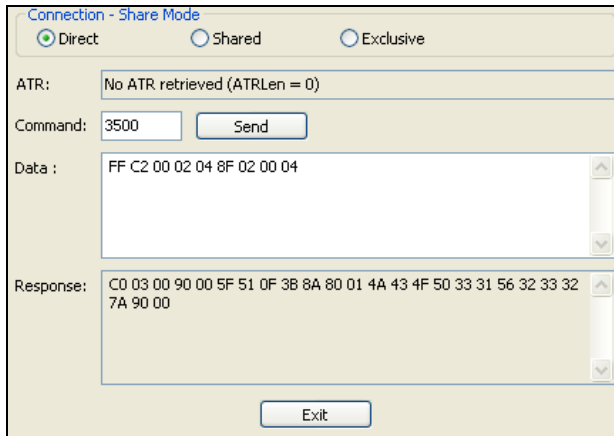


3. ISO 14443-4A 有効。

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)

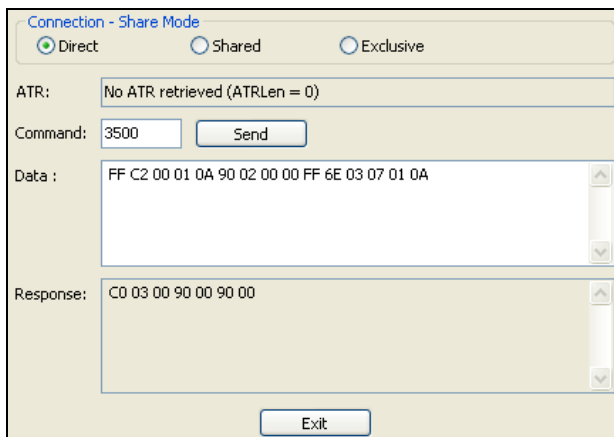
C0 3 00 90 00 5F 51 [ATR] 90 00



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**

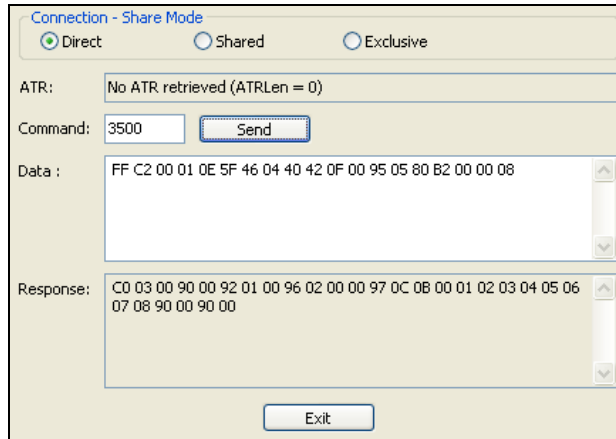
応答 : **C0 03 00 90 00 90 00**



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**

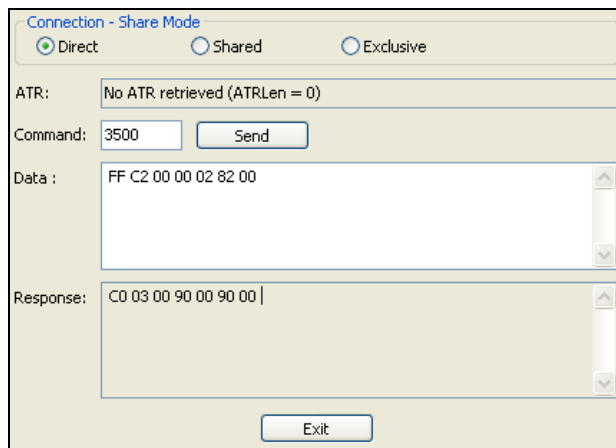
応答 : **C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [カードの応答] 90 00**



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**



5.2.4. MIFARE Classic 1K/4K メモリカードの PICC コマンド (T=CL エミュレーション)

5.2.4.1. 認証キーのダウンロード (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 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ます。リーダーは二種の認証キーのアドレスが提供されている：失いやすいキーのアドレスと失いにくいキーのアドレス。

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 = キーが失いやすい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 – 01Fh = キーを保存するための失いにくいキーのメモリ。キーは永遠にリーダーに保存されます。リーダーが PC から切断された場合でも、リーダーのメモリ内に保持されます。その失いやすいキーのメモリには最大 3 2 個キーをストアできる。

