Digital
Human Machine Interface

Pro-face®

PSシリーズAタイプ
(Eden™ ESP6000 - 667MHz Model)

APIリファレンスマニュアル

目次

1．3概要

本書では、PS-3700A(Eden™ **ESP**6000 - 667MHz Model)、および PS-3701A(Eden™ **ESP**6000 - 667MHz Model) 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、Psa_Ioc.dll、Psa_Ras.dll および Psa_Blc.dll の 3 種類があります。

Psa_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 Psa_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)
- システムモニタ現在情報 (電圧、ファン)
- ウォッチドッグパラメータ
- 警告処理
- イベント処理
- Psa_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 Psa_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
-

Psa_Blc.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 Psa_Blc.dll を経由して以下の機能を使用できます。

- バックライト制御
- バックライト輝度制御

各 dll ファイルは以下のフォルダに格納されています。

dll ファイル	格納場所
Psa_Ioc.dll Psa_Ras.dll Psa_Blc.dll	Windows ディレクトリの System32 例) C:\Winnt\System32

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

MEMO

・ Visual C++ .NET、Visual Basic .NET には対応していません。

3. 必要ファイル

3.1 Psa_Ioc 用ファイル

Psa_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	Psa_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
	Psa_Ioc.lib	ライブラリ定義ファイル
Visual C++	Psa_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
	Psa_Iocall.h	CPSA_Iocall クラス定義インクルードファイル
	Psa_Ioctl.h	CPSA_Ioctl クラス定義インクルードファイル
	Psa_Ioc.lib	ライブラリ定義ファイル
Visual Basic	Psa_Ioc.bas	ドライバインターフェイス定義ファイル
	Psa_Ioc.lib	ライブラリ定義ファイル

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。

Psa_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 Psa_Iocif.h
#include Psa_Ioctl.h
```

Psa_Blcall.h は自動でインクルードされます。

```
#include Psa_Blcif.h
#include Psa_Blctl.h
```

MEMO

・ 各ファイルは以下のフォルダに格納されています。

C:\Proface\Psaapi

3.2 Psa_Ras 用ファイル

Psa_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	Psa_Ras.h	共有メモリ Rasd/Write 定義ファイル
	Psa_Ras.lib	共有メモリ Rasd/Write ライブラリファイル
Visual C++	Psa_Ras.h	共有メモリ Rasd/Write 定義ファイル
	Psa_Ras.lib	共有メモリ Rasd/Write ライブラリファイル

MEMO
・ 各ファイルは以下のフォルダに格納されています。
C:\Proface\Psaapi

3.3 Psa_Blc 用ファイル

Psa_Blc.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	Psa_Blci.h	バックライトドライバインターフェイス定義ファイル
	Psa_Blc.lib	バックライトライブラリ定義ファイル
Visual C++	Psa_Blc.lib	バックライトライブラリ定義ファイル
	Psa_Blcif.h	バックライトドライバインターフェース定義ファイル
	Psa_Blcall.h	CPSA_Blcall クラス定義インクルードファイル
	Psa_Blctl.h	CPSA_Blctl クラス定義インクルードファイル
Visual Basic	Psa_Blc.bas	バックライトドライバインターフェイス定義ファイル
	Psa_Blc.lib	バックライトライブラリ定義ファイル

MEMO
・ 各ファイルは以下のフォルダに格納されています。
C:\Proface\Psaapi

4. クラス内容

4.1 CPSA_Ioctl クラス

CPSA_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 CPSA_Iocall クラス

CPSA_Ioctl 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 CPSA_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 CPSA_Blctl クラス

CPSA_Blctl 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

4.4 CPSA_Blcall クラス

CPSA_Blctl 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）を呼び出す。但し、このクラスは CPSA_Blctl から継承されているので直接使用することはない。

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. 関数仕様

5.1 Visual C用関数仕様

5.1.1 Psa_Ioc.dll用関数

関数名	説明
InitIoctl	CPSA_Ioctl オブジェクト作成
EndIoctl	CPSA_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
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値取得
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
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
GetLightblowErr	バックライト管切れ取得

5.1.2 Psa_Ras.dll 用関数

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

5.1.3 Psa_Blc.dll 用関数

関数名	説明
InitBlctl	CPSA_Blctl オブジェクト作成
EndBlctl	CPSA_Blctl 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
SetBlBrightness	バックライト輝度設定
GetBlBrightness	バックライト輝度取得

重 要

・ 作成したアプリケーションから dll ファイルを使用するためには、以下に dll ファイルを格納する必要があります。

OS	格納場所
Microsft® Windows®2000 Microsft® Windows®XP	・起動プログラムと同一ディレクトリ または ・Windows ディレクトリの System32 例）C:\Winnt\System32

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

・ PsaDevWordWriteとPsaDevWordReadのみを使用する場合は、ドライバオフ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。

5.2.1 サンプルプログラム例

◆Psa_Ioc.dllを使用する場合

