Advanced Card Systems Ltd.
Card & Reader Technologies

ACR1255U-J1

ACS 安全 Bluetooth™ NFC リーダー

リファレンスマニュアル　V1.13

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

改定履歴

リリス日付	改訂説明	バージョン
2015/09/07	• 初回発布	1.00
2016/02/02	• 2.0 節更新ー特性 • セクション 4.1.2 更新 – 電池寿命 • セクション 4.5.1 更新 – キー • セクション 4.5.3 更新 – LED ステータス • セクション 4.5.4 更新 – ブザー • セクション 5.3 更新 – 認証 • セクション 6.7.2 更新 – シリアルナンバー取得 • セクション 6.7.16 更新 – Ｓｌｅｅｐモードオプション • セクション 6.6.18 更新 – Tx パワーの読み取り	1.01

リリス日付	改訂説明	バージョン
2016/06/06	• セクション 5.6.5 更新–カードステータス通知コマンド • セクション 6.2.1 更新 – ATR 生成 • 6.4.1 節更新ー認証キーアップロード • セクション 6.7.6 更新 – LED とブザーステータスインジケータ設定 • セクション 6.7.7 更新 – LED とブザーステータスインジケータ読み取り • セクション 6.7.8 更新 – 自動 PICC ポーリング設定 • セクション 6.7.10 更新 – PICC 操作パラメーター設定 • セクション 6.7.11 更新 – PICC 操作パラメーター読み取り • セクション 6.7.12 更新 – 自動 PPS 設定 • セクション 6.7.13 更新 – 自動 PPS 読み取り • セクション 6.7.16 更新 – Ｓｌｅｅｐモードオプション • セクション 6.7.17 更新 – Tx パワーコマンド変更 • セクション 6.7.18 更新 – Tx パワーの読み取り • セクション 6.7.19 追加 – 顧客マスターキー書き換え	1.02
2016/09/16	• 商品マーケティング名更新	1.03

リリス日付	改訂説明	バージョン
2017/01/10	• コマンドとサンプルの更新 • 相互認証通信例の追加（セクション 5.7.1） • 表 5 の値の更新 • PIN パスコード削除説明 • 2.0 節更新ー特性 • セクション 4.1.2 更新 – 電池寿命 • セクション 5.6.5 更新–カードステータス通知コマンド • セクション 6.4 更新 – PC/SC 2.0 パート 3 APDU コマンド • セクション 6.7.4 更新 – LED の色 • セクション 6.7.17 更新 – Tx パワーコマンド変更 • セクション 6.7.18 更新 – Tx パワーの読み取り	1.04
2017/10/02	• 表 5 の更新：サービスハンドルと UUID メッセージリスト • セクション 5.3 更新 – 認証 • セクション 5.7 更新：相互認証 • セクション 6.7 追：FeliCa タグへアクセス	1.05
2017/12/28	• 更新の製品写真	1.06
2018/05/24	• 更新 5.2 セクション – 配置ファイル選択 • 更新 5.6 セクション – ブルートゥース通信協議	1.07

リリス日付	改訂説明	バージョン
2018/09/13	• 書式設定更新 • 更新 2.0 セクションー特性 • 更新 3.0 セクション – システムブロック図 • 更新 5.6.4 セクション – APDU コマンド • 更新 5.6.6 セクション – ハードウェアエラー応答 • 更新 5.7.6 セクション – RDR_to_SPH_ACK (エラー管理) • 更新 6.3.5 セクション– ブザー制御（Buzzer Control） • 更新 6.3.6 セクション – LED とブザーステータスインジケータ設定 • 更新 6.3.8 セクション – 自動 PICC ポーリング設定 • セクション 6.3.10 更新 – PICC 操作パラメーター設定 • セクション 6.3.11 更新 – PICC 操作パラメーター読み取り • セクション 6.3.12 更新 – 自動 PPS 設定 • 更新 6.3.16 セクション – Ｓｌｅｅｐモード設置オプション • 更新 6.3.17 セクション – Ｓｌｅｅｐモード読み取りオプション • 更新 6.3.18 セクション – Tx パワー設置 • 更新 6.3.19 セクション – Tx パワーの読み取り • 追加 6.3.21 セクション – バッテリー残量取得	1.08
2018/10/08	• 更新 6.3.17 セクション – Ｓｌｅｅｐモード読み取りオプション • 更新 6.3.18 セクション – Tx パワー設置 • 追加 6.3.21 セクション – バッテリー残量取得	1.09
2019/06/14	• 商品マーケティング名更新 • 更新 5.0 セクション – ソルト設計 • 更新 5.1.3 セクション – 認証	1.10

リリス日付	改訂説明	バージョン
2019/07/08	• 更新 5.1.3 セクション – 認証	1.11
2019/08/28	• 更新 6.2.4.7 セクション - コピー値ブロック (Copy Value Block) • 更新 6.3.8 セクション - 自動 PICC ポーリング設置（Set Automatic PICC Polling） • 更新 6.3.9 セクション - 自動 PICC ポーリング設置（Set Automatic PICC Polling） • 更新 6.3.17 セクション – Ｓｌｅｅｐモード読み取りオプション	1.12
2020/08/07	• 追加 6.2.2.2 セクション – PICC データ取得 • 追加 6.3.22 セクション – PICC タイプ読み取り • 更新 6.1 セクション – PC/SC API • セクション 6.3.6 更新 – LED とブザーステータスインジケータ設定 • セクション 6.3.7 更新 – LED とブザーステータスインジケータ読み取り • 更新 6.3.9 セクション – 自動 PICC ポーリング読み取り	1.13

目次

図示一覧表

チャート一覧表

1.0. 紹介

ACR1255U-J1 ACS セキュリティ Bluetooth™NFC リーダーは、コンピュータ/モバイルデバイスと非接触型スマートカード/NFC デバイスとの間の通信インターフェースです。ACR1255U-J1 が様々のカードにコンピュータ/スマートカードリーダーから統一されたインタフェースを確立します。スマートカードのたくさんの特性を持っているため、コンピュータ·ソフトウェア·プログラマがスマートカードの操作を詳しく了解する必要をなくなります。多くの場合、これらの操作の技術的詳細はスマートカードシステムの実装と関係がありません。

1.1. シンボルと略語

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―タ―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―タ―

表1 ：シンボルと略語

2.0. 特性

- USB フルスピード·インターフェース
- ブルートゥース™インターフェース
- プラグアンドプレイーCCID 準拠、高い柔軟性を持っている
- スマートカードリーダー：
  - 非接触インターフェース：
    - 最大 424 kbps の書き込み速度
    - 内蔵アンテナを使って、通信距離は最大 50 mm（タグのタイプに応じて）
    - ISO 14443 パート 4 の A および B カード、MIFARE®、FeliCa カードの 4 タイプすべての NFC タグ（ISO/IEC 18092）もサポートしています。
    - 衝突防止機能保有（一枚のタグのみアクセス）
    - AES-128 暗号化アルゴリズム(CBC 暗号化モード)サポート
    - NFC サポート
      - カードリーダ/ライタモード
- 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 の上のラッパー経由で）
- 内蔵の部品：
  - ザー制御可能な二色の LED パイロットランプ二つのユーザー制御可能な二色の LED パイロットランプ
  - ユーザー制御可能のブザー
- USB ファームウェアのアップグレード機能[1]
- Android™ 4.3 と以降のバージョンサポートしています[2]
- iOS 8.0 と以降のバージョンサポートしている[3]
- 以下の規格に準拠：
  - EN 60950/IEC 60950
  - ISO 14443
  - ISO 18092
  - ブルートゥース
  - PC/SC
  - CCID
  - CE
  - FCC
  - RoHS
  - REACH
  - VCCI（日本）
  - TELEC（日本）
  - Microsoft® WHQL

[1] *PC リンクのモードに適用しています。*

[2] *ACS 定義された Android ライブラリを使用しています*

[3] *ACS の iOS ライブラリを使用*

3.0. システムのブロック図

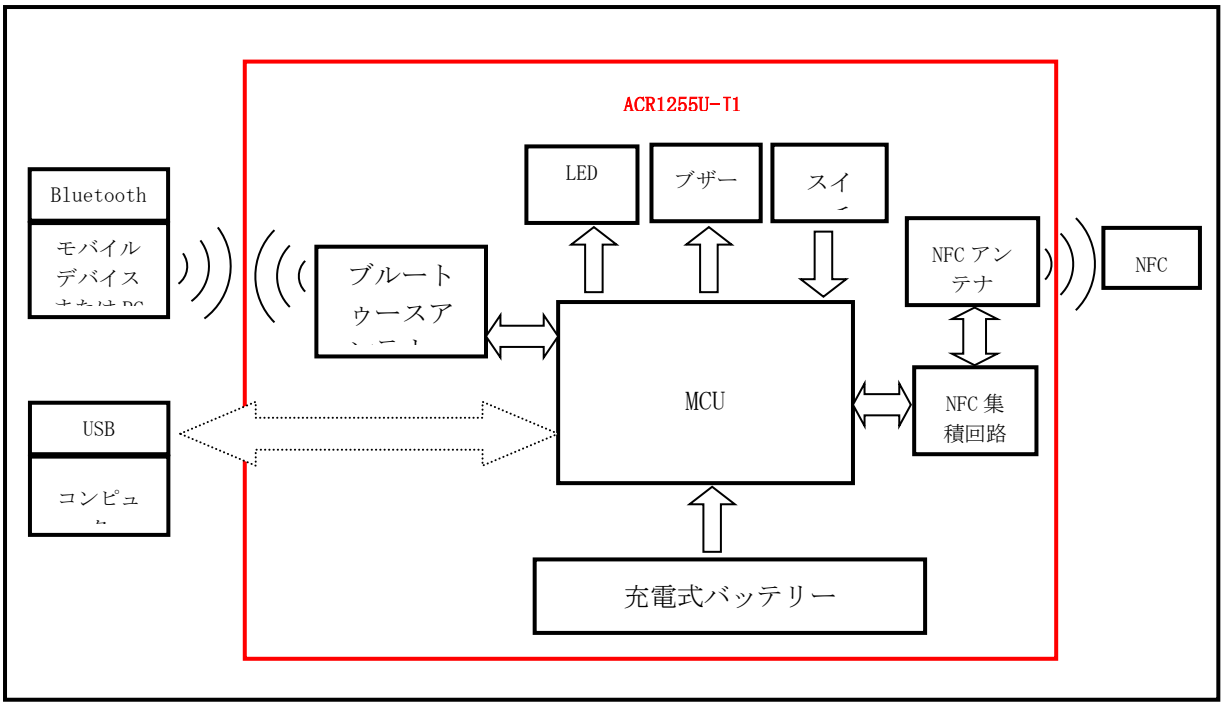


図1 ：ACR1255U-J1 のアーキテクチャ

ACR1255U-J1 がほかのデバイスとの USB 通信アーキテクチャは CCID プロトコルを採用していますが、すべての NFC 通信は PC/SC 仕様に基準しています。

