人にやさしく、地球にやさしく
Thoughtfully, Living with the Earth
WELCOM DESIGN
http://www.e-welcom.com/
ISO14001Certified

CIPHER LAB

C/C++言語プログラミング

for 9200 Mobile Computers

英文･和文で相違がある場合は、英文を優先して解釈をお願いします

Version 1.02

人にやさしく、地球にやさしく
Thoughtfully, Living with the Earth
WELCOM DESIGN
http://www.e-welcom.com/
ISO14001Certified

ウェルコムデザイン株式会社
本社 神戸市西区井吹台東町1-1-1 西神南センタービル 〒651-2242
Phone. 078-993-6010(代) Fax. 078-993-6020
東京 東京都文京区湯島3-14-9 湯島ビル 〒113-0034
Phone. 03-3836-9411(代) Fax. 03-3836-9412

改訂記録	
改訂番号	改訂日
Rev.1.0	May. 2015（初版）

目 次

Blank page

はじめに

CipherLab 製品をご利用いただきありがとうございます。
このプログラミングガイドは、リーダモジュールがデータをキャプチャーしたり、Windows Embedded Handheld 6.5 多機能デバイスを搭載した 9200 モバイルコンピュータを制御したりするアプリケーションを構築するためのガイドです。
専用の Reader Configuration ユーティリティでキャプチャーされたデータは、Windows Message または Windows Event でデコードできます。よりフレキシブルなプログラミングのために、さまざまなリーダの機能を実装するためのリーダ DLL(ダイナミックリンクライブラリ)が当該モバイルコンピュータとあわせて提供されています。
開発するアプリケーションでモバイルコンピュータ上のハードウェアモジュールを制御するためのシステム機能は、ここに含まれています。
9200 プラットフォーム上のアプリケーション開発を容易にするためにも、このガイドのコピーを手元に置いておくことをお勧めします。

開発ツール

ここに記述されている API を使用するには、Windows プログラミング開発の知識が必要です。Windows のアプリケーションプログラミングインターフェイス（API）の詳細については、Microsoft の Web サイトをご参照ください。

http://msdn2.microsoft.com/en-us/library/default.aspx

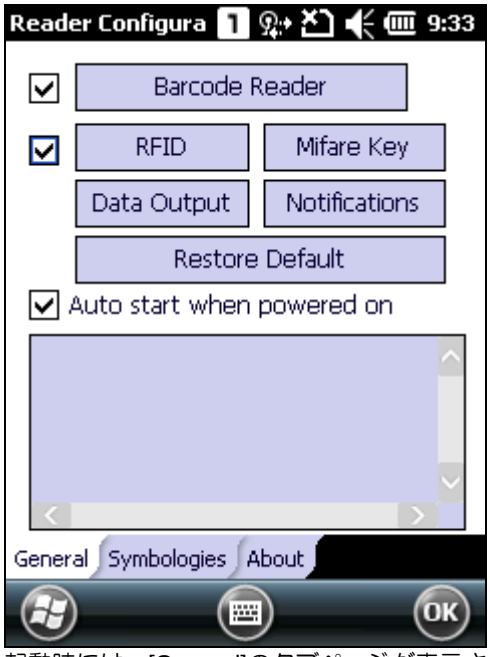
9200 モバイルコンピュータは Windows Embedded Handheld 6.5 を搭載しており、開発環境には以下のものが必要です。

- Visual Studio 2005、または Visual Studio 2008
- 9200 SDK

1　Reader Configurations

Reader Configuration はリーダモジュールを設定するためにあらかじめインストールされているユーティリティです。使用するには

(1) スタート画面で、[設定]→[システム]→[Reader Configurations]  とタップします。
モバイルコンピュータ上の使用可能なリーダの設定が表示されます。



起動時には、[General]のタブページが表示されます。[Symbologies]と[About]はタブのみが表示されています。

(2) [General]タブのページでは使用するリーダ(複数可)を有効にします。[Symbologies]のページではバーコード読取りを設定します。

(3) [General]タブで[Data OutPut]をタップすると、デコード設定の画面が表示されます。

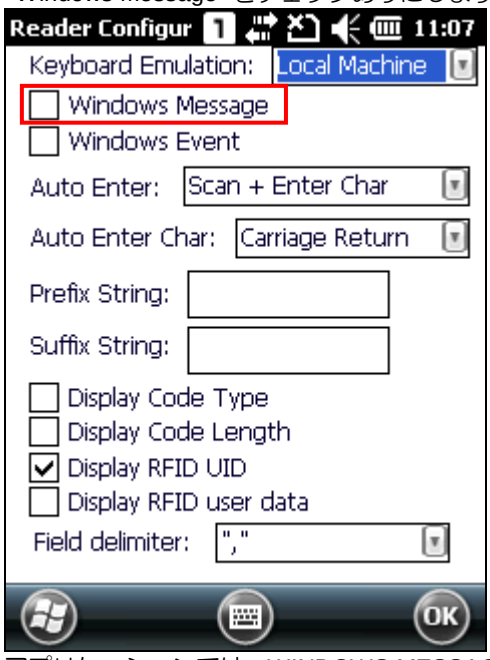
(4) Keyboard Emulation、Windows Message、または Windows Event を選択することにより、データの出力先を設定します。

設定	説明
Keyboard Emulation	まるでタイプしたかのようにアクティブなアプリケーションにデータをデコードします。データを入力するアプリケーションまたは任意のテキストエディタを起動してください。
Windows Message	データをデコード後、アプリケーションに Windows Message を送信します。 詳細は、『1.1 Windows Message』を参照してください。
Windows Event	データをデコード後、アプリケーションに Windows Event を送信します。 詳細は、『1.2 Windows Event』を参照してください。

1.1 WINDOWS MESSAGE

読取ったデータをアプリケーションで WINDOWS MESSAGE として使用します。

(1) スタート画面で、[設定]→[システム]→[Reader Configurations]とタップし、表示された画面で[Data Output]をタップします。
(2) <Windows Message>をチェックありにします。



アプリケーションでは、WINDOWS MESSAGE を受信できるようにしておいてください。

- WM_DECODE に WINDOWS MESSAGE を登録します。
  DecodeMsg = RegisterWindowMessage(TEXT("WM_DECODE"));
  データがデコードされると、WINDOWS MESSAGE 通信の識別子とデータが WINDOWS MESSAGE キューに格納されます。
- WM_DECODE からデコードされたデータを取得します。
- デコードされたデータを取得するには Windows API の ReadMsgQueue をコールします。
  ReadMsgQueue(hQueue,&buffer,sizeof(buffer),&dwRW,0,&dwFlag);

1.1.1 デコードデータ構造体 - COPYDATASTRUCT -

データはヘッダファイル" winuser.h"にある COPYDATASTRUCT 構造体に格納されます。

```
Typedef struct tagCOPYDATASTRUCT {
        DWORD           dwData;
        DWORD           cbData;
        PVOID           lpData;
} COPYDATASTRUCT, *PCOPYDATASTRUCT;
```

1.1.2 メッセージキュー

ReaderConfigQueue という名前のキューを次の例のように作成します。

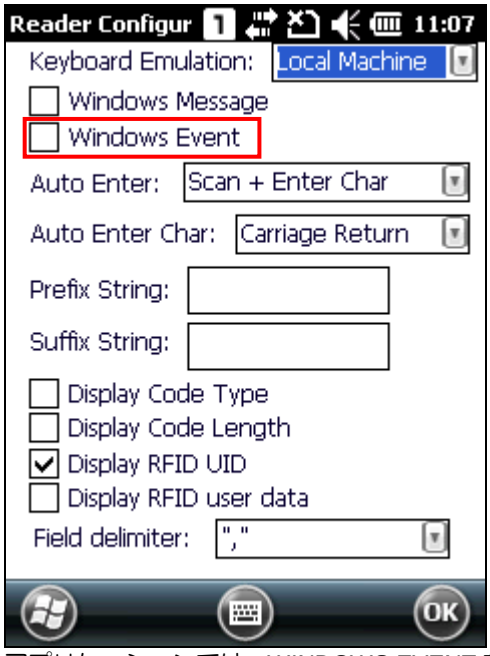
```
HANDLE                  hQueue;
COPYDATASTRUCT          buffer;
MSGQUEUEOPTIONS         QueueOptions;

QueueOptions.dwSize = sizeof(QueueOptions);
QueueOptions.dwMaxMessages = 1;
QueueOptions.cbMaxMessage = sizeof(COPYDATASTRUCT);
QueueOptions.bReadAccess = TRUE;
hQueue = CreateMsgQueue(TEXT("ReaderConfigQueue"),&QueueOptions);
```

1.2 WINDOWS EVENT

読取ったデータをアプリケーションで WINDOWS EVENT として使用します。

(1) スタート画面で、[設定]→[システム]→[Reader Configurations]とタップし、表示された画面で[Data Output]をタップします。
(2) <Windows Event>をチェックありにします。



アプリケーションでは、WINDOWS EVENT を受信できるようにしておいてください。

- WINDOWS EVENT オブジェクトの DecodeEvent を作成します。
  hEvent = CreateEvent(NULL,FALSE,FALSE,TEXT("DecodeEvent"));
  データがデコードされると、WINDOWS EVENT オブジェクトがシグナル状態となり、WINDOWS MESSAGE キューに格納されます。
- DecodeEvent からデコードイベントを取得します。
- デコードされたデータを検索するには Windows API の ReadMsgQueue をコールします。
  ReadMsgQueue(hQueue,&DecodeMsg,sizeof(DecodeMsg),&dwRW,0,&dwFlag);

1.2.1 デコードデータ構造体 - DECODEMSG -

データは DECODEMSG 構造体に格納されます。

```
Typedef struct tagDecodeMsg {
        BYTE        cDeviceType;
        BYTE        cCodeType[4];
        INT         nCodeLen;
        BYTE        szRFID_UID[21];
        BYTE        Reserve[100];
        BYTE        szCodeData[4096+1];
} DECODEMSG, *LPDECODEMSG;
```

1.2.2 メッセージキュー

ReaderConfigQueue という名前のキューを次の例のように作成します。

```
HANDLE                  hQueue;
DECODEMSG               DecodeMsg;
MSGQUEUEOPTIONS         QueueOptions;

QueueOptions.dwSize = sizeof(QueueOptions);
QueueOptions.dwMaxMessages = 1;
QueueOptions.cbMaxMessage = sizeof(DECODEMSG);
QueueOptions.bReadAccess = TRUE;
hQueue = CreateMsgQueue(TEXT(“ReaderConfigQueue”),&QueueOptions);
```

2　リーダ DLL

ダイナミックリンクライブラリ(DLL)は、アプリケーション、または別の DLL で便利なﾘﾝｸ可能な機能とデータをモジュールとして提供しています。明示的にリンクすることで、アプリケーションまたは別の DLL は実行時にエクスポートされた DLL 関数の情報のみを組み込むことができます。
DLL はオペレーティングシステムではなくアプリケーションによってロードされます。そして、アプリケーションロード時ではなく、実行時にロードされます。ロードする DLL がない場合、LoadLibrary()関数が失敗します。
LoadLibrary()の返したエラーをチェックし、それに応じて処理するアプリケーションがあります。DLL のロードに成功すると、アプリケーションは GetProcAddress()をコールし、エクスポートされた DLL 関数のアドレスを取得する必要があります。
リーダ DLL はリーダアプリケーション上の簡単な制御を実現するために前記のメカニズムに基づいて機能します。ご使用にあたって、下記の必要なファイルを CD-ROM から取得してください。

- ReaderDllMobile.dll
- ReaderDll.h

2.1 コンパイル方法

ヘッダファイルは、C ソースコードのディレクトリに置いておく必要があります。LoadLibrary()と FreeLibrary()のある DLL のパスを指定します。

(1) プロセスで、実行時に DLL をロードするために LoadLibrary()をコールする必要があります。
パラメータを DLL のファイル名のみとすると、相対ディレクトリパスとなります。
例）
HINSTANCE hDllInst = LoadLibrary("ReaderDllMobile.dll");

この場合、関数は、アプリケーションと同一のフォルダにある DLL をロードします。これは、アプリケーションと DLL を同じフォルダに配置することを意味します。
パラメータをフルパスとすると、モバイルコンピュータストレージ上の DLL の絶対ディレクトリパスとなります。
例）
HINSTANCE hDllInst = LoadLibrary(“\\Windows\\ReaderDllMobile.dll”);

この場合、DLL ファイルは、”Windows”フォルダにあるものとなります。

(2) リーダ DLL ロード後、プロセスで DLL 内のエクスポートされた関数のアドレスを取得するために GetProcAddress()をコールします。GetProcAddress()によって返された関数ポインタを使用してエクスポートされた DLL 関数を呼び出します。
例）
InitReader = (INITREADER)GetProcAddress(hDllInst,TEXT(“InitReader”));
InitReader は関数へのポインタ。

(3) プロセス終了時または、アプリケーション内で DLL を使用しなくなる場合は、DLL をアンロードし、リーダモジュールが電源オフにしてリソースを開放できるように、FreeLibrary()をコールする必要があります。

[注]：LoadLibrary()、FreeLibrary()でフルパスを指定する場合、指定のパスにロードする DLL をコピーしておいてください。

2.2 実行方法

アプリケーションの実行方法です。

(1) アプリケーションの実行イメージ(.exe)をモバイルコンピュータの内部ストレージにコピーします。内部ストレージの空き容量が不足している場合は、外部ストレージにコピーします。

[注]：外部ストレージにアプリケーションを置く場合、ロードする DLL のパスに注意してください。

(2) アプリケーションの実行イメージ(.exe)を実行します。
アプリケーションは LoadLibrary()で指定されたパスを参照し、DLL をロードします。
例）
HINSTANCE hDllInst = LoadLibrary("\\Windows\\ReaderDllMobile.dll");

アプリケーションは、FreeLibrary()で指定された DLL をアンロードし終了します。

2.3 デコードメッセージの登録

デコードされたデータを取得するには、アプリケーションでメッセージ WM_DECODEDATA を登録する必要があります。
例）
UNIT nDecodeMsg = RegisterWindowMessage(TEXT("WM_DECODEDATA"));

この例の場合、nDecodeMsg に 1D(レーザー)リーダまたは RFID リーダがデータをデコードし、ブロードキャストする際のメッセージ識別子が格納されます。
データがデコードされるたびに以下のようになります。

- 1D(レーザー)リーダまたは RFID リーダがデータをデコードすると、WM_DECODEDATA メッセージがブロードキャストされます。
- アプリケーションは、メッセージ WM_DECODEDATA を取得し、デコードされたデータを識別するための wParam を使用します。

  例）

```
if(wParam == DC_READER_BC) {
        GetDecodeType()
        GetDecodeData()
        GetOutputRecord()
}
else if(wParam == DC_READER_RFID) {
        GetDecodeTagType()
        GetDecodeRfidUID()
        GetDecodeRfidData()
        GetOutputRecord()
 }
```

[注]：ReadBarcodeData()と ReadRfidData()は、GUI のボタンをタップしてスキャンしたデータを WM_DECODEDATA から取得することなく、スキャンして取得します。
それらは、クリックイベントハンドラ関数内の関数を呼び出す必要があります。

3　リーダ API

リーダ DLL は、作成するアプリケーションで利用するための関数を多数提供しており、Windows アプリケーションプログラミングインターフェイス(API)を実装します。ライブラリルーチンの関数プロトタイプはライブラリのヘッダファイルに宣言されています。InitReader()で処理を開始してください。
アプリケーションのコーディングを始める前に必要なファイルを CD-ROM から取得してください。

必須ヘッダファイル	必須 Lib ファイル	必須 DLL
ReaderDll.h	該当なし	ReaderDllMobile.dll

3.1　リーダの初期化／確認

3.1.1　初期化

InitReader

目的　リーダモジュールを初期化します。

書式　INT InitReader(VOID);

使用例
```
INT nRlt;
nRlt = InitReader();
```

戻り値　0=成功、それ以外=エラー。

-102	E_READER_INIT_FAILED
-103	E_NO_READER_INSTALLED
-111	E_READER_BC_RESET_FAILED
-112	E_READER_RFID_RESET_FAILED (下記備考を参照)

備考　この関数は、バーコードリーダを準備状態にするためのものであり、他の関数より先にコールする必要があります。

- リーダ設定にアクセスできなかった場合、リーダは自動的にﾘｾｯﾄされます。
- GetReaderType()をコールすることで、モバイルコンピュータに搭載されているリーダモジュールをチェックすることができます。

参照　GetActiveDevice, GetReaderType, SetActiveDevice

3.1.2　デバイス起動

GetActiveDevice

目的　現在起動中のリーダを取得します。

書式　INT GetActiveDevice(VOID);

使用例
```
INT nRdrType;
nRdrType = GetActiveDevice();
```

戻り値　取得に成功した場合、以下のリーダ種別を返します。

7	DC_READER_BC	バーコードリーダのみが起動中。
32	DC_READER_RFID	RFID リーダのみが起動中。
255	ALL_DEVICE	両方のリーダが起動中。

取得に失敗した場合は、エラーコードを返します。

0	NO_ACTIVE_DEVICE
-101	E_READER_NOT_INIT

備考　リーダが起動中なのは次の場合です。

- 1D または 2D のバーコードリーダのみが起動中である。
- RFID リーダのみが起動中である。
- バーコードリーダと RFID リーダの両方が起動中である。

参照　GetReaderType, InitReader, SetActiveDevice

SetActiveDevice

目的 リーダを起動します。

書式 INT GetActiveDevice(INT actDC);

引数 actDC
[入力]起動するデバイス。

0	NO_ACTIVE_DEVICE	起動中のリーダをすべて停止。
7	DC_READER_BC	バーコードリーダのみを起動。
32	DC_READER_RFID	RFID リーダのみを起動。
255	ALL_DEVICE	両方のリーダを起動。

使用例
```
INT nRit;
nRit = SetActiveDevice(ALL_DEVICE);
```

戻り値 0=成功、それ以外=エラー。

-101	E_READER_NOT_INIT

備考 トリガキーは起動中のスキャナのみスキャンします。起動中でないリーダはイベントが発生しません。
指定されたリーダ種別が存在しない場合、エラーが発生しません。

参照 GetReaderType, GetActiveDevice, InitReader

3.1.3 リーダ種別

GetReaderType

目的 搭載されているリーダの種別を取得します。

書式 INT GetReaderType(VOID);

使用例
```
INT nRdrType;
nRdrType = GetReaderType();
```

戻り値 取得に成功した場合、以下のリーダ種別(論理和)を返します。

128	ID_MOD_1D_955
256	ID_MOD_MP_RFID
512	ID_MOD_2D_4500
1024	ID_MOD_1D_SM1
384	ID_MOD_1D_955 + ID_MOD_MP_RFID
768	ID_MOD_2D_4500 + ID_MOD_MP_RFID
1280	ID_MOD_1D_SM1 + ID_MOD_MP_RFID

取得に失敗した場合は、エラーコードを返します。

0	(1)リーダが初期化されていない。 (2)リーダがインストールされていない。

備考 リーダの搭載パターンは次の 3 種です。
- 1D または 2D のバーコードリーダのみである。
- RFID リーダのみである。
- バーコードリーダと RFID リーダの両方がある。

参照 InitReader, GetActiveDevice, SetActiveDevice

3.2 データ取得

3.2.1 データ出力設定

処理データ

DataOutputSettings()は、デコードされたバーコードデータにアタッチするための情報をセットします。

情報	長さ(Byte)	解説
コード種別	1	バーコード種別。DataOutputSettings()のパラメータ showCodeType を参照してください。
コードの長さ	4	デコードされたバーコードデータの長さ(プリフィックス／サフィックスコードを除く)。DataOutputSettings()のパラメータ showCodeLen を参照してください。
プリフィックスコード	0～10	プリフィックスコードがない場合、値は 0 となります。 DataOutputSettings()のパラメータ prefixCode と prefixCodeLen を参照してください。
デコードデータ	1～4096	デコードされたバーコードデータ。
サフィックスコード	0～10	サフィックスコードがない場合、値は 0 となります。 DataOutputSettings()のパラメータ suffixCode と suffixCodeLen を参照してください。

- キーボードエミュレーションを有効にすると、バーコードデータは、アクティブなアプリケーションのフォーカス部に出力されます。
- キーボードエミュレーションが無効になっている場合、メッセージ WM_DECODEDATA 受信時に処理されたデータを取得するためには GetOutputRecord()を使用します。『2.3 デコードメッセージの登録』を参照してください。

[注]：シーケンスによるデータフィールドが含まれています。
[コード種別]、[コードの長さ]、[プリフィックスコード]、[デコードデータ]、[サフィックスコード]

DataOutputSettings()と RfidSetting()は、デコードされた RFID データにアタッチするための情報をセットします。

情報	長さ(Byte)	解説	
タグ種別	1	RFID タグ種別。DataOutputSettings()のパラメータ showCodeType を参照してください。	
コードの長さ	4	デコードされた RFID データの長さ(プリフィックス／サフィックスコードを除く)。DataOutputSettings()のパラメータ showCodeLen を参照してください。	
		タグの長さ	➢ UID のみ ➢ データのみ ➢ UID + データ
プリフィックスコード	0～10	プリフィックスコードがない場合、値は 0 となります。 DataOutputSettings()のパラメータ prefixCode と prefixCodeLen を参照してください。	
タグ UID	20	RFID タグのユニバーサル識別(UID)。RfidSettings()のパラメータ readUid を参照してください。	
サフィックスコード	0～10	サフィックスコードがない場合、値は 0 となります。 DataOutputSettings()のパラメータ suffixCode と suffixCodeLen を参照してください。	
区切り文字	1	UID とデータを区切る文字。RfidSettings()のパラメータ useDelim を参照してください。 0x00 は区切り文字なしを意味します。	
タグデータ	1～4096	デコードされた RFID データ。RfidSettings()のパラメータ readUserData を参照してください。	

- キーボードエミュレーションを有効にすると、RFID データは、アクティブなアプリケーションのフォーカス部に出力されます。
- キーボードエミュレーションが無効になっている場合、メッセージ WM_DECODEDATA 受信時に処理されたデータを取得するためには GetOutputRecord()を使用します。

[注]：シーケンスによるデータフィールドが含まれています。
[タグ種別]、[タグの長さ]、[プリフィックスコード]、[タグ UID]、[サフィックスコード]、[区切り文字]、[タグデータ]
UID が無効の場合、出力データにプリフィックスコード、UID、サフィックスコード、区切り文字は含まれません。

生データ

生データを取得するには次の関数を使用します。

- GetDecodeData()と GetDecodeType()
- GetDecodeRfidData()、GetDecodeRfidUID()と GetDecodeTagType()

DataOutputSettings

目的　出力データの出力先と方法をセットします。

- 「出力方法」とは、コード／タグ種別、データの長さ、プリフィックスコード、サフィックスコード、等のようなデコードされたデータにアタッチするための情報を意味します。

書式

```
INT DataOutputSettings (INT rw,
                        INT *enableKeyboardEmulation,
                        INT *autoEnterWay,
                        INT *autoEnterChar,
                        INT *showCodeType,
                        INT *showCodeLen,
                        CHAR *prefixCode,
                        INT prefixCodeSize,
                        CHAR *suffixCode,
                        INT suffixCodeSize);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	出力設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	出力設定書込み。

enableKeyboardEmulation

[入/出力] 入力テキストとしてデータをエミュレートし、アクティブなアプリケーションに渡すかどかを指定する値。

0(※)	キーボードエミュレーション無効。 ➢ GetOutputRecord()がコールされない限り、他のパラメータは有効になりません。
1	ローカルマシン上のキーボード入力をエミュレート。
2	リモート PC 上のキーボード入力をエミュレート。(リモートデスクトップ接続、ターミナルサーバクライアント、などのプログラム経由の RDP サーバ)

autoEnterWay

[入/出力] デコード後の文字の自動接辞。

0	無効。
1(※)	デコードデータの末尾に追加。(デコードデータ+入力文字)
2	デコードデータの先頭に追加。(入力文字+デコードデータ)

autoEnterChar

[入/出力]自動接辞する文字。

0	なし。
1(※)	CR。(= 0x0d)
2	タブ
3	スペース(=0x20)
4	カンマ(=0x2c)
5	セミコロン(=0x3b)

showCodeType

[入/出力] データレコード内のバーコード種類または RFID タグ種別の送信。

0(※)	送信しない。
1	送信する。

showCodeLen

[入/出力] データレコード内のバーコードまたは RFID タグのコードの長さの送信。

0(※)	送信しない。
1	送信する。

prefixCode
[入/出力] プリフィックスコードが格納されているバッファのポインタ。
➢ RFID デコードでは、出力にプリピックスコード、UID、サフィックスコードを含むので UID は有効である必要があります

prefixCodeSize
[入/出力] プリフィックスコードのサイズ。

sufixCode
[入/出力] サフィックスコードが格納されているバッファのポインタ。
➢ RFID デコードでは、出力にプリピックスコード、UID、サフィックスコードを含むので UID は有効である必要があります。

sufixCodeSize
[入/出力] サフィックスコードのサイズ。

戻り値　0=成功、1=エラー。

備考　リーダの種類と関連するリーダの設定によっては、出力レコードのフィールドが異なる場合があります。
バーコードリーダからデータを取得したとき、データフィールドが含まれる場合があります。
[コード種別]、[コードの長さ]、[プリフィックスコード]、[デコードデータ]、[サフィックスコード]

コード種別	showCodeType の値が 0 でない場合のみ出力されます。
コードの長さ	showCodeLen の値が 0 でない場合のみ出力されます。 (プリフィックス／サフィックスコードは含まれません。)
プリフィックスコード	prefixCode の値が 0 でない場合のみ出力されます。
デコードデータ	データがデーコードされた場合のみ出力されます。
サフィックスコード	sufixCode の値が 0 でない場合のみ出力されます。

RFID リーダからデータを取得したとき、データフィールドが含まれる場合があります。
➢ UID が無効の場合、出力データにプリフィックスコード、UID、サフィックスコード、区切り文字は含まれません。
[タグ種別]、[タグの長さ]、[プリフィックスコード]、[タグ UID]、[サフィックスコード]、[区切り文字]、[タグデータ]

タグ種別	showCodeType の値が 0 でない場合のみ出力されます。
タグの長さ	showCodeLen の値が 0 でない場合のみ出力されます。
プリフィックスコード	prefixCode の値が 0 でない場合のみ出力されます。
タグ UID	RfidSettings()の引数 readUID の値が 0 でない場合のみ出力されます。
サフィックスコード	sufixCode の値が 0 でない場合のみ出力されます。
区切り文字	RfidSettings()の引数 useDelim の値が 0 でない場合のみ出力されます。
タグデータ	RfidSettings()の引数 readData の値が 0 でない場合のみ出力されます。

参照　GetDecodeData, GetDecodeType, GetOutputRecord
GetDecodeRfidData, GetDecodeRfidUID, GetDecodeTagType, RfidSettings

GetOutputRecord

目的

キーボードエミュレーションのデータ出力方法を設定します。

- 設定とは、コード／タグ種別、データの長さ、プリフィックスコード、サフィックスコード、等のようなデコードされたデータにアタッチするための情報を意味します。

書式

```
INT GetOutputRecord (INT target,
                     CHAR *buf,
                     INT bufsize)
```

引数

target

[入力]デーコードデータの取得元。

7	DC_READER_BC	バーコードリーダから取得。
32	DC_READER_RFID	RFID リーダから取得。

buf

[入/出力] デコードデータが格納されるバッファへのポインタ。

bufsize

[入力] 最大受信バイト数。通常はバッファサイズと同じ値をセットします。

使用例

