ZylTimer 1.35

ZylTimer is a high resolution, long-term Delphi / C++Builder timer component which provides a higher precision than the standard Delphi / C++ Builder TTimer component.

TTimer component which ships with Delphi / C++Builder uses the Windows Message Queue to generate the OnTimer event. Due to this approach it's impossible to get accurate timer intervals smaller than 15 milliseconds.

ZylTimer is a thread based timer and due to this architecture provides a higher precision, close to 1 millisecond (it's possible that you cannot obtain a clear resolution of 1ms on all the systems, but it must be at least 2ms) which is inevitable in time critical applications. The OnTimer event is always fired in time which is not available for the standard TTimer, when there are too many messages processed.

The standard TTimer component uses the SetTimer API function, so the interval is limited to 2147483647 milliseconds (about 25 days). ZylTimer is a thread based timer and the interval could be set even to hundred years if necessary.

The demo version is fully functional in Delphi and C++Builder IDE, but it displays a nag dialog (the licensed version will, of course, not have a nag dialog and will not be limited to the IDE). The package includes demo programs for Delphi and C++Builder and a help file with the description of the component.

Supported Operating Systems: 2000/XP/Server2003/Vista/Server2008/7/8/Server2012/10/11
Available for: Delphi 21.0 (Win 32 & Win64), Delphi 11.0 (Win 32 & Win64), Delphi 10.4 (Win 32 & Win64), Delphi 10.3 (Win 32 & Win64), Delphi 10.2 (Win 32 & Win64), Delphi 10.1 (Win 32 & Win64), Delphi 10 (Win 32 & Win64), Delphi XE8 (Win 32 & Win64), Delphi XE7 (Win 32 & Win64), Delphi XE6 (Win 32 & Win64), Delphi XE5 (Win 32 & Win64), Delphi XE4 (Win 32 & Win64), Delphi XE3 (Win 32 & Win64), Delphi XE2 (Win 32 & Win64), Delphi XE, Delphi 2010, Delphi 2009, Delphi 2007, Delphi 2006, Delphi 2005, Delphi 7, Delphi 6, Delphi 5, Delphi 4, C++Builder 12.0 (Win 32 & Win64), C++Builder 11.0 (Win 32 & Win64), C++Builder 10.4 (Win 32 & Win64), C++Builder 10.3 (Win 32 & Win64), C++Builder 10.2 (Win 32 & Win64), C++Builder 10.1 (Win 32 & Win64), C++Builder 10 (Win 32 & Win64), C++Builder XE8 (Win 32 & 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, C++Builder 2007, C++Builder 2006, C++Builder 6, C++Builder 5, Turbo Delphi, Turbo C++
Remarks:
- The Delphi 2006 version is fully compatible with Turbo Delphi
- The C++Builder 2006 version is fully compatible with Turbo C++
- Avoid to use time consuming code in the OnTimer event, but if it's necessary, run it in a different thread.

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 ZylTimerPack package in the list and click the Remove button

-Open Tools - Environment Options - Library and remove the library path pointing to ZyTimer folder

-Close the IDE

-Browse to the folder where your bpl and dcp files are located (default is $(DELPHI)\Projects\Bpl for Delphi, $(BCB)\Projects\Bpl for C++ Builder). -Delete all of the files related to ZylTimer

-Delete or rename the top folder where ZylTimer 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 ZylTimer. (<compiler> is either "Delphi" or "C++Builder", <version> is the IDE version you have installed)

-Unzip the zip file and open the ZylTimerPack.dpk file in Delphi (ZylTimerPack.bpk file in C++Builder), compile and install it

and add to Tools/Environment Options/[Language]/Library (in Delphi/C++Builder menu) the path of the installation (where the ZylTimer.dcu or ZylTimer.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.

If you still have problems in C++Builder, running an application, which contains the component, then open the project and in C++Builder menu, Project/Options/Packages and uncheck "Build with runtime packages".

C++BuilderXE2: If you get access violation, running an application with this component, then comment out in the project's main form's cpp file the following: //#pragma link "ZylTimer".

-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", otherwise you will have a break at all the handled exceptions

-Avoid to use time consuming code in the OnTimer event, but if it's necessary, run it in a differnet thread.

Properties:

Cycled: Boolean - if this property is true, then the OnTimer event will be fired periodically. Otherwise the event will be fired only once.

Enabled: Boolean - enables or disables the timer. If this property is true, then the timer will be started, otherwise it will be stopped.

Interval: Double - the time interval in seconds (0.001 second = 1 millisecond). The OnTimer event is fired repeatedly after this measured interval.

NeedSynchronization: Boolean - set this property to true for thread safety. If you use the component in ActiveX environment, set this property to false and use a CriticalSection from the main application if you need synchronization. The default value is false.

Priority: TThreadPriority - priority of the timer thread.

RepeatCount: Longword - specifies how many times the OnTimer event will be fired (only if Cycled is true). If this property is 0 and Cycled is true, then the event will be fired continually. Please keep in mind, that this property has no effect if Cycled is set to false.

Tag: Longint - stores an integer value as part of a component. Tag has no predefined meaning. The Tag property is provided for the convenience of developers. It can be used for storing an additional integer value or it can be typecast to any 32-bit value such as a component reference or a pointer.

Functions:

constructor Create(AOwner: TComponent) - constructor
destructor Destroy - destructor
function GetCycleNumber: Longword - returns the elapsed cycles
procedure Start - starts the timer (sets Enabled to true)
procedure Restart - restarts the timer
procedure Stop - stops the timer (sets Enabled to false)

Events:
OnTimer: TNotifyEvent - This event is fired repeatedly after the measured interval. Write the code that you want to occur at the specified time interval inside this event.

Buy Now!

Copyright by Zyl Soft 2003 - 2023
http://www.zylsoft.com
info@zylsoft.com