Digital
Human Machine Interface

Pro-face®

PS-2000Bシリーズ

RAS-API
リファレンスマニュアル

1　概要

本書では、PS-2000B上で動作するRAS機能にアプリケーションからアクセスす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、PSB_Ioc.dllとPSB_Ras.dllの２種類があります。
PSB_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PSB_Ioc.dllを経由して以下の機能を使用できます。

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

PSB_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PSB_Ras.dllを経由して以下の機能を使用できます。

○共有メモリの読み出し
○共有メモリへの書き込み

2　動作環境

API-DLLが動作する環境は以下のとおりです。

OS	対応言語
Microsoft® Windows®2000 Microsoft® Windows®XP	Microsoft® Visual C Microsoft® Visual C++® Microsoft® Visual Basic®

※1 ソフトミラー(PL-SM900)使用時のみ

3　必要ファイル

PSB_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。

開発言語	ファイル名	備考
Visual C	PSB_Iocif.h	ドライバライブラリインターフェイス定義インクルードファイル
	PSB_Ioc.LIB	ライブラリ定義ファイル
	PSB_Ioc.dll	ダイナミックリンクライブラリファイル
Visual C++	PSB_Iocif.h	ドライバライブラリインターフェイス定義インクルードファイル※1
	PSB_Iocall.h	CPSB_Iocallクラス定義インクルードファイル※1
	PSB_Ioctl.h	CPSB_Ioctlクラス定義インクルードファイル
	PSB_Ioc.LIB	ライブラリ定義ファイル
	PSB_Ioc.dll	ダイナミックリンクライブラリファイル
	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)
	PSB_Smiloctl.h	CPSB_Smiloctl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)
Visual Basic	PSB_Ioc.bas	ドライバインターフェイス定義ファイル
	PSB_Ioc.LIB	ライブラリ定義ファイル
	PSB_Ioc.dll	ダイナミックリンクライブラリファイル

PSB_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。

開発言語	ファイル名	備考
Visual C	PSB_Ras.h	ドライバライブラリインターフェイス定義インクルードファイル
	PSB_Ras.LIB	ライブラリ定義ファイル
	PSB_Ras.dll	ダイナミックリンクライブラリファイル
Visual C++	PSB_Ras.h	ドライバライブラリインターフェイス定義インクルードファイル
	PSB_Ras.LIB	ライブラリ定義ファイル
	PSB_Ras.dll	ダイナミックリンクライブラリファイル
Visual Basic	PSB_Ras.LIB	ライブラリ定義ファイル
	PSB_Ras.dll	ダイナミックリンクライブラリファイル

MEMO

各ファイルは[Proface]フォルダの中の[PSBApi]フォルダ内に用意されています。必要なファイルを以下の位置またはアプリケーションと同一フォルダに格納してください。

OS	位置
Microsoft® Windows®2000	C:Winnt\System32
Microsoft® Windows®XP	C:Windows\System32

※1 インクルードするヘッダファイルの順序は以下のとおりです。

PSB_Iocall.hは自動的にインクルードされますので、直接インクルードしないでください。

```
#include PSB_Iocif.h
#include PSB_Ioctl.h
```

4　クラス内容

4.1 CPSB_Ioctlクラス

CPSB_Ioctl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

4.2 CPSB_Iocallクラス

CPSB_Ioctlでセットされたパラメータを使用し、DeviceControl(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CPSB_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 CPSB_SmiIoctlクラス

CPSB_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

5　関数仕様

5.1 Visual C用関数仕様

PSB_Ioc.dll用関数

関数名	説明
InitIoctl	CPSB_Ioctlオブジェクト作成
EndIoctl	CPSB_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
GetFanParam	FAN監視用パラメータ取得
GetCurrentFan	現在FAN値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
SetWdtCounter	ウォッチドックタイマカウンタ値設定
GetWdtCounter	ウォッチドックタイマカウンタ取得
SetWdtDOutMask	ウォッチドックタイマタイムアウト時の警告マスク設定
GetWdtDOutMask	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
GetWdtStatus	ウォッチドッグタイマ動作状況取得
SetDOut	汎用出力設定
GetDOut	汎用出力取得
GetDIn	汎用入力取得
ClearDIn	汎用入力ラッチ状態解除
SetDInMask	汎用入力マスク設定
GetDInMask	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
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
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
SetWdtResetMask	ウォッチドッグタイマのリセットマスク設定
GetWdtResetMask	ウォッチドッグタイマのリセットマスク取得
GetSmiDrvHandle	ソフトミラードライバのハンドルを取得
CloseSmiDrvHandle	ソフトミラードライバのハンドルを破棄
GetSmiAryStatus	ミラー状態の取得
GetSmiDevStatus	ディスク状態の取得

PSB_Ras.dll用関数

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

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。

MEMO　PsbDevWordWriteとPsbDevWordRead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
// CPSB_Ioctlオブジェクトの作成
InitIoctl();
iRet = GetDrvHandle(&hDrv);
    ・
    ・
// DOUT3への出力
bRet = SetDOut(PORT_DOUT3, 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

InitIoctl

呼び出し形式

void WINAPI InitIoctl(void)

戻り値

なし

引数

なし

処理概要

CPSB_Ioctlオブジェクトを作成します。作成したオブジェクトはEndIoctlが呼ばれるまで破棄されません。

使用例

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

CPSB_Ioctlオブジェクトを破棄します。

使用例

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します。

使用例

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

MEMO　RAS/システムモニタ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で取得したデバイスドライバハンドルを破棄します。

