

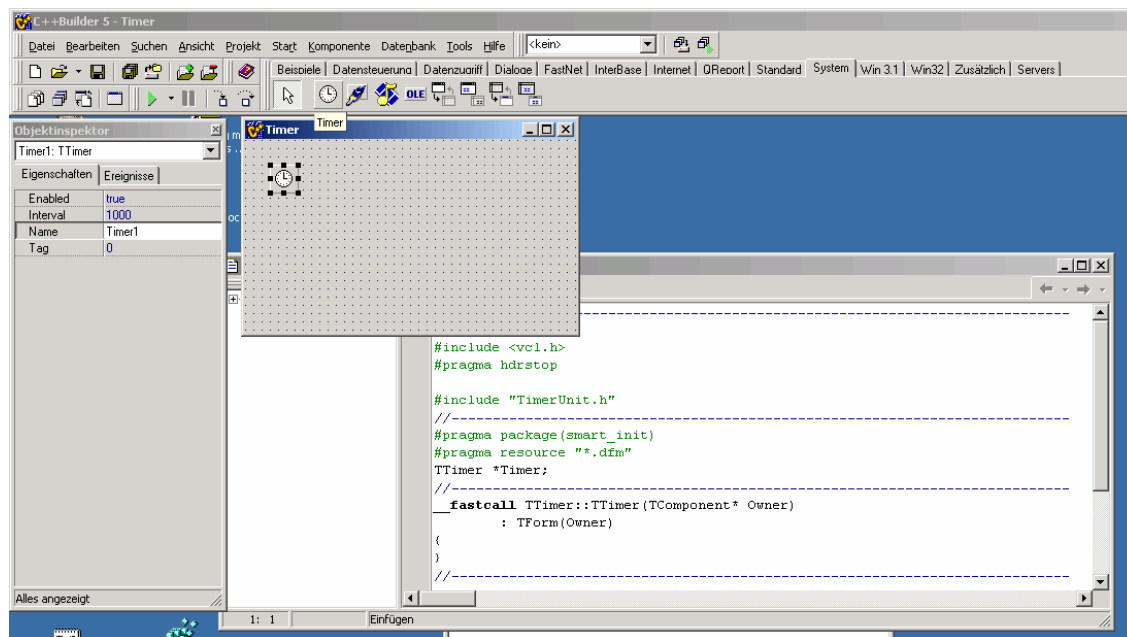
Programm-Ressourcen II

Überblick

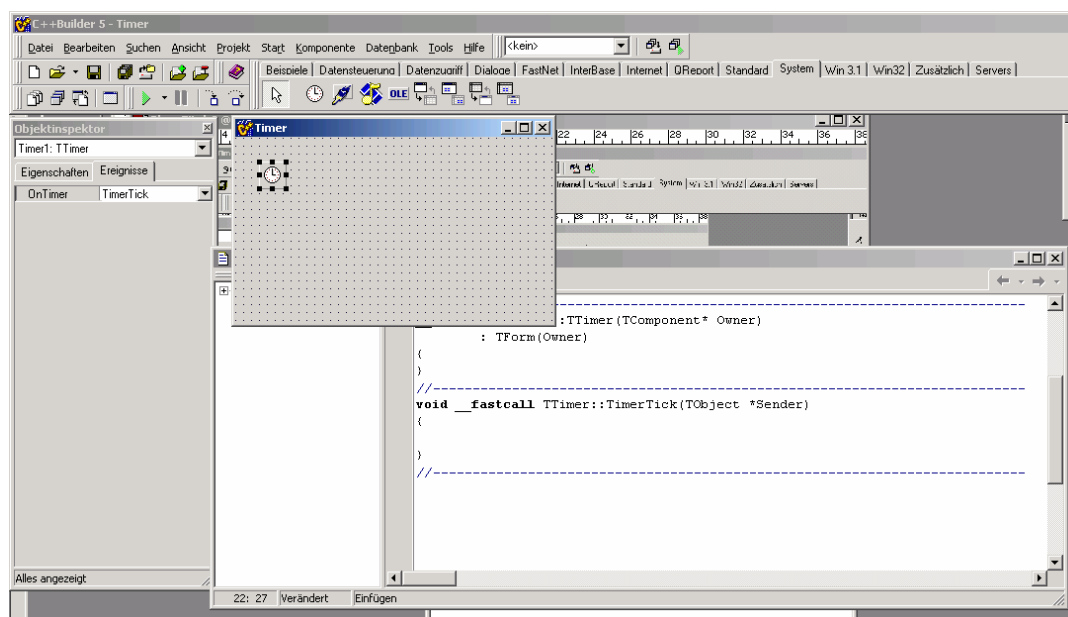
Neben den Standardressourcen wie Label, Button, Dialoge Menüs, etc. zur Gestaltung der grafischen Benutzeroberfläche werden auch Systemressourcen zur Verfügung gestellt. Sie umfassen das Umfeld der Kommunikation (OLE, DDE, Netzwerk, Internet, SQL) wie auch direkte Systemressourcen, z.B. Timer.

