Advanced Card Systems Ltd.
Card & Reader Technologies

ACR3x
モバイルカードリーダー

リファレンスマニュアル　V1.06

事前に告知なく変更される場合があります

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

改定履歴

リリース日付	改訂説明	バージョン
2014/06/16	• 初回発布	1.00
2014/09/26	• 2.0 項更新ー特性 • 9.0 項追加ー非接触カードコマンド	1.01
2015/02/25	• 10.0 項更新ー極秘データ導入方法	1.02
2015/07/27	• 2.0 項更新ー特性	1.03
2015/05/26	• 2.0 項更新ー特性 • 9.2.1 項更新ー認証キーアップロード	1.04
2019/01/11	• ACR32 を取り外し、ACR3201 と交換してください。	1.05
2019/10/30	• 2.0 項更新ー特性 • 5.3 項更新ーMicro USB インターフェース	1.06

目次

図示目次

チャート目次

1.0. 紹介

ACR3xはモバイルカードリーダーはモバイルデバイスと磁気カード/接触カード/非接触式カードが通信する用のインターフェースです。カードのタイプによって、コマンドと通信プロトコルが違っていますが、ACR3x モバイルカードリーダーはモバイルデバイスとカードの間の均一なインターフェースを確立します。
3.5mm オーディオジャックインターフェースによって ACR3x はモバイルデバイスと接続します。それで、カードリーダー中のデコーダによってカード中の情報を読み取って、スマートフォンやタブレット PC などのモバイルデバイスに送信することができます。また、セキュリティを高めるために、カード中の情報をバックエンドサーバに送信する前に、AES-128 暗号化アルゴリズムに通じて暗号化します。
本文は ACR3x のハードウェアとソフトウェアのデザインを紹介すると同時に、ACR3x とモバイルデバイスが通信する時に使われるコマンドをリスト化します。

1.1. 専門用語の定義

略語	説明
ACS キー	リセットを実行する時認証用のキーです。このキーはファームウェアにハードコードされているため、コマンドメッセージで変更することはできません。また、このキーは ACS だけによって安全に設定されなければなりません。
AES	高度な暗号基準
AES キー	AES-128　CBC モードで磁気ストライプトラックデータを暗号化に使用されるキーです。このキーは顧客が変更することができます。
カスタム ID	顧客が設定された 10 バイトの識別コードで、顧客が変更することができます。
顧客マスターキー	このキーは顧客により割り当てられ、AES キー、新しい顧客マスターキー、カスタム ID および DUKPT オプション導入する際に使われます。DUKPT 初期化を実行する前に ACR3x に認証する時にも用います。顧客がこのキーを変更することができます。
デバイス ID	ACR3x で使われた MCU のユニークな識別コードです。(8 バイト)この ID を使って、カスタム ID もしくは DUKPT の初期化データを導出します。MCU メーカーはこのキーを MCU 内にハードコードして、どんな状況でも変更することはできません。
マスタリセット	工場から出荷時の設定を復元するに相当するものです。マスタリセットを実行すると、フラッシュメモリ中の全てのデータが削除されて、デフォルト値に回復します。
マスタリセットセッション鍵	マスタリセットを実行するために、相互認証してから生成したユニークなキーです。
セッション鍵	機密データを導入するために、相互認証してから生成したユニークなキーです。
TDES	トリプル・データ暗号化規格

表1　:専門用語の定義

2.0. 特性

2.1. ACR31

- 3.5mm オーディオジャックインタフェース
- 電源:
  - CR2016 電池搭載
- スワイプカードリーダー:
  - カードデータの 2 トラックまで読み込み
  - 双方向読み取り可能
  - AES-128 暗号化アルゴリズムサポート
  - DUKPT 鍵管理システムサポート
  - ISO 7810/7811 磁気カードサポート
  - 高保磁力、低保磁力の磁気カードサポート
  - JIS1 および JIS2 対応
- Android™ 2.3 と以降のバージョンサポート [1]
- iOS 5.0 と以降のバージョンサポート [2]
- 以下の規格に準拠:
  - CE
  - FCC
  - VCCI
  - RoHS
  - REACH

1 ACS の Android ライブラリを使用
2 ACS の iOS ライブラリを使用します
***注:** 注釈:サポートしているデバイスのリストについて、www.acs.com.hk をアクセスしてください。*

2.2. ACR3201

- 3.5mm オーディオジャックインタフェース
- 電源:
  - o リチウムイオン電池から(Micro-USB インターフェースで充電)
- スマートカードリーダー:
  - o 非接触インターフェース:
    - ISO 7816 クラス A 、B、C の(5V、3V および 1.8V)カードをサポート
    - T = 0 または T = 1 プロトコルのマイクロプロセッサーカードをサポート
    - 様々のメモリカードサポート
    - PPS サポート (プロトコルとパラメータの選択)
    - 短絡保護保有
- スワイプカードリーダー:
  - o カードデータの 2 トラックまで読み込み
  - o 双方向読み取り可能
  - o AES-128 暗号化アルゴリズムサポート
  - o DUKPT 鍵管理システムサポート
  - o ISO 7810/7811 磁気カードサポート
  - o 高保磁力、低保磁力の磁気カードサポート
  - o JIS1 および JIS2 対応
- Android™ 2.0 と以降のバージョンサポート 3
- iOS 5.0 と以降のバージョンサポート 4
- 以下の規格に準拠:
  - o EN 60950/IEC 60950
  - o ISO 7816
  - o ISO 7811
  - o CE
  - o FCC
  - o RoHS
  - o REACH
  - o VCCI (日本)

3 ACS の Android ライブラリー使用;PC/SC および CCID に適用しません。
4 ACS の iOS ライブラリー使用;PC/SC および CCID に適用しません。
***注:** 注釈:サポートしているデバイスのリストについて、www.acs.com.hk をアクセスしてください。*

2.3. ACR35

- 3.5mm オーディオジャックインタフェース
- 電源:
  - o リチウムイオン電池から（Micro-USB インターフェースで充電します）
- スマートカードリーダー:
  - o 非接触インターフェース:
    - 内蔵アンテナを使って、通信距離は最大 30 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。5
    - 衝突防止機能保有（一枚のタグのみアクセス）
    - NFC サポート
      - カードリーダ/ライタモード
- スワイプカードリーダー:
  - o カードデータの 2 トラックまで読み込み
  - o 双方向読み取り可能
  - o AES-128 暗号化アルゴリズムサポート
  - o DUKPT 鍵管理システムサポート
  - o ISO 7810/7811 磁気カードサポート
  - o 高保磁力、低保磁力の磁気カードサポート
  - o JIS1 および JIS2 対応
- Android™ 2.0 と以降のバージョンサポート 6
- iOS 5.0 と以降のバージョンサポート 7
- 以下の規格に準拠:
  - o EN 6095/IEC 60950
  - o ISO 14443
  - o ISO 18092
  - o ISO 7811
  - o CE
  - o FCC
  - o RoHS
  - o REACH
  - o VCCI (日本)
  - o KC (韓国)

*5 Topaz は含むいません。*詳しい情報は、ACS までお問い合わせください。
6 ACS 定義された Android ライブラリを使用しています
7 ACS の iOS ライブラリを使用します
***注:** 注釈: サポートしているデバイスのリストについて、www.acs.com.hk をアクセスしてください。*

3.0. サポートしているカード

3.1. 磁気ストライプカード

ISO 7810/7811 高保磁力と低保磁力の磁気ストライプカードをサポート。

3.2. 接触式スマートカード

3.2.1. MCU カード

ACR3201 スマートカードリーダーは、ISO 7816 クラス A、クラス B、およびクラス C（5V、3V、および 1.8V）スマートカードをサポートし、T = 0 または T = 1 プロトコルに準拠する MCU カードを読み書きできます。
カードの ATR が専用の操作モードを指定すれば（TA2が存在している；TA2 のビット 5 は 0 でなければなりません）、しかし ACR3201 がこのモードをサポートできない場合、カードを交渉モードに設置します。交渉モードを設置できないと、ACR3201 がこのカードを拒否します。
カードの ATR が交渉のモード（TA2 が存在指定ない）および通信パラメータ（デフォルトパラメータじゃなくて）を指定すれば、ACR3201 がその通信パラメータを使用してみて、PPS を実行します。ACR3201 が PPS を拒否したら、デフォルトパラメ―タ―を使用します（F=372，D=1）。
上記のパラメ―タ―の意味について、ISO 7816-3 仕様を参照してください。

3.2.2. メモリカード

ACR3201 が様々のメモリカードをサポートしている、例：

- I2C バスプロトコルに準拠し、一回で 128 バイト/ページを書くことができるメモリカード（フリーメモリカード）、以下を含む：
  - Atmel®：AT24C01/02/04/08/16/32/64/128/256/512/1024
  - SGS-Thomson：ST14C02C、ST14C04C
  - Gemplus：GFM1K、GFM2K、GFM4K、GFM8K
- パスワードと認証によるセキュアなメモリ IC カード、以下を含む：
  - Atmel®：AT88SC153 および AT88SC1608
- インテリジェントな 1K バイトの EEPROM、カード書き込み保護機能付カード、以下を含む：
  - Infineon®：SLE4418、SLE4428、SLE5518 および SLE5528
- インテリジェント 256 バイトの EEPROM、書き込みのカードプロテクト機能付カード、以下を含む：
  - Infineon®：SLE4432、SLE4442、SLE5532 および SLE5542
- ‘104’タイプ EEPROM（読み取りオンリー型トークンカウンタカード、以下を含む：
  - Infineon®：SLE4406、SLE4436、SLE5536 および SLE6636
- インテリジェント 416 バイトの EEPROM、書き込みのカードプロテクト機能付カード、以下を含む：
  - Infineon®：SLE4404
- アプリケーションゾーンでのセキュリティロジックを使用したカード、以下を含む：
  - Atmel®：AT88SC101、AT88SC102 および AT88SC1003

3.3. 非接触スマートカード

ACR35 が様々の非接触カードとタグをサポートしています。例：

- ISO 14443 A タイプのカード
- ISO 14443 B タイプのカード
- ISO/IEC 18092（NFC）カード
- MIFARE® Classic 1K/4K カード
- FeliCa
- MIFARE Ultralight®
- MIFARE Ultralight® C
- MIFARE® DESFire® EV1