使用例

```
BOOL ret;
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　　バージョン情報へのポインタ

(I/O) int *pMinor　　バージョン情報へのポインタ

処理概要

ドライババージョン情報を取得します。

使用例

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

MEMO　例えばドライババージョンが1.00の場合は、以下のようになります。

Major : 1 (10進数)

Minor : 00 (10進数)

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

(I/O)　　int *pMajor　　バージョン情報へのポインタ

(I/O)　　int *pMinor　　バージョン情報へのポインタ

処理概要

機種情報とドライババージョン情報を取得します。

使用例

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

MEMO

例えば機種がPS-2000Bでドライババージョンが1.00の場合は、以下のようになります。

Product : 1 (10進数)

Major : 1 (10進数)

Minor : 00 (10進数)

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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
	MONITOR_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pSetup	取得データへのポインタ	
	0 : Disable	
	1 : Enable	

処理概要

現在のモニタ禁止/許可設定を取得します。

使用例

```
BOOL ret;
int Setup;
// CPUコア電圧セットアップ状態取得
ret = GetMonitorSetup(MONITOR_VOLT_CPU, &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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
(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します。

使用例

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

MEMO

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)単位で使用するとき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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
(I/O) int *pData	電圧値(単位 mV)へのポインタ	

処理概要

電圧値を取得します。

使用例

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

MEMO

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)単位で使用するとき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

GetFanParam

呼び出し形式

BOOL WINAPI GetFanParam(int Selector, 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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pLLimit	FAN(RPM)下限回転数へのポインタ	
	(RPM : 1分間あたりの回転数)	

処理概要

FAN監視用のパラメータを取得します。

使用例

```
BOOL ret;
int LLimit;
// CPU FAN下限回転数取得
ret = GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

GetCurrentFan

呼び出し形式

BOOL WINAPI GetCurrentFan(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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pData	FAN(RPM)回転数へのポインタ	
	(RPM : 1分間あたりの回転数)	

処理概要

FANの現在の回転数を取得します。

使用例

```
BOOL ret;
int Data;
// CPU FAN回転数取得
ret = GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

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_CPU	CPU温度
	MONITOR_TEMP_SYSTEM	SYSTEM温度
(I/O) int *pULimit	温度(℃)の上限値へのポインタ	

処理概要

温度監視用のパラメータを取得します。

使用例

```
BOOL ret;
int ULimit;
// CPU 温度上限値取得
ret = GetTempParam(MONITOR_TEMP_CPU, &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_CPU　　CPU温度

MONITOR_TEMP_SYSTEM　　SYSTEM温度

(I/O) int *pData　　現在温度(℃)へのポインタ

処理概要

現在の温度を取得します。

使用例

```
BOOL ret;
int Data;
// CPU 温度取得
ret = GetCurrentTemp(MONITOR_TEMP_CPU, &Data);
```

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　　ウォッチドックタイマのカウント値

5～255秒

処理概要

ウォッチドッグタイマのカウント値を設定します。

使用例

```
BOOL ret;
// ウォッチドッグタイマのカウント値を10秒に設定
ret = SetWdtCounter(10);
```

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　　ウォッチドックタイマのカウント値へのポインタ

処理概要

ウォッチドッグタイマのカウント値を取得します。

使用例

```
BOOL ret;
int Counter;
// ウォッチドッグタイマのカウント値取得
ret = GetWdtCounter(&Counter);
```

SetWdtDOutMask

呼び出し形式

BOOL WINAPI SetWdtDOutMask(int Selector, int Mask)

戻り値

TRUE	正常
FALSE	エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Mask	マスク情報	
	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RASポートへのマスクを設定します。

使用例

```
BOOL ret;
// RASポートのDOUT0をマスク
ret = SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

GetWdtDOutMask

呼び出し形式

BOOL WINAPI GetWdtDOutMask(int Selector, int *pMask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(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

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RASポートへのマスクを取得します。

使用例

```
BOOL ret;
int Mask;
// RASポートのDOUT0のマスク情報取得
ret = GetWdtDOutMask(PORT_DOUT0, &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します。

使用例

```
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します。

使用例

```
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

カウントダウン中にウォッチドッグタイマのカウントダウン値を初期値に戻し、再度カウントダウンを開始します。

使用例

```
BOOL ret;
ret = RestartWdt();
```

MEMO 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RestarWdtを使用する場合はClearWdtTimeout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から使用してください。

GetWdtStatus

呼び出し形式

BOOL WINAPI GetWdtStatus(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	動作中

処理概要

ウォッチドッグタイマの動作状態を取得します。

使用例

```
BOOL ret;
int RunFlag;
ret = GetWdtStatus(&RunFlag);
```

SetDOut

呼び出し形式

BOOL WINAPI SetDOut(int Selector, int Dout)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Dout	出力状態	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を設定します。

使用例

```
BOOL ret;
ret = SetDOut(PORT_DOUT0, OUTPUT_ON);
```

GetDOut

呼び出し形式

BOOL WINAPI GetDOut(int Selector, int *pDout)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I/O) int *pDout	出力状態へのポインタ	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を取得します。

使用例

```
BOOL ret;
int Dout;
ret = GetDOut(PORT_DOUT0, &Dout);
```

GetDIn

呼び出し形式

BOOL WINAPI GetDIn(int Selector, int *pDin)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pDin	入力状態へのポインタ	
	INPUT_OFF	入力OFF
	INPUT_ON	入力ON

処理概要

汎用入力ポート(DIN)の入力情報を取得します。

使用例

```
BOOL ret;
int Din;
ret = GetDIn(PORT_DIN0, &Din);
```

ClearDIn

呼び出し形式

BOOL WINAPI ClearDIn(int Selector)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3

処理概要

汎用入力ポート(DIN)の入力状態を解除します。

使用例

```
BOOL ret;
ret = ClearDIn(PORT_DIN0);
```

SetDInMask

呼び出し形式

BOOL WINAPI SetDInMask(int Selector, int Mask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I) int Mask	マスク情報	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を設定します。

使用例

