

Prof. Dr. U. Bergmann

**Beschreibung zur Bibliothek Trans3Dll
(Koordinatentransformation)**

Stand: 27.05.2015

Programmersteller:
Prof. Dr. U. Bergmann
Beuth Hochschule für Technik Berlin
Fachbereich III
Luxemburger Str. 10
13353 Berlin

Telefon: 4504-2646

Vorwort

Durch das Zusammenwachsen der beiden Teile Deutschland ist eine Vielzahl von Koordinatensystemen im Einsatz, die auf den unterschiedlichen Grundlagen in den beiden Teilen Deutschlands basieren. Um in Zukunft in einem einheitlichen Koordinatensystem in Deutschland und eventuell auch europaweit arbeiten zu können, wurde von der „Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder“ ein weiteres neues Koordinatensystem definiert. Neben der Notwendigkeit Koordinaten dieser Systeme ineinander umrechnen zu können, wird durch den verstärkten Einsatz von „Vermessungsmethoden mit Satelliten“ auch die Transformation dieser geozentrischen kartesischen Koordinaten oder der daraus abgeleiteten geographischen Koordinaten in ebene Gebrauchskoordinaten erforderlich.

Die Bibliothek Trans3Dll enthält die Programme zur Transformation zwischen beliebigen Koordinatensystemen. Es ist durch Parameterdateien für die Koordinatensysteme an die vorliegenden Verhältnisse anpassbar. Daneben ist durch die Hinzunahme von Stützpunkten für die Geoidundulation auch ein Übergang zu den Normalhöhen möglich. Zur Nutzung der Bibliothek in beliebigen Programmiersprachen sind die Schnittstellen zu den gegebenen Unterprogrammen mit Parametern in Standard-C aufgebaut.

Der Nachteil älterer Algorithmen bei der Umrechnung in die Abbildungsebenen, dass durch den Abbruch der Reihenentwicklungen nur eine beschränkte Genauigkeit zur Verfügung steht, ist durch den /Klotz/-schen Formelkörper eliminiert worden. Somit lassen sich auch beliebig große Ausdehnungen von Koordinatensystemen in der Umrechnung mit ausreichender Genauigkeit berücksichtigen.

Inhaltsverzeichnis

1	Allgemeines	4
1.1	Einleitung	4
1.2	Erstellungsdatum	4
2	Nutzung der Bibliothek	4
3	Programmablauf	4
3.1	Berechnungsablauf der Koordinatentransformation	6
3.2	Formeln	8
3.2.1	Ebene Berechnungen	8
3.2.2	Umrechnung in Abbildungsebenen	9
3.2.3	Überführung in geozentrische kartesische Koordinaten	9
3.2.4	Ellipsoidübergang	9
3.2.5	Höhenanomalien	10
4	Programmaufrufe	10
4.1	Aufruf für die vollständige Bearbeitung	10
4.2	Aufrufe zur schrittweisen Bearbeitung	14
4.2.1	Das Unterprogramm translD_dll	15
4.2.2	Das Unterprogramm transla_dll	16
4.2.3	Das Unterprogramm tranzlz_dll	18
4.2.4	Das Unterprogramm transr_dll	20
4.3	Das Unterprogramm transfm_dll	21
5	Notwendige Daten	22
5.1	Steuerungsdateien	22
5.1.1	Steuerungsdatei für die Geoidstützpunkte	22
5.1.2	Steuerungsdatei DEFAULT	22
5.1.3	Parameterdateien	23
5.1.3.1	Unverschlüsselte Parameterdateien mit dem Suffix TRA	23
5.1.3.2	Verschlüsselte Parameterdateien mit dem Suffix TR0	25
6	Nutzung der Bibliothek	25
6.1	Programmiersprache C ⁺⁺	25
6.2	Programmiersprache C [#]	32
6.3	Programmiersprache VB.Net	40
7	Ausgabedaten	49
8	Meldungen der Programme	49
8.1	Meldungen vom Unterprogramm translD_dll	49
8.2	Meldungen von den Unterprogrammen transla_dll und tranzlz_dll	50
8.3	Meldungen vom Unterprogramm transr_dll	52
8.4	Meldungen vom Unterprogramm trans_dll	56
9	Verfügbare Koordinatensysteme	56
10	Genauigkeiten	60
11	Beschränkungen	61
12	Literaturverzeichnis	61

1. Allgemeines

1.1 Einleitung

Die Unterprogramme in der Bibliothek TRANS3DLL.DLL ermöglichen die Transformation zwischen beliebigen Koordinatensystemen. Die Bibliothek ist in 32-Bit Adressierung aus Quellcode in der Programmiersprache C++ übersetzt worden. Die Bibliotheksprogramme benötigen für den Ablauf eine Definitionsdatei mit dem Namen DEFAULT (siehe 5.1.2) und die Parameterdateien mit dem Suffix TRA bzw. TR0 (siehe 5.1.3). Daneben ist zur Berechnung der Geoidundulationen die Datei hoehe.ini (siehe 5.1.1) erforderlich. Durch die Berücksichtigung verschiedenster Umrechnungen in den einzelnen Koordinatensystemarten und die Festlegung der Koordinatensystemparameter (siehe 3) in frei zugänglichen Parameterdateien sind die Unterprogramme individuell einsetzbar. Bei Änderung der amtlichen Parameterdateien wird jedoch darauf hingewiesen, dass Sie die Richtigkeit der Ergebnisse selbstständig sicherstellen müssen.

1.2 Erstellungsdatum

Versionsnummer der Bibliothek: 1.0 vom 27.05.2015

Erstellungsdatum der Definitionsdatei: 20.09.2006

Erstellungsdatum der Parameterdateien: 23.07.2000 bis 14.07.2005

Ersterstellung der Beschreibung: 01.06.2015

letzte Überarbeitung der Beschreibung: keine

2. Nutzung der Bibliothek

Die Bibliothek wird über die Einstiegspunkte für die vollständige Bearbeitung (trans_dll) oder die schrittweise Nutzung der Unterprogramme translD_dll, transla_dll, translz_dll und transr_dll (siehe 4.2) genutzt. Weiterhin steht das Unterprogramm transfm_dll für das Erzeugen einer ausformulierten Fehlermeldung zur Verfügung (siehe 4.3). Voraussetzung für die Nutzung der Koordinatentransformation ist das Vorhandensein der Höhendefinitionsdatei (siehe 5.1.1) und der Parameterdateien der Koordinatensysteme mit der zugehörigen Definitionsdatei (siehe 5.1.2 und 5.1.3). Hierzu werden für den impliziten, den expliziten und den direkten Zugriff neben der Datei **trans3dll.dll** auch die Dateien **trans3dll.exp** und **trans3dll.lib** zur Verfügung gestellt.

3. Programmablauf

Der eigentliche Berechnungsablauf der Koordinatentransformation ist unter dem nächsten Punkt beschrieben.

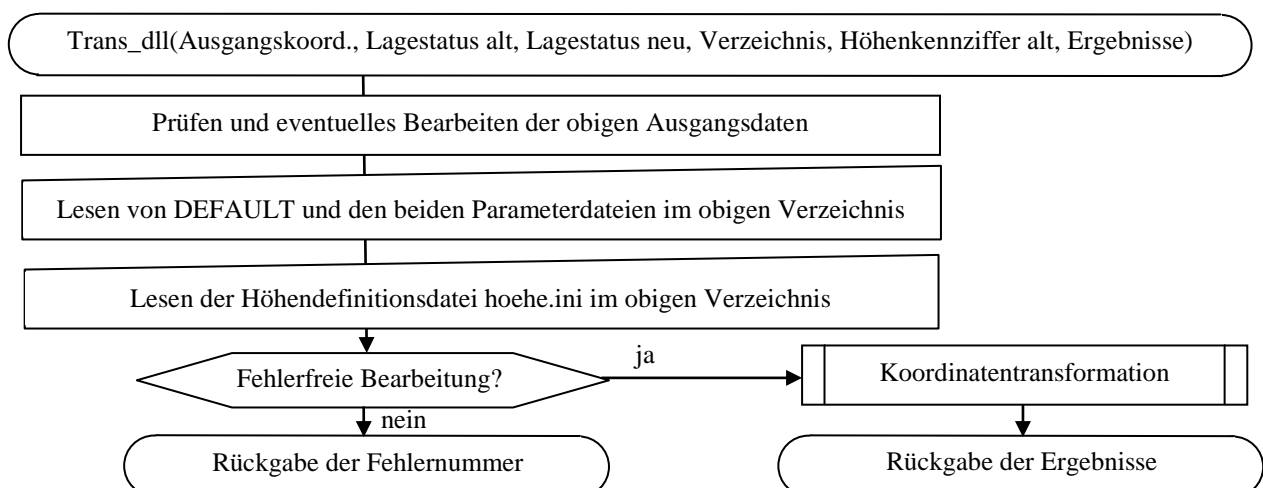


Abbildung 3.1: Ablauf des Unterprogramms Trans_dll

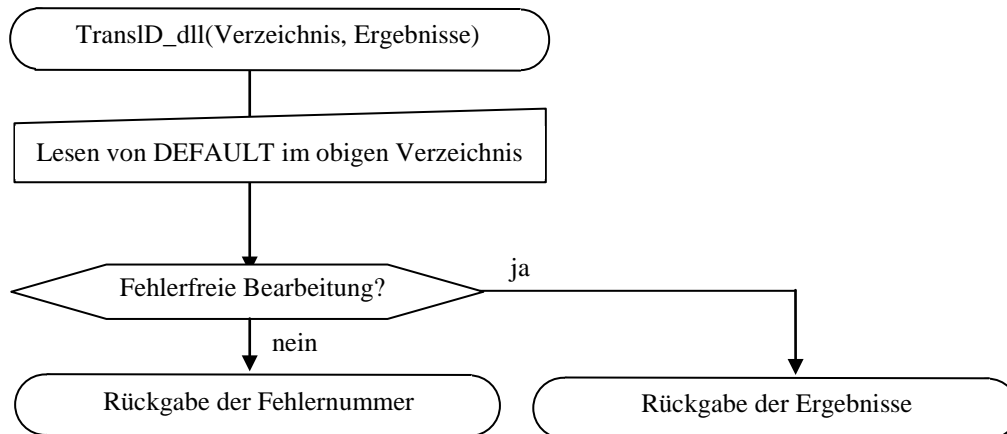


Abbildung 3.2: Ablauf des Unterprogramms TransID_dll

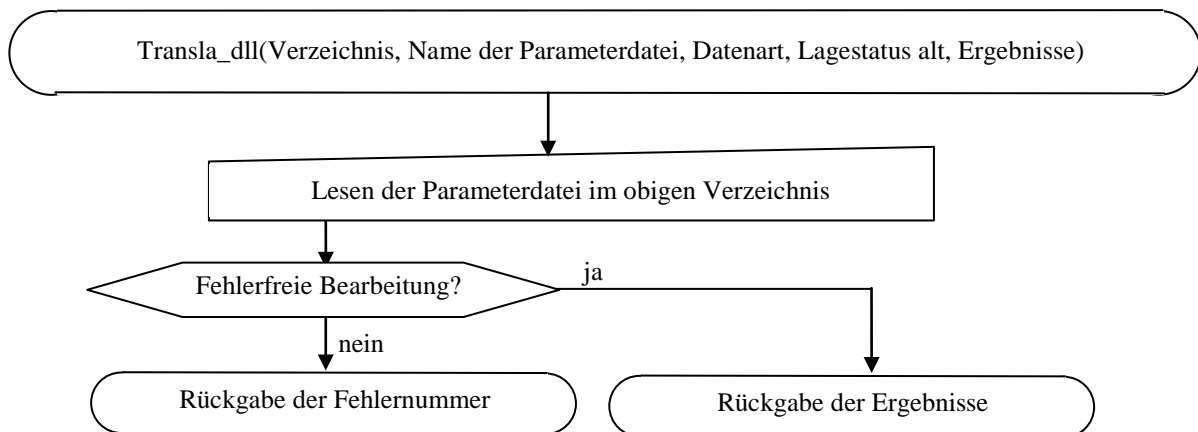


Abbildung 3.3: Ablauf des Unterprogramms Transla_dll

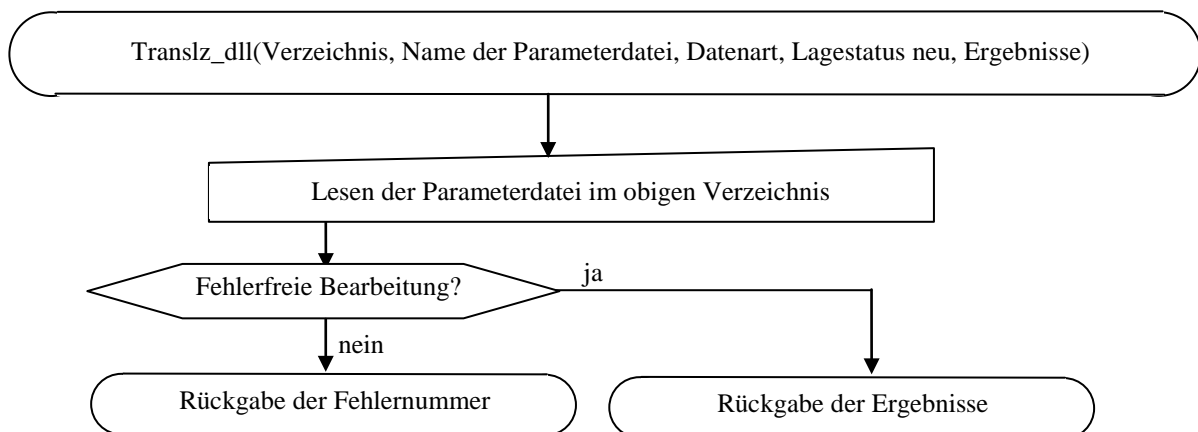


Abbildung 3.4: Ablauf des Unterprogramms Translz_dll

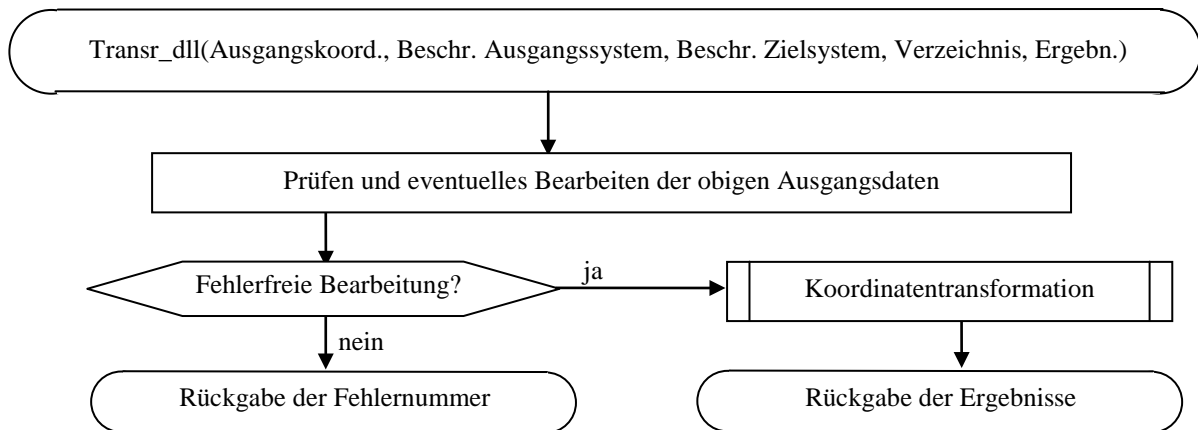


Abbildung 3.5: Ablauf des Unterprogramms Transr_dll

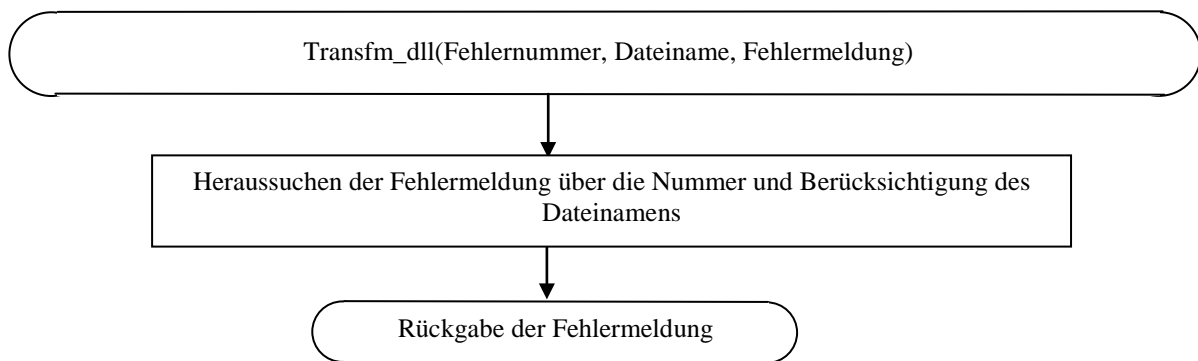


Abbildung 3.6: Ablauf des Unterprogramms Transfm_dll

3.1 Berechnungsablauf der Koordinatentransformation

Nachfolgend ist der Berechnungsablauf anhand eines Ablaufplans in Abhängigkeit von den Ein- und den Ausgabekoordinatensystemen beschrieben. Ausgangs- und Zielsysteme können hierbei zweidimensionale ebene Koordinatensysteme, Systeme mit geographischen Koordinaten und dreidimensionale geozentrische kartesische Koordinatensysteme sein. Hierbei wird für ebene Koordinatensysteme die konforme (Gauß-Krüger- und UTM-Systeme) oder die ordinatentreue (Soldnersysteme) Abbildungsart angeboten. Alle Systeme sind in ihren Parametern frei definierbar. Die ebenen Systeme lassen vor bzw. nach der Umrechnung in geographische Koordinaten eine Reihe von weiteren Umrechnungen in der Ebene zu. Ferner können Systeme auf unterschiedlichen Ellipsoiden ineinander umgerechnet werden. Hierzu wird das Ausgangssystem auf das Besselellipsoid und das Zielsystem vom Besselellipsoid transformiert. Diese Ellipsoidübergänge können dabei in einem Schritt oder auch in zwei Schritten über ein Zwischensystem durchgeführt werden.

Alle Abbildungen basieren auf exakten Formeln und geben dementsprechend bei einer Transformation in den Ergebnissen die Ausgangsgenauigkeiten wider. Alle Ellipsoidübergänge sind anhand von identischen Punkten ermittelt worden. Bei einem Ellipsoidübergang in der Transformation werden die Ergebnisse mit der Genauigkeit ermittelt, die durch die identischen Punkte vorgegeben ist.