4.0. システムのブロックデザイン

4.1. ACR31

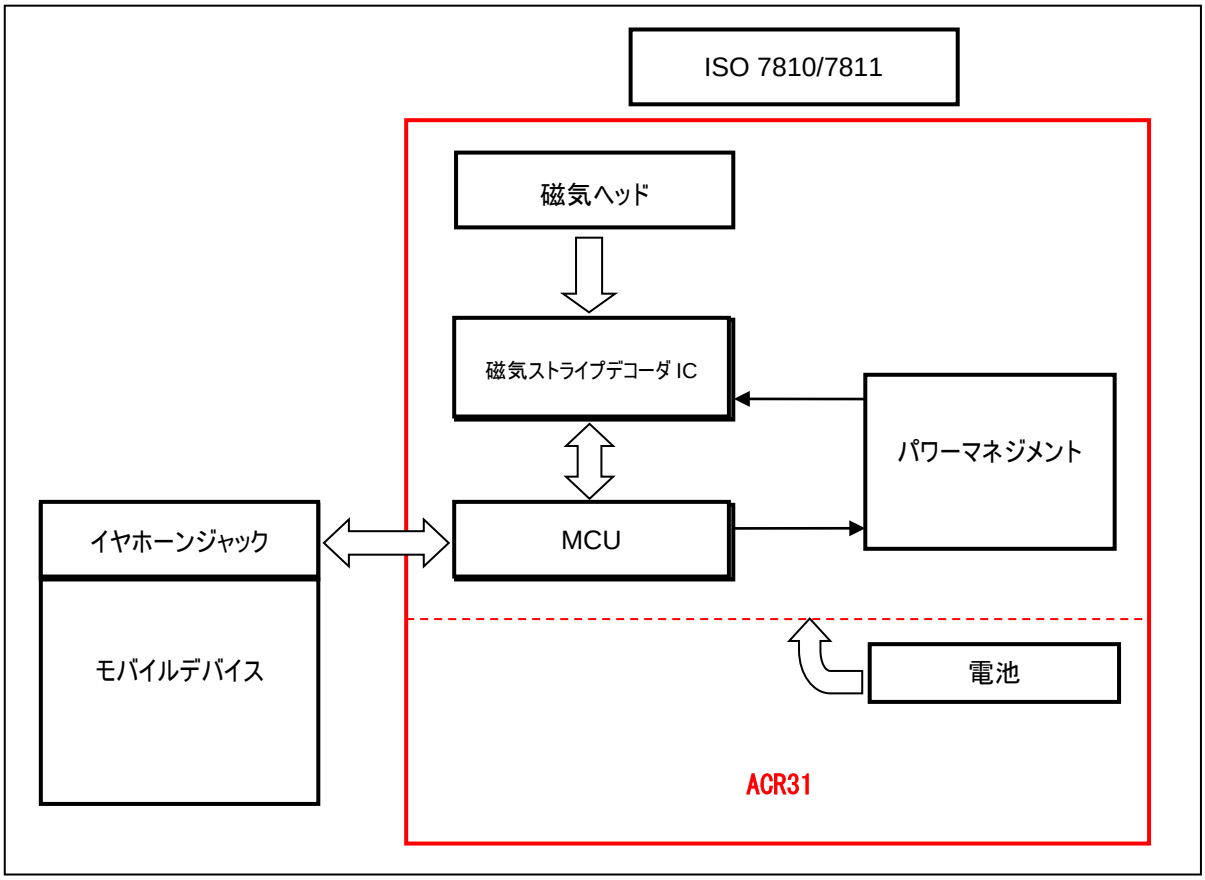

図1 ：ACR31 アーキテクチャ

4.2. ACR3201

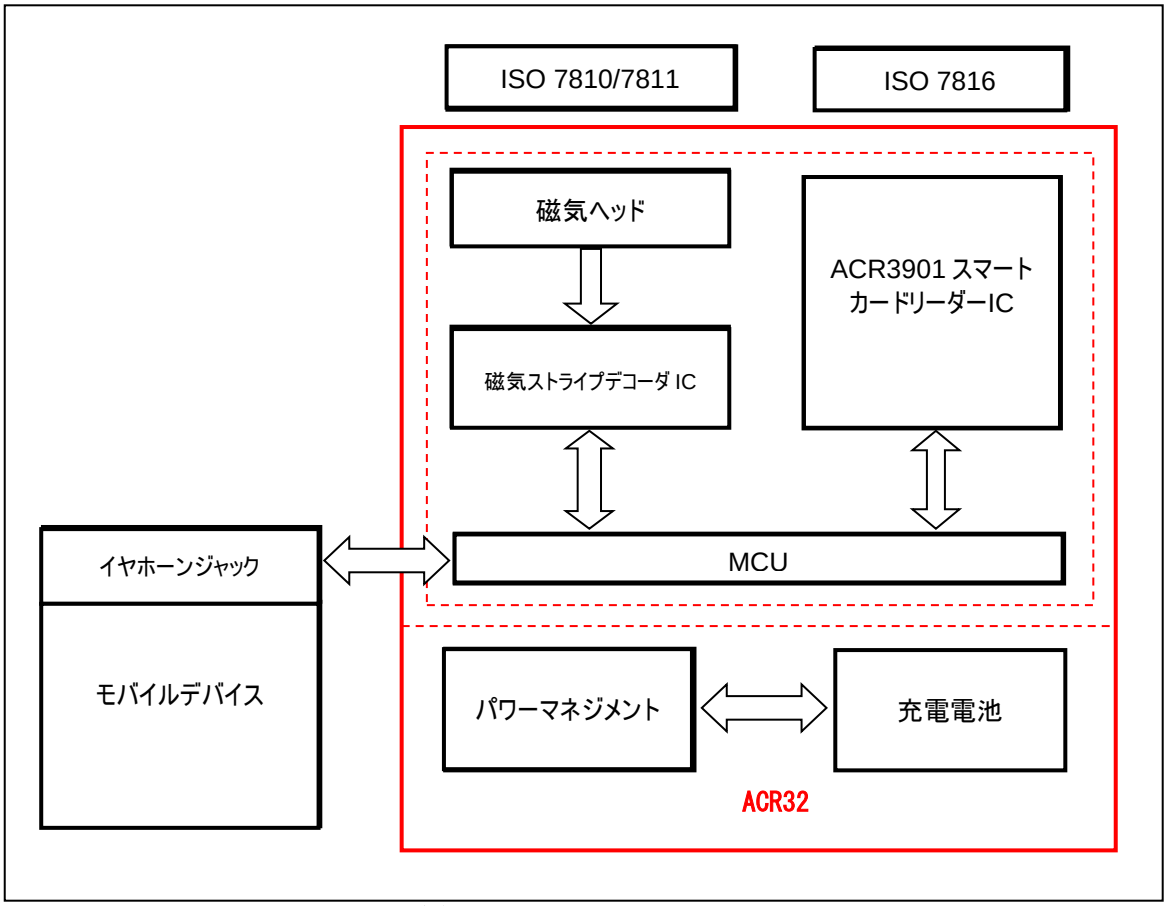

図2 ：ACR3201 アーキテクチャ

4.3. ACR35

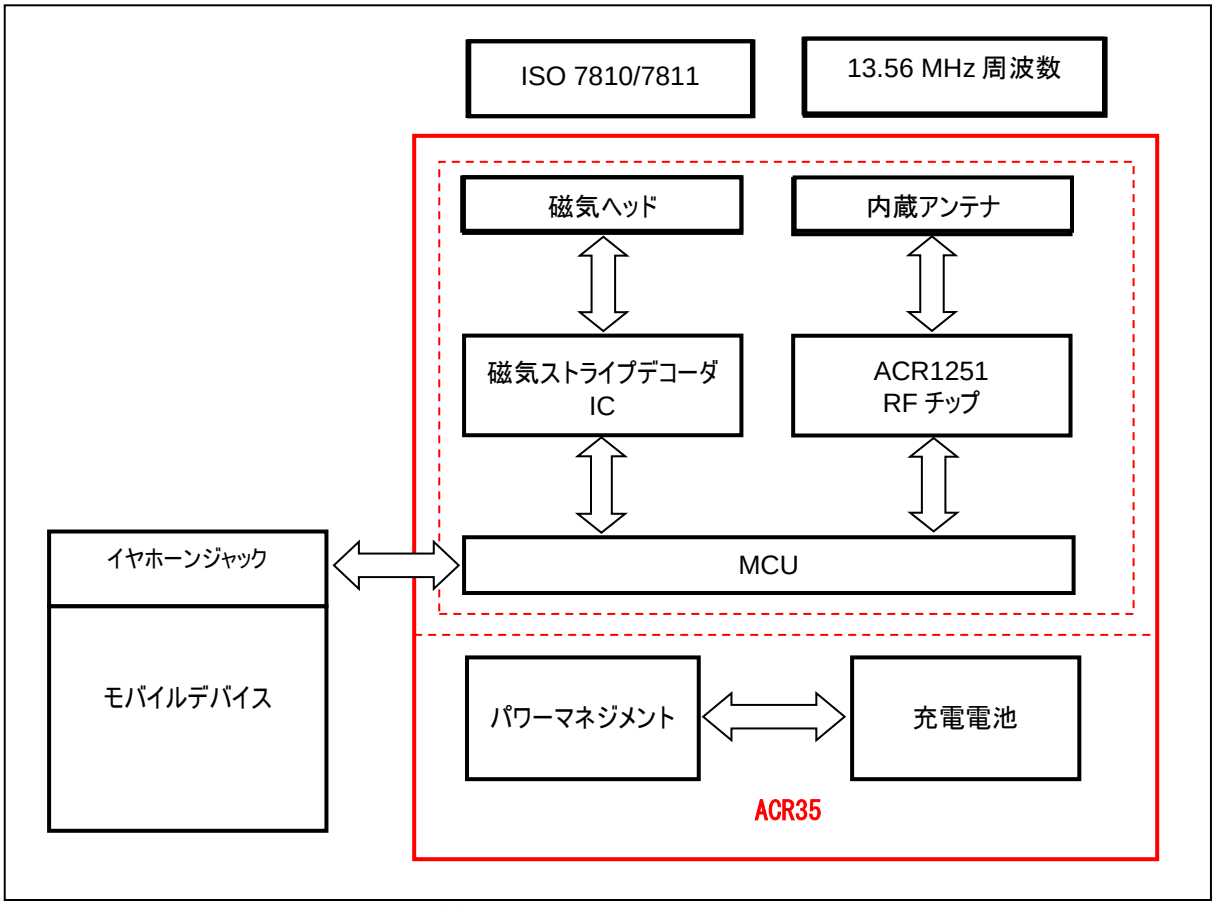

図3 : ACR35 アーキテクチャ

5.0. ハードウェアのデザイン

5.1. 電池

ACR31 は 90mAh 容量仕様の CR2016 で電源を供給しています。ACR3201 と ACR35 はリチウムイオンバッテリーで電源を供給しています。電池容量は 200mAh です。

5.2. LED ステータスインジケータ

異なる色は ACR3201 と ACR35 の違っているステータスを表示します。

- **緑の LED** – 動作中
- 赤の LED - 電池状態

5.3. Micro USB インターフェース

Micro USB インターフェースは充電ポートとして、ACR3201 と ACR35 をコンピュータと接続して充電します。

5.4. オーディオチャンネル

5.4.1. 通信パラメーター

ACR3x はオーディオチャンネルに通じてモバイルデバイスと接続を構築します。

ピン	信号	機能
1	左	ACR3x にデータを送信
2	右	デバイス信号を目覚めさせる
3	GND	GND
4	MIC	スマートフォンにデータを送信します

表2 : 3.5mm オーディオソケット配線

5.5. 磁気カード インターフーイ スパラメーター

ACR3x は全ての ISO 7810/7811 仕様に準拠している磁気カードを読み取れます。ISO 7810 仕様は各種のカードの物理特性を説明しますが、ISO 7811 仕様は識別に使用される記録技術を指定します。
高保磁力の磁気ストライプ（Hi-Co）の色は一般的に黒色であり、強い磁場（2750 エルステッド）で符号化されます。ストライプ上に符号化されたデータが外部磁場に曝されたときに消去される可能性が低いので、このカードはより丈夫です。磁気ヘッドにカードをスワイプすると、高保磁力の磁気ストライプはより大きな信号パルスを誘導することができて、より容易に検出されて、デコードされます。
低保磁力の磁気カードストライプ（Lo-Co）の色は一般的にブラウンであり、弱い磁場（300 エルステッド）で符号化されます。磁気ヘッドにカードをスワイプすると、低保磁力の磁気カードストライプが誘導できる信号パルスが高保磁力の磁気ストライプより小さいです。結果として、S/N が比較的に低いです。そして、彼らはノイズの干渉により弱いです。正しく信号をデコードするために、より洗練されたハードウェアサポートと信号処理アルゴリズムが必要とされています。
ハイ保磁力カードと低保磁力の磁気カードの磁場が違っているため、自動利得制御を用いた磁気ストライプデコーダ IC カードで、この 2 種類のカードのニーズを満たすように設計されています。

