20.6.2 Task-Projekte 3 und 4

Interessant ist bei beiden Projekten die Realisierung der Datenübertragung. Obgleich bei beiden Projekten die Daten mit der Print- oder Error-Anweisung gesendet werden, stehen im Projekt 4 die Funktionen zur Serialisierung und De-Serialisierung im Mittelpunkt, mit denen die Übertragung von Daten mit unterschiedlichen nativen Daten-Typen möglich wird.

20.6.2.1 Projekt 3

Das Projekt 3 ist eine Variante des Projektes 1. Aus dem gestarteten Task werden *kontinuierlich* Daten an das (Haupt-)Programm gesendet.

Mit festem Takt werden nacheinander der Wochentag, das aktuelle Datum und die lokale Zeit für New York berechnet und als *String* an das (Haupt-)Programm geschickt. Die Taktzeit wird als Argument einmalig an den Task übergeben.

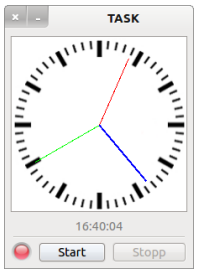

Abbildung 20.6.2.1.1: Normal-Betrieb mit analoger und digitaler (Lokal-)Zeit

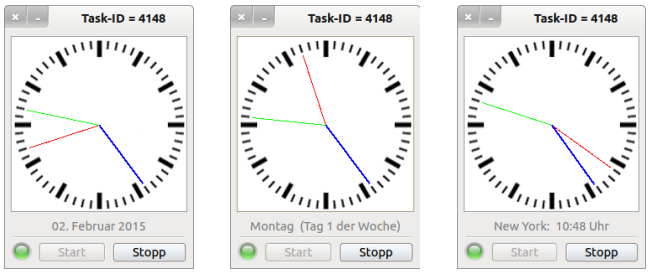

Abbildung 20.6.2.1.2: Hauptprogramm mit gestartetem Task

Den Quelltext der Klasse *TripleTask.class* können Sie vollständig nachlesen:

```
' Gambas class file

Inherits Task

Public iWaitTime As Integer ' Start-Argument für die Wartezeit zwischen den drei Anzeigen

Public Sub Main()
  Dim aTagesListe As String[]
  Dim dTNY, dDate As Date

  aTagesListe = Split("Sonntag,Montag,Dienstag,Mittwoch,Donnerstag,Freitag,Samstag", ",")

  Do ' Kontinuierliche, aber getaktete Ausgabe von Daten (Task → Programm)
    dDate = Now() ' Zeitstempel
    Print aTagesListe[WeekDay(dDate)] & "  (Tag " & Str(WeekDay(dDate)) & " der Woche)" ' DatenTyp = String
    Wait iWaitTime ' Wartezeit
    Print Format$(dDate, "dd. mmmmm yyyy") ' DatenTyp = String
    Wait iWaitTime
    dTNY = Time(DateAdd(Now, gb.Hour, -6)) ' NewYork-LocalTime
    Print "New York:  " & Format(dTNY, "hh:nn") & " Uhr" ' DatenTyp = String
    Wait iWaitTime
  Loop

End ' Main()
```

Der Quelltext für das (Haupt-)Programm wird nur in Auszügen angegeben:

```
Public Sub btnTaskStart_Click()
  If $hTask Then $hTask = Null ' Ein bestehendes Task-Objekt wird zerstört
  $hTask = New TripleTask As "TripleTask" ' = Task-Klassenname
  $hTask.iWaitTime = 3 ' Belegung einer globalen Variablen in der Klasse TripleTask
...
End ' btnTaskStart_Click()

Public Sub TripleTask_Read(Data As String)
  lblAdditionalInformations.Text = Replace(Data, gb.NewLine, "")
End ' TripleTask_Read(Data As String)

Public Sub TripleTask_Error(Data As String)
 lblAdditionalInformations.Text = Replace(Data, gb.NewLine, "")
End ' TripleTask_Error(Data As String)

Public Sub TripleTask_Kill()
  SetLEDColor("red")
  btnTaskStopp.Enabled = False
  FMain.Text = "TASK"
  lblAdditionalInformations.Text = "Task-Fehler!"
End ' TripleTask_Kill()
```

