

Soliton PCI Express Test Solution

PETS COM Component Programming Description

Rev 1.2
July 27, 2006

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
1.1	Add PEX and PEC support	Dec/30/2005	Vincent
1.2	Add property, meas3v3imax, meas1v5imax, and ponrst	Jul/27/2006	Vincent

1. Installation.....	4
2. METHOD Description.....	4
STDMETHODIMP CPemXCtrl::Init().....	4
STDMETHODIMP CPemXCtrl::Exit().....	4
STDMETHODIMP CPemXCtrl::SelPem(long maddr).....	4
STDMETHODIMP CPemXCtrl::SelDevByID(long vid, long did, long index)	5
STDMETHODIMP CPemXCtrl::SelDevBySlot(long busno, long devno).....	5
STDMETHODIMP CPemXCtrl::WinEn(long delay)	6
STDMETHODIMP CPemXCtrl::WinDis(long delay)	6
STDMETHODIMP CPemXCtrl::LedGo()	6
STDMETHODIMP CPemXCtrl::LedNg()	7
STDMETHODIMP CPemXCtrl::LedOff().....	7
STDMETHODIMP CPemXCtrl::Beep(long freq, long time)	7
STDMETHODIMP CPemXCtrl::POn()	8
STDMETHODIMP CPemXCtrl::POff().....	8
STDMETHODIMP CPemXCtrl::ICal()	8
3. Properties Description.....	9
Property: currvid	9
Property: currdid	9
Property: currbusno.....	9
Property: currdevno.....	10
Property: pntype	10
Property: powerstatus.....	11
Property: chkshort	11
Property: cardstatus.....	12
Property: validdev	12
Property: validpem.....	13
Property: meas3v3v.....	13
Property: meas3v3i	14
Property: meas1v5v.....	14
Property: meas1v5i	15
Property: meas12v.....	15
Property: meas12i	16
Property: pemcount	16
Property: drvenabled	17
Property: pemstatus.....	17
Property: usecal.....	18

Property: meas3v3imax.....	18
Property: meas1v5imax.....	18
Property: ponrst.....	19
3. Operation Flow	20
Initialize Flow	20
Operation/Test Flow.....	21
4. Sample Program.....	22
C# Sample.....	22
VB Sample	22
VB.Net Sample	22
5. Contact	22

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 PETS control. Please find all the needed samples and development resources in the **Program files/Soliton/Pem1x/DLLDev** folder.

2. METHOD Description

STDMETHODIMP CPemXCtrl::Init()

Syntax : HRESULT CPemXCtrl::Init()

Description :

Initialize all the PETS cards on mainboard.

Input Value: None

Return Value :

Return S_OK if successful.

Usage :

```
CPemXCtrl pemctl;  
pemctl.Init();
```

STDMETHODIMP CPemXCtrl::Exit()

Syntax : HRESULT CPemXCtrl::Exit()

Description :

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

Input Value: None

Return Value :

Return S_OK if successful.

Usage :

```
CPemXCtrl pemctl;  
pemctl.Exit();
```

STDMETHODIMP CPemXCtrl::SelPem(long maddr)

Syntax : HRESULT CPemXCtrl::SelPciDev(long maddr)

Description :

Select handle for PETS module on mainboard. If success, user can call other operational function without specifying the module address.

Input Value: maddr: Module addr, range 0~3.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.SelPciDev (long maddr);
```

STDMETHODIMP CPemXCtrl::SelDevByID(long vid, long did, long index)

Syntax : HRESULT CPemXCtrl::SelDevByID(long vid, long did, long 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: Vid Vendor ID.
 Did Device ID.
 index. Card number.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.SelDevByID(long vid, long did, long index)
```

STDMETHODIMP CPemXCtrl::SelDevBySlot(long busno, long devno)

Syntax : HRESULT CPemXCtrl::SelDevBySlot(long busno, long devno)

Description :

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

Input Value: busno Bus Number.
 devno Device Number.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.SelDevBySlot(long busno, long devno)
```

STDMETHODIMP CPemXCtrl::WinEn(long delay)

Syntax : HRESULT CPemXCtrl::WinEn(long delay)

Description :

Enable the windows driver of the specified PCI-Express Device on PETS module.

Input Value: delay Delay time between every drivers being enabled.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.WinEn(long delay)
```

STDMETHODIMP CPemXCtrl::WinDis(long delay)

Syntax : HRESULT CPemXCtrl::WinDis(long delay)

Description :

Disable the windows driver of the specified PCI-Express Device on PETS module.

