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 __stdcall PETS_INIT()

Syntax:

Form 1: int __stdcall PETS_INIT()

Description:

Initialize all the PEM-1X cards on mainboard.

Input Value: None

Return Value:

Return 0 if successful; 1 if error.

Usage:

Borland C++

PETS_INIT();

int __stdcall PETS_INITEX()
int __stdcall PETS_INITEX1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_INITEX()

Form 2: int __stdcall PETS_INTEX1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_INITEX();

Form 4: PETS_INITEX1(addr);

int __stdcall PETS_EXIT()

Syntax:

int __stdcall PETS_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:

Borland C++

PETS_EXIT();

void __stdcall PETS_GETLIBVER(int* major, int* minor)

Syntax:

void __stdcall PETS_GETLIBVER(int* major, int* minor)

Description:

Get the version number of PETs DLL

Input Value: None

Return Value: None

Usage:

Borland C++

PETS_GETLIBVER(&majorno, &minorno);

int __stdcall PETS_SELPEM(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_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:

Borland C++

```
DWORD addr = 0;
PETS_SELPEM(addr);
```

int __stdcall PETS_VALIDPEM(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_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:

Borland C++

```
DWORD addr = 0;
int status;
status = PETS_VALIDPEM (addr);
```

int __stdcall PETS_VALIDDEV()
int __stdcall PETS_VALIDDEV1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_VALIDDEV()

Form 2: int __stdcall PETS_VALIDDEV1(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:

Borland C++

```
DWORD addr = 0;
int status;
```

Form 3:

```
status = PETS_VALIDDEV();
```

Form 4:

```
status = PETS_VALIDDEV(addr);
```

int __stdcall PETS_SELDEVVD(DWORD Vid, DWORD Did, int index)
int __stdcall PETS_SELDEVVD1(DWORD addr, DWORD Vid, DWORD Did, int index)

Syntax:

Form 1: int __stdcallll PETS_SELDEVVD(DWORD Vid, DWORD Did, int index)

Form 2: int __stdcallll PETS_SELDEVVD1(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:

Borland C++

```
DWORD addr = 0;
DWORD Vid = 0x1234;
DWORD Did = 0;
int index = 0
```

Form 1: PETS_SELDEVVD(Vid, Did, index);

Form2: PETS_SELDEVVD1(addr, Vid, Did, index);

int __stdcall PETS_SELDEVBD(DWORD Busno, DWORD Devno)
int __stdcall PETS_SELDEVBD1(DWORD addr, DWORD Busno, DWORD Devno)

Syntax:

Form 1: int __stdcall PETS_SELDEVBD(DWORD Busno, DWORD Devno)

Form 2: int __stdcall PETS_SELDEVBD1(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:

Borland C++

DWORD addr = 0;

DWORD Busno = 3;

DWORD Devno = 0;

Form 3: PETS_SELDEVBD(Busno, Devno);

Form 4: PETS_SELDEVBD1(addr, Busno, Devno);

int __stdcall PETS_PON()
int __stdcall PETS_PON1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_PON ()

Form 2: int __stdcall PETS_PON1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_PON ();

Form 4: PETS_PON1(addr);

int __stdcall PETS_POFF()
int __stdcall PETS_POFF1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_POFF()

Form 2: int __stdcall PETS_POFF1(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_POFF ();

Form 2: PETS_POFF1(addr);

int __stdcall PETS_GETPWRSTS()
int __stdcall PETS_GETPWRSTS1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_GETPWRSTS()

Form 2: int __stdcall PETS_GETPWRSTS1(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:

Borland C++

DWORD addr = 0;

int status;

Form 3: status = PETS_GETPWRSTS();

Form 4: status = PETS_GETPWRSTS1(addr);

int __stdcall PETS_CHKSHORT()
int __stdcall PETS_CHKSHORT1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_CHKSHORT()

Form 2: int __stdcall PETS_CHKSHORT1(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:

Borland C++

DWORD addr = 0;

int status;

Form 3: status = PETS_CHKSHORT();

Form 4: status = PETS_CHKSHORT1(addr)

int __stdcall PETS_GET3V3V(double* volatge)
int __stdcall PETS_GET3V3V1(DWORD addr, double* volatge)

Syntax:

Form 1: int __stdcall PETS_GET3V3V(double* volatge)

Form 2: int __stdcall PETS_GET3V3V1(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:

Borland C++

DWORD addr = 0;

double voltage;

Form 3: PETS_GET3V3V(&volatge);

Form 4: PETS_GET3V3V1(addr, &volatge);

int __stdcall PETS_GET1V5V(double* volatge)
int __stdcall PETS_GET1V5V1(DWORD addr, double* volatge)

Syntax:

Form 1: int __stdcall PETS_GET1V5V(double* volatge)

Form 2: int __stdcall PETS_GET1V5V1(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:

Borland C++

DWORD addr = 0;

double voltage;

Form 3: PETS_GET1V5V(&volatge);

Form 4: PETS_GET1V5V(addr, &volatge);

int __stdcall PETS_GET12V(double* volatge)
int __stdcall PETS_GET12V1(DWORD addr, double* volatge)

Syntax:

Form 1: int __stdcall PETS_GET12V(double* volatge)

Form 2: int __stdcall PETS_GET12V1(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:

Borland C++

DWORD addr = 0;

double voltage;

Form 3: PETS_GET12V(&volatge);

Form 4: PETS_GET12V1(addr, &volatge);

int __stdcall PETS_GET3V3I(double* current)
int __stdcall PETS_GET3V3I1(DWORD addr, double* current)

Syntax:

Form 1: int __stdcall PETS_GET3V3I(double* current)

Form 2: int __stdcall PETS_GET3V3I1(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:

Borland C++

DWORD addr = 0;

double current;

Form 3: PETS_GET3V3I (&current);

Form 4: PETS_GET3V3I1 (addr, & current);

int __stdcall PETS_GET1V5I(double* current)
int __stdcall PETS_GET1V5I1(DWORD addr, double* current)

Syntax:

Form 1: int __stdcall PETS_GET1V5I(double* current)

Form 2: int __stdcall PETS_GET1V5I1(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:

Borland C++

DWORD addr = 0;

double current;

Form 3: PETS_GET1V5I (&current);

Form 4: PETS_GET1V5I1(addr, & current);

int __stdcall PETS_GET12I(double* current)
int __stdcall PETS_GET12I1(DWORD addr, double* current)

Syntax:

Form 1: int __stdcall PETS_GET12I(double* current)

Form 2: int __stdcall PETS_GET12I1(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:

Borland C++

DWORD addr = 0;

double current;

Form 3: PETS_GET12I (&current);

Form 4: PETS_GET12I1(addr, & current);

int __stdcall PETS_GET3V3IMAX(double* current)
int __stdcall PETS_GET3V3IMAX1(DWORD addr, double* current)

Syntax:

Form 1: int __stdcall PETS_GET3V3IMAX(double* current)

Form 2: int __stdcall PETS_GET3V3IMAX1(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:

Borland C++

DWORD addr = 0;

double current;

Form 3: PETS_GET3V3I MAX(&current);

Form 4: PETS_GET3V3IMAX1(addr, &current);

int __stdcall PETS_GET1V5IMAX(double* current)
int __stdcall PETS_GET1V5IMAX1(DWORD addr, double* current)

Syntax:

Form 1: int __stdcall PETS_GET1V5IMAX(double* current)

Form 2: int __stdcall PETS_GET1V5IMAX1(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:

Borland C++

DWORD addr = 0;

double current;

Form 3: PETS_GET1V5I MAX(&current);

Form 4: PETS_GET1V5IMAX1(addr, &current);

int __stdcall PETS_SET1V5ICAL()
int __stdcall PETS_SET1V5ICAL1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_SET1V5ICAL()