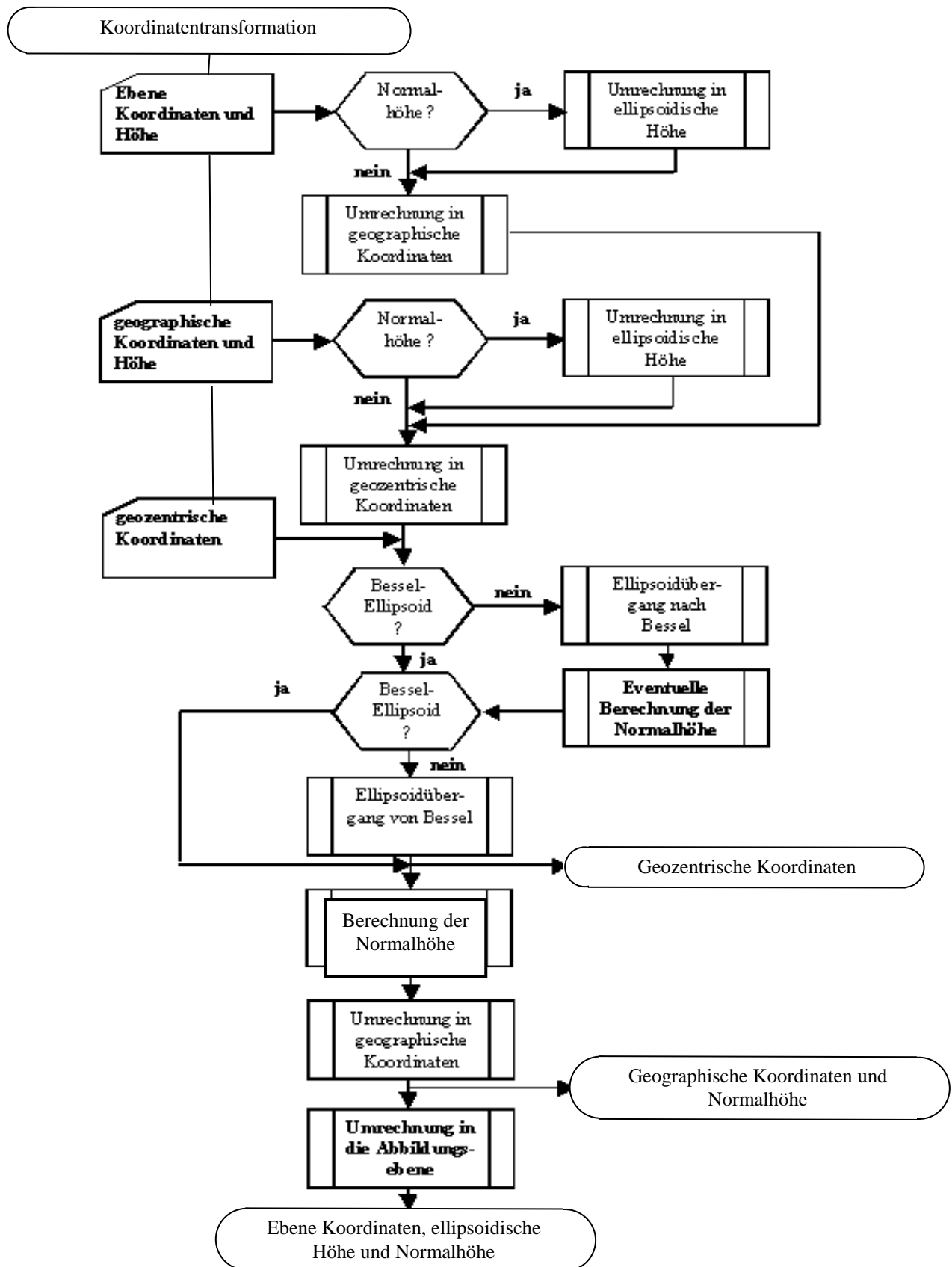


Abbildung 3.7: Ablauf der Koordinatentransformation

3.2 Formeln

3.2.1 Ebene Berechnungen

Ebene Umrechnungen vom Ausgangssystem bis zu den Rohdaten in der Abbildungsebene

gegeben: Y, X im Ausgangssystem

gesucht: Y_R, X_R als Rohdaten in der Abbildungsebene

Ebene Maßstabskorrektur für $M \neq 1$.

$$Y_E = Y_M + M \cdot (Y - Y_M) \quad (1)$$

$$X_E = X_M + M \cdot (X - X_M) \quad (2)$$

mit M .. Maßstabszahl

Y_M .. Nullpunktskoordinate in Y für die Maßstabskorrektur

X_M .. Nullpunktskoordinate in X für die Maßstabskorrektur

Ebene Translationen

$$Y_T = Y_E - Y_N \quad (3)$$

$$X_T = X_E - X_N \quad (4)$$

mit Y_N .. Nullpunktskoordinate in Y

X_N .. Nullpunktskoordinate in X

Dekadische Ergänzung für $E \neq 0$.

$$Y_D = Y_T - E \quad \text{für } Y_T > E/2. \quad (5)$$

$$X_D = X_T - E \quad \text{für } X_T > E/2. \quad (6)$$

mit E .. Wert der dekadischen Ergänzung

Ebene Drehung und Verschiebung für einen der Parameter ungleich 0.

$$Y_R = \frac{Y_D - dy - D \cdot (X_D - dx)}{1 + D \cdot D} + Y_V \quad (7)$$

$$X_R = \frac{X_D - dx + D \cdot (Y_D - dy)}{1 + D \cdot D} + X_V \quad (8)$$

mit Y_V .. Nullpunktskoordinate in Y

X_V .. Nullpunktskoordinate in X

dy .. Verschiebung in Y

dx .. Verschiebung in X

D .. Drehwinkel in rad

Ebene Umrechnungen von den Rohdaten in der Abbildungsebene bis zum Zielsystem

gegeben: Y_R, X_R als Rohdaten in der Abbildungsebene

gesucht: Y, X im Ausgangssystem

Ebene Drehung und Verschiebung für einen der Parameter ungleich 0.

$$Y_D = Y_R - Y_V + dy + D \cdot (X_R - X_V) \quad (9)$$

$$X_D = X_R - X_V + dx - D \cdot (Y_R - Y_V) \quad (10)$$

mit Y_V .. Nullpunktskoordinate in Y

X_V .. Nullpunktskoordinate in X

dy .. Verschiebung in Y

dx .. Verschiebung in X

D .. Drehwinkel in rad

Dekadische Ergänzung für $E \neq 0$.

$$Y_T = Y_D + E \quad \text{für } Y_D < 0. \quad (11)$$

$$X_T = X_D + E \quad \text{für } X_D < 0. \quad (12)$$

mit E .. Wert der dekadischen Ergänzung

Ebene Translationen

$$Y_E = Y_T + Y_N \quad (13)$$

$$X_E = X_T + X_N \quad (14)$$

mit Y_N .. Nullpunktskoordinate in Y

X_N .. Nullpunktskoordinate in X

Ebene Maßstabskorrektur für $M \neq 1$.

$$Y = Y_M + \frac{Y_E - Y_M}{M} \quad (15)$$

$$X = X_M + \frac{X_E - X_M}{M} \quad (16)$$

mit M .. Maßstabszahl

Y_M .. Nullpunktskoordinate in Y für die Maßstabskorrektur

X_M .. Nullpunktskoordinate in X für die Maßstabskorrektur

3.2.2 Umrechnung in Abbildungsebenen

Für die Umrechnung aus der Abbildungsebene in geographische Koordinaten und deren Umkehrung werden die Lösungen über kanonische Gleichungen nach /Klotz; 1991/ bzw. /Klotz; 1993/ benutzt. Diese Formeln erlauben eine beliebig genaue Berechnung für die konformen und die ordinatentreuen Abbildungen auch für große Ausdehnungen des Koordinatensystems. In der Transformationsbibliothek sind für metrische Berechnungen besser als 0,001mm und für Rechnungen im Winkelmaß besser als $1 \cdot 10^{-9}$ Altgradsekunden eingestellt.

3.2.3 Überführung in geozentrische kartesische Koordinaten

Für die Überführung von geographischen in geozentrische kartesische Koordinaten und deren Umkehrung werden die Formeln nach /Heck/ benutzt.

3.2.4 Ellipsoidübergang

Für die Ellipsoidübergänge werden zwei Bearbeitungswege angeboten. Zum einen der direkte Weg vom Ausgangs- zum Zielellipsoid und zum anderen über ein Zwischensystem. Der erste Weg wird dann durchgeführt, wenn die Translations- und Rotationsparameter des zweiten Übergangs durchgehend auf 0. gesetzt sind.

Direkter Weg für $X_{TE1}=0 \wedge Y_{TE1}=0 \wedge Z_{TE1}=0 \wedge \alpha_1=0 \wedge \beta_1=0 \wedge \gamma_1=0$

gegeben: X_A, Y_A, Z_A im Ausgangsellipsoid

gesucht: X_Z, Y_Z, Z_Z im Zielellipsoid

$$\begin{aligned} X_Z &= X_{TE} + M_T \cdot \cos(\beta) \cdot \cos(\gamma) \cdot (X_A - X_{T0}) \\ &+ M_T \cdot (\cos(\alpha) \cdot \sin(\gamma) + \sin(\alpha) \cdot \sin(\beta) \cdot \cos(\gamma)) \cdot (Y_A - Y_{T0}) \\ &+ M_T \cdot (\sin(\alpha) \cdot \sin(\gamma) - \cos(\alpha) \cdot \sin(\beta) \cdot \cos(\gamma)) \cdot (Z_A - Z_{T0}) \end{aligned} \quad (17)$$

$$\begin{aligned}
Y_Z &= Y_{TE} - M_T \cdot \cos(\beta) \cdot \sin(\gamma) \cdot (X_A - X_{T0}) \\
&\quad + M_T \cdot (\cos(\alpha) \cdot \cos(\gamma) - \sin(\alpha) \cdot \sin(\beta) \cdot \sin(\gamma)) \cdot (Y_A - Y_{T0}) \\
&\quad + M_T \cdot (\sin(\alpha) \cdot \cos(\gamma) + \cos(\alpha) \cdot \sin(\beta) \cdot \sin(\gamma)) \cdot (Z_A - Z_{T0})
\end{aligned} \tag{18}$$

$$\begin{aligned}
Z_Z &= Z_{TE} + M_T \cdot \sin(\beta) \cdot (X_A - X_{T0}) - M_T \cdot \sin(\alpha) \cdot \cos(\beta) \cdot (Y_A - Y_{T0}) \\
&\quad + M_T \cdot \cos(\alpha) \cdot \cos(\beta) \cdot (Z_A - Z_{T0})
\end{aligned} \tag{19}$$

mit X_{T0} .. Schwerpunktkoordinate in X
 Y_{T0} .. Schwerpunktkoordinate in Y
 Z_{T0} .. Schwerpunktkoordinate in Z
 X_{TE} .. Translation in X
 Y_{TE} .. Translation in Y
 Z_{TE} .. Translation in Z
 α .. Rotation um X
 β .. Rotation um Y
 γ .. Rotation um Z
 M .. Maßstab

Lösung über ein Zwischensystem für $X_{TE1} \neq 0 \cup Y_{TE1} \neq 0 \cup Z_{TE1} \neq 0 \cup \alpha_1 \neq 0 \cup \beta_1 \neq 0 \cup \gamma_1 \neq 0$

gegeben: X_A, Y_A, Z_A im Ausgangsellipsoid

gesucht: X_E, Y_E, Z_E im Zielellipsoid

Berechnung von X_Z, Y_Z, Z_Z auf dem Zwischensystem nach den Formeln (17) bis (19)

$$X_E = X_{T01} + \cos(\beta_1) \cdot \cos(\gamma_1) \cdot X_Z / M_1 - \cos(\beta_1) \cdot \sin(\gamma_1) \cdot Y_Z / M_1 + \sin(\beta_1) \cdot Z_Z / M_1 \tag{20}$$

$$\begin{aligned}
Y_E &= Y_{T01} + (\cos(\alpha_1) \cdot \sin(\gamma_1) + \sin(\alpha_1) \cdot \sin(\beta_1) \cdot \cos(\gamma_1)) \cdot X_Z / M_1 \\
&\quad + (\cos(\alpha_1) \cdot \cos(\gamma_1) - \sin(\alpha_1) \cdot \sin(\beta_1) \cdot \sin(\gamma_1)) \cdot Y_Z / M_1 - \sin(\alpha_1) \cdot \cos(\beta_1) \cdot Z_Z / M_1
\end{aligned} \tag{21}$$

$$\begin{aligned}
Z_E &= Z_{T01} + (\sin(\alpha_1) \cdot \sin(\gamma_1) - \cos(\alpha_1) \cdot \sin(\beta_1) \cdot \cos(\gamma_1)) \cdot X_Z / M_1 \\
&\quad + (\sin(\alpha_1) \cdot \cos(\gamma_1) + \cos(\alpha_1) \cdot \sin(\beta_1) \cdot \sin(\gamma_1)) \cdot Y_Z / M_1 + \cos(\alpha_1) \cdot \cos(\beta_1) \cdot Z_Z / M_1
\end{aligned} \tag{22}$$

mit X_{T01} .. Schwerpunktkoordinate in X
 Y_{T01} .. Schwerpunktkoordinate in Y
 Z_{T01} .. Schwerpunktkoordinate in Z
 X_{TE1} .. Translation in X
 Y_{TE1} .. Translation in Y
 Z_{TE1} .. Translation in Z
 α_1 .. Rotation um X
 β_1 .. Rotation um Y
 γ_1 .. Rotation um Z
 M_1 .. Maßstab

3.2.5 Höhenanomalien

Für die Berechnung der relativen Höhenanomalien zwischen den ellipsoidischen Höhen auf dem Bessel-ellipsoid und den Normalhöhen im Datum DHHN 92 liegen in Deutschland die absoluten Anomalien auf dem GRS80-Ellipsoid vor. Ausgehend von einem Raster wird über die Länge und die Breite die absolute Höhenanomalie auf dem GRS80-Ellipsoid über die bilineare Interpolation ermittelt. Die Höhendifferenz zwischen dem Bessel- und dem GRS80-Ellipsoid wird sodann über die Formel von Vening-Meinesz /Heiskanen and Moritz, 1967/

berechnet, woraus sich als Ergebnis als Differenz die relative Höhenanomalie auf dem Besselellipsoid ergibt.

Gegeben: Länge l_B und Breite b_B auf dem Besselellipsoid

Gesucht: relative Höhenanomalie auf dem Besselellipsoid ζ_r

Bearbeitungsschritte:

Überführung von l_B und b_B nach den Formeln (17) bis (19) bzw. bis (22) auf das GRS80-Ellipsoid als l_G und b_G

Berechnung der Höhendifferenz h_{BE} nach Vening-Meinesz als Funktion von l_G und b_G , den Ellipsoidangaben und den Angaben im Zentralpunkt Rauenberg

Bestimmung der absoluten Anomalie über die bilineare Interpolation als Funktion von l_G und b_G

$$\zeta = \zeta_1 + \frac{db \cdot (\zeta_2 - \zeta_1)}{dsb} + \frac{dl \cdot \left(\zeta_3 - \zeta_1 + \frac{db \cdot (\zeta_4 - \zeta_3 - \zeta_2 + \zeta_1)}{dsb} \right)}{dsl} \quad (23)$$

mit db .. Breitenabstand von b_G zum nächsten Stützpunkt

dsb .. Breitenabstand zwischen den Stützpunkten

dl .. Längenabstand von l_G zum nächsten Stützpunkt

dsl .. Längenabstand zwischen den Stützpunkten

ζ_i .. Anomalien der vier benachbarten Stützpunkte

Berechnung der relativen Anomalie

$$\zeta_r = \zeta - h_{BE} \quad (24)$$

4. Programmaufrufe

Die Unterprogramme der Bibliothek sind so aufgebaut, dass entweder ein Gesamtdurchlauf über das Unterprogramm Trans_dll oder alle Zwischenschritte der Koordinatentransformation einzeln sich mit den gegebenen Unterprogrammen TranslD_dll, Transla_dll, Translz_dll und Transr_dll erledigen lassen. Daneben werden auch alle Informationen und Parameter durch diese Programme zur Verfügung gestellt. Der nachfolgende Ablauf zeigt die entsprechenden Zusammenhänge.

4.1 Aufruf zur vollständigen Bearbeitung

Bei der Nutzung dieses Unterprogramms erfolgt die Auswahl des Ausgangs- und des Zielsystems über die Statusangaben und über die Art der Ausgangshöhen. Für eine Einzelauswertung steht damit eine komfortable Lösung zur Verfügung, welche allerdings jeweils mit dem Auslesen der Parameterdateien verbunden ist. Für eine wiederholte Überführung aus dem gleichen Ausgangs- in das gleiche Zielsystem sollte auf Grund der kürzeren Bearbeitungszeiten die schrittweise Bearbeitung genutzt werden, da hier die Parameterdateien nur ein Mal gelesen werden müssen.

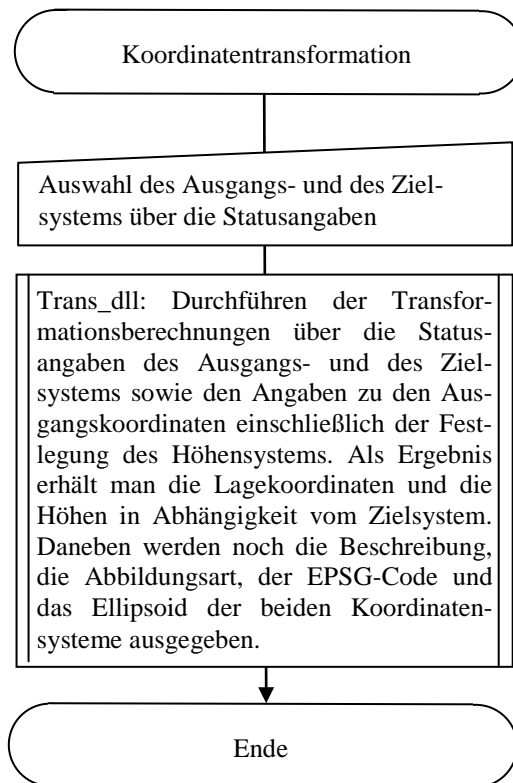


Abbildung 4.1: Vollständige Bearbeitung

Das Unterprogramm `Trans_dll` dient der Transformation eines Einzelpunktes. Als Eingabewerte sind hierbei die dreidimensionalen Koordinaten einschließlich der zugehörigen Lagestatusangabe, die Lagestatusangabe zum Zielkoordinatensystem, die Art der Höhenangabe, solange diese im Koordinatensatz enthalten ist, und die Pfadangabe für die Steuerungsdateien (Datei `DEFAULT` und Parameterdateien in unverschlüsselter oder verschlüsselter Form) erforderlich. Als Ausgabedaten werden die transformierten Koordinaten grundsätzlich mit den Höhenangaben auf dem Ellipsoid und dem Geoid (Normalhöhe), soweit das Zielsystem mit einer Höhenangabe verbunden ist, sowie die Beschreibungen zu den beiden Koordinatensystemen zurückgegeben. Als Beschreibung werden die Bezeichnung, die Abbildungsart und das Ellipsoid ausgegeben. Beim Aufruf dieses Unterprogramms werden bei jedem Aufruf alle notwendigen Parameterdateien wie die Datei `DEFAULT`, die zugehörigen Parameterdateien der beiden Koordinatensysteme und die Datei `hoehe.ini` ausgelesen. Bei der Angabe von `.` (Punkt) für das Verzeichnis wird das aktuelle Verzeichnis beim Suchen nach den Dateien `DEFAULT` und den beiden Parameterdateien genutzt.

Bei den Datentypen ist der Unterschied zwischen den in `C++` genutzten und den eigenen zu beachten. Bei Texten ist außerdem darauf zu achten, dass eventuell noch eine Wandlung von Unicode- in die ASCII-Verschlüsselung vorgenommen werden muss.

Bei einem erfolgreichen Aufruf erhält man im Returncode den Zahlenwert 0, ansonsten Zahlenwerte größer als Null, die auf einen in 8.4 dokumentierten Fehler hinweisen. Der Rückgabewert ist eine Ganzzahl, die in 4 Bytes gespeichert ist. Fehlermeldungen bzw. Hinweise lassen sich in ausformulierter Form über den Returncode und die Angabe zur bearbeiteten Datei mit dem Unterprogramm `transfm_dll` erzeugen (siehe 4.3).

Definition:

```
extern "C" __declspec(dllexport) long trans_dll(double x,
double y, double z, long l1, long l2, const wchar_t kat[100],
wchar_t a_hk[2], double xout[4], wchar_t l_koord[100],
long epsg[2], wchar_t ellip[11], wchar_t abb[14],
wchar_t z_koord[100], wchar_t ellipa[11], wchar_t abba[14]);
```

Anmerkungen:

- long wird in Visual Studio 2010 C++ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
- wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 2010 C++
- double wird in Visual Studio 2010 C++ als Fließzahlvariable mit 8 Bytes mit 15 signifikanten Stellen gespeichert

Bedeutung der Parameter:

Ausgangsparameter:

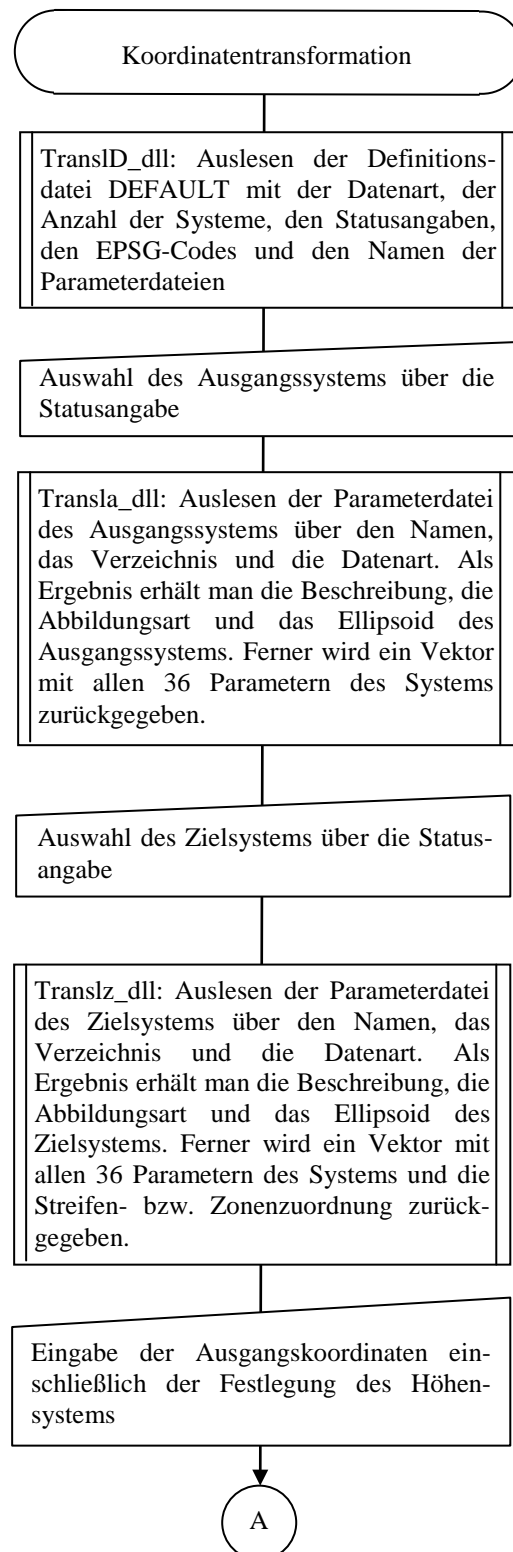
double x	.. X-Koordinate der geozentrischen und ebenen Systeme bzw. die geographische Breite als Fließkommavariablen in 8 Bytes
double y	.. Y-Koordinate der geozentrischen und ebenen Systeme bzw. die geographische Länge als Fließkommavariablen in 8 Bytes
double z	.. Z-Koordinate der geozentrischen Systeme bzw. die ellipsoidische Höhe oder die Normalhöhe als Fließkommavariablen in 8 Bytes
long l1	.. Lagestatus des Ausgangssystems als Ganzzahl in 4 Bytes.
long l2	.. Lagestatus des Zielsystems als Ganzzahl in 4 Bytes.
wchar_t kat[100]	.. Verzeichnis der Parameterdateien und von DEFAULT mit maximal 99 Zeichen in Unicode
wchar_t a_hk[2]	.. Kennziffer zu den gegebenen Höhen in z mit dem ersten Zeichen (n,N .. Normalhöhe; ≠n,N .. Ellipsoidische Höhe) in Unicode

Rückgabeparameter:

double xout[4]	.. Vektor der Koordinaten als Fließkommavariablen in 8 Bytes in der Reihenfolge X-Koordinate der geozentrischen oder ebenen Systeme bzw. die geographische Breite, Y-Koordinate der geozentrischen oder ebenen Systeme bzw. die geographische Länge, Z-Koordinate der geozentrischen Systeme bzw. die ellipsoidische Höhe und der Normalhöhe in allen nichtgeozentrischen Systemen
wchar_t l_koord[100]	.. Beschreibung des Ausgangssystems in maximal 99 Zeichen in Unicode
wchar_t ellip[11]	.. Name des Ellipsoids des Ausgangssystems mit maximal 10 Zeichen in Unicode
wchar_t abb[14]	.. Abbildungsart des Ausgangssystems mit maximal 13 Zeichen in Unicode
wchar_t z_koord[100]	.. Beschreibung des Zielsystems in maximal 99 Zeichen in Unicode.

wchar_t ellipa[11] .. Name des Ellipsoids des Zielsystems mit maximal 10 Zeichen in Unicode
wchar_t abba[14] .. Abbildungsart des Zielsystems mit maximal 13 Zeichen in Unicode
long epsg[2] .. EPSG-Code des Ausgangs- und des Zielsystems

4.2 Aufrufe zur schrittweisen Bearbeitung



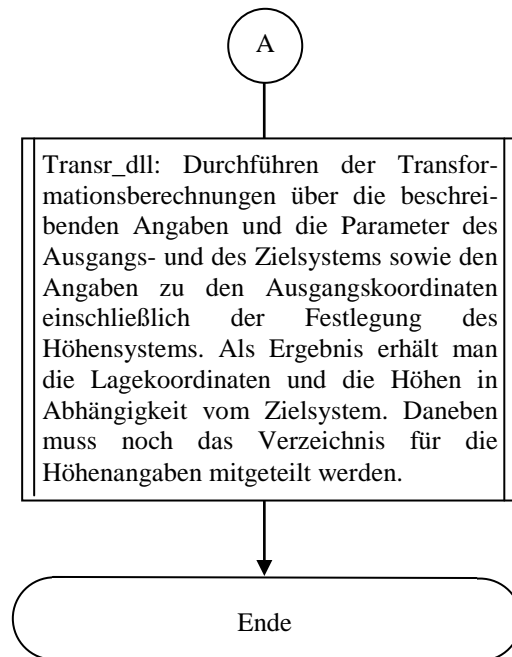


Abbildung 4.2: Schrittweise Bearbeitung

Aus der obigen Abbildung ist ersichtlich, dass nicht unbedingt alle Unterprogramme zum Einsatz kommen müssen. So wird das erste Unterprogramm nur benötigt, wenn eine Übersicht über die zur Verfügung stehenden Koordinatensysteme extern gewonnen werden soll. Die beiden Leseprogramme Transla_dll und Translz_dll müssen dann zum Einsatz kommen, wenn vor der Transformation Informationen zu den Systemen erforderlich sind und die Parameter der beiden Systeme noch nicht bekannt sind.

