CASIO

NFC ライブラリマニュアル

このマニュアルは、NFC ライブラリの
仕様について記載します。

Ver. 1.06

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	2009.07		新規作成
1.01	2009.09	11	NFCGetCardResponse関数の引数を修正
		15	NFCExchangeData関数の引数を修正
1.02	2010.05	15	NFCExchangeData関数の解説を追加
		25	NFCSetEventNotification関数の戻り値を追加
		28	NFCSetAutoRadioOff関数の戻り値を追加
		41-48	プログラミング上の注意点のフローチャートを修正
1.03	2010.10	15	NFCExchangeData関数の解説を修正
1.04	2011.08	-	対象機種に DT-X8 を、対象 IC カードに ISO15693 を追加し、全面改訂
1.05	2011.11	5	NFCOpen関数の注意を追加
		11	NFCGetCardResponse関数の説明を追加
		34	NFCGetCardResponseEx関数の説明を追加
1.06	2011.11	-	対象機種に IT-9000 を追加
	2012.01	49	多段起動時の注意点を追加
	2012.02	30	NFCSetPollingMode関数の補足を追加
		49-50	多段起動時のコールバック関数に関する記述を削除

目次

1. 概要

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

NFC クラスライブラリは、NFC 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。

NFC 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。

NFC 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。
NFC 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。

※ NFC 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

NFC 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#
NFCLib.h	○	-
NFCLib.lib	○	-
NFCLib.dll	○	○
NFCLibNet.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 NFCLib.h をインクルードし、リンカの依存ファイルとして NFCLib.lib を指定してください。
- NFCLib.dll は本体に内蔵されています。

Visual Basic または Visual C#の場合

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

NFC ﾗｲﾌﾞ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
NFCOpen	通信許可状態を設定	○	○	○
NFCClose	通信禁止状態を設定	○	○	○
NFCPollingCard	通信範囲内の IC カード検索を開始	○	○	○
NFCStopPolling	通信範囲内の IC カード検索を停止	○	○	○
NFCGetCardResponse	起動した IC カードの詳細情報を取得	○	○	○
NFCRadioOff	電波送信を停止	○	○	○
NFCExchangeData	起動した IC カードとの通信を実行	○	○	○
NFCSetEventNotification	電波自動停止のﾀｲﾐﾝｸﾞ通知方法を設定	○	○	○
NFCGetEventNotification	電波自動停止のﾀｲﾐﾝｸﾞ通知方法を取得	○	○	○
NFCSetAutoRadioOff	電波自動停止までの時間を設定	○	○	○
NFCGetAutoRadioOff	電波自動停止までの時間を取得	○	○	○
NFCSetPollingMode	IC カードの検索方法を設定	-	○	○
NFCGetPollingMode	IC カードの検索方法を取得	-	○	○
NFCGetCardResponseEx	起動した IC カードの詳細情報を取得	-	○	○
NFCSetHFTagAFI	起動を有効にする ISO15693 カードの AFI を設定	-	○	○
NFCGetHFTagAFI	起動を有効にする ISO15693 カードの AFI を取得	-	○	○
NFCSetFelicaSystemCode	起動を有効にする Felica のシステムコードを設定	-	○	○
NFCGetFelicaSystemCode	起動を有効にする Felica のシステムコードを取得	-	○	○

○ 関数サポート
－ 関数未サポート

3.1 NFCOpen

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 NFCOpen(
    HWND  hWnd
)
```

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

```
[C#]
public static Int32 NFCOpen(
    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NFCClose関数を実行するまで有効です。
Open 状態時に、NFCPollingCard関数を実行すると、通信を開始します。
NFC デバイスを電源 ON にしても微量の電力しか消費しません。
また、電源 ON 処理には時間がかかるため、アプリケーション起動時にあらかじめ電源 ON にし、アプリケーション終了時に電源 OFF にしてください。

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

対応情報

機種	： DT-5300 / DT-X8 / IT-9000
ヘッダ	： NFCLib.h
ライブラリ	： NFCLib.lib

注意

DT-X8 / IT-9000 において、NFC_ERROR_MODULE が返る場合は、本関数を 3 回ﾘﾄﾗｲしてください。

3.2 NFCClose

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 NFCClose()
```

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

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

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

対応情報

機種	:	DT-5300 / DT-X8 / IT-9000
ヘッダ	:	NFCLib.h
ライブラリ	:	NFCLib.lib

3.3 NFCPollingCard

通信可能範囲内にある IC カードを検索します。

```
[C++]
int NFCPollingCard(
    DWORD  dwBaudRate,
    DWORD  dwTimeout,
    BOOL   (*fpCallBack)(void),
    DWORD  *pdwActBaudRate,
    DWORD  dwParam
)
```

```
[Visual Basic]
Public Shared Function NFCPollingCard( _
    ByVal dwBaudRate      As Int32, _
    ByVal dwTimeout       As Int32, _
    ByVal fpCallBack      As IntPtr, _
    ByRef pdwActBaudRate  As Int32, _
    ByVal dwParam         As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCPollingCard(
    Int32      dwBaudRate,
    Int32      dwTimeout,
    IntPtr     fpCallBack,
    ref Int32  pdwTargetSize,
    Int32      dwParam
)
```

解説

本関数は、通信可能範囲内にある IC カードを検索します。
IC カードを発見した場合は、その IC 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 IC 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。
IC カードの検索方式についてはNFCSetPollingMode関数を参照してください。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

dwBaudRate

起動する IC カードのボーレートを以下から指定します。(OR 指定可能)

NFC_BAUD_TYPEA	:	ISO / IEC14443 TypeA (106 kbps TypeA)
NFC_BAUD_TYPEB	:	ISO / IEC14443 TypeB (106 kbps TypeB)
NFC_BAUD_FELICA1	:	Felica (212 kbps)
NFC_BAUD_FELICA2	:	Felica (424 kbps)
NFC_BAUD_HFTAG	:	ISO15693 (DT-5300 以外)

dwTimeout

IC 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 IC カードを検索します。

fpCallBack

IC 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。

pdwActBaudRate

IC カードの起動に成功した場合は、その IC カードのボーレートを取得します。取得する値については、dwBaudRate を参照してください。
また、IC カードの起動に失敗した場合は、何も取得しません。

dwParam

IC カード検索を行う際の動作モードを指定します。本機能を使用しない場合は 0 を指定してください。
DT-5300 では 0 を指定してください。

NFC_PL_SAVE　：IC カード検索時の消費電力を抑えます
長時間連続して IC カードの検索を行う場合は本パラメータを使用してください
※ IC カードの検出レスポンスは低下します

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_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 DeviceEmulator では発生しません

対応情報

機種	：DT-5300 / DT-X8 / IT-9000
ヘッダ	：NFCLib.h
ライブラリ	：NFCLib.lib

3.4 NFCStopPolling

通信可能範囲内にある IC カードの検索を停止します。

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

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

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

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NFCPollingCard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 NFCGetCardResponse

起動した IC カードの応答情報を取得します。

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

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

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

解説

NFCPollingCard関数成功後に本関数を実行すると、起動した 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

pbyTargetData
IC カードからの応答情報を取得します。バッファサイズは 261 バイト以上確保してください。

■ ISO / IEC14443 Type A (Mifare Standard / Ultralight)の場合

2バイト	1バイト	1バイト	nバイト	mバイト
ATQA	SAK	UIDバイト数(n)	UID	ATS (ISO14443-4 プロトコル対応時のみ取得、先頭1バイトはサイズ)

■ ISO / IEC14443 Type B の場合

12バイト	1バイト	nバイト
ATQB	ATTRIB応バイト数(n)	ATTRIB応答

※ DT-X8 / IT-9000 において ATTRIB 応答バイト数(n)は 0 です。

ATQB の形式は以下のようになります。

1バイト	4バイト	4バイト	3バイト
’50’	PUPI	応用データ	プロトコル情報

※ DT-X8 / IT-9000 においてプロトコル情報の 3 バイトはすべて 0 になります。

■ Felica の場合

8バイト	8バイト	2バイト
IDm	PMm	システムコード

■ ISO15693 の場合

1バイト	8バイト
DSFID	UID

pdwActualSize

IC 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。

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

実行例

■ ISO / IEC14443 Type A (Mifare Ultralight)の場合

ATQA	SAK	UIDLen	UID(7バイト)
04 00	08	07	01 02 03 04 05 06 07

※ Mifare は ISO14443-3 までの準拠となり、ISO14443-4 で規定しているプロトコルは使用していません。そのため、ATS 情報は取得しません。

■ ISO / IEC14443 Type B の場合

DT-5300 の場合

ATQB	ATTRIB応答サイズ	ATTRIB応答
50 01 02 03 04 00 00 00 00 00 00 00	01	01

DT-X8 / IT-9000 の場合

ATQB	ATTRIB応答サイズ	ATTRIB応答
50 01 02 03 04 00 00 00 00 00 00 00	00	-

■ Felica の場合

IDm	PMm	システムコード
01 01 06 01 67 02 A5 15	03 00 4B 02 4F 49 8A 8A	FF FF

■ ISO15693 の場合

DSFID	UID
00	7E C6 DF 01 00 00 07 E0

対応情報

機種　　：DT-5300 / DT-X8 / IT-9000
ヘッダ　　：NFCLib.h
ライブラリ　：NFCLib.lib

3.6 NFCRadioOff

NFC モジュールの電波送信を停止します。

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

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

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

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

対応情報

機種	:	DT-5300 / DT-X8 / IT-9000
ヘッダ	:	NFCLib.h
ライブラリ	:	NFCLib.lib

3.7 NFCExchangeData

起動した IC カードとのデータ通信を行います。