```
CHAR buf[2048] = {'\0'};
INT nRlt;
nRlt = GetOutputRecord(DC_READER_BC, buf, sizeof(buf));
nRlt = GetOutputRecord(DC_READER_RFID, buf, sizeof(buf));
```

戻り値

成功すると、取得したバイト数を返します。(NULL 終端文字を除く)

失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

備考

この関数は、次のデータでコード前にコールされます。リーダの種類と関連するリーダの設定によって、出力が異なる場合があります。

バーコードリーダからデータを取得したとき、データフィールドが含まれる場合があります。

[コード種別]、[コードの長さ]、[プリフィックスコード]、[デコードデータ]、[サフィックスコード]

コード種別	DataOutputSettings()の引数 showCodeType の値が 0 でない場合のみ出力されます。
コードの長さ	DataOutputSettings()の引数 showCodeLen の値が 0 でない場合のみ出力されます。 (プリフィックス／サフィックスコードは含まれません。)
プリフィックスコード	DataOutputSettings()の引数 prefixCode の値が 0 でない場合のみ出力されます。
デコードデータ	データがデーコードされた場合のみ出力されます。
サフィックスコード	DataOutputSettings()の引数 sufixCode の値が 0 でない場合のみ出力されます。

RFID リーダからデータを取得したとき、データフィールドが含まれる場合があります。

- UID が無効の場合、出力データにプリフィックスコード、UID、サフィックスコード、区切り文字は含まれません。

[タグ種別]、[タグの長さ]、[プリフィックスコード]、[タグ UID]、[サフィックスコード]、[区切り文字]、[タグデータ]

タグ種別	DataOutputSettings()の引数 showCodeType の値が 0 でない場合のみ出力されます。
タグの長さ	DataOutputSettings()の引数 showCodeLen の値が 0 でない場合のみ出力されます。 (プリフィックス／サフィックスコードは含まれません。)
プリフィックスコード	DataOutputSettings()の引数 prefixCode の値が 0 でない場合のみ出力されます。

タグ UID	RfidSettings()の引数 readUID の値が 0 でない場合のみ出力されます。
サフィックスコード	DataOutputSettings()の引数 sufixCode の値が 0 でない場合のみ出力されます。
区切り文字	RfidSettings()の引数 useDelim の値が 0 でない場合のみ出力されます。
タグデータ	RfidSettings()の引数 readData の値が 0 でない場合のみ出力されます。

参照　DataOutputSettings, GetDecodeData, GetDecodeType, GetDecodeRfidData, GetDecodeRfidUID, GetDecodeTagType, RfidSettings

3.2.2 リーダ設定

Rfidettings

目的

RFID タグからの収集方法と出力方法を設定します。

- 「RFID の出力方法」とは、データの長さ、プリフィックスコード、タグ UID、サフィックスコード、区切り文字等のようなデコードされたデータにアタッチするための情報を意味します。

書式

```
INT RfidSettings (INT rw,
                  INT *readUid
                  INT *readUserData,
                  INT *userDataStartByte,
                  INT *readUserDataLen,
                  INT *useDelim,
                  INT *timeout,
                  CHAR *mifareLoginKey,
                  INT loginKeyLength,
                  INT *loginKeyType);
```

引数

※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	出力設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	出力設定書込み。

readUid

[入/出力]UID のデコード。

0	デコードしない。
1(※)	デコードする。

readUserData

[入/出力] エンコードデータのデコード。

0	デコードしない。
1(※)	デコードする。

userDataStartByte

[入/出力]収集データ開始位置。

RFID タグによって容量は異なりますが、このパラメータで容量を超える値を渡しても、タグの最大容量に自動調節されます。

-1	収集データ開始位置を指定します。”-1”を指定した場合、RFID リーダは RFID タグのデフォルトページから読込みを開始します。 RFID タグのデフォルトページは次の通りです。

タグ種別	規格	デフォルトページ
Mifare	(ISO 14443A)	4
ICODE SLI	(ISO 15693)	3
LRI512	(ISO 15693)	0
SRF55VxxP	(ISO 15693)	3
Tag-it	(ISO 15693)	0
Others	(ISO 15693)	0
ICODE	(Phillips)	5

readUserDataLen

[入/出力] データの収集サイズ。デフォルトは 128 バイトです。RFID タグによって容量は異なりますが、このパラメータで容量を超える値を渡しても、タグの最大容量に自動調節されます。

useDelim

[入/出力] 使用する区切り文字()ASCII 値)。

0(※)	区切り文字なし。
1～127	デコード時に UID とデータの間に追加される区切り文字。

timeout

[入/出力] ｽｷｬﾝセッション終了までの時間(秒)。

0～5	デフォルトは 3(秒)。

mifareLoginKey

[入/出力] リーダ DLL で(Mifare などの)セキュリティーキーが格納されているバッファへのポインタ。特定の RFID にアクセスするための認証キー。デフォルトは”FFFFFFFFFFFF”。

- このキーは 12 バイトの長さの 16 進文字列である必要があります。
- このキーは"userDataStartByte"、” readUserDataLen”で指定された対象範囲のすべてのセクタに適用されます。

0	(現在の文字列を保持)。
“F1F2F3F4F5F6”	F1F2F3F4F5F6。

loginKeyLength

[入力] 現在のキー文字列を上書きする場合、このパラメータを無視するように、NULL を渡します。

リーダ DLL に格納されている現在のキー文字列を取得する場合は、入力バッファの長さを渡します。

- この関数はで、Mifare カードのセキュリティキー(キーA または B キー)を変更することはできません。ChangeMifareKey()を使用してください。

loginKeyType

[入/出力] ログインキー種別。

0(※)	キーA。
1	キーB。

戻り値

0=成功、それ以外=エラー。

-253	E_WRONG_READER_TYPE

備考

データフィールドに含まれています。

[タグ種別]、[タグの長さ]、[プリフィックスコード]、[タグ UID]、[サフィックスコード]、[区切り文字]、[タグデータ]

タグ種別	DataOutputSettings()の引数 showCodeType の値が 0 でない場合のみ出力されます。
タグの長さ	DataOutputSettings()の引数 showCodeLen の値が 0 でない場合のみ出力されます。
プリフィックスコード	DataOutputSettings()の引数 prefixCode の値が 0 でない場合のみ出力されます。
タグ UID	RfidSettings()の引数 readUID の値が 0 でない場合のみ出力されます。
サフィックスコード	DataOutputSettings()の sufixCode の値が 0 でない場合のみ出力されます。
区切り文字	RfidSettings()の引数 useDelim の値が 0 でない場合のみ出力されます。
タグデータ	RfidSettings()の引数 readData の値が 0 でない場合のみ出力されます。

- UID が無効の場合、出力にプリフィックスコード、UID、サフィックスコードは含まれませんので注意してください。

参照

ChangeMifareKey, GetDecodeRfidData, GetDecodeRfidUID GetDecodeTagType, DataOuputSettings, GetOutputRecord

[注]：
(1)スキャンセッションはトリガキーを押すと始まります。
(2)タイムアウトの値は、トリガキー押す時間より長く設定してください。
(3)RFID タグは、Mifare スタンダード 1K/4K などのようなセキュリティ対策の認証をサポートしています。

ChangeMifareKey

目的　Mifare カードのセキュリティーキー(特定のセクタにあるキーA またはキーB)を変更します。

書式

```
INT ChangeMifareKey (INT loginKeyType,
                     INT useDefaultKey,
                     INT defaultKeyIndex,
                     INT targetSector,
                     CHAR *loginKey,
                     CHAR *newKeyA,
                     CHAR *newKeyB,
                     CHAR *accessControl);;
```

引数

loginKeyType(将来)
[入力] NULL を指定してください。

useDefaultKey(将来)
[入力] NULL を指定してください。

defaultKeyIndex(将来)
[入力] NULL を指定してください。

targetSector
[入力] Mifare のセキュリティーキーを変更するセクタ。

loginKey
[入力] 変更前キーA。

newKeyA
[入力] 変更後キーA。変更しない場合は、変更前キーA を渡します。6 バイトの文字列を指定してください。

newKeyB
[入力] 変更後キーB。変更しない場合は、変更前キーB を渡します。6 バイトの文字列を指定してください。

accessControl(将来)
[入力] NULL を指定してください。

戻り値　0=成功、1=エラー。

備考　セキュリティー上、セキュリティーは取得できません。キーの値が正しいことを確認してください。

参照　RfidSettings

3.2.3 バーコードデータ

GetDecodeData()と GetDecodeType()は、バーコードからの生データを取得する関数です。

GetDecodeData

目的 トリガが押されるとデコードされたバーコードデータを取得します。

書式
```
INT GetDecodeData (BYTE *buf,
                   INT bufSize);
```

引数

buf
[出力] デコードされたデータが格納されるバッファのポインタ。

bufSize
[入力] 最大受信バイト数。通常はバッファサイズと同じ値をセットします。

使用例
```
INT nRlt;
BYTE szData[128] = {'\0'};
nRlt = GetDecodeData(szData, sizeof(szData));
```

戻り値 成功すると、取得したバイト数を返します。(NULL 終端文字を除く)
失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

備考 この関数は、直前にデコードされたデータを取得するのに使用します。プリフィックスコード、サフィックスコードが適用されている場合、デコードされたデータに追加されます。

参照 DataOuputSettings, GetDecodeType, GetOutputRecord

GetDecodeType

目的 トリガが押されるとデコードされたバーコードの種別を取得します。

書式
```
INT GetDecodeType (VOID);
```

使用例
```
INT nRlt;
nRlt = GetDecodeType();
```

戻り値 成功すると、バーコード種別を返します。
失敗するとエラーを返します。

-101	E_READER_NOT_INIT

備考 この関数は、直前にデコードされたバーコードの種別を取得するのに使用します。

コード種別		バーコード種別
0x40	64 (@)	ISBT 128
0x41	65 (A)	Code 39
0x42	66 (B)	Italian Pharmacode (Code 32)
0x43	67 (C)	French Pharmacode (CIP 39)
0x44	68 (D)	Industrial 25
0x45	69 (E)	Interleaved 25
0x46	70 (F)	Matrix 25
0x47	71 (G)	Codabar (NW7)
0x48	72 (H)	Code 93
0x49	73 (I)	Code 128
0x4A	74 (J)	UPC-E0
0x4B	75 (K)	UPC-E0 with Addon 2
0x4C	76 (L)	UPC-E0 with Addon 5
0x4D	77 (M)	EAN-8
0x4E	78 (N)	EAN-8 with Addon 2
0x4F	79 (O)	EAN-8 with Addon 5

0x50	80 (P)	EAN-13 (also UPC-A on CCD/Laser scan engine)
0x51	81 (Q)	EAN-13 with Addon 2
0x52	82 (R)	EAN-13 with Addon 5
0x53	83 (S)	MSI
0x54	84 (T)	Plessey
0x55	85 (U)	GS1-128 (EAN-128)
0x56	86 (V)	未定義
0x57	87 (W)	未定義
0x58	88 (X)	未定義
0x59	89 (Y)	未定義
0x5A	90 (Z)	Telepen
0x5B	91 ([ )	GS1 DataBar Omnidirectional (RSS-14)
0x5C	92 (\)	GS1 DataBar Limited (RSS Limited)
0x5D	93 ( ])	GS1 DataBar Expanded (RSS Expanded)
0x5E	94 (^)	UPC-A
0x5F	95 (_)	UPC-A with Addon 2
0x60	96 (‘)	UPC-A with Addon 5
0x61	97 (a)	UPC-E1
0x62	98 (b)	UPC-E1 with Addon 2
0x63	99 (c)	UPC-E1 with Addon 5
0x64	100 (d)	TLC 39 (TCIF Linked Code 39)
0x65	101 (e)	Trioptic (Code 39)
0x66	102 (f)	Bookland (EAN)
0x67	103 (g)	Code 11
0x68	104 (h)	Code 39 Full ASCII
0x69	105 (i)	IATANote (Code 25 used on flight tickets)
0x6A	106 (j)	Industrial 25 (Discrete 25)
0x6B	107 (k)	PDF417
0x6C	108 (l)	MicroPDF417
0x6D	109 (m)	Data Matrix
0x6E	110 (n)	Maxicode
0x6F	111 (o)	QR Code
0x70	112 (p)	US Postnet
0x71	113 (q)	US Planet
0x72	114 (r)	UK Postal
0x73	115 (s)	Japan Postal
0x74	116 (t)	Australian Postal
0x75	117 (u)	Dutch Postal
0x76	118 (v)	Composite Code
0x77	119 (w)	Macro PDF
0x78	120 (x)	Coupon Code
0x79	121 (y)	Chinese 25
0x7A	122 (z)	Aztec
0x7B	123 ({)	MicroQR
0x7C	124 (|)	USPS 4CB / One Code / Intelligent Mail
0x7D	125 (})	UPU FICS Postal
0x7E	126 (~)	Macro MicroPDF417

参照　　DataOuputSettings, GetDecodeType, GetOutputRecord

[注]：IATA は国際航空運送協会(International Air Transport Association)の略で、航空チケットに使用されているバーコード種別です。

ReadBarcodeData

目的

(WM_DECODEDATA から取得せずに)GUI のボタンを押してバーコードデータを読み取ります。

書式

```
INT ReadBarcodeData (INT *codeType,
                     CHAR *buf,
                     INT bufSize,
                     INT timeout);
```

引数

CodeType
[出力] バーコード種別が格納される変数のポインタ。

buf
[出力] デコードされたデータが格納されるバッファのポインタ。

bufSize
[入力]バッファサイズ。

bufSize(将来)
[入力] NULL を指定してください。この値は、UserPreferences_1D_SE955()または UserPreferences_2D_4500()で指定されます。

使用例

```
int b1 = 0, codeType = 0;
char outBuf[200];
memset(outBuf, 0, sizeof(outBuf));
b1 = ReadBarcodeData(&codeType, outBuf, sizeof(outBuf), 3);
```

戻り値

成功すると、取得したバイト数を返します。(NULL 終端文字を除く)
失敗するとエラーを返します。

0	E_READER_UNKNOWN_ERROR
-101	E_READER_NOT_INIT

備考

この関数は、WM_DECODEDATA を使用せずにデータを収集します。例えば、ユーザーが GUI ボタンを押してスキャンする場合、すべきことはクリック時にイベントバンドラでこの関数を呼び出すことです。

参照

DataOutputSettings, GetOutputRecord

3.2.4 RFID データ

GetDecodeRfidData()、GetDecodeRfidUID()と GetDecodeTagType()は、RFID タグからの生データを取得する関数です。RFID の種類によっては、ページ単位で読込み/書込みを行うことができます。RFID タグへの読込み/書込みには ReadRfidData()と WriteRfidData()を使用します。
RFID タグが異なれば容量も異なります。容量にあわせてデータは自動切り捨てされます。
RFID タグの読込み/書込みのデフォルトの開始位置は以下の通りです。

タグ種別	規格	デフォルトページ
Mifare	(ISO 14443A)	4
ICODE SLI	(ISO 15693)	3
LRI512	(ISO 15693)	0
SRF55VxxP	(ISO 15693)	3
Tag-it	(ISO 15693)	0
Others	(ISO 15693)	0
ICODE	(Phillips)	5

[注]：タグ種別が異なると、ページの開始位置も容量も異なります。使用する RFID タグのメモリ構成の仕様を確認してください。

GetDecodeRfidData

目的　トリガが押されるとデコードされた RFID データを取得します。

書式
```
INT GetDecodeRfidData (BYTE *buf,
                       INT bufSize);
```

引数

buf
[出力] デコードされたデータが格納されるバッファのポインタ。

bufSize
[入力] 最大受信バイト数。通常はバッファサイズと同じ値をセットします。

使用例
```
INT nRlt;
BYTE szData[128] = {'\0'};
nRlt = GetDecodeRfidData(szData, sizeof(szData));
```

戻り値　成功すると、取得したバイト数を返します。(NULL 終端文字を除く)
失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

備考　この関数は、直前にデコードされたデータを取得するのに使用します。プリフィックスコード、サフィックスコードが適用されている場合、デコードされたデータに追加されます。

参照　GetDecodeRfidUID, GetDecodeTagType, DataOutputSettings, GetOutputRecord, RfidSettings

GetDecodeRfidUID

目的　トリガが押されると UID を取得します。

書式
```
INT GetDecodeRfidUID (BYTE *buf,
                      INT bufSize);
```

引数

buf
[出力] UID が格納されるバッファのポインタ。

bufSize
[入力] 最大受信バイト数。通常はバッファサイズと同じ値をセットします。

使用例
```
INT nRlt;
BYTE szData[128] = {'\0'};
nRlt = GetDecodeRfidUID(szData, sizeof(szData));
```

戻り値　成功すると、取得したバイト数を返します。(NULL 終端文字を除く)
失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-300	E_BUFF_IS_TOO_SMALL

備考　この関数は、直前にデコードされた UID を取得するのに使用します。プリフィックスコード、サフィックスコードが適用されている場合、デコードされたデータに追加されます。

参照　GetDecodeRfidData, GetDecodeTagType, DataOutputSettings, GetOutputRecord, RfidSettings

GetDecodeTagType

目的　トリガが押されるとデコードされた RFID の種別を取得します。

書式　INT GetDecodeTagType (VOID);

使用例　INT nRlt;
nRlt = GetDecodeTagType();

戻り値　成功すると、タグ種別を返します。

戻り値	16 進	RFID タグ / 規格
0	0x00	不明
80	0x50	ISO15693
81	0x51	I-CODE SLI
82	0x52	I-CODE SLI-L
85	0x55	I-CODE SLI–S
87	0x57	55V10P
88	0x58	55V02P
89	0x59	55V10S
91	0x5B	Tag-it HF-I Plus
92	0x5C	Tag-it HF-I Pro
93	0x5D	MB89R118
94	0x5E	LRI 2K
160	0xA0	Mifare DESFire
161	0xA1	Mifare UL (Ultralight)
162	0xA2	Mifare Mini
163	0xA3	Mifare Classic
164	0xA4	ISO14443-4A
165	0xA5	Mifare S50 1K
166	0xA7	Mifare S50 4K
170	0xAA	Jewel
176	0xB0	ISO14443 B
240	0xF0	FeliCa

失敗するとエラーを返します。

-101	E_READER_NOT_INIT

備考　この関数は、直前にデコードされたタグの種別を取得するのに使用します。

参照　GetDecodeRfidData, GetDecodeRfidUID, DataOutputSettings, GetOutputRecord, RfidSettings

ReadRfidData

目的

(WM_DECODEDATA から取得せずに)GUI のボタンを押して RFID データを読み取ります。

書式

```
INT ReadRfidData (INT type,
                  BYTE *readData,
                  INT bufSize,
                  INT startByte,
                  INT readLen,
                  INT timeout,
                  CHAR *loginKey,
                  INT loginKeyType);
```

引数

type

[入力] 読取るデータの種類。(UID またはエンコードされたデータ)

1	READ_UID	UID 読取り
2	READ_DATA	データ読取り

readData

[出力] デコードされたデータが格納されるバッファのポインタ。

bufSize

[入力] 最大読取りバイト数。通常はバッファサイズと同じ値をセットします。

startByte

[入/出力] 収集データ開始位置。

-1	収集データ開始位置を指定します。”-1”を指定した場合、RFID リーダは RFID タグのデフォルトページから読込みを開始します。 RFID タグのデフォルトページは次の通りです。

タグ種別	規格	デフォルトページ
Mifare	(ISO 14443A)	4
ICODE SLI	(ISO 15693)	3
LRI512	(ISO 15693)	0
SRF55VxxP	(ISO 15693)	3
Tag-it	(ISO 15693)	0
Others	(ISO 15693)	0
ICODE	(Phillips)	5

readLen

[入力] 読取りバイト数。タグ種別が異なれば読取り可能なページ数も異なります。読取り可能な長さを超える値を設定しても、そのタグで読取り可能な長さに自動調節されます。

timeout(将来)

loginKey

[入力] Mifare スタンダード 1K/4K、SLE66R35 などの特定の RFID タグにアクセスするためのログインキーを格納した変数のポインタ。

- このキーは 12 バイトの長さの 16 進文字列である必要があります。
- このキーは"startByte"、” readLen”で指定された対象範囲のすべてのセクタに適用されます。

loginKeyType

[入力] ログインキー種別。

0(※)	キーA。
1	キーB。

使用例

```
INT nRlt, nRead = 0;
BYTE szRead[128] = {‘\0’};
nRlt = ReadRfidData(READ_UID, szRead,
                    sizeof(szRead), -1, 48, 10, ”FFFFFFFFFFFF”);
```

戻り値　成功すると、取得したバイト数を返します。(NULL 終端文字を除く)
失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-272	E_TAG_READ_FAILED
-273	E_TAG_VALUE_OVER_RANGE
-274	E_TAG_INVALID_LENGTH
-278	E_TAG_NO_TAG
-280	E_TAG_CANNOT_READ
-281	E_TAG_READ_TIMEOUT

備考　この関数は、WM_DECODEDATA を使用せずにデータを収集します。例えば、ユーザーが GUI ボタンを押してスキャンする場合、すべきことはクリック時にイベントバンドラでこの関数を呼び出すことです。

参照　DataOutputSettings, GetOutputRecord, RfidSettings, WriteRfidData

WriteRfidData

目的　GUI のボタンを押して RFID タグにデータを書込みます。

書式

```
INT WriteRfidData (BYTE *writeData,
                   INT startByte,
                   INT writeLen,
                   INT *written,
                   INT timeout,
                   INT mode,
                   CHAR *loginKey,
                   INT loginKeyType);
```

引数

writeData
[入力] 書き込むデータが格納されているバッファのポインタ。

startByte
[入/出力] 書込みデータ開始位置。

-1	デフォルトページからの書込みを開始します。 RFID タグのデフォルトページは次の通りです。

タグ種別	規格	デフォルトページ
Mifare	(ISO 14443A)	4
ICODE SLI	(ISO 15693)	3
LRI512	(ISO 15693)	0
SRF55VxxP	(ISO 15693)	3
Tag-it	(ISO 15693)	0
Others	(ISO 15693)	0
ICODE	(Phillips)	5

writeLen
[入力] 最大書込みバイト数。タグ種別が異なれば書込み可能なページ数も異なります。書込み可能な長さを超える値を設定しても、そのタグで書込み可能な長さに自動調節されます。

written
[出力] 実際に書き込まれたバイト数を格納するバッファへのポインタ。

timeout(将来)

loginKey
[入力]書込みモード。

0	すぐに書込み。
0 以外	トリガ押下時に書込み。

loginKey

[入力] Mifare スタンダード 1K/4K、SLE66R35 などの特定の RFID タグにアクセスするためのログインキーが格納されている変数のポインタ。

- このキーは 12 バイトの長さの 16 進文字列である必要があります。
- このキーは"startByte"、” writeLen”で指定された対象範囲のすべてのセクタに適用されます。

loginKeyType

[入力] ログインキー種別。

0(※)	キーA。
1	キーB。

使用例

```
INT nRlt, nWritten = 0;
BYTE szWriteData[] = ”Written by CipherLab”;
nRlt = WriteRfidData(szWriteData,
                -1, strlen(szWriteData), &nWritten, 10, 0, 0);
nRlt = Beeper(-1, szPath);
```

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-273	E_TAG_VALUE_OVER_RANGE
-274	E_TAG_INVALID_LENGTH
-275	E_TAG_GET_DATA_FAILED
-276	E_TAG_WRITE_FAILED
-277	E_TAG_CANNOT_WRITE
-278	E_TAG_NO_TAG
-279	E_TAG_WRITE_TIMEOUT

備考

この関数は、RFID タグへ何を書込むかを定義し、書込みを行います。

参照

DataOutputSettings, GetOutputRecord, ReadRfidData, RfidSettings

3.3　ステータス表示操作

リーダ DLL は NotificationSettings()での設定に応じて、デコード成功時に音をや鳴らしたり、バイブを振動させたりします。デコード以外のイベント受信の合図には Beeper()をコールします。

3.3.1　通知設定

NotificationSettings

目的　通知設定を構成します。

書式

```
INT NotificationSettings (INT rw,
                          INT *goodRead,
                          INT *enableVibrator,
                          INT *vibrationTime,
                          INT *ledDuration);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	出力設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	出力設定書込み。

goodread

[入/出力] 読取り成功時に.WAV ファイルを再生するかどうか指定する値。

0	無音
1～9	音を鳴らす。(デフォルトは 2)

enableVibrator

[入力] 読取り成功時にバイブするかどうか指定する値。

0(※)	バイブなし
1	バイブあり

vibrationTime

[入力] 読取り成功時のバイブの振動時間。

0(※)	バイブなし
1～9	デフォルトは 2 秒

ledDuration

[入力] 読取り成功時に緑色の LED を点灯するかどうかを指定する値。

0(※)	点灯なし
1～9	点灯時間(ミリ秒)

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

参照　Beeper

3.3.2 BEEPER

Beeper

目的　スピーカから音を出します。

書式　INT Beeper (INT mode,
BYTE *path);

引数　mode
[入力] 再生するサウンド。

0	無音
1～9	音を鳴らす。
-1	ユーザー定義パスで指定されたサウンド。

path
[入力] ユーザー定義パスが格納されているバッファへのポインタ。
➢ Mode の値が-1 の場合のみ有効です。

使用例
```
INT nRlt;
BYTE szPath[] = {"\window\test.wav"};
nRlt = Beeper(9, NULL);
nRlt = Beeper(-1, szPath);
```

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER
-241	E_SOUND_INVALID_INDEX
-242	E_SOUND_INVALID_PATH
-243	E_SOUND_PLAY_FAILED

備考　この関数は、非同期的に音や WAV ファイルを再生します。再生を停止するには Beeper(0, NULL)をコールします。

参照　NotificationSettings

3.4 重要情報取得

3.4.1 リーダ DLL バージョン

GetDllVer

目的 現在のリーダ DLL バージョンを取得します。

書式
```
INT GetDllVer (CHAR *buf,
               INT bufSize);
```

引数

buf
[入力] バージョンが格納されるバッファへのポインタ。

bufSize
[入力] バッファサイズ。

使用例
```
CHAR buf[100];
GetDllVer(buf, 100);
```

戻り値 0=成功。1=失敗。

3.4.2 デコーダバージョン

GetDecoderVersion

目的 デコーダのバージョンを取得します。

書式
```
INT GetDecoderVersion (INT target,
                       CHAR *buf,
                       INT bufSize);
```

引数

target
[入力] バージョンを取得するリーダ。

7	DC_READER_BC	バーコードリーダのバージョン取得。
32	DC_READER_RFID	RFID リーダのバージョン取得。

buf
[入力] バージョン情報が格納されるバッファへのポインタ。

bufSize
[入力] 最大受信りバイト数。通常はバッファサイズと同じ値をセットします。

使用例
```
CHAR buf[64] = {'\0'};
INT nRlt = 0;
nRlt = GetDecoderVersion(DC_READER_BC, buf, sizeof(buf));
nRlt = GetDecoderVersion(DC_READER_RFID, buf, sizeof(buf));
```

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	E_READER_UKNOWN_ERROR
-101	E_READER_NOT_INIT

3.5　スキャンエンジン設定 ー 1D CCD スキャンエンジン

UserPreferences_1D_SM1()は 1 次元の CCD バーコードリーダーを構成します。

3.5.1 プリファレンス

UserPreferences_1D_SM1	**CCD**

目的　1D CCD バーコードリーダを設定します。

書式

```
int  UserPreferences_1D_SM1 (int rw,
                             int *laserOnTime,
                             int *timeoutBetweenSameBarcode,
                             int *redundancyLevel,
                             int *triggerMode);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

laserOnTime

[入/出力] スキャン時のバーコードのデコード最大時間。

5～99	デフォルトは 30(0.1 秒単位)

timeoutBetweenSameBarcode

[入/出力] 2 度続けて同じバーコードを読むことができるまでの最小時間。誤って同じバーコードを 2 度続けて読むことを阻止するための設定です。

➢ コンティニュアスモード時に適用されます。

0～99	デフォルトは 10(0.1 秒単位)

redundancyLevel

[入/出力] デコードの冗長性。バーコードの品質が悪い場合は高い冗長性を選択してください。

1(※)	以下のバーコードを、正常なデコードのために 2 度読取り照合を行います。 バーコード種別 / コードの長さ Codabar / すべて MSI / 4 文字以下 Industrial 25 (Discrete 25) / 8 文字以下 Interleaved 25 / 8 文字以下
2	すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。
3	すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。 ただし、以下のバーコードでは正常なデコードのために 3 度読取り照合を行います。 バーコード種別 / コードの長さ MSI / 4 文字以下 Industrial 25 (Discrete 25) / 8 文字以下 Interleaved 25 / 8 文字以下
4	すべてのバーコードで正常なデコードのために 3 度読取り照合を行います。

triggerMode

[入/出力] スキャンモード。

4	コンティニュアスモード
8(※)	レーザーモード

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.2 UPC_1D_SM1

Upc_1D_SM1	**CCD**

目的　UPC-A と UPC-E のシンボル設定を構成します。

書式

```
int  Upc_1D_SM1 (int rw,
                 int *enableUPC_A,
                 int *enableUPC_E,
                 int *enableUPC_E1,
                 int *UPC_EAN_JAN_decodeSupplementals,
                 int *UPC_EAN_JAN_supplementalsRedundancy,
                 int *transmitUPC_AcheckDigit,
                 int *transmitUPC_EcheckDigit,
                 int *transmitUPC_E1checkDigit,
                 int *preambleUPC_A,
                 int *preambleUPC_E,
                 int *preambleUPC_E1,
                 int *convertUPC_EtoA,
                 int *convertUPC_E1toA);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableUPC_A

[入/出力] UPC-A 有効/無効。

0	無効。
1(※)	有効。

enableUPC_E

[入/出力] UPC-E 有効/無効。

0	無効。
1(※)	有効。

enableUPC_E1

[入/出力] UPC-E 有効/無効。

0(※)	無効。
1	有効。

UPC_EAN_JAN_decodeSupplementals

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

UPC_EAN_JAN_supplementalsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 7(読取り照合回数)。

transmitUPC_AcheckDigit

[入/出力] UPC-A チェックデジット送信有無。

0	送信なし。
1(※)	送信あり。

transmitUPC_EcheckDigit

[入/出力] UPC-E チェックデジット通信有無。

0	なし。
1(※)	あり。

preambleUPC_A

[入/出力] UPC-A プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUPC_E

[入/出力] UPC-E プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUPC_E1

[入/出力] UPC-E1 プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

convertUPC_EtoA

[入/出力] UPC-E0 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

convertUPC_E1toA

[入/出力] UPC-E1 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.3 EANJAN_1D_SM1

