20.3.2 Projekte 2

20.3.2.1 Beispiel 5 – CountDown 1

Abbildung 20.3.2.1.1: Geschenke schon beisammen?

Quelltext:

```
Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
  CountDown.Delay = 1000 ' Auslese-Takt für die aktuelle Zeit (kleiner oder gleich 1000ms)!
  CountDown.Start
  CountDown.Trigger
  MovieBox1.Playing = True
  lblText2.Text &= " " & Format$(Now(), "yyyy")
End ' Form_Open()

Public Sub CountDown_Timer()
  Dim dCountDown As Date
  Dim dMonth, dWeek, dDay, dHour, dMinute, dSecond As Integer
  Dim iRestZeitInSekunden, iZeitInSekunden, iTage, iStunden, iMinuten, iSekunden As Integer

  dCountDown = Date(Year(Now()), 12, 24, 0, 0, 0, 0) ' Weihnachten; 24.12.

  If DateDiff(Now, dCountDown, gb.Second) > 0 Then
     iZeitInSekunden = DateDiff(Now, dCountDown, gb.Second)
     iTage = Int(iZeitInSekunden / (24 * 60 * 60))
     lblTage.Text = Str(iTage)
     iRestZeitInSekunden = iZeitInSekunden - iTage * 24 * 60 * 60
     iStunden = Int(iRestZeitInSekunden / (60 * 60))
     lblStunden.Text = Str(iStunden)
     iRestZeitInSekunden = iRestZeitInSekunden - (iStunden * 60 * 60)
     iMinuten = Int(iRestZeitInSekunden / 60)
     lblMinuten.Text = Str(iMinuten)
     iRestZeitInSekunden = iRestZeitInSekunden - (iMinuten * 60)
     iSekunden = iRestZeitInSekunden
     lblSekunden.Text = Str(iSekunden)
  Endif ' DateDiff(Now, dCountDown, gb.Second) > 0 ?

End ' CountDown_Timer()
```

20.3.2.2 Beispiel 6 – CountDown 2

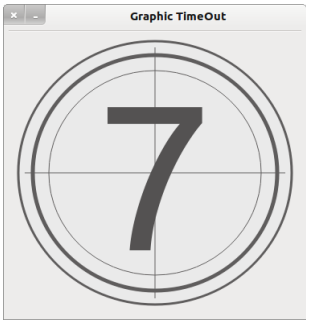

Abbildung 20.3.2.2.1: CountDown im Bereich von 1 bis 99

Die Besonderheit bei diesem Projekt besteht darin, dass eine selbst entwickelte *TimeOut-Komponente* mit integriertem Timer verwendet wird. Sie können den CountDown-Zähler im Intervall von 1 bis 99 vorgeben. Im Bereich von 1-9 werden die Ziffern in einer automatisch angepassten Größe dargestellt. Die Taktfrequenz sowie diverse Farbeinstellungen für Flächen, Kreise, Linien oder den Text (Ziffern) können Sie zum Beispiel im Eigenschaften-Fenster der TimeOut-Komponente einstellen.

Quelltext:

```
Public iCount As Integer

Public Sub Form_Open()
  FCountDown.Center
  FCountDown.Resizable = False
  Timer1.Delay = 100
  Timer1.Start
  Timer1.Trigger
' Eigenschaften der Komponente TimeOut festlegen:
' Entweder über das Eigenschaften-Fenster oder im Quelltext

' Ein TimeOut-Steuerelement ist bereits auf das Formular gezogen worden
  TimeOut1.Delay = 1000
  TimeOut1.Count = 9

End ' Form_Open()

Public Sub Timer1_Timer()

  If TimeOut1.Finished Then
     Timer1.Stop
       Wait 1
     TimeOut1.Delete
       Wait 1
     FCountDown.Close
    ' Hauptprogramm starten ...
  Endif ' TimeOut1.Finished = TRUE ?

End ' Timer1_Timer()
```

Der Timer *Timer1* wird im *Testprogramm* nur benötigt, um in einem 100ms-Intervall zu prüfen, ob der Countdown-Zähler auf Null steht.

