CASIO

NFCMifare ライブラリマニュアル

このマニュアルは、NFCMifare ライブラリの
仕様について記載します。

Ver. 1.011

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	2009.07		開発推進版作成
1.00	2010.05		正版作成
1.01	2012.01		対象機種に DT-X8 および IT-9000 を追加
			関数を追加
	2012.02		プログラミング上の注意点のフローチャートを追加
			概要を修正
			関数呼び出し手順を追加

目次

1. 概要

NFC（Near Field Communication）Mifare ライブラリは、Mifare カードとの通信を行う関数を提供します。

ＮＦＣライブラリを使用して Mifare カードにアクセスする場合、業務アプリケーションは自分で Mifare コマンドを作成し、Mifare カードに送信する必要があります。NFCMifare ライブラリは、業務アプリケーションの代わりに Mifare コマンドの作成を行なうことで、Mifare カードへのアクセスをサポートします。

対象の IC カードが Mifare に限定される場合においては、NFC ライブラリを使用するよりも、NFCMifare ライブラリを使用する方が効率的にアプリケーションを開発することができます。

NFCMifare 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。

NFCMifare 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。
NFCMifare 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。

※ NFCMifare 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

NFCMifare ライブラリの動作環境を以下に示します。

対象機種

- DT-5300
- DT-X8
- 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#
NFCMifareLib.h	○	-
NFCMifareLib.lib	○	-
NFCMifareLib.dll	○	○
NFCMifareLibNet.dll	-	○

使用方法

Visual C++の場合

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

Visual Basic または Visual C#の場合

- NFCMifareLibNet.dll をプロジェクトの参照に追加してください。
- NFCMifareLib.dll は本体に内蔵されています。
- NFCMifareLibNet.dll を実行モジュールと同じフォルダー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NFCMifareLibNet.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	NFCMifareLibNet.Api	関数参照用クラス
	NFCMifareLibNet.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 NFCMifareLibNet.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. 関数

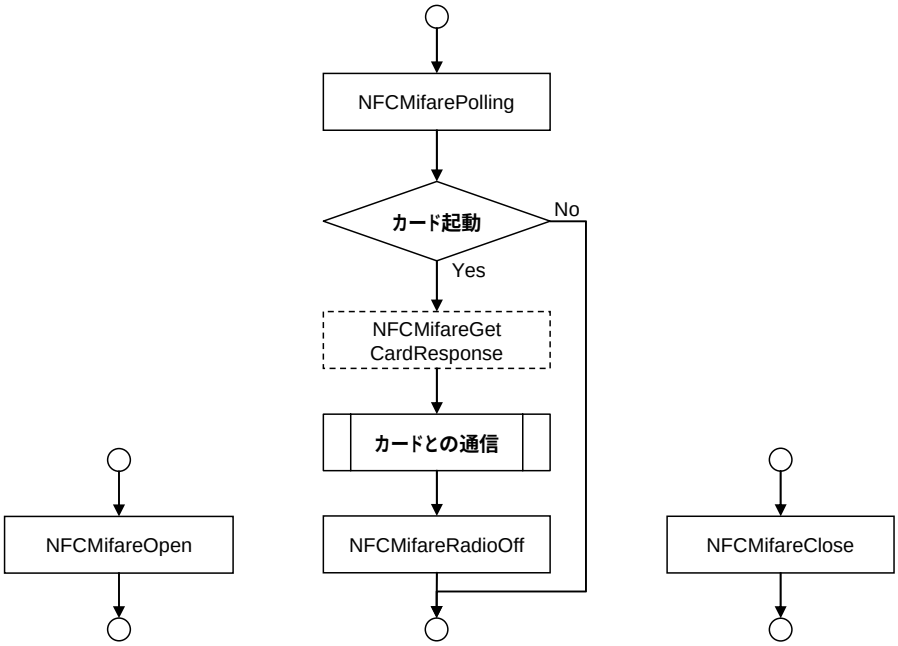
NFCMifare ライブラリの提供する関数は、以下のとおりです。

関数名	機能	DT-5300	DT-X8	IT-9000
NFCMifareOpen	通信許可状態の設定	○	○	○
NFCMifareClose	通信禁止状態の設定	○	○	○
NFCMifarePolling	通信範囲内の Mifare カードの検索および UID の取得	○	○	○
NFCMifareStopPolling	通信範囲内の Mifare カード検索を停止	○	○	○
NFCMifareGetCardResponse	起動した Mifare カードの詳細情報を取得	○	○	○
NFCMifareRadioOff	電波送信を停止	○	○	○
NFCMifareAuthentication	鍵のチェックによるアクセス許可の実行	○	○	○
NFCMifareRead	指定したアドレスのデータ読み出し（16 バイト）	○	○	○
NFCMifareWrite	指定したアドレスにデータ書き込み（16 バイト）	○	○	○
NFCMifareWrite4	指定したアドレスにデータ書き込み（4 バイト）	○	○	○
NFCMifareValue	指定したアドレスのデータの計算	○	○	○
NFCMifareSetEventNotification	電波自動停止のタイミング通知方法の設定	○	○	○
NFCMifareGetEventNotification	電波自動停止のタイミング通知方法の取得	○	○	○
NFCMifareSetAutoRadioOff	電波自動停止までの時間の設定	○	○	○
NFCMifareGetAutoRadioOff	電波自動停止までの時間の取得	○	○	○
NFCMifareSetPollingMode	IC カードの検索方法を設定	-	○	○
NFCMifareGetPollingMode	IC カードの検索方法を取得	-	○	○
NFCMifareGetCardResponseEx	起動した IC カードの詳細情報を取得	-	○	○

○ 関数サポート
－ 関数未サポート

■関数呼び出し手順

1. アプリケーション開始時に、NFCMifareOpen関数により、NFC デバイスの電源を ON にします。 (※1)
2. 通信処理開始時に、NFCMifarePolling関数により、通信可能範囲内にある Mifare カードを検索/起動します。
3. Mifare カードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCMifareGetCardResponse関数により、応答情報を取得します。(任意)
4. Mifare カードとの通信を行います。(次ページを参照)
5. Mifare カードとの通信が終了した場合は、NFCMifareRadioOff関数により、電波出力を停止します。
6. アプリケーション終了時に、NFCMifareClose関数により、NFC デバイスの電源を OFF にします。

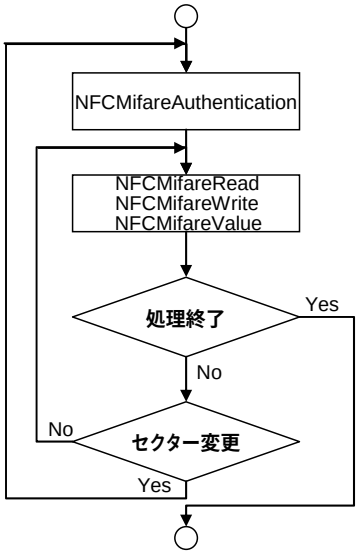


アプリケーション起動時　Mifare カードとの通信実行時　アプリケーション終了時

※1 Mifare カードと通信していないとき、NFC デバイス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。

■Mifare カードとの通信

1. NFCMifareAuthentication関数により、Mifare カードへのアクセス許可を行います。
2. NFCMifareRead関数、NFCMifareWrite関数、または、NFCMifareValue関数を実行します。
3. 処理を続行、かつ、アクセスするセクターを変更する場合は、1.に戻って処理を繰り返します。
4. 処理を続行、かつ、アクセスするセクターを変更しない場合は、2.に戻って処理を繰り返します。



