

**PS-2000B Series**

**RAS-API Reference Manual**

**Product names used in this manual are  
the trademarks of their respective manufacturers.**

# 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-2000B Series unit.

The API-DLLs explained in this document are PSB\_Ioc.dll and PSB\_Ras.dll.

PSB\_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, temperature)
- System Monitor current data (voltage, fan, temperature)
- Watchdog parameters
- Alarm processing
- General input processing
- Reset (of PS-B unit)
- Event handling
- Get Soft Mirror status\*1

PSB\_Ras.dll is used to access the Remote RAS feature's Common Memory API-DLL. PSB\_Ras.dll is used by applications to access the following features.

- Reads from common memory
- Writes to common memory

# 2 Compatible Operating Systems and Languages

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

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

\*1 Only when using the Soft Mirror (PL-SM900)

### 3 Required Files

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

Language	File Name	Description
Visual C	PSB_Iocif.h	Driver interface definition "include" file
	PSB_Ioc.lib	Library definition file
	PSB_Ioc.dll	Dynamic link library file
Visual C++	PSB_Iocif.h	Driver interface definition "include" file
	PSB_Iocall.h	CPSB_Iocall class definition "include" file *1
	PSB_Ioct1.h	CPSB_Ioct1 class definition "include" file
	PSB_Ioc.lib	Library definition file
	PSB_Ioc.dll	Dynamic link library file
	Sm.h	Soft Mirror definition file (Only when using Soft Mirror)
	PSB_SmiOct1.h	CPSB_SmiOct1 Class definition "include" file (Only when using Soft Mirror)
Visual Basic	PSB_Ioc.bas	Driver interface definition file
	PSB_Ioc.lib	Library definition file
	PSB_Ioc.dll	Dynamic link library file

\*1 “#include” header files should be “included” in the following order.  
 PSB\_Iocall.h is automatically included, and does not need to be directly designated.  
 #include PSB\_Iocif.h  
 #include PSB\_Ioct1.h

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

Language	File Name	Description
Visual C	PSB_Ras.h	Driver interface definition "include" file
	PSB_Ras.LIB	Library definition file
	PSB_Ras.dll	Dynamic link Library file
Visual C++	PSB_Ras.h	Driver interface definition "include" file
	PSB_Ras.LIB	Library definition file
	PSB_Ras.dll	Dynamic link Library file
Visual Basic	PSB_Ras.LIB	Library definition file
	PSB_Ras.dll	Dynamic link Library file

**MEMO**

Each file is located in the [Proface] folder's [PSBApi] folder. Place the files that you need in one of the following folders, or in the same folder as your application.

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

## 4 Class Contents

### 4.1 CPSB\_Ioctl Class

This class is used to set the parameters for device driver access using CPSB\_Ioctl class.

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

### 4.2 CPSB\_Iocall Class

This uses the parameters set in CPSB\_Ioctl and, calls up DeviceIOControl (Driver Access function).

However, since this class succeeds CPSB\_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	LPOVERLAPPED	m_ovlp	Address of overlap design

### 4.3 CPSB\_Smiocctl Class

This class is used to set the parameters for device driver access using CPSB\_Smiocctl class.

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

## 5 Function Specifications

### 5.1 Visual C Functions

#### PSB\_loc.dll Functions

Function Name	Description
InitIoctl	Creates the CPSB_Ioctl object
EndIoctl	Destroys the CPSB_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
GetTempParam	Gets the temperature monitoring parameter
GetCurrentTemp	Gets the current temperature value
SetWdtCounter	Sets the watchdog timer counter
GetWdtCounter	Gets the watchdog timer counter
SetWdtDOutMask	Sets warning masking in case of watchdog timer timeout
GetWdtDOutMask	Gets warning masking in case of watchdog timer timeout
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog timer operation status
SetDOut	Sets universal output
GetDOut	Gets universal output
GetDIn	Gets universal input
ClearDIn	Clear the universal input latched status
SetDInMask	Sets universal input masking
GetDInMask	Gets universal input masking
SetResetMask	Sets reset-masking
GetResetMask	Gets reset-masking
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 ResetMask of the watchdog timer
GetWdtResetMask	Gets the ResetMask of the watchdog timer
GetSmiDrvHandle	Gets the soft mirror driver handle
CloseSmiDrvHandle	Destroys the soft mirror driver handle
GetSmiAryStatus	Gets the soft mirror array status
GetSmeDevStatus	Gets the soft mirror device status

#### PSB\_Ras.dll Functions

Function Name	Description
PsbDevWordWrite	Writes to common memory
PsbDevWordRead	Reads from common memory

## 5.2 Visual C Programming 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.

### MEMO

Only when using PsbDevWordWrite and PsbDevWordRead, it is not necessary to create/destroy the driver object and the device handle.

### Sample Program

```
// Example of using an API-DLL
// Variable language
BOOL bRet;
int iRet;
HANDLE hDrv;
// Create the driver object and get the device handle
// Create CPSB_Ioctl object
InitIoctl();
iRet = GetDrvHandle(&Drv);
.
.
.
// Output to DOUT3
bRet = SetDOut(PORT_DOUT3, OUTPUT_ON);
.
.
// Application shut-down processing
// Destroy both the device handle and the driver object
bRet = CloseDrvHandle();
EndIoctl();
```

## 5.3 Visual C Function Specifications (Details)

### InitIoctl

Call Format

```
void WINAPI InitIoctl(void)
```

Return Value

None

Arguments

None

Processing

Creates a CPL\_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 CPSB\_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);
```

### **MEMO**

Error (Return Value : 1) 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);
```

### MEMO

For example, 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);
```

### MEMO

For example, if the H/W type is PS-2000B and the version is 1.00, then you will get

Product : 1 (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_CPU	CPU Voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU Voltage2
	MONITOR_TEMP_SYSTEM	System temperature
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER 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 CPU core voltage setup status.
```

```
ret = GetMonitorSetup(MONITOR_VOLT_CPU, &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\_CPU CPU voltage

MONITOR\_VOLT\_P33 +3.3V

MONITOR\_VOLT\_P50 +5.0V

MONITOR\_VOLT\_P12 +12V

MONITOR\_VOLT\_M12 -12V

MONITOR\_VOLT\_VIT CPU voltage2

(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 CPU core voltage.
```

```
ret = GetVoltParam(MONITOR_VOLT_CPU, &ULimit, &LLimit);
```

### MEMO

The data taken from this function is shown in mV units.

Perform the following conversion 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_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
(I/O) int *pData	Pointer to the voltage value (unit: mV)	

### Processing

Gets the current voltage value.

### Example

```
BOOL ret;
```

```
int Data;
```

```
// Gets the CPU core voltage value.
```

```
ret = GetCurrentVolt(MONITOR_VOLT_CPU, &Data);
```

### MEMO

The data taken from this function is shown in mV units.