```
// 変数宣言
BOOL bRet;
int iRet;
int iData;
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
// CPSA_Ioctl オブジェクトの作成
InitIoctl();
iRet = GetDrvHandle(&hDrv);
    ・
    ・
//+3.3V の監視
bRet = GetCurrentVolt(MONITOR_VOLT_P33, &iData);
    ・
    ・
// 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();
```

◆ Psa_Blc.dll を使用する場合

```
// 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
// CPSA_Ioctl オブジェクトの作成
InitBlctl();
iRet = GetBlDrvHandle(&hDrv);
    ・
    ・
//バックライトの OFF
bRet = SetBlControl(BACKLIGHT_OFF);
    ・
    ・
// 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 = CloseBlDrvHandle();
EndBlctl();
```

5.3 Visual C 用関数仕様詳細

5.3.1 Psa_Ioc.dll 用関数

InitIoctl

呼び出し形式

void WINAPI InitIoctl(void)

戻り値

なし

引数

なし

処理概要

CPSA_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

CPSA_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

・ 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 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

・ 機種が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) 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 : 3 (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_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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
	MONITOR_FAN_SYSTEM	SYSTEM 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;
// +3.3V セットアップ状態取得
ret = GetMonitorSetup(MONITOR_VOLT_P33, &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_P33	+3.3V 電圧
MONITOR_VOLT_P50	+5.0V 電圧
MONITOR_VOLT_P12	+12V 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します。

使用例

```
BOOL ret;
int ULimit, LLimit;
// +3.3V 上限下限値取得
ret = VoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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 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_P33	+3.3V 電圧
MONITOR_VOLT_P50	+5.0V 電圧
MONITOR_VOLT_P12	+12V 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

電圧値を取得します。

使用例

```
BOOL ret;
int Data;
// +3.3V 取得
ret = GetCurrentVolt(MONITOR_VOLT_P33, &Data);
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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

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
MONITOR_FAN_SYSTEM	SYSTEM 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 の下限回転数を取得します。

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
	MONITOR_FAN_SYSTEM	SYSTEM 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);
```

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;
int Counter;
// ウォッチドッグタイマのカウント値を 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) 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);
```

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

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

・ 1度、StartWdt をコールし、カウントダウンを開始していないとエラーになります。Time out が発生したあとはエラーをクリアし再度 StartWdt をコールしてカウントダウンを開始していないとエラーになります。

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);
```

GetEvent

呼び出し形式

BOOL WINAPI GetEvent(int Selector, int *pRasEvent)

戻り値

TRUE　正常

FALSE　エラー

引数

(I) int Selector	取得パラメータ	
	EVENT_VOLT_P33	+3.3V 電圧
	EVENT_VOLT_P50	+5.0V 電圧
	EVENT_VOLT_P12	+12V 電圧
	EVENT_VOLT_M12	-12V 電圧
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_FAN_SYSTEM	SYSTEM FAN
	EVENT_WDT_TIMEOUST	ウォッチドッグタイマ
	EVENT_BACKLIGHT	バックライト管切れ
(I) int *pRasEvent	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 RasEvent;
ret = GetEvent(EVENT_VOLT_P33, &RasEvent);
```

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_P33	+3.3V 電圧
EVENT_VOLT_P50	+5.0V 電圧
EVENT_VOLT_P12	+12V 電圧
EVENT_VOLT_M12	-12V 電圧
EVENT_FAN_CPU	CPU FAN
EVENT_FAN_SYSTEM	SYSTEM FAN
EVENT_FAN_POWER	POWER FAN
EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
EVENT_BACKLIGHT	バックライト管切れ

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_VOLT_P33);
```