3.1 NFCMifareOpen

NFC ドライバを通信許可状態（Open 状態）にし、NFC デバイスの電源を ON にします。

```
[C++]
int NFCMifareOpen(
    HWND  hWnd
)
```

```
[Visual Basic]
Public Shared Function NFCMifareOpen(
    ByVal hWnd As IntPtr _
) As Int32
```

```
[C#]
public static Int32 NFCMifareOpen(
    IntPtr hWnd
)
```

解説

本関数は、NFC ドライバを通信許可状態（Open 状態）にし、NFC デバイスの電源を ON にします。
この状態はNFCMifareClose関数を実行するまで有効です。
Open 状態時に、NFCMifarePolling関数を実行すると、通信を開始します。

パラメータ

hWnd
アプリケーションのウィンドウハンドルを指定します。
電波自動停止が有効、かつ、イベント通知方法がメッセージの場合、指定したウィンドウハンドルに対して、メッセージを送信します。
NULL を指定した場合は、BROADCAST に対してメッセージを送信します。

戻り値

以下の値を返します。

NFC_OK	：正常終了
NFC_PON	：オープン済み※
NFC_NOT_DEVICE	：NFC ドライバエラー DeviceEmulator では発生しません
NFC_ERROR_MODULE	：モジュール未応答エラー DeviceEmulator では発生しません

※ 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

3.2 NFCMifareClose

NFC ドライバを通信禁止状態（Close 状態）にし、NFC デバイスの電源を OFF にします。

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

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

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

解説

本関数は、NFC ドライバを通信禁止状態（Close 状態）にし、NFC デバイスの電源を OFF にします。

パラメータ

なし

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません

3.3 NFCMifarePolling

通信可能範囲内にある Mifare カードを検索します。

```
[C++]
int NFCMifarePolling(
    DWORD  dwTimeout,
    BOOL   (*fpCallBack)(void),
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifarePolling( _
    ByVal dwTimeout     As Int32, _
    ByVal fpCallBack    As IntPtr, _
    ByVal dwReserved    As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifarePolling(
    Int32      dwTimeout,
    IntPtr     fpCallBack,
    Int32      dwReserved
)
```

解説

本関数は、通信可能範囲内にある Mifare カードを検索します。
Mifare カードを発見した場合は、その Mifare カードを起動し、データ通信可能な状態にします。
本関数は Mifare カードを発見する、指定したタイムアウト時間経過する、または、指定したコールバック関数が FALSE を返すまで、通信範囲内の IC カードを検索します。
DeviceEmulator ではパラメータチェックのみを行います。

パラメータ

dwTimeout
Mifare カードが起動するまでのタイムアウト時間を 100～60,000（msec 単位）の範囲で指定します。
また、0 を指定した場合は、タイムアウトなしで Mifare カードを検索します。

fpCallBack
Mifare カードの検索を続行するかどうかを判定するコールバック関数を指定します。
コールバック関数が TRUE を返す場合は処理を続行し、FALSE を返す場合は処理を停止します。
また、NULL を指定した場合は、常に続行します。

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。

NFC_OK	：正常終了
NFC_NOT_DEVICE	：NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	：未オープンエラー
NFC_PRM	：パラメータエラー
NFC_ERROR_TIMEOUT	：タイムアウトエラー DeviceEmulator では発生しません
NFC_ERROR_CALLBACK	：コールバック関数エラー DeviceEmulator では発生しません
NFC_ERROR_MODULE	：モジュール未応答エラー DeviceEmulator では発生しません
NFC_ERROR_STOP	：停止関数による中断エラー DeviceEmulator では発生しません

3.4 NFCMifareStopPolling

通信可能範囲内にある Mifare カードの検索を停止します。

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

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

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

解説

本関数は、通信可能範囲内にある IC カードの検索を停止します。
コールバック関数を指定しないでNFCMifarePolling関数を実行した場合は、本関数を実行することにより検索を停止することができます。

パラメータ

なし

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー

3.5 NFCMifareGetCardResponse

起動した Mifare カードの応答情報を取得します。

```
[C++]
int NFCMifareGetCardResponse(
    BYTE   *pATQ,
    BYTE   *pSAK,
    BYTE   *pUid,
    BYTE   *pUidLen,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareGetCardResponse( _
     ByVal pATQ         As Byte(), _
     ByVal pSAK         As Byte(), _
     ByVal pUid         As Byte(), _
     ByVal pUidLen      As Byte(), _
     ByVal dwReserved   As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareGetCardResponse(
     Byte[]     pATQ,
     Byte[]     pSAK,
     Byte[]     pUid,
     Byte[]     pUidLen,
     Int32      dwReserved
)
```

解説

NFCMifarePolling関数成功後に本関数を実行すると、起動した Mifare カードの応答情報を取得します。
応答情報は Mifare カード起動成功時にドライバに記憶し、本関数によりドライバにある応答情報を取得します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

pATQ
起動に成功した Mifare カードからの ATQ を取得します。
2 バイト領域のポインタを指定してください。

pSAK
起動に成功した Mifare カードからの SAK を取得します。
1 バイト領域のポインタを指定してください。

pUid
起動に成功した Mifare カードの Uid を取得します。

7 バイト領域のポインタを指定してください。

pUidLen

起動に成功した Mifare カードの Uid の長さを取得します。
Mifare Standard の場合は 4 を、Mifare Ultralight の場合は 7 を取得します。
1 バイト領域のポインタを指定してください。

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

3.6 NFCMifareRadioOff

NFC モジュールの電波送信を停止します。

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

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

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

解説

本関数は、NFC モジュールの電波送信を停止します。

パラメータ

なし

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません

3.7 NFCMifareAuthentication

鍵のチェックによるアクセス許可を実行します。

```
[C++]
int NFCMifareAuthentication(
    DWORD  dwMode,
    BYTE   *pKey,
    DWORD  dwBlockNumber,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareAuthentication( _
     ByVal dwMode        As Int32, _
     ByVal pKey          As Byte(), _
     ByVal dwBlockNumber As Int32, _
     ByVal dwReserved    As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareAuthentication(
     Int32      dwMode,
     Byte[]     pKey,
     Int32      dwBlockNumber,
     Int32      dwReserved
)
```

解説

本関数は、起動した Mifare カード（Standard）に対して、指定した鍵情報と指定したブロック（Sector Trailer）に記録している鍵情報とを比較し、一致した場合はこのブロックが属するセクターへのアクセスを許可します。
また、他のセクターへアクセスする場合は、セクターごとに鍵のチェックを行う必要があります。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwMode
指定したブロックのどの鍵と比較するかを指定してください。
NFC_MIFARE_KEYA　：鍵 A と比較
NFC_MIFARE_KEYB　：鍵 B と比較

pKey
鍵を指定します。（6 バイト）

dwBlockNumber
鍵を格納しているブロック番号を指定します。
Standard1k　：0～63
Standard4k　：0～255

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー
NFC_ERROR_TIMEOUT	: タイムアウトエラー DeviceEmulator では発生しません
NFC_NOT_ACTIVATION	: カード未起動エラー DeviceEmulator では発生しません
NFC_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません
NFC_ERROR_SUSPEND	: 本体 OFF 発生エラー DeviceEmulator では発生しません
NFC_ERROR_AUTOOFF	: 電波自動停止エラー DeviceEmulator では発生しません