EanJan_1D_SM1	**CCD**

目的　EAN/JAN-8 と ENA/JAN-13 のシンボル設定を構成します。

書式

```
int  EanJan_1D_SM1 (int rw,
                    int *enableEAN8_JAN8,
                    int *enableEAN13_JAN13,
                    int *enableBooklandEAN,
                    int *UPC_EAN_JAN_decodeSupplementals,
                    int *UPC_EAN_JAN_supplementalsRedundancy,
                    int *EAN8_JAN8_extend);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableEAN8_JAN8

[入/出力] EAN/JAN-8 有効/無効。

0	無効。
1(※)	有効。

enableEAN13_JAN-13

[入/出力] EAN/JAN13 有効/無効。

0	無効。
1(※)	有効。

enableBooklandEAN

[入/出力] Bookland EAN 有効/無効。

- EAN/JAN13 は有効でなければなりません。

0	無効。
1(※)	有効。

UPC_EAN_JAN_decodeSupplementals

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

UPC_EAN_JAN_supplementalsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 10(読取り照合回数)。

EAN8_JAN8_extend

[入/出力] EAN/JAN-8 から EAN/JAN-13 への変換有無。

0(※)	変換なし。
1	変換あり。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.4 Code39_1D_SM1

Code39_1D_SM1	**CCD**

目的　CODE39 のシンボル設定を構成します。

書式

```
int  Code39_1D_SM1 (int rw,
                    int *enable,
                    int *code39ToCode32,
                    int *code32Prefix,
                    int *checkDigitVerification,
                    int *transmitCheckDigit,
                    int *fullASCII,
                    int *length1,
                    int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code39 有効/無効。

0	無効。
1(※)	有効。

code39ToCode32

[入/出力] Code39 から Code32(Italian Pharmacode)への変換有無。

0(※)	変換なし。
1	変換あり。

code32Prefix

[入/出力] Code32 プリフィックスの送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

fullASCII

[入/出力] Code39 フルアスキーのサポート有無。

0(※)	なし。(Standard Code 39)
1	あり。(Code 39 Full ASCII)

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.5 Code93_1D_SM1

Code93_1D_SM1	**CCD**

目的　CODE93 のシンボル設定を構成します。

書式
```
int  Code93_1D_SM1 (int rw,
                    int *enable,
                    int *length1,
                    int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code93 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.6 Interleaved2Of5_1D_SM1

Interleaved2Of5_1D_SM1	**CCD**

目的　Interleaved 25 のシンボル設定を構成します。

書式

```
int  Interleaved2Of5_1D_SM1 (int rw,
                             int *enable,
                             int *length1,
                             int *length2,
                             int *checkDigitVerification,
                             int *transmitCheckDigit);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Interleaved 25 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.7 INDUSTRIAL2OF5_1D_SM1

INDUSTRIAL2OF5_1D_SM1	**CCD**

目的　Industrial 25 (Discrete 25)のシンボル設定を構成します。

書式

```
int  INDUSTRIAL2OF5_1D_SM1 (int rw,
                            int *enable,
                            int *length1,
                            int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Industrial 25 (Discrete 25)有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.8 Codabar_1D_SM1

Codabar_1D_SM1	**CCD**

目的　Codabar のシンボル設定を構成します。

書式

```
int  Codabar_1D_SM1 (int rw,
                     int *enable,
                     int *length1,
                     int *length2,
                     int *clsiEditing,
                     int *notisEditing);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Codabar 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

clsiEditing

[入/出力] CLSI 編集有無。

0(※)	編集なし。
1	編集あり。

- CLSI 編集ありの場合、スタート/ストップ文字を消去し、1 文字目、5 文字目、10 文字目の後にスペースを挿入します。
- 14 文字のバーコードの長さにスタート/ストップ文字は含みません。

notisEditing

[入/出力] NOTIS 編集有無。

0(※)	編集なし。
1	編集あり。

- NOTIS 編集はスタート/ストップ文字を取り除きます。つまり「スタート/ストップ文字送信無効」と同じ意味になります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.9 MSI_1D_SM1

MSI_1D_SM1	**CCD**

目的　MSI のシンボル設定を構成します。

書式

```
int  MSI_1D_SM1 (int rw,
                 int *enable,
                 int *length1,
                 int *length2,
                 int *checkDigitVerification,
                 int *transmitCheckDigit
                 int *checkDigitAlgorithm);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] MSI 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitAlgorithm

[入/出力] 適用するアルゴリズム。

0	Modulo 10 / Modulo 11。
1(※)	Double Modulo 10。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.10 GS1_DATABAR_1D_SM1

GS1_DATABAR_1D_SM1	**CCD**

目的　GS1 DataBar (RSS family)のシンボル設定を構成します。

書式
```
int  GS1_DATABAR_1D_SM1 (int rw,
                         int *enableDataBarOmnidirectional,
                         int *enableGS1_DataBarLimited,
                         int *enableGS1_DataBarExpanded);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableDataBarOmnidirectional

[入/出力] GS1 DataBar Omnidirectional (RSS-14)有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarLimited

[入/出力] GS1 DataBar Limited (RSS Limited) 有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarExpanded

[入/出力] GS1 DataBar Expanded (RSS Expanded)有効/無効。

0	無効。
1(※)	有効。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.5.11 Code128_1D_SM1

Code128_1D_SM1	**CCD**

目的　Code 128 のシンボル設定を構成します。

書式
```
int  Code128_1D_SM1 (int rw,
                     int *enableCode128,
                     int *enableGS1_128,
                     int * enableISBT128);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableCode128

[入/出力] Code 128 有効/無効。

0	無効。
1(※)	有効。

enableGS1_128

[入/出力] GS1-128 有効/無効。

0	無効。
1(※)	有効。

enableISBT128

[入/出力] ISBT 128 有効/無効。

0	無効。
1(※)	有効。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6　スキャンエンジン設定 ― 1D レーザースキャンエンジン

UserPreferences_1D_SE955()と MiscellaneousOption_1D_SE955()は、1D(レーザー)バーコードリーダを設定するための機能です。

3.6.1 プリファレンス

UserPreferences_1D_SE955 レーザー

目的　1D レーザーバーコードリーダを設定します。

書式

```
INT  UserPreferences_1D_SE955 (INT rw,
                               INT *laserOnTime,
                               INT *scanAngle,
                               INT *timeoutBetweenSameBarcode,
                               INT *redundancyLevel,
                               INT *bidirectionalRedundancy,
                               INT *aimDuration,
                               INT *triggerMode);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

laserOnTime

[入/出力] スキャン時のバーコードのデコード最大時間。

5～99	デフォルトは 30(0.1 秒単位)

scanAngle

[入/出力] スキャン角度。

0x05	狭角(35°)。
0x06(※)	広角(47°)。

timeoutBetweenSameBarcode

[入/出力] 2 度続けて同じバーコードを読むことができるまでの最小時間。誤って同じバーコードを 2 度続けて読むことを阻止するための設定です。

➢ コンティニュアスモード時に適用されます。

0～99	デフォルトは 10(0.1 秒単位)

redundancyLevel
[入/出力] デコードの冗長性。バーコードの品質が悪い場合は高い冗長性を選択してください。

<table>
<tr><td>1(※)</td><td>以下のバーコードを、正常なデコードのために 2 度読取り照合を行います。
<table>
<tr><td>バーコード種別</td><td>コードの長さ</td></tr>
<tr><td>Codabar</td><td>すべて</td></tr>
<tr><td>MSI</td><td>4 文字以下</td></tr>
<tr><td>Industrial 25 (Discrete 25)</td><td>8 文字以下</td></tr>
<tr><td>Interleaved 25</td><td>8 文字以下</td></tr>
</table></td></tr>
<tr><td>2</td><td>すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。</td></tr>
<tr><td>3</td><td>すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。
ただし、以下のバーコードでは正常なデコードのために 3 度読取り照合を行います。
<table>
<tr><td>バーコード種別</td><td>コードの長さ</td></tr>
<tr><td>MSI</td><td>4 文字以下</td></tr>
<tr><td>Industrial 25 (Discrete 25)</td><td>8 文字以下</td></tr>
<tr><td>Interleaved 25</td><td>8 文字以下</td></tr>
</table></td></tr>
<tr><td>4</td><td>すべてのバーコードで正常なデコードのために 3 度読取り照合を行います。</td></tr>
</table>

bidirectionalRedundancy (将来)

aimDuration (将来)

triggerMode
[入/出力] スキャンモード。

4	コンティニュアスモード
8(※)	レーザーモード

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

MiscellaneousOption_1D_SE955 レーザー

目的 1D レーザーバーコードリーダを設定します。

書式
```
INT  MiscellaneousOption_1D_SE955 (INT rw,
                                   INT *transmitCodeIdChar,
                                   INT *sendNoReadMessage);
```

引数 ※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

transmitCodeIdChar

[入/出力] コード ID 文字送信有無。

0(※)	送信なし
1	AIM コード ID 文字送信あり。 データの先頭にあり、AIM コード ID にはそれぞれ 3 文字の文字列”[CM”が含まれています。 ➢ [・・・ フラグ文字(ASCII 93) ➢ C・・・ コード文字(下記参照) ➢ M・・・修飾文字(下記参照)

sendNoReadMessage (将来)

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

AIM コード ID　ー コード文字

コード文字	コード種別
A	Code 39
C	Code 128
E	UPC/EAN
F	Codabar
G	Code 93
H	Code 11
I	Interleaved 25
M	MSI
S	Industrial 25(Discrete 25)、IATA 2 of 5

AIM コード ID　ー 修飾文字

コード種別	値	意味
Code 39	0	チェック文字なし、またはフル ASCII 処理。
	1	チェックディジット確認済み。
	3	チェックディジットは確認済みで、剥離済み。
	4	フル ASCII 変換されている。
	5	1 と 4 の両方。
	7	3 と 4 の両方。
Code 128	0	標準のデータパケット。最初の文字位置にファンクションコード 1”FNC1”はありません。
	1	ファンクションコード 1”FNC1”が最初の文字位置にあります。
	2	ファンクションコード 1”FNC1”が 2 番目の文字位置にあります。
Interleaved 25	0	チェックディジット処理なし。
	1	チェックディジット確認済み。
	3	チェックディジットは確認済みで、剥離済み。
Codabar	0	常に 0。
Code 93	0	常に 0。
MSI	0	Modulo 10 チェックディジットは確認済みで、送信済み。
	1	Modulo 10 チェックディジットは確認済みだか、送信なし。
Industrial 25 (Discrete 25)	0	常に 0。
UPC/EAN	0	(補足データなしの)13 桁の UPC-A および UPC-E で、フル EAN カントリーコードフォーマットの標準データパケット。
	1	2 桁の補足データのみ。
	2	5 桁の補足データのみ。
	4	EAN-8 データパケット。
	UPC-A アドオン 2 のバーコード” 012345678905-10 は 21 桁の 文字列 ”]E0012345678905]E110”で送信されます。	
Bookland EAN	0	常に 0。
Trioptic Code 39	0	常に 0。

3.6.2 UPC_1D_SE955

Upc_1D_SE955	**レーザー**

目的　UPC-A と UPC-E のシンボル設定を構成します。

書式

```
INT  Upc_1D_SE955 (INT rw,
                   INT *enableUpcA,
                   INT *enableUpcE,
                   INT *enableUpcE1,
                   INT *enableAddons,
                   INT *addonsRedundancy,
                   INT *transmitUpcACheckDigit,
                   INT *transmitUpcECheckDigit,
                   INT *transmitUpcE1CheckDigit,
                   INT *preambleUpcA,
                   INT *preambleUpcE,
                   INT *preambleUpcE1,
                   INT *convertUpcEtoA,
                   INT *convertUpcE1toA,
                   INT *uccCouponExtendedCode,
                   INT *upcEanSecurityLevel);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableUpcA

[入/出力] UPC-A 有効/無効。

0	無効。
1(※)	有効。

enableUpcE

[入/出力] UPC-E(UPC-E0)有効/無効。

0	無効。
1(※)	有効。

enableUpcE1

[入/出力] UPC-E1 有効/無効。

0(※)	無効。
1	有効。

enableAddons

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

addonsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 7(読取り照合回数)。

transmitUpcACheckDigit

[入/出力] UPC-A チェックデジット送信有無。

0	送信なし。
1(※)	送信あり。

transmitUpcECheckDigit

[入/出力] UPC-E0 チェックデジット通信有無。

0	なし。
1(※)	あり。

transmitUpcE1CheckDigit

[入/出力] UPC-E1 チェックデジット通信有無。

0	なし。
1(※)	あり。

preambleUpcA

[入/出力] UPC-A プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUpcE

[入/出力] UPC-E0 プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUpcE1

[入/出力] UPC-E1 プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

convertUPC_EtoA

[入/出力] UPC-E0 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

convertUPC_E1toA

[入/出力] UPC-E1 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

uccCouponExtendedCode

[入/出力] UCC クーポンコード有効/無効。

0(※)	無効。
1	有効。

upcEanSecurityLevel

[入/出力] UPC/EAN バーコードの読取りレベル。バーコードの品質が落ちる場合はより高いレベル値を指定する必要があります。ただし、レベルが高くなるほど読取り速度は遅くなります。アプリケーションの必要に応じて設定してください。

0	UPC/EAN を最大機能で読取ります。
1	バーコードの品質が悪く、特定の文字を読み損ねている。読取りミスが印刷不良が原因で発生している場合、このレベルを設定します。
2(※)	読取りミスが印刷不良が原因で発生していて特定の文字に限定されていない場合、このレベルを設定します。
3	レベル 2 でも読取りミスがある場合、このレベルを設定します。このレベルは最終的な手段であり、読取り速度が犠牲となります。それよりは、バーコードの品質を向上させることを推奨します。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.3 EANJAN_1D_SE955

EanJan_1D_SE955	**レーザー**

目的　EAN/JAN-8 と ENA/JAN-13 のシンボル設定を構成します。

書式

```
INT  EanJan_1D_SE955 (INT rw,
                      INT *enableEanJan8,
                      INT *enableEanJan13,
                      INT *enableBooklandEan,
                      INT *enableAddons,
                      INT *addonsRedundancy,
                      INT *EanJan8Extend,
                      INT *uccCouponExtendedCode,
                      INT *upcEanSecurityLevel);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableEanJan8

[入/出力] EAN/JAN-8 有効/無効。

0	無効。
1(※)	有効。

enableEanJan13

[入/出力] EAN/JAN13 有効/無効。

0	無効。
1(※)	有効。

enableBooklandEAN

[入/出力] Bookland EAN 有効/無効。

➢ EAN/JAN13 は有効でなければなりません。

0	無効。
1(※)	有効。

enableAddons

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

addonsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 10(読取り照合回数)。

EanJan8Extend

[入/出力] EAN/JAN-8 から EAN/JAN-13 への変換有無。

0(※)	変換なし。
1	変換あり。

uccCouponExtendedCode

[入/出力] UCC クーポンコード有効/無効。

0(※)	無効。
1	有効。

upcEanSecurityLevel

[入/出力] UPC/EAN バーコードの読取りレベル。バーコードの品質が落ちる場合はより高いレベル値を指定する必要があります。ただし、レベルが高くなるほど読取り速度は遅くなります。アプリケーションの必要に応じて設定してください。

0	UPC/EAN を最大機能で読取ります。
1	バーコードの品質が悪く、特定の文字を読み損ねている。読取りミスが印刷不良が原因で発生している場合、このレベルを設定します。
2(※)	読取りミスが印刷不良が原因で発生していて特定の文字に限定されていない場合、このレベルを設定します。
3	レベル 2 でも読取りミスがある場合、このレベルを設定します。このレベルは最終的な手段であり、読取り速度が犠牲となります。それよりは、バーコードの品質を向上させることを推奨します。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.4 Code39_1D_SE955

Code39_1D_SE955	**レーザー**

目的　CODE39 のシンボル設定を構成します。

書式

```
INT  Code39_1D_SE955 (INT rw,
                      INT *enable,
                      INT *enableTrioptic,
                      INT *convertToCode32,
                      INT *prefix Code32,
                      INT *checkDigitVerification,
                      INT *transmitCheckDigit,
                      INT *fullASCII,
                      INT *length1,
                      INT *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code39 有効/無効。

0	無効。
1(※)	有効。

enableTrioptic

[入/出力] Trioptic Code39 有効/無効。

0	無効。
1(※)	有効。

convertToCode32

[入/出力] Code39 から Code32(Italian Pharmacode)への変換有無。

0(※)	変換なし。
1	変換あり。

prefix Code32

[入/出力] Code32 プリフィックスの送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

fullASCII

[入/出力] Code39 フルアスキーのサポート有無。

0(※)	なし。(Standard Code 39)
1	あり。(Code 39 Full ASCII)

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.5 Code93_1D_SE955

Code93_1D_SE955	レーザー

目的　CODE93 のシンボル設定を構成します。

書式

```
INT  Code93_1D_SE955 (INT rw,
                      INT *enable,
                      INT *length1,
                      INT *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code93 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.6 Interleaved2Of5_1D_SE955

Interleaved2Of5_1D_SE955	**レーザー**

目的　Interleaved 25 のシンボル設定を構成します。

書式

```
INT  Interleaved2Of5_1D_SE955 (INT rw,
                               INT *enable,
                               INT *length1,
                               INT *length2,
                               INT *checkDigitVerification,
                               INT *transmitCheckDigit
                               INT *convertToEAN13);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Interleaved 25 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

convertToEAN13

[入/出力] EAN-13 変換有無。

- まず、チェックディジット検査が無効でなければなりません。またバーコードが先行ゼロで、有効な EAN-13 チェックディジットがある場合のみ機能します。

0(※)	変換なし。
1	変換あり。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.7 INDUSTRIAL2OF5_1D_SE955

INDUSTRIAL2OF5_1D_SE955	レーザー

目的　Industrial 25 (Discrete 25)のシンボル設定を構成します。

書式

```
int  INDUSTRIAL2OF5_1D_SE955 (int rw,
                              int *enable,
                              int *length1,
                              int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Industrial 25 (Discrete 25)有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.8 CHINESE2OF5_1D_SE955

CHINESE2OF5_1D_SE955	レーザー

目的　Chinese 25 のシンボル設定を構成します。

書式　int CHINESE2OF5_1D_SE955 (int rw,
int *enable)

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Chinese 25 有効/無効。

0	無効。
1(※)	有効。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.9 Codabar_1D_SE955

Codabar_1D_SE955	レーザー

目的　Codabar のシンボル設定を構成します。

書式

```
int  Codabar_1D_SE955 (int rw,
                       int *enable,
                       int *length1,
                       int *length2,
                       int *enableCLSI_Editing,
                       int *enableNOTIS_Editing);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Codabar 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

enableCLSI_Editing

[入/出力] CLSI 編集有無。

0(※)	編集なし。
1	編集あり。

- CLSI 編集ありの場合、スタート/ストップ文字を消去し、1 文字目、5 文字目、10 文字目の後にスペースを挿入します。
- 14 文字のバーコードの長さにスタート/ストップ文字は含みません。

enableNOTIS_Editing

[入/出力] NOTIS 編集有無。

0(※)	編集なし。
1	編集あり。

- NOTIS 編集はスタート/ストップ文字を取り除きます。つまり「スタート/ストップ文字送信無効」と同じ意味になります。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.10 MSI_1D_SE955

MSI_1D_SE955	レーザー

目的　MSI のシンボル設定を構成します。

書式

```
int  MSI_1D_SE955 (int rw,
                   int *enable,
                   int *length1,
                   int *length2,
                   int *checkDigitVerification,
                   int *transmitCheckDigit
                   int *checkDigitAlgorithm);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] MSI 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitAlgorithm

[入/出力] 適用するアルゴリズム。

0	Modulo 10 / Modulo 11。
1(※)	Double Modulo 10。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.11 GS1_DATABAR_1D_SE955

GS1_DATABAR_1D_SE955	レーザー

目的　GS1 DataBar (RSS family)のシンボル設定を構成します。

書式
```
int  GS1_DATABAR_1D_SE955 (int rw,
                           int *enableGS1_DataBarOmnidirectional,
                           int *enableGS1_DataBarLimited,
                           int *enableGS1_DataBarExpanded,
                           int *convertToUpcEan);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableGS1_DataBarOmnidirectional

[入/出力] GS1 DataBar Omnidirectional (RSS-14)有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarLimited

[入/出力] GS1 DataBar Limited (RSS Limited) 有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarExpanded

[入/出力] GS1 DataBar Expanded (RSS Expanded)有効/無効。

0	無効。
1(※)	有効。

convertToUpcEan

[入/出力] RSS の UPC/EAN 変換有無。

0(※)	変換なし。
1	変換あり。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.12 Code128_1D_SE955

Code128_1D_SE955	レーザー

目的　Code 128 のシンボル設定を構成します。

書式
```
int  Code128_1D_SE955 (int rw,
                       int *enableCode128,
                       int *enableGS1_128,
                       int *enableISBT128);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableCode128

[入/出力] Code 128 有効/無効。

0	無効。
1(※)	有効。

enableGS1_128

[入/出力] GS1-128 有効/無効。

0	無効。
1(※)	有効。

enableISBT128

[入/出力] ISBT 128 有効/無効。

0	無効。
1(※)	有効。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.6.13 CODE11_1D_SE955

CODE11_1D_SE955	レーザー

目的　Oode 11 のシンボル設定を構成します。

書式

```
int  CODE11_1D_SE955 (int rw,
                      int *enable,
                      int *length1,
                      int *length2,
                      int *checkDigitVerification,
                      int *transmitCheckDigit
                      int *checkDigitAlgorithm);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code 11 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitAlgorithm

[入/出力] 適用するアルゴリズム。

0	Modulo 10 / Modulo 11。
1(※)	Double Modulo 10。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

-253	E_WRONG_READER_TYPE

3.7　スキャンエンジン設定 ― 2D レーザースキャンエンジン

UserPreferences_2D_SE4500()と MiscellaneousOption_2D_SE4500()は、2D バーコードリーダを設定するための機能です。

3.7.1 プリファレンス

UserPreferences_2D_SE4500 2D

目的　1D レーザーバーコードリーダを設定します。

書式

```
INT  UserPreferences_2D_SE4500 (INT rw,
                                INT *laserOnTime,
                                INT *timeoutBetweenSameBarcode,
                                INT *triggerMode);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

laserOnTime

[入/出力] スキャン時のバーコードのデコード最大時間。

5～99	デフォルトは 30(0.1 秒単位)

timeoutBetweenSameBarcode (将来)

triggerMode

[入/出力] スキャンモード。

0(※)	レベル
7	プレゼンテーションモード
9	自動照準

戻り値　0=成功。1=失敗。

SymbologySecurityLevel_2D_SE4500 2D

目的 デコードの冗長性を設定します。

書式
```
INT  SymbologySecurityLevel_2D_SE4500 (INT rw,
                                       INT *redundancyLevel,
                                       INT *securityLevel,
                                       INT *interCharGapSize);
```

引数 ※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

redundancyLevel

[入/出力] デコードの冗長性。バーコードの品質が悪い場合は高い冗長性を選択してください。

1(※)	以下のバーコードを、正常なデコードのために 2 度読取り照合を行います。 バーコード種別 / コードの長さ Codabar / 8 文字以下 MSI / 4 文字以下 Industrial 25 (Discrete 25) / 8 文字以下 Interleaved 25 / 8 文字以下
2	すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。
3	すべてのバーコードで正常なデコードのために 2 度読取り照合を行います。ただし、以下のバーコードでは正常なデコードのために 3 度読取り照合を行います。 バーコード種別 / コードの長さ Codabar / 8 文字以下 MSI / 4 文字以下 Industrial 25 (Discrete 25) / 8 文字以下 Interleaved 25 / 8 文字以下
4	すべてのバーコードで正常なデコードのために 3 度読取り照合を行います。

securityLevel

[入/出力] Code 128、Code 93、UPC/EAN のようなデルタバーコードを読取るときに、いくつかの印刷品質の問題を修正するデコードのセキュリティレベル。

0(※)	レベル 0 – デフォルト。スキャンエンジンの性能を最大限にいかすことで、ほとんどの仕様の範囲内のバーコードをデコードできます。
1	レベル 1 - デコードミスが発生する場合はこのオプションを選択します。
2	レベル 2 – レベル 1 でもデコードミスが発生する場合はこのオプションを選択します。
3	レベル 2 – レベル 2 でもデコードミスが発生する場合はこのオプションを選択します。ただし、このオプションを選択するとデコード性能が劣ります。バーコードの品質を向上させることをお勧めします。

interCharGapSize

[入/出力] 一般的には非常に小さい、Code 39 と Codabar ための文字間のギャップサイズを指定する値。各種のバーコード印刷技術により、ギャップサイズが規定よりも大きいものも作成され、スキャナがデコードできない場合があります。このような場合、文字間ギャップ大を設定して仕様外のバーコードを読めるようにします。

0(※)	通常の文字間隔。
1	大きい文字間隔。

戻り値 0=成功。1=失敗。

MiscellaneousOption_2D_SE4500 2D

目的 2D スキャナを設定します。

書式 INT MiscellaneousOption_2D_SE4500 (INT rw,
INT *transmitCodeIdChar,
INT *sendNoReadMessage);

引数 ※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

transmitCodeIdChar

[入/出力] コード ID 文字送信有無。

0(※)	送信なし
1	AIM コード ID 文字送信あり。 データの先頭にあり、AIM コード ID にはそれぞれ 3 文字の文字列”[CM”が含まれています。 ➢ [・・・ フラグ文字(ASCII 93) ➢ C・・・ コード文字(下記参照) ➢ M・・・修飾文字(下記参照)

sendNoReadMessage (将来)

戻り値 0=成功。1=失敗。

AIM コード ID ー コード文字

コード文字	コード種別
A	Code 39、Code 39 フルアスキー、Code 32
C	Code 128、Coupon (Code 128 portion)
d	Data Matrix
E	UPC/EAN、Coupon (UPC portion)
e	GS1 DataBar (RSS)
F	Codabar
G	Code 93
H	Code 11
I	Interleaved 25
L	PDF417、Macro PDF417、Micro PDF417
M	MSI
Q	QR Code
S	Industrial 25(Discrete 25)、IATA 2 of 5
U	Maxicode
X	Code 39 Trioptic、Bookland EAN、US Postnet、US Planet、UK Postal、Japan Postal、Australian Postal、Dutch Postal

AIM コード ID　ー 修飾文字

コード種別	値	意味
Code 39	0	チェック文字なし、またはフル ASCII 処理。
	1	チェックディジット確認済み。
	3	チェックディジットは確認済みで、剥離済み。
	4	フル ASCII 変換されている。
	5	1 と 4 の両方。
	7	3 と 4 の両方。
Code 128	0	標準のデータパケット。最初の文字位置にファンクションコード 1”FNC1”はありません。
	1	ファンクションコード 1”FNC1”が最初の文字位置にあります。
	2	ファンクションコード 1”FNC1”が 2 番目の文字位置にあります。
Interleaved 25	0	チェックディジット処理なし。
	1	チェックディジット確認済み。
	3	チェックディジットは確認済みで、剥離済み。
Codabar	0	常に 0。
Code 93	0	常に 0。
MSI	0	Modulo 10 チェックディジットは確認済みで、送信済み。
	1	Modulo 10 チェックディジットは確認済みだか、送信なし。
Industrial 25 (Discrete 25)	0	常に 0。
UPC/EAN	0	(補足データなしの)13 桁の UPC-A および UPC-E で、フル EAN カントリーコードフォーマットの標準データパケット。
	1	2 桁の補足データのみ。
	2	5 桁の補足データのみ。
	4	EAN-8 データパケット。
	UPC-A アドオン 2 のバーコード” 012345678905-10 は 21 桁の文字列 ”**]E0**00012345678905**]E1**10”で送信されます。	
Bookland EAN	0	常に 0。
Trioptic Code 39	0	常に 0。
Code 11	0	シングルチェックディジット。
	1	2 チェックディジット。
	3	チェックディジット確認済み、送信なし。
GS1 DataBar	0	常に 0。
	RSS-14 と RSS Limited はアプリケーション識別子”01”で送信されます。例えば、RSS-14 バーコード、10012345678902 は、] e00110012345678902 と送信されます。	
EAN、 UCC Composites (RSS、 GS1-128、 UPC composite の 2D 部分)	通常モード	
	0	標準のデータパケット。
	1	符号化されたシンボルの区切り文字に続くデータを含むデータパケット。
	2	エスケープ文字に続くデータを含むデータパケット。データパケットは、ECI プロトコルをサポートしていません。
	3	エスケープ文字に続くデータを含むデータパケット。データパケットは、ECI プロトコルをサポートしています。
	GS1-128	
	1	データパケットは GS1-128 バーコード(すなわち、データの先頭に]JC1 が付く)となります。 ➢ GS1-128 エミュレーションモードでは、RSS は、コード 128 の規則(]JC1)で送信されます。 ➢ UPC の複合部分は UPC 規則で送信されます。
PDF417、 Micro PDF417	0	スキャンエンジンは 1994 PDF417 シンボル仕様で定義されたプロトコルに準拠するよう設定されます。 ➢ この設定で送信された場合、受信側は ECIS が呼び出されたかどうか、データバイト 92DEC が伝送で倍増されているかどうかを確実に判断することはできません。
	1	スキャンエンジンが ECI プロトコル(拡張チャネル翻訳)準拠に設定されます。すべてのデータ文字 92DEC が倍になります。

