CASIO®

システムライブラリ
マニュアル

このマニュアルは、システムライブラリの
仕様について記載します。

Ver. 1.18

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	2008.12	29	SysCheckIOBOX 関数のパラメータ説明を修正
1.02	2009.03	2	開発環境とプログラミング言語の対応表を訂正
		8	SysGetSystemName 関数の文字列サイズを変更
		10	SysSetBootup 関数のパラメータ説明を訂正
		17	SysSetPowerOnKeyTime 関数のパラメータ説明を訂正
		104	SysSetOneKeyLock 関数のパラメータ説明を訂正
		108	SysSetInputMode 関数のパラメータ説明を修正
		109	SysGetInputMode 関数のパラメータ説明を修正
		117	SysGetNormalUserDefineKey 関数の宣言の誤記を訂正
1.03	2009.07	－	対象機種に DT-5300 を追加
1.04	2010.01	110	SysSetNormalUserDefineKey 関数のパラメータ説明を追加
1.05	2010.05	63	SysGetLED 関数のパラメータ説明を追加
		68	SysSetLEDState 関数のパラメータ説明を追加
1.06	2011.01	－	対象機種に IT-300 および DT-X8 を追加
		57	SysMultiTouchOn 関数を追加
		58	SysMultiTouchOff 関数を追加
		59	SysGetMultiTouchState 関数を追加
1.07	2011.05	110	キーID 一覧表の訂正
		81	SysSetBuzzerVolume 関数のパラメータ説明を追加
		83	SysSetBuzzerMute 関数のパラメータ説明を追加
		147	SysConvertGestureMessage 関数を追加
1.08	2011.11	－	対象機種に IT-9000 を追加
1.09	2013.09	85	SysSetCPUMode 関数のパラメータ説明を修正
1.10	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
		85	SysSetCPUMode 関数のパラメータ説明を修正
		110	SysSetNormalUserDefineKey 関数にキーID を追加
1.11	2015.02	－	対象機種に DT-X100 および DT-X200 を追加
1.12	2016.02	85	SysSetCPUMode 関数の補足を修正
1.13	2016.05	90	SysSetBLBattery 関数のパラメータ説明を修正
		92	SysSetBLExpower 関数のパラメータ説明を修正
1.14	2016.06	108	SysSetInputMode 関数のパラメータ説明を修正
		141	SysSetEnableKeyMode 関数のパラメータ説明を修正
1.15	2017.05	11	SysSetBootup 関数の注意を追加
		25	SysDisableAPO 関数の補足を追加
1.16	2017.08	25	SysDisableAPO 関数の補足を修正
		109	SysGetInputMode 関数の補足を追加
1.17	2019.03	29	SysCheckIOBOX 関数の戻り値を修正
1.18	2019.11	109	SysGetInputMode 関数の戻り値および補足を修正
		110-112	SysSetNormalUserDefineKey 関数のキーID 一覧表を修正

目次

1. 概要

システムライブラリは、キー制御/電源制御など、携帯情報端末のシステム動作に対する、補助/拡張機能を提供します。

システムクラスライブラリは、システムライブラリを .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。

システム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。

システムライブラリでは、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。
システム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。

※ システムライブラリは、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

システム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-9800 / DT-5300 / IT-300 / 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 Compact 7
- Microsoft WindowsMobile 6.5
- Microsoft WindowsMobile 6.5.3
- 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 + SP1	○	○

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

Visual Basic または Visual C# の場合

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

※ 関数の戻り値として記載されている TRUE と FALSE に対しては、定数が定義されていません。戻り値の TRUE は -1、FALSE は 0 と読み替えてください。

クラス定義の詳細については、Microsoft Visual Studio で SystemLibNet.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 SysGetModelName

機種情報を取得します。

```
[C++]
DWORD SysGetModelName(
  DWORD *dwModel,
  DWORD *dwVersion,
  DWORD *dwPlatform
)
```

```
[Visual Basic]
Public Shared Function SysGetModelName( _
  ByRef dwModel As Int32, _
  ByRef dwVersion As Int32, _
  ByRef dwPlatform As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetModelName(
  ref Int32 dwModel,
  ref Int32 dwVersion,
  ref Int32 dwPlatform
);
```

解説

使用中の HT の機種、OS バージョン、プラットフォームバージョンを取得します。

パラメータ

dwModel

機種をあらわす値を取得します。値と機種の関係は以下のとおりです。

DT-9800	: 4
DT-5200	: 7
DT-X7	: 9
DT-5300CE	: 12
DT-5300WM	: 13
IT-300	: 15
DT-X8	: 16
IT-9000CE	: 19
IT-9000WM	: 20
IT-G500WEC	: 21
IT-G500WEH	: 22
DT-X100	: 23
DT-X200	: 25
デバイスエミュレータ	: 100

dwVersion
OS ビルド ID を取得します。

dwPlatform
WindowsCE バージョンを取得します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.2 SysGetDeviceIDCode

DeviceID を取得します。

```
[C++]
DWORD SysGetDeviceIDCode(
  TCHAR *pdwDevID
)
```

```
[Visual Basic]
Public Shared Function SysGetDeviceIDCode( _
  ByVal pdwDevID As Char() _
) As Int32
```

```
[C#]
public static Int32 SysGetDeviceIDCode(
  Char[] pdwDevID
);
```

解説

EEPROM に書き込まれている DeviceID を取得します。

パラメータ

pdwDevID

DeviceID をあらわす UUID を取得します。
UUID は、最大 32 文字の文字列で返される場合があります。必ず 33 文字以上の領域を確保して本関数呼び出してください。また呼び出し前には、確保した領域を0でクリアしてください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.3 SysGetUserIDCode

UserID を取得します。

```
[C++]
DWORD SysGetUserIDCode(
  DWORD *pwUserID
)
```

```
[Visual Basic]
Public Shared Function SysGetUserIDCode( _
  ByRef pwUserID As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetUserIDCode(
  ref Int32 pwUserID
);
```

解説

EEPROM に書き込まれている UserID を取得します。

パラメータ

pwUserID
ユーザ ID を取得します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.4 SysGetSystemName

機種により異なる以下の名前およびポート名を取得します。

- Flash フォルダ名
- CF1 カードフォルダ名
- CF2 カードフォルダ名
- SD カードフォルダ名
- Bluetooth クライアントシリアルポート名
- Bluetooth サーバーシリアルポート名
- Bluetooth ダイアルアップポート名
- Bluetooth LAN ポート名
- IrDA LawCOM ポート名
- USB シリアル COM ポート名
- シリアル COM ポート 1 名
- シリアル COM ポート 2 名
- カードシリアル COM ポート 1 名
- カードシリアル COM ポート 2 名
- レーザー設定ファイル保存パス
- イメージャ設定ファイル保存パス

```
[C++]
DWORD SysGetSystemName(
  DWORD   dwValueID,
  TCHAR   *lpValueName
)
```

```
[Visual Basic]
Public Shared Function SysGetSystemName( _
  ByVal dwValueID As Int32 _
  ByVal lpValueName As Char() _
) As Int32
```

```
[C#]
public static Int32 SysGetSystemName(
  Int32 dwValueID,
  Char[] lpValueName
);
```

解説

機種により異なるフォルダ名やポート名を取得します。

パラメータ

dwValueID

取得したい ID を設定します。

STORAGE_Name_Flash : Flash フォルダ名
STORAGE_Name_CF1 : CF1 カードフォルダ名
STORAGE_Name_CF2 : CF2 カードフォルダ名
STORAGE_Name_SD : SD カードフォルダ名

BT_ClientSerial	:	Bluetooth クライアントシリアルポート名
BT_ServerSerial	:	Bluetooth サーバーシリアルポート名
BT_DaialUP	:	Bluetooth ダイアルアップポート名
BT_LAN	:	Bluetooth LAN ポート名
IrDA_Law	:	IrDA RawCOM ポート名
USB_Serial	:	USB シリアル COM ポート名
Serial_Port1	:	シリアル COM ポート 1 名
Serial_Port2	:	シリアル COM ポート 2 名
Card_Serial1	:	カードシリアル COM ポート 1 名
Card_Serial2	:	カードシリアル COM ポート 2 名
OBRIni_FilePath	:	レーザー設定ファイル保存パス
IMGIni_FilePath	:	イメージャ設定ファイル保存パス

lpValueName

ID で指定されたフォルダ名、ポート名または、保存パスが格納されます。
メモリは、80 文字分を確保してください。また、機種により対応されていないフォルダ名、ポート名または、保存パスが指定された場合は、関数は「正常終了」を返しますが、文字は入りません。

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.5 SysSetBootup

Wakeup 要因による電源 ON の有効/無効を設定します。

```
[C++]
DWORD SysSetBootup(
  DWORD BootMode
)
```

```
[Visual Basic]
Public Shared Function SysSetBootup( _
  ByVal BootMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBootup(
  Int32 BootMode
);
```

解説

Wakeup 要因による電源 ON の有効/無効を設定します。本体 OFF 状態で Wakeup 要因が発生することにより、本体電源を ON します。本関数で、電源 ON 動作を制御することができます。
Device Emulator 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、SysGetBootup 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

BootMode

以下の電源 ON の有効/無効状態を、以下の値の論理和で指定します。

表 3-1 設定値一覧

設定値	設定内容	DT-5200	DT-X7	DT-9800	DT-5300	IT-300	DT-X8	IT-9000	IT-G500
BOOT_MULTI	マルチキーによる電源 ON 有効								
BOOT_LEFT	左キーによる電源 ON 有効								
BOOT_RIGHT	右キーによる電源 ON 有効								
BOOT_LEFTTRIGGER	左トリガーキーによる電源 ON 有効	○	○	○	○	○	○	○	○
BOOT_RIGHTTRIGGER	右トリガーキーによる電源 ON 有効	○	○	○	○	○	○	○	○
BOOT_IOBOX	IOBOX 載せによる電源 ON 有効	○	○	○	○	○	○	○	○
BOOT_NONE	Wakeup 要因による電源 ON 無効	○	○	○	○	○	○	○	○
BOOT_PGBUTTON	プログラムキーによる電源 ON 有効								
BOOT_APL1	アプリケーションキー1 による電源 ON 有効								
BOOT_APL2	アプリケーションキー2による電源 ON 有効								
BOOT_APL3	アプリケーションキー3による電源 ON 有効								
BOOT_APL4	アプリケーションキー4による電源 ON 有効								
BOOT_APL5	アプリケーションキー5による電源 ON 有効								
BOOT_APL6	アプリケーションキー6による電源 ON 有効								

BOOT_USB	USB 接続による電源オン有効								
BOOT_CFCARD	CF カード挿入による電源オン有効								
BOOT_GUNTRIGGER	ガントリガーキーによる電源オン有効	○							
BOOT_CENTERTRIGGER	センタートリガーキーによる電源オン有効	○	○		○	○	○		○

設定値	設定内容	DT-X100	DT-X200
BOOT_MULTI	マルチキーによる電源 ON 有効		
BOOT_LEFT	左キーによる電源 ON 有効		
BOOT_RIGHT	右キーによる電源 ON 有効		
BOOT_LEFTTRIGGER	左トリガーキーによる電源 ON 有効	○	○
BOOT_RIGHTTRIGGER	右トリガーキーによる電源 ON 有効	○	○
BOOT_IOBOX	IOBOX 載せによる電源 ON 有効	○	○
BOOT_NONE	Wakeup 要因による電源 ON 無効	○	○
BOOT_PGBUTTON	プログラムキーによる電源 ON 有効		
BOOT_APL1	アプリケーションキー1 による電源 ON 有効		
BOOT_APL2	アプリケーションキー2による電源 ON 有効		
BOOT_APL3	アプリケーションキー3による電源 ON 有効		
BOOT_APL4	アプリケーションキー4による電源 ON 有効		
BOOT_APL5	アプリケーションキー5による電源 ON 有効		
BOOT_APL6	アプリケーションキー6による電源 ON 有効		
BOOT_USB	USB 接続による電源オン有効		
BOOT_CFCARD	CF カード挿入による電源オン有効		
BOOT_GUNTRIGGER	ガントリガーキーによる電源オン有効		
BOOT_CENTERTRIGGER	センタートリガーキーによる電源オン有効	○	○

デフォルトの設定値は以下のとおりです。
BOOT_NONE

戻り値

TRUE　　：正常終了
FALSE　　：内部エラー
SYS_PARAMERR　　：パラメータエラー
FUNCTION_UNSUPPORT　　：未サポートエラー

注意

IT-G500 では、BOOT_LEFTTRIGGER、BOOT_RIGHTTRIGGER、BOOT_CENTERTRIGGER を個別に設定することはできません。いずれかを設定した場合は、すべてのトリガーキーが有効になります。

3.6 SysGetBootup

Wakeup 要因による電源 ON の有効/無効を取得します。

```
[C++]
DWORD SysGetBootup(
  DWORD *pBootMode
)
```

```
[Visual Basic]
Public Shared Function SysGetBootup( _
  ByRef pBootMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBootup(
  ref Int32 pBootMode
);
```

解説

Wakeup 要因による電源 ON の有効/無効状態を取得します。

パラメータ

pBootMode

電源 ON の有効/無効状態を取得します。取得する値は SysSetBootup 関数を参照してください。

戻り値

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

3.7 SysSetOffMaskTime

電源 ON 後、電源 OFF を禁止する時間を設定します。

```
[C++]
DWORD SysSetOffMaskTime(
  DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetOffMaskTime( _
  ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOffMaskTime(
  Int32 dwTime
);
```

解説

電源 ON 後、電源 OFF を禁止する時間を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetOffMaskTime 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

dwTime
電源 ON 後の電源キーによる OFF 禁止時間(sec)を指定
します。デフォルトの設定値は以下のとおりです。
■DT-5200 / DT-X7 / DT-5300 / IT-300 / DT-X8 / IT-9000
10 秒
■DT-9800
5 秒

パラメータの部分にデフォルトの値より小さい値を設定し
ないでください。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.8 SysGetOffMaskTime

電源 ON 後、電源 OFF を禁止する時間を取得します。

```
[C++]
DWORD SysGetOffMaskTime(
  DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetOffMaskTime( _
  ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetOffMaskTime(
  ref Int32 pdwTime
);
```

解説

電源 ON 後、電源 OFF を禁止する時間を取得します。

パラメータ

pdwTime

電源 ON 後の電源キーによる OFF 禁止時間を取得します。取得する値は SysSetOffMaskTime 関数を参照してください。

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.9 SysSetStorageOffMaskTime

ストレージ使用後、電源 OFF を禁止する時間を設定します。

```
[C++]
DWORD SysSetStorageOffMaskTime(
  DWORD dwStorage,
  DWORD dwStorageType
)
```

```
[Visual Basic]
Public Shared Function SysSetStorageOffMaskTime ( _
  ByVal dwStorage As Int32 _
  ByVal dwStorageType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetStorageOffMaskTime (
  Int32 dwStorage
  Int32 dwStorageType
);
```

解説

ストレージ使用後、電源 OFF を禁止する時間を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetStorageOffMaskTime 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

dwStorage
ストレージ使用後の電源キーによる OFF 禁止時間を秒単位で設定します。

dwStorageType
ストレージの種類を設定します。

STORAGE_MASK_CF	: CF カード
STORAGE_MASK_SD	: SD カード
STORAGE_MASK_FD	: フラッシュディスク

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.10 SysGetStorageOffMaskTime

