Digital
Human Machine Interface

Pro-face

PL-5910 Series Panel Computer

API Reference Manual

Table of Contents

Memo

1 Overview

The following information explains the Dynamic Link Libraries (API-DLL) used by the System Monitor/RAS feature, Remote RAS feature, and Backlight Control feature with a PL-5910 Series unit.

This chapter's explanation describes three types of API-DLLs; PL_Ioc.dll, PL_Ras.dll, and PL_BIoc.dll.

■ PL_Ioc.dll

PL_Ioc.dlls provide the interface for applications to access the System Monitor/RAS feature (System Monitor/RAS Device Driver). Applications can use DLLs to access the following types of features.

1. Driver Version information
2. System Monitor feature status
3. Read out (Get) various monitoring parameters (voltage, temperature)
4. System Monitor current data (voltage, temperature)
5. Watchdog parameters
6. Alarm processing
7. General input processing
8. Reset (of PL unit)
9. Event handling
10. Software mirroring *1 condition readout

■ PL_Ras.dll

The PL_Ras.dll provides access to the Remote RAS feature's single shared memory feature. Applications can use the PL_Ras.dll to access the following types of features.

1. Shared memory readout
2. Shared memory write

■ PL_BLIoc.dll

The PL_BLIoc.dll is used to control the backlight. Applications can use the PL_BLIoc.dll to access the following feature.

1. Backlight Control

**1 Optional item (under development).*

Memo

2 Usage Environment

■ Compatible Operating Systems

The API-DLLs contained on the PL unit's CD-ROM are compatible with the following OS types.

- Microsoft® WindowsNT®4.0 (Windows Service Pack 6a or higher)
- Microsoft® Windows®2000 (Windows Service Pack 4 or higher)

Also, each OS must use its corresponding System Monitor/RAS Device Driver.

■ Compatible Languages

- Microsoft® Visual C Ver. 6.0
- Microsoft® Visual C++ Ver. 6.0
- Microsoft® Visual Basic Ver. 6.0

Memo

3 Required Files

3.1 PL_Ioc Files

To use PL_Ioc.dll files, each language requires its own set of files.

■ Visual C

Filename	Description
PL_Iocif.h	Driver interface definition "include" file
PL_Ioc.LIB	Library definition file
PL_Ioc.dll	Dynamic link library file

■ Visual C++

Filename	Description
PL_Iocif.h	Driver interface definition "include" file
PL_Iocall.h	CPL_Iocall class definition "include" file
PL_Ioctl.h	CPL_Ioctl class definition "include" file
PL_Ioc.LIB	Library definition file
PL_Ioc.dll	Dynamic Link library file
PL_SmiIoctl.h	CPL_SmiIocrlclass definition "include" file (used only with software mirroring feature)
Sm.h	Software mirroring definition file (used only with software mirroring feature)

#include header files should be "included" in the following order.

#include PL_Iocif.h

#include PL_Ioctl.h

#include Sm.h ***(Only when the Software Mirroring feature*1 is used.)***

#include PL_SmiIoctl.h ***(Only when the Software Mirroring feature*1 is used.)***

PL_Iocall.h is automatically included, and does not need to be directly designated.

■ Visual Basic

Filename	Description
PL_Ioc.bas	Driver interface definition file
PL_Ioc.dll	Dynamic link library file

■ Dynamic Link Library (DLL)

In order for an application to use PL_Ioc.dll, it should be copied to the following folder.

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

**1 Optional item (under development).*

3.2 PL_Ras Files

To use PL_Ras.dll files, each language requires its own set of files.

■ Visual C

Filename	Description
PL_Ras.h	Driver interface definition "include" file
PL_Ras.LIB	Library definition file
PL_Ras.dll	Dynamic link library file

■ Visual C++

Filename	Description
PL_Ras.h	Driver interface definition "include" file
PL_Ras.LIB	Library definition file
PL_Ras.dll	Dynamic Link library file

■ Visual Basic

Filename	Description
PL_Ras.dll	Dynamic link library file
PL_Ras.bas	Driver Interface definition file

■ Dynamic Link Library (DLL)

In order for an application to use PL_Ras.dll, it should be copied to the following folder.

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

3.3 PL_BLIoc Files

To use PL_BLIoc.dll files, each language requires its own set of files.

■ Visual C

Filename	Description
PL_BLIocif.h	Driver interface definition "include" file
PL_BLIoc.LIB	Library definition file
PL_BLIoc.dll	Dynamic link library file

■ Visual C++

Filename	Description
PL_BLIocif.h	Driver interface definition "include" file
PL_BLIocall.h	CPL_Iocall class definition "include" file
PL_BLIoctl.h	CPL_Ioctl class definition "include" file
PL_BLIoc.LIB	Library definition file
PL_BLIoc.dll	Dynamic Link library file

#include header files should be "included" in the following order.

#include PL_BLIocif.h
#include PL_BLIoctl.h

PL_BLIocall.h is automatically included, and does not need to be directly designated.

■ Visual Basic

Filename	Description
PL_BLIoc.bas	Driver interface definition file
PL_BLIoc.dll	Dynamic link library file

■ Dynamic Link Library (DLL)

In order for an application to use PL_BLIoc.dll, it should be copied to the following folder.

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

Memo

4 Class Contents

4.1 CPL_Ioctl Class

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

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

4.2 CPL_local Class

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

However, since this class succeeds CPL_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 CPL_SmiIoctl Class

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

This class is only used when using the Software Mirroring driver.

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

4.4 CPL_BLIoctl Class

CPL_BLIoctl Class sets the parameters used for accessing the CPL_BLIoctl Class device drivers.

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

4.5 CPL_BLIocal Class

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

However, since this class succeeds CPL_BLIoctl, 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

5 Visual C Functions

5.1 Visual C Function List

5.1.1 PL_Ioc.dll Function List

Function Name	Description
InitIoctl	Creates the CPL_Ioctl object
EndIoctl	Destroys the CPL_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
GetTempParam	Gets the temperature monitoring parameter
GetCurrentTemp	Gets the current temperature value
GetEvent	Gets the error event
ClearEvent	Clears the error event
SetWdtCounter	Sets the watchdog timer counter
GetWdtCounter	Gets the watchdog timer counter
SetWdtMask	Sets warning masking in case of watchdog timer timeout
GetWdtMask	Gets warning masking in case of watchdog timer timeout
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
RunningWdt	Gets the watchdog timer operation status
SetWarningOut	Sets warning output
GetWarningOut	Gets warning output
GetUniversalIn	Gets universal input
ClearUniversalIn	Clears the universal input latched status
SetUniversalInMask	Sets universal input masking
GetUniversalInMask	Gets universal input masking
SetResetMask	Sets reset-masking
GetResetMask	Gets reset-masking
GetLightblowErr	Gets backlight burnout status
SetWdtResetMask	Sets reset-masking during watchdog timer timeout
GetWdtResetMask	Gets reset-masking during watchdog timer timeout
SetWarningDOUT	Sets the warning output DOUT
GetWarningDOUT	Gets the warning output DOUT
GetWdtTimeout	Gets the timeout status of the watchdog timer
ClearWdtTimeout	Clears the timeout status of the watchdog timer
GetSmiDrvHandle	Gets Software Mirroring driver handle
CloseSmiDrvHandle	Destroys Software Mirroring driver handle
GetSmiAryStatus	Gets status of Software Mirroring Array
GetSmiDevStatus	Gets status of Software Mirroring Device

5.1.2 PL_Ras.dll Function List

Function Name	Description
PIDevWordWrite	Writes to common memory.
PIDevWordRead	Reads from common memory.

5.1.3 PL_BLIoc.dll Function List

Function Name	Description
InitBLIoctl	Creates the CPL_Ioctl object
EndBLIoctl	Destroys the CPL_Ioctl object
GetBLDrvHandle	Gets the driver handle
CloseBLDrvHandle	Destroys the driver handle
GetBLDrvVersion	Gets the driver version
GetBLDrvVersionEx	Gets the hardware version and the driver version
SetBLControl	Sets the backlight status
GetBLControl	Gets the backlight status

5.2 Visual C Programming Notes

In order to use an API-DLL, you must first create the driver object and get the device handle. After using the API-DLL, you must destroy both the device handle and the driver object. Please refer to the following sample program when developing your own program.

When using only PlDevWordWrite and PlDevWordRead, creation and destruction of the driver object and the device driver handle is not required.

Sample Program

```
//API-DLL Example
//Variable language
BOOL bRet;
int iRet;
HANDLE hDrv;
//Create driver object and get driver handle
//Create CPL_IOctl object
InitIoctl();
iRet=GetDrvHandle(&hDrv)
•
•
//Output to DOUT
bRet=SetWarningDOUT(OUTPUT_ON);
•
•
//Application completion processing
//Destroy device handle and driver object
bRet=CloseDrvHandle();
EndIoctl();
```

5.3 Visual C Function Specifications (Details)

5.3.1 PL_Ioc.dll Function

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 the 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);

An error (Return Value: 1) occurs if the System Monitor/RAS Device Driver is not running.

