CASIO

イメージャライブラリ
マニュアル

このマニュアルは、イメージャライブラリの
仕様について記載します。

Ver.1.14

ご注意

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

変更履歴

バージョン	変更日付	ページ	内容
1.00	2008.11		新規作成
1.01	2009.03	2	開発環境とプログラミング言語の対応表を訂正
		46	IMGSetAztec 関数の解説の表現を修正
		70	IMGSetCode39 関数のパラメータ説明を訂正
		77	IMGSetCode93 関数のパラメータ説明を訂正
		89	IMGSetEAN13 関数の解説の表現を修正
		181	IMGMakeImageFile 関数のパラメータ説明を修正
1.02	2009.07	-	対象機種に DT-5300 を追加
		20-27	IMGSetDecodeWindow 関数を追加
		25-29	IMGGetDecodeWindow 関数を追加
		27	IMGSetDecodeReverse 関数を追加
		28	IMGGetDecodeReverse 関数を追加
		68	IMGSetCode32 関数を追加
		69	IMGGetCode32 関数を追加
		97-108	IMGSetHX 関数を追加
		99-110	IMGGetHX 関数を追加
		198	コード識別表に Code32 と HanXin を追加
1.03	2010.01	20	IMGSetDecodeWindow 関数のパラメータ説明を修正
1.04	2011.01	-	対象機種に DT-X8 を追加
		159	IMGSetIlluminationEx 関数を追加
		161	IMGGetIlluminationEx 関数を追加
1.05	2011.09	-	対象機種に DT-X8-4x シリーズ（フルレンジイメージャモデル）を追加
		190	IMGSetFocus 関数を追加
		191	IMGGetFocus 関数を追加
1.06	2011.11	-	対象機種に IT-9000 を追加
1.07	2014.11	-	対象機種に IT-G500 を追加
		-	関数一覧表をデバイスライブラリ基本マニュアルへ移動
		20	IMGSetDecodeWindow 関数に IT-G500 での注意点を追加
1.08	2015.02	-	対象機種に DT-X100 および DT-X200 を追加
1.09	2016.02	131	IMGSetQR 関数の誤記を修正
1.10	2016.08	20-24	IMGSetDecodeWindow 関数に仕様を追加
1.11	2016.12	194	フローの誤記を修正
1.12	2017.08	20-24	IMGSetDecodeWindow 関数の解説を修正
1.13	2018.07	20-24	IMGSetDecodeWindow 関数の解説を修正
1.14	2021.03	13	IMGWaitForDecode 関数の解説を修正

目次

1. 概要

イメージャライブラリは、本機に搭載されている CMOS 型イメージャを使用してバーコード・2 次元シンボルのデコード（読み取り）、イメージキャプチャ、ストリーミング、サインキャプチャを行う機能を提供します。

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

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

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

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

2. 動作環境

イメージャライブラリの動作環境を以下に示します。

対象機種

DT-5200 / DT-X7 / DT-5300 / DT-X8 / IT-9000 / IT-G500 / DT-X100 / DT-X200

対象 OS

- Microsoft Windows CE 5.0
- Microsoft Windows CE 6.0
- Microsoft Windows Embedded Comapct 7
- Microsoft Windows Mobile 6.5
- Microsoft Windows Embedded Handheld 6.5

開発環境とプログラミング言語

表 2-1

開発環境	Visual C++	Visual Basic, Visual C#
Microsoft embedded Visual C++ Version 4.0 + SP4	○	-
Microsoft Visual Studio.NET 2003 + SP1	×	○
Microsoft Visual Studio 2005 + SP1	○	○
Microsoft Visual Studio 2008	○	○

（○:利用可、×:利用不可、-:機能なし）

提供ファイル

表 2-2

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

（○:必要、-:不要）

使用方法

Visual C++ の場合

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

Visual Basic または Visual C# の場合

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

名前空間とクラス

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

表 2-3

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

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

3. 関数

3.1 IMGInit

イメージャライブラリの初期化を行います。

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

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

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

解説

イメージャライブラリの初期化を行います。この関数を実行するとライブラリで使用する資源を確保し、その初期化を行います。他の関数をコールする前に必ず本関数を実行してください。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_MEMORY	: メモリーエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.2 IMGDeinit

イメージャライブラリの後処理を行います。

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

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

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

解説

イメージャライブラリの後処理を行います。この関数を実行すると IMGInit 関数で確保した全ての資源を解放します。アプリケーションを終了する前に必ず本関数を実行してください。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
FUNCTION_UNSUPPORT	: 未サポートエラー

3.3 IMGConnect

イメージャデバイスを初期化します。

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

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

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

解説

イメージャデバイスを使用可能にします。この関数は IMGDisconnect 関数を実行するまで有効です。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない
IMG_ERR_DRIVER	: イメージャドライバ内でエラー発生 Device Emulator では発生しません
IMG_ERR_INVALID_ACCESS	: 排他エラー（イメージャを使用している他のプログラムを終了してください） Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.4 IMGDisconnect

イメージャデバイスの後処理を行います。

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

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

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

解説

イメージャデバイスの電源をオフし、使用不可状態にします。イメージャを長時間使用しない場合は、本関数をコールしてイメージャを使用不可状態にしてください。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.5 IMGSetDecodeMode

スキャナ読み取り方式を設定します。

```
[C++]
int IMGSetDecodeMode(
  DWORD dwMode,
  DWORD dwNum,
  TCHAR tcSeparator
)
```

```
[Visual Basic]
Public Shared Function IMGSetDecodeMode( _
  ByVal dwMode As Int32, _
  ByVal dwNum As Int32, _
  ByVal tcSeparator As Char _
) As Int32
```

```
[C#]
public static Int32 IMGSetDecodeMode(
  Int32 dwMode,
  Int32 dwNum,
  char tcSeparator
);
```

解説

IMGWaitForDecode 関数、IMGWaitForDecodeRaw 関数の読み取り方式を設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetDecodeMode 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwMode
デコーダの読み取り方式を、以下の値で指定します。

IMG_DECODEMODE_NORMAL	: 通常読み(デフォルト)
IMG_DECODEMODE_MULTISTEP	: 多段読み
IMG_DECODEMODE_PACKAGE	: 一括読み

dwNum
段数(IMG_DECODEMODE_NORMAL 指定時は無視します)を 2～10 の範囲で指定します。

tcSeparator
区切り記号を ASCII 文字で指定します。区切り記号は一括読み指定の時のみ有効となります。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません

IMG_ERR_PARAMETER　　：パラメータエラー
　　Device Emulator では発生しません
FUNCTION_UNSUPPORT　　：未サポートエラー

補足

デコードモードは以下のとおりです。

通常読み	1 回の読み取りで 1 個のシンボルドをデコードします。
多段読み	1 回の読み取りで 1 個のシンボルをデコードしますが、異なるシンボルを連続して読み取ることができます。dwNum パラメータで指定した回数まで読み取りを行えます。
一括読み	1 回の読み取りで複数個のシンボルをデコードます。dwNum パラメータで指定した数のシンボルを読み取ります。読み取ったデータは 1 つの文字列に順番に格納され、tcSeparator パラメータで指定した区切り記号で区切られます。

3.6 IMGGetDecodeMode

スキャナ読み取り方式を取得します。

```
[C++]
int IMGGetDecodeMode(
  LPDWORD pMode,
  LPDWORD pNum,
  PTCHAR ptcSeparator
)
```

```
[Visual Basic]
Public Shared Function IMGGetDecodeMode( _
  ByRef pMode As Int32, _
  ByRef pNum As Int32, _
  ByRef ptcSeparator As Char _
) As Int32
```

```
[C#]
public static Int32 IMGGetDecodeMode(
  ref Int32 pMode,
  ref Int32 pNum,
  ref char ptcSeparator
);
```

解説

IMGWaitForDecode 関数、IMGWaitForDecodeRaw 関数の読み取り方式を取得します。

パラメータ

pMode
読み取り方式を取得します。取得する値の詳細は IMGSetDecodeMode 関数を参照してください。

pNum
読み取り段数を取得します。取得する値の詳細は IMGSetDecodeMode 関数を参照してください。

ptcSeparator
区切り記号を取得します。取得する値の詳細は IMGSetDecodeMode 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.7 IMGWaitForDecode

シンボルのデコードを開始します。

```
[C++]
int IMGWaitForDecode(
  DWORD dwTime,
  PTCHAR pMessage,
  PTCHAR pCodeID,
  PTCHAR pAimID,
  PTCHAR pSymModifier,
  LPDWORD pLength,
  BOOL (*fpCallBack)(void)
)
```

```
[Visual Basic]
Public Shared Function IMGWaitForDecode( _
  ByVal dwTime As Int32, _
  ByVal pMessage As String, _
  ByVal pCodeID As String, _
  ByVal pAimID As String, _
  ByVal pSymModifier As String, _
  ByRef pLength As Int32, _
  ByVal fpCallBack As IntPtr _
) As Int32
```

```
[C#]
public static Int32 IMGWaitForDecode(
  Int32 dwTime,
  string pMessage,
  string pCodeID,
  string pAimID,
  string pSymModifier,
  ref Int32 pLength,
  IntPtr fpCallBack
);
```

解説

この関数を実行するとイメージャはシンボルのデコードを開始します。シンボルのデコードに成功した場合、その結果はこの関数の引数に文字列として返されます。ただし、この関数では NULL や拡張 ASCII コード文字を含むシンボルをデコードすることはできません。NULL 文字を含むシンボルをデコードする場合には IMGWaitForDecodeRaw 関数を使用してください。IMGSetLED 関数、IMGSetBuzzer 関数、IMGSetVibrator 関数により LED、ブザー、バイブレータの通知を有効に設定にした場合、シンボルのデコード成功または失敗時(LED のみ)に通知を行います。また、本関数は IMGSetDecodeMode 関数で設定される読み取り方式(通常読み、多段読み、一括読み)により異なる動作をします。

Device Emulator では I/O Simulator で指定したコードをデコードします。ただし、Code32 および Han Xin(Chinese Sensible)コードのデコードはできません。

■通常読み
1 個のシンボルをデコードします。シンボルがデコードされるか、タイムアウトするか、引数のユーザ関数が FALSE を返すか、IMGStopDecode 関数がコールされた場合にこの関数は終了します。

■多段読み
トリガキーを押し続けている間(ユーザ関数が TRUE を返している間)、連続してデコードをする機能です。一度デコードしたシンボルを再び読み取ることはありません。1 個のシンボルを読むと読み取った結果が pMessage に格納され、制御はアプリケーションに戻ります。アプリケーションは本関数を繰り返しコールすることにより複数シンボルの読み取りを行います。1 個のシンボルがデコードされるか、タイムアウトするか、引数のユーザ関数が FALSE を返すか、IMGStopDecode 関数がコールされた場合にこの関数は終了します。

■一括読み
複数のシンボルを読み取った結果を一括して出力する機能です。IMGSetDecodeMode の dwNum パラメータで指定された数のシンボルがデコードされるか、タイムアウトするか、引数のユーザ関数が FALSE を返すか、または IMGStopDecode 関数がコールされた場合にこの関数は終了します。読み取ったデータは pMessage に IMGSetDecodeMode で指定した区切り記号で区切られて出力されます。区切り記号に NULL を指定した場合は区切り記号を挿入しません。IMGSetDecodeMode の dwNum パラメータで指定された数のシンボルをデコードし終える前に、タイムアウトまたはユーザ関数が FALSE を返すことによりこの関数が終了したときでも、途中までのデコード結果が pMessage に格納されます。

パラメータ

dwTime
イメージャがデコードを開始してから、デコードが完了する前に強制的に終了するまでのタイムアウト時間を、0 より大きい値で、1 ミリ秒単位で指定します。

pMessage
デコードされたシンボルを取得します。デコードされたシンボルを格納するために必要な桁数＋1文字（NULL 文字含む）分の領域を確保してください。デコードされる文字列は現在のコードページ（英語版 OS のデフォルトは ACP）に合わせて UniCode に変換されます。

一括読み方式の場合は、下記のようにデータを取得します。

	1 段目のデータ								区切り記号	2 段目のデータ												終了文字
データ	4	9	1	2	3	4	5	6	/	A	B	C	D	E	F	G	H	I	J	K	L	NULL
バッファ	0							7	8	9											20	21

pCodeID

デコードされたシンボルの Code ID を取得します。取得する値については「コード識別表」を参照してください。

一括読み方式の場合は、下記のようにデータを取得します。

	1 段目のデータ	2 段目のデータ	終了文字
データ	D	S	NULL
バッファ	0	1	2

pAimID

デコードされたシンボルの AIM ID を取得します。取得する値については「コード識別表」の「AIM ID」を参照してください。

一括読み方式の場合は、下記のようにデータを取得します。

	1 段目のデータ	2 段目のデータ	終了文字
データ	E	Q	NULL
バッファ	0	1	2

pSymModifier

デコードされたシンボルの修飾詞を取得します。取得する値については「コード識別表」の「Possible AIM ID Modifiers」を参照してください。

一括読み方式の場合は、下記のようにデータを取得します。

	1 段目のデータ	2 段目のデータ	終了文字
データ	4	1	NULL
バッファ	0	1	2

pLength

デコードされたシンボル文字列の長さを取得します。デコードされたシンボル文字列の長さは現在のコードページ（英語版 OS のデフォルトは ACP）に合わせて算出されます。

一括読み方式の場合は以下の定式で算出される値が格納されます。
（デコードされたシンボル桁数の合計）＋（区切り記号の数）

「区切り記号の数」は（デコードされたシンボルの数－1）であり、区切り記号に NULL を指定した場合は（0）になります。

fpCallBack

引数なしの BOOL 型ユーザ関数アドレスを指定します。（※）
指定のユーザ関数が TRUE を返している間デコードを継続します。FALSE を返した場合、IMGWaitForDecode 関数は IMG_ERR_NOTRIGGER を返してデコードを停止します。この引数に NULL を指定した場合、IMGWaitForDecode 関数は dwTimeout で指定した時間が経過するか（タイムアウトはユーザ関数を指定した場合も有効）、IMGStopDecode 関数がコールされるまでデコードを継続します。

戻り値

IMG_SUCCESS　　：正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない(DT-5200 以外)
IMG_ERR_NOTCONNECTED : IMGConnect が実行されていない
IMG_ERR_ENGINEBUSY : イメージャデバイスがビジー状態のためエラー発生
IMG_ERR_PARAMETER : 不正なパラメータによるエラー発生
Device Emulator では発生しません
IMG_ERR_NOTRIGGER : ユーザ関数により終了した、または IMGStopDecode 関数によりデコードが終了した
IMG_ERR_NODECODE : デコードせずに終了
IMG_ERR_NOIMAGE : 有効なイメージデータが返せない
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

補足

■多段読みと一括読みの違い

多段読みと一括読みの違い

	多段読み	一括読み
最大読み取りシンボル数 (最大読み取り段数)	10 個	10 個(※1)
読み取り最大桁数	4,095 桁	4,095 桁(※1)
読み取り合計桁数	4,095 x 10 = 40,950 桁	4,095 桁
読み取り方法	トリガキーを押している間、指定した数のシンボルを読み取ります。一括読みとは異なり、1 回のスキャンで複数のシンボルを読み取るのではなく、個々のシンボルを連続して読み取ります。 1個のシンボルを読み取り終わると、ブザーが鳴り、LED、バイブレータが点灯します。	1 回のスキャンで複数のシンボルを読み取ります。 指定したすべてのシンボルを読み取るまでブザー鳴動、LED 点灯、バイブレーターは行いません。
推奨する使用方法	以下の場合の読み取りに適しています。 ・シンボルの桁数が多いとき ・読み取るシンボルが離れているとき ・確実に読み取りたいとき	以下の場合の読み取りに適しています。 ・桁数の少ないバーコードが隣接している場合(書籍 JAN コードの読み取りなど)

(※1) 理論的には最大 10 個、4,095 桁のシンボルを読み取ることができますが、一括読みで桁数の多いシンボルを読み取ることは推奨しません。シンボル数が 4 個以上、合計桁数が 100 桁以上となる場合は、多段読みをお使いください。

■デコード処理の中断方法について
デコード関数(IMGWaitForDecode、IMGWaitForDecodeRaw)を、処理の途中で終了させるには、以下の 2 通りの方法があります。

(1) コールバック関数を使用する方法
以下のようなコールバック関数を作成し、その関数のポインタを、IMGWaitForDecode 関数もしくは IMGWaitForDecodeRaw 関数の第 7 引数(fpCallBack)に指定してください。トリガキーが押されている間だけデコードを行う場合は、この方法を使用してください。

```
#define VKEY_L_TRIGGER 0x87
#define VKEY_R_TRIGGER 0x84

// Callback Function fCallBack
// Returns TRUE  : L or R trigger key is pushed down
//         FALSE : L or R trigger key is not pushed down
BOOL fpCallBack( VOID)
{
    if( GetAsyncKeyState(VKEY_L_TRIGGER) < 0 || GetAsyncKeyState(VKEY_R_TRIGGER)
  < 0)
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}
```

(2) IMGStopDecode 関数を使用する方法
デコードを中断したいタイミングで、デコード関数(IMGWaitForDecode、IMGWaitForDecodeRaw 関数)とは別のスレッドから IMGStopDecode 関数をコールしてください。デコード関数の第 7 引数 (fpCallBack)には NULL を指定してください。任意のタイミングでデコード処理を中断したい場合は、この方法を使用してください。

■データ連結モードについて
連結識別子を含む Code93 コード、Code49 コード、QR コードを読むと、デコーダはデータ連結モードとなります。これらのシンボルを読み取ると、デコード関数(IMGWaitForDecode、IMGWaitForDecodeRaw 関数)の pMessage パラメータにはデコードデータは格納されず、pLength パラメータは 0 が返り、戻り値は IMG_SUCCESS となります。最終データとなるシンボル(シンボルにより異なります)を読むとデータ連結モードを終了し、これまでに読み取ったデコードデータ(連結識別子は省かれます)と最後に読み取ったデコードデータが連結して出力されます。
たとえば、Code93 コードの連結識別子は先頭のスペースです。データが(1)「<space>123」、(2)「<space>456」、(3)「789」の Code93 コードを順番に読むと、(1)、(2)を読み取ったときには pMessage にデコード結果は出力されません。そのあと(3)のデータを読み取ると、pMessage に「123456789」が返り、pLength は 9 となります。
また、データ連結モードでコードを読み取る場合は、デコードモードを通常モードに指定してください。

3.8 IMGWaitForDecodeRaw

バイナリデータを含むシンボルコードのデコードを実行します。

```
[C++]
int IMGWaitForDecodeRaw(
  DWORD dwTime,
  LPBYTE pMessage,
  PTCHAR pCodeID,
  PTCHAR pAimID,
  PTCHAR pSymModifier,
  LPDWORD pLength,
  BOOL (*fpCallBack)(void)
)
```

```
[Visual Basic]
Public Shared Function IMGWaitForDecodeRaw( _
  ByVal dwTime As Int32, _
  ByRef pMessage As Byte, _
  ByVal pCodeID As String, _
  ByVal pAimID As String, _
  ByVal pSymModifier As String, _
  ByRef pLength As Int32, _
  ByVal fpCallBack As IntPtr _
) As Int32
```

```
[C#]
public static Int32 IMGWaitForDecodeRaw(
  Int32 dwTime,
  ref Byte pMessage,
  string pCodeID,
  string pAimID,
  string pSymModifier,
  ref Int32 pLength,
  IntPtr fpCallBack
);
```

解説

バイナリデータを含むシンボルコードのデコードを実行します。この関数はバイナリデータを含むシンボルデータを読み取る場合に使用します。NULL 文字や拡張 ASCII コードを含むシンボルデータを読み取る場合にもこの関数を使用してください。関数を実行するとイメージャは、デコードを始めます。シンボルコードがデコードされた場合、その結果はこの関数のパラメータの中に返されます。また、IMGSetLED 関数、IMGSetBuzzer 関数、IMGSetVibrator 関数により LED、ブザー、バイブレーターの通知を行う設定にした場合、シンボルのデコード成功または失敗時(LED のみ)に通知を行います。また、本関数は IMGSetDecodeMode 関数で設定される読み取り方式(通常読み、多段読み)により異なる動作をします。

Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

dwTime
イメージャがデコードを開始してから、デコードが完了する前に強制的に終了するまでのタイムアウト時間を、0 より大きい値で、ミリ秒単位で指定します。詳細は、IMGWaitForDecode 関数を参照してください。

pMessage
デコードされたシンボルコードを格納するための BYTE 型変数へのポインタ。
デコードされるシンボルコードを格納するために必要なバイト数分の領域を、BYTE 型の配列として確保してください。

pCodeID
デコードされたシンボルの Code ID を取得します。詳細は IMGWaitForDecode 関数を参照してください。

pAimID
デコードされたシンボルの AIM ID を取得します。詳細は IMGWaitForDecode 関数を参照してください。

pSymModifier
デコードされたシンボルの修飾詞を取得します。詳細は IMGWaitForDecode 関数を参照してください。

pLength
デコードされたシンボルコードのバイト数を取得します。

fpCallBack
引数なしの BOOL 型ユーザ関数アドレスを指定します。詳細は IMGWaitForDecode 関数を参照してください。

戻り値

戻り値については IMGWaitForDecode 関数を参照してください。

補足

一括読みはサポートしません。
通常読み/多段読みについては、IMGWaitForDecode 関数を参照してください。

■デコード処理の中断方法について
本関数を処理の途中で終了する方法については、IMGWaitForDecode 関数を参照してください。

3.9 IMGStopDecode

デコードを強制停止します。

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

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

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

解説

デコードを強制停止します。IMGWaitForDecode 関数または IMGWaitForDecodeRaw 関数を実行中にこの関数をコールすると、イメージャはデコードを停止します。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.10 IMGSetDecodeWindow

デコードウィンドウ(シンボル読み取り範囲)を設定します。

```
[C++]
int IMGSetDecodeWindow(
  int nMode,
  int nLeft,
  int nTop,
  int nRight,
  int nBottom
)
```

```
[Visual Basic]
Public Shared Function IMGSetDecodeWindow( _
  ByVal nMode As Int32, _
  ByVal nLeft As Int32, _
  ByVal nTop As Int32, _
  ByVal nRight As Int32, _
  ByVal nBottom As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetDecodeWindow(
  Int32 nMode,
  Int32 nLeft,
  Int32 nTop,
  Int32 nRight,
  Int32 nBottom
)
```

解説

デコードウィンドウ(シンボル読み取り範囲)を設定します。
デコードウィンドウを設定することにより、シンボルを読み取る範囲を限定することができます。
デコードウィンドウに IMG_DW_CENTER1 もしくは IMG_DW_CENTER2 を設定すると、エイマー中心付近のシンボルのみを読み取ることができます。
また、デコードウィンドウに IMG_DW_USER を設定すると、読み取り範囲を座標で指定することができます。

IMG_DW_LIMITED_CENTER1、IMG_DW_LIMITED_CENTER2、IMG_DW_LIMITED_USER は、より厳密にエイマー中心付近のシンボルのみを読むときに使用します。狭い範囲に小さいシンボルが密集していて、エイマー中心付近に複数のシンボルが入る可能性がある場合に本パラメータを使用してください。

DeviceEmulator では、設定値を内部変数として格納するため、何も動作しませんが、IMGGetDecodeWindow 関数を実行することにより、設定値を確認することができます。

パラメータ

nMode

デコードウィンドウモードを指定します。

IMG_DW_NONE	: デコードウィンドウ無効(デフォルト)
IMG_DW_CENTER1	: エイマー中心付近の単一シンボルを読み取ります
IMG_DW_CENTER2	: エイマー中心付近の Composite コードを読み取ります
IMG_DW_USER	: デコードウィンドウを座標で指定します
IMG_DW_LIMITED_CENTER1	: エイマー中心付近の単一シンボルを読み取ります(密集したシンボルを読む場合)
IMG_DW_LIMITED_CENTER2	: エイマー中心付近の Composite コードを読み取ります(密集したシンボルを読む場合)
IMG_DW_LIMITED_USER	: デコードウィンドウを座標で指定します(密集したシンボルを読む場合)

※ DT-X8-4x シリーズでは IMG_DW_NONE および IMG_DW_CENTER1 のみ有効です

※ IMG_DW_LIMITED_CENTER1、IMG_DW_LIMITED_CENTER2、IMG_DW_LIMITED_USER は IT-G500、DT-X100、DT-X200 で ServicePack 1.05 以上をインストールした場合のみ有効になります。

nLeft

デコードウィンドウ左上の X 座標を指定します。
nMode に IMG_DW_USER もしくは IMG_DW_LIMITED_USER を指定したときのみ有効となります。

■ DT-X7、DT-X8、DT-5300、IT-9000A/B の場合
0～751(デフォルト 0)の範囲で指定します。
■ その他のモデルの場合
0～830(デフォルト 0)※の範囲で指定します。
※ 下記のパッチをインストールしていない場合は 0～414 の範囲で指定してください。
- IT-G500、DT-X100、DT-X200 の ServicePack1.08 以上
- IT-9000C CE の ImgaerPatch1.02 以上
- IT-9000C WEH の ImgaerPatch1.01 以上

nTop

デコードウィンドウ左上の Y 座標を指定します。
nMode に IMG_DW_USER もしくは IMG_DW_LIMITED_USER を指定したときのみ有効となります。

■ DT-X7、DT-X8、DT-5300、IT-9000A/B の場合
0～479(デフォルト 0)の範囲で指定します。
■ その他のモデルの場合
0～638(デフォルト 0)※の範囲で指定します。
※ 下記のパッチをインストールしていない場合は 0～318 の範囲で指定してください。
- IT-G500、DT-X100、DT-X200 の ServicePack1.08 以上
- IT-9000C CE の ImgaerPatch1.02 以上
- IT-9000C WEH の ImgaerPatch1.01 以上

nRight

デコードウィンドウ右下の X 座標を指定します。
nMode に IMG_DW_USER もしくは IMG_DW_LIMITED_USER を指定したときのみ有効となります。

■ DT-X7、DT-X8、DT-5300、IT-9000A/B の場合

0～751（デフォルト 751）の範囲で指定します。
■ その他のモデルの場合
1～831（デフォルト 831）※の範囲で指定します。
※ 下記のパッチをインストールしていない場合は 416～831 の範囲で指定してください。
- IT-G500、DT-X100、DT-X200 の ServicePack1.08 以上
- IT-9000C CE の ImgaerPatch1.02 以上
- IT-9000C WEH の ImgaerPatch1.01 以上

nBottom
デコードウィンドウ右下の Y 座標を指定します。
nMode に IMG_DW_USER もしくは IMG_DW_LIMITED_USER を指定したときのみ有効となります。