```
BOOL ret;
ret = SetDInMask(PORT_DIN0, MASK_OFF);
```

GetDInMask

呼び出し形式

BOOL WINAPI GetDInMask(int Selector, int *pMask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pMask	マスク情報へのポインタ	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を取得します。

使用例

```
BOOL ret;
int Mask;
ret = GetDInMask(PORT_DIN0, &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_ON　　マスクON

MASK_OFF　　マスクOFF

処理概要

リセットポートのマスク情報を設定します。

使用例

```
BOOL ret;
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_ON　　マスクON

MASK_OFF　　マスクOFF

処理概要

リセットポートのマスク情報を取得します。

使用例

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

GetEvent

呼び出し形式

BOOL WINAPI GetEvent (int Selector, int *pRasevnt)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector　取得パラメータ

EVENT_VOLT_CPU	CPU電圧
EVENT_VOLT_P33	+3.3V電圧
EVENT_VOLT_P50	+5V電圧
EVENT_VOLT_P12	+12V電圧
EVENT_VOLT_M12	-12V電圧
EVENT_VOLT_VIT	CPU電圧2
EVENT_FAN_CPU	CPU FAN
EVENT_FAN_POWER	POWER FAN
EVENT_TEMP_SYSTEM	SYSTEM温度
EVENT_TEMP_CPU	CPU温度
EVENT_DIN0	DIN0
EVENT_DIN1	DIN1
EVENT_DIN2	DIN2
EVENT_DIN3	DIN3
EVENT_WDT_TIMEOUT	ウォッチドッグタイマ

(I/O) int *pRasevnt　イベント情報へのポインタ

ERROR_EVENT_ON	イベントON
ERROR_EVENT_OFF	イベントOFF

処理概要

イベント情報を取得します。

使用例

```
BOOL ret;
int Rasevnt;
ret = GetEvent(EVENT_DIN0, &Rasevnt);
```

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	取得パラメータ	
	EVENT_VOLT_CPU	CPU電圧
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_VOLT_VIT	CPU電圧2
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ

処理概要

イベント情報をキャンセルします。

使用例

```
BOOL ret;
ret = ClearEvent(EVENT_DIN0);
```

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　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_ERR　　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します。

使用例

```
BOOL ret;
int Timebuf;
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します。

使用例

```
BOOL ret;
ret = ClearWdtTimeout();
```

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　　マスク状態

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によるリセットのマスク状態を設定します。

使用例

```
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

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します。

使用例

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

PsbDevWordWrite

呼び出し形式

long PsbDevWordWrite(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。

使用例

```
//アドレス255へデータ255を書き込む
long ret;
ret = PsbDevWordWrite(255, 255);
```

PsbDevWordRead

呼び出し形式

long PsbDevWordRead(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。

使用例

```
//アドレス255のデータ読み出し
long ret;
long wData;
ret = PsbDevWordRead(255, &wData);
```

GetSmiDrvHandle

呼び出し形式

```
int WINAPI GetSmiDrvHandle(void)
```

戻り値

0　正常

1　エラー

引数

なし

処理概要

ソフトミラードライバハンドルを取得する。

使用例

```
//ハンドルを取得する。
int ret;
ret = GetSmiDrvHandle();
```

MEMO　ソフトミラー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

ソフトミラードライバハンドルを破棄する。

使用例

```
//ハンドルを破棄する。
BOOL ret;
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　　ミラー状態へのポインタ

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

処理概要

ソフトミラーのミラー状態を取得する。

使用例

```
BOOL ret;
int Status;
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	ATAデバイス
	ATAPIDEVICE	CD-ROMドライブ
	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。

使用例

```
//マスターHDDのデバイス状態を取得する。
BOOL ret;
int Type, Status;
ret = GetSmiDevStatus (0, &Type, &Status);
```

5.4 Visual C++用関数仕様

PSB_Ioc.dll用関数

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
GetFanParam	FAN監視用パラメータ取得
GetCurrentFan	現在FAN値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
SetWdtCounter	ウォッチドックタイマカウンタ値設定
GetWdtCounter	ウォッチドックタイマカウンタ取得
SetWdtDOutMask	ウォッチドックタイマタイムアウト時の警告マスク設定
GetWdtDOutMask	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
GetWdtStatus	ウォッチドッグタイマ動作状況取得
SetDOut	汎用出力設定
GetDOut	汎用出力取得
GetDIn	汎用入力取得
ClearDIn	汎用入力ラッチ状態解除
SetDInMask	汎用入力マスク設定
GetDInMask	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
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
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
SetWdtResetMask	ウォッチドッグタイマのリセットマスク設定
GetWdtResetMask	ウォッチドッグタイマのリセットマスク取得
GetSmiDrvHandle	ソフトミラードライバハンドル取得
CloseSmiDrvHandle	ソフトミラードライバハンドル破棄
GetSmiAryStatus	ソフトミラーアレイ状態取得
GetSmiDevStatus	ソフトミラーデバイス状態取得

PSB_Ras.dll用関数

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

5.5 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。

MEMO PsbDevWordWriteとPsbDevWordRead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;
// デバイスハンドルの取得
CPSB_Ioctl m_Ioc;
iRet = m_Ioc.GetDrvHandle();
    ・
    ・
// DOUT0への出力
bRet = m_Ioc.SetDOut(PORT_DOUT0, OUTPUT_ON);
    ・
    ・
// デバイスハンドルの破棄
bRet = m_Ioc.CloseDrvHandle();
```

5.6 Visual C++用関数仕様詳細

GetDrvHandle

呼び出し形式

int GetDrvHandle(HANDLE *pHndl) または、int GetDrvHandle()

戻り値

0　正常

1　エラー

引数

(I/O) HANDLE *pHndl　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うためのデバイスドライバハンドルを取得します。

使用例1

```
int ret;
HANDLE Hndl;
ret = ::GetDrvHandle(&Hndl);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
int ret;
ret = m_Ioctl.GetDrvHandle();
```