補足

■ Mifare Standard のメモリ構成
Standard1k の場合、64 セクターがあり、1 セクターは 4 ブロック（1 ブロック：16 バイト）です。
4 ブロックのうち、1 ブロック目が Sector Trailer となり、ここに鍵を記憶しています。
また、残りの 3 ブロックは Data Block となり、データ用です。

3.8 NFCMifareRead

起動した Mifare カード（Standard、Ultralight）のデータを読み出します。

```
[C++]
int NFCMifareRead(
    DWORD  dwBlockNumber,
    BYTE   *pData,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareRead( _
    ByVal dwBlockNumber   As Int32, _
    ByVal pData           As Byte(), _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareRead(
    Int32      dwBlockNumber,
    Byte[]     pData,
    Int32      dwReserved
)
```

解説

本関数は、起動した Mifare カード（Standard、Ultralight）のデータを読み出します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockNumber
データを読み出すアドレスを指定します。

Standard1k	: 0～63
Standard4k	: 0～255
Ultralight	: 0～15

pData
読み出したデータを取得します。
16 バイト領域のポインタを指定してください。

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません

NFC_POF : 未オープンエラー
NFC_PRM : パラメータエラー
NFC_ERROR_TIMEOUT : タイムアウトエラー
DeviceEmulator では発生しません
NFC_NOT_ACTIVATION : カード未起動エラー
DeviceEmulator では発生しません
NFC_ERROR_MODULE : モジュール未応答エラー
DeviceEmulator では発生しません
NFC_ERROR_SUSPEND : 本体 OFF 発生エラー
DeviceEmulator では発生しません
NFC_ERROR_AUTOOFF : 電波自動停止エラー
DeviceEmulator では発生しません

3.9 NFCMifareWrite

起動した Mifare カード(Standard) にデータを書き込みます。

```
[C++]
int NFCMifareWrite(
    DWORD  dwBlockNumber,
    BYTE   *pData,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareWrite( _
    ByVal dwBlockNumber   As Int32, _
    ByVal pData           As Byte(), _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareWrite(
    Int32      dwBlockNumber,
    Byte[]     pData,
    Int32      dwReserved
)
```

解説

本関数は、起動した Mifare カード(Standard) にデータを書き込みます。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockNumber
データを書き込むアドレスを指定します。

Standard1k	: 0～63
Standard4k	: 0～255

pData
書き込むデータを指定します。
16 バイト領域のポインタを指定してください。

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー

NFC_PRM : パラメータエラー
NFC_ERROR_TIMEOUT : タイムアウトエラー
DeviceEmulator では発生しません
NFC_NOT_ACTIVATION : カード未起動エラー
DeviceEmulator では発生しません
NFC_ERROR_MODULE : モジュール未応答エラー
DeviceEmulator では発生しません
NFC_ERROR_SUSPEND : 本体 OFF 発生エラー
DeviceEmulator では発生しません
NFC_ERROR_AUTOOFF : 電波自動停止エラー
DeviceEmulator では発生しません

3.10 NFCMifareWrite4

起動した Mifare カード(Ultralight) にデータを書き込みます。

```
[C++]
int NFCMifareWrite4(
    DWORD  dwBlockNumber,
    BYTE   *pData,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareWrite4( _
    ByVal dwBlockNumber   As Int32, _
    ByVal pData           As Byte(), _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareWrite4(
    Int32      dwBlockNumber,
    Byte[]     pData,
    Int32      dwReserved
)
```

解説

本関数は、起動した Mifare カード(Ultralight) にデータを書き込みます。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockNumber
データを書き込むアドレスを指定します。
Ultralight : 0～15

pData
書き込むデータを指定します。
16 バイト領域のポインタを指定してください。

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

NFC_ERROR_TIMEOUT　：タイムアウトエラー
DeviceEmulator では発生しません
NFC_NOT_ACTIVATION　：カード未起動エラー
DeviceEmulator では発生しません
NFC_ERROR_MODULE　：モジュール未応答エラー
DeviceEmulator では発生しません
NFC_ERROR_SUSPEND　：本体 OFF 発生エラー
DeviceEmulator では発生しません
NFC_ERROR_AUTOOFF　：電波自動停止エラー
DeviceEmulator では発生しません

3.11 NFCMifareValue

起動した Mifare カード（Standard）のデータの計算（加算、減算、ブロックコピー）を実行します。

```
[C++]
int NFCMifareValue(
    DWORD  dwMode,
    DWORD  dwBlockNumber,
    BYTE   *pValue,
    DWORD  dwTransBlock,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareValue( _
    ByVal dwMode          As Int32, _
    ByVal dwBlockNumber   As Int32, _
    ByVal pValue          As Byte(), _
    ByVal dwTransBlock    As Int32 _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareValue(
    Int32       dwMode,
    Int32       dwBlockNumber,
    Byte[]      pData,
    Int32       dwTransBlock
    Int32       dwReserved
)
```

解説

本関数は、起動した Mifare カード（Standard）のデータを読み込み、指定した計算（加算、減算、ブロックコピー）を実行します。
また、計算結果を指定したアドレスに書き込みます。
本関数を使用して計算を実行する場合は、対象メモリ領域が Value Block である必要があります。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwMode
計算方法を指定します。

NFC_MIFARE_INCREMENT	：加算
NFC_MIFARE_DECREMENT	：減算
NFC_MIFARE_RESTORE	：ブロックコピー

dwBlockNumber
計算元データがあるアドレスを指定します。

Standard1k	：0～63
Standard4k	：0～255

pValue

計算に使用する値を指定します。
4 バイト領域のポインタを指定してください。

dwTransBlock

計算後データを書き込むアドレスを指定します。

Standard1k	: 0～63
Standard4k	: 0～255

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー
NFC_ERROR_TIMEOUT	: タイムアウトエラー DeviceEmulator では発生しません
NFC_NOT_ACTIVATION	: カード未起動エラー DeviceEmulator では発生しません
NFC_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません
NFC_ERROR_SUSPEND	: 本体 OFF 発生エラー DeviceEmulator では発生しません
NFC_ERROR_AUTOOFF	: 電波自動停止エラー DeviceEmulator では発生しません

3.12 NFCMifareSetEventNotification

電波自動停止のタイミング通知方法を設定します。

```
[C++]
int NFCMifareSetEventNotification(
    DWORD dwMode
)
```

```
[Visual Basic]
Public Shared Function NFCMifareSetEventNotification(
    ByVal dwMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareSetEventNotification(
    Int32 dwMode
)
```

解説

本関数は、電波自動停止のタイミング通知方法を設定します。

■ ウィンドウメッセージ通知
WM_NFC_AUTORADIOOFF(WM_USER + 0x580)のウィンドウメッセージを指定したウィンドウハンドルに対して送信します。

■ イベント通知
電波自動停止時に発行されるイベントは“NFCEventAutoRadioOff”です。WindowsCE では、名前は Unicode のため、プログラム上では TEXT(“NFCEventAutoRadioOff”)と指定します。