注釈：デフォルト値は FF FF FF FF FF FFh です。

キー 6 バイト。

リーダーにロードされるキーの値。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 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 :

// 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}

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

このコマンドはリーダーにストアされているキーで MIFARE Classic® カード (PICC) を認証する時に使われます。二種の認証キーを使われる：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失いやすくないキーのメモリ。リーダーが PC から切断された時、キーが消えます。二つの失いやすいキーを提供しています。これらは異なるセッションにセッション鍵として使用できる。

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 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 :

// {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}

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まれている)	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
..
..
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まれている)	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まれている)	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3 : MIFARE Classic 4K 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, 廃止される

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



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 ビット
または
64 バイト

表4 : MIFARE Ultralight カードのメモリマップ

注釈 : 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. バイナリブロックの読み取る (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。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

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	更新していないバイト

その中 :

ブロックの番号 1 バイト。開始ブロック

更新されていない1 バイト。

更新していない MIFARE Classic 1K/4K のバイトは 16 の倍数です ; 更新していない MIFARE Ultralight のバイトは 4 の倍数です。

更新していない MIFARE Ultralight のバイトは最大に 16 です。

更新していない MIFARE Classic 1K カードのバイトは最大に 48 です。(複数のブロックモード ; 3 個の連続のブロック)

更新していない MIFARE Classic 4K カードのバイトは最大に 240 です。(複数のブロックモード ; 15 個の連続のブロック)

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)

注釈 : 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 :

// バイナリブロックの 04h から 16 バイトを読み出す (MIFARE Classic 1K/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	更新していないバイト	データブロック (16 バイトの倍数)

その中：

ブロックの番号 1 バイト。更新していない開始ブロック

更新していないバイト： 1 バイト。

更新していない MIFARE Classic 1K/4K のバイトは 16 の倍数です；更新していない MIFARE Ultralight カードのバイトは 4 の倍数です。

更新していない MIFARE Classic 1K カードのバイトは最大に 48 です。（複数のブロックモード；3 個の連続のブロック）

更新していない MIFARE Classic 4K カードのバイトは最大に 240 です。（複数のブロックモード；15 個の連続のブロック）

ブロックデータ 16 の倍数 + 2 バイト、または 6 バイトバイナリブロックに書き入れていないデータ。

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）

注釈： 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る。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 の応答コード (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 **04h** 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に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:

ブロックの番号 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しか実行されません。

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しか実行されません。

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)

Read Value Block コマンドは数値ブロックの数値を入手するために使われます。数値ブロックしか実行されません。

Read Value Block 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	Value {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}

Value			
MSB			LSB
FFh	FFh	FFh	FCh

例 2 : Decimal 1 = {00h, 00h, 00h, 01h}

Value			
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	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。

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 にストアします。

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

// 数値ブロック 05h を読み取ります。

APDU = {FF B1 00 05 04h}

数値ブロック 05h 中の数値を数値ブロック 06h にコピーします。

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。ACR1251U-A1 カードリーダーは ISO 7816-4 の APDU および応答を交換することによって、ISO14443-4 基準に準拠しているカードと通信します。ACR1251U-A1 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。詳しい情報が 5.2.2 セクション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 部分のコマンド	-	-	-	-	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	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 インターフェースと接続します。
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コマンドは、PCSC タグおよび MIFARE をアクセスするコマンドと違って、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 (長さのバイトから始まる)

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

RES = [IDM (8bytes)] 90 00h

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

3. FeliCa コマンドをアクセスします。

メモリブロックデータの読み取りを例として :

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**

RES = Memory Block Data

5.3. 非接触スマートカード プロトコル

5.3.1. ACOS6-SAM コマンド

このセクションでは、SAM 固有のコマンドについて説明します。

注：ACOS6-SAM のコマンドとシナリオの詳細については、ACOS6-SAM リファレンスマニュアルをもらうには ACS の担当者にお問い合わせください。

5.3.1.1. キー生成 (Generate Key)

このコマンドは、クライアントカードのシリアル番号などの偏差データから ACOS3 / 6 カードまたは他のカードにロードするための多様なキーを生成するために使用されます。このコマンドは、クライアントカードの発行目的で使用されます。