Perform the following conversion 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
(I/O) int *pLLimit	Pointer to the lower-limit fan rotation speed (Unit: RPM)	
	(RPM : Revolutions Per Minute)	

### Processing

Gets the fan monitoring parameter.

### 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
(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);
```

## GetTempParam

### Call Format

```
BOOL WINAPI GetTempParam(int Selector, int *pULimit)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Parameters	
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_TEMP_SYSTEM	System temperature
(I/O) int *pULimit	Pointer to the upper-limit temperature (Unit:Degrees Celsius).	

### Processing

Gets the temperature monitoring parameter.

### Example

```
BOOL ret;
int ULimit;
// Gets the CPU temperature upper-limit value.
ret = GetTempParam(MONITOR_TEMP_CPU, &ULimit);
```

## GetCurrentTemp

### Call Format

BOOL WINAPI GetCurrentTemp(int Selector, int \*pData)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Parameters
	MONITOR_TEMP_CPU CPU temperature
	MONITOR_TEMP_SYSTEM System temperature
(I/O) int *pData	Pointer to the temperature (Unit:Degrees Celsius).

### Processing

Gets the current temperature value.

### Example

```
BOOL ret;
```

```
int Data;
```

```
// Gets the CPU temperature value.
```

```
ret = GetCurrentTemp(MONITOR_TEMP_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;
```

```
// Sets the current watchdog timer's initial counter value to 10 sec.
```

```
ret = SetWdtCounter(10);
```

## GetWdtCounter

### Call Format

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

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I/O) 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);
```

## SetWdtDOutMask

### Call Format

```
BOOL WINAPI SetWdtDOutMask(int Selector, int Mask)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector

Setting Item

PORT\_DOUT0 DOUT0

PORT\_DOUT1 DOUT1

PORT\_DOUT2 DOUT2

PORT\_DOUT3 DOUT3

(I) int Mask

Masking Information

MASK\_OFF Masking disabled

MASK\_ON Masking enabled

### Processing

Sets mask for the RAS port used for Watchdog Timer Timeout output.

### Example

```
BOOL ret;
```

```
// Mask the RAS port's DOUT0.
```

```
ret = SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

## GetWdtDOutMask

### Call Format

BOOL WINAPI GetWdtDOutMask(int Selector, int \*pMask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I/O) int *pMask	Pointer to Masking Information	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

### Processing

Gets the masking information used for RAS port when watchdog timer time-out occurs.

### Example

```
BOOL ret;
```

```
int Mask;
```

```
// Gets the masking information for the DOUT0 of RAS port.
```

```
ret = GetWdtDOutMask(PORT_DOUT0, &Mask);
```

## 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 countdown.

### 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 that is currently counting (down) to its initial value, and restarts the countdown.

### Example

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

**MEMO**

RestartWdt can only be used after the StartWdt countdown has started. If RestartWdt is used after the timeout, it should be after ClearWdtTimeout has used to clear the timeout condition and after StartWdt has started the countdown.

## 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);
```



## SetDOut

### Call Format

BOOL WINAPI SetDOut(int Selector, int Dout)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Dout	Output Status	
	OUTPUT_OFF	Output OFF
	OUTPUT_ON	Output ON

### Processing

Sets the Common Output port (DOUT) output data.

### Example

BOOL ret;

ret = SetDOut(PORT\_DOUT0, OUTPUT\_ON);

## GetDOut

### Call Format

BOOL WINAPI GetDOut(int Selector, int \*pDout)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I/O) int *pDout	Pointer to Output Status	
	OUTPUT_OFF	Output OFF
	OUTPUT_ON	Output ON

### Processing

Gets the Common Output port (DOUT) output data.

### Example

BOOL ret;

int Dout;

ret = GetDOut(PORT\_DOUT0, &Dout);

## GetDIn

### Call Format

BOOL WINAPI GetDIn(int Selector, int \*pDin)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pDin	Pointer to Input Status	
	INPUT_OFF	Input OFF
	INPUT_ON	Input ON

### Processing

Gets Input information.

### Example

```

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

```

## ClearDIn

### Call Format

BOOL WINAPI ClearDIn(int Selector)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3

### Processing

Releases the Common Input port (DIN) input status.

### Example

```

BOOL ret;
ret=ClearDIn(PORT_DIN0);

```

## SetDInMask

### Call Format

BOOL WINAPI SetDInMask(int Selector, int Mask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I) int Mask	Masking Information	
	MASK_ON	Masking enabled
	MASK_OFF	Masking disabled

### Processing

Sets the Common Input Port (DIN) input status.

### Example

BOOL ret;

ret = SetDInMask(PORT\_DIN0, MASK\_OFF);

## GetDInMask

### Call Format

BOOL WINAPI GetDInMask(int Selector, int \*pMask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pMask	Pointer to Masking Information	
	MASK_ON	Masking enabled
	MASK_OFF	Masking disabled

### Processing

Gets the masking information for the designated port (DIN)

### Example

BOOL ret;

int Mask;

ret = GetDInMask(PORT\_DIN0, &Mask);

## SetResetMask

### Call Format

```
BOOL WINAPI SetResetMask(int Mask)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

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

### Processing

Sets the reset port's mask data.

### Example

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

## GetResetMask

### Call Format

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

### Return Value

TRUE Normal

FALSE Error

### Arguments

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

### Processing

Gets the current reset-masking information.

### Example

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

## GetEvent

### Call Format

BOOL WINAPI GetEvent(int Selector, int \*pRasevnt)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector

#### Parameters

EVENT_VOLT_CPU	CPU voltage
EVENT_VOLT_P33	+3.3V
EVENT_VOLT_P50	+5V
EVENT_VOLT_P12	+12V
EVENT_VOLT_M12	-12V
EVENT_VOLT_VIT	CPU voltage2
EVENT_FAN_CPU	CPU fan
EVENT_FAN_POWER	POWER fan
EVENT_TEMP_SYSTEM	System temperature
EVENT_TEMP_CPU	CPU temperature
EVENT_DIN0	DIN0
EVENT_DIN1	DIN1
EVENT_DIN2	DIN2
EVENT_DIN3	DIN3
EVENT_WDT_TIMEOUT	Watchdog Timer
(I/O) int *pRasevnt	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 Rasevnt;

ret = GetEvent(EVENT\_DIN0, &Rasevnt);

## ClearEvent

### Call Format

BOOL WINAPI ClearEvent(int Selector)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector

### Parameters

EVENT_VOLT_CPU	CPU voltage
EVENT_VOLT_P33	+3.3V
EVENT_VOLT_P50	+5V
EVENT_VOLT_P12	+12V
EVENT_VOLT_M12	-12V
EVENT_VOLT_VIT	CPU voltage2
EVENT_FAN_CPU	CPU fan
EVENT_FAN_POWER	POWER fan
EVENT_TEMP_SYSTEM	System temperature
EVENT_TEMP_CPU	CPU temperature
EVENT_DIN0	DIN0
EVENT_DIN1	DIN1
EVENT_DIN2	DIN2
EVENT_DIN3	DIN3
EVENT_WDT_TIMEOUT	Watchdog Timer

### Processing

Cancels the error event.

### Example

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

## GetWdtTimeout

### Call Format

BOOL WINAPI GetWdtTimeout(int \*pTimebuf)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I/O) 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) int Mask                      Masking Information  
                                    MASK\_OFF              Masking disabled  
                                    MASK\_ON                Masking enabled  
Processing  
Sets the H/W reset mask used at WDT 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 WDT timeout.  
Example  
BOOL ret;  
int Mask;  
ret = GetWdtResetMask(&Mask);

## PsbDevWordWrite

### Call Format

