Pro-face
Human Machine Interface

Pro-face

PS Series Type A

(Eden™ ESP6000 - 667MHz Model)

API Reference Manual

Table of Contents

1. Operation Environment

This document explains the use of Dynamic Link Libraries (API-DLL) to access RAS features from an application running on a PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) Series unit.
The API-DLLs explained in this document are Psa_Ioc.dll, Psa_Ras.dll and Psa_Blc.dll.

Psa_Ioc.dll is used by applications to access the following System Monitor/RAS features.

- Driver Version information
- System Monitor feature status
- Read out (Get) various monitoring parameters (voltage, fan)
- System Monitor current data (voltage, fan)
- Watchdog parameters
- Alarm processing
- Event handling

Psa_Ras.dll is used to access the Remote RAS feature’s Common Memory API-DLL.Psa_Ras.dll is used by applications to access the following features.

- Reads from common memory
- Writes to common memory

Psa_Blc.dll is used to controll the Backlight feature’s APl -DLL.
Psa_Blc.dll is used by applications to access the following features.

- Controll the Backlight
- Controll the Backlight Brightness

Each dll files are placed in the following locations.

dll file	Location
Psa_Ioc.dll Psa_Ras.dll Psa_Blc.dll	• Windows directory's System32 folder Ex.) C:\Winnt\System32

2. Compatible Operating Systems and Languages

The API-DLLs are compatible with the following OS types and languages.

OS	Languages
Microsoft® Windows© 2000 Microsoft® Windows® XP	Microsoft® Visual C Microsoft® Visual C++ Microsoft® Visual Basic

NOTE

- Visual C++ .NET or Visual Basic .NET cannot be used.

3. Required Files

3.1 Psa_Ioc.dll Files

The following files are required when using Psa_Ioc.dll. Each language requires its own set of files.

Language	File Name	Description
Visual C	Psa_Iocif.h	Driver interface definition "include" file
	Psa_Ioc.lib	Library definition file
Visual C++	Psa_Iocif.h	Driver interface definition "include" file
	Psa_Iocall.h	CPSA_Iocall class definition "include" file[1]
	Psa_Ioctl.h	CPSA_Ioctl class definition "include" file
	Psa_Ioc.lib	Library definition file
Visual Basic	Psa_Ioc.bas	Driver interface definition file
	Psa_Ioc.lib	Library definition file

#include header files should be included in the following order. Psa_Iocall.h is automatically included, and does not need to be directly designated.

#include Psa_Iocif.h

#include Psa_Ioctl.h

Psa_Blc.dll is automatically included.

#include Psa_Blcif.h

#include Psa_Blctl.h

NOTE

- These files are placed in the following folder.
  C:\Proface\Psaapi

3.2 Psa_Ras.dll Files

The following files are required when using Psa_Ras.dlls. Each language requires its own set of files.

Language	File Name	Description
Visual C	Psa_Ras.h	Common Memory Read/Write driver interface definition "include" file
	Psa_Ras.lib	Common Memory Read/Write Library definition file
Visual C++	Psa_Ras.h	Common Memory Read/Write Driver interface definition "include" file
	Psa_Ras.lib	Common Memory Read/Write Library definition file

NOTE

- These files are placed in the following folder.
  C:\Proface\Psaapi

3.3 Psa_Blc.dll Files

The following files are required when using Psa_Blc.dll. Each language requires its own set of files.

Language	File Name	Description
Visual C	Psa_Blcif.h	Backlight driver interface definition file
	Psa_Blc.lib	Backlight library definition file
Visual C++	Psa_Blcif.h	Backlight driver interface definition file
	Psa_Blcall.h	CPSA_Blacall class dedefinition "include" file
	Psa_Blctl.h	CPSA_Bloctal class dedefinition "include" file
	Psa_Blc.lib	Backlight library definition file
Visual Basic	Psa_Blc.bas	Backlight driver interface definition file
	Psa_Blc.lib	Backlight library definition file

NOTE

- These files are placed in the following folder.
  C:\Proface\Psaapi

4. Class Contents

4.1 CPSA_Ioctl Class

This class is used to set the parameters for device driver access using CPSA_Ioctl class.

Key Word	Type	Variable name	Description
public	HANDLE	m_Drvhandle	Device driver handle

4.2 CPSA_Iocall Class

This uses the parameters set in CPSA_Ioctl and, calls up DeviceIoControl (Driver Access function).

However, since this class succeeds CPSA_Ioctl, it cannot be used directly.

Key Word	Type	Variable Name	Description
public	HANDLE	m_h	Device driver handle
public	LONG	m_long	Control code for action to perform
public	void *	m_ibp	Input data buffer address
public	ULONG	m_ibsize	Input data buffer size
public	void *	m_obp	Output data buffer address
public	ULONG	m_obsize	Output data buffer size
public	DWORD	m_retsize	Address for actual no. of output bytes
public	LPOVER-LAPPED	m_ovlp	Address of overlap design

4.3 CPSA_Blctl Class

This class is used to set the parameters for device driver access using CPSA_Blctl class.

Key Word	Type	Variable Name	Description
public	HANDLE	m_h	Device driver handle

4.4 CPSA_Blcall Class

This uses the parameters set in CPSA_Blctl and, calls up DeviceIoControl (Driver Access function).

However, since this class succeeds CPSA_Bloctl, it cannot be used directly.

Key Word	Type	Variable Name	Description
public	HANDLE	m_h	Device driver handle
public	LONG	m_long	Control code for action to perform
public	void *	m_ibp	Input data buffer address
public	ULONG	m_ibsize	Input data buffer size
public	void *	m_obp	Output data buffer address
public	ULONG	m_obsize	Output data buffer size
public	DWORD	m_retsize	Address for actual no. of output bytes
public	LPOVER-LAPPED	m_ovlp	Address of overlap design

5. Function Specifications

5.1 Visual C Functions

5.1.1 Psa_Ioc.dll Functions

Function Name	Description
InitIoctl	Creates the CPSA_Ioctl object
EndIoctl	Destroys the CPSA_Ioctl object
GetDrvHandle	Gets the driver handle
CloseDrvHandle	Destroys the driver handle
GetDrvVersion	Gets the driver version
GetDrvVersionEx	Gets the hardware type and driver version
GetMonitorSetup	Gets the monitoring enabled/disabled setting
GetVoltParam	Gets the voltage monitoring parameter
GetCurrentVolt	Gets the current voltage value
GetFanParam	Gets the fan monitoring parameter
GetCurrentFan	Gets the current fan value
SetWdtCounter	Sets the watchdog timer counter value
GetWdtCounter	Gets the watchdog timer counter value
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog timer operation status
GetEvent	Gets the error event
ClearEvent	Clear the error event
GetWdtTimeout	Gets the timeout status of the watchdog timer
ClearWdtTimeout	Clear the timeout status of the watchdog timer
SetWdtResetMask	Sets the Reset Mask of the watchdog timer
GetWdtResetMask	Gets the Reset Mask of the watchdog timer
GetLightblowErr	Gets the Backlight blowout error

5.1.2 Psa_Ras.dll Functions

Function Name	Description
PsaDevWordWrite	Writes to common memory
PsaDevWordRead	Reads from common memory

5.1.3 Psa_Blc.dll Functions

Function Name	Description
InitBlctl	Creates the CPSA_Blctl object
EndBlctl	Destroys the CPSA_Blctl object
GetBlDrvHandle	Gets the driver handle
ClearBlDrvHandle	Destroys the driver handle
GetBlDrvVersion	Gets the driver version
GetBlDrvVersionEx	Gets the hardware version and driver version
SetBlControl	Sets the backlight status
GetBlControl	Gets the backlight status
SetBlBrightness	Sets the backlight blightness
GetBlBrightness	Gets the backlight blightness

IMPORTANT

• When using the dll file from a created application, place the dll files in one of the following locations.