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) 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/O) 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);
```

GetLightblowErr

呼び出し形式

BOOL WINAPI GetLightblowErr (int *pLight)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int *pLight　バックライト管切れ状態へのポインタ

BACKLIGHT_BLOWOUT　バックライト管切れ

BACKLIGHT_GLOW　正常

処理概要

バックライト管切れ状態を取得します。

使用例

```
BOOL ret;
int Light;
ret = GetLightblowErr(&Light);
```

5.3.2 Psa_Ras.dll 用関数

PsaDevWordWrite

呼び出し形式

long PsaDevWordWrite(long Addr, long wData)

戻り値

0　　正常

0 以外　エラー

引数

(I) long Addr　　書き込むメモリへのワードアドレス　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。

使用例

```
// アドレス 255 へデータ 100 を書き込む
long ret;
ret = PsaDevWordWrite(255, 100);
```

PsaDevWordRead

呼び出し形式

long PsaDevWordRead(long Addr, long *wData)

戻り値

0　　正常

0 以外　エラー

引数

(I)　long Addr　　読み出すメモリへのワードアドレス　0 ～ 255

(I/O) long *wData　読み出すデータ　　0 ～ 65535

処理概要

共有メモリからの読み出しを行います。

使用例

```
// アドレス 255 のデータを読み出す
long ret;
long wData;
ret = PsaDevWordRead(255, &wData);
```

5.3.3 Psa_Blc.dll 用関数

InitBlctl

呼び出し形式

void WINAPI InitBlctl(void)

戻り値

なし

引数

なし

処理概要

CPSA_Blctl オブジェクトを作成します。作成されたオブジェクトは EndBlctrl 関数が呼ばれるまで破棄されません。

使用例

```
//CPSA_Blctl オブジェクトの作成
InitBlctl();
```

EndBlctl

呼び出し形式

void WINAPI EndBlctl()

戻り値

なし

引数

なし

処理概要

InitBlctl 関数で作成したオブジェクトを破棄します。

使用例

```
EndBlctl();
```

GetBlDrvHandle

呼び出し形式

int WINAPI GetBlDrvHandle(HANDLE *pHndl)

戻り値

0　　正常

0以外　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やり取りをおこな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 = GetBlDrvHandl(& hndl);
```

CloseBlDrvHandle

呼び出し形式

BOOL WINAPI 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します。

使用例

```
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を取得します。

使用例

```
BOOL ret;
int Major, Minor;
GetBlDrvVersion(&Major, &Minor);
```

MEMO

・ 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)

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

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、機種情報を取得します。

使用例

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

MEMO

・ バージョンが 1.00 で製品が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) の場合は、以下のようになります。

Product : 3 (10 進数)

Major : 1 (10 進数)

Minor : 00 (10 進数)

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　　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 ON/OFF を設定します。

使用例

```
BOOL ret;
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_ON　　バックライト ON

BACKLIGHT_OFF　　バックライト OFF

処理概要

バックライト状態を取得します。

使用例

```
BOOL ret;
int BlFlag;
ret = GetBlControl(&BlFlag);
```

SetBlBrightness

呼び出し形式

BOOL WINAPI SetBlBrightness(int BlBright)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int BlBright　バックライト輝度

BRIGHT_LEVEL_0	輝度レベル０（とても暗い）
BRIGHT_LEVEL_1	輝度レベル１（暗い）
BRIGHT_LEVEL_2	輝度レベル２（明るい）
BRIGHT_LEVEL_3	輝度レベル３（とても明るい）

処理概要

バックライトの輝度を設定します。

使用例

```
BOOL ret;
ret = GetBlBrightness(BRIGHT_LEVEL_1);
```

GetBlBrightness

呼び出し形式

BOOL WINAPI GetBlBrightness(int *pBlBright)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int *pBlBright バックライト輝度へのポインタ

BRIGHT_LEVEL_0	輝度レベル０（とても暗い）
BRIGHT_LEVEL_1	輝度レベル１（暗い）
BRIGHT_LEVEL_2	輝度レベル２（明るい）
BRIGHT_LEVEL_3	輝度レベル３（とても明るい）

処理概要

バックライトの輝度を取得します。

使用例

```
BOOL ret;
int BlBright;
ret = GetBlBrightness(&BlBright);
```

5.4 Visual C++ 用関数仕様

5.4.1 Psa_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
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値取得
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
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	リセットマスク設定
GetWdtResetMask	リセットマスク取得
GetLightblowErr	バックライト管切れ取得