5.6. 接触式スマートカードインターフェース

ACR3201と挿入されたカードの間のインターフェースがISO 7816-3仕様プロトコルに準拠して、ACR3201の実用的な機能性を高めるために一定の制限や機能を拡張します。

5.6.1. スマートカード電源 VCC（C1）

挿入されたカードの消費電流は50mAよりも高くてはなりません。

5.6.2. プログラミング電圧 VPP（C6）

ISO7816-3仕様によると、スマートカードコンタクトC6（VPP）がスマートカードにプログラミング電圧を供給します。市場内のすべてのスマートカードがEEPROMベースであり、外部プログラミング電圧の供給の必要がないです。
ACR3201のコンタクトC6（VPP）が通常の制御信号として実装されました。このコンタクトの電気仕様は信号RST（コンタクトC2）の電気仕様と同じです。

5.6.3. カードタイプのセレクション

MCUベースカードに対して、リーダーがT=0またはT=1中から望ましいプロトコルを選びます。しかしながら、挿入されたカードは両方のプロトコルタイプをサポートできる場合は、リーダーがPPSを通じて、このセレクションを受け入れられて、実行します。プログラマベースのカードは、1つだけのプロトコルタイプ（T=0またはT=1）をサポートする時に、アプリケーションがどのプロトコルを選ぶことと関係なくて、リーダーは自動的にこのプロトコルタイプを選択します。

5.6.4. マイクロコントローラベースカードのためのインターフェース

マイクロコントローラベースカードはC1（VCC）、C2（RST)、C3（CLK）、C5（GND）およびC7（I/O）これらのコンタクトだけ使用します。4MHzの周波数がCLK信号（C3）に適用します。

5.6.5. カード引き裂き保護

電気入れる状態で、急に引き出されたカードを保護するために、ACR3xがメカニズムを提供しています。カードが取り外されている時、ACR3xとカード間の信号線への電力供給がすぐに非アクティブ化されます。原則として、電気的な損傷を回避するために、パワーダウンしてから、カードをリーダーから除去されるべきです。
注:*ACR3201は決してそれ自体で挿入されたカードへの電源供給に切り替わりません。制御のコンピュータがリーダーに送信した適切なコマンドを使用して、明示的にこれを行う必要があります。*

6.0. 通信プロトコル

ACR3x はスレーブデバイスであるために、ほとんどの操作はモバイルデバイスによって開始されます。バイルデバイスが送信したコマンドは、連続的なコマンドの要求と応答の交換の形で実施されます。さらに、以前の応答メッセージが受信されるまでに、新しい要求メッセージが待つはずです。

オーディオジャックインターフェースによって ACR3x はモバイルデバイスと接続して通信します。通信チャネルは双方向性です。オーディオジャックの MIC 端子を介してモバイルデバイスにデータを送信します、一方モバイルデバイスがオーディオジャックの右チャンネルを介してリーダーにコマンドを送信します。

動作していない時に、ACR3x はディープスリープモードに入ります。オーディオジャックの左チャンネルからモバイルデバイスのウェイクアップ信号を受信すると、ACR3x はスリープモードから戻して、モバイルデバイスに確認応答信号を返信します。ACR3x は、タイムアウト制限内に磁気ストライプカードのスワイプを待ちます。成功した後スワイプカードからデータを取得して、ACR3x は受信したカード情報を AES-128 で暗号化して、暗号化されたデータをモバイルデバイスに送り返します。リーダーがタイムアウト時間内にカードのスワイプ情報またはコマンドメッセージの取得に失敗した場合、ACR3x はモバイルデバイスに対応するステータスを送り返します。この後、バッテリ電力を節約するために ACR3x はディープスリープモードに戻ります。

ACR3x とモバイルデバイスとの間の通信プロトコルが直接信号給電を採用する前に、ACR3x から受信した信号は DC オフセットキャンセルフィルタを通じます。マンチェスターコーディング方式で（IEEE802.3 に準拠）送信すべきデータは 10 kHZ に設定されます。マンチェスター符号化方式には、データ伝送速度は常にクロック速度と一致しているために、約 10 kbps の最大ボーレートを達成できます。

モバイルデバイスと ACR3x 上の信号解釈は対応している入力波形をサンプリングに基づいている。サンプリング周波数は少なくとも、マンチェスターコーディング方式で使用されるクロック周波数の 2 倍です。信号をサンプリングした後、信号中に符号化されたデータは、論理ゼロクロスタイムにより受信することができます。

7.0. アプリケーション　プログラミング　インターフェース

ACR3x Android ライブラリ ACR3x iOS ライブラリ中の HTML ファイルを参照して下さい。
これらのライブラリは ACS ウェブサイトでダンロードできます。

8.0. 接触カードコマンド

本章は ACR3201 のメモリカードのコマンドについて紹介します。

8.1. メモリカード – 1、2、4、8 および 16 kilobit I2C カード

8.1.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

注釈:SCardConnect( ) API *によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	01h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.1.2. SELECT_PAGE_SIZE

このコマンドはスマートカードを読み取られるページサイズを選択します。デフォルト値は8バイトの書き込みページ。カードが削除されているか、またはリーダーの電源がオフになっている時に、デフォルト値にリセットされます。

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Page Size**
FFh	01h	00h	00h	01h	

その中:

Page size = 03h:8 バイトの書き込みページ

= 04h:16 バイトの書き込みページ

= 05h:32 バイトの書き込みページ

= 06h:64 バイトの書き込みページ

= 07h:128 バイトの書き込みページ

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.1.3. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B0h			

その中:

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h(エラーなしの場合)

8.1.4. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	Byte 1	…	Byte n
		MSB	LSB				
FFh	D0h						

その中:

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリに書き入れていないデータの長さ
Byte x メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.2. メモリカード – 32、64、128、256、512 および 1024 kilobit I2C カード

8.2.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	02h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.2.2. SELECT_PAGE_SIZE

このコマンドはスマートカードを読み取られるページサイズを選択します。デフォルト値は8バイトの書き込みページ。カードが削除されているか、またはリーダーの電源がオフになっている時に、デフォルト値にリセットされます。

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Page size**
FFh	01h	00h	00h	01h	

その中:

Data カードに送信されていない TPDU

Page size = 03h:8 バイトの書き込みページ
= 04h:16 バイトの書き込みページ
= 05h:32 バイトの書き込みページ
= 06h:64 バイトの書き込みページ
= 07h:128 バイトの書き込みページ

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.2.3. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**Byte Address**		**MEM_L**
		MSB	**LSB**	
FFh				

その中:

INS = B0h:32、64、128、256 および 512 kilobit IIC カード
= 1011 000*b:1024 kilobit IIC カード,
その中 * はアドレッシング 17 ビットの MSB を示しています。

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中：

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h（エラーなしの場合）

8.2.4. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	Byte 1	....	Byte n
		MSB	LSB				
FFh							

その中：

INS = D0h：32、64、128、256 および 512 kilobit IIC カード
= 1101 000*b：1024 kilobit IIC カード,
その中 * はアドレッシング 17 ビットの MSB を示しています。
Byte Address メモリカードのメモリアドレス位置
MEM_L メモリに書き入れていないデータの長さ
Byte x メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.3. メモリカード – Atmel® AT88SC153

8.3.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。それはまた、8 バイトのページの書き込みページサイズを選択します。

注釈:SCardConnect( ) API *によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	INS	P1	P2	Lc	Card Type
FFh	A4h	00h	00h	01h	03h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.3.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	INS	P1	Byte Address	MEM_L
FFh		00h		

その中:

INS = B0h:00b を読み取る
= B1h:01b ゾーンを読み取る
= B2h:10b ゾーンを読み取る
= B3h:11b ゾーンを読み取る
= B4h: ヒューズを読み取る

Byte Address メモリカードのメモリアドレス位置

MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	...	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ

SW1 SW2 = 90 00h(エラーなしの場合)

8.3.3. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	P1	Byte Address	MEM_L	Byte 1	....	Byte n
FFh		00h					

その中:

INS = D0h:00b ゾーンを書く
= D1h:01b ゾーンを書く
= D2h:10b ゾーンを書く
= D3h:11b ゾーンを書く

= D4h：ヒューズを書き入れる
Byte Address メモリカードのメモリアドレス位置
MEM_L メモリに書き入れていないデータの長さ
MEM_D メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.3.4. VERIFY_PASSWORD

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	P1	P2	Lc	Pw(0)	Pw(1)	Pw(2)
FFh	20h	00h		03h			

その中：
Pw(0),Pw(1),Pw(2) メモリカードに送信していないシークレットコード
P2 = 0000 00rpb
その中、2ビットの“rp”は比較されていないパスワードを示します
r = 0 ：パスワードを書く
r = 1 ：パスワードを読み取る
p：パスワード設定番号、
rp = 01：セキュリティコード。

応答データフォーマット

SW1	SW2 ErrorCnt
90h	

その中：
SW1 = 90h
SW2 (ErrorCnt) = エラー　カウンター。FFh は検証が正しいことを示している。00H はパスワードがロックされていることを示しています（最大再試行回数を超過した）。他の値は現在の認証が失敗したことを示しています。

8.3.5. INITIALIZE_AUTHENTICATION

コマンドのフォーマット

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	…	Q(7)
FFh	84h	00h	00h	08h				

その中：
Q(0),Q(1)…Q(7) ホストチャレンジ、8 バイト

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.3.6. VERIFY_AUTHENTICATION

コマンドのフォーマット

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Ch(0)**	**Ch(1)**	**…**	**Ch(7)**
FFh	82h	00h	00h	08h				

その中：

Ch(0),Ch(1)…Ch(7) ホストチャレンジ、8 バイト

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.4. メモリカード – Atmel® AT88C1608

8.4.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。それはまた、16 バイトのページの書き込みページサイズを選択します。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	04h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.4.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**Zone Address**	**Byte Address**	**MEM_L**
FFh				

その中:

INS = B0h:ユーザーゾーンを読み取る
= B1h:コンフィギュレーション·ゾーンまたはヒューズを読み取る

Zone Address = 0000 $0A_{10}A_9A_8$b、その中 A_{10} はゾーンアドレスの MSB です
= ヒューズの書き込みと関係ない

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である
= 1000 0000b ヒューズを読み取る

MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h(エラーなしの場合)

8.4.3. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**Zone Address**	**Byte Address**	**MEM_L**	**Byte 1**	**…**	**Byte n**
FFh							

その中:

INS = D0h:ユーザーゾーンを書く
= D1h:コンフィギュレーション·ゾーンまたはヒューズを書く

Zone Address = 0000 $0A_{10}A_9A_8$b、その中 A_{10} はゾーンアドレスの MSB です
= ヒューズの書き込みと関係ない

Byte Address = $A_7A_6A_5A_4\ A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である
= 1000 0000b：ヒューズを書く
MEM_L メモリに書き入れていないデータの長さ
Byte x メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.4.4. VERIFY_PASSWORD

コマンドのフォーマット

Pseudo-APDU								
CLA	INS	P1	P2	Lc	データ			
FFh	20h	00h	00h	04h	RP	Pw(0)	Pw(1)	Pw(2)