Form 2: int __stdcall PETS_SET1V5ICAL1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_SET1V5ICAL();

Form 4: PETS_SET1V5ICAL1(addr);

int __stdcall PETS_SET3V3ICAL()
int __stdcall PETS_SET3V3ICAL1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_SET3V3ICAL()

Form 2: int __stdcall PETS_SET3V3ICAL1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_SET3V3ICAL();

Form 4: PETS_SET3V3ICAL1(addr);

int __stdcall PETS_SET12ICAL()
int __stdcall PETS_SET12ICAL1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_SET12ICAL()

Form 2: int __stdcall PETS_SET12ICAL1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_SET12ICAL();

Form 4: PETS_SET12ICAL1(addr);

int __stdcall PETS_SETUSECAL(int i)
int __stdcall PETS_SETUSECAL1(DWORD addr, int i)

Syntax:

Form 1: int __stdcall PETS_SETUSECAL(int i)

Form 2: int __stdcall PETS_SETUSECAL1(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:

Borland C++

int i = 0;

DWORD addr = 0;

Form 1: PETS_ SETUSECAL (i);

Form 2: PETS_ SETUSECAL 1(addr, i);

int __stdcall PETS_GETUSECAL()
int __stdcall PETS_GETUSECAL1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_GETUSECAL()

Form 2: int __stdcall PETS_GETUSECAL1(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_GETUSECAL();

Form 2: PETS_GETUSECAL1(addr);

int __stdcall PETS_WINEN(int delaytime)
int __stdcall PETS_WINEN1(DWORD addr, int delaytime)

Syntax:

Form 1: int __stdcall PETS_WINEN(int delaytime)

Form 2: int __stdcall PETS_WINEN1(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_WINEN(100);

Form 2: PETS_WINEN1(addr,100);

int __stdcall PETS_WINEN2(int delaytime)
int __stdcall PETS_WINEN21(DWORD addr, int delaytime)

Syntax:

Form 1: int __stdcall PETS_WINEN2(int delaytime)

Form 2: int __stdcall PETS_WINEN21(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_WINEN2(100);

Form 2: PETS_WINEN21(addr,100);

int __stdcall PETS_WINDIS(int delaytime)
int __stdcall PETS_WINDIS1(DWORD addr, int delaytime)

Syntax:

Form 1: int __stdcall PETS_WINDIS(int delaytime)

Form 2: int __stdcall PETS_WINDIS1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_WINDIS(100);

Form 4: PETS_WINDIS1(addr,100);

int __stdcall PETS_WINCHECK()
int __stdcall PETS_ WINCHECK1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_WINCHECK()

Form 2: int __stdcall PETS_WINCHECK1(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:

Borland C++

Int status;

DWORD addr = 0;

Form 1: status = PETS_WINCHECK();

Form 2: status = PETS_WINCHECK1(addr);

int __stdcall PETS_WAITCARDREMOVE()
int __stdcall PETS_WAITCARDREMOVE1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_WAITCARDREMOVE()

Form 2: int __stdcall PETS_WAITCARDREMOVE1(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:

Borland C++

DWORD addr = 0;

Form 1: while(PETS_WAITCARDREMOVE())

Form 2: while(PETS_WAITCARDREMOVE1(addr))

int __stdcall PETS_WAITCARDPLUG()
int __stdcall PETS_WAITCARDPLUG1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_WAITCARDPLUG()

Form 2: int __stdcall PETS_WAITCARDPLUG1(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:

Borland C++

DWORD addr = 0;

Form 3: while(PETS_WAITCARDPLUG())

Form 4: while PETS_WAITCARDPLUG1(addr))

int __stdcall PETS_CHECKCD()
int __stdcall PETS_CHECKCD1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_CHECKCD()

Form 2: int __stdcall PETS_CHECKCD1(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_CHECKCD()

