1 Die Testo „Toolbox“

Die Testo Toolbox ist eine Programmbibliothek, welche das einfache Ansprechen der Messgeräte der Testo AG aus einem eigenen Anwendungsprogramm heraus ermöglicht.

Die Schnittstelle ist geräteunabhängig realisiert. Damit können Sie die Funktionen, die bei allen Geräten vorhanden sind, immer auf dieselbe Art und Weise ansprechen. Diese sind:

- Aufbau der Verbindung zum Gerät
- Auslesen der aktuellen Messwerte (Online-Messung)
- Speichern von Bezeichner (Messorte)
- Auslesen der gespeicherten Messwerte (Protokolle)

Für einige Geräte (testo 174, testo 175, testo 177, testo 350 M/XL) stehen gerätespezifische Erweiterungen zur Verfügung, mit Hilfe derer Sie weitergehende Eigenschaften des Geräts ausnutzen können. Beispielsweise können Sie so ein Loggerprogramm realisieren, wo ab einer bestimmten Zeit in einer gewissen Art und Weise gemessen wird.

Die Toolbox ist als OLE Automation Server realisiert. Somit kann die Integration mit jeder Programmiersprache erfolgen, welche ActiveX Controls unterstützt. Diese sind beispielsweise

- Microsoft Visual Basic ab Version 5
- Microsoft Visual Basic .NET
- Microsoft Visual Basic for Applications (VBA) wie es in Microsoft Word, Excel und Access enthalten ist.
- Microsoft Visual C++ ab Version 5
- Microsoft Visual C++ .NET
- Borland Delphi Version ab Version 3
- National Instruments LabView ab Version 5

Die Toolbox wird zusammen mit der Comfort Software installiert. Es wird zusätzlich eine Online Dokumentation der Schnittstelle installiert zusammen mit Beispielprogrammen für Microsoft Visual Basic, Microsoft Visual C++ und National Instruments LabView.

Inhalt:

Version 2	Oct 2005	Geräte mit USB Schnittstelle hinzugefügt

2 Arbeiten mit der Testo „Toolbox“

Machen Sie zunächst die Typeninformation ihrer Programmierungebung bekannt:

Microsoft VB ab version 5	Projekt, References	Tcddka
Microsoft VC ab version 5	#import	<installationsverzeichnis>\Tcddka.dll

In den meisten anderen Sprachen werden sie in der Kategorie „ActiveX Steuerelemente“ mit dem Schlüsselwort „tcddka“ fündig.

2.1 Gerät anschliessen

Das Objekt tcddk erfordert von Ihnen besondere Pflege, hier insbes. die Kontrolle der „Lebenszeit“. Ein häufig gemachter Fehler ist die Verwendung von globalen Variablen. Diese werden in aller Regel beim Programmende in irgendeiner Reigenfolge „geschlossen“, was fast zwangsläufig zu Problemen führt. Tcddk erfordert ein manuelles „schliessen“. In VB durch das zuweisen von „nothing“.

Keine zwei Objekt können gleichzeitig mit einem angeschlossenen Gerät arbeiten. Darauf wird bei Busverbindungen nochmals hingewiesen werden.

VB

```
Dim inst As tcddk
Dim bRet As Boolean

Set inst = New tcddk
bRet = inst.Init(0, "testo175-177", 5000)

If True = bRet Then
    '...
End If

Set inst = Nothing
```

2.1.1 Geräte mit serieller Schnittstelle

Verwenden Sie die Methode Tcddk::Init um die Arbeit mit einem seriell angeschlossenen Gerät zu beginnen.
Init hat drei Parameter

WidCom	Nummer des COM Ports	0 Basiert, verwenden sie 0 für die erste serielle schnittstelle (COM1) usw.
LpszDevicename	Gerätekennung	Folgende Werte sind gültig (Gross-kleinschreibung beachten): testostor175,

		testostor171 testo400-650-950 testo445-645-945-946-545 testo454-2000 testo175-177 testo174 testo300-M-XL
DwTimeOut	Max. Initialisierungszeit in [msec].	Diese variiert nach Gerät. Da die Initialisierung auch das vollständige inhaltsverzeichnis des Gerätespeichers liest kann eine erfolgreiche initialisierung bei t400, t454-2000 bis zu 30 sekunden dauern.