```
long PsbDevWordWrite(long Addr, long wData)
```

### Return Value

0                    Normal

Other than 0        Error

### Arguments

(I) long Addr                Write memory word address

(I) long wData               Write data (0 to 65535)

### Processing

Writes to common memory.

### Example

```
// Writes data 255 to address 255.
```

```
long ret;
```

```
ret = PsbDevWordWrite(255, 255);
```

## PsbDevWordRead

### Call Format

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

### Return Value

0                    Normal

Other than 0        Error

### Arguments

(I) long Addr                Read memory word address

(I/O) 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 = PsbDevWordRead(255, &wData);
```



## GetSmiDrvHandle

### Call Format

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

### Return Value

0 Normal

1 Error

### Arguments

None

### Processing

Gets the Soft Mirror driver handle.

### Example

```
// Gets the handle.
```

```
int ret;
```

```
ret = GetSmiDrvHandle();
```

### MEMO

Error (Return Value : 1) will result if the soft mirror device driver is not operating.

## CloseSmiDrvHandle

### Call Format

```
BOOL WINAPI CloseSmiDrvHandle(void)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

None

### Processing

Destroys the Soft Mirror driver handle.

### Example

```
// Destroys the handle.
```

```
BOOL ret;
```

```
ret = CloseSmiDrvHandle();
```

## GetSmiAryStatus

### Call Format

```
BOOL WINAPI GetSmiAryHandle(int *pStatus)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I/O) int \*pStatus Pointor to mirroring status

AYRSTAT\_GOOD Normal

ARYSTAT\_UNCONFIG Under Configuration

ARYSTAT\_REBUILD Rebuild

ARYSTAT\_REDUCE Reduced

ARYSTAT\_DEAD Destroy the mirror status

### Processing

Gets the Soft Mirror's mirroring status.

### Example

```
BOOL ret;
```

```
int Status;
```

```
ret = GetSmiAryStatus();
```

## GetSmiDevStatus

### Call Format

```
BOOL WINAPI GetSmiDevStatus(int Id, int *pType, int *pStatus)
```

### Return Value

TRUE            Normal

FALSE          Error

### Arguments

(I) int Id	Device ID 0 : Master HDD 1 : Slave HDD	
(I/O) int *pType	Device Type ATADEVICE ATAPIDEVICE NODEVICE	ATA device CD-ROMdrive Not connected
(I/O) int *pStatus	Device Status DEVSTAT_GOOD DEVSTAT_NOTEXIST DEVSTAT_BROKEN	Normal Not connected Broken

### Processing

Gets the soft mirror's device status.

### Example

```
//Gets the master HDD's device status
BOOL ret;
int Type, Status;
ret = GetSmiDevStatus(0, &Type, &Status);
```

*Memo*

## 5.4 Visual C++ Functions

## PSB\_loc.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
GetTempParam	Gets the temperature monitoring parameter
GetCurrentTemp	Gets the current temperature value
SetWdtCounter	Sets the watchdog timer counter
GetWdtCounter	Gets the watchdog timer counter
SetWdtDOutMask	Sets the warning masking in case of watchdog timer time-out
GetWdtDOutMask	Gets the warning masking in case of watchdog timer time-out
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog timer operation status
SetDOut	Sets universal output
GetDOut	Gets universal output
GetDIn	Gets universal input
ClearDIn	Clears the universal input latched status
SetDInMask	Sets universal input masking
GetDInMask	Gets universal input masking
SetResetMask	Sets reset-masking
GetResetMask	Gets reset-masking
GetEvent	Gets the error event
ClearEvent	Clears the error event
GetWdtTimeout	Gets watchdog timeout status
ClearWdtTimeout	Clears the watchdog timeout status
SetWdtResetMask	Sets the RestMask of the watchdog timer
GetWdtResetMask	Gets the ResetMask of the watchdog timer
GetSmiDrvHandle	Gets the soft mirror driver handle
CloseSmiDrvHandle	Destroys the soft mirror driver handle
GetSmiAryStatus	Gets the soft mirror array status
GetSmiDevStatus	Gets the soft mirror device status

## PSB\_Ras.dll Functions

Function Name	Description
PsbDevWordWrite	Writes to common memory
PsbDevWordRead	Reads from common memory

## 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.

### MEMO

Only when using `PsbDevWordWrite` and `PsbDevWordRead`, it is not necessary to create/destroy the driver object and the device handle.

### Sample Program

```
// Example of using an API-DLL
// Variable language
BOOL bRet;
int iRet;
// Get the device handle
CPSB_Ioc1 m_Ioc;
iRet = m_Ioc.GetDrvHandle();
.
.
.
// Output to DOUT0
bRet=m_Ioc.SetDOut(PORT_DOUT0,OUTPUT_ON);
.
.
// Destroy the device handle
bRet = m_Ioc.CloseDrvHandle();
```

## 5.6 Visual C++ Function Specifications (Details)

### 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

```
CPSB_Ioc1 m_Ioc1;
```

```
int ret;
```

```
ret = m_Ioc1.GetDrvHandle();
```

### MEMO

Error (Return Value : 1) will result 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

```
CPSB_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

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Major, Minor;
```

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

### MEMO

For example, 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

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Product, Major, Minor;
```

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

**MEMO**

For example, if the H/W type is PS-2000B and the version is 1.00, then you will get

Product : 1 (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_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
	MONITOR_TEMP_SYSTEM	System temperature
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER 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 CPU core voltage setup status.
```

```
ret = ::GetMonitorSetup(MONITOR_VOLT_CPU, &Setup);
```

#### Example2

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Setup;
```

```
// Gets the CPU core voltage setup status.
```

```
ret=m_Ioctl.GetMonitorSetup(MONITOR_VOLT_CPU,&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_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
(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 CPU core voltage.
ret = ::GetVoltParam(MONITOR_VOLT_CPU, &ULimit, &LLimit);
```

### Example2