OS	Location
Windows®2000 Windows®XP	• the same directory as the start up program or • Windows directory's System32 folder Ex.) C:\Winnt\System32

5.2 Visual C Programing Cautions

When using API-DLLs, it is important to first create the driver object and get the device handle.

When you finish using the API-DLL, you will need to destroy both the device handle and the driver object.

Refer to the following example when developing your programs.

NOTE

- Only when using PsaDevWordWrite and PsaDevWordRead, it is not necessary to create/destroy the driver object and the device handle.

5.2.1 Sample Program

◆When using a Psa_Ioc.dll

```
// Variable language
BOOL bRet;
int iData;
int iRet;
HANDLE hDrv;
// Create the driver object and get the device handle
// Create CPSA_Ioctl object
InitIoctl();
iRet = GetDrvHandle(&hDrv);
        .
        .
        .
// Watch the +3.3V
bRet = GetCurrentVolt(MONITOR_VOLT_P33, &iData);
        .
        .
// Application shut-down processing
// Destroy both the device handle and the driver object
bRet = CloseDrvHandle();
EndIoctl();
```

◆When using a Psa_Blc.dll

```
// Variable language
BOOL bRet;
int iRet;
HANDLE hDrv;
// Create the driver object and get the device handle
// Create CPSA_Blctl object
InitBlctl();
iRet = GetBlDrvHandle(&hDrv);
        .
        .
        .
// Set the backlight OFF
bRet = SetBlControl(BACKLIGHT_OFF);
        .
        .
// Application shut-down processing
// Destroy both the device handle and the driver object
bRet = CloseBlDrvHandle();
EndBlctl();
```

5.3 Visual C Function Specifications (Details)

5.3.1 Psa_Ioc.dll Functions

InitIoctl

Call Format

void WINAPI InitIoctl(void)

Return Value

None

Arguments

None

Processing

Cleates a CPSA_Ioctl object. The object is not destroyed until the EndIoctl function is called.

Example

InitIoctl();

EndIoctl

Call Format

void WINAPI EndIoctl(void)

Return Value

None

Arguments

None

Processing

Destroys CPSA_Ioctl object created using the InitIoctl function.

Example

EndIoctl();

GetDrvHandle

Call Format

int WINAPI GetDrvHandle(HANDLE *pHndl)

Return Value

0	Normal
1	Error

Arguments

(I/O) HANDLE *pHndl	Pointer to the device driver handle

Processing

Gets the device driver handle to communicate with the device driver.

Example

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

NOTE

- An error will result if the system monitor/RAS device driver is not operating.

CloseDrvHandle

Call Format

BOOL WINAPI CloseDrvHandle(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Destroys the device driver handle created using the GetDrvHandle function.

Example

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

GetDrvVersion

Call Format

BOOL WINAPI GetDrvVersion(int *pMajor, int *pMinor)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMajor	Pointer to version information
(I/O) int *pMinor	Pointer to version information

Processing

Gets the driver's version information

Example

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

NOTE

- If the version is 1.00, then you will get

  Major : 1 (decimal)

  Minor : 00 (decimal).

GetDrvVersionEx

Call Format

BOOL WINAPI GetDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pProduct	Pointer to hardware type
(I/O) int *pMajor	Pointer to version information
(I/O) int *pMinor	Pointer to version information

Processing

Gets the hardware type and driver version.

Example

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

NOTE

- If the H/W type is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will get

  Product : 3 (decimal)

  Major : 1 (decimal)

  Minor : 00 (decimal)

GetMonitorSetup

Call Format

BOOL WINAPI GetMonitorSetup(int Selector, int *pSetup)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pSetup	Pointer to Data	
	0 : Disable	
	1 : Enable	

Processing

Gets the current monitoring status (enabled/disabled).

Example

```
BOOL ret;
int Setup;
// Gets the +3.3V setup status.
ret = GetMonitorSetup(MONITOR_VOLT_P33, &Setup);
```

GetVoltParam

Call Format

BOOL WINAPI GetVoltParam(int Selector, int *pULimit, int *pLLimit)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
(I/O) int *pULimit	Pointer to upper-limit voltage value (Unit: mV)	
(I/O) int *pLLimit	Pointer to lower-limit voltage value (Unit: mV)	

Processing

Gets the voltage monitoring parameter.

Example

```
BOOL ret;
int ULimit, LLimit;
// Gets the upper and lower-limit values of the +3.3V.
ret = GetVoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

NOTE

- Since the data taken from this function is shown in mV units, thefollowing conversion is needed for use in (Volt) units:

  Data in Volt unit = Data in mV unit /1000

GetCurrentVolt

Call Format

BOOL WINAPI GetCurrentVolt(int Selector, int *pData)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
(I/O) int *pData	Pointer to the voltage value (unit: mV)	

Processing

Gets the current voltage value.

Example

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

NOTE

- Since the data taken from this function is shown in mV units, the following conversion is needed for use in (Volt) units:
  Data in Volt unit = Data in mV unit /1000

GetFanParam

Call Format

BOOL WINAPI GetFanParam(int Selector, int *pLLimit)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pLLimit	Pointer to the lower-limit fan rotation speed (Unit: RPM)	
	(RPM : Revolutions Per Minute)	

Processing

Gets the lower-limit fan rotation speed.

Example

```
BOOL ret;
int LLimit;
// Gets the lower-limit CPU fan rotation speed.
ret = GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

GetCurrentFan

Call Format

```
BOOL WINAPI GetCurrentFan(int Selector, int *pData)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameter	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pData	Pointer to the fan rotation speed (Unit: RPM)	
	(RPM : Revolutions Per Minute)	

Processing

Gets the current fan rotational speed.

Example

```
BOOL ret;
int Data;
// Gets the CPU fan rotational speed.
ret = GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

SetWdtCounter

Call Format

BOOL WINAPI SetWdtCounter(int Counter)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Counter	Sets to the watchdog timer’s initial counter value (5 to 255)(Unit:Seconds)

Processing

Sets the current watchdog timer’s initial counter value.

Example

```
BOOL ret;
int Counter;
// Sets the current watchdog timer’s initial counter value to 10 sec.
Counter = 10;
ret = SetWdtCounter(Counter);
```

GetWdtCounter

Call Format

```
BOOL WINAPI GetWdtCounter(int *pCounter)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int *pCounter	Pointer to the watchdog timer’s initial counter value (Unit:Seconds).

Processing

Gets the current watchdog timer’s initial counter value.

Example

```
BOOL ret;
int Counter;
// Gets the current watchdog timer’s initial counter value.
ret = GetWdtCounter(&Counter);
```

StartWdt

Call Format

```
BOOL WINAPI StartWdt(void)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Starts watchdog timer count down.