Init liefert als Ergebnis „True“ oder „False“. Da Initialisierung ein kritischer Vorgang ist und aus den verschiedensten Gründen fehlschlagen kann, benutzen sie in jedem Fall die asynchrone Fehlerbehandlungstechnik ihrer Programmierumgebung: C++Exceptions oder die VB OnError Direktiven. Ein unter Umständen auftretendes Fehlerobjekt enthält eine Beschreibung der möglichen Ursache.

VB

```
On Error GoTo ErrorLabel
Dim inst As tcddk
Dim bRet As Boolean

Set inst = New tcddk
bRet = inst.Init(1, "testo175-177", 5000)

If True = bRet Then
    '...
End If

GoTo EndLabel

ErrorLabel:
MsgBox Err.Description

EndLabel:
Set inst = Nothing
```

2.1.2 Geräte mit Busschnittstelle

Diese Geräte werden über ihre Busadresse angesprochen. Das Objekt Tcbus ermittelt die Adressen aller am Bus angeschlossenen Testo Geräte (sofern sie betriebsbereit sind). Tcbus::item liefert das entsprechende Object tcddk.

Item liefert das bekannte tcddk Objekt. Bitte beachten Sie, das dafür nicht mehr mit tcddk::init aufgerufen werden muss. Dieses Objekt ist bereits initialisiert

Item wird bei jedem Aufruf ein neues tcddk Objekt erzeugen. Wiederholte Aufrufe

	des selben Items (mit der selben Adresse) können zu unerwünschten Ergebnissen führen. Stellen Sie sicher, dass ein Item „geschlossen“ wird (in VB durch zuweisung von „Nothing“), bevor sie seine Adresse erneut verwenden.

VB

```
Dim bus As tcbus
Set bus = New tcbus

bus.Init

Dim dev(255) As tcddk

Dim addr As Variant

i% = 0
For Each addr In bus
    Set dev(i) = bus.Item(addr)
    i% = i% + 1
Next
```

2.1.3 Geräte mit USB Schnittstelle

Verwenden sie die Funktion InitSerial um ein Gerät mit USB Schnittstelle wie z.B. Testo 435,635,735 anzuschliessen.

VB

```
Set dev = New tcddk
dev.InitSerial serial, "testo435-635-735", 5000
Set dev = Nothing
```

Der Parameter “serial” ist hier die Seriennummer des Geräts. Diese ist auf dem Etikett aufgedruckt.

Die Datenlogger 175-177 werden ebenfalls optional mit USB Schnittstelle angeboten. Hier wird jedoch eine serielle Schnittstelle emuliert. Für die Testo Toolbox sind dies keine USB Geräte.

Bei Logger 175-176-2010 ist Initserial zu verwenden

Seriennummern können auch programmgesteuert ermittelt werden. Das Objekt tcusbserials ermittelt die Seriennummern aller über USB angeschlossenen Testo Geräte eines bestimmten Typs.

VB

```
Dim usbserials As tcusbserials
Set usbserials = New tcusbserials

usbserials.Init ("testo435-635-735")

Dim serial As Variant
```

```
For Each serial In usbserials
' …
Next
```

2.2 Online messen

Messbereite Geräte zeigen das in tcddk::NumCols durch einen Wert grössen 0 an.

VB

```
If inst.NumCols > 0 Then
        inst.Get
        For i% = 0 To inst.NumCols - 1 Step 1
            If inst.IsNonNumeric(i%) Then
                msg$ = Format(i%, "0")
                msg$ = msg$ + " " + inst.StringVal(i%)
                MsgBox (msg$)
            Else
                msg$ = Format(i%, "0")
                msg$ = msg$ + " " + Format(inst.RecentVal(i%),
"##0.0") + inst.Unit(i%)
                MsgBox (msg$)
            End If
        Next i
    End If
```

Tcddk::MinRate gibt das kürzeste zulässige Wiederholintervall für Online“Anfragen“ aus. Der Wert ist in [msec]. Sollten sie tcddk::Get() zeitgesteuert aufrufen, vermeiden sie in jedem fall diesen Grenzwert zu unterschreiten. Eine Unterschreitung hier kann zu ungültigen Ergebnissen führen

Tcddk::MinRate ist von Gerät und Fühlerbelegung abhängig. Es ist nicht korrekt Tcddk::MinRate einmal zu ermitteln und danach als Konstante zu verwenden. Andere Geräte können durchaus ein anderes Wiederholintervall erfordern.

