CASIO

NFCHFTag ライブラリマニュアル

このマニュアルは、NFCHFTag ライブラリの
仕様について記載します。

Ver. 1.02

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
1.00	2012.02	-	正版作成
1.01	2014.11	-	対象機種に IT-G500 を追加
		-	関数一覧表をデバイスライブラリ基本マニュアルへ移動
1.02	2015.02	-	対象機種に DT-X200 を追加

目次

1. 概要

NFC（Near Field Communication）HFTag ライブラリは、ISO15693 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 ISO15693 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 ISO15693 コマンドを作成し、ISO15693 カードに送信する必要があります。NFCHFTag ライブラリは、業務アプリケーションの代わりに ISO15693 コマンドの作成を行なうことで、ISO15693 カードへのアクセスをサポートします。

そのため、業務アプリケーションが ISO15693 コマンドを作成しなくても ISO15693 カードにアクセスできるようにするため、NFCHFTag ライブラリを提供します。

対象の IC カードが ISO15693 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、NFCHFTag 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。

NFCHFTag 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。

NFCHFTag 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。
NFCHFTag 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。

※ NFCHFTag 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

NFCHFTag ライブラリの動作環境を以下に示します。

対象機種

- DT-X8
- IT-9000
- IT-G500
- DT-X200

対象 OS

- Microsoft Windows CE 6.0
- Microsoft Windows Embedded Compact 7
- Microsoft Windows Mobile 6.5
- Microsoft Windows Embedded Handheld 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#
NFCHFTagLib.h	○	-
NFCHFTagLib.lib	○	-
NFCHFTagLib.dll	○	○
NFCHFTagLibNet.dll	-	○

使用方法

Visual C++の場合

- プログラムソース内に NFCHFTagLib.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 NFCHFTagLib.lib を指定してください。
- NFCHFTagLib.dll は本体に内蔵されています。

Visual Basic または Visual C#の場合

- NFCHFTagLibNet.dll をプロジェクトの参照に追加してください。
- NFCHFTagLib.dll は本体に内蔵されています。
- NFCHFTagLibNet.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 NFCHFTagLibNet.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	NFCHFTagLibNet.Api	関数参照用クラス
	NFCHFTagLibNet.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 NFCHFTagLibNet.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

HFTag ライブラリの下記 API について、ISO15693 タグの種類により対応状況が異なります。対応状況について、下記の表を参照してください。

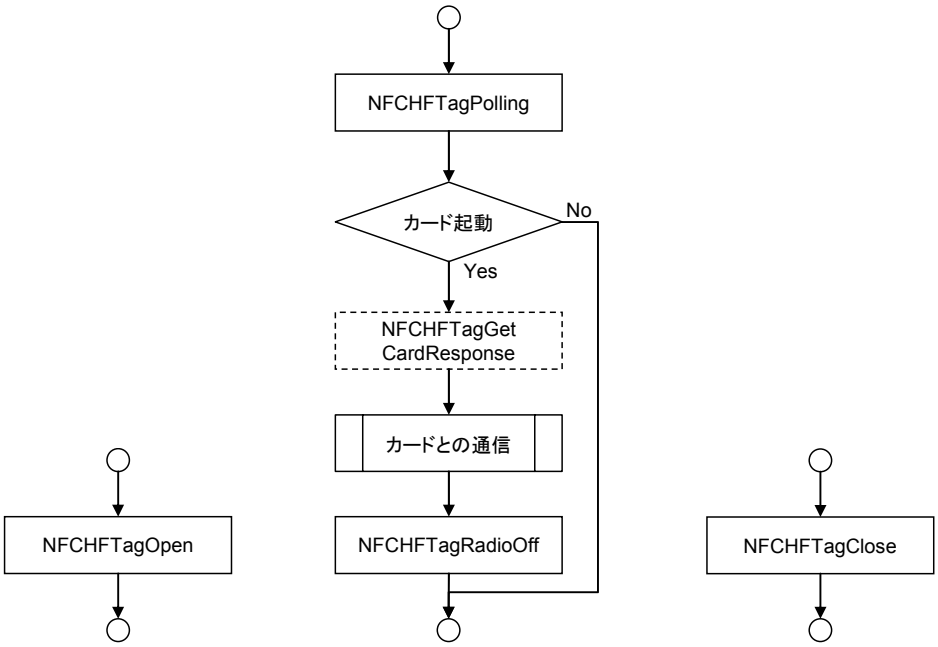
関数名	ICODE SLI	ICODE SLI-L	ICODE SLI-S	my-d V10 Plain	my-d Light	Tag-it HF-I Plus	Tag-it HF-I Pro	Tag-it HF-I Standard
NFCHFTagRead	○	○	○	○	○	○	○	○
NFCHFTagWrite	○	○	○	○	○	○ ※1	○ ※1	○ ※1
NFCHFTagLock	○	○	○	○	－	○ ※1	○ ※1	○ ※1
NFCHFTagReadMulti	○	－	－	○	－	○	－	－
NFCHFTagWriteAFI	○	○	○	○	○	○ ※1	－	－
NFCHFTagLockAFI	○	○	○	○	○	× ※2	－	－
NFCHFTagWriteDSFID	○	○	○	－	－	○ ※1	－	－
NFCHFTagLockDSFID	○	○	○	－	－	× ※2	－	－
NFCHFTagGetSystemInfo	○	○	○	－	－	○	－	－
NFCHFTagGetSecurityStatus	○	－	－	○	－	○	－	－

○：対応
－：カード仕様により未対応
×：NFC コントローラの仕様によりサポート対象外

※1 Tag-it™ シリーズについて、アプリケーションにおいて、コマンドのリトライ処理を行う必要があります。詳細について、「Tag-it™ カードとの通信について」を参照してください。
※2 Tag-it™ シリーズについて、NFC コントローラの仕様により、サポート対象外となります。

■関数呼び出し手順

1. アプリケーション開始時に、NFCHFTagOpen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、NFCHFTagPolling関数により、通信可能範囲内にある ISO15693 カードを検索/起動します。
3. ISO15693 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、NFCHFTagGetCardResponse関数により、応答情報を取得します。(任意)
4. ISO15693 カードとの通信を行います。(ISO15693カードとの通信を参照)
5. ISO15693 カードとの通信が終了した場合は、NFCHFTagRadioOff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、NFCHFTagClose関数により、NFC デバイスの電源を OFF にします。

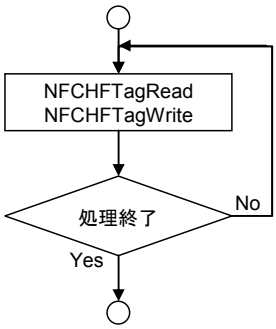


アプリケーション起動時　ISO15693 カードとの通信実行時　アプリケーション終了時

※1 ISO15693 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。

■ISO15693 カードとの通信

1. NFCHFTagRead関数または NFCHFTagWrite関数を実行します。



3.1 NFCHFTagOpen

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 NFCHFTagOpen(
    HWND  hWnd
)
```

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

```
[C#]
public static Int32 NFCHFTagOpen(
    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 NFCHFTagClose関数を実行するまで有効です。
Open 状態時に、NFCHFTagPolling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_INVALID_ACCESS	: デバイス排他エラー
NFC_ERROR_MODULE	: モジュール未応答エラー DeviceEmulator では発生しません

※ 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 NFCHFTagClose

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 NFCHFTagClose()
```

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

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

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 NFCHFTagPolling

通信可能範囲内にある ISO15693 カードを検索します。

```
[C++]
int NFCHFTagePolling(
    DWORD  dwTimeout,
    BOOL   (*fpCallBack)(void),
    BYTE   byAFI,
    DWORD  dwParam,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagPolling( _
    ByVal dwTimeout     As Int32, _
    ByVal fpCallBack    As IntPtr, _
    ByVal byAFI         As Byte, _
    ByVal dwParam       As Int32, _
    ByVal dwReserved    As Int32 _

) As Int32
```

```
[C#]
public static Int32 NFCHFTagPolling(
    Int32      dwTimeout,
    IntPtr     fpCallBack,
    Byte       byAFI,
    Int32      dwParam,
    Int32      dwReserved
)
```

解説

本関数は、通信可能範囲内にある ISO15693 カードを検索します。
ISO15693 カードを発見した場合は、その ISO15693 カードまたはタグを起動し、データ通信可能な状態にします。
本関数は ISO15693 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
ISO15693 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 ISO15693 カードを検索します。

fpCallBack
ISO15693 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。

byAFI

NFCPollingCard 関数により、起動を許可する ISO15693 カードの AFI を指定してください。0x00 を指定すると、全ての ISO15693 カードの起動を許可します。
(範囲:0x00～0xFF)

dwParam

IC カード検索を行う際の動作モードの指定。以下の機能を使用しない場合、0 を指定してください。

NFC_PL_SAVE : 電波の送信間隔を長めに調整した状態で、IC カードを検索することで、消費電力を抑えることができます。ただし、送信間隔が長くなるため、IC カードの検出レスポンスが低下します。長時間連続して IC カードの待ち受けを行う場合に使用してください。

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_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
NFC_ERROR_DUPLICATION	: 重複 IC カード起動

3.4 NFCHFTagStopPolling

通信可能範囲内にある ISO15693 カードの検索を停止します。

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

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

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

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 NFCHFTagPolling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 NFCHFTagGetCardResponse

起動した ISO15693 カードの応答情報を取得します。