ストレージ使用後、電源 OFF を禁止する時間を取得します。

```
[C++]
DWORD SysGetOffMaskTime(
  DWORD *pdwStorage,
  DWORD *pdwStorageType
)
```

```
[Visual Basic]
Public Shared Function SysGetOFFMaskTime ( _
  ByRef pdwStorage As Int32 _
  ByRef pdwStorageType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetOFFMaskTime (
  ref Int32 pdwStorage
  ref Int32 pdwStorageType
);
```

解説

ストレージ使用後、電源 OFF を禁止する時間を取得します。

パラメータ

pdwStorage
ストレージ使用後の電源キーによる OFF 禁止時間を秒単位で取得します。

pdwStorageType
ストレージの種類を取得します。
取得される値は SysSetStorageOffMaskTime 関数を参照してください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.11 SysSetPowerOnKeyTime

電源 ON するまでの電源ボタン連続押し時間を設定します。

```
[C++]
DWORD SysSetPowerOnKeyTime(
  DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetPowerOnKeyTime( _
  ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetPowerOnKeyTime(
  Int32 dwTime
);
```

解説

電源 ON するまでの電源ボタン連続押し時間を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetPowerOnKeyTime 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

dwTime
電源 ON するまでの電源ボタン連続押し時間(msec)を、250msec 単位で指定します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.12 SysGetPowerOnKeyTime

電源 ON するまでの電源ボタン連続押し時間を取得します。

```
[C++]
DWORD SysGetPowerOnKeyTime(
  DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetPowerOnKeyTime( _
  ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetPowerOnKeyTime(
  ref Int32 pdwTime
);
```

解説

電源 ON するまでの電源ボタン連続押し時間を取得します。

パラメータ

pdwTime
電源 ON するまでの電源ボタン連続押し時間(msec)を取得します。取得する値は SysSetPowerOnKeyTime 関数を参照してください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.13 SysSetPowerOffKeyTime

電源 OFF するまでの電源ボタン連続押し時間を設定します。

```
[C++]
DWORD SysSetPowerOffKeyTime(
  DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetPowerOffKeyTime( _
  ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetPowerOffKeyTime(
  Int32 dwTime
);
```

解説

電源 OFF するまでの電源ボタン連続押し時間を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetPowerOffKeyTime 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

dwTime
電源 OFF するまでの電源ボタン連続押し時間(msec)を、250msec 単位で指定します。
設定可能な値は、250msec ～ 1750msec です。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.14 SysGetPowerOffKeyTime

電源 OFF するまでの電源ボタン連続押し時間を取得します。

```
[C++]
DWORD SysGetPowerOffKeyTime(
  DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetPowerOffKeyTime( _
  ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetPowerOffKeyTime(
  ref Int32 pdwTime
);
```

解説

電源 OFF するまでの電源ボタン連続押し時間を取得します。

パラメータ

pdwTime
電源 OFF するまでの電源ボタン連続押し時間(msec)を取得します。取得する値は SysSetPowerOffKeyTime 関数を参照してください。

戻り値

TRUE	： 正常終了
FLASE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.15 SysPowerOff

本体電源を OFF します。

```
[C++]
DWORD SysPowerOff()
```

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

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

解説

本体電源を OFF します。
Device Emulator では、何も動作しません。

パラメータ

なし

戻り値

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

3.16 SysDisablePowerOff

電源ボタンによる本体電源 OFF を禁止します。

```
[C++]
DWORD SysDisablePowerOff()
```

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

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

解説

電源ボタンによる本体電源 OFF を禁止します。本体電源 OFF を禁止した場合、電源ボタンを押されたときに WM_POWERBROADCAST の PBT_APMSUSPEND を発生します。アプリケーションは、このメッセージを取得して必要なオフ処理を行った後、ソフト OFF してください。
Device Emulator 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、SysGetPowerOff 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

本設定は、電源 ON 時に解除されますので、必要に応じて再設定してください。
電池蓋開けや緊急電源 OFF は、本設定をした後でもすぐに OFF します。

3.17 SysEnablePowerOff

電源スイッチによる本体電源 OFF を許可します。

```
[C++]
DWORD SysEnablePowerOff()
```

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

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

解説

電源スイッチによる本体電源 OFF を許可します。
Device Emulator 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、
SysGetPowerOff 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.18 SysGetPowerOff

電源スイッチによる本体電源 OFF の許可/禁止状態を取得します。

```
[C++]
DWORD SysGetPowerOff()
```

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

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

解説

電源スイッチによる本体電源 OFF の許可/禁止状態を取得します。

パラメータ

なし

戻り値

TRUE	： 電源 OFF 禁止
FALSE	： 電源 OFF 許可　デフォルト値
FUNCTION_UNSUPPORT	： 未サポートエラー

3.19 SysDisableAPO

オートパワーオフを無効にします。

```
[C++]
DWORD SysDisableAPO()
```

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

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

解説

オートパワーオフを無効にします。本関数を実行すると、コントロールパネルの設定に関わらず、オートパワーオフが無効になります。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetAPO 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

本関数を実行すると、機種によりオートバックライトオフも無効になります。

IT-G500	： オートバックライトオフは無効にならない※
他	： オートバックライトオフは無効になる

IT-G500 のオートバックライトオフについては、コントロールパネルの明るさ（CE モデル）またはバックライト（WM モデル）から設定してください。

3.20 SysEnableAPO

オートパワーオフを有効にします。

```
[C++]
DWORD SysEnableAPO()
```

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

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

解説

オートパワーオフを有効にします。この関数を実行すると、オートパワーオフはコントロールパネルの設定通りに動作します。
Device Emulator 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、SysGetAPO 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

本関数を実行しても、他のプロセスやスレッドが SysDisableAPO 関数を実行しているときは、APO は有効になりません。APO を有効にするためには、SysDisableAPO 関数を実行した回数分、本関数を実行する必要があります。

3.21 SysGetAPO

オートパワーオフの有効/無効状態を取得します。

```
[C++]
DWORD SysGetAPO()
```

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

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

解説

オートパワーオフの有効/無効状態を取得します。

パラメータ

なし

戻り値

TRUE	: APO 無効
FALSE	: APO 有効　デフォルト値
FUNCTION_UNSUPPORT	: 未サポートエラー

3.22 SysSoftReset

システムをリセットします。

```
[C++]
DWORD SysSoftReset()
```

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

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

解説

システムをリセットします。
Device Emulator では、何も動作しません。

パラメータ

なし

戻り値

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

補足

本関数を実行すると、ただちにシステムがリセットされるため、書き込み中のファイルやデータが失われる可能性があります。オープン中のファイルやデバイスは全てクローズしてから実行してください。

3.23 SysCheckIOBOX

本体と IO ボックスとの接続状態を取得します

```
[C++]
DWORD SysCheckIOBOX(
  DWORD time_out
)
```

```
[Visual Basic]
Public Shared Function SysCheckIOBOX( _
  ByVal time_out As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysCheckIOBOX(
  Int32 time_out
);
```

解説

本体と IO ボックスとの接続状態を取得します。タイムアウト時間を設定することにより、IO ボックスの接続を監視することも可能です。

パラメータ

time_out

接続状態監視時間を指定します。

200～3600000　：接続状態監視時間(msec) ※

INFINITE　：監視時間無限大（タイムアウト無し）

※ 監視時間に 200 未満の値を指定した場合は、200 に切り上げます

戻り値

0　：接続検出

1　：タイムアウト発生

FUNCTION_UNSUPPORT　：未サポートエラー

補足

本関数を連続して使用する場合は、最低 1 秒以上の間隔をあけてください。

3.24 SysCheckCharger

本体とバッテリーチャージャの接続状態を取得します。

```
[C++]
DWORD SysCheckCharger(
  DWORD time_out
)
```

```
[Visual Basic]
Public Shared Function SysCheckCharger( _
  ByVal time_out As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysCheckCharger(
  Int32 time_out
);
```

解説

本体とバッテリーチャージャの接続状態を取得します。タイムアウト時間を設定することにより、バッテリーチャージャの接続を監視することも可能です。

パラメータ

time_out

接続状態監視時間を指定します。

200～3600000	：	接続状態監視時間(msec) ※
INFINITE	：	監視時間無限大（タイムアウト無し）

※ 監視時間に 200 未満の値を指定した場合は、200 に切り上げます

戻り値

0	：	接続検出
1	：	タイムアウト発生
FUNCTION_UNSUPPORT	：	未サポートエラー

補足

本関数を連続して使用する場合は、最低 1 秒以上の間隔をあけてください。

3.25 SysSetBackupLife

RAM バックアップ時間を設定します。

```
[C++]
DWORD SysSetBackupLife(
  DWORD dwSetBackupLife
)
```

```
[Visual Basic]
Public Shared Function SysSetBackupLife( _
  ByVal dwSetBackupLife As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBackupLife(
  Int32 dwSetBackupLife
);
```

解説

VDET1 発生後の RAM バックアップ時間を設定します。NORMAL_LIFE の場合は、約 1.5 日バックアップされます。LONG_LIFE の場合は、約 3 日パックアップされます。

パラメータ

dwSetBackupLife

RAM バックアップ時間を指定します。

LONG_LIFE	:	RAM バックアップ約 3 日
NORMAL_LIFE	:	RAM バックアップ約 1.5 日

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.26 SysGetBackupLife

RAM バックアップ時間を取得します。

```
[C++]
DWORD SysGetBackupLife()
```

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

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

解説

VDET1 発生後の RAM バックアップ時間の設定値を取得します。

パラメータ

なし

戻り値

LONG_LIFE	:	RAM バックアップ約 3 日
NORMAL_LIFE	:	RAM バックアップ約 1.5
FUNCTION_UNSUPPORT	:	未サポートエラー

3.27 SysSetWakeOn

モジュールの WakeOn 機能有効/無効を設定します。

```
[C++]
DWORD SysSetWakeOn(
  DWORD dwWakeOnModules
)
```

```
[Visual Basic]
Public Shared Function SysSetWakeOn( _
  ByVal dwWakeOnModules As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetWakeOn(
  Int32 dwWakeOnModules
);
```

解説

DT-9800 の一部のモジュールには WakeOn が備わっています。本関数により、この WakeOn 機能の有効/無効を制御することができます。

パラメータ

dwWakeOnModules

WakeOn 機能の有効/無効を指定します。

WAKEON_SERIAL14PIN	： シリアル 14 ピンの WakeOn 機能有効化
WAKEON_NON	： WakeOn 機能無効化

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.28 SysGetWakeOn

モジュールの WakeOn 機能有効/無効設定状態を取得します。

```
[C++]
DWORD SysGetWakeOn(
  DWORD *pdwWakeOnModules
)
```

```
[Visual Basic]
Public Shared Function SysGetWakeOn( _
  ByRef pdwWakeOnModules As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetWakeOn(
  ref Int32 pdwWakeOnModules
);
```

解説

DT-9800 の一部のモジュールには WakeOn が備わっています。
本関数により、この WakeOn 機能の有効/無効を制御することができます。

パラメータ

pdwWakeOnModules
WakeOn 機能の有効/無効を取得します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.29 SysSetVirtualOffMode

擬似 OFF 機能の有効/無効を設定します。

```
[C++]
DWORD SysSetVirtualOffMode(
  DWORD dwVirtualOffMode
)
```

```
[Visual Basic]
Public Shared Function SysSetVirtualOffMode( _
  ByVal dwVirtualOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetVirtualOffMode(
  Int32 dwVirtualOffMode
);
```

解説

擬似 OFF 機能とは、電源 ON 後瞬時に SS 無線接続がされたように見せるために、「見た目は OFF しているが、中は動作していて SS 無線は接続されたまま」という状態を作り出すものです。
Device Emulator 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、SysGetVirtualOffMode 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

dwVirtualOffMode
擬似 OFF 機能の有効/無効を指定します。

VIRTUALOFF_ENABLE	: 擬似 OFF 有効
VIRTUALOFF_DISABLE	: 擬似 OFF 無効（デフォルト）

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
SYS_PARAMERR	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

以下の手順で関数を呼び出すと、端末が擬似 OFF 状態になります。

1. SysSetVirtualOffMode 関数を使用して擬似 OFF を有効に設定
2. SysDisablePowerOff 関数を使用して電源キーを無効に設定、または SysSetVirtualOffModeEx 関数を使用（詳細は下記参照）

上記の設定を行うと、電源ドライバは電源キー押下により以下のメッセージを発行するようになります。

■ 電源 OFF 時
WM_POWERBROADCAST（0x0218）の PBT_APMSUSPEND（0x0004）
（通常時、本メッセージは電源ドライバより発行されません。擬似 OFF 設定時のみ発行されます。）

■ 電源 ON 時
WM_POWERBROADCAST の PBT_APMRESUMESUSPEND（0x0007）

上記のシステムメッセージの処理内で、以下の設定を行ってください。

■ SysDisablePowerOff 関数を使用した場合
以下の設定を個別に行ってください。
- ディスプレイ電源制御
- キー入力制御
- 音声制御
- ブザー制御
- バイブレータ制御
- LED 制御

■ SysSetVirtualOffModeEx 関数を使用した場合
擬似 OFF 有効／無効

ただし、キー入力制御に関しては、SysSetAllKeyLock 関数で無効にできないキーがあります。電源キーについては SysDisablePowerOff 関数を、トリガーキーについては OBRClose 関数や IMGDeinit 関数をそれぞれ実行することにより、無効になります。
また、擬似 OFF 中に VDET1 が発生した場合、無線 LAN の電源が ON の状態では、電池が即座になくなり、VDET2 が発生します。そのため、VDET1 発生時に発行する WM_POWERBROADCAST の PBT_APMBATTERYLOW（0x0009）のメッセージを受け取ったら、SysPowerOff 関数を使用して、本体電源を OFF にすることを推奨します。
擬似 OFF の設定状態は、VIRTUALOFF_ENABLE を複数回設定しても、VIRTUALOFF_DISABLE を 1 回実行するだけで解除できます。

3.30 SysSetVirtualOffModeEx

疑似オフ状態設定(タッチパネル、画面、キー動作、APO の禁止、電源 OFF 禁止、CPU スピード設定)の有効無効を設定します。

```
[C++]
DWORD SysSetVirtualOffModeEx(
  DWORD dwVirtualOFFMode
)
```

```
[Visual Basic]
Public Shared Function SysSetVirtualOffModeEx( _
  ByVal dwVirtualOFFMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetVirtualOffModeEx(
  Int32 dwVirtualOFFMode
);
```

解説

疑似オフ状態設定(タッチパネル、画面、キー動作、APO の禁止、電源 OFF 禁止、CPU スピード設定)の有効無効を設定します。SysSetVirtualOffMode 関数と合わせて利用し、擬似オフ状態の制御を行います。
Device Emulator 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、SysGetVirtualOffMode 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

dwVirtualOFFMode

擬似 OFF 機能の有効/無効を指定します。

VIRTUALOFF_ENABLE	:	擬似 OFF 有効
VIRTUALOFF_DISABLE	:	擬似 OFF 無効(デフォルト)