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;
	//Destroys the handle
	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 (Major, 0 to 99).
	(I/O) int *pMinor Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's version information.
Example	BOOL ret;
	int Major, Minor;
	ret = GetDrvVersion(&Major, &Minor);

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 product version.
	(I/O) int *pMajor Pointer to version information (Major, 0 to 99).
	(I/O) int *pMinor Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's product and version information.
Example	BOOL ret;
	int Product, Major, Minor;
	ret = GetDrvVersionEx(&Product, &Major, &Minor);

If the PL-5910 Series unit's driver version is 1.00, then you will get

Product: 4 (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_VCOREA	CPU core voltage
		MONITOR_VOLT_VCOREB	CPU core voltage2
		MONITOR_VOLT_P33	+3.3 V
		MONITOR_VOLT_P50	+5.0 V
		MONITOR_VOLT_P12	+12 V
		MONITOR_VOLT_M50	-5.0 V
		MONITOR_VOLT_M12	-12 V
		MONITOR_TEMP_SYSTEM	System temperature
		MONITOR_TEMP_CPU	CPU temperature
	(I/O) int *pSetup	Pointer to Get Data	
		1: Enabled	
Processing:	Gets the current monitoring status (enabled).		
Example:	BOOL ret;		
	int Setup;		
	// Gets the CPU core voltage setup status.		
	ret = GetMonitorSetup(MONITOR_VOLT_VCOREA, &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_VCOREA CPU core voltage MONITOR_VOLT_VCOREB CPU core voltage2 MONITOR_VOLT_P33 +3.3 V MONITOR_VOLT_P50 +5.0 V MONITOR_VOLT_P12 +12 V MONITOR_VOLT_M50 -5.0 V MONITOR_VOLT_M12 -12 V (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;
	// Get the upper and lower-limit values of the CPU core
	// voltage.
	ret = GetVoltParam(MONITOR_VOLT_VCOREA, &ULimit,
	&LLimit);

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 WINAPI GetCurrentVolt(int Selector, int *pData)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters
	MONITOR_VOLT_VCOREA CPU core voltage
	MONITOR_VOLT_VCOREB CPU core voltage2
	MONITOR_VOLT_P33 +3.3 V
	MONITOR_VOLT_P50 +5.0 V
	MONITOR_VOLT_P12 +12 V
	MONITOR_VOLT_M50 -5.0 V
	MONITOR_VOLT_M12 -12 V
	(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_VCOREA, &Data);

Since the data taken 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

GetTempParam

Call Format	BOOL WINAPI GetTempParam(int Selector, int *pULimit)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters
	MONITOR_TEMP_SYSTEM System temperature
	MONITOR_TEMP_CPU CPU 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 system temperature upper-limit value.
	ret = GetTempParam(MONITOR_TEMP_SYSTEM, &ULimit);

GetCurrentTemp

Call Format	BOOL WINAPI GetCurrentTemp(int Selector, int *pData)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters
	MONITOR_TEMP_SYSTEM System temperature
	MONITOR_TEMP_CPU CPU 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 system temperature value.
	ret = GetCurrentTemp(MONITOR_TEMP_SYSTEM, &Data);

GetEvent

Call Format	BOOL WINAPI GetEvent(int Selector, int *pEvent)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters

MONITOR_VOLT_VCOREA	CPU core voltage
MONITOR_VOLT_VCOREB	CPU core voltage2
EVENT_VOLT_P33	+3.3 V
EVENT_VOLT_P50	+5.0 V
EVENT_VOLT_P12	+12 V
EVENT_VOLT_M50	-5.0 V
EVENT_VOLT_M12	-12 V
EVENT_TEMP_CPU	CPU temperature
EVENT_TEMP_SYSTEM	SYSTEM temperature
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	Backlight burnout

(I/O) int *pEvent Pointer to Error Event Information

ERROR_EVENT_OFF	No error event
ERROR_EVENT_ON	With error event

Processing	Checks the machine for voltage and temperature alarms, and the Universal Input information (event) and Watchdog Timeout information.

Example	BOOL ret;
	int Event;
	// Gets the error event information for the CPU core voltage.
	ret = GetEvent(EVENT_VOLT_VCOREA, &Event);

ClearEvent

Call Format	BOOL WINAPI ClearEvent(int Selector)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters used for cancelling error events

MONITOR_VOLT_VCOREA	CPU core voltage
MONITOR_VOLT_VCOREB	CPU core voltage2
EVENT_VOLT_P33	+3.3 V
EVENT_VOLT_P50	+5.0 V
EVENT_VOLT_P12	+12 V
EVENT_VOLT_M50	-5.0 V
EVENT_VOLT_M12	-12 V
EVENT_TEMP_CPU	CPU temperature
EVENT_TEMP_SYSTEM	SYSTEM temperature
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	Backlight burnout

Processing	Cancels the error event.
Example	BOOL ret;
	// Cancels the CPU core voltage error event.
	ret = ClearEvent(EVENT_VOLT_VCOREA);

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 watchdog timer's initial counter value to 10 seconds.
	Counter = 10;
	ret = SetWdtCounter(Counter);

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 (5 to 255) (Unit: Seconds)
Processing	Gets the current watchdog timer's initial counter value.
Example	BOOL ret; int Counter; ret = GetWdtCounter(&Counter);

SetWdtMask

Call Format	BOOL WINAPI SetWdtMask(int Selector, int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Setting Item	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets masking for the warning output used when watchdog timer time-out occurs.		

Example	BOOL ret;
	// Mask lamp output.
	ret = SetWdtMask(WARNING_LAMP, MASK_ON);
	// Release masking for alarm output.
	ret = SetWdtMask(WARNING_ALARM, MASK_OFF);		

GetWdtMask

Call Format	BOOL WINAPI GetWdtMask(int Selector, int *pMask)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Setting Item
	WARNING_ALARM ALARM
	WARNING_LAMP LAMP
	(I/O) int *pMask Pointer to Masking Information
	MASK_OFF Release Mask
	MASK_ON Set Mask
Processing	Gets the masking information used for warning output when watchdog timer time-out occurs.
Example	BOOL ret;
	int Mask;
	// Gets the masking information for the LAMP.
	ret = GetWdtMask(WARNING_LAMP, &Mask);
	// Gets the masking information for the alarm.
	ret = GetWdtMask(WARNING_ALARM, &Mask);

StartWdt

Call Format	BOOL WINAPI StartWdt (void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Starts watchdog timer countdown.
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	Restarts watchdog timer countdown after resetting to the initial value.
Example	BOOL ret; ret = RestartWdt();

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.

RunningWdt

Call Format	BOOL WINAPI RunningWdt(int *pRunFlag)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I/O) int *pRunFlag Pointer to Watchdog Timer Operation Status
	WATCHDOG_STOP Stopped
	WATCHDOG_COUNTDOWN Count down in progress
Processing	Gets the watchdog timer's operation status.
Example	BOOL ret; int RunFlag; ret = RunningWdt(&RunFlag);

SetWarningOut

Call Format	BOOL WINAPI SetWarningOut(int Selector, int WarnOut)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Setting Item	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
	(I) int WarnOut	Output Status	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Sets setting item warning information (LAMP or ALARM).		

Example

```
BOOL ret;
// Sets the LAMP output status to ON.
ret = SetWarningOut ( WARNING_LAMP, OUTPUT_ON );
// Sets the ALARM output status to OFF.
ret = SetWarningOut ( WARNING_ALARM, OUTPUT_OFF );
```

GetWarningOut

Call Format	BOOL WINAPI GetWarningOut (int Selector, int *pWarnOut)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Setting Item	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
	(I/O) int *pWarnOut	Pointer to Output Status	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Gets currently set item's warning status (LAMP or ALARM).		

Example

```
BOOL ret;
 int WarnOut;
// Gets the LAMP output status.
ret = GetWarningOut( WARNING_LAMP, &WarnOut );
// Gets the ALARM output status.
ret = GetWarningOut( WARNING_ALARM, &WarnOut );
```

GetUniversalIn

Call Format	BOOL WINAPI GetUniversalIn(int Selector, int *pUniIn)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I) int Selector	Designated Port
		PORT_UNI0 Universal Input 0
		PORT_UNI1 Universal Input 1
	(I/O) int *pUniIn	Pointer to Input Status
		INPUT_OFF Input OFF
		INPUT_ON Input ON
Processing	Gets the input status of the designated port (Universal Input 0, Universal Input 1).	
Example	BOOL ret;	
	int UniIn;	
	// Get the input status of Universal Input 0.	
	ret = GetUniversalIn(PORT_UNI0, &UniIn);	
	// Get the input status of Universal Input 1.	
	ret = GetUniversalIn(PORT_UNI1, &UniIn);	

ClearUniversalIn

Call Format	BOOL WINAPI ClearUniversalIn (int Selector)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I) int Selector	Designated Port
		PORT_UNI0 Universal Input 0
		PORT_UNI1 Universal Input 1
Processing	Cancels the input status of the designated port (Universal Input 0, Universal Input 1).	
Example	BOOL ret;	
	// Cancels the input status of Universal Input 0.	
	ret = ClearUniversalIn(PORT_UNI0);	
	// Cancels the input status of Universal Input 1.	
	ret = ClearUniversalIn(PORT_UNI1);	

SetUniversalInMask

Call Format	BOOL WINAPI SetUniversalInMask (int Selector, int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Designated Port	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the masking information for the designated port (Universal Input 0, Universal Input 1).		

Example	BOOL ret;
	// Release masking for Universal Input 0.
	ret = SetUniversalInMask(PORT_UNI0, MASK_OFF);
	// Mask Universal Input 1.
	ret = SetUniversalInMask(PORT_UNI1, MASK_ON);		

GetUniversalInMask

Call Format	BOOL WINAPI GetUniversalInMask(int Selector, int *pMask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Designated Port	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	(I/O) int *pMask	Pointer to Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the masking information for the designated port (Universal Input 0, Universal Input 1).		

Example	BOOL ret;
	int Mask;
	// Gets the masking information for Universal input 0.
	ret = GetUniversalInMask(PORT_UNI0, &Mask);
	// Gets the masking information for Universal input 1.
	ret = GetUniversalInMask(PORT_UNI1, &Mask);		

SetResetMask

Call Format	BOOL WINAPI SetResetMask(int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets reset-masking.		
Example	BOOL ret;		
	// Release reset-masking.		
	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_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the current reset-masking information.		
Example	BOOL ret;		
	int Mask;		
	ret = GetResetMask(&Mask);		

GetLightblowErr

Call Format	BOOL WINAPI GetLightblowErr (int *pLightErr)		
	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pLightErr	Error Information	
		BACKLIGHT_OK	Normal
		BACKLIGH_ERR	Error
Processing	Gets Backlight's current burnout error output.		
Example	BOOL ret;		
	int LightErr;		
	// Gets backlight's burnout condition.		
	ret = GetLightblowErr(&LightErr);		

SetWdtResetMask

Call Format	BOOL WINAPI SetWdtResetMask(int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Mask	Mask condition settings	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the reset mask status, depending on the watchdog timer.		
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	Mask condition settings	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the reset mask status, depending on the watchdog timer.		
Example	BOOL ret;		
	int Mask;		
	ret = GetWdtResetMask(&MASK);		

SetWarningDOUT

Call Format	BOOL WINAPI SetWarningDOUT(int WarningOut)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int WarningOut	Output status	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Sets DOUT warning status of current setting item.		
Example	BOOL ret;		
	// Sets warning DOUT output status to OFF.		
	ret = SetWarningDOUT(OUTPUT_OFF);		

GetWarningDOUT

Call Format	BOOL WINAPI GetWarningDOUT(int *pWarningOut)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I/O) int *pWarningOut	Pointer to Output Status
	OUTPUT_OFF	Output OFF
	OUTPUT_ON	Output ON
Processing	Gets DOUT warning status of current setting item.	
Example	BOOL ret;	
	int WarningOut;	
	// Gets DOUT Output status.	
	ret = GetWarningDOUT(&WarningOut);	

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	Not timeout
	TIMEOUT_ERROR	Timeout
Processing	Gets watchdog timeout status.	
Example	BOOL ret;	
	int Timebuf;	
	// Gets watchdog timeout status.	
	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;
	// Clears the watchdog timeout status.
	ret = GetWdtTimeout();

GetSmiDrvHandle

Call Format	BOOL WINAPI GetSmiDrvHandle (void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Gets Software Mirroring Device Driver Handle to communicate with the Software Mirroring Device Driver.
Example	BOOL ret; ret = GetSmiDrvHandle();

When the Software Mirroring Device Driver is not operating, an error (Return Value: FALSE) occurs.

