SDK-Programmierung bei TurboCAD™

Teil II

**Der erste Versuch eines Handbuches für den deutschen Sprachraum
etwas Licht in die Höhen und Tiefen der Makroprogrammierung (SDK-VBA o.ä)
für das CAD-Programm "TurboCAD™" zu bringen war noch lange nicht vollständig.**

**"TurboCAD™" ist eine weit verbreitete und für ihren Funktionsumfang,
sowie AutoCAD-Kompatibilität sehr preiswerte CAD-Software.**

Basis war TurboCAD 10.1, bzw. auch die neueren TC-Versionen bis TC18 …

Im Teil II sollen die bisher noch nicht behandelten Inhalte, wie z.B. die ganze 3D-Problematik, oder auch die Bemaßungs-Funktionalität eingehender untersucht werden.

Beginn der Bearbeitung im Jahre 2015, laufend ergänzt in den folgenden Jahren

Inhaltsverzeichnis

1. Vorbemerkungen
2. Berichtigungen zum I. Teil
2.1 Spline-Generierung
2.2 Layer-Verwaltung

3. Ergänzungen
3.1 Bemassung
3.2 SDK-Funktionalität „Bemaßung“
3.3 Analyse des SDK-Beispiels *AddDimensionsSample*
3.4 Die `Properties` und ihre Bedeutung im Eigenschaftsmenü
3.5 Fazit aus der DLL-Funktionalität „Bemaßung“
3.6 Alternative Bemaßungsfunktion
3.6.1 Die `Selection`-Funktionalität
3.6.2 Vorbereitende Überlegungen
3.6.3 Menü-Dialog zur Bemaßung XXX
3.6.4 Das AutoIt-Programm

4. Konverter 2D -> 3D
4.1 Zeichnungen in 2D
4.2 Voraussetzungen zur Konvertierung 2D -> 3D
4.3 Konverter-Code 2Dto3D

5. DLL-Liste und deren Funktionen

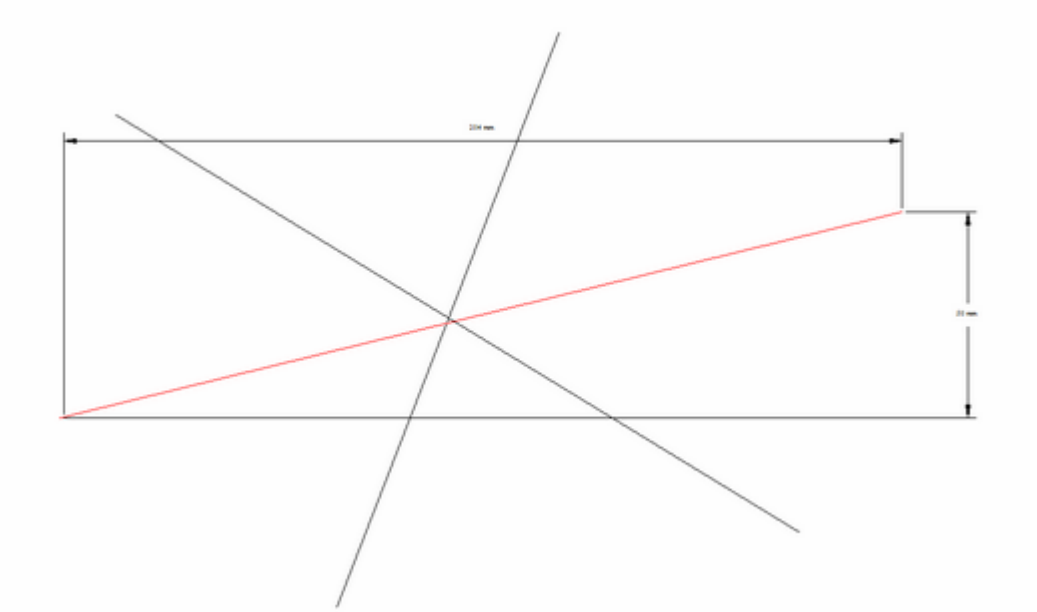
3. Ergänzungen

3.1 Bemaßung

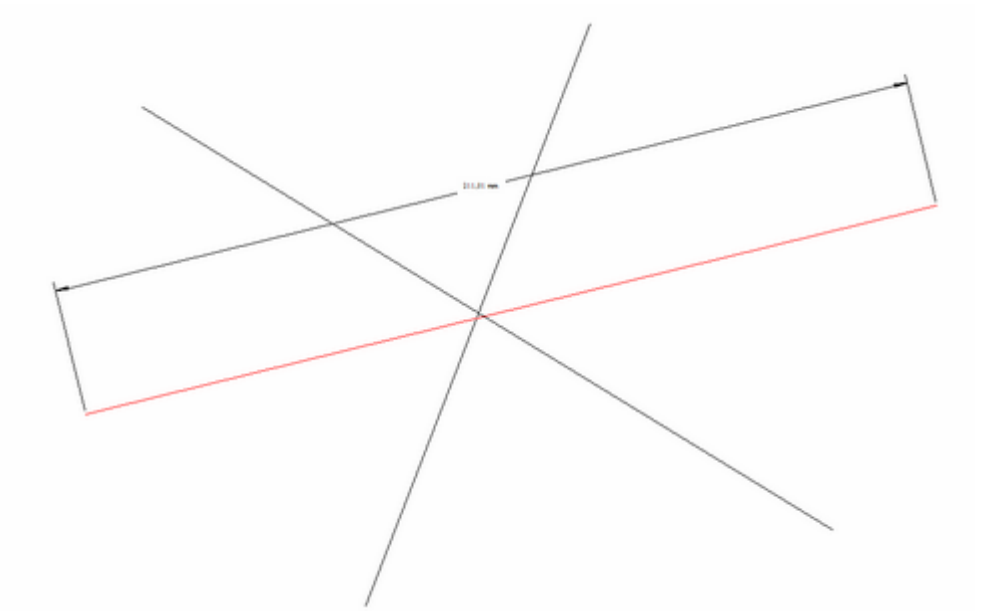
Für die Bemaßung von Grafik-Objekten gibt es in TC ein einfaches und universell verwendbares Werkzeug im Menü:

Einfügen / Bemaßung mit den Unterfunktionen:

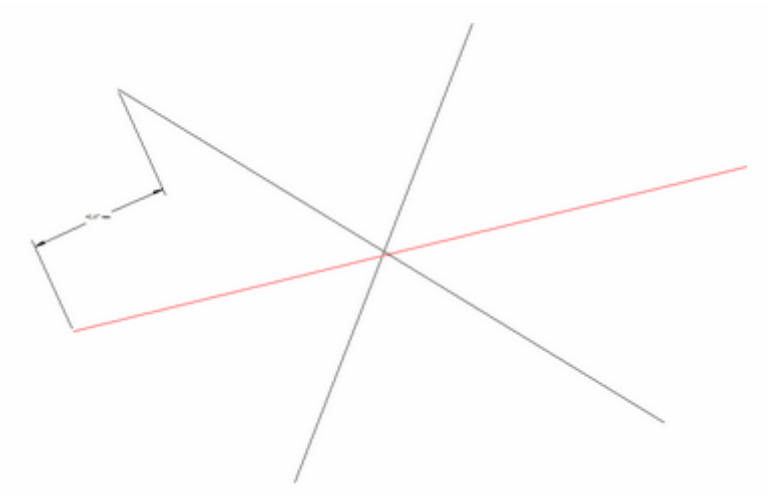
Orthogonal



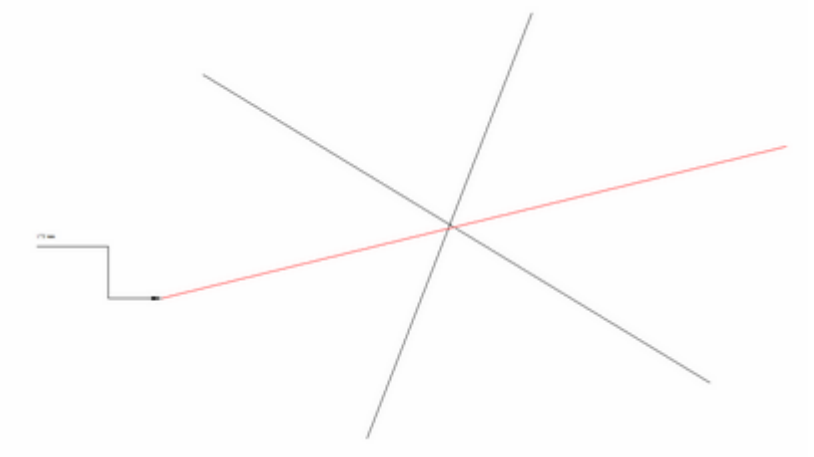
Parallel



Gedreht

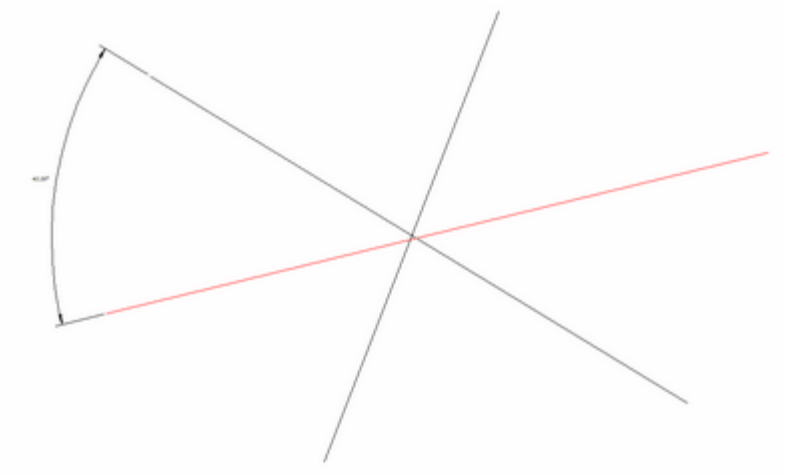


Bezugsgröße

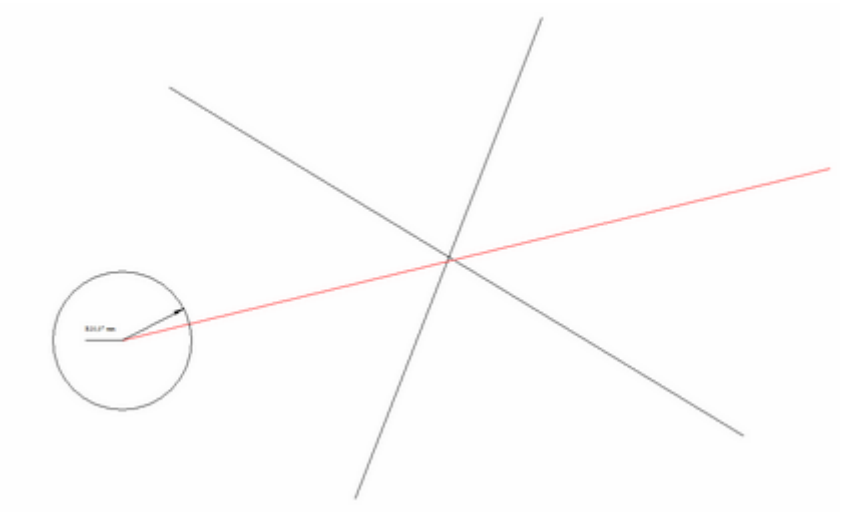


und

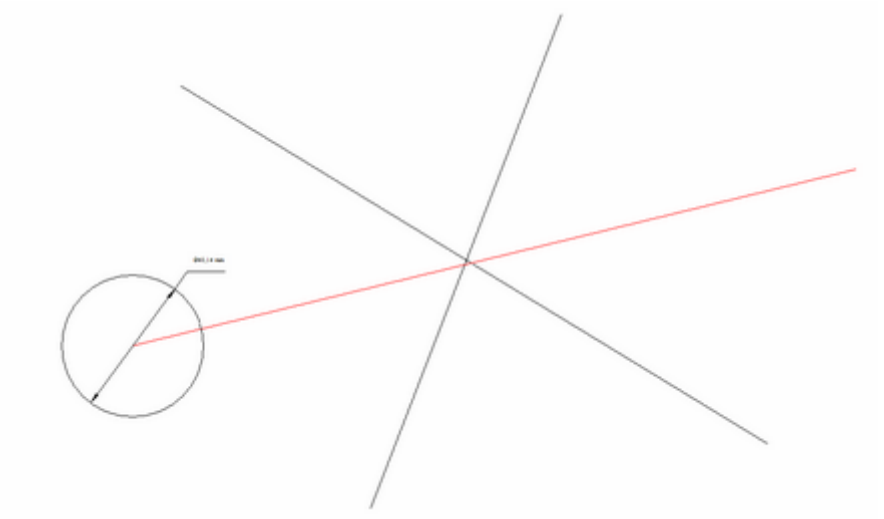
Winkel



Radius

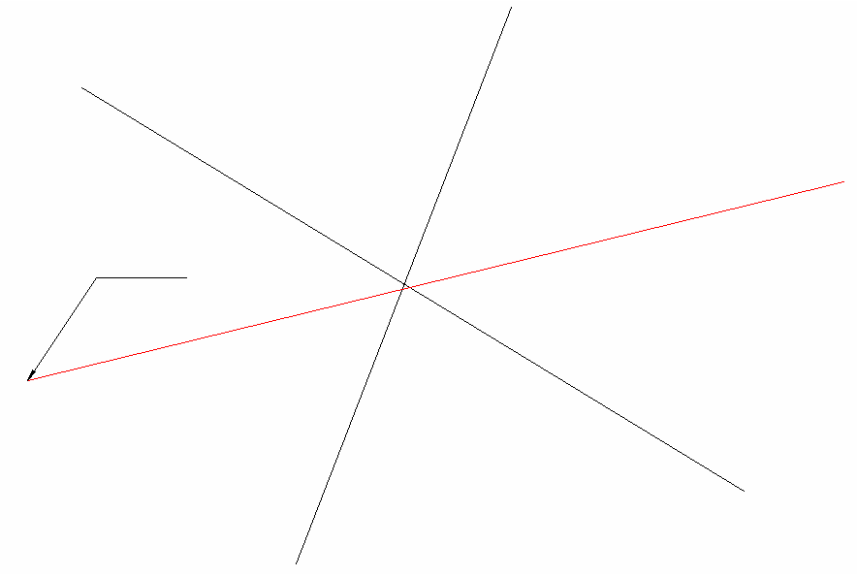


Durchmesser



bzw. allgemein

Führungslinie



3.2 SDK-Funktionalität „Bemaßung“

In der SDK-Funktionalität – Objekt-Modell von TC – wird man die dazu gehörenden Funktionen vergeblich suchen. Auch wenn man den – zugegebenermaßen ziemlich unvollkommenen - Makrorecorder bemüht, um eine diesbezügliche Zeichnungs-Sequenz aufzuzeichnen, wird man aus dem dabei aufgezeichneten Skript auch nicht viel schlauer. Der Makrorecorder ist ein Relikt aus vergangen TC-Versionen, der auch bei der neuesten (TC18) nicht mehr verbessert wurde.

Es konnte jedoch nachgewiesen werden, dass die Bemaßungs-Funktionalität in der DLL:
tcdim.dll (im *Regens*-Ordner)
steckt, die leider nicht nach den COM-Richtlinien aufgebaut ist. Somit hat man auf die einzelnen Funktionen der DLL so ohne weiteres keinen Zugriff.

Im Prinzip würde dann ggf. nichts weiter übrig bleiben, als eine Bemaßung „zu Fuß“, d.h. mit den TC-Bordmitteln (Line, Text) herzustellen. Als besondere Herausforderung erweist sich dabei die Berücksichtigung der zahlreichen Eigenschafts-Parameter. Allerdings hätte das dann auch wiederum den Vorteil, unzweckmäßige (im Eigenschaftsfenster wählbare) Parameter zu ignorieren, bzw. durch anders geartete, bessere zu ersetzen.