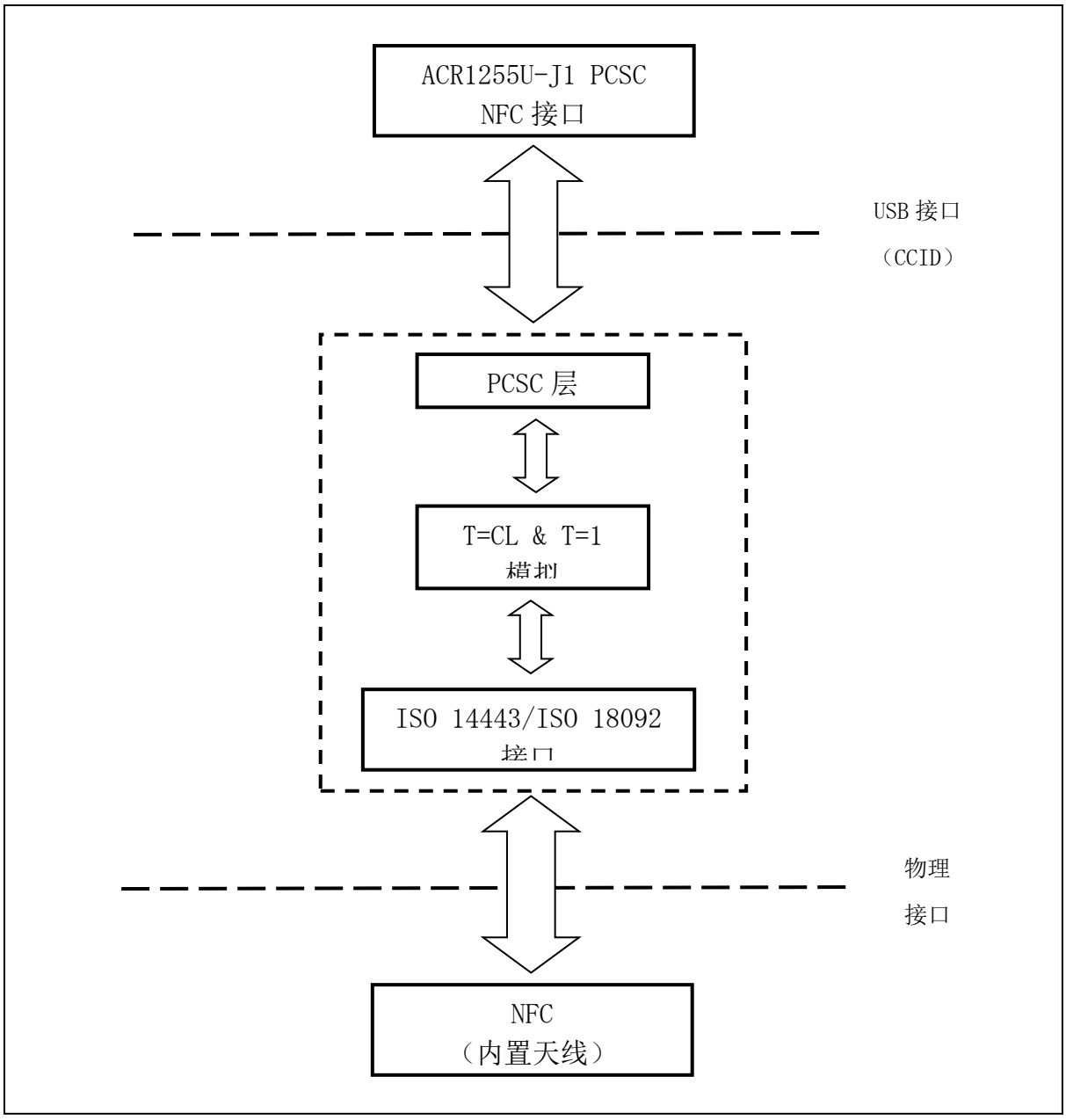


図2 ：ACR1255U-J1 の USB 通信アーキテクチャ

ブルートゥースプロトコル・スタック・アーキテクチャは次のとおりです：

<table>
<tr><td>GAP 周辺役割で配置</td><td>SIM アクセス権限の設定</td><td>GAP 周辺ボンドマネジャー</td></tr>
<tr><td colspan="2">GAP</td><td>GATT</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">SMP</td><td>ATT</td></tr>
<tr><td>L2CAP</td></tr>
<tr><td colspan="3">HCI</td></tr>
<tr><td colspan="3">リンク層</td></tr>
<tr><td colspan="3">物理層</td></tr>
</table>

図3 ：ブルートゥースプロトコル・スタック

4.0. ハードウェアのデザイン

4.1. 電池

ACR1255U-J1 は 320mAh の充電式リチウムイオン電池を使用しています。

4.1.1. 充電

ACR1255U-J1 のバッテリーが切れると、次のいずれかのモードで充電することができる：OFF モード、USB モード、Bluetooth モード;前提条件は電源コンセントに接続されています。

4.1.2. 電池の寿命

電池寿命は装置の使用に依存しています。モード条件に応じて、バッテリー寿命の推定値は以下のようになります。

モード	推定の電池寿命
動作モード	14 日* (1)
スタンバイモード	22 日*(2)
オフモード	2 年

表2 ：推定の電池寿命

***注：** 結果はスマートカードによって異なります。.

(1) *Bluetooth* モードでは、*1* 分間の操作で *1* 日に *10* 回の操作を実行します。

(2) *Bluetooth* モードでは、スリープ時間を *60* 秒に設定し、*1* 日に *1* 回ウェイクアップします。

4.2. ブルートゥースインターフェース

ACR1255U-J1 はコンピュータやモバイルデバイスとのペアリングメディアとして、Bluetooth を使用しています。

4.3. USB インターフェース

mini USB が充電バッテリーのポートとして、コンピュータに ACR1255U-J1 を接続するために使用されています。このポートは、PC-リンクモードで ACR1255U-J1 を動作させるために使用されています。

4.3.1. 通信パラメーター

ACR1255U-J1 は USB 2.0 仕様の USB インターフェース介してコンピュータに接続されます。フルスピードモードをサポートできて、12 mbps で働いています。

ピン	信号	機能
1	V_{BUS}	カードに+5 V の電源を供給
2	D-	ACR1255U-J1 と PC は差動信号でデータを転送します。
3	D+	ACR1255U-J1 と PC は差動信号でデータを転送します。
4	GND	電源用の参照電圧レベル

表3 ：USB インターフェース配線

4.3.2. エンドポイント

ACR1255U-J1 が下記のエンドポイントを介して、ホストの PC と通信します：

エンドポイント制御（Control Endpoint）	設置と制御の用
バルクアウト（Bulk OUT）	ホストから ACR1255U-J1 に送信するコマンドに対して（データパケットサイズは 64 バイト）
バルクイン（Bulk IN）	ACR1255U-J1 からホストに返す応答に対して（データパケットサイズは 64 バイト）
割り込み入力（Interrupt IN）	ACR1255U-J1 からホストに送信する状態メッセージに対して（データパケットサイズは 8 バイト）

4.4. NFC インターフェース

ACR1255U-J1 と挿入された非接触カードの間のインターフェースが ISO 14443 または ISO 18092 仕様プロトコルに準拠しています。ACR1255U-J1 の実用的な機能性を高めるために一定の制限や機能拡張をします。

4.4.1. 搬送波周波数

ACR1255U-J1 の搬送波周波数は 13.56 MHz です。

4.4.2. ポーリング

ACR1255U-J1 は自動にアンテナフィールドにある非接触カードを検出します。この機能は ISO 14443-4 の A と B タイプのカード、Topaz カード、MIFARE カード、FeliCa および NFC タグをサポートします。

4.5. ユーザーインターフェース

4.5.1. キー

ACR1255U-J1 には三つのキーがあります。場所は下記の図示のように：

キー	説明
ANT_KEY	スリープモードから戻す
MODEL_KEY	モード選択
UPDATE_KEY	ブートローダを有効にします

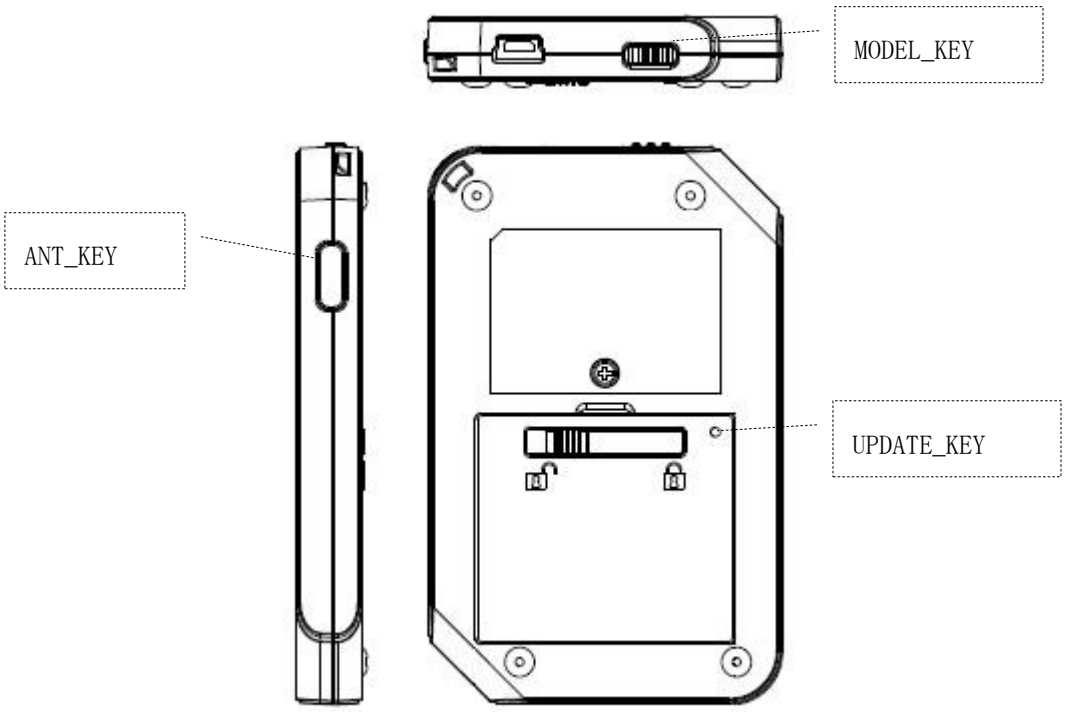


図4 ：ACR1255U-J1 のキー

4.5.2. モード選択のスイッチ

ACR1255U-J1 は三つのモードを提供しています：USB、OFF とブルートゥース。ユーザーはデータ伝送インターフェースとして、一つのモードを選択することができます。

シンボル	スイッチ	モードを有効
USB	USB	PC リンク
O	OFF	パワーOFF
ᛒ	ブルートゥース	ブルートゥース

4.5.3. LED ステータスインジケータ

様々な動作を示すために、ACR1255U-J1 二つの二色 LED を提供しています。

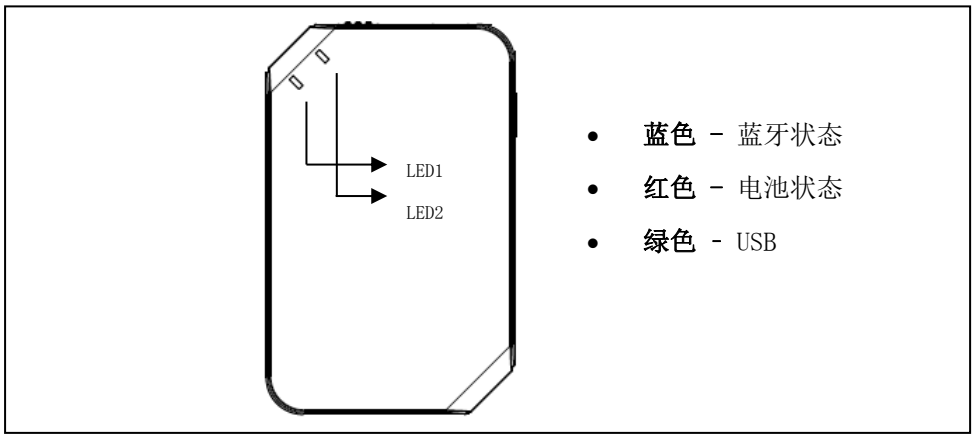


图5 ：LED ステータス

モード	色	LED ステータス	状態
Bluetooth モード	青 (LED1)	OFF	• パワーOFF • ペアリングされたブルートゥースがない • USB モード
		ゆっくり点滅 v1[44]	ペアリングされたブルートゥースデバイスは成功に認証されて、コマンドを待っています
		ゆっくり点滅 v2[55]	ブルートゥースデバイスは成功にペアリングして、カードのポーリングを開始しましたが、何のカードが存在しません。
		速く点滅する	リーダーとペアリングされたデバイスがデータを通信しています。
		速くーゆっくり点滅	• 電気入れる • ペアリングデバイスを探します
		ON	• ブルートゥースデバイスはリーダーとペアリングしました • カードある
	赤 (LED2)	OFF	• 充電完了 • パワーOFF • 電池の電圧は 3.5V より高く、USB の電源がない
		ゆっくり点滅	バッテリー不足
		ON	充電中
USB モード	緑 (LED2)	OFF	パワーOFF
		ゆっくり点滅	カードまたはリーダーがない、コマンドを待っています
		速く点滅する	スマートカードはリーダーと読み/書きます
		ON	カードまたはリーダーがある、コマンドを待っています
	オレンジ[66] (LED2)	ON	デバイスは電気入れられた
	赤 (LED1)	OFF	• 充電完了 • パワーOFF • 電池の電圧は 3.5V より高く、USB の電源がない
		ゆっくり点滅	バッテリー不足
		ON	充電中

