AN701.65-117

für

LI-Ion Ladeschaltung Nμ701.65

Application Note 117:

Programmierbeispiel für National Instruments LabView

Rev. 0.1

Inhaltsverzeichnis

1 Zweck

Der Nµ701.65 lässt sich über eine serielle Schnittstelle prüfen. (Siehe AN701.65-115) Dazu ist der Zugriff auf interne Parameter und Funktionen vorgesehen. Die implementierten Funktionen und Befehle können z. B. für den Baugruppentest über einen externen Controller genutzt werden. Ebenfalls kann die Schnittstelle in der Qualitätssicherung und im Service zur Fehleranalyse genutzt werden.
Es folgt ein Programmierbeispiel, in dem ein National Instruments DAQ-Pad mit dem Nµ701.65 kommuniziert.
Ein weiteres Programmierbeispiel für ANSI-C ist zu finden im AN701.65-116.

1.1 Vorbemerkungen

Während der Kommunikation agiert der Nµ701.65 als Slave, das DAQ-Pad agiert als Master und treibt das Clock-Signal SCLK. Das SD-Signal ist bidirektional.

2 Programmierbeispiel für LabView

2.1 Allgemeine Definitionen:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
using NationalInstruments;
using NationalInstruments.DAQmx;
using Neutron.NuCsLib;
using System.Windows.Forms;
private static int InstanceCount = 0;

private DAQPad daq;
private bool ChannelsDefined = false;
private Task tsDigOut;                // Gleichzeitiges Schreiben aller Leitungen
private Task tsDigOut_SCLK_RW_Trig;   // Gleichzeitiges Schreiben von SCLK, RW, Trig
private Task tsDigOut_SCLK_SD;        // Gleichzeitiges Schreiben von SCLK und SD
private Task tsDigOut_SD;             // Schreiben von SD
private Task tsDigIn_SD;              // Lesen von SD
private Task tsAnaOut_EE;             // Schreiben der Rampe für EE
private DigitalSingleChannelWriter writerDig;
private DigitalSingleChannelWriter writerDig_SCLK_RW_Trig;
private DigitalSingleChannelWriter writerDig_SCLK_SD;
private DigitalSingleChannelWriter writerDig_SD;
private DigitalSingleChannelReader readerDig_SD;
private AnalogSingleChannelWriter writerAna_EE;

public enum pSCLK { Low = 0, Heigh = 1 }
public enum pSD { Low = 0, Heigh = 1 }
public enum pTrig { Low = 0, Heigh = 1 }
public enum pRW { Low = 0, Heigh = 1 }
```

2.2 Standalone-Betrieb aktivieren

2.3 Schreiben vom Microcontroller auf den Nμ701.65

2.3.1 Schreiben eines Bits (0 / 1 / Startbit)

```
public void SetLine(pSCLK sclk, pSD sd, pRW rw, pTrig tr)
    {
       // Beim ersten Aufruf werden die Tasks erzeugt
       if (!ChannelsDefined)
          CreateChannelsAndTasks();

       //writerDig.WriteSingleSampleMultiLine(true, new bool[,] { { tb(sclk) }, { tb(sd) }, { tb(rw) }, { tb(tr) } });

       if( this.InvertDataSignals )
          writerDig.WriteSingleSampleMultiLine(true, new bool[] { !tb(sclk), !tb(sd), tb(rw), tb(tr) });
       else
          writerDig.WriteSingleSampleMultiLine(true, new bool[] { tb(sclk), tb(sd), tb(rw), tb(tr) });
    }

    /// <summary>
    /// Setzt die SCLK-, und SD-Leitungen synchron auf die angegebenen Werte
    /// </summary>
    public void SetLine(pSCLK sclk, pSD sd)
    {
       // Beim ersten Aufruf werden die Tasks erzeugt
       if (!ChannelsDefined)
          CreateChannelsAndTasks();

       //writerDig_SCLK_SD.WriteSingleSampleMultiLine(true, new bool[,] { { tb(sclk) }, { tb(sd) } });
       if (this.InvertDataSignals)
          writerDig_SCLK_SD.WriteSingleSampleMultiLine(true, new bool[] { !tb(sclk), !tb(sd) });
```

```
        else
            writerDig_SCLK_SD.WriteSingleSampleMultiLine(true, new bool[] { tb(sclk), tb(sd) });
    }

    /// <summary>
    /// Setzt die SCLK-, RW- und Tr-Leitungen synchron auf die angegebenen Werte
    /// </summary>
    public void SetLine(pSCLK sclk, pRW rw, pTrig tr)
    {
        // Beim ersten Aufruf werden die Tasks erzeugt
        if (!ChannelsDefined)
            CreateChannelsAndTasks();

        if (this.InvertDataSignals)
            writerDig_SCLK_RW_Trig.WriteSingleSampleMultiLine(true, new bool[] { !tb(sclk), tb(rw), tb(tr) });
        else
            writerDig_SCLK_RW_Trig.WriteSingleSampleMultiLine(true, new bool[] { tb(sclk), tb(rw), tb(tr) });
    }

    /// <summary>
    /// Setzt die SD-Leitung auf den angegebenen Wert. Die SCLK-Leitung bleibt unverändert.
    /// </summary>
    public void SetLine(pSD sd)
    {
        // Beim ersten Aufruf werden die Tasks erzeugt
        if (!ChannelsDefined)
            CreateChannelsAndTasks();

        if (this.InvertDataSignals)
            writerDig_SD.WriteSingleSampleMultiLine(true, new bool[] { !tb(sd) });
        else
            writerDig_SD.WriteSingleSampleMultiLine(true, new bool[] { tb(sd) });
    }
```

2.3.2 Schreiben eines Adress- und Datenworts mit Start- RW- und Stopbit

