Advanced Card Systems Ltd.
Card & Reader Technologies

APG8201
PINhandy 1

リファレンスマニュアル　V1.02

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

目次

図のリスト

表のリスト

1.0. 紹介

APG8201 PINhandy 1 はポータブルで利用できるスマートカードデバイスです。リンクモードとオフラインモード両方をサポートできて、多種の認定機能を持っています。PIN パッドと複数の言語と英数字をサポートするグラフィカルな LCD を内蔵しています。APG82101 は、デバイス内の認証プロセスを使用して、セキュリティ攻撃から PIN コードを保護するキュア PIN エントリ（SPE）をサポートしています高品質で高信頼性のデバイスである APG8201 PINhandy 1 は、付加価値のあるワンタイムパスワード（OTP）と電卓機能を備えています。

2.0. 特性

- コンパクトでポータブルなハンドヘルドデバイス
- 二種の操作モード：
  - PC リンク
  - スタンドアローン
- USB 電源供給（PC リンクモード）：
  - USB 2.0 フルスピードインターフェース
  - CCID 準拠
  - アプリケーション プログラミング インターフェース
    - PC/SC サポート
    - (PC/SC の上のラッパー経由で) 、CT- API をサポート
  - プロトコルとパラメータの選択サポート（PPS ）
  - PC/SC 2.01 パート 10 - 安全な PIN 入力サポート
- スタンドアロン動作：
  - ワンタイムパスワード（OTP ） 、チャレンジレスポンスやトランザクションデータ署名モードをサポート
  - 電源用 CR2032 電池 × 2
  - インテリジェントバッテリ管理または 5 年の平均寿命（使用状況に依る）
- スマートカードリーダー：
  - フルサイズのマイクロプロセッサカード（ T = 0 、T = 1 プロトコル）対応
  - ISO 7816 クラス A カードをサポート
  - カードの半挿入許可
  - 短絡保護機能
- 内蔵されている周辺機器：
  - ロゴおよび複数言語文字用のグラフィカル LCD
  - モノトーンブザー
  - 20 キーの耐久性に優れたシリカゲルのキーパッドメンブレン
  - SPE モードを認識するための LCD のキーシンボル
- 付加価値を付与した電卓や電子財布等の機能
- Android™ 3.1 と以降のバージョンサポートしています [1]
- 以下の基準に一致している：
  - ISO 7816
  - EMV™ Level 1 (接触)
  - MasterCard® チップ認証プログラム（CAP）

[1] *ACS 定義されたアンドリュースライブラリを使用しています*

- MasterCard®高いレベルのチップ認証（AA4C/PLA）
- VISA®ダイナミックパスワード認証（DPA）
- PC/SC
- CCID
- CE
- FCC
- RoHS 2
- FIPS 201 認証（アメリカ）
- Microsoft® WHQL

3.0. サポートしているカードタイプ

APG8201 PINhandy 1 のメインカードスロットは T = 0 および T = 1 プロトコルの MCU カード Class A（5 V）をサポートできます。 また、内部プログラミング電圧（VPP）生成を備えた EEPROM マイクロコントローラベースのカードもサポートしています。ATR で送信されるプログラミングパラメータは：

- PI1 = 0 また 5
- I = 25 また 50

APG8201 はスタンドアロンモードで自動的にプロトコルとパラメータの選択（PPS）を実行したり、USB 接続モードで PPS を手動で実行することができます。

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 APG8201 がこのモードをサポートできない場合、APG8201 はカードをリセットして、交渉モードに設置します。交渉モードを設置できないと、APG8201 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、APG8201 がその通信パラメータを使用してみて、PPS を実行します。ACR32 が PPS を拒否しったら、デフォルトパラメ—タ—を使用する（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。

4.0. スマートカードインターフェ-ス

APG8201 と挿入されたカードの間のインターフェースが ISO 7816-2 仕様プロトコルに準拠して、APG8201 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。

- スマートカード電源 VCC（C1）
  - o 挿入されたカードの消費電流は 50mA よりも高くてはならない。
- プログラミング電圧 VPP（C6）
  - o VPP ピンは未接続のまま
- リセット信号（C2）
  - o EMV 2000 バージョン BOOK 1 を参照してください
- クロック信号（C3）
  - o EMV 2000 バージョン BOOK 1 を参照してください
- 周囲（C5）
  - o EMV 2000 バージョン BOOK 1 を参照してください
- I/ O データ入出力（C7）
  - o EMV 2000 バージョン BOOK 1 を参照してください

5.0. カードタイプのセレクション

制御 PC は、挿入されたカードをアクティする前に、APG8201 に適切なコマンドを送信してカードタイプを選択する必要があります。MCU ベースのカードの場合、カードリーダーは優先的にプロトコル T = 0 または T = 1 を選択することができます。 しかし、この選択は、リーダに挿入されたカードが両方のプロトコルタイプをサポートしている場合にのみ、PPS を介してリーダによって受け入れられ、実行されます。 MCU ベースのカードが 1 つのプロトコルタイプ（T = 0 または T = 1）しかサポートしていない場合は、アプリケーションによって選択されたプロトコルタイプに関係なく、リーダは自動的にそのプロトコルタイプを使用します。

6.0. USB インターフェース

APG8201 は、「USB 規格 2.0」に規定されているように、フルスピードモード、すなわち 12Mbps で動作する USB を介してコンピュータに接続されます。APG8201 が USB インタフェースを介して正しく機能するためには、ACS 独自のデバイスドライバまたは ACS PC/SC デバイスドライバのいずれかをインストールする必要があります。

7.0. USB 通信プロトコル（CCID）

APG8201 は USB を介して、ホストとのインターフェースを確立します。業界の規範 - CCID 標準は、USB チップ - スマートカードインタフェース装置に関わっているプロトコルを定義します。CCID は、スマートカードと PIN を動作させるために必要なすべてのプロトコルをカバーしています。APG8201 の USB エンドポイントの装置と使用は CCID 標準に準拠するはずです。

APG8201 が処理する必要のあるいくつかの必須 CCID コマンドプロトコルがあります。後続のセクションで列挙されます。

7.1. PC からリーダーへ

7.1.1. PC_to_RDR_IccPowerOn

このコマンドはスロットを活性化して、カードから ATR を返すために使われます。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	62h	-
1	*dwLength*	4	00000000h	メッセージ固有のデータ長さ。
2	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
5	*bSeq*	1	00-FFh	コマンドのシーケンス番号
6	*bPowerSelect*	1	01h	ICC に印加される電圧： 01h - 5 V
7	*abRFU*	2	-	保留して将来使います。

このメッセージの応答は *RDR_to_PC_DataBlock* メッセージです。返されたデータはリセット応答（ATR）です。

7.1.2. PC_to_RDR_IccPowerOff

スロットの活性化をキャンセルする時、このコマンドを使います。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	63h	-
1	*dwLength*	4	00000000h	メッセージ固有のデータ長さ。
5	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	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。

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

7.1.3. PC_to_RDR_XfrBlock

ICC にデータブロックを転送する時にこのコマンドを使用します。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	6Fh	-
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	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	コマンドのシーケンス番号
7	*bBWI*	1	00-FFh	現在転送している CCIDs ブロックの待機タイムアウトを拡張するために使用します。「この数値にブロックの待機時間を掛け」の期限が切れた後、CCID はブロックをタイムアウトにします。
8	*wLevelParameter*	2	0000h	RFU（TPDU 交換レベル）。
10	*abData*	バイト配列	-	CCID に送信されるデータブロック。データは ICC に「そのまま」送信されます（TPDU 交換レベル）

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

7.1.4. PC_to_RDR_GetParameters

スロットのパラメーターを取得する時にこのコマンドを使用します。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	6Ch	-
1	*dwLength*	4	00000000h	メッセージ固有のデータ長さ。
5	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	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。

