```
/* File: ScaleDriverBizerbaIS30.cs
 * Proj: ESAprocess.Driver.Scale.BizerbaIS30
 * Date: 09.03.2021 11:06:20
 * Desc: ScaleDriverBizerbaIS30 - TCP-Waagentreiber für Bizerba iS30
 * Elem: CLASS  ScaleDriverBizerbaIS30 - -"-
 * Auth: © ESA GmbH 2014-2021 */


using ESAprocess.Shared.Log;
using System;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ESAprocess.Driver.Scale.BizerbaIS30
{
    #region CLASS ScaleDriverBizerbaIS30

    /// <summary>
    /// TCP-Waagentreiber für Bizerba iS30
    /// </summary>
    public class ScaleDriverBizerbaIS30 : ScaleDriverNetworkBase,
      IScaleDriver
    {
        #region FIELDS

        char m_cSTX = (char)02;
        char m_cETX = (char)03;

        decimal m_dEpsilon;
        decimal m_dCurrentTareWeight = 0;

        #endregion // FIELDS

        #region CONSTRUCTION & INITIALIZATION

        /// <summary>
        /// Erzeugt und initialisiert eine neue Instanz der <see
          cref="ScaleDriverBizerbaIS30"/> Klasse.
        /// </summary>
        /// <param name="guidID"></param>
        /// <param name="scaleConnectionNetwork"></param>
        /// <param name="esaLogger"></param>
        /// <param name="scaleConfiguration"></param>
        /// <param name="iObserverCycle"></param>
        /// <param name="dEpsilon">(Optional) Epsilon für den Vergleich
           mit 0</param>
        public ScaleDriverBizerbaIS30(Guid guidID,
          IScaleConnectionNetwork scaleConnectionNetwork, IESALogger
          esaLogger, ScaleConfiguration? scaleConfiguration = null, int
           iObserverCycle = m_iDEFAULT_OBSERVATION_CYCLE, decimal
```

```
            dEpsilon = 0.00001m)
48              : base(guidID, scaleConnectionNetwork, esaLogger,
                  scaleConfiguration, iObserverCycle)
49          {
50              m_dEpsilon = dEpsilon;
51          }
52
53          #endregion // CONSTRUCTION & INITIALIZATION
54
55
56          #region PRIVATE METHODS
57
58          /// <summary>
59          /// Gibt <see langword="true"/> zurück wenn der Betrag des
              übergebenen Werts kleiner als Epsilon ist.
60          /// </summary>
61          /// <param name="dValue"></param>
62          /// <returns></returns>
63          bool IsZero(decimal dValue)
64          {
65              return Math.Abs(dValue) < m_dEpsilon;
66          }
67
68          /// <summary>
69          /// Holt den Waagenwert und decodiert diesen.
70          /// </summary>
71          /// <param name="bNetWeightRequested"></param>
72          /// <returns></returns>
73          ScaleWeighingResult GetAndDecodeValue(bool bNetWeightRequested)
74          {
75              // Befehl zum Auslesen der Gewichtswerte
76
77              string cmd = m_cSTX + "________" + m_cETX;
78
79              var result = m_scaleConnectionNetwork.Send(cmd.ToCharArray
                  ());
80
81              //Prüfe auf allgemeinen Fehler
82              if (result?.Response == null || !result.Success)
83                  throw new InvalidOperationException("Fehler beim
                      Auslesen des Gewichtswerts.");
84              string strResponseMessage = string.Join(", ",
                  result.Response.Select(c => ((int)c).ToString()));
85              // BSP:  <#02>-,)   0089,0kg<#03>
86              //           "-" -> negativ, "" -> positiv
87              string strData = result.Response;
88              int iIndecACK = strData.IndexOf(m_cSTX);
89              int iIndexCR = strData.IndexOf(m_cETX);
90
91              if (iIndecACK < 0 || iIndexCR < 0 || iIndecACK > iIndexCR)
92                  throw new InvalidOperationException($"Fehler beim
                      Decodieren des Antworttelegramms! Länge:
                      '{strData.Length}' IndexACK: '{iIndecACK}' IndexCR:
```

```
                    '{iIndexCR}' Antwort: '{strResponseMessage}'");
 93
 94             strData = strData.Substring(iIndecACK + 1);
 95             strData = strData.Substring(0, iIndexCR - 1);
 96
 97
 98
 99             bool bIsNetWeight;
100             char strNetIdentifier = strData[0]; // Nach ACK kommt die
                  Gewichtskennung ('+' -> Brutto, ',' -> Netto)
101             if (strNetIdentifier == ',')
102                 bIsNetWeight = true;
103             else if (strNetIdentifier == '+')
104                 bIsNetWeight = false;
105             else
106                 throw new InvalidOperationException($"Unbekannte
                      Kennung des Gewichtswerts ({strNetIdentifier}).
