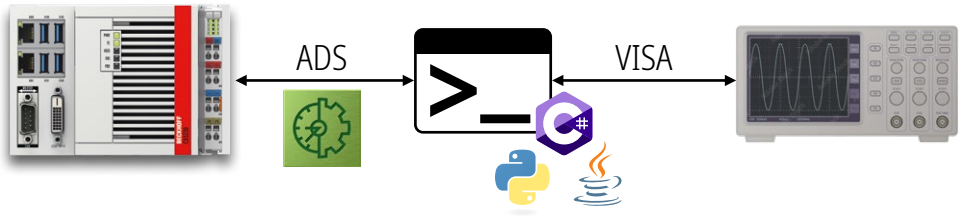
BECKHOFF

Application Note

Keywords
TwinCAT
.NET
ADS
VISA
IVI
SCPI
GPIB
IEEE 488

Ansteuerung von Test- und Messequipment über ADS und TwinCAT



Soll der Funktionsumfang einer Anlage durch Test- und Messtechnik (T&M) erweitert oder ein aufwändiger Laboraufbau automatisiert werden (ATE: Automated Test Equipment), bedarf es einer Schnittstelle zwischen diesen beiden Welten. Programmiersprachen und Tools, die im T&M-Umfeld verbreitet sind, wie MATLAB® oder LabVIEW™, lassen sich über Softwarepakete in TwinCAT integrieren. Besteht Interesse daran, Messgeräte, ohne die weiteren Vorzüge dieser Werkzeuge zu nutzen, ist dies ebenfalls möglich. Der Einsatz eines PCs als Automatisierungssystem bietet dabei den Vorteil einer großen Bandbreite an Hardwareschnittstellen, die bereits ab Werk verfügbar sind oder sich auf einfache Weise erweitern lassen.

Im Folgenden wird beschrieben, wie die Messgerätekommunikation aus einer PC-basierten Steuerung auf Basis von TwinCAT realisiert werden kann. Es wird ein minimales ADS-Server-Programm erstellt („Service“), um Code aus Drittherstellerbibliotheken im User-Mode auszuführen. Auf diese Weise ist es möglich, Geräte aus TwinCAT heraus über das VISA-Protokoll anzusteuern.