```
CPSB_Ioct1 m_Ioct1;
BOOL ret;
int ULimit, LLimit;
// Gets the upper and lower-limit values of the CPU core voltage.
ret = m_Ioct1.GetVoltParam(MONITOR_VOLT_CPU, &ULimit, &LLimit);
```

### MEMO

The data taken from this function is shown in mV units.  
 Perform the following conversion 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_CPU	CPU voltage
MONITOR_VOLT_P33	+3.3V
MONITOR_VOLT_P50	+5.0V
MONITOR_VOLT_P12	+12V
MONITOR_VOLT_M12	-12V
MONITOR_VOLT_VIT	CPU voltage2

(I/O) int \*pData

Pointer to the voltage value (Unit: mV)

### Processing

Gets the current voltage value.

### Example1

```
BOOL ret;
```

```
int Data;
```

```
// Gets the CPU core voltage value.
```

```
ret = ::GetCurrentVolt(MONITOR_VOLT_CPU, &Data);
```

### Example2

```
CPSB_Ioct1 m_Ioct1;
```

```
BOOL ret;
```

```
int Data;
```

```
// Gets the CPU core voltage value.
```

```
ret = m_Ioct1.GetCurrentVolt(MONITOR_VOLT_CPU, &Data);
```

### MEMO

The data taken from this function is shown in mV units.

Perform the following conversion 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
(I/O) int *pLLimit	Pointer to the lower-limit fan rotation speed (Unit: RPM) (RPM : Revolutions Per Minute)	

### Processing

Gets the fan monitoring parameter.

### Example1

```
BOOL ret;
```

```
int LLimit;
```

```
// Gets the lower-limit CPU fan rotational speed.
```

```
ret = ::GetFanParam(MONITOR_FAN_CPU, &LLimit);
```

### Example2

```
CPSB_Ioct1 m_Ioct1;
```

```
BOOL ret;
```

```
int LLimit;
```

```
// Gets the lower-limit CPU fan rotational speed.
```

```
ret = m_Ioct1.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
(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

```
CPSB_Ioct1 m_Ioct1;
```

```
BOOL ret;
```

```
int Data;
```

```
// Gets the CPU fan rotation speed.
```

```
ret = m_Ioct1.GetCurrentFan(MONITOR_FAN_CPU, &Data);
```

## GetTempParam

### Call Format

```
BOOL GetTempParam(int Selector, int *pULimit)
```

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Parameters	
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_TEMP_SYSTEM	System temperature
(I/O) int *pULimit		Pointer to the upper-limit temperature (Unit: Degrees Celsius)

### Processing

Gets the temperature monitoring parameter.

### Example1

```
BOOL ret;
int ULimit;
// Gets the CPU temperature upper-limit value.
ret = ::GetTempParam(MONITOR_TEMP_CPU, &ULimit);
```

### Example2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int ULimit;
// Gets the CPU temperature upper-limit value.
ret = m_Ioctl.GetTempParam(MONITOR_TEMP_CPU, &ULimit);
```

## GetCurrentTemp

### Call Format

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

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Parameters	
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_TEMP_SYSTEM	System temperature
(I/O) int *pData		Pointer to the temperature (Unit: Degrees Celsius)

### Processing

Gets the current temperature value.

### Example1

```
BOOL ret;
int Data;
// Gets the CPU temperature value.
ret = ::GetCurrentTemp(MONITOR_TEMP_CPU, &Data);
```

### Example2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
int Data;
// Gets the CPU temperature value.
ret = m_Ioctl.GetCurrentTemp(MONITOR_TEMP_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 timers initial counter value.

### Example1

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

### Example2

```
CPSB_Ioctl m_Ioctl;
BOOL ret;
// Sets the watchdog timers initial counter value to 10.
ret = m_Ioctl.SetWdtCounter(10);
```

## GetWdtCounter

### Call Format

BOOL GetWdtCounter(int \*pCounter)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I/O) 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

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

## SetWdtDOutMask

### Call Format

BOOL SetWdtDOutMask(int Selector, int Mask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Mask	Masking Information	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

### Processing

Sets mask for the RAS port used for Watchdog Timer Timeout output.

#### Example1

```
BOOL ret;
```

```
// Mask the RAS port's DOUT0.
```

```
ret = ::SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

#### Example2

```
CPSB_Ioctlm_Ioctl;
```

```
BOOL ret;
```

```
// Mask the RAS port's DOUT0.
```

```
ret = m_Ioctl.SetWdtDOutMask(PORT_DOUT0, MASK_ON);
```

## GetWdtDOutMask

### Call Format

BOOL GetWdtDOutMask(int Selector, int \*pMask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I/O) int *pMask	Pointer to Masking Information	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

### Processing

Gets the RAS port masking information that is created when a watchdog timer time-out occurs.

### Example1

```
BOOL ret;
```

```
int Mask;
```

```
// Gets the DOUT0 of RAS port masking information.
```

```
ret = ::GetWdtDOutMask(PORT_DOUT0, &Mask);
```

### Example2

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Mask;
```

```
// Gets the DOUT0 of RAS port masking information.
```

```
ret = m_Ioctl.GetWdtDOutMask(PORT_DOUT0, &Mask);
```

## 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

```
CPSB_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 countdown.

Example1

```
BOOL ret;
```

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

Example2

```
CPSB_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 that is currently counting (down) to its initial value, and restarts the countdown.

Example1

```
BOOL ret;
```

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

Example2

```
CPSB_Ioctlm_Ioctl;
```

```
BOOL ret;
```

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

### MEMO

RestartWdt can only be used after the StartWdt countdown has started. If RestartWdt is used after the timeout, it should be after ClearWdtTimeout has used to clear the timeout condition and after StartWdt has started the countdown.



## 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

```

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

```

## SetDOut

### Call Format

BOOL SetDOut(int Selector, int Dout)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I) int Dout	Output Status	
	OUTPUT_OFF	Output OFF
	OUTPUT_ON	Output ON

### Processing

Sets the Common Output port (DOUT) output data.

### Example1

```

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

```

### Example2

```

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

```

## GetDOut

### Call Format

BOOL GetDOut(int Selector, int \*pDout)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
(I/O) int *pDout	Pointer to Output Status	
	OUTPUT_OFF	Output OFF
	OUTPUT_ON	Output ON

### Processing

Gets the Common Output port (DOUT) output data.

### Example1

```

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

```

### Example2

```

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

```

## GetDIn

### Call Format

BOOL GetDIn(int Selector, int \*pDin)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pDin	Pointer to Input Status	
	INPUT_OFF	Input OFF
	INPUT_ON	Input ON

### Processing

Gets the input status of the designated port (DIN).

### Example1

