

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	1 st Release	Oct/05/2005	Vincent

1. Installation.....	4
2. Function Description.....	4
int CPEMDll::INIT()	4
int CPEMDll::INITEX()	4
int CPEMDll::INITEX(DWORD addr)	4
int CPEMDll::EXIT()	5
int CPEMDll::SELPEN(DWORD addr)	5
int CPEMDll::VALIDPEN(DWORD addr)	5
int CPEMDll::VALIDDEV()	6
int CPEMDll::VALIDDEV(DWORD addr)	6
int CPEMDll::SELDEV(DWORD Vid, DWORD Did, int index)	6
int CPEMDll::SELDEV(DWORD addr, DWORD Vid, DWORD Did, int index)	6
int CPEMDll::SELDEV(DWORD Busno, DWORD Devno)	7
int CPEMDll::SELDEV(DWORD addr, DWORD Busno, DWORD Devno)	7
int CPEMDll::PON()	8
int CPEMDll::PON(DWORD addr)	8
int CPEMDll::POFF()	8
int CPEMDll::POFF(DWORD addr)	8
int CPEMDll::GETPWRSTS()	9
int CPEMDll::GETPWRSTS(DWORD addr)	9
int CPEMDll::CHKSHORT()	9
int CPEMDll::CHKSHORT(DWORD addr)	9
int CPEMDll::GET3V3V(double* volatge)	10
int CPEMDll::GET3V3V(DWORD addr, double* volatge)	10
int CPEMDll::GET1V5V(double* volatge)	10
int CPEMDll::GET1V5V(DWORD addr, double* volatge)	10
int CPEMDll::GET3V3I(double* current)	11
int CPEMDll::GET3V3I(DWORD addr, double* current)	11
int CPEMDll::GET1V5I(double* current)	11
int CPEMDll::GET1V5I(DWORD addr, double* current)	11
int CPEMDll::WINEN(int delaytime)	12
int CPEMDll::WINEN(DWORD addr, int delaytime)	12
int CPEMDll::WINDIS(int delaytime)	13
int CPEMDll::WINDIS(DWORD addr, int delaytime)	13
int CPEMDll::WINCHECK()	13
int CPEMDll::WINDIS(DWORD addr)	13
int CPEMDll::WAITCARDREMOVE()	14

int CPEMDll::WAITCARDREMOVE(DWORD addr)	14
int CPEMDll::WAITCARDPLUG()	14
int CPEMDll::WAITCARDPLUG(DWORD addr)	14
int CPEMDll::LEDGO()	15
int CPEMDll::LEDGO(DWORD addr)	15
int CPEMDll::LEDNG()	15
int CPEMDll::LEDNG(DWORD addr)	15
int CPEMDll::LEDOFF()	16
int CPEMDll::LEDOFF(DWORD addr)	16
int CPEMDll::BEEP(int freq, int time)	16
int CPEMDll::BEEP(DWORD addr, int freq, int time)	16
3. Operation Flow	18
Initialize Flow	18
Operation/Test Flow	19
4. Sample Program	20
VC++ Sample	20
VC++ MFC Sample	22
5. Contact	23

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: 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:

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

int CPEMDll::INITEX()

int CPEMDll::INITEX(DWORD addr)

Syntax:

Form1: int CPEMDll::INITEX()
Form2: int CPEMDll::INITEX(DWORD addr)

Description:

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

Input Value:

Form1: None
Form2: 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:

```
CPEMDLL pemctl;  
DWORD addr = 0;
```

```
Form1: pemctl.INITEX();  
Form2: 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:

```
CPEMDLL pemctl;  
pemctl.EXIT();
```

int CPEMDll::SELPEM(DWORD addr)

Syntax: 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:

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

int CPEMDll::VALIDPEM(DWORD addr)

Syntax: 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 valid; -1 if error.

Usage:

```
CPEMDLL pemctl;  
DWORD addr = 0;
```

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

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

Syntax:

Form1: int CPEMDll::VALIDDEV()
Form2: int CPEMDll::VALIDDEV(DWORD addr)

Description:

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

Input Value:

Form1: None
Form2: 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:

```
CPEMDLL pemctl;  
DWORD addr = 0;  
int status;  
Form1: status = pemctl.VALIDDEV();  
Form2: 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:

Form1: int CPEMDll::SELDEV(DWORD Vid, DWORD Did, int index)
Form2: 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:

Form1: Vid Vendor ID. Did Device ID. Index Card number.
Form2: 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:

```
CPEMDLL pemctl;  
DWORD addr = 0;  
DWORD Vid = 0x1234;  
DWORD Did = 0;  
int index = 0  
Form1: pemctl.SELDEV(Vid, Did, index);  
Form2: pemctl.SELDEV(addr, Vid, DWORD Did, index);
```

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

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

Syntax:

```
Form1: int CPEMDll::SELDEV(DWORD Busno, DWORD Devno)  
Form2: int CPEMDll::SELDEV(DWORD addr, DWORD Busno, DWORD Devno)
```

Description:

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

Input Value:

Form1: Busno	Bus number.
Devno	Device number.
Form2: 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:

```
CPEMDLL pemctl;  
DWORD addr = 0;  
DWORD Busno = 3;  
DWORD Devno = 0;  
Form1: pemctl.SELDEV(Busno, Devno);  
Form2: pemctl.SELDEV(addr, Busno, Devno);
```

int CPEMDll::PON()

int CPEMDll::PON(DWORD addr)

Syntax:

Form1: int CPEMDll::PON ()

Form2: int CPEMDll::PON(DWORD addr)

Description:

Turn the specified PEM-1X module power on.

Input Value:

Form1: None

Form2: DWORD addr: module address, range 0~3

Return Value:

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

Usage:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl.PON ();

Form2: pemctl.PON(addr);

int CPEMDll::POFF()

int CPEMDll::POFF(DWORD addr)

Syntax:

Form1: int CPEMDll::POFF(

Form2: int CPEMDll::POFF(DWORD addr)

Description:

Turn the specified PEM-1X module power off.

Input Value:

Form1: None

Form2: DWORD addr: module address, range 0~3

Return Value:

Return 0 if successful; 1 if error.

Usage:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl.POFF ();

Form2: pemctl.POFF(addr);

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

Syntax:

Form1: int CPEMDll::GETPWRSTS()

Form2: int CPEMDll::GETPWRSTS(DWORD addr)

Description:

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

Input Value:

Form1: None

Form2: DWORD addr: module address, range 0~3

Return Value:

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

Usage:

CPEMDLL pemctl;

DWORD addr = 0;

int status;

Form1: status = pemctl.GETPWRSTS();

Form2: status = pemctl.GETPWRSTS(addr);

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

Syntax:

Form1: int CPEMDll::CHKSHORT()

Form2: int CPEMDll::CHKSHORT(DWORD addr)

Description:

Check which power rail is shorted.

Input Value:

Form1: None

Form2: 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:

CPEMDLL pemctl;

```
DWORD addr = 0;
int sstatus;
Form1: sstatus = pemctl.CHKSHORT();
Form2: sstatus = pemctl.CHKSHORT(addr);
```

int CPEMDll::GET3V3V(double* volatge)

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

Syntax:

```
Form1: int CPEMDll::GET3V3V(double* volatge)
Form2: int CPEMDll::GET3V3V(DWORD addr, double* volatge)
```

Description:

Get 3.3V rail voltage reading.

Input Value:

```
Form1: (double* volatge)
Form2: (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:

```
CPEMDLL pemctl;
DWORD addr = 0;
double voltage;
Form1: pemctl.GET3V3V(&volatge);
Form2: pemctl.GET3V3V(addr, &volatge);
```

int CPEMDll::GET1V5V(double* volatge)

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

Syntax:

```
Form1: int CPEMDll::GET1V5V(double* volatge)
Form2: int CPEMDll::GET1V5V(DWORD addr, double* volatge)
```

Description:

Get 1.5V rail voltage reading.

Input Value:

```
Form1: (double* volatge)
```

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

double voltage;

Form1: pemctl. GET1V5V(&volatge);

Form2: pemctl. GET1V5V(addr, &volatge);

int CPEMDll::GET3V3I(double* current)

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

Syntax:

Form1: int CPEMDll::GET3V3I(double* current)

Form2: int CPEMDll::GET3V3I(DWORD addr, double* current)

Description:

Get 3.3V rail current reading.

Input Value:

Form1: (double* current)

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

double current;

Form1: pemctl. GET3V3I (¤t);

Form2: pemctl. GET3V3I (addr, & current);

int CPEMDll::GET1V5I(double* current)

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

Syntax:

Form1: int CPEMDll::GET1V5I(double* current)

Form2: int CPEMDll::GET1V5I(DWORD addr, double* current)

Description:

Get 1.5V rail current reading.

Input Value:

Form1: (double* current)

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

double current;

Form1: pemctl.GET1V5I(¤t);

Form2: pemctl.GET1V5I(addr, & current);

int CPEMDll::WINEN(int delaytime)

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

Syntax:

Form1: int CPEMDll::WINEN(int delaytime)

Form2: int CPEMDll::WINEN(DWORD addr, int delaytime)

Description:

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

Input Value:

Form1: delaytime delay between enabling each device. The unit is milli-second

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl.WINEN(100);

Form2: pemctl.WINEN(addr,100);

int CPEMDll::WINDIS(int delaytime)

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

Syntax:

Form1: int CPEMDll::WINDIS(int delaytime)

Form2: int CPEMDll::WINDIS(DWORD addr, int delaytime)

Description:

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

Input Value:

Form1: (int delaytime)

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl. WINDIS (100);

Form2: pemctl. WINDIS (addr,100);

int CPEMDll::WINCHECK()

int CPEMDll::WINDIS(DWORD addr)

Syntax:

Form1: int CPEMDll::WINCHECK(int delaytime)

Form2: int CPEMDll::WINCHECK (DWORD addr, int delaytime)

Description:

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

Input Value:

Form1: None

Form2: (DWORD addr)

DWORD addr: module address, range 0~3

Return Value:

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

Usage:

CPEMDLL pemctl;

```
Int status;  
DWORD addr = 0;  
Form1: status = pemctl. WINCHECK ();  
Form2: status = pemctl. WINCHECK (addr);
```

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

Syntax:

```
Form1: int CPEMDll::WAITCARDREMOVE()  
Form2: int CPEMDll::WAITCARDREMOVE (DWORD addr)
```

Description:

Wait the PCI-Express Card on slot is removed.

Input Value:

```
Form1: None  
Form2: (DWORD addr)  
        DWORD addr: module address, range 0~3
```

Return Value:

Return 0 if successful.

Usage:

```
CPEMDLL pemctl;  
DWORD addr = 0;  
Form1: while(pemctl.WAITCARDREMOVE())  
Form2: while(pemctl.WAITCARDREMOVE(addr))
```

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

Syntax:

```
Form1: int CPEMDll::WAITCARDPLUG()  
Form2: int CPEMDll::WAITCARDPLUG(DWORD addr)
```

Description:

Wait the PCI-Express Card on slot is plugged.

Input Value:

```
Form1: None  
Form2: (DWORD addr)  
        DWORD addr: module address, range 0~3
```

Return Value:

Return 0 if successful.

Usage:

CPEMDLL pemctl;
DWORD addr = 0;
Form1: while(pemctl. WAITCARDPLUG ())
Form2: while(pemctl. WAITCARDPLUG (addr))

int CPEMDll::LEDGO()

int CPEMDll::LEDGO(DWORD addr)

Syntax:

Form1: int CPEMDll::LEDGO()
Form2: int CPEMDll::LEDGO(DWORD addr)

Description:

Turn on the GO LED to indicate the test status.

Input Value:

Form1: None
Form2: (DWORD addr)
DWORD addr: module address, range 0~3

Return Value:

Return 0 if successful; 1 if error.

Usage:

CPEMDLL pemctl;
DWORD addr = 0;
Form1: pemctl.LEDGO();
Form2: pemctl.LEDGO(addr);

int CPEMDll::LEDNG()

int CPEMDll::LEDNG(DWORD addr)

Syntax:

Form1: int CPEMDll::LEDNG()
Form2: int CPEMDll::LEDNG(DWORD addr)

Description:

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

Input Value:

Form1: None
Form2: (DWORD addr)

DWORD addr: module address, range 0~3

Return Value:

Return 0 if successful; 1 if error.

Usage:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl.LEDNG();

Form2: pemctl.LEDNG(addr);