20.3.2.3 Beispiel 7 – CountDown 3

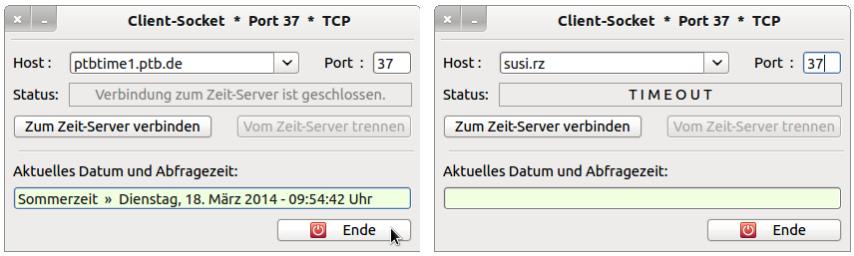

Abbildung 20.3.2.3.1: Zeitserver

Wenn ein Zeitserver nicht innerhalb einer vorgegebenen Zeit erreichbar ist und einen Zeitstempel zurückgibt, dann wird die Verbindungsaufnahme abgebrochen und und der Zustand mit TIMEOUT dokumentiert.

Quelltext:

Innerhalb der Prozedur 'Verbindungsaufbau zum Zeitserver' wird die TimeOut-Zeit festgelegt und der TimeOut-Timer gestartet, der hier nach 5 Sekunden gestoppt wird:

```
 If TCPIP_Socket.Status > Net.Inactive Then
     btnDisconnect.Enabled = True
     lblStatus.Text = "Verbindung zum Zeit-Server wird aufgebaut ..."
     TimeOut.Delay = 5000 ' TimeOut = 5 Sekunden
     TimeOut.Start
  Endif ' TCPIP_Socket.Status > Net.Inactive ?
```

```
Public Sub TimeOut_Timer()
  TimeOut.Stop
  If TCPIP_Socket.Status <> Net.Connected Or lTimeResult = 0 Then
     If TCPIP_Socket.Status > 0 Then Close #TCPIP_Socket
     Set_Interface(False)
     lblStatus.Text = "T I M E O U T"
  Endif ' TCPIP_Socket.Status <> Net.Connected ?
End ' TimeOut_Timer()
```

20.3.2.4 Beispiel 8 – Alarm-Simulation

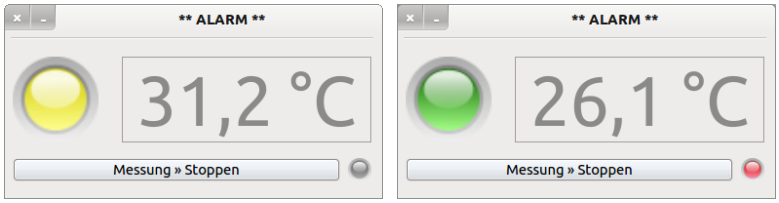

Abbildung 20.3.2.4.1: Simulation

In diesem Projekt werden fünf Timer sowie die Funktion Timer() eingesetzt, um ein Experiment zu simulieren. Die große LED zeigt den Temperatur-Zustand *qualitativ* an (grün, gelb, orange und rot). Die genaue Temperatur wird zeitgesteuert ausgelesen und angezeigt.
Überschreitet die Temperatur einen bestimmten Wert, dann wird die Heizung ausgeschaltet und die kleine LED *blinkt* rot, um den Alarmzustand zu dokumentieren – selbst dann, wenn die Temperatur auf den Normalwert zurückgegangen ist und die Heizung im Intervall-Betrieb (Heizung im oszillierenden Modus) arbeitet. In diesem Modus ändert sich die Temperatur mit (25 ± 0,2)°C nur geringfügig. Auch dieses Verhalten wird mit einem Timer und einem Zufallsgenerator simuliert.
In der Prozedur *pibAlarmFlash_Enter()* wird die Timer-Funktion *Timer()* verwendet, um aus der Startzeit der Messung – ermittelt mit fStartZeit = Timer() – die Zeit-Differenz zu berechnen, wie lange der Alarm bereits besteht. Die Zeit wird in einem Balloon angezeigt, wenn man mit der Maus über der blinkenden Alarm-LED steht:

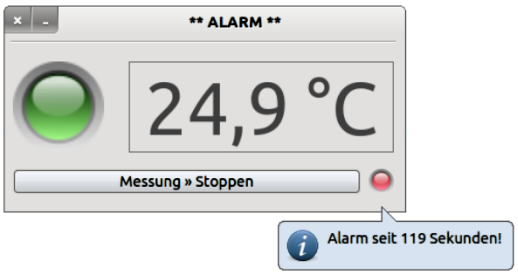

Abbildung 20.3.2.4.2: Anzeige Alarmzeit

Quelltext-Auszug mit ausgewählten Prozeduren:

```
Public fTemperatur As Float = 20.0
Public fStartZeit As Float

Public Sub Form_Open()
  FMain.Center
  FMain.Resizable = False
  TimerAlarm.Delay = 100
  TimerUp.Delay = 100
  TimerDown.Delay = 100
  TimerFlash.Delay = 500
  TimerZufall.Delay = 1000
  SetLEDColor("green")
  lblTemperatur.Text = Format(fTemperatur, "##.0 °C")
  btnStart.Text = "Messung " & String.Chr(187) & " Starten"
  Randomize
End ' Form_Open()

Public Sub TimerZufall_Timer()
  Dim fZufallsTemperatur As Float

  fZufallsTemperatur = Rnd(24.8, 25.2)
  lblTemperatur.Text = Format(fZufallsTemperatur, "##.0 °C")
End ' TimerZufall_Timer()

Public Sub TimerAlarm_Timer()
  If fTemperatur < 30 Then
     SetLEDColor("green")
  Else If (fTemperatur >= 30 And fTemperatur < 32) Then
       SetLEDColor("yellow")
  Else If (fTemperatur >= 33 And fTemperatur < 34) Then
       SetLEDColor("orange")
  Else If fTemperatur >= 34 Then
       SetLEDColor("red")
```

```
    ' Die Heizung wird ausgeschaltet ...
      TimerUp.Stop
      TimerDown.Start
      TimerFlash.Start
      TimerFlash.Trigger
      fStartZeit = Timer()
  Endif ' Temperatur im Bereich ?
End ' TimerAlarm_Timer()

Public Sub pibAlarmFlash_Enter()
  Dim fAlarmZeit As Float
  Dim sMessage As String

  If TimerFlash.Enabled = True Then
     fAlarmZeit = Round(Timer() - fStartZeit, 0)
     sMessage = "Alarm seit " & Str(fAlarmZeit) & " Sekunden! "
     Balloon.Info(sMessage, Last)
  Endif ' fStartzeit > 0

End ' pibAlarmFlash_Enter()
```

20.3.2.5 Beispiel 9 – Zeitgesteuerte Messwerterfassung

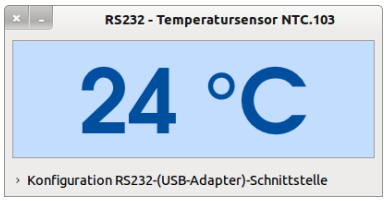

Abbildung 20.3.2.5.1: Temperatur-Anzeige

Nach einem Analog-Digital-Wandler wird der digitale Temperaturwert über ein serielles USB-RS232-Interface zeitgesteuert eingelesen und angezeigt.

Relevante Quelltext-Abschnitte zur Demonstration des Konzepts für den Einsatz der Klasse Timer:

```
PUBLIC SUB Start()
...
  GetValueTimer.Delay = 100 ' Alle 100·ms wird die Temperatur ausgelesen
  GetValueTimer.Start       ' Synonym für GetValueTimer.Enabled = TRUE
  GetValueTimer.Trigger     ' Löst das Timer-Event sofort aus
...
END ' Start()

PUBLIC SUB GetValueTimer_Timer()
  IF RS232.Status <> Net.Active THEN
     lblTemperaturAnzeige.Text = "--- °C"
     rbLED.ForeColor = Color.Red
  ELSE
     lblTemperaturAnzeige.Text = iTemperaturByte & " °C"
     rbLED.ForeColor = Color.Green
  ENDIF
END ' GetValueTimer_Timer()
```