	2	スキャンエンジンはベーシックチャンネル動作(エスケープ文字伝送プロトコル)用に設定されます。データ文字 92DEC は倍になりせん。 ➢ このモードに設定すると、デコーダはバッファリングされていないマクロシンボルと、ECI エスケープシーケンス伝送でデコーダが必要なシンボルを送信することができません。
	3	バーコードは GS1-128 のシンボルを含み、最初のコードワードは、903～907、912、914、915 です。
	4	バーコードは GS1-128 のシンボルを含み、最初のコードワードは、908～909 の範囲内です。
	5	バーコードは GS1-128 のシンボルを含み、最初のコードワードは、910～911 の範囲内です。
	すべての伝送プロトコルが無効になっている PDF417 バーコード"ABCD"は、"]L2ABCD"と送信されます。	
Data Matrix	0	ECC 000-140。サポートされていません。
	1	ECC 200。
	2	ECC 200。”FNC1”が最初または 5 番目の文字位置にあります。
	3	ECC 200。”FNC1”が 2 番目または 6 番目の文字位置にあります。
	4	ECC200。ECI プロトコル実装。
	5	ECC 200。”FNC1”が最初または 5 番目の文字位置にあります。ECI プロトコル実装。
	6	ECC 200。”FNC1”が 2 番目または 6 番目の文字位置にあります。ECI プロトコル実装。
Maxicode	0	Mode 4 または 5。
	1	Mode 2 または 3。
	2	Mode 4 または 5。ECI プロトコル実装。
	3	Mode 2 または 3。ECI プロトコルは２次メッセージで実装。
QR Code	0	Mode 1。
	1	Mode 2。ECI プロトコル未実装。
	2	Mode 2。ECI プロトコル実装。
	3	Mode 2。ECI プロトコル未実装。”FNC1”が最初の文字位置にあります。
	4	Mode 2。ECI プロトコル実装。”FNC1”が最初の文字位置にあります。
	5	Mode 2。ECI プロトコル未実装。”FNC1”が 2 番目の文字位置にあります。
	6	Mode 2。ECI プロトコル実装。”FNC1”が 2 番目の文字位置にあります。

3.7.2 UPC_2D_SE4500

Upc_2D_SE4500	2D

目的　UPC-A と UPC-E のシンボル設定を構成します。

書式

```
INT  Upc_2D_SE4500 (INT rw,
                    INT *enableUpcA,
                    INT *enableUpcE,
                    INT *enableUpcE1,
                    INT *enableAddons,
                    INT *addonsRedundancy,
                    INT *transmitUpcACheckDigit,
                    INT *transmitUpcECheckDigit,
                    INT *transmitUpcE1CheckDigit,
                    INT *preambleUpcA,
                    INT *preambleUpcE,
                    INT *preambleUpcE1,
                    INT *convertUpcEtoA,
                    INT *convertUpcE1toA,
                    INT *uccCouponExtendedCode);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableUpcA

[入/出力] UPC-A 有効/無効。

0	無効。
1(※)	有効。

enableUpcE

[入/出力] UPC-E(UPC-E0)有効/無効。

0	無効。
1(※)	有効。

enableUpcE1

[入/出力] UPC-E1 有効/無効。

0(※)	無効。
1	有効。

enableAddons

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

addonsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 7(読取り照合回数)。

transmitUpcACheckDigit

[入/出力] UPC-A チェックデジット送信有無。

0	送信なし。
1(※)	送信あり。

transmitUpcECheckDigit

[入/出力] UPC-E0 チェックデジット通信有無。

0	なし。
1(※)	あり。

transmitUpcE1CheckDigit

[入/出力] UPC-E1 チェックデジット通信有無。

0	なし。
1(※)	あり。

preambleUpcA

[入/出力] UPC-A プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUpcE

[入/出力] UPC-E0 プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

preambleUpcE1

[入/出力] UPC-E1 プリアンブル送信方法。

0	送信なし。
1(※)	システム番号のみ送信。
2	システム番号と国別コードを送信。

convertUPC_EtoA

[入/出力] UPC-E0 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

convertUPC_E1toA

[入/出力] UPC-E1 から UPC-A への変換有無。

0(※)	変換なし。
1	変換あり。

uccCouponExtendedCode

[入/出力] UCC クーポンコード有効/無効。

0(※)	無効。
1	有効。

戻り値　0=成功。1=失敗。

3.7.3 EANJAN_2D_SE4500

EanJan_2D_SE4500	2D

目的　EAN/JAN-8 と ENA/JAN-13 のシンボル設定を構成します。

書式

```
INT  EanJan_2D_SE4500 (INT rw,
                       INT *enableEanJan8,
                       INT *enableEanJan13,
                       INT *enableBooklandEan,
                       INT *enableAddons,
                       INT *addonsRedundancy,
                       INT *enableEanJan8Extend,
                       INT *uccCouponExtendedCode,
                       INT *booklandIsbnFormat,
                       INT *EnableIssnEan);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableEanJan8

[入/出力] EAN/JAN-8 有効/無効。

0	無効。
1(※)	有効。

enableEanJan13

[入/出力] EAN/JAN13 有効/無効。

0	無効。
1(※)	有効。

enableBooklandEan

[入/出力] Bookland EAN 有効/無効。

➢ EAN/JAN13 は有効でなければなりません。

0	無効。
1(※)	有効。

enableAddons

[入/出力] アドオン 2 とアドオン 5 の有効/無効。

0(※)	アドオン無視。
1	アドオンのみデコード。
2	アドオンでデコード。(=自動識別)

addonsRedundancy

[入/出力]「アドオンでデコード」選択時の冗長性。

2～30	デフォルトは 10(読取り照合回数)。

enableEanJan8Extend

[入/出力] EAN/JAN-8 から EAN/JAN-13 への変換有無。

0(※)	変換なし。
1	変換あり。

uccCouponExtendedCode

[入/出力] UCC クーポンコード有効/無効。

0(※)	無効。
1	有効。

booklandIsbnFormat

[入/出力] Bookland EAN 有効時の Bookland データのフォーマット。

0(※)	Bookland ISBN-10
1	Bookland ISBN-13

EnableIssnEan

[入/出力] ISSN EAN 有効/無効。

0(※)	無効。
1	有効。

戻り値　0=成功。1=失敗。

3.7.4 Code39_2D_SE4500

Code39_2D_SE4500	2D

目的　CODE39 のシンボル設定を構成します。

書式

```
INT  Code39_2D_SE4500 (INT rw,
                       INT *enable,
                       INT *enableTrioptic,
                       INT *convertToCode32,
                       INT *prefix Code32,
                       INT *checkDigitVerification,
                       INT *transmitCheckDigit,
                       INT *fullASCII,
                       INT *length1,
                       INT *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code39 有効/無効。

0	無効。
1(※)	有効。

enableTrioptic

[入/出力] Trioptic Code39 有効/無効。

0(※)	無効。
1	有効。

convertToCode32

[入/出力] Code39 から Code32(Italian Pharmacode)への変換有無。

0(※)	変換なし。
1	変換あり。

prefix Code32

[入/出力] Code32 プリフィックスの送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

fullASCII

[入/出力] Code39 フルアスキーのサポート有無。

0(※)	なし。(Standard Code 39)
1	あり。(Code 39 Full ASCII)

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。1=失敗。

3.7.5 Code93_2D_SE4500

Code93_2D_SE4500	2D

目的　CODE93 のシンボル設定を構成します。

書式

```
INT  Code93_2D_SE4500 (INT rw,
                       INT *enable,
                       INT *length1,
                       INT *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code93 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。1=失敗。

3.7.6 Interleaved2Of5_2D_SE4500

Interleaved2Of5_2D_SE4500	2D

目的　Interleaved 25 のシンボル設定を構成します。

書式

```
INT  Interleaved2Of5_2D_SE4500 (INT rw,
                                INT *enable,
                                INT *length1,
                                INT *length2,
                                INT *checkDigitVerification,
                                INT *transmitCheckDigit
                                INT *convertToEAN13);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Interleaved 25 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	USS チェックディジット。
2	OPCC チェックディジット。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

convertToEAN13

[入/出力] EAN-13 変換有無。

- まず、チェックディジット検査が無効でなければなりません。またバーコードが先行ゼロで、有効な EAN-13 チェックディジットがある場合のみ機能します。

0(※)	変換なし。
1	変換あり。

戻り値　0=成功。1=失敗。

3.7.7 INDUSTRIAL2OF5_2D_SE4500

INDUSTRIAL2OF5_2D_SE4500	**2D**

目的　Industrial 25 (Discrete 25)のシンボル設定を構成します。

書式

```
int  INDUSTRIAL2OF5_2D_SE4500 (int rw,
                               int *enable,
                               int *length1,
                               int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Industrial 25 (Discrete 25)有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。1=失敗。

3.7.8 Matrix2Of5_2D_SE4500

Matrix2Of5_2D_SE4500	2D

目的　Matrix25 のシンボル設定を構成します。

書式

```
INT  Matrix2Of5_2D_SE4500 (INT rw,
                           INT *enable,
                           INT *length1,
                           INT *length2,
                           INT *redundancy,
                           INT *checkDigitVerification,
                           INT *transmitCheckDigit);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Matrix25 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

redundancy

[入/出力] デコードの冗長性有効/無効。

0(※)	無効。
1	有効。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

戻り値　0=成功。1=失敗。

3.7.9 CHINESE2OF5_2D_SE4500

CHINESE2OF5_2D_SE4500	**2D**

目的　Chinese 25 のシンボル設定を構成します。

書式　int CHINESE2OF5_2D_SE4500 (int rw,
int *enable)

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Chinese 25 有効/無効。

0	無効。
1(※)	有効。

戻り値　0=成功。1=失敗。

3.7.10 Codabar_2D_SE4500

Codabar_2D_SE4500	**2D**

目的　Codabar のシンボル設定を構成します。

書式

```
int  Codabar_2D_SE4500 (int rw,
                        int *enable,
                        int *length1,
                        int *length2,
                        int *enableCLSI_Editing,
                        int *enableNOTIS_Editing);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Codabar 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

enableCLSI_Editing

[入/出力] CLSI 編集有無。

0(※)	編集なし。
1	編集あり。

- CLSI 編集ありの場合、スタート/ストップ文字を消去し、1 文字目、5 文字目、10 文字目の後にスペースを挿入します。
- 14 文字のバーコードの長さにスタート/ストップ文字は含みません。

enableNOTIS_Editing

[入/出力] NOTIS 編集有無。

0(※)	編集なし。
1	編集あり。

- NOTIS 編集はスタート/ストップ文字を取り除きます。つまり「スタート/ストップ文字送信無効」と同じ意味になります。

戻り値　0=成功。それ以外=1。

3.7.11 MSI_2D_SE955

MSI_2D_SE4500	2D

目的　MSI のシンボル設定を構成します。

書式

```
int  MSI_2D_SE4500 (int rw,
                    int *enable,
                    int *length1,
                    int *length2,
                    int *checkDigitVerification,
                    int *transmitCheckDigit
                    int *checkDigitAlgorithm);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] MSI 有効/無効。

0	無効。
1(※)	有効。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

checkDigitVerification

[入/出力] チェックディジット検査有無。

0(※)	なし。
1	あり。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

checkDigitAlgorithm

[入/出力] 適用するアルゴリズム。

0	Modulo 10 / Modulo 11。
1(※)	Double Modulo 10。

戻り値　0=成功。1=失敗。

3.7.12 GS1_DATABAR_2D_SE4500

GS1_DATABAR_2D_SE4500	2D

目的　GS1 DataBar (RSS family)のシンボル設定を構成します。

書式

```
int  GS1_DATABAR_2D_SE4500 (int rw,
                    int *enableGS1_DataBarOmnidirectional,
                    int *enableGS1_DataBarLimited,
                    int *enableGS1_DataBarExpanded,
                    int *convertToUpcEan);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableGS1_DataBarOmnidirectional

[入/出力] GS1 DataBar Omnidirectional (RSS-14)有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarLimited

[入/出力] GS1 DataBar Limited (RSS Limited) 有効/無効。

0	無効。
1(※)	有効。

enableGS1_DataBarExpanded

[入/出力] GS1 DataBar Expanded (RSS Expanded)有効/無効。

0	無効。
1(※)	有効。

convertToUpcEan

[入/出力] RSS の UPC/EAN 変換有無。

0(※)	変換なし。
1	変換あり。

戻り値　0=成功。1=失敗。

3.7.13 Code128_2D_SE4500

Code128_2D_SE4500	**2D**

目的　Code 128 のシンボル設定を構成します。

書式

```
int  Code128_2D_SE4500 (int rw,
                        int *enableCode128,
                        int *enableGS1_128,
                        int *enableISBT128,
                        int *enableIsbtConcatenation,
                        int *isbtConcatenationRedundancy);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableCode128

[入/出力] Code 128 有効/無効。

0	無効。
1(※)	有効。

enableGS1_128

[入/出力] GS1-128 有効/無効。

0	無効。
1(※)	有効。

enableISBT128

[入/出力] ISBT 128 有効/無効。

0	無効。
1(※)	有効。

enableIsbtConcatenation

[入/出力] ISBT バーコードの連結有無。

0(※)	連結無効。 ➢ ペアの ISBT バーコードは連結されません。
1	連結有効。 ➢ ただし、ISBT バーコードが２つない場合、デコードと連結は行われません。シングル ISBT バーコードはデコードできません。
2	自動判別有効。 ➢ ペアの ISBT バーコードはデコード、連結されます。シングル ISBT バーコードの場合、スキャナは送信前に 2 つめのバーコードがないことを確認するために 10 回はデコードします。

isbtConcatenationRedundancy

[入/出力] ISBT 連結が自動判別となっている場合の連結冗長性(2～20 回)を指定する値。デフォルトは 10。

戻り値　0=成功。1=失敗。

3.7.14 CODE11_2D_SE4500

CODE11_2D_SE4500	2D

目的　Code 11 のシンボル設定を構成します。

書式

```
int  CODE11_2D_SE4500 (int rw,
                       int *enable,
                       int *numberOfCheckDigits,
                       int *transmitCheckDigit,
                       int *length1,
                       int *length2);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable

[入/出力] Code 11 有効/無効。

0	無効。
1(※)	有効。

numberOfCheckDigits

[入/出力] チェックディジット検査。

0(※)	なし。
1	1 チェックディジット。
2	2 チェックディジット。

transmitCheckDigit

[入/出力] チェックディジット送信有無。

0(※)	送信なし。
1	送信あり。

length1

[入/出力] バーコードの長さ。

0～55	デフォルトは 4

length2

[入/出力] バーコードの長さ。

0～55	デフォルトは 55

- Length1<Length2 の場合、読取り最少桁数、読取り最大桁数となります。それ以外の場合は固定長となります。

戻り値　0=成功。1=失敗。

3.7.15 POSTALCODE_2D_SE4500

PostalCode_2D_SE4500	**2D**

目的　Postal Code のシンボル設定を構成します。

書式

```
int  PostalCode_2D_SE4500 (int rw,
                           int *enableUSPostnet,
                           int *enableUSPlanet,
                           int *enableUKPostal,
                           int *transmitUKCheckDigit,
                           int *enableJapanPostal,
                           int *enableAustralianPostal,
                           int *enableDutchPostal,
                           int *transmitUSCheckDigit);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableUSPostnet

[入/出力] US Postnet 有効/無効。

0	無効。
1(※)	有効。

enableUSPlanet

[入/出力] US Planet 有効/無効。

0	無効。
1(※)	有効。

enableUKPostal

[入/出力] UK Postal 有効/無効。

0	無効。
1(※)	有効。

transmitUKCheckDigit

[入/出力] UK Postal のチェックディジット送信有無。

0	送信なし。
1(※)	送信あり。

enableJapanPostal

[入/出力] Japan Postal 有効/無効。

0	無効。
1(※)	有効。

enableAustralianPostal

[入/出力] Australian Postal 有効/無効。

0	無効。
1(※)	有効。

enableDutchPostal

[入/出力] Dutch Postal 有効/無効。

0	無効。
1(※)	有効。

transmitUSCheckDigit
[入/出力] US Postal のチェックディジット送信有無。

0	送信なし。
1(※)	送信あり。

戻り値　0=成功。1=失敗。

3.7.16 COMPOSITE_2D_SE4500

Composite_2D_SE4500	**2D**

目的　Composite バーコードのシンボル設定を構成します。

書式
```
int  Composite_2D_SE4500 (int rw,
                          int *enableCC_C,
                          int *enableCC_AB,
                          int *enableTLC39,
                          int *enableUpcMode,
                          int *beepMode,
                          int *enableEmulationMode);
```

引数　※印はデフォルト値です。

rw
[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enableCC_C
[入/出力] Compsite CC-C 有効/無効。

0	無効。
1(※)	有効。

enableCC_AB
[入/出力] Compsite CC-A/B 有効/無効。

0(※)	無効。
1	有効。

enableTLC39
[入/出力] TLC39(TCIF Linked Code 39)有効/無効。

0(※)	無効。
1	有効。

enableUpcMode
[入/出力] UPC Composite モード有効/無効。
- 送信中、UPC バーコードと 2D バーコードが 1 つのバーコードであるかのように“リンク”されます。

0	UPC リンクなし。(UPC、2D いずれのバーコードもある場合は 2D 部を破棄)
1(※)	常に UPC リンクあり。(2D バーコードがない場合、バーコードデータ全部を破棄)
2	UPC Composite を自動判別。

beepMode (将来)

enableEmulationMode
[入/出力] UCC/EAN Composite Code の場合の GS-1 エミュレーションモード有効/無効。

0(※)	無効。
1	有効。

戻り値　0=成功。1=失敗。

3.7.17 INVERSE1D_2D_SE4500

Inverse1D_2D_SE4500	**2D**

目的　1D(白黒)反転バーコードのシンボル設定を構成します。

書式　int Inverse1D_2D_SE4500 (int rw,
int *enable1DInverse);

引数　※印はデフォルト値です。
rw
[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable1DInverse
[入/出力] 1D 反転バーコード有効/無効。

0(※)	通常バーコードのみ。
1	反転バーコードのみ。
2	反転自動検出。通常バーコード、反転バーコードともに読取り。

戻り値　0=成功。1=失敗。

3.7.18 KOREAN3OF5_2D_SE4500

Korean3Of5_2D_SE4500	**2D**

目的　Korean 3oof 5 のシンボル設定を構成します。

書式　int Korean3Of5_SE4500 (int rw,
int *enable);

引数　※印はデフォルト値です。
rw
[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enable
[入/出力] Korean 3 of 5 有効/無効。

0(※)	無効。
1	有効。

戻り値　0=成功。1=失敗。

3.7.19 SYMBOLOGIES_2D_SE4500

Symbologies_2D_SE4500	**2D**

目的　2D シンボル設定を構成します。

書式