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

7.1.5. PC_to_RDR_ResetParameters

スロットのパラメーターをリセットする時にこのコマンドを使用します。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	6Dh	-
1	*dwLength*	4	00000000h	メッセージ固有のデータ長さ。
5	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	コマンドのシーケンス番号。 **注：**スロット番号に関連しなくて、*FFh* の後に *00h* へロールオーバーします。
7	*abRFU*	3	-	保留して将来使います。

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

7.1.6. PC_to_RDR_SetParameters

スロットのパラメーターを設置する時にこのコマンドを使用します。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	61h	-
1	*dwLength*	4	-	このメセッジ中の abProtocolDataStructure フィルドの長さ
5	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	コマンドのシーケンス番号
7	*bProtocolNum*	1	00h、01h	下記は指定されたプロトコルのデータ構造です。 00h ： T=0 プロトコルの構造 01h ： T=1 プロトコルの構造 下記の値を保留して将来使います： 80h ： 2 線プロトコルの構造 81h ： 3 線プロトコルの構造 82h ： I2C プロトコル構造
8	*abRFU*	2	-	保留して将来使います。

オフセット	フィールド	大きさ	数値	説明
10	*abProtocolDataStructure*	バイト配列	-	プロトコルのデータ構造

T=0 プロトコルのデータ構造（*dwLength*=00000005h）

オフセット	フィールド	大きさ	数値	説明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中のテーブル 7 をインデックスして、クロックレートの変換係数を選択します B3-0 – DI - ISO/IEC 7816-3:1997 中のテーブル 8 をインデックスして、ボーレートの変換係数を選択します。
11	*bmTCCKST0*	1	00h、02h	B0 – 0b，B7-2 – 000000b B1 – 使用している約束（b1=0：直接；b1=1：逆） **注：***CCID* がこのビットを無視します。
12	*bGuardTimeT0*	1	00-FFh	2 文字間の余分な GuardTime。正常な GuardTime（12 ETU）に 0ー254 ETU を追加します。 FFh が 00h と同じです。
13	*bWaitingIntegerT0*	1	00-FFh	T=0 の場合 WI が WWT を定義する時に使われます
14	*bClockStop*	1	00 03h	ICC クロック停止サポート 00h = クロックを停止することは許可されていません 01h = クロック信号が低い時に停止されます 02h = クロック信号が高い時に停止されます 03h = クロック信号が低い時または高い時に停止されます

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

T=1 プロトコルのデータ構造（*dwLength*=00000007h）

オフセット	フィールド	大きさ	数値	説明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中のテーブル 7 をインデックスして、クロックレートの変換係数を選択します B3-0 – DI - ISO/IEC 7816-3:1997 中のテーブル 8 をインデックスして、ボーレートの変換係数を選択します。
11	*bmTCCKST1*	1	10h、 11h、 12h、 13h、	B7-2 – 000100b B0 – チェックサムタイプ（b0=0：LRC； b0=1：CRC） B1 – 使用している約束（b1=0：直接；b1=1：逆） **注：***CCID* がこのビットを無視します。
12	*bGuardTimeT1*	1	00-FFh	余計な GuardTime（2 文字間の余分な GuardTime は 0－254etu）は FFh である場合、GuardTime を 1 etu 減らします。
13	*bwaitingIntegerT1*	1	00-9Fh	B7-4 = BWI 値 0-9 有効 B3-0 = CWI 値 0-Fh 有効
14	*bClockStop*	1	00 -03h	ICC クロック停止サポート： 00h = クロックを停止することは許可されていません 01h = クロック信号が低い時に停止されます 02h = クロック信号が高い時に停止されます 03h = クロック信号が低い時または高い時に停止されます
15	*bIFSC*	1	00-FEh	交渉された IFSC の大きさ
16	*bNadValue*	1	-	CCID がデフォルト値以外の値をサポートしていない場合、値= 00h

このメッセージの応答は *RDR_to_PC_Parameters* メッセージです。

7.1.7. PC_to_RDR_Escape

このコマンドは拡張機能の定義およびアクセスのために使用されます。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	6Bh	-
1	*dwLength*	4	00000000h	このメッセージの abData データフィールドのサイズ
5	*bSlot*	1	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	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。

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

オフセット	フィールド	大きさ	数値	説明
10	*bcmdCode*	1	04h	-
11	*wcmdLength*	2	0000h	-
13	*abRFU*	2	-	保留して将来使います。

例：

```
bSendBuffer[0]=04h;
bSendBuffer[1]=00h;
bSendBuffer[2]=00h;
bSendBuffer[3]=00h;
bSendBuffer[4]=00h;
dwSendBufferLen=05h

SCARDStatus = SCardControl(  hSAM, SCARD_CTL_CODE(3500), bSendBuffer,
dwSendBufferLen, bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen) ;
```

7.1.7.2. LCD メッセージ表示（Display LCD Message）

オフセット	フィールド	大きさ	数値	説明
10	*bcmdCode*	1	05h	-
11	*wcmdLength*	2	0020h	-
13	*abRFU*	2	-	保留して将来使います。
15	*abData*	20	-	キーボードに入力して、データを LCD に表示する

例：

bSendBuffer[0]=05h;

bSendBuffer[1]=00h;

bSendBuffer[2]=20h;

bSendBuffer[3]=00h;

bSendBuffer[4]=00h;

bSendBuffer[abData]=(31 32 33 20 20 20 20 20 … 20h)

dwSendBufferLen=25h

SCARDStatus = SCardControl(hSAM, SCARD_CTL_CODE(3500), bSendBuffer, dwSendBufferLen, bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);

7.1.7.3. キー読み取り（Read Key）

オフセット	フィールド	大きさ	数値	説明
10	*bcmdCode*	1	06h	-
11	*wcmdLength*	2	0006h	-
13	*abRFU*	2	-	保留して将来使います。
15	*timeout*	1	00h	-
16	*PinLength*	2	XXYYh	XXh：PIN の最大長さ XXh：PIN の最小長さ
18	*KeyReturnCondition*	1	-	この値はビット単位での OR 演算です 01：最大サイズに達しました 02：Enter キーを押す 04：タイムアウト発生 08：Cancel キーを押す
19	*StartPosition*	1	-	Bit7-4：0000 は LCD の上段を示し、0001 は下の行を示します。 Bit3-0：表示位置を示す。

オフセット	フィールド	大きさ	数値	説明
20	*EchoLCDMode*	1	-	00： ASCII 表示 01： キャラクタ*表示

例：

bSendBuffer[0]=06h;

bSendBuffer[1]=00h;

bSendBuffer[2]=06h;

bSendBuffer[3]=00h;

bSendBuffer[4]=00h;

bSendBuffer[abData]=(00 08 04 01 00 00h)

dwSendBufferLen=0Bh

SCARDStatus = SCardControl(hSAM, SCARD_CTL_CODE(3500), bSendBuffer, dwSendBufferLen, bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);

7.1.7.4. ブザー（Buzzer Beep）

オフセット	フィールド	大きさ	数値	説明
10	*bcmdCode*	1	08h	-
11	*wcmdLength*	2	0000h	-
13	*abRFU*	2	-	保留して将来使います。

例：

bSendBuffer[0]=08h;

bSendBuffer[1]=00h;

bSendBuffer[2]=00h;

bSendBuffer[3]=00h;

bSendBuffer[4]=00h;

dwSendBufferLen=05h

SCARDStatus = SCardControl(hSAM, SCARD_CTL_CODE(3500), bSendBuffer, dwSendBufferLen, bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);

7.1.7.5. リーダーオプションコマンドを設定する（Set Reader Option Command）

オフセット	フィールド	大きさ	数値	説明
10	*bcmdCode*	1	13h	-
11	*wcmdLength*	2	0000h	-
14	*abRFU*	2	0000h	-
15	*ReaderOptions*	1	-	ビットを定義する。表 1 を参照してください**表 1**。

例：

bSendBuffer[0]=13h;
bSendBuffer[1]=00h;
bSendBuffer[2]=00h;
bSendBuffer[3]=00h;
bSendBuffer[4]=00h;
bSendBuffer[5]=02h;
dwSendBufferLen=06h