Input Value: delay Delay time between every drivers being enabled.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.WinDis(long delay)
```

STDMETHODIMP CPemXCtrl::LedGo()

Syntax : HRESULT CPemXCtrl::LedGo()

Description :

Turn on the GO LED to indicate the test status.

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.LedGo()
```

STDMETHODIMP CPemXCtrl::LedNg()

Syntax : HRESULT CPemXCtrl::LedNg()

Description :

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

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.LedNg()
```

STDMETHODIMP CPemXCtrl::LedOff()

Syntax : HRESULT CPemXCtrl::LedOff()

Description :

Turn off both the GO and NG LEDs.

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl.LedOff()
```

STDMETHODIMP CPemXCtrl::Beep(long freq, long time)

Syntax : HRESULT CPemXCtrl::Beep(long freq, long time)

Description :

Play Beeper.

Input Value: freq

Beeper Frequency.

0x00: Highest, 0x01: High; 0x02: Low; 0x03: Lowest.

time

Beeper Time.

0x00:500 msec; 0x01:1000 msec ;0x02: 1500 msec ;0x03:2000 msec.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

```
CPemXCtrl pemctl;  
pemctl. Beep(long freq, long time)
```

STDMETHODIMP CPemXCtrl::POn()

Syntax : HRESULT CPemXCtrl::POn()

Description :

Turn the specified PETS module power on.

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

CPemXCtrl pemctl;
pemctl.POn()

STDMETHODIMP CPemXCtrl::POff()

Syntax : HRESULT CPemXCtrl::POff()

Description :

Turn the specified PETS module power off.

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

CPemXCtrl pemctl;
pemctl.POff()

STDMETHODIMP CPemXCtrl::ICal()

Syntax : HRESULT CPemXCtrl::ICal()

Description :

Calibrate every current rail, calculate the current offset when no card on the PETS module.

Need to remove DUT card on PETS module.

Input Value: None.

Return Value :

Return S_OK if successful, S_FALSE if error.

Usage :

CPemXCtrl pemctl;
pemctl. ICal ()

3. Properties Description

Property: currvid

Syntax : HRESULT get_currvid([out, retval] long *pVal);

Description :

Get the Vendor ID of current selected Device on PETS module.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_currvid(&val);
```

VB:

```
CPemXCtrl pemctl;  
currvid = pemctl.currvid;
```

C#:

```
CPemXCtrl pemctl;  
currvid = pemctl.currvid;
```

Property: currdid

Syntax : HRESULT get_currdid([out, retval] long *pVal);

Description :

Get the Device ID of current selected Device on PETS module.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_currdid (&val);
```

VB:

```
CPemXCtrl pemctl;  
currdid = pemctl.currdid;
```

C#:

```
CPemXCtrl pemctl;  
currdid = pemctl.currdid;
```

Property: currbusno

Syntax : HRESULT get_currbusno([out, retval] long *pVal);

Description :

Get the Bus number of current selected Device on PETS module.

Usage :**C++**

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_currbusno (&val);
```

VB:

```
CPemXCtrl pemctl;  
currbusno = pemctl.currbusno;
```

C#:

```
CPemXCtrl pemctl;  
currbusno = pemctl.currbusno;
```

Property: currdevno

Syntax : HRESULT get_currdevno([out, retval] long *pVal);

Description :

Get the Device number of current selected Device on PETS module.

Usage :**C++**

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_currdevno(&val);
```

VB:

```
CPemXCtrl pemctl;  
currdevno = pemctl.currdevno;
```

C#:

```
CPemXCtrl pemctl;  
currdevno = pemctl.currdevno;
```

Property: pntype

Syntax : HRESULT get_pntype([out, retval] long *pVal);

Description :

Get the product model number.