Timer

Ein Timer ist ein Systemobjekt, welches ein zeitgesteuertes, periodisches Ereignis auslösen kann. Das kleinstmögliche Intervall beträgt 1ms und kann in ms-Inkrementen definiert werden.



Das Ereignis wird nach Ablauf der Intervallzeit ausgelöst. Durch Doppelklick auf den Timer wird das Gerüst für die Ereignisfunktion erzeugt. Sie wird nach dem Ablauf des Timers (Intervall in ms) aufgerufen:



Aufgabe

Zu realisieren ist eine Digitaluhr, welche in einem kleinen Fenster immer On-Top die Zeit anzeigt.



Beim Start des Programms soll die Zeit vom Betriebssystem gelesen und übernommen werden. Alle Sekunden soll durch ein Timerereignis eine interne Uhr weitergezählt werden und die aktuelle Zeit neu angezeigt werden.

Vorgehen

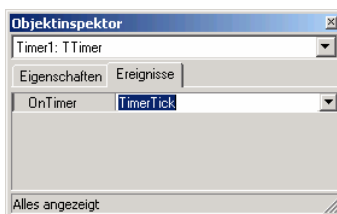
1. Projekt in bekannter Manier aufsetzen. Hauptprogramm und Unit umbenennen und abspeichern. Prüfen, ob Projekt kompilierbar und lauffähig ist.
2. Formulargröße auf ca 160x90 Pixel einstellen. (Mit Maus ziehen oder über Eigenschaften ClientWidth-ClientHeight setzen.

Das Fenster soll in fester Größe, ohne Systemmenü und Minimieren-Maximieren-Felder erscheinen. Dazu sind die Eigenschaften

```
BorderStyle = bsSingle  
FormStyle = fsStayOnTop  
BorderIcons->biMaximize = false  
BorderIcons->biMinimize = false  
BorderIcons->biSystemMenu = false
```

zu setzen.

3. Timerobjekt aus der System-Komponentenleiste auf das Formular setzen. Die Timer-Ereignisfunktion mit Namen definieren, z.B. TimerTick.
Es wird daraufhin ein Codegerüst im Editorfenster erzeugt. Der eigentliche Code wird nachher eingebracht.



Antworten zu Fragen aus dem Unterricht:

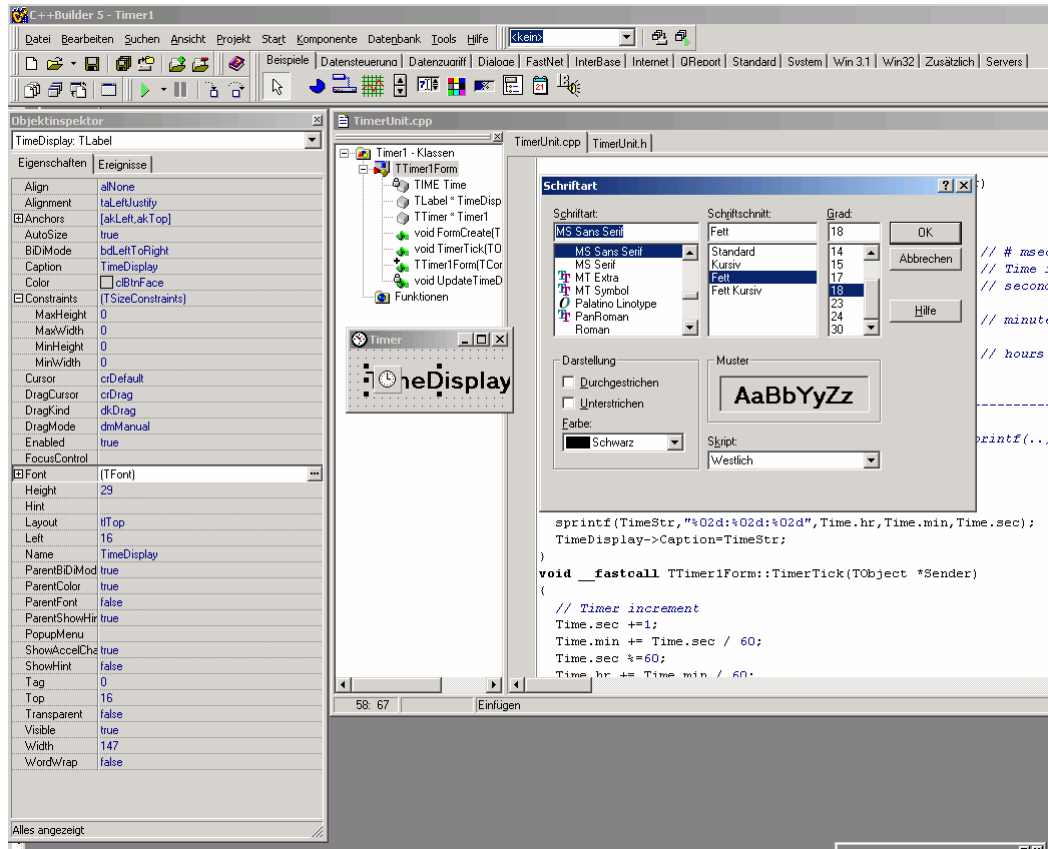
- Wo muss TimerTick genau definiert werden?

TimerTick wird direkt in den Objekteigenschaften eingeschrieben. Es öffnet sich nachher ein Funktionsgerüst im Editorfeld

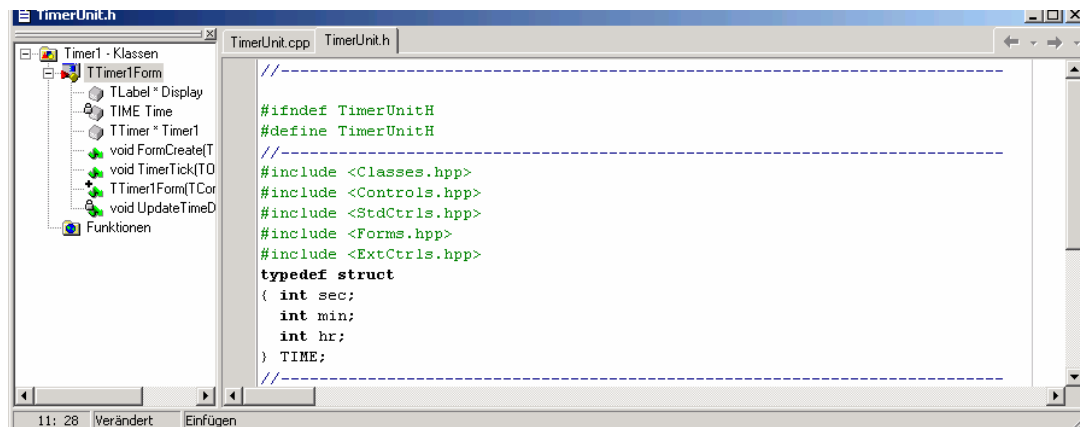
```
void __fastcall TTimerForm::TimerTick(TObject *Sender)  
{  
  
}
```

- Die definierte Funktion TimerTick verschwindet nach dem Kompilieren!
 Vermutlich wurde nur das reine Codegerüst ohne weitere Inhalte kompiliert. C++ Builder entfernt diesen aus Sicht des Programmablaufes unnötigen Code automatisch.