SCARDStatus = SCardControl(hSAM, SCARD_CTL_CODE(3500), bSendBuffer, dwSendBufferLen, bRecvBuffer, dwRecvBufferLen, &dwRecvBufferLen);

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
未使用	未使用	未使用	未使用	未使用	616C タグ	EMV モード	PPS モード

表 1： ReaderOptions ビット構造

7.1.8. PC_to_RDR_Secure

このコマンドは、確認または変更のための PIN を入力します。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	69h	-
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	00-FFh	このコマンドのスロット番号を識別します。
6	*bSeq*	1	00-FFh	コマンドのシーケンス番号

オフセット	フィールド	大きさ	数値	説明
7	*bBWI*	1	00-FFh	現在転送している CCIDs ブロックの待機タイムアウトを拡張するために使用します。「この数値にブロックの待機時間を掛け」の期限が切れた後、CCID はブロックをタイムアウトにします。
8	*wLevelParameter*	2	0000h	RFU（TPDU 交換レベル）。
10	*abData*	バイト配列	-	値は *wLevelParameters* に依存します。*wLevelParameters* = 0000h または 0001h の場合、*abData* = *abPINOperationDataStructure*。

7.1.8.1. abPINOperationDataStructure

オフセット	フィールド	大きさ	数値	説明
10	*bPINOperation*	1	00 -06h	PIN 操作を示すために使用します： 00h：PIN コード認証 01h：PIN コード変更
11	*abPINDataStructure*	バイト配列	-	PIN 検証データ構造 また PIN 変更データ構造

7.1.8.2. PIN 検証データ構造（PIN Verification Data Structure）

オフセット	フィールド	大きさ	数値	説明
11	*bTimeOut*	1	-	タイムアウト時間（秒数）。00h の場合、CCID のデフォルト値が使用されます。
12	*bmFormatString*	1	-	PIN フォーマットオプションのいくつかのパラメータ
13	*bmPINBlockString*	1	-	APDU コマンドで提示する PIN ブロックの長さをバイト単位で定義します。
14	*bmPINLengthFormat*	1	-	APDU コマンドに PIN を挿入できる
15	*wPINMaxExtraDigit*	2	XXYYh	XX：最小の PIN 長さ（ビット） XX：最大の PIN 長さ（ビット）

オフセット	フィールド	大きさ	数値	説明
17	*bEntryValidationCondition*	1	-	この値はビット単位での OR 演算です 01：最大サイズに達しました 02h：検証キーを押す 04h： タイムアウト
18	*bNumberMessage*	1	00h 01h FFh	PIN 検証管理のために表示するメッセージの数： 00h：文字列なし 01h：*bMsgIndex* インデックスには示されたメッセージ FFh：デフォルトの CCID メッセージ
19	*wLangId*	2	-	メッセージを表示するために使用される言語
21	*bMsgIndex*	1	-	Reader CCID メッセージテーブル中のメッセージインデックス（00h である必要があります）。このメッセージは、ユーザーが PIN を入力するためのプロンプトです。
22	*bTeoPrologue*	3	-	T = 1 I-block prologue フィールドに使う。T = 1 プロトコルが使用中である場合にのみ重要です。
25	*abPINApdu*	バイト配列	-	ICD に送る APDU

7.1.8.3. PIN 変更データ構造（PIN Modification Data Structure）

オフセット	フィールド	大きさ	数値	説明
11	*bTimeOut*	1	-	タイムアウト時間（秒数）。00h の場合、CCID のデフォルト値が使用されます。
12	*bmFormatString*	1	-	PIN フォーマットオプションのいくつかのパラメータ
13	*bmPINBlockString*	1	-	APDU コマンドで提示する PIN ブロックの長さをバイト単位で定義します。
14	*bmPINLengthFormat*	1	-	APDU コマンドに PIN を挿入できる

オフセット	フィールド	大きさ	数値	説明
15	*bInsertionOffsetOld*	1	-	現在の PIN が挿入する位置のオフセットバイト
16	*bInsertionOffsetNew*	1	-	新しい PIN が挿入する位置のオフセットバイト
17	*wPINMaxExtraDigit*	2	XXYYh	XX h ：最小の PIN 長さ（ビット） YYh：最大の PIN 長さ（ビット）
19	*bConfirmPIN*	1	00h、 01h、 02h、 03h	新しい PIN を承認する前に確認が要求された場合（ユーザーがこの新しい PIN を受け入れる前に 2 回入力する必要があります） 現在の PIN を同じ APDU フィールドに入力して設定する必要があるかどうかを示します。 b0：（0/1） ＝0 確認要求がない場合 ＝1 確認が要求された場合 b1：（0/1） ＝0 の場合、現在の PIN エントリは要求されていません。（この場合、*bInsertinoOffsetOld* の値は考慮に入れないでください）。 ＝1 現在の PIN エントリが要求された b2 - b7：保留して将来使います
20	*bEntryValidationCondition*	1	-	この値はビット単位での OR 演算です 01：最大サイズに達しました 02h：検証キーを押す 04h： タイムアウト

オフセット	フィールド	大きさ	数値	説明
21	*bNumberMessage*	1	00h、01h、02h、03h または FFh	PIN 変更コマンドのために表示するメッセージの数： 00h：メッセージない 01h：*bMsgIndex1* インデックスには示されたメッセージ 02h：*bMsgIndex1* と *bMsgIndex2* インデックスには示されたメッセージ 03h：*bMsgIndex1*、*bMsgIndex2* と *bMsgIndex3* インデックスには示されたメッセージ FFh：デフォルトの CCID メッセージ
22	*wLangId*	2	-	メッセージを表示するために使用される言語
24	*bMsgIndex1*	1	-	Reader メッセージテーブル中のメッセージインデックス（00h または 01 h である必要があります）。 このメッセージは、*bConfirmPIN* = 00h またはその現在の PIN の場合、新しい PIN を入力するためのプロンプトです。
25	*bMsgIndex2*	1	-	Reader メッセージテーブル中のメッセージインデックス（01h または 02 h である必要があります）。 このメッセージは、*bConfirmPIN* = 02h または 03 h の場合、新しい PIN を入力するためのプロンプトです。 このメッセージは、*bConfirmPIN* = 01 h の場合、もう一回新しい PIN を入力するためのプロンプトです。 （*bNumberMessage* が null でない場合のみ存在する）

オフセット	フィールド	大きさ	数値	説明
26	*bMsgIndex3*	1	-	Reader メッセージテーブル中のメッセージインデックス（02h である必要があります）。 このメッセージは、確認必要の場合、もう一回新しい PIN を入力するためのプロンプトです。 （*bNumberMessage* = 3 の場合のみ存在する）
25 または 2 6 または 27	*bTeoPrologue*	3	-	T = 1 I-block prologue フィールドに使う。T = 1 プロトコルが使用中である場合にのみ重要です。
28 または 29 または 30	*abPINApdu*	バイト配列	-	ICD に送る APDU

このメッセージの応答は *RDR_to_PC_DataBlock* メッセージです。

7.2. カードリーダーからパソコンへ

7.2.1. RDR_to_PC_DataBlock

このコマンドは APG8201 によって送信されて、*PC_to_RDR_IccPowerOn*、*PC_to_RDR_XfrBlock* および *PC_to_RDR_Secure* メッセージに対しての応答です。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	80h	CCID がデータブロックを送信しています。
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。Bulk-OUT メッセージ中の値と同じです。
6	*bSeq*	1	-	対応のコマンドのシーケンス番号。Bulk-OUT メッセージ中の値と同じです。
7	*bStatus*	1	-	スロットステータスレジスタ
8	*bError*	1	-	スロットエラーレジスタ
9	*bChainParameter*	1	00h	RFU（TPDU 交換レベル）。
10	*abData*	バイト配列	-	このデータフィールドは CCID から返されたデータを含めています。