```

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

```

### Example2

```

CPSB_Ioctl m_Ioctl;
BOOL ret;
int Din;
ret = m_Ioctl.GetDIn(PORT_DIN0, &Din);

```

## ClearDIn

### Call Format

BOOL ClearDIn(int Selector)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3

### Processing

Releases the Common Input port (DIN) input status.

### Example1

BOOL ret;

ret = ::ClearDIn(PORT\_DIN0);

### Example2

CPSB\_Ioctl m\_Ioctl;

BOOL ret;

ret = m\_Ioctl.ClearDIn(PORT\_DIN0);

## SetDInMask

### Call Format

BOOL SetDInMask(int Selector, int Mask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I) int Mask	Masking Information	
	MASK_ON	Masking enabled
	MASK_OFF	Masking disabled

### Processing

Sets the Common Input Port (DIN) input status.

### Example1

BOOL ret;

ret = ::SetDInMask(PORT\_DIN0, MASK\_OFF);

### Example2

CPSB\_Ioctl m\_Ioctl;

BOOL ret;

ret = m\_Ioctl.SetDInMask(PORT\_DIN0, MASK\_OFF);

## GetDInMask

### Call Format

BOOL GetDInMask(int Selector, int \*pMask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector	Setting Item	
	PORT_DIN0	DIN0
	PORT_DIN1	DIN1
	PORT_DIN2	DIN2
	PORT_DIN3	DIN3
(I/O) int *pMask	Pointer to Masking Information	
	MASK_ON	Masking enabled
	MASK_OFF	Masking disabled

### Processing

Gets the masking information for the designated port (DIN).

### Example1

```

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

```

### Example2

```

CPSB_Ioctl m_Ioctl;
BOOL ret;
int Mask;
ret = m_Ioctl.GetDInMask(PORT_DIN0, &Mask);

```

## SetResetMask

### Call Format

BOOL SetResetMask(int Mask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

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

### Processing

Sets the reset port's mask data.

### Example1

```

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

```

### Example2

```

CPSB_Ioctl m_Ioctl;
BOOL ret;
ret = m_Ioctl.SetResetMask(MASK_OFF);

```

## GetResetMask

### Call Format

BOOL GetResetMask(int \*pMask)

### Return Value

TRUE Normal

FALSE Error

### Arguments

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

### Processing

Gets the current reset-masking information.

### Example1

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

### Example2

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

## GetEvent

### Call Format

BOOL GetEvent(int Selector, int \*pRasevnt)

### Return Value

TRUE Normal

FALSE Error

### Arguments

(I) int Selector

#### Parameters

EVENT_VOLT_CPU	CPU voltage
EVENT_VOLT_P33	+3.3V
EVENT_VOLT_P50	+5V
EVENT_VOLT_P12	+12V
EVENT_VOLT_M12	-12V
EVENT_VOLT_VIT	CPU voltage2
EVENT_FAN_CPU	CPU fan
EVENT_FAN_POWER	POWER fan
EVENT_TEMP_SYSTEM	System temperature
EVENT_TEMP_CPU	CPU temperature
EVENT_DIN0	DIN0
EVENT_DIN1	DIN1
EVENT_DIN2	DIN2
EVENT_DIN3	DIN3
EVENT_WDT_TIMEOUT	Watchdog Timer
(I/O) int *pRasevnt	Pointer to Event Information
ERROR_EVENT_ON	With Event
ERROR_EVENT_OFF	Without Event

### Processing

Gets event information.

#### Example1

```
BOOL ret;
```

```
int Rasevnt;
```

```
ret::GetEvent(EVENT_DIN0,&Rasevnt);
```

#### Example2

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Rasevnt;
```

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

## ClearEvent

Call Format

BOOL ClearEvent(int Selector)

Return Value

TRUE Normal

FALSE Error

Arguments

(I) int Selector

Parameters

EVENT_VOLT_CPU	CPU voltage
EVENT_VOLT_P33	+3.3V
EVENT_VOLT_P50	+5V
EVENT_VOLT_P12	+12V
EVENT_VOLT_M12	-12V
EVENT_VOLT_VIT	CPU voltage2
EVENT_FAN_CPU	CPU fan
EVENT_FAN_POWER	POWER fan
EVENT_TEMP_SYSTEM	System temperature
EVENT_TEMP_CPU	CPU temperature
EVENT_DIN0	DIN0
EVENT_DIN1	DIN1
EVENT_DIN2	DIN2
EVENT_DIN3	DIN3
EVENT_WDT_TIMEOUT	Watchdog Timer

Processing

Cancels the error event.

Example1

```
BOOL ret;
```

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

Example2

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
ret = m_Icotl.ClearEvent(EVENT_DIN0);
```

## GetWdtTimeout

### Call Format

BOOL GetWdtTimeout(int \*pTimebuf)

### Return Value

TRUE Normal

FALSE Error

### Arguments

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

### Processing

Gets watchdog timeout status.

### Example1

```

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

```

### Example2

```

CPSB_Ioctl 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

```

CPSB_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 WDT Timeout H/W reset-masking.

### Example1

```
BOOL ret;
```

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

### Example2

```
CPSB_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 WDT timeout H/W reset-masking information.

### Example1

```
BOOL ret;
```

```
int Mask;
```

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

### Example2

```
CPSB_Ioctl m_Ioctl;
```

```
BOOL ret;
```

```
int Mask;
```

```
ret = m_Ioctl.GetWdtResetMask(&Mask);
```

## PsbDevWordWrite

### Call Format

```
long PsbDevWordWrite(long Addr, long wData)
```

### Return Value

0                    Normal

Other than 0        Error

### Arguments

(I) long Addr                Write memory word address

(I) long wData               Write data (0 to 65535)

### Processing

Writes to common memory.

### Example

```
// Writes data 255 to address 255.
```

```
long ret;
```

```
ret = PsbDevWordWrite(255, 255);
```

## PsbDevWordRead

### Call Format

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

### Return Value

0                    Normal

Other than 0        Error

### Arguments

(I) long Addr                Read memory word address

(I/O) 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 = PsbDevWordRead(255, &wData);
```

## GetSmiDrvHandle

Call Format

intGetSmiDrvHandle(void)

Return Value

0 Normal

1 Error

Arguments

None

Processing

Gets the soft mirror device driver handle.

Example 1

```
int ret;
```

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

Example 2

```
CPSB_SmiIoctlm_SmiIoctl;
```

```
int ret;
```

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

**MEMO**

Error (Return Value : 1) will result if the soft mirror device driver is not operating.

## CloseSmiDrvHandle

Call Format

BOOLCloseSmiDrvHandle(void)

Return Value

TRUE Normal

FALES Error

Arguments

None

Processing

Destroys the soft mirror device driver handle.

Example 1

```
BOOLret;
```

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

Example 2

