Soliton PCI Express Mini Extender

PETs Lib Programming Description

Rev 1.0
October 5, 2013

Rev. No.	Description	Date	Approved
0.1	Initial	Aug/24/2013	Kikimum

1. Installation

*** make sure these modules are exist.**

a. i2c_dev

b. i2c_i801

*** use i2c-tools to find out i2c device port (untar i2c-tools-3.1.0.tar.bz2 and install it)**

#i2cdetect –l (you should find your SMBus adapter)

i2c-0	**i2c**	**intel drm CRTDDC_A**	**I2C adapter**
i2c-3	**smbus**	**SMBus I801 adapter at 0500**	**SMBus adapter**

*** pets_config.sh exec it before start your test program**

- will store some information of the DUT card to file named config_[modaddr].ext.

 # ./pets_config.sh [mod_addr] [vendor_id]

2. Function Description

int INIT()

Syntax:

Form : int INIT()

Description:

Initialize all the PETs cards on mainboard.

Input Value: None

Return Value:

Return 0 if successful; 1 if error.

int INIT2(unsigned long addr)

Syntax:

Form : int INIT(unsigned long addr**)**

Description:

INIT2 will initialize the PETs module or modules with minimum resources.

Input Value:

unsigned long addr: Module addr, range 0~3. if specified, INIT2 will initialize the dedicate PETs module.

Return Value:

Return 0 if successful; 1 if error.

int EXIT()

Syntax:

int 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.

void GETLIBVER(int* major, int* minor)

Syntax:

void GETLIBVER(int* major, int* minor)

Description:

Get the version number of PETs Lib

Input Value: None

Return Value: None

void GETFIRMVER(int* major, int* minor)

Syntax:

void GETFIRMVER(int* major, int* minor)

Description:

Get the version number of PETs Firmware

Input Value: None

Return Value: None

int SELPEM2(unsigned long addr)

Syntax:

int SELPEM(unsigned long **addr)**

Description:

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

Input Value: unsigned long addr: module address, range 0~3

Return Value:

Return 0 if successful; 1 if error.

int VALIDPEM2(unsigned long addr)

Syntax:

int VALIDPEM(unsigned long **addr)**

Description:

Check if the PETs module exist.

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

Return Value:

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

int VALIDDEV()
int VALIDDEV2(unsigned long addr)

Syntax:

Form 1: int VALIDDEV()

Form 2: int VALIDDEV2(unsigned long **addr)**

Description:

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

Input Value:

Form 1: None

Form 2: unsigned long **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.

int PON()
int PON2(unsigned long addr)

Syntax:

Form 1: int PON ()

Form 2: int PON(unsigned long **addr)**

Description:

Turn the specified PETs module power on.

Input Value:

Form 1: None

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

Return Value:

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

int POFF()
int POFF2(unsigned long addr)

Syntax:

Form 1: int POFF()

Form 2: int POFF(unsigned long **addr)**

Description:

Turn the specified PETs module power off.

Input Value:

Form 1: None

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

Return Value:

Return 0 if successful; 1 if error.

int GETPWRSTS()
int GETPWRSTS2(unsigned long addr)

Syntax:

Form 1: int GETPWRSTS()

Form 2: int GETPWRSTS(unsigned long **addr)**

Description:

Get the power status of the specified PETs module.

Input Value:

Form 1: None

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

Return Value:

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

int CHKSHORT()
int CHKSHORT2(unsigned long addr)

Syntax:

Form 1: int CHKSHORT()

Form 2: int CHKSHORT(unsigned long **addr)**

Description:

Check which power rail is shorted.

Input Value:

Form 1: None

Form 2: unsigned long **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.

int GET3V3V(double* volatge)
int GET3V3V2(unsigned long addr, double* volatge)

Syntax:

Form 1: int GET3V3V(double* volatge)

Form 2: int GET3V3V(unsigned long **addr, double* volatge)**

Description:

Get 3.3V rail voltage reading.

Input Value:

Form 1: (double* volatge)

Form 2: (unsigned long **addr, double* voltage)**

unsigned long **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.

int GET1V5V(double* volatge)
int GET1V5V2(unsigned long addr, double* volatge)

Syntax:

Form 1: int GET1V5V(double* volatge)

Form 2: int GET1V5V(unsigned long **addr, double* volatge)**

Description:

Get 1.5V rail voltage reading.