7.2.2. RDR_to_PC_SlotStatus

このコマンドは APG8201 によって送信されて、*PC_to_RDR_IccPowerOff* メッセージに対しての応答です。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	81h	-
1	*dwLength*	4	00000000h	メッセージ固有のデータ長さ。
5	*bSlot*	1	-	このコマンドのスロット番号を識別します。Bulk-OUT メッセージ中の値と同じです。
6	*bSeq*	1	-	対応のコマンドのシーケンス番号。Bulk-OUT メッセージ中の値と同じです。
7	*bStatus*	1	-	スロットステータスレジスタ
8	*bError*	1	-	スロットエラーレジスタ

オフセット	フィールド	大きさ	数値	説明
9	*bClockStatus*	1	00h、01h、02h、03h	数値： 00h = クロック動作中 01h = L 状態に止まっている 02h = H 状態に止まっている 03h = 不明な状態に止まっている 残された値を保留して将来使います。

7.2.3. RDR_to_PC_Parameters

このコマンドは APG8201 によって送信されて、

PC_to_RDR_GetParameters、*PC_to_RDR_ResetParameters* および *PC_to_RDR_SetParameters* メッセージに対しての応答です。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	82h	-
1	*dwLength*	4	-	このメセッジ中の *abProtocolDataStructure* フィルドの長さ
5	*bSlot*	1	-	このコマンドのスロット番号を識別します。Bulk-OUT メッセージ中の値と同じです。
6	*bSeq*	1	-	対応のコマンドのシーケンス番号。Bulk-OUT メッセージ中の値と同じです。
7	*bStatus*	1	-	スロットステータスレジスタ
8	*bError*	1	-	スロットエラーレジスタ
9	*bProtocolNum*	1	00h, 01h	下記は指定されたプロトコルのデータ構造です。 00h = T=0 プロトコルの構造 01h = T=1 プロトコルの構造 下記の値を保留して将来使います： 80h = 2 線プロトコルの構造 81h = 3 線プロトコルの構造 82h = I2C プロトコル構造
10	*abProtocolDataStructure*	バイト配列	-	プロトコルのデータ構造

T=0 プロトコルのデータ構造（*bProtocolNum=0*、*dwLength=00000005h*）

オフセット	フィールド	大きさ	数値	説明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中のテーブル 7 をインデックスして、クロックレートの変換係数を選択します B3-0 – DI - ISO/IEC 7816-3:1997 中のテーブル 8 をインデックスして、ボーレートの変換係数を選択します
11	*bmTCCKST0*	1	00h, 02h	プロトコルは T=0，B0 – 0b，B7-2 – 000000b の場合 B1 – 使用している約束（b1=0：直接；b1=1：逆）
12	*bGuardTimeT0*	1	00-FFh	2 文字間の余分な GuardTime。正常な GuardTime（12 ETU）に 0−254 ETU を追加します。 FFh が 00h と同じです。
13	*bWaitingIntegerT0*	1	00-FFh	T=0 の場合 WI が WWT を定義する時に使われます
14	*bClockStop*	1	00-03h	ICC クロック停止サポート： 00h = クロックを停止することは許可されていません 01h = クロック信号が低い時に停止されます 02h = クロック信号が高い時に停止されます 03h = クロック信号が低い時または高い時に停止されます

T=1 プロトコルのデータ構造（*bProtocolNum=1*、*dwLength=00000007h*）

オフセット	フィールド	大きさ	数値	説明
10	*bmFindexDindex*	1	-	B7-4 – FI – ISO/IEC 7816-3:1997 中のテーブル 7 をインデックスして、クロックレートの変換係数を選択します B3-0 – DI - ISO/IEC 7816-3:1997 中のテーブル 8 をインデックスして、ボーレートの変換係数を選択します

オフセット	フィールド	大きさ	数値	説明
11	*bmTCCKST1*	1	10h、 11h、 12h、 13h、	プロトコルは T=1、B7-2 – 000100b の場合 B0 – チェックサムタイプ（b0=0：LRC；b0=1：CRC） B1 – 使用している約束（b1=0：直接；b1=1：逆）
12	*bGuardTimeT1*	1	00-FFh	余計な GuardTime（2 文字間の余分な GuardTime は 0－254etu）は FFh である場合、GuardTime から 1 を減らします。
13	*BwaitingIntegerT1*	1	00-9Fh	B7-4 = BWI B3-0 = CWI
14	*bClockStop*	1	00 -03h	ICC クロック停止サポート： 00 = クロックを停止することは許可されていません 01 = クロック信号が低い時に停止されます 02 = クロック信号が高い時に停止されます 03 = クロック信号が低い時または高い時に停止されます
15	*bIFSC*	1	00-FEh	交渉された IFSC の大きさ
16	*bNadValue*	1	00-FFh	CCID によって使用されるナド値

7.2.4. RDR_to_PC_Escape

このコマンドは APG8201 によって送信されて、*PC_to_RDR_Escape* メッセージに対しての応答です。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	83h	-
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。Bulk-OUT メッセージ中の値と同じです。
6	*bSeq*	1	-	対応のコマンドのシーケンス番号。Bulk-OUT メッセージ中の値と同じです。
7	*bStatus*	1	-	スロットステータスレジスタ

オフセット	フィールド	大きさ	数値	説明
8	*bError*	1	-	スロットエラーレジスタ
9	*abRFU*	1	00h	保留して将来使います。
10	*abData*	バイト配列	-	ＣＣID から送信されたでータ

7.2.4.1. リーダー固有のタグを取得する（Get Reader Specific Tag）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	80h	-
11	*wcmdLength*	1	04h	-
12	*abData*	4	00h	-

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

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	84h	-
11	*wcmdLength*	2	0004h	-
13	*abRFU*	2	0000h	保留して将来使います。
15	*abData*	4	-	ファームウェアのバージョン番号

7.2.4.3. LCD メッセージ表示（Display LCD Message）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	85h	-
11	*wcmdLength*	2	00h	-
12	*abRFU*	2	00h	保留して将来使います。

7.2.4.4. キー読み取り（Read Key）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	86h	-
11	*wResLength*	2	-	-
13	*abRFU*	2	0000h	-
15	*abData*	バイト配列	-	カードリーダーから送信されたデータ

7.2.4.5. ブザー（Buzzer Beep）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	88h	-
11	*abRFU*	4	-	-

7.2.4.6. LCD 生産テスト機能（LCD Production Test Function）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	8Fh	-
11	*wResLength*	2	0000h	-
13	*abRFU*	2	0000h	-

7.2.4.7. リーダーオプションコマンドを設定する（Set Reader Option Command）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	93h	-
11	*wResLength*	2	0000h	-
13	*abStatus*	2	-	0000h：成功 0001h：BAD_PARAMETER

7.2.4.8. 生産テストコマンド （Production Test Command）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	8Ch	-
11	*wResLength*	2	0000h	-
13	*abStatus*	2	-	0000h：成功 0001h：BAD_PARAMETER

7.2.4.9. 認証（Authentication）

オフセット	フィールド	大きさ	数値	説明
10	*bRespType*	1	8Dh	-
11	*wResLength*	2	0008h	-
13	*abStatus*	2	-	0000h：成功 0001h：BAD_PARAMETER
15	*abData*	8	-	8 バイトの認証データ

7.2.5. RDR_to_PC_NotifySlotChange

このメッセージは、APG8201 が ICC スロットの挿入状態の変化を検出するたびに送信されます。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	50h	-
1	*bmSlotICCState*	-	-	このフィールドはバイト単位で報告されます。 サイズは（2 ビット*スロット数）で、最も近いバイトに切り上げられます。 各スロットは 2 ビットを有します。最下位ビットは、スロットの現在の状態を報告します（0b = ICC が存在しない、1b = ICC が存在する）。最上位ビットは、最後の *RDR_to_PC_NotifySlotChange* メッセージが送信されてからスロットが状態を変更したかどうかを報告します（0b = 変更なし、1b =変更）。 特定の位置にスロットが存在しない場合、このフィールドはその 2 ビットで 00b を返します。 例：3 つスロットの CCID は、次の形式の 1 バイトを報告しています： Bit 0 = スロット 0 の現在の状態 Bit 1 = スロット 0 の変化の状態 Bit 2 = スロット 1 の現在の状態 Bit 3 = スロット 1 の変化の状態 Bit 4 = スロット 2 の現在の状態 Bit 5 = スロット 2 の変化の状態 Bit 6 = 0b Bit 7 = 0b