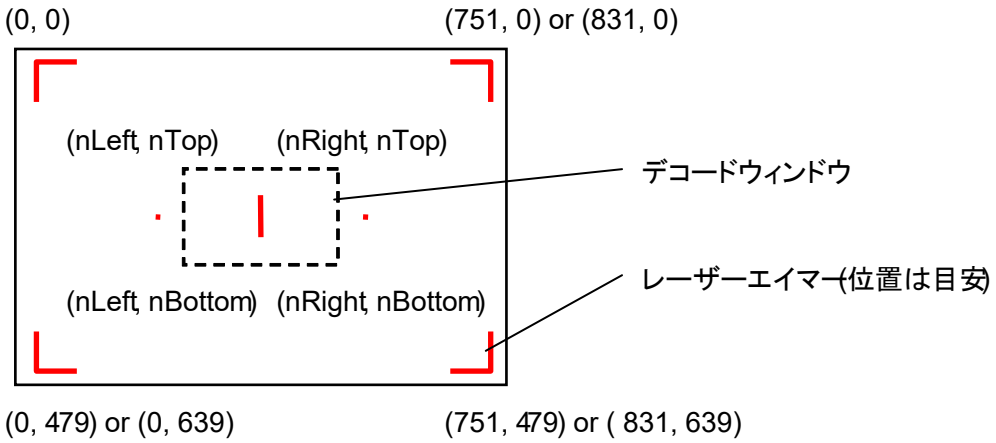
■ DT-X7、DT-X8、DT-5300、IT-9000A/B の場合
0～479（デフォルト 479）の範囲で指定します。
■ その他のモデルの場合
1～639（デフォルト 639）※の範囲で指定します。
※ 下記のパッチをインストールしていない場合は 320～639 の範囲で指定してください。
- IT-G500、DT-X100、DT-X200 の ServicePack1.08 以上
- IT-9000C CE の ImgaerPatch1.02 以上
- IT-9000C WEH の ImgaerPatch1.01 以上

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

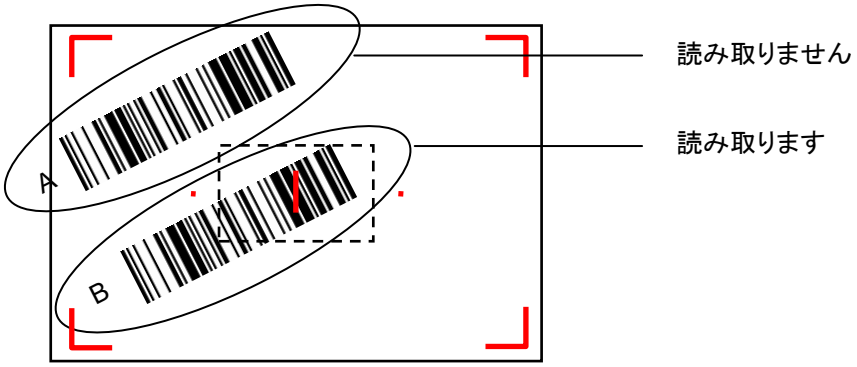
使用方法

イメージャのキャプチャサイズ（752 x 480 pixel もしくは 832 x 640 pixel）に対して、デコードウィンドウ（シンボルを読み取る範囲：矩形）を座標で指定します。



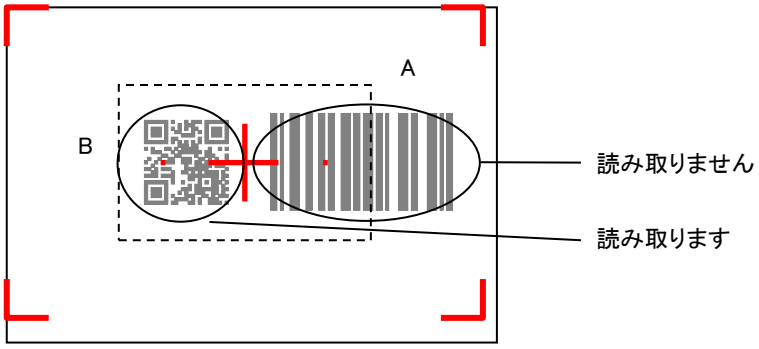
■ IMG_DW_CENTER1、IMG_DW_CENTER2、IMG_DW_USER の場合
デコードウィンドウを指定してシンボルをスキャンすると、デコードウィンドウにシンボルの一部が含まれた場合のみ、そのシンボルを読み取ります。下図の場合、シンボル A はデコードウィンドウ内に含まれ

ていないため読み取りを行いませんが、シンボル B はシンボルの一部が含まれているため読み取りを行います。



■ IMG_DW_LIMITED_CENTER1、IMG_DW_LIMITED_CENTER2、IMG_DW_LIMITED_USER の場合

指定したデコードウィンドウにシンボルの**全体**が含まれた場合のみ、そのシンボルを読み取ります。下図の場合、シンボル A はシンボル全体がデコードウィンドウ内に含まれていないため読み取りを行いませんが、シンボル B はシンボルの全体が含まれているため読み取りを行います。



※ エイマーの照射位置は目安です。シンボルのスキャナの距離や角度などで変わることがあります。

補足

nMode に IMG_DW_CENTER1、IMG_DW_CENTER2、IMG_DW_LIMITED_CENTER1、IMG_DW_LIMITED_CENTER2 を指定した場合は、以下の座標値を自動的に設定します。

■ DT-X7、DT-X8、DT-5300、IT-9000A/B の場合

	nLeft	nTop	nRight	nBottom
IMG_DW_CENTER1	344	234	408	246
IMG_DW_CENTER2	344	214	408	266

■ その他のモデルの場合

	nLeft	nTop	nRight	nBottom
IMG_DW_CENTER1	381	312	451	328
IMG_DW_CENTER2	381	285	451	355
IMG_DW_LIMITED_CENTER1	298	192	610	432
IMG_DW_LIMITED_CENTER2	194	112	714	512

IT-G500、DT-X100、DT-X200、IT-9000C で nMode に IMG_DW_USER を指定した場合は、デコードウィンドウが画面の中心（416, 320）を含むように座標を指定※してください。

※ 下記のパッチをインストールしていない場合のみ

- IT-G500、DT-X100、DT-X200 の ServicePack1.08 以上
- IT-9000C CE の ImgaerPatch1.02 以上
- IT-9000C WEH の ImgaerPatch1.01 以上

3.11 IMGGetDecodeWindow

デコードウィンドウ(シンボル読み取り範囲)を取得します。

```
[C++]
int IMGGetDecodeWindow(
  int *pMode,
  int *pLeft,
  int *pTop,
  int *pRight,
  int *pBottom
)
```

```
[Visual Basic]
Public Shared Function IMGGetDecodeWindow( _
  ByRef pMode As Int32, _
  ByRef pLeft As Int32, _
  ByRef pTop As Int32, _
  ByRef pRight As Int32, _
  ByRef pBottom As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetDecodeWindow(
  ref Int32 pMode,
  ref Int32 pLeft,
  ref Int32 pTop,
  ref Int32 pRight,
  ref Int32 pBottom
)
```

解説

デコードウィンドウ(シンボル読み取り範囲)を取得します。

パラメータ

pMode

デコードウィンドウモードを取得します。取得する値については IMGSetDecodeWindow 関数を参照してください。

pLeft

デコードウィンドウ左上の X 座標を取得します。取得する値については IMGSetDecodeWindow 関数を参照してください。

pTop

デコードウィンドウ左上の Y 座標を取得します。取得する値については IMGSetDecodeWindow 関数を参照してください。

pRight

デコードウィンドウ右下の X 座標を取得します。取得する値については IMGSetDecodeWindow 関数

を参照してください。

pBottom

デコードウィンドウ右下の X 座標を取得します。取得する値については IMGSetDecodeWindow 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

注意

DT-X8-4x シリーズ（フルレンジイメージャモデル）ではデコードウィンドウの座標を取得することはできません。それぞれ NULL を指定してください。

3.12 IMGSetDecodeReverse

デコードの白黒反転の有無を設定します。

```
[C++]
int IMGSetDecodeReverse(
  int nDecodeReverse
)
```

```
[Visual Basic]
Public Shared Function IMGSetDecodeReverse( _
  ByVal nDecodeReverse As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetDecodeReverse(
  Int32 nDecodeReverse
)
```

解説

デコードの白黒反転の有無を設定します。
白黒反転に設定すると、黒地に白で印刷したバーコードやシンボルを読み取れる可能性があります。
DeviceEmulator では、設定値を内部変数として格納するため、何も動作しませんが、
IMGGetDecodeReverse 関数を実行することにより、設定値を確認することができます。

パラメータ

nDecodeReverse
デコードの白黒反転の有無を指定します。

IMG_DR_POSITIVE	: 白黒反転していないシンボルを読み取ります(デフォルト)
IMG_DR_NEGATIVE	: 白黒反転しているシンボルを読み取ります
IMG_DR_POSITIVE_NEGATIVE	: 白黒反転したシンボル、反転していないシンボルの両方を読み取ります

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.13 IMGGetDecodeReverse

デコードの白黒反転の有無を取得します。

```
[C++]
int IMGGetDecodeReverse(
  int *pDecodeReverse
)
```

```
[Visual Basic]
Public Shared Function IMGGetDecodeReverse( _
  ByRef pDecodeReverse As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetDecodeReverse(
  ref Int32 pDecodeReverse
)
```

解説

デコードの白黒反転の有無を取得します。

パラメータ

pDecodeReverse

デコードの白黒反転の有無を取得します。取得する値については IMGSetDecodeReverse 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.14 IMGGetImage

静止画を撮影します。

```
[C++]
int IMGGetImage(
  LPBYTE pImageBuffer,
  LPDWORD pSize,
  int nLeft,
  int nTop,
  int nRight,
  int nBottom,
  DWORD dwSkip,
  DWORD dwFormat,
  DWORD dwWhiteValue
)
```

```
[Visual Basic]
Public Shared Function IMGGetImage( _
  ByRef pImageBuffer As Byte, _
  ByRef pSize As Int32, _
  ByVal nLeft As Int32, _
  ByVal nTop As Int32, _
  ByVal nRight As Int32, _
  ByVal nBottom As Int32, _
  ByVal dwSkip As Int32, _
  ByVal dwFormat As Int32, _
  ByVal dwWhiteValue As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetImage(
  ref Byte pImageBuffer,
  ref Int32 pSize,
  Int32 nLeft,
  Int32 nTop,
  Int32 nRight,
  Int32 nBottom,
  Int32 dwSkip,
  Int32 dwFormat,
  Int32 dwWhiteValue
);
```

解説

静止画を撮影します。イメージ取得座標、イメージデータのサイズおよびイメージのフォーマットを指定し、引数で指定されたバッファにイメージデータ、イメージデータのサイズを格納します。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

pImageBuffer
イメージデータを取得します。撮影するイメージデータのバイト数分領域を確保してください。
pImageBuffer に格納されるデータは、dwFormat の値に関わらず 1 ピクセルにつき 8 ビットとなります。
左上のピクセルから始まり、左から右、上から下の向きに連続的に格納されます。

pSize
pImageBuffer に格納されたイメージデータのバイト数を取得します。
イメージデータのバイト数は以下の定式により算出できます。
イメージデータのバイト数 = ((nRight－nLeft)÷dwSkip)×((nBottom－nTop)÷dwSkip)

nLeft
撮影するイメージデータの始点となる X 座標を指定します。左端の座標は 0 になります。
(nRight－nLeft)が偶数になるように指定してください。

nTop
撮影するイメージデータの始点となる Y 座標を指定します。上端の座標は 0 になります。

nRight
撮影するイメージデータの終点となる X 座標を指定します。
(nRight－nLeft)が偶数になるように指定してください。

nBottom
撮影するイメージデータの終点となる Y 座標を指定します。下端の座標は 480 になります。
上記の座標に関するパラメータの詳細は、後述の＜イメージデータの座標指定について＞を参照してください。

dwSkip
撮影するイメージデータのサイズを縦横の倍率で指定します。
横 1/dwSkip 倍、縦 1/dwSkip 倍のイメージデータが pImageBuffer に格納されます。
このパラメータには(nRight－nLeft)を割り切れる値を指定してください。

dwFormat
イメージのフォーマットを、以下の値で指定します。
IMAGE_256MONO ： モノクロ 256 階調
IMAGE_2MONO ： モノクロ 2 階調

nWhiteValue
イメージャが自動露光制御を行う際に目標とする明るさを、0 から 255 までの範囲で指定します。数値を大きくすると明るめに、小さくすると暗めになります。200 を目安に好みの明るさになるように調整してください。また、0 を指定するとイメージャが自動調整を行いますが、全体的に暗めのイメージデータになります。

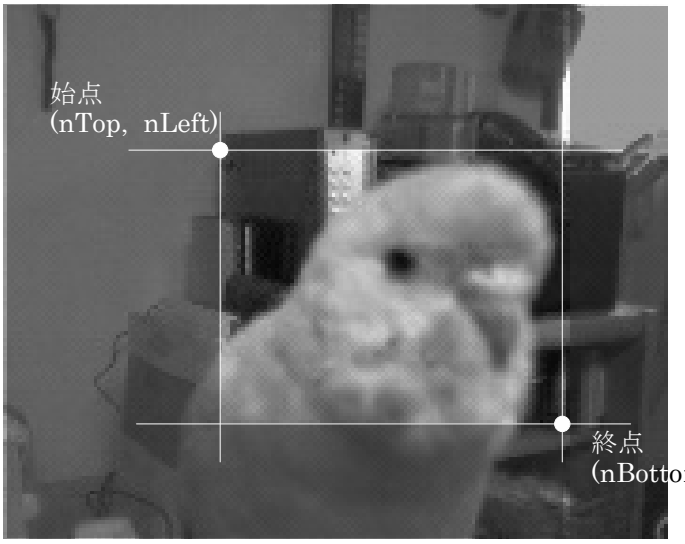
戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-5200 以外) Device Emulator では発生しません
IMG_ERR_NOTCONNECTED	: IMGConnect が実行されていない Device Emulator では発生しません
IMG_ERR_DRIVER	: イメージャドライバ内でエラー発生 Device Emulator では発生しません
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー発生 Device Emulator では発生しません
IMG_ERR_NORESPONSE	: デバイスドライバからの応答がない Device Emulator では発生しません
IMG_ERR_BADREGION	: 座標指定が許容範囲を超えている Device Emulator では発生しません
IMG_ERR_MEMORY	: OS からメモリを割り当てられなくなりエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

イメージデータの座標指定について

取得するイメージデータの座標は以下のように始点と終点を指定します。

(0, 0)　　(640 or 752, 0)



(0, 480)　　(640 or 752, 480)

図 3.1 座標指定例

※(nRight－nLeft)が偶数になるように指定してください。

BMP 形式での保存について

撮影したイメージデータを BMP 形式で保存する場合には(nRight－nLeft)が 4 の倍数になるように指定してください。

3.15 IMGStartStream

イメージストリーミングを開始します。

```
[C++]
int IMGStartStream(
  int nLeft,
  int nTop,
  int nRight,
  int nBottom,
  DWORD dwSkip,
  DWORD dwFormat
)
```

```
[Visual Basic]
Public Shared Function IMGStartStream( _
  ByVal nLeft As Int32, _
  ByVal nTop As Int32, _
  ByVal nRight As Int32, _
  ByVal nBottom As Int32, _
  ByVal dwSkip As Int32, _
  ByVal dwFormat As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGStartStream(
  Int32 nLeft,
  Int32 nTop,
  Int32 nRight,
  Int32 nBottom,
  Int32 dwSkip,
  Int32 dwFormat
);
```

解説

イルミネーション LED を点灯し、イメージストリーミングを開始します。イメージ取得座標、イメージデータのサイズを指定できます。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

nLeft
撮影するイメージデータの始点となる X 座標を指定します。左端の座標は 0 になります。

nTop
撮影するイメージデータの始点となる Y 座標を指定します。上端の座標は 0 になります。

nRight
撮影するイメージデータの終点となる X 座標を指定します。

nBottom
撮影するイメージデータの終点となる Y 座標を指定します。下端の座標は 480 になります。
上記の座標に関するパラメータは後述の＜イメージデータの座標指定について＞も参照してください。

dwSkip
座標で指定したエリアを取得したイメージデータの倍率を指定します。横 1/dwSkip 倍、縦 1/dwSkip 倍のイメージデータが IMGGetStreamData 関数の pImageBuffer に格納されます。
このパラメータには(nRight－nLeft)を割り切れる値を指定してください。

dwFormat
イメージのフォーマットを以下の値で指定します。

IMAGE_256MONO	： モノクロ 256 階調。

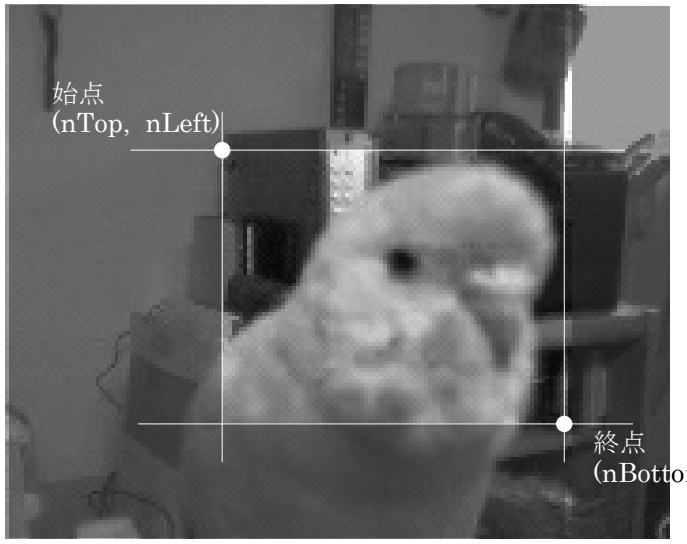
戻り値

IMG_SUCCESS	： 正常終了
IMG_ERR_NOTINITIALIZED	： IMGInit が実行されていない(DT-5200 以外) Device Emulator では発生しません
IMG_ERR_NOTCONNECTED	： IMGConnect が実行されていない Device Emulator では発生しません
IMG_ERR_PARAMETER	： 不正なパラメータによるエラー発生 Device Emulator では発生しません
IMG_ERR_MEMORY	： OS からメモリを割り当てられなくなりエラー発生 Device Emulator では発生しません
IMG_ERR_DRIVER	： イメージャドライバ内でエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	： 未サポートエラー

イメージデータの座標指定について

イメージデータの座標は以下のように始点と終点を指定します。

(0, 0)　　(640 or 752, 0)



(0, 480)　　(640 or 752, 480)

図 3.2 座標指定例

※(nRight－nLeft)が偶数になるように指定してください。

3.16 IMGGetStreamData

イメージストリーミングデータを取得します。

```
[C++]
int IMGGetStreamData(
  LPBYTE pImageBuffer,
  LPDWORD pSize
)
```

```
[Visual Basic]
Public Shared Function IMGGetStreamData( _
  ByRef pImageBuffer As Byte, _
  ByRef pSize As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetStreamData(
  ref Byte pImageBuffer,
  ref Int32 pSize
);
```

解説

イメージストリーミングデータを取得します。この関数で取得したイメージデータを繰り返し表示することにより、ストリーミングを実現します。この関数を呼ぶ前に、IMGStartStream 関数を呼び出す必要があります。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

pImageBuffer
イメージデータを取得します。取得するイメージデータのバイト数分領域を確保してください。
pImageBuffer に格納されるデータは 1 ピクセルにつき 8 ビット。左上のピクセルから始まり、左から右、上から下の向きに連続的に格納されます。

pSize
pImageBuffer に格納されたイメージデータのバイト数を取得します。
イメージデータのバイト数は以下の定式により算出できます。
(((nRight－nLeft)÷dwSkip)×((nBottom－nTop)÷dwSkip))

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_MEMORY	: IMGInit が実行されていない(DT-5200 以外) Device Emulator では発生しません
IMG_ERR_NOIMAGE	: 有効なイメージデータが返せない Device Emulator では発生しません
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.17 IMGStopStream

イメージストリーミングを停止します。

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

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

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

解説

IMGStartStream 関数にて開始したストリーミングを終了します。ストリーミングを終了する場合は、必ずこの関数を実行してください。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
FUNCTION_UNSUPPORT	: 未サポートエラー

3.18 IMGCaptureSign

サインキャプチャデータを取得します。

```
[C++]
int IMGCaptureSign(
  LPBYTE pImageBuffer,
  DWORD dwAspectRatio,
  int nOffsetX,
  int nOffsetY,
  DWORD dwWidth,
  DWORD dwHeight,
  int nResolution,
  DWORD dwFormat
)
```

```
[Visual Basic]
Public Shared Function IMGCaptureSign( _
  ByRef pImageBuffer As Byte, _
  ByVal dwAspectRatio As Int32, _
  ByVal nOffsetX As Int32, _
  ByVal nOffsetY As Int32, _
  ByVal dwWidth As Int32, _
  ByVal dwHeight As Int32, _
  ByVal nResolution As Int32, _
  ByVal dwFormat As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGCaptureSign(
  ref Byte pImageBuffer,
  Int32 dwAspectRatio,
  Int32 nOffsetX,
  Int32 nOffsetY,
  Int32 dwWidth,
  Int32 dwHeight,
  Int32 nResolution,
  Int32 dwFormat
);
```

解説

サインキャプチャデータを取得します。シンボルコードより、指定したオフセット位置にあるイメージデータを切り出し取得します。イメージデータを取得する為に、あらかじめ IMGWaitForDecode 関数または IMGWaitForDecodeRaw 関数を実行してデコードを行ってください。この関数ではデコードを行った際に内部で取得されたイメージデータより指定した位置と大きさのイメージデータを切り出し、バッファに格納します。
サポートしているシンボルコードは Aztec、Codabar(NW-7)、Code128、Code39 および PDF417 です。
また、サポートしているデコードモードは通常読みのみです。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

pImageBuffer
イメージを取得します。取得するイメージデータのサイズ分を確保してください。pImageBuffer に格納されたデータは、1 ピクセルにつき 8 ビット、左上のピクセルから始まり、左から右、上から下の向きに連続的に格納されます。
取得するイメージデータのサイズは次の定式を使用して算出することができます。
(dwWidth×nResolution)×(dwHeight×nResolution)

dwAspectRatio
Aspect 比を指定します。1次元バーコードの場合はシンボルコードの高さと狭バー幅の比率、2次元シンボルコードの場合はシンボルコードの列の高さと最小セル幅の比率を指定します。

nOffsetX
シンボルコードの中心を原点としたときの、取得するイメージの中心の X 座標を指定します。

nOffsetY
シンボルコードの中心を原点としたときの、取得するイメージの中心の Y 座標を指定します。

dwWidth
取得するイメージの幅を指定します。(dwWidth×nResolution)が偶数になるように指定してください。

dwHeight
取得するイメージの高さを指定します。

nResolution
狭バー幅または最小セル幅ごとに使用するピクセルの数を指定します。より高い解像度はよりよい質のイメージを生成しますが、サイズもより大きなものになります。

dwFormat
イメージのフォーマットを、以下の値で指定します。
IMAGE_256MONO　：モノクロ 256 階調
IMAGE_2MONO　：モノクロ 2 階調

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_DRIVER	：イメージャドライバ内でエラー発生 Device Emulator では発生しません
IMG_ERR_PARAMETER	：不正なパラメータによるエラー発生 Device Emulator では発生しません
IMG_ERR_BADREGION	：イメージが無効な領域にあるためエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

パラメータの算出方法

この関数では、シンボルコードの寸法から算出した値をパラメータとして使用します。イメージのサイズと位置のパラメータは、シンボルコードの狭バー幅または最小セル幅を基準としたときの比で指定します。したがって、単位系に依存することなく次の手順でパラメータを算出することができます。

■ Codabar(NW7)、Code39 および Code128

1. dwAspectRatio の算出

狭バー幅とバーの高さを測定し、dwAspectRatio に指定する値を算出します。

dwAspectRatio＝(バーの高さ)÷(狭バーの幅)

2. dwWidth と dwHeight の算出

切り出す部分のイメージの幅と高さを測定し、dwWitdth と dwHeight に指定する値を算出します。

dwWidth＝(切り出すイメージの幅)÷(狭バーの幅)

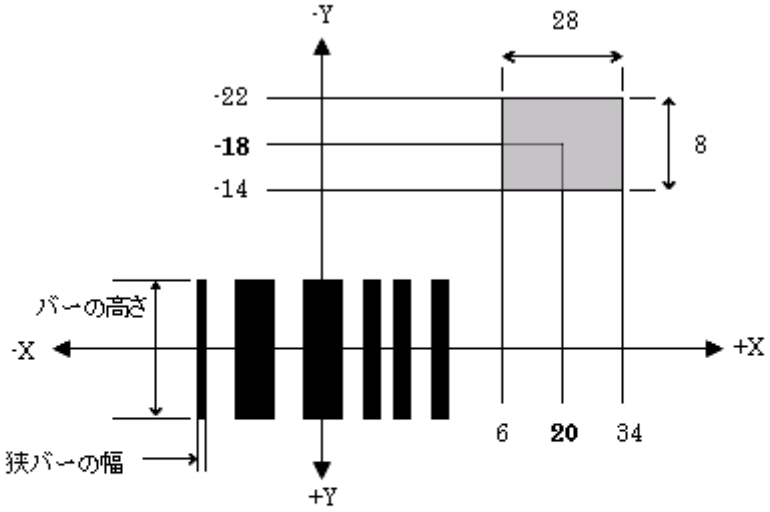
dwHeight＝(切り出すイメージの高さ)÷(狭バーの幅)

3. nOffsetX と nOffsetY の算出

シンボルコードの中心から切り出すイメージの中心までの距離を測定し、nOffsetX と nOffsetY に指定する値を算出します。

nOffsetX＝(シンボルコードの中心からイメージの中心までの横方向距離)÷(狭バーの幅)

nOffsetY＝(シンボルコードの中心からイメージの中心までの縦方向距離)÷(狭バーの幅)



狭バーの幅 = 0.5mm、バーの高さ = 10mmの場合

dwAspectRatio = (10 / 0.5) = 20
dwWidth = (28 / 0.5) = 56
dwHeight = (8 / 0.5) = 16
nOffsetX = (20 / 0.5) = 40
nOffsetY = (-18 / 0.5) = -36

※(dwWidth×nResolution)が偶数になるように指定してください。

■ PDF417

1. dwAspectRatio の算出

最小セルの幅と列の高さを測定し、dwAspectRatio に指定する値を算出します。

dwAspectRatio＝(最小セルの高さ) ÷(最小セルの幅)

2. dwWidth と dwHeight の算出

切り出す部分のイメージの幅と高さを測定し、nWitdth と dwHeight に指定する値を算出します。

dwWidth＝(切り出すイメージの幅) ÷(最小セルの幅)