Nun gibt es ein Code-Beispiel (in „VBA-Beispiele TC.doc“ - *AddDimensionsSample.tcm*), wie man z.B. die Bemaßung einer Doppel-Line vornehmen kann (weil das möglicherweise zum damaligen Zeitpunkt noch nicht mit den TC-Bordmitteln möglich war):

```
Const Pi = 3.14159
Sub Main()
' this sub draw the double line and add three dimensions to it
' Horizontal, Vertical, Parallel -
  Dim Dr As Drawing
    Dim Gr As Graphic
    Dim Grs As Graphics

    Dim Verts
    Set Dr = ActiveDrawing
    Set Grs = Dr.Graphics

    Set Gr = Grs.Add(, "TCW25DblLine")
    Gr.Vertices.Add 0, 0, 0
    Gr.Vertices.Add 5, 5, 0

    Set Gr = LinearDimHorizontal(Grs, 0, 0, 5, 5, -2, True)
    Set Gr = LinearDimVertical(Grs, 0, 0, 5, 5, 2, False)
    Set Gr = LinearDimParallel(Grs, 0, 0, 5, 5, 2, False)
    Dr.Views(0).Refresh
```

```
End Sub
Private Function LinearDimHorizontal(Grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim Gr As Graphic
Dim Vi As View
Dim Vers As Vertices
Dim Ver As Vertex
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'              object and dimension line
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal"  ' "Vertical","Parallel"
H = Length

z1 = 0#
z2 = 0#

'*********************************************************************
Set Gr = Grs.Add(, "TCW25DimLin") ' Creates Linear Dimension
Gr.Properties("$DIMAUN") = 1
x0 = X2
y0 = Y1
z0 = 0#

x3 = X2
y3 = Y2 - H#
z3 = 0#

With Gr.Vertices              ' Add some vertices to Linear Dimension
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  .Add X1, Y1, z1, False, True, False, True, True, False      'V1
  .Add X2, Y2, z2, False, True, False, True, True, False      'V2
  .Add x3, y3, z3, False, True, False, True, False, False     'V3
  .Add x0, y0, z0, False, True, False, True, False, True      'V4
  .Add x0, y0, z0, False, False, False, False, False, False   'V5
End With
If (Associative) Then
    Gr.GetSubjectLink 1       'Link with dimension's Vertex V1
    Gr.GetSubjectLink 2       'Link with dimension's Vertex V2
End If
Set LinearDimHorizontal = Gr


End Function

'*********************************************************************

Private Function LinearDimVertical(Grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim Gr As Graphic
```

```
Dim Vi As View
Dim Vers As Vertices
Dim Ver As Vertex
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'              object and dimension line
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal"  ' "Vertical","Parallel"
H = Length

z1 = 0#
z2 = 0#

'*********************************************************************
Set Gr = Grs.Add(, "TCW25DimLin")   ' Creates Linear Dimension
Gr.Properties("$DIMAUN") = 1
x0 = X1
y0 = Y2
z0 = 0#

x3 = X2 + H
y3 = Y2
z3 = 0#

With Gr.Vertices                        ' Add some vertices to Linear Dimension
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  .Add X1, Y1, z1, False, True, False, True, True, False      'V1
  .Add X2, Y2, z2, False, True, False, True, True, False      'V2
  .Add x3, y3, z3, False, True, False, True, False, False     'V3
  .Add x0, y0, z0, False, True, False, True, False, True      'V4
  .Add x0, y0, z0, False, False, False, False, False, False   'V5
End With
If (Associative) Then
    Gr.GetSubjectLink 1             'Link with dimension's Vertex V1
    Gr.GetSubjectLink 2             'Link with dimension's Vertex V2
End If
Set LinearDimVertical = Gr

End Function
Private Function LinearDimParallel(Grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim Gr As Graphic
Dim Vi As View
Dim Vers As Vertices
Dim Ver As Vertex
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'              object and dimension line
```

```
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal"  ' "Vertical","Parallel"
H = Length

z1 = 0#
z2 = 0#

'*********************************************************************
Set Gr = Grs.Add(, "TCW25DimLin") ' Creates Linear Dimension
Gr.Properties("$DIMAUN") = 1
x0 = X2
y0 = Y2
z0 = 0#

Dim L#, sina#, cosa#
Dim x3Loc#, y3Loc#
L = Sqr((X1 - X2) * (X1 - X2) + (Y1 - Y2) * (Y1 - Y2))
sina = (Y2 - Y1) / L
cosa = (X2 - X1) / L
x3Loc = L
y3Loc = -H

x3 = X1 + x3Loc * cosa - y3Loc * sina
y3 = Y1 + x3Loc * sina + y3Loc * cosa
z3 = 0#

With Gr.Vertices                ' Add some vertices to Linear Dimension
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  .Add X1, Y1, z1, False, True, False, True, True, False      'V1
  .Add X2, Y2, z2, False, True, False, True, True, False      'V2
  .Add x3, y3, z3, False, True, False, True, False, False     'V3
  .Add x0, y0, z0, False, True, False, True, False, True      'V4
  .Add x0, y0, z0, False, False, False, False, False, False   'V5
End With

If (Associative) Then
    Gr.GetSubjectLink 1       'Link with dimension's Vertex V1
    Gr.GetSubjectLink 2       'Link with dimension's Vertex V2
End If
Set LinearDimParallel = Gr

End Function
```

Das funktioniert auch unter TC10 noch einwandfrei – es wird ein Stück schräge Parallel-Line gezeichnet, die mit horizontaler, vertikaler und im schrägen Winkel der Parallel-Linie verlaufenden Bemaßungen versehen ist.

Eine Analyse des Codetextes und Verallgemeinerung stehen noch aus – siehe *Test1.tcw*.

Anfrage im CAD/IMSI-Forum *http://ww3.cad.de/foren/ubb/Forum229/HTML/000052.shtml*

Forum-Text:

Nachdem nun 3 Monaten lang sich noch niemand bemüßigt gefühlt hat sich diesem Thema zu widmen, habe ich selbst versucht dem "Geheimnis" auf die Spur zu kommen.
Wie bereits in meinem SDK-Buch zum Thema "Datenbank" ausgeführt, wird offensichtlich auch die Bemaßungsfunktion (und wie auch noch viele Weitere) außerhalb der trivialen Grafikfunktionen, welche im SDK dokumentiert sind, realisiert.
Dazu sind spezielle DLLs in das Hauptprogramm eingebunden, die bei Bedarf, d.h. beim Aufrufen der entsprechenden Menüfunktion, in Aktion treten.
Für die Datenbankfunktion ist das z.B. die tcdb.dll - habe ich getestet, in dem diese vorübergehend umbenannt wurde. Schon ist die DB-Funktion nicht mehr verfügbar.
Die für die Bemaßungsfunktion zuständige DLL habe ich bis jetzt noch nicht ausfindig machen können - wird aber im Rahmen des 2.Teil vom SDK-Buch geschehen.
Diese DLLs sind in der Regel nicht COM-fähig und somit leider auch in keinster Weise dokumentiert, so dass ein Aufruf ohne die speziellen Kenntnisse der Aufrufparameter scheitern wird.

Für die Realisierung der Bemaßungsfunktion "zu Fuß" gibt es in den SDK-Beispielen einen VBA-Sample: *AddDimensionsSample.tcm* - konnte ich aber aus Zeitgründen bisher noch nicht testen.
Mal sehen, wenn ich dazu komme am 2.Teil weiter zu Arbeiten ...
Grüsse aus Berlin

PSblnkd

3.3 Analyse des SDK-Beispiels *AddDimensionsSample*

Zur detaillierten Analyse und zur besseren Übersicht wurde in *AddDimensionsSample.tcm* zunächst nur die erste Unterfunktion:

```
LinearDimHorizontal(Grs As Graphics, X1 As Double, Y1 As Double, X2 As Double, Y2 As Double, Length As Double, Associative As Boolean)
```

aktiv beibehalten und die anderen beiden:

– `LinearDimVertical(Grs As Graphics, X1 As Double, Y1 As Double, X2 As Double, Y2 As Double, Length As Double, Associative As Boolean)` und

- `LinearDimParallel(Grs As Graphics, X1 As Double, Y1 As Double, X2 As Double, Y2 As Double, Length As Double, Associative As Boolean)`