APDU	説明
CLA	80h
INS	88h
	00h 8 バイトのキーを生成する
P1	01h 16 バイトのキーを生成する
	02h 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、必ず 08h
6A	83h	参照されたキーレコードが EF2 に見つかりません
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 はブロックされています。EF2 がブロックされている
69	83h	使用カウンタはゼロです。
69	82h	セキュリティ条件が満たされていない
6A	87h	参照されたマスターキーは 3-DES 暗号化できません
61	08h	コマンドが完了し、結果を得るために GET RESPONSE を発行する

5.3.1.2. 主要データの多様化（または読み込み）（Diversify (or Load) Key Data）

このコマンドは、キーを多様化してロードすることによって暗号化操作を実行するように SAM カードを準備します。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									
		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	端末キー
P1		-	0	0	0	0	1	0	0	カードキー
		-	0	0	0	0	1	0	1	一括暗号化キー（多様化している）
		-	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
マスターキーの索引： Bit7 : 1 =現在の EF2 のローカルキー。 P2 0 =グローバル KEY EF2 Bit6-Bit5 : 00b - RFU Bit4-Bit0 : キーインデックス										
P1 = 1~4、P3 = 8/16（もし Algo が AES の場合、P3 = 8/16） P1 = 5 の場合、P3 = 0 P3 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 がブロックされているか、または EF2 がブロックされている
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 の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

5.3.1.3. 暗号化する (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. 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								
	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

APDU	説明
	<p>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。</p> <p>1 - セッションキー-Ks でデータを暗号化する 2 - 多様なキー-Sc でデータを暗号化する 3 - 一括暗号鍵でデータを暗号化する 0 - ENC (Sc, Ks) を返す</p> <p>P2</p> <p>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</p>
	<p>P3 < 128</p> <p>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</p> <p>P3</p> <ul style="list-style-type: none"> - 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
	<p>プレーンテキスト</p> <p>P2 b6 = 1 の場合、DATA 形式は次のようになります。</p> <ul style="list-style-type: none"> • プレー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 <p>P1 = A1h、暗号化は MIFAREPlus コマンド用です</p> <ul style="list-style-type: none"> • MFP コマンドが値操作コマンドである場合、DATA 形式はコマンドコード (1 バイト) +ブロック番号 (2/4 バイト) +値 (4 バイト) でなければなりません。 • MFP Command が Proximity Check の場合、DATA フォーマットはコマンドコード (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 Read の場合、DATA 形式はコマンドコード (1 バイト) +ブロック番号 (2 バイト) • MFP コマンドが Write である場合、DATA 形式はコマンドコード (1 バイト) +ブロック番号 (2 バイト) +プレーンテキスト <p>データ</p> <p>P1 = A3h、</p> <ul style="list-style-type: none"> • 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

5.3.1.4. 解読 (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									
		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 Decryption
P1		-	1	0	0	1	1	0	-	MIFARE DESFire EV1 解読化する
		-	0	1	0	0	1	0	-	MIFARE Plus 解読
		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 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。

- | | |
|----|--|
| P2 | <ol style="list-style-type: none"> 1 - セッションキー-Ks でデータを解読化する 2 - 多様なキー-Sc でデータを解読化する 3 - 一括暗号鍵でデータを暗号化する 0 - DEC (Sc, Ks)を返す |
|----|--|



APDU	説明
P3	<p>P3 < 128</p> <p>P1 = A5h の場合、P3 = 16/32/48</p> <p>P1 のビット 3 が 1 に等しくない場合</p> <ul style="list-style-type: none"> - 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
	<p>暗号文</p> <p>P1 = A5h の場合、DATA は暗号化されたテキスト</p> <p>P2 b6 = 1 の場合、DATA 形式は次のようになります。</p> <p>データ</p> <ul style="list-style-type: none"> • 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

5.3.1.5. 認定準備 (Prepare Authentication)

このコマンドは、ACOS 3/6 カードまたは MIFARE Ultralight C /MIFARE DESFire カード/ MIFARE Plus カードの 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) 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）が準備できていません（このキーを生成するために Diversify を使用してください）
69 82h	セキュリティ条件が満たされていない
61 10h	コマンドが完了し、結果を得るために GET RESPONSE を発行する

5.3.1.6. 認証確認 (Verify Authentication)

このコマンドは、ACOS 3/6、MIFARE Ultralight C、MIFARE DESFire/MIFARE DESFireEV1 または MIFARE Plus カードを端末に照合するために使用します。セッション鍵 Ks も内部的に生成される。