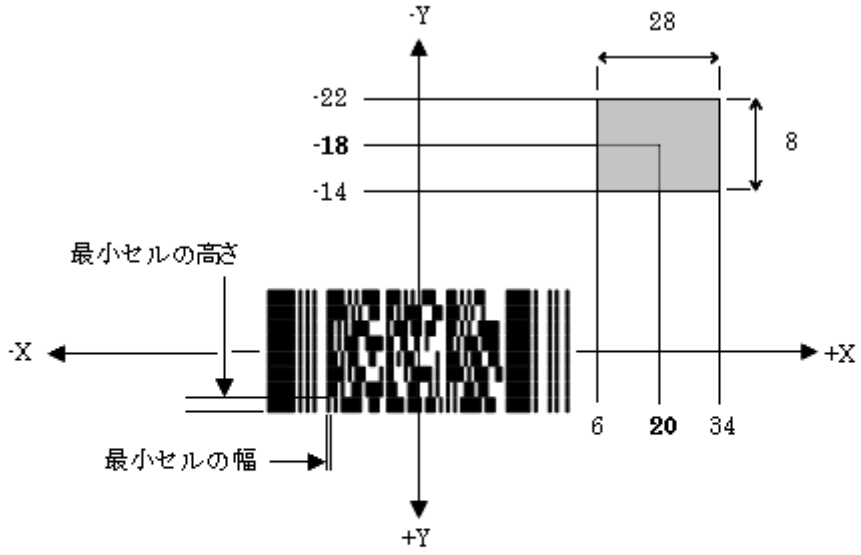
dwHeight＝(切り出すイメージの高さ) ÷(最小セルの幅)

3. nOffsetX と nOffsetY の算出

シンボルコードの中心から切り出すイメージの中心までの距離を測定し、nOffsetX と nOffsetY に指定する値を算出します。

nOffsetX＝(シンボルコードの中心からイメージの中心までの横方向距離) ÷(最小セルの幅)

nOffsetY＝(シンボルコードの中心からイメージの中心までの縦方向距離) ÷(最小セルの幅)



最小セルの幅＝0.5mm、最小セルの高さ＝3mmの場合
dwAspectRatio = (3 / 0.5) = 6
dwWidth = (28 / 0.5) = 56
dwHeight = (8 / 0.5) = 16
nOffsetX = (20 / 0.5) = 40
nOffsetY = (-18 / 0.5) = -36

※(dwWidth×nResolution)が偶数になるように指定してください。

■ Aztec

1. dwAspectRatio の算出

dwAspectRatio＝1(最小セルは正方形のため固定値)

2. dwWidth と dwHeight の算出

最小セルの幅、切り出す部分のイメージの幅と高さを測定し、nWitdth と dwHeight に指定する値を算出します。

dwWidth＝(切り出すイメージの幅)÷(最小セルの幅)

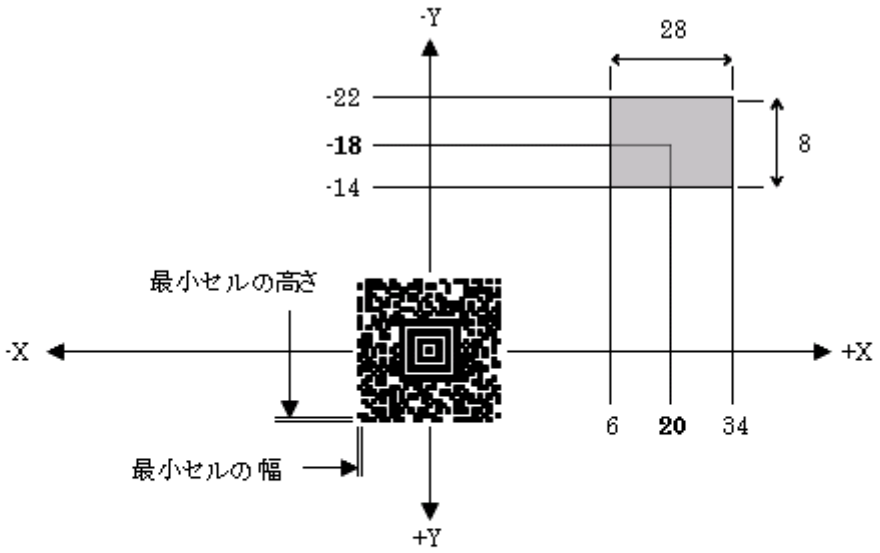
dwHeight＝(切り出すイメージの高さ)÷(最小セルの幅)

3. nOffsetX と nOffsetY の算出

シンボルコードの中心から切り出すイメージの中心までの距離を測定し、nOffsetX と nOffsetY に指定する値を算出します。

nOffsetX＝(シンボルコードの中心からイメージの中心までの横方向距離)÷(最小セルの幅)

nOffsetY＝(シンボルコードの中心からイメージの中心までの縦方向距離)÷(最小セルの幅)



最小セルの幅＝0.5mm、最小セルの高さ＝0.5mmの場合
dwAspectRatio = (0.5 / 0.5) = 1
dwWidth = (28 / 0.5) = 56
dwHeight = (8 / 0.5) = 16
nOffsetX = (20 / 0.5) = 40
nOffsetY = (-18 / 0.5) = -36

※(dwWidth×nResolution)が偶数になるように指定してください。

イメージ取得時の注意点

この関数で取得するのは、IMGWaitForDecode 関数または、IMGWaitForDecodeRaw 関数でデコードを実行したときに取得したイメージ内のシンボルコードの中心を原点とした、相対座標にあるイメージです。指定した部分のイメージがデコード時に取得したイメージの範囲から外れている場合、この関数は IMG_ERR_BADREGION を返します。その場合、イメージャとシンボルとの距離等を調節し、再び IMGWaitForDecode 関数または、IMGWaitForDecodeRaw 関数から実行してください。

3.19 IMGSetAusPost

オーストラリアの郵便番号コードのオプションを設定します。

```
[C++]
int IMGSetAusPost(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetAusPost( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetAusPost(
  Boolean bEnabled
);
```

解説

オーストラリアの郵便番号コードのオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetAusPost 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
オーストラリアの郵便番号コード読み取りの有効/無効を、以下の値で指定します。
TRUE　：有効
FALSE　：無効(デフォルト)

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.20 IMGGetAusPost

オーストラリアの郵便番号コードのオプションを取得します。

```
[C++]
int IMGGetAusPost(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetAusPost( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetAusPost(
  ref Boolean pEnabled
);
```

解説

オーストラリアの郵便番号コードのオプションを取得します。

パラメータ

pEnabled

オーストラリア郵便番号コードの読み取り有効/無効を取得します。取得する値については IMGSetAusPost 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.21 IMGSetAztec

Aztec コードのデコードオプションを設定します。

```
[C++]
int IMGSetAztec(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetAztec( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetAztec(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

Aztec コードのオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetAztec 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Aztec コード読み取りの有効/無効を今の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
読み取るシンボルの最小桁数を指定します。。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
読み取るシンボルの最大桁数を指定します。
最大許容値 : 3750
デフォルト値 : 3750

戻り値

IMG_SUCCESS : 正常終了

IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.22 IMGGetAztec

Aztec コードのデコードオプションを取得します。

```
[C++]
int IMGGetAztec(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetAztec( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetAztec(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

Aztec コードのオプションを取得します。

パラメータ

pEnabled

Aztec コードの読み取り有効/無効を取得します。取得する値は IMGSetAztec 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetAztec 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetAztec 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.23 IMGSetBPO

英国の郵便番号コードのデコードオプションを設定します。

```
[C++]
int IMGSetBPO(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetBPO( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetBPO(
  Boolean bEnabled
);
```

解説

英国の郵便番号コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetBPO 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
英国の郵便番号コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.24 IMGGetBPO

英国の郵便番号コードのデコードオプションを取得します。

```
[C++]
int IMGGetBPO(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetBPO( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetBPO(
  ref Boolean pEnabled
);
```

解説

英国の郵便番号コードのデコードオプションを取得します。

パラメータ

pEnabled

英国の郵便番号コードの読み取り有効/無効を取得します。取得する値は IMGSetBPO 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.25 IMGSetCanPost

カナダの郵便番号コードのデコードオプションを設定します。

```
[C++]
int IMGSetCanPost(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetCanPost( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetCanPost(
  Boolean bEnabled
);
```

解説

カナダの郵便番号コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCanPost 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
カナダの郵便番号コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.26 IMGGetCanPost

カナダの郵便番号コードのデコードオプションを取得します。

```
[C++]
int IMGGetCanPost(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetCanPost( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetCanPost(
  ref Boolean pEnabled
);
```

解説

カナダの郵便番号コードのデコードオプションを取得します。

パラメータ

pEnabled

カナダの郵便番号コードの読み取り有効/無効を取得します。取得する値は IMGSetCanPost 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.27 IMGSetCodabar

Codabar コードのデコードオプションを設定します。

```
[C++]
int IMGSetCodabar(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bSSXmit,
  BOOL bCheckCharOn,
  BOOL bXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetCodabar( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bSSXmit As Boolean, _
  ByVal bCheckCharOn As Boolean, _
  ByVal bXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetCodabar(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bSSXmit,
  Boolean bCheckCharOn,
  Boolean bXmitCheckChar
);
```

解説

Codabar コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCodabar 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Codabar コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
読み取るシンボルの最小桁数を指定します。
最小許容値 : 2
デフォルト値 : 2

dwMaxLength

読み取るシンボルの最大桁数を指定します。

最大許容値　　: 60
デフォルト値　　: 60

bSSXmit

スタート・ストップコード出力の有効/無効を指定します。

TRUE　　: スタート・ストップコードを出力します。
FALSE　　: スタート・ストップコードを出力しません。(デフォルト)

bCheckCharOn

チェックキャラクタ有効/無効を取得します。

TRUE　　: チェックキャラクタを持つ Codabar コードのみをデコードします。
FALSE　　: チェックキャラクタの有無に関わらず、コードをデコードします。(デフォルト)

bXmitCheckChar

チェックキャラクタ出力の有無を取得します。

TRUE　　: チェックキャラクタを出力します。
FALSE　　: チェックキャラクタを出力しません。(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

bCheckCharOn が TRUE にセットされた場合にのみ、bXmitCheckChar が有効となります。
bCheckCharOn が FALSE にセットされた場合、bXmitCheckChar は無視されます。

3.28 IMGGetCodabar

Codabar コードのデコードオプションを取得します。

```
[C++]
int IMGGetCodabar(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pSSXmit,
  LPBOOL pCheckCharOn,
  LPBOOL pXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetCodabar( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pSSXmit As Boolean, _
  ByRef pCheckCharOn As Boolean, _
  ByRef pXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetCodabar(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pSSXmit,
  ref Boolean pCheckCharOn,
  ref Boolean pXmitCheckChar
);
```

解説

Codabar コードのデコードオプションを取得します。

パラメータ

pEnabled
Codabar コードの読み取り有効/無効を取得します。取得する値は IMGSetCodabar 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetCodabar 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetCodabar 関数を参照してください。

pSSXmit

スタート・ストップコード出力の有効/無効を取得します。取得する値は IMGSetCodabar 関数を参照してください。

pCheckCharOn

チェックキャラクタ有効/無効を取得します。取得する値は IMGSetCodabar 関数を参照してください。

pXmitCheckChar

チェックキャラクタ出力の有無を取得します。取得する値は IMGSetCodabar 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.29 IMGSetCodablock

CodablockF コードのデコードオプションを設定します。

```
[C++]
int IMGSetCodablock(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetCodablock( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetCodablock(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

CodablockF コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCodablock 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
CodabarF コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 0(制限なし)
デフォルト値 : 0

dwMaxLength
最大桁数を指定します。
最大許容値 : 2048
デフォルト値 : 2048

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

Codablock A コードには対応していません。

3.30 IMGGetCodablock

CodablockF コードのデコードオプションを取得します。

```
[C++]
int IMGGetCodablock(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetCodablock( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetCodablock(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

CodablockF コードのデコードオプションを取得します。

パラメータ

pEnabled

CodablockF コードの読み取り有効/無効を取得します。取得する値は IMGSetCodablock 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetCodablock 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetCodablock 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

Codablock A コードには対応していません。

3.31 IMGSetCode11

Code11 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode11(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bTwoCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode11( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bTwoCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode11(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bTwoCheckChar
);
```

解説

Code11 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCode11 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Code11 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 4

dwMaxLength
最大桁数を指定します。
最大許容値 : 80
デフォルト値 : 80

bTwoCheckChar

2 桁チェックキャラクタの使用の有無を以下の値で指定します。

TRUE　：2 桁のチェックキャラクタを持つ Code11 バーコードのみをデコードします(デフォルト)。

FALSE　：1 桁のチェックキャラクタを持つ Code11 バーコードのみをデコードします。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.32 IMGGetCode11

Code11 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode11(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pTwoCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode11( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pTwoCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode11(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pTwoCheckChar
);
```

解説

Code11 コードのデコードオプションを取得します。

パラメータ

pEnabled
Code11 コードの読み取り有効/無効を取得します。取得する値は IMGSetCode11 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetCode11 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetCode11 関数を参照してください。

pTwoCheckChar
2 桁チェックキャラクタの使用の有無を取得します。取得する値は IMGSetCode11 関数を参照してください。

戻り値

IMG_SUCCESS　　：正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない（DT-X8-4x シリーズのみ）
DeviceEmulator では発生しません
IMG_ERR_PARAMETER : パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

3.33 IMGSetCode128

Code128 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode128(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode128( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode128(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

Code128 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCode128 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Code128 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 0(制限なし)
デフォルト値 : 0

dwMaxLength
最大桁数を指定します。
最大許容値 : 80
デフォルト値 : 80

戻り値

IMG_SUCCESS : 正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない（DT-X8-4x シリーズのみ）
DeviceEmulator では発生しません
IMG_ERR_PARAMETER : パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

3.34 IMGGetCode128

Code128 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode128(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode128( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode128(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

Code128 コードのデコードオプションを取得します。

パラメータ

pEnabled

Code128 コードの読み取り有効/無効を取得します。取得する値は IMGSetCode128 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetCode128 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetCode128 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.35 IMGSetCode32

Code32 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode32(
  BOOL  bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode32( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode32(
  Boolean bEnabled
)
```

解説

Code32 コードのデコードオプションを設定します。
Code32 コードの読み取りを有効にするためには、Code39 コードの読み取りも有効に設定してください。
DeviceEmulator では、設定値を内部変数として格納するため、何も動作しませんが、IMGGetCode32 関数を実行することにより、設定値を確認することができます。

パラメータ

bEnabled
Code32 コード読み取りの有効/無効を指定します。
TRUE　：有効
FALSE　：無効(デフォルト)

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.36 IMGGetCode32

Code32 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode32(
  LPBOOL  pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode32( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode32(
  ref Boolean pEnabled
)
```

解説

Code32 コードのデコードオプションを取得します。

パラメータ

pEnabled

Code32 コード読み取りの有効/無効を取得します。取得する値は IMGSetCode32 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.37 IMGSetCode39

Code39 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode39(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bSSXmit,
  BOOL bCheckCharOn,
  BOOL bXmitCheckChar,
  BOOL bFullAscii
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode39( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bSSXmit As Boolean, _
  ByVal bCheckCharOn As Boolean, _
  ByVal bXmitCheckChar As Boolean, _
  ByVal bFullAscii As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode39(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bSSXmit,
  Boolean bCheckCharOn,
  Boolean bXmitCheckChar,
  Boolean bFullAscii
);
```

解説

Code39 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCode39 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Code39 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値　　: 0
デフォルト値　　: 2

dwMaxLength
最大桁数を指定します。
最大許容値　　: 48
デフォルト値　　: 48

bSSXmit
スタート・ストップコードの出力の有無を、以下の値で指定します。
TRUE　　: スタート・ストップコードを出力します。
FALSE　　: スタート・ストップコードを出力しません(デフォルト)。

bCheckCharOn
チェックキャラクタの有効/無効を、以下の値で指定します。
TRUE　　: チェックキャラクタを持つ Code39 コードのみをデコードします。
FALSE　　: チェックキャラクタの有無に関わらず、コードをデコードします(デフォルト)。

bXmitCheckChar
チェックキャラクタ出力の有無を、以下の値で指定します。
TRUE　　: チェックキャラクタを出力します。
FALSE　　: チェックキャラクタを出力しません(デフォルト)。

bFullAscii
Full ASCII 変換の有無を、以下の値で指定します。
TRUE　　: Full ASCII 変換を行います。
FALSE　　: Full ASCII 変換を行いません(デフォルト)。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

bCheckCharOn が TRUE にセットされた場合にのみ、bXmitCheckChar は有効となります。
bCheckCharOn が FALSE にセットされた場合、bXmitCheckChar は無視されます。

3.38 IMGGetCode39

Code39 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode39(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pSSXmit,
  LPBOOL pCheckCharOn,
  LPBOOL pXmitCheckChar,
  LPBOOL pFullAscii
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode39( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pSSXmit As Boolean, _
  ByRef pCheckCharOn As Boolean, _
  ByRef pXmitCheckChar As Boolean, _
  ByRef pFullAscii As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode39(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pSSXmit,
  ref Boolean pCheckCharOn,
  ref Boolean pXmitCheckChar,
  ref Boolean pFullAscii
);
```

解説

Code39 コードのデコードオプションを取得します。

パラメータ

pEnabled
Code39 コードの読み取り有効/無効を取得します。取得する値は IMGSetCode39 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetCode39 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetCode39 関数を参照してください。

pSSXmit
スタート・ストップコードの出力の有無を取得します。取得する値は IMGSetCode39 関数を参照してください。

pCheckCharOn
チェックキャラクタの有効/無効を取得します。取得する値は IMGSetCode39 関数を参照してください。

pXmitCheckChar
チェックキャラクタ出力の有無を取得します。取得する値は IMGSetCode39 関数を参照してください。

pFullAscii
Full ASCII 変換の有無を取得します。取得する値は IMGSetCode39 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.39 IMGSetCode49

Code49 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode49(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode49( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode49(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

Code49 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCode49 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Code49 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 81
デフォルト値 : 81

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

Code49 のデータ連結モード（先頭が=M で始まる Code49 コードの読み取り）を使用する場合は注意が必要です。詳細は、IMGWaitForDecode 関数を参照してください。

3.40 IMGGetCode49

Code49 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode49(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode49( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode49(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

Code49 コードのデコードオプションを取得します。

パラメータ

pEnabled
Code49 コードの読み取り有効/無効を取得します。取得する値は IMGSetCode49 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetCode49 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetCode49 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.41 IMGSetCode93

Code93 コードのデコードオプションを設定します。

```
[C++]
int IMGSetCode93(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetCode93( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetCode93(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

Code93 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetCode93 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Code93 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
読み取るシンボルの最小桁数。
最小許容値 : 0
デフォルト値 : 0

dwMaxLength
読み取るシンボルの最大桁数。
最大許容値 : 80
デフォルト値 : 80

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

Code93 のデータ連結モード（先頭がスペースで始まる Code93 コードの読み取り）を使用する場合は注意が必要です。詳細は、IMGWaitForDecode 関数を参照してください。

3.42 IMGGetCode93

Code93 コードのデコードオプションを取得します。

```
[C++]
int IMGGetCode93(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetCode93( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetCode93(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

Code93 コードのデコードオプションを取得します。

パラメータ

pEnabled

Code93 コードの読み取り有効/無効を取得します。取得する値は IMGSetCode93 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetCode93 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetCode93 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.43 IMGSetComposite

Composite コードのデコードオプションを設定します。

```
[C++]
int IMGSetComposite(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bCompositeOnUpcEan
)
```

```
[Visual Basic]
Public Shared Function IMGSetComposite( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bCompositeOnUpcEan As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetComposite(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bCompositeOnUpcEan
);
```

解説

Composite コードのデコードオプションを設定します。
対応しているシンボルは以下の 4 つです。
・EAN8 Composite コード
・EAN13 Composite コード
・Code128 Composite コード
・RSS(RSS14, RSS Limited, RSS Expanded) Composite コード
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetComposite 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Composite コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength

最大桁数を指定します。下段・上段両方のシンボルの合計桁数を指定します。

最大許容値　　：2435
デフォルト値　　：2435

bCompositeOnUpcEan

UPC/EAN コードの読み取り有効/無効を、以下の値で指定します。

TRUE　　：UPC/EAN Composite コードの読み取りが有効。
FALSE　　：UPC/EAN Composite コードの読み取りは無効。(デフォルト)

戻り値

IMG_SUCCESS　　：正常終了
IMG_ERR_PARAMETER　　：パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT　　：未サポートエラー

補足

・bEnabled に TRUE（読み取り有効）を指定する場合は、対になる1次元シンボル(Composite コードの下段に配置されているシンボル)も読み取り有効にしてください。
・EAN Composite でないコードの読み取りを行う場合には、bCompositeOnUpcEan に FALSE を指定することを推奨します。TRUE に指定した状態では Composite でない EAN コードの読み取りが遅くなります。
・Composite コードの読み取りを行う場合には以下の表に従い、各関数ごとにパラメータを指定してください。

読み取るシンボル	設定する関数	指定パラメータ		
		有効指定	bComposite - OnUpcEan	桁数指定
EAN8 Composite	IMGSetComposite	TRUE	TRUE	2D 部の桁数
	IMGSetMicroPDF	TRUE	－	デフォルト
	IMGSetEAN8	TRUE	－	－
EAN13 Composite	IMGSetComposite	TRUE	TRUE	2D 部の桁数
	IMGSetMicroPDF	TRUE	－	デフォルト
	IMGSetEAN13	TRUE	－	－
Code128 Composite (CCA/CCB)	IMGSetComposite	TRUE	FALSE	2D 部の桁数
	IMGSetMicroPDF	TRUE	－	デフォルト
	IMGSetCode128	TRUE	－	1D 部の桁数
Code128 Composite(CCC)	IMGSetComposite	TRUE	FALSE	2D 部の桁数
	IMGSetPDF417	TRUE	－	デフォルト
	IMGSetCode128	TRUE	－	1D 部の桁数
RSS14 Composite RSS Limited Composite RSS Expanded Composite	IMGSetComposite	TRUE	FALSE	2D 部の桁数
	IMGSetMicroPDF	TRUE	－	デフォルト
	IMGSetRSS	TRUE	－	1D 部の桁数

3.44 IMGGetComposite

Composite コードのデコードオプションを取得します。

```
[C++]
int IMGGetComposite(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pCompositeOnUpcEan
)
```

```
[Visual Basic]
Public Shared Function IMGGetComposite( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pCompositeOnUpcEan As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetComposite(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pCompositeOnUpcEan
);
```

解説

Composite コードのデコードオプションを取得します。

パラメータ

pEnabled
Compositeコードの読み取り有効/無効を取得します。取得する値はIMGSetComposite関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetComposite 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetComposite 関数を参照してください。

pCompositeOnUpcEan
UPC/EAN コードの読み取り有効/無効を取得します。取得する値は IMGSetComposite 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.45 IMGSetDataMatrix

DataMatrix コードのデコードオプションを設定します。

```
[C++]
int IMGSetDataMatrix(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetDataMatrix( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetDataMatrix(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

DataMatrix コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetDataMatrix 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
DataMatrix コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 1500
デフォルト値 : 1500

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.46 IMGGetDataMatrix

DataMatrix コードのデコードオプションを取得します。

```
[C++]
int IMGGetDataMatrix(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetDataMatrix( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetDataMatrix(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

DataMatrix コードのデコードオプションを取得します。

パラメータ

pEnabled

DataMatrix コードの読み取り有効/無効を取得します。取得する値は IMGSetDataMatrix 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetDataMatrix 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetDataMatrix 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.47 IMGSetDutchPost

オランダの郵便番号コードのデコードオプションを設定します。

```
[C++]
int IMGSetDutchPost(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetDutchPost( _
  ByVal pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetDutchPost(
  Boolean pEnabled
);
```

解説

オランダの郵便番号コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetDutchPost 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
オランダの郵便番号コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.48 IMGGetDutchPost

オランダの郵便番号コードのデコードオプションを取得します。

```
[C++]
int IMGGetDutchPost(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetDutchPost( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetDutchPost(
  ref Boolean pEnabled
);
```

解説

オランダの郵便番号コードのデコードオプションを取得します。

パラメータ

pEnabled

オランダの郵便番号コードの読み取り有効/無効を取得します。取得する値はIMGSetDutchPost関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulatorでは発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.49 IMGSetEAN13

EAN13/JAN13 コードのデコードオプションを設定します。

```
[C++]
int IMGSetEAN13(
  BOOL bEnabled,
  BOOL bXmitCheckChar,
  BOOL bAddendaReq,
  BOOL bAddendaSeparator,
  BOOL bAddenda2Digit,
  BOOL bAddenda5Digit
)
```

```
[Visual Basic]
Public Shared Function IMGSetEAN13( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean, _
  ByVal bAddendaReq As Boolean, _
  ByVal bAddendaSeparator As Boolean, _
  ByVal bAddenda2Digit As Boolean, _
  ByVal bAddenda5Digit As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetEAN13(
  Boolean bEnabled,
  Boolean bXmitCheckChar,
  Boolean bAddendaReq,
  Boolean bAddendaSeparator,
  Boolean bAddenda2Digit,
  Boolean bAddenda5Digit
);
```

解説

EAN13/JAN13 コードのデコードオプションをセットします。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetEAN13 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
EAN13/JAN13 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

bXmitCheckChar
チェックキャラクタ出力の有無を、以下の値で指定します。
TRUE : チェックキャラクタを出力します。
FALSE : チェックキャラクタを出力しません(デフォルト)。

bAddendaReq

アドオンをもつ EAN13/JAN13 コードのみをデコードするかすべての EAN13/JAN13 コードをデコードするかを、以下の値で指定します。

TRUE　：2 桁あるいは 5 桁のアドオンをもつ EAN13/JAN13 シンボルのみをデコードします。
FALSE　：すべての有効な EAN13/JAN13 シンボルをデコードします(デフォルト)。

bAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを、以下の値で指定します。

TRUE　：バーコードのデータとアドオンをスペースで区切って出力します。
FALSE　：バーコードのデータとアドオンをスペースで区切らずに出力します(デフォルト)。

bAddenda2Digit

2 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：2 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE　：アドオンを無視します(デフォルト)。

bAddenda5Digit

5 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：5 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE　：アドオンを無視します(デフォルト)。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

補足

EAN13 コードを読み取り有効にすると、UPC-A コードの読み取りも可能になります。
UPC-A コードを読み取り有効にすると、先頭が"0"で始まる EAN13 コードも UPC-A コードと認識し、読み取りが可能になります。

