# Soliton PCI Express Mini Extender

## PEM-1X DLL Programming Description

**Rev 1.0**
**October 5, 2005**

| Rev. No. | Description | Date | Approved |
|---|---|---|---|
| 0.1 | Initial | Aug/24/2005 | Vincent |
| 0.9 | Release Candidate | Oct/04/2005 | Vincent |
| 1.0 | 1st Release | Oct/05/2005 | Vincent |
| 3.1 | Add Borland C++ support | Oct/18/2007 | Kikimum |
| | | | |
| | | | |
| | | | |

# 1. Installation

Execute the PEM1XDll_Setup.exe from the CD. All the required component will be installed in the **Program files/Soliton/Pem1x/** folder**.** For developer, who want to integrate test program with the PEM-1X control. Please find all the needed samples and development resources in the **Program files/Soliton/Pem1x/DLLDev** folder**.**

# 2. Function Description

## int CPEMDll::INIT()

**Syntax:**

**Form 1:** int CPEMDll::INIT()

**Description:**

Initialize all the PEM-1X cards on mainboard.

**Input Value:** None

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

```
CPEMDLL pemctl;
pemctl.INIT();
```

## int CPEMDll::INITEX()
## int CPEMDll::INITEX(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::INITEX()

**Form 2:** int CPEMDll::INITEX(DWORD addr)

**Description:**

INITEX will initialize the PEM-1X module or modules with minimum resources.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: Module addr, range 0~3. if specified, INITEX will initialize the dedicate PEM-1X module.

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;
DWORD addr = 0;
**Form 1:** pemctl.INITEX();
**Form 2:** pemctl.INITEX(addr);

## int CPEMDll::EXIT()

**Syntax:**

int CPEMDll::EXIT()

**Description:**

Close and release all the opened resources. Called before terminate the program.

**Input Value:** None

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;
pemctl.EXIT();

## void CPEMDll::GETLIBVER(int* major, int* minor)

**Syntax:**

void CPEMDll::GETLIBVER(int* major, int* minor)

**Description:**

Get the version number of PETs DLL

**Input Value:** None

**Return Value:** None

**Usage:**

**VC++**

CPEMDLL pemctl;
pemctl. GETLIBVER (&majorno, &minorno);

## int CPEMDll::SELPEM(DWORD addr)

**Syntax:**

**From 1:** int CPEMDll::SELPEM(DWORD addr)

**Description:**

Select handle for PEM-1X module on mainboard. If success, user can call other operational

function without specifying the module address.

**Input Value:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

```
CPEMDLL pemctl;
DWORD addr = 0;
pemctl.SELPEM(addr);
```

## int CPEMDll::VALIDPEM(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::VALIDPEM(DWORD addr)

**Description:**

Check if the PEM-1X module exist.

**Input Value:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if device is invalid, 1 if device is validl; -1 if error.

**Usage:**

**VC++**

```
CPEMDLL pemctl;
DWORD addr = 0;
int status;
status = pemctl.VALIDPEM(addr);
```

## int CPEMDll::VALIDDEV()
## int CPEMDll::VALIDDEV(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::VALIDDEV()

**Form 2:** int VALIDDEV(DWORD addr)

**Description:**

Check if the specified PCI-Express Device exist on PEM-1X.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if device is invalid; 1 if device is valid; -1 if error; -2 if power off.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int status;

**Form 1:** status = pemctl.VALIDDEV();

**Form 2:** status = pemctl.VALIDDEV(addr);

## int CPEMDll::SELDEV(DWORD Vid, DWORD Did, int index)
## int CPEMDll::SELDEV(DWORD addr, DWORD Vid, DWORD Did, int index)

**Syntax:**

**Form 1:** int CPEMDll::SELDEV(DWORD Vid, DWORD Did, int index)

**Form 2:** int CPEMDll::SELDEV(DWORD addr, DWORD Vid, DWORD Did, int index)

**Description:**

Select specified PCI-Express Device with Vendor ID and Device ID. If there are multiple devices with same VID and DID, user should specified the index.

**Input Value:**

**Form 1:** Vid Vendor ID. Did Device ID. Index Card number.

**Form 2:** addr Module addr, range 0~3.

Vid Vendor ID.

Did Device ID.

index Card number.

int index: index of the devices

**Return Value:**

Return 0 if successful; 1 if error; 6 if device not match.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

DWORD Vid = 0x1234;

DWORD Did = 0;

int index = 0

**Form 1:** pemctl.SELDEV(Vid, Did, index);

**Form 2:** pemctl.SELDEV(addr, Vid, Did, index);

## int CPEMDll::SELDEV(DWORD Busno, DWORD Devno)
## int CPEMDll::SELDEV(DWORD addr, DWORD Busno, DWORD Devno)

**Syntax:**

**Form 1:** int CPEMDll::SELDEV(DWORD Busno, DWORD Devno)

**Form 2:** int CPEMDll::SELDEV(DWORD addr, DWORD Busno, DWORD Devno)

**Description:**

Select specified PCI-Express Device with Bus number and Device number.

