Digital
Human Machine Interface

Pro-face®

パネルコンピュータ
PL-5910シリーズ

APIリファレンスマニュアル

目次

目 次 1
1 概要 1-1
2 動作環境 2-1
3 必要ファイル 3-1
3.1 PL_Ioc 用ファイル 3-1
3.2 PL_Ras 用ファイル 3-2
3.3 PL_BLIoc 用ファイル 3-3
4 クラス内容 4-1
4.1 CPL_Ioctl クラス 4-1
4.2 CPL_Iocal クラス 4-1
4.3 CPL_SmiIoctl クラス 4-1
4.4 CPL_BLIoctl クラス 4-1
4.5 CPL_BLIocal クラス 4-2
5 Visual C 用関数仕様 5-1
5.1 Visual C 用関数一覧 5-1
5.1.1 PL_Ioc.dll 関数一覧 5-1
5.1.2 PL_Ras.dll 関数一覧 5-2
5.1.3 PL_BLIoc.dll 関数一覧 5-2
5.2 Visual C でのプログラム開発における注意事項 5-2
■サンプルプログラム例 5-2
5.3 Visual C 用関数仕様詳細 5-3
5.3.1 PL_Ioc.dll 関数 5-3
■ InitIoctl 5-3
■ EndIoctl 5-3
■ GetDrvHandle 5-3
■ CloseDrvHandle 5-3
■ GetDrvVersion 5-4
■ GetDrvVersionEx 5-4
■ GetMonitorSetup 5-5
■ GetVoltParam 5-6
■ GetCurrentVolt 5-7

■ GetTempParam 5-7
■ GetCurrentTemp 5-8
■ GetEvent 5-8
■ ClearEvent 5-9
■ SetWdtCounter 5-9
■ GetWdtCounter 5-10
■ SetWdtMask 5-10
■ GetWdtMask 5-11
■ StartWdt 5-11
■ StopWdt 5-11
■ RestartWdt 5-12
■ RunningWdt 5-12
■ SetWarningOut 5-13
■ GetWarningOut 5-13
■ GetUniversalIn 5-14
■ ClearUniversalIn 5-14
■ SetUniversalInMask 5-15
■ GetUniversalInMask 5-15
■ SetResetMask 5-16
■ GetResetMask 5-16
■ GetLightblowErr 5-16
■ SetWdtResetMask 5-17
■ GetWdtResetMask 5-17
■ SetWarningDOUT 5-17
■ GetWarningDOUT 5-18
■ GetWdtTimeout 5-18
■ ClearWdtTimeout 5-18
■ GetSmiDrvHandle 5-19
■ CloseSmiDrvHandle 5-19
■ GetSmiAryStatus 5-19
■ GetSmiDevStatus 5-20
5.3.2 PL_Ras.dll 関数 5-21
■ PlDevWordWrite 5-21
■ PlDevWordRead 5-21
5.3.3 PL_BLIoc.dll 関数 5-22
■ InitBLIoctl 5-22
■ EndBLIoctl 5-22
■ GetBLDrvHandle 5-22
■ CloseBLDrvHandle 5-22

■ GetBLDrvVersion 5-23
■ GetBLDrvVersionEx 5-23
■ SetBLControl 5-24
■ GetBLControl 5-24

6 Visual C++用関数仕様 6-1
6.1 Visual C++用関数一覧 6-1
6.1.1 PL_Ioc.dll関数一覧 6-1
6.1.2 PL_Ras.dll関数一覧 6-2
6.1.3 PL_BLIoc.dll関数一覧 6-2
6.2 Visual C++でのプログラム開発における注意事項 6-2
■サンプルプログラム例 6-2
6.3 Visual C++用関数仕様詳細 6-3
6.3.1 PL_Ioc.dll関数 6-3
■ GetDrvHandle 6-3
■ CloseDrvHandle 6-3
■ GetDrvVersion 6-4
■ GetDrvVersionEx 6-4
■ GetMonitorSetup 6-5
■ GetVoltParam 6-6
■ GetCurrentVolt 6-7
■ GetTempParam 6-8
■ GetCurrentTemp 6-8
■ GetEvent 6-9
■ ClearEvent 6-10
■ SetWdtCounter 6-10
■ GetWdtCounter 6-11
■ SetWdtMask 6-11
■ GetWdtMask 6-12
■ StartWdt 6-12
■ StopWdt 6-13
■ RestartWdt 6-13
■ RunningWdt 6-14
■ SetWarningOut 6-14
■ GetWarningOut 6-15
■ GetUniversalIn 6-16
■ ClearUniversalIn 6-17
■ SetUniversalInMask 6-17

■ GetUniversalInMask 6-18
■ SetResetMask 6-18
■ GetResetMask 6-19
■ GetLightblowErr 6-19
■ SetWdtResetMask 6-20
■ GetWdtResetMask 6-20
■ SetWarningDOUT 6-21
■ GetWarningDOUT 6-21
■ GetWdtTimeout 6-22
■ ClearWdtTimeout 6-22
■ GetSmiDrvHandle 6-23
■ CloseSmiDrvHandle 6-23
■ GetSmiAryStatus 6-24
■ GetSmiDevStatus 6-24
6.3.2 PL_Ras.dll関数 6-25
■ PlDevWordWrite 6-25
■ PlDevWordRead 6-25
6.3.3 PL_BLIoc.dll関数 6-26
■ GetBLDrvHandle 6-26
■ CloseBLDrvHandle 6-26
■ GetBLDrvVersion 6-27
■ GetBLDrvVersionEx 6-27
■ SetBLControl 6-28
■ GetBLControl 6-28
7 Visual Basic用関数仕様 7-1
7.1 Visual Basic用関数一覧 7-1
7.1.1 PL_Ioc.dll関数一覧 7-1
7.1.2 PL_Ras.dll関数一覧 7-2
7.1.3 PL_BLIoc.dll関数一覧 7-2
7.2 Visual Basicでのプログラム開発における注意事項 7-2
■サンプルプログラム例 7-2
7.3 Visual Basic用関数仕様詳細 7-3
7.3.1 PL_Ioc.dll関数 7-3
■ InitIoctl 7-3
■ EndIoctl 7-3
■ GetDrvHandle 7-3
■ CloseDrvHandle 7-3

■ GetDrvVersion 7-4
■ GetDrvVersionEx 7-4
■ GetMonitorSetup 7-5
■ GetVoltParam 7-6
■ GetCurrentVolt 7-7
■ GetTempParam 7-7
■ GetCurrentTemp 7-8
■ GetEvent 7-8
■ ClearEvent 7-9
■ SetWdtCounter 7-10
■ GetWdtCounter 7-10
■ SetWdtMask 7-10
■ GetWdtMask 7-11
■ StartWdt 7-11
■ StopWdt 7-11
■ RestartWdt 7-12
■ RunningWdt 7-12
■ SetWarningOut 7-13
■ GetWarningOut 7-13
■ GetUniversalIn 7-14
■ ClearUniversalIn 7-14
■ SetUniversalInMask 7-15
■ GetUniversalInMask 7-15
■ SetResetMask 7-16
■ GetResetMask 7-16
■ GetLightblowErr 7-16
■ SetWdtResetMask 7-17
■ GetWdtResetMask 7-17
■ SetWarningDOUT 7-17
■ GetWarningDOUT 7-18
■ GetWdtTimeout 7-18
■ ClearWdtTimeout 7-18
■ GetSmiDrvHandle 7-19
■ CloseSmiDrvHandle 7-19
■ GetSmiAryStatus 7-19
■ GetSmiDevStatus 7-20
7.3.2 PL_Ras.dll関数 7-21
■ PlDevWordWrite 7-21
■ PlDevWordRead 7-21

7.3.3 PL_BLIoc.dll関数 7-22
■ InitBLIoctl 7-22
■ EndBLIoctl 7-22
■ GetBLDrvHandle 7-22
■ GetBLDrvVersion 7-23
■ GetBLDrvVersionEx 7-23
■ SetBLControl 7-24
■ GetBLControl 7-24

1 概要

システムモニタ /RAS 機能、リモート RAS 機能、バックライト制御機能を PL-5910 シリーズで動作させるためのダイナミックリンクライブラリ(API-DLL)について説明します。

本書で説明する API-DLL には、PL_Ioc.dll、PL_Ras.dll および PL_BLIoc.dll の 3 種類があります。

■ PL_Ioc.dll

PL_Ioc.dll は、システムモニタ /RAS 機能にアクセスするための API-DLL です。アプリケーションは PL_Ioc.dll を経由して以下の機能を使用できます。

1. ドライバのバージョン管理
2. システムモニタ監視状態
3. 監視用パラメータ取得(電圧、温度)
4. システムモニタ現在情報(電圧、温度)
5. ウォッチドッグパラメータ
6. 警告処理
7. 汎用入力処理
8. リセット処理
9. イベント処理
10. ソフトミラー※1 状態の取得

■ PL_Ras.dll

PL_Ras.dll は、リモート RAS 機能の 1 つである共有メモリ機能にアクセスするための API-DLL です。アプリケーションは PL_Ras.dll を経由して以下の機能を使用できます。

1. 共有メモリの読み出し
2. 共有メモリへの書き込み

■ PL_BLIoc.dll

PL_BLIoc.dll は、バックライトを制御するための API-DLL です。アプリケーションは PL_BLIoc.dll を経由して以下の機能を使用できます。

1. バックライト制御

※1　　オプション品(開発中)です。

MEMO

このページは、空白です。
ご自由にお使いください。

2 動作環境

■オペレーティングシステム

CD-ROM に付属の API-DLL が動作する OS は以下のとおりです。

・Microsoft® WindowsNT® 4.0 (Service Pack 6a 以上)

・Microsoft® Windows® 2000 (Service Pack 4 以上)

また、それぞれのOS用の「システムモニタ/RASデバイスドライバ」が動作していなければなりません。

■対応言語

・Microsoft® Visual C Ver.6.0

・Microsoft® Visual C++ Ver.6.0

・Microsoft® Visual Basic Ver.6.0

MEMO

このページは、空白です。
ご自由にお使いください。

3 必要ファイル

3.1 PL_Ioc用ファイル

PL_Ioc.dllを使用するためには、各開発言語ごとに以下のファイルが必要です。

■Visual C

ファイル名	説明
PL_Iocif.h	ドライバインターフェイス定義インクルードファイル
PL_Ioc.LIB	ライブラリ定義ファイル
PL_Ioc.dll	ダイナミックリンクライブラリファイル

■Visual C++

ファイル名	説明
PL_Iocif.h	ドライバインターフェイス定義インクルードファイル
PL_Iocall.h	CPL_Iocallクラス定義インクルードファイル
PL_Ioctl.h	CPL_Ioctlクラス定義インクルードファイル
PL_Ioc.LIB	ライブラリ定義ファイル
PL_Ioc.dll	ダイナミックリンクライブラリファイル
PL_SmiIoctl.h	CPL_SmiIocrlクラス定義インクルードファイル (ソフトミラー使用時のみ)
Sm.h	ソフトミラー定義ファイル (ソフトミラー使用時のみ)

・ インクルードするヘッダファイルの順番は以下の通りです。

```
#include PL_Iocif.h
#include PL_Ioctl.h
#include Sm.h※1
#include PL_SmiIoctl.h※1
```

PL_Iocall.hは自動でインクルードされるので、直接インクルードしないでください。

■Visual Basic

ファイル名	説明
PL_Ioc.bas	ドライバインターフェイス定義ファイル
PL_Ioc.dll	ダイナミックリンクライブラリファイル

■DLLの格納先

作成したアプリケーションからPL_Ioc.dllを使用するために、以下の位置にDLLを格納する必要があります。

OS	位置
WindowsNT® 4.0/Windows® 2000	C:\Winnt\System32

※1 オプション品のソフトミラー(開発中)使用時のみ必要です。

3.2 PL_Ras用ファイル

PL_Ras.dllを使用するためには、各開発言語ごとに以下のファイルが必要です。

■ Visual C

ファイル名	説明
PL_Ras.h	ドライバインターフェイス定義インクルードファイル
PL_Ras.LIB	ライブラリ定義ファイル
PL_Ras.dll	ダイナミックリンクライブラリファイル

■ Visual C++

ファイル名	説明
PL_Ras.h	ドライバインターフェイス定義インクルードファイル
PL_Ras.LIB	ライブラリ定義ファイル
PL_Ras.dll	ダイナミックリンクライブラリファイル

■ Visual Basic

ファイル名	説明
PL_Ras.bas	ドライバインターフェイス定義ファイル
PL_Ras.dll	ダイナミックリンクライブラリファイル

■ DLLの格納先

作成したアプリケーションからPL_Ras.dllを使用するために、以下の位置にDLLを格納する必要があります。

OS	位置
WindowsNT® 4.0/Windows® 2000	C:\Winnt\System32

3.3 PL_BLIoc用ファイル

PL_BLIoc.dllを使用するためには、各開発言語ごとに以下のファイルが必要です。

■Visual C

ファイル名	説明
PL_BLIocif.h	ドライバインターフェイス定義インクルードファイル
PL_BLIoc.LIB	ライブラリ定義ファイル
PL_BLIoc.dll	ダイナミックリンクライブラリファイル

■Visual C++

ファイル名	説明
PL_BLIocif.h	ドライバインターフェイス定義インクルードファイル
PL_BLIocall.h	CPL_Iocallクラス定義インクルードファイル
PL_BLIoctl.h	CPL_Ioctlクラス定義インクルードファイル
PL_BLIoc.LIB	ライブラリ定義ファイル
PL_BLIoc.dll	ダイナミックリンクライブラリファイル

・ インクルードするヘッダファイルの順番は以下の通りです。

#include PL_BLIocif.h
#include PL_BLIoctl.h
PL_BLIocall.hは自動でインクルードされるので、直接インクルードしないでください。