```
[C++]
int NFCHFTagGetCardResponse(
    BYTE   *pDSFID,
    BYTE   *pUid,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagGetCardResponse( _
     ByVal pDSFID       As Byte(), _
     ByVal pUid         As Byte(), _
     ByVal dwReserved   As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagGetCardResponse(
     Byte[]     pDSFID,
     Byte[]     pUid,
     Int32      dwReserved
)
```

解説

NFCHFTagPolling関数成功後に本関数を実行すると、起動した ISO15693 カードの応答情報を取得します。
応答情報は ISO15693 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

pDSFID
関数成功時に、カードからの DSFID を格納します。
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 ISO15693 カードの Uid を取得します。
8 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 NFCHFTagRadioOff

NFC モジュールの電波送信を停止します。

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

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

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

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 NFCHFTagRead

起動した ISO15693 カードまたはタグに対し、指定されたブロック番号のデータを読み出します。

```
[C++]
int NFCHFTagRead(
    DWORD  dwBlockAddress,
    BYTE   *pData,
    DWORD  *pdwActualSize,
    DWORD  dwTargetNo,
    DWORD  dwOption
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagRead( _
    ByVal dwBlockAddress   As Int32, _
    ByVal pData            As Byte(), _
    ByVal pdwActualSize    As Int32(), _
    ByVal dwTargetNo       As Int32, _
    ByVal dwOption         As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagRead(
    Int32      dwBlockAddress,
    Byte[]     pData,
    Int32[]    pdwActualSize,
    Int32      dwTargetNo,
    Int32      dwOption
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定されたブロック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

dwBlockAddress

読み出し位置を指定するために、先頭のブロック番号を指定します(範囲:0～)。

pData

関数成功時に、読み出したブロックデータを格納します。。格納されるデータのサイズは、通信対象カードの 1 ブロックのサイズとなります。一般的なカードの場合、4 バイトのデータが格納されるため、4 バイトの領域を指定してください。また、dwOption 引数において、NFC_HFTAG_SECURITY を指定する場合、5 バイトのデータ領域を指定してください。

pdwActualSize

関数成功時に、読み出したデータサイズを格納します。。

dwTargetNo

通信したい IC カードに対応するカード番号を指定してください。。通常は 0 を指定してください。。NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

dwOption

ブロックのセキュリティ状態の取得の有無を指定してください。。

NFC_HFTAG_DEFAULT : 指定ブロックのセキュリティ状態の取得無効
NFC_HFTAG_SECURITY : 指定ブロックのセキュリティ状態の取得有効

NFC_HFTAG_SECURITY を指定した場合、関数成功時に pbyData に格納されるデータの書式は以下のとおりとなります。

1byte	4bytes
Status	ブロックデータ

Status は以下の値となります。

0x00 : ロックなし(書込み許可)
0x01 : ロックあり(書込み禁止)

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
NFC_ERROR_INVALID_ACCESS : カードポーリング中に実行エラー

3.8 NFCHFTagWrite

起動した ISO15693 カードまたはタグに対し、指定されたブロック番号のデータを書き込みます。

```
[C++]
int NFCHFTagWrite(
    DWORD  dwBlockAddress,
    BYTE   *pData,
    DWORD  dwWriteSize,
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagWrite( _
    ByVal dwBlockAddress   As Int32, _
    ByVal pData            As Byte(), _
    ByVal dwWriteSize      As Int32, _
    ByVal dwTargetNo       As Int32, _
    ByVal dwReserved       As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagWrite(
    Int32      dwBlockAddress,
    Byte[]     pData,
    Int32      dwWriteSize,
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定されたブロック坂東の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

dwBlockAddress
書き込み位置を指定するために、先頭のブロック番号を指定してください。(範囲:0～)

pData
書き込みを行うブロックデータを指定してください。。データサイズは 1 ブロック分のバイト数を指定する必要があります。※ 1 ブロックのサイズは、IC カードの仕様により異なります。一般的なカードの場合、4 バイトのデータとなります。

dwWriteSize
書き込みを行うデータサイズを指定してください。

dwTargetNo

通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.9 NFCHFTagLock

起動した ISO15693 カードまたはタグに対し、指定されたブロック番号の領域を書き込み不可にします。

```
[C++]
int NFCHFTagLock(
    DWORD  dwBlockAddress,
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagLock ( _
    ByVal dwBlockAddress   As Int32, _
    ByVal dwTargetNo       As Int32, _
    ByVal dwReserved       As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagLock (
    Int32      dwBlockAddress,
    Int32      dwTargetNo,
    Int32      dwReserved

)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定されたブロック番号の領域を書き込み不可にします。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockAddress
書き込み不可にする位置のブロック番号を指定してください(範囲:0～)

dwTargetNo
通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS : カードポーリング中に実行エラー

3.10 NFCHFTagReadMulti

起動した ISO15693 カードまたはタグに対し、指定されたブロック番号から、連続する複数ブロックのデータの一括読み出しを行います。

```
[C++]
int NFCHFTagReadMulti(
    DWORD  dwBlockAddress,
    DWORD  dwBlockNumber,
    BYTE   *pData,
    DWORD  dwActualSize,
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagReadMulti( _
    ByVal dwBlockAddress   As Int32, _
    ByVal dwBlockNumber    As Int32, _
    ByVal pData            As Byte(), _
    ByVal dwActualSize     As Int32, _
    ByVal dwTargetNo       As Int32, _
    ByVal dwReserved       As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagReadMulti(
    Int32      dwBlockAddress,
    Int32      dwBlockNumber,
    Byte[]     pData,
    Int32      dwActualSize,
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定されたブロック番号から、連続する複数ブロックのデータの一括読み出しを行います。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockAddress
読み出し位置を指定するために、先頭のブロック番号を指定してください(範囲:0～)

dwBlockNumber
一括読み出しを行うブロック数を指定してください(範囲:1～27)

pData
関数成功時に、読み出したブロックデータを格納します。。格納されるデータのサイズは、4 バイト(1 ブ

ロックサイズ)×dwBlockNumber となります。また、格納されるデータの最大値は
NFC_MAXSIZE_DATA(※1)となります。

dwActualSize
関数成功時に、読み出したデータのサイズを格納します。

dwTargetNo
通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.11 NFCHFTagWriteAFI

起動した ISO15693 カードまたはタグに対し、指定された AFI(Application Family Identifier)を書き込みます。

```
[C++]
int NFCHFTagWriteAFI(
    BYTE    byAFI,
    DWORD   dwTargetNo,
    DWORD   dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagWriteAFI( _
    ByVal byAFI           As Byte, _
    ByVal dwTargetNo      As Int32, _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagWriteAFI(
    Byte        byAFI,
    Int32       dwTargetNo,
    Int32       dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定された AFI(Application Family Identifier)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

byAFI
ISO15693 カードまたはタグに対し、書き込む AFI を指定してください
AFI のコード表

AFI の上位 4 ビット	AFI の下位 4 ビット	カードの応用分野	例/参考
"0"	"0"	全分野	分野を特定しない
X	"0"	X 分野	広範囲に選択
X	Y	X 分野の Y 小分類	
"0"	Y	Y 小分類に限定	
"1"	"0", Y	交通機関	大量輸送交通、バス、航空機
"2"	"0", Y	金融	銀行
"3"	"0", Y	認識	アクセスコントロール
"4"	"0", Y	電気通信	公衆電話、GSM
"5"	"0", Y	医療	
"6"	"0", Y	マルチメディア	インターネット

"7"	"0", Y	ゲーム	
"8"	"0", Y	データ記憶	可搬型ファイル
"9"	"0", Y	物流管理	
"A"	"0", Y	宅急便	
"B"	"0", Y	郵便	
"C"	"0", Y	航空手荷物	
"D"	"0", Y	RFU	
"E"	"0", Y	RFU	
"F"	"0", Y	RFU	

備考:X と Y は"1"～"F"の範囲で指定

※ カードが AFI を使用するかどうかは、任意となります。

dwTargetNo

通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.12 NFCHFTagLockAFI

起動した ISO15693 カードまたはタグに対し、AFI(Application Family Identifier)を書き込み不可にします。

```
[C++]
int NFCHFTagLockAFI(
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

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

```
[C#]
public static Int32 NFCHFTagLockAFI(
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、AFI(Application Family Identifier)を書き込み不可にします。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwTargetNo
通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS : カードポーリング中に実行エラー

3.13 NFCHFTagWriteDSFID

起動したISO15693カードまたはタグに対し、指定されたDSFID(Data Storage Format Identifier)を書き込みます。

```
[C++]
int NFCHFTagWriteDSFID(
    BYTE    byDSFID,
    DWORD   dwTargetNo,
    DWORD   dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagWriteDSFID( _
    ByVal byDSFID         As Byte, _
    ByVal dwTargetNo      As Int32, _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagWriteDSFID(
    Byte        byDSFID,
    Int32       dwTargetNo,
    Int32       dwReserved
)
```