パラメータ

dwMode
電波自動停止のタイミング通知方法を指定します。

NFC_DISABLE	: 通知無効（デフォルト）
NFC_MESSAGE	: ウィンドウメッセージ通知
NFC_EVENT	: イベント通知

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

3.13 NFCMifareGetEventNotification

電波自動停止のタイミング通知方法を取得します。

```
[C++]
int NFCMifareGetEventNotification(
    DWORD *pMode
)
```

```
[Visual Basic]
Public Shared Function NFCMifareGetEventNotification(
    ByRef pMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareGetEventNotification(
    ref Int32 pMode
)
```

解説

本関数は、電波自動停止のタイミング通知方法を取得します。

パラメータ

pMode

電波自動停止のタイミング通知方法を取得します。取得する値の詳細については、NFCMifareSetEventNotification関数を参照してください。

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

3.14 NFCMifareSetAutoRadioOff

電波自動停止までの時間を設定します。

```
[C++]
int NFCMifareSetAutoRadioOff(
    DWORD dwTimeout
)
```

```
[Visual Basic]
Public Shared Function NFCMifareSetAutoRadioOff(
    ByVal dwTimeout As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareSetAutoRadioOff(
    Int32 dwTimeout
)
```

解説

本関数は、電波自動停止までの時間を設定します。

パラメータ

Timeout

電波自動停止までの時間を 100～60,000（msec 単位）の範囲で指定します（デフォルト：1,000）。
また、0 を指定した場合は、電波自動停止が無効となります。

戻り値

以下の値を返します。

NFC_OK	：正常終了
NFC_NOT_DEVICE	：NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	：未オープンエラー
NFC_PRM	：パラメータエラー

3.15 NFCMifareGetAutoRadioOff

電波自動停止までの時間を取得します。

```
[C++]
int NFCMifareGetAutoRadioOff(
    DWORD *pTimeout
)
```

```
[Visual Basic]
Public Shared Function NFCMifareGetAutoRadioOff(
    ByRef pTimeout As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareGetAutoRadioOff(
    ref Int32 pTimeout
)
```

解説

本関数は、電波自動停止までの時間を取得します。

パラメータ

Timeout

電波自動停止までの時間を取得します。取得する値の詳細については、NFCMifareSetAutoRadioOff 関数を参照してください。

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

3.16 NFCMifareSetPollingMode

IC カードの検索方式を設定します。

```
[C++]
int NFCMifareSetPollingMode(
    DWORD  dwMode,
    DWORD  dwNum,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareSetPollingMode( _
    ByVal dwMode     As Int32, _
    ByVal dwNum      As Int32, _
    ByVal dwReserved As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareSetPollingMode(
    Int32  dwMode,
    Int32  dwNum,
    Int32  dwReserved
)
```

解説

本関数は、IC カードの検索方式を設定します。

パラメータ

dwMode

IC カードの検索方式を指定します。

NFC_PLMODE_NORMAL	: 通常起動（デフォルト）
NFC_PLMODE_MULTISTEP	: 多段起動
NFC_PLMODE_MULTISTEP2	: 多段起動 2
NFC_PLMODE_PACKAGE	: 一括起動

dwNum

多段起動時の段数を指定します。設定範囲は検索方式により異なります。

NFC_PLMODE_NORMAL 指定時	: 0 を指定してください
NFC_PLMODE_MULTISTEP 指定時	: 2～100
NFC_PLMODE_MULTISTEP2 指定時	: 2～100
NFC_PLMODE_PACKAGE 指定時	: 2～4

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。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

補足

■ IC カードの検索方式

通常起動	: 1 回の検索で 1 枚の IC カードを起動します
多段起動/ 多段起動 2	: 段数に指定した回数まで異なる IC カードを連続して起動します （1 回の検索で 1 枚しか起動することができません）※
一括起動	: 1 回の検索で同一タイプの複数枚の IC カードを起動します 段数に指定した回数まで検索を行います

※ 注意

IC カードを 1 つ起動するたびに、起動した IC カードの Uid をドライバに記録し、その記録した IC カードと重複する IC カードの二重起動を防止します。この記録は、指定した枚数の IC カードを起動したとき、タイムアウト時間を経過したとき、コールバック関数が FALSE を返したとき、およびNFCMifareStopPolling関数を実行したときにクリアします。

3.17 NFCMifareGetPollingMode

IC カードの検索方式を取得します。

```
[C++]
int NFCMifareGetPollingMode(
    DWORD  *pdwMode,
    DWORD  *pdwNum,
    DWORD  *pdwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareGetPollingMode( _
    ByRef pdwMode     As Int32, _
    ByRef pdwNum      As Int32, _
    ByRef pdwReserved As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareGetPollingMode(
    ref Int32  pdwMode,
    ref Int32  pdwNum,
    ref Int32  pdwReserved
)
```

解説

本関数は、IC カードの検索方式を取得します。

パラメータ

pdwMode

IC カードの検索方式を取得します。取得する値の詳細については、NFCMifareSetPollingMode関数を参照してください。

pdwNum

多段起動時の段数を取得します。取得する値の詳細については、NFCMifareSetPollingMode関数を参照してください。

pdwReserved

現在のバージョンではこの引数を使用しません。NULL を指定してください。

戻り値

以下の値を返します。

NFC_OK	: 正常終了
NFC_NOT_DEVICE	: NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	: 未オープンエラー
NFC_PRM	: パラメータエラー

3.18 NFCMifareGetCardResponseEx

起動した IC カードの応答情報を取得します。

```
[C++]
int NFCMifareGetCardResponseEx(
    BYTE   *pbyATQ,
    BYTE   *pbySAK,
    BYTE   *pbyUid,
    BYTE   *pbyUidLen,
    DWORD  *pdwDiscoveredNum,
    DWORD  *pdwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCMifareGetCardResponseEx( _
    ByVal pbyATQ           As Byte(), _
    ByVal pbySAK           As Byte(), _
    ByVal pbyUid           As Byte(), _
    ByRef pbyUidLen        As Byte, _
    ByRef pdwDiscoveredNum As Int32, _
    ByRef pdwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCMifareGetCardResponseEx(
    Byte[]     pbyATQ,
    Byte[]     pbySAK,
    Byte[]     pbyUid,
    ref Byte   pbyUidLen,
    ref Int32  pdwDiscoveredNum,
    ref Int32  pdwReserved
)
```

解説

NFCMifareSetPollingMode関数で一括起動モードに設定した状態で、NFCMifarePolling関数成功後に本関数を実行すると、起動した複数枚の IC カードの応答情報を取得します。
応答情報は IC カード起動成功時にドライバに記憶し、本関数によりドライバにある応答情報を取得します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

pbyATQ
起動に成功した IC カードの ATQA を取得します。
バッファサイズは（2×NFCMifareSetPollingMode関数の dwNum）以上確保してください。

2バイト	2バイト	...	2バイト
1枚目のATQA	2枚目のATQA	...	n枚目のATQA

pbySAK

起動に成功した IC カードの SAK を取得します。
バッファサイズは（1×NFCMifareSetPollingMode関数の dwNum）以上確保してください。

1バイト	1バイト	...	1バイト
1枚目のSAK	2枚目のSAK	...	n枚目のSAK

pbyUid

起動に成功した IC カードの Uid を取得します。
サイズは（8×NFCMifareSetPollingMode関数の dwNum）以上確保してください。

1バイト	aバイト	1バイト	bバイト	...	1バイト	xバイト
1枚目のUidの長さ(a)	1枚目のUid	2枚目のUidの長さ(b)	2枚目のUid	...	n枚目のUidの長さ(x)	n枚目のUid

pbyUid

pbyUid に取得したデータのサイズを取得します。

pdwDiscoveredNum

起動に成功した IC カードの枚数を取得します。
NFCMifareAuthentication関数の dwTargetNo に指定可能な値の最大は、（本パラメータで取得した値-1）となります。