CloseSmiDrvHandle

Call Format	BOOL WINAPI CloseSmiDrvHandle (void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Destroys the device driver handle created using the GetSmiDrvHandle function.

Example	BOOL ret;
	// Destroys the device driver handle created using the GetSmiDrvHandle function.
	ret = CloseSmiDrvHandle();

GetSmiAryStatus

Call Format	BOOL WINAPI GetSmiAryStatus (int *pStatus)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pStatus	Pointer to Software Mirroring Disk Status	
		ARYSTAT_GOOD	Good
		ARYSTAT_UNCOMFIG	Unconfigured
		ARYSTAT_REBUILD	Rebuilding
		ARYSTAT_REDUCE	Compressing
		ARYSTAT_DEAD	Dead
Processing	Gets Software Mirroring Status		

Example	BOOL ret;
	int Status;
	// Gets Software Mirroring Status.
	ret = GetSmiAryStatus (&Status);		

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	HDD (ATA DEVICE)
		ATAPIDEVICE	CD-ROM
		NODEVICE	No DEVICE
	(I/O) int *pStatus	Device Status	
		DEVSTAT_GOOD	Good
		DEVSTAT_NOTEXIST	No DEVICE
		DEVSTAT_BROKEN	BROKEN
Processing	Gets Software Mirroring Device Status		

Example

```
BOOL    ret;
int    Id, Type, Status;
// Gets the software mirroring device status
Id = 0;
ret = GetSmiDevStatus (Id , &Type , &Status);
```

5.3.2 PL_Ras.dll Function Details

PlDevWordWrite

Call Format	long PlDevWordWrite(long Addr, long wData)	
Return Value	0: Normal	
	Other than 0: Error	
Arguments	(I) long Addr	Write memory's 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 = PlDevWordWrite(255, 255);	

PlDevWordRead

Call Format	long PlDevWordRead(long Addr, long *wData)	
Return Value	0: Normal	
	Other than 0: Error	
Arguments	(I) long Addr	Write memory's word address
	(I/O) long *wData	Write data (0 to 65535)
Processing	Reads from common memory	
Example	//Reads data from address 255	
	long ret;	
	long wData;	
	ret = PlDevWordRead(255, &wData);	

5.3.3 PL_BLIoc.dll Function Details

InitBLIoctl

Call Format	void WINAPI InitBLIoctl(void)
Return Value	None
Arguments	None
Processing	Creates a CPL_BLIoctl object. The object is not destroyed until the EndBLIoctl function is called.
Example	InitBLIoctl();

EndBLIoctl

Call Format	void WINAPI EndBLIoctl(void)
Return Value	None
Arguments	None
Processing	Destroys the object created using the InitBLIoctl function.
Example	EndBLIoctl();

GetBLDrvHandle

Call Format	int WINAPI GetBLDrvHandle(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 = GetBLDrvHandle(&hndl);

An error (Return Value: 1) occurs if the System Monitor/RAS Device Driver is not running.

CloseBLDrvHandle

Call Format	BOOL WINAPI CloseBLDrvHandle(void)
Return Value	TRUE: Normal FALSE: Error
Arguments	None
Processing	Destroys the device driver handle created using the GetBLDrvHandle function.

Example	BOOL ret;
	//Destroys the handle
	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 (Major, 0 to 99).
	(I/O) int *pMinor Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's version information.
Example	BOOL ret;
	int Major, Minor;
	ret = GetBLDrvVersion(&Major, &Minor);

If the driver 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 product version.
	(I/O) int *pMajor Pointer to version information (Major, 0 to 99).
	(I/O) int *pMinor Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's version information.
Example	BOOL ret;
	int Product, Major, Minor;
	ret = GetBLDrvVersionEx(&Product, &Major, &Minor);

If the PL-5910 Series unit's driver version is 1.00, then you will get
Product: 4 (decimal)
Major: 1 (decimal)
Minor: 0 (decimal).

SetBLControl

Call Format	BOOL WINAPI SetBLControl(int BLFlag)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int BLFlag	Setting Parameter	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Sets the backlight ON/OFF.		
Example	BOOL ret;		
	//Sets the backlight control status.		
	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 backlight status	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Gets the backlight (ON/OFF).		
Example	BOOL ret;		
	//Gets the backlight control status.		
	ret = GetBLControl(&BLFlag);		

6 Visual C++ Function Details

6.1 Visual C++ Functions

6.1.1 PL_Ioc.dll Function List

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
GetTempParam	Gets the temperature monitoring parameter
GetCurrentTemp	Gets the current temperature value
GetEvent	Gets the error event
ClearEvent	Clears the error event
SetWdtCounter	Sets the watchdog timer counter
GetWdtCounter	Gets the watchdog timer counter
SetWdtMask	Sets warning masking in case of watchdog timer timeout
GetWdtMask	Gets warning masking in case of watchdog timer timeout
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
RunningWdt	Gets the watchdog timer operation status
SetWarningOut	Sets warning output
GetWarningOut	Gets warning output
GetUniversalIn	Gets universal input
ClearUniversalIn	Clears the universal input latched status
SetUniversalInMask	Sets universal input masking
GetUniversalInMask	Gets universal input masking
SetResetMask	Sets reset-masking
GetResetMask	Gets reset-masking
GetLightblowErr	Gets backlight burnout status
SetWdtResetMask	Sets reset-masking during watchdog timer timeout
GetWdtResetMask	Gets reset-masking during watchdog timer timeout
SetWarningDOUT	Sets the warning output DOUT
GetWarningDOUT	Gets the warning output DOUT
GetWdtTimeout	Gets the timeout status of the watchdog timer
ClearWdtTimeout	Clears the timeout status of the watchdog timer
GetSmiDrvHandle	Gets Software Mirroring driver handle
CloseSmiDrvHandle	Destroys Software Mirroring driver handle
GetSmiAryStatus	Gets status of Software Mirroring Array
GetSmiDevStatus	Gets status of Software Mirroring Device

6.1.2 PL_Ras.dll Function List

Function Name	Description
PIDevWordWrite	Writes to common memory.
PIDevWordRead	Reads from common memory.

6.1.3 PL_BLIoc.dll Function List

Function Name	Description
GetBLDrvHandle	Gets the driver handle
CloseBLDrvHandle	Destroys the driver handle
GetBLDrvVersion	Gets the driver version
GetBLDrvVersionEx	Gets the hardware version and the driver version
SetBLControl	Sets the backlight status
GetBLControl	Gets the backlight status

6.2 Visual C++ Programming Notes

In order to use an API-DLL, you must first create the driver object and get the device handle. After using the API-DLL, you must destroy both the device handle and the driver object. Please refer to the following sample program when developing your own program.

When using only PlDevWordWrite and PlDevWordRead, creation and destruction of the device driver handle is not required.

Sample Program

```
//API-DLL Example
//Variable language
BOOL bRet;
int iRet;
//Get device handle
CPLIoctl m_Ioc
iRet=m_Ioc.GetDrvHandle();
•
•
//Output to DOUT
bRet=m_Ioc.SetWarningDOUT(OUTPUT_ON);
•
•
//Destroy device handle
bRet=CloseDrvHandle();
```

6.3 Visual C Function Specifications (Details)

6.3.1 PL_Ioc.dll Function

GetDrvHandle

Call Format	int GetDrvHandle(HANDLE *pHndl)
Return Value	0: Normal
	1: Error
Arguments	(I/O) HANDLE *pHandl Pointer to the device driver handle
Processing	Gets the device driver handle to communicate with the device driver. The handle is stored into the member variable m_handle.
Example 1	CPL_Ioctl m_Ioc; int ret; ret = m_Ioc.GetDrvHandle();
Example 2	int ret; HANDLE hndl; ret = ::GetDrvHandle(&hndl);

An error (Return Value:1) 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.

Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	// Destroys the device driver handle.
	ret = m_Ioc.CloseDrvHandle();

Example 2	BOOL ret;
	// Destroys the device driver handle.
	ret = ::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 (Major, 0 to 99).
	(I/O) int *pMinor	Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's version information.	
Example 1	CPL_Ioctl m_Ioc;	
	BOOL ret;	
	int Major, Minor;	
	ret = m_Ioc.GetDrvVersion(&Major, &Minor);	
Example 2	BOOL ret;	
	int Major, Minor;	
	ret = ::GetDrvVersion(&Major, &Minor);	

If the driver 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 product version.
	(I/O) int *pMajor	Pointer to version information. (Major, 0 to 99).
	(I/O) int *pMinor	Pointer to version information. (Minor, 0 to 99).
Processing	Gets the driver's product and version information.	
Example1	BOOL ret;	
	int Product, Major, Minor;	
	ret = GetDrvVersionEx(&Product, &Major, &Minor);	
Example2	CPL_Ioctl m_Ioctl;	
	BOOL ret;	
	int Product, Major, Minor;	
	ret = m_Ioctl.GetDrvVersionEx(&Product, &Major, &Minor);	