■Visual Basic

ファイル名	説明
PL_BLIoc.bas	ドライバインターフェイス定義ファイル
PL_BLIoc.dll	ダイナミックリンクライブラリファイル

■DLLの格納先

作成したアプリケーションからPL_BLIoc.dllを使用するために、以下の位置にDLLを格納する必要があります。

OS	位置
WindowsNT® 4.0/Windows® 2000	C:\Winnt\System32

MEMO

このページは、空白です。
ご自由にお使いください。

4 クラス内容

4.1 CPL_Ioctl クラス

CPL_Ioctl クラスはCPL_Ioctl クラスでデバイスドライバアクセスするためのパラメータをセットします。

キーワード	型	変数名	説明
public	HANDLE	m_Drvhandle	デバイスドライバハンドル

4.2 CPL_Iocal クラス

CPL_Iocalでセットされたパラメータを使用し、DeviceIoControl(ドライバアクセス関数)を呼び出します。

ただし、このクラスはCPL_Ioctl から継承されているので直接使用することはありません。

キーワード	型	変数名	説明
public	HANDLE	m_h	デバイスドライバハンドル
public	LONG	m_long	実行する操作の制御コード
public	void *	m_ibp	入力データバッファアドレス
public	ULONG	m_ibsize	入力データバッファサイズ
public	void *	m_obp	出力データバッファアドレス
public	ULONG	m_obsize	出力データバッファサイズ
public	DWORD	m_retsize	実際の出力バイト数のアドレス
public	LPOVERLAPPED	m_ovlp	オーバーラップ構造体のアドレス

4.3 CPL_SmiIoctl クラス

CPL_SmiIoctl クラスは、CPL_SmiIoctl クラスでデバイスドライバアクセスをするためのパラメータをセットします。

ソフトミラーデバイスドライバを使用する場合にのみ、使用します。

キーワード	型	変数名	説明
public	HANDLE	m_Drvhandle	デバイスドライバハンドル

4.4 CPL_BLIoctl クラス

CPL_BLIoctl クラスは、CPL_BLIoctl クラスでデバイスドライバアクセスをするためのパラメータをセットします。

キーワード	型	変数名	説明
public	HANDLE	m_Drvhandle	デバイスドライバハンドル

4.5 CPL_BLIocal クラス

CPL_BLIocalでセットされたパラメータを使用し、DeviceIoControl(ドライバアクセス関数)を呼び出します。

ただし、このクラスはCPL_BLIoctlから継承されているので直接使用することはありません。

キーワード	型	変数名	説明
public	HANDLE	m_h	デバイスドライバハンドル
public	LONG	m_long	実行する操作の制御コード
public	void *	m_ibp	入力データバッファアドレス
public	ULONG	m_ibsize	入力データバッファサイズ
public	void *	m_obp	出力データバッファアドレス
public	ULONG	m_obsize	出力データバッファサイズ
public	DWORD	m_retsize	実際の出力バイト数のアドレス
public	LPOVERLAPPED	m_ovlp	オーバーラップ構造体のアドレス

5 Visual C用関数仕様

5.1 Visual C用関数一覧

5.1.1 PL_Ioc.dll関数一覧

関数名	説明
InitIoctl	CPL_Ioctlオブジェクト作成
EndIoctl	CPL_Ioctlオブジェクト破棄
GetDrvHandle	ドライバハンドル取得
CloseDrvHandle	GetDrvHandle取得ハンドル破棄
GetDrvVersion	ドライババージョン取得
GetDrvVersionEx	ハードウェアタイプ／ドライババージョン取得
GetMonitorSetup	モニタ許可／禁止設定取得
GetVoltParam	電圧監視用パラメータ取得
GetCurrentVolt	現在電圧値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
SetWdtCounter	ウォッチドッグタイマカウンタ値設定
GetWdtCounter	ウォッチドッグタイマカウンタ取得
SetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク設定
GetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク取得
StartWdt	ウォッチドッグタイマ開始
StopWdt	ウォッチドッグタイマ停止
RestartWdt	ウォッチドッグタイマ再開
RunningWdt	ウォッチドッグタイマ動作状況取得
SetWarningOut	警告出力設定
GetWarningOut	警告出力取得
GetUniversalIn	汎用入力取得
ClearUniversalIn	汎用入力ラッチ状態解除
SetUniversalInMask	汎用入力マスク設定
GetUniversalInMask	汎用入力マスク取得
SetResetMask	リセットマスク設定
GetResetMask	リセットマスク取得
GetLightblowErr	バックライト切れ状態取得
SetWdtResetMask	ウォッチドッグタイマのタイムアウト時のリセットマスクの設定
GetWdtResetMask	ウォッチドッグタイマのタイムアウト時のリセットマスクの取得
SetWarningDOUT	警告出力DOUT設定
GetWarningDOUT	警告出力DOUT取得
GetWdtTimeout	ウォッチドッグタイマのタイムアウト状態取得
ClearWdtTimeout	ウォッチドッグタイマのタイムアウト状態クリア
GetSmiDrvHandle	SoftMirror　ドライバハンドル取得
CloseSmiDrvHandle	SoftMirror　ドライバハンドル破棄
GetSmiAryStatus	SoftMirror　Array Status　取得
GetSmiDevStatus	SoftMirror　Device Status　取得

5.1.2 PL_Ras.dll関数一覧

関数名	説明
PlDevWordWrite	共有メモリへの書き込み
PlDevWordRead	共有メモリからの読み出し

5.1.3 PL_BLIoc.dll関数一覧

関数名	説明
InitBLIoctl	CPL_BLIoctlオブジェクト作成
EndBLIoctl	CPL_BLIoctlオブジェクト破棄
GetBLDrvHandle	ドライバハンドル取得
CloseBLDrvHandle	ドライバハンドル破棄
GetBLDrvVersion	ドライババージョン取得
GetBLDrvVersionEx	ハードウェアバージョン、ドライババージョン取得
SetBLControl	バックライト状態設定
GetBLControl	バックライト状態取得

5.2 Visual Cでのプログラム開発における注意事項

API-DLLを使用するためには､使用前にドライバオブジェクトの作成とデバイスハンドルの取得､使用後にデバイスハンドルの破棄とドライバオブジェクトの破棄を行う必要があります。以下に示す例を参考にプログラム開発を行ってください。

・ PlDevWordWriteとPlDevWordReadのみを使用する場合は、ドライバオブジェクトおよびデバイスハンドルの作成/破棄処理は必要ありません。

■サンプルプログラム例

```
//API-DLL使用例
//変数宣言
BOOL bRet;
int iRet;
HANDLE hDrv;
//ドライバオブジェクトの作成とデバイスハンドルの取得
//CPL_IOctlオブジェクトの作成
InitIoctl();
iRet=GetDrvHandle(&hDrv)
   ・
   ・
//DOUTへの出力
bRet=SetWarningDOUT(OUTPUT_ON);
   ・
   ・
//アプリケーション終了処理
//デバイスハンドルの破棄とドライバオブジェクトの破棄
bRet=CloseDrvHandle();
EndIoctl();
```

5.3 Visual C用関数仕様詳細

5.3.1 PL_Ioc.dll関数

■ InitIoctl

・呼び出し形式　void WINAPI InitIoctl(void)
・戻り値　なし
・引数　なし
・処理概要　CPL_Ioctlオブジェクトを作成する。作成されたオブジェクトはEndIoctl関数が呼ばれるまで破棄されない。
・例

```
InitIoctl();
```

■ EndIoctl

・呼び出し形式　void WINAPI EndIoctl(void)
・戻り値　なし
・引数　なし
・処理概要　InitIoctl関数で作成したオブジェクトを破棄する。
・例

```
EndIoctl();
```

■ GetDrvHandle

・呼び出し形式　int WINAPI GetDrvHandle(HANDLE *pHndl)
・戻り値　0:正常
1：エラー
・引数　(I/O) HANDLE *pHndl　デバイスドライバハンドルへのポインタ
・処理概要　デバイスドライバとのやり取りを行なうためのデバイスドライバハンドルを取得する。
・例

```
int   ret;
HANDLE     hndl;
ret = GetDrvHandle( &hndl );
```

・ システムモニタ/RASデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseDrvHandle

・ 呼び出し形式　BOOL WINAPI CloseDrvHandle(void)
・ 戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・ 処理概要　GetDrvHandle関数で取得したハンドルを破棄する。
・例

```
BOOL  ret;
// ハンドル破棄
ret = CloseDrvHandle();
```

■ GetDrvVersion

・呼び出し形式　BOOL WINAPI GetDrvVersion(int *pMajor, int *pMinor)

・戻り値　TRUE:正常
FALSE：エラー

・引数　(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ

・処理概要　ドライババージョン情報を取得する。

・例
```
BOOL  ret;
int   Major, Minor;
ret = GetDrvVersion( &Major, &Minor );
```

・ ドライババージョンが1.00の場合は、
Major:1　(10進数)
Minor：00　(10進数)
となります。

■ GetDrvVersionEx

・呼び出し形式　BOOL WINAPI GetDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)

・戻り値　TRUE　正常
FALSE　エラー

・引数　(I/O) int *pProduct　機種情報へのポインタ
(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ

・処理概要　ドライババージョン情報を取得する。

・例
```
BOOL ret;
int  Product, Major, Minor;
ret = GetDrvVersionEx(&Product, &Major, &Minor);
```

・ PL-5910シリーズでドライババージョンが1.00の場合は、
Product:4　(10進数)
Major：1　(10進数)
Minor:00　(10進数)
となります。

■ GetMonitorSetup

・呼び出し形式　BOOL WINAPI GetMonitorSetup(int Selector, int *pSetup)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	取得パラメータ	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
(I/O) int *pSetup	取得データへのポインタ	
	1:Enable	

・処理概要　現在のモニタ許可状態を取得する。

・例

```
BOOL  ret;
int   Setup;
// CPUコア電圧セットアップ状態取得
ret = GetMonitorSetup( MONITOR_VOLT_VCOREA, &Setup );
```

■ GetVoltParam

・呼び出し形式　BOOL WINAPI GetVoltParam
(int Selector, int *pULimit, int *pLLimit)

・戻り値　TRUE:正常
FALSE:エラー

・ 引数

(I) int Selector	取得パラメータ	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
(I/O) int *pULimit	電圧上限値(単位:mV)へのポインタ	
(I/O) int *pLLimit	電圧下限値(単位:mV)へのポインタ	

・処理概要　電圧監視用パラメータを取得する。

・例

```
BOOL  ret;
int   ULimit, LLimit;
// CPUコア電圧上限下限値取得
ret = GetVoltParam( MONITOR_VOLT_VCOREA, &ULimit, &LLimit );
```

・ 関数から取得されたデータはmV(ミリボルト)単位になっています。V（ボルト）単位で使用する時は下記のような変換を行ってください。
ボルト単位データ = ミリボルト単位データ/1000

■ GetCurrentVolt

・呼び出し形式　BOOL WINAPI GetCurrentVolt(int Selector, int *pData)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I) int Selector　取得パラメータ

MONITOR_VOLT_VCOREA	CPUコア電圧
MONITOR_VOLT_VCOREB	CPUコア電圧2
MONITOR_VOLT_P33	+3.3V電圧
MONITOR_VOLT_P50	+5.0V電圧
MONITOR_VOLT_P12	+12V電圧
MONITOR_VOLT_M50	-5.0V電圧
MONITOR_VOLT_M12	-12V電圧

(I/O) int *pData　電圧値(単位:mV)へのポインタ

・処理概要　現在の電圧値を取得する。

・例

```
BOOL  ret;
int   Data;
// CPUコア電圧値取得
ret = GetCurrentVolt( MONITOR_VOLT_VCOREA, &Data );
```

・ 関数から取得されたデータはmV(ミリボルト)単位になっています。V（ボルト）単位で使用する時は下記のような変換を行ってください。
ボルト単位データ = ミリボルト単位データ/1000

■ GetTempParam

・ 呼び出し形式　BOOL WINAPI GetTempParam(int Selector, int *pULimit)

・ 戻り値　TRUE:正常
FALSE:エラー

・ 引数　(I) int Selector　取得パラメータ

MONITOR_TEMP_SYSTEM	SYSTEM温度
MONITOR_TEMP_CPU	CPU温度

(I/O) int *pULimit　温度上限値(単位:℃)へのポインタ

・ 処理概要　温度監視用のパラメータを取得する。

・例

```
BOOL  ret;
int   ULimit;
// SYSTEM温度上限値取得
ret = GetTempParam( MONITOR_TEMP_SYSTEM, &ULimit );
```

■ GetCurrentTemp

・呼び出し形式　BOOL WINAPI GetCurrentTemp(int Selector, int *pData)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	取得パラメータ	
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
(I/O) int *pData	温度値(単位:℃)へのポインタ	

・処理概要　現在の温度値を取得する。

・例

```
BOOL  ret;
int   Data;
// SYSTEM温度値取得
ret = GetCurrentTemp( MONITOR_TEMP_SYSTEM, &Data );
```

■ GetEvent

・呼び出し形式　BOOL WINAPI GetEvent (int Selector, int *pEvent)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	取得パラメータ	
	EVENT_VOLT_VCOREA	CPUコア電圧
	EVENT_VOLT_VCOREB	CPUコア電圧2
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5.0V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M50	-5.0V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_TEMP_CPU	CPU温度
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_UNI_IN0	Universal Input 0
	EVENT_UNI_IN1	Universal Input 1
	EVENT_WDT_TIMEOUT	Watchdog Timeout
	EVENT_LIGHT_BLOW	Backlight管切れ
(I/O) int *pEvent	エラーイベント情報へのポインタ	
	ERROR_EVENT_OFF	エラーイベントなし
	ERROR_EVENT_ON	エラーイベントあり

・処理概要　マシンの電圧、温度の異常、また、Universal Input動作の情報(イベント)、Watchdog Timeout情報をチェックする。

・例

```
BOOL ret;
int Evnet;
//CPUコア電圧のエラーイベント情報取得
ret = GetEvent(EVENT_VOLT_VCOREA, &Event);
```

■ ClearEvent

・呼び出し形式　BOOL WINAPI ClearEvent (int Selector)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I)　int Selector　エラーイベントキャンセル対象パラメータ

