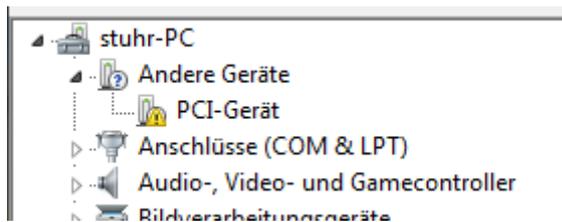


# Kurs 25# NI Daqmx auf Win7

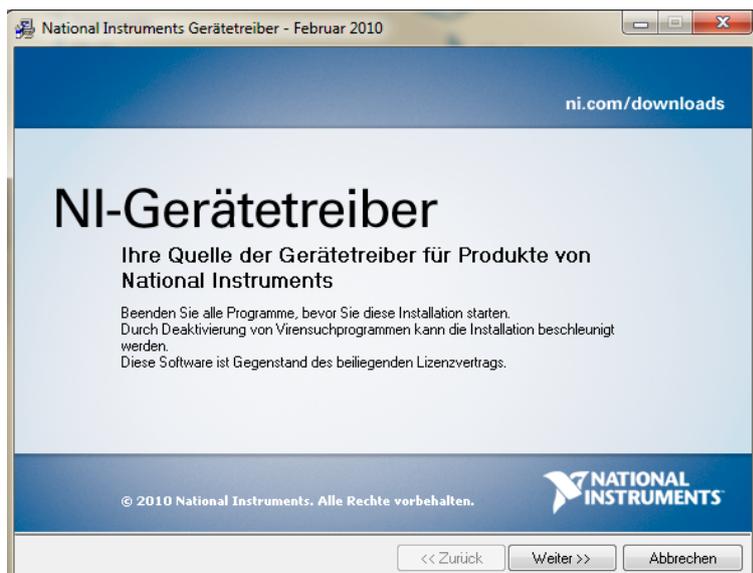
---

## Installation NI- AD- Karte

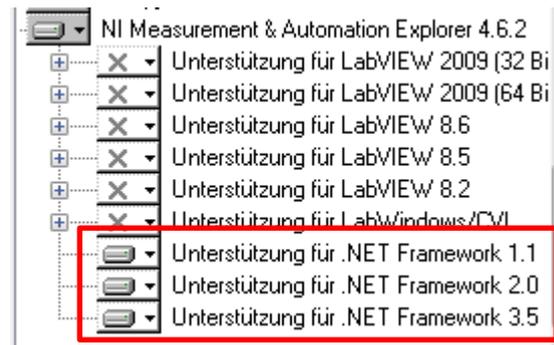
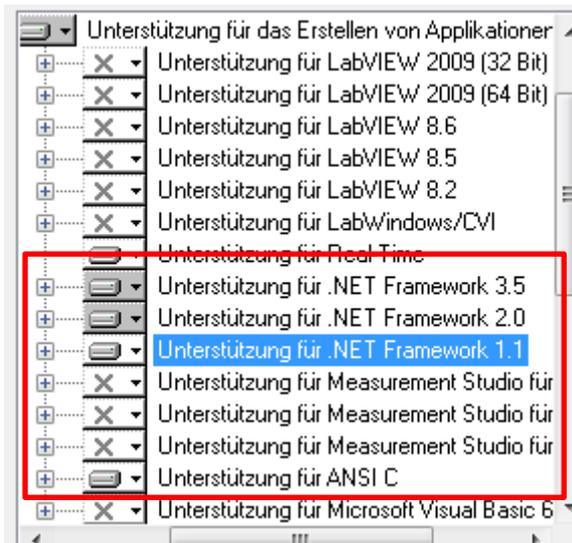
Karte in eine freien PCI- Slot stecken, um zu prüfen, ob sie nicht schon installiert worden ist. Wenn nicht, dann gibt es nach PC starten folgendes Bild: Im Geräte Manager erscheint dann ein unbekanntes Gerät:



Dann die mitgelieferte DVD der NI – Gerätetreibersoftware starten.