解説

本関数は、起動したISO15693カードまたはタグに対し、指定されたDSFID(Data Storage Format Identifier)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

byDSFID
ISO15693カードまたはタグに対し、書き込むDSFIDを指定してください
DSFIDの詳細については、国際標準規格のISO15961を参照してください。

dwTargetNo
通信したいICカードに対応するカード番号を指定してください。通常は0を指定してください。
NFCPollingCard関数により複数ICカードの起動に成功した状態で、2枚目以降のICカードと通信する場合、1以上の値を指定します。詳細はNFCHFTagGetCardResponseEx関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS : カードポーリング中に実行エラー

3.14 NFCHFTagLockDSFID

起動した ISO15693 カードまたはタグに対し、DSFID(Data Storage Format Identifier)を書き込み不可にします。

```
[C++]
int NFCHFTagLockDSFID(
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

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

```
[C#]
public static Int32 NFCHFTagLockDSFID(
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、DSFID(Data Storage Format Identifier)を書き込み不可にします。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwTargetNo
通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS : カードポーリング中に実行エラー

3.15 NFCHFTagGetSystemInfo

起動した ISO15693 カードまたはタグに対し、カードのシステム情報を取得します。

```
[C++]
int NFCHFTagGetSystemInfo(
    BYTE   *pInfo,
    BYTE   *pUID,
    BYTE   *pDSFID,
    BYTE   *pAFI,
    BYTE   *pMemSize,
    BYTE   *pICRef,
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagGetSystemInfo( _
    ByVal pInfo          As Byte(), _
    ByVal pUID           As Byte(), _
    ByVal pDSFID         As Byte(), _
    ByVal pAFI           As Byte(), _
    ByVal pMemSize       As Byte(), _
    ByVal pICRef         As Byte(), _
    ByVal dwTargetNo     As Int32 _
    ByVal dwReserved     As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagGetSystemInfo(
    Byte[]     pInfo,
    Byte[]     pUID,
    Byte[]     pDSFID,
    Byte[]     pAFI,
    Byte[]     pMemSize,
    Byte[]     pICRef,
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、カードのシステム情報を取得します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

pInfo

関数成功時に、カードからの Information フラグを格納します。。1 バイト領域のポインタを指定してください。

■ Information フラグの定義

bit	フラグ名	値	内容
0	DSFID	0	DSFID はサポートされていない/存在しない
		1	DSFID はサポートされている/存在する
1	AFI	0	AFI はサポートされていない/存在しない
		1	AFI はサポートされている/存在する
2	メモリサイズ	0	メモリサイズ情報はサポートされていない/存在しない
		1	メモリサイズ情報はサポートされている/存在する
3	IC リファレンス	0	IC リファレンス情報はサポートされていない/存在しない
		1	IC リファレンス情報はサポートされている/存在する
4	RFU	0	将来拡張用のリザーブ
5	RFU	0	将来拡張用のリザーブ
6	RFU	0	将来拡張用のリザーブ
7	RFU	0	将来拡張用のリザーブ

pUID

関数成功時に、カードからの UID を格納します。。8 バイト領域のポインタを指定してください。

pDSFID

関数成功時に、カードからの DSFID を格納します。。1 バイト領域のポインタを指定してください。

pAFI

関数成功時に、カードからの AFI を格納します。。1 バイト領域のポインタを指定してください。

pMemSize

関数成功時に、カードからのメモリサイズを格納します。。2 バイト領域のポインタを指定してください。

■ メモリサイズ情報定義

MSB		LSB
16 14	13 9	8 1
RFU	ブロックサイズ(1 ブロック単位のバイト数)	ユーザ領域のブロック数

ICRef

関数成功時に、カードからの IC リファレンスを格納します。。1 バイト領域のポインタを指定してください。

dwTargetNo

通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.16 NFCHFTagGetSecurityStatus

起動した ISO15693 カードまたはタグに対し、指定されたブロック番号から指定された範囲に対応するセキュリティ情報を取得します。

```
[C++]
int NFCHFTagGetSecurityStatus(
    DWORD  dwBlockAddress,
    DWORD  dwBlockNumber,
    BYTE   *pStatus,
    DWORD  *pdwActualSize,
    DWORD  dwTargetNo,
    DWORD  dwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagGetSecurityStatus( _
    ByVal dwBlockAddress  As Int32, _
    ByVal dwBlockNumber   As Int32, _
    ByVal pStatus         As Byte(), _
    ByVal pdwActualSize   As Int32(), _
    ByVal dwTargetNo      As Int32, _
    ByVal dwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagGetSecurityStatus(
    Int32      dwBlockAddress,
    Int32      dwBlockNumber,
    Byte[]     pStatus,
    Int32[]    pdwActualSize,
    Int32      dwTargetNo,
    Int32      dwReserved
)
```

解説

本関数は、起動した ISO15693 カードまたはタグに対し、指定されたブロック番号から指定された範囲に対応するセキュリティ情報を取得します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBlockAddress
取得対象ブロック位置を指定するために、先頭のブロック番号を指定してください(範囲:0～)

dwBlockNumber
取得対象ブロック範囲を指定するために、ブロック数を指定してください(範囲:1～256)

pStatus
関数成功時に、カードからのセキュリティ情報を格納します。。dwBlockNumber で指定した値のサイズ

の領域のポインタを指定してください。
例)dwBlockNumber に 4 を指定した場合、4 バイト領域のポインタを指定します。

pdwActualSize
関数成功時に、カードから取得したセキュリティ情報のサイズを格納します。。

dwTargetNo
通信したい IC カードに対応するカード番号を指定してください。通常は 0 を指定してください。
NFCPollingCard 関数により複数 IC カードの起動に成功した状態で、2 枚目以降の IC カードと通信する場合、1 以上の値を指定します。詳細は NFCHFTagGetCardResponseEx 関数の説明を参照してください。

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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.17 NFCHFTagSetEventNotification

電波自動停止のタイミング通知方法を設定します。

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

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

```
[C#]
public static Int32 NFCHFTagSetEventNotification(
    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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.18 NFCHFTagGetEventNotification

電波自動停止のタイミング通知方法を取得します。

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

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

```
[C#]
public static Int32 NFCHFTagGetEventNotification(
    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、NFCHFTagSetEventNotification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

3.19 NFCHFTagSetAutoRadioOff

電波自動停止までの時間を設定します。

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

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

```
[C#]
public static Int32 NFCHFTagSetAutoRadioOff(
    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
NFC_ERROR_INVALID_ACCESS	: カードポーリング中に実行エラー

3.20 NFCHFTagGetAutoRadioOff

電波自動停止までの時間を取得します。

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

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

```
[C#]
public static Int32 NFCHFTagGetAutoRadioOff(
    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、
NFCHFTagSetAutoRadioOff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

3.21 NFCHFTagSetPollingMode

IC カードの検索方式を設定します。

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

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

```
[C#]
public static Int32 NFCHFTagSetPollingMode(
    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 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

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 NFCHFTagStopPolling関数を実行したときにクリアします。

3.22 NFCHFTagGetPollingMode

IC カードの検索方式を取得します。

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

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

```
[C#]
public static Int32 NFCGetPollingMode(
    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、NFCHFTagSetPollingMode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、NFCHFTagSetPollingMode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

3.23 NFCHFTagGetCardResponseEx

起動した IC カードの応答情報を取得します。

