CASIO

SAM ライブラリマニュアル

このマニュアルは、SAM ライブラリの
仕様について記載します。

Ver. 1.00

ご注意

- このソフトウェアおよびマニュアルの、一部または全部を無断で使用、複製することはできません。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

変更履歴

バージョン	変更日付	ページ	内容
0.90	2011.11		仮版作成
0.91	2011.12		解説文修正および関数追加
0.92	2012.01	7	SAMPowerUpCard関数のパラメータを修正
1.00	2012.03	1	SAM カードの解説を追加
		4	関数呼び出し手順を追加

目次

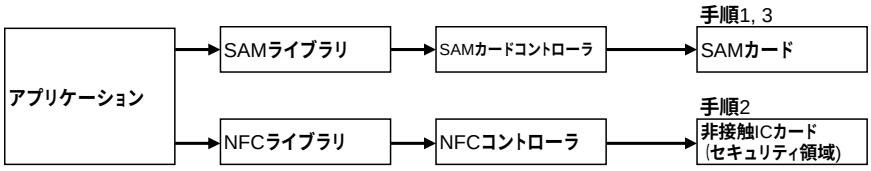
1. 概要

SAM（Secure Application Module）ライブラリは、SAM スロットに挿入した SAM カードとの通信を行う関数を提供します。

SAM カード

SAM カードは、非接触 IC カード内のセキュリティが保護された領域(セキュリティ領域)のデータの読み書きを行なう際に使用します。非接触 IC カードのセキュリティ領域へのアクセス手順の概要は以下のとおりです。

1. SAM カードにコマンドを送信し、暗号化コマンドを生成します。
2. 非接触 IC カードに 1.で生成した暗号化コマンドを送信し、暗号化コマンドの応答情報を受信します。
3. SAM カードに 2.で受信した暗号化コマンドの応答情報を送信し、復号化します。



※ 非接触 IC カードへのコマンド送信と受信は、NFC ライブラリを使って行ないます。詳細は「NFC ライブラリマニュアル」を参照して下さい。
詳細なアクセス手順については、SAM カードの仕様により異なります。各 SAM カードの仕様書を参照してください。

SAM クラスライブラリは、SAM ライブラリを .NET Compact Framework アプリケーションから直接利用できるようにする、ラッパーライブラリです。

SAM ライブラリを使用することにより、機種を意識することなく、アプリケーションのソースコード互換性を高めることかできます。

SAM ライブラリでは、機種を問わず、すべての関数を用意し、アプリケーションから見た「仮想マシン」としての振る舞いを提供します。
SAM ライブラリの各関数は、アプリケーションからの要求に対して、対象のデバイス機能が制御できない場合は、「未サポートエラー」を返します。また搭載デバイスの機能差によって利用できないパラメータを設定した場合は、「パラメータエラー」を返します。

※ SAM ライブラリは、アプリケーションのソースコード互換性の向上を目的としたライブラリであり、搭載デバイスの機能互換性を保障するものではありません。
「未サポートエラー」および「パラメータエラー」を正しく判定し、操作者に対して機能が未サポートである旨を通知する、あるいは処理そのものを無効としてください。

2. 動作環境

SAM ライブラリの動作環境を以下に示します。

対象機種

- IT-9000

対象 OS

- Microsoft WindowsCE 6.0
- Microsoft WindowsMobile 6.5

開発環境とプログラミング言語

開発環境	Visual C++	Visual Basic, Visual C#
Microsoft Visual Studio 2005 + SP1	○	○
Microsoft Visual Studio 2008 + SP1	○	○

提供ファイル

ファイル	Visual C++	Visual Basic, Visual C#
SAMLib.h	○	-
SAMLib.lib	○	-
SAMLib.dll	○	○
SAMLibNet.dll （クラスライブラリ）	-	○

（○：必要、-：不要）

使用方法

Visual C++の場合

- プログラムソース内に SAMLib.h をインクルードし、リンカの依存ファイルとして SAMLib.lib を指定してください。
- SAMLib.dll は本体に内蔵されています。

Visual Basic または Visual C#の場合

- SAMLibNet.dll をプロジェクトの参照に追加してください。
- SAMLib.dll は本体に内蔵されています。
- SAMLibNet.dll を実行モジュールと同じフォルダにコピーしてください。