Form 2: PETS_CHECKCD1(addr)

int __stdcall PETS_LEDGO()
int __stdcall PETS_LEDGO1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_LEDGO()

Form 2: int __stdcall PETS_LEDGO1(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:

Borland C++

DWORD addr = 0;

Form 1: PETS_LEDGO();

Form 2: PETS_LEDGO1(addr);

int __stdcall PETS_LEDNG()
int __stdcall PETS_LEDNG1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_LEDNG()

Form 2: int __stdcall PETS_LEDNG1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_LEDNG();

Form 4: PETS_LEDNG1(addr);

int __stdcall PETS_LEDOFF()
int __stdcall PETS_LEDOFF1(DWORD addr)

Syntax:

Form 1: int __stdcall PETS_LEDOFF()

Form 2: int __stdcall PETS_LEDOFF1(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:

Borland C++

DWORD addr = 0;

Form 3: PETS_LEDOFF();

Form 4: PETS_LEDOFF1(addr);

int __stdcall PETS_BEEP(int freq, int time)
int __stdcall PETS_BEEP1(DWORD addr, int freq, int time)

Syntax:

Form 1: int __stdcall PETS_BEEP(int freq, int time)

Form 2: int __stdcall PETS_BEEP1(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:

Borland C++

DWORD addr = 0;

int freq = 1

int time = 1

Form 3: PETS_BEEP(freq, time);

Form 4: PETS_BEEP1(addr, freq, time);

int __stdcall PETS_SETPONRST(int setting)
int __stdcall PETS_SETPONRST1(DWORD addr, int setting)

Syntax:

Form 1: int __stdcall PETS_SETPONRST(int setting)

Form 2: int __stdcall PETS_SETPONRST1(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:

int setting = 0;

Form 1: pemctl.SETPONRST(setting);

Form 2: pemctl.SETPONRST(addr, setting);

Borland C++

DWORD addr = 0;

int setting = 0

Form 1: PETS_SETPONRST(setting);

Form 2: PETS_SETPONRST1(addr, setting);

int __stdcall PROBESMBCLKRISE(double* time)
int __stdcall PROBESMBCLKRISE1(DWORD addr, double* time)

Syntax:

Form1: int __stdcall PETS_PROBESMBCLKRISE(double* time)

Form 2: int __stdcall PETS_PROBESMBCLKRISE1(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:

Borland C++

DWORD addr = 0;

double time;

Form 3: PETS_PROBESMBCLKRISE(&time);

Form 4: PETS_PROBESMBCLKRISE1(addr, & time);

int __stdcall PETS_ GPIOSETUP(int pinno, int mode, int val)
int __stdcall PETS_ GPIOSETUP1(DWORD addr, int pinno, int mode, int val)

Syntax:

Form 1: int __stdcall PETS_ GPIOSETUP(int pinno, int mode, int val)

Form 2: int __stdcall PETS_ GPIOSETUP1(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:

Borland C++

DWORD addr = 0;
int pinno = 0;
int mode = 0;
int val = 0;
Form 3: PETS_ GPIOSETUP(pinno, mode, val);
Form 4: PETS_ GPIOSETUP1(addr, pinno, mode, val);

int __stdcall PETS_GPIOOUTSET (int pinno)
int __stdcall PETS_GPIOOUTSET1(DWORD addr, int pinno)

Syntax:

Form 1: int __stdcall PETS_GPIOOUTSET(int pinno)
Form 2: int __stdcall PETS_GPIOOUTSET1(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:

Borland C++

DWORD addr = 0;
int pinno = 0;
Form 3: PETS_GPIOOUTSET(pinno);
Form 4: PETS_GPIOOUTSET1(addr, pinno);

int __stdcall PETS_GPIOOUTRESET(int pinno)
int __stdcall PETS_GPIOOUTRESET1(DWORD addr, int pinno)

Syntax:

Form 1: int __stdcall PETS_GPIOOUTRESET (int pinno)