5.4.2 Psa_Ras.dll 用関数

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

5.4.3 Psa_Blc.dll用関数

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	バックライト状態設定
GetBlControl	バックライト状態取得
SetBlBrightness	バックライト輝度設定
GetBlBrightness	バックライト輝度取得

重　要

・ 作成したアプリケーションからdllファイルを使用するためには、以下にdllファイルを格納する必要があります。

OS	格納場所
Microsoft® Windows®2000 Microsoft® Windows®XP	・起動プログラムと同一ディレクトリ または ・Windowsディレクトリの System32 例）C:\Winnt\System32

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

・ PsaDevWordWrite と PsaDevWordRead のみを使用する場合は、ドライバオフ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。

5.5.1 サンプルプログラム例

◆ Psa_Ioc.dll 使用例

```
// 変数宣言
BOOL bRet;
int iData;
int iRet;
// ドライバハンドルの取得
CPSA_Ioctl m_Ioctl;
iRet = m_Ioctl.GetDrvHandle();
   ・
   ・
// +3.3V の監視
bRet = m_Ioctl.GetCurrentVolt(MONITOR_VOLDT_P33, &iData);
   ・
   ・
// ドライバハンドルの破棄
bRet = m_Ioctl.CloseDrvHandle();
```

◆ **Psa_Blc.dll 使用例**

```
// 変数宣言
BOOL bRet;
int iRet;
// ドライバハンドルの取得
CPSA_Blctl m_Blc;
iRet = m_Blc.GetBlDrvHandle();
   ・
   ・
// バックライトの OFF
bRet = m_Blc.SetBlControl(BACKLIGHT_OFF);
   ・
   ・
// ドライバハンドルの破棄
bRet = m_Blc.CloseBlDrvHandle();
```

5.6 Visual C++ 用関数仕様詳細

5.6.1 Psa_Ioc.dll 用関数

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

```
CPSA_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

```
CPSA_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

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

MEMO

・ 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

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

MEMO

・ 機種が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) 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 : 3 (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_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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
	MONITOR_FAN_SYSTEM	SYSTEM 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;
// +3.3V セットアップ状態取得
ret = ::GetMonitorSetup(MONITOR_VOLT_P33, &Setup);
```

使用例 2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Setup;
// +3.3V セットアップ状態取得
ret = m_Ioctl.GetMonitorSetup(MONITOR_VOLT_P33, &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_P33	+3.3V 電圧
	MONITOR_VOLT_P50	+5.0V 電圧
	MONITOR_VOLT_P12	+12V 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します。

使用例 1

```
BOOL ret;
int ULimit, LLimit;
// +3.3V 上限下限値取得
ret = ::GetVoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

使用例 2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int ULimit, LLimit;
// +3.3V 上限下限値取得
ret = m_Ioctl.GetVoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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_P33	+3.3V 電圧
MONITOR_VOLT_P50	+5.0V 電圧
MONITOR_VOLT_P12	+12V 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

電圧値を取得します。

使用例 1

```
BOOL ret;
int Data;
// +3.3V 取得
ret = ::GetCurrentVolt(MONITOR_VOLT_P33, &Data);
```

使用例 2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Data;
// +3.3V 取得
ret = m_Ioctl.GetCurrentVolt(MONITOR_VOLT_P33, &Data);
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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

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
	MONITOR_FAN_SYSTEM	SYSTEM 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 の下限回転数を取得します。

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

```
CPSA_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
MONITOR_FAN_SYSTEM	SYSTEM 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

現在の FAN 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

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Data;
// CPU FAN 回転数取得
ret = m_Ioctl.GetCurrentFan(MONITOR_FAN_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;
int Counter;
// ウォッチドッグタイマのカウント値を10秒に設定
Counter = 10;
ret = ::SetWdtCounter(Counter);
```

使用例2

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

```
CPSA_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);
```

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

```
CPSA_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

```
CPSA_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

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

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

MEMO

・ 1 度、StartWdt をコールし、カウントダウンを開始していないとエラーになります。Time out が発生したあとはエラーをクリアして再度 StartWdt をコールし、カウントダウンを開始していないとエラーになります。

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

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

GetEvent

呼び出し形式

BOOL GetEvent(int Selector, int *pRasEvent)