If the PL-5910 Series unit's driver version is 1.00, then you will get
Product: 4 (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_VCOREA CPU core voltage
	MONITOR_VOLT_VCOREB CPU core voltage2
	MONITOR_VOLT_P33 +3.3 V
	MONITOR_VOLT_P50 +5.0 V
	MONITOR_VOLT_P12 +12 V
	MONITOR_VOLT_M50 -5.0 V
	MONITOR_VOLT_M12 -12 V
	MONITOR_TEMP_SYSTEM System temperature
	MONITOR_TEMP_CPU CPU temperature
	(I/O) int *pSetup Pointer to Get Data
	1: Enable
Processing	Gets the current monitoring enabled status.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int Setup;
	// Gets the CPU core voltage setup status.
	ret=m_Ioc.GetMonitorSetup(MONITOR_VOLT_VCOREA, &Setup);
Example 2	BOOL ret;
	int Setup;
	// Get the CPU core voltage setup status.
	ret = ::GetMonitorSetup(MONITOR_VOLT_VCOREA, &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_VCOREA CPU core voltage
	MONITOR_VOLT_VCOREB CPU core voltage2
	MONITOR_VOLT_P33 +3.3 V
	MONITOR_VOLT_P50 +5.0 V
	MONITOR_VOLT_P12 +12 V
	MONITOR_VOLT_M50 -5.0 V
	MONITOR_VOLT_M12 -12 V
	(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 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int ULimit, LLimit;
	// Get the upper and lower-limit values of the CPU core voltage.
	ret = m_Ioc.GetVoltParam(MONITOR_VOLT_VCOREA,
	&ULimit, &LLimit);
Example 2	BOOL ret;
	int ULimit, LLimit;
	// Get the upper and lower-limit values of the CPU core voltage.
	ret = ::GetVoltParam(MONITOR_VOLT_VCOREA, &ULimit, &LLimit);

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_VCOREA CPU core voltage
	MONITOR_VOLT_VCOREB CPU core voltage2
	MONITOR_VOLT_P33 +3.3 V
	MONITOR_VOLT_P50 +5.0 V
	MONITOR_VOLT_P12 +12 V
	MONITOR_VOLT_M50 -5.0 V
	MONITOR_VOLT_M12 -12 V
	(I/O) int *pData Pointer to the voltage value (Unit: mV)
Processing	Gets the current voltage value.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int Data;
	// Get the CPU core voltage value.
	ret=m_Ioc.GetCurrentVolt(MONITOR_VOLT_VCOREA, &Data);
Example 2	BOOL ret;
	int Data;
	// Get the CPU core voltage value.
	ret = ::GetCurrentVolt(MONITOR_VOLT_VCOREA, &Data);

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

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.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int ULimit;
	// Get the system temperature upper-limit value.
	ret = m_Ioc.GetTempParam(MONITOR_TEMP_SYSTEM, &ULimit);
Example 2	BOOL ret;
	int ULimit;
	ret=::GetTempParam(MONITOR_TEMP_SYSTEM,&ULimit);

GetCurrentTemp

Call Format	BOOL GetCurrentTemp(int Selector, int *pData)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters
	MONITOR_TEMP_SYSTEM System temperature
	MONITOR_TEMP_CPU CPU temperature
	(I/O) int *pData Pointer to the temperature
	(Unit: Degrees Celsius)
Processing	Gets the current temperature value.
Example 1	CPL_Ioctl m_Ioctl;
	BOOL ret;
	int Data;
	// Gets the system temperature value.
	ret = m_Ioctl.GetCurrentTemp(MONITOR_TEMP_SYSTEM, &Data);
Example 2	BOOL ret;
	int Data;
	// Gets the system temperature value.
	ret=::GetCurrentTemp(MONITOR_TEMP_SYSTEM,&Data);

GetEvent

Call Format	BOOL GetEvent(int Selector, int *pEvent)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I) int Selector Parameters	
	MONITOR_VOLT_VCOREA	CPU core voltage
	MONITOR_VOLT_VCOREB	CPU core voltage2
	EVENT_VOLT_P33	+3.3 V
	EVENT_VOLT_P50	+5.0 V
	EVENT_VOLT_P12	+12 V
	EVENT_VOLT_M50	-5.0 V
	EVENT_VOLT_M12	-12 V
	EVENT_TEMP_CPU	CPU temperature
	EVENT_TEMP_SYSTEM	SYSTEM temperature
	EVENT_UNI_IN0	Universal Input 0
	EVENT_UNI_IN1	Universal Input 1
	EVENT_WDT_TIMEOUT	Watchdog Timeout
	EVENT_LIGHT_BLOW	Backlight burnout
	(I/O) int *pEvent Pointer to Error Event Information	
	ERROR_EVENT_OFF	No error event
	ERROR_EVENT_ON	With error event
Processing	Checks the machine for voltage and temperature alarms, and the Universal Input information (event) and Watchdog Timeout information.	
Example 1	CPL_Ioctl m_Ioc;	
	BOOL ret;	
	int Event;	
	// Gets the error event information for the CPU core voltage.	
	ret = m_Ioc.GetEvent(EVENT_VOLT_VCOREA, &Event);	
Example 2	BOOL ret;	
	int Event;	
	// Gets the error event information for the CPU core voltage.	
	ret = ::GetEvent(EVENT_VOLT_VCOREA, &Event);	

ClearEvent

Call Format	BOOL ClearEvent(int Selector)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I) int Selector Parameters used for cancelling error events

MONITOR_VOLT_VCOREA	CPU core voltage
MONITOR_VOLT_VCOREB	CPU core voltage2
EVENT_VOLT_P33	+3.3 V
EVENT_VOLT_P50	+5.0 V
EVENT_VOLT_P12	+12 V
EVENT_VOLT_M50	-5.0 V
EVENT_VOLT_M12	-12 V
EVENT_TEMP_CPU	CPU temperature
EVENT_TEMP_SYSTEM	SYSTEM temperature
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	Backlight burnout

Processing: Cancels the error event.

Example 1:

```
CPL_Ioctl m_Ioc;
BOOL ret;
// Cancels the error event for the CPU core voltage.
ret = m_Ioc.ClearEvent( EVENT_VOLT_VCOREA );
```

Example 2:

```
BOOL ret;
// Cancels the error event for the CPU core voltage.
ret = ::ClearEvent( EVENT_VOLT_VCOREA );
```

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 to 255) (Unit: Seconds)
Processing	Sets the current watchdog timer's initial counter value.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	// Sets the watchdog timer's initial counter value to 10 seconds.
	Counter = 10;
	ret = m_Ioc.SetWdtCounter(Counter);
Example 2	BOOL ret;
	int Counter;
	// Sets the watchdog timer's initial counter value to 10 seconds.
	Counter = 10;
	ret = ::SetWdtCounter(Counter);

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 (5 to 255) (Unit: Seconds)
Processing	Gets the current watchdog timer's initial counter value.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int Counter;
	ret = m_Ioc.GetWdtCounter(&Counter);
Example 2	BOOL ret;
	int Counter;
	ret = ::GetWdtCounter(&Counter);

SetWdtMask

Call Format	BOOL SetWdtMask(int Selector, int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Setting Item	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets masking for the warning that is output when watchdog timer time-out occurs.		

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
//Mask LAMP output.
ret = m_Ioc.SetWdtMask( WARNING_LAMP, MASK_ON );
// Disable masking for ALARM output.
ret = m_Ioc.SetWdtMask( WARNING_ALARM, MASK_OFF );
```

Example 2

```
BOOL ret;
// Mask LAMP output.
ret = ::SetWdtMask( WARNING_LAMP, MASK_ON );
// Disable masking for ALARM output.
ret = ::SetWdtMask( WARNING_ALARM, MASK_OFF );
```

GetWdtMask

Call Format	BOOL GetWdtMask(int Selector, int *pMask)
Return Value	TRUE: Normal FALSE: Error
Arguments	(I) int Selector Setting Item WARNING_ALARM ALARM WARNING_LAMP LAMP (I/O) int *pMask Pointer to Masking Information MASK_OFF Release Mask MASK_ON Set Mask
Processing	Gets the masking information for warning output that is created when a watchdog timer time-out occurs.

Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	int Mask;
	// Gets the LAMP masking information.
	ret = m_Ioc.GetWdtMask(WARNING_LAMP, &Mask);
	// Get the ALARM masking information.
	ret = m_Ioc.GetWdtMask(WARNING_ALARM, &Mask);

Example 2	BOOL ret;
	int Mask;
	// Gets the LAMP masking information.
	ret = ::GetWdtMask(WARNING_LAMP, &Mask);
	// Get the ALARM masking information.
	ret = ::GetWdtMask(WARNING_ALARM, &Mask);

StartWdt

Call Format	BOOL StartWdt(void)
Return Value	TRUE: Normal FALSE: Error
Arguments	None
Processing	Starts watchdog timer countdown.
Example 1	CPL_Ioctl m_Ioc; BOOL ret; ret = m_Ioc.StartWdt();
Example 2	BOOL ret; ret = ::StartWdt();

StopWdt

Call Format	BOOL StopWdt(void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Stops watchdog timer countdown.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	ret = m_Ioc.StopWdt();
Example 2	BOOL ret;
	ret = ::StopWdt();

RestartWdt

Call Format	BOOL RestartWdt(void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Restarts watchdog timer countdown after resetting to the initialvalue.
Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	m_Ioc.RestartWdt();
Example 2	BOOL ret;
	ret = ::RestartWdt();

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.

RunningWdt

Call Format	BOOL RunningWdt(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.	

Example 1

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

Example 2

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

SetWarningOut

Call Format	BOOL SetWarningOut(int Selector, int WarnOut)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Setting Item	
		WARNING_ALARM	ALARM
		WARNING_LAMP	LAMP
	(I) int WarnOut	Output Status	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Sets the warning information for the set item (LAMP or ALARM).		

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
// Sets the LAMP output status to ON.
ret = m_Ioc.SetWarningOut( WARNING_LAMP, OUTPUT_ON );
// Sets the ALARM output status to OFF.
ret = m_Ioc.SetWarningOut( WARNING_ALARM, OUTPUT_OFF );
```

Example 2

```
BOOL ret;
// Sets the LAMP output status to ON.
ret = ::SetWarningOut( WARNING_LAMP, OUTPUT_ON );
// Sets the ALARM output status to OFF.
ret = ::SetWarningOut( WARNING_ALARM, OUTPUT_OFF );
```

GetWarningOut

Call Format	BOOL GetWarningOut(int Selector, int *pWarnOut)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I) int Selector	Settting Item
		WARNING_ALARM ALARM
		WARNING_LAMP LAMP
	(I/O) int *pWarnOut	Pointer to Output Status
		OUTPUT_OFF Output OFF
		OUTPUT_ON Output ON
Processing	Gets the warning status of the current set item (LAMP or ALARM).	