```
[C++]
int NFCHFTagGetCardResponseEx(
    BYTE   *pbyDSFID,
    BYTE   *pbyUid,
    DWORD  *pdwDiscoveredNum,
    DWORD  *pdwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCHFTagGetCardResponseEx( _
    ByVal pbyDSFID         As Byte(), _
    ByVal pbyUid           As Byte(), _
    ByRef pdwDiscoveredNum As Int32, _
    ByRef pdwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCHFTagGetCardResponseEx(
    Byte[]     pbyDSFID,
    Byte[]     pbyUid,
    ref Int32  pdwDiscoveredNum,
    ref Int32  pdwReserved
)
```

解説

NFCHFTagSetPollingMode関数で一括起動モードに設定した状態で、NFCHFTagPolling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

pbyDSFID
関数成功時に、カードからの DSFID を格納します。
バッファサイズは（1×NFCHFTagSetPollingMode関数の dwNum）以上確保してください。

データ書式

1バイト	1バイト
1枚目のDSFID	2枚目のDSFID

pbyUid

関数成功時に、カードからのUidを格納します。
バッファサイズは（8×NFCHFTagSetPollingMode関数の dwNum）以上確保してください。

データ書式

8バイト	8バイト
1枚目のUid	2枚目のUid

pdwDiscoveredNum

NFCHFTagPolling関数で起動に成功した IC カードの枚数を取得します。
NFCHFTagRead関数や NFCHFTagWrite関数などの 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 2 を取得した場合
NFCHFTagRead関数の dwTargetNo に指定可能な値は 0～1 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
NFC_NOT_ACTIVATION	：カード未起動エラー DeviceEmulator では発生しません
NFC_ERROR_INVALID_ACCESS	：電波自動停止タイマー動作中に実行エラー DeviceEmulator では発生しません

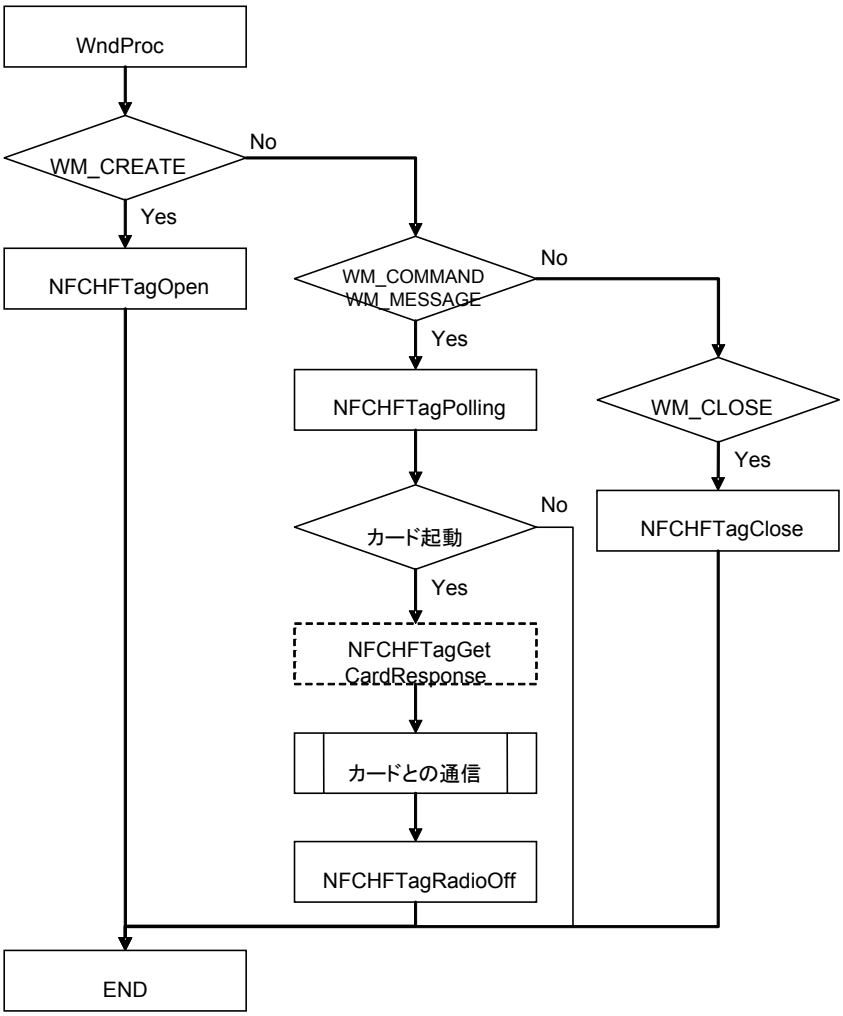
4. プログラミング上の注意点

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

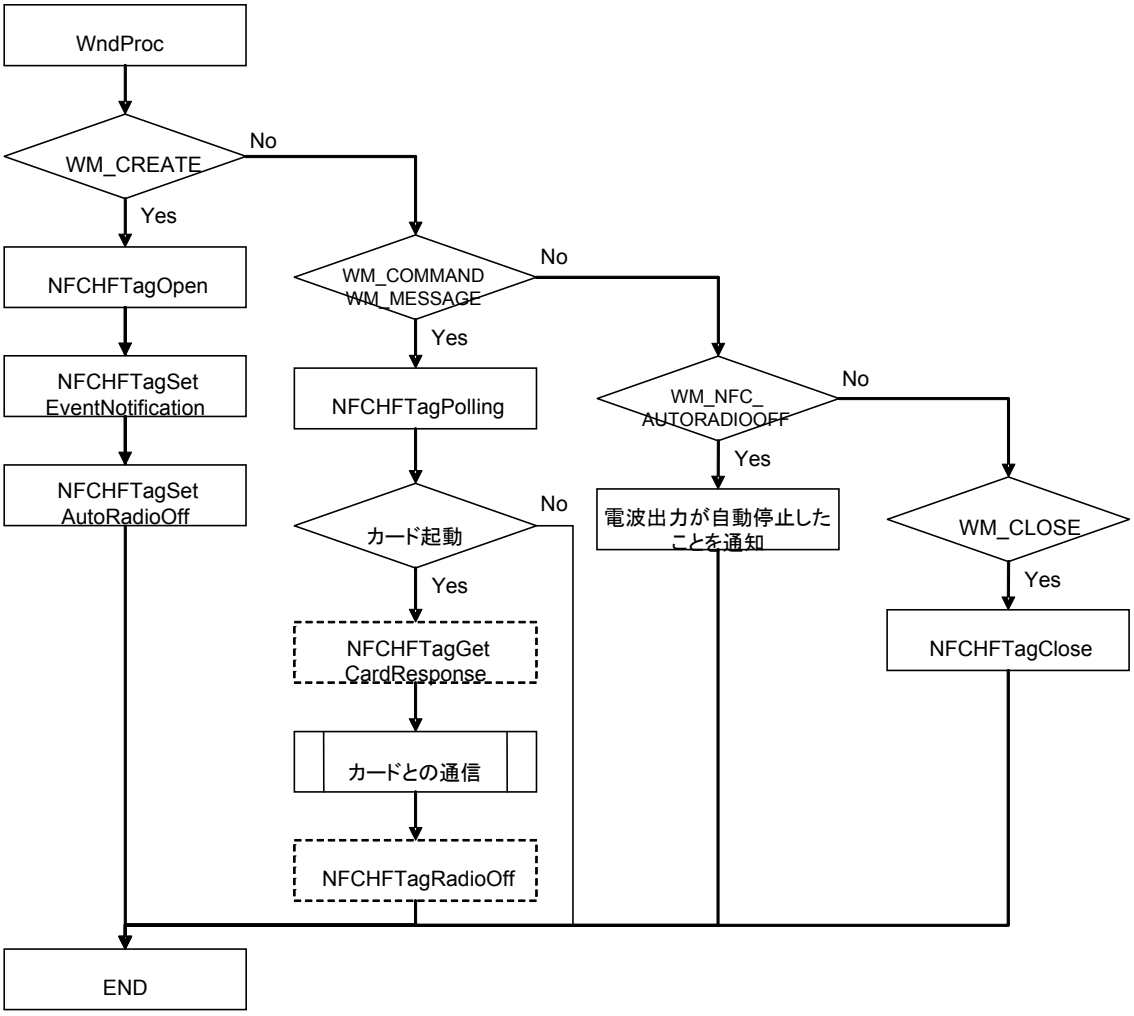
1. WM_CREATE メッセージを受け取った場合は、NFCHFTagOpen関数を実行し、読み取り待機状態にします。
2. WM_COMMAND、WM_KEYDOWN 等のメッセージを受け取った場合は、NFCHFTagPolling関数により、通信可能範囲内にある ISO15693 カードを検索/起動します。
3. ISO15693 カードの起動に成功、かつ、そのカードの詳細な情報が必要な場合は、NFCHFTagGetCardResponse関数により、応答情報を取得します。(任意)
4. ISO15693 カードとの通信を行います※。
5. ISO15693 カードとの通信が終了した場合は、NFCHFTagRadioOff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、NFCHFTagClose関数により、読み取り禁止状態にします。



※ ISO15693 カードとの通信については、「ISO15693 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、NFCHFTagOpen関数を実行し、読み取り待機状態にします。
2. NFCHFTagSetEventNotification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. NFCHFTagSetAutoRadioOff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、NFCHFTagPolling関数により、通信可能範囲内にある ISO15693 カードを検索/起動します。
5. ISO15693 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、NFCHFTagGetCardResponse関数により、応答情報を取得します。（任意）
6. ISO15693 カードとの通信を行います※。
7. ISO15693 カードとの通信が終了した場合は、NFCHFTagRadioOff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、NFCHFTagClose関数により、読み取り禁止状態にします。