3.50 IMGGetEAN13

EAN13/JAN13 コードのデコードオプションを取得します。

```
[C++]
int IMGGetEAN13(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar,
  LPBOOL pAddendaReq,
  LPBOOL pAddendaSeparator,
  LPBOOL pAddenda2Digit,
  LPBOOL pAddenda5Digit
)
```

```
[Visual Basic]
Public Shared Function IMGGetEAN13( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean, _
  ByRef pAddendaReq As Boolean, _
  ByRef pAddendaSeparator As Boolean, _
  ByRef pAddenda2Digit As Boolean, _
  ByRef pAddenda5Digit As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetEAN13(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar,
  ref Boolean pAddendaReq,
  ref Boolean pAddendaSeparator,
  ref Boolean pAddenda2Digit,
  ref Boolean pAddenda5Digit
);
```

解説

EAN13/JAN13 コードのデコードオプションを取得します。

パラメータ

pEnabled

EAN13/JAN13 コードの読み取り有効/無効を取得します。取得する値は IMGSetEAN13 関数を参照してください。

pXmitCheckChar

チェックキャラクタ出力の有無を取得します。取得する値は IMGSetEAN13 関数を参照してください。

pAddendaReq

アドオンをもつ EAN13/JAN13 コードのみをデコードするかすべての EAN13/JAN13 コードをデコードするかを取得します。取得する値は IMGSetEAN13 関数を参照してください。

pAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを取得します。取得する値は IMGSetEAN13 関数を参照してください。

pAddenda2Digit

2 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetEAN13 関数を参照してください。

pAddenda5Digit

5 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetEAN13 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.51 IMGSetEAN8

EAN8/JAN8 コードのデコードオプションを設定します。

```
[C++]
int IMGSetEAN8(
  BOOL bEnabled,
  BOOL bXmitCheckChar,
  BOOL bAddendaReq,
  BOOL bAddendaSeparator,
  BOOL bAddenda2Digit,
  BOOL bAddenda5Digit
)
```

```
[Visual Basic]
Public Shared Function IMGSetEAN8( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean, _
  ByVal bAddendaReq As Boolean, _
  ByVal bAddendaSeparator As Boolean, _
  ByVal bAddenda2Digit As Boolean, _
  ByVal bAddenda5Digit As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetEAN8(
  Boolean bEnabled,
  Boolean bXmitCheckChar,
  Boolean bAddendaReq,
  Boolean bAddendaSeparator,
  Boolean bAddenda2Digit,
  Boolean bAddenda5Digit
);
```

解説

EAN8/JAN8 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetEAN8 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
EAN8/JAN8 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

bXmitCheckChar
チェックキャラクタ出力の有無を、以下の値で指定します。
TRUE : チェックキャラクタを出力します。
FALSE : チェックキャラクタを出力しません(デフォルト)。

bAddendaReq

アドオンをもつ EAN8/JAN8 シンボルのみをデコードするか、すべての有効な EAN8/JAN8 シンボルをデコードするかを、以下の値で指定します。

TRUE : 2 桁あるいは 5 桁のアドオンをもつ EAN8/JAN8 コードのみをデコードします。
FALSE : イメージャはすべての有効な EAN8/JAN8 コードをデコードします(デフォルト)。

bAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを、以下の値で指定します。

TRUE : バーコードのデータとアドオンをスペースで区切って出力します。
FALSE : バーコードのデータとアドオンをスペースで区切らずに出力します(デフォルト)。

bAddenda2Digit

2 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE : 2 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE : アドオンを無視します(デフォルト)。

bAddenda5Digit

5 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE : 5 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE : アドオンを無視します(デフォルト)。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.52 IMGGetEAN8

EAN8/JAN8 のコードのデコードオプションを取得します。

```
[C++]
int IMGGetEAN8(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar,
  LPBOOL pAddendaReq,
  LPBOOL pAddendaSeparator,
  LPBOOL pAddenda2Digit,
  LPBOOL pAddenda5Digit
)
```

```
[Visual Basic]
Public Shared Function IMGGetEAN8( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean, _
  ByRef pAddendaReq As Boolean, _
  ByRef pAddendaSeparator As Boolean, _
  ByRef pAddenda2Digit As Boolean, _
  ByRef pAddenda5Digit As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetEAN8(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar,
  ref Boolean pAddendaReq,
  ref Boolean pAddendaSeparator,
  ref Boolean pAddenda2Digit,
  ref Boolean pAddenda5Digit
);
```

解説

EAN8/JAN8 のコードのデコードオプションを取得します。

パラメータ

pEnabled

EAN8/JAN8 コードの読み取り有効/無効を取得します。取得する値は IMGSetEAN8 関数を参照してください。

pXmitCheckChar

チェックキャラクタ出力の有無を取得します。取得する値は IMGSetEAN8 関数を参照してください。

pAddendaReq

アドオンをもつ EAN8/JAN8 シンボルのみをデコードするか、すべての有効な EAN8/JAN8 シンボルをデコードするかを取得します。取得する値は IMGSetEAN8 関数を参照してください。

pAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを取得します。取得する値は IMGSetEAN8 関数を参照してください。

pAddenda2Digit

2 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetEAN8 関数を参照してください。

pAddenda5Digit

5 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetEAN8 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.53 IMGSetHX

Chinese Sensible (Han Xin)コードのオプションを設定します。

```
[C++]
int IMGSetHX(
  BOOL  bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetHX( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetHX(
  Boolean bEnabled,
  Int32   dwMinLength,
  Int32   dwMaxLength
)
```

解説

Chinese Sensible コードのデコードオプションを設定します。
DeviceEmulator では、設定値を内部変数として格納するため、何も動作しませんが、IMGGetHX 関数を実行することにより、設定値を確認することができます。

パラメータ

bEnabled
Chinese Sensible コード読み取りの有効/無効を指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 6000
デフォルト値 : 6000

戻り値

IMG_SUCCESS : 正常終了

IMG_ERR_PARAMETER　：パラメータエラー
　　Device Emulator では発生しません
FUNCTION_UNSUPPORT　：未サポートエラー

3.54 IMGGetHX

Chinese Sensible (Han Xin)コードのデコードオプションを取得します。

```
[C++]
int IMGGetHX(
  LPBOOL  pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetHX( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetHX(
  ref Boolean pEnabled,
  ref Int32   pMinLength,
  ref Int32   pMaxLength
)
```

解説

Chinese Sensible コードのデコードオプションを取得します。

パラメータ

pEnabled

Chinese Sensible コードの有効/無効を取得します。取得する値は IMGSetHX 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetHX 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetHX 関数を参照してください。

戻り値

以下の値を返します。

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.55 IMGSetIATA

IATA 2 of 5 コードのデコードオプションを設定します。

```
[C++]
int IMGSetIATA(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetIATA( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetIATA(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

IATA 2 of 5 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetIATA 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
IATA 2 of 5 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 4
デフォルト値 : 4

dwMaxLength
最大桁数を指定します。
最大許容値 : 80
デフォルト値 : 80

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.56 IMGGetIATA

IATA 2 of 5 のコードのデコードオプションを取得します。

```
[C++]
int IMGGetIATA(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetIATA( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetIATA(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

IATA 2 of 5 のコードのデコードオプションを取得します。

パラメータ

pEnabled
IATA 2 of 5 コードの有効/無効を取得します。取得する値は IMGSetIATA 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetIATA 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetIATA 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.57 IMGSetITF

ITF(Interleaved 2 of 5)コードのデコードオプションを設定します。

```
[C++]
int IMGSetITF(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bCheckCharOn,
  BOOL bXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetITF( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bCheckCharOn As Boolean, _
  ByVal bXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetITF(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bCheckCharOn,
  Boolean bXmitCheckChar
);
```

解説

ITF(Interleaved 2 of 5)コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetITF 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
ITF(Interleaved 2 of 5)コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

nMinLength
最小桁数を指定します。
最小許容値 : 4
デフォルト値 : 4

nMaxLength
最大桁数を指定します。

最大許容値　　：80
デフォルト値　　：80

bCheckCharOn

チェックキャラクタの有効/無効を以下の値で指定します。

TRUE　　：チェックキャラクタをもつ ITF(Interleaved 2 of 5)コードのみをデコードします。
FALSE　　：チェックキャラクタの有無に関わらず、コードをデコードします(デフォルト)。

bXmitCheckChar

チェックキャラクタ出力の有無を以下の値で指定します。

TRUE　　：チェックキャラクタを出力します。
FALSE　　：チェックキャラクタを出力しません(デフォルト)。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

補足

bCheckCharOn が TRUE にセットされた場合にのみ、bXmitCheckChar は有効となります。
bCheckCharOn が FALSE にセットされた場合、bXmitCheckChar は無視されます。

3.58 IMGGetITF

ITF(Interleaved 2 of 5)コードのデコードオプションを取得します。

```
[C++]
int IMGGetITF(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pCheckCharOn,
  LPBOOL pXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetITF( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pCheckCharOn As Boolean, _
  ByRef pXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetITF(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pCheckCharOn,
  ref Boolean pXmitCheckChar
);
```

解説

ITF(Interleaved 2 of 5)コードのデコードオプションを取得します。

パラメータ

pEnabled

ITF(Interleaved 2 of 5)コードの有効/無効を取得します。取得する値はIMGSetITF関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetITF 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetITF 関数を参照してください。

pCheckCharOn

チェックキャラクタの有効/無効を取得します。取得する値は IMGSetITF 関数を参照してください。

ppXmitCheckChar

チェックキャラクタ出力を取得します。取得する値は IMGSetITF 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.59 IMGSetISBT

ISBT のコードのデコードオプションを設定します。

```
[C++]
int IMGSetISBT(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetISBT( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetISBT(
  Boolean bEnabled
);
```

解説

ISBT のコードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetISBT 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
ISBT コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.60 IMGGetISBT

ISBT のコードのデコードオプションを取得します。

```
[C++]
int IMGGetISBT(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetISBT( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetISBT(
  ref Boolean pEnabled
);
```

解説

ISBT のコードのデコードオプションを取得します。

パラメータ

pEnabled

ISBT コードの有効/無効を取得します。取得する値は IMGSetISBT 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.61 IMGSetMaxicode

Maxicode コードのデコードオプションを設定します。

```
[C++]
int IMGSetMaxicode(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bCarrierMsgOnly
)
```

```
[Visual Basic]
Public Shared Function IMGSetMaxicode( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bCarrierMsgOnly As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetMaxicode(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bCarrierMsgOnly
);
```

解説

Maxicode コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetMaxicode 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Maxicode コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 150
デフォルト値 : 150

pCarrierMsgOnly
FALSE を指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.62 IMGGetMaxicode

Maxicode コードのデコードオプションを取得します。

```
[C++]
int IMGGetMaxicode(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pCarrierMsgOnly
)
```

```
[Visual Basic]
Public Shared Function IMGGetMaxicode( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pCarrierMsgOnly As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetMaxicode(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pCarrierMsgOnly
);
```

解説

Maxicode コードのデコードオプションを取得します。

パラメータ

pEnabled
Maxicode コードの有効/無効を取得します。取得する値は IMGSetMaxicode 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetMaxicode 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetMaxicode 関数を参照してください。

pCarrierMsgOnly
キャリアメッセージの出力を取得します。取得する値は IMGSetMaxicode 関数を参照してください。

戻り値

IMG_SUCCESS　　：正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない（DT-X8-4x シリーズのみ）
DeviceEmulator では発生しません
IMG_ERR_PARAMETER : パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

3.63 IMGSetMicroPDF

MicroPDF コードのデコードオプションを設定します。

```
[C++]
int IMGSetMicroPDF(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetMicroPDF( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetMicroPDF(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

MicroPDF コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetMicroPDF 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
MicroPDF コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 2750
デフォルト値 : 2750

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.64 IMGGetMicroPDF

MicroPDF のコードのデコードオプションを取得します。

```
[C++]
int IMGGetMicroPDF(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetMicroPDF( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetMicroPDF(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

MicroPDF のコードのデコードオプションを取得します。

パラメータ

pEnabled

MicroPDF コードの有効/無効を取得します。取得する値は IMGSetMicroPDF 関数を参照してください。

pMinLength

最小桁数を取得します。取得する値は IMGSetMicroPDF 関数を参照してください。

pMaxLength

最大桁数を取得します。取得する値は IMGSetMicroPDF 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.65 IMGSetMSI

MSI コードのデコードオプションを設定します。

```
[C++]
int IMGSetMSI(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength,
  BOOL bXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetMSI( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32, _
  ByVal bXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetMSI(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength,
  Boolean bXmitCheckChar
);
```

解説

MSI コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetMSI 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
MSI コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 4
デフォルト値 : 4

dwMaxLength
最大桁数を指定します。
最大許容値 : 48
デフォルト値 : 48

bXmitCheckChar

チェックキャラクタ出力の有無を、以下の値で指定します。

TRUE　：チェックキャラクタを出力します。
FALSE　：チェックキャラクタを出力しません(デフォルト)。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.66 IMGGetMSI

MSI コードのデコードオプションを取得します。

```
[C++]
int IMGGetMSI(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength,
  LPBOOL pXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetMSI( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32, _
  ByRef pXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetMSI(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength,
  ref Boolean pXmitCheckChar
);
```

解説

MSI コードのデコードオプションを取得します。

パラメータ

pEnabled
MSI コードの有効/無効を取得します。取得する値は IMGSetMSI 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetMSI 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetMSI 関数を参照してください。

pXmitCheckChar
チェックキャラクタ出力を取得します。取得する値は IMGSetMSI 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません

IMG_ERR_PARAMETER　　：パラメータエラー
　　Device Emulator では発生しません
FUNCTION_UNSUPPORT　　：未サポートエラー

3.67 IMGSetOCR

光学文字読取(Optical Character Recognition)のデコードオプションを設定します。

```
[C++]
int IMGSetOCR(
  OCRMode_t nFont,
  PTCHAR pTemplate,
  PTCHAR pGroupG,
  PTCHAR pGroupH,
  PTCHAR pCheckChar,
  OCRDirection_t nDirection
)
```

```
[Visual Basic]
Public Shared Function IMGSetOCR( _
  ByVal nFont As IMGLibNet.Def.OCRMode_t&, _
  ByVal pTemplate As Char(), _
  ByVal pGroupG As Char(), _
  ByVal pGroupH As Char(), _
  ByVal pCheckChar As Char(), _
  ByVal nDirection As IMGLibNet.Def.OCRDirection_t& _
) As Int32
```

```
[C#]
public static Int32 IMGSetOCR(
  IMGLibNet.Def.OCRMode_t& nFont,
  Char[] pTemplate,
  Char[] pGroupG,
  Char[] pGroupH,
  Char[] pCheckChar,
  IMGLibNet.Def.OCRDirection_t& nDirection
);
```

解説

光学文字読取(Optical Character Recognition)のデコードオプションを設定します。
OCR 文字認識はバーコード読み込みほど精度が高くありません。チェックキャラクタが使用されてない場合、誤読が生じる可能性があります。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetOCR 関数を実行することにより、設定内容を確認することができます。

パラメータ

nFont
デコード対象の OCR フォントを、以下の値で指定します。
OCR_DISABLED(デフォルト)
OCR_A
OCR_B
OCR_MONEY

pTemplate

OCR デコードのため、1 つ以上のテンプレートパターンを、以下の値で指定します。

A-Z	:アルファベットの大文字 1 文字
a	:アルファベットと数字
c	:チェックサム計算を行う位置の指定
d	:0 – 9 の数字
e	:全ての文字(英数字および記号)
g	:グループ G に指定される文字
h	:グループ H に指定される文字
l	:アルファベット文字
r	:列区切り位置
t	:複数テンプレート指定時のテンプレートとテンプレートの区切り位置

pGroupG

テンプレートストリングの中で、小文字「g」に代用することのできる、文字のリストを指定します。

pGroupH

テンプレートストリングの中で、小文字「h」に代用することのできる、文字のリストを指定します。

pCheckChar

シンボルコードをデコードするときのチェックサム計算の方式を設定します。
modulo 10 checksums を使う場合、"0123456789"とセットし、modulo 36 checksums を使う場合、"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"とセットします。

nDirection

文字がイメージに対してどの方向に向いているかを、以下の値で OCR デコーダに伝えます。

LeftToRight
TopToBottom
RightToLeft
BottomToTop(デフォルト)

デコーダはどの方向にもデコードするので、このパラメータを指定することによりデコード時間を短縮させることができます。数字"0, 6, 8, 9"のみを含む番号についてのデコードの信頼性を向上させることにもなります。オペレーターが始めから終わりまでシンボルコードを読む間、方向は固定されます。

戻り値

IMG_SUCCESS	:正常終了
IMG_ERR_PARAMETER	:パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	:未サポートエラー

3.68 IMGGetOCR

光学文字読取のデコードオプション取得します。

```
[C++]
int IMGGetOCR(
  OCRMode_t *pFont,
  PTCHAR pTemplate,
  PTCHAR pGroupG,
  PTCHAR pGroupH,
  PTCHAR pCheckChar,
  OCRDirection_t *pDirection
)
```

```
[Visual Basic]
Public Shared Function IMGGetOCR( _
  ByRef pFont As IMGLibNet.Def.OCRMode_t&, _
  ByRef pTemplate As Char(), _
  ByRef pGroupG As Char(), _
  ByRef pGroupH As Char(), _
  ByRef pCheckChar As Char(), _
  ByRef pDirection As IMGLibNet.Def.OCRDirection_t& _
) As Int32
```

```
[C#]
public static Int32 IMGGetOCR(
  ref IMGLibNet.Def.OCRMode_t& pFont,
  ref Char[] pTemplate,
  ref Char[] pGroupG,
  ref Char[] pGroupH,
  ref Char[] ptcCheckChar,
  ref IMGLibNet.Def.OCRDirection_t& pDirection
);
```

解説

光学文字読取(Optical Character Recognition)のデコードオプションを取得します。

パラメータ

pFont
デコード対象の OCR フォントを取得します。取得する値は IMGSetOCR 関数を参照してください。

pTemplate
OCR デコードのためのテンプレートパターンを取得します。取得する値は IMGSetOCR 関数を参照してください。

pGroupG
テンプレートストリングの中で、小文字「g」に代用することのできる文字のリストを取得します。取得する値は IMGSetOCR 関数を参照してください。

pGroupH

テンプレートストリングの中で、小文字「h」に代用することのできる文字のリストを取得します。取得する値は IMGSetOCR 関数を参照してください。

pCheckChar

シンボルコードをデコードするときに行う現在のチェックサム計算の方式を取得します。取得する値は IMGSetOCR 関数を参照してください。。

pDirection

文字がイメージに対してどの方向に向いているかを OCR デコーダに認識させるためのデータを取得します。取得する値は IMGSetOCR 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.69 IMGSetPDF417

PDF417 コードのデコードオプションを設定します。

```
[C++]
int IMGSetPDF417(
  LPBOOL pEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetPDF417( _
  ByVal pEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetPDF417(
  Boolean pEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

PDF417 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetPDF417 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
PDF417 コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 2750
デフォルト値 : 2750

戻り値

IMG_SUCCESS : 正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない（DT-X8-4x シリーズのみ）
DeviceEmulator では発生しません
IMG_ERR_PARAMETER : パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

3.70 IMGGetPDF417

PDF417 コードのデコードオプションを取得します。

```
[C++]
int IMGGetPDF417(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetPDF417( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetPDF417(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

PDF417 コードのデコードオプションを取得します。

パラメータ

pEnabled
PDF417 コードの有効/無効を取得します。取得数値は IMGSetPDF417 関数を参照してください。

pMinLength
最小桁数を取得します。取得数値は IMGSetPDF417 関数を参照してください。

pMaxLength
最大桁数を取得します。取得数値は IMGSetPDF417 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.71 IMGSetPlanet

Planet コードのデコードオプションを設定します。

```
[C++]
int IMGSetPlanet(
  BOOL bEnabled,
  BOOL bXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetPlanet( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetPlanet(
  Boolean bEnabled,
  Boolean bXmitCheckChar
);
```

解説

Planet コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetPlanet 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Planet コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

bXmitCheckChar
チェックキャラクタを出力するかどうかを、以下の値で指定します。
TRUE : チェックキャラクタを出力します。
FALSE : チェックキャラクタを出力しません(デフォルト)。

戻り値

IMG_SUCCESS : 正常終了
FUNCTION_UNSUPPORT : 未サポートエラー

3.72 IMGGetPlanet

Planet コードのデコードオプションを取得します。

```
[C++]
int IMGGetPlanet(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetPlanet( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetPlanet(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar
);
```

解説

Planet コードのデコードオプションを取得します。

パラメータ

pEnabled

Planet コードの読み取り有効/無効を取得します。取得する値は IMGSetPlanet 関数を参照してください。

pXmitCheckChar

チェックキャラクタを出力するかどうかを取得します。取得する値は IMGSetPlanet 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.73 IMGSetPostnet

Postnet コードのデコードオプションを設定します。

```
[C++]
int IMGSetPostnet(
  BOOL bEnabled,
  BOOL bXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGSetPostnet( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetPostnet(
  Boolean bEnabled,
  Boolean bXmitCheckChar
);
```

解説

Postnet コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetPostnet 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
Postnet コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

bXmitCheckChar
チェックキャラクタを出力するかどうかを、以下の値で指定します。
TRUE : チェックキャラクタを出力します。
FALSE : チェックキャラクタを出力しません(デフォルト).。

戻り値

IMG_SUCCESS : 正常終了
FUNCTION_UNSUPPORT : 未サポートエラー

3.74 IMGGetPostnet

Postnet コードのデコードオプションを取得します。

```
[C++]
int IMGGetPostnet(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar
)
```

```
[Visual Basic]
Public Shared Function IMGGetPostnet( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetPostnet(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar
);
```

解説

Postnet コードのデコードオプションを取得します。

パラメータ

pEnabled

Postnet コードの有効/無効を取得します。取得する値は IMGSetPostnet 関数を参照してください。

pXmitCheckChar

チェックキャラクタを出力するかどうかを取得します。取得する値は IMGSetPostnet 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.75 IMGSetQR

QR コードのデコードオプションを取得します。

```
[C++]
int IMGSetQR(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetQR( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetQR(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

QR コードのデコードオプションを取得します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetQR 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
QR コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

dwMaxLength
最大桁数を指定します。
最大許容値 : 3500
デフォルト値 : 3500

戻り値

IMG_SUCCESS : 正常終了

IMG_ERR_NOTINITIALIZED : IMGInit が実行されていない（DT-X8-4x シリーズのみ）
DeviceEmulator では発生しません
IMG_ERR_PARAMETER : パラメータエラー
Device Emulator では発生しません
FUNCTION_UNSUPPORT : 未サポートエラー

補足

QR Code のデータ連結モード（連結識別子を含む QR Code の読み取り）を使用する場合は注意が必要です。IMGWaitForDecode 関数の項を参照してください。

3.76 IMGGetQR

QR コードのデコードオプションを取得します。

```
[C++]
int IMGGetQR(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetQR( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetQR(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

QR コードのデコードオプションを取得します。

パラメータ

pEnabled
QR コードの読み取り有効/無効を取得します。取得する値は IMGSetQR 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetQR 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetQR 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.77 IMGSetRSS

RSS コードのデコードオプションを設定します。

```
[C++]
int IMGSetRSS(
  BOOL bEnabled,
  DWORD dwMinLength,
  DWORD dwMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGSetRSS( _
  ByVal bEnabled As Boolean, _
  ByVal dwMinLength As Int32, _
  ByVal dwMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetRSS(
  Boolean bEnabled,
  Int32 dwMinLength,
  Int32 dwMaxLength
);
```

解説

RSS コード(RSS-14、RSS Limited、RSS Expanded)のデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetRSS 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
RSS コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

dwMinLength
最小桁数を指定します。
最小許容値 : 1
デフォルト値 : 1

RSS-14、RSS Limited を読み取る場合は、14 を指定してください。RSS Expanded を読み取る場合は、
読み取るシンボルの桁数に合わせて指定してください。

dwMaxLength
最大桁数を指定します。
最大許容値 : 80
デフォルト値 : 80

RSS-14、RSS Limited を読み取る場合は、14 を指定してください。
RSS Expanded を読み取る場合は、読み取るシンボルの桁数に合わせて指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.78 IMGGetRSS

RSS コードのデコードオプションを取得します。

```
[C++]
int IMGGetRSS(
  LPBOOL pEnabled,
  LPDWORD pMinLength,
  LPDWORD pMaxLength
)
```

```
[Visual Basic]
Public Shared Function IMGGetRSS( _
  ByRef pEnabled As Boolean, _
  ByRef pMinLength As Int32, _
  ByRef pMaxLength As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetRSS(
  ref Boolean pEnabled,
  ref Int32 pMinLength,
  ref Int32 pMaxLength
);
```

解説

RSS コード(RSS-14、RSS Limited、RSS Expanded)のデコードオプションを取得します。

パラメータ

pEnabled
RSS コードの有効/無効を取得します。取得する値は IMGSetRSS 関数を参照してください。

pMinLength
最小桁数を取得します。取得する値は IMGSetRSS 関数を参照してください。

pMaxLength
最大桁数を取得します。取得する値は IMGSetRSS 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.79 IMGSetTLC39

TLC39 コードのデコードオプションを設定します。

```
[C++]
int IMGSetTLC39(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetTLC39( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetTLC39(
  Boolean bEnabled
);
```

解説

TLC39 コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetTLC39 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
TLC39 コード読み取りの有効/無効を、以下の値で指定します。
TRUE ： 有効
FALSE ： 無効(デフォルト)

戻り値

IMG_SUCCESS ： 正常終了
IMG_ERR_PARAMETER ： パラメータエラー
FUNCTION_UNSUPPORT ： 未サポートエラー

補足

bEnabled に TRUE(読み取り有効)を指定する場合は、同時に MicroPDF コードと Code39 コードも読み取り有効にしてください。
TLC39 コードをデコードする場合の設定項目は以下のとおりです。

読み取るシンボル	設定する関数	指定パラメータ		
		有効指定	bComposite - OnUpcEan	桁数指定
TLC39	IMGSetTLC39	TRUE	－	－
	IMGSetMicroPDF	TRUE	－	2D 部の桁数
	IMGSetCode39	TRUE	－	－

3.80 IMGGetTLC39

TLC39 コードのデコードオプションを取得します。

```
[C++]
int IMGGetTLC39(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetTLC39( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetTLC39(
  ref Boolean pEnabled
);
```

解説

TLC39 コードのデコードオプションを取得します。

パラメータ

pEnabled

TLC39 コードの読み取り有効/無効を取得します。取得する値は IMGSetTLC39 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.81 IMGSetUPCA