Example

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

StopWdt

Call Format

BOOL WINAPI StopWdt(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Stops watchdog timer.

Example

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

RestartWdt

Call Format

BOOL WINAPI RestartWdt(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

This feature resets the Watchdog timer to its initial value, and restarts the countdown.

Example

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

NOTE

- If StartWdt is called but the countdown has not yet started, an error will occur.
  Also, after a Timeout has occurred, if the error is cleared and StartWdt is called, but the count-down has not yet started , an error will occur.

GetWdtStatus

Call Format

```
BOOL WINAPI GetWdtStatus(int *pRunFlag)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pRunFlag	Pointer to the watchdog timer operation status	
	WATCHDOG_STOP	Stopped
	WATCHDOG_COUNTDOWN	Countdown in progress

Processing

Gets the watchdog timer’s operation status.

Example

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

GetEvent

Call Format

BOOL WINAPI GetEvent(int Selector, int *pRasEvent)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_FAN_SYSTEM	SYSTEM fan
	EVENT_WDT_TIMEOUT	Watchdog timer
	EVENT_BACKLIGHT	Backlight blowout
(I) int *pRasEvent	Pointer to Error Event Information	
	ERROR_EVENT_ON With error event	
	ERROR_EVENT_OFF Without error event	

Processing

Gets the error event.

Example

```
BOOL ret;
int RasEvent;
ret = GetEvent(EVENT_VOLT-P33, &RasEvent);
```

ClearEvent

Call Format

BOOL WINAPI ClearEvent(int Selector)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_FAN_SYSTEM	SYSTEM fan
	EVENT_WDT_TIMEOUT	Watchdog timer
	EVENT_BACKLIGHT	Backlight blowout

Processing

Cancels the error event.

Example

```
BOOL ret;
ret = ClearEvent(EVENT_VOLT-P33);
```

GetWdtTimeout

Call Format

BOOL WINAPI GetWdtTimeout(int *pTimebuf)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int *pTimebuf	Pointer to Watchdog Timeout Status	
	TIMEOUT_OK	Timeout has not occurred
	TIMEOUT_ERR	Timeout has occurred

Processing

Gets watchdog timeout status.

Example

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

ClearWdtTimeout

Call Format

BOOL WINAPI ClearWdtTimeout(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Clears the watchdog timeout status.

Example

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

SetWdtResetMask

Call Format

```
BOOL WINAPI SetWdtResetMask (int Mask)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int Mask Masking Information

MASK_OFF	Masking disabled
MASK_ON	Masking enabled

Processing

Sets the H/W reset mask used at Watchdog Timer timeout.

Example

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

GetWdtResetMask

Call Format

```
BOOL WINAPI GetWdtResetMask(int *pMask)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMask Pointer to Masking Information

MASK_OFF	Masking disabled
MASK_ON	Masking enabled

Processing

Gets the H/W reset mask data used at Watchdog Timer timeout.

Example

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

GetLightblowErr

Call Format

BOOL WINAPI GetLightblowErr(int *pLight)

Return Value

TRUE	Normal
FALSE	Error

Argument

(I/O) int *pLight	Pointer to Backlight blowout Status	
	BACKLIGHT_BLOWOUT	Backlight blowout
	BACKLIGHT_GLOW	Normal

Processing

Gets the Backlight blowout Status

Example1

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

5.3.2 Psa_Ras.dll Functions

PsaDevWordWrite

Call Format

```
long PsaDevWordWrite(long Addr, long wData)
```

Return Value

0	Normal
Other than 0	Error

Arguments

(I) long Addr	Write memory word Address (0 to 255)
(I) long wData	Write data (0 to 65535)

Processing

Writes to common memory.

Example

```
// Writes data 255 to address 100.
long ret;
ret = PsaDevWordWrite(255, 100);
```

PsaDevWordRead

Call Format

```
long PsaDevWordRead(long Addr, long *wData)
```

Return Value

0	Normal
Other than 0	Error

Arguments

(I) long Addr	Read memory word Address (0 to 255)
(I/O) long *wData	Read data (0 to 65535)

Processing

Reads from common memory.

Example

```
// Reads address 255's data.
long ret;
long wData;
ret = PsaDevWordRead(255, &wData);
```

5.3.3 Psa_Blc.dll Functions

InitBlctl

Call Format

void WINAPI InitBlctl(void)

Return Value

None

Arguments

None

Processing

Creates CPSA_Blctl object. The object is not destroyed until the EndBlctrl function is called.

Example

//Creates CPSA_Blctl object.

InitBlctl()

EndBlctl

Call Format

void WINAPI EndBlctl()

Return Value

None

Arguments

None

Processing

Destroys the object created using the InitBlctl function.

Example

EndBlctl()

GetBlDrvHandle

Call Format

int WINAPI GetBlDrvHandle(HANDLE *pHndl)

Return Value

0	Normal
Other than 0	Error

Arguments

(I/O) HANDLE *pHndl	Pointer to Device Driver Handle

Processing

Gets the Device Driver Handle to communicate with the device handle

Example

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

CloseBlDrvHandle

Call Format

BOOL WINAPI CloseBlDrvHandle()

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Destroys the handle in GetBlDrvHandle

Example

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

GetBlDrvVersion

Call Format

BOOL WINAPI GetBlDrvVersion(int *pMajor, int *pMinor)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMajor	Pointer to version information
(I/O) int *pMinor	Pointer to version information

Processing

Gets the Driver Version

Example

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

NOTE

- If the version is 1.00, then you will get

  Major : 1 (decimal)

  Minor : 00 (decimal)

GetBlDrvVersionEx

Call Format

BOOL WINAPI GetBlDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pProduct	Pointer to the product information
(I/O) int *pMajor	Pointer to the version information
(I/O) int *pMinor	Pointer to the version information

Processing

Gets the driver version and product information

Example

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

NOTE

- If the product is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will get

  Product : 3 (decimal)

  Major : 1 (decimal)

  Minor : 00 (decimal)

SetBlControl

Call Format

BOOL WINAPI SetBlControl(int BlFlag)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int BlFlag	Backlight Setting parameter	
	BACKLIGHT_OFF	Backlight OFF
	BACKLIGHT_ON	Backlight ON

Processing

Sets the Backlight ON/OFF setting.

Example

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

GetBlControl

Call Format

BOOL WINAPI GetBlControl(int *pBlFlag)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pBlFlag	Pointer to the backlight status	
	BACKLIGHT_ON	Backlight ON
	BACKLIGHT_OFF	Backlight OFF

Processing

Gets the Backlight Status.

Example

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

SetBlBrightness

Call Format

BOOL WINAPI SetBlBrightness(int BlBright)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int BlBright	Backlight Brightness	
	BRIGHT_LEVEL_0	Brightness level 0 (very dark)
	BRIGHT_LEVEL_1	Brightness level 1 (dark)
	BRIGHT_LEVEL_2	Brightness level 2 (bright)
	BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Sets the Backlight Brightness.

Example

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

GetBlBrightness

Call Format

```
BOOL WINAPI GetBlBrightness(int *pBlBright)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pBlBright	Pointer to the backlight brightness	
	BRIGHT_LEVEL_0	Brightness level 0 (very dark)
	BRIGHT_LEVEL_1	Brightness level 1 (dark)
	BRIGHT_LEVEL_2	Brightness level 2 (bright)
	BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Gets the backlight brightness.