例） 本パラメータで 3 を取得した場合
NFCMifareAuthentication関数の dwTargetNo に指定可能な値は 0～2 となります。

pdwReserved

現在のバージョンではこの引数を使用しません。NULL を指定してください。

戻り値

以下の値を返します。

NFC_OK	： 正常終了
NFC_NOT_DEVICE	： NFC ドライバエラー DeviceEmulator では発生しません
NFC_POF	： 未オープンエラー
NFC_PRM	： パラメータエラー

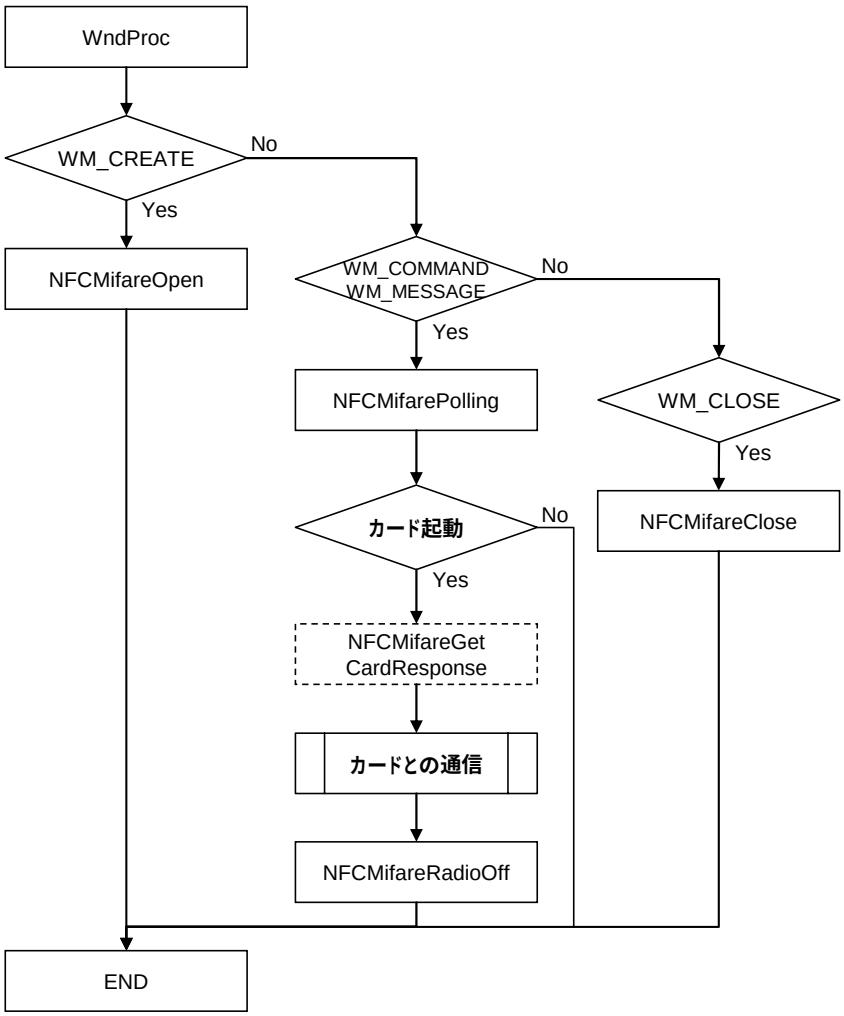
4. プログラミング上の注意点

4.1 電波停止の通知について

ウィンドウメッセージ通知を使用する場合

電波を手動で停止する場合

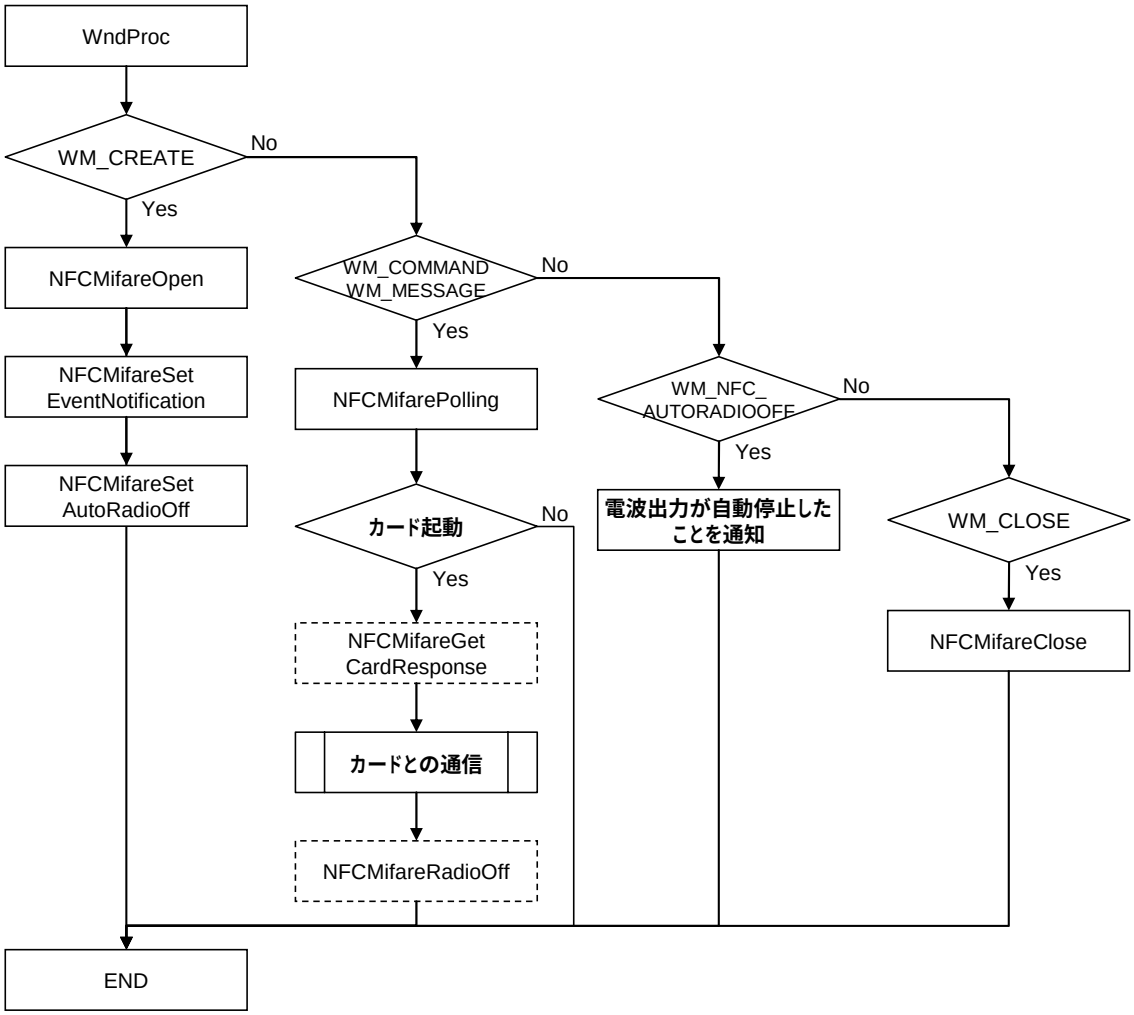
1. WM_CREATE メッセージを受け取った場合は、NFCMifareOpen関数を実行し、読み取り待機状態にします。
2. WM_COMMAND、WM_KEYDOWN 等のメッセージを受け取った場合は、NFCMifarePolling関数により、通信可能範囲内にある Mifare カードを検索/起動します。
3. Mifare カードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCMifareGetCardResponse関数により、応答情報を取得します。(任意)
4. Mifare カードとの通信を行います※。
5. Mifare カードとの通信が終了した場合は、NFCMifareRadioOff関数により、電波出力を停止します。
6. WM_CLOSE メッセージを受け取った場合は、NFCMifareClose関数により、読み取り禁止状態にします。