20.6.2.2 Projekt 4

Kernstück im Projekt 4 ist dieses Modul:

```
' Gambas module file

Public Function Encode(vValue As Variant) As String
  Dim hStream As Stream
  Dim sBinary As String

  hStream = Open String For Write ' Dynamisch wachsender Puffer
  Write #hStream, vValue As Variant
  sBinary = Close #hStream
  Return Base64$(sBinary) ' Newline-freie Kodierung
End ' Encode(..)

Public Sub Send(vValue As Variant)
  Print Encode(vValue)
End ' Send(..)

Public Function Decode(sCode As String) As Variant
  Dim sBinary As String
  Dim hStream As Stream
  Dim vValue As Variant

  sBinary = UnBase64$(sCode)
  hStream = Open String sBinary For Read
  vValue = Read #hStream As Variant
  Return vValue

End ' Decode(..)
```

Die Mächtigkeit des Moduls erschließt sich Ihnen nur dann, wenn Sie das Projekt 4 intensiv erproben und in der Task-Klasse Daten mit unterschiedlichem Datentyp – zum Beispiel String, Boolean, Float oder Collection – an den übergeordneten Prozess schicken.

```
' Gambas class file

Inherits Task

Public Sub Main()

  Do
    TaskIPC.Send(22 / 7)
    TaskIPC.Send(["Now": Now(), "timer": Timer()])
    TaskIPC.Send(["Key": "stringvalue", "log2": Log(2), "YD": DateAdd(Now, gb.Day, -1), "User": User.Name])
    TaskIPC.Send([Pi(3), (1001 / 1000) ^ 1000, 10, (355 / 113) - Pi])
    TaskIPC.Send(["Produkt = ": 6.66 * 34.78])
    TaskIPC.Send(6 < Day(Now()))
    TaskIPC.Send("Task - Hintergrund-Prozess")
    Wait 2
  Loop

End
```

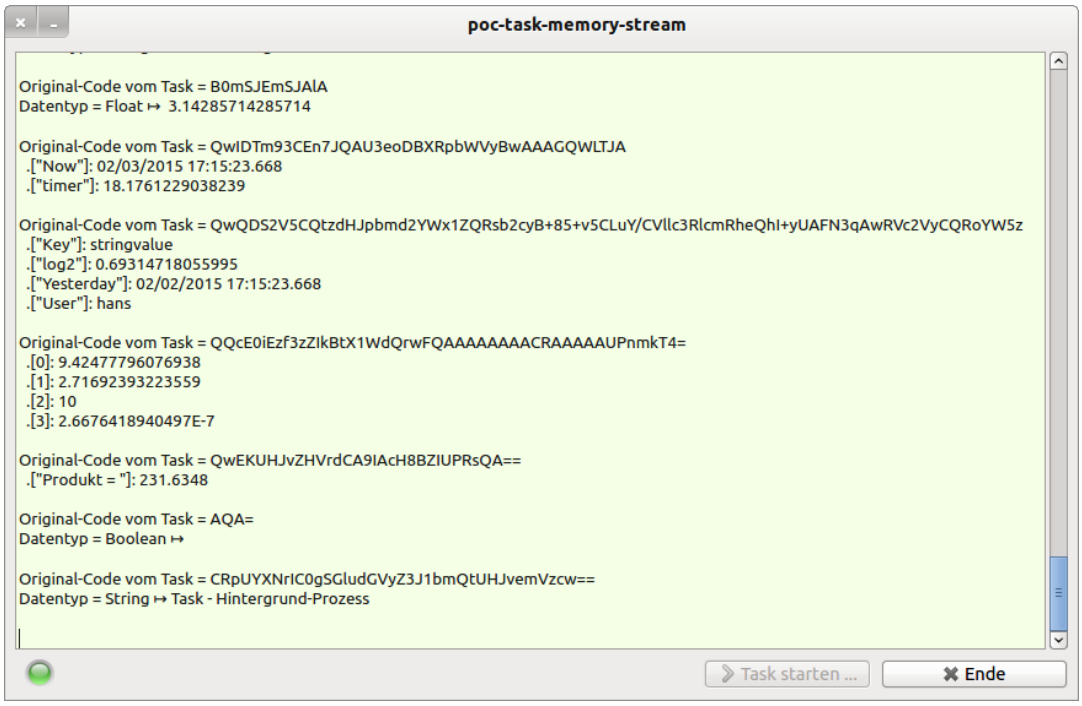

Abbildung 20.6.2.2.1: Demonstrationsprogramm

Einige Daten ändern sich periodisch (Zeit-Werte), während andere sich nicht verändern – wie die Zeichenkette 'Task – Hintergrund-Prozess'.