APDU	説明
CLA	80h
INS	7Ah
P1	00h – 3DES (P2 = 0) 01h – DES (P2 = 0) 02h – 3KDES (P2 = 0, ACOS3) 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 UL-C / DESFire / DESFire EV1 の認証を確認する 02h - MIFARE Plus の認証を確認する
P3	08h - (P2 = 0、P2 = 1、セッションキーは DES / 3DES) 16 時間 - (P2 = 1、セッションキーは 3KDES / AES) 16h - (P2 = 02、MIFARE Plus リターンデータ ek (RndA ')) 32h - (P2 = 02、MIFARE Plus リターンデータ ek (TI + PICCcap2 + PCDCap2))
データ	ACOS 3/6:DES (Ks, RND _T) MIFARE DESFire/ DESFire EV1 リターンデータ : ek (RndA ') MIFARE Plus 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

5.3.1.7. ACOS 問い合わせアカウントの確認 (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																																																																																	
	<table border="1"> <thead> <tr> <th>b7</th> <th>b6</th> <th>b5</th> <th>b4</th> <th>b3</th> <th>b2</th> <th>b1</th> <th>b0</th> <th>説明</th> </tr> </thead> <tbody> <tr> <td>-</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>-</td> <td>0</td> <td>-</td> <td>ACOS INQ_AUT 無効</td> </tr> <tr> <td>-</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>-</td> <td>1</td> <td>-</td> <td>ACOS INQ_AUT 有効</td> </tr> <tr> <td>-</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>-</td> <td>-</td> <td>ACOS INQ_ACC_MAC 無効</td> </tr> <tr> <td>P1</td> <td>-</td> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>1</td> <td>-</td> <td>ACOS INQ_ACC_MAC 無効</td> </tr> <tr> <td></td> <td>0</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>0</td> <td>3DES</td> </tr> <tr> <td></td> <td>0</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>1</td> <td>DES</td> </tr> <tr> <td></td> <td>1</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>0</td> <td>3K DES (ACOS3 のみ)</td> </tr> <tr> <td></td> <td>1</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>-</td> <td>1</td> <td>AES (ACOS3 のみ)</td> </tr> </tbody> </table>	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	P1	-	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)
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																																																																										
P1	-	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KS または KACCT は準備ができていません。KACCT を生成するには DIVERSIFY コマンドを使用します。該当する場合は、「認証の準備」を使用して KS 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



5.3.1.8. ACOS アカウントトランザクションの準備 (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								
	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
P1	0	-	-	-	-	-	-	0	3DES
	0	-	-	-	-	-	-	1	DES
	1	-	-	-	-	-	-	0	3K DES (ACOS3 のみ)
	1	-	-	-	-	-	-	1	AES (ACOS3 のみ)
P2	E2h:Credit								
	E6h:Debit								
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KS または KACCT は準備ができていません。KACCT を生成するには DIVERSIFY コマンドを使用します。該当する場合は、「認証の準備」を使用して KS を生成します。
61 0Bh	コマンドが完了し、結果を得るために GET RESPONSE を発行する

5.3.1.9. 借方証明書を確認する (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								
	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
P1	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、必ず 14h
6A 83h	ACOS キー-KS または KACCT は準備ができていません。KACCT を生成するには DIVERSIFY コマンドを使用します。該当する場合は、「PREPARE AUTHENTICATION」を使用して KS 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

5.3.1.10. キー取得 (Get Key)

このコマンドを使用すると、現在の SAM のキーファイル (SFI = 02h) から別の ACOS6 / ACOS6-SAM に、キーの多様化の有無にかかわらず安全にキーを注入できます。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。

キーコマンドを取得することにより、現在の SAM のキーファイル (SFI = 02h) からキーの多様化を伴う ACOS7 / 10、MIFARE DESFire、MIFARE DESFire EV1 または MIFARE Plus カードへの安全なキーインジェクションも可能になります。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。

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。カードに このビットを設定すると、アクティベーション後の任意の状況下でキーファイルの Read Record コマンドが無効になります。

このコマンドを実行する前に、相互認証の相互認証手順 (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 相互認証手順を使用して、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		
ACOS カードセットキーのキーを取得する			
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	レスポンスデータは、MIFARE PlusCard 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			
P1	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 カードとは異なる	16 バイト
81h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードとは異なる	16 バイト
82h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードとは異なる	24 バイト
88h	MIFARE DESFire	MIFARE DESFire カードとは同じ	16 バイト
89h	MIFARE DESFire EV1	MIFARE DESFire EV1 カードとは同じ	16 バイト