▶ www.beckhoff.com/mess-und-prueftechnik

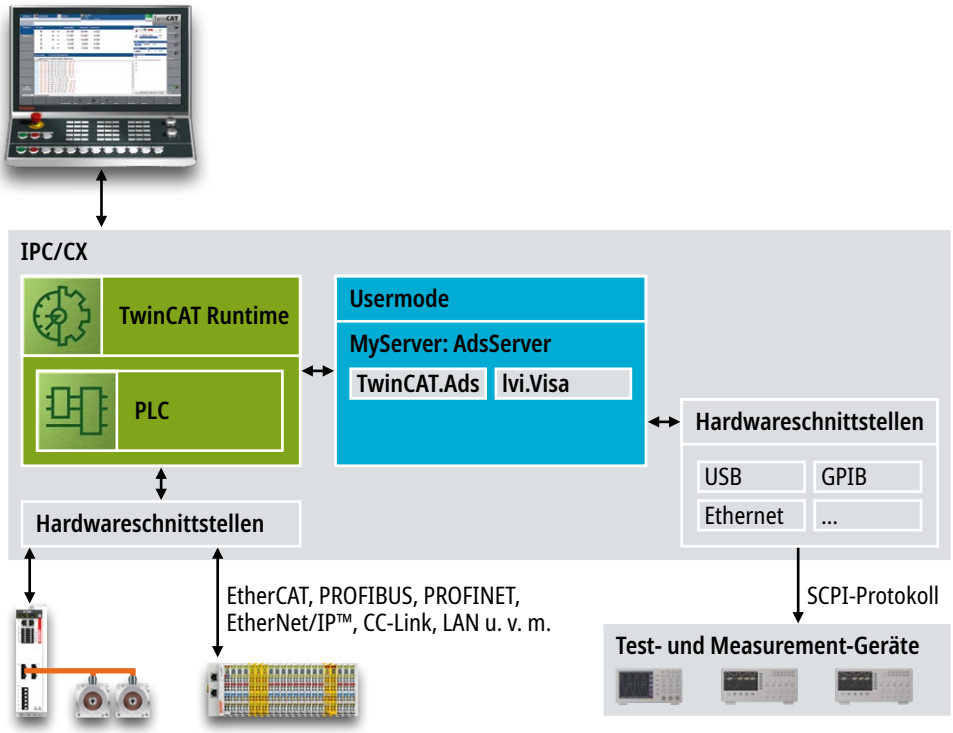


Abbildung 1:
Schematische Darstellung der Systemkomponenten

Dazu im Vorfeld einige Begriffserläuterungen, insbesondere für die Unterscheidung zwischen elektrischer Übertragung (physikalische Kommunikationsschicht, z. B. RS232, LAN, GPIB, ...) und Protokoll:

ADS – Automation Device Specification

ADS ist ein Kommunikationsprotokoll, das in den 1990er Jahren von Beckhoff für die Interaktion zwischen verschiedenen Geräten und Softwarekomponenten innerhalb eines TwinCAT-Automatisierungssystems entwickelt wurde. Die Nutzung des Protokolls ist ohne zusätzliche Lizensierung möglich und kann in verschiedenen Programmiersprachen sowie plattformunabhängig erfolgen. Wird die Kommunikation zu anderen PCs oder Geräten benötigt, setzt das ADS-Protokoll auf TCP/IP auf. Die Teilnehmer werden durch ihre Net ID adressiert und Dienste werden anhand von Port-Nummern unterschieden. Die relevanten Daten werden anhand der Index-Gruppe und des Offsets unterschieden.

Mit dem TF6010 ADS Monitor kann der Datenverkehr mitgelesen werden.

▶ infosys.beckhoff.com

GPIB, IEEE 488

Dies ist eine elektrische parallele Schnittstelle zur Verbindung von Geräten. Es ist nicht definiert, welches Protokoll (Kommandos) darüber gesprochen werden soll. Zum GPIB-Anschluss kommt spezielle Hardware (PC-Einsteckkarten, USB-Konverter) zum Einsatz.

SCPI – Standard Commands for Programmable Instruments

Beim SCPI-Protokoll handelt es sich um standardisierte Kommandos (Befehlssatz) im ASCII-Format,die zur Steuerung von Messgeräten genutzt werden. So kann bei kompatiblen Geräten mit *IDN? beispielsweise die Seriennummer abgefragt werden und *RST setzt das Gerät zurück. Die Kommandos können auch von Herstellern um eigene Befehle erweitert werden, welche an diese Syntax angelehnt sind. Das Protokoll kann über unterschiedliche Physik übertragen werden, z. B. USB, GPIB u. a.

VISA – Virtual Instrument Software Architecture

VISA ist eine API (Application Programming Interface, Softwareschnittstelle), die die Kommunikation zwischen einem Computer (bzw. den dort laufenden Programmen) und Messgeräten über verschiedene Hardwareschnittstellen, wie GPIB, USB oder Ethernet, ermöglicht. Eine VISA-Anwendung versendet/empfängt u. a. das SCPI-Protokoll.