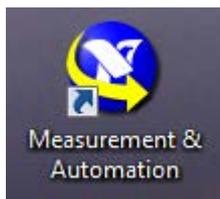
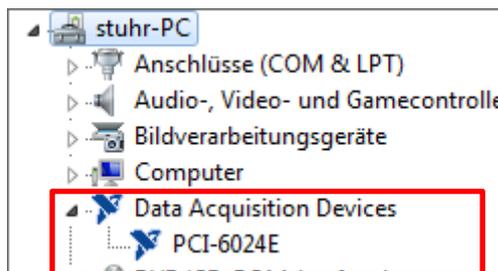


Dabei zusätzlich zu den vorgeschlagenen Komponenten auch unter „Unterstützung für das Erstellen von Applikationen“ folgende Optionen und bei NI Measurement Automation Explorer aktivieren:



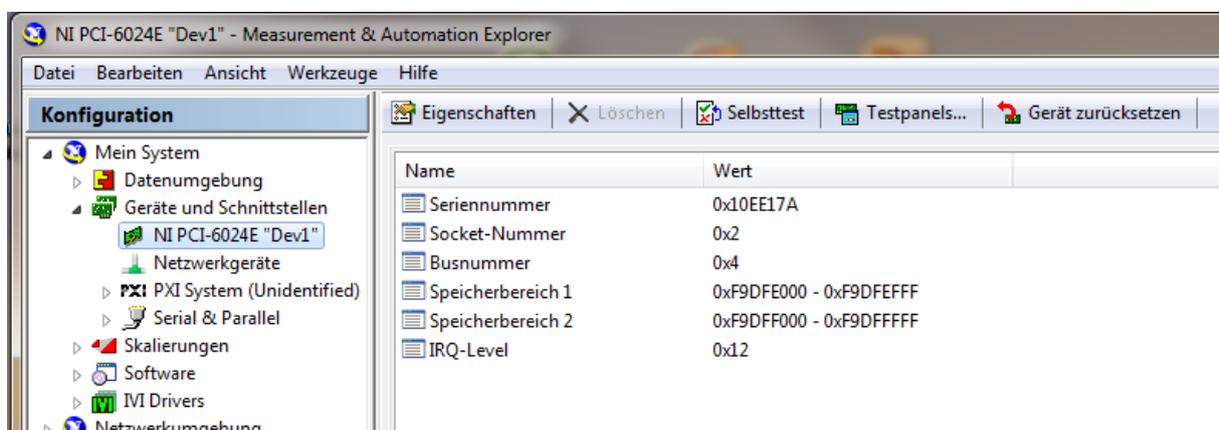
Jetzt braucht mal bloß noch - wie bei NI so üblich – mehrere Stunden zu warten bis alle <1000 Komponenten installiert sind, und dann kann man fortsetzen. Es werden wohl alle Treiber aller in NI jemals produzierten vorhandenen AD-Karten und nicht nur der der vorhandenen Karte installiert. Bei Kartenwechsel braucht dann keine erneute Installation durchgeführt zu werden.

Danach sollte es im Gerätemanager so aussehen (bei mir ist es eine einfache 6024 – Karte):