For PEM-1X & PEC-1X

Input Value:

Form 1: (double* volatge)

Form 2: (unsigned long **addr, double* voltage)**

unsigned long **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.

int GET12V(double* volatge)
int GET12V2(unsigned long addr, double* volatge)

Syntax:

Form 1: int GET12V(double* volatge)

Form 2: int GET12V(unsigned long **addr, double* volatge)**

Description:

Get 12V rail voltage reading.

For PEX-1X & PEX-16X

Input Value:

Form 1: (double* volatge)

Form 2: (unsigned long **addr, double* voltage)**

unsigned long **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.

int GET3V3I(double* current)
int GET3V3I2(unsigned long addr, double* current)

Syntax:

Form 1: int GET3V3I(double* current)

Form 2: int GET3V3I(unsigned long **addr, double* current)**

Description:

Get 3.3V rail current reading.

Input Value:

Form 1: (double* current)

Form 2: (unsigned long **addr, double* current)**

unsigned long **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.

int GET1V5I(double* current)
int GET1V5I2(unsigned long addr, double* current)

Form 1: int GET1V5I(double* current)

Form 2: int GET1V5I(unsigned long addr, double* current)

Description:

Get 1.5V rail current reading.

For PEM-1X & PEC-1X

Input Value:

Form 1: (double* current)

Form 2: (unsigned long **addr, double* current)**

unsigned long **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.

int GET12I(double* current)
int GET12I2(unsigned long addr, double* current)

Syntax:

Form 1: int GET12I(double* current)

Form 2: int GET12I(unsigned long **addr, double* current)**

Description:

Get 12V rail current reading.

For PEX-1X & PEX-16X

Input Value:

Form 1: (double* current)

Form 2: (unsigned long **addr, double* current)**

unsigned long **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.

int CHECKCD()
int CHECKCD2(unsigned long addr)

Syntax:

Form 1: int CHECKCD()

Form 2: int CHECKCD(unsigned long **addr)**

Description:

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

Input Value:

Form 1: None

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

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

int PWRCTL_AUTO()
int PWRCTL_AUTO2(unsigned long addr)

Syntax:

Form 1: int PWRCTL_AUTO()

Form 2: int PWRCTL_AUTO2(unsigned long **addr)**

Description:

Set Power control to Auto mode . .

Input Value:

Form 1: None.

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

Return 0 if successful; 1 if error.

int PWRCTL_MANUAL()
int PWRCTL_MANUAL2(unsigned long addr)

Syntax:

Form 1: int PWRCTL_MANUAL()

Form 2: int PWRCTL_MANUAL2(unsigned long **addr)**

Description:

Set Power control to Manual mode .

Input Value:

Form 1: None.

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

Return 0 if successful; 1 if error.

int LEDGO()
int LEDGO2(unsigned long addr)

Syntax:

Form 1: int LEDGO()

Form 2: int LEDGO2(unsigned long **addr)**

Description:

Turn on the GO LED to indicate the test status.

Input Value:

Form 1: None

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

Return 0 if successful; 1 if error.

int LEDNG()
int LEDNG2(unsigned long addr)

Syntax:

Form 1: int LEDNG()

Form 2: int LEDNG2(unsigned long **addr)**

Description:

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

Input Value:

Form 1: None

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

Return 0 if successful; 1 if error.

int LEDOFF()
int LEDOFF2(unsigned long addr)

Syntax:

Form 1: int LEDOFF()

Form 2: int LEDOFF2(unsigned long **addr)**

Description:

Turn off both the GO and NG LEDs.

Input Value:

Form 1: None

Form 2: (unsigned long **addr)**

unsigned long **addr: module address, range 0~3**

Return Value:

Return 0 if successful; 1 if error.

int BEEP(int freq, int time)
int BEEP2(unsigned long addr, int freq, int time)

Syntax:

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

Form 2: int BEEP2(unsigned long **addr, int freq, int time)**

Description:

Play Beeper.

Input Value:

Form 1: (int freq, int time)

Form 2: (unsigned long **addr, int freq, int time)**

unsigned long **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.