※ Mifare カードとの通信については、「Mifare カードとの通信について」を参照してください。

電波を自動で停止し、停止タイミングを通知する場合

1. WM_CREATE メッセージを受け取った場合は、NFCMifareOpen関数を実行し、読み取り待機状態にします。
2. NFCMifareSetEventNotification関数により、ウィンドウメッセージ通知を有効に設定します。
3. NFCMifareSetAutoRadioOff関数により、電波自動停止を有効に設定します。
4. WM_COMMAND、WM_KEYDOWN 等のメッセージを受け取った場合は、NFCMifarePolling関数により、通信可能範囲内にある Mifare カードを検索/起動します。
5. Mifare カードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCMifareGetCardResponse関数により、応答情報を取得します。(任意)
6. Mifare カードとの通信を行います※。
7. Mifare カードとの通信が終了した場合は、NFCMifareRadioOff関数により、電波出力を停止します。(停止しない場合でも、通信を一定時間行わないと自動的に電波出力を停止します)
8. 電波出力の自動停止が発生したタイミングで WM_NFC_AUTORADIOOFF(WM_USER + 0x580)メッセージを受け取ることができます。このとき、電波出力が自動停止したことをユーザに通知することが可能です。
9. WM_CLOSE メッセージを受け取った場合は、NFCMifareClose関数により、読み取り禁止状態にします。

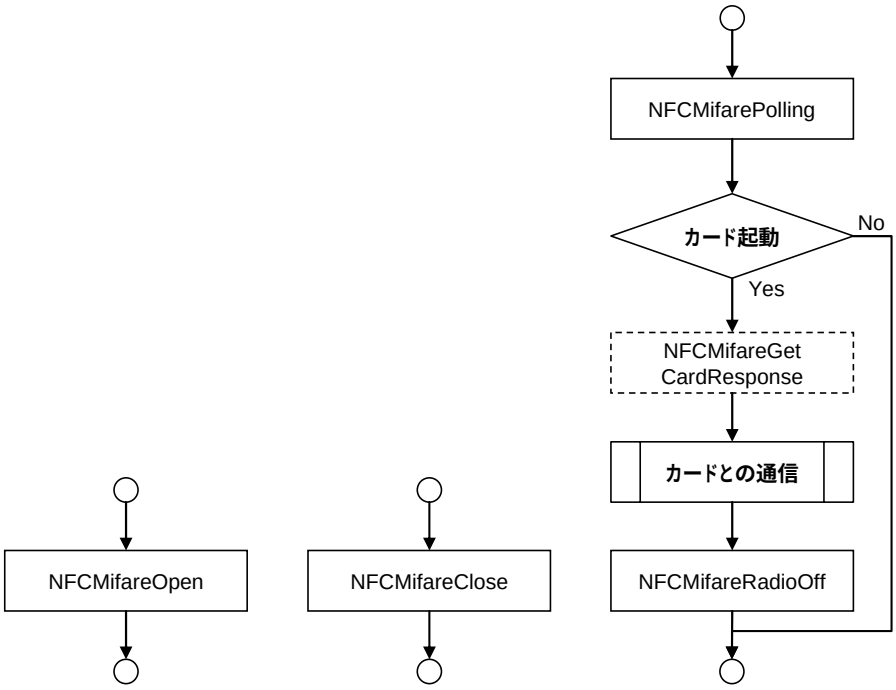


※ Mifare カードとの通信については、「Mifare カードとの通信について」を参照してください。

イベント通知を使用する場合

電波を手動で停止する場合

1. アプリケーション開始時に、NFCMifareOpen関数により、読み取り待機状態にします。
2. 通信処理開始時に、NFCMifarePolling関数により、通信可能範囲内にあるMifareカードを検索/起動します。
3. Mifareカードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCMifareGetCardResponse関数により、応答情報を取得します。(任意)
4. Mifareカードとの通信を行います※。
5. Mifareカードとの通信が終了した場合は、NFCMifareRadioOff関数により、電波出力を停止します。
6. アプリケーション終了時に、NFCMifareClose関数により、読み取り禁止状態にします。



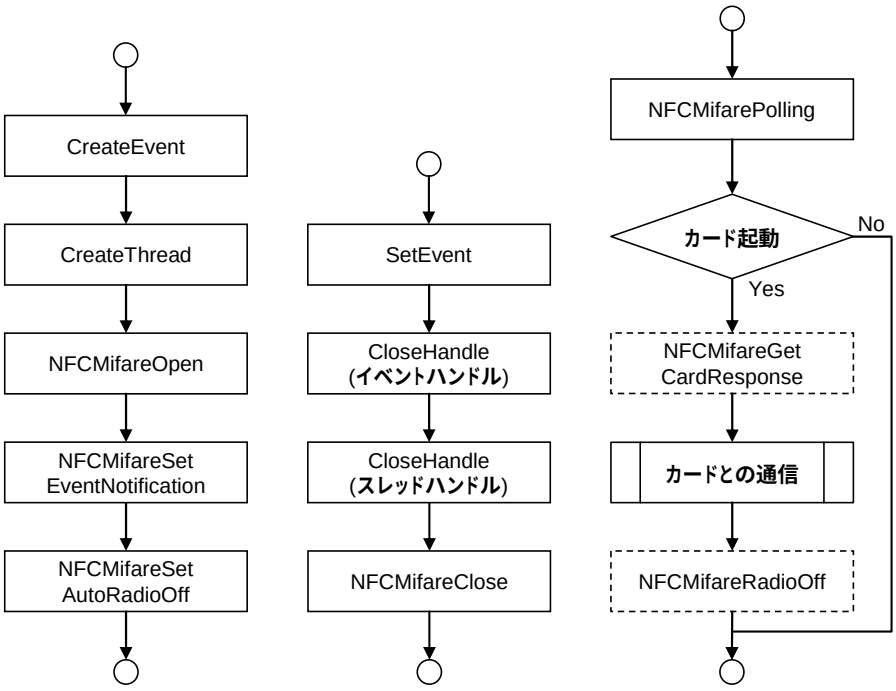
アプリケーション起動時　アプリケーション終了時　Mifareカードとの通信実行時