戻り値

TRUE　正常

FALSE　エラー

引数

(I) int Selector　取得パラメータ

EVENT_VOLT_P33	+3.3V 電圧
EVENT_VOLT_P50	+5.0V 電圧
EVENT_VOLT_P12	+12V 電圧
EVENT_VOLT_M12	-12V 電圧
EVENT_FAN_CPU	CPU FAN
EVENT_FAN_POWER	POWER FAN
EVENT_TEMP_SYSTEM	SYSTEM FAN
EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
EVENT_BACKLIGHT	バックライト管切れ

(I) int *pRasEvent 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 RasEvent;
ret = ::GetEvent(EVENT_VOLT_P33, &RasEvent);
```

使用例 2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int RasEvent;
ret = m_Ioctl.GetEvent(EVENT_VOLT_P33, &RasEvent);
```

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_P33	+3.3V電圧
	EVENT_VOLT_P50	+5.0V電圧
	EVENT_VOLT_P12	+12V電圧
	EVENT_VOLT_M12	-12V電圧
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_TEMP_SYSTEM	SYSTEM FAN
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
	EVENT_BACKLIGHT	バックライト管切れ

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_VOLT_P33);
```

使用例2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.ClearEvent(EVENT_VOLT_P33);
```

GetWdtTimeout

呼び出し形式

BOOL GetWdtTimeout(int *pTimebuf)

戻り値

TRUE　正常

FALSE　エラー

引数

(I) 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

```
CPSA_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

```
CPSA_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

```
CPSA_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

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

GetLightblowErr

呼び出し形式

BOOL GetLightblowErr(int *pLight)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int *pLight　バックライト管切れ状態へのポインタ

BACKLIGHT_BLOWOUT　バックライト管切れ

BACKLIGHT_GLOW　正常

処理概要

バックライト管切れ状態を取得します。

使用例 1

```
BOOL ret;
int Light;
ret = ::GetLightblowErr (&Light);
```

使用例 2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Light;
ret = m_Ioctl.GetLightblowErr(&Light);
```

5.6.2 Psa_Ras.dll用関数

PsaDevWordWrite

呼び出し形式

long PsaDevWordWrite(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。

使用例

```
// アドレス255へデータ100を書き込む
long ret;
ret = PsaDevWordWrite(255, 100);
```

PsaDevWordRead

呼び出し形式

long PsaDevWordRead(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 0～255

(I/O) long *wData　読み出すデータ 0～65535

処理概要

共有メモリからの読み出しを行います。

使用例

```
// アドレス255のデータを読み出す
long ret;
long wData;
ret = PsaDevWordRead(255, &wData);
```

5.6.3 Psa_Blc.dll 用関数

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) 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やり取りをおこな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

```
CPSA_Blctl m_Blc;
int ret;
ret = m_Blctl.GetBlDrvHandl();
```

使用例 2

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

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します。

使用例1

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

使用例2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.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を取得します。

使用例 1

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

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
int Major, Minor;
ret = m_Blctl.GetBlDrvVersion(&Major, &Minor);
```

MEMO

・ 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)

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、機種情報を取得します。

使用例 1

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

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
int Product, Major, Minor;
ret = m_Blctl.GetBlDrvVersionEx(&Product, &Major, &Minor);
```

MEMO

・ バージョンが 1.00 で製品が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) の場合、以下のようになります。

Product : 3 (10 進数)

Major : 1 (10 進数)

Minor : 00 (10 進数)

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 ON/OFF を設定します。

使用例 1

```
BOOL ret;
ret = ::SetBlcontrol(BACKLIGHT_ON);
```

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.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_ON　バックライト ON

BACKLIGHT_OFF　バックライト OFF

処理概要

バックライト状態を取得します。

使用例 1

```
BOOL ret;
int BlFlag;
ret = ::GetBlControl(&BlFlag);
```

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
int BlFlag;
ret = m_Blctl.GetBlControl(&BlFlag);
```

SetBlBrightness

呼び出し形式

BOOL SetBlControl(int BlBright)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int BlBright　バックライト輝度

BRIGHT_LEVEL_0　輝度レベル 0 (とても暗い)

BRIGHT_LEVEL_1　輝度レベル 1 (暗い)

BRIGHT_LEVEL_2　輝度レベル 2 (明るい)

BRIGHT_LEVEL_3　輝度レベル 3 (とても明るい)

処理概要

バックライトの輝度を設定します。

使用例 1