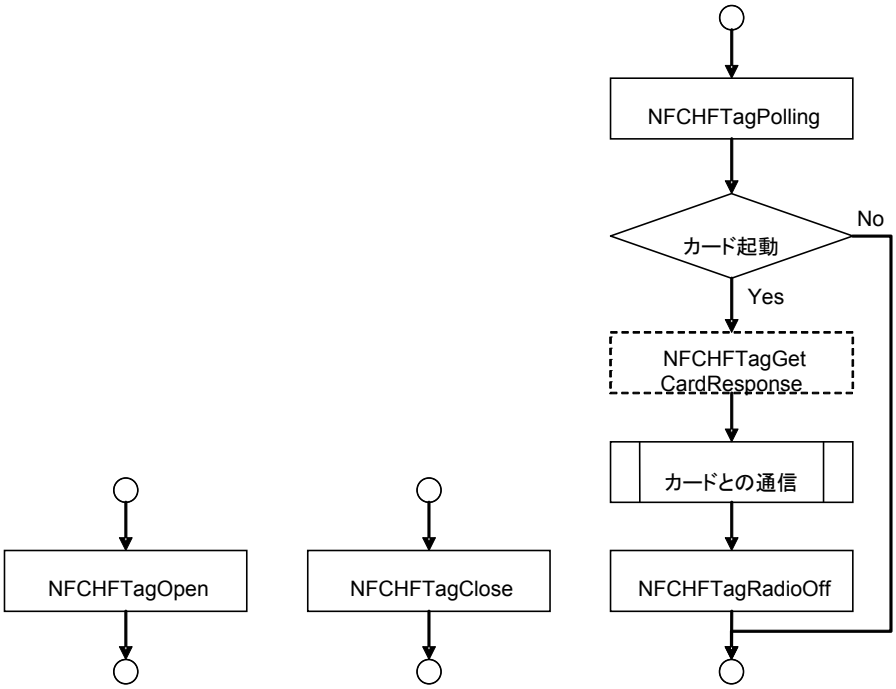


※ ISO15693 カードとの通信については、「ISO15693 カードとの通信について」を参照してください。

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、NFCHFTagOpen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、NFCHFTagPolling関数により、通信可能範囲内にある ISO15693 カードを検索/起動します。
3. ISO15693 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、NFCHFTagGetCardResponse関数により、応答情報を取得します。(任意)
4. ISO15693 カードとの通信を行います※。
5. ISO15693 カードとの通信が終了した場合は、NFCHFTagRadioOff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、NFCHFTagClose関数により、読み取り禁止状態にします。



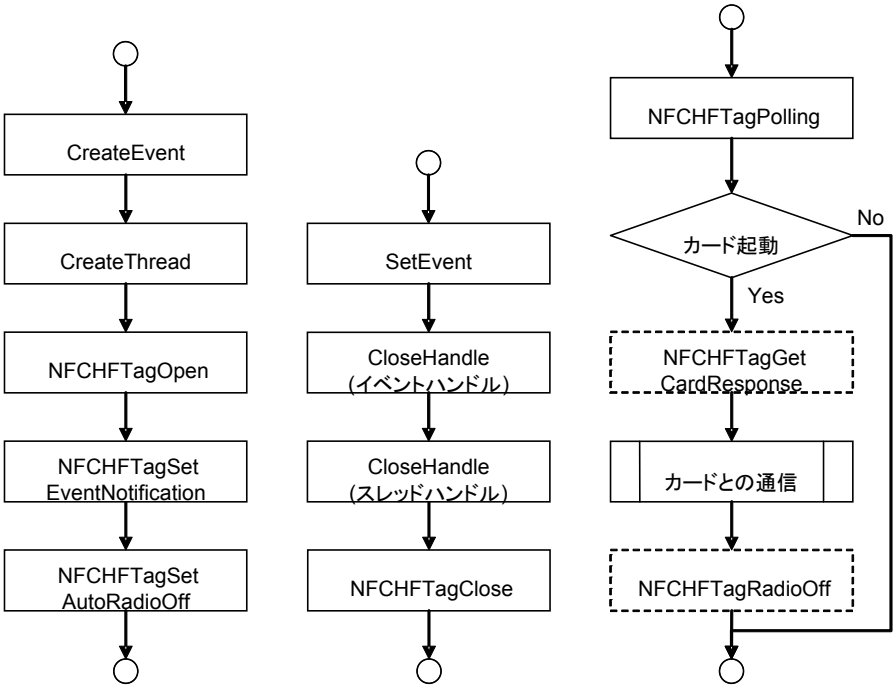
アプリケーション起動時　アプリケーション終了時　ISO15693 カードとの通信実行時

※ ISO15693 カードとの通信については、「ISO15693ISO15693 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. NFCHFTagOpen関数により、読み取り待機状態にします。
4. NFCHFTagSetEventNotification関数により、イベント通知を有効に設定します。
5. NFCHFTagSetAutoRadioOff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、NFCHFTagPolling関数により、通信可能範囲内にある ISO15693 カードを検索/起動します。
7. ISO15693 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、NFCHFTagGetCardResponse関数により、応答情報を取得します。(任意)
8. ISO15693 カードとの通信を行います※。
9. ISO15693 カードとの通信が終了した場合は、NFCHFTagRadioOff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. NFCHFTagClose関数により、読み取り禁止状態にします。

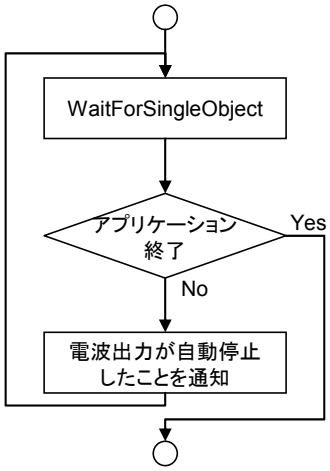


アプリケーション起動時　アプリケーション終了時　ISO15693 カードとの通信実行時

※ ISO15693 カードとの通信については、「ISO15693 カードとの通信について」を参照してください。

■ NFCHFTag スレッド

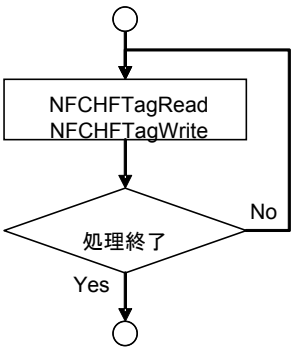
1. WaitForSingleObject 関数により、電波自動停止タイミング通知イベントハンドルに対して待機します。
2. アプリケーション終了時に通知イベントを受け取った場合、電波自動停止の監視を終了します。
3. 上記以外時に通知イベントを受け取った場合、電波出力が自動停止したことを通知することが可能です。



4.2 ISO15693 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. NFCHFTagRead関数または NFCHFTagWrite関数を実行します。

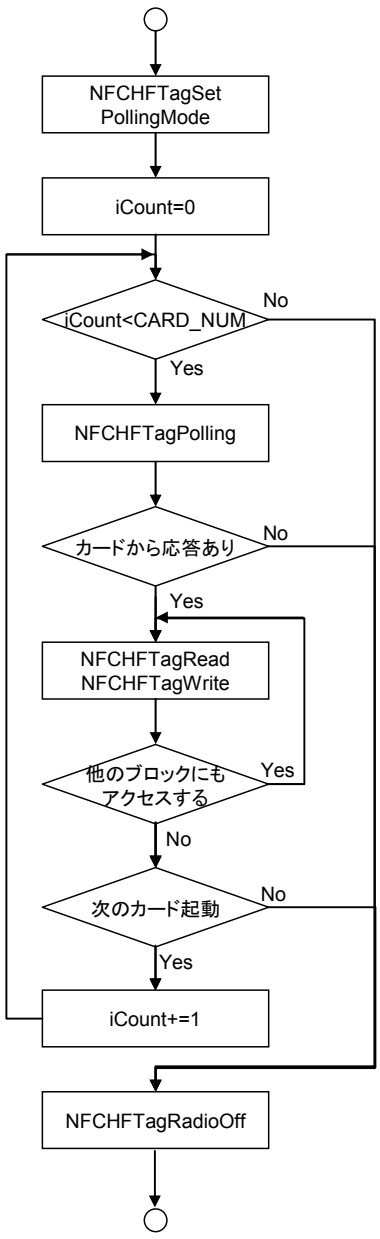


4.3 検索方式について

多段起動を使用する

ISO15693 カードと通信する

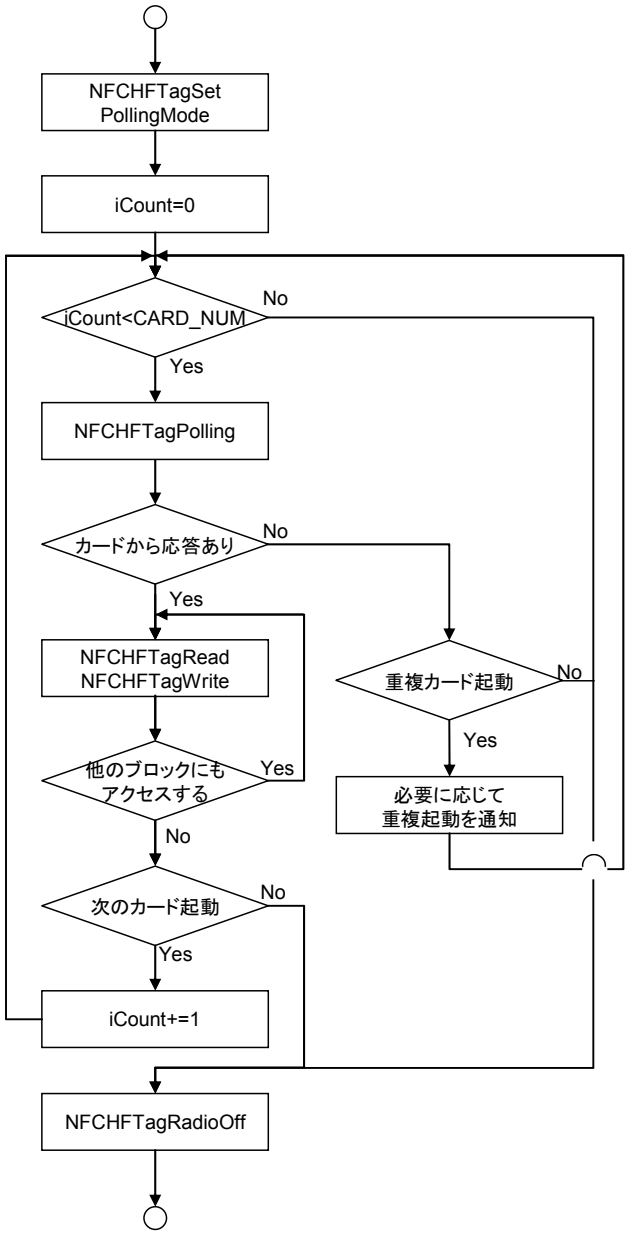
(ア) NFCHFTagSetPollingMode関数により、検索方式に多段起動（NFC_PLMODE_MULTISTEP）を、段数に連続起動するカード枚数 CARD_NUM を指定します。
(イ) iCount=0 をセットします。
(ウ) iCount<CARD_NUM の場合、次の処理に進みます。CARD_NUM は連続起動する IC カードの枚数を表します。
(エ) NFCHFTagPolling関数により通信範囲内の IC カードを検索します。
(オ) IC カードが起動して応答があったら、次の処理へ進みます。
(カ) NFCHFTagRead関数や、NFCHFTagWrite関数などにより、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）
(キ) 他のブロックにもアクセスする場合は、6.に戻って処理を繰り返します。
(ク) 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。
(ケ) 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。
(コ) NFCHFTagRadioOff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