Example

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

5.4 Visual C++ Functions

5.4.1 Psa_Ioc.dll Functions

Function Name	Description
GetDrvHandle	Gets the driver handle
CloseDrvHandle	Destroys the driver handle
GetDrvVersion	Gets the driver version
GetDrvVersionEx	Gets the hardware type and driver version
GetMonitorSetup	Gets the monitoring enabled/disabled setting
GetVoltParam	Gets the voltage monitoring parameter
GetCurrentVolt	Gets the current voltage value
GetFanParam	Gets the fan monitoring parameter
GetCurrentFan	Gets the current fan value
SetWdtCounter	Sets the watchdog timer counter value
GetWdtCounter	Gets the watchdog timer counter value
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog timer operation status
GetEvent	Gets the error event
ClearEvent	Clears the error event
GetWdtTimeout	Gets watchdog timeout status
ClearWdtTimeout	Clears the watchdog timeout status
SetWdtResetMask	Sets the ResetMask of the watchdog timer
GetWdtResetMask	Gets the ResetMask of the watchdog timer
GetLightblowErr	Gets the Backlight blowout error

5.4.2 Psa_Ras.dll Functions

Function Name	Description
PsaDevWordWrite	Writes to common memory
PsaDevWordRead	Reads from common memory

5.4.3 Psa_Blc.dll Functions

Function Name	Description
GetBlDrvHandle	Gets the driver handle
CloseBlDrvHandle	Destroys the driver handle
GetBlDrvVersion	Gets the driver version
GetBlDrvVersionEx	Gets the hardware version and driver version
SetBlControl	Sets the backlight status
GetBlControl	Gets the backlight status
SetBlBrightness	Sets the backlight blightness
GetBlBrightness	Gets the backlight blightness

IMPORTANT

- When using the dll file from a created application, place the dll files in one of the following locations.

OS	Location
Windows®2000 Windows®XP	• the same directory as the start up program or • Windows directory's System32 folder Ex.) C:\Winnt\System32

5.5 Visual C++ Programing Cautions

When using API-DLLs, it is important to first get the device handle. When you finish using the API-DLL, you will need to destroy the device handle object. Refer to the following example when developing your programs.

NOTE

- Only when using PsaDevWordWrite and PsaDevWordRead, it is not necessary to create/destroy the driver object and the device handle.

5.5.1 Sample Program

◆When using a Psa_Ioc.dll

```
// Variable language
BOOL bRet;
int iData;
int iRet;
// Get the device handle
CPSA_Ioctl m_Ioc;
iRet = m_Ioc.GetDrvHandle();
        .
        .
// Watch the +3.3V.
bRet = m_Ioc.GetCurrentVolt(MONITOR_VOLT_P33, &iData)
        .
        .
// Destroy the device handle
bRet = m_Ioc.CloseDrvHandle();
```

◆Example of using a Psa_Blc.dll

```
// Variable language
BOOL bRet;
int iRet;
// Get the device handle
CPSA_Blctl m_Blc;
iRet = m_Blc.GetBlDrvHandle();
        .
        .
// Set the backlight OFF
bRet = m_Blc.SetBlControl(BACKLIGHT_OFF);
        .
        .
// Destroy the device handle
bRet = m_Blc.CloseBlDrvHandle();
```

5.6 Visual C++ Function Specifications (Details)

5.6.1 Psa_Ioc.dll Functions

GetDrvHandle

Call Format

int GetDrvHandle(HANDLE *pHndl) or int GetDrvHandle()

Return Value

0	Normal
1	Error

Arguments

(I/O) HANDLE *pHndl	Pointer to the device driver handle

Processing

Gets the device driver handle to communicate with the device driver.

Example1

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

Example2

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

NOTE

• An error occurs if the System Monitor/RAS Device Driver is not running.

CloseDrvHandle

Call Format

```
BOOL CloseDrvHandle(void)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Destroys the device driver handle created using the GetDrvHandle function.

Example1

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

Example2

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

GetDrvVersion

Call Format

```
BOOL GetDrvVersion(int *pMajor, int *pMinor)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMajor	Pointer to version information
(I/O) int *pMinor	Pointer to version information

Processing

Gets the driver’s version information.

Example1

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

Example2

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

NOTE

- If the version is 1.00, then you will get
  Major : 1 (decimal)
  Minor : 00 (decimal)

GetDrvVersionEx

Call Format

BOOL GetDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pProduct	Pointer to Hardware type
(I/O) int *pMajor	Pointer to version information
(I/O) int *pMinor	Pointer to version information

Processing

Gets the hardware type and driver's version information.

Example1

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

Example2

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

NOTE

- If the H/W type is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will get

  Product : 3 (decimal)

  Major : 1 (decimal)

  Minor : 00 (decimal)

GetMonitorSetup

Call Format

BOOL GetMonitorSetup(int Selector, int *pSetup)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pSetup	Pointer to gotten Data	
	0 : Disable	
	1 : Enable	

Processing

Gets the current monitoring enabled/disabled status.

Example1

```
BOOL ret;
int Setup;
// Gets the +3.3V setup status.
ret = ::GetMonitorSetup(MONITOR_VOLT_CPU, &Setup);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Setup;
// Gets the +3.3V setup status.
ret = m_Ioctl.GetMonitorSetup(MONITOR_VOLT_P33, &Setup);
```

GetVoltParam

Call Format

BOOL GetVoltParam(int Selector, int *pULimit, int *pLLimit)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
(I/O) int *pULimit	Pointer to upper-limit voltage value (Unit: mV)	
(I/O) int *pLLimit	Pointer to lower-limit voltage value (Unit: mV)	

Processing

Gets the voltage monitoring parameter.

Example1

```
BOOL ret;
int ULimit, LLimit;
// Gets the upper and lower-limit values of the +3.3V.
ret = ::GetVoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int ULimit, LLimit;
// Gets the upper and lower-limit values of the +3.3V.
ret = m_Ioctl.GetVoltParam(MONITOR_VOLT_P33, &ULimit, &LLimit);
```

NOTE

- Since the data taken from this function is shown in mV units, the following conversion is needed for use in (Volt) units :

  Data in Volt unit = Data in mV unit / 1000

GetCurrentVolt

Call Format

```
BOOL GetCurrentVolt(int Selector, int *pData)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
(I/O) int *pData	Pointer to the voltage value (Unit: mV)	

Processing

Gets the current voltage value.

Example1

```
BOOL ret;
int Data;
// Gets the +3.3V value.
ret = ::GetCurrentVolt(MONITOR_VOLT_P33, &Data);
```

Example2

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

NOTE

- Since the data taken from this function is shown in mV units, the following conversion is needed for use in (Volt) units :

  Data in Volt unit = Data in mV unit / 1000

GetFanParam

Call Format

```
BOOL GetFanParam(int Selector, int *pLLimit)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pLLimit	Pointer to the lower-limit fan rotation speed (Unit: RPM) (RPM : Revolutions Per Minute)	

Processing

Gets the lower-limit fan rotation speed.