その中：
Pw(0),Pw(1),Pw(2) メモリカードに送信していないシークレットコード
RP = 0000 $rp_2p_1p_0$b
その中、4 ビットの“$rp_2p_1p_0$”は比較されていないパスワードを示します
r = 0 ：パスワードを書く
r = 1 ：パスワードを読み取る
$p_2p_1p_0$ ：パスワード設定番号。
($rp_2p_1p_0$ = 0111：セキュリティコード）

応答データフォーマット

SW1	SW2 ErrorCnt
90h	

その中：
SW1 = 90h
SW2 (ErrorCnt) = エラー　カウンター。FFh は検証が正しいことを示しています。00H はパスワードがロックされていることを示しています（最大再試行回数を超過した）。他の値は現在の認証が失敗したことを示しています。

8.4.5. INITIALIZE_AUTHENTICATION

コマンドのフォーマット

Pseudo-APDU								
CLA	INS	P1	P2	Lc	Q(0)	Q(1)	…	Q(7)
FFh	84h	00h	00h	08h				

その中：
Byte Address メモリカードのメモリアドレス位置
Q(0),Q(1)…Q(7) ホストチャレンジ、8 バイ

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.4.6. VERIFY_AUTHENTICATION

コマンドのフォーマット

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**Lc**	**Q1(0)**	**Q1(1)**	**…**	**Q1(7)**
FFh	82h	00h	00h	08h				

その中：

Byte Address メモリカードのメモリアドレス位置

Q1(0),Q1(1)…Q1(7) ホストチャレンジ、8 バイト

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.5. メモリカード – SLE4418/SLE4428/SLE5518/SLE5528

8.5.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	05h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.5.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**Byte Address**		**MEM_L**
		MSB	**LSB**	
FFh	B0h			

その中:

MSB Byte Address = 0000 00A_9A_8b はメモリカードのメモリアドレス位置である
LSB Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である
MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h(エラーなしの場合)

8.5.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE4428 および SLE5528)

このコマンドがプレゼンテーションエラーカウンタを読み取る時に使われます。

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**P2**	**MEM_L**
FFh	B1h	00h	00h	03h

応答データフォーマット

ERRCNT	DUMMY 1	DUMMY 2	SW1	SW2

その中：

ERRCNT エラーカウンター。FFh は最後の検証が正しいことを示している。00h はパスワードがロックされていることを示している（最大再試行回数を超過した）。他の値は最後の認証が失敗したことを示している。

DUMMY カードから読み取った 2 バイトのダミーデータ

SW1 SW2 = 90 00h（エラーなしの場合）

8.5.4. READ_PROTECTION_BIT

コマンドのフォーマット

Pseudo-APDU				
CLA	INS	Byte Address		MEM_L
		MSB	LSB	
FFh	B2h			

その中：

MSB Byte Address = 0000 00A_9A_8b はメモリカードのメモリアドレス位置である

LSB Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である

MEM_L カードから読み出される保護ビットの長さは 8 ビットの倍数です。最大数値は2です。

MEM_L = 1 + INT((ビットのナンバー - 1)/8)

例えば、メモリ 0010H から始まる 8 保護ビットを読み取るために、以下の擬似 APDU を発行する必要がある：

FF B2 00 10 01h

応答データフォーマット

PROT 1	…	PROT L	SW1	SW2

その中：

PROT y 保護ビットが含むいるバイト

SW1 SW2 = 90 00h（エラーなしの場合）

PROT バイト中で、保護ビットは以下のように並べている：

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

Px は応答データの BYTE x の保護ビットです。

‘0’バイトが書き込み保護されています

‘1’バイトは書き込むことができます

8.5.5. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	Byte 1	....	Byte N
		MSB	LSB				
FFh	D0h						

その中：

MSB Byte Address = 0000 00A_9A_8b はメモリカードのメモリアドレス位置である

LSB Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である

MEM_L メモリに書き入れていないデータの長さ

Byte x メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中：

SW1 、SW2 = 90 00h（エラーなしの場合）

8.5.6. WRITE_PROTECTION_MEMORY_CARD

コマンドで指定された各バイトは内部でカードに指定されたアドレス中のデータと比べる。一致した場合、対応している保護ビットが不可逆的に“0”にプログラムされています。

コマンドのフォーマット

Pseudo-APDU							
CLA	INS	Byte Address		MEM_L	Byte 1	....	Byte N
		MSB	LSB				
FFh	D1h						

その中：

MSB Byte Address = 0000 00A_9A_8b はメモリカードのメモリアドレス位置である
LSB Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である
MEM_L メモリに書き入れていないデータの長さ
Byte x バイト値がバイトアドレスから始まるカード内のデータと比較されます。BYTE 1 と Byte Address 中のデータを比べる；BYTE N と（Byte Address + N -1）中のデータが比べる。

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.5.7. PRESENT_CODE_MEMORY_CARD (SLE4428 および SLE5528)

SLE4428 と SLE5528 に書き込む操作を有効にするために、メモリカードにシークレットコードを提出する時に、このコマンドを使用します。以下の操作を実行します。：

1. プレゼンテーションエラーカウンタにビット‘1’を検索して、‘0’に変更する。
2. 指定されたシークレットコードをカードに提出します。
3. プレゼンテーションエラーカウンタを消去しようとします。

コマンドのフォーマット

Pseudo-APDU						
CLA	INS	P1	P2	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	00h	00h	02h		

その中：

CODE 2 バイトのシークレットコード（PIN）

応答データフォーマット

SW1	SW2 ErrorCnt
90h	

その中：

SW1 = 90h
SW2 (ErrorCnt) = エラー　カウンター。FFh は認証が成功したことを示している。00hはパスワードがロックされていることを示している（最大再試行回数を超過した）。他の値は現在の認証が失敗したことを示している。

8.6. メモリカード – SLE4432/SLE4442/SLE5532/SLE5542

8.6.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行する。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	06h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.6.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	B0h	00h		

その中:

Byte Address = $A_7A_6A_5A_4$ $A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である

MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	...	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ

SW1 SW2 = 90 00h(エラーなしの場合)

8.6.3. READ_PRESENTATION_ERROR_COUNTER_MEMORY_CARD (SLE4442 および SLE5542)

このコマンドがプレゼンテーションエラーカウンタを読み取る時に使われます。

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**P2**	**MEM_L**
FFh	B1h	00h	00h	04h

応答データフォーマット

ERRCNT	DUMMY 1	DUMMY 2	DUMMY 3	SW1	SW2

その中:

ERRCNT エラーカウンター。07h は最後の検証が正しいことを示している。00H はパスワードがロック

されていることを示している（最大再試行回数を超過した）。他の値は最後の認証が失敗したことを示している。

DUMMY 从卡片读取的 3 个字节的虚拟数据

SW1 SW2 = 90 00h（エラーなしの場合）

8.6.4. READ_PROTECTION_BITS

このコマンドは始めの 32 バイトの保護ビットを読み取る時に使われます。

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**P2**	**MEM_L**
FFh	B2h	00h	00h	04h

応答データフォーマット

PROT 1	PROT 2	PROT 3	PROT 4	SW1	SW2

その中：

PROT y 保護ビットが含むいるバイト

SW1 SW2 = 90 00h（エラーなしの場合）

PROT バイト中で、保護ビットは以下のように並べている：

PROT 1								PROT 2								…							
P8	P7	P6	P5	P4	P3	P2	P1	P16	P15	P14	P13	P12	P11	P10	P9	..	..	..	..	..	..	P18	P17

その中：

Px は応答データの BYTE x の保護ビットです。

‘0’バイトが書き込み保護されています

‘1’バイトは書き込むことができます

8.6.5. WRITE_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**Byte 1**	**....**	**Byte N**
FFh	D0h	00h					

その中：

Byte Address = $A_7A_6A_5A_4\ A_3A_2A_1A_0$b はメモリカードのメモリアドレス位置である

MEM_L メモリに書き入れていないデータの長さ

Byte x メモリカードに書き入れていないデータ

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.6.6. WRITE_PROTECTION_MEMORY_CARD

コマンドで指定された各バイトは内部でカードに指定されたアドレス中のデータと比べる。一致した場合、対応している保護ビットが不可逆的に“0”にプログラムされています。

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**Byte 1**	**....**	**Byte N**
FFh	D1h	00h					

その中：

Byte Address = 000A_4 $A_3A_2A_1A_0$b (00h - 1Fh) はメモリカードの保護メモリアドレス位置である
MEM_L メモリに書き入れていないデータの長さ
Byte x バイト値がバイトアドレスから始まるカード内のデータと比較されます。BYTE 1 と Byte Address 中のデータを比べる；BYTE N と（Byte Address + N -1）中のデータが比べる。

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.6.7. PRESENT_CODE_MEMORY_CARD (SLE4442 および SLE5542)

SLE4442 と SLE5542 に書き込む操作を有効にするために、メモリカードにシークレットコードを提出する時に、このコマンドを使用します。以下の操作を実行します。：

1. プレゼンテーションエラーカウンタにビット‘1’を検索して、‘0’に変更します。
2. 指定されたシークレットコードをカードに提出します。
3. プレゼンテーションエラーカウンタを消去しようとします。

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**P1**	**P2**	**MEM_L**	**CODE**		
					Byte 1	**Byte 2**	**Byte 3**
FFh	20h	00h	00h	03h			

その中：

CODE 3 バイトのシークレットコード（PIN）

応答データフォーマット

SW1	SW2 ErrorCnt
90h	

その中：

SW1 = 90h
SW2 (ErrorCnt) = エラー　カウンター。07h は検証が正しいことを示している。00H はパスワードがロックされていることを示している（最大再試行回数を超過した）。他の値は現在の認証が失敗したことを示している。

8.6.8. CHANGE_CODE_MEMORY_CARD (SLE4442 および SLE5542)

指定されたデータを新しいシークレットコードとして、カードに書き入れる時に、このコマンドを使用します。
PRESENT_CODE コマンドでカードに現在のシークレットコードを提出してから、このコマンドを実行します。
コマンドのフォーマット

Pseudo-APDU							
CLA	INS	P1	P2	MEM_L	CODE		
					Byte 1	Byte 2	Byte 3
FFh	D2h	00h	01h	03h			

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.7. メモリカード– SLE4406/SLE4436/SLE5536/SLE6636

8.7.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	07h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.7.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	B0h	00h		

その中:

Byte Address メモリカードのメモリアドレス位置

MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ

SW1 SW2 = 90 00h(エラーなしの場合)

8.7.3. WRITE_ONE_BYTE_MEMORY_CARD

このコマンドは、挿入されたカードの指定されたアドレスに1バイトを書き込むために使用されます。バイトは LSB を初めにカードに書かれている。カードアドレス 0 のビットが 0 バイトの LSB とみなされます。

4 つの異なる書き込みモードは、このカードタイプで使用できます。コマンドデータフィールド内のフラグによって区別され

a) **Write**
マンドで指定されたバイトの値が指定されたアドレスに書き込まれて、カードに個人化情報とカウンタ値を書き入れます。

b) **Write with carry**
コマンドで指定されたバイトの値が指定されたアドレスに書き込まれて、コマンドは次の下位カウンタステージを消去するためにカードに送信されます。この書き込みモードはカードにカウンタ値を更新するためにのみ使用することができます。

c) **Write with backup enabled** (SLE4436, SLE5536 and SLE6636 only)
マンドで指定されたバイトの値が指定されたアドレスに書き込まれて、カードに個人化情報とカウンタ値を書き入れます。バックアップビットを有効にして、カード裂けが発生すると、データの損失を防止することができます。

d) **Write with carry and backup enabled** (SLE4436, SLE5536 and SLE6636 only)
コマンドで指定されたバイトの値が指定されたアドレスに書き込まれて、コマンドは次の下位カウンタステージを消去するためにカードに送信されます。この書き込みモードはカードにカウンタ値を更新するためにのみ使用することができます。バックアップビットを有効にして、カード裂けが発生すると、データの損失を防止することができます。