**Input Value:**

**Form 1:** Busno Bus number.

Devno Device number.

**Form 2:** addr Module addr, range 0~3.

Busno Bus number.

Devno Device number.

**Return Value:**

Return 0 if successful; 1 if error; 6 if device not match.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

DWORD Busno = 3;

DWORD Devno = 0;

**Form 1:** pemctl.SELDEV(Busno, Devno);

**Form 2:** pemctl.SELDEV(addr, Busno, Devno);

## int CPEMDll::PON()
## int CPEMDll::PON(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::PON ()

**Form 2:** int CPEMDll::PON(DWORD addr)

**Description:**

Turn the specified PEM-1X module power on.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error; 4 if power off.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.PON ();

**Form 2:** pemctl.PON(addr);

## int CPEMDll::POFF()
## int CPEMDll::POFF(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::POFF(

**Form 2:** int CPEMDll::POFF(DWORD addr)

**Description:**

Turn the specified PEM-1X module power off.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.POFF ();

**Form 2:** pemctl.POFF(addr);

## int CPEMDll::GETPWRSTS()
## int CPEMDll::GETPWRSTS(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::GETPWRSTS()

**Form 2:** int CPEMDll::GETPWRSTS(DWORD addr)

**Description:**

Get the power status of the specified PEM-1X module.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if power off ; 1 if power on; -1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int status;

**Form 1:** status = pemctl.GETPWRSTS();

**Form 2:** status = pemctl.GETPWRSTS(addr);

## int CPEMDll::CHKSHORT()
## int CPEMDll::CHKSHORT(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::CHKSHORT()

**Form 2:** int CPEMDll::CHKSHORT(DWORD addr)

**Description:**

Check which power rail is shorted.

**Input Value:**

**Form 1:** None

**Form 2:** DWORD addr: module address, range 0~3

**Return Value:**

Return -1 if error,

0 if normal,

0x01 if 1.5V rail short;

0x02 if 3.3V rail short;

0x04 if 3.3VAux rail short.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int status;

**Form 1:** status = pemctl.CHKSHORT();

**Form 2:** status = pemctl.CHKSHORT(addr);

## int CPEMDll::GET3V3V(double* volatge)
## int CPEMDll::GET3V3V(DWORD addr, double* volatge)

**Syntax:**

**Form 1:** int CPEMDll::GET3V3V(double* volatge)

**Form 2:** int CPEMDll::GET3V3V(DWORD addr, double* volatge)

**Description:**

Get 3.3V rail voltage reading.

**Input Value:**

**Form 1:** (double* volatge)

**Form 2:** (DWORD addr, double* voltage)

DWORD addr: module address, range 0~3

double* voltage: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn't exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double voltage;

**Form 1:** pemctl.GET3V3V(&volatge);

**Form 2:** pemctl.GET3V3V(addr, &volatge);

## int CPEMDll::GET1V5V(double* volatge)
## int CPEMDll::GET1V5V(DWORD addr, double* volatge)

**Syntax:**

**Form 1:** int CPEMDll::GET1V5V(double* volatge)

**Form 2:** int CPEMDll::GET1V5V(DWORD addr, double* volatge)

**Description:**

Get 1.5V rail voltage reading.

For **PEM-1X** & **PEC-1X**

**Input Value:**

**Form 1:** (double* volatge)

**Form 2:** (DWORD addr, double* voltage)

DWORD addr: module address, range 0~3

double* voltage: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn’t exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double voltage;

**Form 1:** pemctl. GET1V5V(&volatge);

**Form 2:** pemctl. GET1V5V(addr, &volatge);

## int CPEMDll::GET12V(double* volatge)
## int CPEMDll::GET12V(DWORD addr, double* volatge)

**Syntax:**

**Form 1:** int CPEMDll::GET12V(double* volatge)

**Form 2:** int CPEMDll::GET12V(DWORD addr, double* volatge)

**Description:**

Get 12V rail voltage reading.

For **PEX-1X** & **PEX-16X**

**Input Value:**

**Form 1:** (double* volatge)

**Form 2:** (DWORD addr, double* voltage)

DWORD addr: module address, range 0~3

double* voltage: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn’t exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double voltage;

**Form 1:** pemctl. GET12V(&volatge);

**Form 2:** pemctl. GET12V(addr, &volatge);

## int CPEMDll::GET3V3I(double* current)
## int CPEMDll::GET3V3I(DWORD addr, double* current)

**Syntax:**

**Form 1:** int CPEMDll::GET3V3I(double* current)

**Form 2:** int CPEMDll::GET3V3I(DWORD addr, double* current)

**Description:**

Get 3.3V rail current reading.

**Input Value:**

**Form 1:** (double* current)

**Form 2:** (DWORD addr, double* current)

DWORD addr: module address, range 0~3

double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn't exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double current;

**Form 1:** pemctl. GET3V3I (&current);

**Form 2:** pemctl. GET3V3I (addr, & current);

## int CPEMDll::GET1V5I(double* current)
## int CPEMDll::GET1V5I(DWORD addr, double* current)

**Form 1:** int CPEMDll::GET1V5I(double* current)

**Form 2:** int CPEMDll::GET1V5I(DWORD addr, double* current)

**Description:**

Get 1.5V rail current reading.