Dieser Industriestandard wird von mehreren Unternehmen in der Branche implementiert. Da auf gemeinsame Komponenten der IVI Foundation zurückgegriffen wird, können diese parallel auf demselben System laufen. Die Hersteller unterscheiden sich im Wesentlichen anhand der zusätzlich angebotenen Software. Beispiele für VISA-Implementierungen sind:

Keysight	IO Libraries Suite
National Instruments	NI-VISA
Rohde & Schwarz	R&S®VISA
Tektronix	TekVISA

IVI – Interchangeable Virtual Instruments

Die IVI Foundation wurde als offenes Konsortium gegründet, um Spezifikationen für die Programmierung von Test- und Messgeräten zu fördern. Das Konzept zielt darauf ab, durch standardisierte Treiberarchitekturen den Austausch von Messgeräten in automatisierten Test- und Messsystemen zu erleichtern. Sie bieten Treiber an, welche dazu Klassen an Instrumenten nach Funktion (Digitalmultimeter, Oszilloskop, Funktionsgenerator etc.) zusammenzufassen, und spezifizieren die gemeinsamen Komponenten (Ivi.Visa) für herstellerseitige VISA-Implementierungen. Die IVI pflegt u. a. den SCPI-Befehlssatz und VISA.
Links: The SCPI Standard | IVI Foundation

Erste Schritte mit VISA

Bevor mit der Implementierung eines eigenen Services begonnen wird, kann es von Vorteil sein, sich mit der grundlegenden Funktionsweise der VISA-Kommunikation vertraut zu machen. Einen schnellen Einstieg bieten die grafischen Hilfsprogramme der Gerätehersteller. National Instruments™ bietet „VISA Interactive Control" an. Das Softwarepaket von NI™ ist sehr umfangreich und enthält Komponenten, die für die Verwendung von LabVIEW™ benötigt werden. Dem gegenüber stehen die „VISA-Tools" von Rohde & Schwarz, mit einem vergleichbar kleinen Footprint. Folgend wird der „Connection Expert" von Keysight verwendet, der zusammen mit weiteren nützlichen Werkzeugen in der IO Libraries Suite enthalten ist. Der grundlegende Funktionsumfang ähnelt sich, sodass es für den hier beschriebenen Anwendungsfall unerheblich ist, welches Softwarepaket verwendet wird.

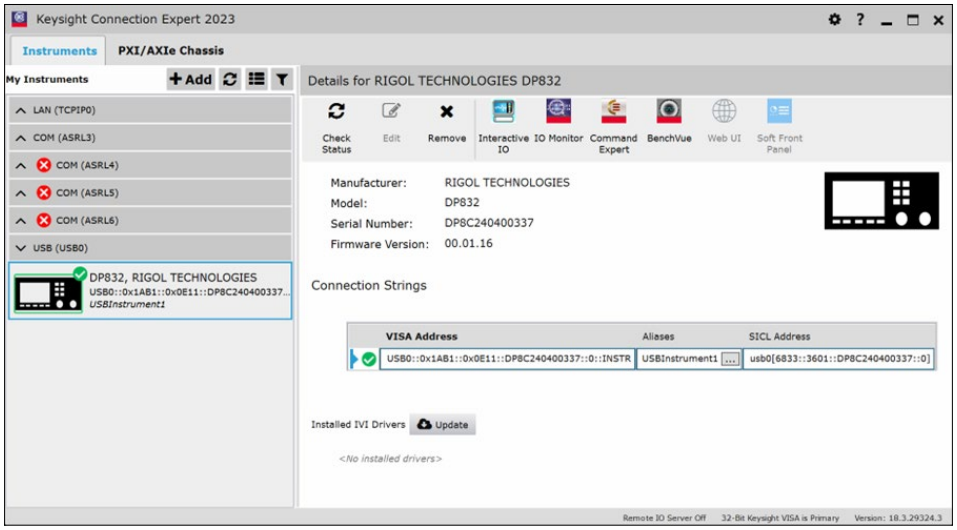


Abbildung 1:
Auflistung verbundener Geräte im Connection Expert

Nach Anschluss eines Netzteils über die USB-Schnittstelle wurde dieses durch den automatischen Scanvorgang erkannt. Bei seriellen Geräten kann es sein, dass diese zunächst über [+ Add] manuell konfiguriert werden müssen. Es wird die VISA-Adresse des Gerätes angezeigt, die im Folgenden genutzt werden kann, um die Kommunikation mit dem Gerät aufzubauen. Der Aufbau der Adresse unterscheidet sich je nach Schnittstellentyp und enthält verpflichtende und optionale Elemente. Die hier dargestellte USB-Adresse lässt sich wie folgt in ihre Komponenten zerlegen.