以下のモードで、指定されたアドレスのバイトは書き込みの操作を実行する前に消去されないので、メモリビットが'1'を'0'にプログラムしかできません。
SLE4436 カードと SLE5536 カード中で利用可能なのバックアップモードは、書き込み動作中に有効または無効にすることができます。

コマンドのフォーマット

Pseudo-APDU						
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**MODE**	**BYTE**
FFh	D0h	00h		02h		

その中:

Byte Address メモリカードのメモリアドレス位置
MODE ライトモードとバックアップオプションを指定します
00h:Write
01h:Write with carry
02h:Write with backup enabled (SLE4436, SLE5536 and SLE6636 only)
03h:Write with carry and with backup enabled (SLE4436, SLE5536 and SLE6636 only)
BYTE カードに書き入れていないバイトの値

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.7.4. PRESENT_CODE_MEMORY_CARD

メモリカードにシークレットコードを提出して、カードの個人化モードを有効する時に、コマンドが以下の操作を実行します:

1. プレゼンテーションエラーカウンタにビット'1'を検索して、'0'に変更します。
2. 指定されたシークレットコードをカードに提出します。

シークレットコードを提出すると、ACR3901x はプレゼンテーションカウンタを消去しようとしません。アプリケーションソフトウェアによって、独立な'*Write with carry*'コマンドを介して行われなければなりません。

コマンドのフォーマット

Pseudo-APDU								
CLA	**INS**	**P1**	**P2**	**MEM_L**	**CODE**			
					ADDR	**Byte 1**	**Byte 2**	**Byte 3**
FFh	20h	00h	00h	04h	09h			

その中:

ADDR カードプレゼンテーションカウンタのビットアドレス
CODE 3 バイトのシークレットコード(PIN)

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.7.5. AUTHENTICATE_MEMORY_CARD (SLE4436、SLE5536 および SLE6636)

このコマンドは SLE5536 カードまたは SLE6636 カードからカードの認証証明書を読み取ることに使われます。
ACR3901x は以下の操作を実行します:

1. コマンドで指定されたカード中の Key 1 または Key 2 を選択します。
2. コマンドで指定された乱数をカードに送信します。
3. カードによって計算した認証データの各ビットに対して指定された CLK パルスの数を生成します。
4. カードから 16 ビットの認証データを読み出す。
5. カードを通常動作モードにリセットします

認証プロセスが二段階で実行されます:ステップ1はカードに認証証明書を送信します。ステップ2は、カードによって計算した 2 バイトの認証データを取り戻す。

ステップ1:カードに認証証明書を送信します
コマンドのフォーマット

Pseudo-APDU												
CLA	INS	P1	P2	MEM_L	CODE							
					KEY	CLK_CNT	Byte 1	Byte 2	……	Byte 5	Byte 6	
FFh	84h	00h	00h	08h								

その中:

KEY 認証証明書を計算するためのキー:
00h:Key 1、暗号ブロック連鎖付いていない
01h:Key 2、暗号ブロック連鎖付いていない
80h:key 1, 暗号ブロック連鎖付き (SLE5536 と SLE6636 のみ)
81h:key 2, 暗号ブロック連鎖付き (SLE5536 と SLE6636 のみ)

CLK_CNT CLK のパルス数は、認証証明書の各ビットの計算のためにカードに供給されます。標準値は 160(A0)です

BYTE 1...6 カードの乱数データ

応答データフォーマット

SW1	SW2
61h	02h

その中:

SW1 SW2 = 61 02h (エラーなしの場合), 2 バイトの認証データが準備ができていることを表す。*GET_RESPONSE* コマンドによって、認証データを入手します。

ステップ 2:認証データを取り戻す(Get_Response)
コマンドのフォーマット

Pseudo-APDU				
CLA	INS	P1	P2	MEM_L
FFh	C0h	00h	00h	02h

応答データフォーマット

CERT		SW1	SW2

その中:

CERT カードによって計算された認証データの 16 ビット。BYTE 1 の LSB はカードから読み出された始めの認証ビット。

SW1 SW2 = 90 00h(エラーなしの場合)

8.8. メモリカード – SLE 4404

8.8.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	08h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.8.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	B0h	00h		

その中:

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	…	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h(エラーなしの場合)

8.8.3. WRITE_MEMORY_CARD

このコマンドは、挿入されたカードの指定されたアドレスにデータを書き込むために使用されます。バイトは LSB を初めにカードに書かれている。カードアドレス 0 のビットが 0 バイトの LSB とみなされます。
指定されたアドレスのバイトは書き込みの操作を実行する前に消去されないので、メモリビットが'1'を'0'にプログラムしかできません。

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**Byte 1**	**…**	**Byte N**
FFh	D0h	00h					

その中:

Byte Address メモリカードのメモリアドレス位置

MEM_L メモリに書き入れていないデータの長さ
BYTE カードに書き入れていないバイトの値

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.8.4. ERASE_SCRATCH_PAD_MEMORY_CARD

挿入されたカードのスクラッチパッド中のデータを消去する時に、このコマンドを使う。スクラッチパッドメモリ内のすべてのメモリビットが“1”にプログラムされます。
エラーカウンターまたはユーザーゾーンを消去する時、パート **8.8.5** の説明のように、*VERIFY_USER_CODE* コマンドを使ってください。

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	D2h	00h		00h

その中：
Byte Address スクラッチパッドのメモリバイトアドレス位置
標準値は 02h です

応答データフォーマット

SW1	SW2

その中：
SW1 SW2 = 90 00h（エラーなしの場合）

8.8.5. VERIFY_USER_CODE

挿入されているカードにユーザーシークレットコードを提出する時、このコマンドが使用されます。｜（2 バイト）カードのメモリがアクセスできるように、ユーザーシークレットコードはそれが意図されています。
以下のアクションが実行されます：

1. 指定されたシークレットコードをカードに提出します。
2. プレゼンテーションエラーカウンタにビット‘1’を検索して、‘0’に変更します。
3. プレゼンテーションエラーカウンタを消去します。提出したシークレットコードが成功に認証されて、ユーザーエラーカウンターを消去ことができます。

コマンドのフォーマット

Pseudo-APDU						
CLA	**INS**	**Error Counter LEN**	**Byte Address**	**MEM_L**	**CODE**	
					Byte 1	**Byte 2**
FFh	20h	04h	08h	02h		

その中：
Error Counter LEN プレゼンテーションエラーカウンタの長さ、ビットの単位です。
Byte Address カード中のキーのバイトアドレス
CODE 2 バイトのユーザーシークレットコード

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）
= 63 00h（これ以上の再試行がない場合）

注釈：SW1 SW2 = 90 00h を受信してから、VERIFY_USER_CODE が正しいかどうかをチェックするために、ユーザーのエラーカウンターを再度読み取るはずです。ユーザーのエラーカウンターが全部消去されて、'FFh'に等しいされている場合、前回の検証が成功した。

8.8.6. VERIFY_MEMORY_CODE

挿入されているカードにメモリコードを提出する時、このコマンドが使用されます。｜（4 バイト）メモリコードはユーザコードとユーザメモリの再ロードを許可するために使用されます。

以下のアクションが実行されます：

1. 指定されたシークレットコードをカードに提出します。
2. プレゼンテーションエラーカウンタにビット'1'を検索して、'0'に変更します。
3. プレゼンテーションエラーカウンタを消去します。メモリ エラー カウンター中のデータが消去されないことを注意してください。

コマンドのフォーマット

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	40h	28h	04h				

その中：

Error Counter LEN プレゼンテーションエラーカウンタの長さ、ビットの単位です。
Byte Address カード中のキーのバイトアドレス
CODE 4 バイトのメモリコード

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）
= 63 00h（これ以上の再試行がない場合）

注釈：SW1 SW2 = 90 00h を受信してから、VERIFY_MEMORY_CODE が正しいかどうかをチェックするために、アプリケーションゾーンを再度読み取るはずです。アプリケーションゾーンのデータが全部消去されて、'FFh'に等しいされている場合、前回の検証が成功した。

8.9. メモリカード – AT88SC101/AT88SC102/AT88SC1003

8.9.1. SELECT_CARD_TYPE

このコマンドはカードリーダーに挿入されて、選択したカードにパワーダウン/アップを実行します。同時にリセットを実行する時に使われます。

***注釈**:SCardConnect( ) API によって確立されたロジックなスマートカードリーダー通信後に使用しかできません。SCardConnect( ) API についての詳しい説明は PC/SC 基準を参照してください。*

コマンドのフォーマット

Pseudo-APDU					
CLA	**INS**	**P1**	**P2**	**Lc**	**Card Type**
FFh	A4h	00h	00h	01h	09h

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)

8.9.2. READ_MEMORY_CARD

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	B0h	00h		

その中:

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリカードから読み出されていないデータの長さ

応答データフォーマット

BYTE 1	...	BYTE N	SW1	SW2

その中:

BYTE x メモリカードから読み出されたデータ
SW1 SW2 = 90 00h(エラーなしの場合)

8.9.3. WRITE_MEMORY_CARD

このコマンドは、挿入されたカードの指定されたアドレスにデータを書き込むために使用されます。バイトは LSB を初めにカードに書かれている。カードアドレス 0 のビットが 0 バイトの LSB とみなされます。
指定されたアドレスのバイトは書き込みの操作を実行する前に消去されないので、メモリビットが'1'を'0'にプログラムしかできません。

コマンドのフォーマット

Pseudo-APDU							
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**	**Byte 1**	**....**	**Byte N**
FFh	D0h	00h					

その中:

Byte Address メモリカードのメモリアドレス位置
MEM_L メモリに書き入れていないデータの長さ
BYTE カードに書き入れていないバイトの値

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.9.4. ERASE_NON_APPLICATION_ZONE

このコマンドがアプリケーションゾーンにストアーされていないデータを消去する時に使われます。EEPROM メモリが 16 ビットのワードで構造されます。独立な 1 ビットのワードを消去しても、ERASE 操作が全てのワードを消去できます。したがって、メモリ中の任意のビットに消去を実行すると、そのメモリのすべての 16 ビットをクリアして、"1"の状態になる。エラーカウンタまたはアプリケーションゾーンのデータを消去する時、以下のコマンドを参照してください：

1. パート **8.9.5** は *ERASE_APPLICATION_ZONE_WITH_ERASE* コマンドを定義している。
2. パート **8.9.6** は *ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE* コマンドを定義している。
3. パート **8.9.7** は *VERIFY_SECURITY_CODE* コマンドを定義している。

コマンドのフォーマット

Pseudo-APDU				
CLA	**INS**	**P1**	**Byte Address**	**MEM_L**
FFh	D2h	00h		00h

その中：

Byte Address 消去していないワードのメモリバイトアドレスの場所

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

8.9.5. ERASE_APPLICATION_ZONE_WITH_ERASE

このコマンドは以下の状況に適用されます：

1. AT88SC101：擦アプリケーションゾーンのデータを消去して、EC 機能が無効になる。
2. AT88SC102：アプリケーションゾーン 1 のデータを消去します。
3. AT88SC102：アプリケーションゾーン 2 のデータを消去して、EC2 機能が無効になる。
4. AT88SC1003：アプリケーションゾーン 1 のデータを消去します。
5. AT88SC1003：アプリケーションゾーン 2 のデータを消去して、EC2 機能が無効になる。
6. AT88SC1003：アプリケーションゾーン 3 のデータを消去します。