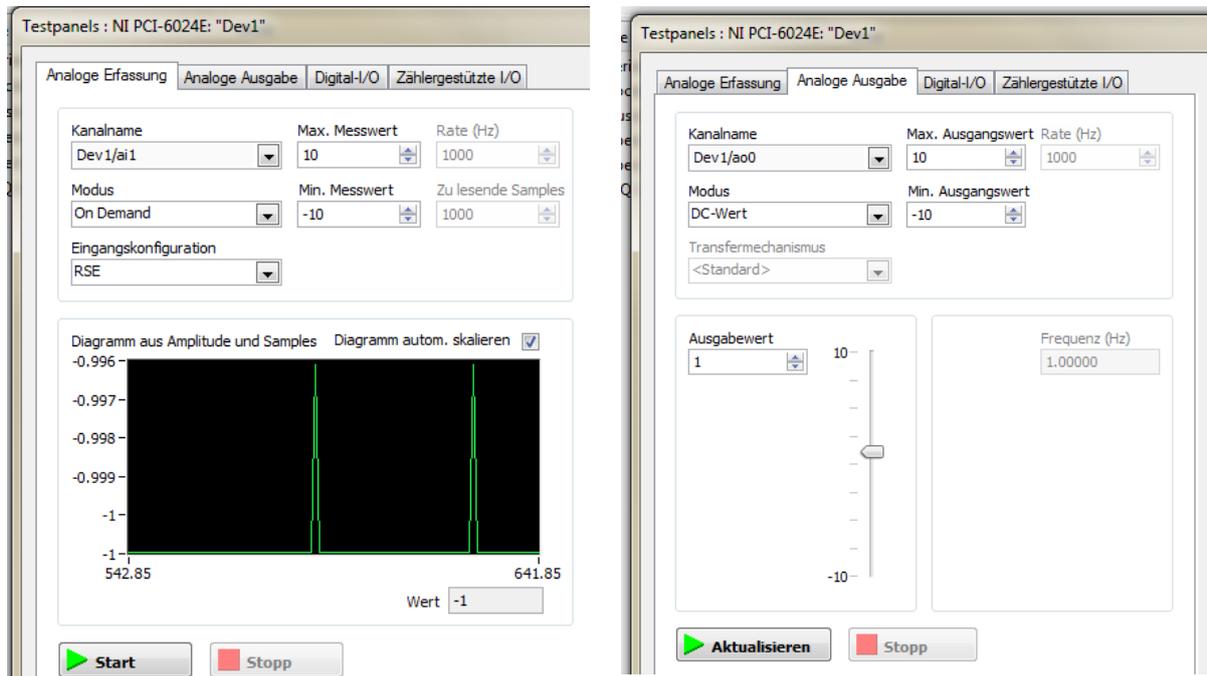


Dann sollte die Hardware getestet werden. Dazu gibt es das Programm NI Measurement Automation Explorer mit Icon auf dem Desktop.

Nach Start dieses Programms:



Mit dem Testpanel kann man so die analogen Eingänge anschauen und die analogen Ausgänge so einstellen (man beachte die Einstellung RSE = referential single ended):



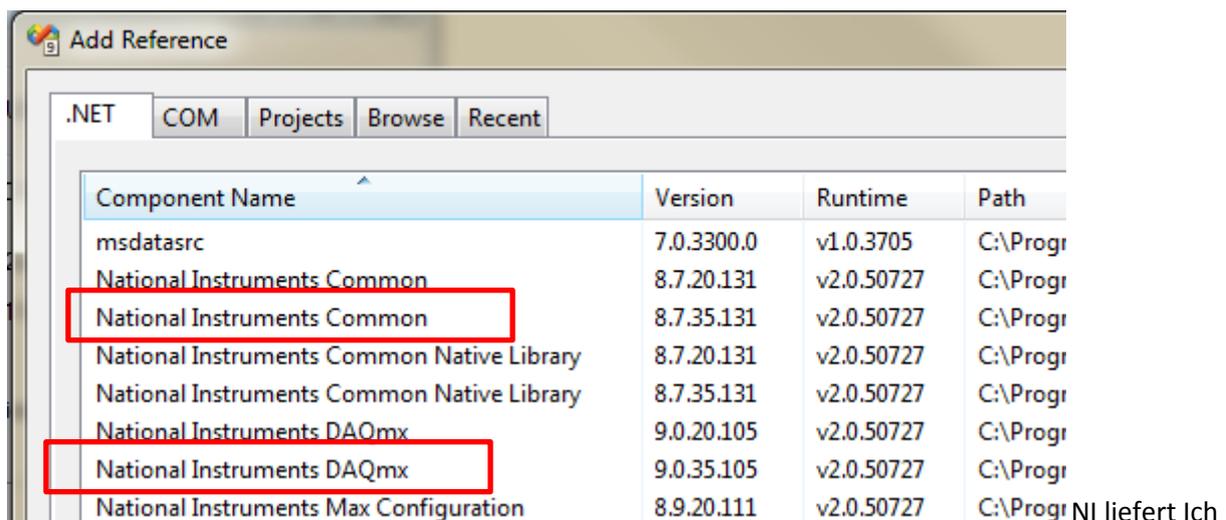
Ich hab wieder AI 0 mit AO 0 und AI 1 mit AO 1 verbunden, AO 0 = +1 Volt, AO 1 = - 1 Volt.