Nachfolgend wird zu den Schnittstellenproblemen beim Einsatz unterschiedlicher Programmiersprachen an vielen Stellen eingegangen. So muss man schon bei den Ganzzahlen darauf achten, mit wie vielen Bytes diese abgespeichert werden. Insbesondere Texte werden in den unterschiedlichen Programmiersprachen grundsätzlich unterschiedlich gesehen und auch die Codierung in ASCII oder Unicode muss beachtet werden. Daneben steht sodann noch, die unterschiedliche Sicht auf Vektoren. Mit der vorliegenden Bibliothek Trans3DLL.dll ist auf den Einsatz von Standard-C bei der Definition der Parameter geachtet worden. Hierbei werden alle Ganzzahlparameter in Long mit 4 Bytes und Vorzeichen gespeichert. Alle Textangaben sind im Datentyp wchar_t als Vektoren mit 2 Bytes (Unicode) in den Aufrufen gespeichert.

Die dynamisch eingebundene Bibliothek wird in Abhängigkeit von der Programmierungsumgebung über die gleichzeitig erzeugte Trans3DLL.lib oder die Datei Trans3DLL.exp eingebunden.

4.2.1 Das Unterprogramm TranslD_dll

Dieses Unterprogramm dient zum Auslesen der Definitionsdatei DEFAULT mit der Datenart, der Anzahl der Systeme, den Statusangaben, den EPSG-Codes und den Namen der Parameterdateien. Hierzu wird dem Unterprogramm das Verzeichnis mitgegeben, in welchem die Definitiondatei DEFAULT (siehe 5.2.1) zu finden ist. Hierbei wird ein Punkt als Angabe für das Verzeichnis als aktuelles Verzeichnis interpretiert. Die Datenart gibt hierbei wieder, ob die Parameterdateien bei einem Wert ungleich Null verschlüsselt sind. Bei unverschlüsselten Dateien weisen die Parameterdateien den Suffix tra und bei verschlüsselten Dateien den Zusatz tr0 auf. Die Länge der eigentlichen Dateinamen ohne den Punkt und den

Zusatz ist begrenzt auf 100 Zeichen. Treten längere Namen auf, so werden diese gekürzt. Auch die Anzahl der Koordinatensysteme ist auf die Zahl 100 begrenzt. Auch hier werden die über die Anzahl 100 hinausgehenden Systeme nicht berücksichtigt. Bei den Datentypen ist der Unterschied zwischen den in C++ genutzten und den eigenen zu beachten. Bei Texten ist außerdem darauf zu achten, dass nach der Entgegennahme aus dem Unterprogramm mit den mitgesendeten Textlängenangaben die einzelnen Texte herausgelöst werden müssen sowie eventuell noch eine Wandlung von Unicode- in die ASCII-Verschlüsselung vorgenommen werden muss.

Bei einem erfolgreichen Aufruf erhält man im Returncode den Zahlenwert 0, ansonsten Zahlenwerte größer als Null, die auf einen in 8.1 dokumentierten Fehler hinweisen. Der Rückgabewert ist eine Ganzzahl, die in 4 Bytes gespeichert ist. Ausformulierte Fehler- bzw. Hinweismeldungen können über das Unterprogramm Transfm_dll erzeugt werden (siehe 8.3). Nachfolgend ist die Definition im Erstellungstool Visual Studio 2010 mit der Sprache C++ wiedergegeben. Hierbei ist darauf zu achten, dass im Vektor datei alle Namen der Parameterdateien ohne Punkt und Suffix fortlaufend eingetragen sind.

Beispiele für die Entgegennahme und das Wandeln der Ergebnisse sind unter dem Menüpunkt 6. untergebracht.

Definition: `extern "C" __declspec(dllexport) long trans1D_dll
(const wchar_t kat[100], long &dart, long &anz, long nr[100],
long epsg[100], wchar_t datei[100][100]);`

Anmerkungen:

- long wird in Visual Studio 2010 C++ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
- wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 6 C++

Bedeutung der Parameter:

Ausgangsparmeter:

wchar_t kat[100] .. Verzeichnis der Parameterdateien und von
DEFAULT mit maximal 99 Zeichen in Unicode

Rückgabeparmeter:

long &dart .. Art der Verschlüsselung (0 .. keine; ≠ 0 ..
verschlüsselt) als Ganzzahl in 4 Bytes.

long &anz .. Anzahl der Koordinatensysteme als Ganzzahl in 4
Bytes

long nr[100] .. Vektor zum Speichern der ganzzahligen
Statuszahlen in 4 Bytes (Startwert: 0; Anzahl der
Elemente: 100)

long epsg[100] .. Vektor zum Speichern der ganzzahligen EPSG-
Codes in 4 Bytes (Startwert: 0; Anzahl der
Elemente: 100)

wchar_t datei[100][100] .. Vektor zum Speichern der Namen der maximal 100
Parameterdateien ohne das Suffix in Unicode-
Verschlüsselung mit einer maximalen Länge von 99
Zeichen.

4.2.2 Das Unterprogramm Transla_dll

Dieses Unterprogramm dient zum Auslesen der Parameterdatei des Ausgangssystems. Hierzu wird dem Unterprogramm das Verzeichnis mitgegeben, in welchem die Parameterdateien zu

finden sind. Dabei wird ein Punkt als Angabe für das Verzeichnis als aktuelles Verzeichnis interpretiert. Ferner wird der Name der Parameterdatei mitgegeben und mit der Datenart auch das Suffix in tra bzw. tr0 vorgegeben. Zurückgegeben werden die Parameter als Fließkommawerte in einem Vektor mit 36 Elementen sowie die verbale Beschreibung, die Abbildungsart und das Ellipsoid des Ausgangssystems. Bei den Datentypen ist der Unterschied zwischen den in C++ genutzten und den eigenen zu beachten. Bei Texten ist außerdem darauf zu achten, dass eventuell noch eine Wandlung von Unicode- in die ASCII-Verschlüsselung vorgenommen werden muss.

Bei einem erfolgreichen Aufruf erhält man im Returncode den Zahlenwert 0, ansonsten Zahlenwerte größer als Null, die auf einen in 8.2 dokumentierten Fehler hinweisen, welche sich mit dem Unterprogramm Transfm_dll in ausformulierte Fehler- bzw. Hinweismeldungen erweitern lassen. Der Rückgabewert ist eine Ganzzahl, die in 4 Bytes gespeichert ist. Nachfolgend ist die Definition im Erstellungstool Visual Studio 2010 mit der Sprache C++ wiedergegeben

Definition:

```
extern "C" __declspec(dllexport) long transla_dll
(const wchar_t kat[100], wchar_t l_name[100], long dart,
long l_status, wchar_t l_koord[100], wchar_t ellip[11],
wchar_t abb[14], double paraa[36]);
```

Anmerkungen:

- long wird in Visual Studio 2010 C++ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
- wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 2010 C++
- double wird in Visual Studio 2010 C++ als Fließzahlvariable mit 8 Bytes mit 15 signifikanten Stellen gespeichert

Bedeutung der Parameter:

Ausgangsparameter:

wchar_t kat[100]	.. Verzeichnis der Parameterdateien und von DEFAULT mit maximal 99 Zeichen in Unicode
wchar_t l_name[100]	.. Name der Parameterdatei ohne Suffix mit maximal 99 Zeichen in Unicode
long dart	.. Datenart (0 .. unverschlüsselt; ≠ 0 .. verschlüsselt) als Ganzzahl in 4 Bytes
long l_status	.. Statusangabe vom Ausgangssystem als Ganzzahl in 4 Bytes

Rückgabeparameter:

wchar_t l_koord[100]	.. Beschreibung des Systems in maximal 99 Zeichen in Unicode.
wchar_t ellip[11]	.. Name des Ellipsoids mit maximal 10 Zeichen in Unicode
wchar_t abb[14]	.. Abbildungsart mit maximal 13 Zeichen in Unicode
double paraa[36]	.. Vektor zum Speichern der 36 Parameter des Ausgangssystems im Datentyp double (Startwert: 0; Anzahl der Elemente: 36).

lfd. Nr.	Eintrag im Vektor paraa	lfd. Nr.	Eintrag im Vektor paraa
1.	Geographische Breite des Nullpunktes in dezimalen Altgrad	20.	Rotation um Z (rad) (γ)
2.	Geographische Länge des Nullpunktes in dezimalen Altgrad	21.	Maßstab (M_T)
3.	Additionskonstante in Y bzw. im Rechtswert in Meter (Y_N)	22.	Schwerpunktkoordinate in X (Meter) (X_{T0})
4.	Additionskonstante in X bzw. im Hochwert in Meter (X_N)	23.	Schwerpunktkoordinate in Y (Meter) (Y_{T0})
5.	Dekadische Ergänzung (E)	24.	Schwerpunktkoordinate in Z (Meter) (Z_{T0})
6.	Maßstab des Meridians	25.	Große Halbachse des Ellipsoids
7.	Nullpunktkoordinate in Y (Meter) (Y_V)	26.	Kleine Halbachse des Ellipsoids
8.	Nullpunktkoordinate in X (Meter) (X_V)		Angaben zum Ellipsoidübergang vom Neusystem in das Zwischensystem
9.	Verdrehung (rad) (D)	27.	Translation in X (Meter) (X_{TE1})
10.	Verschiebung in Y (Meter) (dy)	28.	Translation in Y (Meter) (Y_{TE1})
11.	Verschiebung in X (Meter) (dx)	29.	Translation in Z (Meter) (Z_{TE1})
12.	Nullpunktkoord. in Y (Meter) (Y_M)	30.	Rotation um X (rad) (α_1)
13.	Nullpunktkoord. in X (Meter) (X_M)	31.	Rotation um Y (rad) (β_1)
14.	Maßstab (M)	32.	Rotation um Z (rad) (γ_1)
15.	Angaben zum Ellipsoidübergang nach Bessel bzw. in das Zwischensystem	33.	Maßstab (M_{T1})
16.	Translation in X (Meter) (X_{TE})	34.	Schwerpunktskoordinate in X (Meter) (X_{T01})
17.	Translation in Y (Meter) (Y_{TE})	35.	Schwerpunktskoordinate in Y (Meter) (Y_{T01})
18.	Translation in Z (Meter) (Z_{TE})	36.	Schwerpunktskoordinate in Z (Meter) (Z_{T01})
19.	Rotation um X (rad) (α)		
	Rotation um Y (rad) (β)		

Tabelle 4.1: Inhalt des Parametervektors für das Ausgangssystem

4.2.3 Das Unterprogramm Translz_dll

Dieses Unterprogramm dient zum Auslesen der Parameterdatei des Zielsystems. Hierzu wird dem Unterprogramm das Verzeichnis mitgegeben, in welchem die Parameterdateien zu finden sind. Hierbei wird ein Punkt als Angabe für das Verzeichnis als aktuelles Verzeichnis interpretiert. Ferner wird der Name der Parameterdatei ohne Suffix mitgegeben und mit der Datenart auch das Suffix in tra bzw. tr0 vorgegeben. Zurückgegeben werden die Fließkommaparameter in einem Vektor mit 36 Elementen sowie die verbale Beschreibung, die Abbildungsart und das Ellipsoid des Zielsystems. Zur Festlegung einer Zone- bzw. eines Streifens wird auch dieser Eintrag als ganze Zahl zurückgegeben. Bei den Datentypen ist der Unterschied zwischen den in C⁺⁺ genutzten und den eigenen zu beachten. Bei Texten ist außerdem darauf zu achten, dass eventuell noch eine Wandlung von Unicode- in die ASCII-Verschlüsselung vorgenommen werden muss.

Bei einem erfolgreichen Aufruf erhält man im Returncode den Zahlenwert 0, ansonsten Zahlenwerte größer als Null, die auf einen in 8.2 dokumentierten Fehler hinweisen, welche sich mit dem Unterprogramm Transfm_dll in ausformulierte Fehler- bzw. Hinweismeldungen erweitern lassen. Der Rückgabewert ist eine Ganzzahl, die in 4 Bytes gespeichert ist.

Definition: `extern "C" __declspec(dllexport) long translz_dll
(const wchar_t kat[100], wchar_t z_name[100], long dart,
long z_status, wchar_t z_koord[100], wchar_t ellipa[11],
wchar_t abba[14], long &streifen, double parab[36]);`

- Anmerkungen:**
- long wird in Visual Studio 2010 C++ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
 - wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 2010 C++
 - double wird in Visual Studio 2010 C++ als Fließzahlvariable mit 8 Bytes mit 15 signifikanten Stellen gespeichert

Bedeutung der Parameter:

Ausgangsparameter:

- wchar_t kat[100] .. Verzeichnis der Parameterdateien und von DEFAULT mit maximal 99 Zeichen in Unicode
- wchar_t z_name[100] .. Name der Parameterdatei ohne Suffix mit maximal 99 Zeichen in Unicode
- long dart .. Datenart (0 .. unverschlüsselt; ≠ 0 .. verschlüsselt) als Ganzzahl in 4 Bytes
- long z_status .. Statusangabe zum Zielsystem als Ganzzahl in 4 Bytes

Rückgabeparameter:

- wchar_t z_koord[100] .. Beschreibung des Systems in maximal 99 Zeichen in Unicode.
- wchar_t ellipa[11] .. Name des Ellipsoids mit maximal 10 Zeichen in Unicode
- wchar_t abba[14] .. Abbildungsart mit maximal 13 Zeichen in Unicode
- double paraz[36] .. Vektor zum Speichern der 36 Parameter des Zielsystems im Datentyp double. (Startwert: 0; Anzahl der Elemente: 36).
- long &streifen .. Zonen- bzw. Streifennummer

lfd. Nr.	Eintrag im Parametervektor paraz	lfd. Nr.	Eintrag im Parametervektor paraz
1.	Geographische Breite des Nullpunktes in dezimalen Altgrad	10.	Verschiebung in Y (Meter) (dy)
2.	Geographische Länge des Nullpunktes in dezimalen Altgrad	11.	Verschiebung in X (Meter) (dx)
3.	Additionskonstante in Y bzw. im Rechtswert in Meter (Y _N)	12.	Nullpunktkoord. in Y (Meter) (Y _M)
4.	Additionskonstante in X bzw. im Hochwert in Meter (X _N)	13.	Nullpunktkoord. in X (Meter) (X _M)
5.	Dekadische Ergänzung (E)	14.	Maßstab (M)
6.	Maßstab des Meridians		Angaben zum Ellipsoidübergang von Bessel bzw. vom Zwischensystem
7.	Nullpunktkoordinate in Y (Meter) (Y _V)	15.	Translation in X (Meter) (X _{TE})
8.	Nullpunktkoordinate in X (Meter) (X _V)	16.	Translation in Y (Meter) (Y _{TE})
9.	Verdrehung (rad) (D)	17.	Translation in Z (Meter) (Z _{TE})
		18.	Rotation um X (rad) (α)
		19.	Rotation um Y (rad) (β)

lfd. Nr.	Eintrag im Parametervektor paraz	lfd. Nr.	Eintrag im Parametervektor paraz
20.	Rotation um Z (rad) (γ)	28.	Translation in Y (Meter) (Y_{TE1})
21.	Maßstab (M_T)	29.	Translation in Z (Meter) (Z_{TE1})
22.	Schwerpunktkoordinate in X (Meter) (X_{T0})	30.	Rotation um X (rad) (α_1)
23.	Schwerpunktkoordinate in Y (Meter) (Y_{T0})	31.	Rotation um Y (rad) (β_1)
24.	Schwerpunktkoordinate in Z (Meter) (Z_{T0})	32.	Rotation um Z (rad) (γ_1)
25.	Große Halbachse des Ellipsoids	33.	Maßstab (M_{T1})
26.	Kleine Halbachse des Ellipsoids	34.	Schwerpunktskoordinate in X (Meter) (X_{T01})
	Angaben zum Ellipsoidübergang zum Neusystem aus dem Zwischensystem	35.	Schwerpunktskoordinate in Y (Meter) (Y_{T01})
27.	Translation in X (Meter) (X_{TE1})	36.	Schwerpunktskoordinate in Z (Meter) (Z_{T01})

Tabelle 4.2: Inhalt des Parametervektors für das Zielsystem

4.2.4 Das Unterprogramm Transr_dll

Dieses Unterprogramm dient der Transformation eines Einzelpunktes. Als Eingabewerte sind hierbei die dreidimensionalen Koordinaten, die Beschreibung des Ausgangs- und des Zielsystems mit den Parametern, den Ellipsoiden und den Abbildungsarten, die Art der Höhenangabe des Ausgangssystems, solange diese im Koordinatensatz enthalten ist, und die Pfadangabe für die Steuerungsdatei der Höhen (Datei hoehe.ini) erforderlich. Als Ausgabedaten werden die transformierten Koordinaten grundsätzlich mit den beiden Höhenangaben auf dem Ellipsoid und dem Geoid (Normalhöhe), soweit das Zielsystem mit einer Höhenangabe verbunden ist, zurückgegeben. Bei der Angabe von . (Punkt) für das Verzeichnis wird das aktuelle Verzeichnis beim Suchen nach der Datei hoehe.ini genutzt.

Bei den Datentypen ist der Unterschied zwischen den in C⁺⁺ genutzten und den eigenen zu beachten. Bei Texten ist außerdem darauf zu achten, dass eventuell noch eine Wandlung von Unicode- in die ASCII-Verschlüsselung vorgenommen werden muss.

Bei einem erfolgreichen Aufruf erhält man im Returncode den Zahlenwert 0, ansonsten Zahlenwerte größer als Null, die auf einen in 8.3 dokumentierten Fehler hinweisen, welche sich mit dem Unterprogramm Transfm_dll in ausformulierte Fehler- bzw. Hinweismeldungen erweitern lassen. Der Rückgabewert ist eine Ganzzahl, die in 4 Bytes gespeichert ist.