```
public void SendData(byte Adress, byte Data, bool CreateEEPuls)
    {
        if (Adress > 0x3F)
            throw new System.ArgumentOutOfRangeException("Maximal 6 Bit Adressbreite möglich");

        string message = "0";  // Read: 1, Write: 0
        message += DAQPad.ToBinaryString(Adress, 6);
        message += DAQPad.ToBinaryString(Data, 8);

        // Impulsbreite bzw. Impulspause in us
        long t1 = 0;

        try
        {
            // SCLK und SD auf High
            SetLine(pSCLK.Heigh, pSD.Heigh, pRW.Low, pTrig.Low);
            daq.DelayUS(t1);

            // Beginn der Übertragung: SD auf Low
            SetLine(pSCLK.Heigh, pSD.Low, pRW.Low, pTrig.Heigh);
            daq.DelayUS(t1);

            for (int i = 0; i < 15; i++)
            {
                char bit = message[i];

                // Das Trigger-Signal wird ab dem ersten Datenbit auf Low geschaltet
                SetLine(pSCLK.Low, bit == '0' ? pSD.Low : pSD.Heigh, pRW.Low, (i < 7 ? pTrig.Heigh : pTrig.Low) );
                daq.DelayUS(t1);

                SetLine(pSCLK.Heigh, bit == '0' ? pSD.Low : pSD.Heigh, pRW.Low, (i < 7 ? pTrig.Heigh : pTrig.Low));
                daq.DelayUS(t1);
```

```
        }

        // Nach dem letzten Bit SCLK und SD auf Low
        SetLine(pSCLK.Low, pSD.Low, pRW.Low, pTrig.Low);

        // Ende der Übertragung: SD auf High während SCLK Height ist.
        SetLine(pSCLK.Heigh, pSD.Low, pRW.Low, pTrig.Low);
        daq.DelayUS(t1);
        SetLine(pSCLK.Heigh, pSD.Heigh, pRW.Low, pTrig.Low);

        daq.DelayUS(t1);

      }
      catch (Exception ex)
      {
        ExceptionInfo info = new ExceptionInfo(ex);
        info.ShowDialog();
      }
    }
```

2.4 Lesen vom Nμ701.65 in den Microcontroller

2.4.1 Ein Bit einlesen

```
public bool ReadBit()
        {
            // Beim ersten Aufruf werden die Tasks erzeugt
            if (!ChannelsDefined)
                CreateChannelsAndTasks();

            bool input = readerDig_SD.ReadSingleSampleSingleLine();

            if (this.InvertDataSignals)
                input = !input;

            return input;
```

2.4.2 Eine Leseadresse senden und ein Datum einlesen

```
public byte ReadData(byte Adress)
    {
      if (Adress > 0x3F)
        throw new System.ArgumentOutOfRangeException("Maximal 6 Bit Adressbreite möglich");

      string message = "1";  // Read: 1, Write: 0
      message += DAQPad.ToBinaryString(Adress, 6);

      // Impulsbreite bzw. Impulspause in us
      long t1 = 0;

      try
      {
        // SCLK und SD auf High
        SetLine(pSCLK.Heigh, pSD.Heigh, pRW.Low, pTrig.Low);
        daq.DelayUS(t1);

        // Beginn der Übertragung: SD auf Low
        SetLine(pSCLK.Heigh, pSD.Low, pRW.Low, pTrig.Heigh);
        daq.DelayUS(t1);

        // Adresse senden
        for (int i = 0; i < 7; i++)
        {
          char bit = message[i];

          SetLine(pSCLK.Low, bit == '0' ? pSD.Low : pSD.Heigh, pRW.Low, pTrig.Heigh);
          daq.DelayUS(t1);
```

```
            SetLine(pSCLK.Heigh, bit == '0' ? pSD.Low : pSD.Heigh, pRW.Low, pTrig.Heigh);
            daq.DelayUS(t1);
        }

        // SCLK auf Low und einmal lesen damit die Leitung auf Input geschaltet wird
        SetLine(pSCLK.Low, pRW.Heigh, pTrig.Low);
        daq.DelayUS(t1);

        // SCLK ausgeben und Daten lesen
        byte Data = 0;
        for (int i = 0; i < 8; i++)
        {
            SetLine(pSCLK.Heigh, pRW.Heigh, pTrig.Low);
            daq.DelayUS(t1);

            // Daten bei der neg. Flanke lesen
            SetLine(pSCLK.Low, pRW.Heigh, pTrig.Low);
            bool b = ReadBit();

            Data *= 2;
            Data += (byte)((b == true) ? 1 : 0);

            daq.DelayUS(t1);
        }

        // Ende der Übertragung: SD auf High während SCLK Height ist.
        SetLine(pSCLK.Heigh, pRW.Low, pTrig.Low);
        daq.DelayUS(t1);
        SetLine(pSCLK.Heigh, pSD.Low, pRW.Low, pTrig.Low);
        daq.DelayUS(t1);
        SetLine(pSCLK.Heigh, pSD.Heigh, pRW.Low, pTrig.Low);

        return Data;
    }
    catch (Exception ex)
    {
        ExceptionInfo info = new ExceptionInfo(ex);
        info.ShowDialog();

        return 0;
    }


}
```