VIRTUALOFF_ENABLE に設定するとの以下状態にします。VIRTUALOFF_DISABLE に設定すると設定前の状態に戻します。

タッチパネル	:	禁止
画面	:	OFF
キー動作	:	禁止
APO 禁止	:	禁止
電源 OFF 禁止	:	禁止
CPU スピード	:	Low 設定

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.31 SysGetVirtualOffMode

擬似 OFF 機能の有効/無効を取得します。

```
[C++]
DWORD SysGetVirtualOffMode(
  DWORD *pdwVirtualOffMode
)
```

```
[Visual Basic]
Public Shared Function SysGetVirtualOffMode( _
  ByRef pdwVirtualOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetVirtualOffMode(
  ref Int32 pdwVirtualOffMode
);
```

解説

擬似 OFF 機能の有効/無効設定状態を取得します。

パラメータ

pdwVirtualOffMode
擬似 OFF 機能の有効/無効を取得します。

状態	SysSetVirtualOffMode	SysSetVirtualOffModeEx
VIRTUALOFF_ENABLE	○	○
VIRTUALOFF_DISABLE	－	－
VIRTUALOFF_ENABLE_NOT_EX	○	－
VIRTUALOFF_EX_ENABLE	－	○

戻り値

TRUE　：正常終了
FALSE　：内部エラー
FUNCTION_UNSUPPORT　：未サポートエラー

3.32 SysSetSystemManagedVirtualOffMode

システム制御擬似 OFF の有効無効を設定します。

```
[C++]
DWORD SysSetSystemManagedVirtualOffMode(
  DWORD dwSysVOffMode
)
```

```
[Visual Basic]
Public Shared Function SysSetSystemManagedVirtualOffMode( _
  ByVal dwSysVOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetSystemManagedVirtualOffMode(
  Int32 dwSysVOffMode
);
```

解説

システム制御擬似 OFF の有効無効を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetSystemManagedVirtualOffMode 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

dwSysVOffMode

システム擬似 OFF 機能の有効/無効を指定します。

VIRTUALOFF_ENABLE	：	システム制御擬似 OFF 機能有効
VIRTUALOFF_DISABLE	：	システム制御擬似 OFF 機能無効（デフォルト）

戻り値

TRUE	：	正常終了
FALSE	：	内部エラー
SYS_PARAMERR	：	パラメータエラー
FUNCTION_UNSUPPORT	：	未サポートエラー

3.33 SysGetSystemManagedVirtualOffMode

システム制御擬似 OFF の有効無効を取得します。

```
[C++]
DWORD SysGetSystemManagedVirtualOffMode(
  DWORD *pdwSysVOffMode
)
```

```
[Visual Basic]
Public Shared Function SysGetSystemManagedVirtualOffMode( _
  ByRef pdwSysVOffMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetSystemManagedVirtualOffMode(
  ref Int32 pdwSysVOffMode
);
```

解説

システム制御擬似 OFF の有効/無効を取得します。

パラメータ

pdwSysVOffMode
システム擬似 OFF 機能の有効/無効を取得します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.34 SysDisplayOff

擬似 OFF のために、表示を OFF します。

```
[C++]
DWORD SysDisplayOff()
```

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

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

解説

擬似 OFF のために、表示を OFF します。
Device Emulator 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、
SysGetDisplayPowerState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.35 SysDisplayOn

擬似 OFF のために、表示を ON します。

```
[C++]
DWORD SysDisplayOn()
```

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

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

解説

擬似 OFF のために、表示を ON します。
Device Emulator 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、
SysGetDisplayPowerState 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

なし

補足

以下の順番で設定を行なうと、擬似 OFF 状態になります。

1. SysSetVirtualOffMode 関数で擬似 OFF 有効
2. SysDisablePowerOff 関数で電源キー無効

上記設定を行なうと、電源ドライバは電源キー押下により以下のメッセージを発行します。（通常の電源 OFF/ON 時に発行されるメッセージと同一）

■電源 OFF 時
WM_POWERBROADCAST（0x0218）の PBT_APMSUSPEND（0x0004）
■電源 ON 時
WM_POWERBROADCAST の PBT_APMRESUMESUSPEND（0x0007）

このメッセージをトリガーに、SysSetVirtualOffMode/SysSetAllKeyLock を使用して表示の OFF/ON とキー入力の無効を行ないます。ただし、Mult とトリガーキーは SysSetAllKeyLock で無効にできません。
Multi キーを無効にする場合は、SysSetNormalUserDefineKey で Multi キーにアプリケーションで使用しないキーを設定し、クリック音を OFF にします。擬似 OFF 中にリセット操作をすると、次回立ち上げ時の Multi キーの設定が、擬似 OFF 中のキー設定のままになります。そのため擬似 OFF を使用する場合は、アプリケーションの起動時にデフォルトの Multi キーを設定するようにしてください。
トリガーキーを無効にするには、OBR またはイメージャの Close 処理を行ないます。
擬似 OFF 中に VDET1 が発生した場合、無線 LAN の電源を入れたままにしておくと、電池がすぐになくなって VDET2 が発生します。そのため、VDET1 発生時に発行する WM_POWERBROADCAST の PBT_APMBATTERYLOW（0x0009）のメッセージを受け取ったら、SysPowerOFF を使って本体電源を OFF することを推奨します。
Vibrator/Buzzer/LED については、各アプリケーションで制御できるので擬似 OFF 中は、動作させないようにしてください。

3.36 SysGetDisplayPowerState

表示電源のオン、オフ状態を取得します。

```
[C++]
DWORD SysGetDisplayPowerState()
```

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

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

解説

表示電源のオン、オフ状態を取得します。

パラメータ

なし

戻り値

表示電源の状態を表す値

TRUE	： 表示 ON
FALSE	： 表示 OFF
FUNCTION_UNSUPPORT	： 未サポートエラー

3.37 SysTouchPanelOn

擬似 OFF のために、タッチパネルを ON します。

```
[C++]
DWORD SysTouchPanelOn()
```

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

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

解説

擬似 OFF のために、タッチパネルを ON します。
Device Emulator 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、
SysGetTouchPanelState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.38 SysTouchPanelOff

擬似 OFF のために、タッチパネルを OFF します。

```
[C++]
DWORD SysTouchPanelOff()
```

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

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

解説

擬似 OFF のために、タッチパネルを OFF します。
Device Emulator 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、
SysGetTouchPanelState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.39 SysGetTouchPanelState

タッチパネルのオン／オフ状態を取得します。

```
[C++]
DWORD SysGetTouchPanelState()
```

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

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

解説

タッチパネルのオン／オフ状態を取得します。

パラメータ

なし

戻り値

タッチパネルのオン／オフ状態を表す値

TRUE	：タッチパネル ON
FALSE	：タッチパネル OFF
FUNCTION_UNSUPPORT	：未サポートエラー

3.40 SysSetEmulateMouseState

マウスエミュレータの有効／無効を設定します。

```
[C++]
DWORD SysSetEmulateMouseState(
  DWORD   dwEmulateMouseState
)
```

```
[Visual Basic]
Public Shared Function SysSetEmulateMouseState( _
  ByVal dwEmulateMouseState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetEmulateMouseState(
  Int32 dwEmulateMouseState
);
```

解説

マウスエミュレータの有効／無効を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetEmulateMouseState 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

dwEmulateMouseState

EMUMOUSE_ENABLE	：	マウスエミュレータ有効
EMUMOUSE_DISABLE	：	マウスエミュレータ無効（デフォルト）

戻り値

TRUE	：	正常終了
FALSE	：	内部エラー
FUNCTION_UNSUPPORT	：	未サポートエラー

3.41 SysGetEmulateMouseState

マウスエミュレータの有効／無効を取得します。

```
[C++]
DWORD SysGetEmulateMouseState(
  DWORD   *pdwEmulateMouseState
)
```

```
[Visual Basic]
Public Shared Function SysGetEmulateMouseState( _
  ByRef pdwEmulateMouseState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetEmulateMouseState(
  ref Int32 pdwEmulateMouseState
);
```

解説

マウスエミュレータの有効／無効を取得します。

パラメータ

pdwEmulateMouseState
マウスエミュレータの有効／無効を取得します。

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.42 SysAudioOff

擬似 OFF のために、オーディオを OFF します。

```
[C++]
DWORD SysAudioOff()
```

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

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

解説

擬似 OFF のために、オーディオを OFF します。
Device Emulator 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、
SysGetAudioPowerState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.43 SysAudioOn

擬似 OFF のために、オーディオを ON します。

```
[C++]
DWORD SysAudioOn()
```

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

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

解説

擬似 OFF のために、オーディオを ON します。
Device Emulator 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、
SysGetAudioPowerState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.44 SysGetAudioPowerState

オーディオのオン／オフ状態を取得します。

```
[C++]
DWORD SysGetAudioPowerState(void)
```

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

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

解説

オーディオのオン／オフ状態を取得します。

パラメータ

なし

戻り値

オーディオのオン／オフ状態を表す値

TRUE	：オーディオ ON
FALSE	：オーディオ OFF
FUNCTION_UNSUPPORT	：未サポートエラー

3.45 SysKeyBackLightOn

キーバックライトを ON します。

```
[C++]
DWORD SysKeyBackLightOn()
```

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

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

解説

本関数は、キーバックライトを ON します。
Device Emulator 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、
SysGetKeyBackLightState 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

なし

戻り値

以下の値を返します。

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.46 SysKeyBackLightOff

キーバックライトを OFF します。

```
[C++]
DWORD SysKeyBackLightOff()
```

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

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

解説

本関数は、キーバックライトを OFF します。
Device Emulator 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、
SysGetKeyBackLightState 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

なし

戻り値

以下の値を返します。

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.47 SysGetKeyBackLightState

キーバックライトの ON/OFF 状態を取得します。

```
[C++]
DWORD SysGetKeyBackLightState()
```

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

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

解説

本関数は、キーバックライトの ON/OFF 状態を取得します。

パラメータ

なし

戻り値

以下の値を返します。

TRUE	： キーバックライト ON
FALSE	： キーバックライト OFF
FUNCTION_UNSUPPORT	： 未サポートエラー

3.48 SysMultiTouchOn

マルチタッチモードを有効にします。

```
[C++]
DWORD SysMultiTouchOn()
```

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

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

解説

マルチタッチモードを有効にします。
Device Emulator 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、
SysGetMultiTouchState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

注意

本設定はリセットにより無効となります。

3.49 SysMultiTouchOff

マルチタッチモードを無効にします。

```
[C++]
DWORD SysMultiTouchOff()
```

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

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

解説

マルチタッチモードを無効にします。
Device Emulator 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、
SysGetMultiTouchState 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.50 SysGetMultiTouchState

マルチタッチモードの有効/無効を取得します。

```
[C++]
DWORD SysGetTouchPanelState()
```

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

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

解説

マルチタッチモードの有効/無効を取得します。

パラメータ

なし

戻り値

タッチパネルのオン／オフ状態を表す値

TRUE	： マルチタッチモード ON（デフォルト）
FALSE	： マルチタッチモード OFF
FUNCTION_UNSUPPORT	： 未サポートエラー

3.51 SysSetLED

LED の点灯/消灯を行います。

```
[C++]
DWORD SysSetLED(
  DWORD dwLEDMode,
  DWORD dwNum,
  DWORD dwOnTime,
  DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysSetLED( _
  ByVal dwLEDMode As Int32, _
  ByVal dwNum As Int32, _
  ByVal dwOnTime As Int32, _
  ByVal dwOFFTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetLED(
  Int32 dwLEDMode,
  Int32 dwNum,
  Int32 dwOnTime,
  Int32 dwOFFTime
);
```

解説

LED の点灯/消灯を行います。

パラメータ

dwLEDMode

LED の点灯/消灯を指定します。

表 3-2 設定値一覧

設定値	設定内容	DT-5200	DT-X7	DT-9800	DT-5300	IT-300	DT-X8	IT-9000	IT-G500
LED_OFF	LED を消灯します。(デフォルト値)	○	○	○	○	○	○	○	○
LED_GREEN	LED の緑を点灯します。	○	○	○	○	○	○	○	○
LED_RED	LED の赤を点灯します。	○	○	○	○	○	○	○	○
LED_ORANGE	LED のオレンジを点灯します。	○	○	○	○	○	○	○	○
LED_BLUE	LED の青を点灯します。	○	○		○	○	○	○	○
LED_CYAN	LED の青緑を点灯します。	○	○		○	○	○	○	○
LED_MAGENTA	LED の赤紫を点灯します。	○	○		○	○	○	○	○
LED_ACCESS	アクセス LED を点灯します。								

LED_NOTIFICATION	ノーティフィケーション LED を点灯します。								
LED_BOTH	両方（アクセス LED とノーティフィケーション LED）の LED を点灯します。								
LED_BLINK	指定した色の LED の点灯をします。 色を指定する LED_GREEN,LED_RED, LED_ORANGE と OR で指定してください。								

設定値	設定内容	DT-X100	DT-X200
LED_OFF	LED を消灯します。（デフォルト値）	○	○
LED_GREEN	LED の緑を点灯します。	○	○
LED_RED	LED の赤を点灯します。	○	○
LED_ORANGE	LED のオレンジを点灯します。	○	○
LED_BLUE	LED の青を点灯します。	○	○
LED_CYAN	LED の青緑を点灯します。	○	○
LED_MAGENTA	LED の赤紫を点灯します。	○	○
LED_ACCESS	アクセス LED を点灯します。		
LED_NOTIFICATION	ノーティフィケーション LED を点灯します。		
LED_BOTH	両方（アクセス LED とノーティフィケーション LED）の LED を点灯します。		
LED_BLINK	指定した色の LED の点灯をします。 色を指定する LED_GREEN,LED_RED, LED_ORANGE と OR で指定してください。		

dwNum

LED 連続点灯回数を指定します。点灯、消灯の繰り返し回数を指定してください。

dwOnTime

LED 点灯時間を、1/16（秒）単位で指定します。範囲：0～255

dwOFFTime

LED 消灯時間を 1/16（秒）単位で指定します。範囲：0～255

戻り値

TRUE　：正常終了
FALSE　：内部エラー
SYS_PARAMERR　：パラメータエラー
FUNCTION_UNSUPPORT　：未サポートエラー

補足

dwLEDMode に LED_BLINK を指定した場合（LED_BLINK|LED_RED など）は、システム側で決めた間隔で連続して点灯消灯を繰り返します。LED を消灯する場合は LED_OFF で消灯させてください。この場合、dwNum, dwOnTime, dwOFFTime に設定した値は無視されます。

使用例

LED 橙を素早く 1 回点滅する場合の設定

（ハードの仕様により以下の引数（dwNum, dwOnTime, dwOFFTime）が最低の値となります。）

```
SysSetLED(LED_ORANGE, 1, 1, 0);
```