```
[C++]
int NFCExchangeData(
    BYTE   *pbySendData,
    DWORD  dwSendSize,
    BYTE   *pbyReceiveData,
    DWORD  *pdwActualSize,
    DWORD  dwTargetNo
)
```

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

```
[C#]
public static Int32 NFCExchangeData(
    Byte[]     pbySendData,
    Int32      dwSendSize,
    Byte[]     pbyReceiveData,
    ref Int32  pdwActualSize,
    Int32      dwTargetNo
)
```

解説

本関数は、NFCPollingCard関数により起動した IC カードとのデータ通信を行います。
IC カードからの応答を受信する、または、タイムアウト時間（60 ミリ秒）経過すると通信を終了します。
DeviceEmulator では、パラメータチェックのみを行います。

パラメータ

pbySendData

IC 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、IC カードの種類によって異なります。
送信するバイナリデータは、IC カードに送信する全データを指定するのではなく、一部分のみを指定します。
指定箇所の詳細については、以下を参照してください。
以下の色付箇所が指定箇所となります。（無色箇所は自動付加します）
指定可能なデータの最大値は 261 バイトです。

ISO14443-4 (Type A、Type B)

S	プロローグフィールド	pSendDataの一部※	エピローグフィールド	E

※ pSendData を分割して送信する場合は、分割データごとに自動付加します

Mifare

SOF	pSendData	EOF

※ Mifare は ISO14443-3 までの準拠となり、ISO14443-4 で規定している通信プロトコルは使用していません。そのため、ISO14443-4 とは異なるデータ書式となります。

Felica

プリアンブル	同期コード “B24D”	pSendData(コマンド、データサイズ含む)	CRC

ISO15693

SOF	pSendData(フラグ、コマンドデータ含む)	CRC	EOF

例) Mifare Standard の場合

■ Mifare Standard コマンド一覧

メモリ操作	内容
Read	16 バイト読み込み
Write	16 バイト書き込み
Increment	加算 (計算のみ)
Decrement	減算 (計算のみ)
Transfer	計算結果をメモリに反映
Restore	置換 (計算のみ)

※ 各コマンドの詳細は非公開情報です

■ Mifare Standard メモリ構成

セクタ	ブロック	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	カード製造者用エリア (16バイト)															
	1																
	2																
	3	鍵A (6バイト)						アクセスビット (4バイト)				鍵B (6バイト)					
1	4																
	5																
	6																
	7	鍵A (6バイト)						アクセスビット (4バイト)				鍵B (6バイト)					
(省略)																	
14	56																
	57																
	58																
	59	鍵A (6バイト)						アクセスビット (4バイト)				鍵B (6バイト)					
15	60																
	61																
	62																
	63	鍵A (6バイト)						アクセスビット (4バイト)				鍵B (6バイト)					

※ 空白の領域がユーザ使用可能領域です

例) Mifare Ultralight の場合

■ Mifare Ultralight コマンド一覧

メモリ操作	コマンド (1 バイト)	ページ番号 (1 バイト)	追加パラメータ (可変長)
Read	0x30	0x00～0x0F の いずれか	なし
Write	0xA2	0x00～0x0F の いずれか	書き込みデータ (4 バイト)

■ Mifare Ultralight メモリ構成

ページ番号	0x00	0x01	0x02	0x03
0x00	シリアルナンバー0	シリアルナンバー1	シリアルナンバー2	BCC0
0x01	シリアルナンバー3	シリアルナンバー4	シリアルナンバー5	シリアルナンバー6
0x02	BCC1	Internal	Lock0	Lock1
0x03	OPT0	OPT1	OPT2	OPT3
0x04	データ0	データ1	データ2	データ3
0x05	データ4	データ5	データ6	データ7
0x06	データ8	データ9	データ10	データ11
0x07	データ12	データ13	データ14	データ15
0x08	データ16	データ17	データ18	データ19
0x09	データ20	データ21	データ22	データ23
0x0A	データ24	データ25	データ26	データ27
0x0B	データ28	データ29	データ30	データ31
0x0C	データ32	データ33	データ34	データ35
0x0D	データ36	データ37	データ38	データ39
0x0E	データ40	データ41	データ42	データ43
0x0F	データ44	データ45	データ46	データ47

※ 色づけした領域 (緑色) がユーザ使用可能領域です

例) Felica の場合

■ Felica コマンド一覧

メモリ操作	全体のバイト数 (1 バイト)	コマンド (1 バイト)	IDm (8 バイト)	追加パラメータ (可変長)
Read Without Encryption	12 + 2m + N	0x06	NFCGetCard Response 関数で取得した値	サービス数 m (1 バイト) サービスコードリスト (2m バイト) ブロック数 (1 バイト) ブロックリスト (N バイト)
Write Without Encryption	12 + 2m + N + 16n	0x08	NFCGetCard Response 関数で取得した値	サービス数 m (1 バイト) サービスコードリスト (2m バイト) ブロック数 n (1 バイト) ブロックリスト (N バイト) ブロックデータ (16n バイト)

※ 各コマンドの詳細については Felica カードの仕様を確認してください

例) ISO15693 の場合

■ ISO15693 コマンド一覧

メモリ操作	フラグ (1 バイト)	コマンド (1 バイト)	UID (8 バイト)	追加パラメータ (可変長)
Read Single Block (Tag-it の場合)	0x60	0x20	NFCGetCard Response 関数で取得した値	ブロック番号 (1 バイト)
Read Single Block (Tag-it 以外の場合)	0x20	0x20	NFCGetCard Response 関数で取得した値	ブロック番号 (1 バイト)
Write Single Block (Tag-it 以外の場合)	0x20	0x21	NFCGetCard Response 関数で取得した値	ブロック番号 (1 バイト) データ (1 ブロックサイズ、通常 4 バイト)

※ 各コマンドの詳細については ISO15693 カードの規格を確認してください

※ フラグの設定により UID は不要になります (詳細は ISO15693 カードの規格を確認してください)

※ Tag-It シリーズにおいて、以下のコマンドはサポート対象外です

Lock AFI / Lock DSFID

また、Write Single Block / Lock Block / Write AFI / Write DSFID コマンドを使用する場合は、アプリケーションにてコマンドのリトライ処理を行う必要があります (詳細については、「4.4 コマンド送信について」を参照してください)

■ ICODE SLI メモリ構成

Block番号	Byte 1	Byte 2	Byte 3	Byte 4
-4	UID0	UID1	UID2	UID3
-3	UID4	UID5	UID6	UID7
-2	内部使用	EAS	AFI	DSFID
-1	アクセス情報書込み			
0				
1				
…				
26				
27				

※ 色づけした領域 (緑色) がユーザ使用可能領域です

■ ICODE SLI-L メモリ構成

Page番号	Block番号	Byte 1	Byte 2	Byte 3	Byte 4
-2	-8	UID0	UID1	UID2	UID3
	-7	UID4	UID5	UID6	UID7
	-6	内部使用	EAS	AFI	DSFID
	-5	UID4	UID5	UID6	UID7
-1	-4	内部使用	EAS	AFI	DSFID
	-3	UID4	UID5	UID6	UID7
	-2	内部使用	EAS	AFI	DSFID
	-1	UID4	UID5	UID6	UID7
0	0				
	1				
	2				
	3				
1	4				
	5				
	6				
	7				

※ 色づけした領域（緑色）がユーザ使用可能領域です

■ Tag-It HF-I Plus メモリ構成

Block番号	Byte 1	Byte 2	Byte 3	Byte 4
0				
1				
…				
62				
63				
-	UID0	UID1	UID2	UID3
-	UID4	UID5	UID6	UID7
-	-	-	-	DSFID
-	-	-	-	AFI
-	-	IC Version		

※ 色づけした領域（緑色）がユーザ使用可能領域です

dwSendSize

pSendData に指定するバイナリデータのサイズを指定します。

pbyReceiveData

IC 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 261 バイト以上確保してください。

応答情報の書式は、IC カードの種類によって異なります。

取得する応答情報は、IC カードからの応答の全データを取得するのではなく、一部分のみを取得します。

取得箇所の詳細については、以下を参照してください。

以下の色付箇所が取得箇所となります。（無色箇所は自動カットします）

※ 応答情報の詳細については、IC カードの仕様を確認してください

ISO14443-4 (Type A、Type B)

S	プロローグフィールド	pReceiveDataの一部※	エピローグフィールド	E

※ pSendData を分割して送信する場合は、分割データごとに自動付加します

Mifare

SOF	pReceiveData	EOF

Felica

プリアンブル	同期コード "B24D"	pReceiveData(コマンド、データサイズ含む)	CRC

ISO15693

SOF	pReceiveData(フラグ、コマンドデータ含む)	CRC	EOF

例) Mifare Ultralight の場合

■ Mifare Ultralight 応答一覧

メモリ操作	応答データ (n バイト)
Read	読み込みデータ (16 バイト)
Write	なし (0 バイト)

例) Felica の場合

■ Felica 応答一覧

メモリ操作	全体のバイト数 (1 バイト)	コマンド (1 バイト)	IDm (8 バイト)	追加パラメータ (可変長)
Read Without Encryption	13 + 16n	0x07	NFCGetCard Response 関数で取得した値と同じ	ステータスフラグ 1 (1 バイト) ステータスフラグ 2 (1 バイト) ブロック数 n (1 バイト) ブロックデータ (16n バイト)
Write Without Encryption	12	0x09	NFCGetCard Response 関数で取得した値と同じ	ステータスフラグ 1 (1 バイト) ステータスフラグ 2 (1 バイト)

例) ISO15693 の場合

■ ISO15693 応答一覧

メモリ操作	フラグ (1 バイト)	追加パラメータ (可変長)
Read Single Block (Tag-It の場合)	0x00	セキュリティステー達 (1 バイト) データ (1 ブロックサイズ、通常 4 バイト)
Read Single Block (Tag-It 以外の場合)	0x00	データ (1 ブロックサイズ、通常 4 バイト)
Write Single Block (Tag-It 以外の場合)	0x00	-

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

dwTargetNo
通信したい IC カードに対応するカード番号を指定します。通常は 0 を指定してください。
NFCPollingCard関数により複数の 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に、1 以上の値を指定してください。
詳細はNFCGetCardResponseEx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
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 DeviceEmulator では発生しません

補足

戻り値の NFC_OK は、一部の IC カードの場合を除いて、IC カードとの無線通信の正常終了を表すものであり、メモリに対する読み込み/書き込みの正常終了を表すものではありません。使用する IC カードの種類によっては、メモリに対する読込み/書き込みの成功/失敗を判断するために、別途カードからの応答情報（pbyReceiveData に格納されるデータ）の中身の確認を行う必要があります。各種 IC カードに対する戻り値の意味の差異を以下の表に示します。

IC カード種別	NFC_OK が返る条件	メモリの読み込み/書き込みの成功条件
Mifare Standard Mifare Ultralight	IC カードとの無線通信に成功し、さらにメモリの読み込み/書き込みにも成功	戻り値が NFC_OK のとき
Felica JICSAP（Type B） ISO15693	IC カードとの無線通信に成功 （メモリの読み込み/書き込みに成功しているとは限りません）	戻り値が NFC_OK であり、さらに IC カードからの応答情報（pbyReceiveData に格納されるデータ）におけるメモリアクセスに対するステータス情報が「成功」のとき （ステータス情報は IC カードのコマンド仕様により異なります。Felica の場合、pbyReceiveData において、追加パラメータのステータスフラグ 1 がメモリアクセスに対するステータス情報であり、0x00 の場合は成功、それ以外の場合は失敗となります）

実行例

例) Mifare Ultralight の場合

■ Read コマンド実行時（ページ番号 0x08）

pbySendData

コマンド（1 バイト）	ページ番号（1 バイト）	追加パラメータ（0 バイト）
30	08	

pbyReceiveData

応答データ（16 バイト）
04 03 02 01 00 00 00 00 00 00 00 00 00 00 00 00

■ Write コマンド実行時（ページ番号 0x08）

pbySendData

コマンド（1 バイト）	ページ番号（1 バイト）	追加パラメータ（4 バイト）
A2	08	04 03 02 01

pbyReceiveData

応答データ（0 バイト）

例) Felica の場合

■ Read Without Encryption コマンド実行時（サービスコード 0x1009、ブロック番号 0x00）

pbySendData

全体バイト数（1 バイト）	コマンド（1 バイト）	IDm（8 バイト）	追加パラメータ（6 バイト）
10	06	01 02 03 04 05 06 07 08	01 09 10 01 80 00

pbyReceiveData

全体バイト数（1 バイト）	レスポンス（1 バイト）	IDm（8 バイト）	追加パラメータ（19 バイト）
1D	07	01 02 03 04 05 06 07 08	00 00 01 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 04

■ Write Without Encryption コマンド実行時（サービスコード 0x1009、ブロック番号 0x00）

pbySendData

全体バイト数（1 バイト）	コマンド（1 バイト）	IDm（8 バイト）	追加パラメータ（22 バイト）
20	08	01 02 03 04 05 06 07 08	01 09 10 01 80 00 00 00 00 01 00 00 00 02 00 00 00 03 00 00 00 04

pbyReceiveData

全体バイト数（1 バイト）	レスポンス（1 バイト）	IDm（8 バイト）	追加パラメータ（19 バイト）
0C	09	01 02 03 04 05 06 07 08	00 00

例) ISO15693 の場合

■ Read Single Block コマンド実行時（Tag-It かつブロック番号 0x00 の場合）

pbySendData

フラグ （1 バイト）	コマンド （1 バイト）	UID （8 バイト）	追加パラメータ （1 バイト）
60	20	01 02 03 04 05 06 07 08	00

pbyReceiveData

フラグ （1 バイト）	追加パラメータ （5 バイト）
00	00 01 02 03 04

■ Read Single Block コマンド実行時（Tag-It 以外かつブロック番号 0x00 の場合）

pbySendData

フラグ （1 バイト）	コマンド （1 バイト）	UID （8 バイト）	追加パラメータ （1 バイト）
20	20	01 02 03 04 05 06 07 08	00

pbyReceiveData

フラグ （1 バイト）	追加パラメータ （4 バイト）
00	01 02 03 04

■ Write Single Block コマンド実行時（Tag-It 以外かつブロック番号 0x00 の場合）

pbySendData

フラグ （1 バイト）	コマンド （1 バイト）	UID （8 バイト）	追加パラメータ （5 バイト）
20	21	01 02 03 04 05 06 07 08	00 01 02 03 04

pbyReceiveData

フラグ （1 バイト）
00

対応情報

機種　　：DT-5300 / DT-X8 / IT-9000
ヘッダ　　：NFCLib.h
ライブラリ　：NFCLib.lib

3.8 NFCSetEventNotification

電波自動停止のタイミング通知方法を設定します。

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

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