3. Operation Flow

Initialize Flow

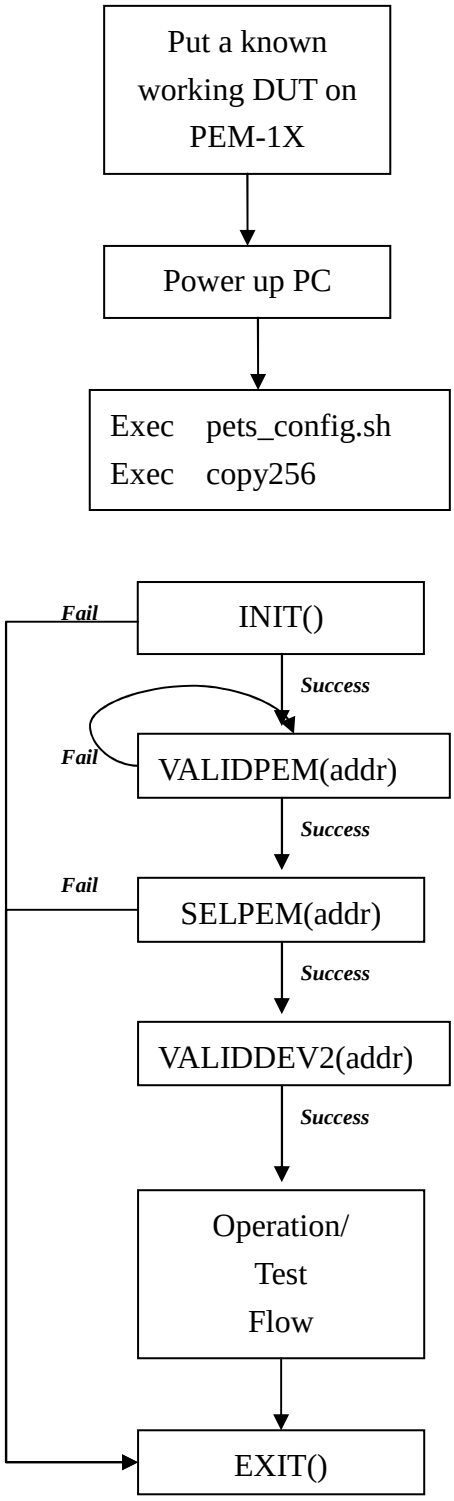

Fig. 1: Initialize Flow

Operation/Test Flow

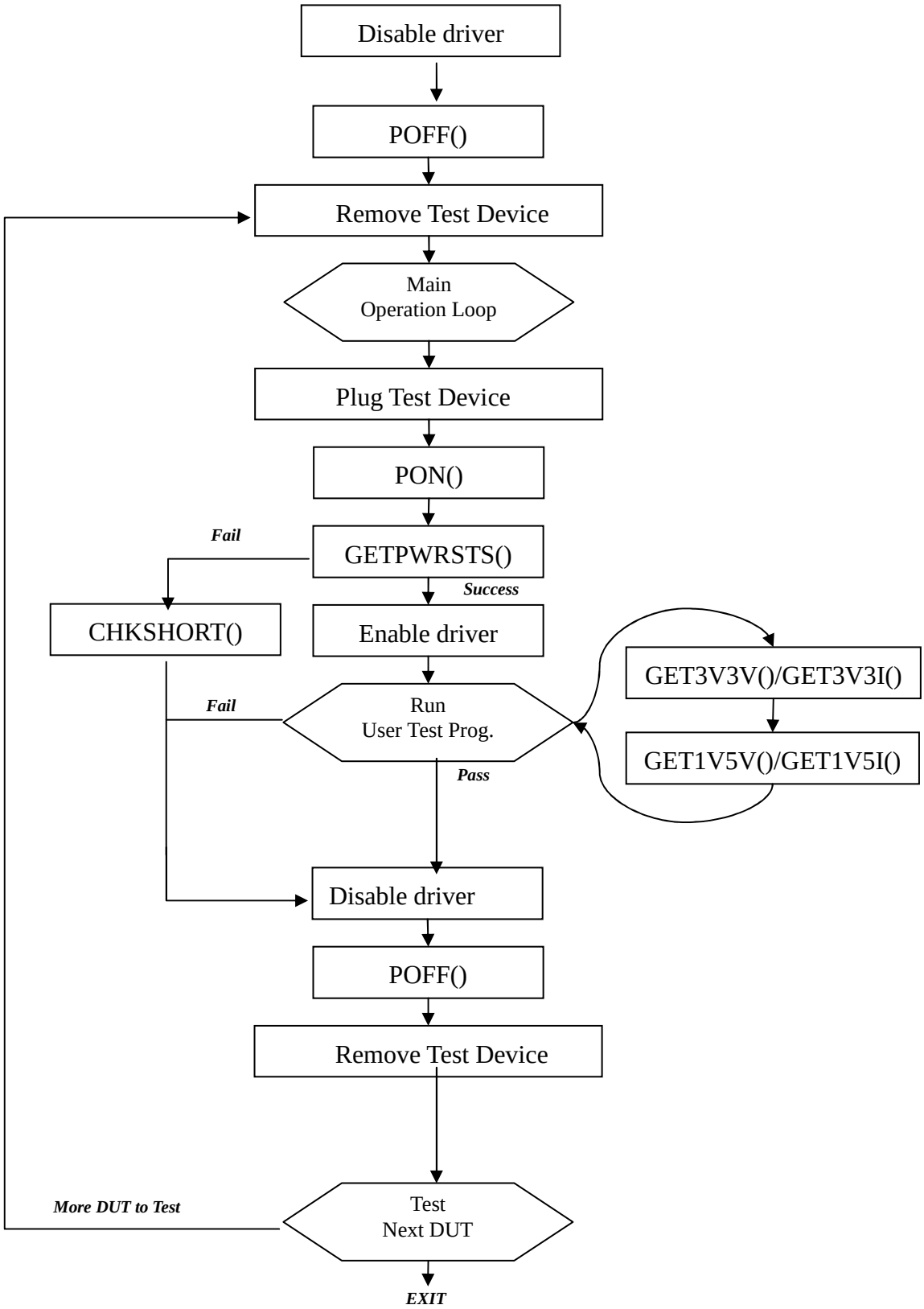

Fig. 2: Operation/Test Flow

4. Sample Program

```
#include <stdio.h>
#include "../include/PETs.h"
int main(int argc, char* argv[])
{
	unsigned int modaddr=0;
	int res;
	double dval = 0;
	int major, minor;
	int cds, status;
	int valid,devvalid;
	unsigned long chkshort;
	if (argc < 2)
	{
		printf("Error!    Please specify [Module Address]!\n");
		printf("Example : single-test 0\n");
		return 1;
	}
	modaddr = atoi(argv[1]);
/*********************************
Single card test
*********************************/
	status = INIT2(modaddr);
	valid = VALIDPEM(modaddr);
	if ( valid == 0)
```

```
	{
		printf("pem[%d] is not exist\n", modaddr);
//		return -1;
	}
	else
	{
		devvalid = VALIDDEV2(modaddr);
		if (devvalid == 0)
		{
			printf("Dev    is not match, try rescan\n");
//			return -1;
		}
		GETLIBVER(&major, &minor);
		printf("lib version = %d.%d, cds=%d\n", major, minor, cds);
		cds= CHECKCD2(modaddr);
		printf("lib version = %d.%d, cds=%d\n", major, minor, cds);
		res = GET3V3I2(modaddr,&dval);
		if (res == 0)
		{
			printf("3.3V Current = %f\n", dval);
		}
		else if (res == PEMCTRL_ERROR_ADCNOTEXIST)
		{
			printf("ADC Not Exist!\n");
		}
		else
```

```
{
    printf("Get 3V3 Current    failed! status = %d\n", res);
}
res = GET3V3V2(modaddr,&dval);