APDU	説明
8Ah	MIFARE DESFire EV1 MIFARE DESFire EV1 カードとは同じ 24 バイト
P2	SAM のキーID (変更のための新しいキー)
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	<p>P1 = 00h の場合、コマンドデータは RNDTarget</p> <p>P1 = 01 / 02h の場合、コマンドデータは RNDTarget +ターゲットカードのシリアル (またはバッチ) 番号です</p> <p>P1 = 03h の場合 P1 = 03h</p> <ul style="list-style-type: none"> - 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) <ul style="list-style-type: none"> - BNr (2 バイト) <p>P1 = 80/81/82/88/89/8Ah :</p> <ul style="list-style-type: none"> - 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)

*異なるキーの変更は、変更するキーが認証キーと異なることを示します。同じキーを変更することは、変更するキーが DESFire カードの認証キーと同じであることを示します



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	成功、GET RESPONSE を使用して結果を取得する

5.4. 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。

5.4.1. 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 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 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 31 55 5F 56 32 30 34 2E 30

ファームウェアのバージョン番号 (HEX) = 41 43 52 31 32 35 31 55 5F 56 32 30 34 2E 30

ファームウェアのバージョン番号 (ASCII) = "ACR1251U_V204.0"

5.4.2. LED 制御 (LED Control)

このコマンドは LED 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

5.4.3. 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 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

5.4.4. ブザー制御 (Buzzer Control)

このコマンドはブザーの出力を制御するために使用されます。

Buzzer 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
Buzzer Control	E0h	00h	00h	28h	01h	ブザーの持続時間

その中：

ブザーの持続時間 1 バイト
 00h = OFF
 01 - FFh = 持続時間 (単位 : 10 ms)

Buzzer 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	00h



5.4.5. ブザーの状態 (Buzzer Status)

このコマンドはブザーの状態を検査するために使用されます。

Buzzer Status コマンドフォーマット (5 バイト)

コマンド	CLA	INS	P1	P2	Lc
Buzzer Status	E0h	00h	00h	28h	00h

Buzzer 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	00h

5.4.6. LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW205 以下)

このコマンドは LED とブザーの状態指示器を設定するために使用されます。

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。205 以下のファームウェアのみ適応します。

Set LED and Buzzer 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 and Buzzer 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	SAM 活性化状態の LED	SAM インタフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
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 インタフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
Bit 3	カードの挿入と除去のイベントブザー	カードの挿入または取り外しのイベントが検出されるたびに、ビーブ音をします。(PICC) 1 = 有効 ; 0 = 無効
Bit 4 – 6	RFU	RFU
Bit 7	カードに操作する時 LED が点滅します。	カードがアクセスされる時 LED が点滅します。

注：操作のデフォルト値 = 8Fh

Set LED and Buzzer 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

5.4.7. LED とブザーの状態指示器を読み取る (Read LED and Buzzer Status Indicator Behavior) (FW205 以下)

このコマンドは LED とブザーのデフォルト操作を読み取る時に使われます。

注: 205 以下のファームウェアのみ適応します。

Set LED and Buzzer 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 and Buzzer Status Indicator Behavior	E0h	00h	00h	21h	00h

Read LED and Buzzer 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	SAM 活性化状態の LED	SAM インタフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
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 インタフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
Bit 3	カードの挿入と除去のイベントブザー	カードの挿入または取り外しのイベントが検出されるたびに、ビーブ音をします。(PICC) 1 = 有効 ; 0 = 無効
Bit 4 – 6	RFU	RFU
Bit 7	カードに操作する時 LED が点滅します。	カードがアクセスされる時 LED が点滅します。

注: 操作のデフォルト値 = 8Fh。

5.4.8. LED とブザーの状態指示器を設定する (Set LED and Buzzer Status Indicator Behavior) (FW206 以降)

このコマンドは PICC インターフェースの LED とブザーの状態指示器を設定するために使用されます。

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。206 以降のバージョンのファームウェア適応します。

Set LED and Buzzer Status Indicator Behavior コマンドフォーマット (6 bytes)