ISO15693 カードと通信する

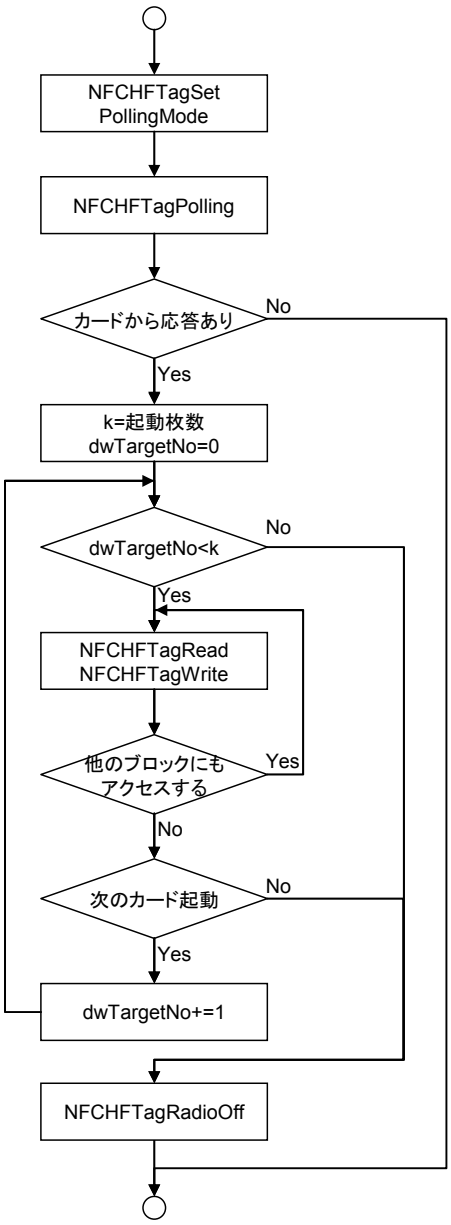
1. NFCHFTagSetPollingMode関数により、検索方式に多段起動（NFC_PLMODE_MULTISTEP2）を、段数に連続起動するカード枚数 CARD_NUM を指定します。
2. iCount=0 をセットします。
3. iCount<CARD_NUM の場合、次の処理に進みます。CARD_NUM は連続起動する IC カードの枚数を表します。
4. NFCHFTagPolling関数により通信範囲内のカードを検索します。
5. カードの起動に失敗し、NFCHFTagPolling 関数の戻り値が重複起動を表す場合、必要に応じて LED 等により重複起動を通知します。その後、3.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. NFCHFTagRead関数や、NFCHFTagWrite 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は、7.に戻って処理を繰り返します。
9. 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。
10. 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。
11. NFCHFTagRadioOff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

ISO15693 カードと通信する

1. NFCHFTagSetPollingMode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. NFCHFTagPolling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。（NFCHFTagRead関数や NFCHFTagWrite関数などの引数）
5. dwTargetNo が k よりも小さい場合、次の処理に進みます。
6. NFCHFTagRead関数や NFCHFTagWrite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）
7. 他のブロックにもアクセスする場合は、6.に戻って処理を繰り返します。
8. 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。
9. 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。
10. NFCHFTagRadioOff関数により、電波を停止します。



4.4 Tag-it™ カードとの通信について

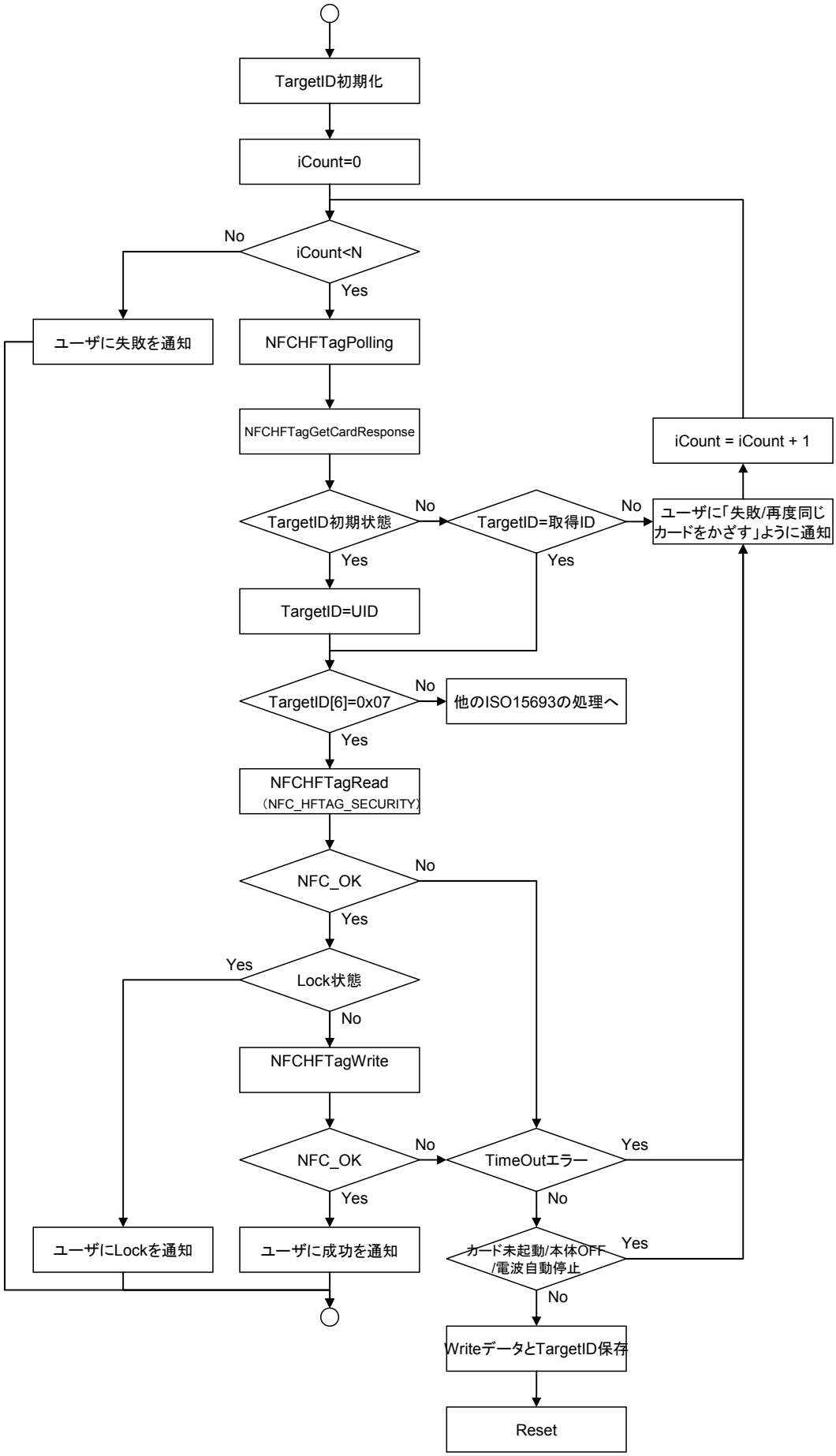
NFC コントローラの仕様により、Tag-it™ に対し NFCHFTagWrite, NFCHFTagLock, NFCHFTagWriteAFI, NFCHFTagWriteDSFID 関数を実施している最中に IC タグを通信可能範囲外に離した場合、戻り値が NFC_ERROR_TIMEOUT 等となりエラーを通知しても、Write や Lock が完了している場合があります。そのため、Write や Lock を確実に行うために、上記関数の戻り値が NFC_ERROR_TIMEOUT 等のエラーとなった場合は、これらの関数の戻り値が NFC_OK となるまで、リトライを行う必要があります。以下のフローの手順に従い、リトライ処理を行ってください。

Tag-it™ に対し HFTagWrite 関数を実行する場合

1. TargetID 配列を初期化します。0 クリア状態等に初期化します。
2. カウンタ変数 i に 0 をセットし、クリアします。
3. カウンタ変数 i が定数 N より小さい場合、次の処理に進みます。定数 N はループ回数の上限を表しており、回数制限を設けることで、無限ループ化を回避します。
4. NFCHFTagPolling 関数により通信範囲内のカードを検索します。
5. IC タグ(Tag-it™)を発見し、IC タグの起動に成功した場合、NFCHFTagGetCardResponse 関数により起動したカードの UID を取得します。
6. TargetID が初期状態の場合、取得した UID をセットします。
7. TargetID[6]をチェックします。TargetID[6]が 0x07 の場合、Tag-it™ シリーズの IC タグであるため、次の処理に進みます。
8. NFCHFTagRead 関数において、dwOption パラメータに NFC_HFTAG_SECURITY を指定して実行します。これは対象ブロックのロック状態を確認するために行います。
9. NFCHFTagRead 関数の戻り値が NFC_OK となった場合、読込データを格納している pbyData 配列をチェックします。pbyData[0]=0x00 の場合、対象ブロックはロックされていないので、次の処理に進みます。
10. NFCHFTagWrite 関数を実行します。
11. NFCHFTagWrite 関数の戻り値が NFC_OK の場合、アプリケーションにおいて、「書き込みに成功しました。」と表示し、処理を終了します。(正常終了)
