POWER USB

4/30/2012

Computer Controlled Power Strip

Software API

Contents

Power USB

SOFTWARE API

1. Introduction

The PowerUSB can be controlled from third party applications through function calls provided in the PowerUSB 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. The basic model will have primary API functions. The Digital IO, Watchdog and Smart models will support additional functions in addition to the primary basic functions.

2. Basic Model

- Works only when connected to computer
- Three controlled outlets and One always on outlets
- Switch on and off controlled outlets
- Set the default on/off state of controlled outlets. Can make the outlet on when connected to computer. Default States are 1: Off, 2: Off, 3: On
- Measure the current consumed and voltage for a given instance and accumulated
- Overload protection in software. Blinks all 3 LED for 10 sec when overload condition happens
- Holding the reset switch for more than 3 seconds will reset the firmware operation

2.1. Initialization

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

Name: InitPowerUSB
Parameters: model:returns the model number(1:basic, 2:digIO, 3:watchdog, 4:Smart), firmware: returns firmware version in ?.? format in a character string (major revision and minor revision)
Return: >0 if successful. Returns number of PowerUSB devices connected
C++ Example:

```
if (!m_pwrUSBInit)
{
    int model;
    char firmware[8];

    if ((ret=InitPowerUSB(&model, firmware)) > 0)
    {
        m_pwrUSBInit = 1;
        m_numDevices = ret;
    }
}
```

2.2. Check PowerUSB connectivity

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

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

```
BOOL CPowerUSBDlg::OnDeviceChange(UINT nEventType, DWORD_PTR dwData)
{
        m_pwrUsbConnected = CheckStatusPowerUSB();
        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 (used 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, all the commands (SetPortPowerUSB) will be directed to the selected PowerUSB. Maximum of 4 powerUSBs are currently supported. This maximum is defined in PwrUSBInc.h as POWER_USB_MAXNUM

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;
   }
   SetCurrentPowerUSB(m_MaxDevices-1);
```

2.5. Set the Outlet State

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

Name: SetPortPowerUSB
Parameters: int port1, int port2, int port3. (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;
m_port3 = 0;
SetPortPowerUSB(m_port1, m_port2, m_port3);
```

2.5. Set the Default power up state

Sets the default power up state of the Power USB (when powerd up through Computer connection). If set to 1, the corresponding outlet will be in on state when powered up.

Name: SetDefaultStatePowerUSB
Parameters: int port1, int port2, int port3. (0=default off state, 1=default on state)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
int m_defaultState1, m_defaultState2, m_defaultState3;
SetDefaultStatePowerUSB(m_defaultState1,m_defaultState2, m_defaultState3);
```

2.6. Read Port State

Reads the on/off state of the outlets

Name: ReadPortPowerUSB
Parameters: int *port1, int *port2, int *port3(0=power is off, 1=power is on)
Returns: >=0 if successful. < 0 on failure
C++ Example:

```
int m_port1, m_port2, m_port3;
ReadPortPowerUSB(&m_port1, &m_port2, &m_port3);
```

2.7. ReadDefaultPortState

Reads the default on/off state of the outlets.

