```
 1 /* File: ScaleDriverSoehnle3010.cs
 2  * Proj: ESAprocess.Driver.Scale.Soehnle3010
 3  * Date: 19.02.2021 12:18:20
 4  * Desc: ScaleDriverSoehnle3010 - TCP-Waagentreiber für Soehnle 3010.
 5  * Elem: CLASS ScaleDriverSoehnle3010 - -"-
 6  * Auth: © ESA GmbH 2014-2021 */
 7
 8 using ESAprocess.Shared.Log;
 9 using System;
10 using System.Diagnostics;
11 using System.Threading;
12 using System.Threading.Tasks;
13
14 namespace ESAprocess.Driver.Scale.Soehnle3010
15 {
16     #region CLASS ScaleDriverSoehnle3010
17
18     /// <summary>
19     /// TCP-Waagentreiber für Soehnle 3010.
20     /// </summary>
21     public sealed class ScaleDriverSoehnle3010 :
         ScaleDriverNetworkBase, IScaleDriver
22     {
23         #region CONSTANTS
24
25         const int m_iDEFAULT_TIME_OUT = 250;
26
27         #endregion // CONSTANTS
28
29         #region CONSTRUCTION & INITIALIZATION
30
31         /// <summary>
32         /// Erzeugt und initialisiert eine neue Instanz der <see
             cref="ScaleDriverSoehnle3010"/> Klasse.
33         /// </summary>
34         public ScaleDriverSoehnle3010(Guid guidID,
             IScaleConnectionNetwork scaleConnectionNetwork, IESALogger
             esaLogger, ScaleConfiguration? scaleConfiguration = null, int
              iObserverCycle = m_iDEFAULT_OBSERVATION_CYCLE)
35             : base(guidID, scaleConnectionNetwork, esaLogger,
                 scaleConfiguration, iObserverCycle)
36         {
37         }
38
39         #endregion // CONSTRUCTION & INITIALIZATION
40
41         #region PROPERTIES
42
43         /// <summary>
44         /// Gibt an ob die Verbindung zur Waage geöffnet ist.
45         /// </summary>
46         public bool IsConnected
47         {
```

```
48              get => m_scaleConnectionNetwork.ConnectionState ==
                  ScaleConnectionState.Opened;
49          }
50
51          #endregion // PROPERTIES
52
53          #region PUBLIC METHODS
54
55
56          /// <inheritdoc/>
57          public bool IsTared()
58          {
59              ScaleWeighingResult resultNetWeight = ReadGrossWeight();
60              return resultNetWeight.Weight == 0m;
61          }
62          /// <summary>
63          /// Tarieren der Waage
64          /// </summary>
65          /// <returns>true, wenn Befehl ausgeführt worden ist, false bei
               Fehler</returns>
66          public bool SetTarePoint()
67          {
68              if (IsTared())
69              {
70                  return true;
71              }
72
73              ScaleWeighingResult resultNetWeight = ReadNetWeight();
74              if (resultNetWeight.Weight <= 0)
75              {
76                  ScaleWeighingResult resultGrossWeight = ReadGrossWeight
                      ();
77                  if (resultGrossWeight.Weight <= 0)
78                      return SetZeroPoint();
79              }
80              //Sende Tara-Befehl
81              string cmd = "<t>";
82
83              //Sende Tara-Befehl
84              //Antwort von der Waage: keine, entgegen der Dokumentation
                  sendet die Testwaage kein ACK.
85              //Daher wird die Ausführung der Tarierung mittels Kontrolle
                   des Nettogewichts überprüft
86              var result = m_scaleConnectionNetwork.Send(cmd.ToCharArray
                  (), m_scaleConfiguration?.CommunicationTimeout ??
                  m_iDEFAULT_TIME_OUT);
87              //Prüfen ob der Befehl erfolgreich versendet worden ist
88              if (result == null || !result.Success)
89              {
90                  throw new InvalidOperationException("Fehler beim
                      Tarieren");
91              }
92
```

```
93              //Warte Ausführung der Tarierung
94              var t = Task.Run(async () => await Task.Delay(250));
95              t.Wait();
96
97              //zur Prüfung ob die Tarierung ausgeführt worden ist, wird
                  das Nettogewicht gelesen
98              decimal dValue = ReadNetWeight().Weight;
99              if (dValue == 0m)
100             {
101                 return true;
102             }
103
104             return false;
105         }
106
107         /// <summary>
108         /// Sendet einen Nullstellen-Befehl an die Waage
109         /// </summary>
110         /// <returns>true, wenn Befehl ausgeführt worden ist, false bei
              Fehler</returns>
111         public bool SetZeroPoint()
112         {
113             string cmd = "<Q007Q>";
114
115             //Sende Nullstellen-Befehl, es wird auf eine Antwort
                  gewartet welche mit 'Q,1>' endet.