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で取得したデバイスドライバハンドルを破棄します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.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　　バージョン情報へのポインタ

(I/O) int *pMinor　　バージョン情報へのポインタ

処理概要

ドライババージョン情報を取得します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Major, Minor;
ret = m_Ioctl.GetDrvVersion(&Major, &Minor);
```

MEMO　例えばドライババージョンが1.00の場合、以下のようになります。

Major : 1 (10進数)

Minor : 00 (10進数)

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

(I/O) int *pMajor　　バージョン情報へのポインタ

(I/O) int *pMinor　　バージョン情報へのポインタ

処理概要

機種情報とドライババージョン情報を取得します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Product, Major, Minor;
ret = m_Ioctl.GetDrvVersionEx(&Product, &Major, &Minor);
```

MEMO

例えば機種がPS-2000Bでドライババージョンが1.00の場合、以下のようになります。

Product : 1 (10進数)

Major : 1 (10進数)

Minor : 00 (10進数)

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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
	MONITOR_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pSetup	取得データへのポインタ	
	0 : Disable	
	1 : Enable	

処理概要

現在のモニタ禁止/許可設定を取得します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Setup;
// CPUコア電圧セットアップ状態取得
ret = m_Ioctl.GetMonitorSetup(MONITOR_VOLT_CPU, &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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
(I/0) int *pULimit	電圧上限値(単位 mV)へのポインタ	
(I/0) 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します。

使用例1

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

使用例2

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

MEMO　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)単位で使用するとき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 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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
(I/O) int *pData	電圧値(単位 mV)へのポインタ	

処理概要

電圧値を取得します。

使用例1

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

使用例2

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

MEMO

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)単位で使用するとき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

GetFanParam

呼び出し形式

```
BOOL GetFanParam(int Selector, 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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pLLimit	FAN(RPM)下限回転数へのポインタ	
	(RPM : 1分間あたりの回転数)	

処理概要

FAN監視用のパラメータを取得します。

使用例1

```
BOOL ret;
int LLimit;
// CPU FAN下限回転数取得
ret = ::GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int LLimit;
// CPU FAN下限回転数取得
ret = m_Ioctl.GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

GetCurrentFan

呼び出し形式

BOOL GetCurrentFan(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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
(I/O) int *pData	FAN(RPM)回転数へのポインタ	
	(RPM : 1分間あたりの回転数)	

処理概要

FANの現在の回転数を取得します。

使用例1

```
BOOL ret;
int Data;
// CPU FAN回転数取得
ret = ::GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Data;
// CPU FAN回転数取得
ret = m_Ioctl.GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

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(℃)の上限値へのポインタ
```

処理概要

温度監視用のパラメータを取得します。

使用例1

```
BOOL ret;
int ULimit;
// CPU 温度上限値取得
ret = ::GetTempParam(MONITOR_TEMP_CPU, &ULimit);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int ULimit;
// CPU 温度上限値取得
ret = m_Ioctl.GetTempParam(MONITOR_TEMP_CPU, &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_CPU	CPU温度
	MONITOR_TEMP_SYSTEM	SYSTEM温度
(I/O) int *pData	現在温度(℃)へのポインタ	

処理概要

現在の温度を取得します。

使用例1

```
BOOL ret;
int Data;
// CPU 温度取得
ret = ::GetCurrentTemp(MONITOR_TEMP_CPU, &Data);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Data;
// CPU 温度取得
ret = m_Ioctl.GetCurrentTemp(MONITOR_TEMP_CPU, &Data);
```

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　　ウォッチドックタイマのカウント値
5～255秒

処理概要

ウォッチドッグタイマのカウント値を設定します。

使用例1

```
BOOL ret;
// ウォッチドッグタイマのカウント値を10秒に設定
ret = ::SetWdtCounter(10);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
// ウォッチドッグタイマのカウント値を10秒に設定
ret = m_Ioctl.SetWdtCounter(10);
```

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　　ウォッチドックタイマのカウント値へのポインタ

処理概要

ウォッチドッグタイマのカウント値を取得します。

使用例1

```
BOOL ret;
int Counter;
// ウォッチドッグタイマのカウント値取得
ret = ::GetWdtCounter(&Counter);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Counter;
// ウォッチドッグタイマのカウント値取得
ret = m_Ioctl.GetWdtCounter(&Counter);
```

SetWdtDOutMask

呼び出し形式

BOOL SetWdtDOutMask(int Selector, int Mask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Mask	マスク情報	
	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RASポートへのマスクを設定します。

使用例1

```
BOOL ret;
// RASポートのDOUT0をマスク
ret = ::SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
// RASポートのDOUT0をマスク
ret = m_Ioctl.SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

GetWdtDOutMask

呼び出し形式

BOOL GetWdtDOutMask(int Selector, int *pMask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(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

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RASポートへのマスク情報を取得します。

使用例1

```
BOOL ret;
int Mask;
// RASポートのDOUT0のマスク情報取得
ret = ::GetWdtDOutMask(PORT_DOUT0, &Mask);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Mask;
// RASポートのDOUT0のマスク情報取得
ret = m_Ioctl.GetWdtDOutMask(PORT_DOUT0, &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します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.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します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.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

カウントダウン中にウォッチドッグタイマのカウントダウン値を初期値に戻し、再度カウントダウンを開始します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.RestartWdt();
```

MEMO

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RestarWdtを使用する場合はClearWdtTimeout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から使用してください。

GetWdtStatus

呼び出し形式

BOOL GetWdtStatus(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　　動作中

処理概要

ウォッチドッグタイマの動作状態を取得します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int RunFlag;
ret = m_Ioctl.GetWdtStatus(&RunFlag);
```

SetDOut

呼び出し形式

BOOL SetDOut(int Selector, int Dout)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Dout	出力状態	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を設定します。

使用例1