```
int  Symbologies_2D_SE4500 (int rw,
                            int *enablePDF417,
                            int *enableMicroPDF417,
                            int *enableCode128Emulation,
                            int *enableDataMatrix,
                            int *enableDataMatrixInverse,
                            int *decodeMirrorImage,
                            int *enableMaxicode,
                            int *enableQRCode,
                            int *enableQRCodeInverse,
                            int *enableMicroQR,
                            int *enableAztec,
                            int *enableAztecInverse);
```

引数　※印はデフォルト値です。

rw

[入力]オペレーション。

“r”	Reader.ReaderEngineAPI.READ_PARAM	設定読込み。
“w”	Reader.ReaderEngineAPI.WRITE_PARAM	設定書込み。

enablePDF417

[入/出力] PDF417 有効/無効。

0	無効。
1(※)	有効。

enableMicroPDF417

[入/出力] MicroPDF417 有効/無効。

0(※)	無効。
1	有効。

enableCode128Emulation

[入/出力] 特定の MicroPDF417 用 Code 128 エミュレーション有効/無効。

0(※)	無効。
1	有効。

- まず、AIM 識別子を有効にする必要があります。
- 適用された場合、MicroPDF417 バーコードは以下のプリフィックスのある Code 128 をデコードしたかのように送信されます。
  - MicroPDF417 の最初の文字列が 903～907、912、914、915。
    元のコードの”]L3”は”]C1”に変換されます。
  - MicroPDF417 の最初の文字列が 908 または 909。
    元のコードの”]L4”は”]C2”に変換されます。
  - MicroPDF417 の最初の文字列が 910 または 911。
    元のコードの”]L5”は”]C0”に変換されます。

enableDataMatrix

[入/出力] DataMatrix 有効/無効。

0	無効。
1(※)	有効。

enableDataMatrixInverse

[入/出力] DataMatrix(白黒)反転バーコード有効/無効。

0(※)	通常バーコードのみ。
1	反転バーコードのみ。
2	反転自動検出。通常バーコード、反転バーコードともに読取り。

decodeMirrorImage

[入/出力] DataMatrix ミラー(左右反転)バーコード有効/無効。

0(※)	なし。
1	常時。
2	自動。

enableMaxicode

[入/出力] Maxicode 有効/無効。

0	無効。
1(※)	有効。

enableQRCode

[入/出力] QR Code 有効/無効。

0	無効。
1(※)	有効。

enableQRCodeInverse

[入/出力] QR Code(白黒)反転バーコード有効/無効。

0(※)	通常バーコードのみ。
1	反転バーコードのみ。
2	反転自動検出。通常バーコード、反転バーコードともに読取り。

enableMicroQRCode

[入/出力] MicroQR 有効/無効。

0	無効。
1(※)	有効。

enableAztec

[入/出力] Aztec 有効/無効。

0	無効。
1(※)	有効。

enableAztecInverse

[入/出力] Aztec (白黒)反転バーコード有効/無効。

0(※)	通常バーコードのみ。
1	反転バーコードのみ。
2	反転自動検出。通常バーコード、反転バーコードともに読取り。

戻り値　0=成功。1=失敗。

3.8 リーダのリセット

ResetReaderToDefault

目的 1D レーザーバーコードリーダを設定します。

書式 INT ResetReaderToDefault (INT readerType);

引数 readerType

[入力]リセットするリーダの種別。

7	DC_READER_BC	バーコードリーダのみ。
32	DC_READER_RFID	RFID リーダのみ。
255	ALL_DEVICE	両方。

ｺｰﾃﾞｨﾝｸﾞ例 INT nRlt;

nRlt = ResetReaderToDefault(DC_READER_BC);

戻り値 1=成功。それ以外=失敗。失敗するとエラーを返します。

-101	E_READER_NOT_INIT
-111	E_READER_BC_RESET_FAILED
-112	E_READER_RFID_RESET_FAILED

備考 リセットに約 2 秒かかります。

参照 InitReader

4　システム API

ダイナミックリンクは下位互換性のために推奨されています。.lib について将来にリリースされる.lib との互換性を保証するものではありません。そのため、最新の.lib で再コンパイルする必要性がある可能性を排除しないでください。
RS-232 データ接続では、モバイルコンピュータの COM0 を使用します。
同梱の CD=ROM で使用するヘッダやライブラリを探してください。

- SystemMobile.h
- SystemMobile.lib

[注]：システム DLL は OS ディレクトリ内に置くことでアクセス可能です。

4.1 システム設定

4.1.1 汎用固有番号識別子(UUID)

GetHALUUID

目的

OEM 定義デバイスハードウェア識別子に基づいた汎用固有番号識別子(UUID)を取得します。

書式

DWORD GetHALUUID (GUID *guid);

引数

guid
[出力]UUID 情報が格納される GUID 構造体へのポインタ。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

1	UUID にアクセスできません。
4	ERROR_PARAMETER(パラメータの誤り)

備考

汎用固有番号識別子(UUID)は、オブジェクトを識別するために使用されるユニークな 128 ビット(16 バイト)の識別子規格です。グローバル固有番号識別子(GUID)は、UUID スタンダードのマイクロソフト実装です。

4.1.2 デバイス名

GetSysDevName

目的　モバイルコンピュータで、「スタート画面」→「設定」→「システム」に表示されているのと同じデバイス名を取得します。

書式　DWORD GetSysDevName (WCHAR *deviceName
BYTE nameSize);

引数　deviceName
[出力]デバイス名が格納されるバッファへのポインタ。
nameSize
[入力]出力バッファのサイズ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

2	ERROR_NOSPACE(バッファサイズの誤り)
4	ERROR_PARAMETER(パラメータの誤り)

備考　初期設定では、工場出荷時のデバイス名が格納されています。

SetSysDevName

目的　デバイス名を変更します。

書式　DWORD SetSysDevName (WCHAR *deviceName
BYTE nameSize);

引数　deviceName
[入力]デバイス名が格納されているバッファへのポインタ。
nameSize
[入力]デバイス名の長さ。最大 15 文字まで。スペース文字は使用できません。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

備考　初期設定では、工場出荷時のデバイス名が格納されています。
- 新しい名称を有効にするにはウォームブートする必要があります。
- デバイス名は最大 15 文字までです。使用できる文字は'A'～'Z'、'a'～'z'、'0'～'9'、"-"、"_"です。最初の文字は'A'～'Z'、'a'～'z'でなければなりません。最後の文字に"-"、"_"は使用できません。

4.1.3 システム情報

GetSysInfo

目的 モバイルコンピュータで、「スタート画面」→「設定」→「システム」に表示されているのと同じデバイス名を取得します。

書式 DWORD GetSysInfo (SYSINFO *sysInfo);

引数 sysInfo
[出力]システム情報が格納される SYSINFO 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

4.1.4 プログラムスタート

SetInitLoaderBind

目的 システムが初期化された後にロードするプログラムをバインドします。

書式 DWORD SetInitLoaderBind (KEYPADBIND *keypadBind);

引数 keypadBind
[入力]プログラム情報が格納されている KEYPADBIND 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	レジストリを開けません。
2	レジストリに書き込めません。
4	ERROR_PARAMETER(パラメータの誤り)

備考 バインドしたプログラムは AutoRun.ini より先に実行され、AutoRun.exe を上書きします。(AutoRun.exe により実行されることはありません。)

4.1.5 ソフトリセット

SystemSoftReset

目的 ソフトウェアリセット(ウォームブート)を実行します。

書式 DWORD SystemSoftReset (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)

4.2 状態識別

4.2.1 LED ライト

SetLEDLight

目的　ステータスランプ(LED)を操作します。

書式　DWORD SetLEDLight (BYTE color
BOOL onoff);

引数　color
[入力]BYTE 変数。

0x00	赤色。
0x01	緑色。
0x02	琥珀色。

onoff
[入力]Boolean 変数。

0	消灯。
1	点灯。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

備考　オンとオフの間には 400ms 以上の間隔をあけてください。

4.2.2 バイブレータ

GetVibratorPower

目的 現在のバイブレータの状態を取得します。

書式 DWORD GetVibratorPower (BYTE *onoff);

引数 onoff
[出力]情報が格納されるバッファへのポインタ。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

SetVibratorPower

目的 バイブレータの状態を設定します。

書式 DWORD SetVibratorPower (BYTE onoff);

引数 onoff
[入力]BYTE 変数。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

4.3 LCD バックライト

4.3.1 バックライトコントロール

GetBacklightCTL

目的 現在のバックライトコントロールの設定を取得します。

書式 DWORD GetBacklightCTL(BKLCTL *bklCtl);

引数 bklCtl
[出力]設定が格納される BKLCTL 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

SetBacklightCTL

目的 バックライトコントロールの設定を変更します。

書式 DWORD SetBacklightCTL(BKLCTL *bklCtl);

引数 bklCtl
[入力]設定が格納されている BKLCTL 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
16	バッテリーモードでのバックライトレベルが範囲外。
32	AC モードでのバックライトレベルが範囲外。

4.3.2 バックライトレベル

GetBacklightLV

目的 現在のバックライトコントロールの設定を取得します。

書式 DWORD GetBacklightLV (BYTE byFromAC,
BYTE *byLevel);

引数 byFromAC
[入力]BYTE 変数。

0	バッテリーモードのバックライトレベル
1	AC モードのバックライトレベル

byLevel
[出力]情報が格納されるバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

備考 バックライトレベルの範囲は 0(暗い)～11(明るい)です。
システムの初期設定はバッテリーモードでレベル 6、AC モードでレベル 11 です。

SetBacklightLV

目的　バックライトコントロールを設定します。

書式　DWORD SetBacklightLV (BYTE byFromAC,
BYTE byLevel);

引数　byFromAC
[入力]BYTE 変数。

0	バッテリーモードのバックライトレベル
1	AC モードのバックライトレベル

byLevel
[入力]BYTE 変数。レベル 0(暗い)～レベル 11(明るい)。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.3.3 バックライトを初期設定にリセット

SetBacklightDefault

目的　バックライトコントロールを初期設定に変更します。

書式　DWORD SetBacklightDefault (VOID);

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
16	バッテリーモードでのバックライトレベルが範囲外。
32	AC モードでのバックライトレベルが範囲外。

備考　システム初期値は次の通りです。
DWORD batttimeout=30;
DWORD lightlevel=3;
DWORD actimeout=60;
DWORD aclightlevel=6;
DWORD backlightontap=1;
DWORD acbacklightontap=1;
DWORD usebattery=1;
DWORD useext=0;

参照　BKLCTL

4.3.4 バックライトを最大に設定

SetBacklightMax

目的 バックライトコントロールを最大に設定します。

書式 DWORD SetBacklightMax (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
16	バッテリーモードでのバックライトレベルが範囲外。
32	AC モードでのバックライトレベルが範囲外。

備考 最大値は次の通りです。
DWORD batttimeout=300;
DWORD lightlevel=10;
DWORD actimeout=600;
DWORD aclightlevel=10;
DWORD backlightontap=1;
DWORD acbacklightontap=1;
DWORD usebattery=1;
DWORD useext=1;

参照 SetBacklightDefault , BKLCTL

4.3.5 バックライトを最小に設定

SetBacklightMin

目的 バックライトコントロールを最小に設定します。

書式 DWORD SetBacklightMin (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
16	バッテリーモードでのバックライトレベルが範囲外。
32	AC モードでのバックライトレベルが範囲外。

備考 最小値は次の通りです。
DWORD batttimeout=30;
DWORD lightlevel=0;
DWORD actimeout=60;
DWORD aclightlevel=0;
DWORD backlightontap=0;
DWORD acbacklightontap=1;
DWORD usebattery=1;
DWORD useext=1;

参照 SetBacklightDefault , BKLCTL

4.4 キーパッドバックライト

GetKeylightOnOff

目的 現在のキーパッドバックライトの状態を取得します。

書式 DWORD GetKeylightOnOff (BYTE *onoff);

引数 onoff
[出力]情報が格納されるバッファへのポインタ。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

SetKeylightOnOff

目的 キーパッドバックライトの状態を設定します。

書式 DWORD SetKeylightOnOff (BYTE onoff);

引数 onoff
[入力]BYTE 変数。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.5 タッチパネル

4.5.1 タッチパネルの状態

GetTchLock

目的 現在のタッチパネルのロック状態を取得します。

書式 DWORD GetTchLock (BYTE *lockState);

引数 lockState
[出力]情報が格納されるバッファへのポインタ。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

SetTchLock

目的 タッチパネルのロック状態を設定します。

書式 DWORD SetTchLock (BYTE lockState);

引数 lockState
[入力]BYTE 変数。

0	オフ。
1	オン。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
5	ERROR_API_FUNCTION(API 呼び出しに失敗)

[注]：タッチパネルとキーパッドは同時にロックできません。

4.6 キーパッド

4.6.1 感度

GetKeypadSensitivity

目的 キーが 1 秒以上押されているときに繰り返す文字数を取得します。

書式 DWORD GetKeypadSensitivity (DWORD *value);

引数 value
[出力]情報が格納されるバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

備考 値が大きいほど、1 秒以上キーが押されたときに。より多く押したキーの文字が繰り返されます。
- 初期値は、1 秒ごとに 6 文字です。

SetKeypadSensitivity

目的 キーが 1 秒以上押されているときに繰り返す文字数を設定します。

書式 DWORD SetKeypadSensitivity (DWORD value);

引数 value
[入力]32BIT 符号なし変数。
値が大きいほど、1 秒以上キーが押されたときに。より多く押したキーの文字が繰り返されます。(最大は 1 秒当たり 20 文字)

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.6.2 キーパッドのロック

GetKeypadLock

目的　キーパッドのロック状態を取得します。

書式　DWORD GetKeypadLock (DWORD key,
DWORD *lockState);

引数　key
[入力] 32BIT 符号なし変数。パラメータの値は次の通りです。この値は OR の組み合わせで使用することはできません。

1	KEY_GENERAL	一般入力キー。
2	KEY_ALPHA	[Alpha]キー。
4	KEY_FN	[Fn]キー。
32	KEY_SCAN	[Scan]キー。
64	KEY_POWER	[Power]キー。
128	KEY_LSTRIGGER	左のトリガーキー。
256	KEY_RSTRIGGER	右のトリガーキー。
512	KEY_SHIFT	[Shift]キー。
0xFFFFFFFF	KEY_ALL	キーパッド全体。

lockState
[出力]情報が格納されるバッファへのポインタ。

0	ロック解除
1	ロック

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	キーハンドル取得失敗
4	ERROR_PARAMETER(パラメータの誤り)

備考　KEY_ALL = KEY_GENERAL + KEY_ALPHA + KEY_FN + KEY_SCAN + KEY_POWER + KEY_LSTRIGGER + KEY_RSTRIGGER + KEY_SHIFT。

4.6.3 キーパッドモード

- 自動再開モード - ファンクションキーを押すことにより切り替わります。組み合わせの 2 つ目のキーを押すことで OFF に切り替わります。ステータスを表すアイコンが画面上に表示されます。
- トグルモード - ファンクションキーを押すことにより切り替わります。再度ファンクションキーを押すと OFF に切り替わります。ステータスを表すアイコンが画面上に表示されます。
- マルチキーモード – 組み合わせになっている任意の 2 つのキーは同時に押すか、２つ目のキーを押してからファンクションキーを押します。

GetKeypadMode

目的　特殊キーのモードを取得します。

書式　DWORD GetKeypadMode (DWORD key,
DWORD *mode);

引数　key
[入力] 32BIT 符号なし変数。

4	KEY_FN	[Fn]キー。

mode
[出力]情報が格納されるバッファへのポインタ。

0	トグルモード
1	自動再開モード
2	マルチキーモード

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	キーハンドル取得失敗
4	ERROR_PARAMETER(パラメータの誤り)

SetKeypadMode

目的　特殊キーのモードを設定します。

書式　DWORD SetKeypadMode (DWORD key,
DWORD mode);

引数　key
[入力] 32BIT 符号なし変数。

4	KEY_FN	[Fn]キー。

mode
[入力] 32BIT 符号なし変数。

0	トグルモード
1	自動再開モード
2	マルチキーモード

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	キーハンドル取得失敗
4	ERROR_PARAMETER(パラメータの誤り)

4.6.4 キーパッド状態

GetKeypadState

目的　特殊キーの状態を取得します。

書式　DWORD GetKeypadState (DWORD key,
DWORD *state);

引数　key
[入力] 32BIT 符号なし変数。

2	KEY_ALPHA	[Alpha]キー。
4	KEY_FN	[Fn]キー。

state
[出力]情報が格納されるバッファへのポインタ。

	Alpha キー	FN キー
0	数値	アクティブでない
1	該当なし	アクティブ
2	アルファベット小文字	該当なし

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	キーハンドル取得失敗
4	ERROR_PARAMETER(パラメータの誤り)

SetKeypadState

目的　特殊キーの状態を設定します。

書式　DWORD SetKeypadState(DWORD key,
DWORD state);

引数　key
[入力] 32BIT 符号なし変数。

2	KEY_ALPHA	[Alpha]キー。
4	KEY_FN	[Fn]キー。

state
[入力] 32BIT 符号なし変数。

	Alpha キー	FN キー
0	数値	アクティブでない
1	該当なし	アクティブ
2	アルファベット小文字	該当なし

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	キーハンドル取得失敗
2	キーはロックされています
4	ERROR_PARAMETER(パラメータの誤り)

4.7 マイク

4.7.1 マイクデバイス

SetAMicGain

目的 カスタマイズされたマイクデバイスの増加レベルを設定します。

書式 DWORD SetAMicGain (unsigned int level);

引数 value

[入力]符号なし int 変数。マイクデバイスの増加レベル。

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

GetAMicGain

目的 カスタマイズされたマイクデバイスの増加レベルを取得します。

書式 DWORD GetAMicGain (unsigned int *level);

引数 value

[出力]符号なし int 変数。マイクデバイスの増加レベル。

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

➢ この関数を使う前に、まず SetAMicGain で初期レベルを設定してください。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.7.2 ヘッドセットマイク

SetHSMicGain

目的 カスタマイズされたヘッドセットマイクの増加レベルを設定します。

書式 DWORD SetHSMicGain (unsigned int level);

引数 level

[入力]符号なし int 変数。ヘッドセットマイクの増加レベル。

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

GetHSMicGain

目的 カスタマイズされたヘッドセットマイクの増加レベルを取得します。

書式 DWORD GetHSMicGain (unsigned int *level);

引数 level

[出力]符号なし int 変数。ヘッドセットマイクの増加レベル。

0	0
1	7281
2	14563
3	21845
4	29126
5	36408
6	43690
7	50971
8	58253
9	65535

- この関数を使う前に、まず SetHSMicGain で初期レベルを設定してください。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.8 ワイヤレス LAN

4.8.1 Wi-Fi 電源

GetWiFiPower

目的 現在の Wi-Fi モジュールの電源状態を取得します。

書式 DWORD GetWiFiPower (BYTE *onoff);

引数 onoff
[出力]情報が格納されるバッファへのポインタ。

0	Wi-Fi モジュールの電源 OFF
1	Wi-Fi モジュールの電源 ON

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	電源状態取得失敗
4	ERROR_PARAMETER(パラメータの誤り)

SetWiFiPower

目的 Wi-Fi モジュールの電源状態を設定します。

書式 DWORD SetWiFiPower(BYTE onoff);

引数 onoff
[入力]BYTE 変数。

0	Wi-Fi モジュールの電源 OFF
1	Wi-Fi モジュールの電源 ON

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	Wi-Fi デバイスハンドルオープン失敗
2	Wi-Fi 電源設定失敗
4	ERROR_PARAMETER(パラメータの誤り)

4.8.2 無線

RadioDisable

目的 無線を無効に設定します。

書式 SDCERR RadioDisable (VOID);

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

RadioEnable

目的 無線を有効に設定します。

書式 SDCERR RadioEnable (VOID);

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.3 IP 情報

GetWlanIpInfo

目的 現在のワイヤレスネットワークの IP 情報を取得します。

書式 DWORD GetWlanIpInfo (WLANADPTINFO *adpt);

引数 adpt
[出力]情報が格納される WLANADPTINFO 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

備考 DHCP 有効の場合は、DHCP ステータスのみ取得します。

SetWlanIpInfo

目的 ワイヤレスネットワークの IP 情報を設定します。

書式 DWORD SetWlanIpInfo (WLANADPTINFO *adpt);

引数 adpt
[入力] 情報が格納されている WLANADPTINFO 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)

4.8.4 ドメイン、SDK、MAC 情報

GetCurrentDomain

目的 規制ドメインまたは無線設定のドメインを取得します。

書式 REG_DOMAIN GetCurrentDomain (VOID);

戻り値 取得に成功すると、以下の規制ドメインを返します。

0	REG_FCC	(アメリカ)
1	REG_ETSI	(ヨーロッパ)
2	REG_TELEC	(日本)
3	REG_WW	(ワールドワイド)
4	REG_KCC	(韓国)

備考 この関数は SROM から値を取得するので実行時間が重要になります。そのため、頻繁にこの関数を呼び出さないでください。

GetSDKVersion

目的 無線を無効に設定します。

書式 SDCERR GetSDKVersion (DWORD *version);

引数 version
[出力] 情報が格納されるバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

GetWiFiMac

目的 サミット無線の MAC アドレスを取得します。

書式 DWORD GetWiFiMac (ULONGLONG *wifiMac);

引数 wifiMac
[出力]情報が格納される ULONGLONG 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
2	ERROR_API_FUNCTION(API 呼び出しに失敗)
3	Wi-Fi 電源が入ってない
4	ERROR_PARAMETER(パラメータの誤り)

4.8.5 ネットワーク一覧

AddWlanSsidToPreferredList

目的 優先リストにワイヤレスネットワーク(アクセスポイント経由)を追加します。

書式 DWORD AddWlanSsidToPreferredList (WLANCTL *wlanCtl1);

引数 wlanCtl1
[入力]構造体変数。WLANCTL を参照してください。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE(リソース取得失敗)
4	ERROR_PARAMETER(パラメータの誤り)
8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)
16	ERROR_WLAN_SSID_REPEAT

参照 GetWlanPreferredList, ResetWlanPreferredList

GetWlanAvailableList

目的 利用可能なワイヤレスネットワーク(アクセスポイント経由)のリストを取得します。

書式 DWORD GetWlanAvailableList (RAW_DATA* pAvailableList);

引数 pAvailableList
[入/出力]リストが格納される RAW_DATA 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

2	ERROR_NOSPACE(バッファが小さすぎます)
4	ERROR_PARAMETER(パラメータの誤り)
8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)
19	ERROR_WLAN_NO_AVAILABLE_LIST

備考 システム API のサンプルコードを参照してください。

参照 GetWlanPreferredList

GetWlanPreferredList

目的 優先ワイヤレスネットワーク(アクセスポイント経由)のリストを取得します。

書式 DWORD GetWlanPreferredList (RAW_DATA* pPreferredList);

引数 pPreferredList
[入/出力]リストが格納される RAW_DATA 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

2	ERROR_NOSPACE(バッファが小さすぎます)
4	ERROR_PARAMETER(パラメータの誤り)
8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)
17	ERROR_WLAN_NO_PRSSID_LIST

備考 システム API のサンプルコードを参照してください。

参照 AddWlanSsidToPreferredList, GetWlanAvailableList, ReconnectWlanPreferredList, ResetWlanPreferredList

ResetWlanPreferredList

目的 優先ワイヤレスネットワーク(アクセスポイント経由)のリストをクリアします。

書式 DWORD ResetWlanPreferredList (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)

備考 現在の接続も終了します。

参照 RAW_DATA, AddWlanSsidToPreferredList

4.8.6 ネットワークステータス

GetWlanConnectedStatus

目的 現在の接続ステータスを取得します。

書式 DWORD GetWlanConnectedStatus (WLANCONNECTEDST* pWlanConnSt);

引数 pWlanConnSt
[入/出力]情報が格納される WLANCONNECTEDST 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER(パラメータの誤り)
8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)
18	ERROR_WLAN_NO_ASSOCIATED
20	ERROR_WLAN_NO_CONNECTED

ReconnectWlanPreferredList

目的 優先ワイヤレスネットワークのリストにあるアクセスポイントに再接続します。

書式 DWORD ReconnectWlanPreferredList (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

8	ERROR_WLAN_QUIRY_INTERFACE(インターフェース問い合わせ失敗)

備考 現在の接続も終了します。

参照 RAW_DATA, GetWlanPreferredList

4.8.7 BSSID

ScanWiFiBSSID

目的 利用可能なアクセスポイントをスキャンします。

書式 DWORD ScanWiFiBSSID (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER(パラメータの誤り)

GetWiFiBSSIDList

目的 BSSID のリストを取得します。

書式 DWORD GetWiFiBSSIDList (BYTE* buffer);

引数 buffer
[出力]情報が格納されるバッファへのポインタ。バッファサイズは
sizeof(NDISUIO_QUERY_OID) + sizeof(NDIS_802_11_BSSID_LIST_EX) + 50 *
sizeof(NDIS_WLAN_BSSID_EX)より大きいサイズにしてください。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER(パラメータの誤り)

4.8.8 構成プロファイル

ActivateConfig

目的 構成プロファイルをアクティブにします。

書式 SDCERR ActivateConfig (DWORD *name);

引数 name
[入力] プロファイル名が格納されているバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

GetCurrentConfig

目的 現在の使用中の構成を取得します。

書式 SDCERR GetCurrentConfig (DWORD *num,
CHAR *name);

引数 num
[出力]情報が格納されるバッファへのポインタ。

0	サードパーティの構成プロファイル。(将来)
1～	アクティブな構成プロファイルの識別子。

name
[出力] プロファイル名が格納されるバッファへのポインタ。
➢ バッファサイズは CONFIG_NAME_SZ より大きいサイズにしてください。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

GetNumConfigs

目的 利用可能なプロファイル数を取得します。

書式 SDCERR GetNumConfigs(DWORD *num);

引数 num
[出力]情報が格納されるバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.9 構成プロファイル編集

AddConfig

目的	構成プロファイルを追加します。
書式	SDCERR AddConfig (SDCConfig *config);
引数	config [入力]構成が格納されている SDCConfig 構造体へのポインタ。
戻り値	0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

DeleteConfig

目的	構成プロファイルを削除します。
書式	SDCERR DeleteConfig (CHAR* name);
引数	name [入力]プロファイル名が格納されるバッファへのポインタ。
戻り値	0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。
備考	アクティブな構成プロファイルは削除できません。 name に NULL を指定することはできません。

SetDefaultConfigValues

目的	新しい構成プロファイルのデフォルト値を設定します。
書式	SDCERR SetDefaultConfigValues (SDCConfig *cfg);
引数	cfg [入力]構成が格納されている SDCConfig 構造体へのポインタ。
戻り値	0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

CreateConfig

目的	構成プロファイルをデフォルト値で作成します。
書式	SDCERR CreateConfig (SDCConfig *cfg);
引数	cfg [入力]構成が格納されている SDCConfig 構造体へのポインタ。
戻り値	0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。
備考	構成ファイル作成作成後、追加処理を行ってください。 構成メモリを割り当てる必要があります。

GetConfig

目的 構成プロファイルを取得します。

書式 SDCERR GetConfig (char *name,
SDCConfig *config);

引数 name
[入力]プロファイル名が格納されているバッファへのポインタ。
config
[出力]構成が格納される SDCConfig 構造体へのポインタ。
➢ NULL にはできません。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

ModifyConfig

目的 構成プロファイルを変更します。

書式 SDCERR ModifyConfig (char *name,
SDCConfig *config);

引数 name
[入力]プロファイル名が格納されているバッファへのポインタ。更新するプロファイル名。
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.10 サミット構成設定

GetAllConfigs

目的 すべての構成プロファイルを取得します。

書式 SDCERR GetAllConfigs (SDCConfig *config,
DWORD *num);

引数 config
[出力]構成が格納される SDCConfig 構造体へのポインタ。
➢ この配列の要素数の合計は、ヘッダーファイルで定義され MAX_CFGS よりも大きくしてください。
num
[出力]要素数が格納されているバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetAllConfigs

目的 すべての構成プロファイルを上書きします。

書式 SDCERR SetAllConfigs (DWORD num,
SDCConfig *config);

引数 num
[入力]32BIT、符号なし変数。配列の要素数を指定します。
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.11 グローバル構成設定

GetGlobalSettings

目的 グローバル構成設定を取得します。

書式 SDCERR GetGlobalSettings (SDCGlobalConfig *configGlobal);

引数 configGlobal
[出力]構成が格納される SDCGlobalConfig 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetGlobalSettings

目的 グローバル構成設定を上書きします。

書式 SDCERR SetGlobalSettings (SDCGlobalConfig *configGlobal);

引数 configGlobal
[入力]構成が格納されている SDCGlobalConfig 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.12 サミットレジストリキー

FlushConfigKeys

目的 ストレージにコンフィギュレーションのレジストリキーを書込みます。

書式 SDCERR FlushConfigKeys (INT configNumber);

引数 configNumber
[入力]INT 変数。

-1	グローバル設定書込み。
0	サードパーティコンフィグレーション書込み(予備)。
1～MAX_CFGS	構成プロファイルを識別子で書込み。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

備考 この関数はシステムがレジストリに書き込む前に電源サイクルが発生した場合、サミットパラメータを保存するために、強制的に書込みます。

FlushAllConfigKeys

目的 サミット構成に関するすべてのレジストリキーを書込みます。

書式 SDCERR FlushAllConfigKeys (VOID);

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

備考 この関数はシステムがレジストリに書き込む前に電源サイクルが発生した場合、サミットパラメータを保存するために、強制的に書込みます。

4.8.13 構成ファイル

GetConfigFileInfo

目的 サミット構成ファイルの情報を取得します。

書式 SDCERR GetConfigFileInfo (CHAR *fileName,
CONFIG_FILE_INFO *info);

引数 filename
[入力]ファイル名が格納されているバッファへのポインタ。
info
[出力]情報が格納される CONFIG_FILE_INFO 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

ExportSettings

目的 指定したファイルにコンフィグレーションとグローバル設定を出力します。

書式
```
SDCERR ExportSettings (CHAR *fileName,
                       SDC_ALL *configAll);
```

引数 filename
[入力]ファイル名が格納されているバッファへのポインタ。

引数 configAll
[入力]情報が格納されている SDC_ALL 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

備考 詳細は SDC_ALL 構造体を参照してください。
- configGlobal – グローバル設定。設定の出力を行わない場合は NULL を指定します。
- configThirdParty – サードパーティ設定。設定の出力を行わない場合は NULL を指定します。
- configs – 設定。1、または複数の SDCConfig 構造体を指定します。設定の出力を行わない場合は NULL を指定します。
- numConfigs – 出力するコンフィグレーション(SDCConfig)の数を指定します。0 は SDCConfig 出力なしです。この数に configGlobal や configThirdParty を含まないでください。

ImportSettings

目的 指定したファイルからコンフィグレーションとグローバル設定を取得します。

書式
```
SDCERR ImportSettings (CHAR *fileName,
                       SDC_ALL *configAll);
```

引数 filename
[入力]ファイル名が格納されているバッファへのポインタ。

引数 configAll
[出力]情報が格納される SDC_ALL 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

備考 詳細は SDC_ALL 構造体を参照してください。
- configGlobal – グローバル設定。設定の取得を行わない場合は NULL を指定します。
- configThirdParty – サードパーティ設定。設定の取得を行わない場合は NULL を指定します。
- configs – 設定。1、または複数の SDCConfig 構造体を指定します。設定の取得を行わない場合は NULL を指定します。
- numConfigs – 取得するコンフィグレーション(SDCConfig)の数を指定します。この数に configGlobal や configThirdParty を含まないでください。

SDC_ALL 構造体- configGlobal、configThirdParty、最大 MAX_CFGS の configs-のメモリを割り当ててください。

4.8.14 WEP キー

GetWEPKey

目的　WEP キーを取得します。

書式
```
SDCERR GetWEPKey (SDCConfig *config,
                  INT keyIndex,
                  WEPLEN *keyLength,
                  UNSIGNED CHAR *wepKey,
                  BOOLEAN *keyState);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
keyIndex
[入力]INT 変数。

1～4	取得する WEP キーのインデックス

keyLength
[出力]キーの長さ。NULL の場合このパラメータは無視されます。

0	WEPLEN_NOT_SET	(キークリア)
1	WEPLEN_40BIT	(10 文字の 16 進文字列)
2	WEPLEN_128BIT	(26 文字の 16 進文字列)

wepKey
[出力]WEP キーが格納されるバッファへのポインタ。
➢ 26 バイト以上のバッファサイズが必要です。
keyState
[入力]BOOLEAN 変数。

0	キーはアクティブではない。
1	キーはアクティブ。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetWEPKey

目的　WEP キーを設定します。

書式
```
SDCERR SetWEPKey (SDCConfig *config,
                  INT keyIndex,
                  WEPLEN keyLength,
                  UNSIGNED CHAR *wepKey,
                  BOOLEAN *keyState);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
keyIndex
[入力]INT 変数。

1～4	設定する WEP キーのインデックス

keyLength
[入力]キーの長さ。

0	WEPLEN_NOT_SET	(キークリア)
1	WEPLEN_40BIT	(10 文字の 16 進文字列)
2	WEPLEN_128BIT	(26 文字の 16 進文字列)

wepKey
[入力]WEP キーが格納されているバッファへのポインタ。
➢ 26 バイト以上のバッファサイズが必要です。

keyState
[入力]BOOLEAN 変数。

0	キーをアクティブにしません。
1	キーをアクティブにします。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

GetMultipleWEPKeys

目的　4 つすべての WEP キーを取得します。

書式
```
SDCERR GetMultipleWEPKeys (SDCConfig *config,
                           INT *keyIndex,
                           WEPLEN *keyLength1,
                           UNSIGNED CHAR *wepKey1,
                           WEPLEN *keyLength2,
                           UNSIGNED CHAR *wepKey2,
                           WEPLEN *keyLength3,
                           UNSIGNED CHAR *wepKey3,
                           WEPLEN *keyLength4,
                           UNSIGNED CHAR *wepKey4);
```

引数　config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
keyIndex
[入力]INT 変数。

1～4	アクティブな WEP キーのインデックス

keyLength(1～4)
[出力]キーの長さ。

0	WEPLEN_NOT_SET	(キークリア)
1	WEPLEN_40BIT	(10 文字の 16 進文字列)
2	WEPLEN_128BIT	(26 文字の 16 進文字列)

wepKey(1～4)
[出力]WEP キーが格納されるバッファへのポインタ。
➢ 26 バイト以上のバッファサイズが必要です。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetMultipleWEPKeys

目的　4 つすべての WEP キーを取得します。

書式
```
SDCERR SetMultipleWEPKeys (SDCConfig *config,
                           INT keyIndex,
                           WEPLEN keyLength1,
                           UNSIGNED CHAR *wepKey1,
                           WEPLEN keyLength2,
                           UNSIGNED CHAR *wepKey2,
                           WEPLEN keyLength3,
                           UNSIGNED CHAR *wepKey3,
                           WEPLEN keyLength4,
                           UNSIGNED CHAR *wepKey4);
```

引数　config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
keyIndex
[入力]INT 変数。

1～4	

keyLength(1～4)
[入力]キーの長さ。

0	WEPLEN_NOT_SET	(キークリア)
1	WEPLEN_40BIT	(10 文字の 16 進文字列)
2	WEPLEN_128BIT	(26 文字の 16 進文字列)

wepKey(1～4)
[入力]WEP キーが格納されているバッファへのポインタ。
➢ 26 バイト以上のバッファサイズが必要です。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.15 事前共通キー(PSK)

GetPSK

目的 事前共通キーを取得します。

書式
```
SDCERR GetPSK(SDCConfig *config,
              CHAR *psk);
```

引数
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
psk
[出力]事前共通キーが格納されるバッファへのポインタ。
➢ PSK_SZ(バイト)以上のバッファサイズが必要です。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetPSK

目的 事前共通キーを設定します。

書式
```
SDCERR SetPSK(SDCConfig *config,
              CHAR *psk);
```

引数
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
psk
[入力]事前共通キーが格納されているバッファへのポインタ。
➢ NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。文字列の長さは、PSK_SZ(バイト)以上でなければなりません。
➢ PSK の場合、64 文字の 16 進文字でなければなりません。
➢ パスフレーズの場合、8～63 文字の印刷可能な文字列でなければなりません。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.16 LEAP 認証

GetLEAPCred

目的 LEAP 認証を取得します。

書式
```
SDCERR GetLEAPCred (SDCConfig *config,
                    CHAR *username,
                    CHAR *password);
```

引数
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。

password
[出力]パスワードが格納されるバッファへのポインタ。
- USER_PWD_SZ(バイト)以上のバッファサイズが必要です。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetLEAPCred

目的 LEAP 認証を設定します。

書式
```
SDCERR SetLEAPCred (SDCConfig *config,
                    CHAR *username,
                    CHAR *password);
```

引数
config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[入力]事前共通キーが格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。文字列の長さは、PSK_SZ(バイト)以上でなければなりません。

password
[入力]パスワードが格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。文字列の長さは、PSK_SZ(バイト)以上でなければなりません。

戻り値 0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.17 EAP-FAST 認証

GetEAPFASTCred

目的　EAP-FAST 認証を取得します。

書式

```
SDCERR GetEAPFASTCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CHAR *pacFileName,
                       CHAR *pacPassword);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。

password
[出力]パスワードが格納されるバッファへのポインタ。
- USER_PWD_SZ(バイト)以上のバッファサイズが必要です。

pacFileName
[出力]PAC ファイル名が格納されるバッファへのポインタ。
- CRED_PFILE_SZ(バイト)以上のバッファサイズが必要です。

pacPassword
[出力]PAC パスワードが格納されるバッファへのポインタ。
- CRED_PFILE_SZ(バイト)以上のバッファサイズが必要です。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetEAPFASTCred

目的　EAP-FAST 設定を設定します。

書式

```
SDCERR SetEAPFASTCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CHAR *pacFileName,
                       CHAR *pacPassword);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。
  USER_NAME_SZ(バイト)以上の文字列を指定してください。

password
[出力]パスワードが格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。
  USER_PWD_SZ(バイト)以上の文字列を指定してください。

pacFileName
[出力]PAC ファイル名が格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。
  CRED_PFILE_SZ(バイト)以上の文字列を指定してください。

pacPassword
[出力]PAC パスワードが格納されるバッファへのポインタ。
- NULL で終わる文字列を指定します。引数に NULL を指定するとクリアされます。
  CRED_PFILE_SZ(バイト)以上の文字列を指定してください。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.18 PEAP-GTC 認証

GetPEAPGTCCred

目的　PEAP-GTC 認証を取得します。

書式

```
SDCERR GetPEAPGTCCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CERTLOCATION *certLocation,
                       CHAR *caCert);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

password
[出力]パスワードが格納されるバッファへのポインタ。
- USER_PWD_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

certLocation
[出力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert
[出力]CA 証明書が格納されるバッファへのポインタ。
- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetPEAPGTCCred

目的　PEAP-GTC 認証を設定します。

書式

```
SDCERR SetPEAPGTCCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CERTLOCATION certLocation,
                       CHAR *caCert);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[入力]ユーザー名が格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。
  USER_NAME_SZ(バイト)以上の文字列を指定してください。

password
[入力]パスワードが格納されているバッファへのポインタ。
- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。
  USER_PWD_SZ(バイト)以上の文字列を指定してください。

certLocation
[入力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert
[入力] CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.19 PEAP-MSCHAP 認証

GetPEAPMSCHAPCred

目的　PEAP-MSCHAP 認証を取得します。

書式

```
SDCERR GetPEAPMSCHAPCred (SDCConfig *config,
                          CHAR *username,
                          CHAR *password,
                          CERTLOCATION *certLocation,
                          CHAR *caCert);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

password
[出力]パスワードが格納されるバッファへのポインタ。
- USER_PWD_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

certLocation
[出力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert
[出力]CA 証明書が格納されるバッファへのポインタ。
- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetPEAPMSCHAPCred

目的　PEAP-MSCHAP 認証を設定します。

書式

```
SDCERR SetPEAPMSCHAPCred (SDCConfig *config,
                          CHAR *username,
                          CHAR *password,
                          CERTLOCATION certLocation,
                          CHAR *caCert);
```

引数

config

[入力]構成が格納されている SDCConfig 構造体へのポインタ。

username

[入力]ユーザー名が格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。 USER_NAME_SZ(バイト)以上の文字列を指定してください。

password

[入力]パスワードが格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。 USER_PWD_SZ(バイト)以上の文字列を指定してください。

certLocation

[入力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert

[入力] CA 証明書が格納されているバッファへのポインタ。

- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.20 EAP-TLS 認証

GetEAPTLSCred

目的　EAP-TLS 認証を取得します。

書式

```
SDCERR GetEAPTLSCred (SDCConfig *config,
                      CHAR *username,
                      CHAR *userCert,
                      CERTLOCATION *certLocation,
                      CHAR *caCert);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

userCert
[出力]ユーザー証明書が格納されるバッファへのポインタ。
- 20 バイト以上のバッファサイズが必要です。“userCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。NULL の場合、このパラメータは無視されます。

certLocation
[出力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert
[出力]CA 証明書が格納されるバッファへのポインタ。
- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetEAPTLSCred

目的　EAP-TLS 認証を設定します。

書式

```
SDCERR SetEAPTLSCred (SDCConfig *config,
                      CHAR *username,
                      CHAR *userCert,
                      CERTLOCATION certLocation,
                      CHAR *caCert);
```

引数

config

[入力]構成が格納されている SDCConfig 構造体へのポインタ。

username

[入力]ユーザー名が格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。
  USER_NAME_SZ(バイト)以上の文字列を指定してください。

userCert

[入力] ユーザー証明書が格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。
  “userCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

certLocation

[入力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert

[入力] CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.8.21 EAP-TTLS 認証

GetEAPTTLSCred

目的　EAP-TTLS 認証を取得します。

書式

```
SDCERR GetEAPTTLSCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CERTLOCATION *certLocation,
                       CHAR *caCert);
```

引数

config
[入力]構成が格納されている SDCConfig 構造体へのポインタ。
username
[出力]ユーザー名が格納されるバッファへのポインタ。
- USER_NAME_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

password
[出力]パスワードが格納されるバッファへのポインタ。
- USER_PWD_SZ(バイト)以上のバッファサイズが必要です。NULL の場合、このパラメータは無視されます。

certLocation
[出力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert
[出力]CA 証明書が格納されるバッファへのポインタ。
- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

SetEAPTTLSCred

目的　EAP-TTLS 認証を設定します。

書式

```
SDCERR SetEAPTTLSCred (SDCConfig *config,
                       CHAR *username,
                       CHAR *password,
                       CERTLOCATION certLocation,
                       CHAR *caCert);
```

引数

config

[入力]構成が格納されている SDCConfig 構造体へのポインタ。

username

[入力]ユーザー名が格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。USER_NAME_SZ(バイト)以上の文字列を指定してください。

password

[入力]パスワードが格納されているバッファへのポインタ。

- NULL で終わる文字列を指定します。NULL の場合、このパラメータは無視されます。USER_PWD_SZ(バイト)以上の文字列を指定してください。

certLocation

[入力] 認証サーバーを有効するための CA 証明書を使用するかどうかを指定する値。
NULL の場合、このパラメータは無視されます。

0	CERT_NONE	“caCert”に NULL が設定されています。サーバー認証しないでください。
1	CERT_FILE	“caCert”はルート認証機関のデジタル証書のファイル名です。CRED_CERT_SZ(バイト)以上の長さが必要です。
2	CERT_FULL_STORE	"caCert"は NULL に設定されています。Microsoft 証明書ストアからの有効な証明書を検索します。
3	CERT_IN_STORE	“caCert”は Microsoft の証明書ストアから 1 つの特定の証明書を表す 20 バイトのハッシュ値です。

caCert

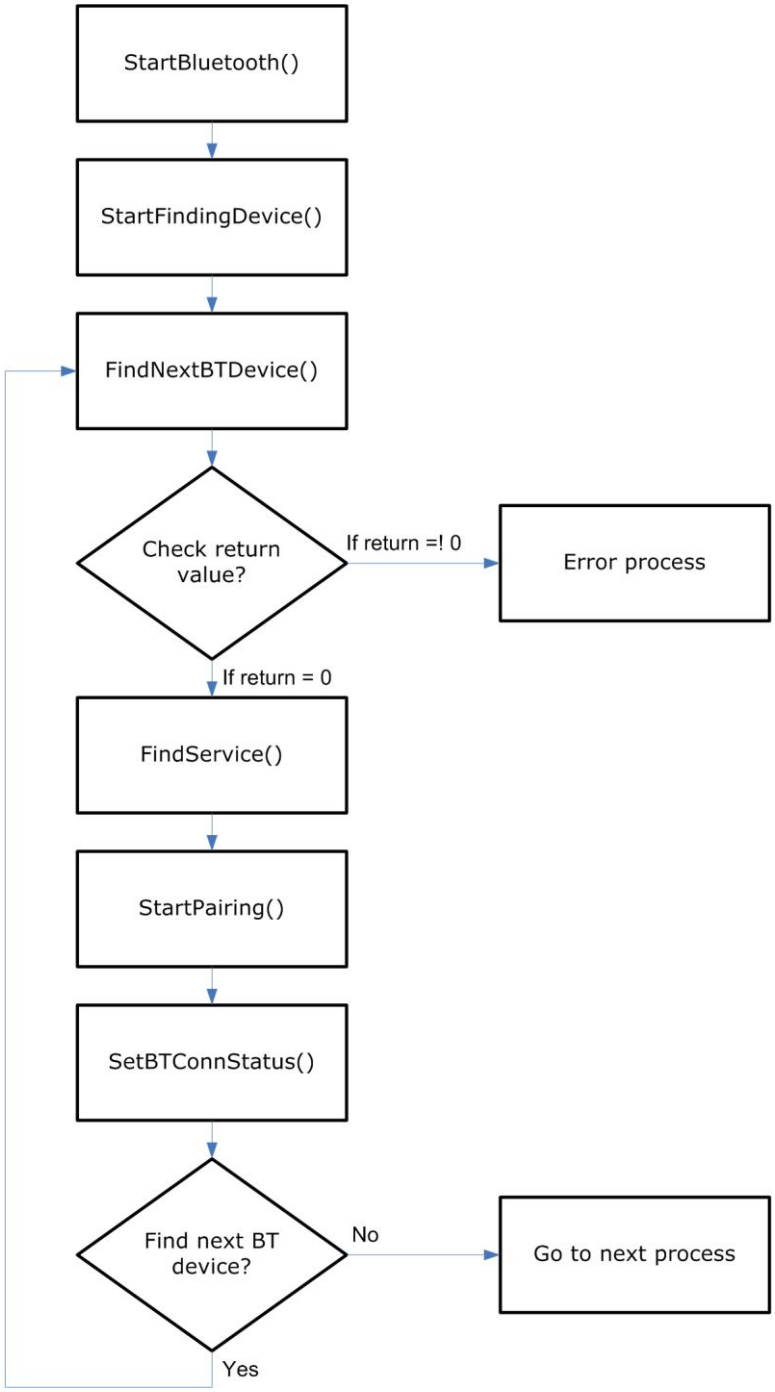
[入力] CA 証明書が格納されているバッファへのポインタ。

- CRED_CERT_SZ(バイト)以上のバッファサイズが必要です。CA 証明書は上記” certLocation”の値に依存します。NULL の場合、このパラメータは無視されます。

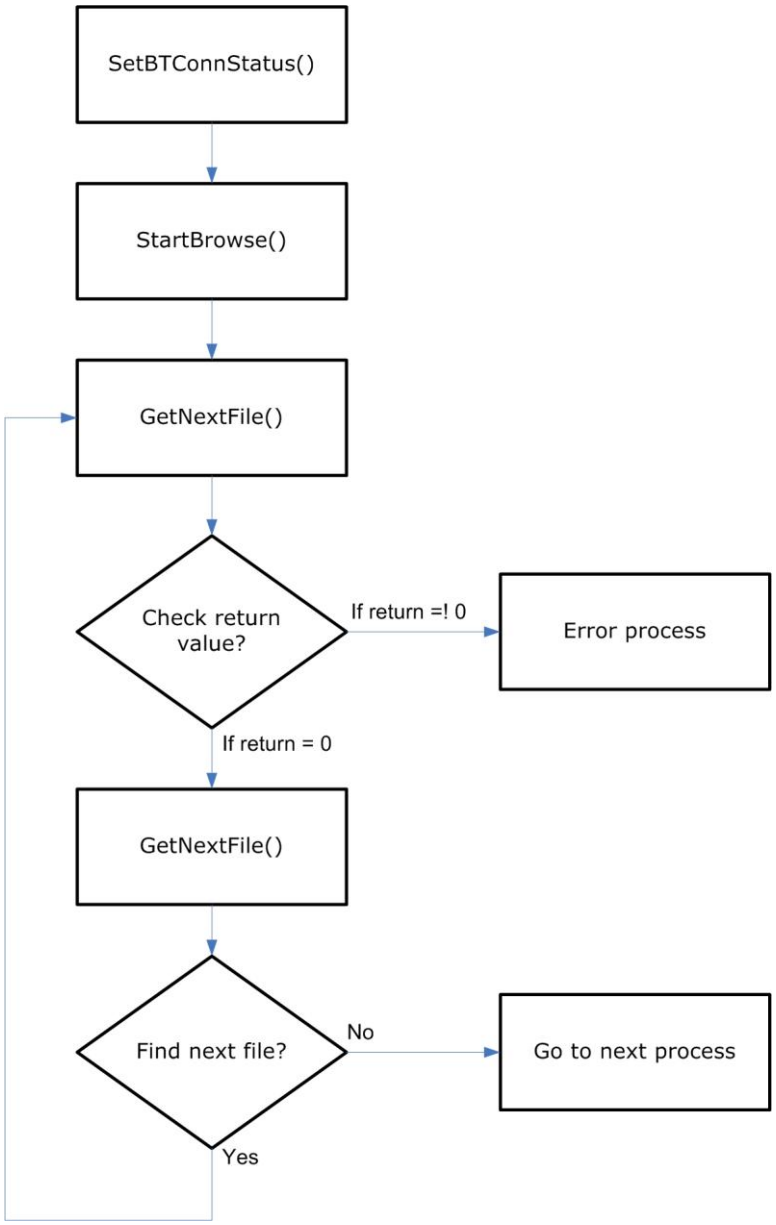
戻り値　0=成功。それ以外=失敗。『エラーコード(SDCERR)』を参照してください。

4.9 Bluetooth

Bluetooth 初期化のプログラミングフローは次の通りです。



初期化後、SetBTConnStatus()をコールすることにより、Bluetooth サービスを開始することができます。下記は、ファイル転送サービス(FTP)を使用するためのプログラミングフローです。


SetBTConnStatus()
StartBrowse()
GetNextFile()
Check return value?
If return =! 0
Error process
If return = 0
GetNextFile()
Find next file?
No
Go to next process
Yes

4.9.1 Bluetooth 開始/終了

StartBluetooth

目的 Bluetooth サービスのアクセスを有効にします。

書式 DWORD StartBluetooth (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

32	ERROR_OPERATION_FAIL
33	ERROR_MACHINE_NOT_SUPPORTED

StopBluetooth

目的 Bluetooth サービスのアクセスを無効にします。

書式 DWORD StopBluetooth (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

32	ERROR_OPERATION_FAIL

4.9.2 デバイス検索

StartFindingDevice

目的 検索プロシージャを初期化します。

書式 DWORD StartFindingDevice (VOID);

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

32	ERROR_OPERATION_FAIL

FindNextBTDevice

目的 StartFindingDevice()コール後に、Bluetooth デバイスを検索します。

書式 DWORD FindNextBTDevice (INT *deviceInfo);

引数 deviceInfo
[出力]デバイス情報が格納される変数へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

30	ERROR_NO_STARTFINDING
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

4.9.3 デバイスのペアリング

StartPairing

目的　特定の Bluetooth デバイスとペアリングします。

書式　DWORD StartPairing (INT deviceInfo,
CHAR *pinCode,
INT pinCodeLen);

引数　deviceInfo
[入力] FindNextBTDevice()の戻り値。
pinCode
[入力] PIN コードが格納されているバッファへのポインタ。
pinCodeLen
[入力] PIN コードの長さ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER
21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

参照　FindNextBTDevice

StartUnpairing

目的　特定の Bluetooth デバイスとペアリングを解除します。

書式　DWORD StartUnpairing (INT deviceInfo);

引数　deviceInfo
[入力] FindNextBTDevice()の戻り値。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

4.9.4 利用可能なサービスとデバイスの情報

FindService

目的　接続されたデバイス上で利用可能な Bluetooth サービスを探します。

書式　DWORD FindService (INT targetService,
INT deviceInfo,
INT *serviceInfo);

引数　targetService
[入力] 問い合わせる Bluetooth サービス。

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM0(クライアント)
3	BT_SPP_SERVICE	COM0(クライアント)
4	BT_OPP_SERVICE	
5	(予備)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

deviceInfo
[入力] FindNextBTDevice()の戻り値。
serviceInfo
[出力] 情報が格納される変数へのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
32	ERROR_OPERATION_FAIL

参照　GetServiceType

GetDeviceAddress

目的　接続されたデバイス上で利用可能な Bluetooth サービスを探します。

書式　DWORD GetDeviceAddress (INT deviceInfo,
ULONGLONG *btAddr);

引数　deviceInfo
[入力] FindNextBTDevice()の戻り値。
btAddr
[出力] MAC アドレスが格納されるバッファへのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
31	ERROR_WRONG_ARG

参照　FindNextBTDevice

GetDeviceName

目的　接続されたデバイス上で利用可能な Bluetooth サービスを探します。

書式　DWORD GetDeviceName (INT deviceInfo,
CHAR *buf,
INT bufSize);

引数　deviceInfo
[入力] FindNextBTDevice()の戻り値。
buf
[出力] デバイス名が格納されるバッファへのポインタ。
bufSize
[入力] 文字バッファのサイズ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
31	ERROR_WRONG_ARG

参照　FindNextBTDevice

GetServiceType

目的　接続されたデバイス上で利用可能な Bluetooth サービスを探します。

書式　DWORD GetServiceType (INT serviceInfo,
INT *serviceType);

引数　serviceInfo
[入力] FindService()の戻り値。
serviceType
[出力] 情報が格納されるバッファへのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
31	ERROR_WRONG_ARG

参照　FindService

4.9.5 Bluetooth 接続とステータス

ConnectByMacAddr

目的　MAC アドレスとサービスタイプによる Bluetooth 接続を確立します。

書式