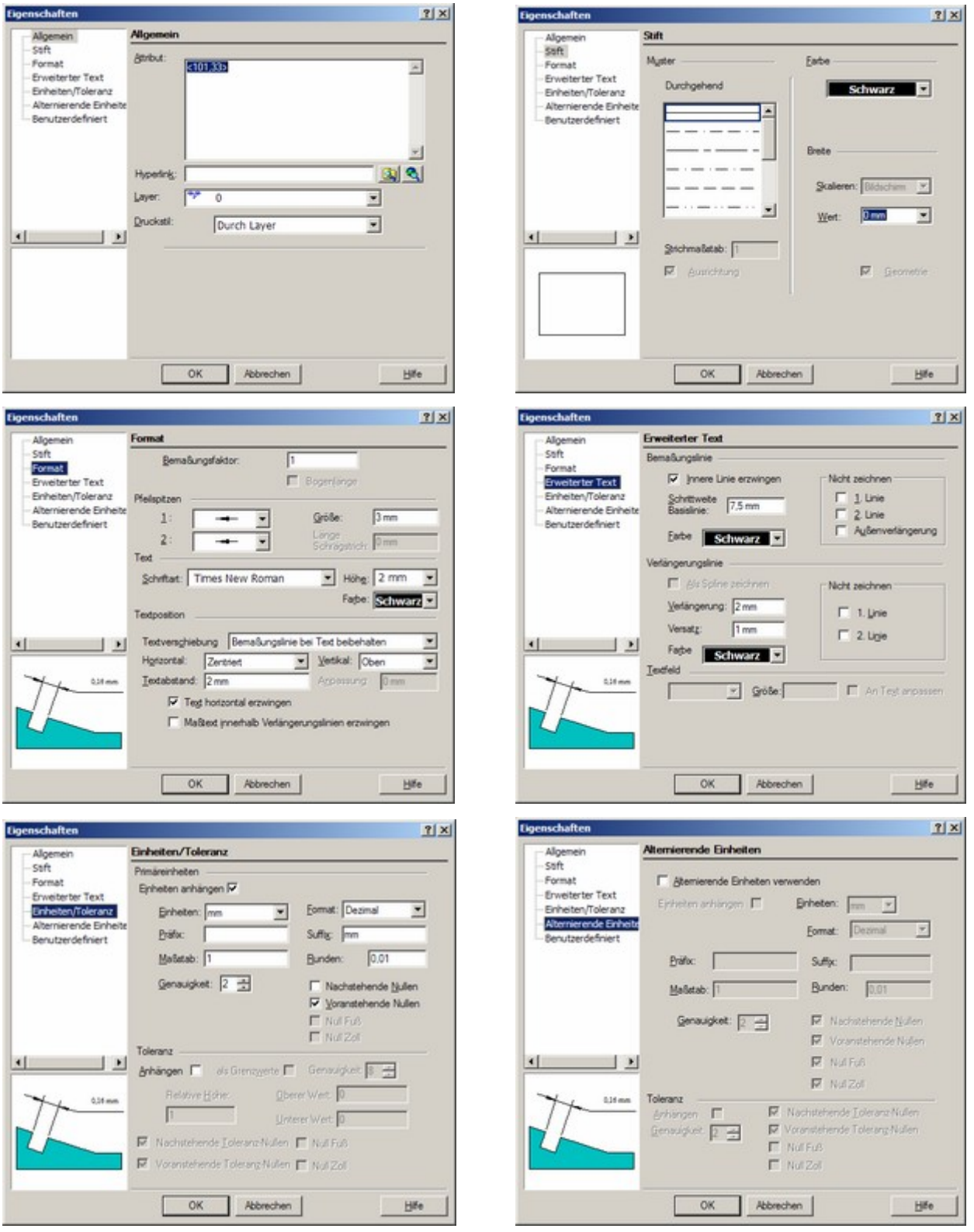
im Hauptprogramm `sub main()` auskommentiert, sowie die Generierung einer Doppellinie durch eine solche für `SingleLine` ersetzt.

Das Hauptprogramm `sub main()` ist sonst nicht weiter interessant – hier wird zunächst das zu bemaßende Grafik-Objekt (Doppel-Linie oder `SingleLine`) generiert, um dann für die verschiedenen Bemaßungs-Varianten entsprechende Unterfunktion aufzurufen.
An die betreffende Unterfunktion werden vom Hauptprogramm verschiedene Parameter übergeben:

- `grs` als Grafik-Collection
- `X1, Y1` und `X2, Y2` die Maß-Parameter
- `Length` als Länge der Maß-Hilfslinien und
- `Associative` als Boole'scher Wert (Wirkungsweise noch unbekannt – siehe unten)

Als Rückgabe erfolgt jeweils das Bemaßungs-Objekt `gr`, was mit einem `View.Refresh` dann auch vollständig dargestellt werden soll, was aber bereits nach Ausführung von `V3` geschieht.

Weitere Parameter-Übergaben sind hier offensichtlich nicht vorgesehen, wobei doch das umfangreiche Eigenschaften-Menü der Bemaßungsfunktion etwas anderes erwarten lässt:



Parameter-Einstellungen für die Bemaßungsfunktion

Offensichtlich müssen die vielfältigen Parameter-Einstellungen doch noch wo anders abgelegt werden.
Später dazu mehr …

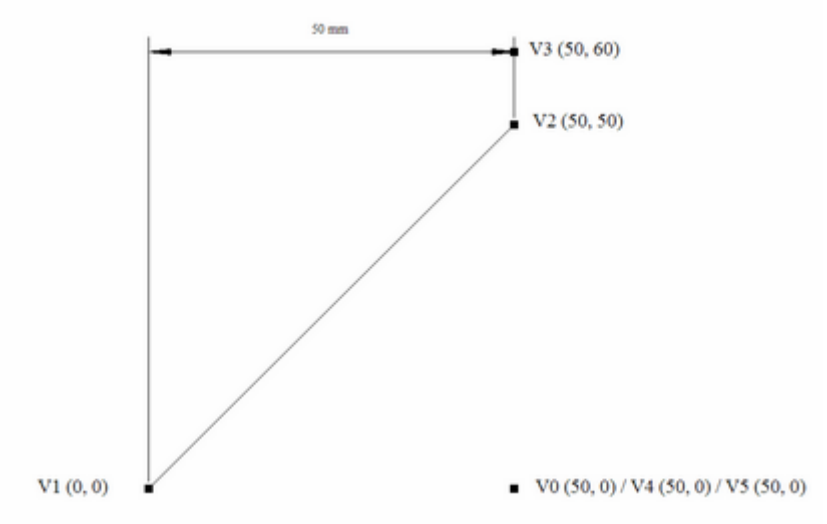
Das eigentlich Interessante jeder Unterfunktion steckt jedoch in dem Aufruf:

```
Set gr = grs.Add(, "TCW25DimLin")   ' Creates Linear Dimension
```

Hier wird offensichtlich doch schon die Funktionalität von *tcdim.dll* bemüht und mit den weiter unten beschriebenen `Vertices`-Array die Aufrufparameter festgelegt.
Mit dem Hinzufügen des Objekts `TCW25DimLin` werden automatisch 80 (!) neue `Properties` (Objekteigenschaften) zu den bereits standardmäßig in TC10 vorhandenen 119 hinzugefügt (in den höheren TC-Versionen ist standardmäßig die Anzahl noch viel größer).
Deren Bedeutung, bzw. Auswirkung ggf. im Zusammenhang mit dem Eigenschafts-Menü ist im Einzelnen noch zu klären.
Im Skript wird als nächstes der `Property (Item157)`:

```
gr.Properties("$DIMAUN") = 1  ' = wahr
```

zugewiesen – Auswirkung noch unklar – d.h. bisher keine …
Dann kommt entsprechend der Definition der 6 notwendigen `Vertices` (Eckpunkte) von `V0` bis `V5` die Berechnung und Zuweisung mit `Add` innerhalb eines `With – EndWith`-Statements.
Bereits nach dem 4. `Add` ist die Bemassung zu sehen.

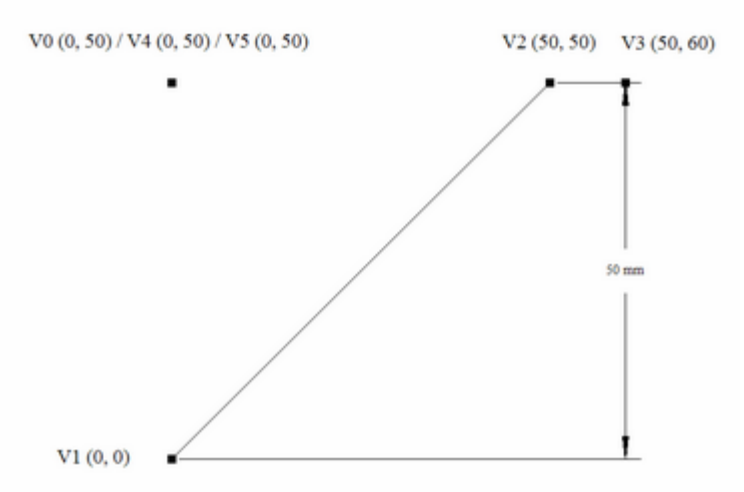


Mit dem Übergabeparameter `Length` wird hier die vertikale Lage von `V3` beeinfusst.
Welche Auswirkungen verschiedene Parameter-Modifikationen der `Vertices V0 … V5` haben, muss noch geprüft werden – z.B.:

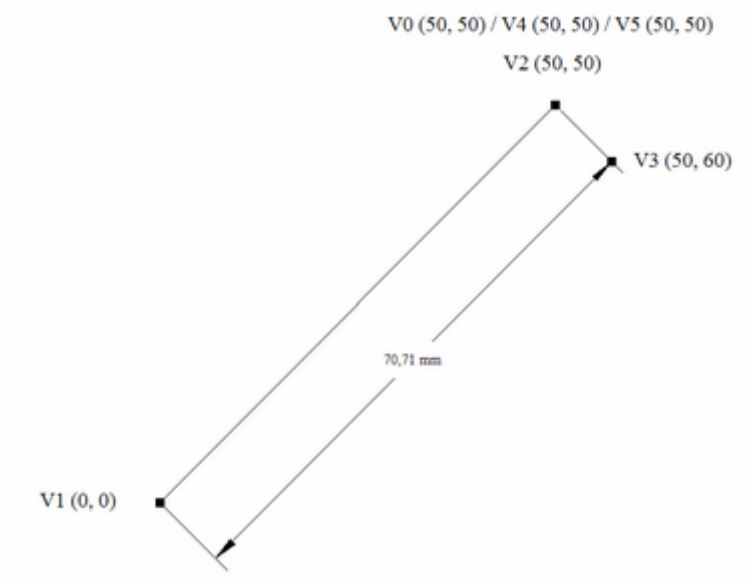
Wenn die für die vertikale Vermassung zuständige Unterfunktion `LinearDimVertical` aufgerufen wird, so werden die `Add`-Zuweisungen für die `Vertices V0, V4` und `V5` wie folgt modifiziert:

`x0 = 0, y0 = 50` und `V3` demzufolge angepasst:

`x3 = 60, y3 = 50`



Für die Unterfunktion `LinearDimParallel` müssen dann zunächst umfangreichere Berechnungen vorgenommen werden, um die richtigen Parameter für die `Add`-Zuweisungen für die `Vertices` bereitstellen zu können:



Besonders Funktions-bedeutsam sind aber die beiden letzten Befehle:

```
gr.GetSubjectLink 1        'Link with dimension's Vertex V1
gr.GetSubjectLink 2        'Link with dimension's Vertex V2
```

welche mit `If-Then` nur dann aktiv sind, wenn die Variable `Associative =` „Wahr“ ist.

Funktions-bedeutsam deshalb, weil damit eine Link-Verbindung des Bemaßungsobjekts mit dem zu vermaßenden Grafikobjekt hergestellt wird. Das bedeutet, dass bei Veränderungen am Grafikobjekt automatisch auch die Vermassung angepasst wird.
Mit der Übergabe des `Associative`-Parameters wird also entschieden, ob die Bemaßung mit dem Grafik-Objekt verknüpft ist, oder nicht. In der TC-Originalfunktion zur Bemaßung ist das immer der Fall.

Als besonders wichtige Information aus dem Beispiel-Skript *AddDimensionsSample.tcm* können die Zugriffsregeln für die DLL *tcdim.dll* entnommen werden.
Dort ist vermerkt (deutsche Übersetzung):
Die Funktion “Dimension“ hat 6 Basis-Scheitelpunkte:
'V0 – nicht sichtbar - definiert die Richtung der Dimension
'V1 - sichtbar, editierbare, verknüpfbar
'V2 - sichtbar, editierbare, verknüpfbar
'V3 - sichtbar, editierbar - definiert den normalen Abstand zwischen dem zu vermaßenden Objekt und Maßlinie
'V4 - sichtbar, editierbar - definiert den Ort des Bemaßungstextes
'V5 - unsichtbar, nicht editierbar - definiert den Ort des Bemaßungstextes

Mit folgender Übersicht soll versucht werden die Zusammenhänge umfassend darzustellen und im Einzelnen zu erklären:

	Koordinaten-Parameter								
Vertice	**X**	**Y**	**Z**	**PenDown**	**Selectable**	**Snappable**	**Editable**	**Linkable**	**Calculated**
0	x	y	z	False	False	False	False	False	False
1	x	y	z	False	True	False	True	True	False
2	x	y	z	False	True	False	True	True	False
3	x	y	z	False	True	False	True	False	False
4	x	y	z	False	True	False	True	False	True
5	x	y	z	False	False	False	False	False	False

Die Anzahl und Bezeichnung der Parameter ist von der `Add`-Funktion abgeleitet, wobei der letzte Parameter (`Befor` und `After`) hier nicht verwendet werden.

Zu Vertice 0:

Wenn für den x-Wert `xEnd` und für den y-Wert `yAnf` eingesetzt wird, erfolgt eine horizontale Bemaßung. Der z-Parameter wird immer auf "0" gesetzt.

Hier ist aber zu beachten, dass der `mLaeng`-Parameter für die Länge der Bemaßungslinien nicht mit negativen Vorzeichen zu verwenden ist, ansonsten die zweite Bemaßungslinie nach unten gezeichnet wird und somit die Maßpfeillinie direkt auf dem Objekt zu liegen kommt. Die erste Bemaßungslinie ist davon nicht betroffen, d.h. ob und wie die Richtung der ersten Bemaßungslinie zu beeinflussen ist, muss noch geklärt werden.

Eine vertikale Bemaßung wird hingegen gezeichnet, wenn für den x-Wert `xAnf` und für den y-Wert `yEnd` eingesetzt wird. Hier ist aber zu beachten, dass der `mLaeng`-Parameter für die Länge der Bemaßungslinien nicht mehr mit negativen Vorzeichen zu verwenden ist, da ansonsten die zweite Bemaßungslinie nach links gezeichnet wird und somit die Maßpfeillinie direkt auf dem Objekt zu liegen kommt. Die erste Bemaßungslinie ist davon nicht betroffen, d.h. ob und wie die Richtung der ersten Bemaßungslinie zu beeinflussen ist, muss noch geklärt werden.

Bei der Objekt-parallelen Vermassung wird für den x-Wert `xEnd` und für den y-Wert `yEnd` eingesetzt. Die Bemaßung erfolgt unterhalb des Objektes. Im Gegensatz zur horizontalen und vertikalen Vermassung wird hier bei Verwendung eines negativen Vorzeichens für den `mLaeng`-Parameter die Vermassung komplett, d.h. mit beiden Bemaßungslinien nach oben gezeichnet.

Eine Variation der Bool'schen Werte hat keinerlei Einfluss auf die Funktion, d.h. die können auch weggelassen werden.

Zu Vertice 1 + 2:

In den x- und y-Werten werden die betreffenden `xAnf / yAnf`, bzw. `xEnd / yEnd` des zu vermaßenden Objekts eingetragen.

Obwohl im Gegensatz zu *Vertice 0* einige der Bool'schen Werte auf `Wahr` gesetzt sind, konnte zunächst ein Einfluss bei einer diesbezüglichen Variation nicht festgestellt werden, aber im Zusammenhang mit den nachfolgenden `Vertices` dann schon.

Im Einzelnen:

Der erste Bool'sche Parameter `PenDown` hat gar keinen Einfluss.

Auch beim zweiten `Selectable`, vierten `Editable` und fünften `Linkable` sind zunächst keine Veränderungen zu sehen. Werden hingegen die Bool'schen Parameter weggelassen, so

wie bei `V0` möglich, so wird bereits beim Abarbeiten von `V2` eine waagerechte Linie, beginnend von `xAnf/yAnf` gezeichnet, die offensichtlich nichts mit der Bemaßung zu tun hat.

Zu Vertice 3:

Für den `x`-Wert wird `xEnd` und für den `y`-Wert `yEnd` – `mLaeng` eingetragen.
Werden beim zweitenBool'schen Parameter `Selectable` und vierten `Editable` `Wahr`-Werte eingetragen sind keine Veränderungen zu sehen. Das trifft auch zu, wenn alle Bool'schen Parameter weggelassen werden (wie in `V0`).

Zu Vertice 4:

Die `x`- und `y`-Parameter sind die gleichen wie bei `V0`.
Nach der Parameterbeschreibung sollte eigentlich damit die Text-Platzierung eingestellt werden können, aber eine solche Funktionalität konnte nicht beobachtet werden.
Bei den Bool'schen Parametern können `Selectable` und `Editable` verändert werden, ohne dass eine sichtbare Veränderung eintritt. Das trifft auch zu, wenn alle Bool'schen Parameter weggelassen werden (wie in `V0`). Dann allerdings tritt eine Beeinflussung der Text-und Maßpfeil-Platzierung bei der Variation der `x`- und `y`-Parameter ein.
Das hängt ursächlich mit dem letzten Bool'schen Parameter `Calculated` zusammen, wenn der auf `Falsch` gesetzt wird ist die Text-/Maßpfeil-Platzierung aktiviert.

Zu Vertice 5:

Die `x`- und `y`-Parameter sind die gleichen wie bei `V0`.
Es tritt keine Veränderung ein, wenn alle Bool'schen Parameter weggelassen werden – mit einer Ausnahme:
`V4[Calculated] = Falsch ,`
dann werden auch die `x`- und `y`-Parameter relevant.