2.2.1 Multithreading

Multithreading ist ein Thema für Fortgeschrittene. Tcddk::Get wird aktualiserte Messdaten von Ihrem Gerät herbeischaffen. Da das einige Zeit dauern kann, wird der aufrufende Thread eingeschläfert bis das Ergebnis vom Gerät eintrifft. Eingeschläfert soll heißen, daß während dessen nur sehr wenig Rechenzeit verbraucht wird. Beachten sie bei der Verwendung mehrerer Threads, daß COM objekte in dem Thread arbeiten, der das Objekt erzeugt. Methodenaufrufe über Threadgrenzen hinweg können unerwartete Ergebnisse zeitigen.

2.3 Arbeiten mit dem Gerätespeicher

Gespeicherte Messdaten werden als Protokoll („Protocol“) abgelegt. Die Speicherorganisation der einzelnen Geräte ist auf deren jeweilige Anwendung abgestimmt und reicht von Langzeitspeichern, die nur ein einziges Protokoll verwalten können bis zu Speicherhierarchien mit Ordnerstruktur.

Um einfachen Programmiersprachen den Umgang mit dem Speicherinhalt zu vereinfachen gibt es Mengenobjekte, die jeweils artgleiche Objekte zusammenfassen. Hier sind das Tfolders, Descs und Protocols.

Die Schnittstelle „tcddk“ enthält drei Mengenobjekte (Collections) von denen je nach Gerät nur eine oder zwei zu den Protokollen führen. Die anderen Properties liefern im jeweiligen ´Fall den Wert NULL.

Tcddk::<Property>	Gültig für Geräte:	
Tfolders	T400	
Descs	Tx45, T400, T454-2000	
Protocols	T171, T174, T175-177	

2.3.1 Beispiel T177

VB

```
Dim inst As tcddk
Dim bRet As Boolean
Dim prot As Protocol
Dim key As Long

On Error GoTo ErrorLabel

Set inst = New tcddk
bRet = inst.Init(1, "testo175-177", 5000)

If True = bRet Then
    For Each prot In inst.Protocols
        key = prot.NumKey
        msg$ = Format(key, "000")
        MsgBox (msg$)
    Next
End If

GoTo EndLabel

ErrorLabel:
MsgBox Err.Description

EndLabel:
Set inst = Nothing
```

2.3.2 Beispiel Tx45

VB

```
Dim inst As tcddk
Dim bRet As Boolean
Dim desc As desc
Dim prot As Protocol
Dim key As Long
Dim pc As ProtocolCollection

On Error GoTo ErrorLabel

Set inst = New tcddk
bRet = inst.Init(1, "testo445-645-945-946-545", 5000)

If True = bRet Then
    For Each desc In inst.Descs
        Set pc = desc.Protocols
        If pc.Count > 0 Then
            For Each prot In pc
                key = prot.NumKey
                msg$ = desc.key + " " + Format(key, "000")
                MsgBox (msg$)
            Next
        End If
        Set pc = Nothing
    Next
End If

GoTo EndLabel

ErrorLabel:
MsgBox Err.Description

EndLabel:
Set inst = Nothing
```

Beachten Sie hier die Abfrage `pc.Count > 0`. Das ForEach Konstrukt sollte auf leere Mengenobjekte nicht angewandt werden.

Achten Sie auf die explizite Nullung der mit set zugewiesenen Variablen (`pc = Nothing`).