```
[C#]
public static Int32 NFCSetEventNotification(
    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	： 電波自動停止タイマ動作中に実行エラー

対応情報

機種　　： DT-5300 / DT-X8 / IT-9000

ヘッダ　　：NFCLib.h
ライブラリ　：NFCLib.lib

3.9 NFCGetEventNotification

電波自動停止のタイミング通知方法を取得します。

```
[C++]
int NFCGetEventNotification(
    DWORD *pdwMode
)
```

```
[Visual Basic]
Public Shared Function NFCGetEventNotification(
    ByRef pdwMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCGetEventNotification(
    ref Int32 pdwMode
)
```

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

pdwMode
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、NFCSetEventNotification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

対応情報

機種	: DT-5300 / DT-X8 / IT-9000
ヘッダ	: NFCLib.h
ライブラリ	: NFCLib.lib

3.10 NFCSetAutoRadioOff

電波自動停止までの時間を設定します。

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

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

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

dwTimeout

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	： 電波自動停止タイマ動作中に実行エラー

対応情報

機種	： DT-5300 / DT-X8 / IT-9000
ヘッダ	： NFCLib.h
ライブラリ	： NFCLib.lib

3.11 NFCGetAutoRadioOff

電波自動停止までの時間を取得します。

```
[C++]
int NFCGetAutoRadioOff(
    DWORD *pdwTimeout
)
```

```
[Visual Basic]
Public Shared Function NFCGetAutoRadioOff(
    ByRef pdwTimeout As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCGetAutoRadioOff(
    ref Int32 pdwTimeout
)
```

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

pdwTimeout

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、NFCSetAutoRadioOff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

対応情報

機種	: DT-5300 / DT-X8 / IT-9000
ヘッダ	: NFCLib.h
ライブラリ	: NFCLib.lib

3.12 NFCSetPollingMode

IC カードの検索方式を設定します。

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

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

```
[C#]
public static Int32 NFCSetPollingMode(
    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※

※ TypeB は未対応です

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	： 4（TypeA） 2（Felica および ISO15693）

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NFCStopPolling関数を実行したときにクリアします。

対応情報

機種	： DT-X8 / IT-9000
ヘッダ	： NFCLib.h
ライブラリ	： NFCLib.lib

3.13 NFCGetPollingMode

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、NFCSetPollingMode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、NFCSetPollingMode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

対応情報

機種　　：DT-X8 / IT-9000
ヘッダ　　：NFCLib.h
ﾗｲﾌﾞﾗﾘ　：NFCLib.lib

3.14 NFCGetCardResponseEx

起動した IC カードの応答情報を取得します。

```
[C++]
int NFCGetCardResponseEx(
    BYTE   *pbyTargetData,
    DWORD  *pdwActualSize,
    DWORD  *pdwDiscoveredNum,
    DWORD  *pdwReserved
)
```

```
[Visual Basic]
Public Shared Function NFCGetCardResponseEx( _
    ByVal pbyTargetData    As Byte(), _
    ByRef pdwActualSize    As Int32, _
    ByRef pdwDiscoveredNum As Int32, _
    ByRef pdwReserved      As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCGetCardResponseEx(
    Byte[]     pbyTargetData,
    ref Int32  pdwActualSize,
    ref Int32  pdwDiscoveredNum,
    ref Int32  pdwReserved
)
```

解説

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

pbyTargetData
IC カードからの応答情報を取得します。
バッファサイズは（262×NFCSetPollingMode関数の dwNum）以上確保してください。

全体のデータ書式

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枚目のカード情報サイズa	1枚目のカード情報	2枚目のカード情報サイズb	2枚目のカード情報	…	n枚目のカード情報サイズx	n枚目のカード情報

カード１枚分のデータ書式

■ ISO / IEC14443 Type A (Mifare Standard / Ultralight)の場合

2バイト	1バイト	1バイト	nバイト	mバイト
ATQA	SAK	UIDバイト数(n)	UID	ATS (ISO14443-4 プロトコル対応時のみ取得、先頭1バイトはサイズ)

■ ISO / IEC14443 Type B の場合

12バイト	1バイト	nバイト
ATQB	ATTRIB応バイト数(n)	ATTRIB応答

※ DT-X8 / IT-9000 において ATTRIB 応答バイト数(n)は 0 です。

ATQB の形式は以下のようになります。

1バイト	4バイト	4バイト	3バイト
'50'	PUPI	応用データ	プロトコル情報

※ DT-X8 / IT-9000 においてプロトコル情報の 3 バイトはすべて 0 になります。

■ Felica の場合

8バイト	8バイト	2バイト
IDm	PMm	システムコード

■ ISO15693 の場合

1バイト	8バイト
DSFID	UID

pdwActualSize

IC カードから取得した応答情報のサイズを取得します。

pdwDiscoveredNum

NFCPollingCard関数で起動に成功した IC カードの枚数を取得します。
NFCExchangeData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
NFCExchangeData関数の dwTargetNo に指定可能な値は 0～2 となります。

dwReserved

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_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

対応情報

機種　：DT-X8 / IT-9000
ヘッダ　：NFCLib.h
ライブラリ　：NFCLib.lib

3.15 NFCSetHFTagAFI

起動を有効にする ISO15693 カードの AFI を設定します。