LED 緑を 1 秒 ON、1秒 OFF を 3 回点滅する場合の設定

```
SysSetLED(LED_GREEN, 3, 16, 16);
```

LED 赤を 2 秒点灯する場合の設定

```
SysSetLED(LED_RED, 1, 32, 0);
```

3.52 SysGetLED

LED の点灯/消灯状態を取得します。

```
[C++]
DWORD SysGetLED()
```

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

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

解説

LED の点灯/消灯状態を取得します。

パラメータ

なし

戻り値

LED の点灯/消灯状態を表す値を返します。(値の詳細は SysSetLED 関数を参照してください)
また、DT-9800 以外の端末では以下の点灯条件が OR 値として返ります。

LED_NOTICE_ON	: ユーザ通知を点灯しています
LED_WLAN_ON	: SS 無線接続を点灯しています
LED_BT_ON	: Bluetooth 接続を点灯しています
LED_SCANOK_ON	: スキャナ読取正常を点灯しています
LED_SCANERR_ON	: スキャナ読取エラーを点灯しています
LED_DISKACCESS_ON	: ディスクアクセスを点灯しています
LED_WWAN_ON	: WAN 接続を点灯しています
LED_GPS_ON	: GPS 接続を点灯しています

3.53 SysPrepareLED

LED の点灯準備を行います

```
[C++]
DWORD SysPrepareLED(
  DWORD dwLEDMode,
  DWORD dwNum,
  DWORD dwOnTime,
  DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysPrepareLED( _
  ByVal dwLEDMode As Int32, _
  ByVal dwNum As Int32, _
  ByVal dwOnTime As Int32, _
  ByVal dwOFFTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPrepareLED(
  Int32 dwLEDMode,
  Int32 dwNum,
  Int32 dwOnTime,
  Int32 dwOFFTime
);
```

解説

LED の点灯準備を行います。SysSetLED 関数ではハードの仕様により LED 点灯まで 130ms かかります。この関数と SysUpdateLED を使用することにより、実行後すぐに LED を点灯することができます。

パラメータ

dwLEDMode

点灯準備対象の LED を指定します。

LED_GREEN　　: LED の緑の点灯準備をします
LED_RED　　: LED の赤の点灯準備をします
LED_ORANGE　　: LED の橙の点灯準備をします

dwNum

LED 連続点灯回数を指定します。点灯、消灯の繰り返し回数を設定してください。

dwOnTime

LED 点灯時間を、1/16 秒単位で指定します。
設定値の範囲は 0～255 です。

dwOFFTime

LED 消灯時間を、1/16 秒単位で指定します。

設定値の範囲は 0～255 です。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

使用例

LED 橙を素早く 1 回点滅する場合の点灯準備設定
（ハードの仕様により以下の引数（dwTime, dwOnTime, dwOFFTime）が最低の値となります。）

```
SysPrepareLED(LED_ORANGE, 1, 1, 3);
```

LED 緑を 1 秒 ON、1秒 OFF を 3 回点滅する場合の点灯準備設定

```
SysPrepareLED(LED_GREEN, 3 16, 16);
```

LED 赤を2秒点灯する場合の点灯準備設定

```
SysPrepareLED(LED_RED, 1 32, 3);
```

LED 赤を連続点灯する場合の点灯準備設定
（連続点灯の場合は dwOnTime, dwOFFTime に 0 使用可）

```
SysPrepareLED(LED_RED, 1 255, 0);
```

補足

LED 点灯中に SysPrepareLED 関数を設定しますと、LED の設定が変更されてしまいますので、SysPrepareLED 関数が実行された場合は強制的に LED は OFF します。

3.54 SysUpdateLED

LED の点灯を行います。

```
[C++]
DWORD SysUpdateLED(
  DWORD dwLEDMode
)
```

```
[Visual Basic]
Public Shared Function SysUpdateLED( _
  ByVal dwLEDMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysUpdateLED(
  Int32 dwLEDMode
);
```

解説

現在の LED 設定値で LED の点灯を行います。SysSetLED 関数ではハードの仕様により LED 点灯まで 130ms かかります。この関数と SysPrepareLED を使用することにより、実行後すぐに LED を点灯することができます。

パラメータ

dwLEDMode

LED_OFF	:	LED を消灯します。
LED_GREEN	:	現在の LED 設定値で LED の緑を点灯します。
LED_RED	:	現在の LED 設定値で LED の赤を点灯します。
LED_ORANGE	:	現在の LED 設定値で LED の橙を点灯します。

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

使用方法

LED を点灯させる前に SysPrepareLED 関数を実行し、LED 点灯準備を設定します。その後 SysUpdateLED 関数を実行し LED を SysPrepareLED 関数にて設定した条件で点灯させます。
(SysUpdateLED 関数を実行した場合、SysPrepareLED 関数が実行されてから 130ms 以上経過していない場合は実行されてから 130ms 待ってから LED 点灯処理を行います。)

補足

SysPrepareLED 関数を行わないで SysUpdateLED 関数を実行した場合は、現在の設定値に基づいて LED を点灯させるため、予期せぬ LED 点灯になりますのでご注意下さい。

3.55 SysSetLEDState

システム使用 LED の点灯/点灯抑止の設定をします。

```
[C++]
DWORD SysSetLEDState (
  DWORD dwType,
  BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetLEDState( _
  ByVal dwType As Int32 _
  ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetLEDState(
  Int32 dwType,
  Boolean bMute
);
```

解説

システム使用 LED の点灯/点灯抑止の設定をします。
Device Emulator 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、
SysGetLEDState 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

dwType

LED 種別を指定します。

LED_BT	: Bluetooth 接続中
LED_WLAN	: SS 無線接続中
LED_SCANOK	: スキャナ読み取り正常
LED_SCANERR	: スキャナ読み取りエラー
LED_ALL	: 全点灯・点滅
LED_WAN	: WAN 接続中
LED_GPS	: GPS 接続中

bMute

点灯/点灯抑止を指定します。

LED_Disable	: 点灯抑止 (LED は点灯・点滅しません)
LED_Enable	: 点灯(LED は点灯・点滅します) (デフォルト)

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

全抑止が設定されている場合、または点灯・点滅したい属性の個別抑止が ON の場合は LED は点灯・点滅しません。
本設定は次回の LED 点灯処理から有効になります。

3.56 SysGetLEDState

システム使用 LED の点灯/点灯抑止の取得をします。

```
[C++]
DWORD SysGetLEDState (
  DWORD dwType,
)
```

```
[Visual Basic]
Public Shared Function SysGetLEDState( _
  ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetLEDState(
  Int32 dwType
);
```

解説

システム使用 LED の点灯/点灯抑止の取得をします。

パラメータ

dwType
LED 種別を指定します。設定値は SysSetLEDState 関数を参照してください。

戻り値

LED_Disable	： 点灯抑止 (LED は点灯・点滅しません)
LED_Enable	： 点灯(LED は点灯・点滅します)
FUNCTION_UNSUPPORT	： 未サポートエラー

3.57 SysDisableCardDetect

カード電源を OFF し、擬似カード挿抜を行います。

```
[C++]
DWORD SysDisableCardDetect(
  DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysDisableCardDetect( _
  ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDisableCardDetect(
  Int32 socket
);
```

解説

カードデテクト端子をディセーブルにして、カード電源を OFF 状態にします。
Device Emulator 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、
SysGetCardDetect 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

socket
擬似挿抜対象のソケットを指定します。

表 設定値一覧

設定値	設定内容	DT-9800	DT-5300	IT-300	DT-X8	IT-9000	IT-G500	DT-X200
TYPE_PCMCIA	PCMCIA カード	○	○	○	○	○	○	○
TYPE_CF	CF カード							
TYPE_SD	SD カード		○	○	○	○		○

戻り値

TRUE : 正常終了
FALSE : 内部エラー
SYS_PARAMERR : パラメータエラー
FUNCTION_UNSUPPORT : 未サポートエラー

注意

本関数は次バージョン(3.01)以降では使用できなくなります。
今後は SysDisableWLAN 関数を実行してください。

3.58 SysEnableCardDetect

カード電源を ON し、擬似カード挿抜を行います。

```
[C++]
DWORD SysEnableCardDetect(
  DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysEnableCardDetect( _
  ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysEnableCardDetect(
  Int32 socket
);
```

解説

カードデテクト端子をイネーブルにして、カード電源を ON 状態にします。Device Emulator 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、SysGetCardDetect 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

socket
擬似挿抜対象のソケットを指定します。設定する値は SysDisableCardDetect 関数を参照してください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
SYS_PARAMERR	： パラメータエラー
FUNCTION_UNSUPPORT	： 未サポートエラー

注意

本関数は次バージョン（3.01）以降では使用できなくなります。
今後は SysEnableWLAN 関数を実行してください。

3.59 SysGetCardDetect

カードデテクト端子の状態を取得します。

```
[C++]
DWORD SysGetCardDetect(
  DWORD socket
)
```

```
[Visual Basic]
Public Shared Function SysGetCardDetect( _
  ByVal socket As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetCardDetect(
  Int32 socket
);
```

解説

カードデテクト端子の状態を取得します。

パラメータ

Socket
取得対象のソケットを指定します。設定する値は SysDisableCardDetect 関数を参照してください。

戻り値

TRUE	： カードデテクト端子有効
FALSE	： カードデテクト端子無効
SYS_PARAMERR	： パラメータエラー
FUNCTION_UNSUPPORT	： 未サポートエラー

注意

本関数は次バージョン（3.01）以降では使用できなくなります。
今後は SysGetWLAN 関数を実行してください。

3.60 SysDisableWLAN

無線 LAN モジュールの電源を OFF にします。

```
[C++]
DWORD SysDisableWLAN()
```

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

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

解説

内蔵している無線 LAN モジュールの電源を OFF 状態にします。
Device Emulator 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、SysGetWLAN 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.61 SysEnableWLAN

無線 LAN モジュールの電源を ON にします。

```
[C++]
DWORD SysEnableWLAN()
```

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

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

解説

内蔵している無線 LAN モジュールの電源を ON 状態にします。
Device Emulator 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、SysGetWLAN 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

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.62 SysGetWLAN

無線 LAN モジュールの電源状態を取得します。

```
[C++]
DWORD SysGetWLAN()
```

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

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

解説

内蔵している無線 LAN モジュールの電源状態を取得します。

パラメータ

なし

戻り値

TRUE	： 内蔵している無線 LAN モジュール電源 ON
FALSE	： 内蔵している無線 LAN モジュール電源 OFF
FUNCTION_UNSUPPORT	： 未サポートエラー

3.63 SysRenewPartition

パーティションのアンマウント→マウントを行い、ストレージの再認識を行います。

```
[C++]
DWORD SysRenewPartition (
  DWORD   dwStorage
)
```

```
[Visual Basic]
Public Shared Function SysRenewPartition( _
  ByVal dwStorage As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysRenewPartition(
  Int32 dwStorage
);
```

解説

パーティションのアンマウント→マウントを行い、ストレージの再認識を行います。
Device Emulator では、何も動作しません。

パラメータ

dwStorage

STORAGE_SD	:	SD メモリを再認識
STORAGE_CF	:	CF メモリを再認識

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

備考

本関数は SD メモリにのみ対応しています。

3.64 SysPlayBuzzer

ブザーを鳴らします。

```
[C++]
DWORD SysPlayBuzzer(
  DWORD dwType,
  DWORD dwHelz,
  DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysPlayBuzzer( _
  ByVal dwType As Int32, _
  ByVal dwHelz As Int32, _
  ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPlayBuzzer(
  Int32 dwType,
  Int32 dwHelz,
  Int32 dwTime
);
```

解説

ブザーを鳴らします。

パラメータ

dwType

対象のブザー音を、下表の設定値で指定します。空欄の組み合わせは無効です。

表 3-3 設定値一覧

設定値	設定内容	DT-X7	その他
B_USERDEF	ユーザ指定音	○	○
B_TAP	タップ音		2200Hz/25msec
B_CLICK	キークリック音		2600Hz/50msec
B_ALARM	アラーム音	2800Hz/150msec	2800Hz/150msec
B_WARNING	警告音	3000Hz/100msec	3000Hz/100msec
B_SCANEND	読み取り完了音	3100Hz/75msec	3100Hz/75msec
B_CARDREAD	カード読み書き音		
B_WIREREAD	無線着信音		

dwHelz

ブザー周波数を指定します。(ユーザ指定音時のみ有効)
BUZ_USERDEF 以外の場合は、BUZ_DEFAULT を使用します。

dwTime

ブザー音連続時間を指定します。(ユーザ指定音時のみ有効)
BUZ_USERDEF 以外の場合は、BUZ_DEFAULT を使用します。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
SYS_PARAMERR	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

全音量ミュートが設定されている場合、または鳴らしたい属性の個別ミュートが ON の場合はブザーは鳴りません。

3.65 SysStopBuzzer

ブザーを止めます。

```
[C++]
DWORD SysStopBuzzer()
```

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

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

解説

ブザーを止めます。

パラメータ

なし

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.66 SysSetBuzzerVolume

ブザー音量を設定します。

```
[C++]
DWORD SysSetBuzzerVolume(
  DWORD dwType
  DWORD dwBuzzerVolume
)
```

```
[Visual Basic]
Public Shared Function SysSetBuzzerVolume( _
  ByVal dwType As Int32, _
  ByVal dwBuzzerVolume As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBuzzerVolume(
  Int32 dwType,
  Int32 dwBuzzerVolume
);
```

解説

ブザー音量を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetBuzzerVolume 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

dwType
対象のブザー音を指定します。指定する値は SysPlayBuzzer 関数を参照してください。
ただし、タップ音とキークリック音の音量は設定しても反映しません。

dwBuzzerVolume
ブザー音量を、以下の値で指定します。

BUZZERVOLUME_MIN	： 音量 小(デフォルト)
BUZZERVOLUME_MID	： 音量 中
BUZZERVOLUME_MAX	： 音量 大

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
SYS_PARAMERR	： パラメータエラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.67 SysGetBuzzerVolume

ブザー音量を取得します。

```
[C++]
DWORD SysGetBuzzerVolume(
  DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetBuzzerVolume( _
  ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBuzzerVolume(
  Int32 dwType
);
```

解説

ブザー音量を取得します。

パラメータ

dwType
対象のブザー音を指定します。指定する値は SysPlayBuzzer 関数を参照してください。

戻り値

BUZZERVOLUME_MIN	： 音量 小(デフォルト)
BUZZERVOLUME_MID	： 音量 中
BUZZERVOLUME_MAX	： 音量 大
FALSE	： 内部エラー
SYS_PARAMERR	： パラメータエラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.68 SysSetBuzzerMute

ブザーの全音量、個別ミュートを設定します。

```
[C++]
DWORD SysSetBuzzerMute(
  DWORD dwType,
  BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetBuzzerMute( _
  ByVal dwType As Int32, _
  ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetBuzzerMute(
  Int32 dwType,
  Boolean bMute
);
```

解説

ブザーの全音量、個別ミュートを設定します。

パラメータ

dwType

対象のブザー音を指定します。指定する値は SysPlayBuzzer 関数を参照してください。
ただし、タップ音とキークリック音の音量は設定しても反映しません。

bMute

ミュートの状態を指定します。

TRUE　：ミュート ON（ブザーは鳴りません）
FALSE　：ミュート OFF（ブザーは鳴ります） デフォルト

戻り値

TRUE	：正常終了
FALSE	：内部エラー
SYS_PARAMERR	：パラメータエラー
FUNCTION_UNSUPPORT	：未サポートエラー

補足

全音量ミュートが設定されている場合、または鳴らしたい属性の個別ミュートが ON の場合はブザーが鳴りません。

3.69 SysGetBuzzerMute

ブザーの全音量、個別ミュートを取得します。

```
[C++]
DWORD SysGetBuzzerMute(
  DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetBuzzerMute( _
  ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBuzzerMute(
  Int32 dwType
);
```