USB0 :: 0x1AB1 :: 0x0E11 :: DP8C240400337 :: 0 :: INSTR

- USB-Karte
- Hersteller-ID
- Produkt-ID
- Gerätetyp
- Schnittstellennummer
- Seriennummer des Gerätes

Beispiele für diverse Schnittstellenbeschreibungen lassen sich in der Dokumentation der VISA-Anbieter finden.[1] Zwei weitere Werkzeuge, die mit installiert wurden, sind „Interactive IO“ und „IO Monitor“. Über Ersteres können SCPI-Kommandos und erweiterte Befehle, die vom Gerätehersteller implementiert wurden, gesendet und die Antwort des Gerätes abgefragt werden. Die standardisierte ID-Abfrage liefert einen Antwort-String mit Gerätedaten.

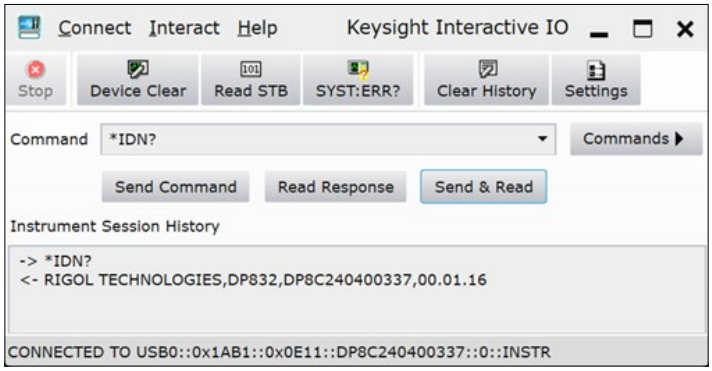


Abbildung 2:
Abfrage der Seriennummer via Interactive IO

Mit dem zweiten Werkzeug kann der Kommunikationsverlauf mit einem Gerät aufgezeichnet werden. Es dient als Hilfsmittel während der Entwicklung, kann aber auch bei der Fehlersuche in Bestandssystemen eingesetzt werden. Wird nach Start der Aufnahme Interactive IO geöffnet, werden eine Reihe von Aufrufen der VISA-C-Bibliothek aufgelistet.