For **PEM-1X** & **PEC-1X**

**Input Value:**

**Form 1:** (double* current)

**Form 2:** (DWORD addr, double* current)

DWORD addr: module address, range 0~3

double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn’t exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double current;

**Form 1:** pemctl. GET1V5I (&current);

**Form 2:** pemctl. GET1V5I (addr, & current);

## int CPEMDll::GET12I(double* current)
## int CPEMDll::GET12I(DWORD addr, double* current)

**Syntax:**

**Form 1:** int CPEMDll::GET12I(double* current)

**Form 2:** int CPEMDll::GET12I(DWORD addr, double* current)

**Description:**

Get 12V rail current reading.

For **PEX-1X** & **PEX-16X**

**Input Value:**

**Form 1:** (double* current)

**Form 2:** (DWORD addr, double* current)

DWORD addr: module address, range 0~3

double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn’t exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double current;

**Form 1:** pemctl. GET12I (&current);

**Form 2:** pemctl. GET12I (addr, & current);

## int CPEMDll::GET3V3IMAX(double* current)
## int CPEMDll::GET3V3IMAX(DWORD addr, double* current)

**Syntax:**

**Form 1:** int CPEMDll::GET3V3IMAX(double* current)

**Form 2:** int CPEMDll::GET3V3IMAX(DWORD addr, double* current)

**Description:**

Get 3.3V Max current Reading.

**Input Value:**

**Form 1:** (double* current)

**Form 2:** (DWORD addr, double* current)

DWORD addr: module address, range 0~3

double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn't exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double current;

**Form 1:** pemctl.GET3V3I MAX(&current);

**Form 2:** pemctl.GET3V3I MAX(addr, &current);

## int CPEMDll::GET1V5IMAX(double* current)
## int CPEMDll::GET1V5IMAX(DWORD addr, double* current)

**Syntax:**

**Form 1:** int CPEMDll::GET1V5IMAX(double* current)

**Form 2:** int CPEMDll::GET1V5IMAX(DWORD addr, double* current)

**Description:**

Get 1.5V Max current Reading.

For **PEM-1X** & **PEC-1X**

**Input Value:**

**Form 1:** (double* current)

**Form 2:** (DWORD addr, double* current)

DWORD addr: module address, range 0~3

double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if error; 3 if ADC doesn't exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

double current;

**Form 1:** pemctl.GET1V5I MAX(&current);

**Form 2:** pemctl.GET1V5I MAX(addr, &current);

## int CPEMDll::SET1V5ICAL()
## int CPEMDll::SET1V5ICAL(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::SET1V5ICAL()

**Form 2:** int CPEMDll::SET1V5ICAL(DWORD addr)

**Description:**

Calibrate 1.5V current measurement.

**Note: Do not plug any DUT on slot.**

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.SET1V5ICAL ();

**Form 2:** pemctl.SET1V5ICAL (addr);

## int CPEMDll::SET3V3ICAL()
## int CPEMDll::SET3V3ICAL(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::SET3V3ICAL()

**Form 2:** int CPEMDll::SET3V3ICAL(DWORD addr)

**Description:**

Calibrate 3.3V current measurement.

**Note: Do not plug any DUT on slot**

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.SET3V3ICAL ();

**Form 2:** pemctl.SET3V3ICAL (addr);

## int CPEMDll::SET12ICAL()
## int CPEMDll::SET12ICAL(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::SET12ICAL()

**Form 2:** int CPEMDll::SET12ICAL(DWORD addr)

**Description:**

Calibrate 12V current measurement.

**Note: Do not plug any DUT on slot**

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.SET12ICAL ();

**Form 2:** pemctl.SET12ICAL (addr);

## int CPEMDll::SETUSECAL(int i)
## int CPEMDll::SETUSECAL(DWORD addr, int i)

**Syntax:**

**Form 1:** int CPEMDll::SETUSECAL(int i)

**Form 2:** int CPEMDll::SETUSECAL(DWORD addr, int i)

**Description:**

Use Calibrate when set to 1, get row measurement data when set to 0..

**Input Value:**

**Form 1:** (int i)

**Form 2:** (DWORD addr, int i)

int i:

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

int i = 0;

DWORD addr = 0;

**Form 1:** pemctl.SETUSECAL(i);

**Form 2:** pemctl.SETUSECAL1(addr, i);

## int CPEMDll::GETUSECAL()
## int CPEMDll::GETUSECAL(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::GETUSECAL()

**Form 2:** int CPEMDll::GETUSECAL(DWORD addr)

**Description:**

Get use_calibrate flag.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.GETUSECAL();

**Form 2:** pemctl.GETUSECAL(addr);

## int CPEMDll::WINEN(int delaytime)
## int CPEMDll::WINEN(DWORD addr, int delaytime)

**Syntax:**

**Form 1:** int CPEMDll::WINEN(int delaytime)

**Form 2:** int CPEMDll::WINEN(DWORD addr, int delaytime)

**Description:**

1st method of windows driver enable.