7.2.6. RDR_to_PC_HardwareError

このメッセージは、*bHardwareErrorCode* フィールドの任意のビットが設定されたときに送信されます。

オフセット	フィールド	大きさ	数値	説明
0	*bMessageType*	1	51h	-
1	*bSlot*	1	00-FFh	ICC スロットナンバー
2	*bSeq*	1	00-FFh	ハードウェアエラー発生時の bulk out コマンドのシーケンス番号
3	*bHardwareErrorCode*	1	XYh	この値は、以下の値に対して実行されるビット単位の OR 演算です。 01h =過電流 定義されるさらなるエラー条件および値。

操作フローの詳細については、CCID 仕様を参照してください。

8.0. PC リンク操作モード

PC リンクモードでは、カードとリーダー間の認証として SPE を選択できます。 スタンドアローンモードで動作する場合、ユーザは、未接続モード PLA および計算機などの付加価値機能を選択することができます。

リーダが PC リンクモードになっている場合、ユーザは PCSC API SCardConnect、SCardTransmit、SCardControl などを選択してリーダにコマンドを送信し、カードとリーダ間の基本的な操作を行うことができます 。SCardControl API を使用して、セキュア・ピン・ベリファイ、セキュア・ピン・モディファイ、および異なるエスケープ・コマンド通信を行うことができます

8.1. SCardConnect API

```
LONG WINAPI SCardConnect(
  _In_   SCARDCONTEXT hContext,
  _In_   LPCTSTR szReader,
  _In_   DWORD dwShareMode,
  _In_   DWORD dwPreferredProtocols,
  _Out_  LPSCARDHANDLE phCard,
  _Out_  LPDWORD pdwActiveProtocol
);
```

8.2. SCardTransmit API

```
LONG WINAPI SCardTransmit(
  _In_         SCARDHANDLE hCard,
  _In_         LPCSCARD_IO_REQUEST pioSendPci,
  _In_         LPCBYTE pbSendBuffer,
  _In_         DWORD cbSendLength,
  _Inout_opt_  LPSCARD_IO_REQUEST pioRecvPci,
  _Out_        LPBYTE pbRecvBuffer,
  _Inout_      LPDWORD pcbRecvLength
);
```

8.3. SCardControl API

```
LONG WINAPI SCardControl(
  _In_   SCARDHANDLE hCard,
  _In_   DWORD dwControlCode,
  _In_   LPCVOID lpInBuffer,
  _In_   DWORD nInBufferSize,
  _Out_  LPVOID lpOutBuffer,
  _In_   DWORD nOutBufferSize,
  _Out_  LPDWORD lpBytesReturned
);
```

8.4. 安全な暗証番号確認

アプリケーションはログオンコマンドを送信します（SPE セキュアピンエントリ）。

例： “00 20 00 01 08 24 12 34 FF FF FF FF FFh”

ユーザが PIN を入力すると、リーダは、SECURE PIN VERIFY コマンドをカードに送信します。

特定のステータスコード：

SW1 SW2	意味
90 00h	エラーなし
63 Cxh	間違っている PIN
64 01h	PIN 入力キャンセル

現在の ACS スマートカードの場合、PIN の入力数は 3 に設定されています。3 回目の試行で誤った PIN が入力された場合、カードはブロックされます。

注：*APG8201 Generic Reader* の正しい *PIN* 入力と誤った *PIN* 入力の両方のテキストは表示されません。

8.5. 安全な暗証番号変更

次の 2 つの方法で SECURE PIN MODIFY コマンドを呼び出すことができます。

1. 明示的なセキュアピンの変更の場合、ホストは SECURE PIN ENTRY と SECURE PIN MODIFY の 2 つのコマンドを別々に送信する必要があります。
   例：PIN 安全入力（ホスト->リーダー）：69 1C 00 00 00 00 F3 00 00 00 00 00 89 47 04 0C 04 07 01 09 04 00 00 00 00 00 20 00 02 08 2C FF FF FF FF FF FF FFh
   LCD ディスプレイ："Enter auth. PIN:"
   ユーザーが入力する 12 ビットの PUK：例 3 3 3 3 3 3 1 1 1 1 1 1
   LCD ディスプレイ："Card inserted"
   PIN 安全入力（リーダー->ホスト）：80 02 00 00 00 00 F3 00 00 00 90 00h
   PIN 安全変更（ホスト->リーダー）：69 1F 00 00 00 00 F4 00 00 00 01 00 89 47 04 00 00 0C 04 00 03 01 09 04 01 00 00 00 00 24 01 01 08 24 FF FF FF FF FF FF FFh
   LCD ディスプレイ："NEW PIN: (key)"
   ユーザーは 4 ビットの新しい PIN を入力します：例 1 2 3 4，そして"OK"をクリックします
   LCD ディスプレイ："CONFIRM PIN: (key)"
   ユーザーはもう一回 4 ビットの新しい PIN を入力します：例 1 2 3 4，そして"OK"をクリックします
   LCD ディスプレイ："Card inserted"
   PIN 安全変更（リーダー->ホスト）：80 02 00 00 00 00 F4 00 00 00 90 00h

2. 暗黙的な SECURE PIN MODIFY の場合、ホストは検証と変更の両方を行うために 1 つのコマンドを送信する必要があります。
   PIN 安全変更（ホスト->リーダー）：69 29 00 00 00 00 CF 00 00 00 01 00 89 47 04 00 08 0C 04 03 03 03 09 04 00 01 02 00 00 00 00 24 00 01 10 24 FF FF FF FF FF FF FF 24 FF FF FF FF FF FF FFh
   LCD ディスプレイ："Enter auth. PIN:"
   ユーザーは 4 ビットの正しい PIN を入力します：例 1 2 3 4，そして"OK"をクリックします
   LCD ディスプレイ："NEW PIN: (key)"
   ユーザーは 4 ビットの新しい PIN を入力します：例 4 3 2 1，そして"OK"をクリックします

LCD ディスプレイ："CONFIRM PIN: (key)"
ユーザーはもう一回４ビットの新しい PIN を入力します：例 4 3 2 1，そして"OK"をクリックします
LCD ディスプレイ："Card inserted"
PIN 安全変更（リーダー->ホスト）：80 02 00 00 00 00 CF 00 00 00 90 00h

暗黙のコマンド中に古い PIN が間違っている場合、カードからの応答は "63 Cxh"でなければなりません。ここで x は PIN がブロックされるまでの残りの試行の回数です。エラーコード "69 83h"が表示されます。

8.6. 直接コマンド

リーダは、リーダ固有タグの取得、ファームウェアバージョンの取得、LCD メッセージの表示、キーパッドからの読み取りキー、ブザービープ音などの異なるエスケープコマンドと、プロダクション専用のいくつかのコマンドを提供します。

8.7. ファームウェアのバージョン番号を取得する

エスケープコマンドコード "04 00 00 00 00h"は、ファームウェアのバージョンを取得するために使用されます。

ファームウェアバージョンを入手する（ホスト - >リーダー）：04 00 00 00 00h

ファームウェアバージョンを取得する（Reader-> Host）：84 00 02 00 00 30 31 39 5Ah

8.8. LCD メッセージを表示する

エスケープコマンドコード "05 00 20 00 00h ..."は、LCD 画面にテキストを表示するために使用されます。

ファームウェアバージョン取得（ホスト→リーダー）：05 00 20 00 00 31 32 33 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20h

ファームウェアのバージョンを入手する（Reader-> Host）：85 00 00 00 00h

8.9. キーを読み取りする

エスケープコマンドコード "06 00 06 00 00 00 08 04 01 00 00h"は、ユーザーがキーパッドで入力したキーを読み取るために使用されます。