このコマンドで以下の操作を実行します：

1. 指定されたコードをカードに提出します。
   a. プレゼンテーションエラーカウンタを消去します。提出したコードが成功に認証されて、アプリケーションゾーンのデータが消去することができます。

コマンドのフォーマット

Pseudo-APDU								
CLA	**INS**	**Error Counter LEN**	**Byte Address**	**MEM_L**	**CODE**			
					Byte 1	**Byte 2**	**…**	**Byte N**
FFh	20h	00h						

その中：

Error Counter LEN プレゼンテーションエラーカウンタの長さ、ビットの単位です。ずっと 00h であるべき。

Byte Address カード中のアプリケーションゾーンのアドレス。精確な数値が下のチャートを参照してください。

	Byte Address	LEN
AT88SC101：擦アプリケーションゾーンを消去して、EC 機能が無効になる。	96h	04h
AT88SC102：アプリケーションゾーン 1 を消去します	56h	06h
AT88SC102：アプリケーションゾーン 2 を消去して、EC2 機能が無効になる。	9Ch	04h
AT88SC1003：アプリケーションゾーン 1 を消去します	36h	06h
AT88SC1003：擦アプリケーションゾーン 2 を消去して、EC2 機能が無効になる。	5Ch	04h
AT88SC1003：アプリケーションゾーン 3 を消去します	C0h	06h

MEM_L 消去キーの長さ。精確な数値が上のチャートを参照してください。
CODE N バイトの消去キー

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

注釈:SW1 SW2 = 90 00h を受信してから，ERASE_APPLICATION_ZONE_WITH_ERASE が正しいかどうかをチェックするために、アプリケーションゾーンのデータを再度読み取るはずです。*アプリケーションゾーンのデータgが全部消去されて、'FFh'に等しいされている場合、前回の検証が成功した。*

8.9.6. ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE

このコマンドは以下の状況に適用されます：

1. AT88SC101：アプリケーションゾーンのデータを消去して、EC 機能が有効になる。
2. AT88SC102：アプリケーションゾーン 2 のデータを消去して、EC2 機能が有効になる。
3. AT88SC1003：アプリケーションゾーン 2 のデータを消去して、EC2 機能が有効になる。

EC または EC2 機能が有効になってから（すなわち：ECEN または EC2EN ヒューズが損害されていなくて、“1”の状態）、以下の操作を実行します：

1. 指定されたシークレットコードをカードに提出します。
2. プレゼンテーションエラーカウンタにビット‘1’を検索して、‘0’に変更します。
3. プレゼンテーションエラーカウンタを消去します。提出したコードが成功に認証されて、アプリケーションゾーンのデータが消去することができます。

コマンドのフォーマット

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Byte 1	Byte 2	Byte 3	Byte 4
FFh	20h	80h		04h				

その中：

Error Counter LEN プレゼンテーションエラーカウンタの長さ、ビットで数値は 80hです。
Byte Address カード中のアプリケーションゾーンのアドレス

	Byte Address
AT88SC101	96h
AT88SC102	9Ch
AT88SC1003	5Ch

CODE 4 バイトの消去キー

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)
= 63 00h(これ以上の再試行がない場合)

注釈:SW1 SW2 = 90 00h を受信してから,
ERASE_APPLICATION_ZONE_WITH_WRITE_AND_ERASE が正しいかどうかをチェックするために、アプリケーションゾーンのデータを再度読み取るはずです。*アプリケーションゾーンのデータgが全部消去されて、'FFh'に等しいされている場合、前回の検証が成功した。*

8.9.7. VERIFY_SECURITY_CODE

挿入されているカードにセキュリティコードを提出する時、このコマンドが使用されます。(2 バイト)カードのメモリがアクセスできるように、セキュリティコードはそれが意図されています。
以下のアクションが実行されます:

1. 指定されたシークレットコードをカードに提出します。
2. プレゼンテーションエラーカウンタにビット'1'を検索して、'0'に変更します。
3. プレゼンテーションエラーカウンタを消去します。提出したコードが成功に認証されて、セキュリティコードの試みるカウンターが消去ことができます。

コマンドのフォーマット

Pseudo-APDU						
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE	
					Byte 1	Byte 2
FFh	20h	08h	0Ah	02h		

その中:

Error Counter LEN プレゼンテーションエラーカウンタの長さ、ビットの単位です。
Byte Address カード中のキーのバイトアドレス
CODE 2 バイトのセキュリティコード

応答データフォーマット

SW1	SW2

その中:

SW1 SW2 = 90 00h(エラーなしの場合)
= 63 00h(これ以上の再試行がない場合)

注釈:SW1 SW2 = 90 00h *を受信した後で、VERIFY_USER_CODE が正しいかどうかを確認するために、セキュリティコードの試みるカウンター(SCAC)を再度読み取ることができます。SCAC が消去されて、'FFh'に等しいされている場合、前回の検証が成功した。*

8.9.8. BLOWN_FUSE

このコマンドは挿入されているカードのヒューズを変更する時に使われます。ヒューズは EC_EN のヒューズ、EC2EN のヒューズ、メーカーのヒューズまたは発行者のヒューズ可能です。
注釈:ヒューズを変更することは不可逆過程です。
コマンドのフォーマット

Pseudo-APDU								
CLA	INS	Error Counter LEN	Byte Address	MEM_L	CODE			
					Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of FUS Pin	State of RST Pin
FFh	05h	00h	00h	04h			01h	00h または 01h

その中：

Fuse Bit Addr (2 バイト) ヒューズのビットアドレス。精確な数値が下のチャートを参照してください。
State of FUS Pin FUS pin の状態。ずっと 01h であるべき。
State of RST Pin RST pin の状態。精確な数値が下のチャートを参照してください。

		Fuse Bit Addr (High)	Fuse Bit Addr (Low)	State of RST Pin
AT88SC101	メーカーのヒューズ	05h	80h	01h
	EC2EN のヒューズ	05h	C9h	01h
	発行者のヒューズ	05h	E0h	01h
AT88SC102	メーカーのヒューズ	05h	B0h	01h
	EC2EN のヒューズ	05h	F9h	01h
	発行者のヒューズ	06h	10h	01h
AT88SC1003	メーカーのヒューズ	03h	F8h	00h
	EC2EN のヒューズ	03h	FCh	00h
	発行者のヒューズ	03h	E0h	00h

応答データフォーマット

SW1	SW2

その中：

SW1 SW2 = 90 00h（エラーなしの場合）

9.0. 非接触カードコマンド

本項は ACR35 の非接触カードのコマンドを紹介します。

9.1. 非接触インターフェースの PseudoAPDU

9.1.1. Get Data

このコマンドが PICC カードのシリアルナンバーもしくは ATS を入手する時に使われます。

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）

例え P1=00h、UID 応答フォーマットを入手します（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 を入手します（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	90 00h	操作が成功に完了しました。
警告	62 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	6C XX	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	63 00h	操作が失敗しました。
エラー	6A 81h	この機能をサポートできません。

例：

```
//PICC カードのシリアルナンバーを入手します。
UINT8 GET_UID[5]={FF, CA, 00, 00, 00h};
//ISO14443 A タイプのカードの ATS を入手します。
UINT8 GET_ATS[5]={FF, CA, 01, 00, 00h};
```

9.2. 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）

9.2.1. 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ます。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ます。リーダーは二種の認証キーのアドレスが提供られている：失いやすいキーのアドレスと失いやすくないキーのアドレス。

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ます。
その他＝予約済み

キーのナンバー 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。キーは永遠にリーダーに保存されて、リーダが 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 2 個キーをストアーできます。
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ときーの値。
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	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:
// 失いやすいキーのメモリに 00h キーをロードします {FF FF FF FF FF FFh}。
APDU = {FF 82 00 00 06 FF FF FF FF FF FFh}

9.2.2. Authentication for MIFARE Classic 1K/4K

このコマンドは ACR3x 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ます:TYPE_A と TYPE_B。

Load Authentication KeysAPDU 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 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 1K カードが 16 個と分けている。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。
例:セクター00h が含まれているブロック{00h、01h、02h および 03h};セクター01h が含まれているブロック{04h、05h、06h および 07h};ラストセクター0Fh が含むいるブロック{3Ch、3Dh、3Eh および 3Fh}。
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ます。
詳しい情報は MIFARE 1K/4K 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、認証用に使われます。
61h = TYPE B キーとして、認証用に使われます。

キーのナンバー　　1 バイト。
00h ~ 01h = キーをストアーするための失いやすいメモリ。リーダーがコンピュータから切断されたとき、それらのキーが削除されます。失いやすいキーを二つ提供されています。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むいる)	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	00 ~ 02h	03h
セクター1	04 ~ 06h	07h
…		
…		
セクター14	38 ~ 0Ah	3Bh
セクター15	3C ~ 3E	3Fh

1 KB

表3 ： MIFARE Classic 1K カードのメモリマップ

セクター (32 個のセクター，各セクターには 4 個の連続的なブロックが含むいる)	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	00 ~ 02h	03h
セクター1	04 ~ 06h	07h
…		
…		
セクター30	78 ~ 7Ah	7Bh
セクター31	7C ~ 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むいる)	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	80 ~ 8Eh	8Fh
セクター33	90 ~ 9Eh	9Fh
…		
…		
セクター38	E0 ~ EEh	EFh
セクター39	F0 ~ FEh	FFh