```
[C++]
int NFCSetHFTagAFI(
    BYTE byAFI
)
```

```
[Visual Basic]
Public Shared Function NFCSetHFTagAFI(
    ByVal byAFI As Byte _
) As Int32
```

```
[C#]
public static Int32 NFCSetHFTagAFI(
    Byte byAFI
)
```

解説

本関数は、起動を有効にする ISO15693 カードの AFI を設定します。

パラメータ

byAFI
起動を有効にする ISO15693 カードの AFI を指定します（0x00～0xFF）。
0x00 を指定すると、すべての ISO15693 カードの起動を有効にします（デフォルト）。

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

対応情報

機種	： DT-X8 / IT-9000
ヘッダ	： NFCLib.h
ライブラリ	： NFCLib.lib

3.16 NFCGetHFTagAFI

起動を有効にする ISO15693 カードの AFI を取得します。

```
[C++]
int NFCGetHFTagAFI(
    BYTE *pbyAFI
)
```

```
[Visual Basic]
Public Shared Function NFCGetHFTagAFI(
    ByRef pbyAFI As Byte _
) As Int32
```

```
[C#]
public static Int32 NFCGetHFTagAFI(
    ref Byte pbyAFI
)
```

解説

本関数は、起動を有効にする ISO15693 カードの AFI を取得します。

パラメータ

pbyAFI
起動を有効にする ISO15693 カードの AFI を取得します。取得する値の詳細については、NFCSetHFTagAFI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

対応情報

機種	: DT-X8 / IT-9000
ヘッダ	: NFCLib.h
ライブラリ	: NFCLib.lib

3.17 NFCSetFelicaSystemCode

起動を有効にする Felica カードのシステムコードを設定します。

```
[C++]
int NFCSetFelicaSystemCode(
    DWORD dwSystemCode
)
```

```
[Visual Basic]
Public Shared Function NFCSetFelicaSystemCode(
    ByVal dwSystemCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCSetFelicaSystemCode(
    Int32 dwSystemCode
)
```

解説

本関数は、起動を有効にする Felica カードのシステムコードを設定します。

パラメータ

dwSystemCode
起動を有効にする Felica カードのシステムコードを指定します（0x0000～0xFFFF）。
0xFFFF を指定すると、すべての Felica カードの起動を有効にします（デフォルト）。

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

対応情報

機種	： DT-X8 / IT-9000
ヘッダ	： NFCLib.h
ライブラリ	： NFCLib.lib

3.18 NFCGetFelicaSystemCode

起動を有効にする Felica カードのシステムコードを取得します。