Definition: `extern "C" __declspec(dllexport) long transr_dll(double xin[3], wchar_t abb[14], wchar_t abba[14], double xout[4], wchar_t ellip[11], double paraa[36], wchar_t ellipa[11], double paraz[36], long streifen, long hart, long harta, wchar_t verz[299]);`

Anmerkungen:

- long wird in Visual Studio 2010 C⁺⁺ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
- wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 2010 C⁺⁺
- double wird in Visual Studio 2010 C⁺⁺ als Fließzahlvariable mit 8 Bytes mit 15 signifikanten Stellen gespeichert

Bedeutung der Parameter:**Ausgangsparameter:**

<code>double xin[3]</code>	.. Vektor als Fließkommavariablen in 8 Bytes mit der X-Koordinate der geozentrischen und ebenen Systeme bzw. die geographische Breite, der Y-Koordinate der geozentrischen und ebenen Systeme bzw. die geographische Länge und der Z-Koordinate der geozentrischen Systeme bzw. die ellipsoidische Höhe oder die Normalhöhe
<code>wchar_t ellip[11]</code>	.. Name des Ellipsoids mit maximal 10 Zeichen in Unicode
<code>wchar_t abb[14]</code>	.. Abbildungsart mit maximal 13 Zeichen in Unicode
<code>double paraa[36]</code>	.. Vektor zum Speichern der 36 Parameter des Ausgangssystems im Datentyp <code>double</code> (Startwert: 0; Anzahl der Elemente: 36) (siehe 8.2.2).
<code>wchar_t ellipa[11]</code>	.. Name des Ellipsoids mit maximal 10 Zeichen in Unicode
<code>wchar_t abba[14]</code>	.. Abbildungsart mit maximal 13 Zeichen in Unicode
<code>double paraz[36]</code>	.. Vektor zum Speichern der 36 Parameter des Zielsystems im Datentyp <code>double</code> . (Startwert: 0; Anzahl der Elemente: 36) (siehe 8.2.3).
<code>wchar_t verz[299]</code>	.. Verzeichnis der Datei <code>hoehe.ini</code> mit maximal 298 Zeichen in Unicode
<code>long streifen</code>	.. Zonen- bzw. Streifennummer des Zielsystems als Ganzzahl in 4 Bytes
<code>long hart</code>	.. Höhenart des Ausgangssystems (Ellipsoidische Höhe .. 0, Normalhöhe .. -1) als Ganzzahl in 4 Bytes
<code>long harta</code>	.. Höhenart des Zielsystems (Ellipsoidische Höhe .. 0, Normalhöhe .. -1) als Ganzzahl in 4 Bytes

Rückgabeparameter:

<code>double xout[4]</code>	.. Vektor mit Fließkommavariablen in 8 Bytes, der Inhalt besteht aus der X-Koordinate der geozentrischen oder ebenen Systeme bzw. der geographischen Breite, der Y-Koordinate der geozentrischen oder ebenen Systeme bzw. der geographischen Länge, der Z-Koordinate der geozentrischen Systeme bzw. der ellipsoidischen Höhe und der Normalhöhe in allen nichtgeozentrischen Systemen
-----------------------------	--

4.3 Das Unterprogramm Transfm_dll

Das Unterprogramm dient der Erzeugung von langschriftlichen Fehler- bzw. Hinweismeldungen. Zur Steuerung wird der Returncode der Unterprogramme `Trans_dll`, `TransID_dll`, `Transla_dll`, `Translz_dll` oder `Transr_dll` genutzt. Außerdem kann ein Dateiname angegeben werden, um zu dokumentieren in welcher Eingabedatei, der Fehler aufgetreten ist. Die zugehörige Beschreibung ist den aufgeführten Fehler- und Hinweismeldungen in 8 zu entnehmen. Die Nutzung dieses Unterprogramms ist in den Beispielprogrammen aufgeführt. Das Unterprogramm erzeugt keinen Rückgabewert, sondern nur die Meldung, solange zu dem gegebenen Code eine Meldung vorhanden ist. Ausnahmen stellen die Meldungen zu den Codes von 600 bis 799 dar, da hier aus der Codezahl sowohl die Fehlerart als auch die Nummer der Angabe als Codezahl abzüglich 599 bzw. 699 ermittelt wird.

Definition: `extern "C" __declspec(dllexport) void transfm_dll(long nr, wchar_t datei[200], wchar_t melde[1000]);`

Anmerkungen:

- long wird in Visual Studio 2010 C++ als Ganzzahlvariable mit Vorzeichen in 4 Bytes gespeichert
- wchar_t ist die interne Festlegung für den Datentyp char mit Zeichen in 2 Bytes-Verschlüsselung (Unicode) in Visual Studio 2010 C++

Bedeutung der Parameter:

Ausgangsparameter:

long nr	.. Fehlernummer, für die eine Fehlermeldung erzeugt werden soll, als Ganzzahl in 4 Bytes
wchar_t datei[200]	.. Dateiname, der in der Fehlermeldung berücksichtigt werden soll, mit 199 Zeichen in Unicode.

Rückgabeparameter:

wchar_t melde[1000]	.. Fehlermeldung mit maximal 999 Zeichen in Unicode.
---------------------	--

5. Notwendige Daten

5.1 Steuerungsdateien

Alle Steuerungsdateien (Steuerungsdateien für die Koordinatentransformationen DEFAULT und *.TRA) werden so interpretiert, dass Zeilen, die mit # beginnen, als Kommentarzeilen verwendet werden und Leerzeilen überlesen werden. Weiterhin werden Zeichenketten innerhalb einer Zeile beginnend mit dem Zeichen # bis zum Ende der Zeile als Kommentar interpretiert. Numerische Einträge und das Zeichen „, für die alphanumerischen Einträge müssen als erstes Zeichen ungleich einem Leerzeichen in der Zeile stehen.

5.1.1 Steuerungsdatei für die Geoidstützpunkte

Bei der Datei hoehe.ini handelt es sich um eine verschlüsselte Textdatei, die von den Programmen der Bibliothek Trans3Dll zur Berechnung der relativen Höhenanomalien auf dem Bessel-Ellipsoid benutzt wird. Diese Datei ist schreibgeschützt und darf vom Nutzer der Bibliotheksprogramme nicht geändert werden. Sollte diese Datei nicht mehr fehlerfrei verfügbar sein, kann nur vom Vertreiber der Bibliothek eine neue Datei zur Verfügung gestellt werden.

Diese Datei enthält Informationen zum Bessel-Ellipsoid, zum Ellipsoidübergang von GRS80 nach Bessel, zum Zentralpunkt Rauenberg und zu den absoluten Höhenanomalien.

5.1.2 Steuerungsdatei DEFAULT

In der Steuerungsdatei DEFAULT sind die Anzahl der zur Verfügung stehenden Parameterdateien, die zugehörigen Lagestatusschlüssel, die EPSG-Codes und die Namen der Parameterdateien verzeichnet. Der EPSG-Code ist eine Festlegung der Arbeitsgruppe der europäischen Öl- und Gaserkundungsunternehmen **European Petroleum Survey Group Geodesy (EPSG)** und wird auch von dieser verwaltet. 2005 wurde die Arbeitsgruppe durch das **Surveying and Positioning Committee der International Association of Oil & Gas Producers (OGP)** ersetzt. Diese Angabe ist nicht zwingend erforderlich, dient nur der Information und muss auch nicht enthalten sein. Bei der Angabe dieses Codes ist als Trennzeichen zwischen dem Lagestatus und dem EPSG-Code wenigstens ein Leerzeichen anzugeben.

Einträge in vorgegebener Reihenfolge:

- Anzahl der Koordinatensysteme (begrenzt auf 100) und Datenart (0 oder keine Angabe für unverschlüsselt, 1 für verschlüsselt)
- Lagestatuskennzahl des ersten Koordinatensystems und durch Leerzeichen abgetrennter EPSG-Code, falls vorhanden
- Name der ersten Parameterdatei eingefasst in „ und ohne den Suffix TRA bzw. TR0
- ...
- Lagestatuskennzahl des letzten Koordinatensystems und durch Leerzeichen abgetrennter EPSG-Code, falls vorhanden
- Name der letzten Parameterdatei eingefasst in „ und ohne den Suffix TRA bzw. TR0

5.1.3 Parameterdateien

Die Programme der Bibliothek Trans3Dll verarbeiten sowohl unverschlüsselte als auch verschlüsselte Parameterdateien mit dem Namen <Name>.TRA bzw. <Name>.TR0. Die Entscheidung über die Datenart wird durch den entsprechenden Parameter in der Datei DEFAULT getroffen.

5.1.3.1 Unverschlüsselte Parameterdateien mit dem Suffix TRA

In den Parameterdateien mit dem Namen <Name>.TRA, wobei der Name in DEFAULT eingetragen werden muss, sind die Parameter zur Definition des Koordinatensystems und für die Ellipsoidübergänge zum Besselipsoid abgelegt.

Einträge in vorgegebener Reihenfolge:

lfd. Nr.	Eintrag	Datentyp	zulässige Einträge
1.	Lagestatuskennzahl	Integer	identisch zu Eintrag in DEFAULT
2.	Bezeichnung (beliebiger Text)	char(100) ²	alphanumerisch in „
3.	Ellipsoid	char(11) ²	alphanumerisch in „
4.	Abbildungsart	char(14) ²	„Soldner“, „Gauss-Krueger“,
5.	Geographische Breite des Nullpunktes in dezimalen Altgrad	Float	„Geographisch“ und
6.	Geographische Länge des Nullpunktes in dezimalen Altgrad ¹	Float(, Int)	„Geozentrisch“
7.	Additionskonstante in Y bzw. im Rechtswert in Meter (Y _N) ³	Float	numerisch
8.	Additionskonstante in X bzw. im Hochwert in Meter (X _N) ³	Float	numerisch
9.	Dekadische Ergänzung (E) ³	Float	numerisch
10.	Maßstab des Meridians	Float	numerisch
	Angaben zur ebenen Verdrehung		
11.	Nullpunktcoordinate in Y (Meter) (Y _V) ³	Float	numerisch
12.	Nullpunktcoordinate in X (Meter) (X _V) ³	Float	numerisch
13.	Verdrehung (rad) (D) ³	Float	numerisch
14.	Verschiebung in Y (Meter) (dy) ³	Float	numerisch
15.	Verschiebung in X (Meter) (dx) ³	Float	numerisch

lfd. Nr.	Eintrag	Datentyp	zulässige Einträge
	Angaben zur ebenen Maßstabskorrektur		
16.	Nullpunktkoord. in Y (Meter) (Y_M) ³	Float	numerisch
17.	Nullpunktkoord. in X (Meter) (X_M) ³	Float	numerisch
18.	Maßstab (M) ³	Float	numerisch
19.	Große Halbachse des Ellipsoids	Float	numerisch
20.	Kleine Halbachse des Ellipsoids	Float	numerisch
	Angaben zum Ellipsoidübergang nach Bessel bzw. in das Zwischensystem (erste Angabe nach Bessel; zweite Angabe von Bessel)		
21.	Translation in X (Meter) (X_{TE}) ³	Float, Float	numerisch, numerisch
22.	Translation in Y (Meter) (Y_{TE}) ³	Float, Float	numerisch, numerisch
23.	Translation in Z (Meter) (Z_{TE}) ³	Float, Float	numerisch, numerisch
24.	Rotation um X (rad) (α) ³	Float, Float	numerisch, numerisch
25.	Rotation um Y (rad) (β) ³	Float, Float	numerisch, numerisch
26.	Rotation um Z (rad) (γ) ³	Float, Float	numerisch, numerisch
27.	Maßstab (M_T) ³	Float, Float	numerisch, numerisch
28.	Schwerpunktkoordinate in X (Meter) (X_{T0}) ³	Float, Float	numerisch, numerisch
29.	Schwerpunktkoordinate in Y (Meter) (Y_{T0}) ³	Float, Float	numerisch, numerisch
30.	Schwerpunktkoordinate in Z (Meter) (Z_{T0}) ³	Float, Float	numerisch, numerisch
	Angaben zum Ellipsoidübergang vom Neusystem in das Zwischensystem (erste Angabe in und zweite Angabe aus dem Zwischensystem)		
31.	Translation in X (Meter) (X_{TE1}) ³	Float, Float	numerisch, numerisch
32.	Translation in Y (Meter) (Y_{TE1}) ³	Float, Float	numerisch, numerisch
33.	Translation in Z (Meter) (Z_{TE1}) ³	Float, Float	numerisch, numerisch
34.	Rotation um X (rad) (α_1) ³	Float, Float	numerisch, numerisch
35.	Rotation um Y (rad) (β_1) ³	Float, Float	numerisch, numerisch
36.	Rotation um Z (rad) (γ_1) ³	Float, Float	numerisch, numerisch
37.	Maßstab (M_{T1}) ³	Float, Float	numerisch, numerisch
38.	Schwerpunktskoordinate in X (Meter) (X_{T01}) ³	Float, Float	numerisch, numerisch
39.	Schwerpunktskoordinate in Y (Meter) (Y_{T01}) ³	Float, Float	numerisch, numerisch
40.	Schwerpunktskoordinate in Z (Meter) (Z_{T01}) ³	Float, Float	numerisch, numerisch

Tabelle 5.1: Aufbau der Parameterdateien

¹ Der Eintrag für die geographische Länge wird bei Gauß-Krüger- und UTM-Systemen entweder als fester Meridian oder als Berechnungshinweis für die automatische Zuordnung genutzt. Hierzu wird bei einem Eintrag von größer als 360° die Differenz in den Vorkommastellen zu 600 als Additionsglied und die Nachkommastelle als Multiplikationsfaktor genutzt. Damit steht das Additionsglied als Beginn der Zählung für die Mittelmeridiane bzw. Zonen und der Multiplikationsfaktor gibt die Streifen- bzw. Zonenbreite in Altgrad an. Ferner kann bei einem festen Eintrag für die Länge zusätzlich als nachfolgender Ganzzahleintrag die Streifen- bzw. Zonennummer in der gleichen Zeile eingetragen werden.

² Die Länge der Zeichenkette muss ein Zeichen kleiner als die Dimension der Variablen sein.

³ Die Bezeichnungen beziehen sich auf die Formeln in 3.2.

5.1.3.2 Verschlüsselte Parameterdateien mit dem Suffix TR0

In den Parameterdateien mit dem Namen <Name>.TR0, wobei der Name in DEFAULT eingetragen werden muss, sind die Parameter zur Definition des Koordinatensystems und für die Ellipsoidübergänge zum Besselellipsoid in verschlüsselter Form abgelegt.

6. Nutzung der Bibliothek

Nach folgend ist beispielhaft Quellcode zur Nutzung der Unterprogramme aus der Bibliothek Trans3dll.dll in den Programmiersprachen C++, C# und VB.Net angegeben.

6.1 Programmiersprache C++

Der nachfolgende Quellcode mit der Header-Datei und dem Testquellcode in C ist in der Programmierungsumgebung Microsoft Visual Studio in der Version 2010 aufgebaut worden. Hierzu ist die Bibliothek trans3dll.lib einzubinden und die Bibliothek trans3dll.dll muss im gleichen Verzeichnis wie das ausführbare aufrufende Programm stehen.

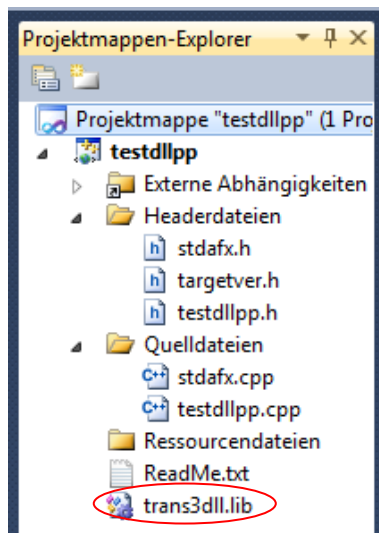


Abbildung 6.1: Einbinden von trans3dll.lib

Header-Datei Testdllpp.h

```
#define DLLImport extern "C" __declspec(dllimport)

DLLImport long trans_dll(double x, double y, double z, long l1, long l2, const wchar_t kat[100],
    wchar_t a_hk[2], double xout[4], wchar_t l_koord[100], long epsg[2], wchar_t ellip[11],
    wchar_t abb[14], wchar_t z_koord[100], wchar_t ellipa[100], wchar_t abba[14]);
DLLImport long translz_dll(const wchar_t kat[100], wchar_t z_name[100], long dart, long z_status,
    wchar_t z_koord[100], wchar_t ellipa[11], wchar_t abba[14], long &streifen, double parab[36]);
DLLImport long transla_dll(const wchar_t kat[100], wchar_t l_name[100], long dart, long l_status,
    wchar_t l_koord[100], wchar_t ellip[11], wchar_t abb[14], double paraa[36]);
DLLImport long transld_dll(const wchar_t kat[100], long &dart, long &anz, long nr[100], long epsg[100],
    wchar_t datei[100][100]);
DLLImport void transfm_dll(long nr, wchar_t datei[200], wchar_t melde[1000]);
DLLImport long transr_dll(double xin[3], wchar_t abb[14], wchar_t abba[14], double xout[4],
    wchar_t ellip[11], double paraa[36], wchar_t ellipa[11], double paraz[36], long streifen, long hart,
    long harta, wchar_t verz[299]);
```

Quellcode Testdllpp.cpp

```
#include "stdafx.h"
#include <wchar.h>
#include <stdlib.h>
#include <string.h>
#include "testdllpp.h"
```

```

#include <time.h>
#include <direct.h>
#include <io.h>
#include <fcntl.h>

struct tm *zeit;
struct tm *localtime();
long time(), uhr;

int main(int argc, char *argv[])
{
    long epsg[2], eps[100];
    long iret, streifen, dart, anz, nr[100], l_status, z_status;
    int icount, fehler, von;
    double x, y, z, xout[4], paraa[36], paraz[36], xin[3];
    wchar_t l_koord[100], ellip[11], abb[14], z_koord[100], ellipa[11], abba[14], kat[100], a_hk[2];
    wchar_t datei[100], ausgabe[1000], param[100][100], melde[1000];
    time_t rawtime;
    _setmode(_fileno(stdout), _O_U16TEXT);

    wprintf(L"Programm T E S T D L L P \n");
    wprintf(L"(c) Dr. Bergmann 2015, Version 1.0\n\n");

    time(&rawtime);
    uhr = time(0L);
    zeit = localtime(&rawtime);
    wprintf(L"Bearbeitung: %02d.%02d.%4d %02d:%02dUhr\n", zeit->tm_mday, (zeit->tm_mon) + 1,
        1900 + (zeit->tm_year), zeit->tm_hour, zeit->tm_min);
    if (argc != 7)
    {
        wprintf(L"Aufruf: testdllpp <y> <x> <z> <l1> <l2> <ha>\n");
        wprintf(L"Eingabedaten: - <y> <x> <z> als Punktkoordinaten\n");
        wprintf(L"                  - <l1> <l2> als Lagestatuskennzahlen\n");
        wprintf(L"                  - <ha> als Kennzahl der Höhenangabe\n");
        wprintf(L"Ergebnisse: - Wiedergabe der Eingabedaten\n");
        wprintf(L"                  - Ergebnisse der Aufrufe der Unterprogramme\n");
        wprintf(L"                  - trans_dll\n");
        wprintf(L"                  - transld_dll\n");
        wprintf(L"                  - transla_dll\n");
        wprintf(L"                  - translz_dll\n");
        wprintf(L"                  - translr_dll\n");
        exit(1);
    }
    y = atof(argv[1]);
    x = atof(argv[2]);
    z = atof(argv[3]);
    l_status = atol(argv[4]);
    z_status = atol(argv[5]);
    a_hk[0] = argv[6][0];
    a_hk[1] = '\\0';
    wprintf(L"Ausgangsdaten:\n");
    if (_wgetcwd(kat, 100) == NULL)
        perror("_wgetcwd error");
    else
        wprintf(L"Akt. Verzeichnis: %ls\n", kat);
    if (a_hk[0] == 'e') a_hk[0] = 'E';
    if (a_hk[0] == 'n') a_hk[0] = 'N';
    if (a_hk[0] != 'E' && a_hk[0] != 'N') a_hk[0] = 'E';
    wcsncpy(kat, L"parameter");
    wprintf(L"x          : %.31f\n", x);
    wprintf(L"y          : %.31f\n", y);
    wprintf(L"z          : %.31f\n", z);
    wprintf(L"l_status   : %ld\n", l_status);
    wprintf(L"z_status   : %ld\n", z_status);
    wprintf(L"kat        : %ls\n", kat);
    wprintf(L"a_hk benutzt : %lc\n", a_hk[0]);
    wprintf(L"a_hk gegeben : %lc\n", argv[6][0]);
    // Umrechnung mit trans
    iret = trans_dll(x, y, z, l_status, z_status, kat, a_hk, xout, l_koord, epsg, ellip, abb, z_koord,
        ellipa, abba);
    wprintf(L"\nUmrechnung mit trans_dll:\n");
    wprintf(L"Return-Code von trans_dll : %ld\n", iret);
    if (iret == 0 || iret > 1000)