EVENT_VOLT_VCOREA	CPUコア電圧
EVENT_VOLT_VCOREB	CPUコア電圧2
EVENT_VOLT_P33	+3.3V電圧
EVENT_VOLT_P50	+5V電圧
EVENT_VOLT_P12	+12V電圧
EVENT_VOLT_M50	-5V電圧
EVENT_VOLT_M12	-12V電圧
EVENT_TEMP_CPU	CPU温度
EVENT_TEMP_SYSTEM	SYSTEM温度
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	Backlight管切れ

・処理概要　エラーイベントをキャンセルする。

・例
```
BOOL ret;
//CPUコア電圧エラーイベントキャンセル
ret = ClearEvent(EVENT_VOLT_VCOREA);
```

■ SetWdtCounter

・呼び出し形式　BOOL WINAPI SetWdtCounter(int Counter)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I) int Counter　ウォッチドッグタイマの初期カウンタ(5～255)(単位:秒)

・処理概要　ウォッチドッグタイマの初期カウンタ値を設定する。

・例
```
BOOL  ret;
int Counter;
//ウォッチドッグタイマの初期カウンタ値を10秒に設定
Counter = 10;
ret = SetWdtCounter(Counter);
```

■ GetWdtCounter

・呼び出し形式　BOOL WINAPI GetWdtCounter(int *pCounter)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I/O) int *pCounter　ウォッチドッグタイマの初期カウンター値(単位:秒)へのポインタ

・処理概要　現在のウォッチドッグタイマの初期カウンタ値を取得する。

・例

```
BOOL  ret;
int   Counter;
ret = GetWdtCounter( &Counter );
```

■ SetWdtMask

・呼び出し形式　BOOL WINAPI SetWdtMask(int Selector, int Mask)

・戻り値　TRUE:正常
FALSE:エラー

・ 引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I) int Mask	マスク情報	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　ウォッチドッグタイマタイムアウト時に出力する警告のマスクを設定する。

・例

```
BOOL  ret;
int Mask
// LAMP出力をマスクする
ret = SetWdtMask( WARNING_LAMP, MASK_ON );
// ALARM出力のマスクを解除する
ret = SetWdtMask( WARNING_ALARM, MASK_OFF );
```

■ GetWdtMask

・呼び出し形式　BOOL WINAPI GetWdtMask(int Selector, int *pMask)

・ 戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I/O) int *pMask	マスク情報へのポインタ	
	MASK_OFF	マスク解除
	MASK_ON	マスク

・処理概要　ウォッチドッグタイマタイムアウト時の警告出力マスク情報を取得する。

・例

```
BOOL  ret;
int   Mask;
// LAMP のマスク情報取得
ret = GetWdtMask( WARNING_LAMP, &Mask );
// ALARM のマスク情報取得
ret = GetWdtMask( WARNING_ALARM, &Mask );
```

■ StartWdt

・呼び出し形式　BOOL WINAPI StartWdt (void)

・戻り値　TRUE:正常
FALSE:エラー

・引数　なし

・処理概要　ウォッチドッグタイマのカウントダウンを開始する。

・例

```
BOOL ret:
ret = StartWdt();
```

■ StopWdt

・呼び出し形式　BOOL WINAPI StopWdt (void)

・戻り値　TRUE:正常
FALSE:エラー

・引数　なし

・処理概要　ウォッチドッグタイマのカウントダウンを停止する。

・例

```
BOOL ret:
ret = StopWdt();
```

■ RestartWdt

・呼び出し形式　BOOL WINAPI RestartWdt (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　ウォッチドッグタイマのカウント値を初期値に戻し、再カウントダウンを始める。
・例
```
BOOL ret:
ret = RestartWdt();
```

・ RestartWdtはStartWdtでカウントダウンを開始した後のみ使用できます。タイムアウトした後にRestartWdtを使用する場合はClearWdtでタイムアウト状態を解除し、再度StartWdtでカウントダウンを開始してください。

■ RunningWdt

・呼び出し形式　BOOL WINAPI RunningWdt (int *pRunFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pRunFlag　ウォッチドッグタイマの動作状態へのポインタ
WATCHDOG_STOP　停止中
WATCHDOG_COUNTDOWN　カウントダウン中
・処理概要　ウォッチドッグタイマの動作状態を取得する。
・例
```
BOOL ret；
int RunFlag;
ret = RunningWdt(&RunFlag);
```

■ SetWarningOut

・呼び出し形式　BOOL WINAPI SetWarningOut (int Selector, int WarnOut)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I) int WarnOut	出力状態	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

・処理概要　設定項目(LAMP、ALARM)の警告情報を設定する。

・例

```
BOOL ret:
//LAMPの出力状態をONに設定
ret = SetWarningOut(WARNING_LAMP, OUTPUT_ON);
//ALARMの出力状態をOFFに設定
ret = SetWarningOut(WARNING_ALARM, OUTPUT_OFF);
```

■ GetWarningOut

・呼び出し形式　BOOL WINAPI GetWarningOut (int Selector, int *pWarnOut)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I/O) int *pWarnOut	出力状態へのポインタ	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

・処理概要　現在の設定項目(LAMP,ALARM)の警告状態を取得する。

・例

```
BOOL ret:
int = WarnOut;
//LAMPの出力状態取得
ret = GetWarningOut (WARNING_LAMP, &WarnOut);
//ALARMの出力状態取得
ret = GetWarningOut (WARNING_ALARM, &WarnOut);
```

■ GetUniversalIn

・呼び出し形式　BOOL WINAPI GetUniversalIn (int Selector, int *pUniIn)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
(I/O) int*pUniIn	入力状態へのポインタ	
	INPUT_OFF	入力なし
	INPUT_ON	入力あり

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態を取得する。

・例

```
BOOL ret;
int UniIn;
//Universal Input 0の入力状態取得
ret = GetUniversalIn(PORT_UNI0, &UniIn);
//Universal Input 1の入力状態
ret = GetUniversalIn(PORT_UNI1, &UniIn);
```

■ ClearUniversalIn

・呼び出し形式　BOOL WINAPI ClearUniversalIn (int Selector)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態をキャンセルする。

・例

```
BOOL ret:
//Universal Input 0の入力状態をキャンセルする
ret = ClearUniversalIn(PORT_UNI0);
//Universal Input 1の入力状態をキャンセルする
ret = ClearUniversalIn(PORT_UNI1);
```

■ SetUniversalInMask

・呼び出し形式　BOOL WINAPI SetUniversalInMask (in Selector, int Mask)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
(I) int Mask	マスク情報	
	MASK_OFF	マスク解除
	MASK_OFF	マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を設定する。

・例

```
BOOL ret;
//Universal Input 0をマスク解除
ret = SetUniversalInMask(PORT_UNI0, MASK_OFF);
//Universal Input 1をマスク
ret = SetUniversalInMask(PORT_UNI1, MASK_ON);
```

■ GetUniversalInMask

・呼び出し形式　BOOL WINAPI GetUniversalInMask(int Selector, int *pMask)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
(I/O) int *pMask	マスク情報へのポインタ	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を取得する。

・例

```
BOOL  ret;
int   Mask;
// Universal Input0マスク情報取得
ret = GetUniversalInMask( PORT_UNI0, &Mask );
// Universal Input1マスク情報取得
ret = GetUniversalInMask( PORT_UNI1, &Mask );
```

■ SetResetMask

・呼び出し形式　BOOL WINAPI SetResetMask (int Mask)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I)　int Mask　マスク情報
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　リセットマスクを設定する
・例

```
BOOL ret:
//リセットマスク解除
ret = SetResetMask(MASK_OFF);
```

■ GetResetMask

・呼び出し形式　BOOL WINAPI GetResetMask(int *pMask)
・戻り値　TRUE:正常
FALSE:エラー
・ 引数　(I/O) int *pMask　マスク情報へのポインタ
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　現在のリセットマスク情報を取得する。
・例

```
BOOL  ret;
int   Mask;
ret = GetResetMask( &Mask );
```

■ GetLightblowErr

・呼び出し形式　BOOL WINAPI GetLightblowErr (int *pLightErr)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O)int *pLightErr　エラー出力情報へのポインタ
BACKLIGHT_OK　バックライト正常
BACKLIGHT_ERR　バックライト管切れ
・処理概要　現在のLCDバックライト管切れエラー出力を取得する。
・例

```
BOOL ret;
int LightErr;
//バックライト管切れ状態取得
ret = GetLightblowErr(&LightErr);
```

■ SetWdtResetMask

・呼び出し形式　BOOL WINAPI SetWdtResetMask(int Mask)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I) int Mask　設定するマスク状態
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　ウォッチドッグタイマによるリセットマスクの状態を設定する。
・例
```
BOOL ret;
ret = SetWdtResetMask(MASK_ON);
```

■ GetWdtResetMask

・呼び出し形式　BOOL WINAPI GetWdtResetMask (int *pMask)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pMask　マスク状態へのポインタ
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　ウォッチドッグタイマによるリセットマスクの状態を取得する。
・例
```
BOOL ret;
int Mask;
ret = GetWdtResetMask(&Mask)
```

■ SetWarningDOUT

・呼び出し形式　BOOL WINAPI SetWarningDOUT (int WarningOut)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I) int WarningOut　出力状態
OUTPUT_OFF　出力OFF
OUTPUT_ON　出力ON
・処理概要　現在の設定項目(DOUT)の警告状態を設定する。
・例
```
BOOL ret;
//DOUTの出力状態をOFFに設定
ret = SetWarningDOUT(OUTPUT_OFF);
```

■ GetWarningDOUT

・呼び出し形式　BOOL WINAPI GetWarningDOUT(int *pWarningOut)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pWarningOut　出力状態へのポインタ
OUTPUT_OFF　出力 OFF
OUTPUT_ON　出力 ON
・処理概要　現在の設定項目(DOUT)の警告状態を取得する。
・例

```
BOOL ret;
int   WarningOut
//DOUT の出力状態を取得
ret = GetWarningDOUT(&WarningOut);
```

■ GetWdtTimeout

・呼び出し形式　BOOL WINAPI GetWdtTimeout(int *pTimebuf)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pTimebuf　ウォッチドッグタイムアウト状態へのポインタ
TIMEOUT_OK　タイムアウトしていない
TIMEOUT_ERROR　タイムアウトしている
・処理概要　ウォッチドッグのタイムアウト状態を取得する
・例

```
BOOL  ret;
int   Timebuf;
// ウォッチドッグのタイムアウト状態取得
ret = GetWdtTimeout( &Timebuf );
```

■ ClearWdtTimeout

・呼び出し形式　BOOL WINAPI ClearWdtTimeout (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　ウォッチドッグのタイムアウト状態を解除する。
・例

```
BOOL ret;
//ウォッチドッグタイムアウト状態解除
ret = GetWdtTimeout();
```

■ GetSmiDrvHandle

・呼び出し形式　BOOL WINAPI GetSmiDrvHandle (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　ソフトミラーデバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例
```
BOOL  ret;
ret = GetSmiDrvHandle();
```

・ ソフトミラーデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseSmiDrvHandle

・呼び出し形式　BOOL WINAPI CloseSmiDrvHandle (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　GetSmiDrvHandle関数で取得したハンドルを破棄する。
・例
```
BOOL ret;
//ハンドル破棄
ret = CloseSmiDrvHandle();
```

■ GetSmiAryStatus

・呼び出し形式　BOOL WINAPI GetSmiAryStatus (int *pStatus)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pStatus　ミラーディスクへのポインタ

ARYSTAT_GOOD	正常
ARYSTAT_UNCONFIG	未構築状態
ARYSTAT_REBUILD	再構築中
ARYSTAT_REDUCE	縮退中
ARYSTAT_DEAD	ミラー状態破壊

・処理概要　ソフトミラーの状態を取得する。
・例
```
BOOL ret;
int Status;
//ソフトミラーの状態取得
ret = GetSmiAryStatus (&Status);
```

■ GetSmiDevStatus

・呼び出し形式　BOOL WINAPI GetSmiDevStatus (int Id, int *pType, int *pStatus)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Id	デバイスID	
	0：Master HDD	
	1：Slave HDD	
(I/O) int *pTypeデバイスタイプ		
	ATADEVICE	HDD(ATAデバイス)
	ATAPIDEVICE	CD-ROM
	NODEVICE	未接続
(I/O) int *pStatus	デバイスステータス	
	DEVSTAT_GOOD	正常
	DEVSTAT_NOTEXIST	未接続
	DEVSTAT_BROKEN	故障

・処理概要　ソフトミラーのデバイス状態を取得する。

・例

```
BOOL ret;
int   Id, Type, Status;
//デバイス状態の取得
Id = 0;
ret = GetSmiDevStatus(Id, &Type, &Status);
```

5.3.2 PL_Ras.dll関数

■ PlDevWordWrite

・呼び出し形式　long PlDevWordWrite(long Addr, long wData)
・戻り値　0:正常
0以外:エラー
・引数　(I) long Addr　書き込むメモリのワードアドレス
(I) long wData　書き込むデータ(0～65535)
・処理概要　共有メモリへの書き込みを行います。
・例
```
// アドレス255へデータ255を書き込む
long ret;
ret = PlDevWordWrite(255, 255);
```

■ PlDevWordRead

・呼び出し形式　long PlDevWordRead(long Addr, long *wData)
・戻り値　0:正常
0以外:エラー
・引数　(I) long Addr　読み出すメモリのワードアドレス
(I/O) long *wData　読み出すデータへのポインタ(0～65535)
・処理概要　共有メモリへの読み出しを行います。
・例
```
// アドレス255のデータ読み出し
long ret;
long wData;
ret = PlDevWordRead(255, &wData);
```

5.3.3 PL_BLIoc.dll関数

■ InitBLIoctl

・呼び出し形式　　void WINAPI InitBLIoctl(void)
・戻り値　　なし
・引数　　なし
・処理概要　　CPL_BLIoctl オブジェクトを作成する。作成されたオブジェクトは EndBLIoctl 関数が呼ばれるまで破棄されない。
・例　　InitBLIoctl();

■ EndBLIoctl

・呼び出し形式　　void WINAPI EndBLIoctl(void)
・戻り値　　なし
・引数　　なし
・処理概要　　InitBLIoctl関数で作成したオブジェクトを破棄する。
・例　　EndBLIoctl();