コマンド	CLA	INS	P1	P2	Lc	データイン
Set LED and Buzzer 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。	PICC カードが読められる時、LED が点滅します。 1 = 有効 ; 0 = 無効
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 インターフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
Bit 3	カードの挿入と除去のイベントブザー	カードの挿入または取り外しのイベントが検出されるたびに、ビープ音をします。(PICC) 1 = 有効 ; 0 = 無効
Bit 4	RFU	RFU
Bit 5	カードリーダーの電気ブザー	カードリーダーがビープ音をします。 1 = 有効 ; 0 = 無効
Bit 6	オプションの色 (緑)	状態の変更を示すための緑の LED 1 = 有効 ; 0 = 無効
Bit 7	オプションの色 (赤)	状態の変更を示すための赤い LED 1 = 有効 ; 0 = 無効

注：操作のデフォルト値 = 7Fh。 NFC モードを変更した場合、ブザー音を無効にしません。

Set LED and Buzzer 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

5.4.9. LED とブザーの状態指示器を読み取る（Read LED and Buzzer Status Indicator Behavior）（FW206 以降）

このコマンドは PICC インターフェースの LED とブザーの状態指示器を読み取る時に使われます。

注：206 以降のバージョンのファームウェア適応します。

Read LED and Buzzer 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 and Buzzer Status Indicator Behavior	E0h	00h	00h	21h	00h

Read LED and Buzzer 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。	PICC カードが読められる時、LED が点滅します。 1 = 有効 ; 0 = 無効
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 インタフェースの活性化状態を表示する 1 = 有効 ; 0 = 無効
Bit 3	カードの挿入と除去のイベントブザー	カードの挿入または取り外しのイベントが検出されるたびに、ビープ音をします。(PICC) 1 = 有効 ; 0 = 無効
Bit 4	RFU	RFU
Bit 5	カードリーダの電気ブザー	カードリーダがビープ音をします。 1 = 有効 ; 0 = 無効
Bit 6	オプションの色 (緑)	状態の変更を示すための緑の LED 1 = 有効 ; 0 = 無効
Bit 7	オプションの色 (赤)	状態の変更を示すための赤い LED 1 = 有効 ; 0 = 無効

注：操作のデフォルト値 = 7Fh。 NFC モードを変更した場合、ブザー音を無効できません。

5.4.10. 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 PCSC Escape コマンドのインターフェースで送信されます。エネルギーを節約するために、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。

注釈：この設置は失いやすいキーのメモリに保存されます。

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 が見つからない場合は、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	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	RFU	RFU
Bit 7	ISO 14443-A 4 パートを強制的に実行します。	1 = 有効 ; 0 = 無効

注：ポーリング設置のデフォルト値 = 8Fh



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、省エネルギー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。省エネルギー状態で *Idle* 消費電流は 60 mA です；非省エネルギー状態で *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。

5.4.11. 自動的な PICC のポーリングを読む (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。

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 が見つからない場合は、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	RFU	
Bit 7	ISO 14443-A 4 パートを強制的に実行します。	1 = 有効 ; 0 = 無効

注 : ポーリング設置のデフォルト値 = 8Fh

5.4.12. 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。

注釈：この設置は失いやすい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

5.4.13. PICC 操作のパラメータを読取る (Read PICC Operating Parameter)

このコマンドは PICC 操作のパラメータ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

操作パラメータ (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	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

5.4.14. 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 :

- 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。
- 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。

5.4.15. 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	CL A	INS	P1	P2	Lc
自動的な PPS を読み取る (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

5.5. NFC P2P についてのコマンド

5.5.1. 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。

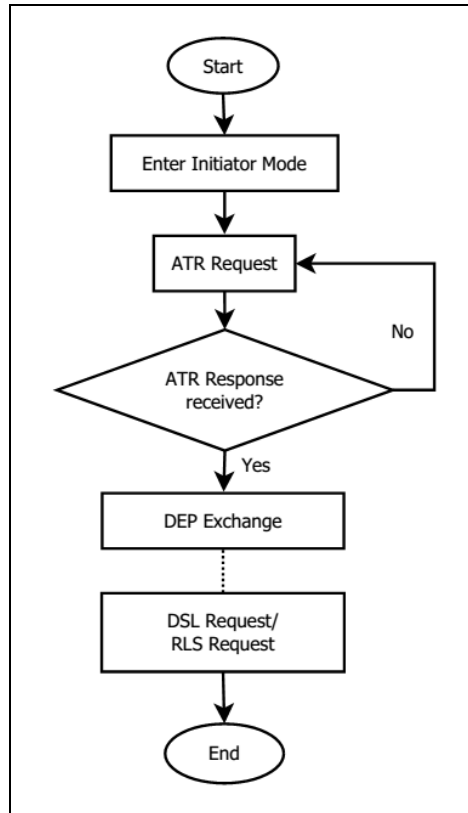


図4 :イニシエータモードでの P2P 流れ

5.5.1.1. 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。リーダーがイニシエータに入ると、SNEP メッセージを成功に交換するために、5 回の試行チャンスがあります (250 ミリ秒間隔のたびに)。