Wie Sie sehen, zeigt AI 1 den richtigen Wert an.

## Software in C#

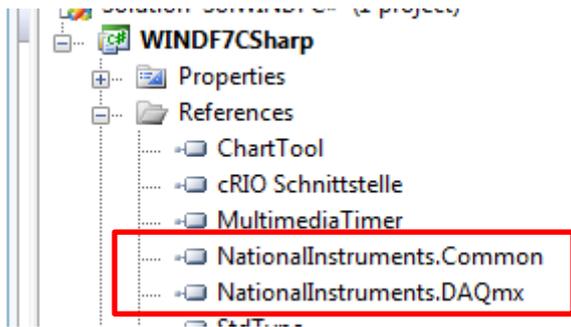
Jetzt zur Software in C#.

Zum Projekt müssen zwei Referenzen / Verweise hinzugefügt werden:



hab die mit der höheren Versionsnummer gewählt.

Dann sieht es im Solution Explorer so unter Referenzen aus:



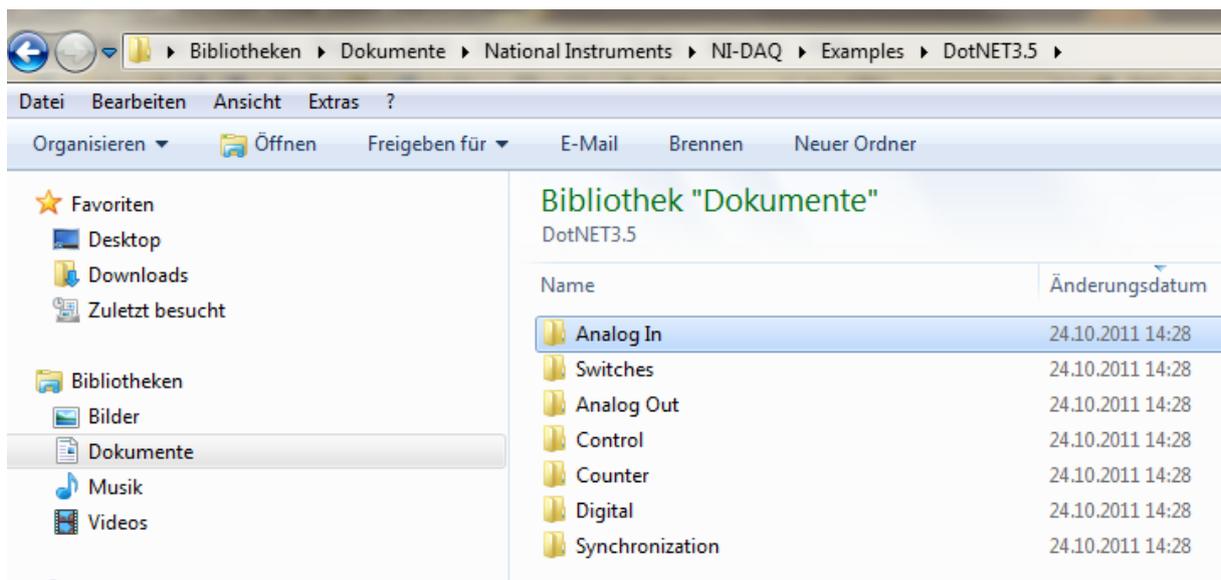
Diese Referenzen verweisen auf zwei DLL-Dateien, die zu finden sind unter:

*C:\Program Files (x86)\National Instruments\MeasurementStudioVS2008\DotNET\Assemblies\Current\NationalInstruments.Common.dll*

*C:\Program Files (x86)\National Instruments\MeasurementStudioVS2008\DotNET\Assemblies\Current\NationalInstruments.DAQmx.dll*

Man erkennt erstaunt, dass sie unter *Measurement Studio* stehen, obwohl dieses Programmpaket nicht auf dem PC vorhanden ist.

NI liefert Beispiel mit, die zu finden sind unter



Dann `using NationalInstruments.DAQmx;`

Und dann sind folgende Befehle möglich:

AD auslesen:

```
double val = rschan.ReadSingleSample();
```

Ausgabe auf DA

```
wschan.WriteSingleSample(true, val);
```

zuvor muss man die beiden Tasks rschan und wschan „kreieren“ mit

```
AnalogSingleChannelReader rschan;  
  
AnalogSingleChannelWriter wschan;
```

Und das geht so: (Mein Kommentar dazu: so einen komplizierten Ablauf kann sich nur ein Informatiker ausdenken):

Erst vorweg einmal die Instanziierung der AD- Tasks: (DA geht ähnlich, siehe Datei .cs):

```
Task taskAI = new Task("abci");
```

Dann:

```
taskAI.AIChannels.CreateVoltageChannel(actdev+"/ai0", "aiChannel0",  
    AITerminalConfiguration.Rse,MinI, MaxI, AIVoltageUnits.Volts);
```

Dann

```
rschan = new AnalogSingleChannelReader(taskAI.Stream);
```

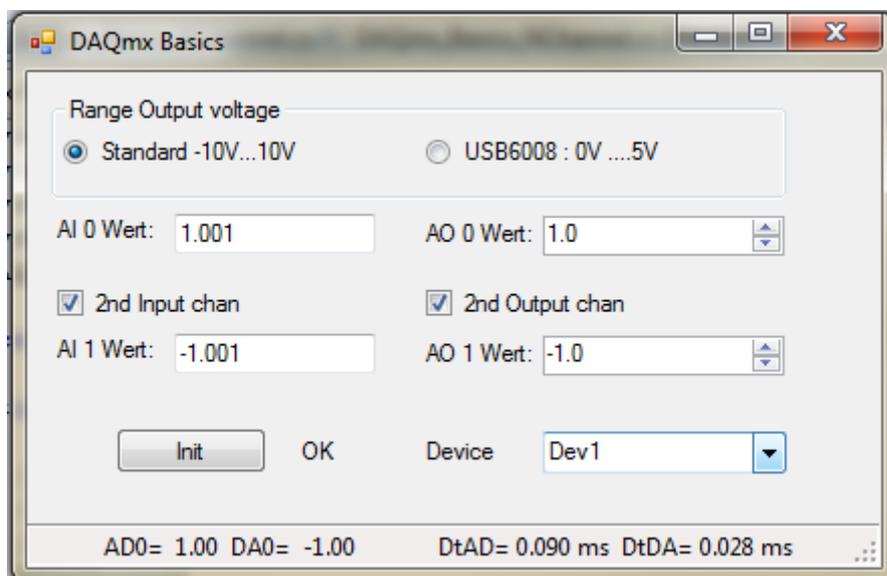
Dann einmalig vorher den Start nicht vergessen, sonst dauert es pro AD oder DA sehr lange (>1ms):

```
taskAI.Start();
```

dann sollte es gehen.