```
BOOL ret;
ret = ::SetDOut(PORT_DOUT0, OUTPUT_ON);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.SetDOut(PORT_DOUT0, OUTPUT_ON);
```

GetDOut

呼び出し形式

```
BOOL GetDOut(int Selector, int *pDout)
```

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector		設定項目	
		PORT_DOUT0	DOUT0
		PORT_DOUT1	DOUT1
		PORT_DOUT2	DOUT2
		PORT_DOUT3	DOUT3
(I/O)	int *pDout	出力状態へのポインタ	
		OUTPUT_OFF	出力OFF
		OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を取得します。

使用例1

```
BOOL ret;
int Dout;
ret = ::GetDOut(PORT_DOUT0, &Dout);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Dout;
ret = m_Ioctl.GetDout(PORT_DOUT0, &Dout);
```

GetDIn

呼び出し形式

BOOL GetDIn(int Selector, int *pDin)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pDin	入力状態へのポインタ	
	INPUT_OFF	入力OFF
	INPUT_ON	入力ON

処理概要

汎用入力ポート(DIN)の出力情報を取得します。

使用例1

```
BOOL ret;
int Din;
ret = ::GetDIn(PORT_DIN0, &Din);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Din;
ret = m_Ioctl.GetDIn(PORT_DOUT0, &Din);
```

ClearDIn

呼び出し形式

BOOL ClearDIn(int Selector)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3

処理概要

汎用入力ポート(DIN)の入力状態を解除します。

使用例1

```
BOOL ret;
ret = ::ClearDIn(PORT_DIN0);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.ClearDIn(PORT_DIN0);
```

SetDInMask

呼び出し形式

BOOL SetDInMask(int Selector, int Mask)

戻り値

TRUE	正常
FALSE	エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I) int Mask	マスク情報	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を設定します。

使用例1

```
BOOL ret;
ret = ::SetDInMask(PORT_DIN0, MASK_OFF);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.SetDInMask(PORT_DIN0, MASK_OFF);
```

GetDInMask

呼び出し形式

BOOL GetDInMask(int Selector, int *pMask)

戻り値

TRUE　　正常

FALSE　　エラー

引数

(I) int Selector	設定項目	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pMask	マスク情報へのポインタ	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を取得します。

使用例1

```
BOOL ret;
int Mask;
ret = ::GetDInMask(PORT_DIN0, &Mask);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Mask;
ret = m_Ioctl.GetDInMask(PORT_DIN0, &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_ON	マスクON
MASK_OFF	マスクOFF

処理概要

リセットポートのマスク情報を設定します。

使用例1

```
BOOL ret;
ret = ::SetResetMask(MASK_OFF);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.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_ON　　マスクON

MASK_OFF　　マスクOFF

処理概要

リセットポートのマスク情報を取得します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int  Mask;
ret = m_Ioctl.GetResetMask(&Mask);
```

GetEvent

呼び出し形式

BOOL GetEvent(int Selector, int *pRasevnt)

戻り値

TRUE	正常
FALSE	エラー

引数

(I) int Selector	取得パラメータ	
	EVENT_VOLT_CPU	CPU電圧
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_VOLT_VIT	CPU電圧2
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
(I/O) int *pRasevnt	イベント情報へのポインタ	
	ERROR_EVENT_ON	イベントON
	ERROR_EVENT_OFF	イベントOFF

処理概要

イベント情報を取得します。

使用例1

```
BOOL ret;
int Rasevnt;
ret = ::GetEvent(EVENT_DIN0, &Rasevent);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Rasevnt;
ret = m_Ioctl.GetEvent(EVENT_DIN0, &Rasevnt);
```

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	取得パラメータ	
	EVENT_VOLT_CPU	CPU電圧
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_VOLT_VIT	CPU電圧2
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ

処理概要

イベント情報をキャンセルします。

使用例1

```
BOOL ret;
ret = ::ClearEvent(EVENT_DIN0);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.ClearEvent(EVENT_DIN0);
```

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　　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_ERR　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します。

使用例1

```
BOOL ret;
int Timebuf;
ret = ::GetWdtTimeout(&Timebuf);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Timebuf;
ret = m_Ioctl.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解除します。

使用例1

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

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.ClearWdtTimeout();
```

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

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します。

使用例1

```
BOOL ret;
ret = ::SetWdtResetMask(MASK_ON);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.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

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します。

使用例1

```
BOOL ret;
int Mask;
ret = ::GetWdtResetMask(&Mask);
```

使用例2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Mask;
ret = m_Ioctl.GetWdtResetMask(&Mask);
```

PsbDevWordWrite

呼び出し形式

long PsbDevWordWrite(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。

使用例

```
//アドレス255へデータ255を書き込む
long ret;
ret = PsbDevWordWrite(255, 255);
```

PsbDevWordRead

呼び出し形式

long PsbDevWordRead(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/0) 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。

使用例

```
//アドレス255のデータ読み出し
long ret;
long wData;
ret = PsbDevWordRead(255, &wData);
```

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のドライバハンドルを取得する。

使用例1

```
int  ret;
ret = ::GetSmiDrvHandle();
```

使用例2

```
CPSB_SmiIoctl  m_SmiIoctl;
int  ret;
ret = m_SmiIoctl.GetSmiDrvHandle();
```

MEMO ソフトミラー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。

CloseSmiDrvHanlde

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

ソフトミラーデバイスドライバのドライバハンドルを破棄する。

使用例1

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

使用例2

```
CPSB_SmiIoctl   m_SmiIoctl;
BOOL  ret;
ret = m_SmiIoctl.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	ミラー状態へのポインタ	
		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。

使用例1