3.4 Die `Properties` und ihre Bedeutung im Eigenschaftsmenü

Wie bereits unter 3.3 beschrieben, kommen mit der Etablierung eines neuen Grafikobjektes für die Bemaßung 80 neue `Properties` (Eigenschaften) hinzu. Da aus den Namen – siehe Tabelle in 3.5 – so ohne weiteres nicht auf die korrespondierenden Eintragungen im Eigenschaftsmenü (3.3) geschlossen werden kann und eine manuelle Untersuchung zu langwierig wäre, muss mittels eines speziellen Hilfsprogramms versucht werden, das betreffende `Property` zu finden, welches im Eigenschaftsmenü des Bemaßungsobjektes dafür zuständig ist.

Dazu sollen zunächst alle 80 `Properties` in einem Array abgelegt, d.h. zwischengespeichert werden und dann bei einer Veränderungen im Eigenschaftsmenü das oder die betreffende(n) `Property / Properties` herausgesucht werden.

Der Code für das Ablegen in einem Array könnte dann so aussehen:

```
Sub PropertyVerifiation()
'untersucht die 80 neuen Properties auf ihre Korrespondenz im Eigenschaftsmenü
    Dim app As Application
    Dim drw As Drawing
    Dim grs As Graphics
    Dim gr As Graphic
    Dim sel As Selection
    Dim array1(80)
    Dim txt As Variant
    Set app = IMSIGX.Application
    Set drw = app.ActiveDrawing
    Set grs = drw.Graphics

    Set gr = grs.Add(, "TCW25DimLin")    ' Creates Linear Dimension –

    For i = 0 To 78
        txt = gr.Properties(120 + i)
        array1(i) = txt
    Next

End Sub
```

Damit wären die Standard-Werte im `Array1` abgelegt, die jedoch nur solange existieren wie das Programm läuft. Ist `End Sub` abgearbeitet, ist auch das `Array1` mit all seinen Inhalten verschwunden. Wie kann man diesen Umstand umgehen?

Am Einfachsten ist es vor dem `For` eine Marke zu setzen und nach dem `Next` einen Sprung auf die Marke mit `GoTo` programmieren, so dass sich eine Endlos-Schleife ergibt:

```
Loop1:
    For i = 0 To 78
        txt = gr.Properties(120 + i)
        array1(i) = txt
    Next
GoTo Loop1                    'muss Loop1 benannt werden wegen Schlüsselwort Loop
```

Damit das Programm nicht endlos läuft, wird auf die GoTo-Anweisung ein Haltepunkt gesetzt – Achtung! – Der Haltepunkt muss jedes Mal neu gesetzt werden, wenn der VBA-Editor geschlossen und wieder neu gestartet wird.

Wird jetzt das Eigenschaftsmenü aufgerufen und z.B. in der Untereigenschaft *Attribut* das noch leere Feld mit einem Eintrag versehen – z.B. “Grafik_1“, so müsste dieser auch in irgendeinem Property auftauchen.
Es wird eine erweiterte zweite For-Next-Schleife programmiert und die Sprunganweisung dementsprechend verändert, dass sie nicht auf die Erstbefüllung des Array1 zeigt:

```
For i = 0 To 78                              ‘Erstbefüllung von array1
        txt = gr.Properties(120 + i)
        array1(i) = txt
    Next
Loop1:
    For i = 0 To 78
      ' nach Bedienung "Eigenschaftsmenü" Veränderungen in den Properties suchen

        If gr.Properties(120 + i) <> array1(i) Then
            nameI = gr.Properties(120 + i).Name
            valueI = gr.Properties(120 + i).Value
            MsgBox ("Name =" & nameI & vbCrLf & "Wert :" & valueI)
        End If
    Next
GoTo Loop1
```

Gleich beim ersten Versuch mit o.g. Untereigenschaft *Attribut* findet die For-Next-Schleife nichts. Wahrscheinlich liegt die *Attribut* -Eigenschaft in einer anderen Property, als in den neu hinzu gekommenden. Wird die Schleife auf

```
For i = 0 To 198
```

erweitert, dann ergibt sich allerdings das Problem, dass nicht alle 199 Properties auch existieren, was sich im Laufzeit-Fehler: `Builtin : <Internal DBAPI failure.>` bemerkbar macht. Diese gilt es vorher abzufangen mit:

```
On Error Resume Next     ' Fehlerbehandlung zurückstellen, nächste Anweisung ausführen.
```

Außerdem sind die Array-Zuweisung und der Vergleich noch nicht richtig, so dass nun der Code wie folgt aussieht (nur das Relevante):

```
For i = 0 To 198                        ' Erstbefüllung von array1
        On Error Resume Next            ' für den Fall, dass die Eigenschaft nicht existiert
        txt = grD.Properties(i).Value
        array1(i) = txt
    Next

Loop1:       ' nach Bedienung im "Eigenschaftsmenü" Veränderungen in den Properties suchen
For i = 0 To 198                         ‘ hier Haltepunkt setzen!
    On Error Resume Next
    nrD = i + 1                          ' da NrD sonst mit "0" beginnt
    nameI = grD.Properties(i).Name       ' Name und Value separat ermitteln
```

```
    valueI = grD.Properties(i).Value
    arrI = array1(i
    If valueI <> arrI Then
       MsgBox ("NrD : " & nrD & vbCrLf & "Name = " & nameI & vbCrLf &
                                          "Wert = " & valueI)     'Ausgabe
    End If
Next

GoTo Loop1
```

Werden die Variablen in der `If-Then`-Abfrage ermittelt, kommt es zu falschen Ergebnissen, was mit der Fehler-Abfangung `On Error Resume Next` zusammenhängt.
Nun kann die eigentliche Untersuchung beginnen.
Nach ersten Variationen am selektierten Grafik-Objekt sind erst einmal gar keine Veränderungen bei den `Properties` zu beobachten – warum?
Weil sich die 199 Properties gar nicht auf das selektierte Grafikobjekt beziehen, sondern auf das bereits angelegte, aber noch nicht wirklich existierendes `Dimension`-Objekt `grD`.
Wird der Objektbezug auf das selektierte Grafikobjekt `gr` geändert, so ergibt sich Folgendes:
Eine Veränderung z.B. bei `NrD 12 Name: Info` wird erkannt, aber bei den nachfolgenden „leeren“, d.h. mit `On Error Resume Next` übergangenen `Properties` wird der `Value`-Wert beibehalten, so dass es zu falschen Aussagen kommt.
Behoben werden kann das Problem mit einer entsprechenden `Nothing`-Zuweisung nach der `MsgBox`-Ausgabe, ebenso bei der `Array1`-Zuweisung:

```
ValueI = Nothing bzw.
txt = Nothing
```

Es gibt zwar immer noch ein paar ungeklärte `Property`-Veränderungen (z.B. `NrD = 9: TextStyle` und `NrD = 13: TextFont`, sowie einige nicht erkannten Veränderungen von `“leer“ -> “0“` – z.B. bei `NrD = 24: TextObliquity -` das kann aber im Weiteren berücksichtigt werden.

Zur Untersuchung der `Properties` für das Bemaßungsobjekt wird

```
’Set gr = grs.Add(, "TCW25DimLin")    ' Creates Linear Dimension –
                                      ’ wird hier erst mal noch nicht gebraucht
```

auskommentiert und dafür ein mit TC-Bordmitteln gezeichnetes Bemaßungs-Objekt selektiert-markiert.
Die Ergebnisse der Untersuchungen sind in der Tabelle 1: “Parameter für die Bemaßungsfunktionen ($...)“ in 3.5. eingetragen.

3.5 Fazit aus der DLL-Funktionalität „Bemaßung“

Alles in Allem muss von einer recht komplizierten und unübersichtlichen Funktionalität der Parameterübergabe-Schnittstelle der *tcdim.dll* ausgegangen werden.

Hierbei ist noch gar nicht untersucht worden, wie die anderen Bemaßungs-Varianten angesprochen werden sollen.

Nach dem verbalen Inhalt – mittels Hex-Editor und PE-Browser ermittelt – sind in der *tcdim.dll* außer der `TCW25DimLin`-Funktionalität auch noch Hinweise zu:

- `TCW25DimAng`
- `TCW25DimDia`
- `TCW25DimRad`
- `TCW25DimDat` und
- `TCW25DimLead`

zu finden.

Ob diese so einfach angesprochen werden können, wie das im `TC25DimLin`-Beispiel *AddDimensionsSample.tcm* dokumentiert wurde, bleibt noch zu untersuchen.