[4] *LED* は点灯 *500ms*、消す *500ms*
[5] *LED* は点灯 *200ms*、消す *1800ms*
[6] 赤と緑の *LED* がオンにされています。

4.5.4. ブザー

ACR1255U-J1 はカードポーリング、Bluetooth 接続、スリープモード、低バッテリー状態をユーザに通知するためのブザーを有します。

ブザー操作	イベント
短いビープ音が 1 回鳴ります	カードが検出されたまたはカードがはなれました
長いビープ音が 1 回鳴ります	リーダーの電源が切られた

5.0. ハードウェアのデザイン

5.1. ブルートゥース通信プロトコル

5.1.1. Bluetooth 接続フロー

下記は Bluetooth の接続フローです：

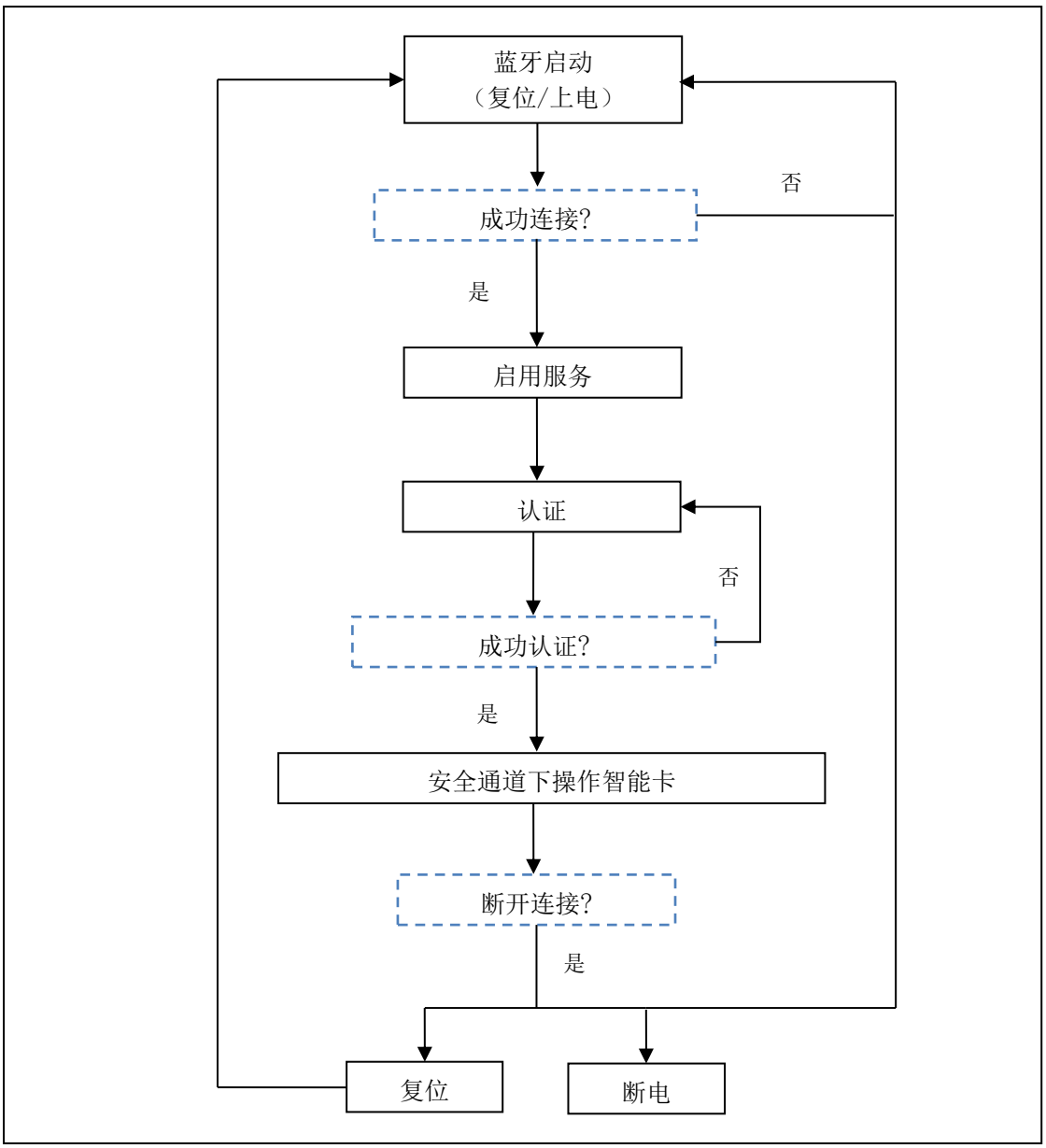


図6 ：Bluetooth 接続フロー

5.1.2. Profile セレクション

ACR1255U-J1 はデータを送信するためのインタフェースとしての Bluetooth 技術を使用するように設計されたスマートカードリーダーです。3 つのパイプでコマンド通信できるカスタマイズされたサービスが使用されている：一番目のパイプがコマンド要求のために使用され、二番目のパイプは、コマンド応答/カード通知するために使用され、三番目は RFU です。

また、リーダーは Bluetooth モードで動作している時、現在の電力消費量が重要なので、従って、カスタマイズのバッテリサービスが現在のバッテリーステータスをペアリング装置に通知するために使用されます。電池状態の変化があった場合、リーダーは、特定のパイプを介してペアリング装置に通知します。簡単に言えば、バッテリーレベルが 3 つのグループに分けています：十分（≥3.78 V）、低バッテリ（<3.78 V と≥3.68 V）、およびバッテリなし（<3.68 V）。

最後に、ユーザに多くのリーダーの情報を提供するために、カスタマイズされたデバイス情報サービスが追加されます。これは、手動で読み、またはアプリケーションの要求によって読むことができます。特徴は、モデル番号、シリアル番号、ファームウェアリビジョン、およびメーカー名が含まれます。

サービス	UUID	パイプ
スマートカード	3C4AFFF1-4783-3DE5-A983-D348718EF133	コマンド請求
	3C4AFFF2-4783-3DE5-A983-D348718EF133	コマンド応答/カード通知
	3C4AFFF3-4783-3DE5-A983-D348718EF133	RFU
電池	2A19	電池レベル
デバイス情報	2A23	システム ID
	2A24	型番
	2A25	シリアルナンバー
	2A26	ファームウェアのバージョン
	2A27	ハードウェアのバージョン
	2A29	メーカ名称

表4 ：ACR1255U-J1 のブルートゥースサービス

属性名称	UUID	ハンドル
DeviceName	2A00	03h
Send(Reader → Paired device)	3C4AFFF1-4783-3DE5-A983-D348718EF133	2Ah
Receive(Paired device → Reader)	3C4AFFF2-4783-3DE5-A983-D348718EF133	2Dh
BatteryLevel	2A19	14h
Manufacturer	2A29	1Eh
SerialNumber	2A25	16h
FW_Version	2A26	18h
ModelNumber	2A24	14h

表5 ：ACR1255U-J1 のサービスハンドルと UUID メッセージリスト

5.1.3. 認証

機密データが ACR1255U-J1 にロードする前に、データ処理サーバは、リーダ内部で保護されたデータを変更する権限のため ACR1255U-J1 によって認証されなければなりません。ACR1255U-J1 においては、相互認証が使用されています。

良い説明のために、下の図示を参照してください（シンプルさとより良い説明のために、下の図示がブリッジデバイスを省略している）

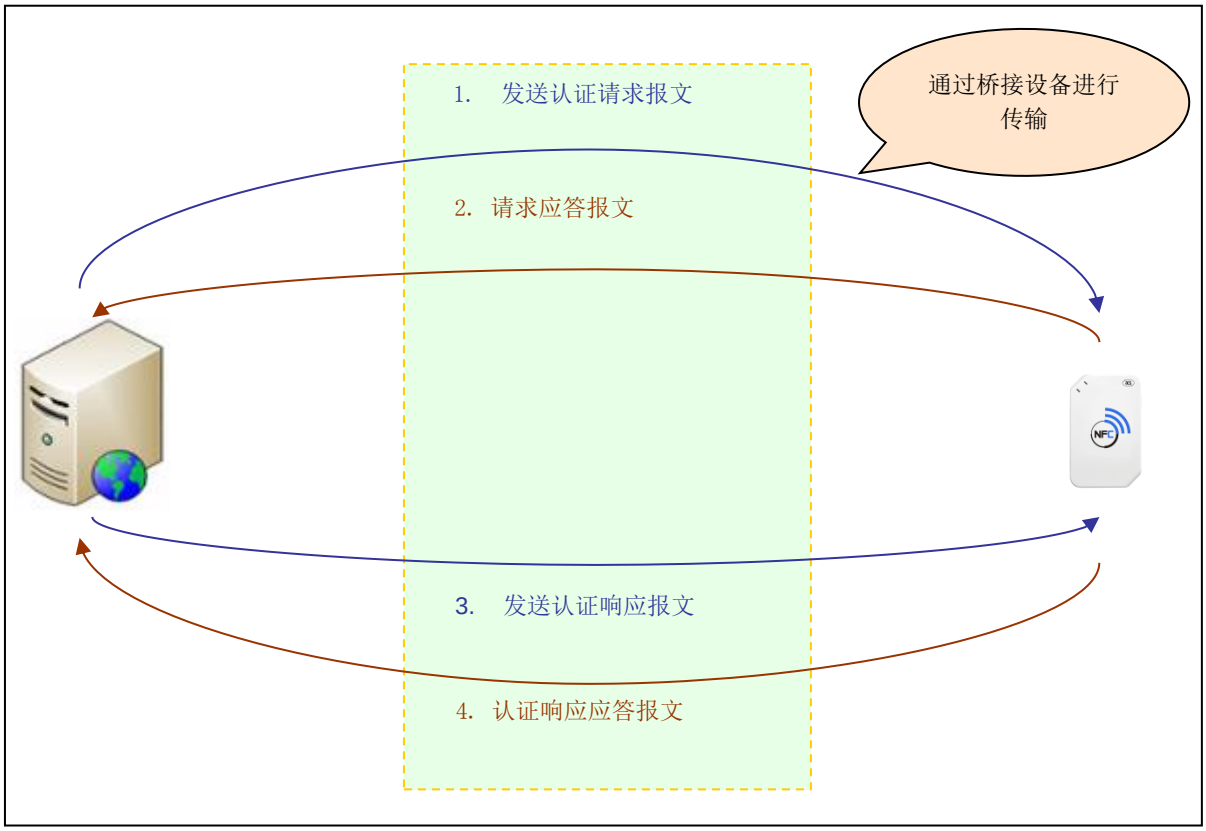


图7 ：認証手順

認証に成功すると、ACR1255U-J1 とデータサーバーでそれぞれ 16 バイトのプロセスキーが生成されます。

デフォルトの顧客マスターキー: **41 43 52 31 32 35 35 55 2D 4A 31 20 41 75 74 68**

注：認証キーの間違いが最大 *6* 回です。一度超えると、リーダーがロックされ、使用できなくなります。

関係情報は *ACS* の販売員までに連絡してください。

5.1.4. 通信配置

通信プロファイルは次のようになります。

开始バイト（Start byte） + 長さ（Len） + データブロック（Datablock） + チェック（Check） + 終了バイト（Stop byte）

データフィールド	大きさ（バイト）	説明
Start byte	1	値：05h
Len	2	Len は Datablock データフィールド中のバイト数を示す
Datablock	N	データ（参照：フレームでの説明**フレームフォーム**セッション)
Check	1	Check は Len と Datablock データフィールド中の各バイトの XOR 値
Stop byte	1	値：0Ah

5.1.5. フレームフォーム

5.1.5.1. データフレームフォーム

コマンド

HID フレーム	長さ（バイト）	説明
CmdMessageType	1	コマンド
Length	2	データ長さ
Slot	1	デフォルト値 ： 00h
Seq	1	番号
Param	1	パラメ―タ―
Checksum	1	チェックサム
Data	0-N	データ

フレームフォームは：

CmdMessageType + Length + Slot + Seq + Param + Checksum + Data

コマンドの長さ（識別子、長さ及びペイロードを含めて）は２０バイトより長い場合、リーダーまたはペアリングデバイスは自動にいくつのフレームに分けます。

応答

HID フレーム	長さ（バイト）	説明
RspMessageType	1	応答
Length	2	データ長さ
Slot	1	デフォルト値 ： 00h
Seq	1	番号
Slot Status/Param	1	カードスロットのステータス/パラメーター
Checksum	1	チェックサム
Data	0-N	データ

データチェックサムは、無線にデータ送信時に導入された可能性のあるエラーを検出するのに使用されます。データチェックサム計算

XOR { RspMessageType, Length, Slot, Seq, SlotStatus/Param, Data }.