```

```

{
    wprintf(L"x                : %.31fm\n", xout[0]);
    wprintf(L"y                : %.31fm\n", xout[1]);
    wprintf(L"z                : %.31fm\n", xout[2]);
    if(wcsncmp(abb, L"Geozentrisch", 12) != 0)
        wprintf(L"nn                : %.31fm\n", xout[3]);
    wprintf(L"Ausgangskoordinatensystem : %ls\n", l_koord);
    if(epsg[0]>-1) wprintf(L"EPSG-Code          : %ld\n", epsg[0]);
    else wprintf(L"EPSG-Code          : unbekannt\n");
    wprintf(L"Abbildungsart Ausg.      : %ls\n", abb);
    wprintf(L"Ellipsoid Ausg.          : %ls\n", ellip);
    wprintf(L"Zielkoordinatensystem    : %ls\n", z_koord);
    if(epsg[1]>-1) wprintf(L"EPSG-Code          : %ld\n", epsg[1]);
    else wprintf(L"EPSG-Code          : unbekannt\n");
    wprintf(L"Abbildungsart Ziel      : %ls\n", abba);
    wprintf(L"Ellipsoid Ziel          : %ls\n\n", ellipa);
    if(iret>1000)
    {
        wcscpy(ausgabe,L"Hinweis ");
        swprintf(datei,L"%ld: ",iret);
        wcscat(ausgabe,datei);
        transfm_dll(iret,L"hoehe.ini",melde);
        wcscat(ausgabe,melde);
        wcscat(ausgabe,L"\n");
        wprintf(L"%ls",ausgabe);
    }
}
else
{
    if(iret>0 && iret<600)
    {
        wcscpy(ausgabe,L"Fehler ");
        swprintf(datei,L"%ld: ",iret);
        wcscat(ausgabe,datei);
        if(iret<3) transfm_dll(iret,L"DEFAULT",melde);
        else transfm_dll(iret,L"Parameterdatei",melde);
        wcscat(ausgabe,melde);
        wcscat(ausgabe,L"\n");
        wprintf(L"%ls",ausgabe);
    }
    if(iret>599 && iret<800)
    {
        wcscpy(ausgabe,L"Fehler ");
        swprintf(datei,L"%ld: ",iret);
        wcscat(ausgabe,datei);
        transfm_dll(iret,L"DEFAULT",melde);
        wcscat(ausgabe,melde);
        wcscat(ausgabe,L"\n");
        wprintf(L"%ls",ausgabe);
    }
}
// schrittweise Bearbeitung
wprintf(L"schrittweise Bearbeitung:\n\n");
wprintf(L"translD_dll:\n");
iret = translD_dll(kat, dart, anz, nr, eps, param);
wprintf(L"Return-Code von translD_dll: %ld\n", iret);
if (iret == 0)
{
    if(dart==0) wprintf(L"Datenart                : %ld (unverschlüsselt, Suffix tra)\n", dart);
    else wprintf(L"Datenart                : %ld (verschlüsselt, Suffix tr0)\n", dart);
    wprintf(L"Anzahl der Parametersätze : %ld\n", anz);
    wprintf(L"\nLagestatl  EPSG-Code  Dateiname ohne Suffix\n");
    for (icount = 0; icount<anz ; icount++)
    {
        if(eps[icount]!=-1)
            wprintf(L"    %3ld      %5ld      %ls\n", nr[icount],eps[icount],param[icount]);
        else wprintf(L"    %3ld      unbekannt      %ls\n", nr[icount],param[icount]);
    }
    wprintf(L"\n");
}
else
{
    wcscpy(ausgabe,L"Fehler ");

```

```

    swprintf(datei,L"%ld: ",iret);
    wcscat(ausgabe,datei);
    wcscpy(datei,kat);
    wcscat(datei,L"\\DEFAULT");
    transfm_dll(iret,datei,melde);
    wcscat(ausgabe,melde);
    wcscat(ausgabe,L"\n");
    wprintf(L"%ls",ausgabe);
}
wprintf(L"\ntransla_dll:\n");
// Suchen der zugehoerigen Parameterdatei
von = -1;
for (icount = 0; icount<anz; icount++)
{
    if (l_status == nr[icount])
    {
        von = icount;
        break;
    }
}
if (von>-1)
{
    fehler = 0;
    iret = transla_dll(kat, param[von], dart, l_status, l_koord, ellip, abb, paraa);
    wprintf(L" Return-Code von transla_dll: %ld\n", iret);
    if(iret==0)
    {
        wprintf(L" Ausgangskoordinatensystem : %ls\n", l_koord);
        wprintf(L" Abbildungsart Ausg. : %ls\n", abb);
        wprintf(L" Ellipsoid Ausg. : %ls\n", ellip);
        wprintf(L" Parameter Ausg. : %14lf %14lf %14lf\n",paraa[0],paraa[1],paraa[2]);
        for (icount = 3; icount<35; icount=icount+3)
        {
            wprintf(L" %14lf %14lf %14lf\n",
                paraa[icount],paraa[icount+1],paraa[icount+2]);
        }
    }
    else
    {
        fehler=1;
        wcscpy(ausgabe,L"Fehler ");
        swprintf(datei,L"%ld: ",iret);
        wcscat(ausgabe,datei);
        wcscpy(datei,kat);
        wcscat(datei,L"\\");
        wcscat(datei,param[von]);
        if(dart==0)wcscat(datei,L".tra");
        else wcscat(datei,L".tr0");
        transfm_dll(iret,datei,melde);
        wcscat(ausgabe,melde);
        wcscat(ausgabe,L"\n");
        wprintf(L"%ls",ausgabe);
    }
}
else
{
    fehler = 1;
    wcscpy(ausgabe,L"Fehler 93: ");
    wcscpy(datei,L "");
    transfm_dll(93,datei,melde);
    wcscat(ausgabe,melde);
    wcscat(ausgabe,L"\n");
    wprintf(L"%ls",ausgabe);
}
wprintf(L"\ntranslz_dll:\n");
// Suchen der zugehoerigen Parameterdatei
von = -1;
for (icount = 0; icount<anz; icount++)
{
    if (z_status == nr[icount])
    {
        von = icount;
        break;
    }
}

```

```

    }
}
if (von>-1)
{
    iret = translz_dll(kat, param[von], dart, z_status, z_koord, ellipa, abba, streifen, paraz);
    wprintf(L" Return-Code von translz_dll: %ld\n", iret);
    if(iret==0)
    {
        wprintf(L" Zielkoordinatensystem      : %ls\n", z_koord);
        wprintf(L" Abbildungsart Ziel          : %ls\n", abba);
        wprintf(L" Ellipsoid Ziel              : %ls\n", ellipa);
        wprintf(L" Streifen Ziel                  : %ld\n", streifen);
        wprintf(L" Parameter Ziel                  : %14lf %14lf %14lf\n", paraz[0], paraz[1], paraz[2]);
        for (icount = 3; icount<35; icount=icount+3)
        {
            wprintf(L"                                %14lf %14lf %14lf\n",
                paraz[icount], paraz[icount+1], paraz[icount+2]);
        }
    }
    else
    {
        fehler = fehler + 1;
        wcscpy(ausgabe, L"Fehler ");
        swprintf(datei, L"%ld: ", iret);
        wcscat(ausgabe, datei);
        wcscpy(datei, kat);
        wcscat(datei, L"\n");
        wcscat(datei, param[von]);
        if(dart==0) wcscat(datei, L".tra");
        else wcscat(datei, L".tr0");
        transfm_dll(iret, datei, melde);
        wcscat(ausgabe, melde);
        wcscat(ausgabe, L"\n");
        wprintf(L"%ls", ausgabe);
    }
}
else
{
    fehler = fehler + 1;
    wcscpy(ausgabe, L"Fehler 93: ");
    wcscpy(datei, L"");
    transfm_dll(93, datei, melde);
    wcscat(ausgabe, melde);
    wcscat(ausgabe, L"\n");
    wprintf(L"%ls", ausgabe);
}
if (fehler == 0)
{
    wprintf(L"\ntransr_dll:\n");
    xin[0]=x;
    xin[1]=y;
    xin[2]=z;
    if ((a_hk[0] != 'N' || wcsncmp(abb, L"Geozentrisch", 12) == 0) && wcsncmp(abba, L"Geozentrisch", 12) == 0)
        iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, 0, 0, kat);
    if ((a_hk[0] != 'N' || wcsncmp(abb, L"Geozentrisch", 12) == 0) && wcsncmp(abba, L"Geozentrisch", 12) != 0)
        iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, 0, -1, kat);
    if (a_hk[0] == 'N' && wcsncmp(abb, L"Geozentrisch", 12) != 0 && wcsncmp(abba, L"Geozentrisch", 12) == 0)
        iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, -1, 0, kat);
    if (a_hk[0] == 'N' && wcsncmp(abb, L"Geozentrisch", 12) != 0 && wcsncmp(abba, L"Geozentrisch", 12) != 0)
        iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, -1, -1, kat);
    wprintf(L"Return-Code von transr_dll : %ld\n", iret);
    if (iret == 0 || iret>1000)
    {
        wprintf(L"x                                : %.31fm\n", xout[0]);
        wprintf(L"y                                : %.31fm\n", xout[1]);
        wprintf(L"z                                : %.31fm\n", xout[2]);
        if(wcsncmp(abba, L"Geozentrisch", 12) != 0)
            wprintf(L"nn                                : %.31fm\n", xout[3]);
        if(iret>1000)
        {
            wcscpy(ausgabe, L"Hinweis ");
            swprintf(datei, L"%ld: ", iret);
            wcscat(ausgabe, datei);
        }
    }
}

```

```

        wscpy(datei,kat);
        wscat(datei,L"\\hoehe.ini");
        transfm_dll(iret,datei,melde);
        wscat(ausgabe,melde);
        wscat(ausgabe,L"\n");
        wprintf(L"%ls",ausgabe);
    }
}
else
{
    wscpy(ausgabe,L"Fehler ");
    swprintf(datei,L"%ld: ",iret);
    wscat(ausgabe,datei);
    wscpy(datei,L"");
    transfm_dll(iret,datei,melde);
    wscat(ausgabe,melde);
    wscat(ausgabe,L"\n");
    wprintf(L"%ls",ausgabe);
}
}
}

```

Zugehörige Bildschirmanzeige

```

C:\Visual Studio 2010\Projects\testdllpp\Release>testdllpp 100 100 30 600 493 n
Programm T E S T D L L P P
(c) Dr. Bergmann 2015, Version 1.0

```

Bearbeitung: 27.05.2015 12:12Uhr

Ausgangsdaten:

```

Akt. Verzeichnis: C:\Visual Studio 2010\Projects\testdllpp\Release
x                : 100.000
y                : 100.000
z                : 30.000
l_status         : 600
z_status         : 493
kat              : parameter
a_hk benutzt     : N
a_hk gegeben     : n

```

Umrechnung mit trans_dll:

Return-Code von trans_dll : 0

```

x                : 5808434.939m
y                : 33406618.788m
z                : 69.293m
nn              : 30.000m
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
EPSG-Code        : 36000
Abbildungsart Ausg. : Soldner
Ellipsoid Ausg.   : Bessel
Zielkoordinatensystem : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
EPSG-Code        : 5650
Abbildungsart Ziel : Gauss-Krueger
Ellipsoid Ziel    : GRS80

```

Schrittweise Bearbeitung:

transld_dll:

Return-Code von transld_dll: 0

Datenart : 0 (unverschlüsselt, Suffix tra)

Anzahl der Parametersätze : 43

Lagestati	EPSG-Code	Dateiname ohne Suffix
100	unbekannt	100
102	31466	102

103	31467	103
104	31468	104
105	31469	105
111	5678	111
112	5679	112
130	unbekannt	130
133	unbekannt	133
134	unbekannt	134
135	unbekannt	135
140	unbekannt	140
142	5664	142
143	5665	143
150	unbekannt	150
153	5673	153
154	5674	154
155	5675	155
350	5828	350
384	4328	384
389	4346	389
400	unbekannt	400
432	23032	432
433	23033	433
489	unbekannt	489
491	5649	491
492	4647	492
493	5650	493
494	unbekannt	494
495	unbekannt	495
500	3068	500
589	unbekannt	589
600	36000	600
610	unbekannt	x610
620	unbekannt	x620
630	unbekannt	x630
640	unbekannt	x640
650	unbekannt	650
660	unbekannt	x660
815	4178	815
850	4314	850
884	4326	884
889	4258	889

transla_dll:

Return-Code von transla_dll: 0
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
Abbildungsart Ausg. : Soldner
Ellipsoid Ausg. : Bessel
Parameter Ausg. : 0.914878 13.627204 0.000000
 0.000000 100000.000000 1.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000
 0.000000 1.000000 0.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000
 6377397.155080 6356078.962900 0.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000
 0.000000 0.000000 0.000000

translz_dll:

Return-Code von translz_dll: 0
Zielkoordinatensystem : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
Abbildungsart Ziel : Gauss-Krueger
Ellipsoid Ziel : GRS80
Streifen Ziel : 33

```

Parameter Ziel      :      0.000000      15.000000      0.000000
                      0.000000      0.000000      0.999600
                      0.000000      0.000000      0.000000
                      500000.000000      0.000000      0.000000
                      0.000000      1.000000      900097.306000
                      3784880.652000 5037371.552000      0.000006
                      0.000003      0.000003      1.000000
                      900067.864000 3784246.245000 5036920.892000
                      6378137.000000 6356752.314140      0.000000
                      0.000000      0.000000      0.000000
                      0.000000      0.000000      0.000000
                      0.000000      0.000000      0.000000

transr_dll:
Return-Code von transr_dll : 0
x      : 5808434.939m
y      : 33406618.788m
z      : 69.293m
nn     : 30.000m

C:\Visual Studio 2010\Projects\testdllpp\Release>

```

6.2 Programmiersprache C[#]

Der nachfolgende Quellcode in C[#] ist in der Programmierumgebung Microsoft Visual Studio in der Version 2010 aufgebaut worden. Hierzu müssen die Datei transansii.exp und die Bibliothek transansii.dll im gleichen Verzeichnis wie das ausführbare aufrufende Programm vorliegen.

Quellcode Programm.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Runtime.InteropServices;

namespace test3dllcs
{
    class Program
    {
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern int trans_dll(double x, double y, double z, int l1, int l2, Char[] kat,
            Char[] a_hk, double[] xout, Char[] l_koord, int[] epsg, Char[] ellip, Char[] abb, Char[] z_koord,
            Char[] ellipa, Char[] abba);
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern int transld_dll(Char[] kat, out int dart, out int anz, int[] nr, int[] epsg,
            Char[] datei);
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern int transla_dll(Char[] kat, Char[] l_name, int dart, int l_status,
            Char[] l_koord, Char[] ellip, Char[] abb, double[] paraa);
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern int translz_dll(Char[] kat, Char[] z_name, int dart, int z_status,
            Char[] z_koord, Char[] ellipa, Char[] abba, out int streifen, double[] parab);
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern int transr_dll(double[] xin, Char[] abb, Char[] abba, double[] xout,
            Char[] ellip, double[] paraa, Char[] ellipa, double[] paraz, int streifen, int hart, int harta,
            Char[] verz);
        [DllImport("Trans3dll.dll", CharSet = CharSet.Auto)]
        public static extern void transfm_dll(int nr, Char[] datei, Char[] melde);

        static void Main(string[] args)
        {
            int icount, fehler, j;
            int iret, streifen=0, dart, anz, l_status, z_status;
            int[] nr = new int[100];
            int[] epsg = new int[2];
            int[] epsga = new int[100];

```



```

double x, y, z;
double[] xin = new double[3];
double[] xout = new double[4];
double[] paraa = new double[36];
double[] paraz = new double[36];
Char[] kat = new char[100];
Char[] a_hk = new char[2];
Char[] l_koord = new char[100];
Char[] ellip = new char[11];
Char[] abb = new char[14];
Char[] z_koord = new char[100];
Char[] ellipa = new char[11];
Char[] abba = new char[14];
Char[] l_name = new char[100];
Char[, ] datei = new char[100,100];
Char[] verz = new char[299];
Char[] name = new char[399];
char[] dummy1 = new char[1000];
string sabb, sabba;

Console.WriteLine("Programm T E S T D L L C S");
Console.WriteLine("(c) Dr. Bergmann 2015, Version 1.0");
Console.WriteLine(string.Format("Bearbeitung: {0} {1}", DateTime.Now.ToLongDateString(),
    DateTime.Now.ToLongTimeString()));
if (args.Length != 6)
{
    Console.WriteLine("Aufruf: testdllcs <y> <x> <z> <l1> <l2> <ha>");
    Console.WriteLine("Eingabedaten: - <y> <x> <z> als Punktkoordinaten");
    Console.WriteLine("                - <l1> <l2> als Lagestatuskennzahlen");
    Console.WriteLine("                - <ha> als Kennzahl der Höhenangabe");
    Console.WriteLine("Ergebnisse: - Wiedergabe der Eingabedaten");
    Console.WriteLine("                - Ergebnisse der Aufrufe der Unterprogramme");
    Console.WriteLine("                - trans_dll");
    Console.WriteLine("                - transld_dll");
    Console.WriteLine("                - transla_dll");
    Console.WriteLine("                - translz_dll");
    Console.WriteLine("                - translr_dll");
    return;
}
Console.WriteLine("Ausgangsdaten:");
if (!double.TryParse(args[0], out y)) y = 0.0;
if (!double.TryParse(args[1], out x)) x = 0.0;
if (!double.TryParse(args[2], out z)) z = 0.0;
if (!int.TryParse(args[3], out l_status)) l_status = 0;
if (!int.TryParse(args[4], out z_status)) z_status = 0;
args[5].CopyTo(0, a_hk, 0, 1);
Console.WriteLine("Akt. Verzeichnis: {0}", Environment.CurrentDirectory);
string dummy = "parameter";
dummy.CopyTo(0, kat, 0, dummy.Length);
Console.WriteLine(" y           : {0,5:f3}m", y);
Console.WriteLine(" x           : {0,5:f3}m", x);
Console.WriteLine(" z           : {0,5:f3}m", z);
Console.WriteLine(" l_status    : {0}", l_status);
Console.WriteLine(" z_status    : {0}", z_status);
for (icount = 0; icount < 2; icount++) { if (a_hk[icount] == '\\')break; }
if (a_hk[0] == 'n') a_hk[0] = 'N';
if (a_hk[0] != 'N') a_hk[0] = 'E';
Console.WriteLine(" a_hk       : " + new string(a_hk, 0, icount));
// Umrechnung mit trans
kat[0] = '.';
kat[1] = '.';
kat[2] = '\\';
dummy.CopyTo(0, kat, 3, dummy.Length);
verz[0] = '.';
verz[1] = '.';
verz[2] = '\\';
dummy.CopyTo(0, verz, 3, dummy.Length);
Console.WriteLine(" kat       : " + new string(kat, 0, 3+dummy.Length));
Console.WriteLine(" verz     : " + new string(verz, 0, 3+dummy.Length));
iret = trans_dll(x, y, z, l_status, z_status, kat, a_hk, xout, l_koord, epsg, ellip, abb,
    z_koord, ellipa, abba);
Console.WriteLine("");
Console.WriteLine("Umrechnung mit trans_dll:");

```

```

Console.WriteLine(" Return-Code von trans_dll : {0}", iret);
if (iret == 0 || iret>1000)
{
    sabb = "";
    for (icount = 0; icount < 14; icount++)
    { if (abb[icount] == '\0')break; sabb = sabb + abb[icount].ToString(); }
    Console.WriteLine(" x          : {0,5:f3}m", xout[0]);
    Console.WriteLine(" y          : {0,5:f3}m", xout[1]);
    Console.WriteLine(" z          : {0,5:f3}m", xout[2]);
    if(!sabb.Equals("Geozentrisch")) Console.WriteLine(" nn          : {0,5:f3}m",
        xout[3]);
    for (icount = 0; icount < 100; icount++) { if (l_koord[icount] == '\0')break; }
    Console.WriteLine(" Ausgangskoordinatensystem : " + new string(l_koord, 0, icount));
    Console.WriteLine(" Abbildungsart Ausg.      : " + sabb);
    for (icount = 0; icount < 11; icount++) { if (ellip[icount] == '\0')break; }
    Console.WriteLine(" Ellipsoid Ausg.          : " + new string(ellip, 0, icount));
    if(epsg[0]!=-1) Console.WriteLine(" EPSG-Code Ausg.          : {0}", epsg[0]);
    else Console.WriteLine(" EPSG-Code Ausg.          : unbekannt");
    for (icount = 0; icount < 100; icount++) { if (z_koord[icount] == '\0')break; }
    Console.WriteLine(" Zielkoordinatensystem    : " + new string(z_koord, 0, icount));
    for (icount = 0; icount < 14; icount++) { if (abba[icount] == '\0')break; }
    Console.WriteLine(" Abbildungsart Ziel       : " + new string(abba, 0, icount));
    for (icount = 0; icount < 11; icount++) { if (ellipa[icount] == '\0')break; }
    Console.WriteLine(" Ellipsoid Ziel          : " + new string(ellipa, 0, icount));
    if(epsg[1]!=-1) Console.WriteLine(" EPSG-Code Ziel         : {0}", epsg[1]);
    else Console.WriteLine(" EPSG-Code Ziel         : unbekannt");
    if(iret>1000)
    {
        name[0]='h'; name[1]='o'; name[2]='e'; name[3]='h'; name[4]='e';
        name[5]='.'; name[6]='i'; name[7]='n'; name[8]='i'; name[9]='\0';
        transfm_dll(iret, name, dummy1);
        Console.WriteLine(string.Format("Hinweis {0}: ", iret));
        for (icount = 0; icount < 1000; icount++)
        { if (dummy1[icount] == '\0')break; Console.WriteLine(dummy1[icount]); }
        Console.WriteLine("\n");
    }
}
else
{
    if((iret>0 && iret<3) || (iret>599 && iret<800))
    {
        name[0]='D'; name[1]='E'; name[2]='F'; name[3]='A'; name[4]='U';
        name[5]='L'; name[6]='T'; name[7]='\0';
        transfm_dll(iret, name, dummy1);
        Console.WriteLine(string.Format("Fehler {0}: ", iret));
        for (icount = 0; icount < 1000; icount++)
        { if (dummy1[icount] == '\0')break; Console.WriteLine(dummy1[icount]); }
        Console.WriteLine("");
    }
    if(iret>2 && iret<600)
    {
        name[0]='P'; name[1]='a'; name[2]='r'; name[3]='a'; name[4]='m';
        name[5]='e'; name[6]='t'; name[7]='e'; name[8]='r'; name[9]='d';
        name[10]='a'; name[11]='t'; name[12]='e'; name[13]='i'; name[14]='\0';
        transfm_dll(iret, name, dummy1);
        Console.WriteLine(string.Format("Fehler {0}: ", iret));
        for (icount = 0; icount < 1000; icount++)
        { if (dummy1[icount] == '\0')break; Console.WriteLine(dummy1[icount]); }
        Console.WriteLine("");
    }
}
// schrittweise Bearbeitung
Console.WriteLine("");
Console.WriteLine("schrittweise Bearbeitung:");
Console.WriteLine("translD_dll:");
iret = translD_dll(kat, out dart, out anz, nr, epsga, datei);
Console.WriteLine(" Return-Code von translD_dll: {0}", iret);
if (iret == 0)
{
    if(dart==0)
        Console.WriteLine(" Datenart          : {0} (unverschlüsselt, Suffix tra)", dart);
    else Console.WriteLine(" Datenart          : {0} (verschlüsselt, Suffix tr0)", dart);
    Console.WriteLine(" Anzahl der Parametersätze : {0}", anz);
}

```