解説

ブザーの全音量、個別ミュートを取得します。

パラメータ

dwType
対象のブザー音を指定します。指定する値は SysPlayBuzzer 関数を参照してください。

戻り値

TRUE	: ミュート ON(ブザーは鳴りません)
FALSE	: ミュート OFF(ブザーは鳴ります) デフォルト
SYS_PARAMERR	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

全音量ミュートが設定されている場合、または鳴らしたい属性の個別ミュートが ON の場合はブザーが鳴りません。

3.70 SysSetCPUMode

CPU 周波数制御を設定します。

```
[C++]
DWORD SysSetCPUMode(
  DWORD dwMode
)
```

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

```
[C#]
public static Int32 SysSetCPUMode(
  Int32 dwMode
);
```

解説

CPU 周波数制御を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetCPUMode 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

dwMode
CPU 周波数を、以下の値で指定します。

設定値	設定内容	DT-9800	その他
CPUMODE_LOW	低速	○	○
CPUMODE_MIDDLE	通常		○
CPUMODE_HIGH	高速	○	○
CPUMODE_AUTO	自動パワーセーブ		○

補足

設定値	DT-5200 DT-X7 DT-9800	DT-5300 IT-300 DT-X8 IT-9000 DT-X7M60SB DT-X7M62SB	IT-G500	DT-X100 DT-X200
CPUMODE_LOW	104MHz	208MHz	400MHz	208MHz
CPUMODE_MIDDLE	208MHz	312MHz	1100.8MHz	624MHz
CPUMODE_HIGH	416MHz	624MHz	1497.6MHz	806MHz

3.71 SysGetCPUMode

CPU 周波数制御を取得します。

```
[C++]
DWORD SysGetCPUMode(
  DWORD *pdwMode
)
```

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

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

解説

CPU 周波数制御を取得します。

パラメータ

pdwMode
CPU 周波数を取得します。（値の詳細は SysSetCPUMode 関数を参照してください）

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.72 SysSetDefaultCPUMode

CPU のスピード設定を工場出荷状態に戻します。

```
[C++]
DWORD SysSetDefaultCPUMode( )
```

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

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

解説

CPU のスピード設定を工場出荷状態に戻します。
Device Emulator では、何も動作しません。

パラメータ

なし

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.73 SysSet180Rotate

表示画像を 180 度回転させます。

```
[C++]
DWORD SysSet180Rotate(
  BOOL bRotate
)
```

```
[Visual Basic]
Public Shared Function SysSet180Rotate( _
  ByVal bRotate As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSet180Rotate(
  Boolean bRotate
);
```

解説

表示画像を 180 度回転させます。
Device Emulator 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、
SysGet180Rotate 関数を呼び出すことにより、設定値を確認することができます。

パラメータ

bRotate
表示画像の回転を指定します。

TRUE	:	180 度回転(反転)
FALSE	:	0 度回転(通常)

戻り値

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

3.74 SysGet180Rotate

表示画像の回転状態を取得します。

```
[C++]
DWORD SysGet180Rotate()
```

```
[Visual Basic]
Public Shared Function SysGet180Rotate() As Int32
```

```
[C#]
public static Int32 SysGet180Rotate()
```

解説

表示画像の回転状態を取得します。

パラメータ

なし

戻り値

TRUE	: 180 度回転(反転)
FALSE	: 0 度回転(通常)
FUNCTION_UNSUPPORT	: 未サポートエラー

3.75 SysSetBLBattery

バッテリー動作時のバックライトの明るさを設定します。

```
[C++]
DWORD SysSetBLBattery(
  DWORD setting
)
```

```
[Visual Basic]
Public Shared Function SysSetBLBattery( _
  ByVal setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBLBattery(
  Int32 setting
);
```

解説

バッテリー動作時のバックライトの明るさを設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetBLBattery 関数を呼び出すことにより、設定値を確認することができます。

パラメータ

setting
バックライトの明るさの値を指定します。明るさは0(暗い)から8(明るい)の9段階で指定してください。
デフォルトは 4 です。

設定できる最大値は、SysGetBLMaximum 関数を呼び出して取得して下さい。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.76 SysGetBLBattery

バッテリー動作時に設定されているバックライトの明るさの値を取得します。

```
[C++]
DWORD SysGetBLBattery(
  DWORD *setting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLBattery( _
  ByRef setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLBattery(
  ref Int32 setting
);
```

解説

バッテリー動作時に設定されているバックライトの明るさの値を取得します。

パラメータ

setting

バックライトの明るさの値を取得します。取得する値は SysSetBLBattery 関数を参照してください

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.77 SysSetBLExpower

AC 電源動作時のバックライトの明るさを設定します。

```
[C++]
DWORD SysSetBLExpower(
  DWORD setting
)
```

```
[Visual Basic]
Public Shared Function SysSetBLExpower( _
  ByVal setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBLExpower(
  Int32 setting
);
```

解説

AC 電源動作時のバックライトの明るさを設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetBLExpower 関数を呼び出すことにより、設定値を確認することができます。

パラメータ

setting
バックライトの明るさの値を指定します。明るさは0(暗い)から8(明るい)の9段階で指定してください。
デフォルトは 8 です。
設定できる最大値は、SysGetBLMaximum 関数を呼び出して取得して下さい。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.78 SysGetBLExpower

AC 電源動作時に設定されているバックライトの明るさの値を取得します。

```
[C++]
DWORD SysGetBLExpower(
  DWORD *setting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLExpower( _
  ByRef setting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLExpower(
  ref Int32 setting
);
```

解説

AC 電源動作時に設定されているバックライトの明るさの値を取得します。

パラメータ

setting
バックライトの明るさの値を取得します。取得する値は SysSetBLExpower 関数を参照してください

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.79 SysGetBLMaximum

バッテリー時、および AC 電源動作時に設定できるバックライトの最大の明るさ値を取得します。

```
[C++]
DWORD SysGetBLMaximum(
  DWORD *BAsetting,
  DWORD *ACsetting
)
```

```
[Visual Basic]
Public Shared Function SysGetBLMaximum( _
  ByRef BAsetting As Int32, _
  ByRef ACsetting As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLMaximum(
  ref Int32 BAsetting,
  ref Int32 ACsetting
);
```

解説

バッテリー時、および AC 電源動作時に設定できるバックライトの最大の明るさ値を取得します。

パラメータ

BAsetting
バッテリー時に設定できる最大値を取得します。

ACsetting
AC 電源時に設定できる最大値を取得します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.80 SysSetBLOffTime

電源ボタン連続押し時間を設定します。

```
[C++]
DWORD SysSetBLOffTime(
  DWORD dwTime
)
```

```
[Visual Basic]
Public Shared Function SysSetBLOffTime( _
  ByVal dwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetBLOffTime(
  Int32 dwTime
);
```

解説

バックライトがオフするまでの電源ボタン連続押し時間を設定します。

パラメータ

dwTime
電源ボタン連続押し時間(msec)を指定します。デフォルトは 2000msec です。

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.81 SysGetBLOffTime

電源ボタン連続押し時間を取得します。

```
[C++]
DWORD SysGetBLOffTime(
  DWORD *pdwTime
)
```

```
[Visual Basic]
Public Shared Function SysGetBLOffTime( _
  ByRef pdwTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetBLOffTime(
  ref Int32 pdwTime
);
```

解説

バックライトがオフするまでの電源ボタン連続押し時間を取得します。

パラメータ

pdwTime
電源ボタン連続押し時間(msec)を取得します。取得する値は SysSetBLOffTime 関数を参照してください。

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.82 SysPlayVibrator

バイブレータを振動します。

```
[C++]
DWORD SysPlayVibrator(
  DWORD dwType,
  DWORD dwCount,
  DWORD dwOnTime,
  DWORD dwOFFTime
)
```

```
[Visual Basic]
Public Shared Function SysPlayVibrator( _
  ByVal dwType As Int32, _
  ByVal dwCount As Int32, _
  ByVal dwOnTime As Int32, _
  ByVal dwOffTime As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysPlayVibrator(
  Int32 dwType,
  Int32 dwCount,
  Int32 dwOnTime,
  Int32 dwOffTime
);
```

解説

バイブレータを振動します。

パラメータ

dwType

対象の振動パターンを以下の値で指定します。

表 3-4 設定値一覧

設定値	設定内容	回数	ON 時間	OFF 時間
B_ALARM	アラーム	1	1000	1000
B_WARNING	警告	1	1000	1000
B_SCANEND	読取完了	1	1000	1000
B_WIREREAD	無線着信	1	1000	1000
B_USERDEF	ユーザー指定	-	-	-

dwCount

振動回数を 1～20 の範囲で指定します。(ユーザ指定時のみ有効)

B_USERDEF 以外の場合は、VIBRATOR_DEFAULT を使用します。

dwOnTime

振動オン時間(msec)を 0～16000 の範囲で指定します。(ユーザ指定時のみ有効)
B_USERDEF 以外の場合は、VIBRATOR _DEFAULT を使用します。

dwOFFTime

振動オフ時間(msec)を 0～1000 の範囲で指定します。(ユーザ指定時のみ有効)
B_USERDEF 以外の場合は、VIBRATOR _DEFAULT を使用します。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

補足

全振動ミュートが設定されている場合、または振動したい属性の個別ミュートが ON の場合はバイブレータは振動しません。

3.83 SysStopVibrator

バイブレータの振動を止めます。

```
[C++]
DWORD SysStopVibrator()
```

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

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

解説

バイブレータの振動を止めます。

パラメータ

なし

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.84 SysSetVibratorMute

バイブレータの全振動、個別ミュートを設定します。

```
[C++]
DWORD SysSetVibratorMute(
  DWORD dwType,
  BOOL bMute
)
```

```
[Visual Basic]
Public Shared Function SysSetVibratorMute( _
  ByVal dwType As Int32, _
  ByVal bMute As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetVibratorMute(
  Int32 dwType,
  Boolean bMute
);
```

解説

バイブレータの全振動、個別ミュートを設定します。

パラメータ

dwType

対象の振動パターンを以下の値で指定します。

B_ALARM	：アラーム
B_WARNING	：警告
B_SCANEND	：読取完了
B_WIREREAD	：無線着信
B_USERDEF	：ユーザ指定
B_ALL	：全音

bMute

ミュートの状態を指定します。

TRUE	：ミュート ON（バイブレータは振動しません）
FALSE	：ミュート OFF（バイブレータは振動します） デフォルト

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

補足

全振動ミュートが設定されている場合、または振動したい属性の個別ミュートが ON の場合はバイブレータが振動しません。

3.85 SysGetVibratorMute

バイブレータの全振動、個別ミュートを取得します。

```
[C++]
DWORD SysGetVibratorMute(
  DWORD dwType
)
```

```
[Visual Basic]
Public Shared Function SysGetVibratorMute( _
  ByVal dwType As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetVibratorMute(
  Int32 dwType
);
```

解説

バイブレータの全振動、個別ミュートを取得します。

パラメータ

dwType
対象の振動パターンを以下の値で指定します。取得する値は SysSetVibratorMute 関数を参照してください。

戻り値

TRUE	：ミュート ON（バイブレータは振動しません）
FALSE	：ミュート OFF（バイブレータは振動します） デフォルト
FUNCTION_UNSUPPORT	：未サポートエラー

補足

全振動ミュートが設定されている場合、または振動したい属性の個別ミュートが ON の場合はバイブレータが振動しません。

3.86 SysSetFnKeyLock

Fn キー動作の許可/禁止を設定します。

```
[C++]
DWORD SysSetFnKeyLock(
  BOOL bFnKey
)
```

```
[Visual Basic]
Public Shared Function SysSetFnKeyLock( _
  ByVal bFnKey As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetFnKeyLock(
  Boolean bFnKey
);
```

解説

Fn キー動作の許可/禁止を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetFnKeyLock 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

bFnKey
動作の許可/禁止を指定します。

TRUE	:	Fn キー無効
FALSE	:	Fn キー有効(デフォルト)

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.87 SysGetFnKeyLock

Fn キー動作の許可/禁止状態を取得します。

```
[C++]
DWORD SysGetFnKeyLock()
```

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

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

解説

Fn キー動作の許可/禁止状態を取得します。

パラメータ

なし

戻り値

TRUE	: Fn キー無効
FALSE	: Fn キー有効
FUNCTION_UNSUPPORT	: 未サポートエラー

3.88 SysSetOneKeyLock

個別キーロックの許可/禁止を設定します。

```
[C++]
DWORD SysSetOneKeyLock(
  DWORD dwLockBits
)
```

```
[Visual Basic]
Public Shared Function SysSetOneKeyLock( _
  ByVal dwLockBits As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOneKeyLock(
  Int32 dwLockBits
);
```

解説

個別キーロックの許可/禁止を設定します。トリガーキー、マルチキー、電源キー以外のキー押下の許可/禁止を設定します。

パラメータ

dwLockBits
設定可能な値は、機種により異なります。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

リセットを行うことにより、禁止設定は全て無効となり、全てのキー入力が可能となります。
また KEY_LOCK_ALL を指定した場合は、トリガーキー、マルチキー、電源キー以外のキー入力を受け付けませんので、リセットによる解除方法しか手段がありません。

3.89 SysGetOneKeyLock

個別キーロックの許可/禁止状態を取得します。

```
[C++]
DWORD SysGetOneKeyLock()
```

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

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

解説

個別キーロックの許可/禁止状態を取得します。トリガーキー、マルチキー、電源キー以外のキー押下の許可/禁止状態を取得します。

パラメータ

なし

戻り値

個別キーロックの許可/禁止状態が返ります。(値の詳細は SysSetOneKeyLock 関数を参照してください)
または、
FUNCTION_UNSUPPORT　：未サポートエラー

3.90 SysSetAllKeyLock

キーロックの許可/禁止を設定します。

```
[C++]
DWORD SysSetAllKeyLock(
  BOOL bKeyLock
)
```

```
[Visual Basic]
Public Shared Function SysSetAllKeyLock( _
  ByVal bKeyLock As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetAllKeyLock(
  Boolean bKeyLock
);
```

解説

キーロックの許可/禁止を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetAllKeyLock 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。
トリガーキー、電源キー以外のキー押下の許可/禁止を設定します。

パラメータ

bKeyLock
キーロックの許可/禁止を指定します。

TRUE	：トリガーキー、電源キー以外は無効
FALSE	：非ロック状態（デフォルト）

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.91 SysGetAllKeyLock

キーロックの許可/禁止状態を取得します。

```
[C++]
DWORD SysGetAllKeyLock()
```

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

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

解説

キーロックの許可/禁止状態を取得します。

パラメータ

なし

戻り値

キーロックの許可/禁止状態が返ります。(値の詳細は SysSetAllKeyLock 関数を参照してください)
または、
FUNCTION_UNSUPPORT　：未サポートエラー