```
DWORD ConnectByMacAddr (ULONGLONG deviceMacAddr,
                        INT targetService,
                        CHAR *pinCode,
                        INT sppComPort,
                        INT *deviceInfo,
                        INT *serviceInfo);
```

引数

deviceMacAddr
[入力] 対象デバイスの MAC アドレス。
targetService
[入力] 接続する Bluetooth サービス。

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM0(クライアント)
3	BT_SPP_SERVICE	COM0(クライアント)
4	(予備)	
5	(予備)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

pinCode
[入力] PIN コードが格納されているバッファへのポインタ。
sppComPort
[入力] SPP 接続するためのシリアルポート。
deviceInfo
[出力] 情報が格納される変数へのポインタ。
serviceInfo
[出力] 情報が格納される変数へのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER
21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
27	ERROR_WRONG_SERVICE_TYPE
32	ERROR_OPERATION_FAIL

GetBTConnStatus

目的　現在の Bluetooth 接続状態を取得します。

書式　DWORD GetBTConnStatus (INT serviceInfo,
INT *connStatus);

引数　serviceInfo
[入力] FindService()の戻り値。
connStatus
[出力] 情報が格納される変数へのポインタ。

0	対象 Bluetooth デバイスと接続中ではない。
1	対象 Bluetooth デバイスと接続中。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER
22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ

参照　FindService

SetBTConnStatus

目的　Bluetooth 接続を設定します。

書式　DWORD SetBTConnStatus (INT serviceType,
INT onOff,
INT *deviceInfo,
INT *serviceInfo);

引数　serviceType
[入力] Bluetooth サービス。

1	BT_HID_SERVICE	
2	BT_DUN_SERVICE	COM0(クライアント)
3	BT_SPP_SERVICE	COM0(クライアント)
4	(予備)	
5	(予備)	
6	BT_HEADSET_SERVICE	
7	BT_HANDSFREE_SERVICE	
8	BT_FTP_SERVICE	

connStatus
[入力] 指定された Bluetooth サービスでの接続確立の有無。

0	対象 Bluetooth デバイスと切断。
1	対象 Bluetooth デバイスと接続。

deviceInfo
[入力] FindNextBTDevice()の戻り値。
serviceInfo
[入力] FindService()の戻り値。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

21	ERROR_NO_DEVICE_INFO
22	ERROR_NO_SERVICE_INFO
25	ERROR_NOT_DEVICEINFO_OBJ
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
28	ERROR_DEVICE_NOT_HAS_SERVICE
32	ERROR_OPERATION_FAIL

参照　FindNextBTDevice, FindService

4.9.6 FTP サービス – ファイル表示

StartBrowse

目的 ブラウズプロシージャを初期化します。

書式
```
DWORD StartBrowse (INT serviceInfo,
                   CHAR *targetPath);
```

引数
serviceInfo
[入力] FindService()の戻り値。
targetPath
[入力] ディレクトリパスが格納されているバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
32	ERROR_OPERATION_FAIL

参照 FindService

GetNextFile

目的 StartBrowse()コール後のファイルを表示します。

書式
```
DWORD GetNextFile (INT serviceInfo,
                   FTP_FILE_INFO *fileInfo);
```

引数
serviceInfo
[入力] FindService()の戻り値。
targetPath
[出力] 情報が格納される FTP_FILE_INFO 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

参照 FindService

4.9.7 FTP サービス – ファイル転送

GetFile

目的 FTP 共有フォルダからファイルをダウンロードします。

書式
```
DWORD GetFile (INT serviceInfo,
               FTP_FILE_INFO *fileInfo,
               HWND progressBar);
```

引数
serviceInfo
[入力] FindService()の戻り値。
fileInfo
[出力] 情報が格納される FTP_FILE_INFO 構造体へのポインタ。
progressBar
[入力] ファイルアップロード進捗状況を表示するプログレスバーのウィンドウハンドル。
引数に NULL を渡すこともできます。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

参照 FindService, GetNextFile, StartBrowse

PutFile

目的 FTP 共有フォルダへファイルをアップロードします。

書式
```
DWORD PutFile (INT serviceInfo,
               CHAR *filePath,
               HWND progressBar);
```

引数
serviceInfo
[入力] FindService()の戻り値。
filePath
[入力] ファイルパスが格納されているバッファへのポインタ。
progressBar
[入力] ファイルアップロード進捗状況を表示するプログレスバーのウィンドウハンドル。
引数に NULL を渡すこともできます。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE
29	ERROR_SERVICE_NOT_CONNECTED
31	ERROR_WRONG_ARG
32	ERROR_OPERATION_FAIL

参照 FindService, GetNextFile, StartBrowse

4.9.8 SPP COM ポート

GetSppComPort

目的 SPP 接続に使用するシリアルポートを検索します。

書式 DWORD GetSppComPort (INT serviceInfo,
INT *comPort);

引数 serviceInfo
[入力] FindService()の戻り値。
comPort
[出力] 情報が格納されるバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

4	ERROR_PARAMETER
22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE

SetSppComPort

目的 SPP 接続に使用するシリアルポートを設定します。

書式 DWORD GetSppComPort (INT serviceInfo,
INT comPort);

引数 serviceInfo
[入力] FindService()の戻り値。
comPort
[入力] 常に COM0 を使用します(クライアント)。

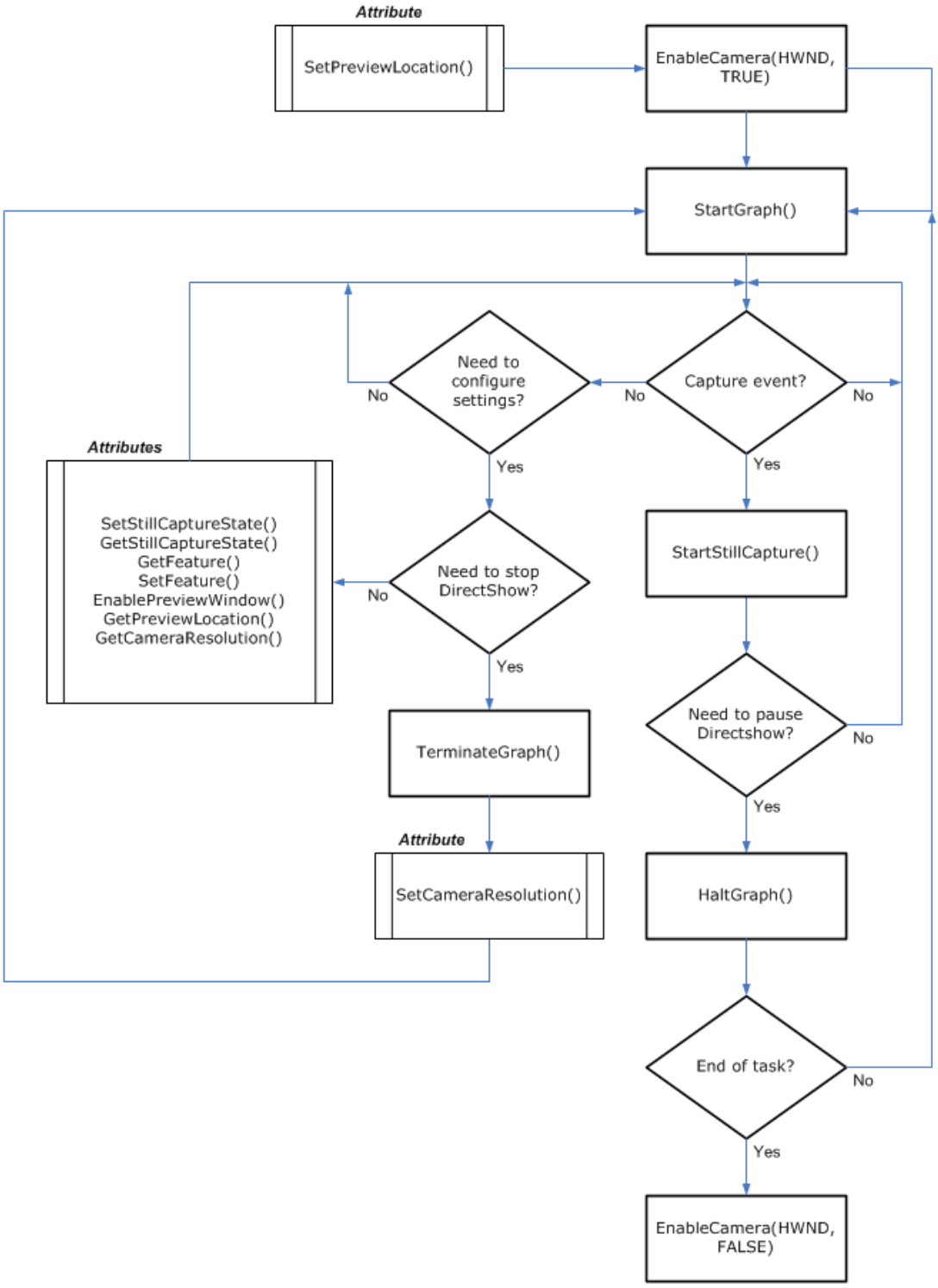
戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

22	ERROR_NO_SERVICE_INFO
26	ERROR_NOT_SERVICEINFO_OBJ
27	ERROR_WRONG_SERVICE_TYPE

備考 デフォルトでは、COM0 が SPP 接続に使用するシリアルポートです。この関数は、SetBTConnStatus()を使用する前にコールする必要があります。

4.10 カメラ

カメラ操作のプログラミングフローは次の通りです。



4.10.1 カメラ電源

EnableCamera

目的　カメラ電源を ON/OFF します。

書式　DWORD EnableCamera (HWND wnd,
BOOL onOff);

引数　wnd
[入力] HWND 変数。
onoff
[入力] BOOL 変数。

0	無効 (カメラ OFF)
1	有効 (カメラ ON)

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

参照　EnablePreviewWindow, SetPreviewLocation

4.10.2 カメラ設定

GetCameraResolution

目的　カメラの解像度を取得します。

書式　DWORD GetCameraResolution (DWORD *resValue);

引数　resValue
[出力] 情報が格納されるバッファへのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetCameraResolution

目的　カメラの解像度を設定します。

書式　DWORD SetCameraResolution (DWORD resValue);

引数　resValue
[入力] 32Bit 符号なし INT 変数。

0	640 × 480 ピクセル
1	1280 × 960 ピクセル
2	1600 × 1200 ピクセル
3	2048 × 1536 ピクセル

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

備考　この関数の前に TerminateGraph()コールする必要があります。新しい解像度は StartGraph()が再びコールされるまで有効になりません。

参照　StartGraph, TerminateGraph

GetFlashLight

目的　カメラのフラッシュ設定を取得します。

書式　DWORD GetFlashLight (FLASH *flashLight);

引数　flashLight
[出力] フラッシュ設定が格納される FLASH 構造体へのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetFlashLight

目的　撮影時にカメラのフラッシュが光るかどうか設定します。

書式　DWORD SetFlashLight (FLASH flashLight);

引数　flashLight
[出力] フラッシュ設定が格納される FLASH 構造体へのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

参照　EnableCamera, StartStillCapture

4.10.3 プレビュー

EnablePreviewWindow

目的 カメラの解像度を設定します。

書式 DWORD EnablePreviewWindow (BOOL onoff);

引数 onoff
[入力] BOOL 変数。

0	無効 (=プレビューなし)
1	有効 (=プレビュー表示)

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)

参照 SetPreviewLocation

GetPreviewLocation

目的 プレビューウィンドウの位置を取得します。

書式 DWORD GetPreviewLocation (INT *posX,
INT *posY);

引数 posX (将来)
[出力] INT 変数。
posY
[出力] INT 変数。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetPreviewLocation

目的 プレビューウィンドウの位置を設定します。

書式 DWORD SetPreviewLocation (INT posX,
INT posY);

引数 posX (将来)
[出力] INT 変数。
posY
[出力] INT 変数。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

参照 プレビューウィンドウのサイズは 240×180 ピクセルです。

参照 EnablePreviewWindow

GetFeature

目的 特殊効果設定を取得します。

書式 DWORD GetFeature (DWORD *level);

引数 level
[出力] 32Bit 符号なし INT 変数。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetFeature

目的 プレビュー画像に特殊効果を適用します。

書式 DWORD SetFeature (DWORD level);

引数 level
[入力] 32Bit 符号なし INT 変数。

0	モノクロ。
1(※)	通常。
2	(白黒)反転。
3	セピア。
4	ポスタリゼーション。
5	ホワイトボード。
6	ブラックボード。
7	アクア。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

参照 SetPreviewLocation

StartGraph

目的 DirectShow のフィルタを実行し、プレビューウィンドウでグラフを開始します。

書式 DWORD StartGraph ();

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)

参照 HaltGraph, TerminateGraph

TerminateGraph

目的 DirectShow フィルタの実行を停止し、プレビューウィンドウをフラッシュします。

書式 DWORD TerminateGraph ();

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)

参照 HaltGraph, StartGraph

HaltGraph

目的 StartGraph()が再度コールされるまで、プレビューウィンドウのグラフを一時停止します。

書式 DWORD TerminateGraph ();

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)

参照 StartGraph, TerminateGraph

4.10.4 静止画キャプチャー

GetStillCaptureState

目的 キャプチャー設定を取得します。

書式 DWORD GetStillCaptureState (PICSTATE *picInfo);

引数 picInfo
[出力] キャプチャー設定が格納される PICSTATE 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetStillCaptureState

目的 キャプチャー設定を設定します。

書式 DWORD SetStillCaptureState (PICSTATE *picInfo);

引数 picInfo
[入力] キャプチャー設定が格納されている PICSTATE 構造体へのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

備考 この関数は StartStillCapture()より前にコールする必要があります。

参照 StartStillCapture

StartStillCapture

目的 静止画キャプチャーを開始します。

書式 DWORD StartStillCapture (INT playSound);

引数 playSound
[入力] INT 変数。

0	通知なし。
1	画像がキャプチャーされたことを通知する音を再生。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

備考 キャプチャーが完了すると、Windows メッセージ”WM_CAMERACAPTUREFINISH”を送信します。

参照 SetPreviewLocation

4.10.5 トーチモード

GetTorchMode

目的 現在のトーチモードを取得します。

書式 DWORD GetTorchMode (DWORD *dwMode);

引数 dwMode
[出力] 情報が格納されるバッファへのポインタ。

0	無効。
1	有効。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

SetTorchMode

目的 トーチライトを有効または無効に設定します。

書式 DWORD SetTorchMode (DWORD dwMode);

引数 dwMode
[出力] 32Bit 符号なし INT 変数。

0	無効。
1	有効。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	ERROR_NORESOURCE (リソース取得失敗)
4	ERROR_PARAMETER (パラメータ誤り)

4.11 GPS

シリアル通信プロトコルは、米国海洋電子機器協会の NMEA0183 ASCII のインターフェイス仕様に基づいています。詳細は、NMEA(www.nmea.org)の NMEA 0183 バージョン 3.01 を参照してください。

以下の表は NMEA 0183 データ伝送の標準特性です。

シリアル設定	NMEA 標準
ボーレート	4800
データビット	8
パリティ	なし
ストップビット	1

COM ポートが開いている限り、モバイルコンピュータ上の GPS 受信機は NMEA メッセージ出力に COM7 使用します。シリアルポートを開いて NMEA 形式のデータを受信するのに、Microsoft の標準 API を使用できます。以下の URL に、シリアルポートのプログラミング技術に関連したドキュメントがあります。

http://msdn.microsoft.com/en-us/library/bb202722.aspx
http://msdn.microsoft.com/en-us/library/ms810467.aspx
http://msdn.microsoft.com/en-us/library/bb201943.aspx

4.12 ボタンの割り当て

GetButtonAssignment

目的　ユーザー定義可能なキーの割り当てを取得します。

書式　DWORD GetButtonAssignment (INT buttonID,
INT *keyCode,
TCHAR *buffer);

引数　buttonID
[入力] INT 変数。

1	Side-Trigger-Left	(デフォルト: Scan)
2	Side-Trigger-Right	(デフォルト: Scan)
3	(将来)	(将来)
4	(将来)	(将来)
5	(将来)	(将来)
6	(将来)	(将来)
7	Send	(デフォルト: Send)
8	End	(デフォルト: End)
9	(将来)	(将来)
10	(将来)	(将来)
11	Up	(デフォルト: Up)
12	Down	(デフォルト: Down)
13	Left	(デフォルト: Left)
14	Right	(デフォルト: Right)
15	ESC	(デフォルト: Esc)
16	TAB	(デフォルト: Tab)
17	Backspace	(デフォルト: Backspace)
18	Volume Up	(デフォルト: Volume Up)
19	Volume Down	(デフォルト: Volume Down)
20	Application (programming key)	(デフォルト:VK_OEM_KEY_1)
21	*Shift	(デフォルト: Shift)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
22	(将来)	(将来)
23	Enter	(デフォルト: Enter)
24	*Alpha	(デフォルト: Alpha)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
25	*Fn	(デフォルト: Fn)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
26	(将来)	(将来)
27	(将来)	(将来)
28	-	(デフォルト: -)
29	.	(デフォルト: .)
107	(将来)	(将来)
108	(将来)	(将来)
109	(将来)	(将来)
110	(将来)	(将来)
111	(将来)	(将来)
112	(将来)	(将来)
113	(将来)	(将来)
114	(将来)	(将来)
115	(将来)	(将来)
116	(将来)	(将来)

117	(将来)	(将来)

keyCode

[出力] キー割り当て情報が格納されるバッファへのポインタ。

buffer

[出力] ユーザー定義のキーコードやプログラムの完全なファイルパスが格納されるバッファへのポインタ。

戻り値

0=成功。それ以外=失敗。失敗するとエラーを返します。

1	レジストリオープン失敗
2	レジストリ書込み失敗
4	ERROR_PARAMETER (パラメータ誤り)

SetButtonAssignment

目的　別のキーとして機能するか、特定のプログラムを起動するためのショートカットキーとして機能するユーザ定義可能なキーを割り当てます。

書式　DWORD SetButtonAssignment (INT buttonID,
INT keyCode,
TCHAR *buffer);

引数　buttonID
[入力] INT 変数。

1	Side-Trigger-Left	(デフォルト: Scan)
2	Side-Trigger-Right	(デフォルト: Scan)
3	(将来)	(将来)
4	(将来)	(将来)
5	(将来)	(将来)
6	(将来)	(将来)
7	Send	(デフォルト: Send)
8	End	(デフォルト: End)
9	(将来)	(将来)
10	(将来)	(将来)
11	Up	(デフォルト: Up)
12	Down	(デフォルト: Down)
13	Left	(デフォルト: Left)
14	Right	(デフォルト: Right)
15	ESC	(デフォルト: Esc)
16	TAB	(デフォルト: Tab)
17	Backspace	(デフォルト: Backspace)
18	Volume Up	(デフォルト: Volume Up)
19	Volume Down	(デフォルト: Volume Down)
20	Application (programming key)	(デフォルト:VK_OEM_KEY_1)
21	*Shift	(デフォルト: Shift)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
22	(将来)	(将来)
23	Enter	(デフォルト: Enter)
24	*Alpha	(デフォルト: Alpha)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
25	*Fn	(デフォルト: Fn)
	➢ このキーに割り当てることはできません。利用可能のロック/アンロックのみです。システムリセットでロック解除されます。	
26	(将来)	(将来)
27	(将来)	(将来)
28	-	(デフォルト: -)
29	.	(デフォルト: .)
107	(将来)	(将来)
108	(将来)	(将来)
109	(将来)	(将来)
110	(将来)	(将来)
111	(将来)	(将来)
112	(将来)	(将来)
113	(将来)	(将来)
114	(将来)	(将来)
115	(将来)	(将来)
116	(将来)	(将来)
117	(将来)	(将来)

keyCode

[入力] INT 変数。

0	ユーザー定義キーコード(0x01 ～ 0xFF)
1	特定のプログラム起動
2	Enter
3	Scan
4	Esc
5	Delete
6	Backspace
7	Space
8	Tab
9 10 ～ 20	F1 F2 ～ F12
21	Start Menu
22	Alt
23	OEM_Key1 (0xE9)
24	OEM_Key2 (0xEA)
25	OEM_Key3 (0xEB)
26	OEM_Key4 (0xEC)
27	OEM_Key5 (0xED)
28	OEM_Key6 (0xEE)
29	OEM_Key7 (0xEF)
30	OEM_Key8 (0xF0)
31	OEM_Key9 (0xF1)
32	OEM_Key10 (0x2A)
33	*
34	#
35	Send(VK_TTALK)
36	End(VK_END)
37	(将来)
38	Up
39	Down
40	Left
41	Right
42	TAB
43	Volume Up
44	Volume Down
45	(将来)
46	(将来)
47	OK
48	(将来)
49	(将来)
50	Home(VK_HOME)
51	End(VK_END)
52	Fn+ESC(0xF5)
53 54 ～ 64	F13 F14 ～ F24
65	Home(VK_LWIN+ VK_APP2)
66	(将来)
67	-
68	.
997	Lock key ➢ロックキー:ロックされたキーはどのモードでも機能がありません。(SHIFT + キー、ブルー + キー、オレンジ + キー)
998	Unlock key

buffer
[入力] ユーザー定義のキーコードやプログラムの完全なファイルパスが格納されているバッファへのポインタ。このパラメータを使用しない場合は NULL を渡します。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	レジストリオープン失敗
2	レジストリ書込み失敗
4	ERROR_PARAMETER (パラメータ誤り)

4.13 署名取り込み

4.13.1 初期化 / 終了

InitSigScreen

目的　署名領域を初期化し表示します。

書式
```
INT InitSigScreen (INT x,
                   INT y,
                   INT width,
                   INT height,
                   HWND parentWnd);
```

引数
x
[入力] 署名取り込みを行う領域の開始位置の X 座標。
y
[入力] 署名取り込みを行う領域の開始位置の Y 座標。
width
[入力] 署名領域の幅。
height
[入力] 署名領域の高さ。
parentWnd
[入力] 署名取り込み機能が実装されている親またはオーナーウィンドウのハンドル。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	SIGERROR_UNKNOWN
2	SIGERROR_MACHINE_NOT_SUPPORTED

備考　この関数は、他の署名関連の関数を使用する前に呼び出す必要があります。署名領域は、初期化完了時に有効になります。

CloseSigScreen

目的　署名領域を終了します。

書式
```
VOID CloseSigScreen (VOID);
```

備考　署名関連の機能を再び有効にするには、InitSigScreen をコールする必要があります。

4.13.2 署名領域の制御

ClearSigArea

目的 署名をクリアします。

書式 VOID ClearSigArea (VOID);

EnableSigScreen

目的 署名領域を有効または無効に設定します。

書式 VOID EnableSigScreen (INT enabled);

引数 enabled

[入力] INT 変数。

0	無効。
1	有効。

ShowSigScreen

目的 署名領域を表示または非表示に設定します。

書式 VOID ShowSigScreen (INT showArea);

引数 showArea

[入力] INT 変数。

0	非表示。
1	表示。

4.13.3 署名領域の制御

ShowSigScreen

目的 署名領域を表示または非表示に設定します。

書式 VOID ShowSigScreen (INT redOfRGB,
INT greenOfRGB,
INT blueOfRGB);

引数 redOfRGB

[入力] INT 変数。

0 ～ 255	デフォルトは 255。

greenOfRGB

[入力] INT 変数。

0 ～ 255	デフォルトは 255。

blueOfRGB

[入力] INT 変数。

0 ～ 255	デフォルトは 255。

4.13.4 ペンの色と太さ

SetSigLineColor

目的 ペンの色を設定します。

書式 VOID SetSigLineColor (INT redOfRGB,
INT greenOfRGB,
INT blueOfRGB);

引数 redOfRGB
[入力] INT 変数。

0 ～ 255	デフォルトは 0。

greenOfRGB
[入力] INT 変数。

0 ～ 255	デフォルトは 0。

blueOfRGB
[入力] INT 変数。

0 ～ 255	デフォルトは 0。

SetSigLineWidth

目的 ペンの色を設定します。

書式 VOID SetSigLineWidth (INT width);

引数 width
[入力] INT 変数。

1 ～ 5	デフォルトは 1。

4.13.5 署名イメージの保存と呼び出し

LoadSigImage

目的 署名イメージをロードします。

書式 INT LoadSigImage (WCHAR *filePath);

引数 filePath
[入力] イメージのフルパスが格納されているバッファへのポインタ。

戻り値 0=成功。それ以外=失敗。失敗するとエラーを返します。

1	SIGERROR_UNKNOWN
3	SIGERROR_SIG_WINDOW_NOT_CREATED
4	SIGERROR_FILE_NOT_FOUND
6	SIGERROR_HIDDEN_WINDOW

備考 署名領域初期化中にこの関数をコールしないでください。また、解像度が 640 × 480 ピクセル以上のイメージをロードできません。

SaveSigImage

目的　署名イメージを保存します。

書式　INT SaveSigImage (INT imageFormat,
WCHAR *filePath);

引数　imageFormat
[入力] INT 変数。

0	BMP_FORMAT	BMP 形式で保存。
1	JPG_FORMAT	JPEG 形式で保存。
2	LOCUS_FORMAT	LOCUS 形式で保存。

filePath
[入力] イメージのフルパスが格納されるバッファへのポインタ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

1	SIGERROR_UNKNOWN
3	SIGERROR_SIG_WINDOW_NOT_CREATED
6	SIGERROR_HIDDEN_WINDOW

4.13.6 署名 DLL バージョン

GetSigDllVer

目的　署名イメージをロードします。

書式　INT GetSigDllVer (CHAR *buf,
INT bufSize);

引数　buf
[出力] バージョン情報が格納されるバッファへのポインタ。
bufSize
[入力] 文字バッファのサイズ。

戻り値　0=成功。それ以外=失敗。失敗するとエラーを返します。

5	SIGERROR_ILLEGAL_PARAMETER

5　データ構造体

5.1 システム情報構造体

5.1.1 SYSINFO

この構造体にはモバイルコンピュータのスタート画面で、[設定]→[システム]→[デバイス情報]で表示される設定と同じシステム情報が格納されます。

```
typedef struct _SYSINFO {
TCHAR           DevDesc[16];
TCHAR           DevCFG[16];
TCHAR           SerialNum[16];
TCHAR           SerialNumReadOnly[16];
TCHAR           SerialNumPCBA[16];
TCHAR           Manufactory[32];
TCHAR           ManuDate[16];
UCHAR           OsVer[32];
UCHAR           BootloaderVer[32];
TCHAR           MicroPVer[32];
UCHAR           BTAddress[32];
UCHAR           WiFiAddress[32];
CHAR            IMEI[32];
} SYSINFO, *PSYSINFO;
```

データ型	メンバ名	説明
TCHAR	DevDesc[16]	デバイスモデル。
TCHAR	DevCFG[16]	(将来)
TCHAR	SerialNum[16]	シリアル番号。
TCHAR	SerialNumReadOnly[16]	工場用シリアル番号。(注 1) (注 2)
TCHAR	SerialNumPCBA[16]	PCBA シリアル番号。(注 2)
TCHAR	Manufactory[32]	メーカー。
TCHAR	ManuDate[16]	製造年月日。
UCHAR	OsVer[32]	OS バージョン。
UCHAR	BootloaderVer[32]	ブートローダのバージョン。
TCHAR	MicroPVer[32]	MicroP ファームウェアバージョン。
UCHAR	BTAddress[32]	Bluetooth MAC アドレス。
UCHAR	WiFiAddress[32]	Wi-Fi MAC アドレス。
TCHAR	IMEI[32]	ユニークな GSM 番号

[注 1]：初期値では、 SerialNum と SerialNumReadOnly は同じ値です。
[注 2]：SerialNumReadOnly と SerialNumPCBA の情報は GetSysInfo()をコールすることでのみアクセスすることができます。

5.2 バックライト構造体

5.2.1 BKLCTL

この構造体にはモバイルコンピュータのスタート画面で、[設定]→[システム]→[バックライト]で表示される設定と同じバックライト情報が格納されます。

バッテリーモード- 使用中のバッテリー電源。

AC モード - クレードル経由で AC 電源に接続。

```
typedef struct _BKLCTL {
DWORD           batttimeout;
DWORD           lightlevel;
DWORD           actimeout;
DWORD           aclightlevel;
DWORD           backlightontap;
DWORD           acbacklightontap;
DWORD           usebattery;
DWORD           useext;
} BKLCTL, *PBKLCTL;
```

データ型	メンバ名	説明
DWORD	batttimeout	バッテリーモードで、指定時間後にバックライトを OFF。 – 15, 30, 60, 120, 300 (秒) ➢ "usebattery"を有効に設定する必要があります。
DWORD	lightlevel	バッテリーモードでのバックライトレベル。 0 ～ 11 : 明暗。
DWORD	actimeout	AC モードで、指定時間後にバックライトを OFF。 – 15, 30, 60, 120, 300 (秒) ➢ "useext"を有効に設定する必要があります。
DWORD	aclightlevel	AC モードでのバックライトレベル。 0 ～ 11 : 明暗。
DWORD	backlightontap	バッテリーモードで、バックライトが OFF になっている場合、タッチスクリーンをタップするかキーを押すことにより自動的に ON になります。 0 : バッテリーモード時、タップで ON にしない。 1 : バッテリーモード時、タップでも ON にする。
DWORD	acbacklightontap	AC モードで、バックライトが OFF になっている場合、タッチスクリーンをタップするかキーを押すことにより自動的に ON になります。 0 : AC モード時、タップで ON にしない。 1 : AC モード時、タップでも ON にする。
DWORD	usebattery	バッテリー電源使用時の、バックライトの省電力モード。 0 : バッテリーモード時の省電力無効。 1 : バッテリーモード時の省電力有効。 ➢ 有効の場合、” batttimeout”と” backlightontap”を設定してください。
DWORD	useext	AC 電源使用時の、バックライトの省電力モード。 0 : AC モード時の省電力無効。 1 : AC モード時の省電力有効。 ➢ 有効の場合、”actimeout”と” acbacklightontap”を設定してください。

5.3 キーパッド構造体

5.3.1 KEYPADBIND

この構造体には、システム初期化時の起動するアプリケーションに関する情報が格納されます。

```
typedef struct {
TCHAR                   szClass[64];
} KEYPADBIND, *PKEYPADBIND;
```

データ型	メンバ名	説明
TCHAR	szClass[64]	起動するアプリケーション名。

5.4 Wi-Fi 構造体

5.4.1 RAW_DATA

この構造体は、WLAN ネットワークに関する情報を格納します。

```
typedef struct {
DWORD           dwDataLen;
BYTE*           pData;
} RAW_DATA, *PRAW_DATA;
```

データ型	メンバ名	説明
DWORD	dwDataLen	pData の長さ。
BYTE*	pData	wzcsapi.h に定義されている PWZC_802_11_CONFIG_LIST 構造体を参照してください。

5.4.2 WLANADPTINFO

構造体 WLANADPTINFO は、スタート画面で、[設定]→[接続]→[Wi-Fi]で表示される Wi-Fi ネットワークに関する情報を格納します。

```
typedef struct _WLANADPTINFO {
DWORD           fUseDHCP;
DWORD           IPAddr;
DWORD           SubnetMask;
DWORD           Gateway;
DWORD           DNSAddr;
DWORD           DNSAltAddr;
DWORD           WINSAddr;
DWORD           WINSAltAddr;
} WLANADPTINFO, *PWLANADPTINFO
```

データ型	メンバ名	説明
DWORD	fUseDHCP	DHCP サーバー使用有無。 0 DHCP 無効。 1 DHCP 有効。
DWORD	IPAddr	モバイルコンピュータの IP アドレス。
DWORD	SubnetMask	サブネットマスクの IP アドレス。
DWORD	Gateway	デフォルトゲートウェイの IP アドレス。
DWORD	DNSAddr	プライマリ DNS の DNS サーバーアドレス。
DWORD	DNSAltAddr	セカンダリ DNS の DNS サーバーアドレス。
DWORD	WINSAddr	プライマリ WINS の WINS サーバーアドレス。
DWORD	WINSAltAddr	セカンダリ WINS の WINS サーバーアドレス。

[注]：DHCP 有効の場合、他の IP 関連情報は 0 がセットされます。

5.4.3 WLANCONNECTEDST

この構造体には、ワイヤレスマネージャ設定ページに表示される情報と同じ無線接続状態に関する情報が格納されます。

```
typedef struct _WLANCONNECTEDST {
CHAR                    SSID[32];
DWORD                   SSIDLength;
BYTE                    BSSID[6];
} WLANCONNECTEDST, *PWLANCONNECTEDST;
```

データ型	メンバ名	説明
CHAR	SSID[32]	アクセスポイントのサービスセット識別子(SSID)、最大 32 文字。
DWORD	SSIDLength	SSID の長さ。(スペースを含む)
BYTE	BSSID[6]	接続されたアクセスポイントの MAC アドレス。

5.4.4 WLANCTL

この構造体には、スタート画面で、[設定]→[接続]→[Wi-Fi]→[ワイヤレス]→[新規]で表示されるものと同じ新しい無線ネットワークに関する情報が格納されます。

```
typedef struct _WLANCTL {
CHAR            SSID[32];
DWORD           authentication;
DWORD           encryption;
DWORD           adhoc;
DWORD           eap;
WCHAR           key[80];
} WLANCTL, *PWLANCTL;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>CHAR</td><td>SSID[32]</td><td>最大 32 文字。</td></tr>
<tr><td>DWORD</td><td>authentication</td><td>使用中の認証。
<table>
<tr><td>0</td><td>オープン (Ndis802_11AuthModeOptn)</td></tr>
<tr><td>1</td><td>共有キー(Ndis802_11AuthModeShared)</td></tr>
<tr><td>3</td><td>WPA (Ndis802_11AuthModeWPA)</td></tr>
<tr><td>4</td><td>WPA-PSK (Ndis802_11AuthModeWPAPSK)</td></tr>
<tr><td>5</td><td>WPA-NONE (Ndis802_11AuthModeWPANone)</td></tr>
<tr><td>6</td><td>WPA2 (Ndis802_11AuthModeWPA2)</td></tr>
<tr><td>7</td><td>WPA2=PSK (Ndis802_11AuthModeWPA2PSK)</td></tr>
</table></td></tr>
<tr><td>DWORD</td><td>encryption</td><td>使用中の暗号化。
<table>
<tr><td>0</td><td>WEP (Ndis802_11WEPEnabled)</td></tr>
<tr><td>1</td><td>なし (Ndis802_11WEPDisabled)</td></tr>
<tr><td>4</td><td>TKIP (Ndis802_11Encryption2Enabled)</td></tr>
<tr><td>6</td><td>AES (Ndis802_11Encryption3Enabled)</td></tr>
</table></td></tr>
<tr><td>DWORD</td><td>adhoc</td><td>ネットワーク種別。
<table>
<tr><td>0</td><td>インフラ</td></tr>
<tr><td>1</td><td>アドホックネットワーク</td></tr>
</table></td></tr>
<tr><td>DWORD</td><td>eap</td><td>ネットワーク種別。
<table>
<tr><td>13</td><td>TLS</td></tr>
<tr><td>25</td><td>PEAP</td></tr>
</table></td></tr>
<tr><td>WCHAR</td><td>key[80]</td><td>以下のいずれかの形式の WEP キー。
<table>
<tr><td>1/0x1234567890</td><td>[index=1,40Bit(HEX10 桁)]</td></tr>
<tr><td>4/zxcvb</td><td>[index=4,40Bit(5 文字)]</td></tr>
<tr><td>3/0x12345678901
234567890123456</td><td>[index=3,104Bit(HEX26 桁)]</td></tr>
<tr><td>2/abcdefghij123</td><td>[index=2,104Bit(13 文字)]</td></tr>
<tr><td>auto</td><td>[EAP から取得]</td></tr>
</table></td></tr>
</table>

5.4.5 CF10G_STATUS

```
typedef struct _CF10G_STATUS {
CARDSTATE       cardState;
CHAR            configName[CONFIG_NAME_SZ];
UCHAR           client_MAC[6];
UCHAR           client_IP[4];
CHAR            clientName[CLIENT_NAME_SZ];
UCHAR           AP_MAC[6];
UCHAR           AP_IP[4];
CHAR            APName[CLIENT_NAME_SZ];
EAPTYPE         eapType;
DWORD           channel;
INT             rssi;
BITRATE         bitRate;
INT             txPower;
DWORD           driverVersion;
RADIOTYPE       radioType;
DWORD           DTIM;
DWORD           beaconPeriod;
DWORD           beaconsReceived;
} CF10G_STATUS;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>CARDSTATE</td><td>cardState</td><td>結合ステータス
<table><tr><td>0</td><td>CARDSTATE_NOT_INSERTED</td></tr><tr><td>1</td><td>CARDSTATE_NOT_ASSOCIATED</td></tr><tr><td>2</td><td>CARDSTATE_ASSOCIATED</td></tr><tr><td>3</td><td>CARDSTATE_AUTHENTICATED</td></tr><tr><td>4</td><td>CARDSTATE_FCCTEST</td></tr><tr><td>5</td><td>CARDSTATE_NOT_SDC</td></tr></table></td></tr>
<tr><td>CHAR</td><td>configName
[CONFIG_NAME_SZ]</td><td>アクティブなプロファイルの名前。
33 文字まで。ヘッダファイルに定義されている
CONFIG_NAME_SZ を参照してください。</td></tr>
<tr><td>UCHAR</td><td>client_MAC[6]</td><td>サミットラジオの MAC アドレス。</td></tr>
<tr><td>UCHAR</td><td>client_IP[4]</td><td>サミットラジオの IP アドレス。</td></tr>
<tr><td>CHAR</td><td>clientName
[CLIENT_NAME_SZ]</td><td>サミットラジオ名。
17 文字まで。ヘッダファイルに定義されている
CLIENT_NAME_SZ を参照してください。</td></tr>
<tr><td>UCHAR</td><td>AP_MAC[6]</td><td>アクセスポイントの MAC アドレス。</td></tr>
<tr><td>UCHAR</td><td>AP_IP[4]</td><td>アクセスポイントの IP アドレス。(AP にサポートされてない場合は NULL)</td></tr>
<tr><td>CHAR</td><td>APName
[CLIENT_NAME_SZ]</td><td>アクセスポイント名。(AP にサポートされてない場合は NULL)
17 文字まで。ヘッダファイルに定義されている
CLIENT_NAME_SZ を参照してください。</td></tr>
<tr><td>EAPTYPE</td><td>eapType</td><td>使用中の EAP 種別。
<table><tr><td>0</td><td>EAP_NONE</td></tr><tr><td>1</td><td>EAP_LEAP</td></tr><tr><td>2</td><td>EAP_EAPFAST</td></tr><tr><td>3</td><td>PEAPMSCHAP</td></tr><tr><td>4</td><td>PEAPGTC</td></tr><tr><td>5</td><td>EAPTLS</td></tr></table></td></tr>
<tr><td>DWORD</td><td>channel</td><td>使用中のチャンネル。サミット無線と AP との間の WLAN の接続に関する情報。</td></tr>
<tr><td>INT</td><td>rssi</td><td>信号の強さ。サミット無線と AP との間の WLAN の接続に関する情報。</td></tr>
</table>

<table>
<tr><td>BITRATE</td><td>bitRate</td><td>データ転送速度。サミット無線と AP との間の WLAN の接続に関する情報。<table><tr><td>0</td><td>BITRATE_AUTO</td></tr><tr><td>2</td><td>BITRATE_1</td></tr><tr><td>4</td><td>BITRATE_2</td></tr><tr><td>11</td><td>BITRATE_5_5</td></tr><tr><td>12</td><td>BITRATE_6</td></tr><tr><td>18</td><td>BITRATE_9</td></tr><tr><td>22</td><td>BITRATE_11</td></tr><tr><td>24</td><td>BITRATE_12</td></tr><tr><td>36</td><td>BITRATE_18</td></tr><tr><td>48</td><td>BITRATE_24</td></tr><tr><td>72</td><td>BITRATE_36</td></tr><tr><td>96</td><td>BITRATE_48</td></tr><tr><td>108</td><td>BITRATE_54</td></tr></table></td></tr>
<tr><td>INT</td><td>txPower</td><td>送信電力。サミット無線と AP との間の WLAN の接続に関する情報。<table><tr><td>0</td><td>TXPOWER_MAX</td></tr><tr><td>1</td><td>TXPOWER_1</td></tr><tr><td>5</td><td>TXPOWER_5</td></tr><tr><td>10</td><td>TXPOWER_10</td></tr><tr><td>20</td><td>TXPOWER_20</td></tr><tr><td>30</td><td>TXPOWER_30</td></tr><tr><td>50</td><td>TXPOWER_50</td></tr></table></td></tr>
<tr><td>DWORD</td><td>driverVersion</td><td>ドライバのバージョン。</td></tr>
<tr><td>RADIOTYPE</td><td>radioType</td><td>使用中の無線種別。<table><tr><td>0</td><td>RADIOTYPE_BG</td></tr><tr><td>1</td><td>RADIOTYPE_ABG</td></tr><tr><td>100</td><td>RADIOTYPE_NOT_SDC</td></tr><tr><td>101</td><td>RADIOTYPE_NOT_SDC_1</td></tr></table></td></tr>
<tr><td>DWORD</td><td>DTIM</td><td>待機中の省電力モードの無線クライアントに対して、どのくらいの頻度で DTIM を含んだビーコンを送信するかを指定します。(例：DTIM 間隔 3 は 3 回に 1 回 DTIM を含んだビーコンを送信します。)</td></tr>
<tr><td>DWORD</td><td>beaconPeriod</td><td>アクセスポイントのビーコン間隔をキロマイクロ秒で指定します。(1Kμsec = 1024 ミリ秒)</td></tr>
<tr><td>DWORD</td><td>beaconsReceived</td><td>ビーコン受信。</td></tr>
</table>

5.4.6 CONFIG_FILE_INFO