```
[C++]
int NFCGetFelicaSystemCode(
    DWORD *pdwSystemCode
)
```

```
[Visual Basic]
Public Shared Function NFCGetFelicaSystemCode(
    ByRef pdwSystemCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 NFCGetFelicaSystemCode(
    ref Int32 pdwSystemCode
)
```

解説

本関数は、起動を有効にする Felica カードのシステムコードを取得します。

パラメータ

pdwSystemCode
起動を有効にする Felica カードのシステムコードを取得します。取得する値の詳細については、NFCSetFelicaSystemCode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

対応情報

機種	: DT-X8 / IT-9000
ヘッダ	: NFCLib.h
ライブラリ	: NFCLib.lib

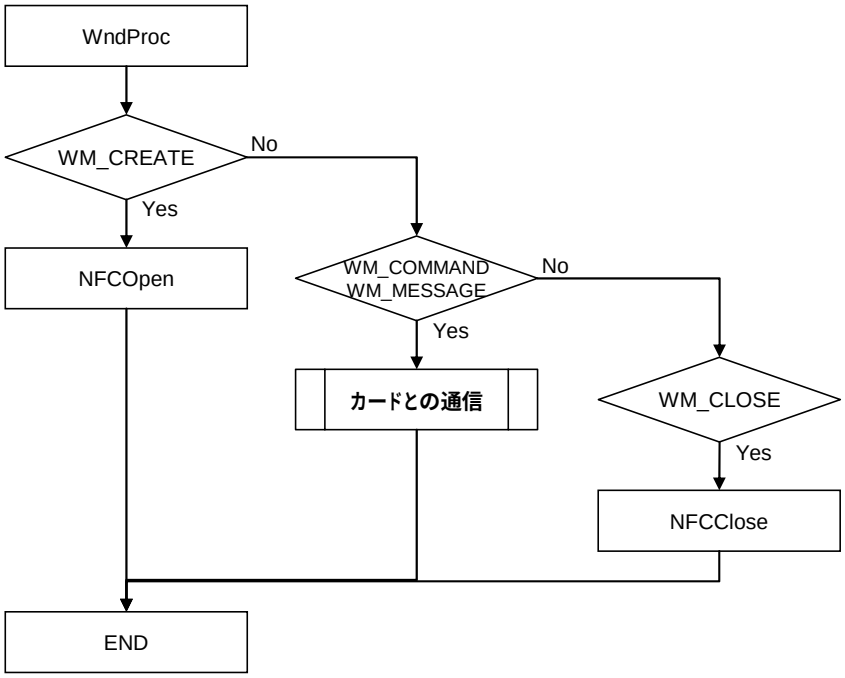
4. プログラミング上の注意点

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

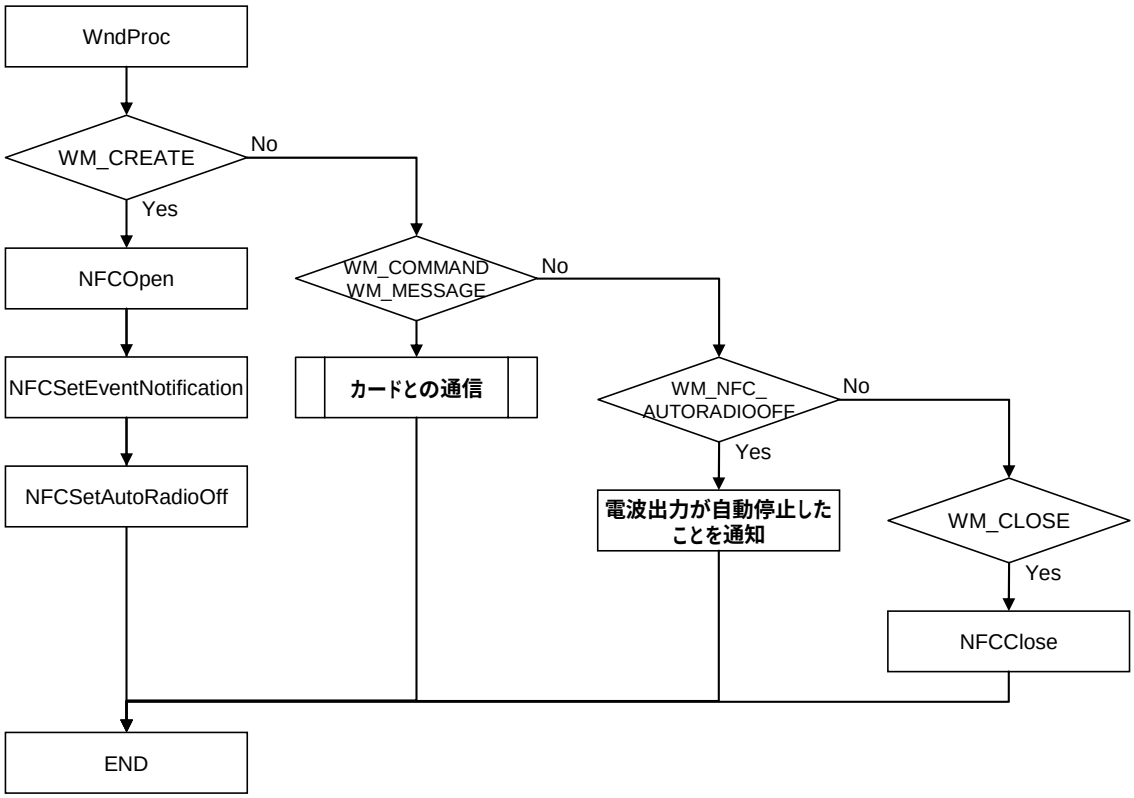
電波を手動で停止する

1. WM_CREATE メッセージを受け取った場合は、NFCOpen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、IC カードとの通信を行います。
   （各 IC カードとの通信については、「IC カードとの通信について」を参照してください）
3. IC カードとの通信が終了した場合は、電波を停止するための処理を行います。
   （電波を停止する処理については、「IC カードとの通信について」を参照してください）
4. WM_CLOSE メッセージを受け取った場合は、NFCClose関数により、読み取り禁止状態にします。



電波を自動で停止し、停止タイミングを通知する

5. WM_CREATE メッセージを受け取った場合は、NFCOpen関数を実行し、読み取り待機状態にします。
6. NFCSetEventNotification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。
7. NFCSetAutoRadioOff関数により、電波自動停止を有効に設定します。
8. WM_COMMAND、WM_KEYDOWN 等のメッセージを受け取った場合は、IC カードとの通信を行います。
   （各 IC カードとの通信については、「IC カードとの通信について」を参照してください）
9. IC カードとの通信が終了後、通信を行わずに一定時間経過すると自動的に電波を停止します。
10. 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。
11. WM_CLOSE メッセージを受け取った場合は、NFCClose関数により、読み取り禁止状態にします。



イベント通知を使用する

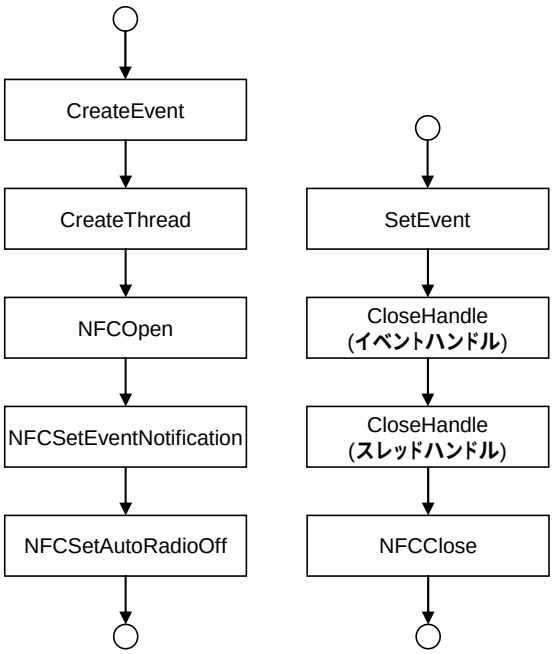
電波を手動で停止する

12. アプリケーション開始時に、NFCOpen関数により、読み取り待機状態にします。
13. 各カードタイプに応じた通信手順により、IC カードとの通信を行います。
    （各 IC カードとの通信については、「IC カードとの通信について」を参照してください）
14. IC カードとの通信が終了した場合は、電波を停止するための処理を行います。
    （電波を停止する処理については、「IC カードとの通信について」を参照してください）
15. アプリケーション終了時に、NFCClose関数により、読み取り禁止状態にします。

電波を自動で停止し、停止タイミングを通知する

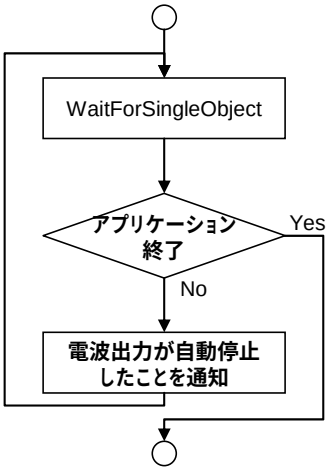
■ メインスレッド

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. NFCOpen関数により、読み取り待機状態にします。
4. NFCSetEventNotification関数により、イベント通知を有効に設定します。
5. NFCSetAutoRadioOff関数により、電波自動停止を有効に設定します。
6. 各カードタイプに応じた通信手順により、IC カードとの通信を行います。
   （各 IC カードとの通信については、「IC カードとの通信について」を参照してください）
7. IC カードとの通信が終了後、通信を行わずに一定時間経過すると自動的に電波を停止します。
8. 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。
9. 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。
10. NFCClose関数により、読み取り禁止状態にします。



■ NFC スレッド

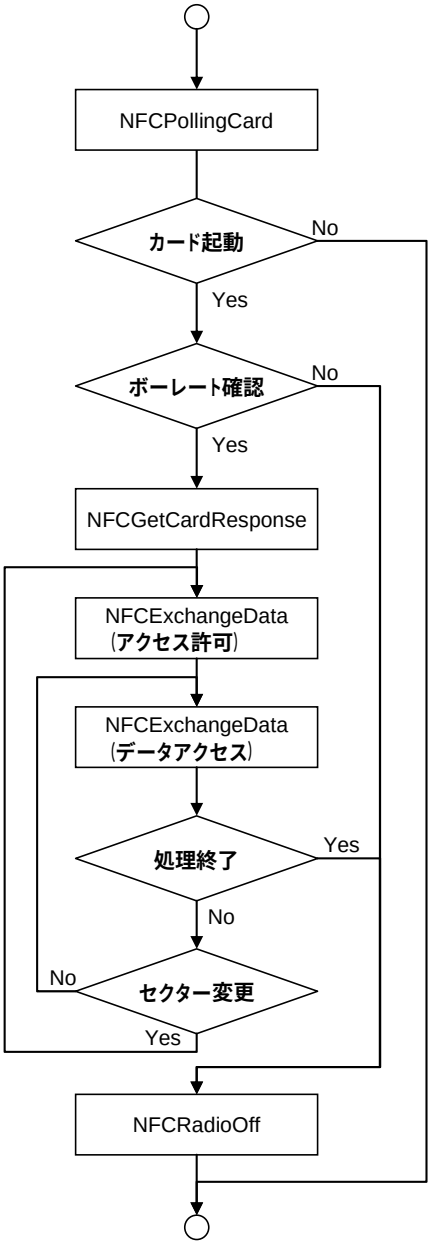
11. WaitForSingleObject 関数により、電波自動停止タイミング通知イベントハンドルに対して待機します。
12. アプリケーション終了時に通知イベントを受け取った場合、電波自動停止の監視を終了します。
13. 上記以外時に通知イベントを受け取った場合、電波出力が自動停止したことを通知することが可能です。



4.2 IC カードとの通信について

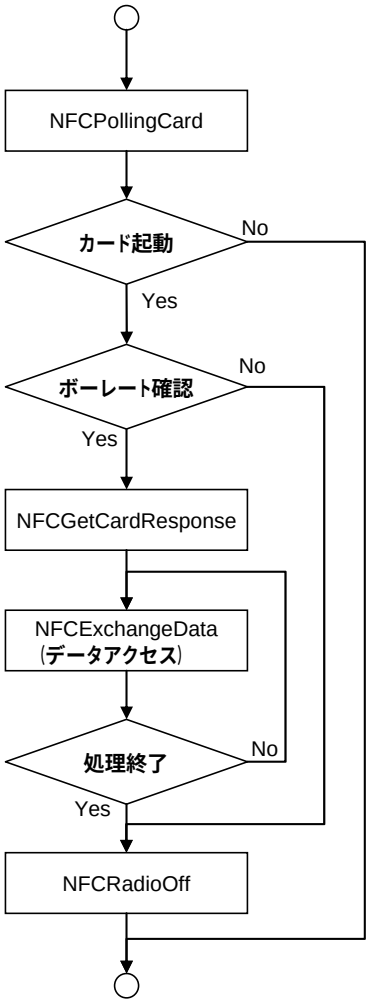
Mifare (Standard) カードと通信する

1. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
2. 起動に成功した IC カードのボーレートが Mifare カードのボーレートと一致しているかを確認します。
   一致している場合は、NFCGetCardResponse関数により Mifare カードの UID を取得します。