■ GetBLDrvHandle

・呼び出し形式　　int WINAPI GetBLDrvHandle(HANDLE *pHndl)
・戻り値　　0:正常
1:エラー
・引数　　(I/O) HANDL *pHndl　　デバイスドライバハンドルへのポインタ
・処理概要　　デバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例

```
int   ret;
HANDLE     hndl;
ret = GetBLDrvHandle( &hndl );
```

・ バックライトコントロールデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseBLDrvHandle

・呼び出し形式　　BOOL WINAPI CloseBLDrvHandle(void)
・戻り値　　TRUE:正常
FALSE:エラー
・引数　　なし
・処理概要　　GetBLDrvHandleで取得したハンドルを破棄する。
・例

```
BOOL ret;
ret = CloseBLDrvHandle();
```

■ GetBLDrvVersion

・呼び出し形式　BOOL WINAPI GetBLDrvVersion(int *pMajor, int *pMinor)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ
・処理概要　ドライババージョン情報を取得する。
・例

```
BOOL  ret;
int   Major, Minor;
ret = GetBLDrvVersion( &Major, &Minor );
```

・ ドライババージョンが1.00の場合は
Major:1　(10進数)
Minor:00　(10進数)
となります。

■ GetBLDrvVersionEx

・呼び出し形式　BOOL WINAPI GetBLDrvVersionEx
(int *pProduct, int *pMajor, int *pMinor)
・戻り値　TRUE:正常
FALSE:エラー
(I/O) int *pProduct　機種情報へのポインタ
(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ
・処理概要　ドライババージョン、機種情報を取得する。
・例

```
BOOL ret;
int Product, Major, Minor
ret = GetBLDrvVersionEx(&Product, &Major, &Minor);
```

・ PL-5910シリーズでドライババージョンが1.0の場合は、
Product : 4　(10進数)
Major : 1　(10進数)
Minor : 0　(10進数)
となります。

■ SetBLControl

・呼び出し形式　BOOL WINAPI SetBLControl(int BLFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I)　int BLFlag　設定パラメータ
BACKLIGHT_OFF バックライトOFF
BACKLIGHT_ON　バックライトON
・処理概要　バックライトのON/OFFを設定する。
・例

```
BOOL ret;
//バックライトコントロールON設定
ret = SetBLControl( BACKLIGHT_ON );
```

■ GetBLControl

・呼び出し形式　BOOL WINAPI GetBLControl(int *pBLFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pBLFlag　バックライト状態へのポインタ
BACKLIGHT_OFF　バックライトOFF
BACKLIGHT_ON　バックライトON
・処理概要　バックライトコントロール状態を取得する。
・例

```
BOOL  ret;
int   BLFlag;
//バックライトコントロール状態取得
ret = GetBLControl( &BLFlag );
```

6 Visual C++用関数仕様

6.1 Visual C++用関数一覧

6.1.1 PL_Ioc.dll関数一覧

関数名	説明
GetDrvHandle	ドライバハンドル取得
CloseDrvHandle	GetDrvHandle取得ハンドル破棄
GetDrvVersion	ドライババージョン取得
GetDrvVersionEx	ハードウェアタイプ／ドライババージョン取得
GetMonitorSetup	モニタ許可／禁止設定取得
GetVoltParam	電圧監視用パラメータ取得
GetCurrentVolt	現在電圧値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
SetWdtCounter	ウォッチドッグタイマカウンタ値設定
GetWdtCounter	ウォッチドッグタイマカウンタ取得
SetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク設定
GetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク取得
StartWdt	ウォッチドッグタイマ開始
StopWdt	ウォッチドッグタイマ停止
RestartWdt	ウォッチドッグタイマ再開
RunningWdt	ウォッチドッグタイマ動作状況取得
SetWarningOut	警告出力設定
GetWarningOut	警告出力取得
GetUniversalIn	汎用入力取得
ClearUniversalIn	汎用入力ラッチ状態解除
SetUniversalInMask	汎用入力マスク設定
GetUniversalInMask	汎用入力マスク取得
SetResetMask	リセットマスク設定
GetResetMask	リセットマスク取得
GetLightblowErr	バックライト切れ状態取得
SetWdtResetMask	ウォッチドッグタイマのタイムアウト時のリセットマスクの設定
GetWdtResetMask	ウォッチドッグタイマのタイムアウト時のリセットマスクの取得
SetWarningDOUT	警告出力DOUT設定
GetWarningDOUT	警告出力DOUT取得
GetWdtTimeout	ウォッチドッグタイマのタイムアウト状態取得
ClearWdtTimeout	ウォッチドッグタイマのタイムアウト状態クリア
GetSmiDrvHandle	SoftMirror　ドライバハンドル取得
CloseSmiDrvHandle	SoftMirror　ドライバハンドル破棄
GetSmiAryStatus	SoftMirror　Array Status　取得
GetSmiDevStatus	SoftMirror　Device Status　取得

6.1.2 PL_Ras.dll関数一覧

関数名	説明
PlDevWordWrite	共有メモリへの書き込み
PlDevWordRead	共有メモリからの読み出し

6.1.3 PL_BLIoc.dll関数一覧

関数名	説明
GetBLDrvHandle	ドライバハンドル取得
CloseBLDrvHandle	ドライバハンドル破棄
GetBLDrvVersion	ドライババージョン取得
GetBLDrvVersionEx	ハードウェアバージョン、ドライババージョン取得
SetBLControl	バックライトコントロール設定
GetBLControl	バックライトコントロール状態取得

6.2 Visual C++でのプログラム開発における注意事項

API-DLLを使用するためには､使用前にドライバオブジェクトの作成とデバイスハンドルの取得､使用後にデバイスハンドルの破棄とドライバオブジェクトの破棄を行う必要があります。以下に示す例を参考にプログラム開発を行ってください。

・ PlDevWordWriteとPlDevWordReadのみを使用する場合は、デバイスハンドルの作成/破棄処理は必要ありません。

■サンプルプログラム例

```
//API-DLL使用例
//変数宣言
BOOL bRet;
int iRet;
//デバイスハンドルの取得
CPLIoctl m_Ioc
iRet=m_Ioc.GetDrvHandle();
   ・
   ・
//DOUTへの出力
bRet=m_Ioc.SetWarningDOUT(OUTPUT_ON);
   ・
   ・
//デバイスハンドルの破棄
bRet=CloseDrvHandle();
```

6.3 Visual C++用関数仕様詳細

6.3.1 PL_Ioc.dll関数

■ GetDrvHandle

・呼び出し形式　int GetDrvHandle(HANDLE *pHndl)

・戻り値　0:正常
1:エラー

・引数　(I/O) HANDLE *pHandl　デバイスハンドルへのポインタ

・処理概要　デバイスドライバとのやり取りを行なうためのデバイスドライバハンドルを取得する。取得されたハンドルはメンバ変数m_handleに格納される。

・例1
```
CPL_Ioctl  m_Ioc;
int ret;
ret = m_Ioc.GetDrvHandle();
```

・例2
```
int ret;
HANDLE hndl;
ret = ::GetDrvHandle( &hndl );
```

MEMO
・ システムモニタ/RASデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseDrvHandle

・ 呼び出し形式　BOOL CloseDrvHandle(void)

・ 戻り値　TRUE:正常
FALSE:エラー

・引数　なし

・処理概要　GetDrvHandle関数で取得したハンドルを破棄する。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL       ret;
// ハンドル破棄
ret = m_Ioc.CloseDrvHandle();
```

・例2
```
BOOL  ret;
// ハンドル破棄
ret = ::CloseDrvHandle();
```

■ GetDrvVersion

・呼び出し形式　　BOOL GetDrvVersion(int *pMajor, int *pMinor)
・戻り値　　TRUE:正常
　　FALSE：エラー
・引数　　(I/O) int *pMajor　　バージョン情報(Major,0～99)へのポインタ
　　(I/O) int *pMinor　　バージョン情報(Minor,0～99)へのポインタ
・処理概要　　ドライババージョン情報を取得する。
・例1　　
```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Major, Minor;
ret = m_Ioc.GetDrvVersion( &Major, &Minor );
```
・例2　　
```
BOOL ret;
int Major, Minor;
ret = ::GetDrvVersion( &Major, &Minor );
```

・ ドライババージョンが1.00の場合は、
Major:1　(10進数)
Minor：00　(10進数)
となります。

■ GetDrvVersionEx

・呼び出し形式　　BOOL GetDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)
・戻り値　　TRUE：正常
　　FALSE：エラー
・引数　　(I/O) int *pProduct　　機種情報へのポインタ
　　(I/O) int *pMajor　　バージョン情報(Major,0～99)へのポインタ
　　(I/O) int *pMinor　　バージョン情報(Minor,0～99)へのポインタ
・処理概要　　ドライババージョン情報を取得する。
・例1　
```
BOOL  ret;
int   Product, Major, Minor;
ret = GetDrvVersionEx(&Product, &Major, &Minor)
```
・例2　
```
CPL_Ioctl  m_Ioctl;
BOOL  ret;
int   Product, Major, Minor;
ret = m_Ioctl.GetDrvVersionEx(&Product, &Major, &Minor);
```

・ PL-5910シリーズでバージョンが1.00の場合は、
Product:4　(10進数)
Major:1　(10進数)
Minor：00　(10進数)
となります。

■ GetMonitorSetup

・呼び出し形式　BOOL GetMonitorSetup(int Selector, int *pSetup)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	取得パラメータ	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_TEMP_CPU	CPU温度
	MONITOR_TEMP_SYSTEM	SYSTEM温度
(I/O) int *pSetup	取得データへのポインタ	
	1:Enable	

・処理概要　現在のモニタ許可状態を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Setup;
// CPUコア電圧取得セットアップ状態取得
ret = m_Ioc.GetMonitorSetup( MONITOR_VOLT_VCOREA, &Setup );
```

・例2

```
BOOL ret;
int Setup;
// CPUコア電圧取得セットアップ状態取得
ret = ::GetMonitorSetup( MONITOR_VOLT_VCOREA, &Setup );
```

■ GetVoltParam

・呼び出し形式　BOOL GetVoltParam (int Selector, int *pULimit, int *pLLimit)

・戻り値　TRUE:正常
FALSE:エラー

・ 引数

(I) int Selector	取得パラメータ	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
(I/O) int *pULimit	電圧上限値(単位:mV)へのポインタ	
(I/O) int *pLLimit	電圧下限値(単位:mV)へのポインタ	

・処理概要　電圧監視用パラメータを取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int ULimit, LLimit;
// CPUコア電圧上限下限値取得
ret = m_Ioc.GetVoltParam( MONITOR_VOLT_VCOREA, &ULimit, &LLimit );
```

・例2

```
BOOL  ret;
int ULimit, LLimit;
// CPUコア電圧上限下限値取得
ret = ::GetVoltParam( MONITOR_VOLT_VCOREA, &ULimit, &LLimit );
```

・関数から取得されたデータはmV(ミリボルト)単位になっています。V(ボルト)単位で使用する時は下記のような変換をする必要があります。
ボルト単位データ = ミリボルト単位データ/1000

■ GetCurrentVolt

・呼び出し形式　BOOL GetCurrentVolt(int Selector, int *pData)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I) int Selector　取得パラメータ

MONITOR_VOLT_VCOREA	CPUコア電圧
MONITOR_VOLT_VCOREB	CPUコア電圧2
MONITOR_VOLT_P33	+3.3V電圧
MONITOR_VOLT_P50	+5.0V電圧
MONITOR_VOLT_P12	+12V電圧
MONITOR_VOLT_M50	-5.0V電圧
MONITOR_VOLT_M12	-12V電圧

(I/O) int *pData　電圧値(単位:mV)へのポインタ

・処理概要　現在の電圧値を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Data;
// CPUコア電圧値取得
ret = m_Ioc.GetCurrentVolt( MONITOR_VOLT_VCOREA &Data );
```

・例2

```
BOOL  ret;
int Data;
// CPUコア電圧値取得
ret = ::GetCurrentVolt( MONITOR_VOLT_VCOREA, &Data );
```

・ 関数から取得されたデータはmV(ミリボルト)単位になっています。V（ボルト）単位で使用する時は下記のような変換を行ってください。
ボルト単位データ = ミリボルト単位データ/1000

■ GetTempParam

・ 呼び出し形式　BOOL GetTempParam(int Selector, int *pULimit)
・ 戻り値　TRUE:正常
FALSE:エラー
・ 引数　(I) int Selector　取得パラメータ
MONITOR_TEMP_CPU　CPU温度
MONITOR_TEMP_SYSTEM　SYSTEM温度
(I/O) int *pULimit　温度上限値(単位:℃)へのポインタ
・ 処理概要　温度監視用のパラメータを取得する。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int ULimit;
// SYSTEM温度上限値取得
ret = m_Ioc.GetTempParam( MONITOR_TEMP_SYSTEM, &ULimit );
```

・例2

```
BOOL  ret;
int   ULimit;
ret = ::GetTempParam( MONITOR_TEMP_SYSTEM, &ULimit );
```

■ GetCurrentTemp

・呼び出し形式　BOOL GetCurrentTemp (int Selector, int *pData)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I)　int Selector　取得パラメータ
MONITOR_TEMP_SYSTEM　SYSTEM温度
MONITOR_TEMP_CPU　CPU温度
(I/O) int *pData　温度値(単位:℃)へのポインタ
・処理概要　現在の温度値を取得する。
・例1

```
CPL_Ioctl  m_Ioctl;
BOOL ret;
int Data;
// SYSTEM温度値取得
ret = m_Ioctl.GetCurrentTemp( MONITOR_TEMP_SYSTEM, &Data );
```

・例2