12. NFCHFTagWrite 関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF のとき、書き込みに失敗している場合と、書き込みには成功しているが、書き込み後のチェックに失敗している場合があります。確実に書き込みを完了させるために、アプリケーションにおいて、「書き込み失敗。同じカードをかざして下さい。」と表示し、3 に戻って一連の処理のリトライを行います。
13. NFCHFTagWrite 関数の戻り値が手順 12 以外のエラー戻り値である場合、Write データと TargetID を保存。本体リセット実施後に 1 に戻り、再度本処理フローを実行します。このとき、手順 1 において、TargetID 配列は初期化せずに、保存した TargetID をセットします。 (※1)

※1 NFCHFTagWrite 関数のエラーには他に下記のエラー処理がありますが、通常は発生しないため、省略します。

NFC_NOT_DEVICE：NFC ドライバ非常駐時に発生。/　NFC_POF：NFCHFTagOpen 未実行時に発生。
NFC_PRM：不正パラメータ指定時に発生。 /
NFC_ERROR_INVALID_ACCESS：NFCHFTagPolling 実行中に NFCHFTagWrite 関数を実行時に発生。

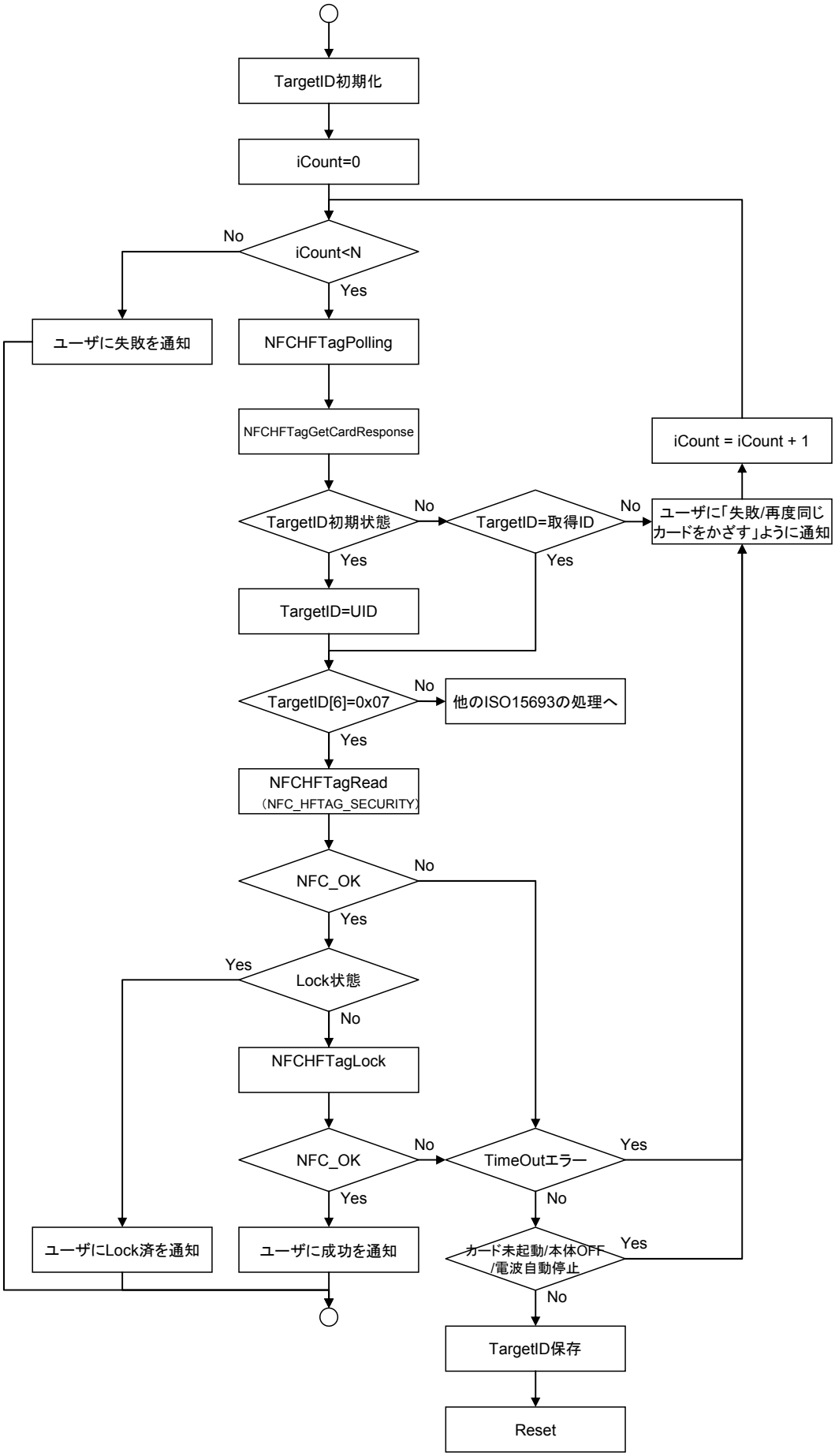

TargetID初期化
iCount=0
iCount<N
No
Yes
ユーザに失敗を通知
NFCHFTagPolling
NFCHFTagGetCardResponse
iCount = iCount + 1
TargetID初期状態
No
TargetID=取得ID
No
ユーザに「失敗/再度同じカードをかざす」ように通知
Yes
Yes
TargetID=UID
TargetID[6]=0x07
No
他のISO15693の処理へ
Yes
NFCHFTagRead
（NFC_HFTAG_SECURITY）
NFC_OK
No
Yes
Yes
Lock状態
No
NFCHFTagWrite
NFC_OK
No
TimeOutエラー
Yes
Yes
No
ユーザにLockを通知
ユーザに成功を通知
カード未起動/本体OFF/電波自動停止
Yes
No
WriteデータとTargetID保存
Reset

Tag-it™ に対し HFTagLock 関数を実行する場合

1. TargetID 配列を初期化します。0 クリア状態等に初期化します。
2. カウンタ変数 i に 0 をセットし、クリアします。
3. カウンタ変数 i が定数 N より小さい場合、次の処理に進みます。定数 N はループ回数の上限を表しており、回数制限を設けることで、無限ループ化を回避します。
4. NFCHFTagPolling 関数により通信範囲内のカードを検索します。
5. IC タグ(Tag-it™)を発見し、IC タグの起動に成功した場合、NFCHFTagGetCardResponse 関数により起動したカードの UID を取得します。
6. TargetID が初期状態の場合、取得した UID をセットします。
7. TargetID[6]をチェックします。TargetID[6]が 0x07 の場合、Tag-it™ シリーズの IC タグであるため、次の処理に進みます。
8. NFCHFTagRead 関数において、dwOption パラメータに NFC_HFTAG_SECURITY を指定して実行します。これは対象ブロックのロック状態を確認するために行います。
9. NFCHFTagRead 関数の戻り値が NFC_OK となった場合、読込データを格納している pbyData 配列をチェックします。pbyData[0]=0x00 の場合、対象ブロックはロックされていないので、次の処理に進みます。
10. NFCHFTagLock 関数を実行します。
11. NFCHFTagLock 関数の戻り値が NFC_OK の場合、アプリケーションにおいて、「ロックに成功しました。」と表示し、処理を終了します。(正常終了)
12. NFCHFTagLock 関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF のとき、ロックに失敗している場合と、ロックには成功しているが、ロック後のチェックに失敗している場合があります。確実にロックを完了させるために、アプリケーションにおいて、「ロック失敗。同じカードをかざして下さい。」と表示し、3 に戻って本処理のリトライを行います。
13. NFCHFTagLock 関数の戻り値が手順 12 以外のエラー戻り値である場合、TargetID を保存。本体リセット実施後に 1 に戻り、再度本処理フローを実行します。このとき、手順 1 において、TargetID 配列は初期化せずに、保存した TargetID をセットします。 (※1)

※1 NFCHFTagLock 関数のエラーには他に下記のエラー処理がありますが、通常は発生しないため、省略します。

NFC_NOT_DEVICE：NFC ドライバ非常駐時に発生。/　NFC_POF：NFCHFTagOpen 未実行時に発生。
NFC_PRM：不正パラメータ指定時に発生。 /
NFC_ERROR_INVALID_ACCESS：NFCHFTagPolling 実行中に NFCHFTagLock 関数を実行時に発生。


TargetID初期化
iCount=0
No