```
CPSB_SmiIoctlm_SmiIoctl;
```

```
BOOLret;
```

```
ret=m_SmiIoctl.CloseSmiDrvHandle();
```

## GetSmiAryStatus

### Call Format

```
BOOL GetSmiAryStatus(int *pStatus)
```

### Return Value

TRUE            Normal

FALSE           Error

### Arguments

(I) int *pStatus	Pointor to the soft mirror's mirroring status
ARYSTAT_GOOD	Normal
ARYSTAT_UNCONFIG	Under configuration
ARYSTAT_REBUILD	Rebuild
ARYSTAT_REDUCED	Reduced
ARYSTAT_DEAD	Destroys the mirror status

### Processing

Gets the soft mirror's mirroring status.

### Example 1

```
BOOL ret;
```

```
int Status
```

```
ret = GetSmiAryStatus(&Status);
```

### Example 2

```
CPSB_SmiIoctl m_SmiIoctl;
```

```
BOOL ret;
```

```
int Status
```

```
ret = m_SmiIoctl.GetSmiAryStatus(&Status);
```

## GetSmiDevStatus

### Call Format

```
BOOL GetSmiDevStatus(int ID, int *pType, int *pStatus)
```

### Return Value

TRUE            Normal

FALSE          Error

### Arguments

(I) int ID	Device ID 0 : Master HDD 1 : Slave HDD	
(I/O) int *pType	pointer to the device type ATADEVICE            ATA device ATAPIDEVICE         CD-ROM NODEVICE             Not connected	
(I/O) int *pStatus	pointer to device status DEVSTAT_GOOD        Normal DEVSTAT_NOTEXIST    Not connected DEVSTAT_BROKEN      Broken	

### Processing

Gets the soft mirror device status.

### Example 1

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

### Example 2

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

## 5.7 Visual Basic Functions

### PSB\_loc.dll Functions

Function Name	Description
InitIoctl	Creates a CFT_Ioctl object
EndIoctl	Destroys a CFT_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
GetWdtCounter	Gets the watchdog timer counter
SetWdtDOutMask	Sets the watchdog timer counter time-out status warning mask
GetWdtDOutMask	Gets the watchdog timer counter time-out status warning mask
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
GetWdtStatus	Gets the watchdog status
SetDOut	Sets the universal output
GetDOut	Gets the universal output
GetDIn	Gets the universal input
ClearDIn	Clears the universal input
SetDInMask	Sets the universal input mask
GetDInMask	Gets the universal input mask
SetResetMask	Sets the reset mask
GetResetMask	Gets the reset mask
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
GetSmiDrvHandle	Gets the soft mirror driver handle
CloseSmiDrvHandle	Destroy the soft mirror driver handle
GetSmiAryStatus	Gets the soft mirror array status
GetSmiDevStatus	Gets the soft mirror device status

### PSB\_Ras.dll Functions

Function Name	Description
PsbDevWordWrite	Writes to common memory
PsbDevWordRead	Reads from common memory

## 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.

### MEMO

Only when using PsbDevWordWrite and PsbDevWordRead, it is not necessary to create/destroy the driver object and the device handle.

### Sample Program

```
// Create the driver object
Call InitIoctl
// Get the device handle
Dim ret As Long
Dim Hndl As Long
ret = GetDrvHandle(Hndl)
.
.
.
// Output to DOUT0
Dim ret As Long
ret = SetDOut(PORT_DOUT0, OUTPUT_ON)
.
.
// Destroy the device handle
Dim ret As Long
ret = CloseDrvHandle()
// Destroy the driver object
Call EndIoctl
```

## 5.9 Visual Basic Function Specifications (Details)

### InitIoctl

Call Format  
 Declare Sub InitIoctl Lib"PSB\_Ioc.dll"  
 Return Value  
 None  
 Argument  
 None  
 Processing  
 Creates a CPSB\_Ioc object. The created object will not be released until the "EndIoctl" function is called.  
 Example  
 CallInitIoctl

## EndIoctl

Call Format

Declare Sub EndIoctl Lib "PSB\_Ioc.dll"()

Return Value

None

Argument

None

Processing

Destroys CPSB\_Iocctl object.

Example

CallEndIoctl

## GetDrvHandle

Call Format

Declare Function GetDrvHandle Lib "PSB\_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)

### **MEMO**

Error (Return Value : Other than 0) will result if the system monitor/RAS device driver is not operating.

## CloseDrvHandle

Call Format

Declare Function CloseDrvHandle Lib "PSB\_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 "PSB\_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)
```

### MEMO

For Example, if the version is 1.00, then you will get  
Major : 1 (Decimal)  
Minor : 00 (Decimal).

## GetDrvVersionEx

### Call Format

Daclare Function GetDrvVersionEx Lib "PSB\_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)
```

### MEMO

For example, if the H/W type is PS-2000B and the version is 1.00, then you will get  
Product : 1 (Decimal)  
Major : 1 (Decimal)  
Minor : 00 (Decimal).

## GetMonitorSetup

### Call Format

Declare Function GetMonitorSetup Lib "PSB\_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_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
	MONITOR_TEMP_SYSTEM	System temperature
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_FAN_CPU	CPU fan
	MONITOR_FAN_POWER	POWER 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 CPU core voltage.

ret = GetMonitorSetup(MONITOR\_VOLT\_CPU, Setup)

## GetVoltParam

### Call Format

Declare Function GetVoltParam Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByVal ULimit As Long, ByVal LLimit As Long)

### Return Value

Other than 0      Normal  
 0                      Error

### Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_VOLT_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
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 CPU core voltage value

ret = GetVoltParam(MONITOR\_VOLT\_CPU, ULimit, LLimit)

### **MEMO**

The data received from this function is in mV units.  
 Perform the following conversion for use in (Volt) units:  
 Data in Volt unit = Data in mV unit / 1000

## GetCurrentVolt

### Call Format

Declare Function GetCurrentVolt Lib "PSB\_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_CPU	CPU voltage
	MONITOR_VOLT_P33	+3.3V
	MONITOR_VOLT_P50	+5.0V
	MONITOR_VOLT_P12	+12V
	MONITOR_VOLT_M12	-12V
	MONITOR_VOLT_VIT	CPU voltage2
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 CPU core voltage value.

ret = GetCurrentVolt(MONITOR\_VOLT\_CPU, Data)

### MEMO

The data received from this function is in mV units.  
Perform the following conversion for use in (Volt) units:  
Data in Volt unit = Data in mV unit / 1000

## GetFanParam

### Call Format

Declare Function GetFanParam Lib "PSB\_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
LLimit As Long	FAN revolution lower-limit value (Unit: RPM) (pass by reference) (RPM : Revolutions per minute)	

### Processing

Gets the parameter for monitoring the FAN.

### 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 "PSB\_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 FAN rpm.

### Example

Dim ret As Long

Dim Data As Long

' Gets the RPM of the CPU FAN.

ret = GetCurrentFan(MONITOR\_FAN\_CPU, Data)

## GetTempParam

### Call Format

Declare Function GetTempParam Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByRef ULimit As Long) As Long

### Return Value

Other than 0      Normal

0                    Error

### Argument

Selector As Long	Parameters (pass by value)	
	MONITOR_TEMP_CPU	CPU temperature
	MONITOR_TEMP_SYSTEM	System temperature
ULimit As Long	Temperature upper-limit (Unit °C) (pass by reference)	

### Processing

Gets the parameter for monitoring the temperature.

### Example

Dim ret As Long

Dim ULimit As Long

' Gets the upper-limit of CPU temperature.

ret = GetTempParam(MONITOR\_TEMP\_CPU, ULimit)

## GetCurrentTemp

### Call Format

Declare Function GetCurrentTemp Lib "PSB\_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_TEMP_CPU	CPU temperature
	MONITOR_TEMP_SYSTEM	System temperature
Data As Long	Temperature value (Unit: °C) (pass by reference)	

### Processing

Gets the current temperature value.

### Example

Dim ret As Long

Dim Data As Long

' Get the current value of CPU temperature.

ret = GetCurrentTemp(MONITOR\_TEMP\_CPU, Data)

## SetWdtCounter

### Call Format

Declare Function SetWdtCounter Lib "PSB\_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

' Sets the initial counter value for the watchdog timer to 10 seconds.

ret = SetWdtCounter(10)

## GetWdtCounter

### Call Format

Declare Function GetWdtCounter Lib "PSB\_Ioc.dll" (ByRef 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 (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)
```