Example1

```
BOOL ret;
int LLimit;
// Gets the lower-limit CPU fan rotational speed.
ret = ::GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int LLimit;
// Gets the lower-limit CPU fan rotational speed.
ret = m_Ioctl.GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

GetCurrentFan

Call Format

BOOL GetCurrentFan(int Selector, int *pData)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
(I/O) int *pData	Pointer to the fan rotation speed (Unit: RPM)	
	(RPM : Revolutions Per Minute)	

Processing

Gets the current fan rotation speed.

Example1

```
BOOL ret;
int Data;
// Gets the CPU fan rotation speed.
ret = ::GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Data;
// Gets the CPU fan rotation speed.
ret = m_Ioctl.GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

SetWdtCounter

Call Format

```
BOOL SetWdtCounter(int Counter)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Counter	Sets to the watchdog timer’s initial counter value. (5 to255)(Unit: Seconds)

Processing

Sets the watchdog timer’s initial counter value.

Example1

```
BOOL ret;
int Counter;
// Sets the watchdog timers initial counter value to 10.
Counter = 10;
ret = ::SetWdtCounter(Counter);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Counter;
// Sets the watchdog timer’s initial counter value to 10.
Counter = 10;
ret = m_Ioctl.SetWdtCounter(Counter);
```

GetWdtCounter

Call Format

```
BOOL GetWdtCounter(int *pCounter)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int *pCounter	Pointer to the watchdog timer’s initial counter value. (Unit: Seconds)

Processing

Gets the current watchdog timer’s initial counter value.

Example1

```
BOOL ret;
int Counter;
// Gets the current watchdog timer’s initial counter value.
ret = ::GetWdtCounter(&Counter);
```

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int Counter;
// Gets the current watchdog timer’s initial counter value.
ret = m_Ioctl.GetWdtCounter(&Counter);
```

StartWdt

Call Format

BOOL StartWdt(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Starts watchdog timer countdown.

Example1

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

Example2

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

StopWdt

Call Format

BOOL StopWdt(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Stops watchdog timer.

Example1

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

Example2

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

RestartWdt

Call Format

BOOL RestartWdt(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

This feature resets the Watchdog timer to its initial value, and restarts the countdown.

Example1

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

Example2

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

NOTE

- If StartWdt is called but the countdown has not yet started, an error will occur.
  Also, after a Timeout has occurred, if the error is cleared and StartWdt is called, but the countdown has not yet started, an error will occur.

GetWdtStatus

Call Format

```
BOOL GetWdtStatus(int *pRunFlag)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pRunFlag	Pointer to Watchdog Timer Operation Status.	
	WATCHDOG_STOP	Stopped
	WATCHDOG_COUNTDOWN	Countdown in progress

Processing

Gets the watchdog timer’s operation status.

Example1

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

Example2

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

GetEvent

Call Format

BOOL GetEvent(int Selector, int *pRasEvent)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_FAN_SYSTEM	SYSTEM fan
	EVENT_WDT_TIMEOUT	Watchdog timer
	EVENT_BACKLIGHT	Backlight blowout
(I) int *pRasEvent	Pointer to Event Information	
	ERROR_EVENT_ON	With Event
	ERROR_EVENT_OFF	Without Event

Processing

Gets event information.

Example1

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

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
int RasEvent;
ret = m_Ioctl.GetEvent(EVENT_DIN0, &RasEvent);
```

ClearEvent

Call Format

BOOL ClearEvent(int Selector)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Selector	Parameters	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_WDT_TIMEOUT	Watchdog Timer
	EVENT_BACKLIGHT	Backlight blowout

Processing

Cancels the error event.

Example1

```
BOOL ret;
ret = ::ClearEvent(EVENT_VOLT_P33);
```

Example2

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

GetWdtTimeout

Call Format

BOOL GetWdtTimeout(int *pTimebuf)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int *pTimebuf	Pointer to Watchdog Status	
	TIMEOUT_OK	Timeout has not occurred
	TIMEOUT_ERR	Timeout has occurred

Timeout has occurred

Processing

Gets watchdog timeout status.

Example1

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

Example2

```
CPSA_Icotl m_Ioctl;
BOOL ret;
int Timebuf;
ret = m_Ioctl.GetWdtTimeout(&Timebuf);
```

ClearWdtTimeout

Call Format

BOOL ClearWdtTimeout(void)

Return Value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Clears the watchdog timeout status.

Example1

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

Example2

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

SetWdtResetMask

Call Format

BOOL SetWdtResetMask (int Mask)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int Mask	Masking Information	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

Processing

Sets Watchdog timer Timeout H/W reset-masking.

Example1

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

Example2

```
CPSA_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.SetWdtResetMask(MASK_ON);
```

GetWdtResetMask

Call Format

```
BOOL GetWdtResetMask(int *pMask)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMask	Pointer to Masking Information	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

Processing

Gets the current Watchdog Timer timeout H/W reset-masking information.

Example1

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

Example2

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

GetLightblowErr

Call Format

BOOL GetLightblowErr(int *pLight)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pLight	Pointer to the backlight blowout	
	BACKLIGHT_BLOWOUT	Backlight blowout
	BACKLIGHT_GLOW	Normal

Normal

Processing

Gets the backlight blowout status

Example 1

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

Example 2

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

5.6.2 Psa_Ras.dll Functions

PsaDevWordWrite

Call Format

long PsaDevWordWrite(long Addr, long wData)

Return Value

0	Normal
Other than 0	Error

Arguments

(I) long Addr	Write memory word address (0 to 255)
(I) long wData	Write data (0 to 65535)

Processing

Writes to common memory.

Example

```
// Writes data 255 to address 100.
long ret;
ret = PsaDevWordWrite(255, 100);
```

PsaDevWordRead

Call Format

long PsaDevWordRead(long Addr, long *wData)

Return Value

0	Normal
Other than 0	Error

Arguments

(I) long Addr	Write memory word address (0 to 255)
(I) long *wData	Pointer to read data (0 to 65535)

Processing

Reads from common memory.

Example

```
// Reads address 255's data.
long ret;
long wData;
ret = PsaDevWordRead(255, &wData);
```

5.6.3 Psa_Blc.dll Functions

GetBlDrvHandle

Call Format

int GetBlDrvHandle(void) or int GetBlDrvHandle(HANDLE *pHndl)

Return Value

0	Normal
Other than 0	Error

Arguments

(I/O) HANDLE *pHndl	Pointer to the device driver handle

Processing

Gets the device driver handle to communicate with device driver.

Example 1

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

Example 2

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

CloseBlDrvHandle

Call Format

BOOL CloseBlDrvHandle()

Return value

TRUE	Normal
FALSE	Error

Arguments

None

Processing

Destorys the handle in GetBlDrvHandle.

Example 1

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

Example 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.CloseBlDrvHandle();
```

GetBlDrvVersion

Call Format

```
BOOL GetBlDrvVersion(int *pMajor, int *pMinor)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pMajor	Pointer to the version information
(I/O) int *pMinor	Pointer to the version information

Processing

Gets the driver version.