例： 62 00 00 00 01 00 63 => チェックサム = 63h

5.1.6. ブルートゥース通信プロトコル

ACR1255U-J1 は、予め定義されたプロトコルを採用して、ブルートゥース・インターフェースでペアリングされたデバイスと通信します。プロトコルは、CCID コマンドパイプと応答パイプの形式に似ています。

コマンド	サポートモード	送信側	説明
62h	認証済	ペアリングデバイス	PICC パワーアップ
63h	認証済	ペアリングデバイス	PICC パワーオフ
65h	認証済	ペアリングデバイス	カードステータスを取得する
6Fh	認証済	ペアリングデバイス	APDU 交換
6Bh	認証済	ペアリングデバイス	ダイレクトコマンド

表6 ：コマンドコードの概要

コマンド	サポートモード	送信側	説明
80h	認証済	リーダー	データブロックの応答
81h	認証済	リーダー	カードスロットステータスの応答
83h	認証済	リーダー	直接コマンドの応答
50h	認証済	リーダー	カードステータスを通知する
51h	認証済	リーダー	ハードウェアエラーの応答

表7 ：応答コードの概要

5.1.6.1. カードパワーアップ

このコマンドはリーダにパワーアップのリクエストを送信するために使用されます。

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	**62h**	-
1	*Length*	2	00 00h	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメ―タ―
6	*Checksum*	1	-	CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**80h**	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	パラメ―タ― Bit0 が LSB Bit 0-1 = 00h = カードあり（アクティブ） 01h = カードある（アクティブされていない） 02h = カードなし Bit 6 = “0” = エラーなし = “1” = エラー発生 例：パラメータ= 41h = 0100 0001b

オフセット	データフィールド	大きさ	数値	説明
				カード（アクティブ化されていない）があるときにエラーが発生しました
6	*Checksum*	1	-	CheckSum は応答中でのすべての XOR 値を示す
7	*Data*	N	-	データ

例：

応答 = 80 00 08 00 00 00 B3 3B 83 80 01 41 07 00 44

ATR = 3B 83 80 01 41 07 00 44

5.1.6.2. カードパワーオフ

このコマンドはリーダにパワーオフのリクエストを送信するために使用されます。

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	**63h**	-
1	*Length*	2	0000h	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメーター
6	*Checksum*	1	-	CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**81h**	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	パラメ―タ― Bit 0 が LSB Bit 0-1 = 00h = カードあり（アクティブ） 01h = カードある（アクティブされていない） 02h = カードなし Bit 6 = “0” = エラーなし = “1” = エラー発生 例：パラメータ = 41h = 0100 0001b カード（アクティブ化されていない）があるときにエラーが発生しました
6	*Checksum*	1	-	CheckSum は応答中でのすべての XOR 値を示す
7	*Data*	N	-	データ

5.1.6.3. カードスロットのステータスを取得する。

このコマンドでカードが挿入されているかどうかを検出します。

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	**65h**	-
1	*Length*	2	00 00h	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメ―タ―
6	*Checksum*	1	-	CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**81h**	-
1	*Length*	2	0000h	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	カードステータス： Bit 0 が LSB Bit 0-1 = 00h = カードあり（アクティブ） 01h = カードある（アクティブされていない） 02h = カードなし Bit 6 = “0” = エラーなし = “1” = エラー発生 例：パラメータ = 41h = 0100 0001b カード（アクティブ化されていない）があるときにエラーが発生しました
6	*Checksum*	1	-	CheckSum は応答中でのすべての XOR 値を示す
7	*Data*	N	-	データ

5.1.6.4. APDU コマンド

このコマンドはリーダに APDU コマンドを送信するために使用されます。

注：拡張 *APDU* をサポートする「*Param*」オプションは、ファームウェアバージョン *2.01.00* 以降に適用されます。。

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	**6Fh**	-
1	*Length*	2	-	*最大データ長さ；最大長さ： 256 バイト
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	パラメ―タ― 短 APDU レベル 00h – ディフォルト 拡張 APDU レベル 00h –APDU コマンドはこのコマンドで開始および終了します。 01h - APDU コマンドがこのコマンドで開始され、次の APDU コマンドで続行されます。 02h - データフィールドは引き続きコマンド APDU を渡し、APDU コマンドを終了します。 03h - データフィールドはコマンド APDU の後に続けて別のデータブロックが続きます。 10h - 空のデータフィールド、次の応答は応答 APDU を渡し続けます
6	*Checksum*	1	-	CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**80h**	-
1	*Length*	2		データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	パラメ―タ― Bit0-5 に対して Bit 0 が LSB 短 APDU レベル 00h – default 拡張 APDU レベル 00h - 応答 APDU がこのコマンドで開始および終了します。 01h - 応答 APDU がこのコマンドで開始および続ける。 02h - データフィールドは引き続き応答 APDU を渡し、APDU 応答を終了します。 03h - データフィールドは引き続き応答 APDU を渡し、応答 APDU の後に続けて別のデータブロックが続きます。 10h–空のデータフィールド。次のコマンドは引き続き APDU コマンドを渡します。, Bit 6 エラービットとして使用 Bit 6 = “0” = エラーなし = “1” = エラー発生
6	*Checksum*	1	-	CheckSum は応答中でのすべての XOR 値を示す
7	*Data*	N	-	データ

例：

カードに 600 バイトのデータを送信する

1. コマンド = 6F 01 00 00 XX 01 チェックサム (256 バイトデータ)

   応答 = 80 00 00 00 XX 10 チェックサム

2. コマンド = 6F 01 00 00 XX 03 チェックサム (256 バイトデータ)

   応答 = 80 00 00 00 XX 10 チェックサム

3. コマンド = 6F 00 58 00 XX 02 チェックサム (88 バイトデータ)

   応答 = 80 00 02 00 XX 00 チェックサム 90 00

カードから 600 バイトのデータを受信する

1. コマンド = 6F 00 07 00 XX 00 チェックサム 00 B0 87 00 00 02 58

   応答 = 80 01 00 00 XX 01 チェックサム (256 バイトデータ)

2. コマンド = 6F 00 00 00 XX 10 チェックサム

   応答 = 80 01 00 00 XX 03 チェックサム (256 バイトデータ)

3. コマンド = 6F 00 00 00 XX 10 チェックサム

   応答 = 80 00 00 00 XX 02 チェックサム (88 バイトデータ)90 00

5.1.6.5. カードステータス通知コマンド

このコマンドはカードのステータスをチェックするために使用されます。

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**50h**	-
1	*Length*	2	00 00h	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	状態： 02h = カードなし 03h = カードある
6	*Checksum*	1	-	CheckSum はメッセージ中でのすべての XOR 値を示す
7	*Data*	N	-	データ

注： *“52 00 00 00 00 01 53”*のメッセージが表示したら、リーダーはスリープモードに入るという意味です。

5.1.6.6. ハードウェアエラーの応答

誤ったコマンドを受け取った場合、このメッセージはエラーメッセージを返します。

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	51h	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	-	状態： 01h = チェックサムエラー 02h = タイムアウト 03h = コマンドエラー 04h = 許可されていません 05h = 未定義のエラー 06h = データ受信エラー 07h = 認証回数が制限を超えるエラー
6	*Checksum*	1	-	CheckSum はメッセージ中でのすべての XOR 値を示す
7	*Data*	N	-	データ

5.1.6.7. ダイレクトコマンド

このコマンドでリーダーの拡張機能をアクセスできます。

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	**6Bh**	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメ―タ―
6	*Checksum*	1	-	CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

応答データフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*RspMessageType*	1	**83h**	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメ―タ―
6	*Checksum*	1	-	CheckSum は応答中でのすべての XOR 値を示す
7	*Data*	N	-	データ

5.1.6.7.1. 直接コマンド例

1. 自動ポーリングを有効する

   リクエストコマンド：E0 00 00 40 01

   応答コマンド：E1 00 00 40 01

   例：

   リクエスト：6B 00 05 00 00 00 CF E0 00 00 40 01

   応答： 83 00 05 00 00 00 66 E1 00 00 40 01

2. 自動ポーリングを無効する

   リクエストコマンド：E0 00 00 40 00

   応答コマンド：E1 00 00 40 00

   例：

   リクエスト：6B 00 05 00 00 00 CE E0 00 00 40 00

   応答： 83 00 05 00 00 00 67 E1 00 00 40 00

3. 認証リクエスト

   リクエストコマンド：E0 00 00 45 00

   応答コマンド：E1 00 00 45 00 + Data (16 バイト)

   その中：Data = 16 バイトの乱数

   例：

   リクエスト (SPH_to_RDR_ReqAuth): 6B 00 05 00 00 00 CB E0 00 00 45 00

   応答 (RDR_to_SPH_AuthRsp1): 83 00 15 00 00 00 21 E1 00 00 45 00 77 59 E8 62 B7 80 0D 0A CE 9A 03 9B E9 48 EF 05

4. 認証応答

   リクエストコマンド：E0 00 00 46 00 + Data (32 バイト)

   応答コマンド：E1 00 00 46 00 + Data (16 バイト)

   例：

   請求 (SPH_to_RDR_AuthRsp)：6B 00 25 00 00 00 EA E0 00 00 46 00 A6 81 17 91 9F 46 07 AE AE 4E 94 8E 05 14 E8 C8 25 F7 90 05 76 F8 DE 7D 6D ED 55 3F 80 10 C2 CA

   応答 (RDR_to_SPH_AuthRsp2)：83 00 15 00 00 00 51 E1 00 00 46 00 47 D5 50 54 F3 49 D4 17 B1 65 40 21 9B DA C9 B2

5.1.7. 相互認証と暗号化プロトコル

ブルートゥースモードで相互認証が成功してから、**ブルートゥース通信プロトコル**セッション中の通信プロトコルを暗号化し、転送します。

5.1.7.1. Bluetooth 認証プロセス：

認証セッションに示されるように、相互認証が man-in-the-middle 攻撃を回避するために使用されています。相互認証で使用するコマンドの概要は以下の表に示します。

番号	コマンド	サポートモード	送信側	説明
1	6Bh	接続済	ペアリングデバイス	SPH_to_RDR_ReqAuth
2	83h	接続済	リーダー	RDR_to_SPH_AuthRsp1
3	6Bh	接続済	ペアリングデバイス	SPH_to_RDR_AuthRsp
4	83h	接続済	リーダー	RDR_to_SPH_AuthRsp2

表8 ：相互認証コマンドの概要

5.1.7.2. SPH_to_RDR_ReqAuth

このコマンドは、鍵生成装置に認証を行うために ACR1255U-J1 を要求します。

認証プロセスの詳細については、認証**認証**セッションを参照してください。

オフセット	データフィールド	大きさ	数値	説明	暗号化
0	*bMessageType*	1	6Bh	-	NO
1	*LEN1 LEN2 (wLength)*	2	0005h	長さは 2 バイトで、データフィールド中での余計のバイトの数を示します。 LEN1 は MSB で、LEN2 は LSB です。	
3	スロット番号	1	00h	-	
4	番号	1	00h	-	
5	パラメーター	1	00h	カードスロットのステータス	
6	*wChecksum*	1	CBh	CSUM はコマンド中でのすべての XOR 値を示す。	
7	データ	5	E0 00 00 45 00h	-	NO