**Input Value:**

**Form 1:** delaytime delay between enabling each device. The unit is milli-second

**Form 2:** (DWORD addr, int delaytime)

DWORD addr: module address, range 0~3

int delaytime: delay between enabling each device. The unit is milli-second.

**Return Value:**

Return 0 if successful; 1 if error; -2 if power off

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.WINEN(100);

**Form 2:** pemctl.WINEN(addr,100);

## int CPEMDll::WINEN2(int delaytime)
## int CPEMDll::WINEN2(DWORD addr, int delaytime)

**Syntax:**

**Form 1:** int CPEMDll::WINEN2(int delaytime)

**Form 2:** int CPEMDll::WINEN2(DWORD addr, int delaytime)

**Description:**

2nd method of windows driver enable.

**Input Value:**

**Form 1:** delaytime delay between enabling each device. The unit is milli-second

**Form 2:** (DWORD addr, int delaytime)

DWORD addr: module address, range 0~3

int delaytime: delay between enabling each device. The unit is milli-second.

**Return Value:**

Return 0 if successful; 1 if error; -2 if power off

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.WINEN2(100);

**Form 2:** pemctl.WINEN2(addr,100);

## int CPEMDll::WINDIS(int delaytime)
## int CPEMDll::WINDIS(DWORD addr, int delaytime)

**Syntax:**

**Form 1:** int CPEMDll::WINDIS(int delaytime)

**Form 2:** int CPEMDll::WINDIS(DWORD addr, int delaytime)

**Description:**

Disable the windows driver of the specified PCI-Express Device on PEM-1X module.

**Input Value:**

**Form 1:** (int delaytime)

**Form 2:** (DWORD addr, int delaytime)

DWORD addr: module address, range 0~3

int delaytime: delay between disabling each device.

**Return Value:**

Return 0 if successful; 1 if error;

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.WINDIS(100);

**Form 2:** pemctl.WINDIS(addr,100);

## int CPEMDll::WINCHECK()
## int CPEMDll:: WINCHECK (DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::WINCHECK()

**Form 2:** int CPEMDll::WINCHECK(DWORD addr)

**Description:**

Check if the windows driver of the specified PCI-Express Device on PEM-1X module is enabled.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if driver are dsiabled; 1 if driver are enabled.

**Usage:**

**VC++**

CPEMDLL pemctl;

Int status;

DWORD addr = 0;

**Form 1:** status = pemctl.WINCHECK();

**Form 2:** status = pemctl.WINCHECK(addr);

## int CPEMDll::WAITCARDREMOVE()
## int CPEMDll::WAITCARDREMOVE(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::WAITCARDREMOVE()

**Form 2:** int CPEMDll::WAITCARDREMOVE(DWORD addr)

**Description:**

Wait the PCI-Express Card on slot is removed.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** while(pemctl.WAITCARDREMOVE())

**Form 2:** while(pemctl.WAITCARDREMOVE(addr))

## int CPEMDll::WAITCARDPLUG()
## int CPEMDll::WAITCARDPLUG(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::WAITCARDPLUG()

**Form 2:** int CPEMDll::WAITCARDPLUG(DWORD addr)

**Description:**

Wait the PCI-Express Card on slot is plugged.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** while(pemctl.WAITCARDPLUG())

**Form 2:** while(pemctl.WAITCARDPLUG(addr))

## int CPEMDll::CHECKCD()
## int CPEMDll::CHECKCD(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::CHECKCD()

**Form 2:** int CPEMDll::CHECKCD(DWORD addr)

**Description:**

Check if the PCI-Express Card on slot is plugged.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 1 if Card Exist, 0 if Card not Exist.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.CHECKCD()

**Form 2:** pemctl.CHECKCD(addr)

## int CPEMDll::LEDGO()
## int CPEMDll::LEDGO(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::LEDGO()

**Form 2:** int CPEMDll::LEDGO(DWORD addr)

**Description:**

Turn on the GO LED to indicate the test status.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.LEDGO();

**Form 2:** pemctl.LEDGO(addr);

## int CPEMDll::LEDNG()
## int CPEMDll::LEDNG(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::LEDNG()

**Form 2:** int CPEMDll::LEDNG(DWORD addr)

**Description:**

Turn on the NO-GO LED to indicate the test status.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.LEDNG();

**Form 2:** pemctl.LEDNG(addr);

## int CPEMDll::LEDOFF()
## int CPEMDll::LEDOFF(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::LEDOFF()

**Form 2:** int CPEMDll::LEDOFF(DWORD addr)

**Description:**

Turn off both the GO and NG LEDs.

**Input Value:**

**Form 1:** None

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

**Form 1:** pemctl.LEDOFF();

**Form 2:** pemctl.LEDOFF(addr);

## int CPEMDll::BEEP(int freq, int time)
## int CPEMDll::BEEP(DWORD addr, int freq, int time)

**Syntax:**

**Form 1:** int CPEMDll::BEEP(int freq, int time)

**Form 2:** int CPEMDll::BEEP(DWORD addr, int freq, int time)

**Description:**

Play Beeper.