```

Console.WriteLine("");
Console.WriteLine("Lagestatik EPSG-Code Dateiname ohne Suffix\n");
for (icount = 0; icount<anz ; icount++)
{
    if(epsga[icount]!=-1) Console.Write(" {0,3} {1,5} ", nr[icount],epsga[icount]);
    else Console.Write(" {0,3} unbekannt ", nr[icount]);
    for (j = 0; j < 100; j++)
    { if (datei[icount,j] == '\0')break; Console.Write(datei[icount,j]); }
    Console.WriteLine("");
}
Console.WriteLine("");
}
else
{
    kat.CopyTo(name,0);
    for (j = 0; j < 200; j++) { if (name[j] == '\0')break; }
    name[j]='\0'; name[j+1]='D'; name[j+2]='E'; name[j+3]='F'; name[j+4]='A'; name[j+5]='U';
    name[j+6]='L'; name[j+7]='T'; name[j+8]='\0';
    transfm_dll(iret, name, dummy1);
    Console.WriteLine(string.Format("Fehler {0}: ", iret));
    for (icount = 0; icount < 1000; icount++)
    { if (dummy1[icount] == '\0')break; Console.Write( dummy1[icount]); }
    Console.WriteLine("");
}
Console.WriteLine("");
Console.WriteLine("transla_dll:");
// Suchen der zugehörigen Parameterdatei
iret = -1;
for (icount = 0; icount < anz; icount++)
{
    if (l_status == nr[icount])
    {
        iret = icount;
        epsg[0]=epsga[icount];
        break;
    }
}
if (iret > -1)
{
    fehler = 0;
    for (icount = 0; icount < 100; icount++)
    {
        name[icount] = datei[iret,icount];
        if(datei[iret,icount]=='\0')break;
    }
    iret = transla_dll(kat, name, dart, l_status, l_koord, ellip, abb, paraa);
    Console.WriteLine(" Return-Code von transla_dll: {0}", iret);
    if(iret==0)
    {
        for (icount = 0; icount < 100; icount++) { if (l_koord[icount] == '\0')break; }
        Console.WriteLine(" Ausgangskoordinatensystem : " + new string(l_koord, 0, icount));
        for (icount = 0; icount < 14; icount++) { if (abb[icount] == '\0')break; }
        Console.WriteLine(" Abbildungsart Ausg. : " + new string(abb, 0, icount));
        for (icount = 0; icount < 11; icount++) { if (ellip[icount] == '\0')break; }
        Console.WriteLine(" Ellipsoid Ausg. : " + new string(ellip, 0, icount));
        Console.WriteLine(" EPSG-Code Ausg. : {0}",epsg[0]);
        Console.WriteLine(" Parameter Ausg. : {0,14:f6} {1,14:f6} {2,14:f6}",
            paraa[0],paraa[1],paraa[2]);
        for (icount = 3; icount<35; icount=icount+3)
        {
            Console.WriteLine(" {0,14:f6} {1,14:f6} {2,14:f6}",
                paraa[icount], paraa[icount + 1], paraa[icount + 2]);
        }
    }
    else
    {
        fehler = 1;
        for (j = 0; j < 200; j++) { if (name[j] == '\0')break; }
        name[j]='.'; name[j+1]='t'; name[j+2]='r';
        if(dart==0)name[j+3]='a';
        else name[j+3]='0';
        name[j+4]='\0';
        transfm_dll(iret, name, dummy1);
    }
}

```

```

        Console.WriteLine(string.Format("Fehler {0}: ", ired));
        for (icount = 0; icount < 1000; icount++)
        { if (dummy1[icount] == '\0')break; Console.WriteLine( dummy1[icount]); }
        Console.WriteLine("");
    }
}
else
{
    fehler = 1;
    name[0]='\0';
    transfm_dll(93, name, dummy1);
    Console.WriteLine(string.Format("Fehler 93: "));
    for (icount = 0; icount < 1000; icount++)
    { if (dummy1[icount] == '\0')break; Console.WriteLine( dummy1[icount]); }
    Console.WriteLine("");
}
Console.WriteLine("");
Console.WriteLine("translz_dll:");
// Suchen der zugehoerigen Parameterdatei
ired = -1;
for (icount = 0; icount < anz; icount++)
{
    if (z_status == nr[icount])
    {
        ired = icount;
        break;
    }
}
if (ired > -1)
{
    for (icount = 0; icount < 100; icount++)
    {
        name[icount] = datei[ired,icount];
        if(datei[ired,icount]=='\0')break;
    }
    ired = translz_dll(kat, name, dart, z_status, z_koord, ellipa, abba, out streifen, paraz);
    Console.WriteLine(" Return-Code von translz_dll: {0}", ired);
    if(ired==0)
    {
        for (icount = 0; icount < 100; icount++) { if (z_koord[icount] == '\0')break; }
        Console.WriteLine(" Zielkoordinatensystem      : " + new string(z_koord, 0, icount));
        for (icount = 0; icount < 14; icount++) { if (abba[icount] == '\0')break; }
        Console.WriteLine(" Abbildungsart Ziel      : " + new string(abba, 0, icount));
        for (icount = 0; icount < 11; icount++) { if (ellipa[icount] == '\0')break; }
        Console.WriteLine(" Ellipsoid Ziel          : " + new string(ellipa, 0, icount));
        Console.WriteLine(" Streifen Ziel          : {0}", streifen);
        Console.WriteLine(" Parameter Ziel          : {0,14:f6} {1,14:f6} {2,14:f6}",
            paraz[0],paraz[1],paraz[2]);
        for (icount = 3; icount<35; icount=icount+3)
        {
            Console.WriteLine("                                {0,14:f6} {1,14:f6} {2,14:f6}",
                paraz[icount], paraz[icount + 1], paraz[icount + 2]);
        }
    }
}
else
{
    fehler = fehler + 1;
    for (j = 0; j < 200; j++) { if (name[j] == '\0')break; }
    name[j]='.'; name[j+1]='t'; name[j+2]='r';
    if(dart==0)name[j+3]='a';
    else name[j+3]='0';
    name[j+4]='\0';
    transfm_dll(ired, name, dummy1);
    Console.WriteLine(string.Format("Fehler {0}: ", ired));
    for (icount = 0; icount < 1000; icount++)
    { if (dummy1[icount] == '\0')break; Console.WriteLine( dummy1[icount]); }
    Console.WriteLine("");
}
}
else
{
    fehler = fehler + 1;
    name[0] = '\0';

```


Zugehörige Bildschirmanzeige

```
C:\Visual Studio 2010\Projects\testdllcs\testdllcs\bin\Release>testdllcs 100 100 30 600 493 k
Programm T E S T D L L C S
(c) Dr. Bergmann 2015, Version 1.0
Bearbeitung: Mittwoch, 27. Mai 2015 14:03:03
Ausgangsdaten:
Akt. Verzeichnis: C:\Visual Studio 2010\Projects\testdllcs\testdllcs\bin\Release
y          : 100,000m
x          : 100,000m
z          : 30,000m
l_status   : 600
z_status   : 493
a_hk       : E
kat        : ..\parameter
verz       : ..\parameter

Umrechnung mit trans_dll:
Return-Code von trans_dll : 0
x          : 5808434,939m
y          : 33406618,788m
z          : 69,229m
nn         : 29,936m
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
Abbildungsart Ausg.       : Soldner
Ellipsoid Ausg.           : Bessel
EPSG-Code Ausg.           : 36000
Zielkoordinatensystem     : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
Abbildungsart Ziel        : Gauss-Krueger
Ellipsoid Ziel            : GRS80
EPSG-Code Ziel            : 5650

schrittweise Bearbeitung:
translD_dll:
Return-Code von translD_dll: 0
Datenart      : 0 (unverschlüsselt, Suffix tra)
Anzahl der Parametersätze : 43

Lagestati  EPSG-Code  Dateiname ohne Suffix

100      unbekannt    100
102      31466        102
103      31467        103
104      31468        104
105      31469        105
111      5678         111
112      5679         112
130      unbekannt    130
133      unbekannt    133
134      unbekannt    134
135      unbekannt    135
140      unbekannt    140
142      5664         142
143      5665         143
150      unbekannt    150
153      5673         153
154      5674         154
155      5675         155
350      5828         350
384      4328         384
389      4346         389
400      unbekannt    400
432      23032        432
433      23033        433
489      unbekannt    489
491      5649         491
```

492	4647	492
493	5650	493
494	unbekannt	494
495	unbekannt	495
500	3068	500
589	unbekannt	589
600	36000	600
610	unbekannt	x610
620	unbekannt	x620
630	unbekannt	x630
640	unbekannt	x640
650	unbekannt	650
660	unbekannt	x660
815	4178	815
850	4314	850
884	4326	884
889	4258	889

transla_dll:

Return-Code von transla_dll: 0
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
Abbildungsart Ausg. : Soldner
Ellipsoid Ausg. : Bessel
EPSG-Code Ausg. : 36000
Parameter Ausg. :

	0,914878	13,627204	0,000000
	0,000000	100000,000000	1,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	1,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	6377397,155080	6356078,962900	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000

translz_dll:

Return-Code von translz_dll: 0
Zielkoordinatensystem : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
Abbildungsart Ziel : Gauss-Krueger
Ellipsoid Ziel : GRS80
Streifen Ziel : 33
Parameter Ziel :

	0,000000	15,000000	0,000000
	0,000000	0,000000	0,999600
	0,000000	0,000000	0,000000
	500000,000000	0,000000	0,000000
	0,000000	1,000000	900097,306000
	3784880,652000	5037371,552000	0,000006
	0,000003	0,000003	1,000000
	900067,864000	3784246,245000	5036920,892000
	6378137,000000	6356752,314140	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000

transr_dll:

Return-Code von transr_dll: 0
x : 5808434,939m
y : 33406618,788m
z : 69,229m
nn : 29,936m

C:\Visual Studio 2010\Projects\testdllcs\testdllcs\bin\Release>

6.3 Programmiersprache VB.Net

Der nachfolgende Quellcode in Visual Basic ist in der Programmierumgebung Microsoft Visual Studio in der Version 2010 aufgebaut worden. Hierzu müssen die Datei transansii.exp und die Bibliothek transansii.dll im gleichen Verzeichnis wie das ausführbare aufrufende Programm vorliegen.

Quellcode Module1.vb

```
Imports System
Imports System.IO
Imports System.Text
Imports System.Runtime.InteropServices
Imports System.Security.Permissions

Module Module1
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Function trans_dll(ByVal x As Double, ByVal y As Double, ByVal z As Double, ByVal l1 As Integer, _
        ByVal l2 As Integer, ByVal kat As Char(), ByVal a_hk As Char(), ByVal xout As Double(), _
        ByVal l_koord As Char(), ByVal epsg As Integer(), ByVal ellip As Char(), ByVal abb As Char(), _
        ByVal z_koord As Char(), ByVal ellipa As Char(), ByVal abba As Char()) As Integer
    End Function
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Sub transfm_dll(ByVal nr As Integer, ByVal datei As Char(), ByVal melde As Char())
    End Sub
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Function transld_dll(ByVal kat As Char(), ByRef dart As Integer, ByRef anz As Integer, _
        ByVal nr As Integer(), ByVal epsg As Integer(), ByVal datei As Char(),) As Integer
    End Function
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Function transla_dll(ByVal kat As Char(), ByVal l_name As Char(), ByVal dart As Integer, _
        ByVal l_status As Integer, ByVal l_koord As Char(), ByVal ellip As Char(), ByVal abb As Char(), _
        ByVal paraa As Double()) As Integer
    End Function
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Function translz_dll(ByVal kat As Char(), ByVal z_name As Char(), ByVal dart As Integer, _
        ByVal z_status As Integer, ByVal z_koord As Char(), ByVal ellipa As Char(), ByVal abba As Char(), _
        ByRef streifen As Integer, ByVal paraa As Double()) As Integer
    End Function
    <DllImport("Trans3dll.dll", ExactSpelling:=True, SetLastError:=True, CharSet:=CharSet.Unicode)> _
    Function transr_dll(ByVal xin As Double(), ByVal abb As Char(), ByVal abba As Char(), _
        ByVal xout As Double(), ByVal ellip As Char(), ByVal paraa As Double(), ByVal ellipa As Char(), _
        ByVal paraz As Double(), ByVal streifen As Integer, ByVal hart As Integer, ByVal harta As Integer, _
        ByVal verz As Char()) As Integer
    End Function

    Sub Main()
        Dim dnow As DateTime = DateTime.Now
        Dim dString As String, dstring1 As String
        Dim x As Double, y As Double, z As Double, epsga(99) As Integer, datei(99, 99) As Char
        Dim fehler As Integer, paraa(35) As Double
        Dim l1 As Integer, l2 As Integer, iret As Integer, icount As Integer, dart As Integer
        Dim anz As Integer, nr(99) As Integer
        Dim hk As Char, kat(99) As Char, verz(99) As Char, a_hk(1) As Char, l_koord(99) As Char
        Dim streifen As Integer, paraz(35) As Double
        Dim ellip(10) As Char, abb(13) As Char, z_koord(99) As Char, ellipa(10) As Char, abba(13) As Char
        Dim epsg(1) As Integer, xout(3) As Double
        Dim name(199) As Char, melde(999) As Char, xin(3) As Double
        Console.WriteLine("Programm T E S T D L L V B")
        Console.WriteLine("(c) Dr. Bergmann, Version 1.0, 2015")
        dString = dnow.ToString()
        Console.WriteLine("Bearbeitung: {0}", dString)
        Console.WriteLine("")
        Console.WriteLine("Eingabe der Parameter: x, y, z, l1, l2, hk")
        Console.WriteLine("  x, y und z sind die Koordinaten des Ausgangssystems im Lagestatus l1")
        Console.WriteLine("  hk ist gleich die Höhenkennzahl im Ausgangssystem mit n oder N für")
        Console.WriteLine("  Normalhöhen, l2 ist der Lagestatus im Zielsystem, dessen Koordinaten über")
        Console.WriteLine("  die Programme der Bibliothek transdll.dll ermittelt werden")
        Console.WriteLine("Eingabe der x-Koordinate in m:")
        dString = Console.ReadLine()
        If Not Double.TryParse(dString, x) Then x = 0.0
    End Sub
End Module
```



```

Console.WriteLine("Eingabe der y-Koordinate in m:")
dString = Console.ReadLine()
If Not Double.TryParse(dString, y) Then y = 0.0
Console.WriteLine("Eingabe der z-Koordinate in m:")
dString = Console.ReadLine()
If Not Double.TryParse(dString, z) Then z = 0.0
Console.WriteLine("Eingabe der Lagestatusangabe des Ausgangssystems:")
dString = Console.ReadLine()
If Not Integer.TryParse(dString, l1) Then l1 = 0
Console.WriteLine("Eingabe der Lagestatusangabe des Zielsystems:")
dString = Console.ReadLine()
If Not Integer.TryParse(dString, l2) Then l2 = 0
Console.WriteLine("Eingabe der Höhenkennzahl des Ausgangssystems")
Console.WriteLine(" (Normalhöhe=N oder n, Ellipsoidische Höhe<>N):")
dString = Console.ReadLine()
If Not Char.TryParse(dString, hk) Then hk = Char.Parse("E")
If hk.CompareTo(Char.Parse("n")) = 0 Then
    hk = Char.Parse("N")
End If
If hk.CompareTo(Char.Parse("N")) <> 0 Then
    hk = Char.Parse("E")
End If
Console.WriteLine("Ausgangsdaten:")
Console.WriteLine(" x           : {0:f3}m", x)
Console.WriteLine(" y           : {0:f3}m", y)
Console.WriteLine(" z           : {0:f3}m", z)
Console.WriteLine(" l1          : {0}", l1)
Console.WriteLine(" l2          : {0}", l2)
Console.WriteLine(" hk          : {0}", hk)
' Umrechnung mit trans
Console.WriteLine(" Akt. Verzeichnis: {0}", Directory.GetCurrentDirectory())
kat(0) = Char.Parse(".") : kat(1) = Char.Parse(".") : kat(2) = Char.Parse("\")
kat(3) = Char.Parse("P") : kat(4) = Char.Parse("a") : kat(5) = Char.Parse("r")
kat(6) = Char.Parse("a") : kat(7) = Char.Parse("m") : kat(8) = Char.Parse("e")
kat(9) = Char.Parse("t") : kat(10) = Char.Parse("e") : kat(11) = Char.Parse("r")
verz(0) = Char.Parse(".") : verz(1) = Char.Parse(".") : verz(2) = Char.Parse("\")
verz(3) = Char.Parse("P") : verz(4) = Char.Parse("a") : verz(5) = Char.Parse("r")
verz(6) = Char.Parse("a") : verz(7) = Char.Parse("m") : verz(8) = Char.Parse("e")
verz(9) = Char.Parse("t") : verz(10) = Char.Parse("e") : verz(11) = Char.Parse("r")
Console.Write(" kat           : ")
For icount = 0 To 99
    If AscW(kat(icount)) = 0 Then
        Exit For
    End If
    Console.Write("{0}", kat(icount))
Next
Console.WriteLine("")
Console.Write(" verz           : ")
For icount = 0 To 99
    If AscW(verz(icount)) = 0 Then
        Exit For
    End If
    Console.Write("{0}", verz(icount))
Next
Console.WriteLine("")
a_hk(0) = Char.Parse(hk) : a_hk(1) = Char.Parse(vbNull)
iret = trans_dll(x, y, z, l1, l2, kat, a_hk, xout, l_koord, epsg, ellip, abb, z_koord, ellipa, abba)
Console.WriteLine("")
Console.WriteLine("Umrechnung mit trans_dll:")
Console.WriteLine(" Return-Code von trans_dll : {0}", iret)
If iret = 0 Or iret > 1000 Then
    Console.WriteLine(" x           : {0,5:f3}m", xout(0))
    Console.WriteLine(" y           : {0,5:f3}m", xout(1))
    Console.WriteLine(" z           : {0,5:f3}m", xout(2))
    dString = ""
    For icount = 0 To 13
        If AscW(abb(icount)) = 0 Then Exit For
        dString = dString & abb(icount).ToString()
    Next
    If Not dString.Equals("Geozentrisch") Then _
        Console.WriteLine(" nn           : {0,5:f3}m", xout(3))
    Console.WriteLine(" Ausgangskoordinatensystem : ")
    For icount = 0 To 99

```

```

    If AscW(l_koord(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", l_koord(icontains))
Next
Console.WriteLine("")
Console.WriteLine("Abbildungsart Ausg.      : ")
For icount = 0 To 13
    If AscW(abb(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", abb(icontains))
Next
Console.WriteLine("")
Console.WriteLine("Ellipsoid Ausg.      : ")
For icount = 0 To 10
    If AscW(ellip(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", ellip(icontains))
Next
Console.WriteLine("")
If epsg(0) <> -1 Then
    Console.WriteLine("EPSG-Code Ausg.      : {0}", epsg(0))
Else
    Console.WriteLine("EPSG-Code Ausg.      : unbekannt")
End If
Console.WriteLine("Zielkoordinatensystem : ")
For icount = 0 To 99
    If AscW(z_koord(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", z_koord(icontains))
Next
Console.WriteLine("")
Console.WriteLine("Abbildungsart Ziel      : ")
For icount = 0 To 13
    If AscW(abba(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", abba(icontains))
Next
Console.WriteLine("")
Console.WriteLine("Ellipsoid Ziel      : ")
For icount = 0 To 10
    If AscW(ellipa(icontains)) = 0 Then Exit For
    Console.WriteLine("{0}", ellipa(icontains))
Next
Console.WriteLine("")
If epsg(1) <> -1 Then
    Console.WriteLine("EPSG-Code Ziel      : {0}", epsg(1))
Else
    Console.WriteLine("EPSG-Code Ziel      : unbekannt")
End If
If iret > 1000 Then
    name(0) = Char.Parse("h") : name(1) = Char.Parse("o") : name(2) = Char.Parse("e")
    name(3) = Char.Parse("h") : name(4) = Char.Parse("e") : name(5) = Char.Parse(".")
    name(6) = Char.Parse("i") : name(7) = Char.Parse("n") : name(8) = Char.Parse("i")
    transfm_dll(iret, name, melde)
    Console.WriteLine("Hinweis {0}: ", iret)
    For icount = 0 To 999
        If AscW(melde(icontains)) = 0 Then Exit For
        Console.WriteLine("{0}", melde(icontains))
    Next
    Console.WriteLine("")
End If
Else
    If (iret > 0 And iret < 3) Or (iret > 599 And iret < 800) Then
        name(0) = Char.Parse("D") : name(1) = Char.Parse("E") : name(2) = Char.Parse("F")
        name(3) = Char.Parse("A") : name(4) = Char.Parse("U") : name(5) = Char.Parse("L")
        name(6) = Char.Parse("T")
        transfm_dll(iret, name, melde)
        Console.WriteLine("Fehler {0}: ", iret)
        For icount = 0 To 999
            If AscW(melde(icontains)) = 0 Then Exit For
            Console.WriteLine("{0}", melde(icontains))
        Next
        Console.WriteLine("")
    End If
    If iret > 2 And iret < 600 Then
        name(0) = Char.Parse("P") : name(1) = Char.Parse("a") : name(2) = Char.Parse("r")
        name(3) = Char.Parse("a") : name(4) = Char.Parse("m") : name(5) = Char.Parse("e")
        name(6) = Char.Parse("t") : name(7) = Char.Parse("e") : name(8) = Char.Parse("r")
    End If
End If

```