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：

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

5.5.1.2. 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	Speed

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	Speed

その中 :

- NfcMode** 1 バイト。NFC デバイスモード
 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
- 速度** 1 バイト。通信速度
 01h = 106 Kbps
 02h = 212 Kbps
 03h = 424 Kbps

5.5.1.3. 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 P2P ターゲットモードのデバイスをポーリングするために使用される。

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	Len	11h	Mode (1バイト)	Speed (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 パラメーター	
			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	Len	ATR 応答 (Len バイト)

その中：

- Mode** 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 パラメーター

5.5.1.4. 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	Len	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	Len	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 P2 P 通信に使われます。

5.5.1.5. 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。

5.5.1.6. 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。デバイス識別

RLS 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。

5.5.1.7. PSL リクエストを送信する (Send PSL Request)

このコマンドは PSL 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。

PSL 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			
PSL request	E0h	00h	00h	46h	04h	11h	DID (1 バイト)	BRS (1 バイト)	FSL (1 バイト)

その中 :

- DID** 1 バイト。デバイス識別
- BRS** 1 バイト。イニシエータとターゲットデバイスのビットレート
- FSL** 1 バイト。フレームの長さ

PSL 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	01h	DID (1 バイト)

その中 :

- DID** 1 バイト。デバイス識別

5.5.2. 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 P 2 P の流れを示します。

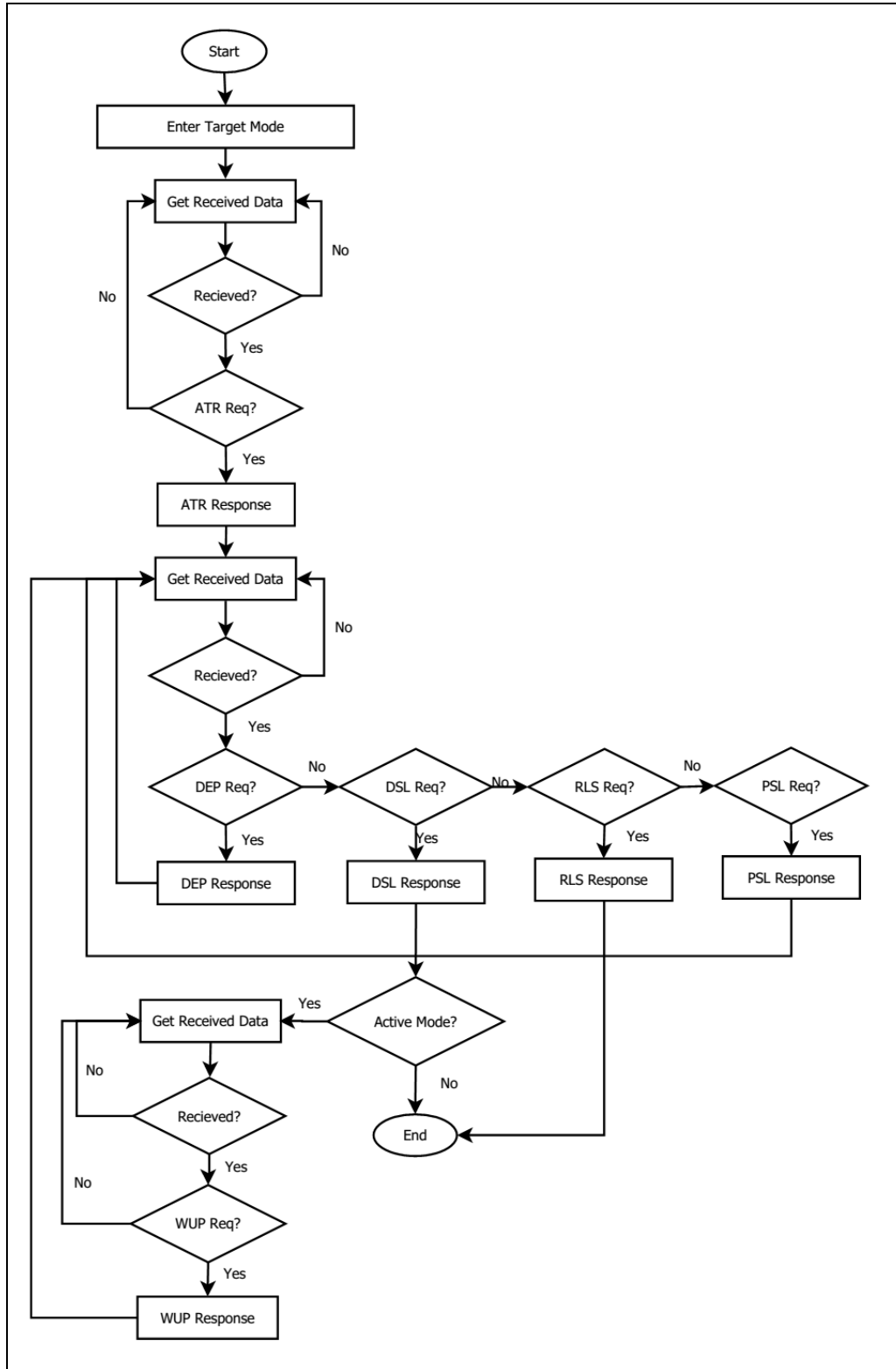


図5 : ターゲットモードでの P2P 流れ

5.5.2.1. 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 μ s ; ターゲットタイムアウトのデフォルト値 = 00 C8h (200 * 100 μ s = 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。イニシエータモードのタイムアウト (単位 = 100 ms) 。

5.5.2.2. 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

コマンド	CLA	INS	P1	P2	Lc
Enter Target Mode	E0h	00h	00h	51h	00h

ボーレート 106 Kbps とパッシブモードでターゲットモードを設定する

または

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

5.5.2.3. 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	Len	LLCP パラメータ — (N バイト)

その中：

LLCP パラメータ— 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。

5.5.2.4. 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	Len	PFB (1 バイト)	DEP Message (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。