**Input Value:**

**Form 1:** (int freq, int time)

**Form 2:** (DWORD addr, int freq, int time)

DWORD addr: module address, range 0~3

int freq: frequency of sound, range 0~3

int time: duration of sound, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

```
CPEMDLL pemctl;
DWORD addr = 0;
int freq = 1
int time = 1
```

**Form 1:** pemctl.BEEP(freq, time);

**Form 2:** pemctl.BEEP(addr, freq, time);

## int CPEMDll::SETPONRST(int setting)
## int CPEMDll::SETPONRST(DWORD addr, int setting)

**Syntax:**

**Form 1:** int CPEMDll::SETPONRST(int setting)

**Form 2:** int CPEMDll::SETPONRST(DWORD addr, int setting)

**Description:**

Set power on reset timing.

**Input Value:**

**Form 1:** (int setting)

int setting:power on reset time, range 0~3

0x00: 25 msec

0x01: 50 msec (default)

0x02: 100 msec

0x03: 150 msec

**Form 2:** (DWORD addr, int setting)

DWORD addr: module address, range 0~3

int setting: power on reset time, , range 0~3
0x00: 25 msec
0x01: 50 msec (default)
0x02: 100 msec
0x03: 150 msec

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;
DWORD addr = 0;
int setting = 0;
**Form 1:** pemctl.SETPONRST(setting);
**Form 2:** pemctl.SETPONRST(addr, setting);

## int CPEMDll::PROBESMBCLKRISE(double* time)
## int CPEMDll::PROBESMBCLKRISE(DWORD addr, double* time)

**Syntax:**

**Form 1:** int CPEMDll::PROBESMBCLKRISE(double* time)
**Form 2:** int CPEMDll::PROBESMBCLKRISE(DWORD addr, double* time)
**Description:**

Get rise time of SMBCLK

**Input Value:**

**Form 1:** (double* time)
**Form 2:** (DWORD addr, double* time)
DWORD addr: module address, range 0~3
double* current: if success, the converted result will stored in the passed double pointer.

**Return Value:**

Return 0 if successful; 1 if bus error; -1 if detection fail (smbclk might be shorted to GND)

**Usage:**

**VC++**

CPEMDLL pemctl;
DWORD addr = 0;
double time;
**Form 1:** pemctl.PROBESMBCLKRISE(&time);
**Form 2:** pemctl.PROBESMBCLKRISE(addr, & time);

## int CPEMDll::GPIOSETUP(int pinno, int mode, int val)
## int CPEMDll:: GPIOSETUP (DWORD addr, int pinno, int mode, int val)

**Syntax:**

**Form 1:** int CPEMDll::GPIOSETUP(int pinno, int mode, int val)

**Form 2:** int CPEMDll:: GPIOSETUP(DWORD addr, int pinno, int mode, int val)

**Description:**

GPIO control mode setting.

**Input Value:**

**Form 1:** (int pinno, int mode, int val)

int pinno: GPIO pin no, range 1~3

int mode: GPIO control mode, 0: input, 1: output

int val: 0: Low, 1: High

**Form 2:** (DWORD addr, int pinno, int mode, int val)

DWORD addr: module address, range 0~3

int pinno: GPIO pin no, range 1~3

int mode: GPIO control mode, 0: input, 1: output

int val: 0: Low, 1: High

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int pinno = 0;

int mode = 0;

int val = 0;

**Form 1:** pemctl. GPIOSETUP(pinno, mode, val);

**Form 2:** pemctl. GPIOSETUP(addr, pinno, mode, val);

## int CPEMDll::GPIOOUTSET(int pinno)
## int CPEMDll:: GPIOOUTSET (DWORD addr, int pinno)

**Syntax:**

**Form 1:** int CPEMDll::GPIOOUTSET(int pinno)

**Form 2:** int CPEMDll::GPIOOUTSET(DWORD addr, int pinno)

**Description:**

Set GPIO output to high.

**Input Value:**

**Form 1:** (int pinno)

int pinno: GPIO pin no, range 1~3

**Form 2:** (DWORD addr, int pinno)

DWORD addr: module address, range 0~3

int pinno: GPIO pin no, range 1~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int pinno = 0;

**Form 1:** pemctl. GPIOOUTSET(pinno);

**Form 2:** pemctl. GPIOOUTSET(addr, pinno);

## int CPEMDll::GPIOOUTRESET(int pinno)
## int CPEMDll::GPIOOUTRESET(DWORD addr, int pinno)

**Syntax:**

**Form 1:** int CPEMDll::GPIOOUTRESET(int pinno)

**Form 2:** int CPEMDll::GPIOOUTRESET(DWORD addr, int pinno)

**Description:**

Set GPIO output to low.