```
BOOL      ret;
int Status;
ret = ::GetSmiAryStatus(&Status);
```

使用例2

```
CPSB_SmiIoctl   m_SmiIoctl;
BOOL      ret;
int       Status;
ret = m_SmiIoctl.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	デバイスタイプへのポインタ	
	ATADEVICE	ATAデバイス
	ATAPIDEVICE	CD-ROM
	NODEVICE	未接続
(I/O)int *pStatus	デバイス状態へのポインタ	
	DEVSTAT_GOOD	正常
	DEVSTAT_NOTEXIST	未接続
	DEVSTAT_BROKEN	故障

処理概要

接続されているデバイスの状態を取得する。

使用例1

```
BOOL      ret;
int       Type, Status
ret = ::GetSmiDrvStatus(0, &Type, &Status);
```

使用例2

```
CPSB_SmiIoctl   m_SmiIoctl;
BOOL      ret;
int       Type, Status
ret = m_SmiIoctl.GetSmiDrvStatus(0, &Type, &Status);
```

MEMO

このページは、空白です。
ご自由にお使いください。

5.7 Visual Basic用関数仕様

PSB_Ioc.dll用関数

関数名	説明
InitIoctl	CPSB_Ioctlオブジェクト作成
EndIoctl	CPSB_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
GetFanParam	FAN監視用パラメータ取得
GetCurrentFan	現在FAN値取得
GetTempParam	温度監視用パラメータ取得
GetCurrentTemp	現在温度値取得
SetWdtCounter	ウォッチドックタイマカウンタ値設定
GetWdtCounter	ウォッチドックタイマカウンタ取得
SetWdtDOutMask	ウォッチドックタイマタイムアウト時の警告マスク設定
GetWdtDOutMask	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
GetWdtStatus	ウォッチドッグタイマ動作状況取得
SetDOut	汎用出力設定
GetDOut	汎用出力取得
GetDIn	汎用入力取得
ClearDIn	汎用入力ラッチ状態解除
SetDInMask	汎用入力マスク設定
GetDInMask	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
GetEvent	エラーイベント取得
ClearEvent	エラーイベント消去
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
SetWdtResetMask	ウォッチドッグタイマのリセットマスク設定
GetWdtResetMask	ウォッチドッグタイマのリセットマスク取得
GetSmiDrvHandle	ソフトミラードライバハンドル取得
CloseSmiDrvHandle	ソフトミラードライバハンドル破棄
GetSmiAryStatus	ソフトミラーアレイ状態取得
GetSmiDevStatus	ソフトミラーデバイス状態取得

PSB_Ras.dll用関数

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

5.8 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。

MEMO　PsbDevWordWriteとPsbDevWordRead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)
    ・
    ・
'DOUT0への出力
Dim ret As Long
ret = SetDOut(PORT_DOUT0, 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
```

5.9 Visual Basic用関数仕様詳細

InitIoctl

呼び出し形式

Declare Sub InitIoctl Lib“PSB_Ioc.dll”()

戻り値

なし

引数

なし

処理概要

CPSB_Ioctlオブジェクトを作成します。作成したオブジェクトはEndIoctlが呼ばれるまで破棄されません。

使用例

Call InitIoctl

EndIoctl

呼び出し形式

Declare Sub EndIoctl Lib“PSB_Ioc.dll”()

戻り値

なし

引数

なし

処理概要

CPSB_Ioctlオブジェクトを破棄します。

使用例

Call EndIoctl

GetDrvHandle

呼び出し形式

```
Declare Function GetDrvHandle Lib ”PSB_Ioc.dll ”(ByRef Hndl As Long) As Long
```

戻り値

0　　正常

0以外　　エラー

引数

Hndl As Long　　デバイスハンドル(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します。

使用例

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

MEMO　RAS/システムモニタ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

```
Declare Function CloseDrvHandle Lib  ”PSB_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で取得したデバイスドライバハンドルを破棄します。

使用例

```
Dim ret As Long
ret = CloseDrvHandle()
```

GetDrvVersion

呼び出し形式

Declare Function GetDrvVersion Lib “PSB_Ioc.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(参照渡し)

Minor As Long　　バージョン情報(参照渡し)

処理概要

ドライババージョン情報を取得します。

使用例

```
Dim ret As Long
Dim Major As Long
Dim Minor As Long
ret = GetDrvVersion(Major, Minor)
```

MEMO　例えばドライババージョンが1.00の場合は、以下のようになります。

Major : 1 (10進数)

Minor : 00 (10進数)

GetDrvVersionEx

呼び出し形式

Daclare Function GetDrvVersionEx Lib “PSB_Ioc.dll” (ByRef Product As Long, ByRef 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(参照渡し)
Minor As Long	バージョン情報(参照渡し)

処理概要

機種情報とドライババージョン情報を取得します。

使用例

```
Dim ret As Long
Dim Product As Long
Dim Major As Long
Dim Minor As Long
ret = GetDrvVersionEx(Product, Major, Minor)
```

MEMO

例えば機種がPS-2000Bでドライババージョンが1.00の場合は、以下のようになります。

Product : 1 (10進数)

Major : 1 (10進数)

Minor : 00 (10進数)

GetMonitorSetup

呼び出し形式

Declare Function GetMonitorSetup Lib “PSB_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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
	MONITOR_TEMP_SYSTEM	SYSTEM温度
	MONITOR_TEMP_CPU	CPU温度
	MONITOR_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
Setup As Long	取得データ(参照渡し)	
	0 : Disable	
	1 : Enable	

処理概要

現在のモニタ禁止/許可設定を取得します。

使用例

```
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_CPU, Setup)
```

GetVoltParam

呼び出し形式

Declare Function GetVoltParam Lib “PSB_Ioc.dll” (ByVal Selector As Long, ByRef ULimit As Long, ByRef LLimit 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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
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します。

使用例

```
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_CPU, ULimit, LLimit)
```

