ZylBattery 1.39

ZylBattery is a visual Delphi component that diplays the charging level of the battery in a chart. You can specify different colors for different charging levels and you can indicate it also in text format in percents.
Beside this functionality it has many other useful methods to:
- get battery status
- get AC line status
- get battery lifetime
- turn off the computer
- hibernate the computer
- suspend (standby) the computer
- reboot the computer
- turn off monitor
- run screen saver
- log off interactive user
- get Windows version
- enable hibernation

It has more power related events like:
- OnBatteryLow
- OnPowerStatusChange
- OnSuspend
- OnQuerySuspend
- OnResumeSuspend
- OnResumeAutomatic and so on.

Supported Operating Systems:
Windows 2000/XP/Server2003/Vista/Server2008/7/8/Server2012/10
Available for:
Delphi 11.0 Alexandria (Win32 & Win64), Delphi 10.4 Sydney (Win32 & Win64), Delphi 10.3 Rio (Win32 & Win64), Delphi 10.2 (Win32 & Win64), Delphi 10.1 (Win32 & Win64), Delphi 10 (Win32 & Win64), Delphi XE8 (Win32 & Win64), Delphi XE7 (Win32 & Win64), Delphi XE6 (Win32 & Win64), Delphi XE5 (Win32 & Win64), Delphi XE4 (Win32 & Win64), Delphi XE3 (Win32 & Win64), Delphi XE2 (Win32 & Win64), Delphi XE, Delphi 2010, Delphi 2009, Delphi 2007, Delphi 2006, Delphi 7, Delphi 6, C++Builder 11.0 Alexandria (Win32 & Win64), C++Builder 10.4 Sydney (Win32 & Win64), C++Builder 10.3 Rio (Win32 & Win64), C++Builder 10.2 (Win32 & Win64), C++Builder 10.1 (Win32 & Win64), C++Builder 10 (Win32 & Win64), C++Builder XE8 (Win32 & Win64), C++Builder XE7, C++Builder XE6, C++Builder XE5, C++Builder XE4, C++Builder XE3, C++Builder XE2, C++Builder XE, C++Builder 2010, C++Builder 2009
Remarks:
- The Delphi 2006 version is fully compatible with Turbo Delphi

Insatallation:

If you have a previous version of the component installed, you must remove it completely before installing this version. To remove a previous installation, proceed as follows:
-Start the IDE, open the packages page by selecting Component - Install Packages
-Select ZylBatteryPack package in the list and click the Remove button
-Open Tools - Environment Options - Library and remove the library path pointing to ZylBattery folder
-Close the IDE
-Browse to the folder where your bpl and dcp files are located (default is $(DELPHI)\Projects\Bpl for Delphi).
-Delete all of the files related to ZylBattery
-Delete or rename the top folder where ZylBattery is installed
-Start regedit (click Start - Run, type "regedit.exe" and hit Enter). Open the key HKEY_CURRENT_USER\Software\Borland\<compiler>\<version>\Palette and delete all name/value items in the list related to ZylBattery. (<compiler> is either "Delphi" or "C++Builder", <version> is the IDE version you have installed)

-Unzip the zip file and open the ZylBatteryPack.dpk file in Delphi, compile and install it and add to Tools/Environment Options/Library (in older Delphi menu) or Tools/Options/Delphi Options/Library/Library Path (in newer Delhi menu) the path of the installation (where the ZylBattery.dcu file is located). The component will be added to the "Zyl Soft" tab of the component palette. After you have the component on your component palette, you can drag and drop it to any form, where you can set its properties by the Object Inspector and you can write event handlers selecting the Events tab of the Object Inspector and double clicking the preferred event.

-It is indicated to use this component with "Stop on Delphi exception" option deactivated. You can do this from Delphi / C++Builder menu, Tools/Debugger Options/Language Exceptions/Stop on Delphi exceptions in older versions or Tools/Options/Debugger Options/Embarcadero Debuggers/Language Exceptions/Notify on language exceptions in newer versions, otherwise you will have a break at all the handled exceptions., otherwise you will have a break at all the handled exceptions

Constants:

PBT_APMQUERYSUSPEND = $0000;
PBT_APMQUERYSTANDBY = $0001;
PBT_APMQUERYSUSPENDFAILED = $0002;
PBT_APMQUERYSTANDBYFAILED = $0003;
PBT_APMSUSPEND = $0004;
PBT_APMSTANDBY = $0005;
PBT_APMRESUMECRITICAL = $0006;
PBT_APMRESUMESUSPEND = $0007;
PBT_APMRESUMESTANDBY = $0008;
PBTF_APMRESUMEFROMFAILURE = $00000001;
PBT_APMBATTERYLOW = $0009;
PBT_APMPOWERSTATUSCHANGE = $000A;
PBT_APMOEMEVENT = $000B;
PBT_APMRESUMEAUTOMATIC = $0012;