**Input Value:**

**Form 1:** (int pinno)

int pinno: GPIO pin no, range 1~3

**Form 2:** (DWORD addr, int pinno)

DWORD addr: module address, range 0~3

int pinno: GPIO pin no, range 1~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int pinno = 0;

**Form 1:** pemctl.GPIOOUTRESET(pinno);

**Form 2:** pemctl.GPIOOUTRESET(addr, pinno);

## int CPEMDll::GPIOIN(int pinno)
## int CPEMDll::GPIOIN(DWORD addr, int pinno)

**Syntax:**

**Form 1:** int CPEMDll::GPIOIN(int pinno)

**Form 2:** int CPEMDll::GPIOIN(DWORD addr, int pinno)

**Description:**

Get GPIO input value.

**Input Value:**

**Form 1:** (int pinno)

int pinno: GPIO pin no, range 1~3

**Form 2:** (DWORD addr, int pinno)

DWORD addr: module address, range 0~3

int pinno: GPIO pin no, range 1~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int pinno = 0;

**Form 1:** pemctl.GPIOIN(pinno);

**Form 2:** pemctl.GPIOIN(addr, pinno);

## int CPEMDll::GPIOGETSETUP(int pinno)
## int CPEMDll::GPIOGETSETUP(DWORD addr, int pinno)

**Syntax:**

**Form 1:** int CPEMDll::GPIOGETSETUP(int pinno)

**Form 2:** int CPEMDll::GPIOGETSETUP (DWORD addr, int pinno)

**Description:**

Get GPIO setting .

**Input Value:**

**Form 1:** (int pinno)

int pinno: GPIO pin no, range 1~3

**Form 2:** (DWORD addr, int pinno)

DWORD addr: module address, range 0~3

int pinno: GPIO pin no, range 1~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**VC++**

CPEMDLL pemctl;

DWORD addr = 0;

int pinno = 0;

**Form 1:** pemctl.GPIOGETSETUP(pinno);

**Form 2:** pemctl.GPIOGETSETUP(addr, pinno);

## int CPEMDll::PWRCTL_AUTO()
## int CPEMDll::PWRCTL_AUTOA(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::PWRCTL_AUTO()

**Form 2:** int CPEMDll::PWRCTL_AUTOA(DWORD addr)

**Description:**

Set Power control to **Auto** mode . .

**Input Value:**

**Form 1:** None.

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**Borland C++**

DWORD addr = 0;

**Form 1:** pemctl.PWRCTL_AUTO();

**Form 2:** pemctl.PWRCTL_AUTOA(addr);

## int CPEMDll::PWRCTL_MANUAL()
## int CPEMDll::PWRCTL_MANUALA(DWORD addr)

**Syntax:**

**Form 1:** int CPEMDll::PWRCTL_MANUAL()

**Form 2:** int CPEMDll::PWRCTL_MANUALA(DWORD addr)

**Description:**

Set Power control to **Manual** mode .

**Input Value:**

**Form 1:** None.

**Form 2:** (DWORD addr)

DWORD addr: module address, range 0~3

**Return Value:**

Return 0 if successful; 1 if error.

**Usage:**

**Borland C++**

DWORD addr = 0;