Return Value:

0 if PEM-1L
1 if PEX-1L

2 if PEC-1L

3 if PEX-16L

Property: powerstatus

Syntax : HRESULT get_powerstatus([out, retval] long *pVal);

Description :

Get the power status of the specified PETS module.

Return Value:

0 if power is off.

1 if power is on.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_ powerstatus (&val);
```

VB:

```
CPemXCtrl pemctl;  
powerstatus = pemctl.powerstatus;
```

C#:

```
CPemXCtrl pemctl;  
powerstatus = pemctl.powerstatus;
```

Property: chkshort

Syntax : HRESULT get_chkshort([out, retval] long *pVal);

Description :

Check which power rail is shorted.

Return Value:

-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 :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_ chkshort (&val);
```

VB:

```
CPemXCtrl pemctl;  
chkshort = pemctl.chkshort;
```

C#:

```
CPemXCtrl pemctl;  
chkshort = pemctl.chkshort;
```

Property: cardstatus

Syntax : HRESULT get_cardstatus([out, retval] long *pVal);

Description :

Check if the PCI-Express Card are plugged on PETS module.

Return Value:

0 if card is removed.
1 if card is plugged.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_cardstatus (&val);
```

VB:

```
CPemXCtrl pemctl;  
cardstatus = pemctl.cardstatus;
```

C#:

```
CPemXCtrl pemctl;  
cardstatus = pemctl.cardstatus;
```

Property: validdev

Syntax : HRESULT get_validdev([out, retval] long *pVal);

Description :

Check if the specified PCI-Express Device exist on PETS.

Return Value:

0 if not valid, the specified device not match.
1 if valid.

Usage :

C++

```
long val;
```

```
CPemXCtrl pemctl;  
pemctl.get_validdev (&val);
```

VB:

```
CPemXCtrl pemctl;  
validdev = pemctl.validdev;
```

C#:

```
CPemXCtrl pemctl;  
validdev = pemctl.validdev;
```

Property: validpem

Syntax : HRESULT get_validpem([out, retval] long *pVal);

Description :

Check if the PETS module exist.

Return Value:

0 if not valid, the current selected PETS module not exist.

1 if valid.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_validpem (&val);
```

VB:

```
CPemXCtrl pemctl;  
validpem = pemctl.validpem;
```

C#:

```
CPemXCtrl pemctl;  
validpem = pemctl.validpem;
```

Property: meas3v3v

Syntax : HRESULT get_meas3v3v([out, retval] double *pVal);

Description :

Get 3.3V rail voltage reading.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;
```

```
pemctl.get_meas3v3v (&val);
```

VB:

```
CPemXCtrl pemctl;  
meas3v3v = pemctl.meas3v3v;
```

C#:

```
CPemXCtrl pemctl;  
meas3v3v = pemctl.meas3v3v;
```

Property: meas3v3i

Syntax : HRESULT get_meas3v3i([out, retval] double *pVal);

Description :

Get 3.3V rail current reading.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.get_meas3v3i (&val);
```

VB:

```
CPemXCtrl pemctl;  
meas3v3i = pemctl.meas3v3i;
```

C#:

```
CPemXCtrl pemctl;  
meas3v3i = pemctl.meas3v3i;
```

Property: meas1v5v

Syntax : HRESULT get_meas1v5v([out, retval] double *pVal);

Description :

Get 1.5V rail voltage reading. If the module type is not PEM or PEC series, the read back value will be 0.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.get_meas1v5v (&val);
```

VB:

```
CPemXCtrl pemctl;
```

```
meas1v5v = pemctl.meas1v5v;
```

C#:

```
CPemXCtrl pemctl;  
meas1v5v = pemctl.meas1v5v;
```

Property: meas1v5i

Syntax : HRESULT get_meas1v5i([out, retval] double *pVal);

Description :

Get 1.5V rail current reading. If the module type is not PEM or PEC series, the read back value will be 0.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.get_meas1v5i (&val);
```

VB:

```
CPemXCtrl pemctl;  
meas1v5i = pemctl.meas1v5i;
```

C#:

```
CPemXCtrl pemctl;  
meas1v5i = pemctl.meas1v5i;
```

Property: meas12v

Syntax : HRESULT get_meas12v([out, retval] double *pVal);