Types:

TBatteryStatus = (bsHigh, bsLow, bsCritical, bsCharging, bsNoBattery, bsUnknown)
TBatteryStatusSet = set of TBatteryStatus;
TACLineStatus = (acsOffline, acsOnline, acsUnknown)
TOEMEvent = procedure(Sender: TObject; EventCode : Integer) of object
TQuerySuspendEvent = procedure(Sender: TObject; var CanSuspend : Boolean) of object
TACLineStatusChangeEvent = procedure(Sender: TObject; NewACLineStatus: TACLineStatus)

of object;
TWindowsVersion = (wvUnknown, wvWin95, wvWin95OSR2, wvWin98, wvWin98SE, wvWinME, wvWinNT31, wvWinNT35, wvWinNT351, wvWinNT4, wvWin2000, wvWinXP, wvWin2003, wvVista)

Properties:

Active: Boolean - if this property is true, the charging level is updated periodically (determined by UpdateInterval property) in the chart.
Align: TAlign- determines how the control aligns within its container (parent control)
Anchors: TAnchors - specifies how the control is anchored to its parent
Constrains: TSizeConstrains - specifies the size constraints for the control
BackColor: TColor - background color
BorderStyle: TBorderStyle - determines the style of the line drawn around the perimeter of the control
CriticalColor: TColor - foreground color used when the battery status is critical
LowColor: TColor - foreground color used when the battery status is low
HighColor: TColor - foreground color used when the battery status is high
NoBatteryColor: TColor - foreground color used when is no battery
ChargingColor: TColor - foreground color used for charging animation
Font: TFont - controls the attributes of text written on or in the control
Kind: TGaugeKind - horizontal / vertical
ParentColor: Boolean - determines where a control looks for its color information
ParentFont: Boolean - determines where a control looks for its font information
ParentShowHint: Boolean - determines where a control looks to find out if its Help Hint should be shown
PopupMenu: TPopupMenu - identifies the pop-up menu associated with the control
ShowCharging: Boolean - shows / hides charging color
ShowText: Boolean - shows charging level in text format too
ShowHint: Boolean - determines whether the control displays a Help Hint when the mouse pointer rests momentarily on the control
UpdateInterval: Cardinal - time period in milli-seconds when the charging level is updated in the chart
Version: Double - returns the verion number of the component
Visible: Boolean - shows / hides battery chart

Public Methods:

constructor Create(AOwner: TComponent) - contructor
destructor Destroy - destructor
procedure UpdateBattery - updates the battery chart
function GetBatteryStatus(): TBatteryStatusSet - returns battery status
function GetACLineStatus(): TACLineStatus - returns AC line status
function GetBatteryLifeTime(): Cardinal - returns battery lifetime
function GetBatteryFullLifeTime(): Cardinal - returns battery full lifetime
function GetBatteryLifePercent(): Cardinal - returns battery life percent
function HasBattery(): Boolean - returns true if battery present, false otherwise
procedure Standby(Force: Boolean = True) - puts the computer in standby
procedure Hibernate(Force: Boolean = True) - hibernates the computer
procedure TurnOff(Force: Boolean = True) - turns the computer off
procedure Reboot(Force: Boolean = True) - restarts the computer
procedure LogOff(Force: Boolean = True) - logs the interactive user off
procedure TurnOffMonitor() - turns the monitor off
procedure ScreenSaver() - lunchs screen saver
procedure EnableHibernation() - enables hibernation

Events:

OnQuerySuspend - fires at request for permission to suspend
OnQueryStandby - fires at request for permission to standby.
OnQuerySuspendFailed - fires when suspension request is denied
OnQueryStandbyFailed - fires when standby request is denied
OnSuspend - fires when system is suspending operation
OnStandby - fires when system is standbying operation
OnResumeCritical - fires when operation resuming after critical suspension
OnResumeSuspend - fires when operation resuming after suspension
OnResumeStandby - fires when operation resuming after standby
OnResumeFromFailure - fires when operation resuming after failure
OnBatteryLow - fires when battery power is low.
OnPowerStatusChange - fires when power status has changed.
OnOEMEvent - fires when OEM-defined event occurred
OnResumeAutomatic - fires when operation resuming automatically after event
OnACLineStatusChange - fires when AC line status change (power failure)

Buy Now!

http://www.zylsoft.com
info@zylsoft.com