```
BOOL  ret;
int   Data;
// SYSTEM温度値取得
ret = ::GetCurrentTemp( MONITOR_TEMP_SYSTEM, &Data );
```

■ GetEvent

・呼び出し形式　BOOL GetEvent (int Selector, int *pEvent)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	取得パラメータ	
	EVENT_VOLT_VCOREA	CPUコア電圧
	EVENT_VOLT_VCOREB	CPUコア電圧2
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5.0V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M50	-5.0V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_TEMP_CPU	CPU温度
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_UNI_IN0	Universal Input 0
	EVENT_UNI_IN1	Universal Input 1
	EVENT_WDT_TIMEOUT	Watchdog Timeout
	EVENT_LIGHT_BLOW	Backlight管切れ
(I/O) int *pEvent	エラーイベント情報へのポインタ	
	ERROR_EVENT_OFF	エラーイベントなし
	ERROR_EVENT_ON	エラーイベントあり

・処理概要　マシンの電圧、温度の異常、また、Universal Input動作の情報(イベント)、Watchdog Timeout情報をチェックする。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Event;
// CPUコア電圧のエラーイベント情報取得
ret = m_Ioc.GetEvent( EVENT_VOLT_VCOREA, &Event );
```

・例2

```
BOOL ret;
int Event;
// CPLコア電圧のエラーイベント情報取得
ret = ::GetEvent( EVENT_VOLT_VCOREA, &Event );
```

■ ClearEvent

・呼び出し形式　BOOL ClearEvent (int Selector)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I)　int Selector　エラーイベントキャンセル対象パラメータ

EVENT_VOLT_VCOREA	CPUコア電圧
EVENT_VOLT_VCOREB	CPUコア電圧2
EVENT_VOLT_P33	+3.3V電圧
EVENT_VOLT_P50	+5.0V電圧
EVENT_VOLT_P12	+12V電圧
EVENT_VOLT_M50	-5.0V電圧
EVENT_VOLT_M12	-12V電圧
EVENT_TEMP_CPU	CPU温度
EVENT_TEMP_SYSTEM	SYSTEM温度
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	バックライト管切れ

・処理概要　エラーイベントをキャンセルする。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL  ret;
// CPUコア電圧エラーイベントキャンセル
ret = m_Ioc.ClearEvent( EVENT_VOLT_VCOREA );
```

・例2
```
BOOL  ret;
// CPUコア電圧エラーイベントキャンセル
ret = ::ClearEvent( EVENT_VOLT_VCOREA );
```

■ SetWdtCounter

・呼び出し形式　BOOL SetWdtCounter(int Counter)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I)　int Counter　ウォッチドッグタイマの初期カウンタ(5～255)(単位:秒)

・処理概要　ウォッチドッグタイマの初期カウンタ値を設定する。

・例1
```
CPL_Ioctl m_Ioc;
BOOL  ret;
//ウォッチドッグタイマの初期カウンタ値を10秒に設定
Counter = 10;
ret = m_Ioctl.SetWdtCounter(Counter);
```

・例2
```
BOOL  ret;
int Counter;
//ウォッチドッグタイマの初期カウンタ値を10秒に設定
Counter = 10;
ret = ::SetWdtCounter(Counter);
```

■ GetWdtCounter

・呼び出し形式　BOOL GetWdtCounter(int *pCounter)

・戻り値
TRUE:正常
FALSE:エラー

・引数　(I/O) int *pCounter　ウォッチドッグタイマの初期カウンター値(5～255)(単位:秒)へのポインタ

・処理概要　現在のウォッチドッグタイマの初期カウンタ値を取得する。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Counter;
ret = m_Ioc.GetWdtCounter( &Counter );
```

・例2
```
BOOL  ret;
int   Counter;
ret = ::GetWdtCounter( &Counter );
```

■ SetWdtMask

・呼び出し形式　BOOL SetWdtMask (int Selector, int Mask)

・戻り値
TRUE:正常
FALSE:エラー

・ 引数

(I)	int Selector	設定項目	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
(I)	int Mask	マスク情報	
		MASK_OFF	マスク解除
		MASK_ON	マスク設定

・処理概要　ウォッチドッグタイマタイムアウト時に出力する警告のマスクを設定する。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
// LAMP出力をマスクする
ret = m_Ioc.SetWdtMask( WARNING_LAMP, MASK_ON );
// ALARM出力のマスクを解除する
ret = m_Ioc.SetWdtMask( WARNING_ALARM, MASK_OFF );
```

・例2
```
BOOL  ret;
// LAMP出力をマスクする
ret = ::SetWdtMask( WARNING_LAMP, MASK_ON );
// ALARM出力のマスクを解除する
ret = ::SetWdtMask( WARNING_ALARM, MASK_OFF );
```

■ GetWdtMask

・呼び出し形式　BOOL GetWdtMask(int Selector, int *pMask)

・ 戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I/0) int *pMask	マスク情報へのポインタ	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　ウォッチドッグタイマタイムアウト時の警告出力マスク情報を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Mask;
// LAMPのマスク情報取得
ret = m_Ioc.GetWdtMask( WARNING_LAMP, &Mask );
// ALARMのマスク情報取得
ret = m_Ioc.GetWdtMask( WARNING_ALARM, &Mask );
```

・例2

```
BOOL ret;
int Mask;
// LAMPのマスク情報取得
ret = ::GetWdtMask( WARNING_LAMP, &Mask );
// ALARMのマスク情報取得
ret = ::GetWdtMask( WARNING_ALARM, &Mask );
```

■ StartWdt

・呼び出し形式　BOOL StartWdt (void)

・戻り値　TRUE:正常
FALSE:エラー

・引数　なし

・処理概要　ウォッチドッグタイマのカウントダウンを開始する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
ret = m_Ioc.StartWdt();
```

・例2

```
BOOL ret;
ret = ::StartWdt();
```

■ StopWdt

・呼び出し形式　　BOOL StopWdt (void)
・戻り値　　TRUE:正常
　　FALSE:エラー
・引数　　なし
・処理概要　　ウォッチドッグタイマのカウントダウンを停止する。
・例1　　CPL_Ioctl　m_Ioc;
　　BOOL ret;
　　ret = m_Ioc.StopWdt();
・例2　　BOOL　ret;
　　ret = ::StopWdt();

■ RestartWdt

・呼び出し形式　　BOOL RestartWdt (void)
・戻り値　　TRUE:正常
　　FALSE:エラー
・引数　　なし
・処理概要　　ウォッチドッグタイマのカウント値を初期値に戻し、再カウントダウンを始める。
・例1　　CPL_Ioctl　m_Ioc;
　　BOOL ret;
　　m_Ioc.RestartWdt();
・例2　　BOOL　ret;
　　ret = ::RestartWdt();

・ RestartWdtはStartWdtでカウントダウンを開始した後のみ使用できます。タイムアウトした後にRestartWdtを使用する場合はClearWdtでタイムアウト状態を解除し、再度StartWdtでカウントダウンを開始してください。

■ RunningWdt

・呼び出し形式　BOOL RunningWdt (int *pRunFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pRunFlag　ウォッチドッグタイマの動作状態へのポインタ
WATCHDOG_STOP　停止中
WATCHDOG_COUNTDOWN　カウントダウン中
・処理概要　ウォッチドッグタイマの動作状態を取得する。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int RunFlag;
ret = m_Ioc.RunningWdt( &RunFlag );
```

・例2

```
BOOL ret;
int RunFlag;
ret = ::RunningWdt( &RunFlag );
```

■ SetWarningOut

・呼び出し形式　BOOL SetWarningOut (int Selector, int WarnOut)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I)　int Selector　設定項目
WARNING_ALARM　ALARM
WARNING_LAMP　LAMP
(I)　int WarnOut　出力状態
OUTPUT_OFF　出力OFF
OUTPUT_ON　出力ON
・処理概要　設定項目(LAMP、ALARM)の警告情報を設定する。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
// LAMPの出力状態をONに設定
ret = m_Ioc.SetWarningOut( WARNING_LAMP, OUTPUT_ON );
// ALARMの出力状態をOFFに設定
ret = m_Ioc.SetWarningOut( WARNING_ALARM, OUTPUT_OFF );
```

・例2

```
BOOL ret;
// LAMPの出力状態をONに設定
ret = ::SetWarningOut( WARNING_LAMP, OUTPUT_ON );
// ALARMの出力状態をOFFに設定
ret = ::SetWarningOut( WARNING_ALARM, OUTPUT_OFF );
```

■ GetWarningOut

・呼び出し形式　BOOL GetWarningOut (int Selector, int *pWarnOut)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	設定項目	
	WARNING_ALARM	ALARM
	WARNING_LAMP	LAMP
(I/O) int *pWarnOut	出力状態へのポインタ	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

・処理概要　現在の設定項目(LAMP,ALARM)の警告状態を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int WarnOut;
// LAMPの出力状態取得
ret = m_Ioc.GetWarningOut( WARNING_LAMP, &WarnOut );
// ALARMの出力状態取得
ret = m_Ioc.GetWarningOut( WARNING_ALARM, &WarnOut );
```

・例2

```
BOOL ret;
int WarnOut;
// LAMPの出力状態取得
ret = ::GetWarningOut( WARNING_LAMP, &WarnOut );
// ALARMの出力状態取得
ret = ::GetWarningOut( WARNING_ALARM, &WarnOut );
```

■ GetUniversalIn

・呼び出し形式　BOOL GetUniversalIn (int Selector, int *pUniIn)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
(I/O) int *pUniIn	入力状態へのポインタ	
	INPUT_OFF	入力なし
	INPUT_ON	入力あり

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int UniIn;
// Universal Input 0の入力状態取得
ret = m_Ioc.GetUniversalIn( PORT_UNI0, &UniIn );
// Universal Input 1の入力状態取得
ret = m_Ioc.GetUniversalIn( PORT_UNI1, &UniIn );
```

・例2

```
BOOL  ret;
int UniIn;
// Universal Input 0の入力状態取得
ret = ::GetUniversalIn( PORT_UNI0, &UniIn );
// Universal Input 1の入力状態取得
ret = ::GetUniversalIn( PORT_UNI1, &UniIn );
```

■ ClearUniversalIn

・呼び出し形式　BOOL ClearUniversalIn (int Selector)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I) int Selector　対象ポート
PORT_UNI0　Universal Input 0
PORT_UNI1　Universal Input 1

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態をキャンセルする。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
// Universal Input 0の出力をキャンセルする
ret = m_Ioc.ClearUniversalIn( PORT_UNI0 );
// Universal Input 1の出力をキャンセルする
ret = m_Ioc.ClearUniversalIn( PORT_UNI1 );
```

・例2
```
BOOL  ret;
// Universal Input 0の出力をキャンセルする
ret = ::ClearUniversalIn( PORT_UNI0 );
// Universal Input 1の出力をキャンセルする
ret = ::ClearUniversalIn( PORT_UNI1 );
```

■ SetUniversalInMask

・呼び出し形式　BOOL SetUniversalInMask (in Selector, int Mask)

・戻り値　TRUE:正常
FALSE:エラー

・引数　(I)　in Selector　対象ポート
PORT_UNI0　Universal Input 0
PORT_UNI1　Universal Input 1
(I)　int Mask　マスク情報
MASK_OFF　マスク解除
MASK_OFF　マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を設定する。

・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
// Universal Input 0をマスク解除
ret = m_Ioc.SetUniversalInMask( PORT_UNI0, MASK_OFF );
// Universal Input 1をマスク
ret = m_Ioc.SetUniversalInMask( PORT_UNI1, MASK_ON );
```

・例2
```
BOOL  ret;
// Universal Input 0をマスク解除
ret = ::SetUniversalInMask( PORT_UNI0, MASK_OFF );
// Universal Input 1をマスク
ret = ::SetUniversalInMask( PORT_UNI1, MASK_ON );
```

■ GetUniversalInMask

・呼び出し形式　BOOL GetUniversalInMask(int Selector, int *pMask)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Selector	対象ポート	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
(I/O) int *pMask	マスク情報へのポインタ	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を取得する。

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Mask;
// Universal Input 0マスク情報取得
ret = m_Ioc.GetUniversalInMask( PORT_UNI0, &Mask );
// Universal Input 1マスク情報取得
ret = m_Ioc.GetUniversalInMask( PORT_UNI1, &Mask );
```

・例2

```
BOOL ret;
int Mask;
// Universal Input 0マスク情報取得
ret = ::GetUniversalInMask( PORT_UNI0, &Mask );
// Universal Input 1マスク情報取得
ret = ::GetUniversalInMask( PORT_UNI1, &Mask );
```

■ SetResetMask

・呼び出し形式　BOOL SetResetMask (int Mask)

・戻り値　TRUE:正常
FALSE:エラー

・引数

(I) int Mask	マスク情報	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　リセットマスクを設定する

・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
// リセットマスク解除
ret = m_Ioc.SetResetMask( MASK_OFF );
```

・例2

```
BOOL  ret;
// リセットマスク解除
ret = ::SetResetMask( MASK_OFF );
```

■ GetResetMask

・呼び出し形式　BOOL GetResetMask(int *pMask)
・戻り値　TRUE:正常
FALSE:エラー
・ 引数　(I/O) int *pMask　マスク情報へのポインタ
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　現在のリセットマスク情報を取得する。
・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Mask;
ret = m_Ioc.GetResetMask( &Mask );
```
・例2
```
BOOL ret;
int Mask;
ret = ::GetResetMask( &Mask );
```

■ GetLightblowErr

・呼び出し形式　BOOL GetLightblowErr (int *pLightErr)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O)int *pLightErr　エラー出力情報へのポインタ
BACKLIGHT_OK　バックライト正常
BACKLIGHT_ERR　バックライト管切れ
・処理概要　現在のLCDバックライト管切れエラー出力を取得する。
・例1
```
CPL_Ioctl  m_Ioc;
BOOL ret;
int LightErr;
// バックライト管切れ状態取得
ret = m_Ioc.GetLightblowErr( &LightErr );
```
・例2
```
BOOL  ret;
int LightErr;
// バックライト管切れ状態取得
ret = ::GetLightblowErr( &LightErr );
```

■ SetWdtResetMask

・呼び出し形式　BOOL SetWdtResetMask (int Mask)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I) int Mask　マスク状態
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　ウォッチドッグタイマによるリセットのマスク状態を設定する。
・例1
```
BOOL ret;
ret = ::SetWdtResetMask(MASK_ON);
```
・例2
```
CPL_SmiIoctl    m_SmiIoctl;
BOOL ret;
ret = m_SmiIoctl.SetWdtResetMask(MASK_ON);
```