```
BOOL ret;
ret = ::GetBlBrightness(BRIGHT_LEVEL_1);
```

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.SetBlBrightness(BRIGHT_LEVEL_1);
```

GetBlBrightness

呼び出し形式

BOOL GetBlControl(int *pBlBright)

戻り値

TRUE　正常

FALSE　エラー

引数

(I/O) int *pBlBright　バックライト輝度へのポインタ

BRIGHT_LEVEL_0　輝度レベル 0 (とても暗い)

BRIGHT_LEVEL_1　輝度レベル 1 (暗い)

BRIGHT_LEVEL_2　輝度レベル 2 (明るい)

BRIGHT_LEVEL_3　輝度レベル 3 (とても明るい)

処理概要

バックライトの輝度を取得します。

使用例 1

```
BOOL ret;
int BlBright;
ret = ::GetBlBrightness(&BlBright);
```

使用例 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
int BlBright;
ret = m_Blctl.GetBlBrightness(&BlBright);
```

5.7 Visual Basic 用関数仕様

5.7.1 Psa_Ioc.dll 用関数

関数名	説明
InitIoctl	CPSA_Ioctl オブジェクト作成
EndIoctl	CPSA_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
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値取得
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
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
GetLightblowErr	バックライト管切れの取得

5.7.2 Psa_Ras.dll 用関数

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

5.7.3 Psa_Blc.dll 用関数

関数名	説明
InitBlctl	CPSA_Blctl オブジェクト作成
EndBlctl	CPSA_Blctl 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
SetBlBrightness	バックライト輝度設定
GetBlBrightness	バックライト輝度取得

重 要

・ 作成したアプリケーションから dll ファイルを使用するためには、以下に dll ファイルを格納する必要があります。

OS	格納場所
Microsoft® Windows®2000 Microsoft® Windows®XP	・起動プログラムと同一ディレクトリ または ・Windows ディレクトリの System32 例）C:\Winnt\System32

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

・ PsaDevWordWrite と PsaDevWordRead のみを使用する場合は、ドライバオフ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。

5.8.1 サンプルプログラム例

◆ Psa_Ioc.dll を使用する場合

```
’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)
    ・
    ・
’+3.3V の監視
Dim ret As Long
Dim Data As Long
ret = GetCurrentVolt(MONITOR_VOLT_P33, Data)
    ・
    ・
’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
```

◆ Psa_Blc.dll を使用する場合

```
’ドライバオブジェクトの作成
Call InitBlctl
’デバイスハンドルの取得
Dim ret As Long
Dim Hndl As Long
ret = GetBlDrvHandle(Hndl)
    ・
    ・
’バックライトの OFF
Dim ret As Long
ret = SetBlControl(BACKLIGHT_OFF)
    ・
    ・
’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 = CloseBlDrvHandle()
’ドライバオブジェクトの破棄
Call EndBlctl
```

5.9 Visual Basic 用関数仕様詳細

5.9.1 Psa_Ioc.dll 用関数

InitIoctl

呼び出し形式

Declare Sub InitIoctl Lib“Psa_Ioc.dll”()

戻り値

なし

引数

なし

処理概要

CPSA_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“Psa_Ioc.dll”()

戻り値

なし

引数

なし

処理概要

CPSA_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 ”Psa_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になります。

CloseDrvHandle

呼び出し形式

Declare Function CloseDrvHandle Lib ”Psa_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 “ Psa_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

・ 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

Daclare Function GetDrvVersionEx Lib “ Psa_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

・ 機種が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) 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 : 3 (10 進数)

Major : 1 (10 進数)

Minor : 00 (10 進数)

GetMonitorSetup

呼び出し形式

Declare Function GetMonitorSetup Lib “ Psa_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_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_FAN_CPU	CPU FAN
	MONITOR_FAN_POWER	POWER FAN
	MONITOR_FAN_SYSTEM	SYSTEM 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
'+3.3V セットアップ状態取得
ret = GetMonitorSetup(MONITOR_VOLT_P33, Setup)
```

GetVoltParam

呼び出し形式

Declare Function GetVoltParam Lib “ Psa_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_P33	+3.3V 電圧
	MONITOR_VOLT_P50	+5.0V 電圧
	MONITOR_VOLT_P12	+12V 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します。