```

name(9) = Char.Parse("d") : name(10) = Char.Parse("a") : name(11) = Char.Parse("t")
name(12) = Char.Parse("e") : name(13) = Char.Parse("i")
transfm_dll(iret, name, melde)
Console.Write(" Fehler {0}: ", ired)
For icount = 0 To 999
    If AscW(melde(icount)) = 0 Then Exit For
    Console.Write("{0}", melde(icount))
Next
Console.WriteLine("")
End If
End If
' schrittweise Bearbeitung
Console.WriteLine("")
Console.WriteLine("schrittweise Bearbeitung:")
Console.WriteLine("translD_dll:")
ired = translD_dll(kat, dart, anz, nr, epsga, datei)
Console.WriteLine(" Return-Code von translD_dll: {0}", ired)
If ired = 0 Then
    If dart = 0 Then
        Console.WriteLine(" Datenart                : {0} (unverschlüsselt, Suffix tra)", dart)
    Else
        Console.WriteLine(" Datenart                : {0} (verschlüsselt, Suffix tr0)", dart)
    End If
    Console.WriteLine(" Anzahl der Parametersätze : {0}", anz)
    Console.WriteLine("")
    Console.WriteLine("Lagestati  EPSG-Code  Dateiname ohne Suffix")
    For icount = 0 To anz - 1
        If epsga(icount) <> -1 Then
            Console.Write("   {0,3}          {1,5}          ", nr(icount), epsga(icount))
        Else
            Console.Write("   {0,3}          unbekannt          ", nr(icount))
        End If
        For j = 0 To 99
            If AscW(datei(icount, j)) = 0 Then Exit For
            Console.Write("{0}", datei(icount, j))
        Next
        Console.WriteLine("")
    Next
Else
    For icount = 0 To 99
        If AscW(kat(icount)) = 0 Then Exit For
        name(icount) = kat(icount)
    Next
    name(icount) = Char.Parse("\") : name(icount + 1) = Char.Parse("D")
    name(icount + 2) = Char.Parse("E") : name(icount + 3) = Char.Parse("F")
    name(icount + 4) = Char.Parse("A") : name(icount + 5) = Char.Parse("U")
    name(icount + 6) = Char.Parse("L") : name(icount + 7) = Char.Parse("T")
    transfm_dll(ired, name, melde)
    Console.Write(" Fehler {0}: ", ired)
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.Write("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
' Suchen der zugehörigen Parameterdatei
ired = -1
For icount = 0 To anz - 1
    If l1 = nr(icount) Then
        ired = icount
        epsg(0) = epsga(icount)
        Exit For
    End If
Next
If ired > -1 Then
    fehler = 0
    For icount = 0 To 99
        name(icount) = datei(ired, icount)
        If AscW(datei(ired, icount)) = 0 Then Exit For
    Next
    ired = transla_dll(kat, name, dart, l1, l_koord, ellip, abb, paraa)
    Console.WriteLine("")
    Console.WriteLine("transla_dll:")

```

```

Console.WriteLine(" Return-Code von transla_dll: {0}", ired)
If ired = 0 Then
    Console.WriteLine(" Ausgangskoordinatensystem : ")
    For icount = 0 To 99
        If AscW(l_koord(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", l_koord(icount))
    Next
    Console.WriteLine("")
    Console.WriteLine(" Abbildungsart Ausg. : ")
    For icount = 0 To 13
        If AscW(abb(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", abb(icount))
    Next
    Console.WriteLine("")
    Console.WriteLine(" Ellipsoid Ausg. : ")
    For icount = 0 To 13
        If AscW(ellip(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", ellip(icount))
    Next
    Console.WriteLine("")
    Console.WriteLine(" EPSG-Code Ausg. : {0}", epsg(0))
    Console.WriteLine(" Parameter Ausg. : {0,14:f6} {1,14:f6} {2,14:f6}", _
        paraa(0), paraa(1), paraa(2))
    For icount = 3 To 35 Step 3
        Console.WriteLine(" {0,14:f6} {1,14:f6} {2,14:f6}", _
            paraa(icount), paraa(icount + 1), paraa(icount + 2))
    Next
Else
    fehler = 1
    For icount = 0 To 99
        name(icount) = datei(ired, icount)
        If AscW(datei(ired, icount)) = 0 Then Exit For
    Next
    name(icount) = Char.Parse(".") : name(icount + 1) = Char.Parse("t")
    name(icount + 2) = Char.Parse("r")
    If dart = 0 Then
        name(icount + 3) = Char.Parse("a")
    Else
        name(icount + 3) = Char.Parse("0")
    End If
    name(icount + 4) = Char.Parse(vbNull)
    transfm_dll(ired, name, melde)
    Console.WriteLine(String.Format("Fehler {0}: ", ired))
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
Else
    fehler = 1
    name(0) = Char.Parse(vbNull)
    transfm_dll(93, name, melde)
    Console.WriteLine(String.Format("Fehler 93: "))
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
' Suchen der zugehoerigen Parameterdatei
ired = -1
For icount = 0 To anz - 1
    If l2 = nr(icount) Then
        ired = icount
        epsg(1) = epsga(icount)
        Exit For
    End If
Next
If ired > -1 Then
    fehler = 0
    For icount = 0 To 99
        name(icount) = datei(ired, icount)

```

```

    If AscW(datei(iret, icount)) = 0 Then Exit For
Next
iret = translz_dll(kat, name, dart, 12, z_koord, ellipa, abba, streifen, paraz)
Console.WriteLine("")
Console.WriteLine("translz_dll:")
Console.WriteLine(" Return-Code von translz_dll: {0}", iret)
If iret = 0 Then
    Console.Write(" Zielkoordinatensystem      : ")
    For icount = 0 To 99
        If AscW(z_koord(icount)) = 0 Then Exit For
        Console.Write("{0}", z_koord(icount))
    Next
    Console.WriteLine("")
    Console.Write(" Abbildungsart Ziel          : ")
    For icount = 0 To 13
        If AscW(abba(icount)) = 0 Then Exit For
        Console.Write("{0}", abba(icount))
    Next
    Console.WriteLine("")
    Console.Write(" Ellipsoid Ziel              : ")
    For icount = 0 To 13
        If AscW(ellipa(icount)) = 0 Then Exit For
        Console.Write("{0}", ellipa(icount))
    Next
    Console.WriteLine("")
    Console.WriteLine(" EPSG-Code Ziel              : {0}", epsg(1))
    Console.WriteLine(" Streifen Ziel              : {0}", streifen)
    Console.WriteLine(" Parameter Ziel             : {0,14:f6} {1,14:f6} {2,14:f6}", _
        paraz(0), paraz(1), paraz(2))
    For icount = 3 To 35 Step 3
        Console.WriteLine(" {0,14:f6} {1,14:f6} {2,14:f6}", _
            paraz(icount), paraz(icount + 1), paraz(icount + 2))
    Next
Else
    fehler = 1
    For icount = 0 To 99
        name(icount) = datei(iret, icount)
        If AscW(datei(iret, icount)) = 0 Then Exit For
    Next
    name(icount) = Char.Parse(".") : name(icount + 1) = Char.Parse("t")
    name(icount + 2) = Char.Parse("r")
    If dart = 0 Then
        name(icount + 3) = Char.Parse("a")
    Else
        name(icount + 3) = Char.Parse("0")
    End If
    name(icount + 4) = Char.Parse(vbNull)
    transfm_dll(iret, name, melde)
    Console.WriteLine(String.Format("Fehler {0}: ", iret))
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.Write("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
Else
    fehler = 1
    name(0) = Char.Parse(vbNull)
    transfm_dll(93, name, melde)
    Console.WriteLine(String.Format("Fehler 93: "))
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.Write("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
Console.WriteLine("")
If fehler = 0 Then
    Console.WriteLine("")
    Console.WriteLine("transr_dll:")
    xin(0) = x
    xin(1) = y
    xin(2) = z

```

```

dString = "" : dstring1 = ""
For icount = 0 To 13
    If AscW(abb(icount)) = 0 Then Exit For
    dString = dString & abb(icount).ToString()
Next
For icount = 0 To 13
    If AscW(abba(icount)) = 0 Then Exit For
    dstring1 = dstring1 & abba(icount).ToString()
Next
If (Not (a_hk(0) = Char.Parse("N"))) Or dString.Equals("Geozentrisch")) And _
    dstring1.Equals("Geozentrisch") Then
    iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, 0, 0, verz)
End If
If (Not (a_hk(0) = Char.Parse("N"))) Or dString.Equals("Geozentrisch")) And Not _
    dstring1.Equals("Geozentrisch") Then
    iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, 0, -1, verz)
End If
If (a_hk(0) = Char.Parse("N") And Not dString.Equals("Geozentrisch")) And _
    dstring1.Equals("Geozentrisch") Then
    iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, -1, 0, verz)
End If
If (a_hk(0) = Char.Parse("N") And Not dString.Equals("Geozentrisch")) And Not _
    dstring1.Equals("Geozentrisch") Then
    iret = transr_dll(xin, abb, abba, xout, ellip, paraa, ellipa, paraz, streifen, -1, -1, verz)
End If
Console.WriteLine(" Return-Code von transr_dll: {0}", iret)
If iret = 0 Or iret > 1000 Then
    Console.WriteLine(" x           : {0,5:f3}m", xout(0))
    Console.WriteLine(" y           : {0,5:f3}m", xout(1))
    Console.WriteLine(" z           : {0,5:f3}m", xout(2))
    If Not dstring1.Equals("Geozentrisch") Then _
        Console.WriteLine(" nn          : {0,5:f3}m", xout(3))
    If iret > 1000 Then
        For icount = 0 To 99
            name(icount) = verz(icount)
            If AscW(verz(icount)) = 0 Then Exit For
        Next
        name(icount) = Char.Parse("\") : name(icount + 1) = Char.Parse("h")
        name(icount + 2) = Char.Parse("o") : name(icount + 3) = Char.Parse("e")
        name(icount + 4) = Char.Parse("h") : name(icount + 5) = Char.Parse("e")
        name(icount + 6) = Char.Parse(".") : name(icount + 7) = Char.Parse("i")
        name(icount + 8) = Char.Parse("n") : name(icount + 9) = Char.Parse("i")
        name(icount + 10) = Char.Parse(vbNull)
        transfm_dll(iret, name, melde)
        Console.WriteLine(String.Format("Hinweis {0}: ", iret))
        For icount = 0 To 999
            If AscW(melde(icount)) = 0 Then Exit For
            Console.WriteLine("{0}", melde(icount))
        Next
        Console.WriteLine("")
    End If
Else
    name(0) = Char.Parse(vbNull)
    transfm_dll(iret, name, melde)
    Console.WriteLine(String.Format("Fehler {0}: ", iret))
    For icount = 0 To 999
        If AscW(melde(icount)) = 0 Then Exit For
        Console.WriteLine("{0}", melde(icount))
    Next
    Console.WriteLine("")
End If
End If
End Sub
End Module

```

Zugehörige Bildschirmanzeige

```

C:\Visual Studio 2010\Projects\testdllvb\testdllvb\bin\Release>testdllvb
Programm T E S T D L L V B
(c) Dr. Bergmann, Version 1.0, 2015
Bearbeitung: 27.05.2015 15:17:10

```

```
Eingabe der Parameter: x, y, z, l1, l2, hk
  x, y und z sind die Koordinaten des Ausgangssystems im Lagestatus l1
  hk ist gleich die Höhenkennzahl im Ausgangssystem mit n oder N für
  Normalhöhen, l2 ist der Lagestatus im Zielsystem, dessen Koordinaten über
  die Programme der Bibliothek transdll.dll ermittelt werden
Eingabe der x-Koordinate in m:
100
Eingabe der y-Koordinate in m:
100
Eingabe der z-Koordinate in m:
30,2
Eingabe der Lagestatusangabe des Ausgangssystems:
600
Eingabe der Lagestatusangabe des Zielsystems:
493
Eingabe der Höhenkennzahl des Ausgangssystems
(Normalhöhe=N oder n, Ellipsoidische Höhe<>N):
k
Ausgangsdaten:
  x          : 100,000m
  y          : 100,000m
  z          : 30,200m
  l1         : 600
  l2         : 493
  hk         : E
Akt. Verzeichnis: C:\Visual Studio 2010\Projects\testdllvb\testdllvb\bin\Release
kat          : ..\Parameter
verz         : ..\Parameter

Umrechnung mit trans_dll:
Return-Code von trans_dll : 0
  x          : 5808434,939m
  y          : 33406618,788m
  z          : 69,429m
  nn         : 30,136m
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
Abbildungsart Ausg.       : Soldner
Ellipsoid Ausg.           : Bessel
EPSG-Code Ausg.           : 36000
Zielkoordinatensystem     : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
Abbildungsart Ziel        : Gauss-Krueger
Ellipsoid Ziel            : GRS80
EPSG-Code Ziel            : 5650

schrittweise Bearbeitung:
translD_dll:
Return-Code von translD_dll: 0
Datenart          : 0 (unverschlüsselt, Suffix tra)
Anzahl der Parametersätze : 43

Lagestati  EPSG-Code  Dateiname ohne Suffix
  100      unbekannt  100
  102      31466      102
  103      31467      103
  104      31468      104
  105      31469      105
  111      5678       111
  112      5679       112
  130      unbekannt  130
  133      unbekannt  133
  134      unbekannt  134
  135      unbekannt  135
  140      unbekannt  140
  142      5664       142
  143      5665       143
  150      unbekannt  150
```

153	5673	153
154	5674	154
155	5675	155
350	5828	350
384	4328	384
389	4346	389
400	unbekannt	400
432	23032	432
433	23033	433
489	unbekannt	489
491	5649	491
492	4647	492
493	5650	493
494	unbekannt	494
495	unbekannt	495
500	3068	500
589	unbekannt	589
600	36000	600
610	unbekannt	x610
620	unbekannt	x620
630	unbekannt	x630
640	unbekannt	x640
650	unbekannt	650
660	unbekannt	x660
815	4178	815
850	4314	850
884	4326	884
889	4258	889

transla_dll:

Return-Code von transla_dll: 0
Ausgangskoordinatensystem : Soldner 18, Mueggelberg
Abbildungsart Ausg. : Soldner
Ellipsoid Ausg. : Bessel
EPSG-Code Ausg. : 36000
Parameter Ausg. :

	0,914878	13,627204	0,000000
	0,000000	100000,000000	1,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	1,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	6377397,155080	6356078,962900	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000

translz_dll:

Return-Code von translz_dll: 0
Zielkoordinatensystem : UTM-Koordinaten ETRS89 DREF91 (Zone 33)
Abbildungsart Ziel : Gauss-Krueger
Ellipsoid Ziel : GRS80
EPSG-Code Ziel : 5650
Streifen Ziel : 33
Parameter Ziel :

	0,000000	15,000000	0,000000
	0,000000	0,000000	0,999600
	0,000000	0,000000	0,000000
	500000,000000	0,000000	0,000000
	0,000000	1,000000	900097,306000
	3784880,652000	5037371,552000	0,000006
	0,000003	0,000003	1,000000
	900067,864000	3784246,245000	5036920,892000
	6378137,000000	6356752,314140	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000
	0,000000	0,000000	0,000000


```

transr_dll:
Return-Code von transr_dll: 0
x                : 5808434,939m
y                : 33406618,788m
z                : 69,429m
nn               : 30,136m