ATR 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。

5.5.2.5. 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。

5.5.2.6. 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。

5.5.2.7. 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 パラメーター。

FSL 1 バイト。FSL パラメーター。

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。

5.5.2.8. 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。



5.5.2.9. 受信したデータを入力する (Get Received Data)

このコマンドでイニシエータモードのデバイスからメッセージを取得します。

Get Received Data コマンドフォーマット

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

Get Received Data 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	Len	受信したメッセージ (N バイト)

5.6. ACR122U 互換性のあるコマンド

5.6.1. 二色の LED とブザー制御 (Bi-color LED and Buzzer Control)

このコマンドは二色の LED とブザー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)
Bi-color LED and Buzzer Control	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 とブザー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

Bi-color LED Blinking Duration Control コマンドフォーマット (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	ブザーの応答

その中：

バイト 3 ブザーの応答。LED が点滅周期にブザーの状態を制御します。

00h = ブザーを有効にしない。

01h = T1 周期にブザーを有効します。

02h = T2 周期にブザーを有効します。

03h = T1 周期と T2 周期にブザーを有効します。

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	63	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	保留	



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= $T1 / (T1 + T2)$ 。

1. ブザーだけを制御したい場合、**P2"LED 状態制御"**を 0 に設置すればいいです。
2. ブザーが動作させる場合、**"繰り返し回数"**が 0 より大きくなければなりません。
3. **LED** だけ制御したい場合は、**"パラメーターのブザー応答"**を 0 に設置すればいいです。



5.6.2. 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 31 55 5F 56 32 30 34 2E 30h = ACR1251U_V204.0 (ASCII)

5.6.3. PICC 操作のパラメータを入手する (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

PICC 操作パラメータ

ビット	パラメータ	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 請求を送信します。	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

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

このコマンドはリーダーの PICC 操作のパラメータ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	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

PICC 操作パラメータ

ビット	パラメータ	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 請求を送信します。	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



Appendix 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	Length	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”

Android は Google Inc.の商標です。

Microsoft は Atmel また子会社がアメリカまたはほかの国の登録商標です。

MIFARE、MIFARE Classic、MIFARE DESFire および MIFARE Ultralight は NXP B.V.の登録商標で、ライセンスに従って使用されます。