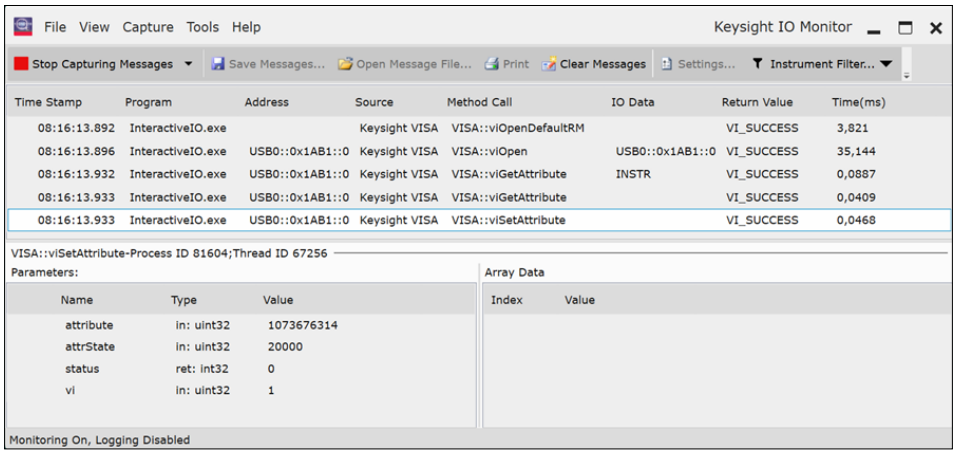


Abbildung 3:
Mitschneiden der VISA-Kommunikation mittels IO Monitor

Die vorgestellte Software ist in C# für das .NET Framework geschrieben. Die C-API wird daher nicht direkt genutzt, sondern VISA.NET, welche ein Wrapper für VISA-C ist. Den Aufrufen kann entnommen werden, dass ein Kommunikationskanal (Session genannt) zum Gerät geöffnet wird, Attribute gelesen und auch ein Attribut gesetzt wird. In .NET werden diese Attribute durch Eigenschaften des Verbindungsobjektes abgebildet.
Im Parameterfeld ist eine genauere Beschreibung des Set-Attribute-Aufrufs dargestellt. Dem Gerät wurde der Handle 1 zugewiesen (vi 1) und es hat den Befehl ohne Fehler

[1] https://www.ni.com/docs/de-DE/bundle/ni-visa/page/visa-resource-syntax-and-examples.html

ausgeführt (status 0). Eine 10-stellige Nummer verweist auf das Attribut, dem der Wert 20.000 zugewiesen wurde. Der Code verweist auf „VI_ATTR_TMO_VALUE“, dem Timeout für diese Verbindung. Dieser lässt sich unter Settings im Interactive IO einstellen. Die Codes sind für alle VISA-Implementierungen festgelegt und auf den Herstellerseiten dokumentiert.[2] Die bislang durchgeführten Operationen lassen sich wie folgt in C# abbilden:

```
// Suche nach allen angeschlossenen Geräten
IEnumerable<string> devices = GlobalResourceManager.Find();

// Erstelle eine textbasierte Verbindung zum Gerät
string VISA_ADDRESS =
"USB0::0x1AB1::0x0E11::DP8C240400337::0::INSTR";
var session = GlobalResourceManager.Open(VISA_ADDRESS) as IMes-
sageBasedSession;
var formattedIO = new MessageBasedFormattedIO(session);

// Lege die Timeout-Eigenschaft fest
session.TimeoutMilliseconds = 20_000;

// Sende das SCPI-Kommando um die ID abzufragen
formattedIO.WriteLine("*IDN?");
string idnResponse = formattedIO.ReadLine();

// Freigeben der Ressource
session.Dispose();
```

Codebeispiel 1: Minimaler ADS-Server durch Überschreiben der Read/Write-Indikation

Ergänzend zur Dokumentation werden von den Geräteherstellern häufig Anwendungsbeispiele als Download angeboten. Diese decken bereits viele typische Standardanwendungen ab und erleichtern den Einstieg in die VISA .NET API. Zu diesem Zeitpunkt sind noch keine VISA-Implementierungen für das moderne .NET vorhanden. Sollen die VISA-Bibliotheken out of the box genutzt werden, ist daher .NET-Framework zu verwenden.

i

In der IVI Spezifikation VPP-4.3.6 wurde angekündigt, dass die VISA.NET-Shared-Components ab der Version 7.3 für .NET 6+ verfügbar sein werden.[3] Um bereits jetzt in .NET 6+ zu arbeiten, können die installierten VISA-Versionen über die Klasse Ivi.Visa.ConflictManager gefunden werden. Mittels der Assembly-Klasse können diese aus %windir%\assembly (bzw. %windir%\Microsoft.NET\assembly für .NET Framework 4+) geladen werden. Hintergrund ist, dass moderne .NET-Anwendungen nicht länger den globalen Assemblycache (GAC) verwenden. Alternativ kann auch ein eigener Wrapper um VISA-C geschrieben werden.