## SetWdtDOutMask

### Call Format

Declare Function SetWdtDOutMask Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByVal Mask As Long)

### Return Value

Other than 0      Normal

0                    Error

### Argument

Selector As Long            Setting Item (pass by value)

PORT\_DOUT0                DOUT0

PORT\_DOUT1                DOUT1

PORT\_DOUT2                DOUT2

PORT\_DOUT3                DOUT3

Mask As Long                Mask Information (pass by value)

MASK\_OFF                    Masking disabled

MASK\_ON                     Masking enabled

### Processing

Sets the mask for the RAS port used for Watchdog Timer Timeout output.

### Example

```
Dim ret As Long
```

```
' Mask the RAS port's DOUT0.
```

```
ret = SetWdtDOutMask(PORT_DOUT0, MASK_ON)
```

## GetWdtDOutMask

### Call Format

Declare Function GetWdtDOutMask Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByRef Mask As Long) As Long

### Return Value

Other than 0      Normal  
0                    Error

### Argument

Selector As Long	Setting Item (pass by value)	
	PORT_DOUT0	DOUT0
	PORT_DOUT1	DOUT1
	PORT_DOUT2	DOUT2
	PORT_DOUT3	DOUT3
Mask As Long	Masking Information (pass by reference)	
	MASK_OFF	Masking disabled
	MASK_ON	Masking enabled

### Processing

Gets the WDT time out warning output of the RAS port's mask data.

### Example

Dim ret As Long

Dim Mask As Long

' Gets DOUT0 of RAS port mask data.

ret = GetWdtDOutMask(PORT\_DOUT0, Mask)

## StartWdt

### Call Format

Declare Function StartWdt Lib "PSB\_Ioc.dll"() As Long

### Return Value

Other than 0      Normal  
0                    Error

### Argument

None

### Processing

Starts the WDT countdown.

### Example

Dim ret As Long

ret = StartWdt()



## StopWdt

Call Format

Declare Function StopWdt Lib "PSB\_Ioc.dll"() As Long

Return Value

Other than 0      Normal

0                    Error

Argument

None

Processing

Stops the WDT countdown.

Example

Dim ret As Long

ret = StopWdt()

## RestartWdt

Call Format

Declare Function RestartWdt Lib "PSB\_Ioc.dll"() As Long

Return Value

Other than 0      Normal

0                    Error

Argument

None

Processing

This feature resets the Watchdog timer that is currently counting (down) to its initial value, and restarts the countdown.

Example

Dim ret As Long

ret = RestartWdt()

**MEMO** RestartWdt can only be used after the StartWdt countdown has started. If RestartWdt is used after the timeout, it should be after ClearWdtTimeout has used to clear the timeout condition and after StartWdt has started the countdown.

## GetWdtStatus

### Call Format

Declare Function GetWdtStatus Lib "PSB\_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)

## SetDOut

### Call Format

Declare Function SetDOut Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByVal Dout As Long) As

Long

### Return Value

Other than 0      Normal

0                      Error

### Argument

Selector As Long      Setting Item (pass by value)

PORT\_DOUT0              DOUT0

PORT\_DOUT1              DOUT1

PORT\_DOUT2              DOUT2

PORT\_DOUT3              DOUT3

Dout As Long              Output Status (pass by value)

OUTPUT\_OFF              Output OFF

OUTPUT\_ON                Output ON

### Processing

Sets the Common Output port (DOUT) output data.

### Example

Dim ret As Long

ret = SetDOut(PORT\_DOUT0, OUTPUT\_ON)

## GetDOut

### Call Format

Declare Function GetDOut Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByRef Dout As Long) As Long

### Return Value

Other than 0      Normal  
 0                      Error

### Argument

Selector As Long	Setting Item (pass by value)
	PORT_DOUT0              DOUT0
	PORT_DOUT1              DOUT1
	PORT_DOUT2              DOUT2
	PORT_DOUT3              DOUT3
Dout As Long	Output Status (pass by reference)
	OUTPUT_OFF              Output OFF
	OUTPUT_ON                Output ON

### Processing

Gets the Common Output port (DOUT) output data.

### Example

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

## GetDIn

### Call Format

Declare Function GetDIn Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByRef Din As Long) As Long

### Return Value

Other than 0      Normal  
 0                      Error

### Argument

Selector As Long	Setting Item (pass by value)
	PORT_DIN0                DIN0
	PORT_DIN1                DIN1
	PORT_DIN2                DIN2
	PORT_DIN3                DIN3
Din As Long	Input Status (pass by reference)
	INPUT_OFF                Input OFF
	INPUT_ON                 Input ON

### Processing

Gets the Input status of the designated port (DIN).

### Example

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

## ClearDIn

### Call Format

Declare Function ClearDIn Lib "PSB\_Ioc.dll" (ByVal Selector As Long) As Long

### Return Value

Other than 0      Normal

0                    Error

### Argument

Selector As Long      Setting Item (pass by value)

PORT\_DIN0            DIN0

PORT\_DIN1            DIN1

PORT\_DIN2            DIN2

PORT\_DIN3            DIN3

### Processing

Releases the Common Input port (DIN) input status.