116             var result = m_scaleConnectionNetwork.Request
                  (cmd.ToCharArray(), "Q,1>".ToCharArray(),
                  m_scaleConfiguration?.CommunicationTimeout ??
                  m_iDEFAULT_TIME_OUT);
117             if (result == null || !result.Success)
118             {
119                 throw new InvalidOperationException("Fehler beim
                      Nullstellen der Waage");
120             }
121
122             //Verbindung neu aufbauen
123             m_scaleConnectionNetwork.Close();
124             m_scaleConnectionNetwork.Open();
125
126             return true;
127         }
128
129         /// <summary>
130         /// BruttoAnzeige aktivieren
131         /// </summary>
132         public void SwitchToGrossView()
133         {
134             throw new NotSupportedException($"Die Aktion '{nameof
                  (SwitchToGrossView)}' wird bei diesem Treiber ('{GetType
                  ().Name}') nicht untersützt!");
135         }
136
```

```
137         /// <summary>
138         /// NettoAnzeige aktivieren
139         /// </summary>
140         public void SwitchToNettoView()
141         {
142             throw new NotSupportedException($"Die Aktion '{nameof
                (SwitchToNettoView)}' wird bei diesem Treiber ('{GetType
                ().Name}') nicht untersützt!");
143         }
144
145         /// <summary>
146         /// Sperren/Entsperren der (Waagen-)Tastatur
147         /// </summary>
148         /// <param name="LockState"></param>
149         public bool SetKeyboardLock(bool LockState)
150         {
151             throw new NotSupportedException($"Die Aktion '{nameof
                (SetKeyboardLock)}' wird bei diesem Treiber ('{GetType
                ().Name}') nicht untersützt!");
152         }
153
154         /// <summary>
155         /// Triggert einen Registrierungs-Vorgang
156         /// </summary>
157         public ScaleRegisterResult RegisterWeight()
158         {
159             throw new NotSupportedException($"Die Aktion '{nameof
                (RegisterWeight)}' wird bei diesem Treiber ('{GetType
                ().Name}') nicht untersützt!");
160         }
161
162         /// <summary>
163         /// Liest Brutto-Wert der Waage
164         /// </summary>
165         public ScaleWeighingResult ReadGrossWeight()
166         {
167             string cmd = "<A>";
168
169             //synchroner Aufruf...
170             NetworkCommunicationResult? result =
                m_scaleConnectionNetwork.Request(cmd.ToCharArray(),
                Environment.NewLine.ToCharArray(),
                m_scaleConfiguration?.CommunicationTimeout ??
                m_iDEFAULT_TIME_OUT);
171
172             //Verwendung der async Methode...
173             //Task<ProtocolCommResult> commTask = ((ScaleConnectionTcp)
                Connection).RequestAsync(cmd.ToCharArray(), 5000,
                Environment.NewLine.ToCharArray());
174
175             //Warte auf Ausführung....
176             //commTask.Wait();
177
```

```
178             //if (commTask.IsCompleted && (!commTask.IsFaulted))
179             //{
180             if (result?.Response != null && result.Success/
                  *commTask.Result.Success*/)
181             {
182                 //Daten wurden empfangen
183                 string response = result.Response;
184
185                 //Auswertung von ev Fehler-Codes der Waage
186
187                 //Auswertung der Empfangsdaten
188                 //z.B.: '001001N      0,130 kg G      0,130 kg
                      <CR><LF>'
189
190                 int iPosGross = response.IndexOf('G');
191                 if (iPosGross < 0)
192                 {
193                     throw new InvalidOperationException("Kein
                      Bruttowert in den Empfangsdaten");
194                 }
195                 //Ausgabe Belastungs-Status
196                 ScaleLoadState loadState;
197                 decimal dValue;
198                 // Bei Unter bzw. Überlast muss der Wagenwert nicht
                      aufbereitet werden da die Waage entweder ______ oder
                      ^^^^^^ anzeigt.