3.92 SysSetInputMode

入力切替キー動作を設定します。

```
[C++]
DWORD SysSetInputMode(
  DWORD dwInputMode
)
```

```
[Visual Basic]
Public Shared Function SysSetInputMode( _
  ByVal dwInputMode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetInputMode(
  Int32 dwInputMode
);
```

解説

入力切替キー動作を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetInputMode 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

dwInputMode
入力切替キー動作を、以下の値で指定します

INPUT_NORMAL	： 通常動作(デフォルト)	切替可能
INPUT_LOCK_NUM	： 数字固定	切替不可
INPUT_LOCK_HIRA	： ひらがな固定	切替不可
INPUT_LOCK_KANA	： カタカナ固定	切替不可
INPUT_LOCK_ALPHA	： 英大文字固定	切替不可
INPUT_LOCK_ALPHAS	： 英小文字固定	切替不可
INPUT_LOCK_PHONE	： 電話固定（DT-5200/DT-5300/IT-300/IT-G500）	切替不可

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
SYS_PARAMERR	： パラメータエラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.93 SysGetInputMode

入力切替キー動作を取得します。

```
[C++]
DWORD SysGetInputMode()
```

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

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

解説

入力切替キー動作を取得します。

パラメータ

なし

戻り値

以下の値が返ります。

入力モード	入力モード固定時	入力モード未固定
数字	INPUT_LOCK_NUM	INPUT_NORMAL
ひらがな	INPUT_LOCK_HIRA	INPUT_LOCK_HIRA
カタカナ	INPUT_LOCK_KANA	INPUT_LOCK_KANA
英大文字	INPUT_LOCK_ALPHA	INPUT_LOCK_ALPHA
英小文字	INPUT_LOCK_ALPHAS	INPUT_LOCK_ALPHAS
電話	INPUT_LOCK_PHONE	INPUT_LOCK_PHONE

または以下の値が返ります。

FUNCTION_UNSUPPORT　：未サポートエラー

補足

本関数は SysSetInputMode 関数で設定した入力切替キー動作を取得します。SysSetInputMode 関数未実行の場合は、入力モードにより戻り値が異なります。

3.94 SysSetNormalUserDefineKey

通常モード(数値入力モード)のキーコードを設定します。

```
[C++]
DWORD SysSetNormalUserDefineKey(
  DWORD dwKeyID,
  DWORD KeySetBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysSetNormalUserDefineKey( _
  ByVal dwKeyID As Int32, _
  ByVal KeySetBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysSetNormalUserDefineKey(
  Int32 dwKeyID,
  Int32[] KeySetBuff
);
```

解説

通常モード(数値入力モード)のキーコードを設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetNormalUserDefineKey 関数を実行することにより、現在の設定値を確認することができます。

パラメータ

dwKeyID
キーID を以下の値で指定します。

表 3-5 キーID 一覧

設定値	設定内容	DT-5200	DT-X7	DT-9800	DT-5300	IT-300	DT-X8	IT-9000	IT-G500
KEYID_LEFTTRIGGER	レフトトリガーキー レフトアップキー(DT-5300M30S シリーズのみ) レフトサイドキー(IT-9000)	○	○	○	○		○	○	○
KEYID_RIGHTTRIGGER	ライトトリガーキー ライトエンターキー(DT-5300M30S シリーズのみ) ライトサイドキー(IT-9000)	○	○	○	○		○	○	○
KEYID_LEFT	左キー	○		○	○	○	○	○	○
KEYID_RIGHT	右キー	○		○	○	○	○	○	○
KEYID_CLEAR	クリアキー	○	○	○	○	○	○	○	○
KEYID_ENTER	決定キー	○	○	○	○	○	○	○	○

KEYID_0	0 キー	○	○	○	○	○	○	○	○
KEYID_1	1 キー	○	○	○	○	○	○	○	○
KEYID_2	2 キー	○	○	○	○	○	○	○	○
KEYID_3	3 キー	○	○	○	○	○	○	○	○
KEYID_4	4 キー	○	○	○	○	○	○	○	○
KEYID_5	5 キー	○	○	○	○	○	○	○	○
KEYID_6	6 キー	○	○	○	○	○	○	○	○
KEYID_7	7 キー	○	○	○	○	○	○	○	○
KEYID_8	8 キー	○	○	○	○	○	○	○	○
KEYID_9	9 キー	○	○	○	○	○	○	○	○
KEYID_UP	上キー	○	○		○	○	○		○
KEYID_DOWN	下キー	○	○		○	○	○		○
KEYID_F1	F1 キー		○		○	○	○	○	○
KEYID_F2	F2 キー		○		○	○	○	○	○
KEYID_F3	F3 キー		○		○	○	○	○	○
KEYID_F4	F4 キー		○		○	○	○	○	○
KEYID_F5	F5 キー		○				○		
KEYID_F6	F6 キー		○				○		
KEYID_F7	F7 キー		○				○		
KEYID_F8	F8 キー		○				○		
KEYID_000	000 キー			○				○	
KEYID_00	00 キー			○				○	
KEYID_HYPHEN	ハイフン	○		○	○	○		○	○
KEYID_BS	BS キー			○				○	
KEYID_PERIOD	ピリオド（・）キー		○				○	○	
KEYID_INPUTMODE	文字キー	○			○	○			○
KEYID_CENTERTRIGGER	センタートリガーキー センターエンターキー（DT-5300M30S シリーズのみ）	○	○		○	○	○		○
KEYID_GUNTRIGGER	ガントリガーキー	○			○				○
KEYID_LEFTDOWN	レフトダウンキー（DT-5300M30S シリーズのみ）				○				○
KEYID_MENU	MENU キー					○		○	
KEYID_LMULTI	レフトマルチキー						○		
KEYID_RMULTI	ライトマルチキー						○		
KEYID_SUBTRACT	ハイフンキー（IT-G500 のみ）								○

設定値	設定内容	DT-X100	DT-X200
KEYID_LEFTTRIGGER	レフトトリガーキー レフトアップキー（DT-5300M30S シリーズのみ） レフトサイドキー（IT-9000）	○	○
KEYID_RIGHTTRIGGER	ライトトリガーキー ライトエンターキー（DT-5300M30S シリーズのみ）	○	○

	ライトサイドキー（IT-9000）		
KEYID_LEFT	左キー		○
KEYID_RIGHT	右キー		○
KEYID_CLEAR	クリアキー	○	○
KEYID_ENTER	決定キー	○	○
KEYID_0	0 キー	○	○
KEYID_1	1 キー	○	○
KEYID_2	2 キー	○	○
KEYID_3	3 キー	○	○
KEYID_4	4 キー	○	○
KEYID_5	5 キー	○	○
KEYID_6	6 キー	○	○
KEYID_7	7 キー	○	○
KEYID_8	8 キー	○	○
KEYID_9	9 キー	○	○
KEYID_UP	上キー	○	○
KEYID_DOWN	下キー	○	○
KEYID_F1	F1 キー	○	○
KEYID_F2	F2 キー	○	○
KEYID_F3	F3 キー	○	○
KEYID_F4	F4 キー	○	○
KEYID_F5	F5 キー	○	○
KEYID_F6	F6 キー	○	○
KEYID_F7	F7 キー	○	○
KEYID_F8	F8 キー	○	○
KEYID_000	000 キー		
KEYID_00	00 キー		
KEYID_HYPHEN	ハイフン		
KEYID_BS	BS キー		
KEYID_PERIOD	ピリオド（・）キー	○	○
KEYID_INPUTMODE	文字キー		
KEYID_CENTERTRIGGER	センタートリガーキー センターエンターキー（DT-5300M30S シリーズのみ）	○	○
KEYID_GUNTRIGGER	ガントリガーキー		
KEYID_LEFTDOWN	レフトダウンキー（DT-5300M30S シリーズのみ）		
KEYID_MENU	MENU キー		
KEYID_LMULTI	レフトマルチキー		○
KEYID_RMULTI	ライトマルチキー		○
KEYID_SUBTRACT	ハイフンキー（IT-G500 のみ）		

KeySetBuff[16]

キーダウン時およびキーアップ時に発生する「仮想キーコード」を、16 要素の配列で指定します。
最大で 16 個の仮想キーコードを指定できますが、配列の途中に、値が 0 のデータがあると、そのデータ以降は無視されます。

仮想キーコードには、ビット OR 演算子を使用して、任意のオプションフラグを指定できます。下記の補足を参照してください。

戻り値

TRUE　　：正常終了
FALSE　　：内部エラー
SYS_PARAMERR　　：パラメータエラー
FUNCTION_UNSUPPORT　　：未サポートエラー

補足

[仮想キーコードのオプションフラグ]
仮想キーコードには、以下に示すオプションフラグを指定できます。
（任意の組み合わせが可能ですが、「VF_PAGE と VF_NO_FIX_PAGE の組み合わせ」は無効です）

表 3-6 仮想キーコードのオプションフラグ

定義名と説明	VF_NO_KEY_UP 指定なし		VF_NO_KEY_UP 指定あり	
	キーダウン時	キーアップ時	キーダウン時	キーアップ時
オプションフラグ指定なし （仮想キーコードのみ）	仮想キーコードをダウン 仮想キーコードをアップ	何もしない	―	―
VF_NO_KEY_UP キーをリピートする場合に指定します。	―	―	仮想キーコードをダウン	仮想キーコードをアップ
VF_PAGE めくりキー動作を指定します。 最初のキーに指定します。	仮想キーコードをダウン 仮想キーコードをアップ	何もしない	仮想キーコードをダウン	仮想キーコードをアップ
VF_NO_FIX_PAGE めくりキーを確定しません。 最初のキーに指定します。 VF_PAGE と同時に指定した場合は、 VF_NO_FIX_PAGE は無視されます。 通常はプログラムキーのような文字を出力しない特殊なキーに指定します。	仮想キーコードをダウン 仮想キーコードをアップ	何もしない	仮想キーコードをダウン	仮想キーコードをアップ
VF_KANA VK_KANA を送ります。 かな文字を出力する場合に指定します。	VK_KANA をダウン VK_KANA をアップ 仮想キーコードをダウン 仮想キーコードをアップ VK_KANA をダウン VK_KANA をアップ	何もしない	VK_KANA をダウン VK_KANA をアップ 仮想キーコードをダウン	仮想キーコードをアップ VK_KANA をダウン VK_KANA をアップ

VF_SHIFT SHIFT キーとの同時押しをシミュレートします。	VK_SHIFT をダウン 仮想キーコードをダウン 仮想キーコードをアップ VK_SHIFT をアップ	何もしない	VK_SHIFT をダウン 仮想キーコードをダウン	仮想キーコードをアップ VK_SHIFT をアップ
VF_CONTROL CTRL キーとの同時押しをシミュレートします。	VK_CONTROL をダウン 仮想キーコードをダウン 仮想キーコードをアップ VK_CONTROL をアップ	何もしない	VK_CONTROL をダウン 仮想キーコードをダウン	仮想キーコードをアップ VK_CONTROL をアップ
VF_MENU ALT キーとの同時押しをシミュレートします。	VK_MENU をダウン 仮想キーコードをダウン 仮想キーコードをアップ VK_MENU をアップ	何もしない	VK_MENU をダウン 仮想キーコードをダウン	仮想キーコードをアップ VK_MENU をアップ
VF_NOP 無効キーコード(0x00)を送ります。 通常は VF_NO_KEY_UP と組み合わせ仮想キーコードをリピートさせたくないが、キーアップ時に仮想キーコードをアップしたい場合に指定します。 プログラムキーのようなアプリケーションで GetAsyncKeyState 関数を使用しキーダウン、キーアップを監視する場合に指定します。	仮想キーコードをダウン 無効キーコードをダウン 無効キーコードをアップ 仮想キーコードをアップ	何もしない	仮想キーコードをダウン 無効キーコードをダウン 無効キーコードをアップ	仮想キーコードをアップ
VF_NOP_CLICK キークリック音を鳴らす無効キーコード(0x00)を送ります。 通常は使用しません。	仮想キーコードをダウン 無効キーコードをダウン 無効キーコードをアップ 仮想キーコードをアップ	何もしない	仮想キーコードをダウン 無効キーコードをダウン 無効キーコードをアップ	仮想キーコードをアップ
KEYBD_DEVICE_SILENT 最初のキークリック音、およびリピート時のキークリック音を鳴らしません。	仮想キーコードをダウン 仮想キーコードをアップ	何もしない	仮想キーコードをダウン	仮想キーコードをアップ

KEYBD_DEVICE_SILENT_REPEAT リピート時のキークリック音を鳴らしません。通常は VF_NO_KEY_UP と組み合わせます。	仮想キーコードをダウン 仮想キーコードをアップ	何もしない	仮想キーコードをダウン	仮想キーコードをアップ

[既定の仮想キーコード]
仮想キーコードの設定例として、主なキーの既定の仮想キーコード（数値入力モード用）を以下に示します。

表 3-7 仮想キーコード＋オプションフラグ

KeyId	KeySetBuff [16] の値
KEYID_000 （000 キー）	VK_0, VK_0 | KEYBD_DEVICE_SILENT, VK_0 | KEYBD_DEVICE_SILENT, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_00 （英語OSの00キー）	VK_0, VK_0 | KEYBD_DEVICE_SILENT, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_0 （0 キー）	VK_0 | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_1 （1 キー）	VK_1 | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_2 （2 キー）	VK_2 | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_3 （3 キー）	VK_3 | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_4 （4 キー）	VK_4 | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_PERIOD （ピリオド（・）キー）	VK_PERIOD, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_HYPHEN （ハイフン（－）キー）	VK_HYPHEN, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_C （取消キー）	VK_ESCAPE, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_BS （後退キー）	VK_BACK, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_ENT （入力キー）	VK_RETURN, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_LEFT （左（＜）キー）	VK_LEFT | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_RIGHT （右（＞）キー）	VK_RIGHT | KEYBD_DEVICE_SILENT_REPEAT | VF_NO_KEY_UP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_LTR （左プログラムキー）	VF_NO_FIX_PAGE | VK_F24 | KEYBD_DEVICE_SILENT | VF_NO_KEY_UP | VF_NOP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
KEYID_RTR （右プログラムキー）	VF_NO_FIX_PAGE | VK_F21 | KEYBD_DEVICE_SILENT | VF_NO_KEY_UP | VF_NOP, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

上記の値は、SysGetDefaultKey 関数で取得できます。仮想キーコードの設定を初期状態に戻したいときには、SysGetDefaultKey 関数で取得した配列を、そのまま SysSetNormalUserDefineKey 関数に渡してください。

[複数の仮想キーコードの指定例]
VF_PAGE | VK_0, VK_1, VK_2, VK_3, VF_KANA | WCH_A, VF_KANA | WCH_I, VF_KANA | WCH_U, VF_KANA | WCH_E, VK_A, VK_B, VK_C, VK_D, VK_SHIFT | VK_A, VK_SHIFT | VK_B, VK_SHIFT | VK_C, VK_SHIFT | VK_D