**Form 1:** pemctl.PWRCTL_MANUAL();

**Form 2:** pemctl.PWRCTL_MANUALA(addr);

### 3. Operation Flow

## Initialize Flow

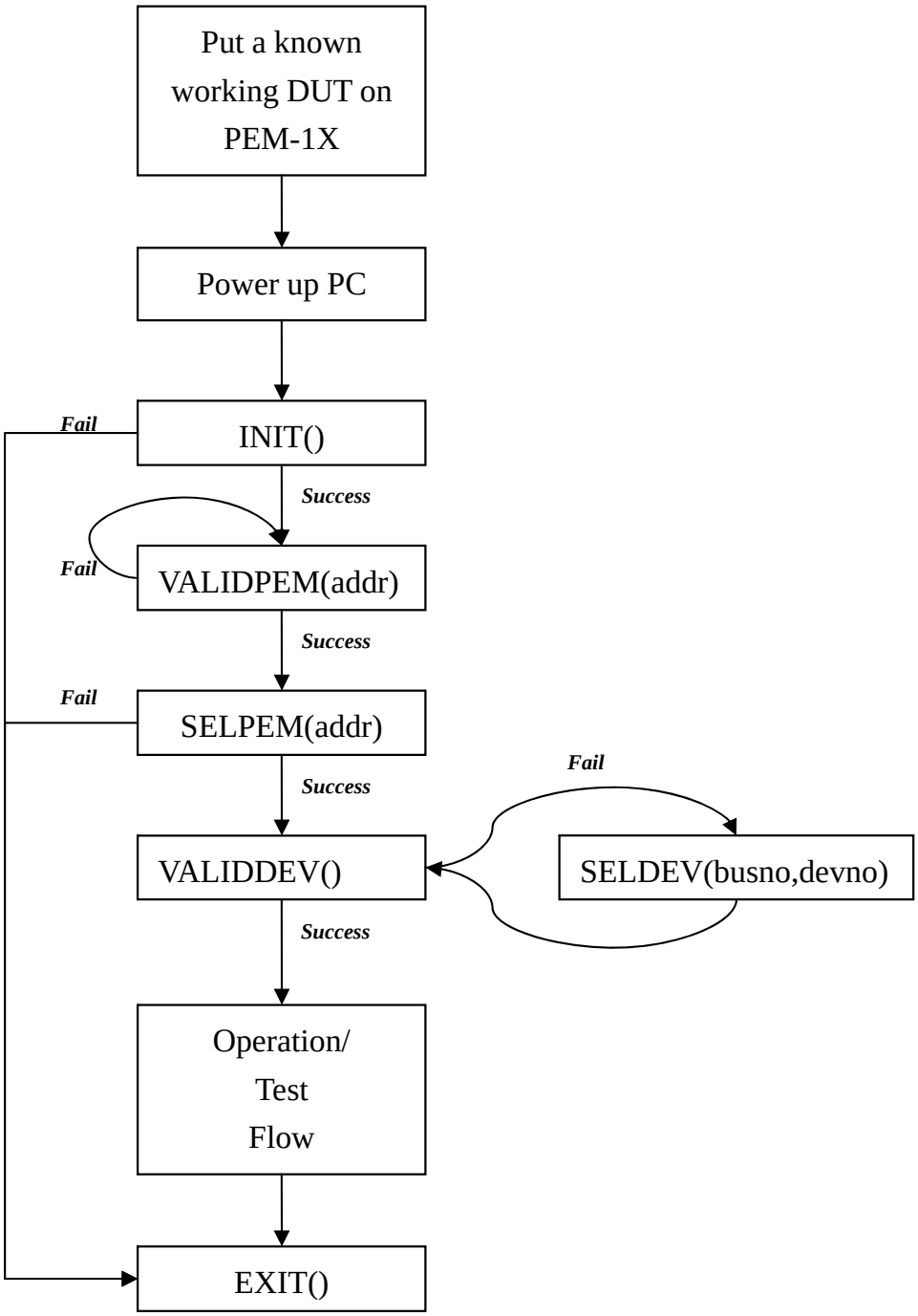


Fig. 1: Initialize Flow

## Operation/Test Flow

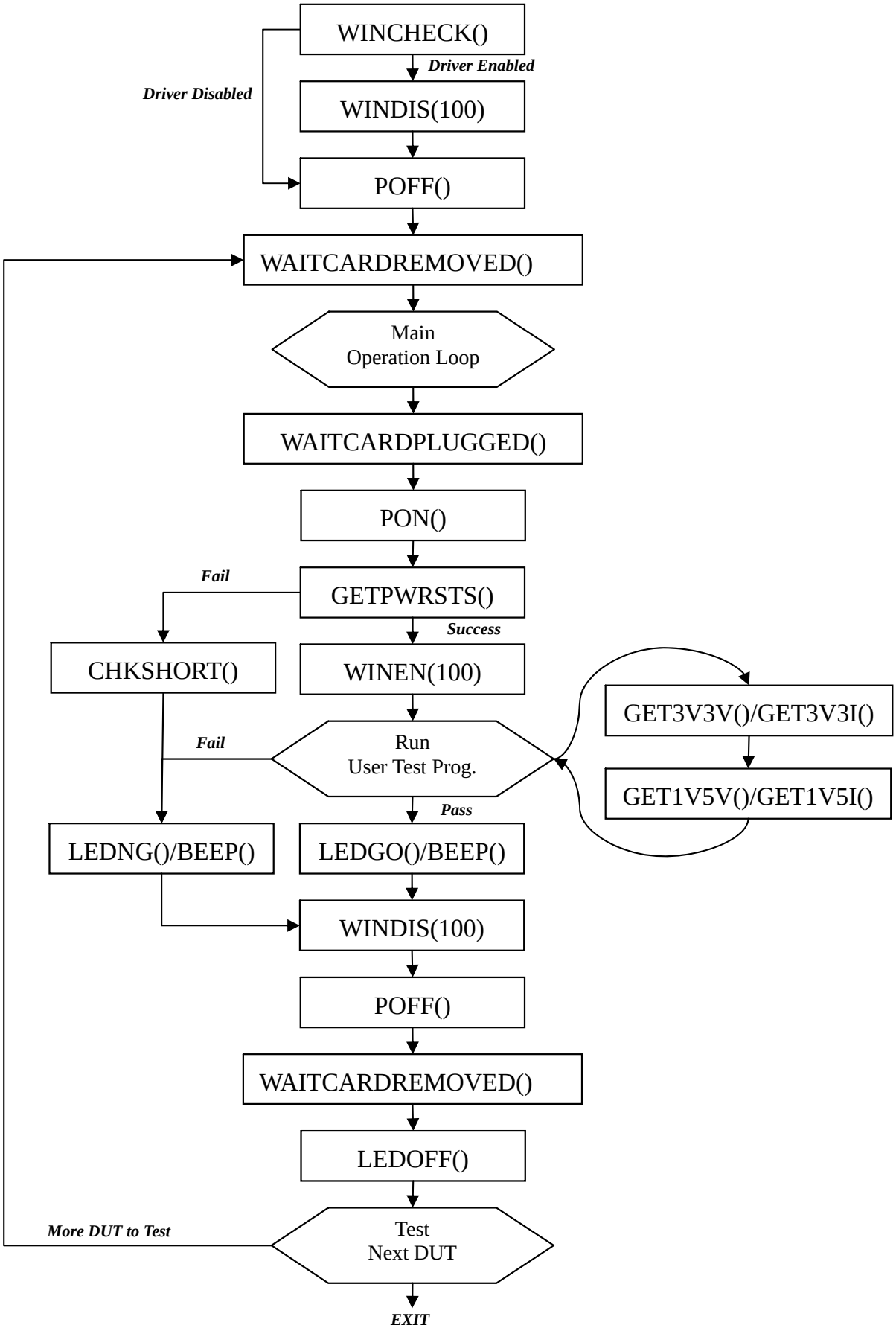


Fig. 2: Operation/Test Flow

# 4. Sample Program

## VC++ Sample

```
#include "stdafx.h"
#include "windows.h"
#include "../include/pemdll.h"
#include "conio.h"

void showpwr(CPEMDll* pem)
{
	int status;
	double dval = 0;
	status = pem->GETPWRSTS();
	if (status == 1)
	{
		printf("Power On!\n");
	}
	else
	{
		printf("Power Off!\n");
	}
	status = pem->GET3V3V(&dval);
	if (status == 0)
	{
		printf("3.3V Voltage = %f\n", dval);
	}
	else if (status == CPEMDLL_ERROR_ADCNOTEXIST)
	{
		printf("ADC Not Exist!\n");
	}
	else
	{
		printf("GET3V3V fail! status = %d\n", status);
	}
	status = pem->GET3V3I(&dval);
	if (status == 0)
	{
		printf("3.3V Current = %f\n", dval);
	}
	else if (status == CPEMDLL_ERROR_ADCNOTEXIST)
	{
		printf("ADC Not Exist!\n");
	}
	else
	{
		printf("GET3V3I fail! status = %d\n", status);
	}
	status = pem->GET1V5V(&dval);
```

```
	if (status == 0)
	{
		printf("1.5V Voltage = %f\n", dval);
	}
	else if (status == CPEMDLL_ERROR_ADCNOTEXIST)
	{
		printf("ADC Not Exist!\n");
	}
	else
	{
		printf("GET1V5V fail! status = %d\n", status);
	}
	status = pem->GET1V5I(&dval);
	if (status == 0)
	{
		printf("1.5V Current = %f\n", dval);
	}
	else if (status == CPEMDLL_ERROR_ADCNOTEXIST)
	{
		printf("ADC Not Exist!\n");
	}
	else
	{
		printf("GET1V5I fail! status = %d\n", status);
	}
}
int main(int argc, char* argv[])
{
	int status;
	int pemvalid[4];
	int currpemno;