iCount<N
Yes
ユーザに失敗を通知
NFCHFTagPolling
NFCHFTagGetCardResponse
iCount = iCount + 1
TargetID初期状態
No
TargetID=取得ID
No
ユーザに「失敗/再度同じカードをかざす」ように通知
Yes
Yes
TargetID=UID
TargetID[6]=0x07
No
他のISO15693の処理へ
Yes
NFCHFTagRead
（NFC_HFTAG_SECURITY）
NFC_OK
No
Yes
Yes
Lock状態
No
NFCHFTagLock
NFC_OK
No
TimeOutエラー
Yes
Yes
No
ユーザにLock済を通知
ユーザに成功を通知
カード未起動/本体OFF/電波自動停止
Yes
No
TargetID保存
Reset

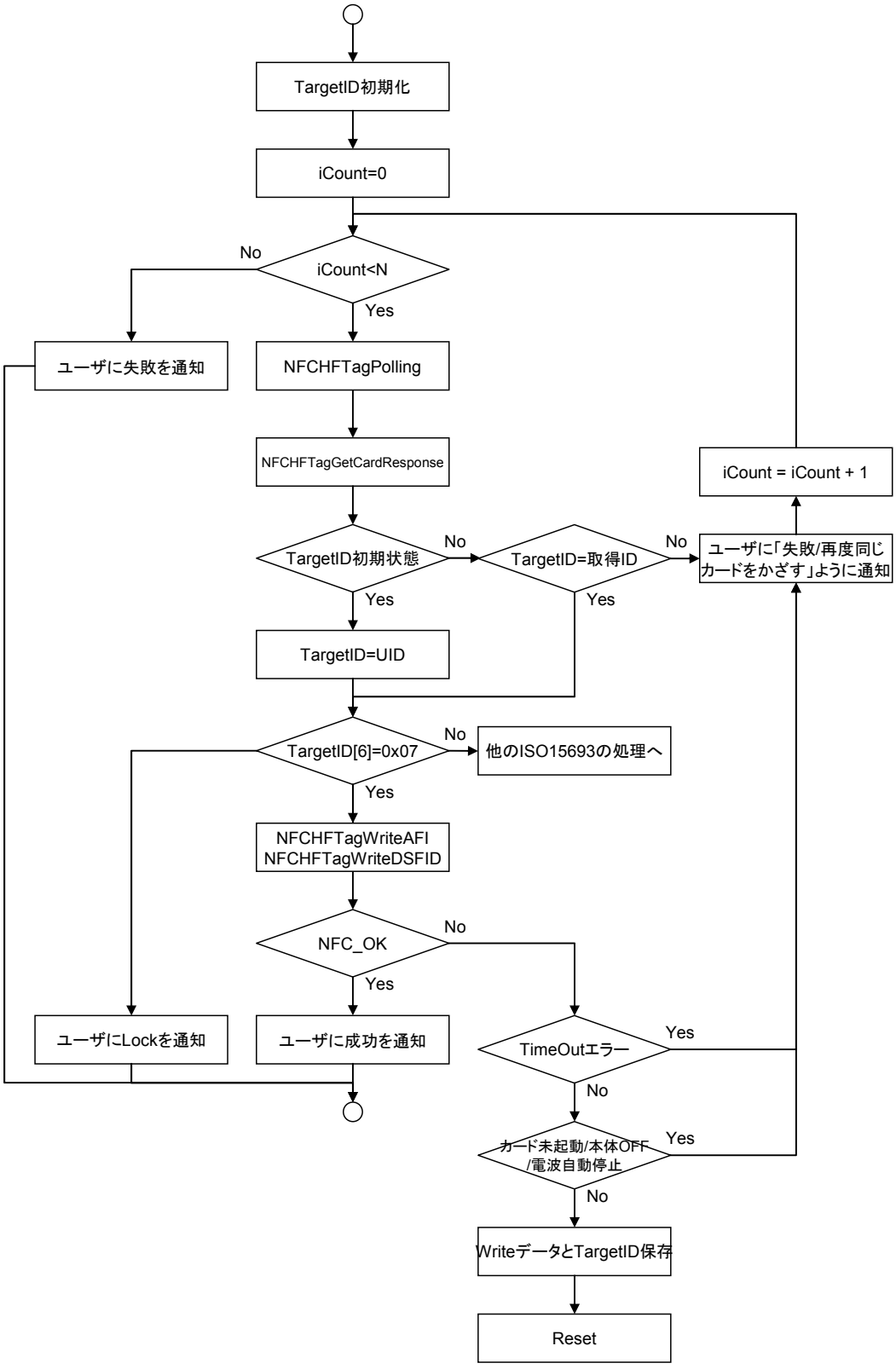
Tag-it™ に対し NFCHFTagWriteAFI、NFCHFTagWriteDSFID 関数を実行する場合

1. TargetID 配列を初期化します。0 クリア状態等に初期化します。
2. カウンタ変数 i に 0 をセットし、クリアします。
3. カウンタ変数 i が定数 N より小さい場合、次の処理に進みます。定数 N はループ回数の上限を表しており、回数制限を設けることで、無限ループ化を回避します。 (※1)
4. NFCHFTagPolling 関数により通信範囲内のカードを検索します。
5. IC タグ(Tag-it™)を発見し、IC タグの起動に成功した場合、NFCHFTagGetCardResponse 関数により起動したカードの UID を取得します。
6. TargetID が初期状態の場合、取得した UID をセットします。
7. TargetID[6]をチェックします。TargetID[6]が 0x07 の場合、Tag-it™ シリーズの IC タグであるため、次の処理に進みます。
8. NFCHFTagWriteAFI または NFCHFTagWriteDSFID 関数を実行します。
9. NFCHFTagWriteAFI 関数または NFCHFTagWriteDSFID 関数の戻り値が NFC_OK の場合、アプリケーションにおいて、「書き込みに成功しました。」と表示し、処理を終了します。(正常終了)
10. NFCHFTagWriteAFI 関数または NFCHFTagWriteDSFID 関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF のとき、書き込みに失敗している場合と、書き込みには成功しているが、書き込み後のチェックに失敗している場合があります。確実に書き込みを完了させるために、アプリケーションにおいて、「書き込み失敗。同じカードをかざして下さい。」と表示し、i に 1 加算し、3 に戻って本処理のリトライを行います。 (※1)
11. NFCHFTagWriteAFI 関数または NFCHFTagWriteDSFID 関数の戻り値が手順 10 以外のエラー戻り値である場合、Write データと TargetID を保存。本体リセット実施後に 1 に戻り、再度本処理フローを実行します。このとき、手順 1 において、TargetID 配列は初期化せずに、保存した TargetID をセットします。 (※2)

※1　AFI または DSFID 領域について、ロック状態を確認する方法がないため、既にロックされているかどうかを確認し、ロックされている場合はループ処理を終了するというようなプログラムを組むことができません。そのため、リトライのためのループ回数に制限を設けて指定回数ループ処理を行っても書き込みできない場合は、エラーとして終了するよう処理を行ってください。

※2　NFCHFTagWriteAFI 関数、NFCHFTagWriteDSFID 関数のエラーには他に下記のエラー処理がありますが、通常は発生しないため、省略します。

NFC_NOT_DEVICE：NFC ドライバ非常駐時に発生。/　NFC_POF：NFCHFTagOpen 未実行時に発生。
NFC_PRM：不正パラメータ指定時に発生。 /
NFC_ERROR_INVALID_ACCESS：NFCHFTagPolling 実行中に NFCHFTagWriteAFI, NFCHFTagWriteDSFID 関数を実行時に発生。


TargetID初期化
iCount=0
No
iCount<N
Yes
ユーザに失敗を通知
NFCHFTagPolling
NFCHFTagGetCardResponse
iCount = iCount + 1
TargetID初期状態
No
TargetID=取得ID
No
ユーザに「失敗/再度同じカードをかざす」ように通知
Yes
Yes
TargetID=UID
TargetID[6]=0x07
No
他のISO15693の処理へ
Yes
NFCHFTagWriteAFI
NFCHFTagWriteDSFID
NFC_OK
No
Yes
ユーザにLockを通知
ユーザに成功を通知
TimeOutエラー
Yes
No
カード未起動/本体OFF/電波自動停止
Yes
No
WriteデータとTargetID保存
Reset

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/

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)