Example 1

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

Example 2

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

NOTE

- If the version is 1.00, then you will get
  Major : 1 (decimal)
  Minor : 00 (decimal)

GetBlDrvVersionEx

Call Format

```
BOOL GetBlDrvVersionEx(int *pProduct, int *pMajor, int *pMinor)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pProduct	Pointer to the product information
(I/O) int *pMajor	Pointer to the version information
(I/O) int *pMinor	Pointer to the version information

Processing

Gets the driver version and product information.

Example 1

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

Example 2

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

NOTE

- If the product is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will get

  Product : 3 (decimal)

  Major : 1 (decimal)

  Minor : 00 (decimal)

SetBlControl

Call Format

```
BOOL SetBlControl(int BlFlag)
```

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I) int BlFlag	Backlight Setting parameter	
	BACKLIGHT_OFF	Backlight OFF
	BACKLIGHT_ON	Backlight ON

Processing

Sets the Backlight ON/OFF setting.

Example 1

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

Example 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.SetBlcontrol(BACKLIGHT_ON);
```

GetBlControl

Call Format

BOOL GetBlControl(int *pBlFlag)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pBlFlag	Pointer to the backlight status	
	BACKLIGHT_ON	Backlight ON
	BACKLIGHT_OFF	Backlight OFF

Processing

Gets the Backlight Status.

Example 1

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

Example 2

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

SetBlBrightness

Call Format

BOOL SetBlControl(int BlBright)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int BlBright	Backlight Brightness	
	BRIGHT_LEVEL_0	Brightness level 0 (very dark)
	BRIGHT_LEVEL_1	Brightness level 1 (dark)
	BRIGHT_LEVEL_2	Brightness level 2 (bright)
	BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Sets the backlight brightness.

Example 1

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

Example 2

```
CPSA_Blctl m_Blctl;
BOOL ret;
ret = m_Blctl.SetBlBrightness(BRIGHT_LEBEL_1);
```

GetBlBrightness

Call Format

BOOL GetBlControl(int *pBlBright)

Return Value

TRUE	Normal
FALSE	Error

Arguments

(I/O) int *pBlBright Pointer to the backlight brightness

BRIGHT_LEVEL_0	Brightness level 0 (very dark)
BRIGHT_LEVEL_1	Brightness level 1 (dark)
BRIGHT_LEVEL_2	Brightness level 2 (bright)
BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Gets the backlight brightness.

Example

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

Example

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

5.7 Visual Basic Functions

5.7.1 Psa_Ioc.dll Functions

Function Name	Description
InitIoctl	Creates a CPSA_Ioctl object
EndIoctl	Destroys a CPSA_Ioctl object
GetDrvHandle	Gets the driver handle
CloseDrvHandle	Destroys the driver handle
GetDrvVersion	Gets the driver version
GetDrvVersionEx	Gets the hardware type and driver version
GetMonitorSetup	Gets the enabled/disabled monitor settings
GetVoltParam	Gets the voltage monitoring parameters
GetCurrentVolt	Gets the current value of the voltage
GetFanParam	Gets the parameters for monitoring the FAN
GetCurrentFan	Gets the current value of the FAN
GetTempParam	Gets the parameters for monitoring the temperature
GetCurrentTemp	Gets the current value of the temperature
SetWdtCounter	Sets the value of the watchdog timer counter value
GetWdtCounter	Gets the watchdog timer counter value
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog status
GetEvent	Gets an error event
ClearEvent	Clears an error event
GetWdtTimeout	Gets the time-out status of the watchdog timer
ClearWdtTimeout	Clears the time-out status of the watchdog timer
SetWdtResetMask	Sets the reset mask of the watchdog timer
GetWdtResetMask	Gets the reset mask of the watchdog timer
GetLightblowErr	Gets the Backlight blowout error

5.7.2 Psa_Ras.dll Functions

Function Name	Description
PsaDevWordWrite	Writes to common memory
PsaDevWordRead	Reads from common memory

5.7.3 Psa-Blc.dll Functions

Function Name	Description
InitBlctl	Creates the CPSA_Blctl object
EndBlctl	Destroys the CPSA_Blctl object
GetBlDrvHandle	Gets the driver handle
CloseBlDrvHandle	Destroys the driver handle
GetBlDrvVersion	Gets the driver version
GetBlDrvVersionEx	Gets the hardware version and driver version
SetBlControl	Sets the backlight status
GetBlControl	Gets the backlight status
SetBlBrightness	Sets the backlight blightness
GetBlBrightness	Gets the backlight blightness

IMPORTANT

- When using the dll file from a created application, place the dll files in one of the following locations.

OS	Location
Windows®2000 Windows®XP	• the same directory as the start up program or • Windows directory's System32 folder Ex.) C:\Winnt\System32

5.8 Visual Basic Programing Cautions

When using API-DLLs, it is important to first create the driver object and get the device handle. When you finish using the API-DLL, you will need to destroy both the device handle and the driver object. Refer to the following example when developing your programs.

NOTE

- When using PsaDevWordWrite and PsaDevWordRead, it is not necessary to create/destroy the driver object and the device handle.

5.8.1 Sample Program

◆When using Psa_Ioc.dll

```
’Create the driver object
Call InitIoctl
’Get the device handle
Dim ret As Long
Dim Data As Long
Dim Hndl As Long
ret = GetDrvHandle(Hndl)
        .
        .
’Watch the +3.3V
ret = GetCurrentVolt(MONITOR_VOLT_P33, Data)
        .
        .
’Destroy the device handle
Dim ret As Long
ret = CloseDrvHandle()
’Destroy the driver object
Call EndIoctl
```

◆Example of using Psa_Blc.dll

```
'Create the driver object
Call InitBlctl
'Get the device handle
Dim ret As Long
Dim Hndl As Long
ret = GetBlDrvHandle(Hndl)
        .
        .
'Set the Backlight OFF
ret = SetBlControl(BACKLIGHT_OFF)
        .
        .
'Destroy the device handle
Dim ret As Long
ret = CloseBlDrvHandle()
'Destroy the driver object
Call EndBlctl
```

5.9 Visual Basic Function Specifications (Details)

5.9.1 Psa_Ioc.dll Functions

InitIoctl

Call Format

Declare Sub InitIoctl Lib”Psa_Ioc.dll”

Return Value

None

Argument

None

Processing

Creates a CPSA_Ioctl object. The created object will not be released until the “EndIoctl” function is called.

Example

CALL InitIoctl

EndIoctl

Call Format

Declare Sub EndIoctl Lib”Psa_Ioc.dll”()

Return Value

None

Argument

None

Processing

Destroys a CPSA_loctl object.

Example

CALL EndIoctl

GetDrvHandle

Call Format

Declare Function GetDrvHandle Lib"Psa_Ioc.dll"(ByRef Hndl As Long) As Long

Return Value

0	Normal
Other than 0	Error

Argument

Hndl As Long	Device driver handle (pass by reference).

Processing

Gets the device driver handle to exchange information with the device driver.

Example

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

NOTE

- An error will result if the system monitor/RAS device driver is not operating.