名前空間とクラス

クラスライブラリ SAMLibNet.dll では、関数および定数の参照用として、下記のクラスが用意されています。

名前空間	クラス名	内容
CaLib	SAMLibNet.Api	関数参照用クラス
	SAMLibNet.Def	定数参照用クラス

クラス定義の詳細については、Microsoft Visual Studio で SAMLibNet.dll を参照設定し、オブジェクトブラウザで確認してください。

3. 機能一覧

3.1 関数一覧

SAM ライブラリの提供する関数は、以下のとおりです。

関数名	機能	IT-9000
SAMOpen	カードコントローラを電源 ON	○
SAMClose	カードコントローラを電源 OFF	○
SAMPowerUpCard	指定したカードスロットの SAM カードを電源 ON	○
SAMPowerDownCard	指定したカードスロットの SAM カードを電源 OFF	○
SAMExchangeData	起動した SAM カードとの通信	○
SAMGetNumberOfSlot	SAM カードのスロット数を取得	○
SAMGetCardResponse	起動した SAM カードの ATR 情報を取得	○
SAMNegotiate	起動した SAM カードとの PPS	○

○ 関数サポート
－ 関数未サポート

※ 複数のカードスロットを搭載するモデルの場合、スロットの番号は左端から順にスロット 1、スロット 2、スロット 3 となります。

関数呼び出し手順

アプリケーション起動時
1. SAMOpen関数により、SAM カードコントローラの電源を ON にします。(注意 1)
2. SAMPowerUpCard関数により、SAM カードの電源を ON にします。(注意 2)

SAM カードとの通信時
1. SAMExchangeData関数により、SAM カードに APDU コマンドを送信し、通信を行ないます。

アプリケーション終了時
1. SAMPowerDownCard関数により、SAM カードの電源を OFF にします。
2. SAMClose関数により、SAM カードコントローラの電源を OFF にします。

注意
1. SAM カードの電源が OFF の間は、SAM カードコントローラはスタンバイモードとなるため、ほとんど電力を使用しません。
2. SAM カードの電源を ON にすると、SAM カードが電力を消費するようになります。電力消費量について、SAM カードの種類により異なるので、各 SAM カードの仕様書を参照してください。電力の使用を抑えたい場合は、SAM カードの電源を OFF にしておき、SAM カードと通信するときだけ電源を ON にしてください。

3.1.1 SAMOpen

カードコントローラの電源を ON にし、SAM ドライバを待機状態にします。

```
[C++]
int SAMOpen()
```

```
[Visual Basic]
Public Shared Function SAMOpen() As Int32
```

```
[C#]
public static Int32 SAMOpen()
```

解説

本関数は、カードコントローラの電源を ON にし、SAM ドライバを待機状態にします。

パラメータ

なし

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_PON	: オープン済み※
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

※ 本関数を同一プロセス内で 2 回以上呼び出した場合は正常終了を返します。また、他のプロセスでオープン済の状態で本関数を実行すると SAM_PON を返します。

3.1.2 SAMClose

カードコントローラの電源を OFF にし、SAM ドライバを通信禁止状態にします。

```
[C++]
int SAMClose()
```

```
[Visual Basic]
Public Shared Function SAMClose() As Int32
```

```
[C#]
public static Int32 SAMClose()
```

解説

本関数は、カードコントローラの電源を OFF にし、SAM ドライバを通信禁止状態にします。

パラメータ

なし

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

3.1.3 SAMPowerUpCard

指定したカードスロットの SAM カードの電源を ON にし、SAM カードを通信可能状態にします。

```
[C++]
int SAMPowerUpCard(
    DWORD  dwSlotNumber,
    DWORD  dwVoltage
)
```

```
[Visual Basic]
Public Shared Function SAMPowerUpCard( _
    ByVal dwSlotNumber As Int32, _
    ByVal dwVoltage    As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMPowerUpCard(
    Int32  dwSlotNumber,
    Int32  dwVoltage
)
```

解説

本関数は、指定したカードスロットの SAM カードの電源を ON にし、SAM カードを通信可能状態にします。

パラメータ

dwSlotNumber

通信対象とする SAM カードを挿入したカードスロット番号を指定します。

1～3	: WAN なしモデル
1～2	: WAN ありモデル

dwVoltage

挿入した SAM カードに入れる電源の電圧を指定します。

SAM_POWER_ISO	: 1.8V、3V、5V の順で電源 ON にし、最初に応答のあった電圧で電源 ON
SAM_POWER_18V	: 1.8V で電源 ON
SAM_POWER_3V	: 3V で電源 ON
SAM_POWER_5V	: 5V で電源 ON