Example 1

```
CPL_Ioctl  m_Ioc;
BOOL  ret;
int  WarnOut;
// Gets the LAMP output status.
ret = m_Ioc.GetWarningOut( WARNING_LAMP, &WarnOut );
// Gets the ALARM output status.
ret = m_Ioc.GetWarningOut( WARNING_ALARM, &WarnOut );
```

Example 2

```
BOOL  ret;
int  WarnOut;
// Gets the LAMP output status.
ret = ::GetWarningOut( WARNING_LAMP, &WarnOut );
// Gets the ALARM output status.
ret = ::GetWarningOut( WARNING_ALARM, &WarnOut );
```

GetUniversalIn

Call Format	BOOL GetUniversalIn(int Selector, int *pUniIn)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Designated Port	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	(I/O) int *pUniIn	Pointer to Input Status	
		INPUT_OFF	Input OFF
		INPUT_ON	Input ON
Processing	Gets the input status of the designated port (Universal Input 0, Universal Input 1).		

Example1

```
CPL_Ioctl m_Ioc;
BOOL ret;
int UniIn;
// Gets the input status of Universal Input 0.
ret = m_Ioc.GetUniversalIn( PORT_UNI0, &UniIn );
// Gets the input status of Universal Input 1.
ret = m_Ioc.GetUniversalIn( PORT_UNI1, &UniIn );
```

Example 2

```
BOOL ret;
int UniIn;
// Gets the input status of Universal Input 0.
ret = ::GetUniversalIn( PORT_UNI0, &UniIn );
// Gets the input status of Universal Input 1.
ret = ::GetUniversalIn( PORT_UNI1, &UniIn );
```

ClearUniversalIn

Call Format	BOOL ClearUniversalIn(int Selector)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I) int Selector	Designated Port
		PORT_UNI0 Universal Input 0
		PORT_UNI1 Universal Input 1
Processing	Clears the input status of the designated port (Universal Input 0, Universal Input 1).	

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
// Cancels the output of Universal Input 0.
ret = m_Ioc.ClearUniversalIn( PORT_UNI0 );
// Cancels the output of Universal Input 1.
ret = m_Ioc.ClearUniversalIn( PORT_UNI1 );
```

Example 2

```
BOOL ret;
// Cancels the output of Universal Input 0.
ret = ::ClearUniversalIn( PORT_UNI0 );
// Cancels the output of Universal Input 1.
ret = ::ClearUniversalIn( PORT_UNI1 );
```

SetUniversalInMask

Call Format	BOOL SetUniversalInMask(int Selector, int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Designated Port	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the masking information for the designated port (Universal Input 0, Universal Input 1).		

Example 1

```
CPL_Ioctl  m_Ioc;
BOOL  ret;
// Release masking for Universal Input 0.
ret = m_Ioc.SetUniversalInMask( PORT_UNI0, MASK_OFF );
// Mask Universal Input 1.
ret = m_Ioc.SetUniversalInMask( PORT_UNI1, MASK_ON );
```

Example 2

```
BOOL  ret;
// Release masking for Universal Input 0.
ret = ::SetUniversalInMask( PORT_UNI0, MASK_OFF );
// Mask Universal Input 1.
ret = ::SetUniversalInMask( PORT_UNI1, MASK_ON );
```

GetUniversalInMask

Call Format	BOOL GetUniversalInMask(int Selector, int *pMask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Selector	Designated Port	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	(I/O) int *pMask	Pointer to Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the masking information for the designated port (Universal Input 0, Universal Input 1).		

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
int Mask;
// Gets the masking information for Universal input 0.
ret = m_Ioc.GetUniversalInMask( PORT_UNI0, &Mask );
// Gets the masking information for Universal input 1.
ret = m_Ioc.GetUniversalInMask( PORT_UNI1, &Mask );
```

Example 2

```
BOOL ret;
int Mask;
// Gets the masking information for Universal input 0.
ret = ::GetUniversalInMask( PORT_UNI0, &Mask );
// Gets the masking information for Universal input 1.
ret = ::GetUniversalInMask( PORT_UNI1, &Mask );
```

SetResetMask

Call Format	BOOL SetResetMask(int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Mask	Masking Information	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets reset-masking.		
Example 1	CPL_Ioctl m_Ioc;		
	BOOL ret;		
	// Disable reset-masking.		
	ret = m_Ioc.SetResetMask(MASK_OFF);		
Example 2	BOOL ret;		
	// Disable reset-masking.		
	ret = ::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_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the current reset-masking information.		
Example 1	CPL_Ioctl m_Ioc;		
	BOOL ret;		
	int Mask;		
	ret = m_Ioc.GetResetMask(&Mask);		
Example 2	BOOL ret;		
	int Mask;		
	ret = ::GetResetMask(&Mask);		

GetLightblowErr

Call Format	BOOL GetLightblowErr(int *pLightErr)		
	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pLightErr	Pointer to Error Information	
		BACKLIGHT_OK	Normal
		BACKLIGH_ERR	Error
Processing	Gets Backlight's current burnout error output.		

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
int LightErr;
// Gets backlight's burnout condition.
ret = m_Ioc.GetLightblowErr( &LightErr );
```

Example 2

```
BOOL ret;
int LightErr;
// Gets backlight's burnout condition.
ret = ::GetLightblowErr( &LightErr );
```

SetWdtResetMask

Call Format	BOOL SetWdtResetMask(int Mask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int Mask	Mask condition settings	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the reset mask status, depending on the watchdog timer.		

Example 1

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

Example 2

```
CPL_SmiIoctl m_SmiIoctl;
BOOL ret;
ret = m_SmiIoctl.SetWdtResetMask(MASK_ON);
```

GetWdtResetMask

Call Format	BOOL GetWdtResetMask(int *pMask)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pMask	Pointer to Mask status	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the reset mask status, depending on the watchdog timer.		
Example 1	BOOL ret;		
	int Mask;		
	ret = ::GetWdtResetMask(&Mask);		
Example 2	CPL_Ioctl m_Ioctl;		
	BOOL ret;		
	int Mask;		
	ret = m_Ioctl.GetWdtResetMask(&Mask);		

SetWarningDOUT

Call Format	BOOL SetWarningDOUT(int WarningOut)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int WarningOut	Output status	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Sets error status of DOUT.		
Example 1	CPL_Ioctl m_Ioc;		
	BOOL ret;		
	// Sets DOUT output status to OFF.		
	ret = m_Ioc.SetWarningDOUT(OUTPUT_OFF);		
Example 2	BOOL ret;		
	// Sets DOUT output status to OFF.		
	ret = ::SetWarningDOUT(OUTPUT_OFF);		

GetWarningDOUT

Call Format	BOOL GetWarningDOUT(int *pWarningOUT)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I/O) int *pWarningOut	Pointer to Output Status
		OUTPUT_OFF Output OFF
		OUTPUT_ON Output ON
Processing	Gets alarm status of DOUT.	

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
int WarningOut;
// Gets alarm status of DOUT.
ret = m_Ioc.GetWarningDOUT( &WarningOut );
```

Example 2

```
BOOL ret;
int WarningOut;
// Gets alarm status of DOUT.
ret = ::GetWarningDOUT( &WarningOut );
```

GetWdtTimeout

Call Format	BOOL GetWdtTimeout(int *pTimebuf)	
Return Value	TRUE: Normal	
	FALSE: Error	
Arguments	(I/O) int *pTimebuf	Pointer to Watchdog Status
		TIMEOUT_OK Not timeout
		TIMEOUT_ERROR Now timeout
Processing	Gets watchdog timeout status.	

Example 1

```
CPL_Ioctl m_Ioc;
BOOL ret;
int Timebuf;
// Gets watchdog timeout status.
ret = m_Ioc.GetWdtTimeout( &Timebuf );
```

Example 2

```
BOOL ret;
int Timebuf;
// Gets watchdog timeout status.
ret = ::GetWdtTimeout( &Timebuf );
```

ClearWdtTimeout

Call Format	BOOL ClearWdtTimeout (void)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	None
Processing	Clears the watchdog timeout status.

Example 1	CPL_Ioctl m_Ioc;
	BOOL ret;
	// Clears the watchdog timeout status.
	ret=m_Ioc.ClearWdtTimeout();

Example 2	BOOL ret;
	// Clears the watchdog timeout status.
	ret=::ClearWdtTimeout();

GetSmiDrvHandle

Call Format	int GetSmiDrvHandle (void)
Return Value	0: Normal
	1 : Error
Arguments	None
Processing	Gets device driver handle for communication with Software Mirroring device driver.

Example 1	CPL_SmiIoctl m_SmiIoc;
	BOOL ret;
	// Gets Software Mirroring driver handle.
	ret=m_SmiIoc.GetSmiDrvHandle();

Example 2	BOOL ret;
	// Gets Software Mirroring driver handle.
	ret=::GetSmiDrvHandle();

When the Software Mirroring Driver is not loaded, an error (Return Value: 1) is returned.

CloseSmiDrvHandle

Call Format	BOOL CloseSmiDrvHandle (void)
Return Value	True: Normal
	False: Error
Arguments	None
Processing	Destroys handle created in GetSmiDrvHandle.
Example 1	CPL_SmiIoctl m_SmiIoc;
	BOOL ret;
	// Destroys Software Mirroring driver handle.
	ret = m_SmiIoc.CloseSmiDrvHandle();
Example 2	BOOL ret;
	// Destroys Software Mirroring driver handle.
	ret = ::CloseSmiDrvHandle();

GetSmiAryStatus

Call Format	BOOL GetSmiAryStatus(int *pStatus)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pStatus	Pointer to Mirroring Status	
		ARYSTAT_GOOD	Good
		ARYSTAT_UNCONFIG	Unconfigured
		ARYSTAT_REBUILD	Rebuilding
		ARYSTAT_REDUCE	Reduced
		ARYSTAT_DEAD	Dead
Processing	Gets Software Mirroring status.		
Example 1	CPL_SmiIoctl m_Smiloc;		
	BOOL ret;		
	int Status;		
	// Gets Software Mirroring status.		
	ret = m_Smiloc.GetSmiAryStatus(&Status);		
Example 2	BOOL ret;		
	int Status;		
	// Gets Software Mirroring status.		
	ret = ::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	Device Type	
		ATADEVICE	HDD (ATA DEVICE)
		ATAPIDEVICE	CD-ROM
		NODEVICE	No DEVICE
	(I/O) int* pStatus	Device Status	
		DEVSTAT_GOOD	Good
		DEVSTAT_NOTEXIST	No DEVICE
		DEVSTAT_BROKEN	BROKEN
Processing	Gets Device Status of software mirroring.		