2 KB

表4 ： 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, 廃止されます
```

APDU = {FF 88 00 04 60 00h};
同様に、
// {TYPE A, キーナンバーの 00h}によって、ブロック 04h を認証します。
// PC/SC V2.07
APDU = {FF 86 00 00 05 01 00 04 60 00h}
注釈:MIFARE Ultralight のメモリは自由にアクセスできます。認証はいりません。

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

512 ビット
または
64 バイト

表5 :MIFARE Ultralight メモリマップ

9.2.3. Read Binary Blocks

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

Read Binary の APDU フォーマット(5 バイト)

コマンド	CLA	INS	P1	P2	Le
Read Binary Blocks	FFh	B0h	00h	ブロック番号	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

更新していない 1 バイト。

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

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

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

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	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:

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

APDU = {FF B0 00 04 10}

从バイナリブロック 80h に 240 バイトを読み取る(MIFARE 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 F0}

9.2.4. 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ます。Update Binary Blocks コマンドを実行する前に、データブロック/トレーラーブロックを認証しなければなりません。

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	Le	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**	ブロックデータ(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 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 1K のバイトは最大に 48 です（複数のブロックモード；3 つの連続ブロック）。
更新していない MIFARE 4K のバイトは最大に 240 です（複数のブロックモード；15 つの連続ブロック）。
更新していない MIFARE Ultralight のバイトは最大に 16 です。

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。

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。

Update Binary Block 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：
// MIFARE 1K または 4K カード中のバイナリブロック 04h を {00 01 … 0F}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 0F}

MIFARE Ultralight 中のバイナリブロック 04hを{00 01 02 03}に更新します

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

9.2.5. Value Block Operation (INC, DEC, STORE)

Value Block Operation コマンドは数値ブロック対しての操作に使われます（例：数値ブロックの数値をインクリメントします）。

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

コマンド	CLA	INS	P1	P2	Le	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数値ブロック対しての操作のみに適用します。
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。数値ブロック対しての操作のみに適用します。
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。

例 1：Decimal –4 = {FFh, FFh, FFh, FCh}

VB_Value			
MSB			LSB

VB_Value			
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。

9.2.6. Read Value Block

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

Read Value Block の APDU フォーマット（5 バイト）

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

その中:

ブロック番号 1 バイト。アクセスされていない数値のブロック

Read Value Block の応答フォーマット（4 + 2 バイト）

応答	データ出力		
結果	数値 {MSB … LSB}	SW1	SW2

その中:

数値 4 バイト。カードから返された数値。
この値は符号付き長い整数です。

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

数値			
MSB			LSB
FFh	FFh	FFh	FCh

例 2: Decimal 1 = {00h, 00h, 00h, 01h}

数値			
MSB			LSB
00h	00h	00h	01h

Read Value Block コマンドの応答コード

結果	SW1 SW2	意味
成功	90 00h	操作が成功に完了しました。

結果	SW1 SW2	意味
エラー	63 00h	操作が失敗しました。

9.2.7. Copy Value Block

Copy Value Block コマンドは、数値を一つの値ブロックから別の値ブロックにコピーすることに使用されています。

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	
Value Block Operation	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：

ソースブロック番号 1 个字节。
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ます。

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 の応答メッセージ（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 をブロック 05h にストアーします。
APDU = {FF D7 00 05 05 00 00 00 00 01}

//ブロック 05h から一つの数値を読み取る。
APDU = {FF B1 00 05 04}

//数値をブロック 05h からブロック 06h にコピーします。
APDU = {FF D7 00 05 04 02 03 06}

//ブロック 05h の値を 5 にインクリメントします。
APDU = {FF D7 00 05 05 01 00 00 00 05}

9.2.8. PC / SC 準拠のタグをアクセスする（ISO14443-4)

基本的に、すべての ISO14443-4 に準拠したカードは、ISO7816-4 の APDU を理解できます。ACR35 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。ACR35 は内部で ISO14443 の1－4パートのプロトコルを処理します。
MIFARE 1K、4K、Mini および 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。

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
ISO78164 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 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

ISO 7816-4 応答コード

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

典型的なシーケンスは：

1. タグを提示して、PICC 画面と接続します。
2. タグ中の情報を読み取り/更新します。

以下のように操作してください：

1. タグと接続します。
   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、乱数を入手します。
   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 を入手します。*

例：
//ISO14443-4 B タイプの PICC から 8 バイトを読み取る
APDU = {80 B2 80 00 08}

CLA	INS	P1	P2	Lc	データイン	Le
80h	B2h	80h	00h	なし	なし	08h

応答：00 01 02 03 04 05 06 07 [$9000]

9.2.9. 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。これらのコマンドは FeliCa 基準に準拠して、ヘッダが追加されています。

FeliCa のコマンドデータフォーマット

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

FeliCa の応答データフォーマット（データ+2 バイト）

応答	データ出力
結果	応答データ

例のメモリブロックデータの読み取り

1. FeliCa を接続します。
   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

10.0.機密データの導入方法

本文は機密データの導入方法を紹介します。例えば：よりセキュアな環境で顧客マスターキー、DUKPT 初期 PIN 暗号化キー、AES 暗号化キーおよび顧客 ID を ACR3x に導入する方法。

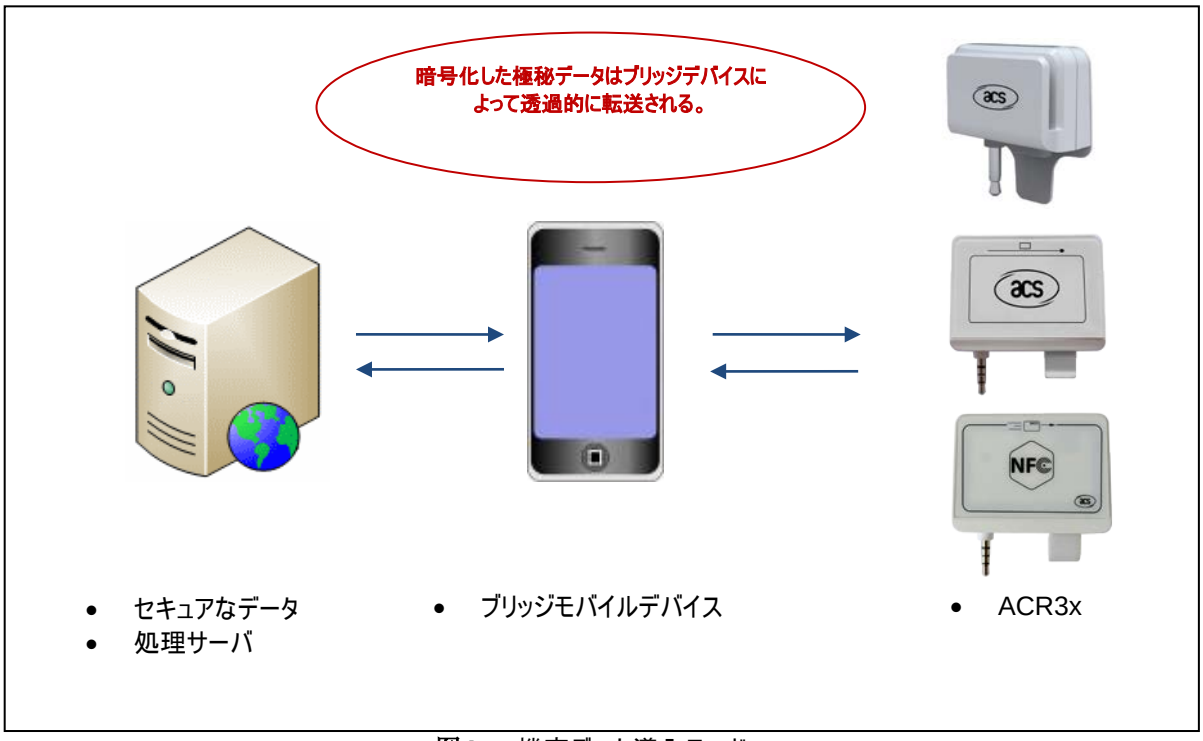

図4 ：機密データ導入モード

上の図示が 3 つのエンティティと関わっている、これらは：セキュアなデータ処理サーバ、ブリッジモバイルデバイスと ACR3x。セキュアなデータ処理サーバは ACR3x を対象としての暗号化した機密データを受信し、生成することを担当します。ブリッジモバイルデバイスはデータ処理サーバと ACR3x の間のメッセージブリッジングチャネルだけとして動作します。そのメッセージはブリッジモバイルデバイスで処理する必要がありません（適切なフレームにデータを再梱包してオーディオチャネルを介して ACR3x に送信される必要がない限り）。

10.1. 認証

機密データが ACR3x に導入される前に、データ処理サーバ（その時、モバイルデバイスは必ずサーバと接続している）が ACR3x に認証されて、ACR3x からその中の機密情報を変更する許可を取らなければなりません。相互認証が ACR3x 中で使われている。
認証リクエストはいつもデータ処理サーバまたはブリッジデバイスから発起されます。そうすると、16 バイトの乱数（RND_A[0:15]）のシーケンスを返すように ACR3x をトリガします。この 16 バイトの乱数が ACR3x から送信される前に、AES-128 CBC 暗号化モードを採用して、ACR3x 中にストレージしている顧客マスターキーで暗号化されます。ブリッジデバイスが必ず暗号化した乱数をデータ処理サーバ送信します。それで、データ処理サーバがその中で使用している顧客マスターキーを通じて、AES-128 CBC 暗号化モードで受信したデータを複合化します（データ処理サーバ中の顧客マスターキーと ACR3x 中の顧客マスターキーが同じで、顧客が保存されます）。そのあとで、複合化した 16 バイトの乱数はデータ処理サーバが生成した別の 16 バイトの乱数（RND_B[0:15]）の後に記入されます。最後に 32 バイトの乱数（RND_C[0:31]）が生成されました、すなわち：

RND_C[0:31] = RND_B[0:15] + RND_A[0:15],

データ処理サーバ中に使用している顧客マスターキーで、この 32 バイトの乱数を複合化します。複合化プロセスからの最終出力データが認証の応答メッセージ送信を使用して、ブリッジデバイスを通じて、ACR3x に送信されます。