※ RC-S251 を使用する場合は SAM_POWER_3V を指定してください

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_POF	: 未オープンエラー
SAM_PRM	: パラメータエラー

SAM_ERROR_NOCARD	: SAM カード未挿入エラー DeviceEmulator では発生しません
SAM_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません
SAM_ERROR_RESPONSE	: SAM カード異常応答発生エラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

3.1.4 SAMPowerDownCard

指定したカードスロットの SAM カードの電源を OFF にし、SAM カードを待機状態にします。

```
[C++]
int SAMPowerDownCard(
    DWORD  dwSlotNumber,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function SAMPowerDownCard( _
    ByVal dwSlotNumber As Int32, _
    ByVal dwReserved   As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMPowerDownCard(
    Int32  dwSlotNumber,
    Int32  dwReserved
)
```

解説

本関数は、指定したカードスロットの SAM カードの電源を OFF にし、SAM カードを待機状態にします。

パラメータ

dwSlotNumber

通信対象とする SAM カードを挿入したカードスロット番号を指定します。

1～3	: WAN なしモデル
1～2	: WAN ありモデル

dwReserved

現在のバージョンではこの引数を使用しません。0 を指定してください。

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_POF	: 未オープンエラー
SAM_PRM	: パラメータエラー
SAM_ERROR_NOCARD	: SAM カード未挿入エラー DeviceEmulator では発生しません
SAM_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません
SAM_ERROR_RESPONSE	: SAM カード異常応答発生エラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

3.1.5 SAMExchangeData

起動した SAM カードとのデータ通信を行います。

```
[C++]
int SAMExchangeData(
    DWORD  dwSlotNumber,
    BYTE   *pbySendData,
    DWORD  dwSendSize,
    BYTE   *pbyReceiveData,
    DWORD  *pdwActualSize,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function SAMExchangeData( _
    ByVal dwSlotNumber    As Int32, _
    ByVal pbySendData     As Byte(), _
    ByVal dwSendSize      As Int32, _
    ByVal pbyReceiveData  As Byte(), _
    ByRef pdwActualSize   As Int32, _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMExchangeData(
    Int32      dwSlotNumber,
    Byte[]     pbySendData,
    Int32      dwSendSize,
    Byte[]     pbyReceiveData,
    ref Int32  pdwActualSize,
    Int32      dwReserved
)
```

解説

本関数は、SAMPowerUpCard関数により起動したSAMカードに対し、APDU形式のコマンドを送信し、それに対する応答を受信します。
DeviceEmulator では、指定した SAM カードの起動確認のみを行います。

パラメータ

dwSlotNumber
通信対象とする SAM カードを挿入したカードスロット番号を指定します。
1～3　　：WAN なしモデル
1～2　　：WAN ありモデル

pbySendData
SAM カードに送信するコマンドおよびパラメータ（バイナリデータ）を指定します。
コマンドおよびパラメータの書式は、SAM カードにより異なります。
送信可能なデータの最大値は 506 バイトです。

dwSendSize
pbySendData に指定するバイナリデータのサイズを指定します。

pbyReceiveData
SAM カードからの応答情報（バイナリデータ）を取得します。
バッファサイズは 506 バイト以上確保してください。

pdwActualSize
pbyReceiveData が取得したバイナリデータのサイズを取得します。

dwReserved
現在のバージョンではこの引数を使用しません。0 を指定してください。

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_POF	: 未オープンエラー
SAM_PRM	: パラメータエラー
SAM_ERROR_NOCARD	: SAM カード未挿入エラー DeviceEmulator では発生しません
SAM_ERROR_TIMEOUT	: タイムアウトエラー DeviceEmulator では発生しません
SAM_ERROR_SUSPEND	: 本体 OFF 発生エラー DeviceEmulator では発生しません
SAM_NOT_ACTIVATION	: カード未起動エラー
SAM_ERROR_RESPONSE	: SAM カード異常応答発生エラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

補足

戻り値の SAM_OK は、SAM カードとの通信の正常終了を表すものであり、指定したコマンド自身の正常終了を表すものではありません。コマンド自身の結果については、応答情報に格納しているコマンドの実行結果を確認してください。

3.1.6 SAMGetNumberOfSlot

SAM カードのスロット数を取得します。

```
[C++]
int SAMGetNumberOfSlot(
    DWORD  *pdwNumber
)
```

```
[Visual Basic]
Public Shared Function SAMGetNumberOfSlot( _
    ByRef pdwNumber As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMGetNumberOfSlot(
    ref Int32  pdwNumber
)
```