MEMO

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)単位で使用するとき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 “PSB_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_CPU	CPU電圧
	MONITOR_VOLT_P33	+3.3V電圧
	MONITOR_VOLT_P50	+5.0V電圧
	MONITOR_VOLT_P12	+12V電圧
	MONITOR_VOLT_M12	-12V電圧
	MONITOR_VOLT_VIT	CPU電圧2
Data As Long	電圧値(単位 mV) (参照渡し)	

処理概要

電圧値を取得します。

使用例

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

MEMO

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)単位で使用するとき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

GetFanParam

呼び出し形式

Declare Function GetFanParam Lib “PSB_Ioc.dll” (ByVal Selector 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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
LLimit As Long	FAN(RPM)下限回転数(参照渡し)	
	(RPM : 1分間あたりの回転数)	

処理概要

FAN監視用のパラメータを取得します。

使用例

```
Dim ret As Long
Dim LLimit As Long
’CPU FAN下限回転数取得
ret = GetFanParam(MONITOR_FAN_CPU, LLimit)
```

GetCurrentFan

呼び出し形式

Declare Function GetCurrentFan Lib “PSB_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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
Data As Long	FAN(RPM)回転数(参照渡し)	
	(RPM : 1分間あたりの回転数)	

処理概要

FANの現在の回転数を取得します。

使用例

```
Dim ret As Long
Dim Data As Long
’CPU FAN回転数取得
ret = GetCurrentFan(MONITOR_FAN_CPU, Data)
```

GetTempParam

呼び出し形式

Declare Function GetTempParam Lib “PSB_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_CPU	CPU温度
	MONITOR_TEMP_SYSTEM	SYSTEM温度
ULimit As Long	温度(℃)の上限値(参照渡し)	

処理概要

温度監視用のパラメータを取得します。

使用例

```
Dim ret As Long
Dim ULimit As Long
’CPU 温度上限値取得
ret = GetTempParam(MONITOR_TEMP_CPU, ULimit)
```

GetCurrentTemp

呼び出し形式

Declare Function GetCurrentTemp Lib “PSB_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_CPU	CPU温度
	MONITOR_TEMP_SYSTEM	SYSTEM温度
Data As Long	現在温度(℃) (参照渡し)	

処理概要

温度を取得します。

使用例

```
Dim ret As Long
Dim Data As Long
’CPU 温度取得
ret = GetCurrentTemp(MONITOR_TEMP_CPU, Data)
```

SetWdtCounter

呼び出し形式

Declare Function SetWdtCounter Lib “PSB_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	ウォッチドックタイマのカウント値 5～255秒(値渡し)

処理概要

ウォッチドッグタイマのカウント値を設定します。

使用例

```
Dim ret As Long
’ウォッチドッグタイマのカウント値を10秒に設定
ret = SetWdtCounter(10)
```

GetWdtCounter

呼び出し形式

```
Declare Function GetWdtCounter Lib “PSB_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　　ウォッチドックタイマのカウント値(参照渡し)

処理概要

ウォッチドッグタイマのカウント値を取得します。

使用例

```
Dim ret As Long
Dim Counter As Long
’ウォッチドッグタイマのカウント値取得
ret = GetWdtCounter(Counter)
```

SetWdtDOutMask

呼び出し形式

Declare Function SetWdtDOutMask Lib “PSB_Ioc.dll” (ByVal Selector As Long, ByVal Mask 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)	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
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RASポートへのマスクを設定します。

使用例

```
Dim ret As Long
’RASポートのDOUT0をマスク
ret = SetWdtDOutMask(PORT_DOUT0, MASK_ON)
```

GetWdtDOutMask

呼び出し形式

Declare Function GetWdtDOutMask Lib “PSB_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)	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
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に出力するRASポートへのマスク情報を取得します。

使用例

```
Dim ret As Long
Dim Mask As Long
’RASポートのDOUT0のマスク情報取得
ret = GetWdtDOutMask(PORT_DOUT0, Mask)
```

StartWdt

呼び出し形式

Declare Function StartWdt Lib ”PSB_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します。

使用例

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

StopWdt

呼び出し形式

Declare Function StopWdt Lib “PSB_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します。

使用例

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

RestartWdt

呼び出し形式

Declare Function RestartWdt Lib “PSB_Ioc.dll”() As Long

戻り値

0以外　　正常

0　　エラー

引数

なし

処理概要

カウントダウン中にウォッチドッグタイマのカウントダウン値を初期値に戻し、再度カウントダウンを開始します。

使用例

```
Dim ret As Long
ret = RestartWdt()
```

MEMO 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RestarWdtを使用する場合はClearWdtTimeout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から使用してください。

GetWdtStatus

呼び出し形式

Declare Function GetWdtStatus Lib “PSB_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	動作中

処理概要

ウォッチドッグタイマの動作状態を取得します。

使用例

```
Dim ret As Long
Dim RunFlag As Long
ret = GetWdtStatus(RunFlag)
```

SetDOut

呼び出し形式

Declare Function SetDOut Lib “PSB_Ioc.dll” (ByVal Selector As Long, ByVal Dout 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)	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
Dout As Long	出力状態(値渡し)	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を設定します。

使用例

```
Dim ret As Long
ret = SetDOut(PORT_DOUT0, OUTPUT_ON)
```

GetDOut

呼び出し形式

Declare Function GetDOut Lib “PSB_Ioc.dll”(ByVal Selector As Long, ByRef Dout 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)	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
Dout As Long	出力状態(参照渡し)	
	OUTPUT_OFF	出力OFF
	OUTPUT_ON	出力ON

処理概要

汎用出力ポート(DOUT)の出力情報を取得します。

使用例

