```
 1  using System;
 2  using System.ComponentModel;
 3  using System.Threading;
 4  using System.Windows.Forms;
 5  using System.Diagnostics; // hinzugefügt
 6
 7
 8
 9  namespace CPU
10      {
11      public partial class Form1 : Form
12      {
13
14          PerformanceCounter prozessorPerformance;
15          bool isRunning;
16
17          public Form1()
18          {
19              InitializeComponent();
20
21              InitializeBackgroundWorker(); // Einschalten
22              prozessorPerformance = new PerformanceCounter("Processor", "%
                  Processor Time", "_Total"); // Prozessorwerte entgegennehmen
23              isRunning = false;
24              this.Text = "CPU Auslastung"; // Text im Formkopf
25
26          }
27          #region Fensterposition und Größe
28          private void FenstergroesseErmitteln() // ermittelt die Fenstergröße
29          {
30              int BildschrimWeite =
                  System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width - 219; //
                   Fenstergröße minus Formgröße
31              int BildschirmHöhe =
                  System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height -
                  168; // Fenstergröße minus Formgröße
32              lblBildschrimgrösse.Text = "max. Fensterposition X: " +
                  BildschrimWeite.ToString() +" und Y: " + BildschirmHöhe.ToString
                  (); // in label die maximalen Werte anzeigen die eingegeben
                  werden sollten
33
34          }
35          private void Fensterpostition(object sender, EventArgs e) //
              Fensterposition
36          {
37              FenstergroesseErmitteln();
38          }
39          #endregion
40          #region Zubehör ohne Code
41          private void progressBar1_Click(object sender, EventArgs e)
42          {
43          }
44          private void toolStripLabel1_Click(object sender, EventArgs e)
45          {
46          }
47          private void toolStripButton1_Click(object sender, EventArgs e) //
```

```
            Hilfebox
48          {
49              AboutBox1 MyAboutBox = new AboutBox1();
50              MyAboutBox.ShowDialog();
51          }
52          private void textBox1KoordinateX_TextChanged(object sender, EventArgs
              e)
53          {
54          }
55          private void textBox2KoordinateY_TextChanged(object sender, EventArgs
              e)
56          {
57          }
58          private void lblBildschrimgrösse_Click(object sender, EventArgs e)
59          {
60          }
61          private void checkBox1Start_CheckedChanged(object sender, EventArgs e)
62          {
63          }
64          private void checkBox2PositionMerken_CheckedChanged(object sender,
              EventArgs e)
65          {
66          }
67
68  #endregion
69
70          private void InitializeBackgroundWorker() // Initialisierung
71          {
72              backgroundWorker1CPU.DoWork += new DoWorkEventHandler
                  (backgroundWorker1CPU_DoWork);
73              backgroundWorker1CPU.ProgressChanged += new
                  ProgressChangedEventHandler
                  (backgroundWorker1CPU_ProgressChanged);
74              backgroundWorker1CPU.WorkerReportsProgress = true;
75              backgroundWorker1CPU.WorkerSupportsCancellation = true;
76          }
77          private void btn_CPU_Start_Click(object sender, EventArgs e) //
              starten
78          {
79              if (checkBox1Start.Checked)
80              {
81                  backgroundWorker1CPU.RunWorkerAsync(); // Asyncron starten
82                  btn_CPU_Start.Enabled = false;  // Button auf inaktiv setzen
83                  btn_CPU_Stop.Enabled = true;    // Button auf aktiv setzen
84                  isRunning = true;
85              }
86              else
87              {
88                  btn_CPU_Start.Enabled = true;  // Button auf inaktiv setzen
89                  btn_CPU_Stop.Enabled = false;    // Button auf aktiv setzen
90                  isRunning = true;
91              }
92          }
93          private void backgroundWorker1CPU_DoWork(object sender,
              DoWorkEventArgs e) // wenn er arbeitet hole die Werte und speichere
              sie in e
```

```
94          {
95              BackgroundWorker worker = sender as BackgroundWorker;
96              consumingMethod(worker, e);
97          }
98          private void consumingMethod(BackgroundWorker worker, DoWorkEventArgs
              e)
99          {
100             if (worker.CancellationPending == true)
101             return;
102
103             while (isRunning)
104             {
105             worker.ReportProgress((int)prozessorPerformance.NextValue());
106             Thread.Sleep(1000);
107
108             }
109         }
110         private void backgroundWorker1CPU_ProgressChanged(object sender,
              ProgressChangedEventArgs e) // Progressbar zeichnen
111         {
112             progressBar1.Value = e.ProgressPercentage; // wert für die
                Progressbar von e holen
113             toolStripLabel1.Text = "Auslastung: " +
                e.ProgressPercentage.ToString() + "%"; // Auslastung in den
                toolstrip schreiben
114         }
115         private void backgroundWorker1CPU_RunWorkerCompleted(object sender,
              RunWorkerCompletedEventArgs e)
116         {
117         }
118         private void btn_CPU_Stop_Click(object sender, EventArgs e) // stoppen
119         {
120             backgroundWorker1CPU.CancelAsync(); // wichtiger befehl zum
                stoppen
121             btn_CPU_Stop.Enabled = false;
122             btn_CPU_Start.Enabled = true;
123             isRunning = false; // worker ist aus
124
125
126         }
127         private void Form1_FormClosed(object sender, FormClosedEventArgs e)
128         {
129             Properties.Settings.Default.checkBox1Start =
                checkBox1Start.Checked;
130             Properties.Settings.Default.checkBox2PositionMerken =
                checkBox2PositionMerken.Checked;
131             // während des schließens den Status der Checkbox speichern
132
133
134             if (this.WindowState == FormWindowState.Normal)
135                 {
136
137                 Properties.Settings.Default.Fensterposition =
                    this.Location; // beim schließen die letzte Fensterposition
                    merken
138                 }
```

```
139             else
140                 {
141                 Properties.Settings.Default.Fensterposition =
                      this.RestoreBounds.Location;
142                 }
143
144
145             Properties.Settings.Default.Save();
146
147             }// beim schließen der Form die persönliche Einstellungen
                  abspeichern
148         private void Form1_Load(object sender, EventArgs e)
149         {
150             if (checkBox1Start.Checked) // wenn checked ist dann
151                 {
152                 checkBox1Start.Checked =
                      Properties.Settings.Default.checkBox1Start; // vor dem
                      öffnen den gespeicherten status der checkbox holn
153                 }
154
155             if (checkBox2PositionMerken.Checked)
156                 {
157                 checkBox2PositionMerken.Checked =
                      Properties.Settings.Default.checkBox2PositionMerken; // vor
                      dem öffnen den gespeicherten status der checkbox holn
158                 this.Location =
                      Properties.Settings.Default.Fensterposition; // gespeicherte
                       letzte Fensterposition holen
159                 }
160
161             }// beim laden der Form die persönliche Einstellungen holen
162
163
164     }
165 }
166
```