Soliton SCTS Extender

SCTS DLL Programming Description

Rev 1.0
May 20, 2008

Rev. No.	Description	Date	Approved
0.1	Initial	May/20/2008	Vincent
1.0	First Relase	Jun/13/2008	Vincent

1. Installation

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

2. Function Description

sctsctl_t __stdcall SCTSCTL_OPEN(DWORD instance)

Syntax:

Sctsctl_t __stdcall SCTSCTL_OPEN(DWORD instance)

Description:

Get a handle to a device(SCTS). When the function returns, sctsctl_t will Contain completed information for th SCTS board.

- ***instance* is SCTS board ID or board Index to determine whitch SCTS board we want to control. The valid range is 0 to 7**

Return Value:

If successful, will return the SCTS handle of data structure sctsctl_t.

Return 0 if fail.

Usage:

```
sctsctl_t board0;
int boarded = 0;
board0 = sctsctl_open(boarded);
```

Notes:

This function should be used to open a channel to a SCTS device and get a handle to the device driver before any PCIAPI functions that require a handle can be used.

void __stdcall SCTSCTL_CLOSE(sctsctl_t board)

Syntax:

void __ stdcall SCTSCTL_CLOSE(sctsctl_t board)

Description:

Close the handle of SCTS board

Return Value:

No return value.

Usage:

SCTSCTL_CLOSE(board0);

Notes:

Be sure to call SCTSCTL_CLOSE before your application exits.

int __stdcall SCTSCTL_CONNECT(sctsctl_t board, void (*sig_handler)(sctsctl_t, int))

Syntax:

int __stdcall SCTSCTL_CONNECT(sctsctl_t board, void (*sig_handler)(sctsctl_t, int))

Description:

To connect user's interrupt process to main program and pass its handle to main program.

***sig_handler: UserISR.**

*** int : INT vector, will be one of the PCITG_IRQ_EXT, PCITG_IRQ_0, PCITG_IRQ_1, PCITG_IRQ_2, PCITG_IRQ_3,**

*** board is a handle of SCTS board include the information of SCTS.**

Return Value:

Return 0 if successful; 1 if error.

Usage:

```
 // Define a data structure
typedef struct
{
        HANDLE userevent;
        ULONG nSampleCount;
        int intvec;
        int event;
 } MY_EXTENSION, *PMY_EXTENSION;
sctsctl_t board0;
board0 = SCTSCTL_OPEN(0);
SCTSCTL_CARDSERVICE_INIT(board0);
HANDLE UserISREvent;
DWORD dwWaitResult;
PMY_EXTENSION pExtension = NULL;
UserISREvent = CreateEvent(NULL, TRUE, FALSE, NULL);

pExtension = (PMY_EXTENSION)malloc(sizeof(MY_EXTENSION));
pExtension->nSampleCount = 0;
pExtension->userevent = UserISREvent;
```

```
SCTSCTL_SET_EXTENSION(board0, pExtension);
SCTSCTL_CONNECT(board0, UserISR);
```

int __stdcall SCTSCTL_SET_EXTENSION(sctsctl_t board, PVOID pext)

Syntax:

int __stdcall SCTSCTL_SET_EXTENSION(sctsctl_t board, PVOID pext)

Description:

Associates a user defined data structure with a particular device.

- ***board*** is a handle of SCTS board include the information of SCTS.
- ***pext*** is the structure pointer of extension.

Return Value:

Return 0 if successful.

Usage:

```
 // Define a data structure
typedef struct
{
      HANDLE userevent;
      ULONG nSampleCount;
      int intvec;
      int event;
 } MY_EXTENSION, *PMY_EXTENSION;
sctsctl_t board0;
board0 = SCTSCTL_OPEN(0);
SCTSCTL_CARDSERVICE_INIT (board0);
HANDLE UserISREvent;
DWORD dwWaitResult;
PMY_EXTENSION pExtension = NULL;
UserISREvent = CreateEvent(NULL, TRUE, FALSE, NULL);
pExtension = (PMY_EXTENSION)malloc(sizeof(MY_EXTENSION));
pExtension->nSampleCount = 0;
pExtension->userevent = UserISREvent;
SCTSCTL_SET_EXTENSION(board0, pExtension);
```

Notes:

The SCTSCTL_SET_EXTENSION and SCTSCTL_GET_EXTENSION routines support a user defined device extension to hold variables and/or buffers associated with a particular device.