UPC-A コードのデコードオプションを設定します。

```
[C++]
int IMGSetUPCA(
  BOOL bEnabled,
  BOOL bXmitCheckChar,
  BOOL bAddendaReq,
  BOOL bAddendaSeparator,
  BOOL bAddenda2Digit,
  BOOL bAddenda5Digit,
  BOOL bXmitNumSys
)
```

```
[Visual Basic]
Public Shared Function IMGSetUPCA( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean, _
  ByVal bAddendaReq As Boolean, _
  ByVal bAddendaSeparator As Boolean, _
  ByVal bAddenda2Digit As Boolean, _
  ByVal bAddenda5Digit As Boolean, _
  ByVal bXmitNumSys As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetUPCA(
  Boolean bEnabled,
  Boolean bXmitCheckChar,
  Boolean bAddendaReq,
  Boolean bAddendaSeparator,
  Boolean bAddenda2Digit,
  Boolean bAddenda5Digit,
  Boolean bXmitNumSys
);
```

解説

UPC-A コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetUPCA 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
UPC-A コード読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

bXmitCheckChar

チェックキャラクタ出力の有無を、以下の値で指定します。

TRUE　：チェックキャラクタを出力します。
FALSE　：チェックキャラクタを出力しません(デフォルト)。

bAddendaReq

アドオンをもつ UPC-A コードのみをデコードするかすべての UPC-A コードをデコードするかを、以下の値で指定します。

TRUE　：2 桁あるいは 5 桁のアドオンをもつ UPC-A コードのみをデコードします。
FALSE　：すべての有効な UPC-A コードをデコードします(デフォルト)。

bAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを、以下の値で指定します。

TRUE　：バーコードのデータとアドオンをスペースで区切って出力します。
FALSE　：バーコードのデータとアドオンをスペースで区切らずに出力します(デフォルト)。

bAddenda2Digit

2 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：2 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE　：アドオンを無視します(デフォルト)。

bAddenda5Digit

5 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：5 桁のアドオンが存在するとき、アドオンデータも読み取ります。
FALSE　：アドオンを無視します(デフォルト)。

bXmitNumSys

UPC ラベルのナンバーシステムキャラクタの出力の有無を、以下の値で指定します。

TRUE　：UPC ラベルのナンバーシステムキャラクタを出力します(デフォルト)。
FALSE　：UPC ラベルのナンバーシステムキャラクタを出力しません。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

補足

UPC-A コードを読み取り有効にすると、先頭が"0"で始まる EAN13 コードも UPC-A コードと認識し、読み取りが可能になります。

3.82 IMGGetUPCA

UPC-A コードのデコードオプションを取得します。

```
[C++]
int IMGGetUPCA(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar,
  LPBOOL pAddendaReq,
  LPBOOL pAddendaSeparator,
  LPBOOL pAddenda2Digit,
  LPBOOL pAddenda5Digit,
  LPBOOL pXmitNumSys
)
```

```
[Visual Basic]
Public Shared Function IMGGetUPCA( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean, _
  ByRef pAddendaReq As Boolean, _
  ByRef pAddendaSeparator As Boolean, _
  ByRef pAddenda2Digit As Boolean, _
  ByRef pAddenda5Digit As Boolean, _
  ByRef pXmitNumSys As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetUPCA(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar,
  ref Boolean pAddendaReq,
  ref Boolean pAddendaSeparator,
  ref Boolean pAddenda2Digit,
  ref Boolean pAddenda5Digit,
  ref Boolean pXmitNumSys
);
```

解説

UPC-A コードのデコードオプションを取得します。

パラメータ

pEnabled

UPC-A コードの読み取り有効/無効を取得します。取得する値は IMGSetUPCA 関数を参照してください。

pXmitCheckChar

チェックキャラクタ出力の有無を取得します。取得する値は IMGSetUPCA 関数を参照してください。

pAddendaReq

アドオンをもつ UPC-A コードのみをデコードするかすべての UPC-A コードをデコードするかを取得します。取得する値は IMGSetUPCA 関数を参照してください。

pAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを取得します。取得する値は IMGSetUPCA 関数を参照してください。

pAddenda2Digit

2 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetUPCA 関数を参照してください。

pAddenda5Digit

5 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetUPCA 関数を参照してください。

pXmitNumSys

UPC ラベルのナンバーシステムキャラクタの出力の有無を取得します。取得する値は IMGSetUPCA 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.83 IMGSetUPCE

UPC-E コードのデコードオプションを設定します。

```
[C++]
int IMGSetUPCE(
  BOOL bEnabled,
  BOOL bXmitCheckChar,
  BOOL bAddendaReq,
  BOOL bAddendaSeparator,
  BOOL bAddenda2Digit,
  BOOL bAddenda5Digit,
  BOOL bXmitNumSys,
  BOOL bExpandVersionE
)
```

```
[Visual Basic]
Public Shared Function IMGSetUPCE( _
  ByVal bEnabled As Boolean, _
  ByVal bXmitCheckChar As Boolean, _
  ByVal bAddendaReq As Boolean, _
  ByVal bAddendaSeparator As Boolean, _
  ByVal bAddenda2Digit As Boolean, _
  ByVal bAddenda5Digit As Boolean, _
  ByVal bXmitNumSys As Boolean, _
  ByVal bExpandVersionE As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetUPCE(
  Boolean bEnabled,
  Boolean bXmitCheckChar,
  Boolean bAddendaReq,
  Boolean bAddendaSeparator,
  Boolean bAddenda2Digit,
  Boolean bAddenda5Digit,
  Boolean bXmitNumSys,
  Boolean bExpandVersionE
);
```

解説

UPC-E コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetUPCE 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
UPC-E コード読み取りの有効/無効を、以下の値で指定します。
TRUE　　: 有効

FALSE　：無効(デフォルト)

bXmitCheckChar

チェックキャラクタ出力の有無を、以下の値で指定します。

TRUE　：チェックキャラクタを出力します。

FALSE　：チェックキャラクタを出力しません(デフォルト)。

bAddendaReq

アドオンをもつ UPC-E コードのみをデコードするか、すべての UPC-E コードを設定するかを、以下の値で指定します。

TRUE　：2 桁または 5 桁のアドオンをもつ UPC-E コードのみをデコードします。

FALSE　：すべての有効な UPC-E コードをデコードします(デフォルト)。

bAddendaSeparator

バーコードのデータとアドオンをスペースで区切って出力するかどうかを、以下の値で指定します。

TRUE　：バーコードのデータとアドオンをスペースで区切って出力します。

FALSE　：バーコードのデータとアドオンをスペースで区切らずに出力します(デフォルト)。

bAddenda2Digit

2 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：2 桁のアドオンが存在するとき、アドオンデータも読み取ります。

FALSE　：アドオンを無視します(デフォルト)。

bAddenda5Digit

5 桁のアドオンの読み取り有効/無効を、以下の値で指定します。

TRUE　：5 桁のアドオンが存在するとき、アドオンデータも読み取ります。

FALSE　：アドオンを無視します(デフォルト)。

bXmitNumSys

UPC ラベルのナンバーシステムキャラクタの出力の有無を、以下の値で指定します。

TRUE　：UPC ラベルのナンバーシステムキャラクタを出力します。

FALSE　：UPC ラベルのナンバーシステムキャラクタを出力しません(デフォルト)。

bExpandVersionE

UPC-E コードを 12 桁 UPC-A フォーマットに拡張するかどうかを、以下の値で指定します。

TRUE　：UPC-E コードを 12 桁 UPC-A フォーマットに拡張します。

FALSE　：UPC-E コードを拡張しません(デフォルト)。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.84 IMGGetUPCE

UPC-E コードのデコードオプションを取得します。

```
[C++]
int IMGGetUPCE(
  LPBOOL pEnabled,
  LPBOOL pXmitCheckChar,
  LPBOOL pAddendaReq,
  LPBOOL pAddendaSeparator,
  LPBOOL pAddenda2Digit,
  LPBOOL pAddenda5Digit,
  LPBOOL pXmitNumSys,
  LPBOOL pExpandVersionE
)
```

```
[Visual Basic]
Public Shared Function IMGGetUPCE( _
  ByRef pEnabled As Boolean, _
  ByRef pXmitCheckChar As Boolean, _
  ByRef pAddendaReq As Boolean, _
  ByRef pAddendaSeparator As Boolean, _
  ByRef pAddenda2Digit As Boolean, _
  ByRef pAddenda5Digit As Boolean, _
  ByRef pXmitNumSys As Boolean, _
  ByRef pExpandVersionE As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetUPCE(
  ref Boolean pEnabled,
  ref Boolean pXmitCheckChar,
  ref Boolean pAddendaReq,
  ref Boolean pAddendaSeparator,
  ref Boolean pAddenda2Digit,
  ref Boolean pAddenda5Digit,
  ref Boolean pXmitNumSys,
  ref Boolean pExpandVersionE
);
```

解説

UPC-E コードのデコードオプションを取得します。

パラメータ

pEnabled

UPC-E コードの読み取り有効/無効を取得します。取得する値は IMGSetUPCE 関数を参照してください。

pXmitCheckChar
チェックキャラクタ出力の有無を取得します。取得する値は IMGSetUPCE 関数を参照してください。

pAddendaReq
アドオンをもつ UPC-E コードのみをデコードするか、すべての UPC-E コードを設定するかを取得します。取得する値は IMGSetUPCE 関数を参照してください。

pAddendaSeparator
バーコードのデータとアドオンをスペースで区切って出力するかどうかを取得します。取得する値は IMGSetUPCE 関数を参照してください。

pAddenda2Digit
2 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetUPCE 関数を参照してください。

pAddenda5Digit
5 桁のアドオンの読み取り有効/無効を取得します。取得する値は IMGSetUPCE 関数を参照してください。

pXmitNumSys
UPC ラベルのナンバーシステムキャラクタの出力の有無を取得します。取得する値は IMGSetUPCE 関数を参照してください。

pExpandVersionE
UPC-E コードを 12 桁 UPC-A フォーマットに拡張するかどうかを取得します。取得する値は IMGSetUPCE 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない（DT-X8-4x シリーズのみ） DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.85 IMGSetMesa

AztecMesa Code のコードのデコードオプションを設定します。

```
[C++]
int IMGSetMesa(
  BOOL bUMSEnabled,
  BOOL bEMSEnabled,
  BOOL b3MSEnabled,
  BOOL b1MSEnabled,
  BOOL bIMSEnabled,
  BOOL b9MSEnabled,
)
```

```
[Visual Basic]
Public Shared Function IMGSetMesa( _
  ByVal bUMSEnabled As Boolean, _
  ByVal bEMSEnabled As Boolean, _
  ByVal b3MSEnabled As Boolean, _
  ByVal b1MSEnabled As Boolean, _
  ByVal bIMSEnabled As Boolean, _
  ByVal b9MSEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetMesa(
  Boolean bUMSEnabled,
  Boolean bEMSEnabled,
  Boolean b3MSEnabled,
  Boolean b1MSEnabled,
  Boolean bIMSEnabled,
  Boolean b9MSEnabled
);
```

解説

AztecMesa Code のコードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetMesa 関数を実行することにより、設定内容を確認することができます。

パラメータ

bUMSEnabled
UPCA Mesa コード読取の有効/無効を以下の値で指定します。
TRUE : UPCA Mesa コードは有効。
FALSE : UPCA Mesa コードは無効。

bEMSEnabled
EAN13 Mesa コード読取の有効/無効を以下の値で指定します。
TRUE : EAN13 Mesa コードは有効。
FALSE : EAN13 Mesa コードは無効。

b3MSEnabled

Code39 Mesa コード読取の有効/無効を以下の値で指定します。

TRUE : Code39 Mesa コードは有効。
FALSE : Code39 Mesa コードは無効。

b1MSEnabled

Code128 Mesa コード読取の有効/無効を以下の値で指定します。

TRUE : Code128 Mesa コードは有効。
FALSE : Code128 Mesa コードは無効。

bIMSEnabled

Interleaved 2 of 5 コード読取の有効/無効を以下の値で指定します。

TRUE : Interleaved 2 of 5 Mesa コードは有効。
FALSE : Interleaved 2 of 5 Mesa コードは無効。

b9MSEnabled

Code93 Mesa コード読取の有効/無効を以下の値で指定します。

TRUE : Code93 Mesa コードは有効。
FALSE : Code93 Mesa コードは無効。

戻り値

IMG_SUCCESS : 正常終了
FUNCTION_UNSUPPORT : 未サポートエラー

補足

最小および最大のパラメータは IMGSetAztec 関数で使用しているものをセットします。UPCA Mesa コードの読み取りを有効にするためには、UPCA コードの読み取りを有効にする必要があります。
IMGSetUPCA 関数を呼び出して、UPCA コードの読み取りを有効にして下さい。

3.86 IMGGetMesa

AztecMesa コードのデコードオプションを取得します。

```
[C++]
int IMGGetMesa(
  LPBOOL pUMSEnabled,
  LPBOOL pEMSEnabled,
  LPBOOL p3MSEnabled,
  LPBOOL p1MSEnabled,
  LPBOOL pIMSEnabled,
  LPBOOL p9MSEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetMesa( _
  ByRef pUMSEnabled As Boolean, _
  ByRef pEMSEnabled As Boolean, _
  ByRef p3MSEnabled As Boolean, _
  ByRef p1MSEnabled As Boolean, _
  ByRef pIMSEnabled As Boolean, _
  ByRef p9MSEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetMesa(
  ref Boolean pUMSEnabled,
  ref Boolean pEMSEnabled,
  ref Boolean p3MSEnabled,
  ref Boolean p1MSEnabled,
  ref Boolean pIMSEnabled,
  ref Boolean p9MSEnabled
);
```

解説

AztecMesa コードのデコードオプションを取得します。

パラメータ

pUMSEnabled

UPCA Mesa コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

pEMSEnabled

EAN13 Mesa コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

p3MSEnabled

Code39 Mesa コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

p1MSEnabled

Code128 Mesa コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

pIMSEnabled

Interleaved 2 of 5 コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

p9MSEnabled

Code93 Mesa コード読取の有効/無効を取得します。取得する値は IMGSetMesa 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.87 IMGSetJaPost

日本の郵便番号コードのデコードオプションを設定します。

```
[C++]
int IMGSetJaPost(
  BOOL bEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGSetJaPost( _
  ByVal bEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGSetJaPost(
  Boolean bEnabled
);
```

解説

日本の郵便番号コードのデコードオプションを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetJaPost 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnabled
日本の郵便番号読み取りの有効/無効を、以下の値で指定します。
TRUE : 有効
FALSE : 無効(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.88 IMGGetJaPost

日本の郵便番号コードのデコードオプションを取得します。

```
[C++]
int IMGGetJaPost(
  LPBOOL pEnabled
)
```

```
[Visual Basic]
Public Shared Function IMGGetJaPost( _
  ByRef pEnabled As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGGetJaPost(
  ref Boolean pEnabled
);
```

解説

日本の郵便番号コードのデコードオプションを取得します。

パラメータ

pEnabled

日本の郵便番号コードの読み取り有効/無効を取得します。取得する値は IMGSetJaPost 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.89 IMGAimerOn

エイマを照射開始/停止します。

```
[C++]
int IMGAimerOn(
  BOOL bEnable
)
```

```
[Visual Basic]
Public Shared Function IMGAimerOn( _
  ByVal bEnable As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGAimerOn(
  bool bEnable
);
```

解説

イメージャのエイマを照射開始、または照射停止します。
Device Emulator では、IMGInit および IMGConnect 関数の実行確認のみを行います。

パラメータ

bEnable
エイマの照射開始/停止を以下の値で指定します。
TRUE : エイマを照射開始。
FALSE : エイマを照射停止。(デフォルト)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない
IMG_ERR_NOTCONNECTED	: IMGConnect が実行されていない
IMG_ERR_DRIVER	: イメージャドライバにエラー発生 Device Emulator では発生しません
IMG_ERR_NORESPONSE	: デバイスドライバからの応答がない Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.90 IMGIlluminationOn

イルミネーション LED を点灯または消灯します。

```
[C++]
int IMGIlluminationOn(
  BOOL bEnable
)
```

```
[Visual Basic]
Public Shared Function IMGIlluminationOn( _
  ByVal bEnable As Boolean _
) As Int32
```

```
[C#]
public static Int32 IMGIlluminationOn(
  Boolean bEnable
);
```

解説

イメージャのイルミネーション LED を点灯または消灯します。
Device Emulator では、IMGInit および IMGConnect 関数の実行確認のみを行います。

パラメータ

bEnable
イルミネーション LED の点灯/消灯を以下の値で指定します。
TRUE　：イルミネーション LED を点灯。
FALSE　：イルミネーション LED を消灯。(デフォルト)

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない
IMG_ERR_NOTCONNECTED	：IMGConnect が実行されていない
IMG_ERR_DRIVER	：イメージャドライバにエラー発生 Device Emulator では発生しません
IMG_ERR_NORESPONSE	：デバイスドライバからの応答がない Device Emulator では発生しません
FUNCTION_UNSUPPORT	：未サポートエラー

3.91 IMGSetAimer

エイマ強度を設定します。

```
[C++]
int IMGSetAimer(
  DWORD dwAim
)
```

```
[Visual Basic]
Public Shared Function IMGSetAimer( _
  ByVal dwAim As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetAimer(
  Int32 dwAim
);
```

解説

IMGWaitForDecode 関数、IMGWaitForDecodeRaw 関数、IMGAimerOn 関数実行時に点灯するイメージャのエイマの強度を設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetAimer 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwAim
エイマの強度を指定します。
照射せず　　: 0
照射します　: 0 以外

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.92 IMGGetAimer

エイマ強度を取得します。

```
[C++]
int IMGGetAimer(
  LPDWORD pAim
)
```

```
[Visual Basic]
Public Shared Function IMGGetAimer( _
  ByRef pAim As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetAimer(
  ref Int32 pAim
);
```

解説

IMGWaitForDecode 関数、IMGWaitForDecodeRaw 関数、IMGAimerOn 関数実行時に点灯するイメージャのエイマの強度を取得します。

パラメータ

pAim
エイマ強度を取得します。取得する値は IMGSetAimer 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.93 IMGSetIllumination

イルミネーション LED 強度を設定します。

```
[C++]
int IMGSetIllumination(
  DWORD dwIll
)
```

```
[Visual Basic]
Public Shared Function IMGSetIllumination( _
  ByVal dwIll As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetIllumination(
  Int32 dwIll
);
```

解説

イメージャのイルミネーション LED の強度を設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetIllumination 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwIll
イルミネーション LED の強度を指定します。設定可能範囲は 0～100(%)です。
0 に設定するとイルミネーション LED は点灯しません。デフォルト値は 70 です。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.94 IMGGetIllumination

イルミネーション LED 強度を取得します。

```
[C++]
int IMGGetIllumination(
  LPDWORD pIll
)
```

```
[Visual Basic]
Public Shared Function IMGGetIllumination( _
  ByRef pIll As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetIllumination(
  ref Int32 pIll
);
```

解説

イメージャのイルミネーション LED の強度設定を取得します。

パラメータ

pIll

イルミネーション LED の強度を取得します。取得する値は IMGSetIllumination 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.95 IMGSetIlluminationEx

イルミネーション LED 強度およびモードを設定します。

```
[C++]
int IMGSetIlluminationEx(
  DWORD dwIll,
  DWORD dwMode,
  DWORD dwReserved
)
```

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

```
[C#]
public static Int32 IMGSetIlluminationEx(
  Int32 dwIll,
  Int32 dwMode,
  Int32 dwReserved
);
```

解説

イメージャのイルミネーション LED の強度およびモードを設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、
IMGGetIlluminationEx 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwIll
イルミネーション LED の強度を指定します。設定可能範囲は 0～100(%)です。
0 に設定するとイルミネーション LED は点灯しません。デフォルト値は 70 です。

dwMode
イルミネーション LED の点灯モードを指定します。
IMG_ILLUM_BLINK : スキャン中にイルミネーション LED が点滅 (デフォルト)
IMG_ILLUM_ON : スキャン中にイルミネーション LED が点灯

dwReserved
0 を指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.96 IMGGetIlluminationEx

イルミネーション LED 強度およびモードを取得します。

```
[C++]
int IMGGetIlluminationEx(
  LPDWORD pIll,
  LPDWORD pMode,
  LPDWORD pReserved
)
```

```
[Visual Basic]
Public Shared Function IMGGetIlluminationEx( _
  ByRef pIll As Int32, _
  ByRef pMode As Int32, _
  ByRef pReserved As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetIlluminationEx(
  ref Int32 pIll,
  ref Int32 pMode,
  ref Int32 pReserved
);
```

解説

イメージャのイルミネーション LED の強度およびモードの設定を取得します。

パラメータ

pIll

イルミネーション LED の強度を取得します。取得する値は IMGSetIlluminationEx 関数を参照してください。

pMode

イルミネーション LED の点灯モードを取得します。取得する値は IMGSetIlluminationEx 関数を参照してください。

pReserved

NULL を指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_MEMORY	: メモリエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.97 IMGSetScanMode

スキャンモードを設定します。

```
[C++]
int IMGSetScanMode(
  DWORD dwScanMode
)
```

```
[Visual Basic]
Public Shared Function IMGSetScanMode( _
  ByVal dwScanMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetScanMode(
  Int32 dwScanMode
);
```

解説

環境に応じたスキャンモードを設定します。スキャンモードとは露光設定パラメータを、各外光条件に合わせてセットするモードのことです。スキャンモードの変更により、極端に明るい場所や、極端に暗い場所など、読み取りにくい環境での読み取り性能の向上を図ります。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetScanMode 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwScanMode
スキャンモードを以下の値で指定します。デフォルトは IMG_SCAN_MODE_WINDOWSIDE です。

IMG_SCAN_MODE_OUTDOOR	: 屋外の明るさに適切な露光設定
IMG_SCAN_MODE_WINDOWSIDE	: 屋内の窓際の明るさに適切な露光設定
IMG_SCAN_MODE_INDOOR	: 屋内の明るさに適切な露光設定
IMG_SCAN_MODE_WAREHOUSE	: 倉庫の明るさに適切な露光設定

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー Device Emulator では発生しません
IMG_ERR_DRIVER	: スキャナドライバエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

場所の指定はあくまで明るさの目安です。指定モードと一致する場所で、必ずスキャンできるようになるというわけではありません。またスキャンモードを変更しても、必ず読み取り性能が良くなるとは限りません。現在のモードでは読み取りにくいといった場合、スキャンモードを変更してみることを推奨します。

3.98 IMGGetScanMode

スキャンモードを取得します。

```
[C++]
int IMGGetScanMode(
  LPDWORD pScanMode
)
```

```
[Visual Basic]
Public Shared Function IMGGetScanMode( _
  ByRef pScanMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetScanMode(
  ref Int32 pScanMode
);
```

パラメータ

pdwScanMode

スキャンモードを取得します。取得する値は IMGSetScanMode 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
FUNCTION_UNSUPPORT	: 未サポートエラー

3.99 IMGSetImagerAPO

オートパワーオフタイマ値を設定します。

```
[C++]
int IMGSetImagerAPO(
  DWORD dwTimeOut
)
```

```
[Visual Basic]
Public Shared Function IMGSetImagerAPO( _
  ByVal dwTimeOut As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetImagerAPO(
  Int32 dwTimeOut
);
```

解説

イメージャのオートパワーオフタイマ値を設定します。省電力機能として、一定時間イメージャが使用されていなかった場合に、イメージャデバイスの電源を自動的にオフする機能です。以下のいずれかの関数が指定時間以上使用されなかった場合、イメージャの電源を自動的にオフします。

・IMGWaitForDecode
・IMGWaitForDecodeRaw
・IMGGetImage
・IMGStopStream
・IMGCaptureSign
・IMGIlluminationOn (消灯(FALSE)指定時)
・IMGAimerOn (消灯(FALSE)指定時)

Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetImagerAPO 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwTimeOut

オートパワーオフタイマのタイムアウト値を 0～1800(秒)の範囲で設定します。タイムアウト値を 0 に指定した場合は、イメージャの電源は自動的にオフしません。デフォルト値は 60 です。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.100 IMGGetImagerAPO

オートパワーオフタイマ値を取得します。

```
[C++]
int IMGGetImagerAPO(
  LPDWORD pTimeOut
)
```

```
[Visual Basic]
Public Shared Function IMGGetImagerAPO( _
  ByRef pTimeOut As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetImagerAPO(
  ref Int32 pTimeOut
);
```

解説

イメージャのオートパワーオフタイマの設定値を取得します。

パラメータ

pTimeOut

オートパワーオフタイマのタイムアウト値を取得します。取得する値は IMGSetImagerAPO 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
FUNCTION_UNSUPPORT	: 未サポートエラー

3.101 IMGSetPrintWeight

デコード時の印字太さ調整値を設定します。

```
[C++]
int IMGSetPrintWeight(
  DWORD dwPrintWeight
)
```

```
[Visual Basic]
Public Shared Function IMGSetPrintWeight( _
  ByVal dwPrintWeight As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetPrintWeight(
  Int32 dwPrintWeight
);
```

解説

デコードをする際に使用する印字太さ調整値(PrintWeight)を設定します。黒バーの細った、あるいは太ったシンボルをデコードする場合、この値を変更することによりデコード時の性能が向上する場合があります。基準より黒バーの細ったシンボルをデコードするときはデフォルト値より減らす方向に、逆に黒バーの太ったシンボルをデコードするときは増やす方向に設定します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetPrintWeight 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwPrintWeight
デコード時の印字太さ調整値を、1～7 の範囲で設定します。デフォルトは 4 です。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.102 IMGGetPrintWeight

デコード時の印字太さ調整値を取得します。

```
[C++]
int IMGGetPrintWeight(
  LPDWORD pPrintWeight
)
```

```
[Visual Basic]
Public Shared Function IMGGetPrintWeight( _
  ByRef pPrintWeight As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetPrintWeight(
  ref Int32 pPrintWeight
);
```

解説

デコードをする際に使用する印字太さ調整値(PrintWeight)を取得します。

パラメータ

pPrintWeight

デコード時の印字太さ調整値を取得します。取得する値は IMGSetPrintWeight 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
FUNCTION_UNSUPPORT	: 未サポートエラー

3.103 IMGSetLED

LED によるスキャナの読み取り通知方式を設定します。

```
[C++]
int IMGSetLED(
  DWORD dwLED
)
```

```
[Visual Basic]
Public Shared Function IMGSetLED( _
  ByVal dwLED As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetLED(
  Int32 dwLED
);
```

解説

シンボルの読み取り(デコード)を行った場合に、インジケータ LED の点灯 ON/OFF による通知を行うかどうかを設定します。

パラメータ

dwLED

LED 点灯 ON/OFF を以下の値で指定します。デフォルトは IMG_LEDON です。

IMG_LEDON	: 読み取りが成功した場合は緑 LED、読み取りが失敗した場合は赤 LED による通知を行います。
IMG_LEDOFF	: 読み取り成功、失敗どちらの場合にも LED による通知は行いません。
IMG_LEDEROF	: 読み取りが成功した場合に緑 LED による通知を行います。 : 読み取りが失敗した場合には赤 LED による通知を行いません。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.104 IMGGetLED