上記のように、KeySetBuff[16] の最初の仮想キーコードに VF_PAGE オプションを指定した場合は、「めくりキー」動作となり、キーを 1 回押すごとに、「0→1→2→3→4→ｱ→ｲ→ｳ→ｴ→a→b→c→d→A→B→C→D→（以下、繰り返し）」の順に出力します。

最初の仮想キー（VF_PAGE | VK_0）の VF_PAGE オプションを削除し、2 番目の仮想キー（VK_1）以降すべてに KEYBD_DEVICE_SILENT オプションを指定すると、キークリック音を一度だけ鳴らして、「01234 ｱｲｳｴ abcdABCD」を一度に出力します。

最初の仮想キーコードの値に 0 を指定すると、当該キーを操作しても、コードが何も発生しません（操作が無視されます）。

注意事項

一通り設定した後、SysSetUserDefineKeyState（TRUE）を呼ぶことでキーボードドライバに通知し、ユーザ定義キーを有効にすることができます。
ユーザ定義キーが有効状態であっても、設定後にリセットするか、SysSetUserDefineKeyState(TRUE) を呼ばない限り、有効とはなりません。

3.95 SysGetNormalUserDefineKey

通常モードのキーコードを取得します。

```
[C++]
DWORD SysGetNormalUserDefineKey(
  DWORD dwKeyID,
  DWORD KeySetBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetNormalUserDefineKey( _
  ByVal dwKeyID As Int32, _
  ByVal KeySetBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetNormalUserDefineKey(
  Int32 dwKeyID,
  Int32[] KeySetBuff
);
```

解説

通常モードのキーコードを取得します。

パラメータ

dwKeyID
キーID を指定します。(値の詳細は SysSetNormalUserDefineKey 関数を参照してください)

KeySetBuff[16]
キーID に対して設定されている、仮想キーコードとオプションフラグを取得します。
仮想キーコードとオプションフラグについては、SysSetNormalUserDefineKey 関数を参照してください。
キーID に対して定義されている、既定の仮想キーコードとオプションフラグを取得するには、SysGetDefaultKey 関数を使用してください。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラーまたはユーザ定義キーは未定義
SYS_PARAMERR	: パラメータエラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.96 SysSetUserDefineKey

ユーザ定義キーを設定します。

```
[C++]
DWORD SysSetUserDefineKey(
  WORD    KeyMode,
  DWORD   KeyID,
  DWORD   UserDefineKeyBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysSetUserDefineKey( _
  ByVal KeyMode As Short, _
  ByVal KeyID As Int32, _
  ByVal UserDefineKeyBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysSetUserDefineKey(
  Short KeyMode,
  Int32 KeyID,
  Int32[] UserDefineKeyBuff
);
```

解説

ユーザ定義キーを設定します。数字・ひらがな／カタカナ・英大文字・英小文字モード時に発行する任意のユーザ定義キーを設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetUserDefineKey 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

KeyMode
キーモードを以下の値で指定します。

KEY_MODE_NUM	：数字
KEY_MODE_KANA	：ひらがな／カタカナ
KEY_MODE_ALPHA	：英大文字
KEY_MODE_ALPHAS	：英小文字

KeyID
キーID を指定します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

UserDefineKeyBuff
キーコードを指定します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

戻り値

TRUE	：正常終了
FALSE	：内部エラー

SYS_PARAMERR　　：パラメータエラー
FUNCTION_UNSUPPORT　：未サポートエラー

3.97 SysGetUserDefineKey

ユーザ定義キーを取得します。

```
[C++]
DWORD SysSetUserDefineKey(
  WORD    KeyMode,
  DWORD   KeyID,
  DWORD   UserDefineKeyBuff[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetUserDefineKey( _
  ByVal KeyMode As Short, _
  ByVal KeyID As Int32, _
  ByVal UserDefineKeyBuff As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetUserDefineKey(
  Short KeyMode,
  Int32 KeyID,
  Int32[] UserDefineKeyBuff
);
```

解説

ユーザ定義キーを取得します。数字・ひらがな／カタカナ・英大文字・英小文字モード時に発行する任意のユーザ定義キーを取得します。

パラメータ

KeyMode

キーモードを以下の値で指定します。

KEY_MODE_NUM	：数字
KEY_MODE_KANA	：ひらがな／カタカナ
KEY_MODE_ALPHA	：英大文字
KEY_MODE_ALPHAS	：英小文字

KeyID

キーID を指定します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

UserDefineKeyBuff

キーコードを取得します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

戻り値

TRUE	：正常終了
FALSE	：内部エラー
SYS_PARAMERR	：パラメータエラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.98 SysSetFnUserDefineKey

Fn モードのキーコードを設定します。

```
[C++]
DWORD SysSetFnUserDefineKey(
  DWORD dwKeyID,
  DWORD dwKeyCode
)
```

```
[Visual Basic]
Public Shared Function SysSetFnUserDefineKey( _
  ByVal KeyID As Int32, _
  ByVal dwKeyCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetFnUserDefineKey(
  Int32 KeyID,
  Int32 dwKeyCode
);
```

解説

Fn モードのキーコードを設定します。

パラメータ

dwKeyID

対象のキーID を以下の値で指定します。

KEYID_LEFTTRIGGER	: 左トリガーキー
KEYID_RIGHTTRIGGER	: 右トリガーキー
KEYID_LEFT	: 左キー
KEYID_RIGHT	: 右キー
KEYID_CLEAR	: クリアキー
KEYID_ENTER	: 決定キー
KEYID_0	: 0 キー
KEYID_1	: 1 キー
KEYID_2	: 2 キー
KEYID_3	: 3 キー
KEYID_4	: 4 キー
KEYID_5	: 5 キー
KEYID_6	: 6 キー
KEYID_7	: 7 キー
KEYID_8	: 8 キー
KEYID_9	: 9 キー
KEYID_MULTI	: マルチキー
KEYID_FUNCTION	: ファンクションキー
KEYID_UP	: 上キー
KEYID_DOWN	: 下キー

KEYID_F1 : F1 キー
KEYID_F2 : F2 キー
KEYID_F3 : F3 キー
KEYID_F4 : F4 キー
KEYID_F5 : F5 キー
KEYID_F6 : F6 キー
KEYID_F7 : F7 キー
KEYID_F8 : F8 キー

dwKeyCode
キーコードを指定します。

戻り値

TRUE : 正常終了
FALSE : 内部エラー
FUNCTION_UNSUPPORT : 未サポートエラー

3.99 SysGetFnUserDefineKey

Fn モードのキーコードを取得します。

```
[C++]
DWORD SysGetFnUserDefineKey(
  DWORD dwKeyID,
  DWORD *pdwKeyCode
)
```

```
[Visual Basic]
Public Shared Function SysGetFnUserDefineKey( _
  ByVal dwKeyID As Int32, _
  ByRef pdwKeyCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetFnUserDefineKey(
  Int32 dwKeyID,
  ref Int32 pdwKeyCode
);
```

解説

Fn モードのキーコードを取得します。

パラメータ

dwKeyID
対象のキーID を指定します。(値の詳細は SysSetFnUserDefineKey 関数を参照してください)

pdwKeyCode
キーコードを取得します。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.100 SysSetOtherUserDefineKey

ひらがな、カタカナ、英字モードのキーコードを設定します。

```
[C++]
DWORD SysSetOtherUserDefineKey(
  WORD wMode,
  WORD wKeyID,
  DWORD *pdwCodes
)
```

```
[Visual Basic]
Public Shared Function SysSetOtherUserDefineKey( _
  ByVal wMode As Short, _
  ByVal wKeyID As Short, _
  ByRef pdwCodes As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetOtherUserDefineKey(
  short wMode,
  short wKeyID,
  ref Int32 pdwCodes
);
```

解説

ひらがな、カタカナ、英字モードのキーコードを設定します。

パラメータ

wMode

対象のモードを、以下の値で指定します。

KEY_CONVERSION_MODE_KANA	：ひらがな、カタカナ
KEY_CONVERSION_MODE_CAPITALALPHA	：英大文字
KEY_CONVERSION_MODE_SMALLALPHA	：英小文字

wKeyID

対象のキーID を、以下の値で指定します。

KEYID_0
KEYID_1
KEYID_2
KEYID_3
KEYID_4
KEYID_5
KEYID_6
KEYID_7
KEYID_8
KEYID_9

pdwCodes[20]

モード、キーID に対応するキーコード（最大 20 個）を指定します。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

設定は必ず0から順番に入れてください。途中に設定なしの 0x00000000 が入っていると、それ以降の設定は無効になります。また設定する場合は、必ず SysGetFnUserDefineKey を行い、現在のキーコード状態を取得した後に、キーコード変更設定を行ってください。

3.101 SysGetOtherUserDefineKey

ひらがな、カタカナ、英字モードのキーコードを取得します。

```
[C++]
DWORD SysGetOtherUserDefineKey(
  WORD wMode,
  WORD wKeyID,
  DWORD *pdwCode
)
```

```
[Visual Basic]
Public Shared Function SysGetOtherUserDefineKey( _
  ByVal wMode As Short, _
  ByVal wKeyID As Short, _
  ByRef pdwCode As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetOtherUserDefineKey(
  short wMode,
  short wKeyID,
  ref Int32 pdwCode
);
```

解説

ひらがな、カタカナ、英字モードのキーコードを取得します。

パラメータ

wMode
対象のモードを指定します。(値の詳細は SysSetOtherUserDefineKey 関数を参照してください)

wKeyID
対象のキーID を指定します。(値の詳細は SysSetOtherUserDefineKey 関数を参照してください)

pdwCode[20]
モード、キーID に対応するキーコード(最大 20 個)を取得します。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.102 SysSetKeyRepeat

キーリピートするキーコードを設定します。

```
[C++]
DWORD SysSetKeyRepeat(
  DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetKeyRepeat( _
  ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetKeyRepeat(
  ref Int32 pdwKeyRepeat
);
```

解説

キーリピートするキーコードを設定します。

パラメータ

pdwKeyRepeat[16]

キーリピートするキーコード（最大 16 個）の配列を指定します。
デフォルトは下記の値が設定されています。

表 3-8 デフォルトキーコード一覧

配列	Virtual Key Code	Virtual Key Code の値
0	VK_UP	0x26
1	VK_DOWN	0x28
2	VK_LEFT	0x25
3	VK_RIGHT	0x27
4	設定なし	0x00
5	設定なし	0x00
6	設定なし	0x00
7	設定なし	0x00
8	設定なし	0x00
9	設定なし	0x00
10	設定なし	0x00
11	設定なし	0x00
12	設定なし	0x00
13	設定なし	0x00
14	設定なし	0x00
15	設定なし	0x00

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

補足

設定は必ず 0 から順番に入れてください。途中に設定なしの 0x00 が入っていると、それ以降の設定は無効になります。また設定する場合は、必ず SysGetKeyRepeat を行い、現在のキーリピート状態を取得した後に、キーリピート変更設定を行ってください。

3.103 SysGetKeyRepeat

キーリピートするキーコードを取得します。

```
[C++]
DWORD SysGetKeyRepeat(
  DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysGetKeyRepeat( _
  ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetKeyRepeat(
  ref Int32 pdwKeyRepeat
);
```

解説

キーリピートするキーコードを取得します。

パラメータ

pdwKeyRepeat[16]
キーリピートするキーコード（最大 16 個）を取得する配列。
取得する値の詳細は SysSetKeyRepeat 関数を参照してください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.104 SysSetFakeRepeat

偽りキーリピートするキーコードを設定します。

```
[C++]
DWORD SysSetFakeRepeat(
  DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetFakeRepeat( _
  ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetFakeRepeat(
  ref Int32 pdwKeyRepeat
);
```

解説

偽りキーリピートするキーコードを設定します。
偽りキーリピートとは、キーを長く押すことで何度も発生するKEY_DOWNイベントを発生させないようにすることです。デフォルトで登録済みのキーコード(VK_F24/VK_F21)はトリガーキーでトリガーキーを長く押した場合に何度もスキャナが発光しないようにしています。

パラメータ

pdwKeyRepeat[4]
キーリピート(偽りリピート)するキーコード(最大4個)を指定します。
デフォルトは下記の値が設定されています。

表 3-9 デフォルトキーコード一覧

配列	Virtual Key Code	Virtual Key Code の値
0	VK_F24	0x87
1	VK_F21	0x84
2	設定なし	0x00
3	設定なし	0x00

戻り値

TRUE　　: 正常終了
FALSE　　: 内部エラー
FUNCTION_UNSUPPORT　　: 未サポートエラー

補足

設定は必ず0から順番に入れてください。途中に設定なしの0x00が入っていると、それ以降の設定は無効になります。また設定する場合は、必ずSysGetFakeRepeatを行い、現在のキーリピート状態を取得した後に、キーリピート(偽りリピート)変更設定を行ってください。

3.105 SysGetFakeRepeat

偽りキーリピートするキーコードを取得します。

```
[C++]
DWORD SysGetFakeRepeat(
  DWORD *pdwKeyRepeat
)
```

```
[Visual Basic]
Public Shared Function SysGetFakeRepeat( _
  ByRef pdwKeyRepeat As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysGetFakeRepeat(
  ref Int32 pdwKeyRepeat
);
```

解説

偽りキーリピートするキーコードを取得します。

パラメータ

pdwKeyRepeat[4]
キーリピート(偽りリピート)するキーコード(最大 4 個)を取得する配列。

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.106 SysSetAllKeyRepeat

キーリピート全ての有効/無効を設定します。

```
[C++]
DWORD SysSetAllKeyRepeat(
  BOOL bRepeat
)
```

```
[Visual Basic]
Public Shared Function SysSetAllKeyRepeat( _
  ByVal bRepeat As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetAllKeyRepeat(
  Boolean bRepeat
);
```

解説

SysSetKeyRepeat でリピート設定したキー全ての有効/無効を設定します。

パラメータ

bRepeat

リピートの有効/無効を指定します。

TRUE　：リピート有効
FALSE　：リピート無効

戻り値

TRUE　：正常終了
FALSE　：内部エラー
FUNCTION_UNSUPPORT　：未サポートエラー

3.107 SysGetAllKeyRepeat

キーリピート全ての有効/無効の状態を取得します。

```
[C++]
DWORD SysGetAllKeyRepeat()
```

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

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

解説

SysSetKeyRepeat でリピート設定したキー全ての有効/無効の状態を取得します。

パラメータ

なし

戻り値

TRUE	：リピート有効
FALSE	：リピート無効
FUNCTION_UNSUPPORT	：未サポートエラー

3.108 SysDeleteUserDefineKey

数値入力モード時に発行する任意のユーザ定義キーを削除します。

```
[C++]
DWORD SysDeleteUserDefineKey(
  int KeyId
)
```

```
[Visual Basic]
Public Shared Function SysDeleteUserDefineKey( _
  ByVal KeyId As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDeleteUserDefineKey(
  Int32 KeyId
);
```

解説

数値入力モード時に発行する任意のユーザ定義キーを削除します。削除した場合は既定のキーを使用します。

パラメータ

KeyId
対象のキーID を指定します。(値の詳細は SysSetNormalUserDefineKey 関数を参照してください)

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.109 SysDeleteUserDefineKeyEx