Example 1

```
CPL_SmiIoctl    m_SmiIoc;
BOOL  ret;
int  Id, Type, Status;
// Gets device status.
Id = 0;
ret = m_SmiIoc.GetSmiDevStatus( Id, &Type, &Status );
```

Example 2

```
BOOL   ret;
int  Id, Type, Status;
// Gets device status.
Id = 0;
ret = ::GetSmiDevStatus( Id, &Type, &Status);
```

6.3.2 PL_Ras.dll Functions

PlDevWordWrite

Call Format	long PlDevWordWrite(long Addr, long wData)	
Return Value	0: Normal	
	Other than 0: Error	
Arguments	(I) long Addr	Write memory's word address (0 to 255)
	(I) long wData	Write data (0 to 65535)
Processing	Writes to common memory	
Example	//Writes data 255 to address 255	
	long ret;	
	ret = PlDevWordWrite(255, 255);	

PlDevWordRead

Call Format	long PlDevWordRead(long Addr, long *wData)	
Return Value	0: Normal	
	Other than 0: Error	
Arguments	(I) long Addr	Write memory's word address
	(I/O) long *wData	Write data (0 to 65535)
Processing	Reads from common memory	
Example	//Reads data from address 255	
	long ret;	
	long wData;	
	ret = PlDevWordRead(255, &wData);	

6.3.3 PL_BLIoc.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 the device driver.
Example 1	CPL_BLIoctl m_BLIoc; m_BLIoc. GetBLDrvHandle()
Example 2	int ret; HANDLE hndl; ret = ::GetBLDrvHandle(&hndl);

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

CloseBLDrvHandle

Call Format	BOOL CloseBLDrvHandle()
Return Value	TRUE: Normal FALSE: Error
Arguments	None
Processing	Destroys the device driver handle created using the GetBLDrvHandle function.
Example 1	BOOL ret; ret = ::CloseBLDrvHandle();
Example 2	CPL_BLIoctl m_BLIoctl; BOOL ret; ret = m_BLIoctl.CloseBLDrvHandle();

GetBLDrvVersion

Call Format	BOOL GetBLDrvVersion(int *pMajor, int *pMinor)
Return Value	TRUE: Normal
	FALSE: Error
Arguments	(I/O) int *pMajor Pointer to version information (Major, 0 to 99).
	(I/O) int *pMinor Pointer to version information (Minor, 0 to 99).
Processing	Gets the driver's version information.
Example 1	CPL_BLIoctl m_BLIoc;
	BOOL ret;
	int Major, Minor;
	ret = m_BLIoc.GetBLDrvVersion(&Major, &Minor);
Example 2	BOOL ret;
	int Major, Minor;
	ret = ::GetBLDrvVersion(&Major, &Minor);

If the driver 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 product version.
	(I/O) int *pMajor Pointer to version information. (Major, 0 to 99)
	(I/O) int *pMinor Pointer to version information. (Minor, 0 to 99)
Processing	Gets the driver's version information.
Example 1	CPL_BLIoctl m_BLIoc;
	BOOL ret;
	int Product, Major, Minor;
	ret=m_BLIoctl.GetBLDrvVersionEx(&Product, &Major, &Minor);
Example 2	BOOL ret;
	int Product, Major, Minor;
	ret = ::GetBLDrvVersionEx(&Product, &Major, &Minor);

If the PL-5910 Series unit's driver version is 1.00, then you will get

Product: 4 (decimal)
Major: 1 (decimal)
Minor: 10 (decimal).

SetBLControl

Call Format	BOOL SetBLControl (int BLFlag)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I) int BLFlag	Parameter	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Sets the backlight ON/OFF.		

Example 1

```
CPL_BLIoc m_BLIoc;
BOOL      ret;
// Sets backlight control ON
ret = m_BLIoc.SetBLControl( BACKLIGHT_ON );
```

Example 2

```
BOOL      ret;
// Sets backlight control ON
ret = ::SetBLControl( BACKLIGHT_ON );
```

GetBLControl

Call Format	BOOL GetBLControl (int *pBLFlag)		
Return Value	TRUE: Normal		
	FALSE: Error		
Arguments	(I/O) int *pBLFlag	Parameter	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Gets the backlight control status.		

Example 1

```
CPL_BLIoc m_BLIoc;
BOOL    ret;
int     BLFlag;
// Gets backlight control status
ret = m_BLIoc.GetBLControl( &BLFlag );
```

Example 2

```
BOOL    ret;
int     BLFlag;
// Gets backlight control status
ret = ::GetBLControl( &BLFlag );
```

Memo

7 Visual Basic Function Specifications

7.1 Visual Basic Functions

7.1.1 PL_Ioc.dll Function List

Function Name	Description
InitIoctl	Creates the CPL_Ioctl object
EndIoctl	Destroys the CPL_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
GetTempParam	Gets the temperature monitoring parameter
GetCurrentTemp	Gets the current temperature value
GetEvent	Gets the error event
ClearEvent	Clears the error event
SetWdtCounter	Sets the watchdog timer counter
GetWdtCounter	Gets the watchdog timer counter
SetWdtMask	Sets warning masking in case of watchdog timer timeout
GetWdtMask	Gets warning masking in case of watchdog timer timeout
StartWdt	Starts the watchdog timer
StopWdt	Stops the watchdog timer
RestartWdt	Restarts the watchdog timer
RunningWdt	Gets the watchdog timer operation status
SetWarningOut	Sets warning output
GetWarningOut	Gets warning output
GetUniversalIn	Gets universal input
ClearUniversalIn	Clears the universal input latched status
SetUniversalInMask	Sets universal input masking
GetUniversalInMask	Gets universal input masking
SetResetMask	Sets reset-masking
GetResetMask	Gets reset-masking
GetLightblowErr	Gets backlight burnout status
GetWdtTimeout	Gets the timeout status of the watchdog timer
ClearWdtTimeout	Clears the timeout status of the watchdog timer
SetWarningDOUT	Sets the warning output DOUT
GetWarningDOUT	Gets the warning output DOUT
GetSmiDrvHandle	Gets Software Mirroring driver handle
CloseSmiDrvHandle	Destroys Software Mirroring driver handle
GetSmiAryStatus	Gets status of Software Mirroring Array
GetSmiDevStatus	Gets status of Software Mirroring Device

7.1.2 PL_Ras.dll Function List

Function Name	Description
PIDevWordWrite	Writes to common memory.
PIDevWordRead	Reads from common memory.

7.1.3 PL_BLIoc.dll Function List

Function Name	Description
InitBLIoctl	Creates the CPL_Ioctl object
EndBLIoctl	Destroys the CPL_Ioctl object
GetBLDrvHandle	Gets the driver handle
CloseBLDrvHandle	Destroys the driver handle
GetBLDrvVersion	Gets the driver version
GetBLDrvVersionEx	Gets the hardware version and the driver version
SetBLControl	Sets the backlight status
GetBLControl	Gets the backlight status

7.2 Visual Basic Programming Notes

In order to use an API-DLL, you must first create the driver object and get the device handle. After using the API-DLL, you must destroy both the device handle and the driver object. Please refer to the following sample program when developing your own program.

When using only PlDevWordWrite and PlDevWordRead, creation and destruction of the driver object and the device driver handle is not required.

Sample Program

```
'API-DLL Example
'Create driver object
Call InitIoctl
'Create device handle
Dim ret As Long
Dim Hndl As Long
ret=GetDrvHandle(Hndl)
•
•
'Output to DOUT
Dim ret As Long
ret=SetWarningDOUT(OUTPUT_ON)
•
•
'Application completion processing
'Destroy device handle
Dim ret As Long
ret=CloseDrvHandle();
'Destroy driver object
Call EndIoctl
```

7.3 Visual Basic Function Specifications (Details)

7.3.1 PL_Ioc.dll Function

InitIoctl

Call format	Declare Sub InitIoctl Lib "PL_Ioc.dll" ()
Return value	None
Argument	None
Processing	Creates a CPL_Ioctl object. The created object will not be released until the "EndIoctl" function is called.
Example	InitIoctl()

EndIoctl

Call format	Declare Sub EndIoctl Lib "PL_Ioc.dll" ()
Return value	None
Argument	None
Processing	Destroys the object created with the "InitIoctl" function.
Example	EndIoctl()

GetDrvHandle

Call format	Declare Function GetDrvHandle Lib "PL_Ioc.dll" (ByRef hndl As Long) As Long
Return value	0: Normal 1: 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)

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

CloseDrvHandle

Call format	Declare Function CloseDrvHandle Lib "PL_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 'Destroy handle ret = CloseDrvHandle()

GetDrvVersion

Call format	Declare Function GetDrvVersion Lib "PL_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 (Major, 0 to 99) (pass by reference)
	Minor As Long	Version data (Minor, 0 to 99) (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)	

When the driver version is 1.00,
Major:1 (Decimal)
Minor:00 (Decimal).

GetDrvVersionEx

Call format	Declare Function GetDrvVersionEx Lib "PL_Ioc.dll"	
	(By Ref Product As Long, By Ref Major As Long, By Ref Minor As Long) As Long	
Return value	Other than 0: Normal	
	0: Error	
Argument	Product As Long	Unit data (pass by reference)
	Major As Long	Version data (Major, 0 to 99) (pass by reference)
	Minor As Long	Version data (Minor, 0 to 99) (pass by reference)
Processing	Gets the driver version.	
Example	Dim ret As Long	
	Dim Product As Long	
	Dim Major As Long	
	Dim Minor As Long	
	ret = GetDrvVersionEx(Product, Major, Minor)	

When the PL-5910 Series unit's driver version is 1.00,
Product:4 (Decimal)
Major:1 (Decimal)
Minor:00 (Decimal).