CloseDrvHandle

Call Format

Declare Function CloseDrvHandle Lib "Psa_Ioc.dll"() As Long

Return Value

Other than 0	Normal
0	Error

Argument

None

Processing

Destroys the handle acquired with the "GetDrvHandle" function.

Example

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

GetDrvVersion

Call Format

Declare Function GetDrvVersion Lib “Psa_Ioc.dll” (ByRef Major As Long, ByRef Minor As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Major As Long	Version Data (pass by reference)
Minor As Long	Version Data (pass by reference)

Processing

Gets the driver version.

Example

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

NOTE

- If the version is 1.00, then you will get

  Major : 1 (Decimal)

  Minor : 00 (Decimal).

GetDrvVersionEx

Call Format

Daclare Function GetDrvVersionEx Lib “Psa_Ioc.dll” (ByRef Product As Long, ByRef Major As Long, ByRef Minor As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Product As Long	Hardware Type (pass by reference)
Major As Long	Version Data (pass by reference)
Minor As Long	Version Data (pass by reference)

Processing

Gets the hardware type and driver version.

Example

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

NOTE

- If the H/W type is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will get

  Product : 3 (Decimal)

  Major : 1 (Decimal)

  Minor : 00 (Decimal).

GetMonitorSetup

Call Format

Declare Function GetMonitorSetup Lib “Psa_Ioc.dll” (ByVal Selector As Long, ByRef Setup As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
Setup As Long	Get data (pass by reference)	
	0 : Disable	
	1 : Enable	

Processing

Gets the current enabled/disabled monitor status.

Example

```
Dim ret As Long
Dim Setup As Long
’ Get the setup status of the +3.3V.
ret = GetMonitorSetup(MONITOR_VOLT_P33, Setup)
```

GetVoltParam

Call Format

Declare Function GetVoltParam Lib “Psa_Ioc.dll” (ByVal Selector As Long, ByRef ULimit As Long, ByRef LLimit As Long)

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
ULimit As Long	Voltage value upper-limit (Unit: mV)(pass by reference)	
LLimit As Long	Voltage value lower-limit (Unit: mV)(pass by reference)	

Processing

Gets the voltage monitoring parameter.

Example

```
Dim ret As Long
Dim ULimit As Long
Dim LLimit As Long
’ Get the upper/lower limit of the +3.3V value
ret = GetVoltParam(MONITOR_VOLT_P33, ULimit, LLimit)
```

NOTE

- Since the data received from this function is in mV units, the following conversion is needed for use in (Volt) units:
  Data in Volt unit = Data in mV unit / 1000

GetCurrentVolt

Call Format

Declare Function GetCurrentVolt Lib “PSa_Ioc.dll” (ByVal Selector As Long, ByRef Data As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
Data As Long	Voltage value (Unit: mV) (pass by reference)	

Processing

Gets the current voltage value.

Example

```
Dim ret As Long
Dim Data As Long
’ Get the +3.3V value.
ret = GetCurrentVolt(MONITOR_VOLT_P33, Data)
```

NOTE

- Since the data received from this function is in mV units, the following conversion is needed for use in (Volt) units:

  Data in Volt unit = Data in mV unit / 1000

GetFanParam

Call Format

Declare Function GetFanParam Lib “PSA_Ioc.dll” (ByVal Selector As Long, ByRef LLimit As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
	MONITOR_FAN_SYSTEM	SYSTEM fan
LLimit As Long	FAN revolution lower-limit value (Unit: RPM) (pass by reference) (RPM : Revolutions per minute)	

Processing

Gets the lower-limit value.

Example

```
Dim ret As Long
Dim LLimit As Long
’ Gets the CPU FAN lower-limit rpm value.
ret = GetFanParam(MONITOR_FAN_CPU, LLimit)
```

GetCurrentFan

Call Format

Declare Function GetCurrentFan Lib “Psa_Ioc.dll” (ByVal Selector As Long, ByRef Data As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER fan
Data As Long	FAN revolution value (Unit: RPM) (pass by reference)	
	(RPM : Revolutions per minute)	

Processing

Gets the current CPU FAN rpm.

Example

```
Dim ret As Long
Dim Data As Long
’ Gets the current CPU FAN rpm.
ret = GetCurrentFan(MONITOR_FAN_CPU, Data)
```

SetWdtCounter

Call Format

```
Declare Function SetWdtCounter Lib "Psa_Ioc.dll" (ByVal Counter As Long) As Long
```

Return Value

Other than 0	Normal
0	Error

Argument

Counter As Long	The initial counter value of the watchdog timer (5 to 255) (Unit: second) (pass by value)

Processing

Sets the initial counter value for the watchdog timer.

Example

```
Dim ret As Long
Dim Counter As Long
' Sets the initial counter value for the watchdog timer to 10 seconds.
Counter = 10
ret = SetWdtCounter(Counter)
```

GetWdtCounter

Call Format

Declare Function GetWdtCounter Lib “Psa_Ioc.dll” (ByRef Counter As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Counter As Long	The initial couter value of the watchdog timer (pass by value)

Processing

Gets the initial counter value of the current watchdog timer.

Example

```
Dim ret As Long
Dim Counter As Long
’ Gets the initial counter value of the current watchdog timer.
ret = GetWdtCounter(Counter)
```

StartWdt

Call Format

Declare Function StartWdt Lib “Psa_Ioc.dll”() As Long

Return Value

Other than 0	Normal
0	Error

Argument

None

Processing

Starts the Watchdog Timer countdown.

Example

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

StopWdt

Call Format

Declare Function StopWdt Lib “Psa_Ioc.dll”() As Long

Return Value

Other than 0	Normal
0	Error

Argument

None

Processing

Stops the Watchdog Timer coutdown.

Example

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

RestartWdt

Call Format

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

Return Value

Other than 0	Normal
0	Error

Argument

None

Processing

This feature resets the Watchdog timer to its initial value, and restarts the countdown.

Example

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

NOTE

- If StartWdt is called but the countdown has not yet started, an error will occur.
  Also, after a Timeout has occurred, if the error is cleared and StartWdt is called, but the count-down has not yet started, an error will occur.

GetWdtStatus

Call Format

Declare Function GetWdtStatus Lib “Psa_Ioc.dll” (ByRef RunFlag As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

RunFlag As Long	Operation Status of the watchdog timer (pass by reference)	
	WATCHDOG_STOP	Stopped
	WATCHDOG_COUNTDOWN	Counting down

Processing

Gets the operation status of the watchdog timer.

Example

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

GetEvent

Call Format

Declare Function GetEvent Lib “Psa_Ioc.dll” (ByVal Selector As Long, ByRef RasEvent As Long)

As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_FAN_SYSTEM	SYSTEM fan
	EVENT_WDT_TIMEOUT	Watchdog timer
	EVENT_BACKLIGHT	Backlight blowout
RasEvent As Long	Error event data (pass by reference)	
	ERROR_EVENT_ON	Error event
	ERROR_EVENT_OFF	No Error event

Processing

Gets the event information.