解説

本関数は、SAM カードのスロット数を取得します。
DeviceEmulator では、常に 3 を取得します。

パラメータ

dwNumber
SAM カードのスロット数を取得します。

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

3.1.7 SAMGetCardResponse

起動した SAM カードの応答情報を取得します。

```
[C++]
int SAMGetCardResponse(
    DWORD  dwSlotNumber,
    BYTE   *pbyTargetData,
    DWORD  *pdwActualSize,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function SAMGetCardResponse( _
    ByVal dwSlotNumber   As Int32, _
    ByVal pbyTargetData  As Byte(), _
    ByRef pdwActualSize  As Int32, _
    ByVal dwReserved     As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMGetCardResponse(
    Int32      dwSlotNumber,
    Byte[]     pbyTargetData,
    ref Int32  pdwActualSize,
    Int32      dwReserved
)
```

解説

本関数は、SAMPowerUpCard関数により起動した SAM カードの応答情報を取得します。
応答情報は SAM カード起動時にドライバに記憶されているため、本関数はドライバ内の応答情報を取得します。
DeviceEmulator では、指定した SAM カードの起動確認のみを行います。

パラメータ

dwSlotNumber

通信対象とする SAM カードを挿入したカードスロット番号を指定します。

1～3	: WAN なしモデル
1～2	: WAN ありモデル

pbyTargetData

SAM カードからの ATR 応答情報（以下参照）を取得します。バッファサイズは 506 バイト以上確保してください。

[TS]	開始キャラクタ（必須）
[T0]	構成表示キャラクタ（必須） 上位 4bit :TD(1)～TA(1)の有無を符号化 下位 4bit :管理情報バイトの個数（0～15）
[TA(1)]	接続情報キャラクタ（任意） 共通、Fi / Di を符号化
[TB(1)]	共通、Ii / Pi を符号化
[TC(1)]	共通、N を符号化
[TD(1)]	TD(2)～TA(2)の有無と T を符号化
[TA(2)]	共通、Fi / Di を符号化
[TB(2)]	共通、Pi2 を符号化
[TC(2)]	固有
[TD(2)]	TD(2)～TA(2)の有無と T を符号化
[TA(3)]	T≠15 のとき伝送プロトコルパラメータ、T=15 のとき IC パラメータ
（省略）	
[T1] （省略） [TK]	管理情報キャラクタ（任意） （最大 15 キャラクタ）
[TCK]	検査キャラクタ TD(1)が存在しないとき TCK は存在してはならない

※ ATR 応答情報の詳細については ISO7816 の規格書を参照してください

pdwActualSize

SAM カードから取得した応答情報のサイズを取得します。

dwReserved

現在のバージョンではこの引数を使用しません。0 を指定してください。

戻り値

以下の値を返します。

SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_POF	: 未オープンエラー
SAM_PRM	: パラメータエラー
SAM_ERROR_NOCARD	: SAM カード未挿入エラー DeviceEmulator では発生しません
SAM_NOT_ACTIVATION	: SAM カード未起動エラー
FUNCTION_UNSUPPORT	: 関数未サポート

3.1.8 SAMNegotiate

起動した SAM カードに対して、PPS を実行します。

```
[C++]
int SAMNegotiate(
    DWORD  dwSlotNumber,
    BYTE   byPPS0,
    BYTE   byPPS1,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function SAMNegotiate( _
    ByVal dwSlotNumber  As Int32, _
    ByVal byPPS         As Byte, _
    ByVal byPPS1        As Byte, _
    ByVal dwReserved    As Int32 _
) As Int32
```

```
[C#]
public static Int32 SAMNegotiate(
    Int32  dwSlotNumber,
    Byte   byPPS0,
    Byte   byPPS1,
    Int32  dwReserved
)
```

解説

本関数は、SAMPowerUpCard関数により起動した SAM カードに対して、PPS を実行します。
DeviceEmulator では、指定した SAM カードの起動確認のみを行います。

