PuslelR™ Multitouch Screen Software SDK Specification

Revision 4.0

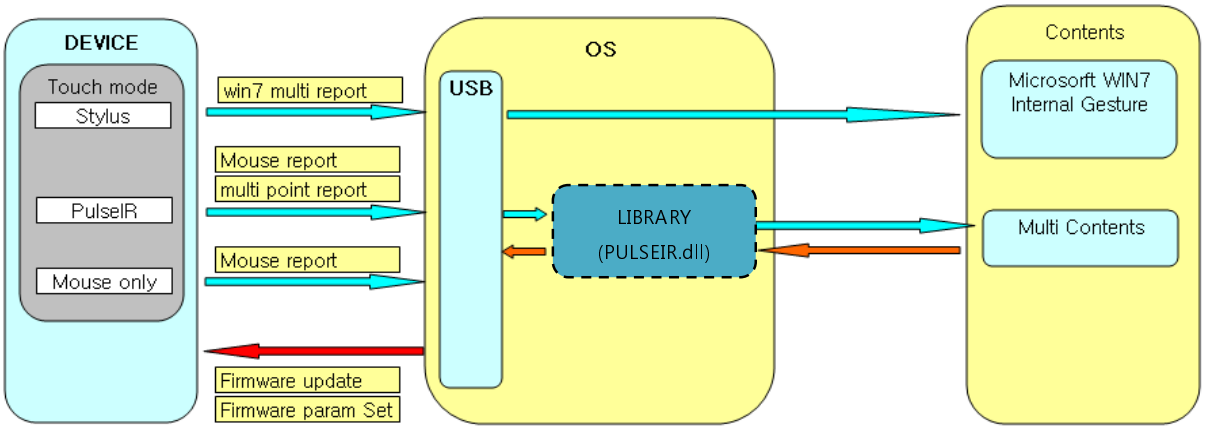
PulseIR™ Touch Screen

Table of Contents

1. Overview

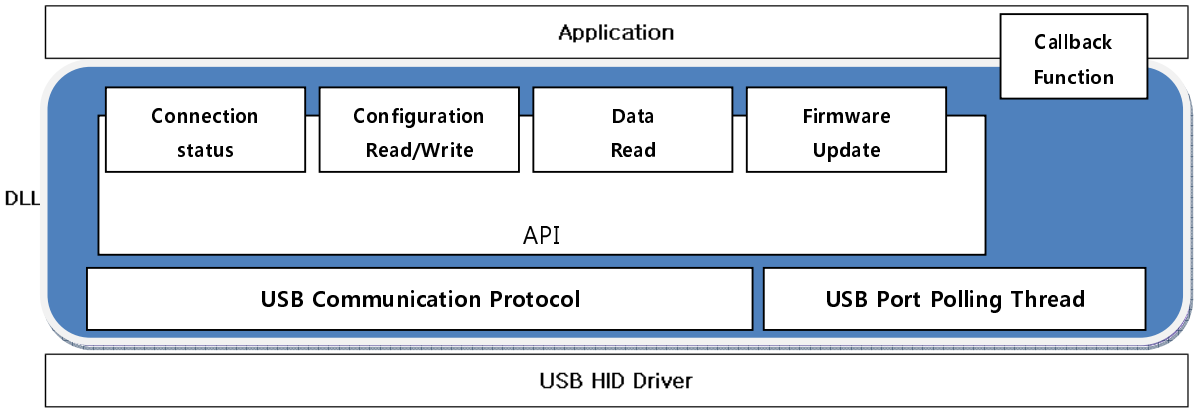
This document describes API specification regarding the communication protocol between PC Application and PulseIR touch device by means of USB.

1.1 Diagram



1.2 PulseIR API Layers Hierarchy

A touch device is connected to a PC by means of USB HID(Human Interface Device). The PC Application communicates with the touch device through API DLL(PULSEIR.DLL) in LIBRARY. The API hierarchical layers structure is shown below.



Setting up a Touch Device
Set the Touch Mode to “PulseIR” for enabling data communication with API.
You can set “Touch Mode” through PulseIR utility.

- **USB Communication Protocol**
  A protocol of request/response command type to communicate with the PulseIR touch device.

- **USB Port Polling Thread**
  This is for checking out if the touch device is properly connected by checking the connection to USB every one second. API DLL periodically must check extra threads to check the connection of the touch device to USB as it can’t receive Windows system events.