受信したコマンドメッセージはエラーがない場合は、RDR_to_SPH_AuthRsp1 が受信するはずです。そうじゃないと、エラーメッセージが含まれた RDR_to_SPH_ACK の応答を受信するはずです。

5.1.7.3. RDR_to_SPH_AuthRsp1

このコマンドは ACR1255U-J1 から送信された SPH_to_RDR_ReqAuth の応答です。

オフセット	データフィールド	大きさ	数値	説明	暗号化
0	*bMessageType*	1	**83h**	-	NO
1	*LEN1 LEN2 (wLength)*	2	0015h	長さは 2 バイトで、abRndNum のデータフィールド中での余計のバイトの数を示します。 LEN1 は MSB で、LEN2 は LSB です。	NO
3	スロット番号	1	00h	-	NO
4	番号	1	00h	-	NO
5	パラメーター	1	00h	カードスロットのステータス	-
6	*wChecksum*	1	-	wChecksum はコマンド中でのすべての XOR 値を示す	NO
7	*abRndNum*	21	E1 00 00 45 00 + 16 バイトの乱数	abRndNum[0:15] – 16 バイトの乱数 すべての 16 バイトの乱数は現在に ACR1255U-J1 中でストレージされている顧客マスターキーで暗号化しなければなりません。 “E1 00 00 45 00”は暗号化される必要がありません。	YES

5.1.7.4. SPH_to_RDR_AuthRsp

このコマンドは認証プロセスの第二段階です。デバイスは SPH_to_RDR_ReqAuth コマンドを ACR1255U-J1 に送信して、エラーがない場合はリーダーは RDR_to_SPH_AuthRsp1 メッセージを戻します。。

RDR_to_SPH_AuthRsp1 には顧客マスターキーで暗号化された 16 バイトの乱数が含まれています。ペアリングされたキーの生成デバイスは正しい顧客マスターキーで復号化する必要があります。また、１６バイトの乱数のあとに追加します。それに顧客マスターキーで３２バイトの乱数全体を復号化して、結果をこのコマンドで ACR1255U-J1 に返すことによって、認証を完了します。

オフセット	データフィールド	大きさ	数値	説明	暗号化
0	*bMessageType*	1	6Bh	-	NO
1	*LEN1 LEN2 (wLength)*	2	0025h	長さは 2 バイトで、abAuthData のデータフィールド中での余計のバイトの数を示します。 LEN1 は MSB で、LEN2 は LSB です。	NO
3	スロット番号	1	00h	-	NO
4	番号	1	00h	-	NO
5	パラメーター	1	00h	カードスロットのステータス	NO
6	*wCheckSum*	1	-	wCheckSum はコマンド中でのすべての XOR 値を示す	NO
7	*abAuthData*	37	E0 00 00 46 00 + 32 バイトの乱数	abAuthData[0:15] – データ処理サーバに生成された 16 バイトの乱数 abAuthData[16:31] – ACR1255U-J1 に受信された 16 バイトの復号化した乱数 すべての 32 バイトの乱数を AES128 CBC 暗号化モードで顧客マスターキーを使用して、復号化します。 “E0 00 00 46 00”は復号化される必要がありません。	YES

受信されたコマンドメッセージはエラーがないで、ペアリングしたデバイスに返された乱数も正しい場合、応答は RDR_to_SPH_AuthRsp2 です。そうじゃないと、エラーメッセージが含まれた RDR_to_SPH_ACK の応答を受信するはずです。

5.1.7.5. RDR_to_SPH_AuthRsp2

このコマンドは ACR1255U-J1 から送信された SPH_to_RDR_AuthRsp の応答です。

オフセット	データフィールド	大きさ	数値	説明	暗号化
0	*bMessageType*	1	**83h**	-	NO
1	*LEN1 LEN2 (wLength)*	2	0015h	長さは 2 バイトで、abRndNum のデータフィールド中での余計のバイトの数を示します。 LEN1 は MSB で、LEN2 は LSB です。	NO
3	スロット番号	1	00h	-	NO
4	番号	1	00h	-	NO
5	パラメーター	1	00h	カードスロットのステータス	NO
6	*wCheckSum*	1		wCheckSum はコマンド中でのすべての XOR 値を示す	NO
20	*abRndNum*	21	E1 00 00 46 00 + 16 バイトの乱数	abRndNum[0:15] – データ処理サーバに受信された 16 バイトの乱数 すべての 16 バイトの乱数は現在に ACR1255U-J1 中でストレージされている顧客マスターキーで暗号化しなければなりません。 “E1 00 00 46 00”は暗号化される必要がありません。	YES

5.1.7.6. RDR_to_SPH_ACK (エラー処理)

このコマンドはコマンドメッセージの受信を確認するように、ACR1255U-J1 からペアリングしたデバイスに送信されるエラー処理を確認するメッセージです。通信中、指定のエラーメッセージは RDR_to_SPH_ACK で転送しなければなりません。コマンドを暗号化する必要がありません。

コマンド	サポートモード	送信側	説明
51h	接続済/認証済	リーダー	RDR_to_SPH_ACK (エラー処理)

必要な場合、エラーコードがこのメッセージに含まれます。

オフセット	データフィールド	大きさ	数値	説明	暗号化
0	*bMessageType*	1	51h	エラー処理の応答ヘッダ	NO
1	*LEN1LEN2 (wLength)*	2	00h	-	
3	スロット番号	1	00h	-	
4	番号	1	00h	-	
3	*bErrorCode*	1	-	この前、コマンドメッセージを処理する時に発生したエラーコードを指定します。可能性があるエラーコードは下記のテーブルをご覧下さい。	
4	*wChecksum*	1	-	CSUM はコマンド中でのすべての XOR 値を示す。	

データフィールド	数値	説明
bErrorCode	01h	チェックサムエラー
	02h	タイムアウト？
	03h	コマンドエラー
	04h	許可されていない
	05h	未定義のエラー
	06h	受信したデータエラー。
	07h	認証回数が制限を超えるエラー

表9 ：相互認証エラーコード

6.0. ホストプログラミング（PC リンク）API

6.1. PC/SC API (For Windows®)

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

6.1.1. SCardEstablishContext

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

この関数は、他の PCSC 操作を実行する前に実行する必要があります。

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
   }
}
```

6.1.2. SCardListReaders

この関数は、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
      }
   }
}
```

6.1.3. SCardConnect

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

参照のウェブサイト：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 ACR1255U-J1 PICC Reader 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
          }
       }
   }
}
```

6.1.4. SCardControl

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

参照のウェブサイト：http://msdn.microsoft.com/en-us/library/windows/desktop/aa379474%28v=vs.85%29.aspx

注：**周辺デバイス制御**のコマンドはこの API を使用して送信しています。

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                  endBuff[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 ACR1255U-J1 PICC Reader 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
            }
```

6.1.5. ScardTransmit

この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 **MIFARE® Classic （1K/4K）メモリカードの PICC コマンド**ド（即ち：接続を確立されらカードに送信するコマンドです。**非接触インターフェースの疑似 APDU コマンド**セクション）はこの *API* で送信します。

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 ACR1255U-J1 PICC Reader 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];
   }
}
```

6.1.6. ScardDisconnect

この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

PCSC 操作を終了するために使われます。

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
      }
   }
}
```