ファームウェアのバージョンを入手する（ホスト - >リーダー）：06 00 06 00 00 08 04 01 00 00h

ファームウェアバージョンを取得する（Reader-> Host）：86 00 09 00 00 31 31 32 33 34 35 36 37 38h

8.10. ブザー

エスケープコマンドコード "08 00 00 00 00h"は、ビープ音を鳴らすために使用されます。

ブザー音（ホスト - >リーダー）：08 00 00 00 00h

ブザー音（リーダー→ホスト）：88 00 00 00 00h

9.0. デバイスコントロール

このセクションでは、システムスマートカードデバイスの IOCTL について説明します。

9.1. プロセス（PC/SC 2.0 パート 10）

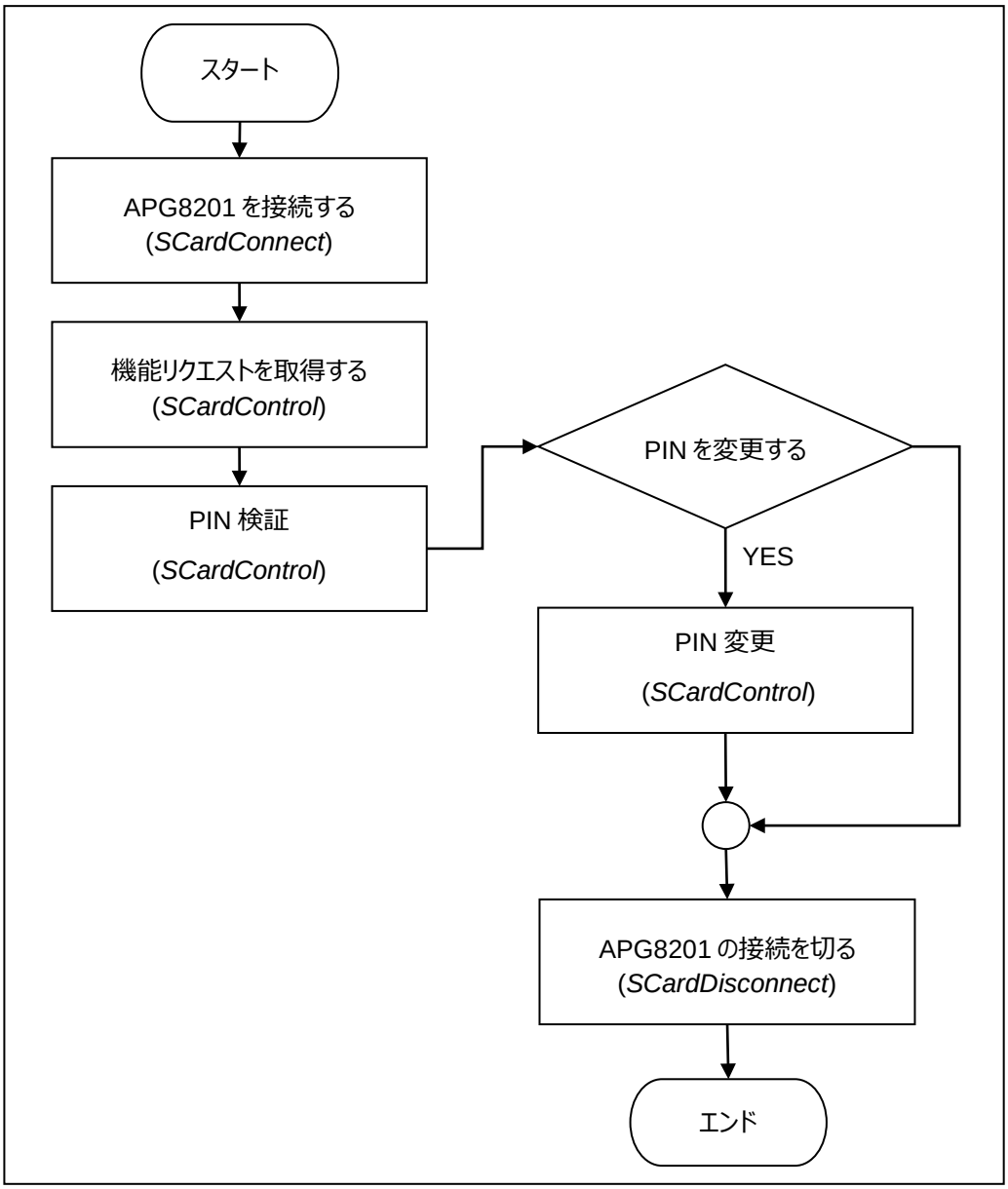


図 1：プロセス

PIN の検証と変更を使用するには、*SCardControl* API を Get Feature Request コントロールコードで呼び出す必要があります。この API は、サポートされている機能のリストをリーダーから返します。

APG8201 では、PIN 直接検証、PIN 直接変更、および IFD PIN のプロパティのみがサポートされています。 これらの機能を使用するには、リストから制御コードを取得します。詳細については、PC / SC 2.0 仕様パート 10 を参照してください。

9.2. 具体の ScardControl 関数

```
LONG SCardControl(
SCARDHANDLE hCard,
DWORD dwControlCode,
LPCVOID lpInBuffer,
DWORD nInBufferSize,
LPVOID lpOutBuffer,
DWORD nOutBufferSize,
LPDWORD lpBytesReturned
);
#define IOCTL_SMARTCARD_GET_FIRMWARE_VERSION SCARD_CTL_CODE(2078)
#define IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE SCARD_CTL_CODE(2079)
#define IOCTL_SMARTCARD_READ_KEY SCARD_CTL_CODE(2080
// PC/SC 2.0 Part 10
#define CM_IOCTL_GET_FEATURE_REQUEST SCARD_CTL_CODE(3400)
```

注：データは、*LSB*（*Least Significant Byte*）が最初にあるリトルエンディアン形式で格納されます。 さらに、*SCardControl* コマンドはソースコードで宣言する必要があります。

9.3. スマートカードデバイス IOCTL

9.3.1. CM_IOCTL_GET_FEATURE_REQUEST

CM_IOCTL_GET_FEATURE_REQUEST は、サポートされている機能のリストをリーダから返します。

hCard	*SCardConnect* から返された参照値
dwControlCode	CM_IOCTL_GET_FEATURE_REQUEST
lpInBuffer	NULL
nInBufferSize	*lpInBuffer* の **sizeof** でなければなりません(ULONG)
lpOutBuffer	PC / SC 2.0 仕様パート 10 によれば、以下の機能が定義されています。

```
#define FEATURE_VERIFY_PIN_START 0x01
#define FEATURE_VERIFY_PIN_FINISH 0x02
#define FEATURE_MODIFY_PIN_START 0x03
#define FEATURE_MODIFY_PIN_FINISH 0x04
#define FEATURE_GET_KEY_PRESSED 0x05
#define FEATURE_VERIFY_PIN_DIRECT 0x06
#define FEATURE_MODIFY_PIN_DIRECT 0x07
#define FEATURE_MCT_READERDIRECT 0x08
#define FEATURE_MCT_UNIVERSAL 0x09
#define FEATURE_IFD_PIN_PROP 0x0A
#define FEATURE_ABORT 0x0B
```

APG8201 は以下の機能を備えています：

```
#define FEATURE_VERIFY_PIN_DIRECT 0x06
#define FEATURE_MODIFY_PIN_DIRECT 0x07
#define FEATURE_IFD_PIN_PROP 0x0A
```

使用しているAPG8201リーダーがPC / SC 2.0 Part 10をサポートしている場合、次のデータが得られます。

06 04 XX XX XX XX 07 04 XX XX XX XX 0A 04 XX XX XX XXh

ここで、XX XX XX XXh は機能の制御コードです。

nOutBufferSize *lpOutBuffer* の **sizeof**(ULONG)