```
typedef struct _CONFIG_FILE_INFO {
DWORD               numConfigs;
BOOLEAN             globalConfigPresent;
BOOLEAN             thirdPartyConfigPresent;
DWORD               sdkVersion;
} CONFIG_FILE_INFO;
```

データ型	メンバ名	説明
DWORD	numConfigs	プロファイルの数。最大 20 まで。ヘッダファイルに定義されている MAX_CFGS を参照してください。
BOOLEAN	globalConfigPresent	グローバル設定があるかどうか。 0 なし 1 あり
BOOLEAN	thirdPartyConfigPresent	(将来機能) サードパーティの設定があるかどうか。 0 なし 1 あり
DWORD	sdkVersion	サミット無線の SDK バージョン。

5.4.7 SDC_ALL

```
typedef struct _SDC_ALL {
DWORD               numConfigs;
SDCConfig           *configs;
SDC3rdPartyConfig   *configThirdParty;
SDCGlobalConfig     *configGlobal;
} SDC_ALL;
```

データ型	メンバ名	説明
DWORD	numConfigs	プロファイルの数。最大 20 まで。ヘッダファイルに定義されている MAX_CFGS を参照してください。
SDCConfig	*configs	SDCConfig を参照。
SDC3rdPartyConfig	*configThirdParty	(将来機能) SDC3rdPartyConfig を参照。
SDCGlobalConfig	*configGlobal	SDCGlobalConfig を参照。

5.4.8 SDCCONFIG

```
typedef struct _SDCConfig {
CHAR            configName[CONFIG_NAME_SZ];
CHAR            SSID[SSID_SZ];
CHAR            clientName[CLIENT_NAME_SZ];
INT             txPower;
AUTH            authType;
EAPTYPE         eapType;
POWERSAVE       powerSave;
WEPTYPE         wepType;
BITRATE         bitRate;
RADIOMODE       radioMode;
CRYPT           userName;
CRYPT           userPwd;
CRYPT           PSK;
CRYPT           WEPKeys;
} SDCConfig;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>CHAR</td><td>configName
[CONFIG_NAME_SZ]</td><td>プロファイル名。33 文字以下。ヘッダファイルに定義されている CONFIG_NAME_SZ を参照してください。</td></tr>
<tr><td>CHAR</td><td>SSID[SSID_SZ]</td><td>サミット無線が接続しているサービスセット識別子(SSID)。33 文字以下。ヘッダファイルに定義されている SSID_SZ を参照してください。</td></tr>
<tr><td>CHAR</td><td>clientName
[CLIENT_NAME_SZ]</td><td>サミット無線名。17 文字以下。ヘッダファイルに定義されている CLIENT_NAME_SZ を参照してください。</td></tr>
<tr><td>INT</td><td>txPower</td><td>使用中の TX 電源
<table><tr><td>0</td><td>現在の既定ドメインで定義されている最大値。</td></tr><tr><td>1</td><td>1mW</td></tr><tr><td>5</td><td>5mW</td></tr><tr><td>10</td><td>10mW</td></tr><tr><td>20</td><td>20mW</td></tr><tr><td>30</td><td>30mW</td></tr><tr><td>50</td><td>50mW</td></tr></table></td></tr>
<tr><td>AUTH</td><td>authType</td><td>使用中の認証。
<table><tr><td>0</td><td>オープン</td></tr><tr><td>1</td><td>共有キー</td></tr><tr><td>2</td><td>LEAP (Network-EAP)</td></tr></table></td></tr>
<tr><td>EAPTYPE</td><td>eapType</td><td>使用中の EAP 種別。
<table><tr><td>0</td><td>EAP_NONE</td></tr><tr><td>1</td><td>EAP_LEAP</td></tr><tr><td>2</td><td>EAP_EAPFAST</td></tr><tr><td>3</td><td>PEAPMSCHAP</td></tr><tr><td>4</td><td>PEAPGTC</td></tr><tr><td>5</td><td>EAPTLS</td></tr></table></td></tr>
<tr><td>POWERSAVE</td><td>powerSave</td><td>省電力設定
➢ 常時起動モード(CAM)はクライアントアダプタで常時電源が入っているため、メッセージの応答時間にほとんど遅れがありません。最も電力を消費しますが、最高のスループットを提供します。AC 電源で使用する場合にお勧めです。
➢ 最大省電力(Max PSP)は、周期的に起動し任意のバッファリングされたメッセージが待っているかどうかアクセスポイントを調査するクライアントアダプタへの着信メッセージをバッファリングするようにします。クライアントアダプタはそれらのメッセージを要求したのち、スリープモードに戻ることができます。バッテリで使用する場合にお勧めです。
➢ パワーセーブモード(Fast PSP)は、ネットワークトラフィックに応じて前述の 2 つもモードが切り替わります。大量のパ</td></tr>
</table>

<table>
<tr><td></td><td></td><td>ケットを受信すると CAM モードに切り替わり、パケットを取り終えると PSP に切り替わります。Max PSP より消費電力が懸念されますが、高いスループットを要求する場合はお勧めします。<table><tr><td>0</td><td>常時起動モード(CAM)</td></tr><tr><td>1</td><td>最大省電力</td></tr><tr><td>2</td><td>省電力</td></tr></table></td></tr>
<tr><td>WEPTYPE</td><td>wepType</td><td>使用中の WEP 種別。<table><tr><td>0</td><td>WEP_OFF</td></tr><tr><td>1</td><td>WEP_ON</td></tr><tr><td>2</td><td>WEP_AUTO</td></tr><tr><td>3</td><td>WEP_PSK</td></tr><tr><td>4</td><td>WEP_TKIP</td></tr><tr><td>5</td><td>WEP2_PSK</td></tr><tr><td>6</td><td>WEP2_AES</td></tr><tr><td>7</td><td>CCKM_TKIP</td></tr><tr><td>8</td><td>WEP_CKIP</td></tr><tr><td>9</td><td>WEP_AUTO_CKIP</td></tr><tr><td>10</td><td>CCKM_AES</td></tr></table></td></tr>
<tr><td>BITRATE</td><td>bitRate</td><td>AP と通信するサミット無線で使用されるデータ転送速度。<table><tr><td>0</td><td>BITRATE_AUTO</td></tr><tr><td>2</td><td>BITRATE_1</td></tr><tr><td>4</td><td>BITRATE_2</td></tr><tr><td>11</td><td>BITRATE_5_5</td></tr><tr><td>12</td><td>BITRATE_6</td></tr><tr><td>18</td><td>BITRATE_9</td></tr><tr><td>22</td><td>BITRATE_11</td></tr><tr><td>24</td><td>BITRATE_12</td></tr><tr><td>36</td><td>BITRATE_18</td></tr><tr><td>48</td><td>BITRATE_24</td></tr><tr><td>72</td><td>BITRATE_36</td></tr></table></td></tr>
<tr><td>RADIOMODE</td><td>radioMode</td><td>使用中の無線モード。<table><tr><td>0</td><td>RADIOMODE_B_ONLY</td></tr><tr><td>1</td><td>RADIOMODE_BG</td></tr><tr><td>2</td><td>RADIOMODE_G_ONLY</td></tr><tr><td>3</td><td>RADIOMODE_BG_LRS</td></tr><tr><td>4</td><td>RADIOMODE_A_ONLY</td></tr><tr><td>5</td><td>RADIOMODE_ABG</td></tr><tr><td>6</td><td>RADIOMODE_BGA</td></tr><tr><td>7</td><td>RADIOMODE_ADHOC</td></tr></table></td></tr>
<tr><td>CRYPT</td><td>userName</td><td>認証用ユーザー名(EAP)。72 文字以下。ヘッダファイルに定義されている CRED_CA_POS を参照してください。</td></tr>
<tr><td>CRYPT</td><td>userPwd</td><td>認証用パスワード(EAP)。72 文字以下。ヘッダファイルに定義されている CRED_UCA_POS を参照してください。</td></tr>
<tr><td>CRYPT</td><td>PSK</td><td>暗号化のための事前共通キー。</td></tr>
<tr><td>CRYPT</td><td>WEPKeys</td><td>暗号化のための WEP キー。</td></tr>
</table>

5.4.9 SDC3RDPARTYCONFIG

```
typedef struct _SDC3rdPartyConfig
{
CHAR            clientName[CLIENT_NAME_SZ];
POWERSAVE       powerSave;
INT             txPower;
BITRATE         bitRate;
RADIOMODE       radioMode;
} SDC3rdPartyConfig;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>CHAR</td><td>clientName
[CLIENT_NAME_SZ]</td><td>サミットラジオ名。
17 文字まで。ヘッダファイルに定義されている
CLIENT_NAME_SZ を参照してください。</td></tr>
<tr><td>POWERSAVE</td><td>powerSave</td><td>省電力設定
➢ 常時起動モード(CAM)はクライアントアダプタで常時電源が入っているため、メッセージの応答時間にほとんど遅れがありません。最も電力を消費しますが、最高のスループットを提供します。AC 電源で使用する場合にお勧めです。
➢ 最大省電力(Max PSP)は、周期的に起動し任意のバッファリングされたメッセージが待っているかどうかアクセスポイントを調査するクライアントアダプタへの着信メッセージをバッファリングするようにします。クライアントアダプタはそれらのメッセージを要求したのち、スリープモードに戻ることができます。バッテリで使用する場合にお勧めです。
➢ パワーセーブモード(Fast PSP)は、ネットワークトラフィックに応じて前述の 2 つもモードが切り替わります。大量のパケットを受信すると CAM モードに切り替わり、パケットを取り終えると PSP に切り替わります。Max PSP より消費電力が懸念されますが、高いスループットを要求する場合はお勧めします。
<table>
<tr><td>0</td><td>常時起動モード(CAM)</td></tr>
<tr><td>1</td><td>最大省電力</td></tr>
<tr><td>2</td><td>省電力</td></tr>
</table>
</td></tr>
<tr><td>INT</td><td>txPower</td><td>使用中の送信電力。
<table>
<tr><td>0</td><td>最大電力。</td></tr>
<tr><td>1</td><td>1mW</td></tr>
<tr><td>5</td><td>5mW</td></tr>
<tr><td>10</td><td>10mW</td></tr>
<tr><td>20</td><td>20mW</td></tr>
<tr><td>30</td><td>30mW</td></tr>
<tr><td>50</td><td>50mW</td></tr>
</table>
</td></tr>
<tr><td>BITRATE</td><td>bitRate</td><td>AP と通信するサミット無線で使用されるデータ転送速度。
<table>
<tr><td>0</td><td>BITRATE_AUTO</td></tr>
<tr><td>2</td><td>BITRATE_1</td></tr>
<tr><td>4</td><td>BITRATE_2</td></tr>
<tr><td>11</td><td>BITRATE_5_5</td></tr>
<tr><td>12</td><td>BITRATE_6</td></tr>
<tr><td>18</td><td>BITRATE_9</td></tr>
<tr><td>22</td><td>BITRATE_11</td></tr>
<tr><td>24</td><td>BITRATE_12</td></tr>
<tr><td>36</td><td>BITRATE_18</td></tr>
<tr><td>48</td><td>BITRATE_24</td></tr>
<tr><td>72</td><td>BITRATE_36</td></tr>
</table>
</td></tr>
</table>

RADIOMODE	radioMode	使用中の無線モード。 0 RADIOMODE_B_ONLY 1 RADIOMODE_BG 2 RADIOMODE_G_ONLY 3 RADIOMODE_BG_LRS 4 RADIOMODE_A_ONLY 5 RADIOMODE_ABG 6 RADIOMODE_BGA 7 RADIOMODE_ADHOC

5.4.10 SDCGLOBALCONFIG

```
typedef struct _SDCGlobalConfig
{
DWORD           fragThreshold;
DWORD           RTSThreshold;
RX_DIV          RxDiversity;
TX_DIV          TxDiversity;
ROAM_TRIG       roamTrigger;
ROAM_DELTA      roamDelta;
ROAM_PERIOD     roamPeriod;
PREAMBLE        preamble;
GSHORTSLOT      g_shortslot;
BT_COEXIST      BTcoexist;
PING_PAYLOAD    pingPayload;
DWORD           pingTimeout;
DWORD           pingDelay;
DWORD           radioState;
DWORD           displayPasswords;
DWORD           adminOverride;
DWORD           txMax;
FCC_TEST        FCCtest;
DWORD           testChannel;
BITRATE         testRate;
TXPOWER         testPower;
DWORD           regDomain;
DWORD           ledUsed;
DWORD           txTestTimeout;
DWORD           WMEenabled;
DWORD           CCXfeatures;
CHAR            certPath[MAX_CERT_PATH];
CRYPT           adminPassword;
DWORD           bLRS;
DWORD           avgWindow;
DWORD           probeDelay;
DWORD           polledIRQ;
DWORD           keepAlive;
DWORD           trayIcon;
DWORD           aggScanTimer;
DWORD           authTimeout;
DWORD           autoProfile;
DWORD           Reserved0[6];
DWORD           txMaxA;
DWORD           adminFiles;
DWORD           DFSchannels;
DWORD           interferenceMode;
DWORD           authServerType;
DWORD           Reserved1[4];
} SDCGlobalConfig;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>DWORD</td><td>fragThreshold</td><td>分割されて送信されるパケットサイズ。256～2346(bytes)。
ヘッダファイルに定義されている FRAG_LOW と FRAG_HIGH を参照してください。</td></tr>
<tr><td>DWORD</td><td>RTSThreshold</td><td>RTS/CTS がリンクで必要なパケットサイズ。0～2347(bytes)。
ヘッダファイルに定義されている RTS_LOW と RTS_HIGH を参照してください。</td></tr>
<tr><td>RX_DIV</td><td>RxDiversity</td><td>AP からデータを受信する時のダイバーシティアンテナの処理方法。<table><tr><td>0</td><td>メインアンテナのみ使用。</td></tr><tr><td>1</td><td>補助アンテナのみ使用。</td></tr><tr><td>2</td><td>起動時に補助アンテナを使用。</td></tr><tr><td>3</td><td>起動時にメインアンテナを使用。</td></tr></table></td></tr>
<tr><td>TX_DIV</td><td>TxDiversity</td><td>AP にデータを送信する時のダイバーシティアンテナの処理方法。<table><tr><td>0</td><td>メインアンテナのみ使用。</td></tr><tr><td>1</td><td>補助アンテナのみ使用。</td></tr><tr><td>2</td><td></td></tr><tr><td>3</td><td>ダイバーシティを使用。</td></tr></table></td></tr>
<tr><td>ROAM_TRIG</td><td>roamTrigger</td><td>現在の AP からの移動平均 RSSI がロームトリガよりも弱い場合、無線機は、少なくともロームデルタ dBm の強い信号と AP のためのローミングスキャンを行います。<table><tr><td>50</td><td>-50dBm</td></tr><tr><td>55</td><td>-55dBm</td></tr><tr><td>60</td><td>-60dBm</td></tr><tr><td>65</td><td>-65dBm</td></tr><tr><td>70</td><td>-70dBm</td></tr><tr><td>75</td><td>-75dBm</td></tr><tr><td>80</td><td>-80dBm</td></tr><tr><td>85</td><td>-85dBm</td></tr><tr><td>90</td><td>-90dBm</td></tr></table></td></tr>
<tr><td>ROAM_DELTA</td><td>roamDelta</td><td>ロームトリガーが満たされ、無線が 2 つ目の AP にローミングしようとする場合、2 つ目のの AP の信号強度(RSSI)は、現在の AP の移動平均 RSSI よりも強いロームデルタ dBm である必要があります。<table><tr><td>5</td><td>5dBm</td></tr><tr><td>10</td><td>10dBm</td></tr><tr><td>15</td><td>15dBm</td></tr><tr><td>20</td><td>20dBm</td></tr><tr><td>25</td><td>25dBm</td></tr><tr><td>30</td><td>30dBm</td></tr><tr><td>35</td><td>35dBm</td></tr><tr><td>40</td><td>40dBm</td></tr><tr><td>45</td><td>45dBm</td></tr><tr><td>50</td><td>50dBm</td></tr><tr><td>55</td><td>55dBm</td></tr><tr><td>60</td><td>60dBm</td></tr></table></td></tr>
</table>

<table>
<tr><td>ROAM_PERIOD</td><td>roamPeriod</td><td>結合または(ローミングなしで)ローミングスキャンした後、無線はローミングする前にローミング期間の数秒、RSSI からスキャンデータを収集します。<table><tr><td>5</td><td>5dBm</td></tr><tr><td>10</td><td>10dBm</td></tr><tr><td>15</td><td>15dBm</td></tr><tr><td>20</td><td>20dBm</td></tr><tr><td>25</td><td>25dBm</td></tr><tr><td>30</td><td>30dBm</td></tr><tr><td>35</td><td>35dBm</td></tr><tr><td>40</td><td>40dBm</td></tr><tr><td>45</td><td>45dBm</td></tr><tr><td>50</td><td>50dBm</td></tr><tr><td>55</td><td>55dBm</td></tr><tr><td>60</td><td>60dBm</td></tr></table></td></tr>
<tr><td>PREAMBLE</td><td>preamble</td><td>(未実装)</td></tr>
<tr><td>GSHORTSLOT</td><td>g_shortslot</td><td>(未実装)</td></tr>
<tr><td>BT_COEXIST</td><td>BTcoexist</td><td>(未実装)</td></tr>
<tr><td>PING_PAYLOAD</td><td>pingPayload</td><td>ピン上で送信されるデータ量。。<table><tr><td>32</td><td>32bytes</td></tr><tr><td>64</td><td>64bytes</td></tr><tr><td>128</td><td>128bytes</td></tr><tr><td>256</td><td>256bytes</td></tr><tr><td>512</td><td>512bytes</td></tr><tr><td>1024</td><td>1024bytes</td></tr></table></td></tr>
<tr><td>DWORD</td><td>pingTimeout</td><td>Ping 要求が失敗と判断される応答なしの時間。0～30000(ミリ秒)。
ヘッダファイルに定義されている PTIME_LOW と PTIME_HIGH を参照してください。</td></tr>
<tr><td>DWORD</td><td>pingDelay</td><td>Ping 要求間隔。0～7200000 (ミリ秒)。
ヘッダファイルに定義されている PDELAY_LOW と PDELAY_HIGH を参照してください。</td></tr>
<tr><td>DWORD</td><td>radioState</td><td>無線の有効/無効。<table><tr><td>0</td><td>無効。</td></tr><tr><td>1</td><td>有効。</td></tr></table></td></tr>
<tr><td>DWORD</td><td>displayPasswords</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>adminOverride</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>txMax</td><td>(未実装)</td></tr>
<tr><td>FCC_TEST</td><td>FCCtest</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>testChannel</td><td>(未実装)</td></tr>
<tr><td>BITRATE</td><td>testRate</td><td>(未実装)</td></tr>
<tr><td>TXPOWER</td><td>testPower</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>regDomain</td><td>使用中規定ドメインの確認。<table><tr><td>0</td><td>FCC(アメリカ)</td></tr><tr><td>1</td><td>ETSI(ヨーロッパ)</td></tr><tr><td>2</td><td>TELEC(日本)</td></tr><tr><td>3</td><td>ワールドワイド</td></tr></table></td></tr>
<tr><td>DWORD</td><td>ledUsed</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>txTestTimeout</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>WMEenabled</td><td>Wi-Fi マルチメディア拡張機能の有効/無効。<table><tr><td>0</td><td>無効。</td></tr><tr><td>1</td><td>有効。</td></tr></table></td></tr>
<tr><td>DWORD</td><td>CCXfeatures</td><td>3 つの CCX 機能(ローミング、送信電力、無線管理)の有効/無効。<table><tr><td>0</td><td>無効。</td></tr><tr><td>1</td><td>有効(Cisco AP 限定)。</td></tr></table></td></tr>
<tr><td>CHAR</td><td>certPath [MAX_CERT_PATH]</td><td>証明書のファイルパス。最大 65 文字。
ヘッダファイルに定義されている MAX_CERT_PATH を参照してください。</td></tr>
<tr><td>CRYPT</td><td>adminPassword</td><td>(未実装)</td></tr>
<tr><td>DWORD</td><td>bLRS</td><td>(未実装)</td></tr>
</table>

DWORD	avgWindow	(未実装)
DWORD	probeDelay	(未実装)
DWORD	polledIRQ	(未実装)
DWORD	keepAlive	CAM モードで、ヌルパケットを秒単位で送信する頻度を指定します。(節電されません。) 0 なし。 9 既定値。
DWORD	trayIcon	システムトレイアイコン有効/無効。 0 無効。 1 有効。
DWORD	aggScanTimer	積極的なスキャンがロームトリガー、ロームデルタ、及びローム期間の設定を使用して構成されている標準的なスキャンと連携して補完し動作します。 同じチャネル上にある AP からのカバレッジの重複による同一チャネル干渉があるため、積極的なスキャンがない限り、有効にすることをお勧めします。 0 無効。 1 有効。
DWORD	authTimeout	EAP 証明書。初期値は 8 秒。
DWORD	autoProfile	(未実装)
DWORD	Reserved0[6]	(予備)
DWORD	txMaxA	(未実装)
DWORD	adminFiles	ファイルへのインポート/エクスポート設定許可/不許可。 0 不許可。 1 許可。
DWORD	DFSchannels	(未実装)
DWORD	interferenceMode	干渉モード。 0 オフ。 1 非 WLAN。 2 WLAN。 3 自動。
DWORD	authServerType	認証サーバの種別。 0 ACS(タイプ 1) 1 SBR(タイプ 2)
DWORD	Reserved1[4]	(予備)

5.4.11 エラーコード(SDCERR)

エラーコード	説明
1	SDCERR_FAIL
2	SDCERR_INVALID_NAME
3	SDCERR_INVALID_CONFIG
4	SDCERR_INVALID_DELETE
5	SDCERR_POWERCYCLE_REQUIRED
6	SDCERR_INVALID_PARAMETER
7	SDCERR_INVALID_EAP_TYPE
8	SDCERR_INVALID_WEP_TYPE
9	SDCERR_INVALID_FILE

5.5 Bluetooth 構造体

5.5.1 FTP_FILE_INFO

```
type struct
{
INT             fileType;
WCHAR           filename[200];
INT             fileSize;
} FTP_FILE_INFO;
```

<table>
<tr><th>データ型</th><th>メンバ名</th><th>説明</th></tr>
<tr><td>INT</td><td>fileType</td><td>保存するファイルの種別。<table><tr><td>0</td><td>ファイル</td></tr><tr><td>1</td><td>フォルダ</td></tr></table></td></tr>
<tr><td>WCHAR</td><td>filename[200]</td><td>保存ファイル名。</td></tr>
<tr><td>INT</td><td>fileSize</td><td>ファイル名のサイズ。</td></tr>
</table>

5.6 カメラ構造体

5.6.1 PICSTATE

```
typedef struct
{
INT             AutoExposure;
INT             AutoWhiteBalance;
INT             ImageFormat;
WCHAR           lpPathName[500];
INT             PathSize;
WCHAR           lpFileName[500];
INT             FileSize;
} PICSTATE, *PPICSTATE;
```

データ型	メンバ名	説明
INT	AutoExposure	(予備)
INT	AutoWhiteBalance	(予備)
INT	ImageFormat	画像イメージのフォーマット。 0 JPEG 1 BMP
WCHAR	lpPathName[500]	キャプチャしたイメージの保存先フォルダ。
INT	PathSize	パス名のサイズ。
WCHAR	lpFileName[500]	保存ファイル名。
INT	FileSize	ファイル名のサイズ。

5.6.2 FLASH

```
type struct
        {
DWORD           PreviewFlash;
DWORD           CaptureFlash;
} FLASH, *PFLASH;
```

データ型	メンバ名	説明
DWORD	PreviewFlash	(例えば、画像プレビュー時の)フラッシュライト有効/無効。 0 無効 1 有効
DWORD	CaptureFlash	撮影時のフラッシュ使用。 0 フラッシュなし 1 フラッシュあり 2 自動判断

付録 1　スキャナエンジン設定

9200 シリーズモバイルコンピュータは、以下のリーダのタイプをサポートしています。リーダの可用性はモバイルコンピュータのハードウェアの統合に依存します。

スキャナエンジン		ID
1D	CCD	SM1
1D	Laser	SE955
2D	2D Imager (Software decode)	SE4500
RFID	ID_MOD_MP_RFID	

リーダの組み合わせは、1D+RFID と 2D+RFID のいずれかです。それぞれの組み合わせは、同時に両方のリーダを初期化することができます(デュアルモード動作)。スキャンキーを押したとき、モバイルコンピュータは、1 つの印刷されたバーコード、または 1 つの近接する RFID タグを読み取ります。

[注]：(1)1D および 2D スキャンエンジンは、モバイルコンピュータに共存しません。
両方ともバーコードリーダであり、モバイルコンピュータは、ボード上にひとつのバーコードリーダのみ搭載可能なためです。
(2) リーダを制御しているアプリケーションを同時に 2 つ以上実行しないでください。例えば、MIRROR ブラウザを実行している間は、ターミナルエミュレーション、または ReaderDLL を使用する他のアプリケーションでリーダの設定を実行しないでください。

対応シンボル体系

搭載されているスキャンエンジンにより、以下のシンボル体系をサポートしています。

		CCD	レーザー	2D
Codabar		○	○	○
Code 11		×	○	○
Code 39	Code 39	○	○	○
	Trioptic Code 39	×	○	○
	Italian Pharmacode (Code 32)	○	○	○
Code 93		○	○	○
Code 128	Code 128	○	○	○
	GS1-128 (EAN-128)	○	○	○
	ISBT 128	○	○	○
Code 2 of 5	Chinese 25	×	○	○
	Industrial 25 (Discrete 25)	○	○	○
	Interleaved 25	○	○	○
	Convert Interleaved 25 to EAN-13	×	○	○
	Matrix 25	×	×	○
Composite Code	Composite CC-A/B	×	×	○
	Composite CC-C	×	×	○
	Compostie TLC 39	×	×	○
GS1 DataBar (RSS)	GS1 DataBar-14 (RSS-14)	○	○	○
	GS1 DataBar Limited (RSS Limited)	○	○	○
	GS1 DataBar Expanded (RSS Expanded)	○	○	○
	Convert to UPC/EAN	×	○	○
Inverse	Inverse 1D barcodes	×	×	○
Korean 3 of 5		×	×	○
MSI		○	○	○
Postal Codes	Australian Postal	×	×	○
	Japan Postal	×	×	○
	Netherlands KIX Code	×	×	○
	US Postnet	×	×	○
	US Planet	×	×	○
	UK Postal	×	×	○
EAN/UPC	EAN-8	○	○	○
	EAN-8 Extend	○	○	○
	EAN-13	○	○	○
	Bookland EAN (ISBN)	○	○	○
	UCC Coupon Extended Code	×	○	○
	ISSN EAN	×	×	○
	UPC-A	○	○	○
	UPC-E	○	○	○
	Convert UPC-E to UPC-A	○	○	○
	UPC-E1	○	○	○
	Convert UPC-E1 to UPC-A	○	○	○
2D Symbologies	Aztec	×	×	○
	Data Matrix	×	×	○
	Maxicode	×	×	○
	MicroPDF417	×	×	○
	MicroQR	×	×	○
	PDF417	×	×	○
	QR Code	×	×	○

対応 RFID タグ

RFID リーダは、RFID データの読取りと書込みの両方に対応しています。対応しているラベルは、ISO14443A、ISO15693 と ISO18092 です。
対応している RFID タグは次の通りです。

ID_MOD_MP_RFID		UID のみ	読取り	書込み
ISO 14443A	Mifare Standard S50 1K	○	○	○
	Mifare Standard S70 4K	○	○	○
	Mifare Ultralight UL/ULC	○	○	○
	Mifare DESFire	○		
	SLE66R35	○	○	○
ISO 15693	ICODE SLI	○	○	○
	SRF55V02P	○	○	○
	SRF55V02S	○		
	SRF55V10P	○	○	○
	TI Tag-it HF-I Pro/Plus	○	○	○
	LRI512	○	○	○
ISO 18092	Felica	○		

[注]：読取る前に RFID の仕様を検討してください。

Blank page