3. NFCExchangeData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。
4. NFCExchangeData関数により、Mifare コマンドを送信し、Mifare 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コマンドを送信)
5. 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、3.に戻って処理を繰り返します。
6. 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、4.に戻って処理を繰り返します。
7. 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は必要ありません)



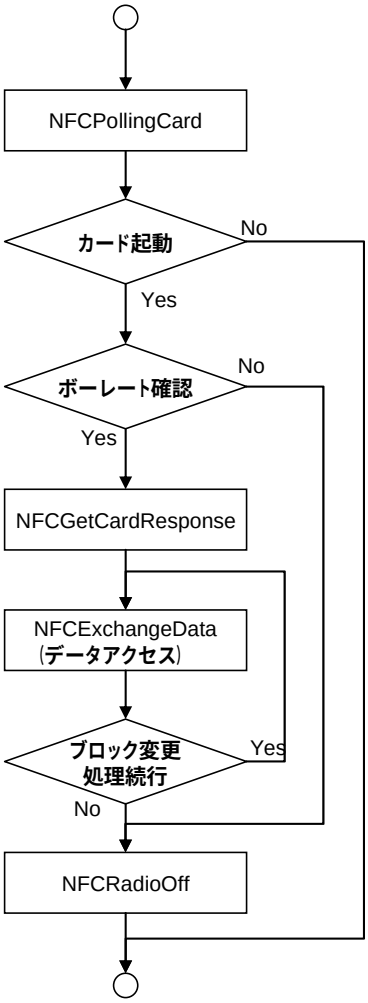
Felica カードと通信する

1. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
2. 起動に成功した IC カードのボーレートが Felica カードのボーレートと一致しているかを確認します。
   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。
3. NFCExchangeData関数により、Felica コマンド 0x06 または 0x08、および、パラメータを送信し、Felica 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コマンドを送信)
4. 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は必要ありません)



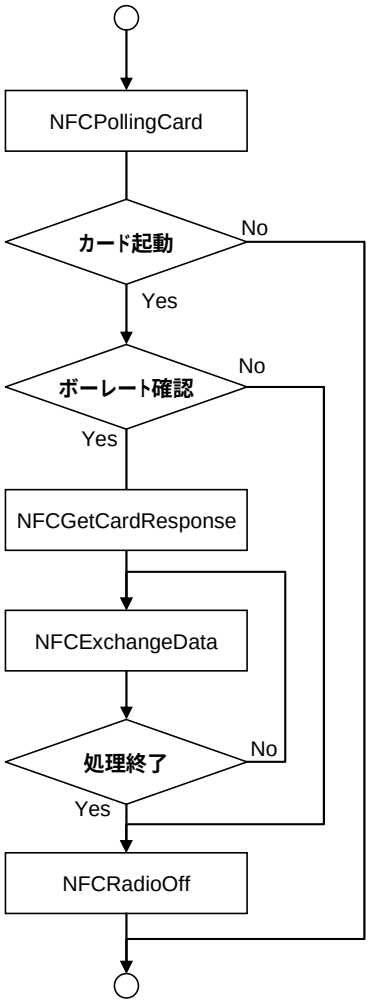
ISO15693 カードと通信する

1. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
2. 起動に成功した IC カードのボーレートが ISO15693 カードのボーレートと一致しているかを確認します。一致している場合は、NFCGetCardResponse関数により ISO15693 カードの UID を取得します。
3. NFCExchangeData関数により、ISO15693 コマンド 0x20 または 0x21、および、パラメータを送信し、ISO15693 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コマンドを送信)
4. ブロックを変更して処理を続行する場合は3.に戻って処理を繰り返します。
5. 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は必要ありません)



他の IC カードと通信する

1. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
2. 起動に成功した IC カードのボーレートが通信対象 IC カードのボーレートと一致しているかを確認します。一致している場合は、NFCGetCardResponse関数により IC カードの情報を取得します。
3. NFCExchangeData関数により、IC カードと通信を行います。
4. 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は必要ありません）

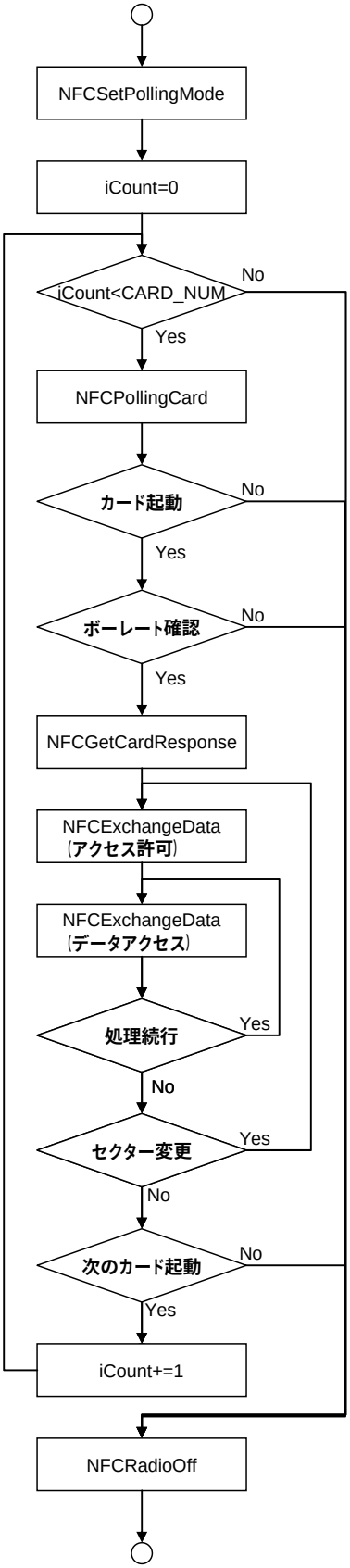


4.3 検索方式について

多段起動を使用する

Mifare (Standard) カードと通信する

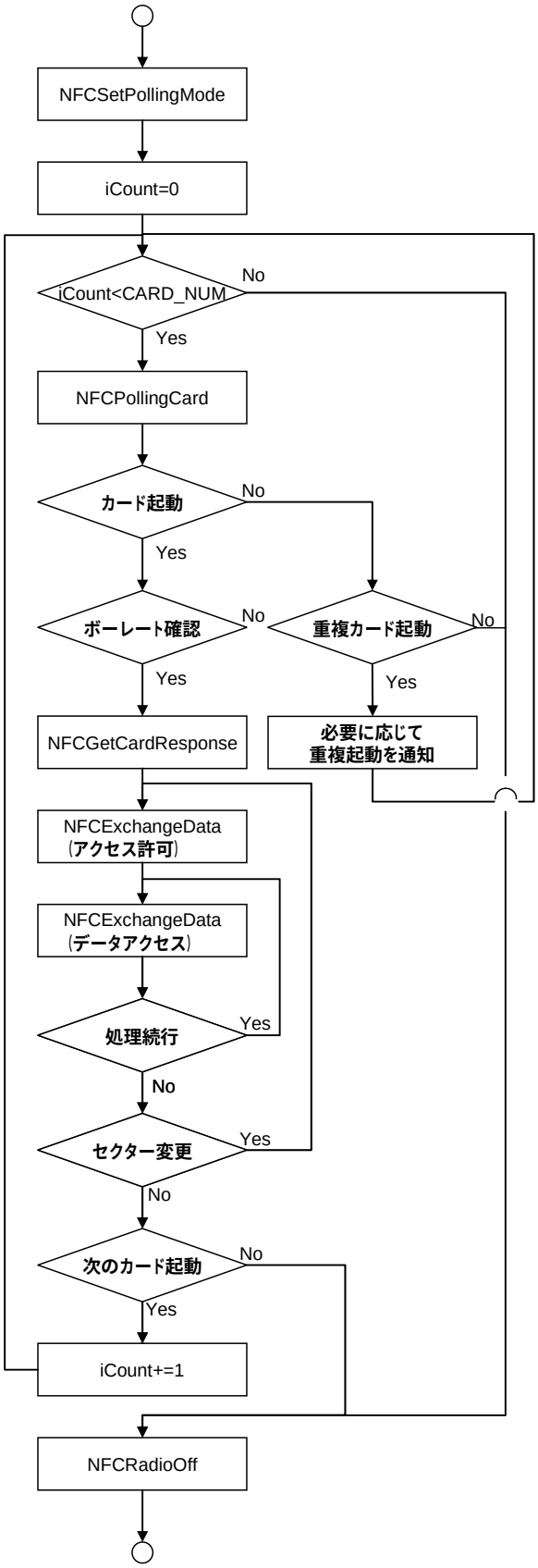
1. NFCSetPollingMode関数により、検索方式に多段起動（NFC_PLMODE_MULTISTEP）を、段数に連続起動するカード枚数 CARD_NUM を指定します。
2. iCount=0 をセットします。
3. iCount<CARD_NUM の場合、次の処理に進みます。CARD_NUM は連続起動する IC カードの枚数を表します。
4. NFCPollingCard関数により通信範囲内の IC カードを検索します。
5. 起動に成功した IC カードのボーレートが Mifare カードのボーレートと一致しているかを確認します。一致している場合は、NFCGetCardResponse関数により Mifare カードの UID を取得します。
6. NFCExchangeData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. 次にNFCExchangeData関数により、Mifare コマンドを送信し、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コマンドを送信）
8. セクターを変更せずに処理を続行する場合、7.に戻って処理を繰り返します。
9. セクターを変更して処理を続行する場合、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. 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

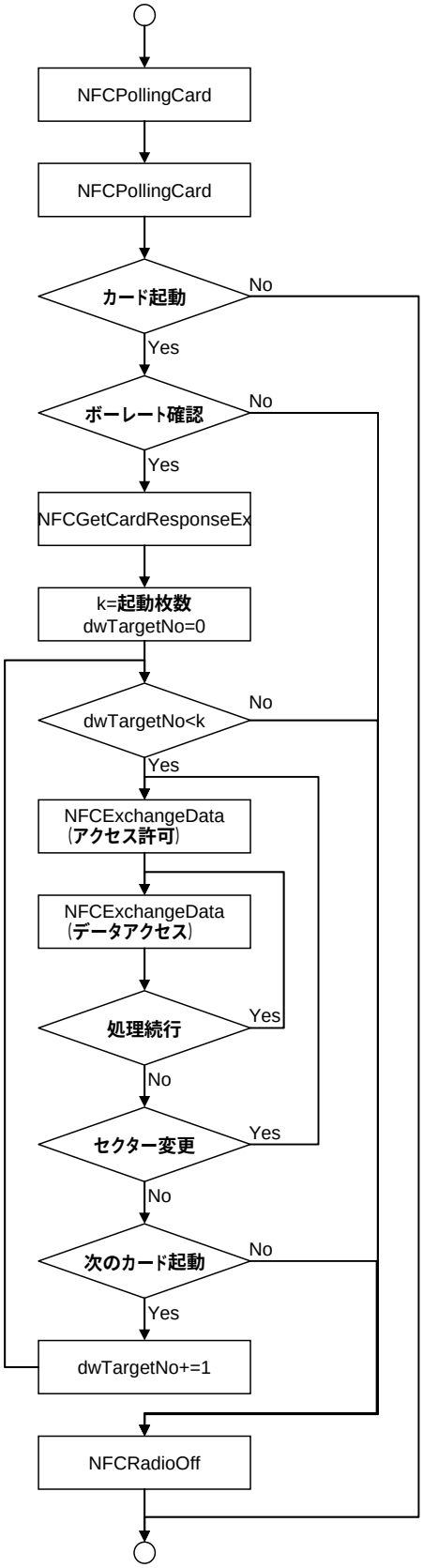
1. NFCSetPollingMode関数により、検索方式に多段起動 (NFC_PLMODE_MULTISTEP2) を、段数に連続起動するカード枚数 CARD_NUM を指定します。
2. iCount=0 をセットします。
3. iCount<CARD_NUM の場合、次の処理に進みます。CARD_NUM は連続起動する IC カードの枚数を表します。
4. NFCPollingCard関数により通信範囲内のカードを検索します。
5. カードの起動に失敗し、NFCPollingCard関数の戻り値が重複起動を表す場合、必要に応じて LED 等により重複起動を通知します。その後、3.に戻って処理を繰り返します。
6. 起動に成功した IC カードのボーレートが Mifare カードのボーレートと一致しているかを確認します。一致している場合は、NFCGetCardResponse関数により Mifare カードの UID を取得します。