■ GetWdtResetMask

・呼び出し形式　BOOL GetWdtResetMask(int *pMask)
・戻り値　TRUE;正常
FALSE;エラー
・引数　(I/O) int *pMask　マスク状態へのポインタ
MASK_OFF マスク解除
MASK_ON マスク設定
・処理概要　ウォッチドッグタイマーによるリセットマスクの状態を取得する。
・例1
```
BOOL ret;
int Mask;
ret = ::GetWdtResetMask(&Mask);
```
・例2
```
CPL_Ioctl  m_Ioctl;
BOOL ret;
int Mask;
ret = m_Ioctl.GetWdtResetMask(&Mask);
```

■ SetWarningDOUT

・呼び出し形式　BOOL SetWarningDOUT (int WarningOut)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I) int WarningOut　出力状態
OUTPUT_OFF　出力OFF
OUTPUT_ON　出力ON
・処理概要　現在の設定項目(DOUT)の警告状態を設定する。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
// DOUTの出力状態をOFFに設定
ret = m_Ioc.SetWarningDOUT( OUTPUT_OFF );
```

・例2

```
BOOL ret;
// DOUTの出力状態をOFFに設定
ret = ::SetWarningDOUT( OUTPUT_OFF );
```

■ GetWarningDOUT

・呼び出し形式　BOOL GetWarningDOUT(int* pWarningOut)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pWarningOut　出力状態へのポインタ
OUTPUT_OFF　出力OFF
OUTPUT_ON　出力ON
・処理概要　現在の設定項目(DOUT)の警告状態を取得する。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int WarningOut
// DOUTの出力状態を取得
ret = m_Ioc.GetWarningDOUT( &WarningOut );
```

・例2

```
BOOL ret;
int WarningOut;
// DOUTの出力状態を取得
ret = ::GetWarningDOUT( &WarningOut );
```

■ GetWdtTimeout

・呼び出し形式　BOOL GetWdtTimeout(int *pTimebuf)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pTimebuf　ウォッチドッグタイムアウト状態へのポインタ
TIMEOUT_OK　タイムアウトしていない
TIMEOUT_ERROR　タイムアウトしている
・処理概要　ウォッチドッグのタイムアウト状態を取得する
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
int Timebuf;
// ウォッチドッグのタイムアウト状態取得
ret = m_Ioc.GetWdtTimeout( &Timebuf );
```

・例2

```
BOOL ret;
int Timebuf;
// ウォッチドッグのタイムアウト状態取得
ret = ::GetWdtTimeout( &Timebuf );
```

■ ClearWdtTimeout

・呼び出し形式　BOOL ClearWdtTimeout (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　ウォッチドッグのタイムアウト状態をクリアする。
・例1

```
CPL_Ioctl  m_Ioc;
BOOL ret;
// ウォッチドッグのタイムアウト状態クリア
ret = m_Ioc.ClearWdtTimeout();
```

・例2

```
BOOL  ret;
// ウォッチドッグのタイムアウト状態クリア
ret = ::ClearWdtTimeout();
```

■ GetSmiDrvHandle

・呼び出し形式　int GetSmiDrvHandle (void)
・戻り値　0:正常
1:エラー
・引数　なし
・処理概要　ソフトミラーデバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例1

```
CPL_SmiIoctl    m_SmiIoc;
BOOL ret;
// ソフトミラードライバハンドルの取得
ret = m_SmiIoc.GetSmiDrvHandle();
```

・例2

```
BOOL ret;
// ソフトミラードライバハンドルの取得
ret = ::GetSmiDrvHandle();
```

・ ソフトミラーデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseSmiDrvHandle

・呼び出し形式　BOOL CloseSmiDrvHandle (void)
・戻り値　TRUE:正常
FALSE:エラー
・引数　なし
・処理概要　GetSmiDrvHandle関数で取得したハンドルを破棄する。
・例1

```
CPL_SmiIoctl    m_SmiIoc;
BOOL ret;
// ハンドル破棄
ret = m_SmiIoc.CloseSmiDrvHandle();
```

・例2

```
BOOL ret;
// ハンドル破棄
ret = ::CloseSmiDrvHandle();
```

■ GetSmiAryStatus

・呼び出し形式　BOOL GetSmiAryStatus (int *pStatus)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pStatus　ミラーディスクへのポインタ

ARYSTAT_GOOD	正常
ARYSTAT_UNCONFIG	未構築状態
ARYSTAT_REBUILD	再構築中
ARYSTAT_REDUCE	縮退中
ARYSTAT_DEAD	ミラー状態破壊

・処理概要　ソフトミラーの状態を取得する。
・例1

```
CPL_SmiIoctl     m_SmiIoc;
BOOL  ret;
int Status;
// ソフトミラーの状態取得
ret = m_SmiIoc.GetSmiAryStatus( &Status );
```

・例2

```
BOOL  ret;
int   Status;
// ソフトミラーの状態取得
ret = ::GetSmiAryStatus( &Status );
```

■ GetSmiDevStatus

・呼び出し形式　BOOL GetSmiDevStatus (int Id, int *pType, int *pStatus)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I) int Id　デバイスID
0：Master HDD
1：Slave HDD
(I/O) int *pType　デバイスタイプ

ATADEVICE	HDD(ATAデバイス)
ATAPIDEVICE	CD-ROM
NODEVICE	未接続

(I/O) int *pStatus　Disk状態へのステータス

DEVSTAT_GOOD	正常
DEVSTAT_NOTEXIST	未接続
DEVSTAT_BROKEN	故障

・処理概要　ソフトミラーのデバイス状態を取得する。
・例1

```
CPL_SmiIoctl     m_SmiIoc;
BOOL  ret;
int Id, Type, Status
// デバイス状態の取得
Id = 0;
ret = m_SmiIoc.GetSmiDevStatus( Id, &Type, &Status );
```

・例2

```
BOOL  ret;
int Id, Type, Status;
// デバイス状態の取得
Id = 0;
ret = ::GetSmiDevStatus( Id, &Type, &Status );
```

6.3.2 PL_Ras.dll関数

■ PlDevWordWrite

・呼び出し形式　long PlDevWordWrite(long Addr, long wData)
・戻り値　0:正常
0以外:エラー
・引数　(I) long Addr　書き込むメモリのワードアドレス(0～255)
(I) long wData　書き込むデータ(0～65535)
・処理概要　共有メモリへの書き込みを行います。
・例
```
// アドレス255へデータ255を書き込む
long ret;
ret = PlDevWordWrite(255, 255);
```

■ PlDevWordRead

・呼び出し形式　long PlDevWordRead(long Addr, long *wData)
・戻り値　0:正常
0以外:エラー
・引数　(I) long Addr　読み出すメモリのワードアドレス
(I/O) long *wData　読み出すデータへのポインタ(0～65535)
・処理概要　共有メモリへの読み出しを行います。
・例
```
// アドレス255のデータ読み出し
long ret;
long wData;
ret = PlDevWordRead(255, &wData);
```

6.3.3 PL_BLIoc.dll関数

■ GetBLDrvHandle

・呼び出し形式　int GetBLDrvHandle(void)またはint GetBLDrvHandle(HANDLE *pHndl)
・戻り値　0:正常
　0以外:エラー
・引数　(I/O) HANDL *pHndl　デバイスドライバハンドルへのポインタ
・処理概要　デバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例1

```
CPL_BLIoctl      m_BLIoc;
m_BLIoc. GetBLDrvHandle();
```

・例2

```
int         ret;
HANDLE      hndl;
ret = ::GetBLDrvHandle( &hndl );
```

・バックライトコントロールデバイスドライバが動作していない場合はエラー（戻り値:0以外)になります。

■ CloseBLDrvHandle

・呼び出し形式　BOOL CloseBLDrvHandle()
・戻り値　TRUE：正常
　FALSE：エラー
・引数　なし
・処理概要　GetBLDrvHandleで取得したハンドルを破棄する。
・例1

```
BOOL ret;
ret = ::CloseBLDrvHandle();
```

・例2

```
CPL_BLIoctl m_BLIoctl;
BOOL ret;
ret = m_BLIoctl.CloseBLDrvHandle();
```

■ GetBLDrvVersion

・呼び出し形式　BOOL GetBLDrvVersion(int *pMajor, int *pMinor)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ
・処理概要　ドライババージョン情報を取得する。
・例1

```
CPL_BLIoctl      m_BLIoc;
BOOL  ret;
int Major, Minor;
ret = m_BLIoc.GetBLDrvVersion( &Major, &Minor );
```

・例2

```
BOOL  ret;
int   Major, Minor;
ret = ::GetBLDrvVersion( &Major, &Minor );
```

・ドライババージョンが1.00の場合は
Major:1　(10進数)
Minor:00　(10進数)
となります。

■ GetBLDrvVersionEx

・呼び出し形式　BOOL GetBLDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pProduct　機種情報へのポインタ
(I/O) int *pMajor　バージョン情報(Major,0～99)へのポインタ
(I/O) int *pMinor　バージョン情報(Minor,0～99)へのポインタ
・処理概要　ドライババージョン、機種情報を取得する。
・例1

```
CPL_BLIoctl      m_BLIoctl;
BOOL  ret;
int Product, Major, Minor;
ret = m_BLIoctl.GetBLDrvVersionEx( &Product, &Major, &Minor );
```

・例2

```
BOOL  ret;
int   Major, Minor;
ret = ::GetBLDrvVersionEx( &Product, &Major, &Minor );
```

・PL-5910シリーズでドライババージョンが1.10の場合は
Product:4　(10進数)
Major:1　(10進数)
Minor:10　(10進数)
となります。

■ SetBLControl

・呼び出し形式　BOOL SetBLControl(int BLFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I)　int BLFlag　バックライト設定パラメータ
BACKLIGHT_OFF　バックライトOFF
BACKLIGHT_ON　バックライトON
・処理概要　バックライトのON/OFFを設定する。
・例1
```
CPL_BLIoc m_BLIoc
BOOL      ret;
//バックライトコントロールON設定
ret = m_BLIoc.SetBLControl( BACKLIGHT_ON );
```
・例2
```
BOOL ret;
//バックライトコントロールON設定
ret = ::SetBLControl( BACKLIGHT_ON );
```

■ GetBLControl

・呼び出し形式　BOOL GetBLControl(int *pBLFlag)
・戻り値　TRUE:正常
FALSE:エラー
・引数　(I/O) int *pBLFlag　バックライト状態へのポインタ
BACKLIGHT_OFF　バックライトOFF
BACKLIGHT_ON　バックライトON
・処理概要　バックライトコントロール状態を取得する。
・例1
```
CPL_BLIoc m_BLIoc;
BOOL  ret;
int   BLFlag;
//バックライトコントロール状態取得
ret = m_BLIoc. GetBLControl( &BLFlag );
```
・例2
```
BOOL  ret;
int   BLFlag;
//バックライトコントロール状態取得
ret = ::GetBLControl( &BLFlag );
```

7 Visual Basic用関数仕様

7.1 Visual Basic用関数一覧

7.1.1 PL_Ioc.dll関数一覧

関数名	説明
InitIoctl	CPL_Ioctlオブジェクト作成
EndIoctl	CPL_Ioctlオブジェクト破棄
GetDrvHandle	ドライバハンドル取得
CloseDrvHandle	GetDrvHandle取得ハンドル破棄
GetDrvVersion	ドライババージョン取得
GetDrvVersionEx	ハードウェアタイプ、ドライババージョン取得
GetMonitorSetup	モニタ許可／禁止設定取得
GetVoltParam	電圧監視用パラメータ取得
GetCurrentVolt	現在電圧値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
SetWdtCounter	ウォッチドッグタイマカウンタ値設定
GetWdtCounter	ウォッチドッグタイマカウンタ取得
SetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク設定
GetWdtMask	ウォッチドッグタイマタイムアウト時の警告マスク取得
StartWdt	ウォッチドッグタイマ開始
StopWdt	ウォッチドッグタイマ停止
RestartWdt	ウォッチドッグタイマ再開
RunningWdt	ウォッチドッグタイマ動作状況取得
SetWarningOut	警告出力設定
GetWarningOut	警告出力取得
GetUniversalIn	汎用入力取得
ClearUniversalIn	汎用入力ラッチ状態解除
SetUniversalInMask	汎用入力マスク設定
GetUniversalInMask	汎用入力マスク取得
SetResetMask	リセットマスク設定
GetResetMask	リセットマスク取得
GetLightblowErr	バックライト切れ状態取得
GetWdtTimeout	ウォッチドッグタイマのタイムアウト状態取得
ClearWdtTimeout	ウォッチドッグタイマのタイムアウト状態クリア
SetWarningDOUT	警告出力DOUT設定
GetWarningDOUT	警告出力DOUT取得
GetSmiDrvHandle	SoftMirror　ドライバハンドル取得
CloseSmiDrvHandle	SoftMirror　ドライバハンドル破棄
GetSmiAryStatus	SoftMirror　Array Status　取得
GetSmiDevStatus	SoftMirror　Device Status　取得

7.1.2 PL_Ras.dll関数一覧

関数名	説明
PlDevWordWrite	共有メモリへの書き込み
PlDevWordRead	共有メモリからの読み出し

7.1.3 PL_BLIoc.dll関数一覧

関数名	説明
InitBLIoctl	CPL_BLIoctlオブジェクト作成
EndBLIoctl	CPL_BLIoctlオブジェクト破棄
GetBLDrvHandle	ドライバハンドル取得
CloseBLDrvHandle	ドライバハンドル取得
GetBLDrvVersion	ドライババージョン取得
GetBLDrvVersionEx	ハードウェアバージョン、ドライババージョン取得
SetBLControl	バックライト状態設定
GetBLControl	バックライト状態取得

7.2 Visual Basicでのプログラム開発における注意事項

API-DLLを使用するためには､使用前にドライバオブジェクトの作成とデバイスハンドルの取得､使用後にデバイスハンドルの破棄とドライバオブジェクトの破棄を行う必要があります。以下に示す例を参考にプログラム開発を行ってください。

・ PlDevWordWriteとPlDevWordReadのみを使用する場合は、ドライバオブジェクトおよびデバイスハンドルの作成/破棄処理は必要ありません。

■サンプルプログラム例