使用例

```
Dim ret As Long
Dim ULimit As Long
Dim LLimit As Long
'+3.3V 上限下限値取得
ret = GetVoltParam(MONITOR_VOLT_P33, ULimit, LLimit)
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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

Declare Function GetCurrentVolt Lib “ Psa_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_P33	+3.3V 電圧
	MONITOR_VOLT_P50	+5.0V 電圧
	MONITOR_VOLT_P12	+12V 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

電圧値を取得します。

使用例

```
Dim ret As Long
Dim Data As Long
'+3.3V 値取得
ret = GetCurrentVolt(MONITOR_VOLT_P33, Data)
```

MEMO

・ 関数から取得されたデータは mV(ミリボルト) 単位になっているため V(ボルト) 単位で使用するとき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

GetFanParam

呼び出し形式

Declare Function GetFanParam Lib “ Psa_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
	MONITOR_FAN_SYSTEM	SYSTEM 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 下限回転数を取得します。

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 “Psa_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
	MONITOR_FAN_SYSTEM	SYSTEM 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

現在の FAN 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)
```

SetWdtCounter

呼び出し形式

Declare Function SetWdtCounter Lib “ Psa_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
Dim Counter As Long
' ウォッチドッグタイマのカウント値を 10 秒に設定
Counter = 10
ret = SetWdtCounter(Counter)
```

GetWdtCounter

呼び出し形式

Declare Function GetWdtCounter Lib “ Psa_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)
```

StartWdt

呼び出し形式

Declare Function StartWdt Lib ”Psa_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 “Psa_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 “Psa_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

・ 1度、StartWdt をコールし、カウントダウンを開始していないとエラーになります。Time out が発生したあとはエラーをクリアして再度 StartWdt をコールし、カウントダウンを開始していないとエラーになります。

GetWdtStatus

呼び出し形式

Declare Function GetWdtStatus Lib “ Psa_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)
```

GetEvent

呼び出し形式

Declare Function GetEvent Lib “ Psa_Ioc.dll ”(ByVal Selector As Long, ByRef RasEvent 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_P33	+3.3V 電圧
	EVENT_VOLT_P50	+5.0V 電圧
	EVENT_VOLT_P12	+12V 電圧
	EVENT_VOLT_M12	-12V 電圧
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_FAN_SYSTEM	SYSTEM FAN
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
	EVENT_BACKLIGHT	バックライト管切れ
RasEvent 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 RasEvent As Long
ret = GetEvent(EVENT_VOLT_P33, RasEVENT)
```

ClearEvent

呼び出し形式

Declare Function ClearEvent Lib “Psa_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_P33	+3.3V 電圧
	EVENT_VOLT_P50	+5.0V 電圧
	EVENT_VOLT_P12	+12V 電圧
	EVENT_VOLT_M12	-12V 電圧
	EVENT_FAN_CPU	CPU FAN
	EVENT_FAN_POWER	POWER FAN
	EVENT_FAN_SYSTEM	SYSTEM FAN
	EVENT_WDT_TIMEOUT	ウォッチドッグタイマ
	EVENT_BACKLIGHT	バックライト管切れ

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_VOLT_P33)
```

GetWdtTimeout

呼び出し形式

Declare Function GetWdtTimeout Lib “Psa_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 “Psa_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 “Psa_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 “Psa_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)
```

GetLightblowErr

呼び出し形式

```
Declare Function GetLightblowErr Lib "Psa_Ioc.dll" (ByRef Light As Long) As Long
```

戻り値

0 以外　正常

0　　エラー

引数

Light As Long	バックライト管切れ (参照渡し)
	BACKLIGHT_BLOWOUT バックライト管切れ
	BACKLIGHT_GLOW　正常

処理概要

バックライト管切れ状態を取得します。

使用例

```
Dim ret As Long
Dim Light As Long
ret = GetLightblowErr(Light)
```

5.9.2 Psa_Ras.dll 用関数

PsaDevWordWrite

呼び出し形式

Declare Function PsaDevWordWrite Lib “ Psa_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　　書き込むメモリへのワードアドレス 0 ～ 255 (値渡し)

wData As Long　　書き込むデータ 0 ～ 65535 (値渡し)

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 100 を書き込む

```
Dim ret As Long
ret = PsaDevWordWrite(255, 100)
```

PsaDevWordRead

呼び出し形式

```
Declare Function PsaDevWordRead Lib “Psa_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 0 ～ 255（値渡し）

wData As Long　　読み出すデータ 0 ～ 65535（参照渡し）

処理概要

共有メモリからの読み込みを行います。

使用例