※ Mifareカードとの通信については、「Mifareカードとの通信について」を参照してください。

電波を自動で停止し、停止タイミングを通知する場合

■ メインスレッド

1. アプリケーション開始時に、CreateEvent 関数により、電波自動停止タイミング通知イベントハンドルを作成します。
2. CreateThread 関数により、電波自動停止を監視するスレッドを作成します。
3. NFCMifareOpen関数により、読み取り待機状態にします。
4. NFCMifareSetEventNotification関数により、イベント通知を有効に設定します。
5. NFCMifareSetAutoRadioOff関数により、電波自動停止を有効に設定します。
6. 通信処理開始時に、NFCMifarePolling関数により、通信可能範囲内にある Mifare カードを検索/起動します。
7. Mifare カードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCMifareGetCardResponse関数により、応答情報を取得します。(任意)
8. Mifare カードとの通信を行います※。
9. Mifare カードとの通信が終了した場合は、NFCMifareRadioOff関数により、電波出力を停止します。(停止しない場合でも、通信を一定時間行わないと自動的に電波出力を停止します)
10. アプリケーション終了時に、SetEvent 関数により、電波自動停止を監視するスレッドに対して通知を行います。
11. イベントハンドルとスレッドハンドルをクローズします。
12. NFCMifareClose関数により、読み取り禁止状態にします。

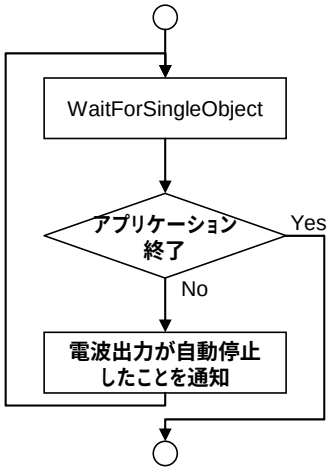


アプリケーション起動時　アプリケーション終了時　Mifare カードとの通信実行時

※ Mifare カードとの通信については、「Mifare カードとの通信について」を参照してください。

■ NFCMifare スレッド

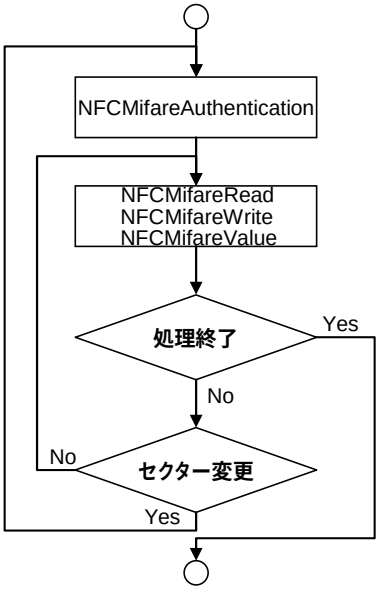
1. WaitForSingleObject 関数により、電波自動停止タイミング通知イベントハンドルに対して待機します。
2. アプリケーション終了時に通知イベントを受け取った場合、電波自動停止の監視を終了します。
3. 上記以外時に通知イベントを受け取った場合、電波出力が自動停止したことを通知することが可能です。



4.2 Mifare カードとの通信について

以下は「電波停止の通知について」におけるカードとの通信部分の手順です。

1. NFCMifareAuthentication関数により、Mifare カードへのアクセス許可を行います。
2. NFCMifareRead関数、NFCMifareWrite関数、または、NFCMifareValue関数を実行します。
3. 処理を続行、かつ、アクセスするセクターを変更する場合は、1.に戻って処理を繰り返します。
4. 処理を続行、かつ、アクセスするセクターを変更しない場合は、2.に戻って処理を繰り返します。