if (res == 0)
{
    printf("3.3V Voltage = %f\n", dval);
}
else if (res == PEMCTRL_ERROR_ADCNOTEXIST)
{
    printf("ADC Not Exist!\n");
}
else
{
    printf("Get 3V3 Voltage    failed! status = %d\n", res);
}
res = GET12VV2(modaddr,&dval);
if (res == 0)
{
    printf("12V Voltage = %f\n", dval);
}
else if (res == PEMCTRL_ERROR_ADCNOTEXIST)
{
    printf("ADC Not Exist!\n");
}
else
```

```
{
    printf("Get 12V Voltage    failed! status = %d\n", res);
}
res = GET12VI2(modaddr,&dval);
if (res == 0)
{
    printf("12V Current = %f\n", dval);
}
else if (res == PEMCTRL_ERROR_ADCNOTEXIST)
{
    printf("ADC Not Exist!\n");
}
else
{
    printf("Get 12V Current failed! status = %d\n", res);
}
// swap card test
printf("Swap card Test....\n");
POFF2(modaddr);
printf("Unplug test card & Plug another test card....\n");
usleep(500000);
getchar();
status = PON2(modaddr);
if (status == 0)
{
    printf("Power on!\n");
```

```
            }
            else
            {
                printf("Power on    failed!\n");
                chkshort = CHKSHORT(modaddr);
                if (chkshort & PEMCTRL_ERROR_3V3AUXSHORT) printf("3.3V Aux Short!\n");
                if (chkshort & PEMCTRL_ERROR_3V3SHORT) printf("3.3V Short!\n");
                if (chkshort & PEMCTRL_ERROR_1V5SHORT) printf("1.5V Short!\n");
            }
        }
        EXIT(modaddr);
        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