GetMonitorSetup

Call format	Declare Function GetMonitorSetup Lib "PL_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_VCOREA	CPU core voltage
	MONITOR_VOLT_VCOREB	CPU core voltage2
	MONITOR_VOLT_P33	+3.3 V
	MONITOR_VOLT_P50	+5.0 V
	MONITOR_VOLT_P12	+12 V
	MONITOR_VOLT_M50	-5.0 V
	MONITOR_VOLT_M12	-12 V
	MONITOR_TEMP_SYSTEM	System temperature
	MONITOR_TEMP_CPU	CPU temperature
	Setup As Long	Get data (pass by reference)
		1:Enable
Processing	Gets the current enabled monitor status.	
Example	Dim ret As Long	
	Dim Setup As Long	
	' Get the setup status of the CPU core voltage	
	ret = GetMonitorSetup(MONITOR_VOLT_VCOREA, Setup)	

GetVoltParam

Call format	Declare Function GetVoltParam Lib "PL_Ioc.dll" (ByVal Selector As Long, ByRef ULimit 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_VOLT_VCOREA CPU core voltage MONITOR_VOLT_VCOREB CPU core voltage2 MONITOR_VOLT_P33 +3.3 V MONITOR_VOLT_P50 +5.0 V MONITOR_VOLT_P12 +12 V MONITOR_VOLT_M50 -5.0 V MONITOR_VOLT_M12 -12 V Ulimit As Long Voltage value upper limit (unit: mV) (pass by reference) LLimitAsLong 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_VCOREA, ULimit, LLimit)

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 "PL_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_VCOREA CPU core voltage
	MONITOR_VOLT_VCOREB CPU core voltage2
	MONITOR_VOLT_P33 +3.3 V
	MONITOR_VOLT_P50 +5.0 V
	MONITOR_VOLT_P12 +12 V
	MONITOR_VOLT_M50 -5.0 V
	MONITOR_VOLT_M12 -12 V
	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_VCOREA, Data)

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

GetTempParam

Call format	Declare Function GetTempParam Lib "PL_Ioc.dll"
	(ByVal Selector As Long, ByRef ULimit As Long) As Long
Return value	Other than 0: Normal
	0: Error
Argument	Selector As Long Get parameter (pass by value)
	MONITOR_TEMP_SYSTEM SYSTEM temp.
	MONITOR_TEMP_CPU CPU temp.
	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 SYSTEM temperature
	ret = GetTempParam(MONITOR_TEMP_SYSTEM, ULimit)

GetCurrentTemp

Call format	Declare Function GetCurrentTemp Lib "PL_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_SYSTEM	SYSTEM temp.
	MONITOR_TEMP_CPU	CPU temp.
	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 SYSTEM temperature	
	ret = GetCurrentTemp(MONITOR_TEMP_SYSTEM, Data)	

GetEvent

Call format	Declare Function GetEvent Lib "PL_Ioc.dll"
	(ByVal Selector As Long, ByRef Event As Long) As Long
Return value	Other than 0: Normal
	0: Error
Argument	Selector As Long Parameters (pass by value)
	EVENT_VOLT_VCOREA CPU core voltage
	EVENT_VOLT_VCOREB CPU core voltage2
	EVENT_VOLT_P33 +3.3 V
	EVENT_VOLT_P50 +5.0 V
	EVENT_VOLT_P12 +12 V
	EVENT_VOLT_M50 -5.0 V
	EVENT_VOLT_M12 -12 V
	EVENT_TEMP_SYSTEM System temperature
	EVENT_TEMP_CPU CPU temperature
	EVENT_UNI_IN0 Universal Input 0
	EVENT_UNI_IN1 Universal Input 1
	EVENT_WDT_TIMEOUT Watchdog Timeout
	EVENT_LIGHT_BLOW Backlightburnout
	Event As Long Error event data (pass by reference)
	ERROR_EVENT_OFF No error event
	ERROR_EVENT_ON Error event
Processing	Checks for the irregularities in the machine voltage, temperature, Universal Input function (event) data, and WatchDog Timeout data.
Example	Dim ret As Long
	Dim Event As Long
	' Gets the error event data of the CPU core voltage
	ret = GetEvent(EVENT_VOLT_VCOREA, Event)

ClearEvent

Call format	Declare Function ClearEvent Lib "PL_Ioc.dll"
	(ByVal Selector As Long) As Long
Return value	Other than 0: Normal
	0: Error
Argument	Selector As Long Designated error event cancel parameters (pass by value)

EVENT_VOLT_VCOREA	CPU core voltage
EVENT_VOLT_VCOREB	CPU core voltage2
EVENT_VOLT_P33	+3.3 V
EVENT_VOLT_P50	+5.0 V
EVENT_VOLT_P12	+12 V
EVENT_VOLT_M50	-5.0 V
EVENT_VOLT_M12	-12 V
EVENT_TEMP_SYSTEM	System temperature
EVENT_TEMP_CPU	CPU temperature
EVENT_UNI_IN0	Universal Input 0
EVENT_UNI_IN1	Universal Input 1
EVENT_WDT_TIMEOUT	Watchdog Timeout
EVENT_LIGHT_BLOW	Backlightburnout

Processing	Cancels the error event.
Example	Dim ret As Long
	' Cancels the error event.
	ret = ClearEvent(EVENT_VOLT_VCOREA)

SetWdtCounter

Call format	Declare Function SetWdtCounter Lib "PL_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 sec. (pass by value) (unit: second)
Processing	Sets the initial counter value of the current watchdog timer.
Example	Dim ret As Long
	Dim Counter As Long
	' Sets the watchdog timer's initial counter value to 10 seconds.
	Counter = 10
	ret = SetWdtCounter(Counter)

GetWdtCounter

Call format	Declare Function GetWdtCounter Lib "PL_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) (unit: second)
Processing	Gets the initial counter value of the current watchdog timer.
Example	Dim ret As Long
	Dim Counter As Long
	ret = GetWdtCounter(Counter)

SetWdtMask

Call format	Declare Function SetWdtMask Lib "PL_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 items (pass by value)	
		WARNING_LAMP	LAMP
		WARNING_ALARM	ALARM
	Mask As Long	Mask data (pass by value)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the warning mask to be output when a watchdog timer time-out occurs.		
Example	Dim ret As Long		
	' Mask the LAMP output		
	ret = SetWdtMask(WARNING_LAMP, MASK_ON)		
	' Release the mask for the ALARM output		
	ret = SetWdtMask(WARNING_ALARM, MASK_OFF)		

GetWdtMask

Call format	Declare Function GetWdtMask Lib "PL_Ioc.dll"		
	(ByVal Selector As Long, ByRef Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Selector As Long	Setup items (pass by reference)	
		WARNING_LAMP	LAMP
		WARNING_ALARM	ALARM
	Mask As Long	(pass by reference)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the WDT timeout warning output mask data.		
Example	Dim ret As Long		
	Dim Mask As Long		
	' Gets LAMP mask data		
	ret = GetWdtMask(WARNING_LAMP, Mask)		
	' Gets ALARM mask data		
	ret = GetWdtMask(WARNING_ALARM, Mask)		

StartWdt

Call format	Declare Function StartWdt Lib "PL_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 "PL_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 "PL_Ioc.dll" () As Long
Return value	Other than 0: Normal
	0: Error
Argument	None
Processing	Resets the initial value of the watchdog timer to the default value, and restarts the countdown.
Example	Dim ret As Long
	ret = RestartWdt()

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.

RunningWdt

Call format	Declare Function RunningWdt Lib "PL_Ioc.dll"		
	(ByRef RunFlag As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	RunFlag As Long	Operating status of the watchdog timer (pass by reference)	
		WATCHDOG_STOP	Stopped
		WATCHDOG_COUNTDOWN	Counting down
Processing	Gets the operating status of the watchdog timer.		
Example	Dim ret As Long		
	Dim RunFlag As Long		
	ret = RunningWdt(RunFlag)		

SetWarningOut

Call format	Declare Function SetWarningOut Lib "PL_Ioc.dll"	
	(ByVal Selector As Long, ByVal WarnOut As Long) As Long	
Return value	Other than 0: Normal	
	0: Error	
Argument	Selector As Long	Setting items (pass by value)
		WARNING_LAMP LAMP
		WARNING_ALARM ALARM
	WarnOut As Long	Output condition (pass by value)
		OUTPUT_OFF Output OFF
		OUTPUT_ON Output ON
Processing	Sets warning data for the setup items (LAMP and ALARM).	
Example	Dim ret As Long	
	' Set the output status of the LAMP to ON	
	ret = SetWarningOut(WARNING_LAMP, OUTPUT_ON)	
	' Set the output status of the ALARM to OFF	
	ret = SetWarningOut(WARNING_ALARM, OUTPUT_OFF)	

GetWarningOut

Call format	Declare Function GetWarningOut Lib "PL_Ioc.dll"	
	(ByVal Selector As Long, ByRef WarnOut As Long) As Long	
Return value	Other than 0: Normal	
	0: Error	
Argument	Selector As Long	Setting items (pass by value)
		WARNING_LAMP LAMP
		WARNING_ALARM ALARM
	WarnOut As Long	Output condition (pass by reference)
		OUTPUT_OFF Output OFF
		OUTPUT_ON Output ON
Processing	Gets the current warning status of the setup items (LAMP and ALARM).	
Example	Dim ret As Long	
	Dim WarnOut As Long	
	' Gets the output status of the LAMP	
	ret = GetWarningOut(WARNING_LAMP, WarnOut)	
	' Get the output status of the ALARM	
	ret = GetWarningOut(WARNING_ALARM, WarnOut)	

GetUniversalIn

Call format	Declare Function GetUniversalIn Lib "PL_Ioc.dll"		
	(ByVal Selector As Long, ByRef UniIn As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Selector As Long	Designated port (pass by value)	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	UniIn As Long	Input status (pass by reference)	
		INPUT_OFF	No input
		INPUT_ON	Input
Processing	Gets the input status of the designated port (Universal Input 0 and Universal Input 1).		
Example	Dim ret As Long		
	Dim UniIn As Long		
	' Get the input status of the Universal Input 0		
	ret = GetUniversalIn(PORT_UNI0, UniIn)		
	' Get the input status of the Universal Input 1		
	ret = GetUniversalIn(PORT_UNI1, UniIn)		

ClearUniversalIn

Call format	Declare Function ClearUniversalIn Lib "PL_Ioc.dll"		
	(ByVal Selector As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Selector As Long	Designated port (pass by value)	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
Processing	Clears the input status of the designated port (Universal Input 0 and Universal Input 1).		
Example	Dim ret As Long		
	' Clear the input status of Universal Input 0		
	ret = ClearUniversalIn(PORT_UNI0)		
	' Clear the input status of Universal Input 1		
	ret = ClearUniversalIn(PORT_UNI1)		

SetUniversalInMask

Call format	Declare Function SetUniversalInMask Lib "PL_Ioc.dll"		
	(ByVal Selector As Long, ByVal Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Selector As Long	Designated port (pass by value)	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	Mask As Long	Mask data (pass by value)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the masking information of the designated ports (Universal Input 0 and Universal Input 1).		
Example	Dim ret As Long		
	' Release the masking for Universal Input 0		
	ret = SetUniversalInMask(PORT_UNI0, MASK_OFF)		
	' Mask Universal Input 1		
	ret = SetUniversalInMask(PORT_UNI1, MASK_ON)		