7. 次にNFCExchangeData関数により、Mifare コマンドを送信し、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コマンドを送信)
8. セクターを変更せずに処理を続行する場合、7.に戻って処理を繰り返します。
9. セクターを変更して処理を続行する場合、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. 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. NFCSetPollingMode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. NFCPollingCard関数により通信範囲内のカードを検索します。
3. 起動に成功した IC カードのボーレートが Mifare カードのボーレートと一致しているかを確認します。一致している場合は、NFCGetCardResponse関数により Mifare カードの UID を取得します。
4. k に起動した枚数を、dwTargetNo に 0 をセットします。（NFCExchangeData関数の引数）
5. dwTargetNo が k よりも小さい場合、次の処理に進みます。
6. NFCExchangeData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. 次にNFCExchangeData関数により、Mifare コマンドを送信し、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コマンドを送信）
8. セクターを変更せずに処理を続行する場合、7.に戻って処理を繰り返します。
9. セクターを変更して処理を続行する場合、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. NFCRadioOff関数により、電波を停止します。



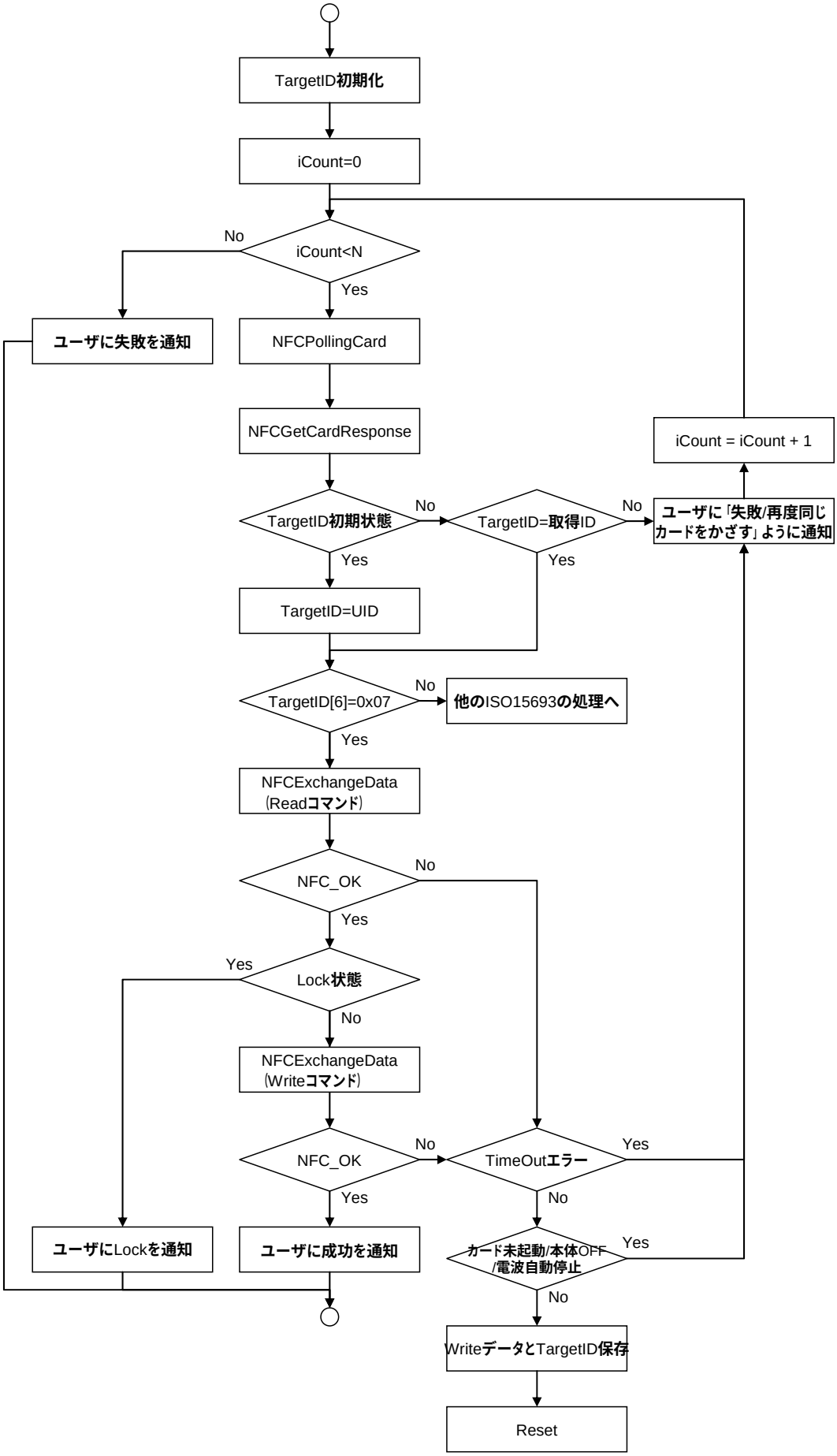
4.4 コマンド送信について

Tag-it にコマンドを送信する

Tag-it に対して、NFCExchangeData関数を使って Write Single Block / Lock Block / Write AFI / Write DSFID コマンドを送信している最中に 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、NFCExchangeData関数の戻り値が NFC_ERROR_TIMEOUT 等のエラーとなった場合は、NFCExchangeData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。

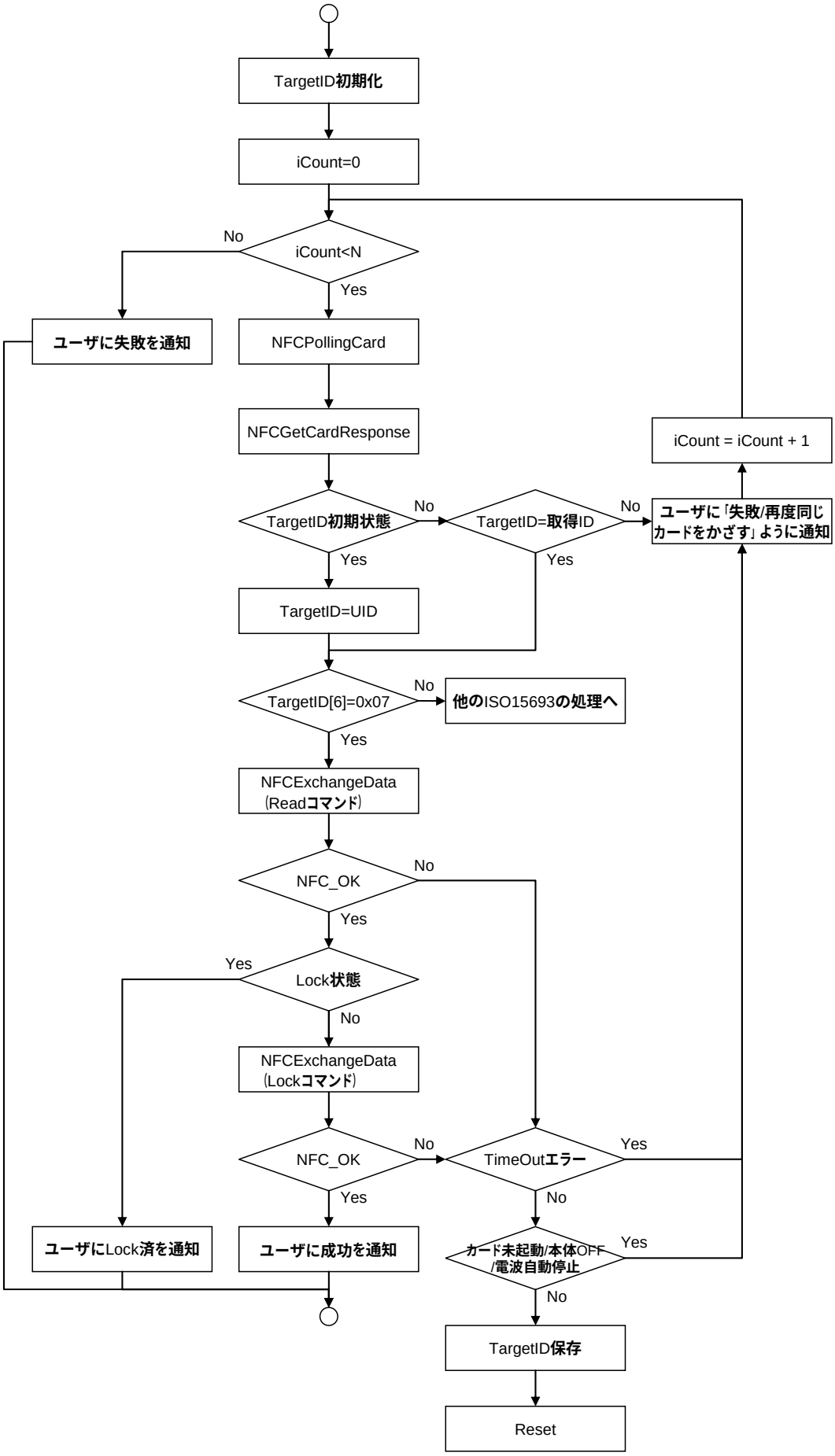
Tag-it に Write Single Block コマンドを送信する

1. TargetID 配列を 0 ｸﾘｱ等で初期化します。
2. iCount=0 をｾｯﾄします。
3. iCount<定数 N の場合、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により、無限ループを回避します。
4. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
5. IC カードの起動に成功した場合は、NFCGetCardResponse関数により IC カードの UID を取得します。
6. TargetID が初期状態の場合は取得した 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 ｼﾘｰｽﾞの IC カードであるため、次の処理に進みます。
8. NFCExchangeData関数により Read コマンドを送信します。（このとき、ISO15693 コマンドの Flag に 0x60 を指定し、対象ﾌﾞﾛｯｸの Lock 状態を確認します）
9. NFCExchangeData関数の戻り値が NFC_OK となった場合は、応答を格納している ReceiveData 配列をﾁｪｯｸします。ReceiveData[1]=0x00 の場合、対象ﾌﾞﾛｯｸは Lock されていないので、次の処理に進みます。
10. NFCExchangeData関数により Write コマンドを送信します。（このとき、ISO15693 コマンドの Flag に 0x60 を指定し、対象ﾌﾞﾛｯｸの Lock 状態を確認します）
11. NFCExchangeData関数の戻り値が NFC_OK の場合は、ｱﾌﾟﾘｹｰｼｮﾝにおいて「書き込みに成功した」旨を表示し、処理を終了します。（正常終了）
12. NFCExchangeData関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF の場合は、書き込みに失敗しているときと、書き込みに成功しているが書き込み後のﾁｪｯｸに失敗しているときがあります。確実に書き込みを完了するためには、ｱﾌﾟﾘｹｰｼｮﾝにおいて「書き込みに失敗/再度同じカードをかざす」旨を表示し、3.に戻って Write Single Block 処理のﾘﾄﾗｲを行います。
13. NFCExchangeData関数の戻り値が上記以外の場合は、Write データと TargetID を保存し、本体ﾘｾｯﾄ後に 1.に戻り、再度 Write Single Block の処理ﾌﾛｰを実行します。このとき、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。


TargetID初期化
iCount=0
iCount<N
No
Yes
ユーザに失敗を通知
NFCPollingCard
NFCGetCardResponse
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
NFCExchangeData
(Readコマンド)
NFC_OK
No
Yes
Yes
Lock状態
No
NFCExchangeData
(Writeコマンド)
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