ACR3X が応答メッセージを受信すると、メッセージデータは自身持っている顧客マスターキーで復号化して、普通の 32 バイトの乱数に変換します。理論的には、前の 16 バイトの乱数は RND_B[0:15]に相当して、データ処理サーバが生成されます。別の 16 バイトの乱数は RND_A[0:15]に相当して、ACR3x が生成されます。

ACR3x はまず RND_A[0:15]と元のデータが同じかどうか比較します。同じの場合、データ処理サーバが ACR3x の認証に合格しました。それで、ACE3x が顧客マスターキーで RND_B[0:15]を暗号化して、ブリッジデバイスによって、認証の応答メッセージ中にデータ処理サーバに返します。

応答メッセージの答えを受信した後で、データ処理サーバがメッセージ中のデータを復号して、16 バイトの乱数と最初生成した RND_B[0:15]が同じかどうか比較します。同じの場合、ACR3x がデータ処理サーバの認証に合格しました。いま、認証プロセクスが完了しました。機密データが ACR3xに導入できます。
認証が成功すると、ACR3x とデータ処理サーバが 16 バイトのセッション鍵を生成します。RND_A の始めの 8 バイトを RND_B の始めの 8 バイトの後に記入されて、そのセッション鍵（SK[0:15]）になる。すなわち：

SK[0:15] = RND_B[0:7] + RND_A[0:7]

全てのセキュアなデータ処理サーバから送信した機密データがセッション鍵を使用して、AES-128 CBC 暗号化モードで暗号化されます。そうすると、ブリッジデバイスが暗号化したデータを捕獲しても、顧客マスターキーがよく分からない状況で元の機密データを捕獲することが難しいです。

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）

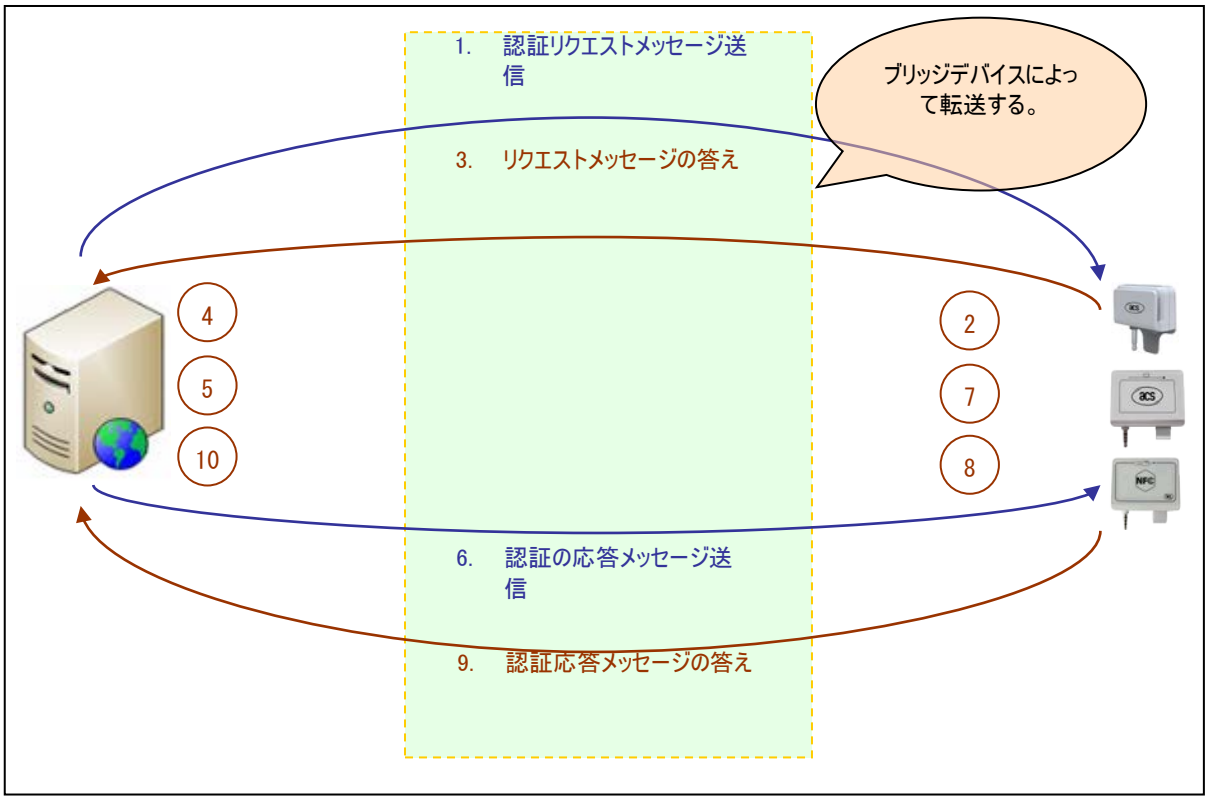

図5 ：認証手順

以下は上記の手順の概要です：

1. データ処理サーバ/ブリッジデバイスは、認証リクエストメッセージを送信することによって、ACR3x からの認証リクエストを発起します。
2. 認証リクエストメッセージを受信すると、ACR3x が 16 バイトの乱数を生成します（RND_A[0:15]）。この 16 バイトの乱数は ACR3x が使用している顧客マスターキーで暗号化されます。
3. 暗号化した RND_A[0:15]が認証応答メッセージに対する答えを介してデータ処理サーバに転送されます。
4. データ処理サーバが受信したダーたを復号化して、RND_A[0:15]を回復します。
5. データ処理サーバが別の 16 バイトの乱数を生成します（RND_B[0:15]）。RND_A[0:15]が RND_B[0:15]の後に記入されて、32 バイトの乱数になる（RND_C[0:31] = RND_B[0:15] + RND_A[0:15]）。全ての 32 バイトの乱数をサーバが使用している顧客マスターキーで暗号化します。
6. 暗号化プロセスからの最終出力データが認証の応答メッセージ送信を通じて、ACR3x に送信されます。
7. ACR3x 中で、受信したデータを復号化して、32 バイトの乱数を回復します。回復した RND_A[0:15]は最初の乱数と同じかどうか、ACR3x がチェックします。同じじゃない場合、認証プロセスが終止されます。
8. ACR3x が RND_B[0:15]を顧客マスターキーで暗号化します。同時に、RND_A の最初の 8 バイトを RND_B の最初の 8 バイトの後に記入して、16 バイトのセッション鍵を生成します。
9. 暗号化した RND_B[0:15]が認証の応答メッセージを通じて、データ処理サーバに送信されます。
10. データ処理サーバが受信したデータを複合化して、元の RND_B[0:15]と同じかどうか比較します。同じじゃない場合、認証プロセスが終止されます。それに対して、認証が終わる。RND_A の最初の 8 バイトを RND_B の最初の 8 バイトの後に記入して、1 セッション鍵が生成されます。

10.2. 顧客マスターキー導入

工場で製造された時、ACR3x のフラッシュメモリは、いくつかのデフォルトの値にリセットされている必要がある。顧客マスターキーは全部ゼロにリセットされます。
顧客マスターキーを変更する場合、データ処理サーバと ACE3x が古いキーで認証しなければなりません。成功に認証した後で、データ処理サーバは ACR3x にマスターキーを設定するコマンドメッセージを送信します。そのコマンドメッセージが顧客マスターキーを含むいる。新しい顧客マスターキーは現在のセッション鍵で暗号化されなければなりません。新しい顧客マスターキーが成功に ACR3x にロードされる後で、ACR3x によって確立された認証が削除されます。新しい機密データを導入する前に、サーバーは新しい顧客マスターキーで新しい認証リクエストを実行する必要がある。

10.3. AES キー導入

ACR3x が工場で製造された時、デフォルト ACR AES キーはフラッシュメモリにリロードされています。成功に認証した後で、ユーザーがこのキーを任意の値に変更できます。
DUKPT が無効にした場合には、AES キーはトラックデータの暗号化に使用されます。新しい AES キーはすぐに有効であり、それは、現在の認証されているセッションには影響しません。

10.4. DUKPT の初期化

DUKPT 鍵管理アルゴリズムが正常に動作する前に、いくつかの初期化処理を実行されなければなりません。まず、データ処理サーバは ACR31 に 10 バイトの最初の鍵のシリアル番号（IKSN）及び ACR31 に 16 バイトの初期 PIN 暗号化キー（IPEK）を提供しなければなりません。数字のこれらの 2 つの配列は、その将来の主要なテーブルやその他の設定を初期化します DUKPT キー管理エンジンによって使用されます。DUKPT エンジンの暗号化カウンタは自動的にリセットされます。

DUKPT が初期化された後、磁気ストライプトラックデータを暗号化するために使用されるキーは、DUKPT アルゴリズムによって生成されるように、DUKPT オプションを有効にする必要がある。カードが成功にスワイプした後に、すべてのトランザクションに固定な AES キーを使用するのではなく、ユニークな暗号化キーが DUKPT にリクエストされます。

スワイプカードデータにエラーがある場合、どのキーも DUKPT エンジンからリクエストされないことに留意すべきです。代わりに、エラーメッセージにトラックデータィールドがゼロで記入されて、エラーコードだけで検出されたすべてのカードデータのエラーのタイプを表す。ユーザーが再びカードをスワイプさせるために、失敗したカードスワイプが DUKPT からキーをリクエストしていません。モバイルデバイスアプリケーションがそのように提示します。無駄なデータをバックエンドサーバへプッシュすることなくて、その同時にサーバとの同期的な暗号カウンタを維持します。

11.0. カードデータの暗号化

カードをスワイプされるたびに、応答メッセージは自動的にモバイルデバイスに送信されます。(初期ベクトルは 16bの0です)メッセージ中にカプセル化されたトラックデータが AES-128 CBC 暗号化モードを使用して暗号化されます。
DUKPT が有効になっている場合、カードを成功にスワイプされると、ラックデータの暗号化に用いる鍵は DUKPT 鍵管理アルゴリズムによって生成されます。結果として、成功したすべてのトランザクションに対して、異なるキーがトラックデータの暗号化に使用されます。
DUKPT が無効にした場合には、AES キーはトラックデータの暗号化に使用されます。**10.010.0 セクション**で紹介した機密データを導入する方法を用いて、AES キーを変更することができます。ACR3x は工場から出荷されたときに、デフォルトの AES キーが ACR3x 内部にプレロードされます。デフォルト AES キーは:

4E 61 74 68 61 6E 2E 4C 69 20 54 65 64 79 20h

カードスワイプ中、データエラーが存在する場合には、トラックデータィールドがゼロで記入されます。エラーメッセージのみに報告されることを留意すべきです。

12.0. AES-128 CBC 暗号化のテストベクトル

以下の表は、ACR3x で使用されているトリプル AES-128 CBC 暗号モードのいくつかのテストベクトルを示している

元データ:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
初期ベクトル	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
キー:	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
暗号化の出力:	6B 1E 2F FF E8 A1 14 00 9D 8F E2 2F 6D B5 F8 76h
元データ:	69 88 44 21 13 84 0A 10 00 0C 02 22 11 88 00 0E 12 84 00 B1 40 80 80 11 31 02 45 20 20 28 E4 00h
初期ベクトル	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
キー:	10 29 02 14 03 90 53 09 12 08 20 20 02 C0 9A 80h
暗号化の出力:	EF 14 C9 C9 F3 48 96 5B 18 36 0A 2F 81 1A 93 C7 E2 FF F3 61 04 B8 D4 5E 13 F7 26 FE 2A 94 2B 69h
元データ:	80 83 11 13 09 D1 11 30 0E 00 0A 49 04 00 26 99 C0 58 D1 7A 45 CD 17 10 30 00 22 08 10 4C 41 51h
初期ベクトル	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
キー:	9A 04 2A 10 21 00 06 20 10 84 20 01 00 00 22 1Ch
暗号化の出力:	56 85 B9 6B A1 B2 09 AB 58 71 58 B5 E0 30 42 71 64 62 51 FA 55 94 52 BC 78 33 24 FB 15 F5 33 62h
元データ:	41 01 06 02 21 A8 C4 40 08 00 44 11 11 88 0D 09 10 81 92 10 01 20 20 2E 20 C4 05 81 58 08 18 86h
初期ベクトル	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
キー:	30 13 30 C8 91 53 49 44 0E 29 98 42 84 17 00 D0h
暗号化の出力:	ED 0F 2E BC 7D EA 58 C4 AB E8 72 91 87 74 2F C3 B1 8B 66 4F F5 E5 3F 8B BD A9 63 40 F8 0D 11 97h

13.0. TDES ECB 暗号化のテストベクトル

以下の表は、ACR3x で使用されているトリプル DES ECB 暗号モードのいくつかのテストベクトルを示している。

元データ:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
キー:	10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00h
暗号化の出力:	49 F9 E7 A6 0C 40 6D BF 49 F9 E7 A6 0C 40 6D BFh
元データ:	02 50 88 82 22 21 13 C4 42 00 08 44 60 24 8A 04h
キー:	00 C0 08 28 8E 28 16 10 01 80 50 4D 72 00 28 88h
暗号化の出力:	8E BF 16 AA B4 59 AA C0 13 DB 32 E5 1D 04 BD 66h
元データ:	61 10 88 19 42 31 01 26 42 02 74 24 00 07 0C 82h
キー:	00 10 80 42 09 20 13 24 82 22 24 89 62 08 09 90h
暗号化の出力:	74 57 DF 51 3B 04 7A F2 2B 26 C4 BF 81 6B 4D 58h

付録A.　トラックデータエラーコード

Bit 7 MSB	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 LSB
MSB	0	0	0	0	LRC エラー	エンドセンチネルエラー	スタートセンチネルエラー

注釈:

1. *b7-からb 1まではエラーコードです。*
2. *エラーなし　＝　0*

付録B.　システムエラーコード

以下のテーブルが全てのシステムエラーコードとそれぞれの説明を一覧表にします。

エラーコード	状態
00h	ERROR_SUCCESS
FFh	ERROR_INVALID_CMD
FEh	ERROR_INVALID_PARAM
FDh	ERROR_INVALID_CHECKSUM
FCh	ERROR_INVALID_STARTBYTE
FBh	ERROR_UNKNOWN
FAh	ECODE_DUKPT_CEASE_OPERATION
F9h	ECODE_DUKPT_DATA_CORRUPTED
F8h	ECODE_FLASH_DATA_CORRTPTED
F7h	ECODE_VERIFICATION_FAILED

表6　: システムエラーコード