                      Antwort: '{strResponseMessage}'");
107
108             strData = strData.Substring(1);
109
110             char cStatus = strData[0];  // Nach der Kennung kommt eine
                  Statuszeichen (Doku S. 141)
111             byte charByte = Encoding.ASCII.GetBytes(new[] { cStatus })
                  [0];
112             bool bIsStagnating = (charByte & 0b00000001) != 0;  // Das
                  erste Bit bestimmt, ob der Waagenwert beruhigt ist.
113             bool bUnderload = (charByte & 0b00000010) != 0;     // Das
                  zweite Bit bestimmt, ob ein Übergewicht vorliegt.
114             bool bOverload = (charByte & 0b00000100) != 0;      // Das
                  dritte Bit bestimmt, ob ein Untergewicht vorliegt.
115
116             strData = strData.Substring(1);
117
118             bool bIsNegative = false;
119             if (strData[0] == '-')
120             {
121                 bIsNegative = true;
122                 strData = strData.Substring(1);
123             }
124             strData = strData.Substring(1);
125
126             strData = strData.Trim();
127             int i;
128             for (i = strData.Length - 1; i >= 0; i--)
129             {
130                 if (int.TryParse(strData[i].ToString(), out int iTemp))
131                     break;
132             }
133             strData = strData.Substring(0, i + 1);
134             if (!Decimal.TryParse(strData, out decimal dValue))
135                 throw new InvalidOperationException($"Fehler bei der
                      Konvertierung von \"{strData}\" in decimal. Antwort:
```

```
                    '{strResponseMessage}'");

            if (bIsNegative)
                dValue *= -1;

            if (bNetWeightRequested)
            {
                if (!bIsNetWeight)
                    dValue -= m_dCurrentTareWeight;
            }
            else
            {
                if (bIsNetWeight)
                    dValue += m_dCurrentTareWeight;
            }

            ScaleLoadState scaleLoadState;
            if (bOverload)
                scaleLoadState = ScaleLoadState.Overload;
            else if (bUnderload)
                scaleLoadState = ScaleLoadState.Underload;
            else
                scaleLoadState = ScaleLoadState.Operational;

            return new ScaleWeighingResult(FormatScaleValue(dValue),
              bIsStagnating, scaleLoadState);
        }

        #endregion // PRIVATE METHODS

        #region PUBLIC METHODS

        #region INTERFACE IScaleDriver

        /// <summary>
        /// Holt den Waagenwert
        /// </summary>
        /// <returns></returns>
        public override ScaleWeight ReadCurrentScaleWeight()
        {
            var netWeight = GetAndDecodeValue(true);

            return new ScaleWeight(netWeight.Weight +
              m_dCurrentTareWeight, netWeight.Weight,
              m_dCurrentTareWeight, netWeight.IsStagnating,
              netWeight.LoadState);
        }

        /// <summary>
        /// Holt das Bruttogewicht
        /// </summary>
        /// <returns></returns>
        public ScaleWeighingResult ReadGrossWeight()
```

```
184         {
185             return GetAndDecodeValue(false);
186         }
187 
188         /// <summary>
189         /// Holt das Nettogewicht
190         /// </summary>
191         /// <returns></returns>
192         public ScaleWeighingResult ReadNetWeight()
193         {
194             return GetAndDecodeValue(true);
195         }
196 
197         /// <summary>
198         /// Triggert einen Registrierungs-Vorgang
199         /// </summary>
200         public ScaleRegisterResult RegisterWeight()
201         {
202             throw new NotSupportedException($"Die Aktion '{nameof