4. Labelfeld für die zu anzeigende Zeit aus der Standardkomponentenleiste platzieren. Schrift in den Eigenschaften mit 18p-SansSerif-Fett wählen:



5. Zur internen Speicherung der Zeitinformation wird eine benutzerdefinierte Struktur TIME im Headerfile definiert. Sie wird direkt im Editorfenster nach den #includes eingegeben:



Dann private Membervariable mit dem Namen time des Typs TIME in der Klassenansicht über „neues Feld“ definieren.

6. Initialisierung der Uhr durch Lesen der Systemzeit in `FormCreate()`. Das Lesen der aktuellen Zeit wird in Borland C++ Builder mit dem `TDateTime`-Objekt unterstützt. Die Memberfunktion `currentTime()` liefert als Nachkommaanteil die Anzahl Millisekunden seit 00:00. Die Millisekunden werden umgerechnet und den einzelnen Strukturkomponenten zugewiesen:

```

//
// Gerhard Krucker
// 17.12.2001

#include <vcl.h>
#pragma hdrstop
#include <stdio.h> // sprintf()
#include "TimerUnit.h"

//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TTimerForm *TimerForm;
//-----

__fastcall TTimerForm::TTimerForm(TComponent* Owner)
: TForm(Owner)
{
}

//-----
// Initialize and preset time counters reading systems time of day clock.
// Systems time can be read using a TDateTime VCL object.
//

void __fastcall TTimerForm::FormCreate(TObject *Sender)
{
    TDateTime DateTime; // DateTime Object
    int time;
    // Get System Clock Time
    time = (int)(1000*60*60*24* DateTime.CurrentTime()); // # mseconds since 00:00
    time = time/1000; // Time in seconds since 00:00
    Time.sec=time % 60; // seconds
    time /=60;
    Time.min=time %60; // minutes
    time /=60;
    Time.hr=time; // hours
    UpdateTimeDisplay();
}

//-----
// Format time string using ANSI C format functions
// (Don't forget to include <stdio.h> headerfile for sprintf(..))

void TTimerForm::UpdateTimeDisplay()
{
    static char TimeStr[10];

    sprintf(TimeStr,"%02d:%02d:%02d",Time.hr,Time.min,Time.sec);
    TimeDisplay->Caption=TimeStr;
}
    
```

7. Die formatierte Ausgabe der Zeit erfolgt durch Einschreiben in die Caption des Labelfeldes (hier mit dem Namen `TimeDisplay`). Dazu wird eine Methode für das Objekt definiert (In der Klassenansicht rechte Maustaste- `UpdateTimeDisplay` des Typs `void` und ohne Parameter definieren). Aus Einfachheitsgründen erfolgt die formatierte Ausgabe in einen String mit `sprintf(..)` der ANSI-C Bibliothek. Hierzu ist `<stdio.h>` einzubinden.

- Der `TimerTick()` bewirkt das Sekundeninkrement. Am Schluss wird die neue Zeit durch Methodenaufruf `UpdateTimeDisplay()` angezeigt.

```
void TTimer1Form::UpdateTimeDisplay()
{ static char TimeStr[10];

  sprintf(TimeStr, "%02d:%02d:%02d", Time.hr, Time.min, Time.sec);
  TimeDisplay->Caption=TimeStr;
}

void __fastcall TTimer1Form::TimerTick(TObject *Sender)
{
  // Timer increment
  Time.sec +=1;
  Time.min += Time.sec / 60;
  Time.sec %=60;
  Time.hr += Time.min / 60;
  Time.min %=60;
  Time.hr %=24;
  UpdateTimeDisplay();
}
//-----
```

- Icon für die Applikation und Beschriftung über die Projektoptionen einbringen. Das Icon ist auf dem Pool-Laufwerk im Verzeichnis TI greifbar.
- Anwendung testen. Man kann die Uhr zur raschen Prüfung der Zählung schneller laufen lassen, indem im Timerobjekt das Intervall herunergesetzt wird, z.B. 10ms oder 1ms.