Form 2: int __stdcall PETS_GPIOOUTRESET1(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:

Borland C++

DWORD addr = 0;

int pinno = 0;

Form 1: PETS_GPIOOUTRESET(pinno);

Form 2: PETS_GPIOOUTRESET1(addr, pinno);

int __stdcall PETS_GPIOIN(int pinno)
int __stdcall PETS_GPIOIN(DWORD addr, int pinno)

Syntax:

Form 1: int __stdcall PETS_GPIOIN(int pinno)

Form 2: int __stdcall PETS_GPIOIN1(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:

Borland C++

DWORD addr = 0;

int pinno = 0;

Form 1: PETS_GPIOIN pinno);

Form 2: PETS_GPIOIN1(addr, pinno);

int __stdcall PETS_GPIOGETSETUP(int pinno)
int __stdcall PETS_GPIOGETSETUP(DWORD addr, int pinno)

Syntax:

Form 1: int __stdcall PETS_GPIOGETSETUP(int pinno)

Form 2: int __stdcall PETS_GPIOGETSETUP1(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:

Borland C++

DWORD addr = 0;

int pinno = 0;

Form 1: PETS_GPIOGETSETUP(pinno);

Form 2: PETS_GPIOGETSETUP1(addr, pinno);

int __stdcall PETS_GPIOGETSETUP(int pinno)
int __stdcall PETS_GPIOGETSETUP(DWORD addr, int pinno)

Syntax:

Form 1: int __stdcall PETS_GPIOGETSETUP(int pinno)

Form 2: int __stdcall PETS_GPIOGETSETUP1(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:

Borland C++

DWORD addr = 0;

int pinno = 0;

Form 1: PETS_GPIOGETSETUP(pinno);

Form 2: PETS_GPIOGETSETUP1(addr, pinno);

int __stdcall PWRCTL_AUTO()
int __stdcall PWRCTL_AUTOA(DWORD addr)

Syntax:

Form 1: int __stdcall PWRCTL_AUTO()

Form 2: int __stdcall 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: PWRCTL_AUTO();

Form 2: PWRCTL_AUTOA(addr);

int __stdcall PWRCTL_MANUAL()
int __stdcall PWRCTL_MANUALA(DWORD addr)

Syntax:

Form 1: int __stdcall PWRCTL_MANUAL()

Form 2: int __stdcall 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: PWRCTL_MANUAL();

Form 2: PWRCTL_MANUALA(addr);