パラメータ

dwSlotNumber

通信対象とする SAM カードを挿入したカードスロット番号を指定します。

1～3　　：WAN なしモデル
1～2　　：WAN ありモデル

byPPS0

伝送プロトコルを指定します。下記のフォーマットにしたがって 8bit の値を指定します。

Bit: 7～5	Bit: 4	Bit: 3～0
Reserved	0: PPS1 を無視 1: PPS1 を使用	伝送プロトコルを指定

byPPS1

クロックレートとビットレートを指定します。下記のフォーマットにしたがって 8bit の値を指定します。

Bit: 7～4	Bit: 3～0
クロックレートを指定	ビットレートを指定

dwReserved
現在のバージョンではこの引数を使用しません。0 を指定してください。

戻り値

以下の値を返します。

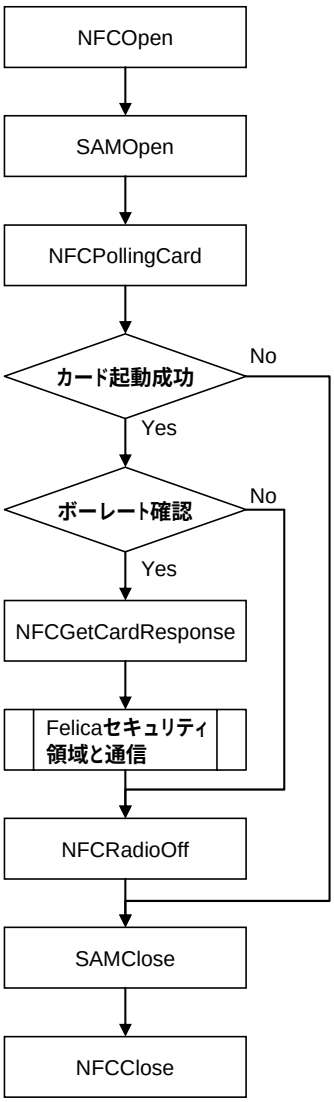
SAM_OK	: 正常終了
SAM_NOT_DEVICE	: SAM ドライバエラー DeviceEmulator では発生しません
SAM_POF	: 未オープンエラー
SAM_PRM	: パラメータエラー
SAM_ERROR_NOCARD	: SAM カード未挿入 DeviceEmulator では発生しません
SAM_NOT_ACTIVATION	: カード未起動エラー
SAM_ERROR_TIMEOUT	: タイムアウトエラー DeviceEmulator では発生しません
SAM_ERROR_SUSPEND	: 本体 OFF 発生エラー DeviceEmulator では発生しません
SAM_ERROR_RESPONSE	: SAM カード異常応答発生エラー DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 関数未サポート

4. プログラミング上の注意点

4.1 FeliCa セキュリティ領域との通信について

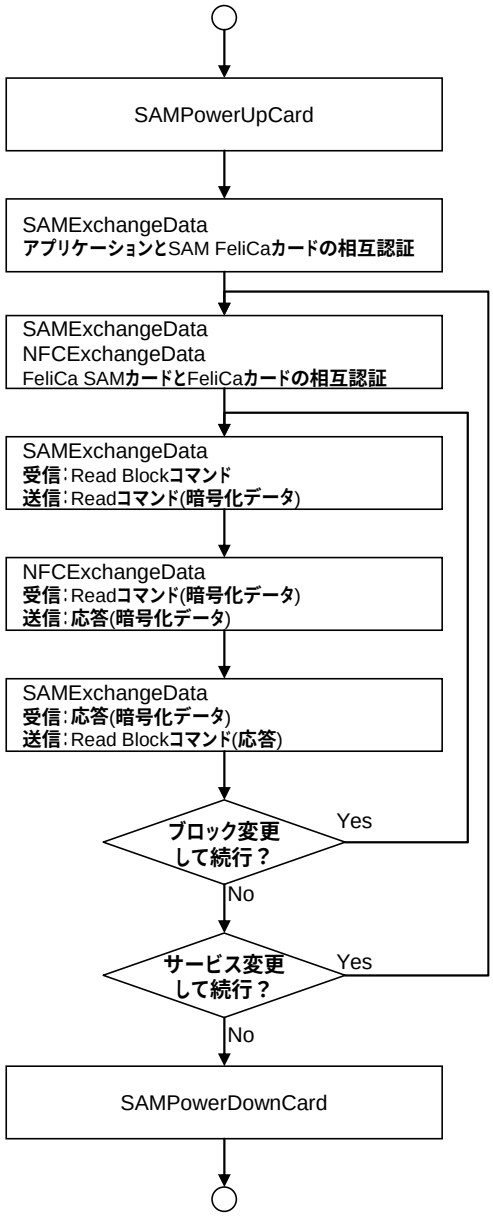
FeliCa セキュリティ領域と通信する場合 (全体)