PVOID __stdcall SCTSCTL_GET_EXTENSION(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_GET_EXTENSION(sctsctl_t board, PVOID pext)

Description:

Return a pointer to the user defined data structure.

- ***board*** is a handle of SCTS board include the information of SCTS.

Return Value:

A point by user defined.

Usage:

```
// User interrupt service routine
void UserISR(sctsctl_t board, int intvec)
{
    //unsigned long datac;
    PMY_EXTENSION pExtension =
        (PMY_EXTENSION)SCTSCTL_GET_EXTENSION(board);
        pExtension->nSampleCount++;
        pExtension->intvec = intvec;
    printf("ISR INT Count=[%d] Vector=[0x%.2X] Diff[0x%.2X]
        val[0x%.2X]\n",pExtension->nSampleCount, pExtension->intvec,
        board->data_diff,       board->data_in);
    SetEvent(pExtension->userevent);
}
```

int __stdcall SCTSCTL_ADCCONV(sctsctl_t board, int ch, double *value)

Syntax:

int __stdcall SCTSCTL_ADCCONVL(sctsctl_t board, int ch, double *value)

Description:

Meauser current or voltage.

board is a handle of SCTS board include the information of SCTS.

ch

0: Voltage meas, gain 0.5

1: Current meas, gain 5.

Return Value:

return 0 if success, 1 if Error (ADC NULL bit not detected)

Usage:

```
status = SCTSCTL_ADCCONVD(board, ch, &val);
```

int __stdcall SCTSCTL_CARDSERVICE_INIT(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_CARDSERVICE_INIT(sctsctl_t board)

Description:

Initial auto test flow.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_CARDSERVICE_INIT(board0);

int __stdcall SCTSCTL_CARDINSERTPROCESS(sctsctl_t board, double *volt, double *current)

Syntax:

int __stdcall SCTSCTL_CARDINSERTPROCESS(sctsctl_t board, double *volt, double *current)

Description:

Card insert process.

board is a handle of SCTS board include the information of SCTS.

double *volt : if success, the measured voltage will stored in the passed double pointer.

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

Return Value:

Return 0 if successful; 1 if error.

Usage:

status = SCTSCTL_CARDINSERTPROCESS(board0, &volt, &current);

int __stdcall SCTSCTL_CARDREMOVEPROCESS(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_CARDREMOVEPROCESS(sctsctl_t board)

Description:

When device removed, switch to disconnected mode.

***board* is a handle of SCTS board include the information of SCTS**.

Return Value:

Return 0 if successful; 1 if error.

Usage:

status = SCTSCTL_CARDREMOVEPROCESS(board0);

int __stdcall SCTSCTL_ CARDOCPROCESS (sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_CARDOCPROCESS(sctsctl_t board)

Description:

Over current process.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

status = SCTSCTL_CARDREMOVEPROCESS(board0);

int __stdcall SCTSCTL_CARDSERVICE_EXIT(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_CARDSERVICE_EXIT(sctsctl_t board)

Description:

Card remove process.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_CARDSERVICE_EXIT(board0);

int __stdcall SCTSCTL_BEEP(sctsctl_t board, int freq, int msec)

Syntax:

int __stdcall SCTSCTL_BEEP(sctsctl_t board, int freq, int msec)

Description:

Play Beeper.

board is a handle of SCTS board include the information of SCTS.

int freq: frequency of sound, range 0~3

int msec: length Millisecond

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_BEEP(board0, 0, 1000);

int __stdcall SCTSCTL_LEDGO(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_LEDGO(sctsctl_t board)

Description:

Turn on the GO LED to indicate the test status.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_LEDGO(board0);

int __stdcall SCTSCTL_LEDNG(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_LEDNG(sctsctl_t board)

Description:

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

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_LEDNG(board0);

int __stdcall SCTSCTL_LEDOFF(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_LEDOFF(sctsctl_t board)

Description:

Turn off both the GO and NG LEDs

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_LEDOFF(board0);

int __stdcall SCTSCTL_MCARD_FORCEPON(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_MCARD_FORCEPON(sctsctl_t board)

Description:

Manual mode force power on.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_MCARD_FORCEPON(board0);

int __stdcall SCTSCTL_MCARD_DISCONNECT(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_ MCARD_DISCONNECT (sctsctl_t board)

Description:

Manual mode disconnect socket.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_MCARD_DISCONNECT(board0);

int __stdcall SCTSCTL_MCARD_NORMAL(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_MCARD_NORMAL(sctsctl_t board)

Description:

Manual mode force normal mode.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_MCARD_NORMAL(board0);

int __stdcall SCTSCTL_MCARD_CHECKSDCD(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_MCARD_CHECKSDCD(sctsctl_t board)

Description:

Check SD card detect pin, check whether SD card is inserted.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_MCARD_CHECKSDCD(board0);

int __stdcall SCTSCTL_MCARD_CHECKOC(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_MCARD_CHECKOC(sctsctl_t board)

Description:

Check over current flag.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_MCARD_CHECKOC(board0);

int __stdcall SCTSCTL_MEAS_VOLTAGE(sctsctl_t board, double *voltage)

Syntax:

int __stdcall SCTSCTL_MEAS_VOLTAGE(sctsctl_t board, double *voltage)

Description:

Measure voltage.

board is a handle of SCTS board include the information of SCTS.

double *volt : if success, the measured voltage will stored in the passed double pointer.

Return Value:

Return 0 if successful; 1 if error.

Usage:

status = SCTSCTL_CARDINSERTPROCESS(board0, &volt, &current);

int __stdcall SCTSCTL_ MEAS_ CURRENT(sctsctl_t board, double *current)

Syntax:

int __stdcall SCTSCTL_MEAS_CURRENT(sctsctl_t board, double *current)

Description:

Measure current.

board is a handle of SCTS board include the information of SCTS.

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

Return Value:

Return 0 if successful; 1 if error.

Usage:

status = SCTSCTL_MEAS_CURRENT(board0, &current);

int __stdcall SCTSCTL_ PON(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_ PON(sctsctl_t board)

Description:

Manual mode force power on.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_PON(board0);

int __stdcall SCTSCTL_ POFF(sctsctl_t board)

Syntax:

int __stdcall SCTSCTL_ POFF(sctsctl_t board)

Description:

Manual mode force power off.

board is a handle of SCTS board include the information of SCTS.

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_POFF(board0);

int __stdcall SCTSCTL_GPIOSET(sctsctl_t board, int pinno)

Syntax:

int __stdcall SCTSCTL_ GPIOSET(sctsctl_t board, int pinno)

Description:

Set GPIO output to high.

board is a handle of SCTS board include the information of SCTS.

pinno: GPIO pin no, range 0~1

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_GPIOSET(board0, 0);

int __stdcall SCTSCTL_GPIORESET(sctsctl_t board, int pinno)

Syntax:

int __stdcall SCTSCTL_ GPIORESET(sctsctl_t board, int pinno)

Description:

Set GPIO output to low.

board is a handle of SCTS board include the information of SCTS.

pinno: GPIO pin no, range 0~1

Return Value:

Return 0 if successful; 1 if error.

Usage:

SCTSCTL_GPIORESET(board0, 0);