	double dval = 0;
	CPEMDll pem;
	int pcidevcount = 0;

	status = pem.INIT();
	for (int i = 0; i < 4; i++)
	{
		pemvalid[i] = pem.VALIDPEM(i);
		if (pemvalid[i] == 1) currpemno = i;
		printf("PEM[%d] %s\n", i, (pemvalid[i]==1) ? "Exist":"Not Exist");
	}
	status = pem.SELPEM(currpemno);
	if (status != 0) printf("SELPEM Error!\n");
	showpwr(&pem);
	pem.SELECTDEV(3, 0);
	status = pem.VALIDDEV();
	if (status == 1)
	{
		printf("VALIDDEV Success!\n");
```

```
	}
	else
	{
		printf("VALIDDEV Fail!    Need to Assign A New Device\n");
		goto ERR;
	}

	printf("Press Any Key to Disbale Driver and Power Off!\n");
	getch();

	status = pem.WINDIS(200);
	if (status == 0) printf("Driver Disabled!\n");
	else printf("Driver Disabled Fail!\n");

	status = pem.POFF();
	if (status == 0) printf("Power Off!\n");
	else printf("Power Off Fail!\n");

	printf("Press Any Key to Enable Driver and Power On!\n");
	getch();

	pem.PON();
	if (status == 0)
	{
		printf("Power On!\n");
		status = pem.WINEN(200);
		if (status == 0) printf("Driver Enabled!\n");
		else printf("Driver Enabled Fail!\n");
	}
	else
	{
		printf("Power On Fail!\n");
		status = pem.CHKSHORT();
		if (status & CPEMDLL_ERROR_3V3AUXSHORT) printf("3.3VAux Short!\n");
		if (status & CPEMDLL_ERROR_3V3SHORT) printf("3.3V Short!\n");
		if (status & CPEMDLL_ERROR_1V5SHORT) printf("1.5V Short!\n");

	}
	showpwr(&pem);

ERR:
	pem.EXIT();
	return 0;
}
```

## VC++ MFC Sample

Please fine the project in the **Program files/Soliton/Pem1x/DLLDev/VC/PemWin** folder.

# 5. Contact

Please contact us if you have experienced any problems.

E-mail: info@soliton.com.tw
Tel: +886-3-6566996
Fax: +886-3-6566883