GetUniversalInMask

Call format	Declare Function GetUniversalInMask Lib "PL_Ioc.dll"		
	(ByVal Selector As Long, ByRef Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Selector As Long	Designated port (pass by value)	
		PORT_UNI0	Universal Input 0
		PORT_UNI1	Universal Input 1
	Mask As Long	Mask data (pass by reference)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the masking information of the subject ports (Universal Input 0 and Universal Input 1).		
Example	Dim ret As Long		
	Dim Mask As Long		
	' Get the masking information for Universal Input 0		
	ret = GetUniversalInMask(PORT_UNI0, Mask)		
	' Get the masking information for Universal Input 1		
	ret = GetUniversalInMask(PORT_UNI1, Mask)		

SetResetMask

Call format	Declare Function SetResetMask Lib "PL_Ioc.dll"		
	(ByVal Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Mask As Long	Mask data (pass by value)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the reset mask.		
Example	Dim ret As Long		
	' Releases the reset mask		
	ret = SetResetMask(MASK_OFF)		

GetResetMask

Call format	Declare Function GetResetMask Lib "PL_Ioc.dll"		
	(ByRef Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Mask As Long	Mask data (pass by reference)	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the current reset mask information.		
Example	Dim ret As Long		
	Dim Mask As Long		
	ret = GetResetMask(Mask)		

GetLightblowErr

Call format	Declare Function GetLightblowErr Lib "PL_Ioc.dll"		
	(ByRef LightblowErr As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	LightblowErr As Long	Error data (pass by reference)	
		BACKLIGHT_OK	Normal
		BACKLIGHT_ERR	Error
Processing	Gets the current backlight error information.		
Example	Dim ret As Long		
	Dim LightblowErr As Long		
	' Gets the backlight error information.		
	ret = GetLightblowErr(LightblowErr)		

SetWdtResetMask

Call Format	Declare Function SetWdtResetMask Lib "PL_Ioc.dll"		
	(ByVal Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Arguments	Mask As Long	Mask condition settings	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Sets the reset mask status, depending on the watchdog timer.		
Example	Dim ret As Long		
	ret = SetWdtResetMask(MASK_ON)		

GetWdtResetMask

Call Format	Declare Function GetWdtResetMask Lib "PL_Ioc.dll"		
	(ByVal Mask As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Arguments	Mask As Long	Mask condition settings	
		MASK_OFF	Release Mask
		MASK_ON	Set Mask
Processing	Gets the reset mask status, depending on the watchdog timer.		
Example	Dim ret As Long		
	Dim Mask As Long		
	ret = GetWdtResetMask(Mask)		

SetWarningDOUT

Call format	Declare Function SetWarningDOUT Lib "PL_Ioc.dll"		
	(ByVal WarningOut As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	WarningOut As Long	Output status (pass by value)	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Sets the warning status of the current setup item (DOUT).		
Example	Dim ret As Long		
	' Set the output status of DOUT to OFF.		
	ret = SetWarningDOUT(OUTPUT_OFF)		

GetWarningDOUT

Call format	Declare Function GetWarningDOUT Lib "PL_Ioc.dll"		
	(ByRef WarningOut As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	WarningOut As Long	Output status (pass by reference)	
		OUTPUT_OFF	Output OFF
		OUTPUT_ON	Output ON
Processing	Gets the warning status of the current setup item (DOUT).		
Example1	Dim ret As Long		
	Dim WarningOut As Long		
	ret = GetWarningDOUT(WarningOut)		

GetWdtTimeout

Call format	Declare Function GetWdtTimeout Lib "PL_Ioc.dll"
	(ByRef Timebuf As Long) As Long
Return value	Other than 0: Normal
	0: Error
Argument	Timebuf As Long WDT status (pass by reference)
Processing	Gets the watchdog timeout status.
Example	Dim ret As Long
	Dim Timebuf As Long
	' Gets the timeout status of the watchdog.
	ret = GetWdtTimeout(Timebuf)

ClearWdtTimeout

Call format	Declare Function ClearWdtTimeout Lib "PL_Ioc.dll" () As Long
Return value	Other than 0: Normal
	0: Error
Argument	None
Processing	Clears the timeout status of the watchdog.
Example	Dim ret As Long
	' Clear the timeout status of the watchdog.
	ret = ClearWdtTimeout()

GetSmiDrvHandle

Call format	Declare Function GetSmiDrvHandle Lib "PL_Ioc.dll" () As Long
Return value	0: Normal
	1: Error
Argument	None
Processing	Gets the device driver handle to exchange information with the software mirroring device driver.
Example1	Dim ret As Long
	ret = GetSmiDrvHandle()

An error (return value: 1) will occur if the software mirroring device driver is not running.

CloseSmiDrvHandle

Call format	Declare Function CloseSmiDrvHandle Lib "PL_Ioc.dll" () As Long
Return value	Other than 0: Normal
	0: Error
Argument	None
Processing	Destroys the handle acquired with the "GetSmiDrvHandle" function.
Example	Dim ret As Long
	' Destroys the handle.
	ret = CloseSmiDrvHandle()

GetSmiAryStatus

Call format	Declare Function GetSmiAryStatus Lib "PL_Ioc.dll"		
	(ByRef Status As Long) As Long		
Return value	Other than 0: Normal		
	0: Error		
Argument	Status As Long	Software mirroring status (pass by reference)	
		ARYSTAT_GOOD	Normal
		ARYSTAT_UNCONFIG	Not configured
		ARYSTAT_REBUILD	Being rebuilt
		ARYSTAT_REDUCE	Being reduced
		ARYSTAT_DEAD	Mirror status destroyed
Processing	Gets the status of the software mirroring feature.		
Example	Dim ret As Long		
	Dim Status As Long		
	' Get the status of the software mirroring feature.		
	ret = GetSmiAryStatus(Status)		

GetSmiDevStatus

Call format	Declare Function GetSmiDevStatus Lib "PL_Ioc.dll" (ByVal Id As Long, ByRef Type As Long, ByRef 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 HDD (ATA type device) ATAPIDEVICE CD-ROM NODEVICE No device
	Status As Long	Device status (pass by reference) DEVSTAT_GOOD Normal DEVSTAT_NOTEXIST Not connected DEVSTAT_BROKEN Device failure
Processing	Gets the device status of the software mirroring feature.	
Example	Dim ret As Long Dim Id As Long Dim Type As Long Dim Status As Long ' Gets the device status of the software mirroring feature. Id = 0 ret = GetSmiDevStatus(Id, Type, Status)	

7.3.2 PL_Ras.dll Functions

PlDevWordWrite

Call Format	Declare Function PlDevWordWrite Lib "PL_Ras.dll" (ByVal Addr As Long, ByVal wData As Long) As Long	
Return Value	0: Normal Other than 0: Error	
Arguments	Addr As Long	Write memory's 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 = PlDevWordWrite(255, 255)	

PlDevWordRead

Call Format	Declare Function PlDevWordRead Lib "PL_Ras.dll" (ByVal Addr As Long, ByRef wData As Long) As Long	
Return Value	0: Normal Other than 0: Error	
Arguments	Addr As Long	Read memory's word address
	wData As Long	Read data (0 to 65535)
Processing	Reads from common memory	
Example	' Reads address 255 data Dim ret As Long Dim wData As Long ret = PlDevWordRead(255, wData)	

7.3.3 PL_BLIoc.dll Function Details

InitBLIoctl

Call Format	Declare Sub InitBLIoctl Lib "PL_BLIoc.dll" ()
Return Value	None
Arguments	None
Processing	Creates a CPL_BLIoctl object. The object is not destroyed until the EndBLIoctl function is called.
Example	Call InitBLIoctl

EndBLIoctl

Call Format	Declare Sub EndBLIoctl Lib "PL_BLIoc.dll" ()
Return Value	None
Arguments	None
Processing	Destroys the object created using the InitBLIoctl function.
Example	Call EndBLIoctl

GetBLDrvHandle

Call Format	Declare Function GetBLDrvHandle Lib "PL_BLIoc.dll" (ByRef hndl As Long) As Long
Return Value	0: Normal 1: Error
Arguments	hndl As Long Pointer to the device driver handle (pass by reference)
Processing	Gets the device driver handle to communicate with the device driver.
Example	Dim ret As Long Dim hndl As Long ret = GetBLDrvHandle(hndl)

An error (Return Value: 1) occurs if the System Monitor/RAS Device Driver is not running.

GetBLDrvVersion

Call Format	Declare Function GetBLDrvVersion Lib "PL_BLIoc.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 (Major, 0 to 99) (pass by reference)
	Minor As Long	Version data (Minor, 0 to 99) (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)	

When the drivers version is 1.10,
Major:1 (Decimal)
Minor:10 (Decimal)

GetBLDrvVersionEx

Call format	Declare Function GetBLDrvVersionEx Lib "PL_BLIoc.dll" (By Ref Product As Long, Major As Long, By Ref Minor As Long) As Long	
Return value	Other than 0: Normal 0: Error	
Argument	Product As Long	PL unit data (pass by reference)
	Major As Long	Version data (Major, 0 to 99) (pass by reference)
	Minor As Long	Version data (Minor, 0 to 99) (pass by reference)
Processing	Gets the driver version.	
Example	Dim ret As Long Dim Product As Long Dim Major As Long Dim Minor As Long ret = GetBLDrvVersionEx(Product, Major, Minor)	

When the PL-5910 Series unit's driver version is 1.00,
Product:4 (Decimal)
Major:1 (Decimal)
Minor:00 (Decimal)

SetBLControl

Call Format	Declare Function SetBLControl Lib "PL_BLIoc.dll" (ByVal BLFlag As Long) As Long		
Return value	Other than 0: Normal 0: Error		
Argument	BLFlag As Long	Setting Parameter (pass by value)	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Sets the backlight ON/OFF.		
Example	Dim ret As Long ' Sets the backlight control to ON. ret = SetBLControl(BACKLIGHT_ON)		

GetBLControl

Call Format	Declare Function GetBLControl Lib "PL_BLIoc.dll" (ByRef BLFlag As Long) As Long		
Return value	Other than 0: Normal 0: Error		
Argument	BLFlag As Long	Setting Parameter (pass by reference)	
		BACKLIGHT_OFF	Backlight OFF
		BACKLIGHT_ON	Backlight ON
Processing	Gets the backlight ON/OFF status.		
Example	Dim ret As Long Dim BLFlag As Long ' Gets the backlight control status. ret = GetBLControl(BLFlag)		

Memo