3. Operation Flow

Initialize Flow

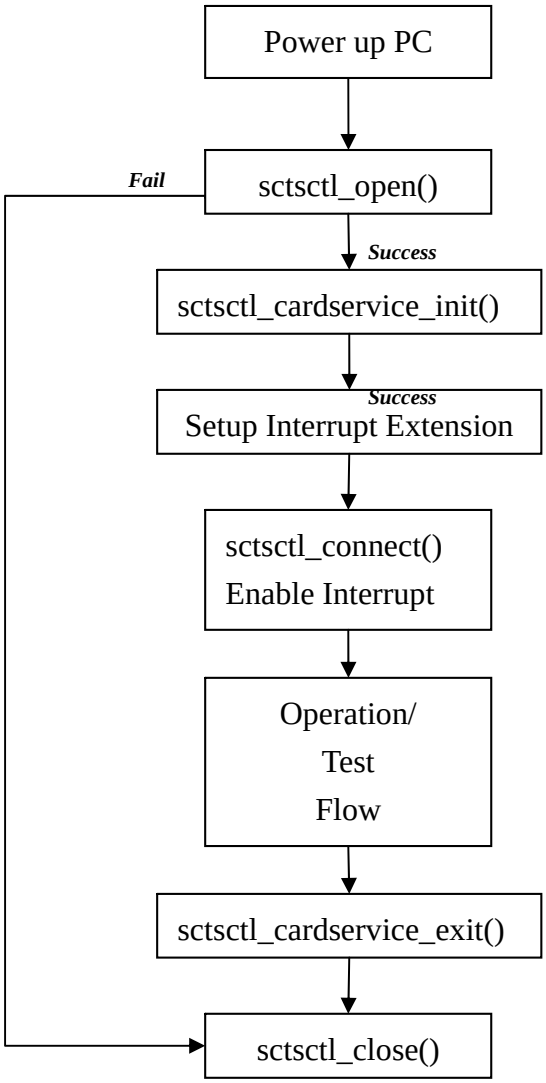

Fig. 1: Initialize Flow

Operation/Test Flow

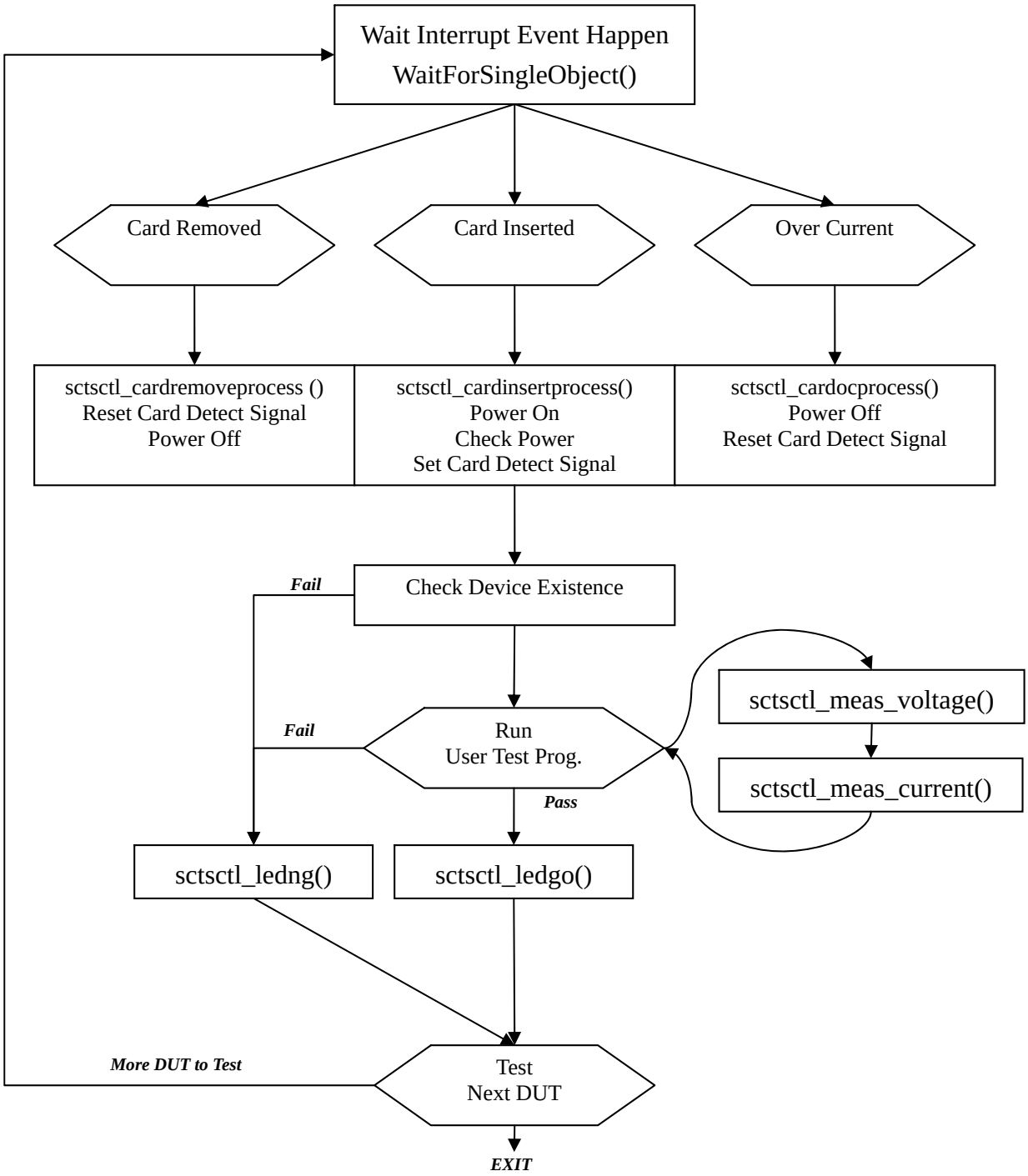

Fig. 2: Operation/Test Flow

4. Sample Program

VC++ Sample

```
#include "stdafx.h"
#include "scts_lib.h"

typedef struct
{
	HANDLE userevent;
	ULONG nSampleCount;
	int intvec;
	int event;
} MY_EXTENSION, *PMY_EXTENSION;


void UserISR(sctsctl_t board, int intvec)
{
	PMY_EXTENSION pExtension = (PMY_EXTENSION)sctsctl_get_extension(board);
	pExtension->nSampleCount++;
	pExtension->intvec = intvec;
	Sleep(10);	// De-Glitch for CD#, more sleep time if needed
	printf("ISR INT Count=[%d] Vector=[0x%.2X] Diff[0x%.2X]
Val[0x%.2X]\n",pExtension->nSampleCount, pExtension->intvec, board->data_diff,
board->data_in);
	SetEvent(pExtension->userevent);
}
int main(int argc, char* argv[])
{
	int board_index = 0;
	sctsctl_t board0;
	unsigned long data;
	unsigned long address, aH, aL;
	int execnt = 0;
	int sleeptime = 10;
	int status = 0;
	int x;
	int error = 0;
	int total_error = 0;
	int verbose = 0;
	double volt;
	double current;
	DWORD dwWaitResult;
	if (argc > 1)	board_index = atoi(argv[1]);  //
	if (argc > 2) verbose = 1;

	board0 = sctsctl_open(0);
	if (board0->hDevice == PCITG_NULL)
	{
```

```
        printf("ERROR opening SCTS[%d]: (%0x) returned from CreateFile\n", board_index,
GetLastError());
        return 1;
    }
    printf("SCTS[%d] found, Device Handle = 0x%.8X\n", board_index, board0->hDevice);

    sctsctl_cardservice_init(board0);

    // Set ISR Extention
    HANDLE UserISREvent;
    PMY_EXTENSION pExtension = NULL;

    UserISREvent = CreateEvent(NULL, TRUE, FALSE, NULL);
    pExtension = (PMY_EXTENSION)malloc(sizeof(MY_EXTENSION));