2.3.3 Beispiel T400

Speicherhierarchien mit Ordnerstruktur können rekursiv abgearbeit werden. Das Beispiel verwendet die Funktion `WalkDescs` um alle Messorte einer Hierarchiestufe aufzulisten. `WalkFolder` listet alle Messorte und Ordner einer Hierarchiestufe. `WalkFolder` wird für jede Unterordnermenge erneut

aufgerufen. Vergleichen sie dies mit einem Verfahren, dass Dateien und Verzeichnisse einer Festplatte bearbeitet. Tfolder entspricht hier dem Verzeichnis, Desc einer Datei. Analog dazu führt zum einzelnen Protokoll dann ein „Pfad“ folgenden Aufbaus: [Tfolder]* Desc NumKey.

VB

```
Private Sub WalkDescs(ADescCollection As DescCollection)
Dim ADesc As Desc
Dim AProtocolCollection As ProtocolCollection
Dim AProt As Protocol

If ADescCollection.Count > 0 Then
    For Each ADesc In ADescCollection
        MsgBox (ADesc.key)
        Set AProtocolCollection = ADesc.Protocols
        If AProtocolCollection.Count > 0 Then
            For Each AProt In AProtocolCollection
                key = AProt.NumKey
                msg$ = ADesc.key + " " + Format(key, "000")
                MsgBox (msg$)
            Next
        End If
        Set AProtocolCollection = Nothing
    Next
End If
End Sub

Private Sub WalkFolder(ATFolderCollection As TFolderCollection)

Dim ATFolder As TFolder
Dim MoreTFolders As TFolderCollection
Dim ADescCollection As DescCollection

If ATFolderCollection.Count > 0 Then
    For Each ATFolder In ATFolderCollection
        MsgBox (ATFolder.key)

        Set ADescCollection = ATFolder.Descs
        WalkDescs ADescCollection
        Set ADescCollection = Nothing

        Set MoreTFolders = ATFolder.TFolders
        WalkFolder MoreTFolders
        Set MoreTFolders = Nothing
    Next
End If

End Sub

Private Sub Ex4_Click()

Dim inst As tcddk
```

```
Dim bRet As Boolean
Dim ADescCollection As DescCollection
Dim ATFolderCollection As TFolderCollection


On Error GoTo ErrorLabel

Set inst = New tcddk
bRet = inst.Init(1, "testo400-650-950", 20000)


If True = bRet Then
    Set ADescCollection = inst.Descs
    WalkDescs ADescCollection
    Set ADescCollection = Nothing

    Set ATFolderCollection = inst.TFolders
    WalkFolder ATFolderCollection
    Set ATFolderCollection = Nothing
End If

GoTo EndLabel

ErrorLabel:
MsgBox Err.Description

EndLabel:
Set inst = Nothing

End Sub
```

Beachten Sie bitte, dass obige Beispiele stets die ForEach Anweisung benutzen, um in einer Schleife alle Mengenobjekte zu behandeln. Der einfache Grund liegt darin, dass Protocols::Item Methode einen Wert als Parameter benötigt, den man nur vom einzelnen Protokoll als Protocol::Numkey Attribut erfahren kann – ein klassisches Henne-Ei Problem. Wenn Ihr Anwender später aus einer Übersicht auswählen können soll, speichern sie am Besten beim Durchwandern des Gerätespeichers die entsprechenden Schlüsselwerte.

Sollten sie beobachten, das im Einzelfall Numkey eine triviale Sequenz der Art 0,1,2 liefert, so darf man das keinesfalls verallgemeinern. Andere Belegungen sind technisch möglich. Wenn Ihr Programm auf einer derartigen Annahme beruht wird es früher oder später u.U. nicht korrekt arbeiten.

z.B.:

```
For Each p in inst.Protocols
        Keys(i) = p.NumKey
        i = i+1;
```

Next

Das spätere „auslesen“ kann dann so aussehen
Set p = inst.Protocols.Item(Keys(selected))
p.Get();

2.4 Weitere Geräteneutrale Funktionen

Ident	Die Seriennummer Ihres Gerätes
Logfile	Ablageort und Name der Protokolldatei können konfiguriert werden. Dies sollte vor dem ersten Verbindungsaufbau geschehen.

2.5 Erweiterte Funktionen einzelner Geräte

Für verschiedene Geräte stehen weitere Funktionen an der Programmierschnittstelle bereit. Sie müssen deren Typeninformation zusätzlich zu tcddka ihrer Programmierungebung bekannt machen.