int CPEMDll::LEDOFF()

int CPEMDll::LEDOFF(DWORD addr)

Syntax:

Form1: int CPEMDll::LEDOFF()

Form2: int CPEMDll::LEDOFF(DWORD addr)

Description:

Turn off both the GO and NG LEDs.

Input Value:

Form1: None

Form2: (DWORD addr)

DWORD addr: module address, range 0~3

Return Value:

Return 0 if successful; 1 if error.

Usage:

CPEMDLL pemctl;

DWORD addr = 0;

Form1: pemctl.LEDOFF();

Form2: pemctl.LEDOFF(addr);

int CPEMDll::BEEP(int freq, int time)

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

Syntax:

Form1: int CPEMDll::BEEP(int freq, int time)

Form2: int CPEMDll::BEEP(DWORD addr, int freq, int time)

Description:

Input Value:

Form1: (int freq, int time)

Form2: (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:

CPEMDLL pemctl;

DWORD addr = 0;

int freq = 1

int time = 1

Form1: pemctl.BEEP(freq, time);

Form2: pemctl.BEEP(addr, freq, time);

3. Operation Flow

Initialize Flow

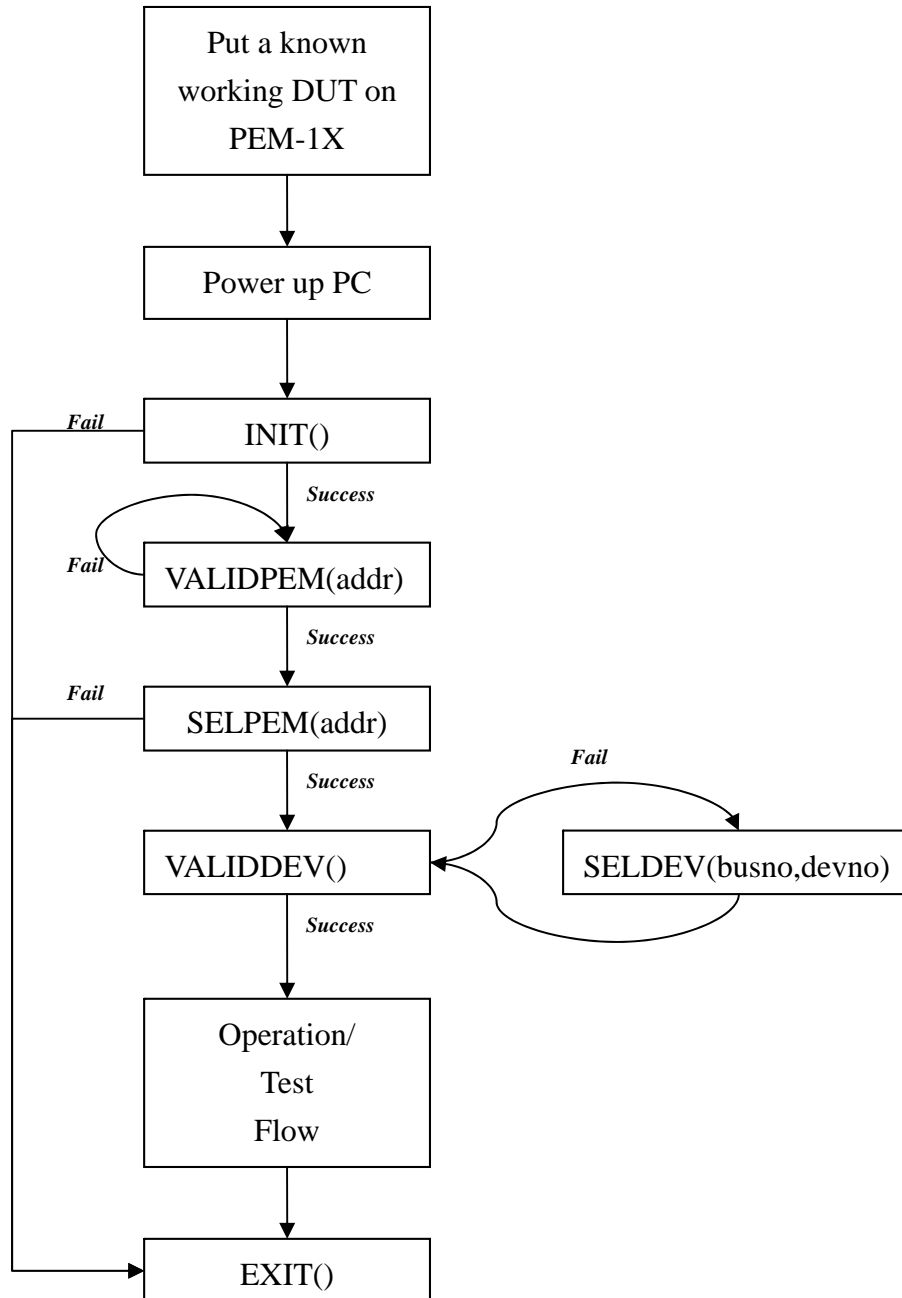


Fig. 1: Initialize Flow

Operation/Test Flow

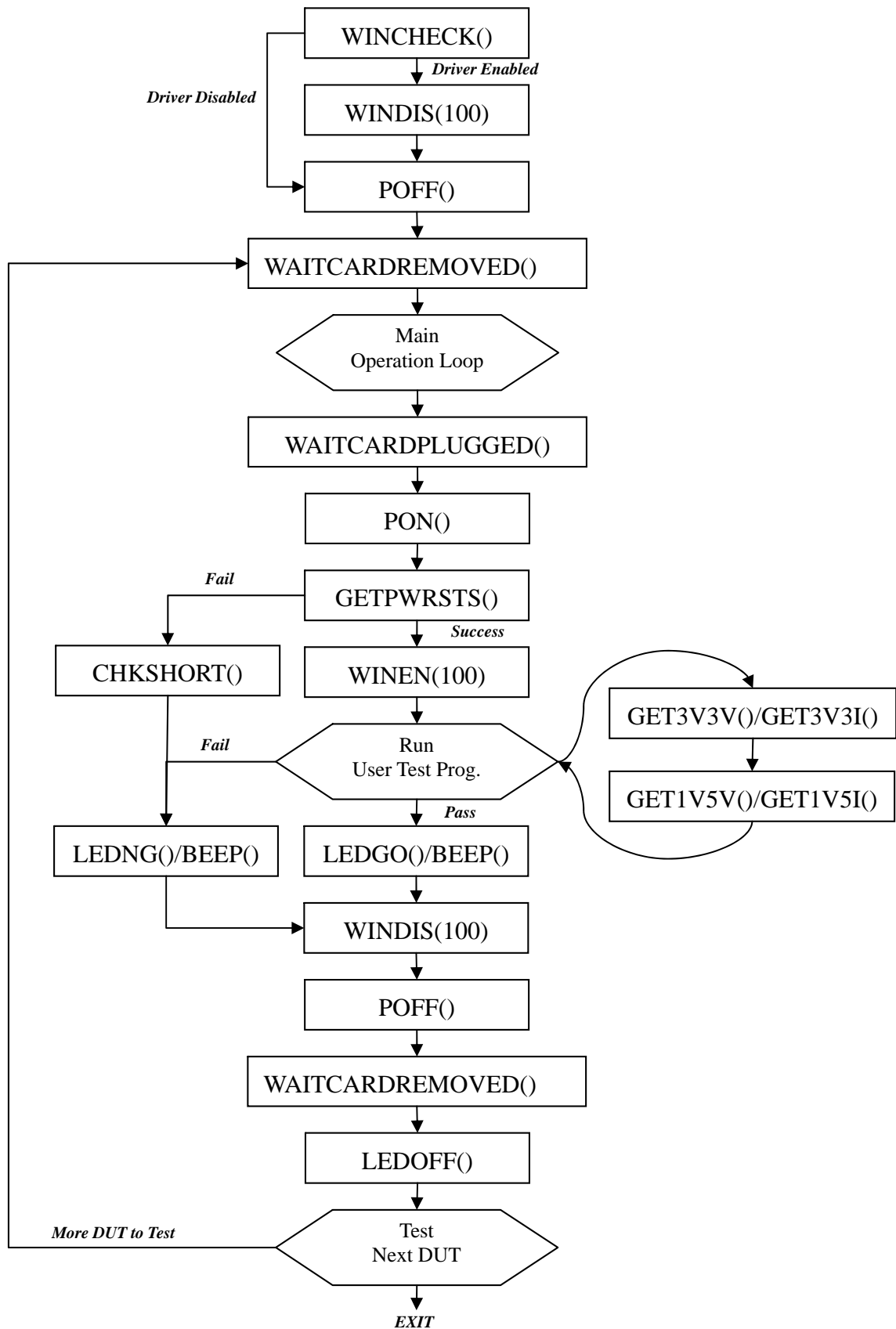


Fig. 2: Operation/Test Flow

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