```
'API-DLL使用例
'ドライバオブジェクトの作成
Call InitIoctl
'デバイスハンドルの取得
Dim ret As Long
Dim Hndl As Long
ret=GetDrvHandle(Hndl)
   ・
   ・
'DOUTへの出力
Dim ret As Long
ret=SetWarningDOUT(OUTPUT_ON)
   ・
   ・
'アプリケーション終了処理
'デバイスハンドルの破棄
Dim ret As Long
ret=CloseDrvHandle()
'ドライバオブジェクトの破棄
Call EndIoctl
```

7.3 Visual Basic用関数仕様詳細

7.3.1 PL_Ioc.dll関数

■ InitIoctl

・呼び出し形式　Declare Sub InitIoctl Lib "PL_Ioc.dll" ()
・戻り値　なし
・引数　なし
・処理概要　CPL_Ioctlオブジェクトを作成する。作成されたオブジェクトはEndIoctl関数が呼ばれるまで破棄されない。
・例　InitIoctl()

■ EndIoctl

・呼び出し形式　Declare Sub EndIoctl Lib "PL_Ioc.dll" ()
・戻り値　なし
・引数　なし
・処理概要　InitIoctl関数で作成したオブジェクトを破棄する。
・例　EndIoctl()

■ GetDrvHandle

・呼び出し形式　Declare Function GetDrvHandle Lib "PL_Ioc.dll" (ByRef hndl As Long) As Long
・戻り値　0:正常
1：エラー
・引数　hndl As Long　デバイスドライバハンドル(参照渡し)
・処理概要　デバイスドライバとのやり取りを行なうためのデバイスドライバハンドルを取得する。
・例

```
Dim ret As Long
Dim hndl As Long
ret = GetDrvHandle(hndl)
```

MEMO
・ システムモニタ/RASデバイスドライバが動作していない場合はエラーになります。

■ CloseDrvHandle

・呼び出し形式　Declare Function CloseDrvHandle Lib "PL_Ioc.dll"() As Long
・戻り値　0以外:正常
0:エラー
・引数　なし
・処理概要　GetDrvHandle関数で取得したハンドルを破棄する。
・例

```
Dim ret As Long
'ハンドル破棄
ret = CloseDrvHandle()
```

■ GetDrvVersion

・呼び出し形式　Declare Function GetDrvVersion Lib "PL_Ioc.dll"
(By Ref Major As Long, By Ref Minor As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Major As Long　バージョン情報(Major,0～99)(参照渡し)
Minor As Long　バージョン情報(Minor,0～99)(参照渡し)

・処理概要　ドライババージョン情報を取得する。

・例
```
Dim ret As Long
Dim Major As Long
Dim Minor As Long

ret = GetDrvVersion(Major, Minor)
```

・ ドライババージョンが1.00の場合は、
Major:1　(10進数)
Minor：00　(10進数)
となります。

■ GetDrvVersionEx

・呼び出し形式　Declare Function GetDrvVersionEx Lib "PL_Ioc.dll"
(By Ref Product As Long, By Ref Major As Long,
By Ref Minor As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Product As Long　機種情報(参照渡し)
Major As Long　バージョン情報(Major,0～99)(参照渡し)
Minor As Long　バージョン情報(Minor,0～99)(参照渡し)

・処理概要　ドライババージョン情報を取得する。

・例
```
Dim ret As Long
Dim Product As Long
Dim Major As Long
Dim Minor As Long

ret = GetDrvVersionEx(Product, Major, Minor)
```

・ PL-5910シリーズでドライババージョンが1.00の場合は、
Product:4　(10進数)
Major:1　(10進数)
Minor：00　(10進数)
となります。

■ GetMonitorSetup

・呼び出し形式　Declare Function GetMonitorSetup Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Setup As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
Setup As Long	取得データ(参照渡し)	
	1:Enable	

・処理概要　現在のモニタ許可状態を取得する。

・例

```
Dim ret As Long
Dim Setup As Long
'CPUコア電圧セットアップ状態取得
ret = GetMonitorSetup(MONITOR_VOLT_VCOREA, Setup )
```

■ GetVoltParam

・呼び出し形式　Declare Function GetVoltParam Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef ULimit As Long, ByRef LLimit As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
Ulimit As Long	電圧上限値(単位:mV) (参照渡し)	
LLimit As Long	電圧下限値(単位:mV) (参照渡し)	

・処理概要　電圧監視用パラメータを取得する。

・例

```
Dim ret As Long
Dim ULimit As Long
Dim LLimit As Long
'CPUコア電圧上限下限値取得
ret = GetVoltParam( MONITOR_VOLT_VCOREA, ULimit, LLimit )
```

・ 関数から取得されたデータはmV(ミリボルト)単位になっています。V（ボルト）単位で使用する時は下記のような変換を行ってください。
ボルト単位データ = ミリボルト単位データ/1000

■ GetCurrentVolt

・呼び出し形式　Declare Function GetCurrentVolt Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Data As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	MONITOR_VOLT_VCOREA	CPUコア電圧
	MONITOR_VOLT_VCOREB	CPUコア電圧2
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M50	-5.0V電圧
	MONITOR_VOLT_M12	-12V電圧
Data As Long	電圧値(単位:mV) (参照渡し)	

・処理概要　現在の電圧値を取得する。

・例

```
Dim ret As Long
Dim Data As Long
'CPUコア電圧値取得
ret = GetCurrentVolt(MONITOR_VOLT_VCOREA, Data)
```

・ 関数から取得されたデータはmV(ミリボルト)単位になっています。V（ボルト）単位で使用する時は下記のような変換を行ってください。
ボルト単位データ = ミリボルト単位データ/1000

■ GetTempParam

・呼び出し形式　Declare Function GetTempParam Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef ULimit As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
Ulimit As Long	温度上限値(単位:℃) (参照渡し)	

・処理概要　温度監視用のパラメータを取得する。

・例

```
Dim ret As Long
Dim ULimit As Long
'SYSTEM温度上限値取得
ret = GetTempParam( MONITOR_TEMP_SYSTEM, ULimit )
```

■ GetCurrentTemp

・呼び出し形式　Declare Function GetCurrentTemp Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Data As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
Data As Long	温度値(単位:℃) (参照渡し)	

・処理概要　現在の温度値を取得する。

・例
```
Dim ret As Long
Dim Data As Long
'SYSTEM温度値取得
ret = GetCurrentTemp( MONITOR_TEMP_SYSTEM, Data )
```

■ GetEvent

・呼び出し形式　Declare Function GetEvent Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Event As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	取得パラメータ(値渡し)	
	EVENT_VOLT_VCOREA	CPUコア電圧
	EVENT_VOLT_VCOREB	CPUコア電圧2
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5.0V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M50	-5.0V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_UNI_IN0	Universal Input 0
	EVENT_UNI_IN1	Universal Input 1
	EVENT_WDT_TIMEOUT	Watchdog Timeout
	EVENT_LIGHT_BLOW	バックライト管切れ
Event As Long	エラーイベント情報(参照渡し)	
	ERROR_EVENT_OFF	エラーイベントなし
	ERROR_EVENT_ON	エラーイベントあり

・処理概要　マシンの電圧、温度の異常、また、Universal Input動作の情報(イベント)、Watchdog Timeout情報をチェックする。

・例
```
Dim ret As Long
Dim Event As Long
'CPUコア電圧のエラーイベント情報取得
ret = GetEvent( EVENT_VOLT_VCOREA, Event )
```

■ ClearEvent

・呼び出し形式　Declare Function ClearEvent Lib "PL_Ioc.dll"
(ByVal Selector As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Selector As Long　エラーイベントキャンセル対象パラメータ(値渡し)

EVENT_VOLT_VCOREA	CPUコア電圧
EVENT_VOLT_VCOREB	CPUコア電圧2
EVENT_VOLT_P33	+3.3V電圧
EVENT_VOLT_P50	+5.0V電圧
EVENT_VOLT_P12	+12V電圧
EVENT_VOLT_M50	-5.0V電圧
EVENT_VOLT_M12	-12V電圧
EVENT_TEMP_SYSTEM	SYSTEM温度
EVENT_TEMP_CPU	CPU温度
EVENT_UNI_IN0	Universal Input0
EVENT_UNI_IN1	Universal Input1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	バックライト管切れ

・処理概要　エラーイベントをキャンセルする。

・例

```
Dim ret As Long
'CPUコア電圧エラーイベントキャンセル
ret = ClearEvent( EVENT_VOLT_VCOREA )
```

■ SetWdtCounter

・呼び出し形式　Declare Function SetWdtCounter Lib "Pl_Ioc.dll"
(ByVal Counter As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Counter As Long　ウォッチドッグタイマの初期カウンタ値
5～255秒(値渡し)

・処理概要　ウォッチドッグタイマの初期カウンタ値を設定する。

・例
```
Dim ret As Long
Dim Counter As Long
' ウォッチドッグタイマの初期カウンタ値を10秒に設定
Counter = 10
ret = SetWdtCounter(Counter)
```

■ GetWdtCounter

・呼び出し形式　Declare Function GetWdtCounter Lib "PL_Ioc.dll"
(ByRef Counter As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Counter As Long　ウォッチドッグタイマの初期カウンター値
(参照渡し)(単位:秒)

・処理概要　現在のウォッチドッグタイマの初期カウンタ値を取得する。

・例
```
Dim ret As Long
Dim Counter As Long
ret = GetWdtCounter(Counter)
```

■ SetWdtMask

・呼び出し形式　Declare Function SetWdtMask Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByVal Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Selector As Long　設定項目(値渡し)

WARNING_LAMP	LAMP
WARNING_ALARM	ALARM

Mask As Long　マスク情報(値渡し)

MASK_OFF	マスク解除
MASK_ON	マスク

・処理概要　ウォッチドッグタイマタイムアウト時に出力する警告のマスクを設定する。

・例
```
Dim ret As Long
'LAMP出力をマスクする
ret = SetWdtMask( WARNING_LAMP, MASK_ON )
'ALARM出力のマスクを解除する。
ret = SetWdtMask( WARNING_ALARM, MASK_OFF )
```

■ GetWdtMask

・呼び出し形式　Declare Function GetWdtMask Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	設定項目(値渡し)	
	WARNING_LAMP	LAMP
	WARNING_ALARM	ALARM
Mask As Long	マスク情報(参照渡し)	
	MASK_OFF	マスク解除
	MASK_ON	マスク

・処理概要　ウォッチドッグタイマタイムアウト時の警告出力マスク情報を取得する。

・例
```
Dim ret As Long
Dim Mask As Long
'LAMPのマスク情報取得
ret = GetWdtMask( WARNING_LAMP, Mask )
'ALARMのマスク情報取得
ret = GetWdtMask( WARNING_ALARM, Mask )
```

■ StartWdt

・呼び出し形式　Declare Function StartWdt Lib "PL_Ioc.dll" () As Long

・戻り値　0以外:正常
0:エラー

・引数　なし

・処理概要　ウォッチドッグタイマのカウントダウンを開始する。

・例
```
Dim ret As Long
ret = StartWdt()
```

■ StopWdt

・呼び出し形式　Declare Function StopWdt Lib "PL_Ioc.dll" () As Long

・戻り値　0以外:正常
0:エラー

・引数　なし

・処理概要　ウォッチドッグタイマのカウントダウンを停止する。

・例
```
Dim ret As Long
ret = StopWdt()
```

■ RestartWdt

・呼び出し形式　Declare Function RestartWdt Lib "PL_Ioc.dll" () As Long
・戻り値　0以外:正常
0:エラー
・引数　なし
・処理概要　ウォッチドッグタイマのカウント値を初期値に戻し､再カウントダウンを始める。
・例
```
Dim ret As Long
ret = RestartWdt()
```

・ RestartWdtはStartWdtでカウントダウンを開始した後のみ使用できます。タイムアウトした後にRestartWdtを使用する場合はClearWdtでタイムアウト状態を解除し、再度StartWdtでカウントダウンを開始してください。

■ RunningWdt

・呼び出し形式
```
Declare Function RunningWdt Lib "PL_Ioc.dll"
(ByRef RunFlag As Long) As Long
```
・戻り値　0以外:正常
0:エラー
・引数　RunFlag As Long　ウォッチドッグタイマの動作状態(参照渡し)

WATCHDOG_STOP	停止中
WATCHDOG_COUNTDOWN	カウントダウン中

・処理概要　ウォッチドッグタイマの動作状態を取得する。
・例
```
Dim ret As Long
Dim RunFlag As Long
ret = RunningWdt( RunFlag )
```

■ SetWarningOut

・呼び出し形式　Declare Function SetWarningOut Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByVal WarnOut As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	設定項目(値渡し)	
	WARNING_LAMP	LAMP
	WARNING_ALARM	ALARM
WarnOut As Long	出力状態(値渡し)	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

・処理概要　設定項目(LAMP、ALARM)の警告情報を設定する。

・例

```
Dim ret As Long
'LAMPの出力状態をONに設定
ret = SetWarningOut( WARNING_LAMP, OUTPUT_ON )
'ALARMの出力状態をOFFに設定
ret = SetWarningOut( WARNING_ALARM, OUTPUT_OFF )
```

■ GetWarningOut

・呼び出し形式　Declare Function GetWarningOut Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef WarnOut As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	設定項目(値渡し)	
	WARNING_LAMP	LAMP
	WARNING_ALARM	ALARM
WarnOut As Long	出力状態(参照渡し)	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

・処理概要　現在の設定項目(LAMP,ALARM)の警告状態を取得する。

・例

```
Dim ret As Long
Dim WarnOut As Long
'LAMPの出力状態取得
ret = GetWarningOut( WARNING_LAMP, WarnOut )
'ALARMの出力状態取得
ret = GetWarningOut( WARNING_ALARM, WarnOut )
```

■ GetUniversalIn

・呼び出し形式　Declare Function GetUniversalIn Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef UniIn As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	対象ポート(値渡し)	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
UniIn As Long	入力状態(参照渡し)	
	INPUT_OFF	入力なし
	INPUT_ON	入力あり

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態を取得する。

・例