```
' アドレス 255 のデータ読み込み
Dim ret As Long
Dim wData As Long
ret = PsaDevWordRead(255, wData)
```

5.9.3 Psa_Blc.dll 用関数

InitBlctl

呼び出し形式

Declare Sub InitBlctl Lib "Psa_Blc.dll" ()

戻り値

なし

引数

なし

処理概要

CPSA_Blctl オブジェクトを作成します。作成されたオブジェクトは EndBlctrl 関数が呼ばれるまで破棄されません。

使用例

'CPSA_Blctl オブジェクトの作成

CALL InitBlctl

EndBlctl

呼び出し形式

Declare Sub EndBlctl Lib "Psa_Blc.dll" ()

戻り値

なし

引数

なし

処理概要

InitBlctl 関数で作成したオブジェクトを破棄します。

使用例

CALL EndBlctl

GetBlDrvHandle

呼び出し形式

Declare Function GetBlDrvHandle Lib "Psa_Blc.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やり取りをおこな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 = GetBlDrvHandle( hndl )
```

CloseBlDrvHandle

呼び出し形式

Declare Function CloseBlDrvHandle Lib "Psa_Blc.dll" () As Long

戻り値

0 以外　正常

0　　エラー

引数

なし

処理概要

GetBlDrvHandle で取得したハンドルを破棄します。

使用例

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

GetBlDrvVersion

呼び出し形式

```
Declare Function GetBlDrvVersionLib "Psa_Blc.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を取得します。

使用例

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

MEMO

・ 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)

GetBlDrvVersionEx

呼び出し形式

Declare Function GetBlDrvVersionEx Lib "Psa_Blc.dll" (ByRef Product As Long, ByRef Major As Long, ByRef Minor 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

ドライババージョン、機種情報を取得します。

使用例

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

MEMO

・ バージョンが 1.00 で製品が PS-3700A(Eden™ ESP6000 - 667MHz Model)、および PS-3701A(Eden™ ESP6000 - 667MHz Model) の場合、以下のようになります。

Product : 3 (10 進数)

Major : 1 (10 進数)

Minor : 00 (10 進数)

SetBlControl

呼び出し形式

Declare Function SetBlControl Lib "Psa_Blc.dll" (ByVal 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 ON/OFF を設定します。

使用例

```
Dim ret As Long
ret = SetBlcontrol(BACKLIGHT_ON)
```

GetBlControl

呼び出し形式

Declare Function GetBlControl"Psa_Blc.dll"(ByRef BlFlag As Long)

戻り値

0 以外　正常

0　　エラー

引数

BlFlag As Long（参照渡し）

BACKLIGHT_ON　　バックライト ON

BACKLIGHT_OFF　　バックライト OFF

処理概要

バックライト状態を取得します。

使用例

```
Dim ret As Long
Dim BlFlag As Long
ret = GetBlControl(BlFlag)
```

SetBlBrightness

呼び出し形式

Declare Function SetBlBrightness"Psa_Blc.dll"(ByVal BlBright As Long)

戻り値

0 以外　正常

0　　エラー

引数

BlBright As Long (値渡し)

BRIGHT_LEVEL_0	輝度レベル 0 (とても暗い)
BRIGHT_LEVEL_1	輝度レベル 1 (暗い)
BRIGHT_LEVEL_2	輝度レベル 2 (明るい)
BRIGHT_LEVEL_3	輝度レベル 3 (とても明るい)

処理概要

バックライトの輝度を設定する。

使用例

```
Dim ret As Long
ret = SetBlBrightness(BRIGHT_LEVEL_1)
```

GetBlBrightness

呼び出し形式

Declare Function GetBlBrightness"Psa_Blc.dll"(ByRef BlBright As Long)

戻り値

0 以外　正常

0　　エラー

引数

BlBright As Long (参照渡し)

BRIGHT_LEVEL_0	輝度レベル 0 (とても暗い)
BRIGHT_LEVEL_1	輝度レベル 1 (暗い)
BRIGHT_LEVEL_2	輝度レベル 2 (明るい)
BRIGHT_LEVEL_3	輝度レベル 3 (とても明るい)

処理概要

バックライトの輝度を取得します。

使用例

```
Dim ret As Long
Dim BlBright As Long
ret = GetBlBrightness(BlBright)
```