6.1.7. APDU の流れ

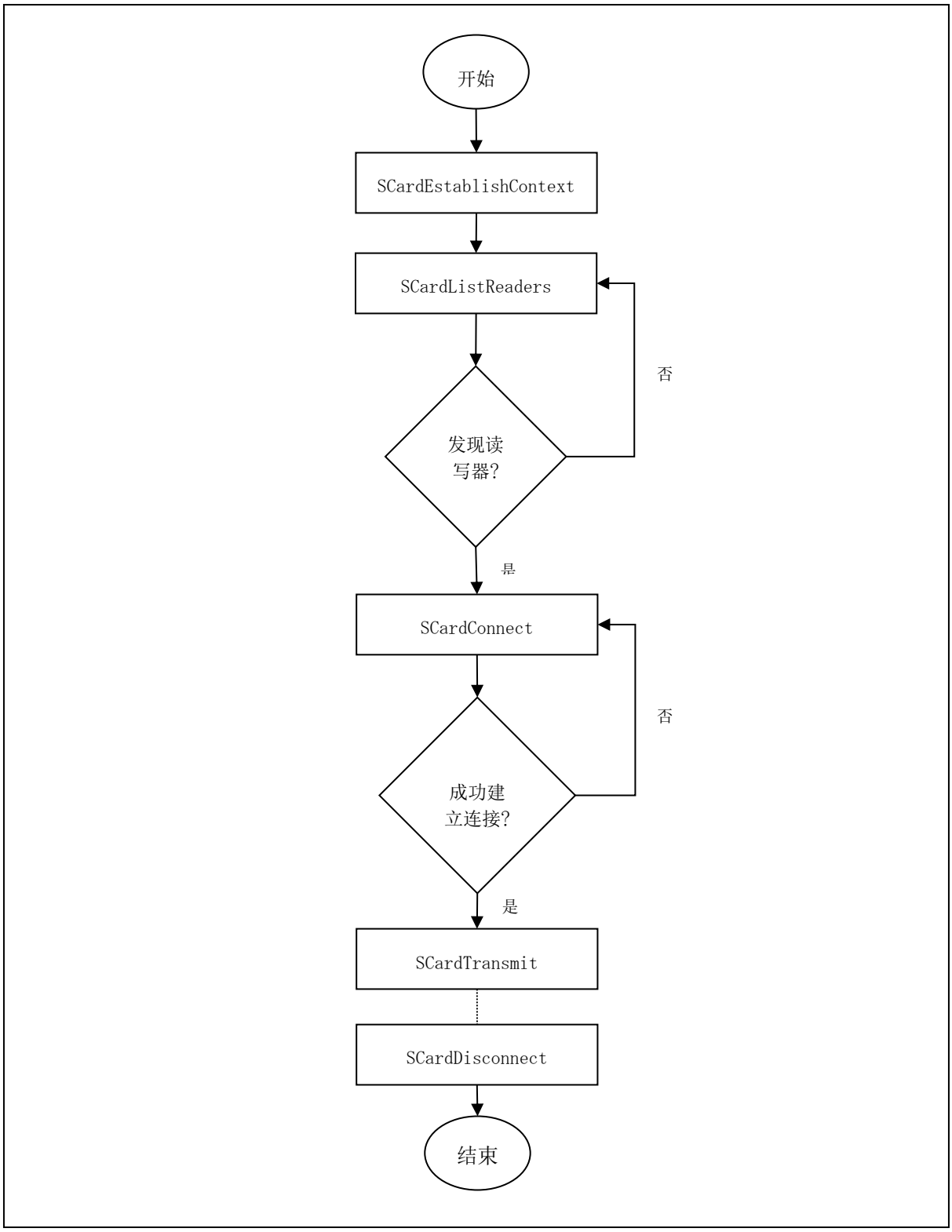


図8 ：ACR1255U-J1 の APDU の流れ

6.1.8. 直接コマンド（Escape Command）の流れ

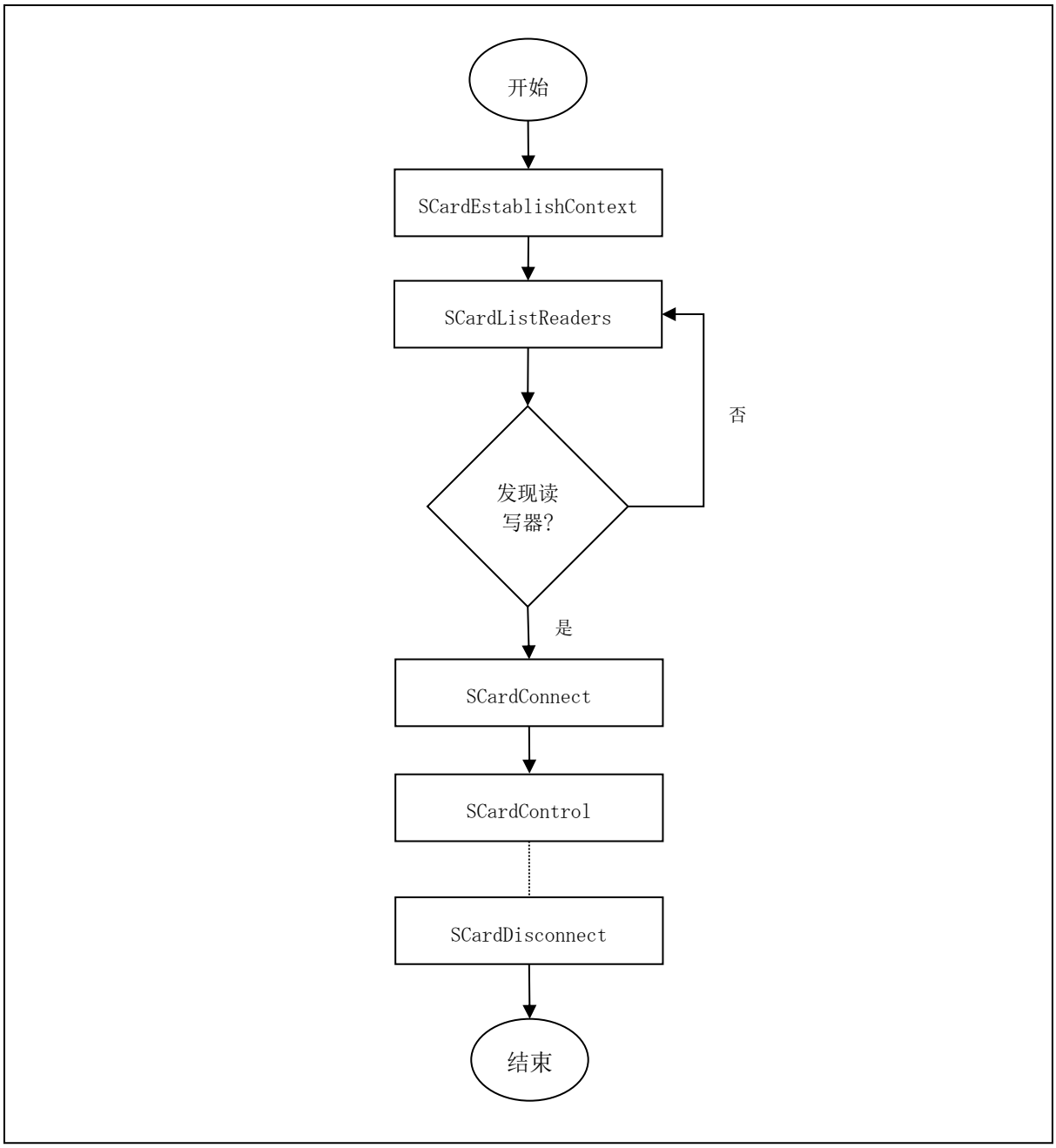


図9 ：ACR1255U-J1 直接なコマンドの流れ

6.2. 非接触スマートカード プロトコル

6.2.1. 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 PCSC ドライバに送されます。

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

バイト	数値	標記	説明
0	3Bh	最初のヘッダー	-
1	8Nh	T0	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		長さ
	RID		登録されたアプリケーションプロバイダ識別子 （RID） # A0 00 00 03 06
	SS		基準のバイト
	C0 ..C1h		カードネームバイト
	00 00 00 00h	RFU	RFU # 00 00 00 00
4+N	UU	TCK	T0 から Tk までのすべてのバイトの排他的論理和

例：

MIFARE Classic 1K 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 38: MIFARE Plus SL2 2K
00 02: MIFARE Classic 4K	00 39: MIFARE Plus SL2 4K
00 03: MIFARE Ultralight	00 30: Topaz and 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]: undefined tags
00 37: MIFARE Plus SL1 4K	00 07: SRIX512

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

バイト	数値	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

<table>
<tr><th>バイト</th><th>数値</th><th>標記</th><th>説明</th></tr>
<tr><td rowspan="2">4
から
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>Byte1-4</th><th>Byte5-7</th><th>Byte8</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
XX
XX</td><td>Tk</td></tr>
<tr><td>4+N</td><td>UU</td><td>TCK</td><td>T0 から Tk までのすべてのバイトの排他的論理和</td></tr>
</table>

例 1：

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

注釈：*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

6.2.2. 非接触インターフェースの疑似 APDU コマンド

6.2.2.1. データを取得する（Get Data）

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

GET UIDAPDU フォーマット（5 バイト）

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

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

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

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

応答	データ出力		
結果	ATS	SW1	SW2

応答コード

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

例：

"接続された PICC"のシリアルナンバーを取得します

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

"接続された ISO 14443-A PICC"の ATS を取得します

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

6.2.2.2. PICC データ取得（Get PICC Data）

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

注： *2.04.xx* 以降のバジョンのみに適用します。

Get PICC DataAPDU フォーマット（5 バイト）

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

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

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

B タイプのカードの場合 ATQB を取得（12 バイト+2 バイト）

応答	データ出力		
結果	ATQB	SW1	SW2

応答コード

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

6.2.3. 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）

PC/SC 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。

6.2.3.1. 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	予想以外の長さのデータオブジェクト XX
XX 6A 80h	予想以外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。

6.2.3.2. 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	DataLen	データオブジェクト （N バイト）

その中：

データオブジェクト（1 バイト)

タグ	データオブジェクト
80h	バージョンのデータオブジェクト
81h	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―タ―を取得する
FF 6Eh	パラメ―タ―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―タ―データオブジェクト

6.2.3.2.1. 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	-

6.2.3.2.2. 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	-

6.2.3.2.3. バージョンのデータオブジェクト（Version Data Object）

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

6.2.3.2.4. 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	-

6.2.3.2.5. 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	-

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

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

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

タイマーデータオブジェクト

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

6.2.3.2.7. 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	バール	TLV_Objects		

TLV_Objects

要求のパラメータ	タグ	長さ
IFD フレームサイズの整数（FSDI）	01h	00h
ICC フレームサイズの整数（FSCI）	02h	00h
フレーム待ち時間の整数（FWTI）	03h	00h
IFD でサポートされている最大な通信速度	04h	00h
ICC の通信速度	05h	00h
指数変調	06h	00h
ISO/IEC14443 基準の PCB	07h	00h
ISO/IEC14443 基準の CID	08h	00h
ISO/IEC14443 基準の NAD	09h	00h
ISO/IEC14443 B タイプのパラメ—タ—1 - 4	0Ah	00h

6.2.3.2.8. 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—タ—1 - 4	0Ah	04h

6.2.3.3. 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	送受信ー送信と受信
FF 6Dh	パラメ―タ―を取得する
FF 6Eh	パラメ―タ―を設定する

透明交換セッションの応答データオブジェクト

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

6.2.3.3.1. 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

6.2.3.3.2. 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。

6.2.3.3.3. 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。

6.2.3.3.4. 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）

6.2.3.3.5. 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	-

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

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

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

データオブジェクトを送受信する

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

6.2.3.3.7. 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れます。じゃないと“00h”を表示します。
		1	0 – 衝突なし 1 – 衝突が検出されました	
		2	0 – パリティエラーなし 1 – パリティエラーが検出されました	
		3	0 – フレームエラーなし 1 – フレームエラーが検出されました	
		4 - 7	RFU	

6.2.3.3.8. データオブジェクトを応答する（Response 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。

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）

6.2.3.4. 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―タ―を取得する
FF 6Eh	パラメ―タ―を設定する

プロトコルの応答データオブジェクトを切り替える

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

6.2.3.4.1. 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

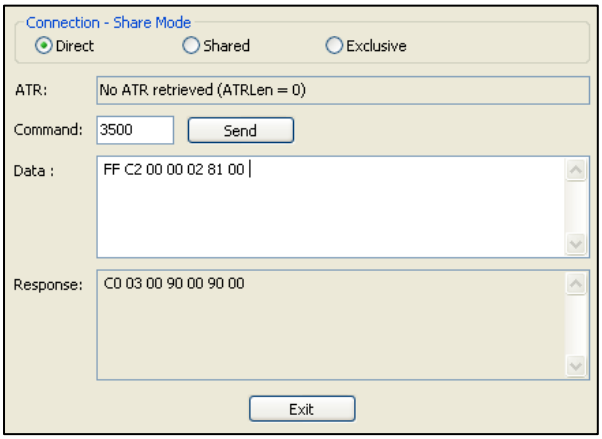
タグ	長さ（1 バイト）	数値	
		バイト 0	バイト 1
8Fh	02h	00h – ISO/IEC14443 Type A 01h – ISO/IEC14443 Type B 03h – FeliCa 他 – RFU	00h – 層分離がない場合 02h – ２層に切り替える 03h – ３層に切り替えるまたは活性化する 04h – ４層に活性化する 他 - RFU

6.2.3.5. 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**

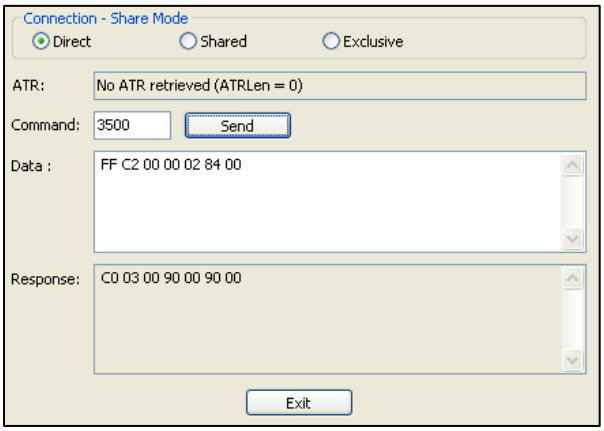
   応答：**C0 03 00 90 00 90 00**



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**

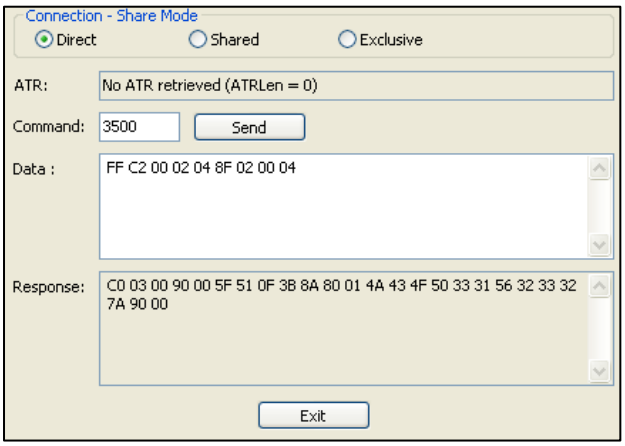


3. ISO 14443-4A 有効。

   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)

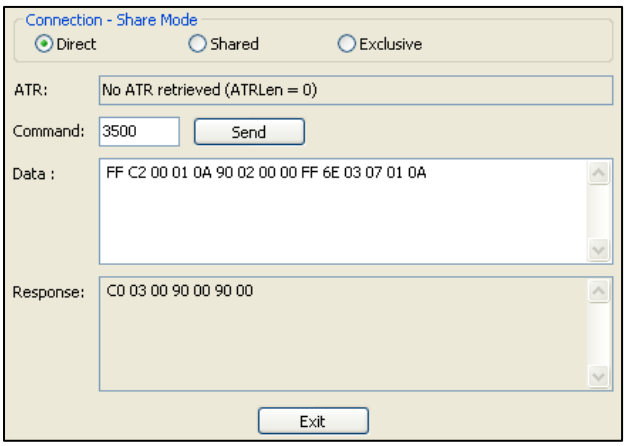
   C0 03 00 90 00 5F 51 [ATR] 90 00



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

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

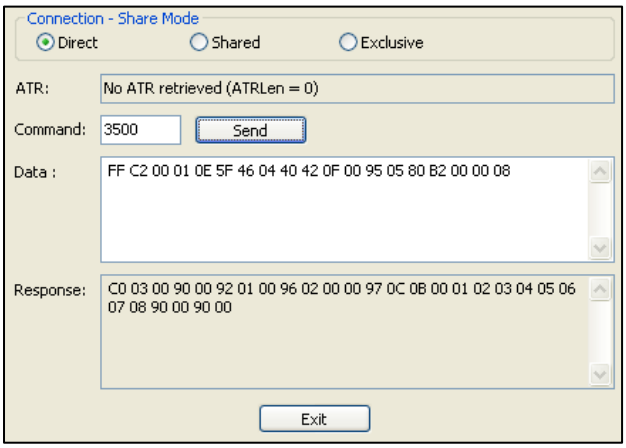
   応答：**C0 03 00 90 00 90 00**



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

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

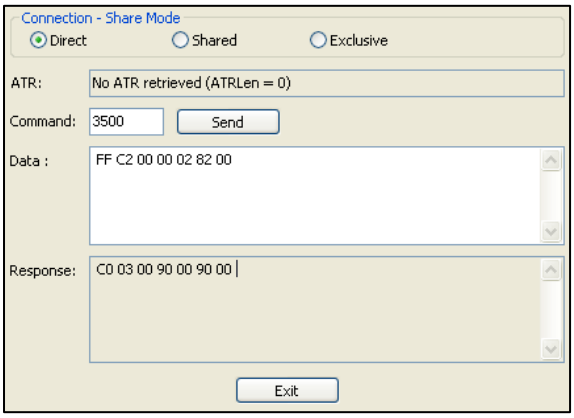
   応答：**C0 03 00 90 00 92 01 00 96 02 00 00 97 0C [カードの応答] 90 00**