Zugang finden Sie über die Eigenschaft tcddk::Instrument. Sie liefert ein Objekt, das sie einer Variable passenden Typs zuweisen. Passender Typ heißt: Sie müssen jetzt Ihr Instrument benennen. Im folgenden Beispiel wird testo175-177 programmiert. Die Typenbibliothek ist t177a, das Instrument ist hier T177aInstrument. Andere Geräte werden hier Objekte von anderem T<xxx>aInstrument Typ zeigen. Die Zuweisung kann bei Typeninkompatibilität oder Fehlen einer detaillierten Schnittstelle fehlschlagen.

VB
```
Dim tcd As tcddk
Dim Inst As T177aInstrument
Set tcd = New tcddk
bInit = tcd.Init(0, "testo175-177", 20000)  ' use com1
If (True = bInit) Then
    Set Inst = tcd.Instrument
    ...
    Set Inst = Nothing
    tcd.Exit
End If
Set tcd = Nothing
```

2.5.1 Testo 175-177

Typenbibliothek: T177a

T177aInstrument enthält untergeordnete Objekte:

Program	T177aProg	Messprogramm
Configuration	T177aConf	Allgemeine Einstellungen wie Display an/aus
Sockets	T177aSocketCollection	Definition der „Fühler“konfiguration

Jedes dieser Unterobjekte enthält eine Methode „Save“. Damit werden geänderte Einstellungen ans Gerät übertragen.

Die meisten Änderungen erfordern für ::Save eine Unterbrechung des Messbetriebs. Achten sie auf T177aInstrument::Mode. Gültige Werte hier sind T177_WAIT, und T177_END. Verwenden sie TxxxAProg::Stop um den Messbetrieb ggf. zu unterbrechen

Logger 175-177 können nur ein einziges Protokoll verwalten. ::Save wird vorhandene gespeicherte Daten ungültig machen, d.h. sie gehen dadurch unwiederbringlich verloren. Sichern sie vor ::Save den Inhalt des Gerätespeichers.

Details siehe Online-Help „t177a.hlp“

2.5.2 Testo 174

Typenbibliothek: T174a

2.5.3 Testo 300XXL, Testo 350 M/XL

Für das Testo 300XXL, Testo 350 M/XL Messsystem steht die gerätespezifische Erweiterung t35na zur Verfügung.

Diese realisiert folgende gerätespezifische Erweiterungen.

- Es kann der Fehlerzustand des Geräts jederzeit ausgelesen werden. Zu einem Zeitpunkt können mehrere Gerätefehler vorhanden sein. Zu jedem Fehler wird die Fehlernummer und ein Fehlertext in der jeweiligen Sprachversion ausgegeben. Es werden sowohl kritische Fehler (z.B. Defekt im Gerät) als auch Warnungen (z.B. Sensor verbraucht) ausgegeben. Es ist Aufgabe des Anwendungsprogramms, darauf korrekt zu reagieren.
- Es kann ein Loggerprogramm geladen und gestartet werden.
  - Festlegen der Startart (per Befehl oder zu festgelegter Zeit)
  - Festlegen der Stoppbedingung (Speicher voll, Anzahl von Messungen, festgelegte Zeit)
  - Festlegen vom Messintervall
  - Festlegen vom Gas- und Frischluftzyklus
  - Festlegen des Brennstoffs
  - Ein- oder Ausschalten des Pellistors (wenn vorhanden)
  - Festlegen der Verdünnung
  - Festlegen der Messortbezeichnung
  - Laden des letzten Loggerprogramms, Speichern der geänderten Einstellungen
  - Starten des Loggerprogramms
  - Stoppen des Loggerprogramms
  - Abfrage des Zustands (warte auf Start, Messung läuft, Messung beendet)
- Folgende Eigenschaften vom Gerät lassen sich abfragen
  - Spannung des Akkus bzw. der Batterie. Es ist Aufgabe des Anwendungsprogramms, diese Spannung zu interpretieren, also z.B. auf eine drohende Akkuentladung zu reagieren.