### Example

Dim ret As Long

ret = ClearDIn(PORT\_DIN0)

## SetDInMask

### Call Format

Declare Function SetDInMask Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByVal Mask As Long) As Long

### Return Value

Other than 0      Normal

0                    Error

### Argument

Selector As Long      Setting Item (pass by value)

PORT\_DIN0            DIN0

PORT\_DIN1            DIN1

PORT\_DIN2            DIN2

PORT\_DIN3            DIN3

Mask As Long          Masking Information (pass by value)

MASK\_ON              Masking enabled

MASK\_OFF             Masking disabled

### Processing

Sets the Common Input Port (DIN) input status.

### Example

Dim ret As Long

ret = SetDInMask(PORT\_DIN0, MASK\_OFF)

## GetDInMask

### Call Format

Declare Function GetDInMask Lib "PSB\_Ioc.dll" (ByVal Selector As Long, ByRef Mask As Long)

As Long

### Return Value

Other than 0      Normal

0                    Error

### Argument

Selector As Long      Setting Item (pass by value)

PORT\_DIN0            DIN0

PORT\_DIN1            DIN1

PORT\_DIN2            DIN2

PORT\_DIN3            DIN3

Mask As Long          Masking Information (pass by reference)

MASK\_ON              Masking enabled

MASK\_OFF             Masking disabled

### Processing

Gets the masking information of the subject ports (DIN).

### Example

Dim ret As Long

Dim Mask As Long

ret = GetDInMask(PORT\_DIN0, Mask)

## SetResetMask

### Call Format

Declare Function SetResetMask Lib "PSB\_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\_ON              Masking enabled

MASK\_OFF             Masking disabled

### Processing

Sets the reset port's mask data.

### Example

Dim ret As Long

ret = SetResetMask(MASK\_OFF)

## GetResetMask

### Call Format

Declare Function GetResetMask Lib "PSB\_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_ON Masking enabled
	MASK_OFF Masking disabled

### Processing

Gets the current reset mask information.

### Example

Dim ret As Long

Dim Mask As Long

ret = GetResetMask(Mask)

## GetEvent

### Call Format

Declare Function GetEvent Lib "PSB\_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_CPU	CPU voltage
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_VOLT_VIT	CPU voltage2
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_TEMP_SYSTEM	System temperature
	EVENT_TEMP_CPU	CPU temperature
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	Watchdog Timer
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\_DIN0, Rasevent)

## ClearEvent

### Call Format

Declare Function ClearEvent Lib "PSB\_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_CPU	CPU voltage
	EVENT_VOLT_P33	+3.3V
	EVENT_VOLT_P50	+5V
	EVENT_VOLT_P12	+12V
	EVENT_VOLT_M12	-12V
	EVENT_VOLT_VIT	CPU voltage2
	EVENT_FAN_CPU	CPU fan
	EVENT_FAN_POWER	POWER fan
	EVENT_TEMP_SYSTEM	System temperature
	EVENT_TEMP_CPU	CPU temperature
	EVENT_DIN0	DIN0
	EVENT_DIN1	DIN1
	EVENT_DIN2	DIN2
	EVENT_DIN3	DIN3
	EVENT_WDT_TIMEOUT	Watchdog Timer

### Processing

Cancels the error event.

### Example

Dim ret As Long

ret = ClearEvent(EVENT\_DIN0)

## GetWdtTimeout

### Call Format

Declare Function GetWdtTimeout Lib "PSB\_Ioc.dll" (ByRef Timebuf As Long) As Long

### Return Value

Other than 0 Normal

0 Error

### Argument

Timebuf As Long	WDT timeout status (pass by reference)	
	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 Function ClearWdtTimeout Lib "PSB\_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 "PSB\_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 WDT timeout.

### Example

Dim ret As Long

ret = SetWdtResetMask(MASK\_ON)

## GetWdtResetMask

### Call Format

Declare Function GetWdtResetMask Lib "PSB\_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 WDT timeout's H/W reset mask data.

### Example

Dim ret As Long

Dim Mask As Long

ret = GetWdtResetMask(Mask)



## PsbDevWordWrite

### Call Format

Declare Function PsbDevWordWrite Lib "PSB\_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.

wData As Long Write data (0 to 65535)

### Processing

Writes to common memory.

### Example

' Writes data 255 to address 255.

Dim ret As Long

ret = PsbDevWordWrite(255, 255)

## PsbDevWordRead

### Call Format

Declare FunctionPsbDevWordRead Lib "PSB\_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

wData As Long Read data (0 to 65535)

### Processing

Reads from common memory.

### Example

' Reads address 255's data.

Dim ret As Long

Dim wData As Long

ret = PsbDevWordRead(255, wData)

## GetSmiDrvHandle

### Call Format

Declare Function GetSmiDrvHandle Lib "PSB\_Ioc.dll" () AS Long

### Return Value

0 Normal

Other than 0 Error

### Argument

None

### Processing

Gets the soft mirror device driver handle.

### Example

Dim ret As Long

ret = GetSmiDrvHandle()

### MEMO

Error (Return Value : Other than 0) will result if the soft mirror device driver is not operating.

## CloseSmiDrvHandle

### Call Format

Declare Function CloseSmiDrvHandle Lib "Psb\_Ioc.dll" () As Long

### Return Value

Other than 0 Normal

0 Error

### Argument

None

### Processing

Destroys the soft mirror device driver handle.

### Example

Dim ret As Long

ret = CloseSmiDrvHandle()

## GetSmiAryStatus

### Call Format

Declare Function GetSmiAryStatus Lib "Psb\_Ioc.dll"

### Return Value

Other than 0 Normal

0 Error

### Argument

Status As Long soft mirror's mirroring status (pass by reference)

### Processing

Get the soft mirror's mirroring status.

### Example

Dim ret As Long

Dim Status As Long

ret = GetSmiAryStatus(Status)

## GetSmiDevStatus

### Call Format

Declare Function GetSmiDevStatus Lib "Psb\_Ioc.dll"

(ByVal ID As Long, ByRef Type As Long, By Ref Status As Long) As Long

### Return Value

Other than 0      Normal

0                    Error

### Argument

ID As Long        Device ID (pass by value)

0 : Master HDD

1 : Slave HDD

Type As Long     Device Type (pass by reference)

ATADEVICE            ATA device

ATAPIDEVICE         CD-ROM

NODEVICE             Not connected

Status As Long   Device Status (pass by reference)

DEVSTAT\_GOOD        Normal

DEVSTAT\_NOTEXIST    Not connected

DEVSTAT\_BROKEN     Broken

### Processing

Gets the soft mirror device status.

### Example

Dim ret As Long

Dim Type As Long

Dim Status As Long

ret = GetSmiDrvStatus(0, Type, Status)