1.3 DLL File

OS	File	Description
Window	PULSEIR.DLL	PULSEIR DLL file
	PULSEIRDLL.h	PULSEIRDLL header file
Mac OS X	libPulseIRAPIDriver.dylib	PULSEIR DLL file
	PIRParam.h	PULSEIRDLL header file
Linux	libPulseIRCtrl.so	PULSEIR SO file
	PIRParam.h	PULSEIRDLL header file

2. Data Structure

Structure	Size	Description	Remarks
PULSEIR_POINT	24 byte	Point Data	

2.1 Point Data (PULSEIR_POINT)

● **Structure Prototype**

```
typedef struct {
        int id;
        int type;
        int x;
        int y;
        int w;
        int h;
} PULSEIR_POINT;
```

● **Parameter Description**

Parameter	Size	Description
id	4 byte	Unique Point ID
type	4 byte	0x01 = Up 0x02 = Down 0x03 = Move
x	4 byte	X coordinates Display resolution unit
y	4 byte	Y coordinates Display resolution unit
w	4 byte	X radius Display resolution unit
h	4 byte	Y radius Display resolution unit

3. Interface Function

Classification	Function Name	Description	Remarks
DLL information	getDLLVersion	get DLL version	
	setPulseIRCB	transfer the address of Callback function to library	static

3.1 getDLLVersion

Syntax	int getDLLVersion(char* Version)	
Description	get DLL version	
Parameter	Version	Pointer of a variable to get DLL version data
Return Value	0 = Success	
Example		
See Also		
Remark		

3.2 setPulseIRMultiCB

Syntax	void setPulseIRMultiCB(void* pFunc)	
Description	Transfer the address of CALLBACK function to LIBRARY	
Parameter	pFunc	Address of Application
Return Value		
Example		
See Also		
Remark		

3.3 Basic Structure of CALLBACK Function

```
void CPulseIR_DemoDlg::OnPulseIRMultiEvent(int usbindex, int count,PULSEIR_POINT* pnt)
 {
;        for(int i=0; i<count;i++)
         {
                  switch (pnt[i].type) {
                  case PIR_CURSOR_TOUCH_UP: //0x01
                           // up point
                           break;
                  case PIR_CURSOR_TOUCH_DOWN: //0x02
                           / /Down point
                           break;
                  case PIR_CURSOR_TOUCH_MOVE: //0x03
                           //move    point
                           break;
                  }

         }
 }
```

- **Parameter Description**

usbindex : index of a touch device
count: number of points to receive from the touch device
pnt: (point data) Array

pnt[i].id - unique point ID
For case of usbindex=0, id should be any number from 1 to 99 to support more than one touch device
For case of Usbindex=1, id is assigned any number from 101 to 199 to each touch device
pnt[i].type - status of point (up , down , move)
pnt[i].x - x coordinates
pnt[i].y - y coordinates
pnt[i].w - width
pnt[i].h - height

4. Sample Codes

4.1 C++ Sample code

```
/* dump.cpp   */
#include <stdio.h>
#include <windows.h>
#include "pulseirdll.h"

HMODULE hModule;
getDLLVersion_t         pGetDLLVersion;
setPulseIRMultiCB_t     pSetPulseIRMultiCB;

static void OnPulseIRMultiEvent(int usbindex, int count, PULSEIR_POINT *pnt);

bool LoadDLL()
{
        hModule = LoadLibrary("pulseirlib.dll");
        if(!hModule) return false;

        pSetPulseIRMultiCB = (setPulseIRMultiCB_t)GetProcAddress(hModule,
"setPulseIRMultiCB");
        pSetPulseIRMultiCB(OnPulseIRMultiEvent);
        return true;
}

void UnloadDLL()
{
        FreeLibrary(hModule);
        hModule = NULL;
}


void OnPulseIRMultiEvent(int usbindex, int count, PULSEIR_POINT *pnt)
{
        const char *strTYPE[] = {"", "UP", "DOWN", "MOVE" };

        printf("usbindex=%d ", usbindex);
        for(int i = 0; i < count; i++)
        {
                printf(" P%d=[%4d,%4d,%-4s]",   pnt[i].id, pnt[i].x, pnt[i].y, strTYPE[pnt[i].type]);
        }
        printf("\n");
}

int main()
{
        if(LoadDLL() == false)
        {
                printf("DLL load fail!\n");
                Sleep(3000);
                return 1;
        }
        char Version[64];
        pGetDLLVersion(Version);
        printf("DLL Version %s\n",Version);
```

```
        getchar();          // wait to enter key.
        UnloadDLL();
        return 0;
}
```