In nachfolgender Tabelle werden die 80 `Properties` aufgeführt und wie sie möglicherweise im Zusammenhang mit den im Eigenschaften-Menü eingetragenen Parametern stehen.

Alle beginnen mit `$`:

Tabelle 1: Parameter für die Bemaßungsfunktionen ($...)

ItemNr	Name	Type	Value	Eigenschaftsmenü	
119	$DIMASO	2	0		
120	$DIMASZ	5	3	Pfeilspitzengröße	
121	$DIMSTYLE	2	0		
122	$DIMBLK	2	1		
123	$DIMBLK1	2	1	1.Pfeilspitze	
124	$DIMBLK2	2	1	2.Pfeilspitze	
125	$DIMCLRD	3	0	Maßlinienfarbe	
126	$DIMCLRE	3	0		
127	$DIMCLRT	3	0	Maßzahlfarbe	
128	$DIMDLE	5	0		
129	$DIMEXE	5	2	Verlängerungsl. Länge	
130	$DIMEXO	5	1		
131	$DIMGAP	5	2	Textabstand	
132	$DIMSCALE	5	1	Bemaßungsfaktor	
133	$DIMSD1	2	0	1.Linie nicht zeichnen	

134	$DIMSD2	2	0	2.Linie nicht zeichnen	
135	$DIMSE1	2	0	1.Verl. nicht zeichnen	
136	$DIMSE2	2	0	2.Verl. nicht zeichnen	
137	$DIMSOXD	2	0	Außenverläng. nicht	
138	$DIMTAD	2	1	Textposition vertikal	
139	$DIMJUST	2	0	Textposition horiz.	
140	$DIMTFAC	5	1	rel. Höhe Toleranz	
141	$DIMTIH	2	1	Text horiz. erzwingen	
142	$DIMTIX	2	0	Text innerhalb Verl.	
143	$DIMTP	5	0	oberer Wert Toleranz	
144	$DIMTM	5	0	unterer Wert Toleranz	
145	$DIMTOFL	2	1	innere Linie erzwingen	
146	$DIMTOH	2	0	Text innerhalb Verl.	
147	$DIMTOL	2	0	Toleranz anhängen	
148	$DIMTSZ	2	0		
149	$DIMTVP	5	0	Textposition / Anpass.	
150	$DIMTXT	5	2	Schrifthöhe	
151	$DIMZIN	2	8	Vor-/nachstehende Nullen	
152	$$ArrowInside	2	0	Pfeilspitzen innen/außen (berechn.)	
153	$DIMDIL	5	7,5	Schrittweite Basisl.	
154	$DIMLFAC	5	1	Maßstab (berechn.)	
155	$DIMRND	5	0,01	Runden	
156	$DIMAUN	2	1	Einheiten anhängen	
157	$DIMTBST	2	0		
158	$DIMTBSZ	5	5		
159	$DIMTBFT	2	0		
160	SDIMTOLLIM	2	0	Toleranz als Grenzw.	
161	$DIMPOST	8	<>mm	Text, Einheiten vor-/anhängen	
162	$DIMASSPLINE	2	0		
163	PenColor	3	0		
164	ExplodeScala	5	1		
165	$DIMARCLENGTH	2	0		
166	$DIMOBLIQUEANGLE	5	0		
167	$DIMLUNIT	2	2		
168	$DIMUNIT	2	0	Einheiten-Format	
169	$DIMFRAC	2	0		
170	$DIMDEC	2	2	Genauigkeit	
171	$DIMDSEP	8	“”		

172	$DIMAZIN	2	0		
173	$DIMADEC	2	2		
174	$DIMAUNIT	2	0		
175	$DIMCEN	5	0,25		
176	$DIMSAH	2	0		
177	$DIMATFIT	2	3		
178	$DIMLDRBLK	8	“0“		
179	$DIMUPT	2	0		
180	$DIMTXSTY	8	“STANDART“		
181	$DIMTMOVE	2	0	Textposition / Textverschiebung	
182	$DIMFIT	2	3		
183	$DIMSHO	2	1		
184	$DIMLWD	2	4		
185	$DIMLWE	2	4		
186	$DIMTDEC	2	8	Toleranz Genauigkeit	
187	$DIMTOLJ	2	1		
188	$DIMTZIN	2	0		
189	$DIMALTU	2	2		
190	$DIMALT	2	0	altern. Einheiten verwenden	
191	$DIMALTD	2	2		
192	$DIMALTF	5	1		
193	$DIMALTRND	5	0,01		
194	$DIMALTZ	2	1		
195	$DIMALTTD	2	2		
196	$DIMAPOST	8	“”		
197	$DIMALTTZ	2	0		
198	$DIMSIGNTOL	8	“”		
199	(nicht vorhanden)				

Die korrespondierenden Parameter aus dem Eigenschaftsmenü sind noch nicht vollständig, aber zum vorläufigen Verständnis ausreichend.

Die `Value`-Werte betreffen die Standard-Einstellungen, wie sie mit der Befehlssequenz:

```
Set Gr = Grs.Add(, "TCW25DimLin")   ' Creates Linear Dimension
Gr.Properties("$DIMAUN") = 1
```

erzeugt werden.

Aus dem Namen der `Properties` kann man leider nicht unbedingt direkt auf deren Wirkungsweise, bzw. das korrespondierende Element im Eigenschaftsmenü des Bemaßungsobjektes (s.o.) schließen. Dazu bedarf es eines geeigneten Prüfprogramms, welches die Unterschiede auflistet, die ggf. die einzelnen `Properties` bei unterschiedlichen Einstellungen annehmen.
Inwiefern sich andere Einstellungen im Eigenschaftsmenü auf die `Value`-Werte in der Variablen-Auflistung auswirken, wäre demnach noch zu untersuchen.

-> ist erfolgt – siehe Tabelleneinträge
Des Weiteren siehe auch dazu Alternative Bemaßungsfunktion.