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

9.3.2. FEATURE_VERIFY_PIN_DIRECT

hCard *SCardConnect* から返された参照値

dwControlCode CM_IOCTL_GET_FEATURE_REQUEST

lpInBuffer

オフセット	データフィールド	大きさ	数値	説明
0	*bTimeOut*	1	-	秒数値が 00h である場合、デフォルト値が使用されます。
1	*bTimeOut2*	1	00h	サポートできない最初のキーストローク後の秒数。
2	*bmFormatString*	1	-	PIN フォーマットオプションのいくつかのパラメータ。詳細については、**bmFormatString 説明**を参照してください。
3	*bmPINBlockString*	1	-	APDU コマンド中の PIN ブロックの長さをバイト単位で定義します。詳しい情報が **bmPINBlockString 説明** を参照してください。
4	*bmPINLengthFormat*	1	-	APDU コマンドに PIN を挿入できる詳しい情報が **bmPINLength フォーマット**を参照してください。
5	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN の最大長さ（ビット） XXh：PIN の最小長さ（ビット）
7	*bEntryValidationCondition*	1	-	この値はビット単位の OR 演算です 01 h ＝最大サイズに達しました 02h = 確認のボタンを押す 04h = タイムアウト発生
8	*bNumberMessage*	1	FFh	PIN 検証のために表示するメッセージの数

オフセット	データフィールド	大きさ	数値	説明
9	*wLangId*	2	0409h	メッセージの言語
11	*bMsgIndex*	1	00h	メッセージインデックス（00 h であるはずです）
12	*bTeoPrologue*	3	000000h	使用する T = 1（I-block）のプロローグフィールド（00h を入れ込む）
15	*ulDataLength*	4	-	ICC に送信するデータの長さ
19	*abData*	-	-	ICC に送信するデータ

nInBufferSize 19 + *uLDataLength*

lpOutBuffer

オフセット	データフィールド	大きさ	数値	説明
0	*abStatus*	2	-	6400h：SPE 操作タイムアウト 6401h：「キャンセル」ボタンで SPE 操作をキャンセルしました 6402h：2 つの「新しい PIN」エントリが一致しないため、PIN 操作の変更に失敗しました 6403h：MIN / MAX PIN の長さに関して、ユーザーが短すぎたり長すぎたりする PIN を入力しました。 **注：** *APG8201* は、入力中に *PIN* の長さをチェックするため、このステータスを戻しません。 6B80h：渡された構造体のパラメータが無効です SW1SW2：カードからの結果

nOutBufferSize 2

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取ります。

9.3.3. FEATURE_MODIFY_PIN_DIRECT

hCard *SCardConnect* から返された参照値

dwControlCode CM_IOCTL_GET_FEATURE_REQUEST

lpInBuffer

オフセット	データフィールド	大きさ	数値	説明
0	*bTimeOut*	1	-	秒数値が 00h である場合、デフォルト値が使用されます。
1	*bTimeOut2*	1	00h	サポートできない最初のキーストローク後の秒数。
2	*bmFormatString*	1	-	PIN フォーマットオプションのいくつかのパラメータ。詳細については、**bmFormatString 説明**を参照してください。
3	*bmPINBlockString*	1	-	APDU コマンド中の PIN ブロックの長さをバイト単位で定義します。詳しい情報が **bmPINBlockString 説明**を参照してください。
4	*bmPINLengthFormat*	1	-	APDU コマンドに PIN を挿入できる詳しい情報が **bmPINLength 格式**を参照してください。
5	*bInsertionOffsetOld*	1	-	現在の PIN が挿入する位置のオフセット（バイト）
6	*bInsertionOffsetNew*	1	-	新しい PIN が挿入する位置のオフセット（バイト）
7	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN の最大長さ（ビット） XXh：PIN の最小長さ（ビット）

オフセット	データフィールド	大きさ	数値	説明
9	*bConfirmPIN*	1	00h, 01h, 02h, 03h	新しい PIN の承認前に確認が要求されたかどうかを示します（ユーザーがこの新しい PIN を受け入れる前に 2 回入力する必要があることを意味します） 現在の PIN を同じ APDU フィールドに入力して設定する必要があるかどうかを示します。 b0：(0/1) 0 =確認要求がない場合 1 =確認が要求された場合 b1：(0/1) 0＝現在の PIN エントリは要求されていません。（この場合、*bInsertinoOffsetOld* の値は考慮に入れないでください）。 1＝ 現在の PIN エントリが要求された b2 – b7：RFU
10	*bEntryValidationCondition*	1	-	この値はビット単位の OR 演算です 01 h ＝最大サイズに達しました 02h = 確認のボタンを押す 04h = タイムアウト発生
11	*bNumberMessage*	1	FFh	PIN 検証のために表示するメッセージの数
12	*wLangId*	2	0409h	メッセージの言語
14	*bMsgIndex1*	1	00h	第 1 回のプロンプトメッセージの索引
15	*bMsgIndex2*	1	01h	第 2 回のプロンプトメッセージの索引
16	*bMsgIndex3*	1	02h	第 3 回のプロンプトメッセージの索引
17	*bTeoPrologue*	3	000000h	使用する T = 1（I-block）のプロローグフィールド （00h を入れ込む）
20	*ulDataLength*	4	-	ICC に送信するデータの長さ
24	*abData*		-	ICC に送信するデータ

nInBufferSize 24 + ulDataLength

lpOutBuffer

オフセット	データフィールド	大きさ	数値	説明
0	*abStatus*	2	-	6400h：SPE 操作タイムアウト 6401h：「キャンセル」ボタンで SPE 操作をキャンセルしました 6402h：2 つの「新しい PIN」エントリが一致しないため、PIN 操作の変更に失敗しました 6403h：MIN / MAX PIN の長さに関して、ユーザーが短すぎたり長すぎたりする PIN を入力しました。 **注：***APG8201* は、入力中に *PIN* の長さをチェックするため、このステータスを戻しません。 6B80h：渡された構造体のパラメータが無効です SW1SW2：カードからの結果

nOutBufferSize 2

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

9.3.4. FEATURE_IFD_PIN_PROP

hCard *SCardConnect* から返された参照値

dwControlCode CM_IOCTL_GET_FEATURE_REQUEST から返されます。

lpInBuffer NULL

LpOutBuffer

オフセット	データフィールド	大きさ	数値	説明
0	*wLcdLayout*	2	0210h	表示特性：2 行、1 行につき 16 文字

オフセット	データフィールド	大きさ	数値	説明
2	*bEntryValidationCondition*	1	07h	サポートのタイムアウトに達しました。 最大 PIN サイズに達しました。確認ボタンを押しました。
3	*bTimeOut2*	1	00h	0＝IFD は bTimeOut と bTimeOut2 を区別できない 1＝IFD は bTimeOut と bTimeOut2 を区別できる

nOutBufferSize 4

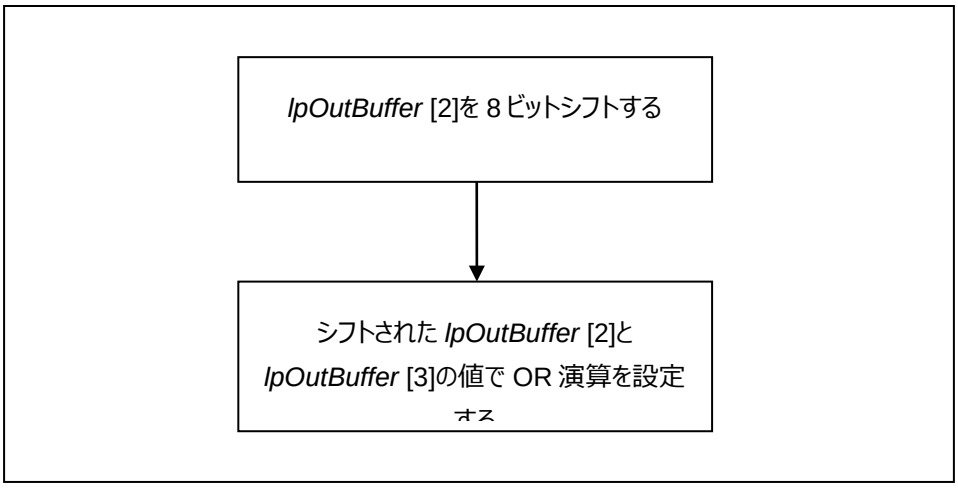
lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