LED によるスキャナ読み取り通知方式を取得します。

```
[C++]
int IMGGetLED(
  LPDWORD pLED
)
```

```
[Visual Basic]
Public Shared Function IMGGetLED( _
  ByRef pLED As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetLED(
  ref Int32 pLED
);
```

解説

シンボルの読み取り(デコード)を行った場合に、インジケータ LED の点灯 ON/OFF による通知を行うかどうかの設定を取得します。

パラメータ

pLED
LED 点灯 ON/OFF を取得します。取得する値は IMGSetLED 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.105 IMGSetBuzzer

ブザーによるスキャナの読み取り通知方式を設定します。

```
[C++]
int IMGSetBuzzer(
  DWORD dwBuzzer
)
```

```
[Visual Basic]
Public Shared Function IMGSetBuzzer( _
  ByVal dwBuzzer As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetBuzzer(
  Int32 dwBuzzer
);
```

解説

シンボルの読み取り(デコード)に成功した場合、ブザーによる通知を行うかどうかの設定を行います。

パラメータ

dwBuzzer

ブザーによる通知の ON/OFF を、以下の値で指定します。

IMG_BUZON　：読み取り成功時、ブザーによる通知を行います。(デフォルト)
IMG_BUZOFF　：読み取り成功時、ブザーによる通知を行いません。

戻り値

IMG_SUCCESS	：正常終了
IMG_ERR_NOTINITIALIZED	：IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	：パラメータエラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.106 IMGGetBuzzer

ブザーによるスキャナ読み取り通知方式を取得します。

```
[C++]
int IMGGetBuzzer(
  LPDWORD pBuzzer
)
```

```
[Visual Basic]
Public Shared Function IMGGetBuzzer( _
  ByRef pBuzzer As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetBuzzer(
  ref Int32 pBuzzer
);
```

解説

シンボルの読み取り(デコード)に成功した場合、ブザーによる通知を行うかどうかの設定を取得します。

パラメータ

pBuzzer
ブザーによる通知の ON/OFF を取得します。取得する値は IMGSetBuzzer 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.107 IMGSetVibrator

バイブレータによるスキャナの読み取り通知方式を設定します。

```
[C++]
int IMGSetVibrator(
  DWORD dwVibrator
)
```

```
[Visual Basic]
Public Shared Function IMGSetVibrator( _
  ByVal dwVibrator As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetVibrator(
  Int32 dwVibrator
);
```

解説

シンボルの読み取り(デコード)に成功した場合、バイブレータにより通知を行うかどうかの設定を行います。

パラメータ

dwVibrator

バイブレータの ON/OFF を、以下の値で指定します。

IMG_VIBON　　: 読み取り成功時、バイブレータによる通知を行います。(デフォルト)
IMG_VIBOFF　　: 読み取り成功時、バイブレータによる通知を行いません。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_PARAMETER	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.108 IMGGetVibrator

バイブレータによるスキャナ読み取り通知方式を取得します。

```
[C++]
int IMGGetVibrator(
  LPDWORD pVibrator
)
```

```
[Visual Basic]
Public Shared Function IMGGetVibrator( _
  ByRef pVibrator As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetVibrator(
  ref Int32 pVibrator
);
```

解説

シンボルの読み取り(デコード)に成功した場合、バイブレータによる通知を行うかどうかの設定を取得します。

パラメータ

pVibrator
バイブレータの ON/OFF を取得します。取得する値は IMGSetVibrator 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.109 IMGSetDeliberation

シンボルのデコードを行う際の熟考度を設定します。

```
[C++]
int IMGSetDeliberation(
  DWORD dwDeliberateTime
)
```

```
[Visual Basic]
Public Shared Function IMGSetDeliberation( _
  ByVal dwDeliberateTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetDeliberation(
  Int32 dwDeliberateTime
);
```

解説

シンボルのデコードを行う際の熟考度を設定します。熟考度は5段階で設定することができます。熟考度をIMG_DECODE_VERYQUICKやIMG_DECODE_QUICKに指定するとデコードできるシンボルは限定されますが、デコードのスピードは速くなります。逆に熟考度をIMG_DECODE_DELIBERATEやIMG_DECODE_VERYDELIBERATEに指定すると多くのシンボルを読むことができますが、デコードのスピードは遅くなります。
Device Emulatorでは、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetDeliberation関数を実行することにより、設定内容を確認することができます。

パラメータ

dwDeliberateTime
デコードを行う際の熟考度を以下の値で指定します。

IMG_DECODE_VERYQUICK	: 読み取り可能なシンボルは非常に限定されるが、デコードスピードは非常に速い
IMG_DECODE_QUICK	: 読み取り可能なシンボルはやや限定されるが、デコードスピードは速い
IMG_DECODE_NORMAL	: 読み取り可能なシンボル、デコードスピードは普通(デフォルト)
IMG_DECODE_DELIBERATE	: 読み取り可能なシンボルはやや多いが、デコードスピードは遅い
IMG_DECODE_VERYDELIBERATE	: 読み取り可能なシンボルは多いが、デコードスピードは非常に遅い

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInitが実行されていない(DT-X8-4xシリーズのみ) DeviceEmulatorでは発生しません
IMG_ERR_PARAMETER	: パラメータエラー Device Emulatorでは発生しません

FUNCTION_UNSUPPORT　　：未サポートエラー

補足

読み取るシンボルに適した熟考度は以下のとおりです。

デコード熟考度	読み取り対象シンボル	読み取りスピード
IMG_DECODE_VERYQUICK	印字品質の良い 1D バーコード	非常に速い
IMG_DECODE_QUICK	印字品質の悪い 1D バーコード 500 桁以下の PDF417	速い
IMG_DECODE_NORMAL	1000 桁以下の PDF417、DataMatrix、QR、Aztec MicroPDF、Code49、Codablock F Composite コード Maxicode	普通
IMG_DECODE_DELIBERATE	1000～2000 桁の PDF417、DataMatrix、QR、Aztec 80 桁以下の TLC39 RSS-14 Stacked/RSS Expanded Stacked	遅い
IMG_DECODE_VERYDELIBERATE	2000 桁以上の PDF417、DataMatrix、QR、Aztec 80 桁以上の TLC39	非常に遅い

上記の設定は目安であり、シンボルの印字状態(分解能、PCS など)及び周囲の環境によって変化します。デコードモードが一括読み（IMGSetDecodeMode 参照）に設定されている場合には、IMG_DECODE_NORMAL～IMG_DECODE_VERYDELIBERATE を指定してください。

3.110 IMGGetDeliberation

シンボルのデコードを行う際の熟考度を取得します。

```
[C++]
int IMGGetDeliberation(
  LPDWORD pDeliberateTime
)
```

```
[Visual Basic]
Public Shared Function IMGGetDeliberation( _
  ByRef pDeliberateTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetDeliberation(
  ref Int32 pDeliberateTime
);
```

解説

シンボルのデコードを行う際の熟考度を取得します。

パラメータ

pDeliberateTime

デコードを行う際の熟考度を取得します。取得する値は IMGSetDeliberation 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.111 IMGSetDecodeCenteringWindow

デコードセンタリングモードを設定します。

```
[C++]
int IMGSetDecodeCenteringWindow(
  BOOL bEnable,
  RECT *pIntersectRect
)
```

```
[Visual Basic]
Public Shared Function IMGSetDecodeCenteringWindow( _
  ByVal bEnabled As Boolean, _
  ByRef pIntersectRect As IMGLibNet.RECT& _
) As Int32
```

```
[C#]
public static Int32 IMGSetDecodeCenteringWindow(
  Boolean bEnabled,
  ref IMGLibNet.RECT& pIntersectRect
);
```

解説

デコードセンタリングモードを設定をします。このモードを有効にした場合、イメージの中心より指定した長方形の範囲にあるシンボルのみをデコードします。この関数により、2D スキャナは近接するシンボルを区別することができます。そのため、1回のキャプチャの間にひとつのシンボルをデコードします。Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetDecodeCenteringWindow 関数を実行することにより、設定内容を確認することができます。

パラメータ

bEnable

センタリングモードを、以下の値で指定します。

TRUE : センターリングを ON
FALSE : センターリングを OFF

pIntersectRect

有効なデコードデータと認識されるためにシンボルがオーバーラップしなければならない長方形のイメージ領域を指定します。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.112 IMGGetDecodeCenteringWindow

デコードセンタリングモードの設定情報を取得します。

```
[C++]
int IMGGetDecodeCenteringWindow(
  BOOL *pbEnabled,
  RECT *pIntersectRect
)
```

```
[Visual Basic]
Public Shared Function IMGGetDecodeCenteringWindow( _
  ByRef pbEnabled As Boolean, _
  ByRef pIntersectRect As IMGLibNet.RECT& _
) As Int32
```

```
[C#]
public static Int32 IMGGetDecodeCenteringWindow(
  ref Boolean pbEnabled,
  ref IMGLibNet.RECT& pIntersectRect
);
```

解説

デコードセンタリングモードの設定情報を取得します。

パラメータ

pbEnabled

センタリングモードを取得します。取得する値は IMGSetDecodeCenteringWindow 関数を参照してください。

pIntersectRect

有効なデコードデータと認識されるために、シンボルがオーバーラップしなければならない長方形のイメージ領域を取得します。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.113 IMGLoadConfigFile

設定ファイルによるイメージャの設定を行います。

```
[C++]
int IMGLoadConfigFile(
  PTCHAR pFileName
)
```

```
[Visual Basic]
Public Shared Function IMGLoadConfigFile( _
  ByVal pFileName As String _
) As Int32
```

```
[C#]
public static Int32 IMGLoadConfigFile(
  string pFileName
);
```

解説

指定された設定ファイルを読み込み、読み込んだ設定をイメージャに設定します。指定されたファイルがない場合は、設定は現在の設定のままとなります。設定ファイルの書式については「設定ファイルの書式について」を参照してください。設定ファイルの書式にあてはまらないファイルを読み込んだ場合の動作は保証しません。設定ファイルを直接編集する場合は注意してください。

パラメータ

pFileName

読み込み元となるファイルのフルパスを指定します。
TEXT("¥¥FlashDisk¥¥ImagerConfig.ini")と指定した場合、「FlashDisk」フォルダ内の「ImagerConfig.ini」というファイルを読み込み、設定をイメージャにセットします。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_FILE	: ファイル操作に失敗 Device Emulator では発生しません
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー発生 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.114 IMGSaveConfigFile

イメージャの設定を取得して保存します。

```
[C++]
int IMGSaveConfigFile(
  PTCHAR pFileName
)
```

```
[Visual Basic]
Public Shared Function IMGSaveConfigFile( _
  ByVal pFileName As String _
) As Int32
```

```
[C#]
public static Int32 IMGSaveConfigFile(
  string pFileName
);
```

解説

現在のイメージャの設定を取得し、指定されたテキストファイルに保存します。指定されたファイルと同じ名前のファイルが既に存在する場合、上書きして保存を行います。設定ファイルの書式については「設定ファイルの書式について」を参照してください。
Device Emulator では、規定のフォルダに設定内容をファイルとして保存します。

パラメータ

pFileName
保存先のファイルのフルパスを指定します。
TEXT("¥¥FlashDisk¥¥ImagerConfig.ini")と指定した場合、「FlashDisk」フォルダ内に「ImagerConfig.ini」というファイル名で設定が保存されます。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない(DT-X8-4x シリーズのみ) DeviceEmulator では発生しません
IMG_ERR_FILE	: ファイル操作に失敗 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.115 IMGMakeImageFile

イメージデータを保存します。

```
[C++]
int IMGMakeImageFile(
  TCHAR *pFileName,
  BYTE *pBuffer,
  DWORD dwSize,
  WORD nWidth,
  WORD nHeight,
  DWORD dwFormat
)
```

```
[Visual Basic]
Public Shared Function IMGMakeImageFile( _
  ByVal pFileName As String, _
  ByRef pBuffer As Byte, _
  ByVal dwSize As Int32, _
  ByVal nWidth As Short, _
  ByVal nHeight As Short, _
  ByVal dwFormat As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGMakeImageFile(
  string pFileName,
  ref Byte pBuffer,
  Int32 dwSize,
  short nWidth,
  short nHeight,
  Int32 dwFormat
);
```

解説

イメージデータを保存します。IMGGetImage 関数で取得した画像データを、指定したファイル形式に保存します。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

pFileName
保存先のファイルのフルパスを指定します。
TEXT("¥¥My Documents¥¥Picture.bmp")と指定した場合、「マイドキュメント」フォルダ内に「Picture.bmp」というファイル名で画像が保存されます。

pBuffer
IMGGetImage により取得した画像データが格納されているバッファを指定します。

dwSize

IMGGetImage により取得した画像データのサイズを指定します。

nWidth

画像データの幅を指定します。

nHeight

画像データの高さを指定します。

dwFormat

ファイルフォーマットを、以下の値で指定します。

IMG_FORMAT_BMP	: 8bit グレースケールビットマップ形式
IMG_FORMAT_JPEG_LOW	: 低画質の 24Bit JPEG 形式
IMG_FORMAT_JPEG_MID	: 中画質の 24Bit JPEG 形式
IMG_FORMAT_JPEG_HI	: 高画質の 24Bit JPEG 形式

次の値との論理和を指定すると、画像処理を行ってからファイル保存を行います。

IMG_FORMAT_PROG_STRETCH	: 画像のヒストグラム平滑化を行います。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_DRIVER	: ドライバ内でエラー発生 Device Emulator では発生しません
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー発生 Device Emulator では発生しません
IMG_ERR_MEMORY	: OS のメモリ不足 Device Emulator では発生しません
IMG_ERR_FILE	: ファイル操作に失敗 Device Emulator では発生しません
FUNCTION_UNSUPPORT	: 未サポートエラー

3.116 IMGSetDecodePreview

デコード時に指定したウィンドウにデコード画像の縮小画像が表示されます。

```
[C++]
int IMGSetDecodePreview(
  DWORD dwPreviewMode,
  HWND hWnd,
  DWORD dwPosX,
  DWORD dwPosY,
  DWORD dwResSize
)
```

```
[Visual Basic]
Public Shared Function IMGSetDecodePreview ( _
  ByVal dwPreviewMode As Int32, _
  ByVal hWnd As IntPtr, _
  ByVal dwPosX As Int32, _
  ByVal dwPosY As Int32, _
  ByVal dwResSize As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetDecodePreview(
  Int32 dwPreviewMode,
  IntPtr hWnd,
  Int32 dwPosX,
  Int32 dwPosY,
  Int32 dwResSize
)
```

解説

デコード画像のプレビュー表示のオプションを設定します。本関数を用いてプレビュー表示を有効に設定すると、デコード時(IMGWaitForDecode/IMGWaitForDecodeRaw 実行時)に指定したウィンドウにデコード画像の縮小画像が表示されます。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetDecodePreview 関数を実行することにより、設定内容を確認することができます。

パラメータ

dwPreviewMode
プレビュー表示モードの有効/無効を指定します。この引数に IMG_PREVIEW を指定すると、デコード時にデコード画像の縮小画像が表示されます。

IMG_PREVIEW	: デコード画像のプレビューを表示します
IMG_NOPREVIEW	: デコード画像のプレビューを表示しません(デフォルト)

hWnd
プレビューを表示するウィンドウのハンドルを指定します。

dwPosX

プレビューの左上の X 座標を指定します。

dwPosY

プレビューの左上の Y 座標を指定します。

dwResSize

プレビューの解像度を指定します。

IMG_4PER9VGA	: 4/9VGA サイズ(426 x 320)
IMG_QVGA	: QVGA:サイズ (320 x 240)
IMG_1PER9VGA	: 1/9VGA サイズ (213 x 160)

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: 不正なパラメータによるエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.117 IMGGetDecodePreview

デコード時に指定したウィンドウに、デコード画像の縮小画像を表示するかを取得します。

```
[C++]
int IMGGetDecodePreview(
  LPDWORD pPreviewMode,
  HWND hWnd,
  LPDWORD pPosX,
  LPDWORD pPosY,
  LPDWORD pResSize
)
```

```
[Visual Basic]
Public Shared Function IMGGetDecodePreview ( _
  ByRef pPreviewMode As Int32, _
  ByRef hWnd As IntPtr, _
  ByRef pPosX As Int32, _
  ByRef pPosY As Int32, _
  ByRef pResSize As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetDecodePreview(
  ref Int32 pPreviewMode,
  ref IntPtr hWnd,
  ref Int32 pPosX,
  ref Int32 pPosY,
  ref Int32 pResSize
)
```

解説

デコード画像のプレビュー表示のオプションを取得します。
デコード時(IMGWaitForDecode/IMGWaitForDecodeRaw 実行時)に、指定されたウィンドウとデコード画像の縮小画像を表示するかどうかを取得できます。

パラメータ

pPreviewMode
プレビュー表示モードの有効/無効を指定します。この引数に IMG_PREVIEW を指定すると、デコード時にデコード画像の縮小画像が表示されます。取得する値は IMGSetDecodePreview 関数を参照してください。

hWnd
プレビューを表示するウィンドウのハンドルを取得します。

pPosX
プレビューの左上の X 座標を取得します。

pPosY

プレビューの左上の Y 座標を取得します。

pResSize

プレビューの解像度を取得します。取得する値は IMGSetDecodePreview 関数を参照してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.118 IMGStartPreview

デコードのプレビューを表示します。

```
[C++]
int IMGStartPreview(
  HWND hWnd,
  DWORD dwPosX,
  DWORD dwPosY,
  DWORD dwResSize,
  DWORD dwFrame
)
```

```
[Visual Basic]
Public Shared Function IMGStartPreview ( _
  ByVal hWnd As IntPtr, _
  ByVal dwPosX As Int32, _
  ByVal dwPosY As Int32, _
  ByVal dwResSize As Int32 _
  ByVal dwFrame As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGStartPreview(
  IntPtr hWnd,
  Int32 dwPosX,
  Int32 dwPosY,
  Int32 dwResSize,
  Int32 dwFrame
)
```

解説

読み取るシンボルの位置合わせをしやすくするために、プレビュー上に読み取る範囲を示す枠を表示することができます。このプレビュー表示は、IMGWaitForDecode、IMGWaitForDecodeRaw、IMGStopPreview 関数がコールされるまで続行され、上記関数がコールされると終了します。デコード後にプレビューを再開する場合は、再度本関数をコールしてください。
本関数によりプレビューを表示した状態でトリガキーをスキャンし、シンボルの読み取りを行うことにより、読み取りの操作性が向上します。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

hWnd
プレビューを表示するウィンドウのハンドルを指定します。

dwPosX
プレビューの左上の X 座標を指定します。

dwPosY
プレビューの左上の Y 座標を指定します。

dwResSize

プレビューの解像度を指定します。

IMG_4PER9VGA : 4/9VGA サイズ(426 x 320)
IMG_QVGA : QVGA:サイズ (320 x 240)
IMG_1PER9VGA : 1/9VGA サイズ (213 x 160)

dwFrame

読み取り範囲を示す枠表示の有無を指定します。

IMG_PREVIEW_NOFRAME : 枠を表示しない
IMG_PREVIEW_FRAME1 : 枠を表示する

戻り値

IMG_SUCCESS : 正常終了
IMG_ERR_PARAMETER : パラメータエラー
FUNCTION_UNSUPPORT : 未サポートエラー

3.119 IMGStopPreview

デコードのプレビュー表示を停止します。

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

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

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

解説

デコードのプレビュー表示を停止し、カメラの電源をオフします。
Device Emulator では常に IMG_SUCCESS を返します。

パラメータ

なし

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.120 IMGSetFocus

シンボルをデコードする際のフォーカスを設定します。

```
[C++]
int IMGSetFocus(
  int nFocus,
  int nReserved
)
```

```
[Visual Basic]
Public Shared Function IMGSetFocus( _
  ByVal nFocus As Int32, _
  ByVal nReserved As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGSetFocus(
  Int32 nFocus,
  Int32 nReserved
);
```

解説

シンボルをデコードする際のフォーカスを設定します。
フォーカスを自動に設定すると、すべての距離のシンボルを読み取ることができるが、読取レスポンスは低下します。(高速モードに設定すると読取レスポンスは向上するが、遠距離の読取精度が低下します)
フォーカスを固定に設定すると、読み取る距離が限定されるが、読取レスポンスは向上します。
Device Emulator では、設定値を内部変数として格納するため、なにも動作しませんが、IMGGetFocus 関数を実行することにより、設定内容を確認することができます。

パラメータ

nFocus
デコードする際のフォーカスを指定します。

IMG_FOCUS_AUTO	: 自動フォーカス/通常モード
IMG_FOCUS_AUTO_FAST	: 自動フォーカス/高速モード
IMG_FOCUS_NEAR	: 固定フォーカス(近距離)
IMG_FOCUS_MIDDLE	: 固定フォーカス(中距離)
IMG_FOCUS_FAR	: 固定フォーカス(遠距離)

nReserved
0 を指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない
FUNCTION_UNSUPPORT	: 未サポートエラー

3.121 IMGGetFocus

シンボルをデコードする際のフォーカスを取得します。

```
[C++]
int IMGGetFocus(
  int *pFocus,
  int *pReserved
)
```

```
[Visual Basic]
Public Shared Function IMGGetFocus( _
  ByRef nFocus As Int32, _
  ByRef nReserved As Int32 _
) As Int32
```

```
[C#]
public static Int32 IMGGetFocus(
  ref Int32 nFocus,
  ref Int32 nReserved
);
```

解説

シンボルをデコードする際のフォーカスを取得します。

パラメータ

nFocus

デコードする際のフォーカスを取得します。取得する値の詳細については、IMGSetFocus 関数を参照してください。

nReserved

NULL を指定してください。

戻り値

IMG_SUCCESS	: 正常終了
IMG_ERR_PARAMETER	: パラメータエラー
IMG_ERR_NOTINITIALIZED	: IMGInit が実行されていない
FUNCTION_UNSUPPORT	: 未サポートエラー

4. プログラミング上の注意点

4.1 基本手順

デコード、イメージキャプチャ、ストリーミング、サインキャプチャのいずれかの処理を行う際に必要な手順を以下に示します。

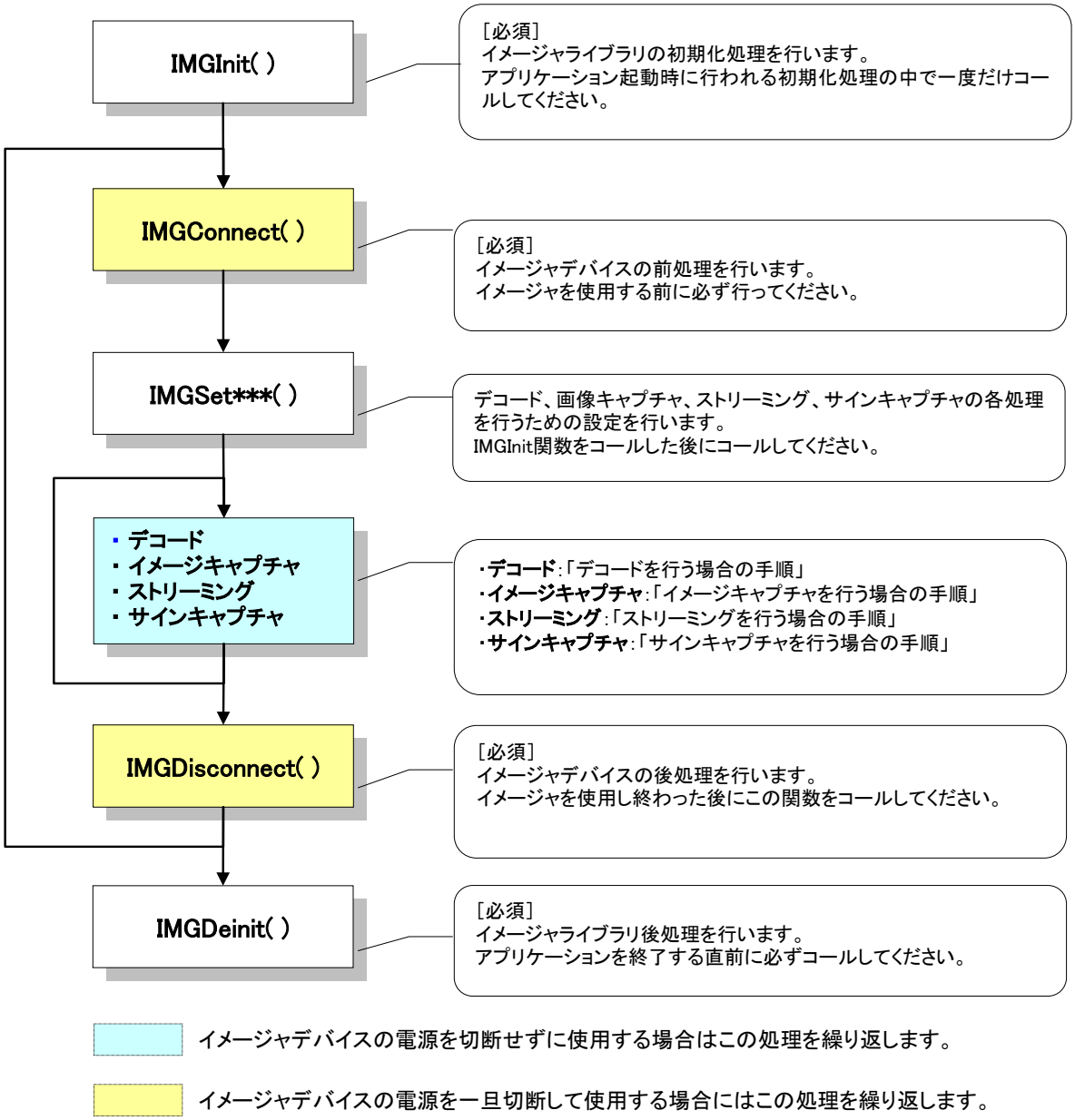


図 4.1

■デコードを行う場合の手順

デコード処理には、2 種類の処理手順があります。トリガキーが押されている間だけデコード処理を行う場合は下記の(1)を、任意のタイミングでデコード処理を行う場合は(2)を参照してください。

(1) トリガキーを監視してデコードを行う場合

1. IMGSet*** 関数(***はコードの種類を表す)をコールして、デコードを行うためのデータを設定します。デコードの対象となるコードの種類、条件などを指定します。初期状態では全てのシンボル読み取りは無効になっているため、必ず読み取り対象のシンボルを有効にしてください。

2. トリガキーを押下したタイミング等で IMGWaitForDecode または IMGWaitForDecodeRaw 関数をコールし、デコードを行います。ユーザより指定されたバッファにデコード結果を格納します。

3. デコードを途中で停止させる場合には、コールバック関数でトリガキーの押下状況を監視します。トリガキーが離されたら FALSE を返すことにより、デコード処理は中断します。