6. 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**



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

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

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

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

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

その中：

キー構造 1 バイト。

00h = キーが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 – 01h = キーを保存するための失いやすいキーのメモリ。リーダーが PC から切断された時、キーが消えます。失いやすいキーは 2 つが設けられるので、異なるセッションのセッション鍵として使用することができます。

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きーの値。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：

// 揮発性のメモリに 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}

6.2.4.2. 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ーられているキーで MIFARE Classic（1K/4K）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 01 h	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。例えば：セクター00h がブロック{00h、01h、02 h と 03h}を含めている；セクター01h がブロック{{04h、05h、06 h と 07h}を含めている；最後のセクター0Fh がブロック{{3Ch、3Dh、3E h と 3 Fh}を含めている。当ブロックが成功認証されると、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：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。失いやすいキーは 2 つが設けられるので、異なるセッションのセッション鍵として使用することができます。

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

応答	データ出力	
結果	SW1	SW2

Load Authentication Keys 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。

セクター (16 個のセクター，各セクターには 4 個の連続的なブロックが含めている)	データブロック (3 個のブロック，各には 16 バイト)	トレーラーブロック (1 個のブロック，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10　　：MIFARE Classic 1K カードのメモリマップ

セクター (32 個のセクター，各セクターには 4 個の連続的なブロックが含めている)	データブロック (3 個のブロック，各には 16 バイト)	トレーラーブロック (1 個のブロック，16 バイト)
セクター0	00h – 02h	03h
セクター1	04h – 06h	07h
..	..	..
..	..	..
セクター30	78h – 7Ah	7Bh
セクター31	7Ch – 7Eh	7Fh

2 KB

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

2 KB

表11 ：MIFARE Classic 4K カードのメモリマップ

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

512 bits
or
64 bytes

表12 :MIFARE Ultralight® カードのメモリマップ

例：

//{TYPE A, キーナンバー00h}によりブロック 04h を認証します。PC/SC V2.01、廃止

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

// {TYPE A,キーナンバー00h}によりブロック 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。

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

複数のデータブロックを PICC カードから取り出すことに使われます。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：

ブロック番号 1 バイト。開始ブロック

読み取られていないバイト **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 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 1 6 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； 1 5 個の連続のブロック）

例 1：10h（16 バイト）。開始ブロックだけ（シングルブロックモード）

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

6.2.4.4. 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 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。
- 読み取られていない 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）
- 読み取られていない 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； 1 5 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（シングルブロックモード）

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ます。トレーラーブロックは複数のブロックモードでアクセスされません。単一のブロックモードを使用してください。

Update Binary Block 応答コード（2 バイト）

結果	SW1	SW2	意味
成功	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：

// 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

6.2.4.5. 数値ブロックの操作コマンド増加、減少、格納)[Value Block Operation (INC, DEC, STORE)]

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

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

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

その中：

ブロック番号 1 バイト。操作されていない数値のブロック

VB_OP 1 バイト。

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

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

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

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

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB
FFh	FFh	FFh	FCh

例 2：Decimal 1 = {00h, 00h, 00h, 01h}

VB_Value			
MSB			LSB
00h	00h	00h	01h

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

応答	データ出力	
結果	SW1	SW2

Value Block Operation 応答コード

結果	SW1	SW2	意味
成功	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。

6.2.4.6. 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る。数値ブロック対しての操作のみに適用する。

Read Value BlockAPDU 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 応答フォーマット（4 + 2 バイト）

応答	データ出力		
結果	数値 {MSB ..LSB}	SW1	SW2

その中：

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

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

数値			
MSB			LSB
FFh	FFh	FFh	FCh

例 2：Decimal 1 = {00h, 00h, 00h, 01h}

数値			
MSB			LSB
00h	00h	00h	01h

Read Value Block コマンドの応答コード

結果	SW1	SW2	意味
成功	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。

6.2.4.7. 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	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：

//数値"1"を数値ブロック 05h にストアーします。

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

// 数値ブロック 05h を読み取ります。

APDU = FF B1 00 05 04h

//数値をブロック 05h からブロック 06h にコピーする。

APDU = FF D7 00 05 02 03 06h

//ブロック 05h の値を 5 にインクリメントする。

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

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

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。ACR1255U-J1 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。ACR1255U-J1 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 及び 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 **MIFARE® Classic （1K/4K）メモリカードの PICC コマンド**を参照してください。

ISO 7816-4 APDU フォーマット

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

ISO 7816-4 仕様の応答データフォーマット（データ+2 バイト）

応答	データ出力		
結果	応答データ	SW1	SW2

一般的な ISO 7816-4 コマンドの応答コード

結果	SW1	SW2	意味
成功	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。

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。

これを実行するように：

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 の応答データ = 00 00 00 00

   ATQB のプロトコル情報 = 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]

6.2.6. MIFARE DESFire タグを読み取り(ISO14443-4)

MIFARE® DESFire®は ISO7816-4 APDU 包装モードとネイティブモードをサポートできます。MIFARE® DESFire®タグが活性化されると、MIFARE® DESFire®タグに送られた最初の APDU が"コマンドモード"を決定します。例えば最初の APDU が"ネイティブモード"を選択したら、残りの APDU は必ず"ネイティブモード"です。それと同じ、最初の APDU が"ISO 7816-4 APDU 包装モード"を選択する場合、残りの APDU は必ず"ISO 7816-4 APDU 包装モード"です。

例 1：MIFARE® DESFire® ISO 7816-4 APDU 包装

//ISO 14443-4 Type A PICC (DESFire)から 8 バイトの乱数を読み取ります

APDU = {90 0A 00 00 01 00 00}

CLA = 90h; INS = 0Ah (DESFire Instruction); P1 = 00h; P2 = 00h

Lc = 01h; Data In = 00h; Le = 00h (Le = 00h 最大の長さを示す)

応答：7B 18 92 9D 9A 25 05 21 [$91AF]

ステータスコード{91 AF}は DESFire 仕様で定義されています。詳細については、DESFire 仕様を参照してください。

例 2：MIFARE® DESFire®フレームレベルの連鎖（ISO 7816 APDU 包装モード）

// この例では、アプリケーションは"フレームレベルの連鎖"と関わっています。

// DESFire カードのバージョン番号を取得するために：

ステップ 1：APDU {90 60 00 00 00}を送信して、最初のフレームを取得します。INS=60h

応答：04 01 01 00 02 18 05 91 AF [$91AF]

ステップ 2：APDU {90 AF 00 00 00}を送信して、二番目のフレームを取得します。INS=AFh

応答：04 01 01 00 06 18 05 91 AF [$91AF]

ステップ 3：APDU {90 AF 00 00 00}を送信して、最後のフレームを取得します。INS=AFh

応答：04 52 5A 19 B2 1B 80 8E 36 54 4D 40 26 04 91 00 [$9100]

6.2.7. FeliCa タグのアクセス

FeliCa タグのアクセスコマンドは PC/SC タグおよび MIFARE カードのアクセスコマンドはちょっと違う。このコマンドは FeliCa 基準に準拠して、ヘッダが追加されています。

FeliCa コマンドのフォーマット

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

FeliCa の応答データフォーマット（データ 2 バイト）

応答	データ出力
結果	応答データ

例のメモリブロックデータの読み取り

1. FeliCa と接続。

   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。

   CMD = FF CA 00 00 00h

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

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

3. FeliCa コマンドアクセス。

   例：メモリブロックデータの「読み取り」

   CMD = 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 = メモリブロックデータ

6.3. 周辺デバイス制御

リーダーの周辺機器制御コマンドはブルートゥースモードで Escape コマンド（0x6B）を介して、USB モードで PC_to_RDR_Escape コマンドを介して実現されます。

PC モード：

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*bMessageType*	1	6Bh	-
1	*dwLength*	4	-	このメッセージの abData データフィールドのサイズ
5	*bSlot*	1	-	このコマンドのスロット番号を識別します。
6	*bSeq*	1	-	コマンドのシーケンス番号
7	*abRFU*	3	-	保留して将来使います。
10	*abData*	バイト配列	-	CCID に送信されるデータブロック。

abData はコマンドを示す

ブルートゥースモードで：

コマンドのフォーマット

オフセット	データフィールド	大きさ	数値	説明
0	*CmdMessageType*	1	6Bh	-
1	*Length*	2	-	データ長さ
3	*Slot*	1	00h	-
4	*Seq*	1	00h	番号
5	*Param*	1	00h	パラメーター
6	*Checksum*			CheckSum はコマンド中でのすべての XOR 値を示す
7	*Data*	N	-	データ

Data はコマンドを示す。

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

このコマンドはファームウェアのバージョンを入手する時に使われます。

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

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

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

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

例：

応答 = E1 00 00 00 14 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

ファームウェアのバージョン番号（HEX） = 41 43 52 31 32 35 35 55 2D 4A 31 20 53 57 56 20 31 2E 30 35

ファームウェアのバージョン番号（ASCII） = “ACR1255U-J1 SWV 1.05”

6.3.2. シリアルナンバーを取得する（Get Serial Number）

このコマンドはリーダーのシリアル番号を取得する時に使われます。

Get Serial Number のコマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Get Serial Number	E0h	00h	00h	47h	00h

Get Serial Number の応答フォーマット（5 バイト）

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	シリアルナンバー

例：

応答 = E1 00 00 00 0C 52 52 34 33 31 2D 30 30 30 30 31 36

シリアルナンバー(HEX) = 52 52 34 33 31 2D 30 30 30 30 31 36

シリアルナンバー(ASCII) = “RR431-000016”

6.3.3. 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 コマンドフォーマット（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	LED	色	説明
Bit 0	1	緑	1 = ON；0 = OFF
Bit 1	1	赤	1 = ON；0 = OFF
Bit 2	2	青	1 = ON；0 = OFF
Bit 3	2	赤	1 = ON；0 = OFF
Bit 4 - 7		RFU	RFU

6.3.4. 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	LED	色	説明
Bit 0	1	緑	1 = ON；0 = OFF
Bit 1	1	赤	1 = ON；0 = OFF
Bit 2	2	青	1 = ON；0 = OFF
Bit 3	2	赤	1 = ON；0 = OFF
Bit 4 - 7		RFU	RFU

6.3.5. 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

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

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

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：この設置は失いにくいキーのメモリに保存されます。（ファームウェア *2.03.xx* 以降から）

Bit4 と *Bit5* 操作オプションが *2.04.xx* 以降のファームウェアに提供されます。

Set LED and Buzzer Status Indicator Behaviors 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	デフォルト操作

操作（1 バイト）

操作	モード	説明
Bit 0	充電状態 LED	充電状態表示 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 インタフェース活性化状態表示** 1 = 有効；0 = 無効
Bit 3	カード挿入イベントブザー	カード検出ごとにビップ音をする 1 = 有効；0 = 無効
Bit 4	カード外しイベントブザー	カード外しを検出するごとにビップ音をする 1 = 有効；0 = 無効 （この機能を使うために *Bit3* を有効する必要があります）
Bit 5	カードリーダーパワーオンのブザー	カードリーダーがパワーオン際ビップする 1 = 有効；0 = 無効
Bit 6	RFU	