9.3.5. IOCTL_SMARTCARD_GET_FIRMWARE_VERSION

IOCTL_SMARTCARD_GET_FIRMWARE_VERSION は *Get Firmware Version* コマンドを有効する時に使われます。

9.3.5.1. ファームウェアのバージョン番号

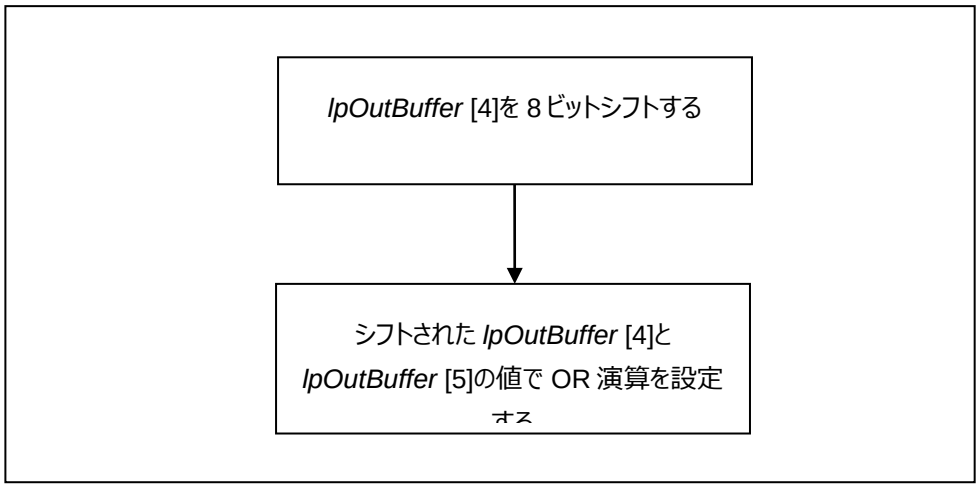
デバイスファームウェアのバージョンを取得するには、受信したバッファの 3 番目の要素を取り出し、8 ビットシフトします。その結果を受信したバッファの 4 番目の要素とともに OR 演算に設定します。



例： *Firmware_Version = (Common.RecvBuff[2] << 8) | Common.RecvBuff[3]);*

9.3.5.2. LCD

デバイス LCD 情報を取得するには、受信したバッファの 5 番目の要素を取り出し、8 ビットシフトします。 その結果を受信したバッファの 6 番目の要素とともに OR 演算に設定します。



データ入力：

hCard *SCardConnect* から返された参照値

dwControlCode IOCTL_SMARTCARD_GET_FIRMWARE_VERSION

データ出力：

lpOutBuffer コマンドの出力値

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

オフセット	データフィールド	大きさ	数値	説明
0	*abStatus*	2	0000h	成功
2	*wACR83Firmware*	2		-
4	*LCD*	2		-

9.3.6. IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE

IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE は *Display LCD Message* コマンドを有効する時に使われます。

hCard *SCardConnect* から返された参照値

dwControlCode IOCTL_SMARTCARD_DISPLAY_LCD_MESSAGE

lpInBuffer Display LCD Message オプションの設定値

nOutBufferSize *lpOutBuffer* の **sizeof**(ULONG)

オフセット	データフィールド	大きさ	数値	説明
0	*abLCDmessage*	0-32	-	LCD メッセージ（最大 32 文字）

データ出力：

lpOutBuffer コマンドの出力値

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

オフセット	データフィールド	大きさ	数値	説明
0	*abStatus*	2	0000h 0001h	成功 BAD_PARAMETER

9.3.7. IOCTL_SMARTCARD_READ_KEY

IOCTL_SMARTCARD_READ_KEY は *Read Key* コマンドを有効する時に使われます。

データ入力：

hCard *SCardConnect* から返された参照値

dwControlCode IOCTL_SMARTCARD_READ_KEY

lpInBuffer Display LCD Message オプションの設定値

nOutBufferSize *lpOutBuffer* の **sizeof**(ULONG)

オフセット	データフィールド	大きさ	数値	説明
0	*bTimeOut*	1	-	秒数値が 00h である場合、デフォルト値が使用されます。
1	*wPINMaxExtraDigit*	2	XXYYh	XXh：PIN の最大長さ（ビット） XXh：PIN の最小長さ（ビット）
3	*bKeyReturnCondition*	1	-	この値はビット単位の OR 演算です 01 h ：最大サイズに達しました 02h：[E]を押しました 04h：タイムアウト発生 08h：[C]を押しました
4	*bEchoLCDStartPosition*	1	-	開始位置（0～31）
5	*bEchoLCDMode*	1	-	00h：Echo キーを ASCII 表示で LCD に表示 01h：Echo キーをアスタリスク "*" で LCD に表示

データ出力：

lpOutBuffer コマンドの出力値

nOutBufferSize *lpOutBuffer* 的 **sizeof**(ULONG)

lpBytesReturned DWORD へのポインタ、*lpOutBuffer* に指定するバッファに格納されたデータのサイズをバイト数で受け取る

オフセット	データフィールド	大きさ	数値	説明
0	*abStatus*	2	0000h 0001h	成功 BAD_PARAMETER
2	*bKeyReturnCondition*	1	31h 32h 33h 34h	最大サイズに達しました [E]を押しました タイムアウト発生 [C]を押しました
3	*abNumericInputKeys*	0-32	-	-

付録 A. bKeyReturnCondition 設定

bKeyReturnCondition	OR オペランド
最大の PIN サイズに達した場合	01h
APG8201 デバイスの KEY_E が押された場合	02
APG8201 セッションの TIMEOUT に達した場合	04h
APG8201 デバイスの KEY_C が押された場合	08h
APG8201 デバイスの KEY_BACK が押された場合	10h
APG8201 デバイスの KEY_FN が押された場合	20h

注：特定の *OR* オペランドを再び *OR* 演算に設定してください。

付録 B. 応答エラーコード

下記のテーブルは APG8201（CCID）が返す可能なエラーコードをまとめています：

エラーコード	状態
0001h	BAD_PARAMETER
0083h	SLOTERROR_LCDCOMMANDERROR
0084h	SLOTERROR_WRONGCONFIRMPIN
0085h	SLOTERROR_UNKNOWN_LCD
0086h	SLOTERROR_MAXPINSIZE_EQUAL_ZERO
00EFh	SLOTERROR_PIN_CANCELLED
00F0h	SLOTERROR_PIN_TIMEOUT

付録 C. bmFormatString 説明

ビットナンバー	説明
Bit 7	システムユニットのタイプインジケータ： 0h の場合：システム単位はビットです 0h の場合：システム単位はバイトです このビットは、次のパラメータ（単位移動）を定量化します。
Bit 6 – 3	APDU コマンドでフォーマット後の PIN 位置を定義します（*Lc* の後の最初のデータを基準にします）。この位置は、システム・ユニットのタイプ・インジケータに基づいています（15 システム・ユニットで最大 1111）。
Bit 2	PIN の正当化のためのビットマスク： 0h の場合：データの左寄せ 1h の場合：データの右寄せ
Bit 1-0	PIN フォーマットタイプのビット単位： 00h：バイナリ 01h：BCD 10h：ASCII

付録 D. bmPINBlockString 説明

ビットナンバー	説明
Bit 7 - 4	APDU コマンドに挿入された PIN 長のサイズ、ビット単位で。（値が 0h の場合、有効な PIN 長は APDU コマンドに挿入されません）
Bit 3 - 0	PIN 長さ情報：両端揃えと書式設定後の PIN ブロックサイズ（バイト単位）

付録 E. bmPINLength フォーマット

ビットナンバー	説明
Bit 7-5	RFU
Bit 4	システムユニットのタイプインジケータ： 0h の場合：システム単位はビットです 0h の場合：システム単位はバイトです
Bit 3 - 0	APDU コマンドの PIN の長さの位置は、前のパラメータ（15 システム単位で最大 1111）に従って指定します。