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_Defport1, m_Defport2, m_Defport3;
ReadDefaultPortPowerUSB(&m_Defport1, &m_Defport2,  &m_DefPort3);
```

2.6 Get Firmware Version

Reads the firmware version of the PowerUSB

Name: GetFirmwareVersionPowerUSB
Parameters: char firmware[] . Returns firmware in format A.B where A is the major revision and B is minor revision.
Returns: >= upon success

C++ Example:
```
char firmware[8];
GetFirmwareVersionPowerUSB(firmware);
```

2.7. Get Model

Returns the model of the current connected PowerUSB. The model numbers are Basic=1, DigitalIO=2, Watchdog=3, Smart=4

Name: GetModelPowerUSB
Parameters: None
Returns: Model number integer
C++ Example:
```
modelNumber = GetModelPowerUSB();
```

2.8. Read Current

Returns the current consumed presently by the connected devices. The units returned is in milliamps (ma).
Name: `ReadCurrentPowerUSB`
Parameters: int *current
Returns: >= 0 upon success
C++ Example:
```
int m_current;
ReadCurrentPowerUSB(&m_current);
```

2.9 Read Current Cumulative

Returns the current consumed by the device over a period. The unit returned is in milliamps accumulated each minute (average power consumed each minute). The current consumed is accumulated within the PowerUSB. The accumulation is done till the integer overflow or Reset Current counter function is called. Total power consumed (in kilowatt hour) can be calculated using the example given below.
Name: `ReadCurrentCumPowerUSB`
Parameters: int *current
Returns: >= 0 upon success
C++ Example:
```
int m_currentCum, m_pwrCum, defVoltage=110;
ReadCurrentCumPowerUSB(&m_currentCum);
// kwh= /60:convert min to hr, defV: conver to watt, /1000: convert to kw, /1000: convert ma to Amp
m_pwrCum = (double)m_currentCum / 60 * defVoltage / 1000 / 1000;
```

2.10. Reset Cumulative Current Counter

Resets the accumulated current counter to 0.
Name: `ReadCurrentCumPowerUSB`
Parameters: None
Returns: >= 0 upon success
C++ Example:
```
ResetCurrentCounterPowerUSB();
```

2.11. Set the current Ratio

The default current ratio is 1 and current consumed is returned is in ma. This function should not be used by the user.

2.12. Enable Overload Protection

Enables the sotware overload protection for controlled outlets. By default, the overload protection is enabled in PowerUSB. When overload is detected in the power sensing, the controlled outlets are switched off till further reset. This function should be used at owners risk as the overload condition can kill the Solid State Relay before the fused reset.
Name: `SetOverloadPowerUSB`
Parameters: int 1: enable overload, 0: disable overload
Returns: >= 0 upon success
C++ Example:

```
SetOverloadPowerUSB(1);
```

2.13. Check Overload Protection

Checks if the overload protection is currently enabled.

Name: `GetOverloadPowerUSB`
Parameters: None
Returns: 1: overload enabled, 0: overload disabled
C++ Example:

```
overloadProtection = GetOverloadPowerUSB ();
```

2.14. Reset the Board

Resets the board and restarts the firmware within the board. This function is noramlly not required to be called.
Name: `ResetBoard`
Parameters: None
Returns: >= 0 upon success
C++ Example:

```
ResetBoard ();
```

2.15. Sets the Current Read Offset

Each PowerUSB will have an offset value in the current sensed. This function will set this offset to 0 within the PowerUSB. Once this function is called, the current sensing offset is stored in the flash memory of the PowerUSB. This function should be called only when there is no load in any of the outlets. After this function is called, the future calls to `ReadCurrentPowerUSB` will return offset compensated readings.
Name: `SetCurrentOffset`
Parameters: None
Returns: >= 0 upon success
C++ Example:

```
SetCurrentOffset();
```

3. Digital I/O Model

- Supports all functions of the basic model
- Provides 7 I/O Channels (4 input and 3 output), 5VDC and Ground connections
- Output will be 5-12VDC opto Isolated. 12V input will be sent to output
- Input will be 5-12VDC opto Isolated
- Input will be polling based from the host. Input read resolution is about 250milli second

4.1 SetOutputStatePowerUSB

Sets the output states of the PowerUSB digital output ports. Currently you can set only ports 4 and 5 which are output ports

Parameters: int ouptput[]. 2 output port states are set. Only byte 3 and 4 are used in the array
Returns: >= upon success

C++ Example:
```
int output[7], i;

if (m_dioOptions.DoModal() == IDOK)
{
      for (i = 0; i < 7; i++)
            output[i] = 0;
      output[3] = m_dioOptions.m_port1;
      output[4] = m_dioOptions.m_port2;
      SetOutputStatePowerUSB(output);
}
```

4.2 GetIInputStatePowerUSB

Reads the port states of input ports. Currently reads only port 1, 2 and 3 which are input ports

Parameters: int input[], Sets only byte 0, 1, 2 in the array which are input ports

C++ Example:
```
int input[7];

GetIInputStatePowerUSB(input);
m_outStr2.Format("Input Port Status: %d %d %d", input[0], input[1], input[2]);
```

4.3 GenerateClockPowerUSB (Currently Not Available)

Generates a clock pulse for the output port. Currently sets only port 4 and 5 which are output ports

Parameters:
int port: port number to set the clock (3 or 4)
int onTime: On time in milliseconds
int offTime: Off time in milliseconds

C++ Example:

```
GenerateClockPowerUSB(3, 30, 100);
```

4.4 SetInputTriggerPoewrUSB

Sets a trigger action based on change in input state of one of the 4 inputs.

Parameters:
int input: Trigger Inputs1 to 4. 0: input1 … 3: input4
int fallingSignal: Trigger Condition 0: lo->hi signal, 1: hi->lo signal
int outlet: Outlet to take action on. 1-3 Power Outlets 1, 2, 3
int output: Output to take action on. 1-3 Outputs 1, 2, 3
int outtime: Amount of time in seconds to keep the outlet or output on. -3: noaction, -2: Toggles the output/outlet, -1: Makes it always on, 0: Switch off.
char cond and mask:
A combination bit for the state of the other inputs to be in. The action is taken only if the other inputs are in this state. This will be documented in the next version. For now use 0 for both bytes.

C++ Example:

```
// input, lo->hi, outlet, output, outtime, cond, mask
SetInputTriggerPowerUSB(0, 1, 1, 0, 9, 0, 0);
If the Input 1 goes from lo to hi, power outlet 1 will be switched on for 9
seconds. This is done irrespective of the state of other inputs


SetInputTriggerPowerUSB(3, 0, 0, 2, -2, 0, 0);
If the Input 3 goes from hi to lo, output 2 will be toggled from on to off (vice
versa) state. This is done irrespective of the state of other inputs
```

4.5 SetPLCPowerUSB

Starts or stops the PLC operation in the PowerUSB. Once started, the trigger actions set through SetInputTriggerPowerUSB will be initiated. If set to 0, the actions will not be taken.

Parameters:
int state: 0: switch off trigger, 1: switch on trigger

C++ Example:

```
SetPLCPowerUSB(1);
```

4.6 GetPLCPowerUSB

Gets the status of the PLC running in the PowerUSB.

Parameters:
int state: 0: PLC trigger is off in PowerUSB 1: PLC trigger is running in PowerUSB

C++ Example:

```
int state;
SetPLCPowerUSB(&state);
```

4.7 ClearPLCPowerUSB

Clears the PLC table inside PowerUSB. After each call to SetInputTriggerPowerUSB, a table entry is filled for input trigger. Call this function to erase this table. Typically this function is called before multiple calls to SetInputTriggerPowerUSB.

Parameters:
None

C++ Example:

```
ClearPLCPowerUSB();
```

4. WatchDog Model

- Monitors the attached computer for hang-ups and hard power resets the computer when no heartbeat is seen
- No control over outlet 3 which is controlled through watchdog timer functions
- Other outlets will work normally with full manual control. No timed control

Note: In the watchdog version the outlet3 is always on except during powercycle time during which the PowerUSB expects the reboot the attached computer in outlet1

5.1 StartWatchdogTimerPowerUSB

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

C++ Example:
```
StartWatchdogTimerPowerUSB(20, 3, 15);
```

5.2 StopWatcdogTimerPowerUSB

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

C++ Example:
```
StopWatchdogTimerPowerUSB();
```

5.3 GetWatchdogStatusPowerUSB

Gets the current mode of the Watchdog

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

C++ Example:
```
m_wdStatus = GetWatchdogStatusPowerUSB();
```

5.4 SendHeartBeatPowerUSB

Send a hearbeat message to PowerUSB.

C++ Example:

```
SendHeartBeatPowerUSB();
```

5.5 PowerCyclePowerUSB

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

C++ Example:

```
PowerCyclePowerUSB(15);
```

5.6 ShutdownOffOnPowerUSB

The outlet1 is switched off after a specified period of time and kept off for another specified duration. This feature will provide following benefits

- Prevents hang-ups during Windows shutdown
- Saves vampire power when computer is off
- Provides a clean start of the computer after a determined period of time

Parameter:

offDelay: The computer outlet will be switched off after this period in minutes

onDelay: The computer outlet will be switched back on after this delay in minutes. Outlet will be off for this duration.

C++ Example:

```
ShutdownOffOnPowerUSB(5, 720);

Computer outlet will be switched off after 5 minutes and will remain off for 12
hours
```