199                 if (response[0] == '1')
200                 {
201                     loadState = ScaleLoadState.Underload;
202                     dValue = decimal.MinValue;
203                 }
204                 else if (response[1] == '1')
205                 {
206                     loadState = ScaleLoadState.Overload;
207                     dValue = decimal.MaxValue;
208                 }
209                 else
210                 {
211                     loadState = ScaleLoadState.Operational;
212                     //Nettowert decodieren und Dezimalseparator
                      anpassen
213                     string strGrossWeight = response.Substring
                      (iPosGross + 1, 11).Replace(",",
                      Thread.CurrentThread.CurrentCulture.NumberFormat.Numb
                      erDecimalSeparator).Trim();
214                     if (!decimal.TryParse(strGrossWeight, out dValue))
215                         throw new InvalidCastException($"Der Wert
                      '{strGrossWeight}' konnte nicht in den Decimal Wert
                      geparst werden! Antwort '{response}'!");
216                 }
217                 //Bruttowert decodieren und Dezimalseparator anpassen
218
219                 //ScaleUnit = response.Substring(iPosGross +
```

```
                    12).TrimEnd(Environment.NewLine.ToCharArray()).Trim
                    ();
220
221                 //Waagenstillstand
222                 bool bIsStagnating = (response[2] == '1');
223
224                 //Leermeldung
225                 bool bIsEmpty = (response[3] == '1');
226
227
228                 return new ScaleWeighingResult(FormatScaleValue
                    (dValue), bIsStagnating, loadState);
229
230             }
231             else
232             {
233                 if (result?.Response != null)
234                     throw new InvalidOperationException
                    (result.Response.ToString());
235                 else
236                     throw new InvalidOperationException("Kommando-
                    Antwort konnte nicht empfangen werden.");
237             }
238         }
239
240         /// <summary>
241         /// Liest Netto-Wert der Waage
242         /// </summary>
243         public ScaleWeighingResult ReadNetWeight()
244         {
245             string cmd = "<A>";
246
247             var result = m_scaleConnectionNetwork.Request
                  (cmd.ToCharArray(), Environment.NewLine.ToCharArray(),
                  m_scaleConfiguration?.CommunicationTimeout ??
                  m_iDEFAULT_TIME_OUT);
248
249             if (result?.Response != null && result.Success)
250             {
251                 //Auswertung von ev Fehler-Codes der Waage
252                 string response = result.Response;
253                 int ResponseLen = response.Length;
254
255                 //Auswertung der Empfangsdaten
256                 //z.B.:'001001N      0,125 kg G      0,125 kg <CR><LF>'
257
258                 int iPosNet = response.IndexOf('N');
259                 if (iPosNet < 0)
260                 {
261                     throw new InvalidOperationException("Kein Nettowert
                    in den Empfangsdaten");
262                 }
263
```

```
264                 //Ausgabe Belastungs-Status
265                 ScaleLoadState loadState;
266                 decimal dValue;
267                 // Bei Unter bzw. Überlast muss der Wagenwert nicht
                      aufbereitet werden da die Waage entweder ______ oder
                      ^^^^^^ anzeigt.
268                 if (response[0] == '1')
269                 {
270                     loadState = ScaleLoadState.Underload;
271                     dValue = decimal.MinValue;
272                 }
273                 else if (response[1] == '1')
274                 {
275                     loadState = ScaleLoadState.Overload;
276                     dValue = decimal.MaxValue;
277                 }
278                 else
279                 {
280                     loadState = ScaleLoadState.Operational;
281                     //Nettowert decodieren und Dezimalseparator
                      anpassen
282                     string strNetWeight = response.Substring(iPosNet +
                      1, 11).Replace(",",
                      Thread.CurrentThread.CurrentCulture.NumberFormat.Numb
                      erDecimalSeparator).Trim();
283                     if (!decimal.TryParse(strNetWeight, out dValue))
284                         throw new InvalidCastException($"Der Wert
                      '{strNetWeight}' konnte nicht in den Decimal Wert
                      geparst werden! Antwort '{response}'!");
285                 }
286
287
288                 //ScaleUnit = response.Substring(iPosNet + 12,
                      3).TrimEnd(Environment.NewLine.ToCharArray()).Trim();
289
290                 //Waagenstillstand
291                 bool bIsStagnating = (response[2] == '1');
292
293                 //Leermeldung
294                 bool bIsEmpty = (response[3] == '1');
295
296                 return new ScaleWeighingResult(FormatScaleValue
                      (dValue), bIsStagnating, loadState);
297             }
298             else
299             {
300                 if (result?.Response != null)
301                     throw new InvalidOperationException
                      (result.Response.ToString());
302                 else
303                     throw new InvalidOperationException("Kommando-
                      Antwort konnte nicht empfangen werden.");
304             }
```

```
305         }
306
307         /// <inheritdoc/>
308         public override ScaleWeight ReadCurrentScaleWeight()
309         {
310             //Stopwatch sw = new Stopwatch();
311             //sw.Start();
312             var netto = ReadNetWeight();
313             var gross = ReadGrossWeight();
314             //var tara = ReadTareWeight();
315             decimal dTara = gross.Weight - netto.Weight;
316             //sw.Stop();
317             //Console.WriteLine($"{DateTime.Now:HH:mm:ss.fff} Brutto
                  und Nettowert in {sw.ElapsedMilliseconds}ms abgerufen");
318             return new ScaleWeight(gross.Weight, netto.Weight, dTara,
                  gross.IsStagnating && netto.IsStagnating,
                  gross.LoadState);
319         }
320
321         #endregion // PUBLIC METHODS
322     }
323
324     #endregion // CLASS ScaleDriverSoehnle3010
325 }
326
327
```