3. Operation Flow

Initialize Flow

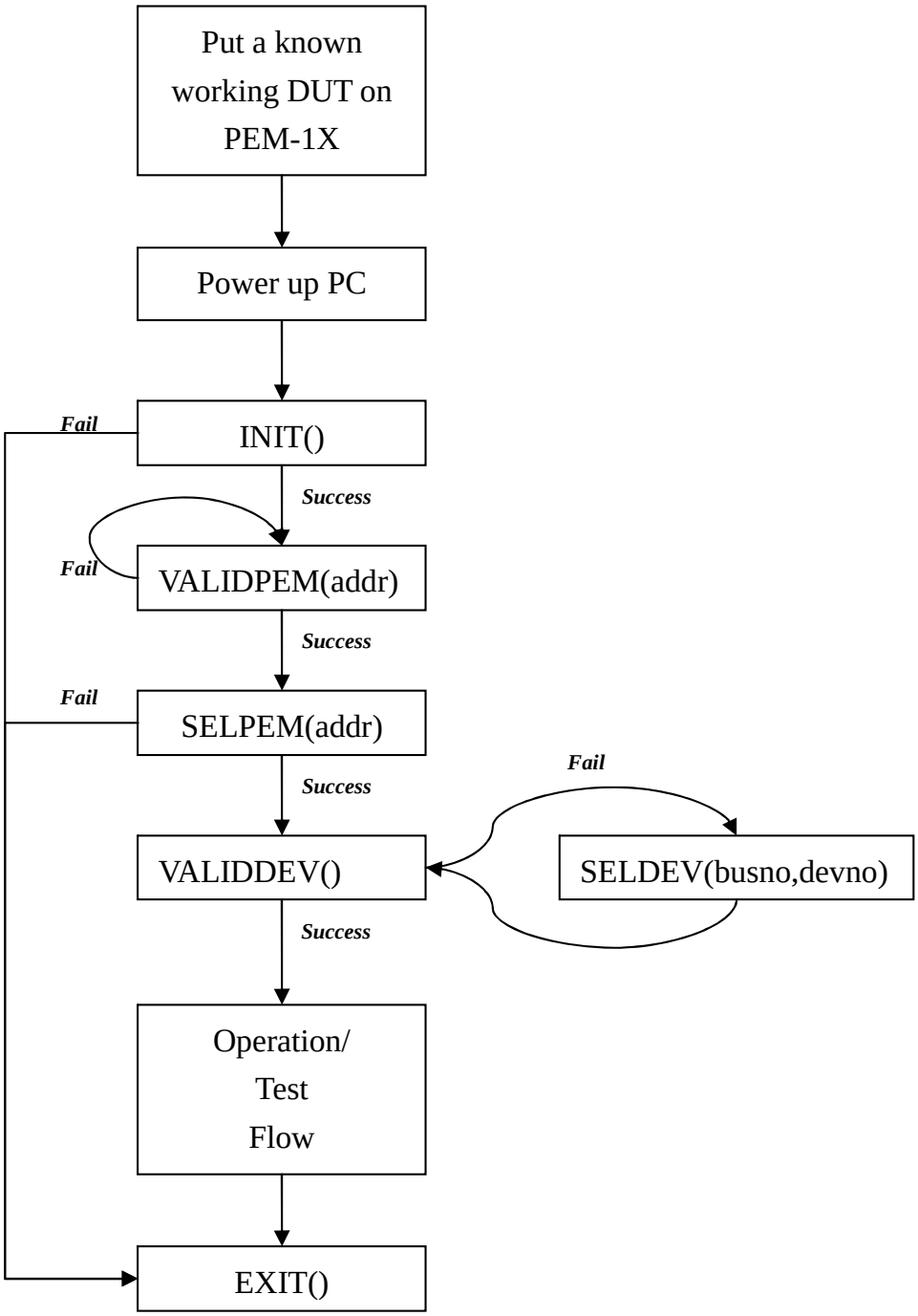

Fig. 1: Initialize Flow

Operation/Test Flow

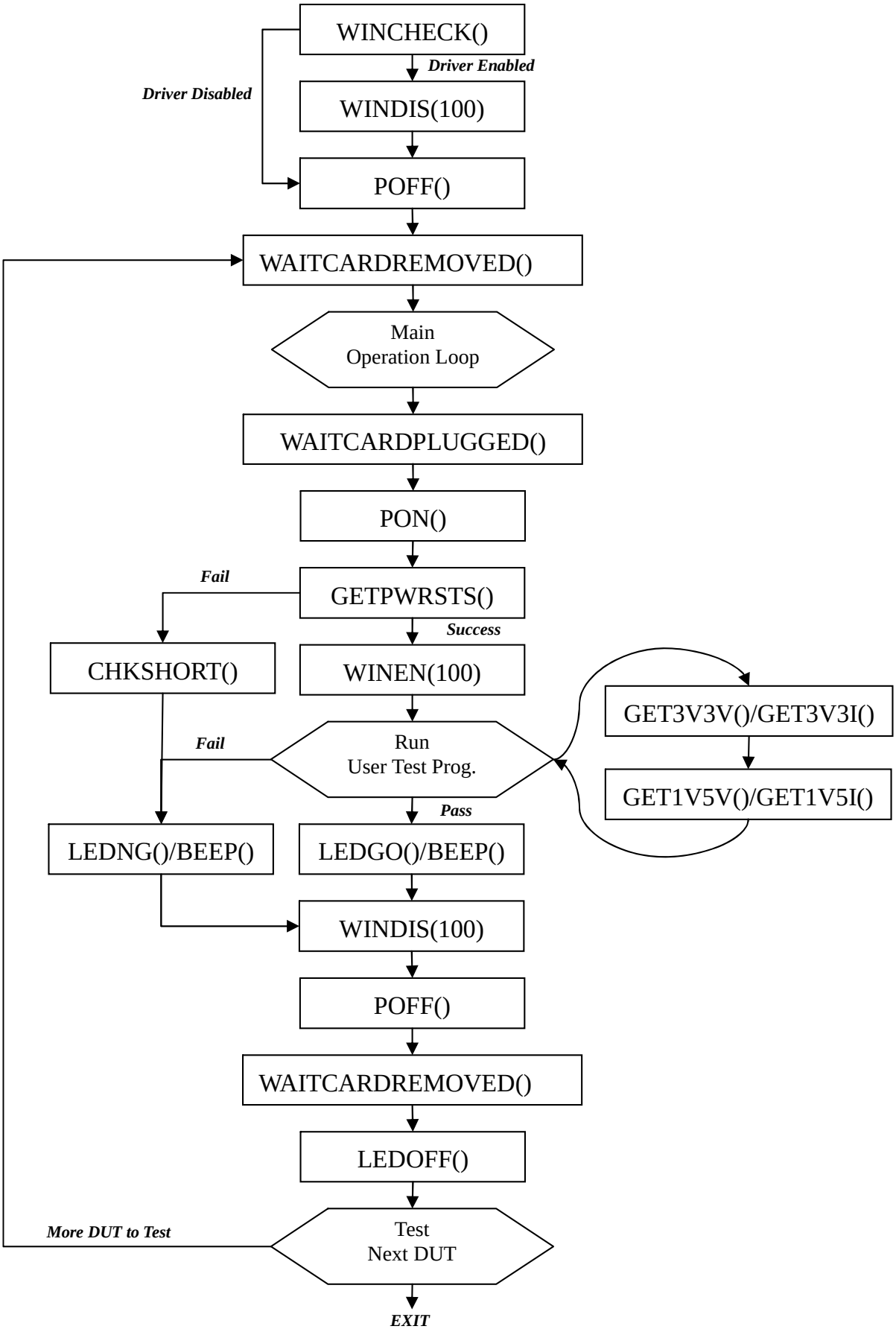

Fig. 2: Operation/Test Flow

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 = PETS_GETPWRSTS();
	if (status == 1)
	{
		printf("Power On!\n");
	}
	else
	{
		printf("Power Off!\n");
	}
	status = PETS_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 = PETS_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 = PETS_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 = PETS_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 = PETS_INIT();
	for (int i = 0; i < 4; i++)
	{
		pemvalid[i] = PETS_VALIDPEM(i);
		if (pemvalid[i] == 1) currpemno = i;
		printf("PEM[%d] %s\n", i, (pemvalid[i]==1) ? "Exist":"Not Exist");
	}
	status = PETS_SELPEM(currpemno);
	if (status != 0) printf("SELPEM Error!\n");
	showpwr(&pem);
	PETS_SELECTDEV(3, 0);
	status = PETS_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 = PETS_WINDIS(200);
	if (status == 0) printf("Driver Disabled!\n");
	else printf("Driver Disabled Fail!\n");

	status = PETS_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();

	PETS_PON();
	if (status == 0)
	{
		printf("Power On!\n");
		status = PETS_WINEN(200);
		if (status == 0) printf("Driver Enabled!\n");
		else printf("Driver Enabled Fail!\n");
	}
	else
	{
		printf("Power On Fail!\n");
		status = PETS_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:
	PETS_EXIT();
	return 0;
}
```

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