1. NFCOpen 関数により IC カードと通信可能状態にします。
2. SAMOpen関数によりカードコントローラの電源を ON にします。
3. NFCPollingCard 関数により通信範囲内の IC カードを検索します。
4. IC カードの発見･起動に成功し、その IC カードのボーレートが FeliCa のボーレートと一致している場合は、NFCGetCardResponse 関数により FeliCa カードの UID を取得します。
5. FeliCa セキュリティ領域との通信を行います。(詳細は次ページ以降を参照してください)
6. NFCRadioOff 関数により電波を停止します。(電波を自動で停止する場合、本手順は不要です)
7. SAMClose関数によりカードコントローラの電源を OFF にします。
8. NFCClose 関数により IC カードと通信禁止状態にします。



FeliCa セキュリティ領域と通信する場合 (詳細)

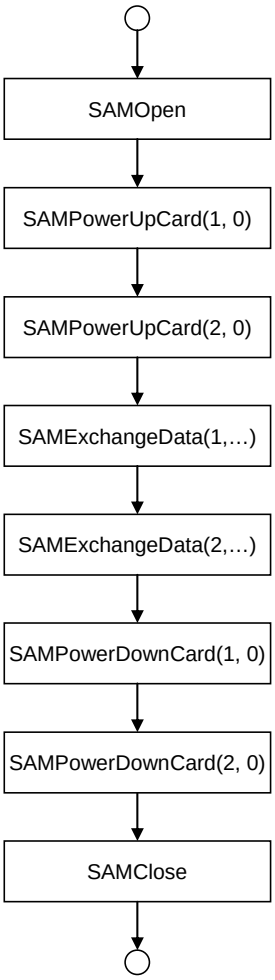
1. SAMPowerUpCard関数により SAM カードの電源を ON にし、通信可能状態にします。
2. SAMExchangeData関数により FeliCa SAM カードに対して通信を行い、アプリケーションと FeliCa SAM カードの相互認証を行います。
3. SAMExchangeData関数により FeliCa SAM カードに対して通信を行い、また、NFCExchangeData 関数により FeliCa カードと通信を行うことで、FeliCa SAM カードと FeliCa カードの相互認証を行います。
4. SAMExchangeData関数により、FeliCa SAM カードに対して、FeliCa Read Block コマンドを送信し、暗号化された Read コマンドを応答情報として取得します。
5. NFCExchangeCard 関数により、FeliCa SAM カードに対して、手順 4.で取得した暗号化された Read コマンドを送信すると、暗号化された応答情報を取得します。
6. SAMExchangeData関数により、FeliCa SAM カードに対して、手順 5.で取得した暗号化された応答情報を送信すると、手順 4.で送信した Read Block コマンドに対する応答情報を取得します。
7. Read/Write を行うブロックを変更し、処理を続行する場合は手順 4.に戻り、同様の処理を繰り返します。
8. Read/Write を行うサービスを変更し、処理を続行する場合は手順 3.に戻り、同様の処理を繰り返します。
9. SAMPowerDownCard関数により SAM カードの電源を OFF にし、待機状態にします。



4.2 SAM スロットを切り替えながらの通信について

複数の SAM を切り替えながら通信を行う場合

1. SAMOpen関数によりカードコントローラの電源を ON にします。
2. SAMPowerUpCard関数により、カードスロット 1 の SAM カードの電源を ON にし、通信可能状態にします。
3. SAMPowerUpCard関数により、カードスロット 2 の SAM カードの電源を ON にし、通信可能状態にします。
4. SAMExchangeData関数によりカードスロット 1 の SAM カードとの通信を行います。このとき、カードスロット 1 の SAM カードは手順 2.により通信可能状態になっているため、カードスロットを素早く切り替えてアクセスすることができます。
5. SAMExchangeData関数によりカードスロット 2 の SAM カードとの通信を行います。このとき、カードスロット 2 の SAM カードは手順 3.により通信可能状態になっているため、カードスロットを素早く切り替えてアクセスすることができます。
6. SAMPowerDownCard関数により、カードスロット 1 の SAM カードの電源を OFF にし、待機状態にします。
7. SAMPowerDownCard関数により、カードスロット 2 の SAM カードの電源を OFF にし、待機状態にします。
8. SAMClose関数によりカードコントローラの電源を OFF にします。



カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

http://casio.jp/support/ht/

製品の取扱い方法のお問い合わせ

●情報機器コールセンター

0570-022066
市内通話料金でご利用いただけます。
携帯電話・PHS 等をご利用の場合、**048-233-7241**

カシオ計算機株式会社
〒151-8543　東京都渋谷区本町 1-6-2
TEL 03-5334-4638（代）