```
Dim ret As Long
Dim UniIn As Long
'Universal Input 0の入力状態取得
ret = GetUniversalIn( PORT_UNI0, UniIn )
'Universal Input 1の入力状態取得
ret = GetUniversalIn( PORT_UNI1, UniIn )
```

■ ClearUniversalIn

・呼び出し形式　Declare Function ClearUniversalIn Lib "PL_Ioc.dll"
(ByVal Selector As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	対象ポート(値渡し)	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1

・処理概要　対象ポート(Universal Input 0, Universal Input 1)の入力状態をキャンセルする。

・例

```
Dim ret As Long
'Universal Input 0の入力状態をキャンセルする
ret = ClearUniversalIn( PORT_UNI0 )
'Universal Input 1の入力状態をキャンセルする
ret = ClearUniversalIn( PORT_UNI1 )
```

■ SetUniversalInMask

・呼び出し形式　Declare Function SetUniversalInMask Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByVal Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	対象ポート(値渡し)	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
Mask As Long	マスク情報(値渡し)	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を設定する。

・例

```
Dim ret As Long
'Universal Input 0をマスク解除
ret = SetUniversalInMask( PORT_UNI0, MASK_OFF )
'Universal Input 1をマスク
ret = SetUniversalInMask( PORT_UNI1, MASK_ON )
```

■ GetUniversalInMask

・呼び出し形式　Declare Function GetUniversalInMask Lib "PL_Ioc.dll"
(ByVal Selector As Long, ByRef Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数

Selector As Long	対象ポート(値渡し)	
	PORT_UNI0	Universal Input 0
	PORT_UNI1	Universal Input 1
Mask As Long	マスク情報(参照渡し)	
	MASK_OFF	マスク解除
	MASK_ON	マスク設定

・処理概要　対象ポート(Universal Input 0, Universal Input 1)のマスク情報を取得する。

・例

```
Dim ret As Long
Dim Mask As Long
'Universal Input0マスク情報取得
ret = GetUniversalInMask( PORT_UNI0, Mask )
'Universal Input1マスク情報取得
ret = GetUniversalInMask( PORT_UNI1, Mask )
```

■ SetResetMask

・呼び出し形式　Declare Function SetResetMask Lib "PL_Ioc.dll"
(ByVal Mask As Long) As Long
・戻り値　0以外:正常
0:エラー
・引数　Mask As Long　マスク情報(値渡し)
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　リセットマスクを設定する
・例
```
Dim ret As Long
'リセットマスク解除
ret = SetResetMask( MASK_OFF )
```

■ GetResetMask

・呼び出し形式　Declare Function GetResetMask Lib "PL_Ioc.dll"
(ByRef Mask As Long) As Long
・戻り値　0以外:正常
0:エラー
・引数　Mask As Long　マスク情報(参照渡し)
MASK_OFF　マスク解除
MASK_ON　マスク設定
・処理概要　現在のリセットマスク情報を取得する。
・例
```
Dim ret As Long
Dim Mask As Long
ret = GetResetMask( Mask )
```

■ GetLightblowErr

・呼び戻し形式　Declare Function GetLightblowErr Lib "PL_Ioc.dll"
(ByRef LightblowErr As Long) As Long
・戻り値　0以外:正常
0:エラー
・引数　LightblowErr As Long　エラー情報(参照渡し)
BACKLIGHT_OK　正常
BACKLIGHT_ERR　エラー
・処理概要　現在のLCDバックライト管切れエラー出力を取得する。
・例
```
Dim ret As Long
Dim LightblowErr As Long
'バックライトエラー情報取得
ret = GetLightblowErr( LightblowErr )
```

■ SetWdtResetMask

・呼び出し形式　Declare Function SetWdtResetMask Lib "PL_Ioc.dll"
(ByVal Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Mask As Long　マスク状態(値渡し)
MASK_OFF　マスク解除
MASK_ON　マスク設定

・処理概要　ウォッチドッグタイマによるリセットマスク状態を取得する

・例
```
Dim ret As Long
ret = SetWdtResetMask( MASK_ON )
```

■ GetWdtResetMask

・呼び出し形式　Declare Function GetWdtResetMask Lib "PL_Ioc.dll"
(ByRef Mask As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Mask As Long　マスク状態(参照渡し)
MASK_OFF　マスク解除
MASK_ON　マスク設定

・処理概要　ウォッチドッグタイマによるリセットマスクの状態を取得する

・例
```
Dim ret As Long
Dim Mask As Long
ret = GetWdtResetMask( Mask )
```

■ SetWarningDOUT

・呼び出し形式　Declare Function SetWarningDOUT Lib "PL_Ioc.dll"
(ByVal WarningOut As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　WarningOut As Long　出力状態(値渡し)
OUTPUT_OFF　出力OFF
OUTPUT_ON　出力ON

・処理概要　現在の設定項目(DOUT)の警告状態を設定する。

・例
```
Dim ret As Long
'DOUTの出力状態をOFFに設定
ret = SetWarningDOUT( OUTPUT_OFF )
```

■ GetWarningDOUT

・呼び出し形式　Declare Function GetWarningDOUT Lib "PL_Ioc.dll"
(ByRef WarningOut As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　WarningOut As Long　出力状態(参照渡し)
OUTPUT_OFF　出力 OFF
OUTPUT_ON　出力 ON

・処理概要　現在の設定項目(DOUT)の警告状態を取得する。

・例
```
Dim ret As Long
Dim WarningOut As Long
ret = GetWarningDOUT( WarningOut )
```

■ GetWdtTimeout

・呼び出し形式　Declare Function GetWdtTimeout Lib "PL_Ioc.dll"
(ByRef Timebuf As Long) As Long

・戻り値　0以外:正常
0:エラー

・引数　Timebuf As Long　ウォッチドッグタイムアウト状態(参照渡し)

・処理概要　ウォッチドッグのタイムアウト状態を取得する

・例
```
Dim ret As Long
Dim Timebuf As Long
'ウォッチドッグのタイムアウト状態取得
ret = GetWdtTimeout( Timebuf )
```

■ ClearWdtTimeout

・呼び出し形式　Declare Function ClearWdtTimeout Lib "PL_Ioc.dll" () As Long

・戻り値　0以外:正常
0:エラー

・引数　なし

・処理概要　ウォッチドッグのタイムアウト状態をクリアする。

・例
```
 Dim ret As Long
'ウォッチドッグのタイムアウト状態クリア
ret = ClearWdtTimeout()
```

■ GetSmiDrvHandle

・呼び出し形式　Declare Function GetSmiDrvHandle Lib "PL_Ioc.dll" () As Long
・戻り値　0:正常
1:エラー
・引数　なし
・処理概要　ソフトミラーデバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例
```
Dim ret As Long
ret = GetSmiDrvHandle()
```

・ ソフトミラーデバイスドライバが動作していない場合はエラー(戻り値:1)になります。

■ CloseSmiDrvHandle

・呼び出し形式　Declare Function CloseSmiDrvHandle Lib "PL_Ioc.dll" () As Long
・戻り値　0以外:正常
0:エラー
・引数　なし
・処理概要　GetSmiDrvHandle関数で取得したハンドルを破棄する。
・例
```
Dim ret As Long
'ハンドルを破棄する
ret = CloseSmiDrvHandle()
```

■ GetSmiAryStatus

・呼び出し形式　Declare Function GetSmiAryStatus Lib "PL_Ioc.dll"
(ByRef Status As Long) As Long
・戻り値　0以外:正常
0:エラー
・引数　Status As Long　ソフトミラーステータス(参照渡し)

ARYSTAT_GOOD	正常
ARYSTAT_UNCONFIG	未構築状態
ARYSTAT_REBUILD	再構築中
ARYSTAT_REDUCE	縮退中
ARYSTAT_DEAD	ミラー状態破壊

・処理概要　ソフトミラーの状態を取得する。
・例
```
Dim ret As Long
Dim Status As Long
'ソフトミラーの状態を取得する
ret = GetSmiAryStatus( Status )
```

■ GetSmiDevStatus

・呼び出し形式　Declare Function GetSmiDevStatus Lib "PL_Ioc.dll"
(ByVal Id As Long, ByRef Type As Long, ByRef Status As Long)As Long

・戻り値　0以外:正常
0:エラー

・引数

Id As Long	デバイスID(値渡し)	
	0 : Master HDD	
	1 : Slave HDD	
Type As Long	デバイスタイプ(参照渡し)	
	ATADEVICE	HDD(ATAデバイス)
	ATAPIDEVICE	CD-ROM
	NODEVICE	未接続
Status As Long	デバイスステータス(参照渡し)	
	DEVSTAT_GOOD	正常
	DEVSTAT_NOTEXIST	未接続
	DEVSTAT_BROKEN	故障

・処理概要　ソフトミラーのデバイス状態を取得する。

・例

```
Dim ret As Long
Dim Id As Long
Dim Type As Long
Dim Status As Long
'ソフトミラーのデバイス状態を取得する。
Id = 0
ret = GetSmiDevStatus( Id, Type, Status )
```

7.3.2 PL_Ras.dll関数

■ PlDevWordWrite

・呼び出し形式　Declare Function PlDevWordWrite Lib "PL_Ras.dll"
(ByVal Addr As Long, ByVal wData As Long) As Long

・戻り値　0:正常
0以外:エラー

・引数

Addr As Long	書き込むメモリのワードアドレス
wData As Long	書き込むデータ(0～65535)

・処理概要　共有メモリへの書き込みを行います。

・例

```
' アドレス255へデータ255を書き込む
Dim ret As Long
ret = PlDevWordWrite(255, 255)
```

■ PlDevWordRead

・呼び出し形式　Declare Function PlDevWordRead Lib "PL_Ras.dll"
(ByVal Addr As Long, ByRef wData As Long) As Long

・戻り値　0:正常
0以外:エラー

・引数

Addr As Long	読み出すメモリのワードアドレス
wData As Long	読み出すデータ(0～65535)

・処理概要　共有メモリへの読み出しを行います。

・例

```
' アドレス255のデータ読み出し
Dim ret As Long
Dim wData As Long
ret = PlDevWordRead(255, wData)
```

7.3.3 PL_BLIoc.dll関数

■ InitBLIoctl

・呼び出し形式　Declare Sub InitBLIoctl Lib "PL_BLIoc.dll" ()
・戻り値　なし
・引数　なし
・処理概要　CPL_BLIoctlオブジェクトを作成する。作成されたオブジェクトはEndBLIoctl関数が呼ばれるまで破棄されない。
・例　Call InitBLIoctl

■ EndBLIoctl

・呼び出し形式　Declare Sub EndBLIoctl Lib "PL_BLIoc.dll" ()
・戻り値　なし
・引数　なし
・処理概要　InitBLIoctl関数で作成したオブジェクトを破棄する。
・例　Call EndBLIoctl

■ GetBLDrvHandle

・呼び出し形式　Declare Function GetBLDrvHandle Lib "PL_BLIoc.dll"
(ByRef hndl As Long) As Long
・戻り値　0:正常
1:エラー
・引数　hndl As Long　　デバイスドライバハンドル(参照渡し)
・処理概要　デバイスドライバとのやり取りを行うためのデバイスドライバハンドルを取得する。
・例

```
Dim ret  As Long
Dim hndl As Long
ret = GetBLDrvHandle( hndl )
```

・バックライトコントロールデバイスドライバが動作していない場合は、エラー（戻り値:1）となります。

■ GetBLDrvVersion

・呼び出し形式　Declare Function GetBLDrvVersion Lib "PL_BLIoc.dll"
(ByRef Major As Long, ByRef Minor As Long) As Long

・戻り値　0以外: 正常
0:エラー

・引数　Major As Long　バージョン情報(Major,0～99)　(参照渡し)
Minor As Long　バージョン情報(Minor,0～99)　(参照渡し)

・処理概要　ドライババージョン情報を取得する。

・例
```
Dim ret As Long
Dim Major As Long
Dim Minor As Long
ret = GetBLDrvVersion(Major, Minor)
```

・ドライババージョンが1.10の場合は
Major:1　(10進数)
Minor:10　(10進数)
となります。

■ GetBLDrvVersionEx

・呼び出し形式　Declare Function GetBLDrvVersionEx Lib "PL_BLIoc.dll"
(ByRef Product As Long, Major As Long, ByRef Minor As Long)
As Long

・戻り値　0以外: 正常
0:エラー

・引数　Product As Long　機種情報(参照渡し)
Major As Long　バージョン情報(Major,0～99)　(参照渡し)
Minor As Long　バージョン情報(Minor,0～99)　(参照渡し)

・処理概要　ドライババージョン情報を取得する。

・例
```
Dim ret As Long
Dim Product As Long
Dim Major As Long
Dim Minor As Long
ret = GetBLDrvVersionEx(Product, Major, Minor)
```

・PL-5910シリーズでドライババージョンが1.00の場合は、
Product:4　(10進数)
Major:1　(10進数)
Minor:00　(10進数)
となります。

■ SetBLControl

・呼び出し形式　Declare Function SetBLControl Lib "PL_BLIoc.dll"
(ByVal BLFlag As Long) As Long

・戻り値　0以外: 正常
0:エラー

・引数　BLFlag As Long　設定パラメータ(値渡し)
BACKLIGHT_OFF　バックライトOFF
BACKLIGHT_ON　バックライトON

・処理概要　バックライトのON/OFFを設定する。

・例
```
Dim ret As Long
'バックライトコントロールON設定
ret = SetBLControl( BACKLIGHT_ON )
```

■ GetBLControl

・呼び出し形式　Declare Function GetBLControl Lib "PL_BLIoc.dll"
(ByRef BLFlag As Long) As Long

・戻り値　0以外: 正常
0:エラー

・引数　BLFlag As Long　バックライト状態(参照渡し)
BACKLIGHT_OFF　バックライトOFF
BACKLIGHT_ON　バックライトON

・処理概要　バックライトコントロール状態を取得する。

・例
```
Dim ret As Long
Dim BLFlag As Long
'バックライトコントロール状態取得
ret = GetBLControl( BLFlag )
```