＜この例では IMGWaitForDecode 関数を使って説明します。＞

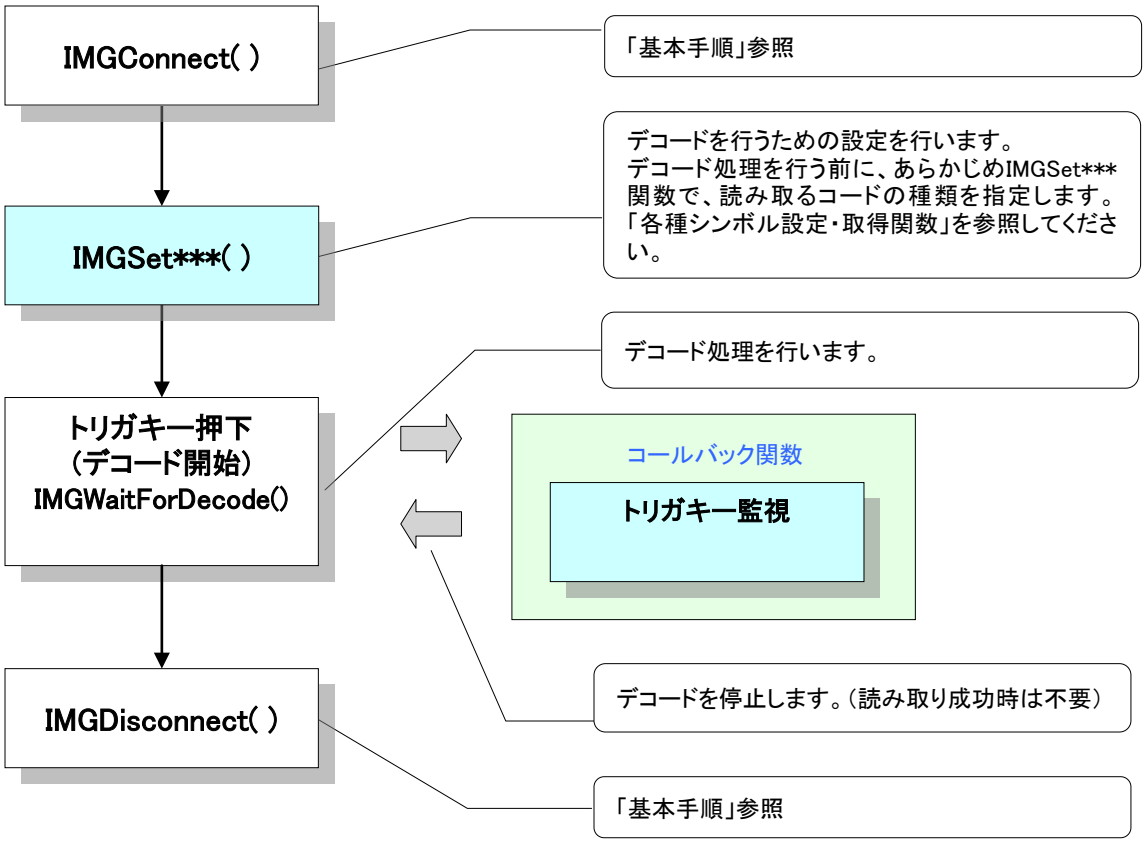


図 4.2

(2) 任意のタイミングでデコードを行う場合

1. IMGSet*** 関数(***はコードの種類を表す)をコールして、デコードを行うためのデータを設定します。デコードの対象となるコードの種類、条件などを指定します。初期状態では全てのシンボル読み取りは無効になっているため、必ず読み取り対象のシンボルを有効にしてください。

2. デコードを開始したいタイミングで IMGWaitForDecode または IMGWaitForDecodeRaw 関数をコールし、デコードを行います。ユーザより指定されたバッファにデコード結果を格納します。

3. デコードを途中で停止させる場合には、中断したいタイミングで IMGStopDecode 関数をコールします。

＜この例では IMGWaitForDecode 関数を使って説明します。＞

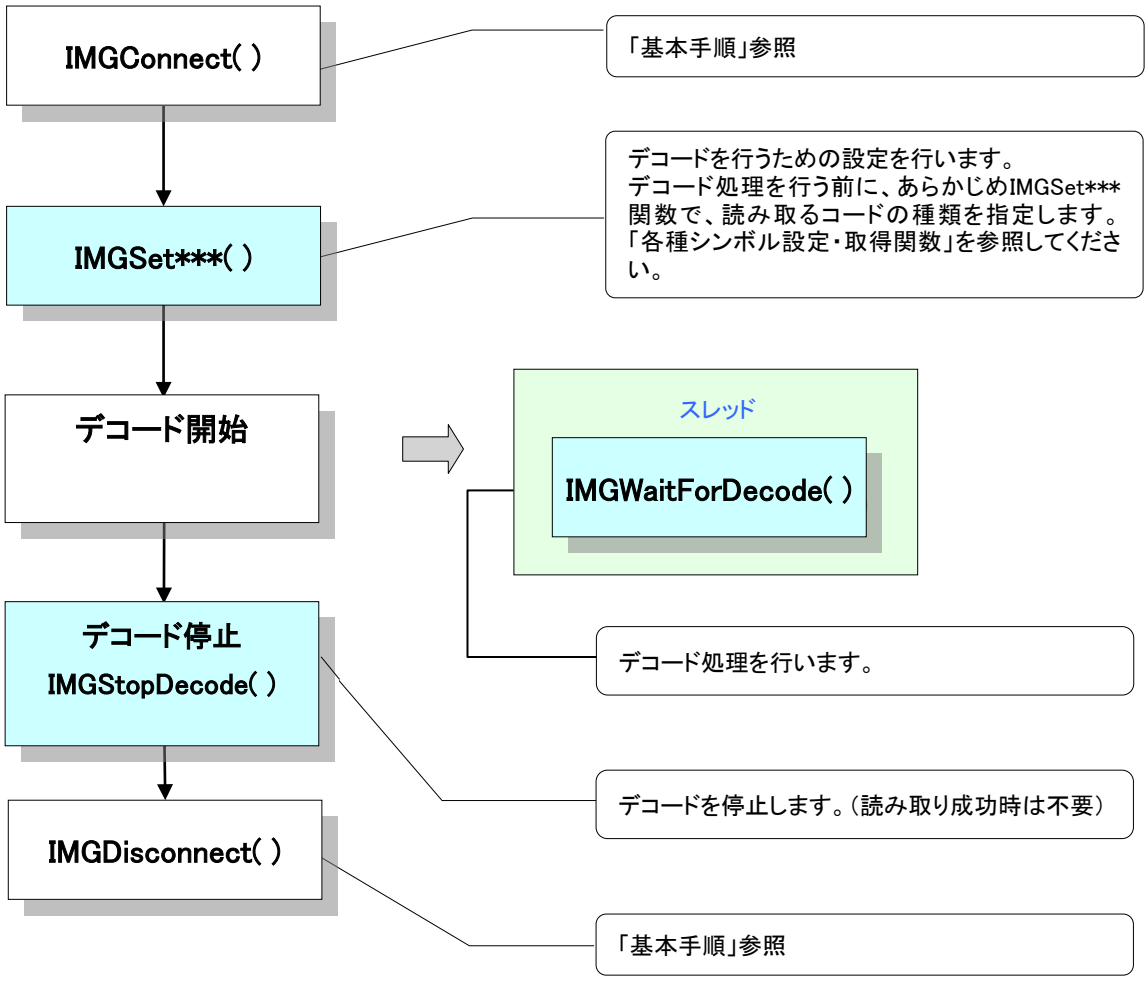


図 4.3

■イメージキャプチャを行う場合の手順

必要に応じ、IMGSet***関数にてデータを設定してください。

1. IMGGetImage 関数をコールします。ユーザより指定されたバッファにイメージデータを格納します。

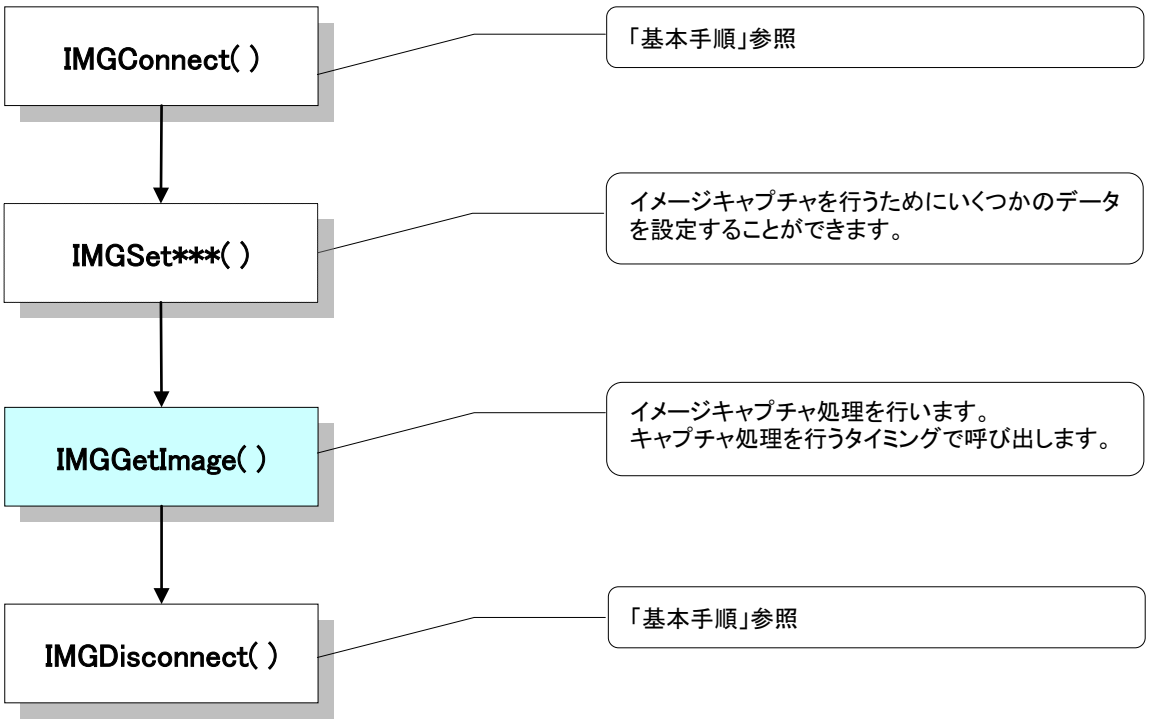


図 4.4

■ストリーミングを行う場合の手順

必要に応じて IMGSet***関数にてデータを設定してください。

1. IMGStartStream 関数をコールし、ストリーミング開始処理を行います。ストリーミングデータを格納するバッファ、画像サイズなどを指定します。イルミネーション LED を ON します。省電力モードの場合はイメージャデバイスの電源を投入します。

2. IMGGetStreamData 関数を繰り返しコールします。ユーザより指定されたバッファにストリーミングイメージデータを格納します。この関数を繰り返しコールすることにストリーミングを実現します。

3. IMGStopStream 関数をコールし、ストリーミングを停止します。イルミネーション LED を OFF します。

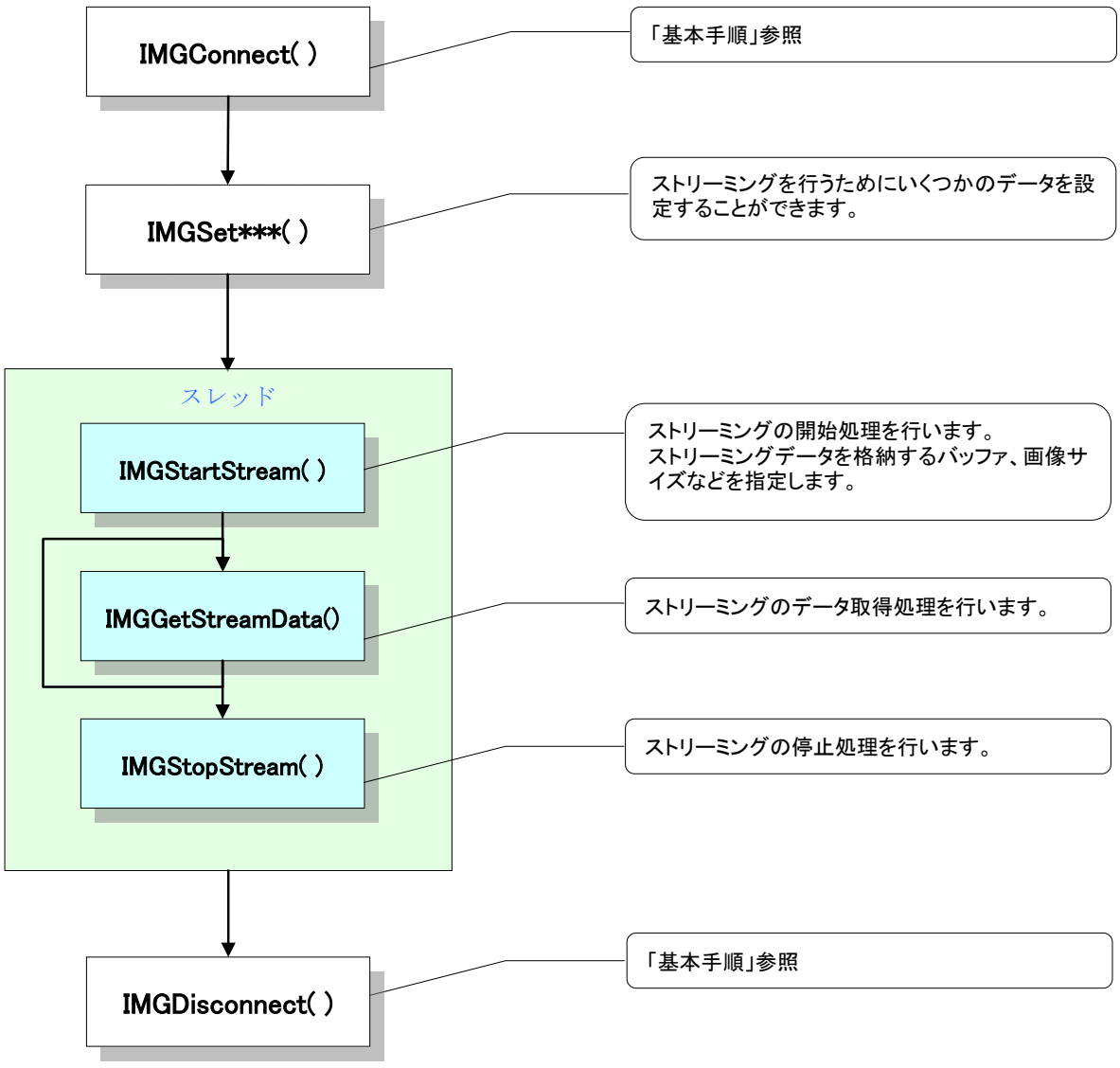


図 4.5

■サインキャプチャを行う場合の手順

1. IMGSet*** 関数(***はコードの種類を表す)をコールして、デコードを行うためのデータを設定します。デコードの対象となるシンボルコードの種類、条件などを指定します。サインキャプチャ機能をサポートしているシンボルコードは Aztec、Codabar(NW-7)、PDF417、Code128、Code39 です。

2. IMGWaitForDecode または IMGWaitForDecodeRaw 関数をコールします。ユーザより指定されたバッファにデコード結果を格納します。IMGWaitForDecode または IMGWaitForDecodeRaw 関数が正常に終了したら 3 へ進みます。ただし、上記の関数はサインキャプチャ機能でサポートしていないコードを読み取った場合でも正常に終了することがあります。必ず 1 の IMGSet***関数でサポートしているコードを指定してください。

3. IMGCaptureSign 関数をコールします。ユーザより指定されたサインエリアを切り出し、さらにそのイメージをバッファに格納します。

＜この例では IMGWaitForDecode 関数を使って説明します。＞

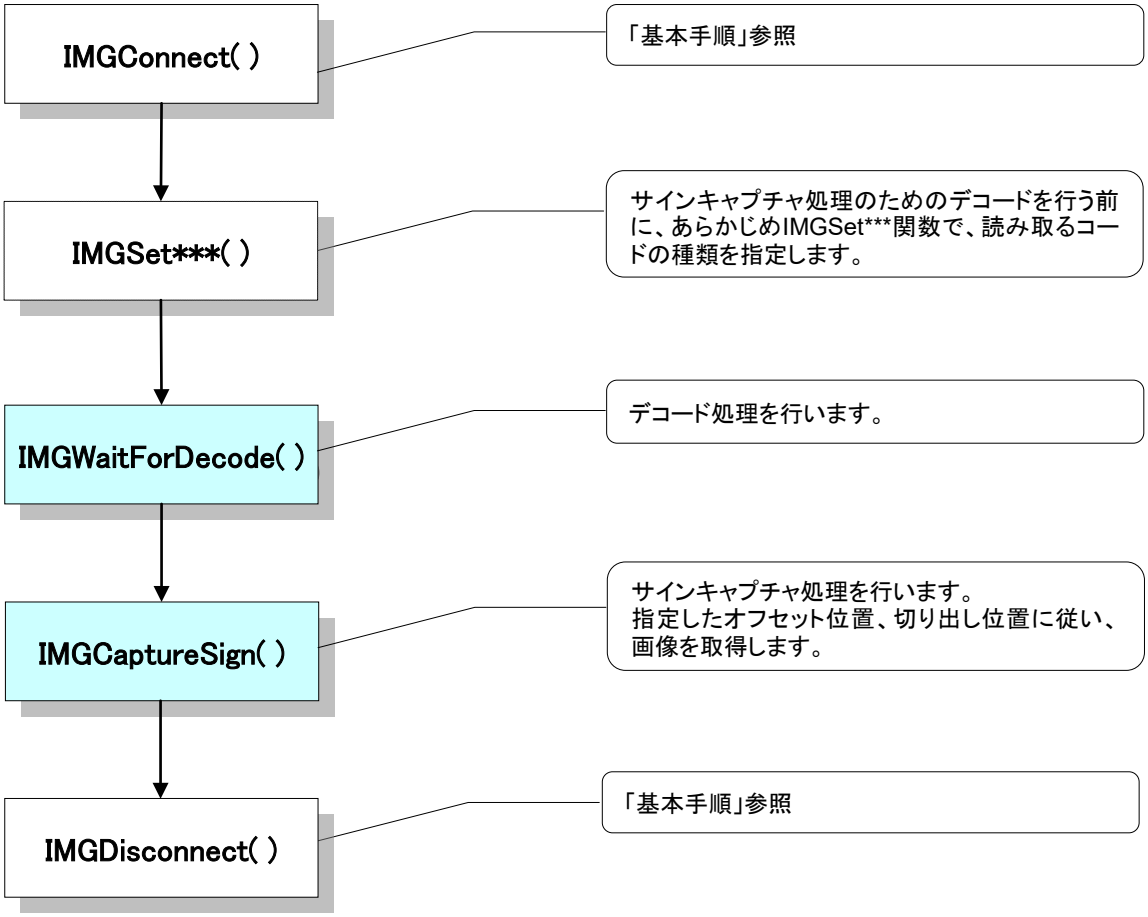


図 4.6

4.2 コード識別表

表 4-1

Symbology	Code ID		AIM ID	Possible AIM ID Modifiers
Australian Postal(4 state)	SYMID_AUSPOST	'A'	'X'	0-9,A-C
Aztec	SYMID_AZTEC	'z'	'z'	0-1
AztecMesa	SYMID_MESA	'Z'	'z'	0-9,A-C
British Postal(4 state)	SYMID_BPO	'B'	'X'	0
Canadian Postal(4 state)	SYMID_CANPOST	'C'	'X'	0
Codabar	SYMID_CODABAR	'a'	'F'	0-1
Codablock F	SYMID_CODABLOCK	'q'	'O'	0, 1, 4, 5, 6
Code11	SYMID_CODE11	'h'	'H'	0,1,3
Code128	SYMID_CODE128	'j'	'C'	0,1,2,4
Code32	SYMID_CODE32	'<'	'X'	0
Code39	SYMID_CODE39	'b'	'A'	0,1,3,4,5,7
Code49	SYMID_CODE49	'l'	'T'	0, 1, 2, 4
Code93	SYMID_CODE93	'i'	'G'	0-9, A-Z, a-m
UCC/EAN Composite	SYMID_COMPOSITE	'y'	'e'	0-3
DataMatrix	SYMID_DATAMATRIX	'w'	'd'	0-6
Dutch Postal	SYMID_DUTCHPOST	'K'	'X'	0
EAN 128	SYMID_EAN128	'I'	'C'	0,1,2,4
EAN13	SYMID_EAN13	'd'	'E'	0-4
EAN8	SYMID_EAN8	'D'	'E'	0-4
IATA 2 of 5	SYMID_IATA	'f'	'R'	0, 1, 3
ITF(Interleaved 2 of 5)	SYMID_ITF	'e'	'I'	0, 1, 3
ISBT	SYMID_ISBT	'j'	'C'	4
Japanese Postal	SYMID_JAPOST	'J'	'X'	0
Maxicode	SYMID_MAXICODE	'x'	'U'	0-3
MicroPDF	SYMID_MICROPDF	'R'	'L'	3-5
MSI	SYMID_MSI	'g'	'M'	0, 1
PDF417	SYMID_PDF417	'r'	'L'	0-2
Planet Code	SYMID_PLANET	'L'	'X'	0
Postnet	SYMID_POSTNET	'P'	'X'	0
OCR	SYMID_OCR	'O'	'o'	0-3
QR	SYMID_QR	's'	'Q'	0-6
RSS	SYMID_RSS	'y'	'e'	0
TLC39	SYMID_TLC39	'T'	'L'	2
UPC versions A	SYMID_UPCA	'c'	'E'	0-4
UPC versions E0,E1	SYMID_UPCE	'E'	'E'	0-4
Chinese Sensible Code (Han Xin)	SYMID_HX	'H'	'X'	0

デコードしたシンボルの種類を判別する場合は Code ID の値を判別してください。ただし、Code128 と ISBT、RSS と UCC/EAN Composite は同じ値が返ります。

4.3 設定ファイルの書式について

設定ファイルはテキスト形式で、以下の規則に従って記述されています。
例:
[Code39] : [(セクション名)] この(セクション名)に対するキー設定を以下に記述します。
Enable=1 : Code39 の読み取り有効／無効設定
Min=2 : Code39 の有効最小桁数の指定
Max=48 : Code39 の有効最大桁数の指定

[Codabar(NW7)] : [(次の項目名)] (次の項目名)の設定が以下に記述されます。
Enable=1

表 4-2

項目名	設定内容		
シンボル読み取り設定一覧			
Enable	シンボルの読み取り有効／無効		
Min	読み取り有効な最小桁数		
Max	読み取り有効な最大桁数		
Output Start/Stop Code	スタート・ストップコード出力有効／無効		
Read On Check Char	チェックキャラクタをもつシンボルのみ読み取り有効／無効		
Read On Check Digit			
Output Check Char	チェックキャラクタ出力有効／無効		
Output Check Digit			
Read On 2 Check Digit	2 桁のチェックキャラクタをもつシンボルのみ読み取り有効／無効		
Output Start/Stop Code	スタート／ストップコード出力有効／無効		
Full ASCII	Full ASCII 変換出力有効／無効		
Only Carrier Message	キャリアメッセージのみ出力有効／無効		
Append 2Digit Addon	2 桁のアドオンを出力有効／無効		
Append 5Digit Addon	5 桁のアドオンを出力有効／無効		
Request Addon	アドオンをもつシンボルのみ読み取り有効／無効		
Separate Addon	アドオンをスペースで区切って出力有効／無効		
Output System Number	ナンバーシステムキャラクタを出力有効／無効		
Expand Version E	UPCA 12 桁に拡張して出力有効／無効		
OCR 読み取り設定一覧(※2)			
Font	OCR 読み取りフォントの指定	0	OCR の読み取り無効
		1	OCR A
		2	OCR B
		3	US MONEY
Direction	読み取り OCR の表記される向き指定	0	左から右
		1	上から下
		2	右から左
		3	下から上
Template	テンプレートの指定	499 文字までの文字列で指定	
GroupG	ユーザ定義文字の指定	49 文字までの文字列で指定	
GroupH	ユーザ定義文字の指定	49 文字までの文字列で指定	
CheckSum	チェックサム計算方式の指定	49 文字までの文字列で指定	
PrintWeight(印字太さ調整設定)			

PrintWeight	印字太さ調整値の指定	1 から 7 の範囲で指定		
Intensity（レーザ/LED 強度設定）				
Aimer	エイミングレーザの強度指定	0 から 100 の範囲で指定		
Illumination	イルミネーション LED の強度指定	0 から 100(%)の範囲で指定		
Multi Step（多段読み設定）				
ReadMode	読み取り方式の指定	0	通常読み	
		1	多段読み	
		2	一括読み	
ReadTimes	連続読み取り回数の指定	2 から 10 の範囲で指定		
Separator	一括読み時の区切り記号指定	文字を int 型にキャストした値で指定		
Scan Mode（スキャンモード設定）		DT-X7		
Scan Mode	スキャンモードの指定			デフォルトモード
		0		屋外モード
		1		屋内（窓際）モード
		2		屋内モード
		3		倉庫モード
Decode Deliberation（デコード熟考度の設定）(※3)		DT-X7		
Decode Deliberation	熟考度の設定	100		読み取り非常に速い、コード限定
		200		読み取り速い、コードやや限定
		400		普通
		800		読み取り遅い、コードやや多い
		0		読み取り非常に遅い、コード多い
Search Mode（サーチモードの設定）※この項目のパラメータは変更しないでください。(※3)				
Search Mode	サーチモード	1		
Auto Power OFF（イメージャ APO タイマの設定）				
Auto Power OFF	イメージャ APO タイマ	0から 1800(秒)の範囲で指定。0=APO 無効、1～1800=設定時間経過後にイメージャ電源 OFF		
Indicator（読み取り通知設定）				
LED	読み取り完了時の LED 点灯指定	0：無効　1：緑有効 2：緑と赤有効		
Buzzer	読み取り成功時のブザー鳴動指定	0：無効　1：有効		
Vibrator(※4)	読み取り成功時のバイブレータ振動指定	0：無効　1：有効		