```
Dim ret As Long
Dim Dout As Long
ret = GetDOut(PORT_DOUT0, Dout)
```

GetDIn

呼び出し形式

Declare Function GetDIn Lib “PSB_Ioc.dll” (ByVal Selector As Long, ByRef Din 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)	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
Din As Long	入力状態(参照渡し)	
	INPUT_OFF	入力OFF
	INPUT_ON	入力ON

処理概要

汎用入力ポート(DIN)の出力情報を取得します。

使用例

```
Dim ret As Long
Dim Din As Long
ret = GetDIn(PORT_DIN0, Din)
```

ClearDIn

呼び出し形式

```
Declare Function ClearDIn Lib “PSB_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	設定項目(値渡し)	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3

処理概要

汎用入力ポート(DIN)の入力状態を解除する。

使用例

```
Dim ret As Long
ret = ClearDIn(PORT_DIN0)
```

SetDInMask

呼び出し形式

Declare Function SetDInMask Lib “PSB_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）	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
Mask As Long	マスク情報（値渡し）	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を設定します。

使用例

```
Dim ret As Long
ret = SetDInMask(PORT_DIN0, MASK_OFF)
```

GetDInMask

呼び出し形式

Declare Function GetDInMask Lib “PSB_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)	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
Mask As Long	マスク情報(参照渡し)	
	MASK_ON	マスクON
	MASK_OFF	マスクOFF

処理概要

汎用入力ポート(DIN)のマスク情報を取得します。

使用例

```
Dim ret As Long
Dim Mask As Long
ret = GetDInMask(PORT_DIN0, Mask)
```

SetResetMask

呼び出し形式

Declare Function SetResetMask Lib “PSB_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_ON　　マスクON

MASK_OFF　　マスクOFF

処理概要

リセットポートのマスク情報を設定します。

使用例

```
Dim ret As Long
ret = SetResetMask(MASK_OFF)
```

GetResetMask

呼び出し形式

Declare Function GetResetMask Lib “PSB_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_ON　　マスクON

MASK_OFF　　マスクOFF

処理概要

リセットポートのマスク情報を取得します。

使用例

```
Dim ret As Long
Dim Mask As Long
ret = GetResetMask(Mask)
```

GetEvent

呼び出し形式

Declare Function GetEvent Lib “PSB_Ioc.dll” (ByVal Selector As Long, ByRef Rasevnt 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_CPU	CPU電圧
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_VOLT_VIT	CPU電圧2
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
Rasevnt As Long	イベント情報(参照渡し)	
	ERROR_EVENT_ON	イベントON
	ERROR_EVENT_OFF	イベントOFF

処理概要

イベント情報を取得します。

使用例

```
Dim ret As Long
Dim Rasevnt As Long
ret = GetEvent(EVENT_DIN0, Rasevnt)
```

ClearEvent

呼び出し形式

Declare Function ClearEvent Lib “PSB_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	取得パラメータ(値渡し)	
	EVENT_VOLT_CPU	CPU電圧
	EVENT_VOLT_P33	+3.3V電圧
	EVENT_VOLT_P50	+5V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_VOLT_VIT	CPU電圧2
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM温度
	EVENT_TEMP_CPU	CPU温度
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ

処理概要

イベント情報をキャンセルします。

使用例

```
Dim ret As Long
ret = ClearEvent(EVENT_DIN0)
```

GetWdtTimeout

呼び出し形式

Declare Function GetWdtTimeout Lib “PSB_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　　タイムアウト状態(参照渡し)

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します。

使用例

```
Dim ret As Long
Dim Timebuf As Long
ret = GetWdtTimeout(Timebuf)
```

ClearWdtTimeout

呼び出し形式

Declare Funciton ClearWdtTimeout Lib “PSB_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解除します。

使用例

```
Dim ret As Long
ret = ClearWdtTimeout()
```

SetWdtResetMask

呼び出し形式

Declare Function SetWdtResetMask Lib “PSB_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

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します。

使用例

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

GetWdtResetMask

呼び出し形式

Declare Function GetWdtResetMask Lib “PSB_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

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します。

使用例

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

PsbDevWordWrite

呼び出し形式

```
Declare Function PsbDevWordWrite Lib “PSB_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。

使用例

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

PsbDevWordRead

呼び出し形式

```
Declare Function PsbDevWordRead Lib “PSB_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。

使用例

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

GetSmiDrvHandle

呼び出し形式

Declare Function GetSmiDrvHandle Lib “Psb_Ioc.dll”() As Long

戻り値

0　　正常

0以外　　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のドライバハンドルを取得する。

使用例

```
Dim ret As Long
ret = GetSmiDrvHandle()
```

MEMO ソフトミラー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。

CloseSmiDrvHanlde

呼び出し形式

Declare Function CloseSmiDrvHandle Lib ”Psb_Ioc.dll”() As Long

戻り値

0以外　　正常

0　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のドライバハンドルを破棄する。

使用例

```
Dim ret As Long
ret = CloseSmiDrvHandle()
```

GetSmiAryStatus

呼び出し形式

Declare Function GetSmiAryStatus Lib “Psb_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	ミラー状態(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。

使用例

```
Dim ret As Long
Dim Status As Long
ret = GetSmiAryStatus(Status)
```

GetSmiDevStatus

呼び出し形式

Declare Function Get SmiDevStatus lib “Psb_Ioc.dll”
(ByValID As Long, ByRef Type As Long, By Ref 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	ATAデバイス
	ATAPIDEVICE	CD-ROM
	NODEVICE	未接続
Status As Long	デバイス状態(参照渡し)	
	DEVSTAT_GOOD	正常
	DEVSTAT_NOTEXIST	未接続
	DEVSTAT_BROKEN	故障

処理概要

接続されているデバイスの状態を取得する。

使用例

```
Dim ret As Long
Dim Type As Long
Dim Status As Long
ret = GetSmiDrvStatus(0, Type, Status)
```

MEMO

このページは、空白です。
ご自由にお使いください。