  - Zur Verfügung stehender Speicher für Messdaten.
  - Bezeichnung des Bedienteils bzw. der Messbox. Diese Bezeichnung kann auch neu geschrieben werden.
  - Abfrage des Gerätetyps (300 XXL, 350 M, 350 XL, jeweils Bedienteil oder Messbox)
  - Bezeichnung der Russpumpe
- Der Gerätespeicher (Messdaten und Messorte) kann komplett gelöscht werden.

Eine ausführliche Dokumentation ist in der Datei t35na.hlp enthalten.

Die Verbindung erfolgt entweder über die serielle Schnittstelle vom PC zum Bedienteil oder unmittelbar über eine CAN Verbindung von der CAN Einsteckkarten im PC zur Messbox.

3 Hinweise zu einzelnen Entwicklungsumgebungen

3.1 Borland Delphi®

Der Import der Typelibrary [Project|Import Type Library] erzeugt eine „unit“ tcddka_TLB.pas. Der generierte Code kann Nacharbeit erfordern. Das ist ein Problem der Delphi Entwicklungsumgebung.

Ändern Sie in der Datei tcddka_TLB.pas alle fehlerhaften Funktionsaufrufe der Art DefaultInterface.Get_<xxx>() in DefaultInterface.<xxx>[] ab. Achten Sie darauf, die runden durch eckige Klammern zu ersetzen.

3.2 National Instruments LabView®

Bei Versionswechsel kann es notwendig sein, die „Library“-Einstellung aller „Property“ und „Invoke“ Knoten zu aktualisieren. Gehen Sie dazu folgendermaßen vor:

Wählen Sie auf dem Frontpanel eines Vis das „Automation Refnum“ Symbol mit der Beschriftung „tcddka.Itcddk“ aus, schneiden es aus und fügen es erneut ein. Stellen Sie dann im Kontextmenü „ActiveX Klasse auswählen“ erneut „tcddka.Itcddk“ ein.

Überprüfen Sie in der Diagrammansicht „Property“ und „Invoke“-Knoten. Im Kontextmenü (rechte Maustaste) sehen Sie entweder „Eigenschaften“ (wenn es ein Property Knoten ist) oder „Methoden“ (wenn es sich um einen Invoke Knoten handelt). Stellen Sie sicher, daß im jeweiligen Untermenü der Eintrag mit dem passenden Methoden- bzw Eigenschaftsnamen markiert ist.

3.2.1 Beispiele

Beispiele im Ordner Labview5 wurden erstellt mit Labview Version 5.1, Beispiele im Ordner Labview6 mit Version 6.1.

CAN Measure Example : Beispiel um Messwerte online aus Testogeräten (t454) via CAN-BUS auszulesen.

Serial Measure Example : Beispiel um Messwerte online aus Testogeräten via serielle Schnittstelle auszulesen.

t171 Example : Beispiel um Messwerte online aus einem t171 auszulesen.

t171_t177 Example : Beispiel um Messwerte online aus einem t171 bzw. t177 auszulesen.

t350 gas analyser serial Example : Beispiel um Messwerte seriell aus einer t350M/XL Messbox auszulesen.

Control Unit serial Example : Beispiel um Messwerte seriell aus einer t350M/XL Control -Unit auszulesen.

T454 serial Example : Beispiel um Messwerte seriell aus einer t350M/XL Messbox auszulesen.

3.2.1.1 Serielle Geräteanbindung:

Gerätebeispiel zum vorliegendem Testogerät auswählen.
COM –Schnittstelle auswählen und ggf. Waittime einstellen.
VI starten.

3.2.1.2 CAN Geräteanbindung:

Beispiel **CAN Measure Example.vi** auswählen.
CAN-BUS Adresse einstellen und ggf. Waittime einstellen.
VI starten.

3.2.1.3 In eigenes VI einbinden:

Verdrahten Sie die Icons (Init, Measure, Close), die sich in der **testo toolbox.llb** befinden, je nach gewählter Schnittstelle wie in **Serial Measure Example.vi** bzw. **CAN Measure Example.vi**. Oder kopieren Sie die jeweiligen VI-Diagramme in das eigene Diagramm und gestalten Sie die Anzeigeelementen im Frontpanel nach den eigenen Wünschen.

Eine Beschreibung der jeweiligen Methoden und Eigenschaften finden Sie in tcddk.hlp.