Example

```
Dim ret As Long
Dim RasEvent As Long
ret = GetEvent(EVENT_VOLT_P33, RasEvent)
```

ClearEvent

Call Format

Declare Function ClearEvent Lib “Psa_Ioc.dll” (ByVal Selector As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Selector As Long	Parameters (pass by value)	
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5.0V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_FAN_SYSTEM	SYSTEM fan
	EVENT_WDT_TIMEOUT	Watchdog Timer
	EVENT_BACKLIGHT	Backlight blowout

Processing

Cancels the error event.

Example

```
Dim ret As Long
ret = ClearEvent(EVENT_VOLT_P33)
```

GetWdtTimeout

Call Format

Declare Function GetWdtTimeout Lib “Psa_Ioc.dll” (ByRef Timebuf As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Timebuf As Long	Pointer to Watchdog Timer timeout status
	TIMEOUT_OK Timeout has not occurred
	TIMEOUT_ERR Timeout has occurred

Processing

Gets the watchdog timeout status.

Example

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

ClearWdtTimeout

Call Format

Declare Funciton ClearWdtTimeout Lib “Psa_Ioc.dll”() As Long

Return Value

Other than 0	Normal
0	Error

Argument

None

Processing

Clears the watchdog timeout status.

Example

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

SetWdtResetMask

Call Format

Declare Function SetWdtResetMask Lib “Psa_Ioc.dll” (ByVal Mask As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Mask As Long	Masking information (pass by value)	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

Processing

Sets the H/W reset mask for the Watchdog Timer timeout.

Example

Dim ret As Long

ret = SetWdtResetMask(MASK_ON)

GetWdtResetMask

Call Format

Declare Function GetWdtResetMask Lib “Psa_Ioc.dll” (ByRef Mask As Long) As Long

Return Value

Other than 0	Normal
0	Error

Argument

Mask As Long	Masking Information (pass by reference)	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

Processing

Gets the Watchdog Timer timeout’s H/W reset mask data.

Example

Dim ret As Long

Dim Mask As Long

ret = GetWdtResetMask(Mask)

GetLightblowErr

Call Format

Declare Function GetLightblowErr Lib "Psa_Ioc.dll"

(ByRef Light As Long) As Long

Return Value

Other than 0	Normal
0	Error

Arguments

Light As Long	Backlight blowout (pass by reference)	
	BACKLIGHT_BLOWOUT	Backlight blowout
	BACKLIGHT_GLOW	Normal

Processing

Gets the Backlight blowout status.

Example

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

5.9.2 Psa_Ras.dll Functions

PsaDevWordWrite

Call Format

Declare Function PsaDevWordWrite Lib “Psa_Ras.dll” (ByVal Addr As Long, ByVal wData As Long) As Long

Return Value

0	Normal
Other than 0	Error

Argument

Addr As Long	Write memory word address. (0 to 255) (pass by value)
wData As Long	Write data (0 to 65535) (pass by value)

Processing

Writes to common memory.

Example

```
’ Writes data 100 to address 255.
Dim ret As Long
ret = PsaDevWordWrite(255, 100)
```

PsaDevWordRead

Call Format

Declare Function PsaDevWordRead Lib “Psa_Ras.dll” (ByVal Addr As Long, ByRef wData As Long) As Long

Return Value

0	Normal
Other than 0	Error

Argument

Addr As Long	Read memory word address (0 to 255) (pass by value)
wData As Long	Read data (0 to 65535) (pass by reference)

Processing

Reads from common memory.

Example

```
’ Reads address 255’s data.
Dim ret As Long
Dim wData As Long
ret = PsaDevWordRead(255, wData)
```

5.9.3 Psa_Blc.dll Functions

InitBlctl

Call Format

Declare Sub InitBlctl Lib "Psa_Blc.dll" ()

Return Value

None

Arguments

None

Processing

Creates CPSA_Blctl object. The object once created is not destroyed until the EndBLIoctl function is called.

Example

’ Creates CPSA_Blctl object.

CALL InitBlctl

EndBlctl

Call Format

Declare Sub EndBlctl Lib "Psa_Blc.dll" ()

Return value

None

Arguments

None

Processing

Destroys the object of InitBlctl function.

Example

CALL EndBlctl

GetBlDrvHandle

Call Format

Declare Function GetBlDrvHandle Lib "Psa_Blc.dll" (byRef hndl As Long) As Long

Return Value

0	Normal
Other than 0	Error

Arguments

hndl As Long	Device driver handle (pass by reference)

Processing

Gets the device driver handle to communicate with device driver.

Example

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

CloseBlDrvHandle

Call Format

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

Return Value

Other than 0	Normal
0	Error

Arguments

None

Processing

Destroys the handle with GetBlDrvHandle.

Example

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

GetBlDrvVersion

Call Format

Declare Function GetBlDrvVersionLib "Psa_Blc.dll" (ByRef Major As Long, ByRef Minor As Long) As Long

Return value

Other than 0	Normal
0	Error

Arguments

Major As Long	the version information (pass by reference)
Minor As Long	the version information (pass by reference)

Processing

Gets the driver version.

Example

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

NOTE

- If the version is 1.00, the you will get

  Major : 1 (decimal)

  Minor : 00 (decimal)

GetBlDrvVersionEx

Call Format

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

Return Format

Other than 0	Normal
0	Error

Arguments

Product As Long	Product information (pass by reference)
Major As Long	Version information (pass by reference)
Minor As Long	Version information (pass by reference)

Processing

Gets the driver version and product information.

Example

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

NOTE

- If the product is PS-3700A (Eden™ ESP6000 - 667MHz Model) or PS-3701A (Eden™ ESP6000 - 667MHz Model) and the version is 1.00, then you will be get

  Product : 3 (decimal)

  Major : 1 (decimal)

  Minor : 00 (decimal)

SetBlControl

Call Format

Declare Function SetBlControl Lib "Psa_Blc.dll" (ByVal BlFlag As Long) As Long

Return Value

Other than 0	Normal
0	Error

Arguments

BlFlag As Long	Backlight setting parameter (pass by value)		
	BACKLIGHT_OFF	Backlight OFF	
	BACKLIGHT_ON	Backlight ON	

Processing

Sets the backlight ON/OF setting.

Example

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

GetBlControl

Call Format

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

Return Value

Other than 0	Normal
0	Error

Arguments

BlFlag As Long (pass by reference)

BACKLIGHT_ON	Backlight ON
BACKLIGHT_OFF	Backlight OFF

Processing

Gets the Backlight Status.

Example

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

SetBlBrightness

Call Format

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

Return Value

Other than 0Normal

0Error

Arguments

BlBright As Long (pass by value)

BRIGHT_LEVEL_0	Brightness level 0 (very dark)
BRIGHT_LEVEL_1	Brightness level 1 (dark)
BRIGHT_LEVEL_2	Brightness level 2 (bright)
BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Sets the backlight brightness.

Example

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

GetBlBrightness

Call Format

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

Return value

Other than 0	Normal
0	Error

Arguments

BlBright As Long (pass by reference)

BRIGHT_LEVEL_0	Brightness level 0 (very dark)
BRIGHT_LEVEL_1	Brightness level 1 (dark)
BRIGHT_LEVEL_2	Brightness level 2 (bright)
BRIGHT_LEVEL_3	Brightness level 3 (very bright)

Processing

Gets the backlight brightness.

Example

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