操作	モード	説明
Bit 7	カードに操作する時 LED が点滅します。	カードを読み書きする時 LED が点灯します。

注：

(1) 操作のデフォルト値 *= 8Fh*

(2) *USB* または *Bluetooth* モードでは、充電ステータスがオフに設定されるまたはリーダーが完全に充電されている場合を除き、充電中は赤い *LED* インジケーターが常に点灯します。

(3) *Bluetooth* モードでは、青色の *LED* インジケーター（*LED 2*）が連続的に点滅し、変更できません。

(4) *USB* モードでは、ポーリングステータス設定がオフでない限り、緑色の *LED* インジケーター（*LED 2*）は常に点灯です。

Set LED and Buzzer Status Indicator Behaviors の応答フォーマット（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

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

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 *Bit4* と *Bit5* 操作オプションが *2.04.xx* 以降のファームウェアに提供されます。

Set LED and Buzzer Status Indicator Behaviors 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 Behaviors	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	充電状態 LED	充電状態表示 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 インタフェース活性化状態表示** 1 = 有効；0 = 無効
Bit 3	カード挿入イベントブザー	カード検出ごとにビップ音をする 1 = 有効；0 = 無効
Bit 4	カード外しイベントブザー	カード外しを検出するごとにビップ音をする 1 = 有効；0 = 無効 （この機能を使うために *Bit3* を有効する必要があります）
Bit 5	カードリーダーパワーオンのブザー	カードリーダーがパワーオン際ビップする 1 = 有効；0 = 無効
Bit 6	RFU	
Bit 7	カードに操作する時 LED が点滅します。	カードを読み書きする時 LED が点灯します。

注：操作のデフォルト値 = *8Fh*

6.3.8. 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。エネルギを節約するために、PICC が活動していない、または PICC が見つからない時、いつでもアンテナフィールドをオフにするための特別なモードが設けられている。省電力モードで、リーダーはもっと少ない電流を消費します。

注：この設置は失いにくいキーのメモリに保存されます。（ファームウェア *2.03.xx* 以降から）

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

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

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

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

ポーリング設定（1 バイト）

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

注：ポーリング設置のデフォルト値 = *8Bh*

提示：

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

   注釈：dle 消費電流=PICC が活性化されていない。
3. リーダーは自動的に*"ISO14443A-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。

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

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

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

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

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

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

ポーリング設定（1 バイト）

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

注：ポーリング設置のデフォルト値 = *8Bh*

6.3.10. PICC 操作のパラメ—タ—を設定する（Set PICC Operating Parameter）

このコマンドは PICC 操作のパラメ—タ—を設定するために使われます。

注：この設置は失いにくいキーのメモリに保存されます。（ファームウェア *2.03.xx* 以降から）

Set the 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 the PICC Operating Parameter	E0h	00h	00h	20h	01h	操作パラメ—タ—

Set the 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—タ—

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	Calypso		1 = 検出 0 = スキップ
Bit 6	SRIX		1 = 検出 0 = スキップ
Bit 7	RFU	RFU	RFU

注：操作のデフォルト値 = *7Fh*

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

このコマンドは PICC 操作のパラメーターを検査するために使用されます。

Read the 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 the PICC Operating Parameter	E0h	00h	00h	20h	00h

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

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

操作パラメーター（1 バイト）

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

注：操作のデフォルト値 = *7Fh*

6.3.12. 自動的な PPS を設定する（Set Auto PPS）

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

注：この設置は失いにくいキーのメモリに保存されます。（ファームウェア *2.03.xx* 以降から）

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	最大の Tx 速度	最大の Rx 速度

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	最大の Tx 速度	現在の Tx 速度	最大の Rx 速度	現在の Rx 速度

その中：

最大の Tx 速度 1 バイト。最大の通信速度。

最大の Rx 速度 1 バイト。最大の受信速度。

現在の Tx 速度 **1 バイト。**現在の通信速度。

現在の Rx 速度 **1 バイト。**現在の受信速度。

値： 00h = 106 Kbps
01h = 212 Kbps
02h = 424 Kbps

注：デフォルトに *00h* に設定されています。

注釈：

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

6.3.13. 自動的な PPS を読み取る（Read Auto PPS）

このコマンドは現在の自動的な PPS の設置を検査するために使用されます。

Read Auto PPS コマンドフォーマット（5 バイト）

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

Read Auto PPS 応答フォーマット（9 バイト）

応答	CLA	INS	P1	P2	Le	データ出力			
結果	E1	00h	00h	00h	04h	最大の Tx 速度	現在の Tx 速度	最大の Rx 速度	現在の Rx 速度

その中：

最大の Tx 速度 1 バイト。最大の通信速度。

最大の Rx 速度 1 バイト。最大の受信速度。

現在の Tx 速度 **1 バイト。**現在の通信速度。

現在の Rx 速度 **1 バイト。**現在の受信速度。

値： 00h = 106 Kbps

01h = 212 Kbps

02h = 424 Kbps

注：デフォルトに *00h* に設定されています。

6.3.14. アンテナフィールド制御（Antenna Field Control）

このコマンドはアンテナフィールドを ON/OFF にする時に使われます。

Antenna Field 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
Antenna Field Control	E0h	00h	00h	25h	01h	状態

Antenna Field 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	状態

その中：

**状態　　1 バイト　　**。

01h = アンテナフィールドを ON にする

00h = アンテナフィールドを OFF にする

注：アンテナフィールドを ON にする前に、自動的な PICC のポーリングは OFF 状態ではなければなりません。

6.3.15. アンテナフィールドの状態を読み取る（Read Antenna Field Status）

このコマンドはアンテナフィールドの状態を検査するために使用されます。

Read Antenna Field Status コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read Antenna Field Status	E0h	00h	00h	25h	00h

Read Antenna Field 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	状態

その中：

状態　1 バイト。

00h = PICC オフ

01h = PICC 空いてる [非接触式のタグへのポリング準備が出来ていますが、このタイプのタグが検出されていません]

02h = PICC レディー [PICC 請求が成功で（ISO 14443 参照）、すなわち、非接触タグが検出されました。]

03h = PICC 選定された [PICC が成功に選定された（ISO 14443 を参照）]

04h = PICC 活性化された [PICC が成功に活性化された（ISO 14443 を参照）、APDU との交換の準備が出来ています]

6.3.16. スリープモード設置オプション（Set Sleep Mode Option）

60 秒後に何も操作しない場合、デフォルトでは、リーダーがスリープモードに入ます。

このコマンドはデバイスがスリープモードに入る前にの時間間隔を設定する時に使用されます。

注：この設置は失いにくいキーのメモリに保存されます。

Set Sleep Time Interval コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Set Sleep Time Interval	E0h	00h	00h	48h	時間間隔

Set Sleep Time Interval 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	時間間隔

その中：

時間間隔　　1 バイト

00h = 60 秒

01h = 90 秒

02h = 120 秒

03h = 180 秒

04h = Disable

6.3.17. スリープモード読み取りオプション（Read Sleep Mode Option）

このコマンドはデバイスがスリープモードに入る前にの時間間隔を確認する時に使用されます。

注： 2.03.xx 以降のバジョンのみに適用します。

Read Sleep Time Interval コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Read Sleep Time Interval	E0h	00h	00h	50h	00h

Read Sleep Time Interval 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	Time

その中：

時間間隔　　1 バイト

00h = 60 秒

01h = 90 秒

02h = 120 秒

03h = 180 秒

04h = スリープモードなし

6.3.18. TX パワーのコマンドを変更する（Change Tx Power command）

ブルートゥースの通信パワーを設定するために使われます。

注：この設置は失いにくいキーのメモリに保存されます。（ファームウェア *2.03.xx* 以降から）

Change Tx Power コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	49h	Tx パワー

Change Tx Power 応答フォーマット（5 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	Tx パワー

その中：

Tx パワー 1 バイト

00h = -23 dBm (デフォルト)、距離： ~3 メーター

01h = -6 dBm、距離： ~7 メーター

02h = 0 dBm、距離： ~17 メーター

03h = 4 dBm、距離： ~25 メーター

6.3.19. Tx パワーを読み取る（Read Tx Power Value）

ブルートゥースの通信パワーを読取るために使われます。

注： 2.03.xx 以降のバジョンのみに適用します。

Read Tx Power Value コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Set Tx power	E0h	00h	00h	51h	00h

Read Tx Power Value 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	Tx パワー

その中：

Tx パワー 1 バイト

00h = -23 dBm (デフォルト)

01h = -6 dBm

02h = 0 dBm

03h = 4 dBm

6.3.20. 顧客マスターキーリライトー（Customer Master Key Rewrite）

顧客マスターキーを設置する時にこのコマンドを使用します。

Customer Master Key reset コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Customer Master Key reset	E0h	00h	00h	60h	00h

Customer Master Key reset 応答フォーマット（21 バイト）（成功）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	10h	16 バイトの乱数

Customer Master Key rewrite コマンドフォーマット（36 バイト）

コマンド	CLA	INS	P1	P2	Lc
Customer Master Key rewrite	E0h	00h	00h	61h	16 バイトの暗号化された乱数 + 16 バイトの暗号化された新しいマスターキー

Customer Master Key rewrite 応答フォーマット（7 バイト）（成功）

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

Customer Master Key rewrite 応答フォーマット（5 バイト）（失敗）

応答	CLA	INS	P1	P2	Le
結果	E1h	00h	00h	00h	00h

例：顧客マスターキーを書き込む時、

リーダーによって生成された乱数は “11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66”

ホストデバイス → カードリーダー: (Customer Master Key Reset リクエスト)

Frame1 “E0 00 00 60 00”

カードリーダー → ホストデバイス: (Customer Master Key Reset コマンド応答)

Frame1 “E1 00 00 00 10 11 22 33 44 55 66 77 88 99 00 11 22 33 44 55 66”

ホストデバイス → カードリーダー: (Rewrite Master Key コマンドリクエスト)

Frame1 “E0 00 00 61 xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx”

カードリーダー → ホストデバイス: (Rewrite Master Key コマンド応答)

Frame1 “E1 00 00 00 02 90 00”

応答	説明
E1 00 00 00 02 90 00	成功
E1 00 00 00 00	失敗

6.3.21. バッテリー残量を取得（Get Battery Level）

バッテリー残量を取得するためにこのコマンドを使います（ブルートゥースモードのみ適用）

注： *2.03.xx* 以降バージョンのファームウェアのみ適用します。またカードリーダーがブルートゥースモードに設置される必要があります。.

Get Battery Level コマンドフォーマット（5 バイト）

コマンド	CLA	INS	P1	P2	Lc
Get Battery Level	E0h	00h	00h	52h	00h

Get Battery Level コマンドフォーマット（6 バイト）

応答	CLA	INS	P1	P2	Le	データ出力
結果	E1h	00h	00h	00h	01h	バッテリー残量

その中：

バッテリー残量　1 バイト

100 = 100%レベル

90 = 90%レベル

80 = 80%レベル

70 = 70%レベル

60 = 60%レベル

50 = 50%レベル

40 = 40%レベル

30 = 30%レベル

20 = 20%レベル

15 = 15%レベル

6.3.22. PICC タイプ読み取り（Read PICC Type）

このコマンドは現在の PICC タイプを確認するために使用されます。

注： 2.04.xx 以降のバジョンのみに適用します。

Read PICC Type 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 Type	E0h	00h	00h	35h	00h

Read PICC Type 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	02h	カードタイプ	カードステータス

その中：

カードのタイプ　1 バイト

CCh = ない

04h = Topaz

10h = Mifare

11h = Felica 212 Kbps

12h = Felica 424 Kbps

20h = ISO 14443-4 A タイプ

23h = ISO 14443-4 B タイプ

25h = Calypso

28h = Srix

カード状態　1 バイト

00h = PICC パワーダウン（タグが検出されていません）

他 = PICC が検出（非接触タグが検出されました。）