(※2)DT-X7 のみ設定可能
(※3)DT-X7 のみ設定可能
(※4)DT-X7 のみ設定可能

5. サンプルソースコード

5.1 デコード処理（トリガキーを使用する場合）

この例では、左右どちらかのトリガキーが押されるとデコードを開始し、トリガキーが離されると中断します。
デコードした結果は画面のエディットボックス上に表示されます。

```
#define   ID_HOTKEY_L 1                 // Hotkey ID of L Trigger
#define   ID_HOTKEY_R 2                 // Hotkey ID of R Trigger
#define   VKEY_TRIGGER_L  VK_F24        // Virtual Key of L Trigger
#define   VKEY_TRIGGER_R  VK_F21        // Virtual Key of R Trigger

BOOL fTrigger(VOID);

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HWND hEditMessage;
 TCHAR chCodeID, chAimID, chModifier, szDecodeMessage[ 512];
 DWORD dwLength;
 int nResult;
 BOOL ResPeek;
 MSG msg;

 switch ( message)
 {
 case WM_HOTKEY:
  switch( wParam)
  {
  case ID_HOTKEY_L:
  case ID_HOTKEY_R:
     nResult = IMGWaitForDecode( 5000, szDecodeMessage, &chCodeID, &chAimID,
         &chModifier, &dwLength, fTrigger);
     if ( nResult == IMG_SUCCESS)
     {
         SetWindowText( hEditMessage, szDecodeMessage);
     }
     break;
  }
  do
  {
     ResPeek = PeekMessage(&msg, hWnd, WM_HOTKEY, WM_HOTKEY, PM_REMOVE);
  }while(ResPeek == TRUE);
  break;
 case WM_CREATE:
  ...
  hEditMessage = CreateWindow( TEXT( "edit"), NULL,
     WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
     8, 30, 96, 40, hWnd, ( HMENU)1, hInst, NULL);
  RegisterHotKey( hWnd, ID_HOTKEY_L,  0, VKEY_TRIGGER_L);
```

```
   RegisterHotKey( hWnd, ID_HOTKEY_R,  0, VKEY_TRIGGER_R);

   IMGInit();
   IMGConnect();
   IMGSetEAN13( TRUE, FALSE, FALSE, FALSE, TRUE, TRUE);
   IMGSetDeliberation(IMG_DECODE_QUICK);
   break;
  case WM_DESTROY:
   IMGDisconnect();
   UnregisterHotKey( hWnd, ID_HOTKEY_L);
   UnregisterHotKey( hWnd, ID_HOTKEY_R);
   IMGDeinit();
   PostQuitMessage( 0);
   break;
  default:
   return DefWindowProc( hWnd, message, wParam, lParam);
  }
    return 0;
}

BOOL fTrigger( VOID)
{
  if( GetAsyncKeyState( VKEY_TRIGGER_L) < 0 || GetAsyncKeyState( VKEY_TRIGGER_R) < 0)
  {
   return TRUE;
  }
  else
  {
   return FALSE;
  }
}
```

5.2 デコード処理（任意のタイミングでデコードを中断する場合）

この例では、"Start"ボタンが押されるとデコードを開始し、"Stop"ボタンが押されると中断します。デコードした結果は画面のエディットボックス上に表示されます。

```
#define IDC_BTNSTART 101
#define IDC_BTNSTOP  102
#define VKEY_CLR VK_DELETE        // Virtual Key of CLR key

DWORD WINAPI DecodeThread( LPVOID);
HANDLE g_hDecodeEvent[ 2];
HWND g_hEditMessage;

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HWND hBtnStart, hBtnStop;
 static HANDLE hDecodeThread;
 DWORD dwID;

 switch ( message)
 {
 case WM_COMMAND:
  switch( LOWORD( wParam))
  {
  case IDC_BTNSTART:
     SetEvent( g_hDecodeEvent[ 0]);
     break;
  case IDC_BTNSTOP:
     IMGStopDecode();
     break;
  }
  break;
 case WM_CREATE:
  ...
  g_hEditMessage = CreateWindow( TEXT( "edit"), NULL,
     WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
     8, 30, 96, 40, hWnd, ( HMENU)1, hInst, NULL);
  hBtnStart = CreateWindow( TEXT("button"), TEXT("Start"),
     WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
     8, 80, 40, 20, hWnd, ( HMENU)IDC_BTNSTART, hInst, NULL);
  hBtnStop = CreateWindow( TEXT("button"), TEXT("Stop"),
     WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
     64, 80, 40, 20, hWnd, ( HMENU)IDC_BTNSTOP, hInst, NULL);
  g_hDecodeEvent[ 0] = CreateEvent( NULL, FALSE, FALSE, TEXT("StartDecode"));
  g_hDecodeEvent[ 1] = CreateEvent( NULL, FALSE, FALSE, TEXT("Quit"));
  hDecodeThread = CreateThread( NULL, 0, DecodeThread, NULL, 0, &dwID);

  IMGInit();
  IMGConnect();
IMGSetEAN13( TRUE, FALSE, FALSE, FALSE, TRUE, TRUE);
```

```
    IMGSetDeliberation(IMG_DECODE_QUICK);
    break;
  case WM_DESTROY:
    SetEvent( g_hDecodeEvent[ 1]);
    IMGDisconnect();
    IMGDeinit();
    WaitForSingleObject( hDecodeThread, INFINITE);
    CloseHandle( hDecodeThread);
    CloseHandle( g_hDecodeEvent[ 1]);
    CloseHandle( g_hDecodeEvent[ 0]);
    PostQuitMessage( 0);
    break;
  default:
    return DefWindowProc( hWnd, message, wParam, lParam);
  }
    return 0;
}

DWORD WINAPI DecodeThread( LPVOID pParameter)
{
  BOOL bDecodeLoop = TRUE;
  DWORD dwObjectNum, dwLength;
  int nResult;
  TCHAR chCodeID, chAimID, chModifier, szDecodeMessage[ 512];

  while( bDecodeLoop)
  {
    dwObjectNum = WaitForMultipleObjects( 2, g_hDecodeEvent, FALSE, INFINITE);
    switch( dwObjectNum)
    {
    case WAIT_OBJECT_0:
       nResult = IMGWaitForDecode( 20000, szDecodeMessage, &chCodeID, &chAimID,
           &chModifier, &dwLength, NULL);
       if ( nResult == IMG_SUCCESS)
       {
           SetWindowText( g_hEditMessage, szDecodeMessage);
       }
       break;
    case ( WAIT_OBJECT_0+1):
       bDecodeLoop = FALSE;
       break;
    }
  }
  ExitThread( 0);
  return 0;
}
```

5.3 多段読み処理

この例では、多段読み機能を使って 3 個のシンボルをデコードし、それぞれの結果をエディットボックスに表示しています。トリガキーが押されている間にデコードを行います。

```
#define   ID_HOTKEY_L 1           // Hotkey ID of L Trigger
#define   ID_HOTKEY_R 2           // Hotkey ID of R Trigger
#define VKEY_TRIGGER_L VK_F24 // Virtual Key of L Trigger
#define VKEY_TRIGGER_R VK_F21 // Virtual Key of R Trigger
#define NUMBER_OF_SYMBOL 3

BOOL fTrigger( VOID);

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HWND hEditMessage[ NUMBER_OF_SYMBOL];
 HDC hdc;
 PAINTSTRUCT ps;
 TCHAR chCodeID, chAimID, chModifier, szDecodeMessage[ 512];
 DWORD dwLength;
 int nResult, i;
 BOOL ResPeek;
 MSG msg;

 switch (message)
 {
 case WM_HOTKEY:
  switch( wParam)
  {
  case ID_HOTKEY_L:
  case ID_HOTKEY_R:
     for( i = 0; i < NUMBER_OF_SYMBOL; i++)
     {
         nResult = IMGWaitForDecode( 5000, szDecodeMessage, &chCodeID, &chAimID,
             &chModifier, &dwLength, fTrigger);
         if ( nResult == IMG_SUCCESS)
         {
             SetWindowText( hEditMessage[ i], szDecodeMessage);
         }
         else
         {
             break;
         }
     }
     break;
  default:
     break;
  }
  do
  {
```

```
      ResPeek = PeekMessage(&msg , hWnd, WM_HOTKEY, WM_HOTKEY, PM_REMOVE);
   }while(ResPeek == TRUE);
   break;
  case WM_CREATE:
   ...
   hEditMessage[ 0] = CreateWindow( TEXT( "edit"), NULL,
      WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
      8, 28, 112, 34, hWnd, ( HMENU)1, hInst, NULL);
   hEditMessage[ 1] = CreateWindow( TEXT( "edit"), NULL,
      WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
      8, 66, 112, 34, hWnd, ( HMENU)1, hInst, NULL);
   hEditMessage[ 2] = CreateWindow( TEXT( "edit"), NULL,
      WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
      8, 104, 112, 34, hWnd, ( HMENU)1, hInst, NULL);
   RegisterHotKey( hWnd, ID_HOTKEY_L,  0, VKEY_TRIGGER_L);
   RegisterHotKey( hWnd, ID_HOTKEY_R,  0, VKEY_TRIGGER_R);

   IMGInit();
   IMGConnect();
IMGSetEAN13( TRUE, FALSE, FALSE, FALSE, TRUE, TRUE);
   IMGSetDecodeMode( IMG_DECODEMODE_MULTISTEP, 3, 0);
   IMGSetDeliberation(IMG_DECODE_QUICK);

   break;
  case WM_DESTROY:
   IMGDisconnect();
   UnregisterHotKey( hWnd, ID_HOTKEY_L);
   UnregisterHotKey( hWnd, ID_HOTKEY_R);
   IMGDeinit();
   PostQuitMessage( 0);
   break;
  default:
   return DefWindowProc( hWnd, message, wParam, lParam);
  }
    return 0;
}

BOOL fTrigger( VOID)
{
  if( GetAsyncKeyState( VKEY_TRIGGER_L) < 0 || GetAsyncKeyState( VKEY_TRIGGER_R) < 0)
  {
   return TRUE;
  }
  else
  {
   return FALSE;
  }
}
```

5.4 一括読み処理

この例では、一括読み機能を使って 2 個のシンボルをデコードし、それぞれの結果をエディットボックスに表示しています。"Start"ボタンが押されるとデコードを開始し、"Stop"ボタンが押されると中断します。

```
#define IDC_BTNSTART 101
#define IDC_BTNSTOP  102
#define VKEY_CLR VK_DELETE            // Virtual Key of CLR Key

DWORD WINAPI DecodeThread( LPVOID);
HANDLE g_hDecodeEvent[ 2];
HWND g_hEditMessage1, g_hEditMessage2;

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HWND hBtnStart, hBtnStop;
 static HANDLE hDecodeThread;
 DWORD dwID;

 switch ( message)
 {
 case WM_COMMAND:
  switch( LOWORD( wParam))
  {
  case IDC_BTNSTART:
     SetEvent( g_hDecodeEvent[ 0]);
     break;
  case IDC_BTNSTOP:
     IMGStopDecode();
     break;
  }
  break;
 case WM_CREATE:
  ...
  g_hEditMessage1 = CreateWindow( TEXT( "edit"), NULL,
     WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
     8, 30, 112, 40, hWnd, ( HMENU)1, hInst, NULL);
  g_hEditMessage2 = CreateWindow( TEXT( "edit"), NULL,
     WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
     8, 74, 112, 40, hWnd, ( HMENU)1, hInst, NULL);
  hBtnStart = CreateWindow( TEXT("button"), TEXT("Start"),
     WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
     8, 118, 48, 20, hWnd, ( HMENU)IDC_BTNSTART, hInst, NULL);
  hBtnStop = CreateWindow( TEXT("button"), TEXT("Stop"),
     WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON,
     72, 118, 48, 20, hWnd, ( HMENU)IDC_BTNSTOP, hInst, NULL);

  g_hDecodeEvent[ 0] = CreateEvent( NULL, FALSE, FALSE, TEXT("StartDecode"));
  g_hDecodeEvent[ 1] = CreateEvent( NULL, FALSE, FALSE, TEXT("StopDecode"));
  hDecodeThread = CreateThread( NULL, 0, DecodeThread, NULL, 0, &dwID);
```

```
    IMGInit();
    IMGConnect();
    IMGSetEAN13( TRUE, FALSE, FALSE, FALSE, TRUE, TRUE);
    IMGSetDecodeMode( IMG_DECODEMODE_PACKAGE, 2, ',');
    break;
  case WM_DESTROY:
    SetEvent( g_hDecodeEvent[ 1]);
    IMGDisconnect();
    IMGDeinit();
    WaitForSingleObject( hDecodeThread, INFINITE);
    CloseHandle( hDecodeThread);
    CloseHandle( g_hDecodeEvent[ 1]);
    CloseHandle( g_hDecodeEvent[ 0]);
    PostQuitMessage( 0);
    break;
  default:
    return DefWindowProc( hWnd, message, wParam, lParam);
  }
    return 0;
}

DWORD WINAPI DecodeThread( LPVOID pParameter)
{
  BOOL bDecodeLoop = TRUE;
  DWORD dwObjectNum, dwLength;
  int nResult;
  TCHAR chCodeID, chAimID, chModifier, szDecodeMessage[ 512];

  while( bDecodeLoop)
  {
    dwObjectNum = WaitForMultipleObjects( 2, g_hDecodeEvent, FALSE, INFINITE);
    switch( dwObjectNum)
    {
    case WAIT_OBJECT_0:
       nResult = IMGWaitForDecode( 20000, szDecodeMessage, &chCodeID, &chAimID,
           &chModifier, &dwLength, NULL);
       if ( nResult == IMG_SUCCESS)
       {
           PTCHAR pToken;
           pToken = wcstok( szDecodeMessage, TEXT(","));
           SetWindowText( g_hEditMessage1, pToken);
           pToken = wcstok( NULL, TEXT(","));
           SetWindowText( g_hEditMessage2, pToken);
       }
       break;
    case ( WAIT_OBJECT_0+1):
       bDecodeLoop = FALSE;
       break;
    }
  }
  ExitThread( 0);
  return 0;
```

```
}
```

5.5 イメージキャプチャ・ストリーミング

この例では、プログラムが起動されるとすぐにストリーミング表示が開始されます。ストリーミング表示中にトリガキーが押されるとQVGAサイズ(320x240 pixels)の画像をキャプチャし、ビットマップファイルとして保存します。

```
DWORD WINAPI CaptureThread( HWND);
HANDLE g_hCaptureEvent[ 2];
HWND g_hEditMessage;
HPALETTE g_hPalette;

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HANDLE hCaptureThread;
 HDC hdc;
 PAINTSTRUCT ps;
 DWORD dwID;

 switch ( message)
 {
 case WM_KEYDOWN:
  switch( LOWORD( wParam))
  {
  case VK_F21:                    // Trigger R key
  case VK_F24:                    // Trigger L key
     SetEvent( g_hCaptureEvent[ 0]);
     break;
  case VK_DELETE:                      // CLR key
     PostMessage( hWnd, WM_DESTROY, 0, 0);
     break;
  }
  break;
 case WM_CREATE:
  ...
  g_hCaptureEvent[ 0] = CreateEvent( NULL, FALSE, FALSE, TEXT("Capture"));
  g_hCaptureEvent[ 1] = CreateEvent( NULL, FALSE, FALSE, TEXT("Quit"));
  IMGInit();
  IMGConnect();
  hCaptureThread = CreateThread( NULL, 0,
     ( LPTHREAD_START_ROUTINE)CaptureThread, ( LPVOID)hWnd, 0, &dwID);
  break;
 case WM_DESTROY:
  SetEvent( g_hCaptureEvent[ 1]);
  IMGDisconnect();
  IMGDeinit();
  WaitForSingleObject( hCaptureThread, INFINITE);
  CloseHandle( hCaptureThread);
  CloseHandle( g_hCaptureEvent[ 1]);
  CloseHandle( g_hCaptureEvent[ 0]);
  PostQuitMessage( 0);
```

```
   break;
  default:
   return DefWindowProc( hWnd, message, wParam, lParam);
  }
    return 0;
}

HPALETTE SetPalette( void);
void ShowImage( HWND hWnd, LPBYTE ImageBuffer, int dwWidth, int dwHeight );

DWORD WINAPI CaptureThread( HWND hWnd)
{
  BOOL bCaptureLoop = TRUE;
  DWORD dwObjectNum, dwCaptureSize, dwStreamSize;
  int nResult;
  LPBYTE pImageBuffer, pStreamBuffer;

  pStreamBuffer = (LPBYTE)VirtualAlloc( NULL, 128 *  96, MEM_COMMIT, PAGE_READWRITE);
  pImageBuffer  = (LPBYTE)VirtualAlloc( NULL, 320 * 240, MEM_COMMIT, PAGE_READWRITE);
  // カラーパレット作成
  g_hPalette = SetPalette();
  IMGStartStream( 64, 48, 640-64, 480-48, 4, IMAGE_256MONO);

  while( bCaptureLoop)
  {
   dwObjectNum = WaitForMultipleObjects( 2, g_hCaptureEvent, FALSE, 2);
   switch( dwObjectNum)
   {
   case WAIT_OBJECT_0:        // Trigger key pushed
          IMGStopStream();
          nResult = IMGGetImage( pImageBuffer, &dwCaptureSize, 0, 0,
              640, 480, 2, IMAGE_256MONO, 200);
          if ( nResult == IMG_SUCCESS)
          {
              IMGMakeImageFile(   TEXT("¥¥My Documents¥¥IMGSample.bmp"),
pImageBuffer,dwCaptureSize,320,240,IMG_FORMAT_BMP);
}
          IMGStartStream( 64, 48, 640-64, 480-48, 4, IMAGE_256MONO);
          ResetEvent( g_hCaptureEvent[ 0]);
      break;
   case ( WAIT_OBJECT_0+1):       // C key pushed
      bCaptureLoop = FALSE;
      break;
   default:
      nResult = IMGGetStreamData( pStreamBuffer, &dwStreamSize);
      if ( nResult == IMG_SUCCESS)
      {
          ShowImage( hWnd, pStreamBuffer, 128, 96 );
      }
      break;
   }
  }
  IMGStopStream();
```

```
  VirtualFree( pImageBuffer , 0, MEM_RELEASE);
  VirtualFree( pStreamBuffer, 0, MEM_RELEASE);
  // カラーパレット破棄
  DeleteObject( g_hPalette);

  ExitThread( 0);
  return 0;
}
HPALETTE SetPalette( void )
{
  int i;
  LOGPALETTE *lpPal;
  HPALETTE hp;

  lpPal = ( LOGPALETTE *)VirtualAlloc( NULL,
    sizeof(WORD)*2 + 256*sizeof(PALETTEENTRY), MEM_COMMIT, PAGE_READWRITE);

  lpPal->palVersion = 0x300;
  lpPal->palNumEntries = 256;

  // 256 階調グレースケールカラーパレット作成
  for( i=0; i<256; i++ )
  {
    lpPal->palPalEntry[i].peRed   = i;
    lpPal->palPalEntry[i].peGreen = i;
    lpPal->palPalEntry[i].peBlue  = i;
  }
  hp = CreatePalette( lpPal );
  VirtualFree( lpPal, 0, MEM_RELEASE);

  return hp;
}

void ShowImage( HWND hWnd, LPBYTE ImageBuffer, int dwWidth, int dwHeight )
{
  HBITMAP     hBmp, hOrgBmp;
  HDC         hdcMem, hdc;

  hdc     = GetDC( hWnd );

  hBmp    = CreateBitmap( dwWidth, dwHeight, 1, 8, ImageBuffer ); // 8bit Bitmap
  hdcMem  = CreateCompatibleDC( hdc );

  // カラーパレット指定
  SelectPalette( hdcMem, g_hPalette, FALSE );
  RealizePalette( hdcMem );

  hOrgBmp = ( HBITMAP )SelectObject( hdcMem, hBmp );

  BitBlt( hdc, 60, 100, dwWidth, dwHeight, hdcMem, 0, 0, SRCCOPY );

  SelectObject( hdcMem, hOrgBmp );
  DeleteObject( hBmp );
```

```
  DeleteDC( hdcMem );

  ReleaseDC( hWnd, hdc );
}
```

5.6 サインキャプチャ

この例では、トリガキーが押されるとシンボルのデコードを行い、そのシンボルの右下にある画像を取得して表示します。
以下の各コードのサインキャプチャが行えます。

表 5-1

Code128	PDF417	Aztec
		

```
#define   ID_HOTKEY_L 1          // Hotkey ID of L Trigger
#define   ID_HOTKEY_R 2          // Hotkey ID of R Trigger
#define VKEY_TRIGGER_L VK_F24 // Virtual Key of L Trigger
#define VKEY_TRIGGER_R VK_F21 // Virtual Key of R Trigger
#define VKEY_CLR VK_ESCAPE

HPALETTE g_hPalette;

HPALETTE SetPalette( void );
BOOL fTrigger( VOID);
VOID ShowImage( HWND , LPBYTE , int , int);
VOID LowerGradation( LPBYTE, const LPBYTE, DWORD);

LRESULT CALLBACK WndProc( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)
{
 static HWND hEditMessage;
 HDC hdc;
 PAINTSTRUCT ps;
 TCHAR chCodeID, chAimID, chModifier, szDecodeMessage[ 512];
 DWORD dwLength;
 int nResult;
 static LPBYTE pSignCaptureBuffer;
 static DWORD dwWidth, dwHeight, dwFormat, dwAspectRatio;
 static int nResolution, nOffsetX, nOffsetY;
 BOOL ResPeek;
 MSG msg;

 switch (message)
 {
 case WM_KEYDOWN:
  switch( LOWORD( wParam))
  {
```

```
 case VKEY_CLR:
    PostMessage( hWnd, WM_DESTROY, 0, 0);
    break;
 }
 break;

case WM_HOTKEY:
 switch( wParam)
 {
 case ID_HOTKEY_L:
 case ID_HOTKEY_R:
    nResult = IMGWaitForDecode( 5000, szDecodeMessage, &chCodeID, &chAimID,
        &chModifier, &dwLength, fTrigger);
    if ( nResult == IMG_SUCCESS)
    {
        SetWindowText( hEditMessage, szDecodeMessage);
        if ( chCodeID == SYMID_PDF417)
        {
            dwAspectRatio = 2;      // element height/element width
        }
        else if ( chCodeID == SYMID_AZTEC)
        {
            dwAspectRatio = 1;      // element height/element width = 1
        }
        else
        {
            dwAspectRatio = 30;     // bar height/Narrow bar width
        }
        nResult = IMGCaptureSign( pSignCaptureBuffer, dwAspectRatio,
             nOffsetX, nOffsetY, dwWidth, dwHeight, nResolution, dwFormat);
        if ( nResult == IMG_SUCCESS)
        {
            ShowImage( hWnd, pSignCaptureBuffer,
                dwWidth * nResolution, dwHeight * nResolution);
        }
    }
    break;
 default:
    break;
 }
 do
 {
    ResPeek = PeekMessage(&msg , hWnd, WM_HOTKEY, WM_HOTKEY, PM_REMOVE);
 }while(ResPeek == TRUE);
 break;

case WM_CREATE:
 ...
 hEditMessage = CreateWindow( TEXT( "edit"), NULL,
    WS_CHILD | WS_VISIBLE | WS_BORDER | ES_MULTILINE,
    8, 30, 112, 20, hWnd, ( HMENU)1, hInst, NULL);
 RegisterHotKey( hWnd, ID_HOTKEY_L,  0, VKEY_TRIGGER_L);
 RegisterHotKey( hWnd, ID_HOTKEY_R,  0, VKEY_TRIGGER_R);
```

```
  nOffsetX      = 33;
  nOffsetY      = 63;
  dwWidth       = 60;
  dwHeight      = 40;
  nResolution   = 2;
  dwFormat      = IMAGE_256MONO;
  pSignCaptureBuffer = ( LPBYTE)VirtualAlloc( NULL,
     dwWidth * dwHeight * nResolution * nResolution, MEM_COMMIT, PAGE_READWRITE);

  // カラーパレット作成
  g_hPalette = SetPalette();
  IMGInit();
  IMGConnect();
  IMGSetCode39( TRUE, 2, 20, FALSE, FALSE, FALSE, FALSE);
  IMGSetCode128( TRUE, 1, 56);
  IMGSetCodabar( TRUE, 2, 24, FALSE, FALSE, FALSE);
  IMGSetPDF417( TRUE, 1, 500);
  IMGSetAztec( TRUE, 1, 500);
  break;

 case WM_PAINT:
  hdc = BeginPaint( hWnd, &ps);
  EndPaint( hWnd, &ps);
  break;

 case WM_DESTROY:
  IMGDisconnect();
  VirtualFree( pSignCaptureBuffer, 0, MEM_RELEASE);
  UnregisterHotKey( hWnd, ID_HOTKEY_L);
  UnregisterHotKey( hWnd, ID_HOTKEY_R);
  IMGDeinit();
  // カラーパレット破棄
  DeleteObject( g_hPalette);
  CommandBar_Destroy( hwndCB);
  PostQuitMessage( 0);
  break;

 default:
  return DefWindowProc( hWnd, message, wParam, lParam);
 }
  return 0;
}

BOOL fTrigger( VOID)
{
 if( GetAsyncKeyState( VKEY_TRIGGER_L) < 0 || GetAsyncKeyState( VKEY_TRIGGER_R) < 0)
 {
  return TRUE;
 }
 else
 {
  return FALSE;
```

```
  }
}

HPALETTE SetPalette( void)
{
  int i;
  LOGPALETTE *lpPal;
  HPALETTE hp;

  lpPal = ( LOGPALETTE *)VirtualAlloc( NULL,
    sizeof(WORD)*2 + 256*sizeof(PALETTEENTRY), MEM_COMMIT, PAGE_READWRITE);

  lpPal->palVersion = 0x300;
  lpPal->palNumEntries = 256;

  // 256 階調グレースケールカラーパレット作成
  for( i=0; i<256; i++)
  {
    lpPal->palPalEntry[i].peRed   = i;
    lpPal->palPalEntry[i].peGreen = i;
    lpPal->palPalEntry[i].peBlue  = i;
  }
  hp = CreatePalette( lpPal );
  VirtualFree( lpPal, 0, MEM_RELEASE);

     return hp;
}

VOID ShowImage( HWND hWnd, LPBYTE ImageBuffer, int dwWidth, int dwHeight )
{
  HBITMAP     hBmp, hOrgBmp;
  HDC         hdcMem, hdc;

  hdc     = GetDC( hWnd );

  hBmp    = CreateBitmap( dwWidth, dwHeight, 1, 8, ImageBuffer ); // 8bit Bitmap
  hdcMem  = CreateCompatibleDC( hdc );

  // カラーパレット指定
  SelectPalette( hdcMem, g_hPalette, FALSE );
  RealizePalette( hdcMem );

  hOrgBmp = ( HBITMAP )SelectObject( hdcMem, hBmp );

  BitBlt( hdc, 60, 100, dwWidth, dwHeight, hdcMem, 0, 0, SRCCOPY );

  SelectObject( hdcMem, hOrgBmp );
  DeleteObject( hBmp );
  DeleteDC( hdcMem );

  ReleaseDC( hWnd, hdc );
}
```

カシオ計算機お問い合わせ窓口

●製品サポートサイト

https://casio.jp/support/ht/

カシオ計算機株式会社

〒151-8543　東京都渋谷区本町 1-6-2

TEL 03-5334-4638(代)