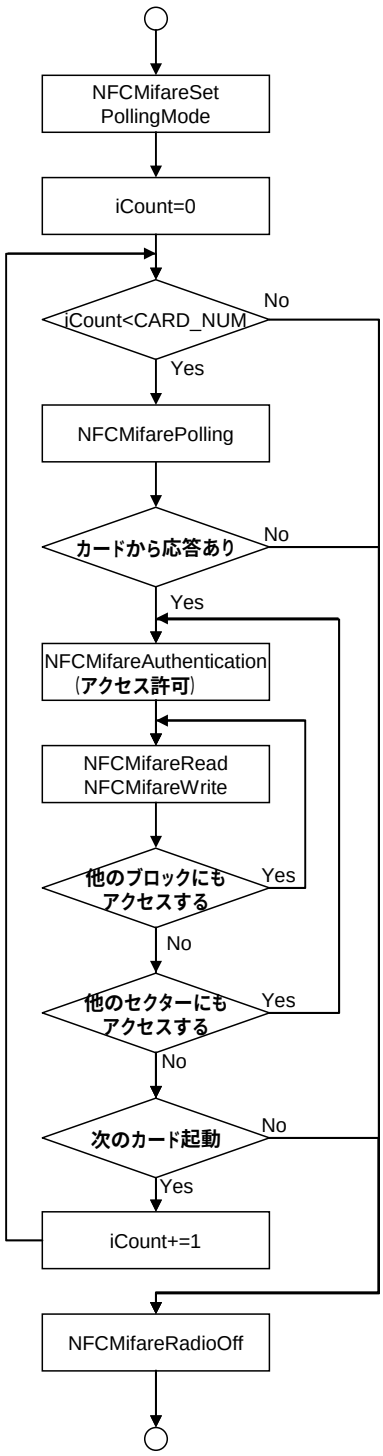


4.3 検索方式について

多段起動を使用する

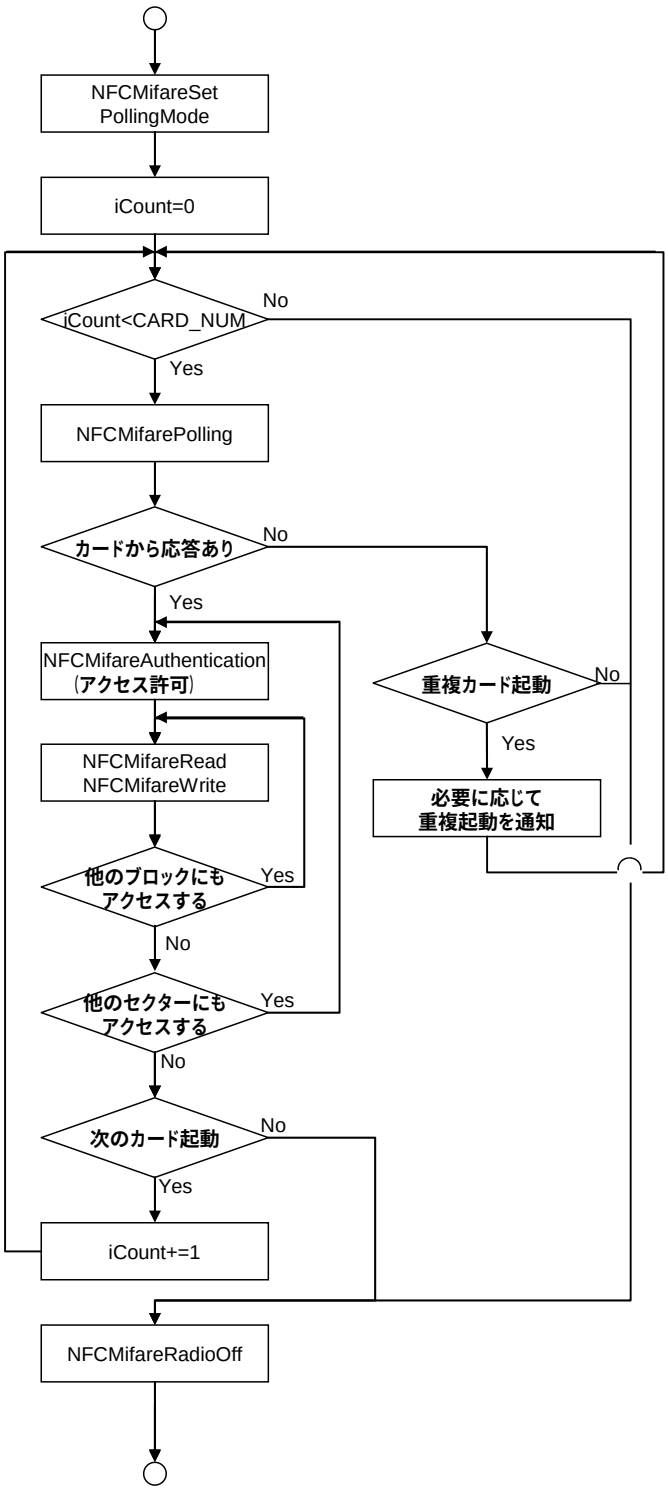
Mifare (Standard) カードと通信する

1. NFCMifareSetPollingMode関数により、検索方式に多段起動（NFC_PLMODE_MULTISTEP）を、段数に連続起動するカード枚数 CARD_NUM を指定します。
2. iCount=0 をセットします。
3. iCount<CARD_NUM の場合、次の処理に進みます。
   CARD_NUM は連続起動する IC カードの枚数を表します。
4. NFCMifarePolling関数により通信範囲内の IC カードを検索します。
5. IC カードが起動して応答があったら、次の処理へ進みます。
6. NFCMifareAuthentication関数により、Mifare コマンドを送信し、セクターへのアクセス許可を行います。
7. NFCMifareRead関数や、NFCMifareWrite関数などにより、カードとのデータアクセスを行います。（必要な動作に応じて各種関数を実行）
8. 他のブロックにもアクセスする場合、7.に戻って処理を繰り返します。
9. 他のセクターにもアクセスする場合、6.に戻って処理を繰り返します。
10. 次のカードを起動する場合、iCount に 1 を加算し、3.に戻って同様の処理を繰り返します。
11. 3.において、iCount が CARD_NUM より大きい場合、ループ処理を終了します。
12. NFCMifareRadioOff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は、本手順は必要ありません。）



多段起動 2 を使用する

Mifare (Standard) カードと通信する

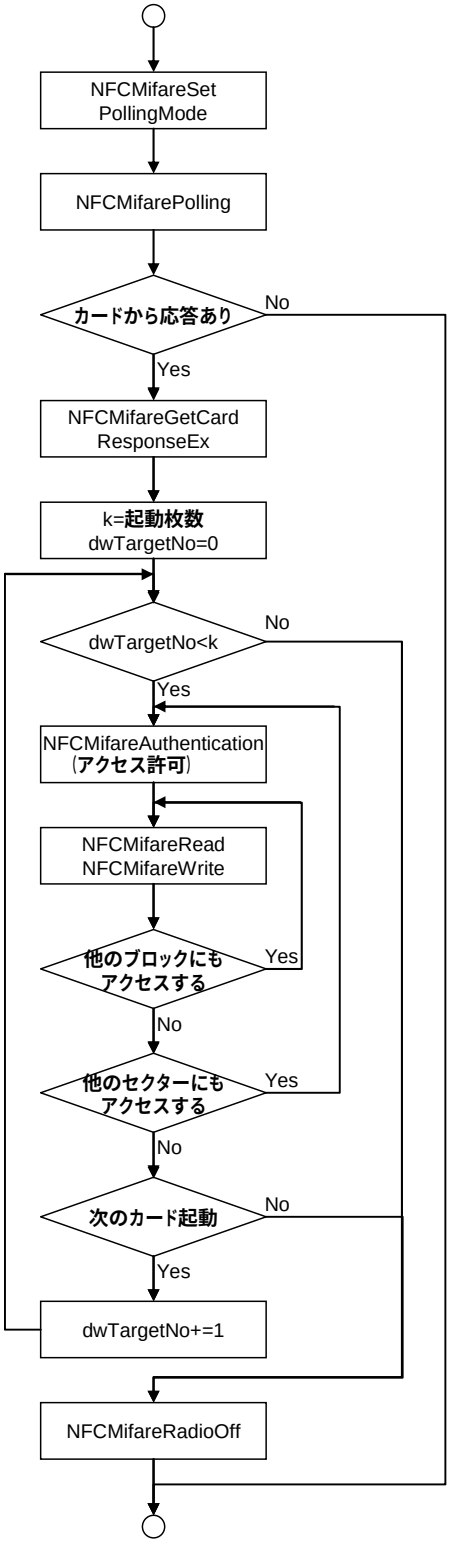


1. NFCMifareSetPollingMode関数により、検索方式に多段起動 (NFC_PLMODE_MULTISTEP2) を、段数に連続起動するカード枚数 CARD_NUM を指定します。
2. iCount=0 をセットします。
3. iCount<CARD_NUM の場合、次の処理に進みます。CARD_NUM は連続起動する IC カードの枚数を表します。
4. NFCMifarePolling関数により通信範囲内のカードを検索します。
5. カードの起動に失敗し、NFCMifarePolling関数の戻り値が重複起動を表す場合、必要に応じて LED 等により重複起動を通知します。その後、3.に戻って処理を繰り返します。
6. IC カードが起動して応答があったら、次の処理へ進みます。
7. NFCMifareAuthentication関数により、Mifare コマンドを送信し、セクターへのアクセス許可を行います。
8. NFCMifareRead関数や、NFCMifareWrite関数などにより、カードとのデータアクセスを行います。(必要な動作に応じて各種関数を実行)
9. 他のブロックにもアクセスする場合、8.に戻って処理を繰り返します。
10. 他のセクターにもアクセスする場合、7.に戻って処理を繰り返します。
11. 次のカードを起動する場合、iCount に 1 を加算し、3.に戻って同様の処理を繰り返します。
12. 3.において、iCount が CARD_NUM より大きい場合、ループ処理を終了します。
13. NFCMifareRadioOff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は、本手順は必要ありません。)

一括起動を使用する

Mifare (Standard) カードと通信する

1. NFCMifareSetPollingMode関数により、検索方式に一括起動 (NFC_PLMODE_PACKAGE) を、段数に一括起動する枚数を指定します。
2. NFCMifarePolling関数により通信範囲内のカードを検索します。
3. IC カードが起動して応答があったら、次の処理へ進みます。
4. k に起動した枚数を、dwTargetNo に 0 をセットします。(NFCMifareAuthentication関数等の引数)
5. dwTargetNo が k よりも小さい場合、次の処理に進みます。
6. NFCMifareAuthentication関数により、Mifare コマンドを送信し、セクターへのアクセス許可を行います。
7. NFCMifareRead関数やNFCMifareWrite関数などにより、カードとのデータアクセスを行います。(必要な動作に応じて各種関数を実行)
8. 他のブロックにもアクセスする場合、7.に戻って処理を繰り返します。
9. 他のセクターにもアクセスする場合、6.に戻って処理を繰り返します。
10. 次のカードと通信する場合、dwTargetNo に 1 加算し、5.に戻って同様の処理を繰り返します。
11. 5.において、dwTargetNo が k よりも大きい場合、ループ処理を終了します。
12. NFCMifareRadioOff関数により、電波を停止します。



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）