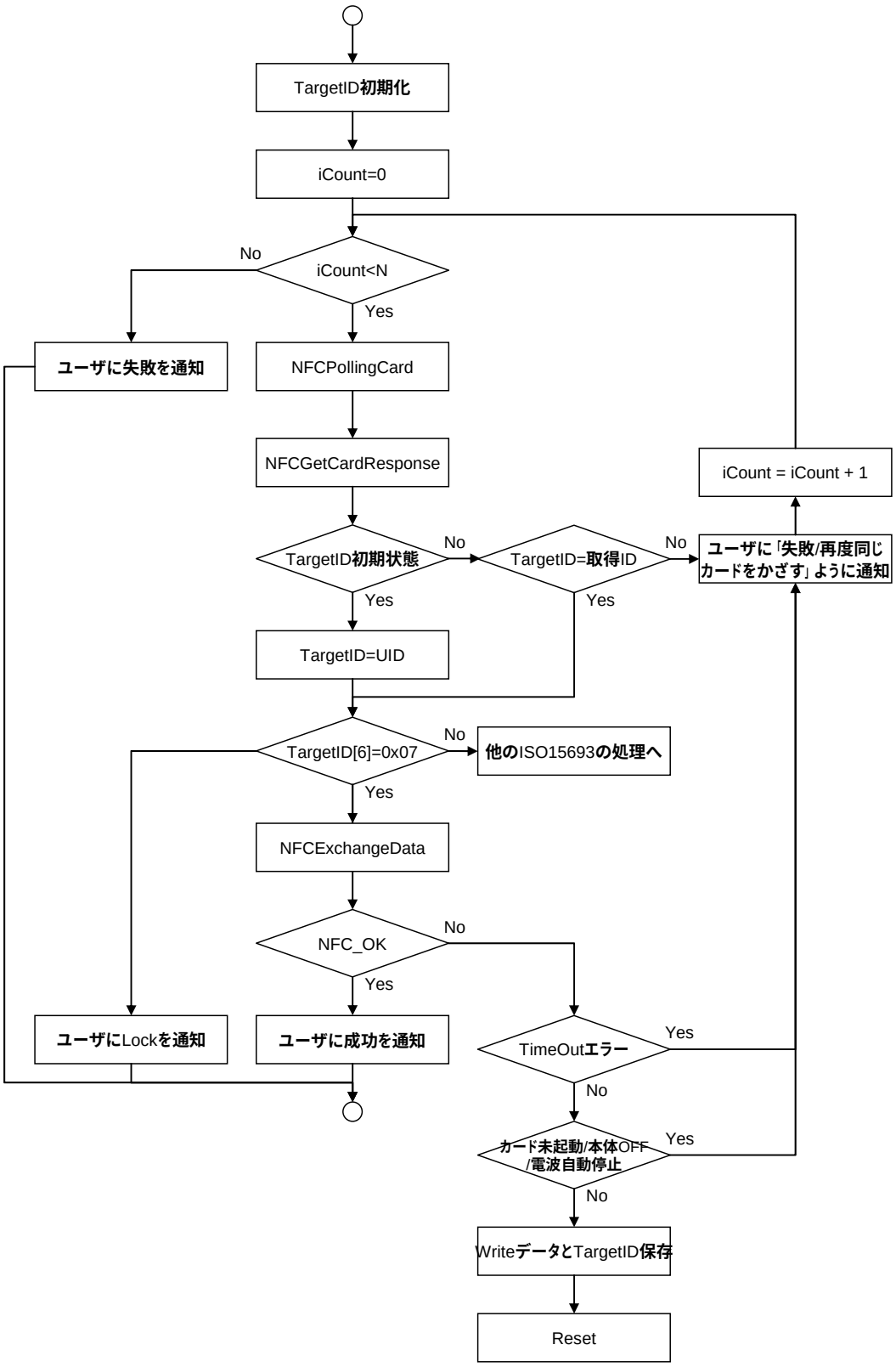
Tag-it に Lock Block コマンドを送信する

1. TargetID 配列を 0 クリア等で初期化します。
2. iCount=0 をセットします。
3. iCount<定数 N の場合、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により、無限ループを回避します。
4. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
5. IC カードの起動に成功した場合は、NFCGetCardResponse関数により IC カードの UID を取得します。
6. TargetID が初期状態の場合は取得した 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 カードであるため、次の処理に進みます。
8. NFCExchangeData関数により Read コマンドを送信します。（このとき、ISO15693 コマンドの Flag に 0x60 を指定し、対象ブロックの Lock 状態を確認します）
9. NFCExchangeData関数の戻り値が NFC_OK となった場合は、応答を格納している ReceiveData 配列をチェックします。ReceiveData[1]=0x00 の場合、対象ブロックは Lock されていないので、次の処理に進みます。
10. NFCExchangeData関数により Lock Block コマンドを送信します。（このとき、ISO15693 コマンドの Flag に 0x60 を指定し、対象ブロックの Lock 状態を確認します）
11. NFCExchangeData関数の戻り値が NFC_OK の場合は、アプリケーションにおいて「Lock に成功した」旨を表示し、処理を終了します。（正常終了）
12. NFCExchangeData関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF の場合は、Lock に失敗しているときと、Lock に成功しているが Lock 後のチェックに失敗しているときがあります。確実に Lock を完了するためには、アプリケーションにおいて「Lock に失敗/再度同じカードをかざす」旨を表示し、3.に戻って Lock Block 処理のリトライを行います。
13. NFCExchangeData関数の戻り値が上記以外の場合は、Write データと TargetID を保存し、本体リセット後に 1.に戻り、再度 Lock Block の処理フローを実行します。このとき、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。


TargetID初期化
iCount=0
iCount<N
No
Yes
ユーザに失敗を通知
NFCPollingCard
NFCGetCardResponse
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
NFCExchangeData
(Readコマンド)
NFC_OK
No
Yes
Yes
Lock状態
No
NFCExchangeData
(Lockコマンド)
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 Write AFI / Write DSFID コマンドを送信する

1. TargetID 配列を 0 クリア等で初期化します。
2. iCount=0 をセットします。
3. iCount<定数 N の場合、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により、無限ループを回避します。
4. NFCPollingCard関数により、通信範囲内の IC カードを検索します。
5. IC カードの起動に成功した場合は、NFCGetCardResponse関数により IC カードの UID を取得します。
6. TargetID が初期状態の場合は取得した 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 カードであるため、次の処理に進みます。
8. NFCExchangeData関数により WriteAFI コマンド（または Write DSFID コマンド）を送信します。（このとき、ISO15693 コマンドの Flag に 0x60 を指定し、対象ブロックの Lock 状態を確認します）
9. NFCExchangeData関数の戻り値が NFC_OK の場合は、アプリケーションにおいて「書き込みに成功した」旨を表示し、処理を終了します。（正常終了）
10. NFCExchangeData関数の戻り値が NFC_ERROR_TIMEOUT、NFC_NOT_ACTIVATION、NFC_ERROR_SUSPEND、NFC_ERROR_AUTOOFF の場合は、書き込みに失敗しているときと、書き込みに成功しているが書き込み後のチェックに失敗しているときがあります。確実に書き込みを完了するためには、アプリケーションにおいて「書き込みに失敗/再度同じカードをかざす」旨を表示し、3.に戻って WriteAFI（または Write DSFID）処理のリトライを行います。
11. NFCExchangeData関数の戻り値が上記以外の場合は、Write データと TargetID を保存し、本体リセット後に 1.に戻り、再度 WriteAFI（または Write DSFID）の処理フローを実行します。このとき、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。


TargetID初期化
iCount=0
iCount<N
No
Yes
ユーザに失敗を通知
NFCPollingCard
NFCGetCardResponse
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
NFCExchangeData
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/

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)