                  (RegisterWeight)}' wird bei diesem Treiber ('{GetType
                  ().Name}') nicht untersützt!");
203         }
204 
205         /// <summary>
206         /// Sperrt/Entsperrt das Keyboard
207         /// </summary>
208         /// <param name="bLockState"></param>
209         /// <returns></returns>
210         public bool SetKeyboardLock(bool bLockState)
211         {
212             // Befehl zum Sperren der Wiegeterminal Tastatur
213 
214             string cmd = m_cSTX + "q" + (bLockState ? '6' : '5') +
                  m_cETX;
215 
216             var result = m_scaleConnectionNetwork.Send(cmd.ToCharArray
                  ());
217 
218             //Prüfe auf allgemeinen Fehler
219             if (result == null || !result.Success)
220                 throw new InvalidOperationException("Fehler beim
                      Tarieren");
221 
222             return true;
223         }
224 
225         /// <inheritdoc/>
226         public bool IsTared()
227         {
228             // Wenn Nettogewicht = 0 -> keine Tarierung nötig
229             return IsZero(GetAndDecodeValue(true).Weight);
230         }
231 
```

```
232         /// <summary>
233         /// Tariert die Waage
234         /// </summary>
235         /// <returns></returns>
236         public bool SetTarePoint()
237         {
238             // Wenn Nettogewicht = 0 -> keine Tarierung nötig
239             if (IsTared())
240                 return true;
241 
242             // Befehl zum Tarieren
243 
244             string cmd = m_cSTX + "________" + m_cETX;
245 
246             var result = m_scaleConnectionNetwork.Send(cmd.ToCharArray
                  ());
247 
248             if (result == null || !result.Success)
249                 throw new InvalidOperationException("Fehler beim
                      Tarieren");
250 
251             // Prüfen ob Nettogewicht nach dem Tarieren = 0 ist
252             if (IsZero(GetAndDecodeValue(true).Weight))
253                 return true;
254 
255             return false;
256         }
257 
258         /// <summary>
259         /// Stellt die Waage auf Null
260         /// </summary>
261         /// <returns></returns>
262         public bool SetZeroPoint()
263         {
264             // Befehl für Nullstellen
265 
266             string cmd = m_cSTX + "________" + m_cETX;
267 
268 
269             var result = m_scaleConnectionNetwork.Send(cmd.ToCharArray
                  ());
270             //Prüfe auf allgemeinen Fehler
271             if (result == null || !result.Success)
272                 throw new InvalidOperationException("Fehler beim
                      Nullstellen der Waage");
273 
274             // Beim Nullstellen wird auf das Bruttogewicht geprüft
275             if (!IsZero(GetAndDecodeValue(false).Weight))
276                 return false;
277 
278             return true;
279         }
280 
```

```
281         /// <summary>
282         /// BruttoAnzeige aktivieren
283         /// </summary>
284         public void SwitchToGrossView()
285         {
286             throw new NotSupportedException($"Die Aktion '{nameof
                (SwitchToGrossView)}' wird bei diesem Treiber ('{GetType
                ().Name}') nicht untersützt!");
287         }
288
289         /// <summary>
290         /// NettoAnzeige aktivieren
291         /// </summary>
292         public void SwitchToNettoView()
293         {
294             throw new NotSupportedException($"Die Aktion '{nameof
                (SwitchToNettoView)}' wird bei diesem Treiber ('{GetType
                ().Name}') nicht untersützt!");
295         }
296
297         #endregion // INTERFACE IScaleDriver
298
299         #endregion //  PUBLIC METHODS
300     }
301
302     #endregion // CLASS ScaleDriverBizerbaIS30
303 }
304
```