C:\Visual Studio 2010\Projects\testdllvb\testdllvb\bin\Release>

```

7. Ausgabedaten

Alle Unterprogramme der Bibliothek weisen keine eigenen Bildschirm- oder Dateiausgaben auf. Die Kommunikation mit dem aufrufenden Programm erfolgt über den Returncode der eingesetzten Programme (siehe hierzu 8).

8. Meldungen der Programme

Die Zahlenwerte der Returncodes der Unterprogramme der Bibliothek geben die Unterscheidung nach fehlerfreien oder fehlerbehafteten Durchlauf wieder. Im Fehlerfall wird außerdem im Returncode nach der Fehlerursache differenziert. Nachfolgend sind die Returncodes der vier Unterprogramme mit ihren Bedeutungen angegeben.

8.1 Meldungen vom Unterprogramm transID_dll

Die nachfolgenden Returncodes sind identisch mit denen vom Unterprogramm trans_dll. Mit dem Unterprogramm transfm_dll können mit der Angabe des Returncodes als Nummer und dem Namen der Datei, hier DEFAULT eventuell mit dem Verzeichnisnamen, Fehlermeldungen erzeugt werden.

Returncode	Bedeutung	Ursache bzw. Abhilfe
0	Fehlerfreier Durchlauf	
1	Definitionsdatei <Dateiname> kann zum Lesen nicht geöffnet werden	Bereinigen bzw. Bereitstellen der Datei <Dateiname> (<Katalog>\DEFAULT)
2	Lesen der Anzahl der Koordinatensysteme in der Definitionsdatei <Dateiname>	
100	Lesefehler in <Dateiname>	
599+n	Lesen der Lagestatusangabe in der Definitionsdatei <Dateiname> der n. Angabe	
699+n	Lesen der Dateiangabe in der Definitionsdatei <Dateiname> der n. Angabe	

Tabelle 8.1: Meldungen des Unterprogramms TransID_dll

8.2 Meldungen von den Unterprogrammen transla_dll und translz_dll

Die nachfolgenden Returncodes sind identisch mit denen vom Unterprogramm trans_dll. Da beim Auslesen der Parameterdateien für die Ausgangs- und die Zielsysteme nur bei den Angaben für den Ellipsoidübergang unterschiedliche Werte gelesen werden, sind die ersten Returncodes für beide Unterprogramme identisch. Mit dem Unterprogramm transfm_dll können mit der Angabe des Returncodes als Nummer und dem Namen der Datei, hier also der Name der zugehörigen Parameterdatei eventuell mit dem Verzeichnisnamen und dem Suffix, Fehlermeldungen erzeugt werden.

Return-code	Bedeutung	Ursache bzw. Abhilfe
0	Fehlerfreier Durchlauf	
7	Parameterdatei <Dateiname> existiert nicht	Bereinigen bzw. Bereitstellen der Parameterdatei für das Ausgangs- oder Zielsystem (Meldungen von transla_dll oder translz_dll)
8	Lesefehler in <Dateiname> beim Lesen der Lagestatusangabe	
9	Lesefehler in <Dateiname> beim Vergleichen der Lagestatusangaben	
10	Lesefehler in <Dateiname> beim Lesen der Bezeichnung des Systems	
11	Lesefehler in <Dateiname> beim Lesen der Bezeichnung des Ellipsoids	
12	Lesefehler in <Dateiname> beim Lesen der Abbildungsart	
13	Lesefehler in <Dateiname> beim Lesen der Breite des Nullpunkts	
14	Lesefehler in <Dateiname> beim Lesen der Länge des Nullpunkts	
15	Lesefehler in <Dateiname> beim Lesen der Additionskonstante in Y	
16	Lesefehler in <Dateiname> beim Lesen der Additionskonstante in X	
17	Lesefehler in <Dateiname> beim Lesen der dekadischen Ergänzung	
18	Lesefehler in <Dateiname> beim Lesen des Maßstabs des Meridians	
19	Lesefehler in <Dateiname> beim Lesen der Verschiebung in Y	
20	Lesefehler in <Dateiname> beim Lesen der Verschiebung in X	
21	Lesefehler in <Dateiname> beim Lesen der Verdrehung	
22	Lesefehler in <Dateiname> beim Lesen der Verschiebung in Y	
23	Lesefehler in <Dateiname> beim Lesen der Verschiebung in X	
24	Lesefehler in <Dateiname> beim Lesen der Nullpunktskoordinate in Y	
25	Lesefehler in <Dateiname> beim Lesen der Nullpunktskoordinate in X	
26	Lesefehler in <Dateiname> beim Lesen des Maßstabs	
27	Lesefehler in <Dateiname> beim Lesen der großen Halbachse	
28	Lesefehler in <Dateiname> beim Lesen der kleinen Halbachse	
100	Lesefehler in <Dateiname>	

Return-code	Bedeutung	Ursache bzw. Abhilfe
29	Lesefehler in <Dateiname> beim Lesen der Translation in X	Bereinigen bzw. Bereitstellen der Parameter-datei für das Ausgangs-system (Meldungen von transla_dll)
30	Lesefehler in <Dateiname> beim Lesen der Translation in Y	
31	Lesefehler in <Dateiname> beim Lesen der Translation in Z	
32	Lesefehler in <Dateiname> beim Lesen der Rotation um X	
33	Lesefehler in <Dateiname> beim Lesen der Rotation um Y	
34	Lesefehler in <Dateiname> beim Lesen der Rotation um Z	
35	Lesefehler in <Dateiname> beim Lesen des Maßstabs	
36	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in X	
37	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Y	
38	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Z	
39	Lesefehler in <Dateiname> beim Lesen der Translation in X zum Zwischensystem	
40	Lesefehler in <Dateiname> beim Lesen der Translation in Y zum Zwischensystem	
41	Lesefehler in <Dateiname> beim Lesen der Translation in Z zum Zwischensystem	
42	Lesefehler in <Dateiname> beim Lesen der Rotation um X zum Zwischensystem	
43	Lesefehler in <Dateiname> beim Lesen der Rotation um Y zum Zwischensystem	
44	Lesefehler in <Dateiname> beim Lesen der Rotation um Z zum Zwischensystem	
45	Lesefehler in <Dateiname> beim Lesen des Maßstabs zum Zwischensystem	
46	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in X zum Zwischensystem	
47	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Y zum Zwischensystem	
48	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Z zum Zwischensystem	
112	Lesefehler in <Dateiname> beim Prüfen der Abbildungsart	
113	Lesefehler in <Dateiname> beim Prüfen der Breite des Nullpunkts	
114	Lesefehler in <Dateiname> beim Prüfen der Länge des Nullpunkts	
127	Lesefehler in <Dateiname> beim Prüfen der großen Halbachse	
128	Lesefehler in <Dateiname> beim Prüfen der kleinen Halbachse	
70	Lesefehler in <Dateiname> beim Lesen der Translation in X	Bereinigen bzw. Bereitstellen der Parameterdatei für das Zielsystem (Meldungen von translz_dll)
71	Lesefehler in <Dateiname> beim Lesen der Translation in Y	
72	Lesefehler in <Dateiname> beim Lesen der Translation in Z	
73	Lesefehler in <Dateiname> beim Lesen der Rotation um X	
74	Lesefehler in <Dateiname> beim Lesen der Rotation um Y	
75	Lesefehler in <Dateiname> beim Lesen der Rotation um Z	
76	Lesefehler in <Dateiname> beim Lesen des Maßstabs	

Returncode	Bedeutung	Ursache bzw. Abhilfe
77	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in X	Bereinigen bzw. Bereitstellen der Parameterdatei für das Zielsystem (Meldungen von translz_dll)
78	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Y	
79	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Z	
80	Lesefehler in <Dateiname> beim Lesen der Translation in X zum Zwischensystem	
81	Lesefehler in <Dateiname> beim Lesen der Translation in Y zum Zwischensystem	
82	Lesefehler in <Dateiname> beim Lesen der Translation in Z zum Zwischensystem	
83	Lesefehler in <Dateiname> beim Lesen der Rotation um X zum Zwischensystem	
84	Lesefehler in <Dateiname> beim Lesen der Rotation um Y zum Zwischensystem	
85	Lesefehler in <Dateiname> beim Lesen der Rotation um Z zum Zwischensystem	
86	Lesefehler in <Dateiname> beim Lesen des Maßstabs zum Zwischensystem	
87	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in X zum Zwischensystem	
88	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Y zum Zwischensystem	
89	Lesefehler in <Dateiname> beim Lesen der Nullpunkt-reduktion in Z zum Zwischensystem	

Tabelle 8.2: Meldungen der Unterprogramme Transla_dll und Translz_dll

8.3 Meldungen vom Unterprogramm transr_dll

Die nachfolgenden Returncodes sind identisch mit denen vom Unterprogramm trans_dll. Mit dem Unterprogramm transfm_dll können mit der Angabe des Returncodes als Nummer und dem Namen der Datei, hier also einer Leerangabe bzw. der Datei hoehe.ini eventuell mit Verzeichnis, Fehlermeldungen erzeugt werden.

Returncode	Bedeutung	Ursache bzw. Abhilfe
90	Keine Konvergenz bei der Berechnung der geographischen Koordinaten	Interne Fehler von trans
92	Keine Konvergenz bei der Berechnung der Undulation zur Reduzierung der Normalhöhe	
201	Keine Konvergenz bei der Bestimmung von e^2 bei der Umwandlung von Soldner- in geographische Koordinaten	Interne Fehler des Unterprogramms sdgeo (Umw. von Soldner- in geographische Koordinaten)
202	Keine Konvergenz bei der Berechnung des Meridianbogens bei der Umwandlung von Soldner- in geographische Koordinaten	

Return-code	Bedeutung	Ursache bzw. Abhilfe
203	Keine Konvergenz bei der Berechnung von $e(\text{phif})$ bei der Umwandlung von Soldner- in geographische Koordinaten	Interne Fehler des Unterprogramms sdgeo (Umwandlung von Soldner- in geographische Koordinaten)
204	Keine Konvergenz bei der Berechnung der Fußpunkt-breite bei der Umwandlung von Soldner- in geographische Koordinaten	
205	Keine Konvergenz bei der Berechnung der Gesamtbogenlänge bei der Umwandlung von Soldner- in geographische Koordinaten	
206	Keine Konvergenz bei der Berechnung der Breite bei der Umwandlung von Soldner- in geographische Koordinaten	
207	Keine Konvergenz bei der Berechnung der Länge bei der Umwandlung von Soldner- in geographische Koordinaten	
208	Keine Konvergenz bei der Berechnung der Länge bei der Umwandlung von Soldner- in geographische Koordinaten	
301	Keine Konvergenz bei der Bestimmung von e^{**2} bei der Umwandlung von konformen in geographische Koordinaten	Interne Fehler des Unterprogramms gkgeo (Umwandlung von Gauß-Krüger- in geographische Koordinaten)
302	Keine Konvergenz bei der Berechnung des Meridian-bogens bei der Umwandlung von konformen in geographische Koordinaten	
303	Keine Konvergenz bei der Berechnung von $e(\text{phi})$ bei der Umwandlung von konformen in geographische Koordinaten	
304	Keine Konvergenz bei der Berechnung der geo-graphischen Koordinaten bei der Umwandlung von konformen in geographische Koordinaten	
305	Keine Konvergenz bei der Bestimmung von ϕ bei der Umwandlung von konformen in geographische Koordinaten	
401	Keine Konvergenz bei der Bestimmung von e^{**2} bei der Umwandlung von geographischen in Soldner-Koordinaten	Interne Fehler des Unterprogramms geosd (Umwandlung von geographische in Soldner-Koordinaten)
402	Keine Konvergenz bei der Berechnung des Meridianbogens bei der Umwandlung von geographischen in Soldner-Koordinaten	
403	Keine Konvergenz bei der Fußpunktbreite bei der Umwandlung von geographischen in Soldner-Koordinaten	
404	Keine Konvergenz bei der y-Koordinatenberechnung bei der Umwandlung von geographischen in Soldner-Koordinaten	
405	Keine Konvergenz bei der Meridianbogenberechnung bei der Umwandlung von geographischen in Soldner-Koordinaten	
501	Keine Konvergenz bei der Bestimmung der Koordinaten bei der Umwandlung von geographischen in konforme Koordinaten	Interne Fehler des Unterprogramms geogk (Umwandlung von geo-graphische in Gauß-Krüger-Koordinaten)
502	Keine Konvergenz bei der Bestimmung von e^{**2} bei der Umwandlung von geographischen in konforme Koordinaten	
503	Keine Konvergenz bei der Berechnung des Meridianbogens bei der Umwandlung von geographischen in konforme Koordinaten	

Return-code	Bedeutung	Ursache bzw. Abhilfe
1027	Lesefehler in <Dateiname> beim Lesen der großen Halbachse von Bessel	Bereinigen bzw. Bereitstellen von hoehe.ini (Beim Auftreten dieser Fehlercodes ist die Umrechnung der Koordinaten erfolgreich durchgeführt worden, nur die der Normalhöhen ist fehlerhaft)
1028	Lesefehler in <Dateiname> beim Lesen der kleinen Halbachse von Bessel	
1029	Lesefehler in <Dateiname> beim Lesen der X-Verschiebung	
1030	Lesefehler in <Dateiname> beim Lesen der Y-Verschiebung	
1031	Lesefehler in <Dateiname> beim Lesen der Z-Verschiebung	
1032	Lesefehler in <Dateiname> beim Lesen des ersten Drehwinkels	
1033	Lesefehler in <Dateiname> beim Lesen des zweiten Drehwinkels	
1034	Lesefehler in <Dateiname> beim Lesen des dritten Drehwinkels	
1035	Lesefehler in <Dateiname> beim Lesen des Maßstabs	
1036	Lesefehler in <Dateiname> beim Lesen des X-Wertes im Schwerpunkt	
1037	Lesefehler in <Dateiname> beim Lesen des Y-Wertes im Schwerpunkt	
1038	Lesefehler in <Dateiname> beim Lesen des Z-Wertes im Schwerpunkt	
1039	Lesefehler in <Dateiname> beim Lesen der X-Verschiebung zum Zwischensystem	
1040	Lesefehler in <Dateiname> beim Lesen der Y-Verschiebung zum Zwischensystem	
1041	Lesefehler in <Dateiname> beim Lesen der Z-Verschiebung zum Zwischensystem	
1042	Lesefehler in <Dateiname> beim Lesen des ersten Drehwinkels zum Zwischensystem	
1043	Lesefehler in <Dateiname> beim Lesen des zweiten Drehwinkels zum Zwischensystem	
1044	Lesefehler in <Dateiname> beim Lesen des dritten Drehwinkels zum Zwischensystem	
1045	Lesefehler in <Dateiname> beim Lesen des Maßstabs zum Zwischensystem	
1046	Lesefehler in <Dateiname> beim Lesen des X-Wertes im Schwerpunkt zum Zwischensystem	
1047	Lesefehler in <Dateiname> beim Lesen des Y-Wertes im Schwerpunkt zum Zwischensystem	
1048	Lesefehler in <Dateiname> beim Lesen des Z-Wertes im Schwerpunkt zum Zwischensystem	
1049	Lesefehler in <Dateiname> beim Prüfen der großen Halbachse von Bessel	
1050	Lesefehler in <Dateiname> beim Prüfen der kleinen Halbachse von Bessel	
1090	Keine Konvergenz bei der Berechnung der geographischen Koordinaten	

Return-code	Bedeutung	Ursache bzw. Abhilfe
1100	Lesefehler in <Dateiname>	Bereinigen bzw. Bereitstellen von hoehe.ini
1101	Lesefehler in <Dateiname> beim Lesen der Länge des Fundamentpunktes Rauenberg	
1102	Lesefehler in <Dateiname> beim Lesen der Breite des Fundamentpunktes Rauenberg	
1103	Lesefehler in <Dateiname> beim Lesen der Lotabweichung im Fundamentpunkt Rauenberg	
1104	Lesefehler in <Dateiname> beim Lesen der Lotabweichung im Fundamentpunkt Rauenberg	
1105	Lesefehler in <Dateiname> beim Lesen der Höhenanomalie im Fundamentpunkt Rauenberg	
1106	Lesefehler in <Dateiname> beim Lesen der Bereichsangaben für die Länge der absoluten Höhenanomalien	
1107	Lesefehler in <Dateiname> beim Lesen der Bereichsangaben für die Breite der absoluten Höhenanomalien	
1108	Für diese Länge bzw. Breite liegt kein Geoidmodell vor	
1109	Lesefehler in <Dateiname> beim Lesen der absoluten Undulationen	
1110	Datei <Dateiname> existiert nicht oder kann zum Lesen nicht geöffnet werden	
1127	Lesefehler in <Dateiname> beim Lesen der großen Halbachse vom GRS80	
1128	Lesefehler in <Dateiname> beim Lesen der kleinen Halbachse vom GRS80	
1149	Lesefehler in <Dateiname> beim Prüfen der großen Halbachse vom GRS80	
1150	Lesefehler in <Dateiname> beim Prüfen der kleinen Halbachse vom GRS80	
1151	Lesefehler in <Dateiname> beim Prüfen der Länge des Fundamentpunktes Rauenberg	
1152	Lesefehler in <Dateiname> beim Prüfen der Breite des Fundamentpunktes Rauenberg	
1156	Lesefehler in <Dateiname> beim Prüfen der Bereichsangaben für die Länge der absoluten Höhenanomalien	
1157	Lesefehler in <Dateiname> beim Prüfen der Bereichsangaben für die Breite der absoluten Höhenanomalien	
1158	Lesefehler in <Dateiname> beim Prüfen der ersten absoluten Undulation	
1159	Lesefehler in <Dateiname> beim Prüfen der dritten absoluten Undulation	
1160	Lesefehler in <Dateiname> beim Prüfen der zweiten absoluten Undulation	
1161	Lesefehler in <Dateiname> beim Prüfen der vierten absoluten Undulation	

Tabelle 8.3: Meldungen des Unterprogramms Transr_dll

8.4 Meldungen vom Unterprogramm trans_dll

Da der Ablauf des Unterprogramms trans_dll auf den Funktionen der vorhergehenden Unterprogramme basiert, sind die Rückgabecodes vollkommen identisch und können wiederum über das Unterprogramm transfm_dll mit vollständigen Fehlermeldungen versehen werden. Der einzige zusätzliche Fehlerfall tritt mit dem Returncode 93 in diesem Unterprogramm auf, wenn zu einem angegebenen Lagestatus keine Parameterdatei existiert. Ein fehlerfreier Durchlauf endet auch hier wieder mit dem Rückgabewert 0. Bei Returncodes über 1000 ist die Koordinatenberechnung fehlerfrei durchführbar gewesen, nur bei der Umrechnung aus oder in eine Normalhöhe sind Probleme aufgetreten.

Return-code	Bedeutung	Ursache bzw. Abhilfe
93	Zur Lagestatusangabe existiert keine Parameterdatei	Falsche Lagestatusauswahl

9. Verfügbare Koordinatensysteme

Lage-status	Kordinatensystem	programminterne Bezeichnung der Abbildungsart	Übergabewerte
100	Gauß-Krüger-Koordinaten auf dem Besselellipsoid mit automatischer Streifenzuordnung	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
103	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im 3. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
104	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im 4. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
105	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im 5. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
130	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im System 40/83 mit automatischer Streifenzuordnung	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
133	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im System 40/83 im 3. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
134	Gauß-Krüger-Koordinaten auf dem Besselellipsoid im System 40/83 im 4. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z

Lage-status	Kordinatensystem	programminterne Bezeichnung der Abbildungsart	Übergabewerte
135	Gauß-Krüger-Koordinaten auf dem Besselipsoid im System 40/83 im 5. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
140	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 6 Grad-Streifen mit automatischer Streifen-zuordnung	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
142	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 6 Grad-Streifen im 2. Streifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
143	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 6 Grad-Streifen im 3. Streifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
150	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 3 Grad-Streifen mit automatischer Streifen-zuordnung	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
153	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 3 Grad-Streifen im 3. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
154	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 3 Grad-Streifen im 4. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
155	Gauß-Krüger-Koordinaten auf dem Krassowskiellipsoid im System 42/83 in 3 Grad-Streifen im 5. Meridianstreifen	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
384	Koordinaten im System ETRS 89 mit der Netzfestlegung EUREF 89 (WGS84-Ellipsoid)	Geozentrisch	dreidimensionale geozentrische metrische Koordinaten
389	Koordinaten im System ETRS 89 mit der Netzfestlegung ETRS 89 (GRS80-Ellipsoid)	Geozentrisch	dreidimensionale geozentrische metrische Koordinaten

Lage-status	Kordinatensystem	programminterne Bezeichnung der Abbildungsart	Übergabewerte
400	UTM-Koordinaten auf dem Hayford-Ellipsoid mit automatischer Zonenzuordnung	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
432	UTM-Koordinaten auf dem Hayford-Ellipsoid in der 32. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
433	UTM-Koordinaten auf dem Hayford-Ellipsoid in der 33. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
489	UTM-Koordinaten im ETRS 89 mit automatischer Zonenzuordnung	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
491	UTM-Koordinaten im ETRS 89 in der 31. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
492	UTM-Koordinaten im ETRS 89 in der 32. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
493	UTM-Koordinaten im ETRS 89 in der 33. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
494	UTM-Koordinaten im ETRS 89 in der 34. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
495	UTM-Koordinaten im ETRS 89 mit automatischer Zonenzuordnung ab 30. Zone	Gauss-Krueger	ebene metrische North- und Eastwerte y, x und die Höhe in z
500	Soldner-Berlin	Soldner	ebene metrische Soldnerkoordinaten y, x und die Höhe in z
600	Soldner im bestehenden Lagefestpunktfeld (18. Soldnersystem)	Soldner	ebene metrische Soldnerkoordinaten y, x und die Höhe in z
610	System S 18/1 (konforme Koordinaten auf dem Krasowski-Ellipsoid mit ebener Verdrehung und Maßstab)	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z

Lage-status	Kordinatensystem	programminterne Bezeichnung der Abbildungsart	Übergabewerte
620	System S 18/2 (konforme Koordinaten auf dem Krasowski-Ellipsoid mit ebener Verdrehung)	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
630	System S 18/3 (konforme Koordinaten auf dem Krasowski-Ellipsoid mit ebener Verdrehung)	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
640	konforme Koordinaten bezogen auf den Müggelberg (Reinickendorf/Pankow)	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
650	Soldner Götzer Berg (19. Soldnersystem)	Soldner	ebene metrische Soldnerkoordinaten y, x und die Höhe in z
660	konforme Koordinaten bezogen auf den Rathausturm in Berlin Mitte	Gauss-Krueger	ebene metrische Rechts- und Hochwerte y, x und die Höhe in z
850	Geographische Koordinaten auf dem Besselipsoid	Geographisch	Länge und Breite in Altgrad und der Höhe
884	Geographische Koordinaten zu dem System 384	Geographisch	Länge und Breite in Altgrad und der Höhe
889	Geographische Koordinaten zu dem System 389	Geographisch	Länge und Breite in Altgrad und der Höhe

Tabelle 9.1: Verfügbare Koordinatensysteme

Nachfolgend sind die programminternen Bezeichnungen für die Abbildungsart aufgeführt, welche auch nur in der absolut identischen Schreibweise genutzt werden kann.

Programminterne Bezeichnung der Abbildungsart	Bedeutung	Beispiel
Soldner	Ordinatentreue zweidimensionale ebene kartesische Koordinatensysteme mit einer Höhenangabe	Soldner-Berlin mit dem Lagestatus 500
Gauss-Krueger	Konforme zweidimensionale ebene kartesische Koordinatensysteme mit einer Höhenangabe	Gauß-Krüger Koordinaten im Lagestatus 100
Geographisch	Geographische Winkelangaben für die Länge und die Breite in Altgrad mit einer Höhenangabe	Geographische Koordinaten auf dem Besselipsoid mit dem Lagestatus 850
Geozentrisch	Dreidimensionale geozentrische kartesische Koordinatenangaben	Koordinaten im System ETRS 89 mit der Netzfestlegung ETRS 89 mit dem Lagestatus 389

Tabelle 9.2: Abbildungsarten

In der nachfolgenden Tabelle sind die Namen der genutzten Ellipsoide erläutert. Hierbei ist bei der Nutzung der Programmbibliothek nur darauf zu achten, dass der Name für das Besselipsoid in der nachfolgenden Schreibweise festgelegt ist und dass ein Ellipsoidname maximal 10 Buchstaben aufweisen darf.

Bezeichnung der Ellipsoide	Nutzung	Hinweis
Bessel	Deutschland	Festgeschriebene Schreibweise
Hayford	Europa	
Krassowski	Ehemaliger Ostblock	
WGS84	Weltweit	
GRS80	Weltweit	

Tabelle 9.3: Ellipsoidbezeichnungen

10. Genauigkeiten

Alle Abbildungen basieren auf exakten Formeln und geben dementsprechend bei einer Transformation in den Ergebnissen die Ausgangsgenauigkeiten wieder. Die Ellipsoidübergänge sind anhand von identischen Punkten ermittelt worden. Somit können bei einem Ellipsoidübergang in der Transformation die Ergebnisse nur mit einer Genauigkeit ermittelt werden, die durch die identischen Punkte vorgegeben ist.

Von älteren Koordinatensystemen wie dem 18. Soldnersystem (Soldnerkoordinaten im bestehenden Lagefestpunktfeld, Lagestatus 600) und dem System S 18/1 (Lagestatus 610) ist bekannt, dass lokale Sprünge bis in den Dezimeterbereich in den Netzen auftreten. Für die

Transformationsansätze geht man jedoch davon aus, dass global transformiert werden kann, so dass diese lokalen Ungenauigkeiten nicht berücksichtigt sind.

11. Beschränkungen

In der nachfolgenden Tabelle sind die Beschränkungen für den Programmablauf aufgeführt.

Beschränkung in	maximal zulässig
Anzahl der Parameterdateien	100
Länge des Namens des Verzeichnisses der Parameterdateien	99 Zeichen
Länge des Namens einer Parameterdatei	99 Zeichen
Länge der Bezeichnung für ein Koordinatensystem (freier Text)	99 Zeichen
Länge der Bezeichnung für eine Koordinatenart	13 Zeichen
Länge der Bezeichnung für das Ellipsoid	10 Zeichen

Tabelle 11.1: Beschränkungen

12. Literaturverzeichnis

- Heck, Bernhard Rechenverfahren und Auswertemodelle der Landesvermessung; Herbert Wichmann Verlag 1987
- Heiskanen, W. and Moritz, H. Physical Geodesy, W. H. Freeman and Company, San Francisco and London, 1967
- Klotz, Jürgen Eine analytische Lösung kanonischer Gleichungen der geodätischen Linie zur Transformation ellipsoidischer Flächenkoordinaten; Deutsche Geodätische Kommission, Reihe C, Nr. 385; 1991
- Klotz, Jürgen Eine analytische Lösung der Gauß-Krüger-Abbildung; Zeitschrift für Vermessungswesen 118 , Nr. 3, S. 106; 1993