[2] https://helpfiles.keysight.com/IO_Libraries_Suite/English/IOLS_2024_U1_Windows/VISA/Content/visa/VISA%20Attributes%20Table.html
[3] https://www.ivifoundation.org/specifications/default.html

Implementierung des ADS-Servers

Um aus der SPS auf Drittanbieterbibliotheken wie VISA zugreifen zu können, soll ein virtueller Funktionsserver implementiert werden, der steuerungsseitig über ADS kommuniziert. Das Programm wird im Usermode ausgeführt und ist daher, im Gegensatz zur TwinCAT Runtime, nicht echtzeitfähig. Die .NET API und Beschreibungen für Programmiersprachen außerhalb dieses Frameworks sind im Infosys[4] zu finden.

```
public class MyServer : AdsServer
{
    public MyServer() : base(25_000, "MyServer") { }

    protected override Task<ResultReadWriteBytes> OnRead-
WriteAsync(
        AmsAddress sender, uint invokeId,
        uint indexGroup, uint indexOffset,
        int readLength, <byte> writeData,
        CancellationToken cancel)
    {
        ResultReadWriteBytes result;
        switch (indexGroup)
        {
            case 1:
                result = ResultReadWriteBytes.CreateSuc-
cess(Memory<byte>.Empty);
                break;

            default:
                result = ResultReadWriteBytes.CreateError(
                AdsErrorCode.DeviceServiceNotSupported);
                break;
        }
        return Task.FromResult(result);
    }
}
```

Codebeispiel 2: Minimaler ADS-Server durch Überschreiben der Read/Write-Indikation

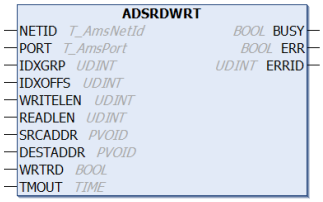
Für die ADS-Seite der Applikation sind alle benötigten Software-Komponenten im NuGet Paket Beckhoff.TwinCAT.Ads[5] enthalten. Für eigens erstellte Serveranwendungen werden seitens Beckhoff die Portnummern von 25000–25999 reserviert, um eine Überschneidung mit bereits bestehenden Funktionen[6] zu vermeiden. Im Codebeispiel 2 wird ein ADS-Server definiert, der lokal auf dem Port 25000 gestartet werden kann. Die Read/Write-Methode wurde überschrieben, sodass diese bei einer Anfrage für die Index-Gruppe 1 einen Erfolg und für andere Anfragen einen NotSupported-Fehler zurückliefern würde (ADS return code 1793). Aus dem SPS-Programm heraus kann diese Anfrage mit dem ADSRDWRT Baustein aus der Tc2_System Bibliothek gestellt werden.

[4] https://infosys.beckhoff.com/index.php?content=../content/1031/tc3_ads.net/9407515403.html&id=
[5] https://www.nuget.org/packages/Beckhoff.TwinCAT.Ads/
[6] https://infosys.beckhoff.com/index.php?content=../content/1031/tc3_grundlagen/116159883.html&id=

```
var
    AdsReadWrite : ADSRDWRT;
    Payload : usint;
    Response : usint;
end_var

AdsReadWrite(
    NETID:= '',
    PORT:= 25_000,
    IDXGRP:= 1,
    IDXOFFS:= 0,
    WRITELEN:= sizeof(Payload),
    READLEN:= sizeof(Response),
    SRCADDR:= adr(Payload),
    DESTADDR:= adr(Response),
    WRTRD:= true);
```

Codebeispiel 3: Definition und Aufruf des parametrierten AdsRead-Write-FBs



Der ADS-Server kann nun mit den benötigten Funktionen ausgestattet werden. Die konkreten Implementierungsdetails richten sich nach der jeweiligen Applikation.

Weiterführende Links:
ADS-Beispiele von Beckhoff
VISA-Beispiele von Keysight

BECKHOFF

Sie möchten mehr erfahren?

Wir informieren Sie gern und
freuen uns auf Ihre Anfrage:

support@beckhoff.com

▶ www.beckhoff.com/mess-und-prueftechnik

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 52469630
info@beckhoff.com
www.beckhoff.com