## Zeitmessung

Mit einer Stopwatch kann man nun die Laufzeiten der Ein- und Ausgabe messen. Im fertigen Projekt hab ich die Messung eingebaut. Folgende Zeiten ergeben sich mit dieser Karte auf einem Dell Optiplex 780 mit Pentium Dual Core 2,6 GHz:



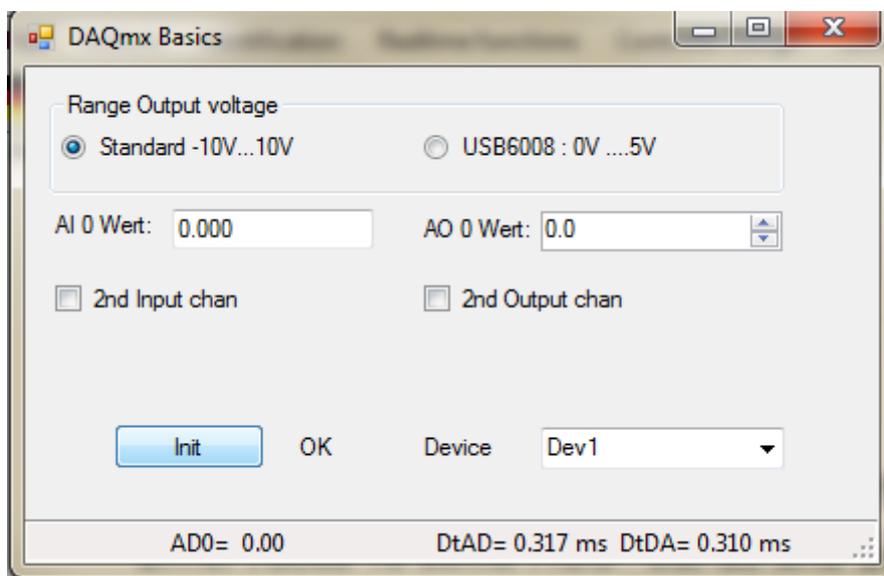
Mit zwei AD – DA – Aufrufen insgesamt 120  $\mu$ sec, so geht das locker noch in einer 1 msec- Schleife.  
NI im Vergleich zu Meilhaus:

AD- Eingabe: Meilhaus : ca. 60 $\mu$ sec! NI : 90  $\mu$ sec (1,5 – mal langsamer)

DA Ausgabe: Meilhaus : ca. 20  $\mu$ sec! NI ca. 28  $\mu$ sec (1,5 – mal langsamer)

Die Zeiten sind absolut akzeptabel.

Was interessant ist: Wenn man das Projekt auf einem 64 – Bit Win7 kompiliert mit Platform Target „Any CPU“, dann ergeben sich mit dem exakt gleichen Programm deutlich größere Zeiten (pro AD oder DA ca. 300  $\mu$ sec.):



Startet man dieses Programm nicht von der Entwicklungsumgebung, sondern das exe. File, dann sind die Zeiten wieder niedrig wie zuvor.

Kompiliert man mit Platform Target „x86“, dann ist es auch in der Entwicklungsumgebung schnell.

## Portieren

Nun gibt es das Problem, dass alles auf dem PC läuft, auf dem alles installiert ist. Wenn man das Projekt auch auf dem PC laufen und kompilieren möchte, auf dem keine NI – Karte installiert ist, wird es sofort eine Fehlermeldung geben, da die DLL- Dateien *NationalInstruments.Common.dll* und *NationalInstruments.DAQmx.dll* dann natürlich nicht gefunden werden und die Referenzen ins Leere zeigen.

Dann lässt sich besser folgender Weg gehen. Diese beiden Dateien werden in das aktuelle bin/Debug – Verzeichnis kopiert und die Referenz wird dann per Browse auf diese beiden DLL gesetzt. Dann ist ein Kompilieren auf jedem PC möglich.

Gezeichnet

Prof. Dr. Bayerlein