Weitere Anpassungen und Vereinheitlichungen zu den bisherigen Makro-Schreibweisen, sowie den versuchsweisen Variationen führten dann zu folgender Test-Programm(en):

```
Const Pi = 3.14159

Sub Main()
' this sub draw the double line and add three dimensions to it
' Horizontal, Vertical, Parallel -
' Vereinfachte Basis: SingleLine
    Dim app As Application
    Dim drw As Drawing
    Dim grs As Graphics
    Dim gr As Graphic
    'Dim Verts                                  ' wird nicht gebraucht
    Set app = IMSIGX.Application
    Set drw = app.ActiveDrawing
    Set grs = drw.Graphics
    Dim xAnf, yAnf, xEnd, yEnd, mLaeng As Double
    xAnf = 10
    yAnf = 20
    xEnd = 50
    yEnd = 40
    mLaeng = 10
    'Set gr = grs.Add("SingleLine")          ' einfache Linie
    Set gr = grs.Add                         ' vereinfacht
    'Set gr = grs.Add(, "TCW25DblLine")      ' Regen-Method für Doppel-Linie
    gr.Vertices.Add xAnf, yAnf, 0
    gr.Vertices.Add xEnd, yEnd, 0
    'gr.Vertices.Add 5, 5, 0

    'Set gr = LinearDimHorizontal(grs, 0, 0, 5, 5, -2, True)
                                             ' mit Übergabe der x/y-Werte, Maßlinienlänge
     'Set gr = LinearDimHorizontal(grs, (xAnf), (yAnf), (xEnd), (yEnd), -
    (mLaeng), True)           ' Klammersetzung verhindert "By Ref", d.h. ' erzwingt "By Val"!

    'Set gr = LinearDimVertical(grs, 0, 0, 5, 5, 2, False)
    'Set gr = LinearDimVertical(grs, (xAnf), (yAnf), (xEnd), (yEnd), -
(mLaeng), False)
    'Set Gr = LinearDimParallel(grs, 0, 0, 5, 5, 2, False)
    'Set gr = LinearDimParallel(grs, (xAnf), (yAnf), (xEnd), (yEnd), -
(mLaeng), False)
```

```
    Set gr = AngleDim(grs, (xMP), (yMP), (radP), (mLaeng))
'Versuch für Radius-Bemaßung
    drw.Views(0).Refresh

End Sub

'*********************************************************************

Private Function LinearDimHorizontal(grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim gr As Graphic
'Dim Vi As View                  ' wird nicht gebraucht
'Dim Vers As Vertices            ' wird nicht gebraucht
'Dim Ver As Vertex               ' wird nicht gebraucht
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'              object and dimension line
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal", "Vertical", "Parallel"
H = Length

z1 = 0#
z2 = 0#

'Set gr = grs.Add(, "TCW25DimLin")  ' Creates Linear Dimension
Set gr = grs.Add(, "TCW25DimLin")
gr.Properties("$DIMAUN") = 1        ' Item 157, Auswirkung unklar
x0 = X2
y0 = Y1
z0 = 0#

x3 = X2
y3 = Y2 - H#
z3 = 0#

With gr.Vertices            ' Add some vertices to Linear Dimension
'Add([X], [Y], [Z], [PenDown], [Selectable], [Snappable], [Editable],
[Linkable], [Calculated], [Before], [After])
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  '.Add x0, y0, z0                                            'V0 - Variante
  '.Add X1, Y1, z1, False, True, False, True, True, False     'V1
  .Add X1, Y1, z1, False, False, False, False, False, False   'V1 - Varianten
  '.Add X2, Y2, z2, False, True, False, True, True, False     'V2
  .Add X2, Y2, z2, False, False, False, False, False, False   'V2 - Varianten
  '.Add x3, y3, z3, False, True, False, True, False, False    'V3
  .Add x3, y3, z3                                             'V3 - Varianten
  '.Add x0, y0, z0, False, True, False, True, False, True     'V4
  .Add x0 - 10, y0 - 10, z0, False, False, False, False, False, False
                                                              'V4 - Varianten
  '.Add x0, y0, z0, False, False, False, False, False, False  'V5
  .Add x0, y0, z0, False, False, False, False, False, False
```

```
                                                                    ' V5 - Varianten
End With
If (Associative) Then
    'gr.GetSubjectLink 1                    ' Link with dimension's Vertex V1
    gr.GetSubjectLink 2                     ' Link with dimension's Vertex V2
End If
Set LinearDimHorizontal = gr

End Function

'*********************************************************************

Private Function LinearDimVertical(grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim gr As Graphic
Dim Vi As View
Dim Vers As Vertices
Dim Ver As Vertex
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'              object and dimension line
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal"  ' "Vertical","Parallel"
H = Length

z1 = 0#
z2 = 0#

Set gr = grs.Add(, "TCW25DimLin")    ' Creates Linear Dimension
gr.Properties("$DIMAUN") = 1
x0 = X1
y0 = Y2
z0 = 0#

x3 = X2 + H
y3 = Y2
z3 = 0#

With gr.Vertices                        ' Add some vertices to Linear Dimension
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  .Add X1, Y1, z1, False, True, False, True, True, False      'V1
  .Add X2, Y2, z2, False, True, False, True, True, False      'V2
  .Add x3, y3, z3, False, True, False, True, False, False     'V3
  .Add x0, y0, z0, False, True, False, True, False, True      'V4
  .Add x0, y0, z0, False, False, False, False, False, False   'V5
End With
If (Associative) Then
    gr.GetSubjectLink 1              'Link with dimension's Vertex V1
    gr.GetSubjectLink 2              'Link with dimension's Vertex V2
End If
Set LinearDimVertical = gr
```

```
End Function

'*********************************************************************

Private Function LinearDimParallel(grs As Graphics, X1 As Double, Y1 As
Double, X2 As Double, Y2 As Double, Length As Double, Associative As
Boolean) As Graphic
Dim gr As Graphic
Dim Vi As View
Dim Vers As Vertices
Dim Ver As Vertex
' Dimension has 6 base vertices
' V0 - invisible - defines the direction of the dimension
' V1 - visible,editable,linkable
' V2 - visible,editable,linkable
' V3 - visible,editable - defines the normal distance between measured
'                         object and dimension line
' V4 - visible,editable - defines the place of the dimension text
' V5 - invisible,noteditable - defines the place of the dimension text

Dim H#
Dim x0#, y0#, z0#
Dim x3#, y3#, z3#
Dim TypeDimension$
'TypeDimension = "Horizontal"  ' "Vertical","Parallel"
H = Length

z1 = 0#
z2 = 0#

Set gr = grs.Add(, "TCW25DimLin")         ' Creates Linear Dimension
gr.Properties("$DIMAUN") = 1
'For i = 119 To 200                       ' nur zum Test der Properties
'    name1 = gr.Properties(i).Name
'    value1 = gr.Properties(i).Value
'    type1 = gr.Properties(i).Type
'Next


x0 = X2
y0 = Y2
z0 = 0#

Dim L#, sina#, cosa#
Dim x3Loc#, y3Loc#
L = Sqr((X1 - X2) * (X1 - X2) + (Y1 - Y2) * (Y1 - Y2))
sina = (Y2 - Y1) / L
cosa = (X2 - X1) / L
x3Loc = L
y3Loc = -H

x3 = X1 + x3Loc * cosa - y3Loc * sina
y3 = Y1 + x3Loc * sina + y3Loc * cosa
z3 = 0#

With gr.Vertices                    ' Add some vertices to Linear Dimension
  .Add x0, y0, z0, False, False, False, False, False, False   'V0
  .Add X1, Y1, z1, False, True, False, True, True, False      'V1
  .Add X2, Y2, z2, False, True, False, True, True, False      'V2
  .Add x3, y3, z3, False, True, False, True, False, False     'V3
  .Add x0, y0, z0, False, True, False, True, False, True      'V4
```

```
  .Add x0, y0, z0, False, False, False, False, False, False   'V5
End With

If (Associative) Then
    gr.GetSubjectLink 1      'Link with dimension's Vertex V1
    gr.GetSubjectLink 2      'Link with dimension's Vertex V2
End If
Set LinearDimParallel = gr

End Function

'*********************************************************************

Private Function AngleDim(grs As Graphics, xMP As Double, yMP As Double,
radP As Double, Length As Double) As Graphic



End Function

'*********************************************************************

Sub SelectionDim()
' untersucht die Bedingungen für die Parameter-Rückgewinnung aus einem SelectionObjekt

Dim gr As Graphic
'Set gr = grs.Select
Dim app As Application
Set app = IMSIGX.Application
Dim sel As Selection
Set sel = app.Selection
anz = sel.Count
Set gr = sel.Item(0)
If sel.Count < 1 Then
    Set vert1 = gr.Vertices.Item(0)
    xAnf = vert1.X
    yAnf = vert1.Y
    Set vert2 = gr.Vertices.Item(1)
    xEnd = vert2.X
    yEnd = vert2.Y
Else
    grAnz = sel.Count
    For i = 0 To grAnz - 1
        Set gr = sel.Item(i)
        anz = gr.Vertices.Count
        For k = 0 To anz - 1
            Set vert = gr.Vertices.Item(k)
            xAnf = vert.X
            yAnf = vert.Y
        Next
    Next
End If

End Sub

'*********************************************************************

Sub PropertyVerifiation()
' untersucht die 80 neuen Properties auf ihre Korrespondenz im Eigenschaftsmenü

Dim app As Application
Dim drw As Drawing
```

```
Dim grs As Graphics
Dim gr As Graphic
Dim sel As Selection
Dim array1(199)
Dim txt As Variant
Set app = IMSIGX.Application
Set drw = app.ActiveDrawing
Set grs = drw.Graphics
Set sel = app.Selection
anz = sel.Count
Set gr = sel.Item(0)                    ' selektiertes Grafikobjekt definieren
'Set grD = grs.Add(, "TCW25DimLin")     ' Creates Linear Dimension

For i = 0 To 198                        ' Erstbefüllung von array1 für Bemaßungsobjekt
'For i = 0 To 119                       ' für die gr-Variante
        On Error Resume Next            ' für den Fall, dass die Eigenschaft nicht existiert
        'txt = grD.Properties(i).Value
        txt = gr.Properties(i).Value
        array1(i) = txt
        txt = Nothing
Next

Loop1:                                  ' nach Bedienung "Eigenschaftsmenü"
For i = 0 To 198                        ' Veränderungen in den Properties suchen
    'For i = 0 To 119                   ' für die gr-Variante
        On Error Resume Next
        nrD = i + 1                     ' da NrD sonst mit "0" beginnt
        'nameI = grD.Properties(i).Name          ' Name und Value separat ermitteln
        nameI = gr.Properties(i).Name
        'valueI = grD.Properties(i).Value
        ValueI = gr.Properties(i).Value
        arrI = array1(i)                ' Array-Wert auslesen
        If ValueI <> arrI Then          ' mit dem Array-Wert vergleichen
            MsgBox ("NrD : " & nrD & vbCrLf & "Name = " & nameI & vbCrLf &
"Wert = " & ValueI)                     ' Ausgabe
            ValueI = Nothing            ' Value-Wert löschen
        End If
    Next
GoTo Loop1

End Sub
```