Description :

Get 12V rail voltage reading. If the module type is not PEX series, the read back value will be 0.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.get_meas12v (&val);
```

VB:

```
CPemXCtrl pemctl;  
meas1v5v = pemctl.meas12v;
```

C#:

```
CPemXCtrl pemctl;
```

```
meas1v5v = pemctl.meas12v;
```

Property: meas12i

Syntax : HRESULT get_meas12i([out, retval] double *pVal);

Description :

Get 12V rail current reading. If the module type is not PEX series, the read back value will be 0.

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.get_meas12i (&val);
```

VB:

```
CPemXCtrl pemctl;  
meas1v5v = pemctl.meas12i;
```

C#:

```
CPemXCtrl pemctl;  
meas1v5v = pemctl.meas12i;
```

Property: pemcount

Syntax : HRESULT get_pemcount([out, retval] long *pVal);

Description :

Get the total amount of PETS modules found on mainboard.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_pemcount (&val);
```

VB:

```
CPemXCtrl pemctl;  
pemcount = pemctl.pemcount;
```

C#:

```
CPemXCtrl pemctl;  
pemcount = pemctl.pemcount;
```


Property: drvenabled

Syntax : HRESULT get_drvenabled([out, retval] long *pVal);

Description :

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

Return Value:

0 if the driver is disabled

1 if the driver is enabled.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_drvenabled (&val);
```

VB:

```
CPemXCtrl pemctl;  
drvenabled = pemctl.drvenabled;
```

C#:

```
CPemXCtrl pemctl;  
drvenabled = pemctl.drvenabled;
```

Property: pemstatus

Syntax : HRESULT get_pemstatus([out, retval] long *pVal);

Description :

Get PETS status.

Return Value:

0x01 if module address="0" PETS module exist.

0x02 if module address="1" PETS module exist.

0x04 if module address="2" PETS module exist.

0x08 if module address="3" PETS module exist.

Value can be combination of upper values.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.get_pemstatus (&val);
```

VB:

```
CPemXCtrl pemctl;  
pemstatus = pemctl.pemstatus;
```

C#:

```
CPemXCtrl pemctl;  
pemstatus = pemctl.pemstatus;
```

Property: usecal

Syntax : HRESULT put_usecal([out, retval] long *pVal);

Description :

Use Current Calibration.

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.put_usecal (1);
```

VB:

```
CPemXCtrl pemctl;  
pemctl.usecal = 1;
```

C#:

```
CPemXCtrl pemctl;  
pemctl.usecal = 1;
```

Property: meas3v3imax

Syntax : HRESULT get_meas3v3imax([out, retval] double *pVal);

Description :

Get maximum current on 3.3V Rail

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.meas3v3imax (&double);
```

VB:

```
CPemXCtrl pemctl;  
meas3v3imax = pemctl.meas3v3imax;
```

C#:

```
CPemXCtrl pemctl;  
meas3v3imax = pemctl.meas3v3imax;
```

Property: meas1v5imax

Syntax : HRESULT get_meas1v5imax([out, retval] double *pVal);

Description :

Get maximum current on 1.5V Rail

Usage :

C++

```
double val;  
CPemXCtrl pemctl;  
pemctl.meas1v5imax (&double);
```

VB:

```
CPemXCtrl pemctl;  
Meas1v5imax = pemctl.meas1v5imax;
```

C#:

```
CPemXCtrl pemctl;  
Meas1v5imax = pemctl.meas1v5imax;
```

Property: ponrst

Syntax : HRESULT put_ponrst ([out, retval] long *pVal);

Description :

Set Power On Rest Duration.

0: 25ms
1: 50ms (default)
2: 100ms
3: 150ms

Usage :

C++

```
long val;  
CPemXCtrl pemctl;  
pemctl.put_ponrst (1);
```

VB:

```
CPemXCtrl pemctl;  
pemctl.ponrst = 1;
```

C#:

```
CPemXCtrl pemctl;  
pemctl.ponrst = 1;
```

3. Operation Flow

Initialize Flow

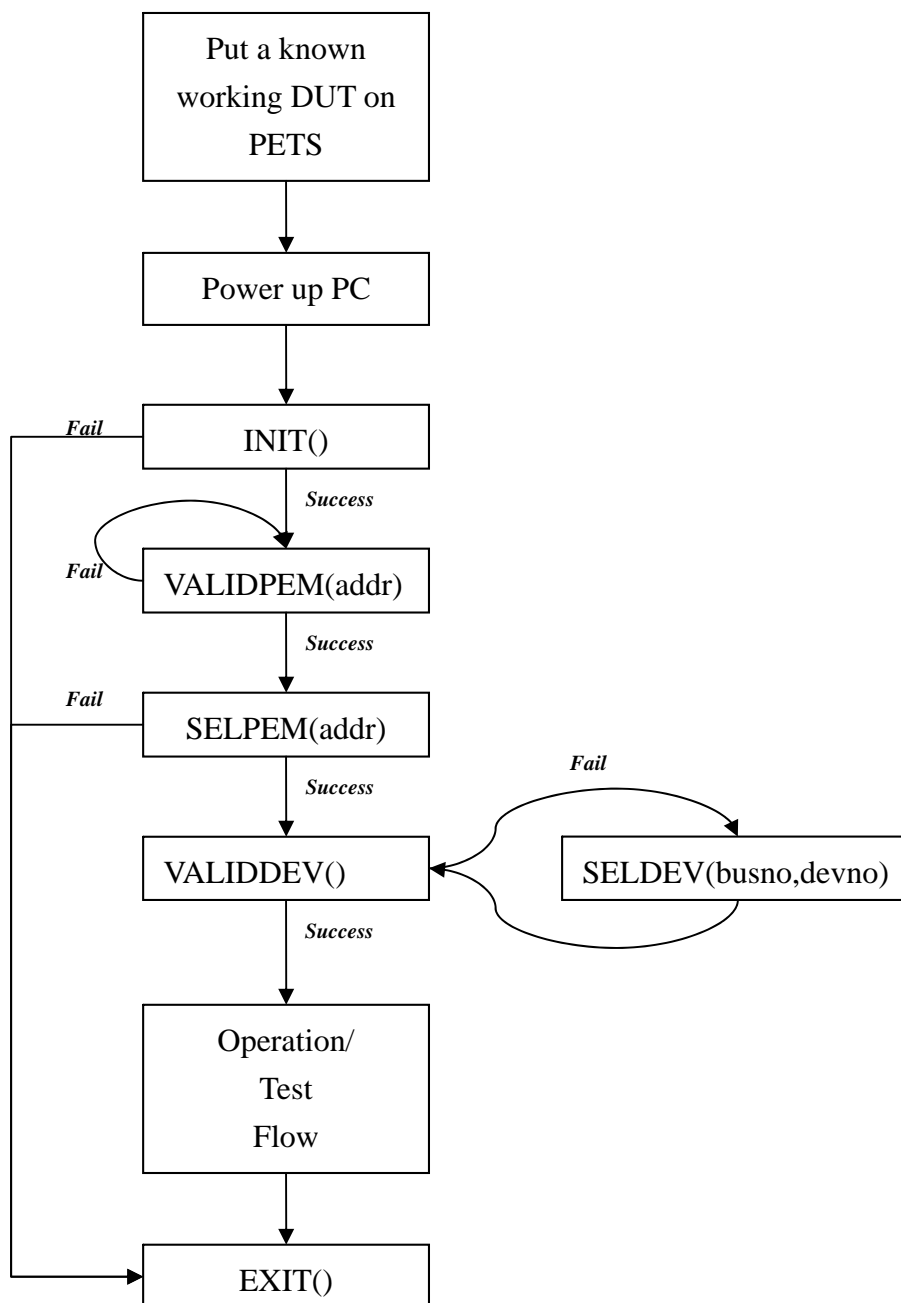


Fig. 1: Initialize Flow

Operation/Test Flow

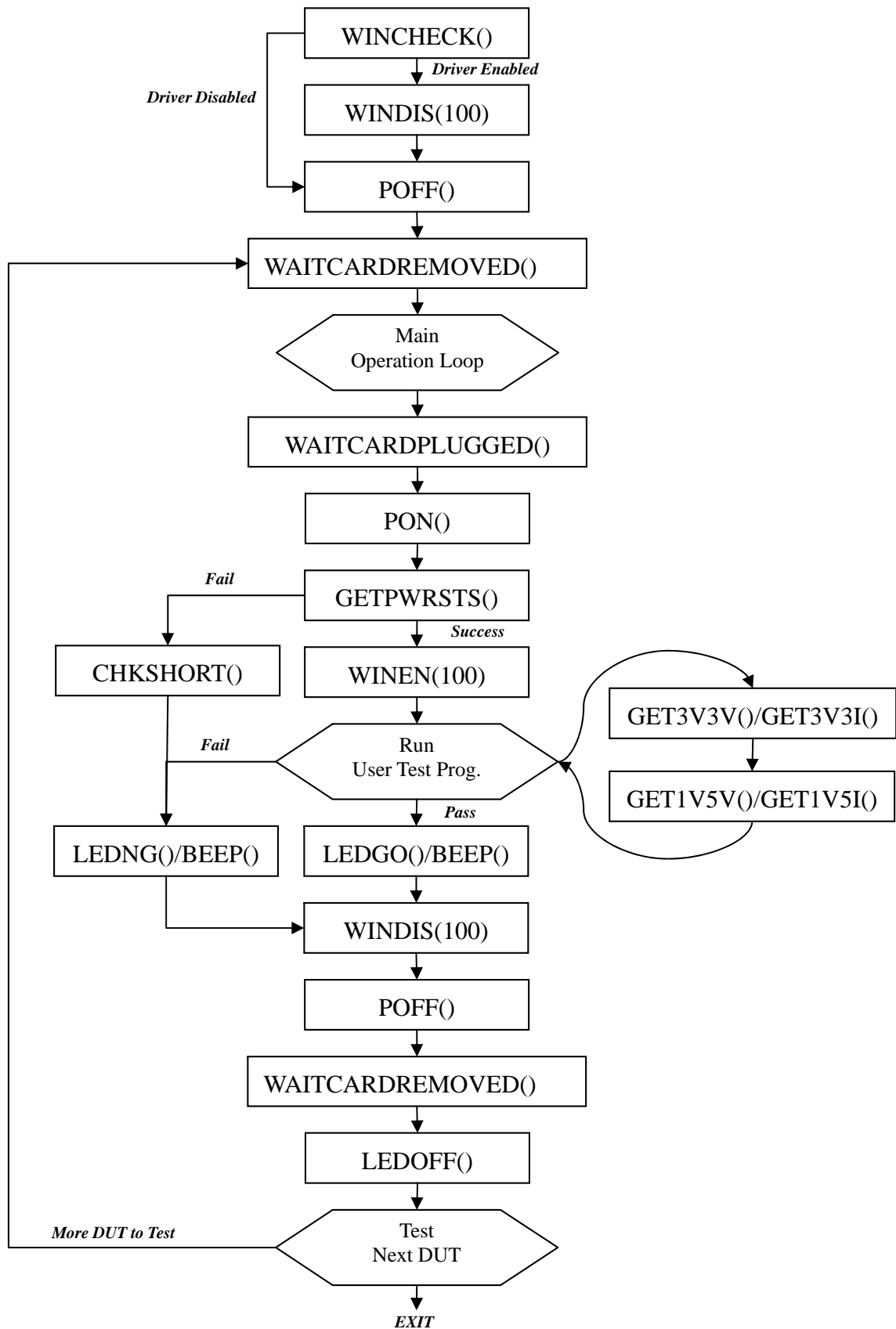


Fig. 2: Operation/Test Flow

4. Sample Program

C# Sample

Please find the project in the **Program files/Soliton/Pem1x/DLLDev/CS.NET** folder.

VB Sample

Please find the project in, **Program files/Soliton/Pem1x/DLLDev/VB**

VB.Net Sample

Please find the project in, **Program files/Soliton/Pem1x/DLLDev/VB.NET**

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