ユーザ定義キーを削除します。

```
[C++]
DWORD SysDeleteUserDefineKeyEx(
  WORD    KeyMode,
  DWORD   KeyID
)
```

```
[Visual Basic]
Public Shared Function SysDeleteUserDefineKeyEx( _
  ByVal KeyMode As Short, _
  ByVal KeyID As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysDeleteUserDefineKeyEx(
  Short KeyMode,
  Int32 KeyID
);
```

解説

ユーザ定義キーを削除します。数字・ひらがな／カタカナ・英大文字・英小文字モード時に発行する任意のユーザ定義キーを削除します。
削除した場合は既定のキーを使用します。

パラメータ

KeyMode
キーモードを以下の値で指定します。

KEY_MODE_NUM	：数字
KEY_MODE_KANA	：ひらがな／カタカナ
KEY_MODE_ALPHA	：英大文字
KEY_MODE_ALPHAS	：英小文字

KeyID
キーID を指定します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

戻り値

TRUE	：正常終了
FALSE	：内部エラー
SYSPARAMERR	：パラメータエラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.110 SysGetDefaultKey

数値入力モード時に発行する既定のキーを取得します。

```
[C++]
DWORD SysGetDefaultKey(
  int KeyId,
  DWORD DefaultKeyBuf[16]
)
```

```
[Visual Basic]
Public Shared Function SysGetDefaultKey( _
  ByVal KeyId As Int32, _
  ByVal DefaultKeyBuf As Int32() _
) As Int32
```

```
[C#]
public static Int32 SysGetDefaultKey(
  Int32 KeyId,
  Int32[] DefaultKeyBuf
);
```

解説

数値入力モード時に発行する既定のキーを取得します。

パラメータ

KeyId
対象のキーID を指定します。（値の詳細は SysSetNormalUserDefineKey 関数を参照してください）

DefaultKeyBuf[16]
キーID に対して設定されている、既定の仮想キーコードとオプションフラグを取得します。
仮想キーコードとオプションフラグについては、SysSetNormalUserDefineKey 関数を参照してください。

戻り値

TRUE	： 正常終了
FALSE	： 内部エラー
FUNCTION_UNSUPPORT	： 未サポートエラー

3.111 SysSetUserDefineKeyState

ユーザ定義キーの有効/無効を設定し、キーボードドライバに状態を通知します。

```
[C++]
DWORD SysSetUserDefineKeyState(
  BOOL State
)
```

```
[Visual Basic]
Public Shared Function SysSetUserDefineKeyState( _
  ByVal State As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetUserDefineKeyState(
  Boolean State
);
```

解説

ユーザ定義キーの有効/無効を設定し、キーボードドライバに状態を通知します。
[コントロールパネル]－[キーボード]－[その他]タブの「ユーザ定義キー:有効にする」の状態を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetUserDefineKeyState 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

State
ユーザ定義キーの有効/無効を指定します。

TRUE	:	有効
FALSE	:	無効

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.112 SysGetUserDefineKeyState

ユーザ定義キーの有効/無効を取得します。

```
[C++]
DWORD SysGetUserDefineKeyState()
```

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

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

解説

ユーザ定義キーの有効/無効を取得します。
[コントロールパネル]－[キーボード]－[その他]タブの「ユーザ定義キー:有効にする」の状態を取得します。

パラメータ

なし

戻り値

TRUE	： 有効
FALSE	： 無効
FUNCTION_UNSUPPORT	： 未サポートエラー

3.113 SysSetResetUserDefineKeyState

リセット時のユーザ定義キー無効の有効/無効を設定します。

```
[C++]
DWORD SysSetResetUserDefineKeyState(
  BOOL State
)
```

```
[Visual Basic]
Public Shared Function SysSetResetUserDefineKeyState( _
  ByVal State As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetResetUserDefineKeyState(
  Boolean State
);
```

解説

リセット時のユーザ定義キー無効の有効/無効を設定します。
[コントロールパネル]－[キーボード]－[その他]タブの「ユーザ定義キー:リセット時に無効にする」の状態を設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、SysGetResetUserDefineKeyState 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

State
ユーザ定義キーの有効/無効を指定します。

TRUE	:	有効
FALSE	:	無効

戻り値

TRUE	:	正常終了
FALSE	:	内部エラー
FUNCTION_UNSUPPORT	:	未サポートエラー

3.114 SysGetResetUserDefineKeyState

リセット時のユーザ定義キー無効の有効/無効を取得します。

```
[C++]
DWORD SysGetResetUserDefineKeyState()
```

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

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

解説

リセット時のユーザ定義キー無効の有効/無効を取得します。
[コントロールパネル]－[キーボード]－[その他]タブの「ユーザ定義キー:リセット時に無効にする」の状態を取得します。

パラメータ

なし

戻り値

TRUE	： 有効
FALSE	： 無効
FUNCTION_UNSUPPORT	： 未サポートエラー

3.115 SysSetEnableKeyMode

入力切替時のキーモード遷移の有効/無効を設定します。

```
[C++]
DWORD SysSetEnableKeyMode(
  DWORD EnableState
)
```

```
[Visual Basic]
Public Shared Function SysSetEnableKeyMode( _
  ByVal EnableState As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetEnableKeyMode(
  Int32 EnableState
);
```

解説

入力切替時のキーモード遷移の有効/無効を設定します。
[1]→[あ]→[ア]→[A]→[a]→[P]→[N]→[1]（繰り返し）の順に切り替わるのを[A]、[a]を無効にすると、[1]→[あ]→[ア]→[P]→[N]→[1]（繰り返し）の順に切り替わるようになります。（[P]、[N]は、DT-5100 のみ設定できます。）
ただし、SysSetInputMode を使用すれば、無効にされているキーモードでも切り替えることができます。
Device Emulator 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、SysGetEnableKeyMode 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

EnableState
有効にするキーモードを以下の値の論理和で指定します。

SYS_ENABLE_KEYMODE_NUM	：数字
SYS_ENABLE_KEYMODE_HIRA	：ひらがな(日本語版のみ)
SYS_ENABLE_KEYMODE_KANA	：カタカナ(日本語版のみ)
SYS_ENABLE_KEYMODE_ALPHA	：英大文字
SYS_ENABLE_KEYMODE_ALPHAS	：英小文字
SYS_ENABLE_KEYMODE_PHONE	：電話(DT-5200/DT-5300/IT-300/IT-G500)

戻り値

TRUE	：成功
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.116 SysGetEnableKeyMode

入力切替時のキーモード遷移の有効/無効を取得します。

```
[C++]
DWORD SysGetEnableKeyMode()
```

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

```
[C#]
public static Int32 SysGetEnableKeyMode();
```

解説

入力切替時のキーモード遷移の有効/無効を取得します。キーモードを変更したときに利用できる入力モードを取得します。

パラメータ

なし

戻り値

有効となっているキーモードの値を論理和で取得します。(値の詳細は SysSetEnableKeyMode 関数を参照してください。)
または、
FUNCTION_UNSUPPORT　：未サポートエラー

3.117 SysSetEnableTriggerKey

トリガーキー入力の有効/無効を設定します。

```
[C++]
DWORD SysSetEnableTriggerKey(
  BOOL EnableTriggerKey
)
```

```
[Visual Basic]
Public Shared Function SysSetEnableTriggerKey( _
  ByVal EnableTriggerKey As Boolean _
) As Int32
```

```
[C#]
public static Int32 SysSetEnableTriggerKey(
  Boolean EnableTriggerKey
);
```

解説

トリガーキー入力の有効/無効を設定します。トリガーキーを無効化した場合は、トリガーキーが押された状態でも他のキー入力を受け付けることができます。
Device Emulator 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、SysGetEnableTriggerKey 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

EnableTriggerKey
トリガーキーの有効/無効を指定します。
TRUE:　有効(デフォルト)
FALSE:　無効

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.118 SysGetEnableTriggerKey

トリガーキー入力の有効/無効を取得します。

```
[C++]
DWORD SysGetEnableTriggerKey()
```

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

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

解説

トリガーキー入力の有効/無効を取得します。

パラメータ

なし

戻り値

TRUE	：有効
FALSE	：無効
FUNCTION_UNSUPPORT	：未サポートエラー

3.119 SysSetFnKeyOperation

Fn キーによる特殊動作の有効/無効を個別に設定します。

```
[C++]
DWORD SysSetFnKeyOperation(  DWORD dwfnkey)
```

```
[Visual Basic]
Public Shared Function SysSetFnKeyOperation( _
  ByVal dwfnkey As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysSetFnKeyOperation (
  Int32 dwfnkey
);
```

解説

Fn キーによる特殊動作の有効/無効を個別に設定します。
Device Emulator では、設定値を内部変数として格納するため、何も動作しませんが、
SysGetFnKeyOperation 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

dwfnkey
動作を禁止する Fn キー特殊動作を論理和で指定します。

STATE_FN_NON	: すべて有効(デフォルト)
STATE_FN_0	: Fn+0
STATE_FN_1	: Fn+1
STATE_FN_2	: Fn+2
STATE_FN_3	: Fn+3
STATE_FN_4	: Fn+4
STATE_FN_5	: Fn+5
STATE_FN_6	: Fn+6
STATE_FN_7	: Fn+7
STATE_FN_8	: Fn+8
STATE_FN_9	: Fn+9
STATE_FN_MENU	: Fn+MENU
STATE_FN_LMULTI	: Fn+レフトマルチキー
STATE_FN_RMULTI	: Fn+ライトマルチキー
STATE_FN_ALL	: すべて無効

戻り値

TRUE	: 正常終了
FALSE	: 内部エラー
FUNCTION_UNSUPPORT	: 未サポートエラー

3.120 SysGetFnKeyOperation

Fn キーによる特殊動作が無効になっているキーの情報を取得します。

```
[C++]
DWORD SysGetFnKeyOperation(  )
```

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

```
[C#]
public static Int32 SysGetFnKeyOperation ( );
```

解説

Fn キーによる特殊動作が無効になっているキーの情報を論理和で取得します。
戻り値として STATE_FN_NON が返った場合は、全ての動作が有効になっています。（デフォルト）

戻り値

個別 Fn キーによる特殊動作が禁止状態のキーの値が論理和で返ります。（値の詳細は SysSetFnKeyOperation 関数を参照してください）
または、
FUNCTION_UNSUPPORT　：未サポートエラー

3.121 SysConvertGestureMessage

WM_GESTURE メッセージを解析し、マルチタッチジェスチャ情報を取得します。

```
[C++]
DWORD SysConvertGestureMessage(
  UINT    *pMessage,
  WPARAM  *pwParam,
  LPARAM  *plParam,
  DWORD   *pdwDistance,
  DWORD   *pdwAngle
)
```

```
[Visual Basic]
Public Shared Function SysConvertGestureMessage ( _
  ByRef pMessage As UInt32, _
  ByRef pwParam  As UInt32, _
  ByRef plMessage As Int32, _
  ByRef pdwDistance  As Int32, _
  ByRef pdwAngle As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysConvertGestureMessage (
  ref UInt32 *pMessage,
  ref UInt32 *pwParam,
  ref Int32 *plParam,
  ref Int32 *pdwDistance,
  ref Int32 *pdwAngle
);
```

解説

OS が発行する WM_GESTURE メッセージを解析し、マルチタッチジェスチャ情報を取得します。
アプリケーションでマルチタッチジェスチャを使用する場合は、WM_GESTURE メッセージ受信後に、必ず本関数を呼び出してください。
Device Emulator では、何も動作しません。

パラメータ

pMessage
OS が発行したメッセージを保持する変数のアドレスを指定します。

pwParam
OS が発行したメッセージに付随する WPARAM 変数のアドレスを指定します。
マルチタッチジェスチャと認識した場合は、以下のジェスチャ ID を取得します。シングルタッチジェスチャと認識した場合は、指定したジェスチャ ID をそのまま返します。

GID_ZOOM　　：ピンチイン/ピンチアウト
GID_ROTATE　　：ローテート

plParam
OS が発行したメッセージに付随する LPARAM 変数のアドレスを指定します。

pdwDistance
指の 2 点間の距離（0～800）を取得する変数のアドレスを指定します。
ジェスチャ ID が GID_ZOOM のときのみ距離を取得し、それ以外のときは 0 を取得します。

pdwAngle
指の 2 点の回転角（-180°～180°）を取得する変数のアドレスを指定します。
ジェスチャ ID が GID_ROTATE のときのみ回転角を取得し、それ以外のときは 0 を取得します。
回転角は GID_BEGIN 発生時を 0°とし、半時計回りを正、時計回りを負とした回転角を取得します。

戻り値

TRUE	：正常終了
FALSE	：内部エラー
FUNCTION_UNSUPPORT	：未サポートエラー

3.122 SysWaitForEvent

ドライバ等で発生させるイベントを待機します。

この関数は、Windows API の WaitForSingleObject 関数に相当する機能を、Visual Basic および C#でも利用できるように用意されているものです。このため、C++からは呼び出せません。

```
[Visual Basic]
Public Shared Function SysWaitForEvent( _
  ByVal EventHandle As System.IntPtr, _
  ByVal EventName As String, _
  ByVal TimeOut As Int32 _
) As Int32
```

```
[C#]
public static Int32 SysWaitForEvent(
  System.IntPtr EventHandle,
  string EventName,
  Int32 TimeOut
)
```

解説

本関数は、ドライバ等で発生するイベントを待機します。イベントが発生するか、指定したタイムアウト時間が経過するまで、この関数は戻りません。アプリケーションのメインスレッドとは別のスレッドで呼び出してください。

待機状態を強制終了させるには、アプリケーションのメインスレッドから SysTerminateWaitEvent 関数を呼び出してください。強制終了されると、この関数から戻ってきますが、強制終了されたのかどうかは、戻り値だけでは判別できません。イベントの状態を調べて確認してください。

この関数が必要になるのは、入出力の完了をイベントとして受け取りたい場合です。具体的には、レーザスキャナライブラリの OBRSetScanningNotification 関数で、完了通知の方法として「イベント通知」を指定した場合です。

パラメータ

EventHandle
待機するイベントのハンドルを指定します。EventName で指定する場合は、System.IntPtr.Zero を指定してください。

EventName
待機するイベントの名前を指定します。EventHandle に System.IntPtr.Zero 以外を指定した場合は、このパラメータは無視されます。

TimeOut
この関数で待機するときのタイムアウト時間を、1 ミリ秒単位で指定します。無限に待機する場合は、「Def.INFINITE(-1)」を指定してください。

戻り値

FUNCTION_UNSUPPORT	： 未サポートエラー
WAIT_TIMEOUT	： タイムアウト時間が経過しました
WAIT_OBJECT_0	： イベントが発生しました。または、強制終了されました。
WAIT_FAILED	： イベント待機できませんでした

3.123 SysTerminateWaitEvent

SysWaitForEvent 関数でのイベント待機を強制終了させます。
SysWaitForEvent 関数と同様に、この関数も、C++からは呼び出せません。

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

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

解説

本関数は、SysWaitForEvent 関数でのイベント待機を強制終了させます。SysWaitForEvent 関数呼び出し時、アプリケーションを終了させる時に、必ず本関数を呼び出してください。

戻り値

FUNCTION_UNSUPPORT	： 未サポートエラー
True	： 正常終了
False	： 内部エラー

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

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)