    pExtension->nSampleCount = 0;
    pExtension->userevent = UserISREvent;
    sctsctl_set_extension(board0, pExtension);
    sctsctl_connect(board0, UserISR);

    Sleep(100);
    printf("Wait for Interrupt!\n");
    while(!kbhit())
    {
        dwWaitResult = WaitForSingleObject(UserISREvent, 3000);
        if (dwWaitResult == WAIT_OBJECT_0)
        {
            switch (pExtension->intvec)
            {
            case SCTS_EVENT_SDMMCCARD_INSERT:
            case SCTS_EVENT_MSCARD_INSERT:
            case SCTS_EVENT_XDCARD_INSERT:
                status = sctsctl_cardinsertprocess(board0, &volt, &current);
                if (status == 0)
                {
                    printf("Card Insert Process OK! V=[%f]V I=[%f]mA\n", volt, current*1e3);
                }
                else
                {
                    printf("Card Insert Process Fail!\n");
                }
            break;
            case SCTS_EVENT_SDMMCCARD_REMOVE:
            case SCTS_EVENT_MSCARD_REMOVE:
            case SCTS_EVENT_XDCARD_REMOVE:
                status = sctsctl_cardremoveprocess(board0);
                if (status == 0) printf("Card Removed OK\n");
                else printf("Card Removed Fail\n");

            break;
            case SCTS_EVENT_CARDPOWER_OC:
```

```
                    status = sctsctl_cardocprocess(board0);
                break;
                default:
                    // Do nothing
                break;

                }
                ResetEvent(UserISREvent);
            }
        }
        sctsctl_cardservice_exit(board0);
        sctsctl_close(board0);
        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