POWER USB

11/11/2010

Computer Controlled Power Strip

Software API

Power USB

SOFTWARE API

1. Introduction

The PowerUSB can be controlled from third party applications through functions calls provided in the DLL. The DLL provides functions to initialize the PowerUSB and set the outlet power states. The DLL can be loaded statically or dynamically from the calling application. The DLLs have been tested under Visual Studio .Net and VC++ 6.0 in Windows XP and Windows 7 environments.

2. Function Calls

2.1 Initialization

Initializes the Power USB API. No other functions can be called till the API is initialized.

Name: InitPowerUSB
Parameters: mode. Returns the driver mode. The default mode is HID driver
Return: >=0 if successful. < 0 on failure
C++ Example:

```
if (!m_pwrUSBInit)
{

    int mode;


    if (InitPowerUSB(&mode) >= 0)
    {
        m_pwrUSBInit = 1;
        m_pwrUsbConnected = CheckStatusPowereUSB();
     }
}
```

2.2 Check PowerUSB connectivity

Checks to see if the power USB is connected to the computer. Returns the number of Power USBs connected. Windows function OnDeviceChange can be used to monitor the connection/disconnection of the powerUSB dynamically.

Name: CheckStatusPowereUSB
Parameters: None
Returns: Number of PowerUSB devices connected
C++ Example:

```
BOOL CPowerUSBDlg::OnDeviceChange(UINT nEventType, DWORD_PTR dwData)
{
```

```
        m_pwrUsbConnected = CheckStatusPowereUSB();
        UpdateData(TRUE);
        if (m_pwrUsbConnected)
                m_connectionStr = "PwrUSB Connected";
        else
                m_connectionStr = "PwrUSB Not Connected";
        UpdateData(FALSE);
        return TRUE;
}
```

2.3 Close the Device

Closes the PowerUSB API. Should be called in application exit function such as OnDestroy.

Name: ClosePowerUSB
Parameters: None
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
if (m_pwrUSBInit)
{
        ClosePowerUSB();
        m_pwrUSBInit = 0;
}
```

2.4 Set the Current PowerUSB (Useful when multiple powerUSBs are connected)

Changes the current PowerUSB. By default the first connected PowerUSB is active. If multiple PowerUSBs are connected, you can use this function to change the current PowerUSB. Once this function is called, the commands (SetPortPowerUSB) will be directed to the selected PowerUSB. Maximum of 4 powerUSBs are currently supported.

Name: SetCurrentPowerUSB
Parameters: int count (0 to 3)
Returns: Returns current selected PowerUSB.
C++ Example:

```
   if ((m_MaxDevices=InitPowerUSB(&m_pwrUSBMode)) > 0)
   {
      m_pwrUSBInit = 1;
      m_pwrUsbConnected = CheckStatusPowereUSB();
   }
   SetCurrentPowerUSB(m_MaxDevices-1);
```

2.5 Set the Outlet State

Sets the power on/off state of the two outlets.

Name: SetPortPowerUSB
Parameters: int port1, int port2. (0=switch off the power, 1=switch on the power)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
m_port1 = 0;
m_port2 = 1;
SetPortPowerUSB(m_port1, m_port2);
```

2.5 Set the Default power up state

Sets the default power up state of the Power USB (when connected to Computer). If set to 1, the outlet will come on when the attached computer is booted up.

Name: SetDefaultStatePowerUSB
Parameters: int port1, int port2. (0=default off state, 1=default on state)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
SetDefaultStatePowerUSB(m_defaultState1,.m_defaultState2);
```

The functions listed below are available for PowerUSB units released after Nov 2010 firmware Version2

2.6 ReadPortState

Reads the outlet on/off state of the outlets

Name: ReadPortPowerUSB
Parameters: int *port1, int *port2. (0=power is off, 1=power is on)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
Int m_port1, m_port2;
ReadPortPowerUSB(&m_port1, &m_port2);
```

2.7 ReadDefaultPortState

Reads the default outlet on/off state of the outlets. The outlets go into this state when PowerUSB is powered on.

Name: ReadDefaultPortPowerUSB
Parameters: int *port1, int *port2. (0=default power is on, 1=default power is off)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
Int m_port1, m_port2;
ReadDefaultPortPowerUSB(&m_port1, &m_port2);
```

2.6 GetFirmwareVersion

Reads the firmware version of the PowerUSB

Name: GetFirmwareVersion
Parameters: None
Returns: Firmware Version

Version1:
- Basic power control functions

Version2:
- Read the outlet states
- Watchdog functions

The Below Watchdog functions are available for PowerUSB-Watchdog version units released after November 2010 Version 2

Note: In the watchdog version the outlet1 is always on except during powercycle time during which the PowerUSB expects the reboot the attached computer in outlet1

3.1 StartWatchdogTimer

Starts a watchdog in the PowerUSB. The PowerUSB starts a monitoring thread and expects a heartbeat from this moment. If the attached computer misses the heartbeat for certain number of times (probably due to hangup), it switches off the outlet 1 for a certain duration and switches it on to do a PowerCycle. After the PowerCycle, the watchdog timer is stopped. The watchdog timer has to be restarted by the attached computer after the reboot

Name: StartWatchdogTimer
Parameters:

HbTime: This is the time within witch the PowerUSB exepects a heartbeat
numHbMisses: If this many hearbeats are missed, the PowerUSB starts the PowerCycle to reboot
resetTime: The amount of time to swich off the outlet during the PowerCycle

Returns:

>=0 if successful. < 0 on failure

3.2 StopWatcdogTimer

Stops the watchdog timer. The PowerUSB ignores any received heartbets and will not monitor the attached computer

3.3 GetWatchdogStatus

Gets the current mode of the Watchdog

Returns:

0: Watchdog is not active
1: Watchdog is active
2: PowerCycle

3.4 SendHeartBeat

Send a hearbeat message to PowerUSB.

3.5 PowerCycle

The outlet1 is switched off for a specified duration and switched on to reboot the attached computer

Parameter:

resetTime: Amount of time to switch off the outlet 1