

## Inhaltsverzeichnis

<b>VORWORT</b> .....	<b>2</b>
<b>BESCHREIBUNG</b> .....	<b>3</b>
<b>Definierende Aufrufe</b> .....	<b>3</b>
Ausgabedatei öffnen (ohne Hintergrundbild, 2 Farben).....	3
Ausgabedatei öffnen (ohne Hintergrundbild, 256 Graustufen).....	3
Ausgabedatei öffnen (ohne Hintergrundbild, 256 Farben).....	3
Ausgabedatei öffnen (mit Hintergrundbild) .....	4
Bildbeschneidung übermitteln .....	5
Nullpunktverschiebung.....	5
Stiftfarbe definieren .....	5
Füllfarbe definieren.....	5
Eigene Farbe definieren .....	6
Strichstärke definieren.....	6
Ausgabegröße definieren .....	6
Outline-Farbe definieren .....	6
Outline-Strichstärke definieren .....	6
Outline-Ende.....	7
Rotation.....	7
<b>Zeichnende Aufrufe</b> .....	<b>7</b>
Zeichne einen Punkt.....	7
Zeichne eine Linie vom zuletzt erreichten Punkt nach x, y .....	7
Zeichne ein Polygon, wahlweise weiß, schwarz oder nicht gefüllt. ....	7
Zeichne einen Bogen vom zuletzt erreichten Punkt nach x, y .....	7
Zeichne eine Bezier-Kurve vom zuletzt erreichten Punkt nach x, y .....	8
Zeichne eine Bezier-Kurve (1).....	8
Zeichne eine Bezier-Kurve (2).....	8
Zeichne eine Bezier-Kurve (3).....	8
Zeichne eine Kurve durch vorgegebene Punkte .....	8
Move .....	9
Zeichne einen Text.....	9
Zeichne eine Festkomma-Zahl .....	9
Zeichne eine Gleitkomma-Zahl einfacher Genauigkeit .....	9
Zeichne eine Gleitkomma-Zahl doppelter Genauigkeit .....	10
Zeichne einen Kreis oder Kreisbogen.....	10
Abschluß. Die Datei wird ausgegeben.....	10
<b>Andere Aufrufe</b> .....	<b>11</b>
Festkommazahl in Text konvertieren .....	11
Gleitkommazahl einfacher Genauigkeit in Text konvertieren .....	11
Gleitkommazahl doppelter Genauigkeit in Text konvertieren.....	11
(Quasi-) Zufallszahl ermitteln .....	11
<b>Aufruf der Modulbibliothek aus C/C++-Programmen:</b> .....	<b>12</b>

## Vorwort

Diese Bibliothek ist vornehmlich für Fortran-Programmierer gedacht. Sie wendet sich nicht eigentlich an C++ Programmierer, die mit ihrer Programmierumgebung viele der Funktionen bereits vorfinden. Dennoch ist beschrieben, wie sei aus C / C++ aufgerufen werden kann.

Voraussetzung für die Verwendung der Bibliothek ist, dass einfache Programmierkenntnisse in Fortran oder C / C++ vorhanden sind. C++ Programmierer mit tieferen Systemkenntnissen sind nicht die Zielgruppe, für diese Bibliothek interessant ist.

Die Bibliothek ist komplett in Fortran 2003 geschrieben und mit GNU-Fortran 2003 (Fortran 95) übersetzt. Sie setzt keine speziellen weiteren Bibliotheken voraus.

Die Erstellung von Grafikdateien als Strichzeichnungen aus Linien (Geraden, Flächen, Bögen, Kurven) und Texten incl. Zahlen machen Sinn als Monochrombilder oder Graustufen- und Farbgrafiken mit 256 Farben. Die Erstellung von (neuen) True-Color-Bildern (24 oder 32 Bit) macht hier wenig Sinn. Wohl aber macht es Sinn, True-Color-Bilder als Hintergrundbilder zu laden oder zum Beschriften einzulesen, z. B. bei Fotos.

Beides ist mit dieser Bibliothek möglich. Speziell für die Beschriftung von Fotos wurde eine Funktion eingeführt, die es erlaubt, Bilder mit einem zusätzlichen Weissraum zur Beschriftung unterhalb des Bildes zu versehen.

Die Verarbeitung mehrerer Bilder nacheinander ist möglich.

Wenn nicht anders angegeben, ist die im kostenlosen Download heruntergeladene Bibliothek nur im Testmodus zu verwenden. Im Testmodus ist die Bibliothek nur zeitlich befristet zu verwenden. Nach Ablauf der Testphase von normalerweise ca. 90 Tagen werden die danach erstellten Bilder mit einem Hintergrundmuster übermalt, das auf den Testmodus hinweist. Die Farben weiss (0) und schwarz (1 oder 255) können dann auch nicht mehr undefiniert werden. Es kann aber jederzeit eine neue Bibliothek heruntergeladen werden, die dann weitere ca. 90 Tage funktioniert. Ab 30 Tagen vor Ablauf wird darauf hingewiesen, dass eine neue Bibliothek heruntergeladen werden muss.

## **Beschreibung**

Die Beschreibung gliedert sich in definierende Aufrufe und zeichnende Aufrufe.

### **Definierende Aufrufe**

#### **Ausgabedatei öffnen (ohne Hintergrundbild, 2 Farben)**

call qqopen (x, y, dateiname, x-maßstab, y-maßstab)

Parameter:

x, y: real\*4, Bildgröße in Einheiten

dateiname: character\*255, Dateiname (mit/ohne Pfad)

x-maßstab, y-maßstab: real\*4, Umrechnungsfaktor für Einheiten

Sämtliche Koordinaten werden zur Umrechnung in Bildpunkte mit diesen Werten multipliziert.

Bei unterschiedlichen Maßstäben in x und y ist Rotation nicht möglich.

#### **Ausgabedatei öffnen (ohne Hintergrundbild, 256 Graustufen)**

call qqopengr (x, y, dateiname, x-maßstab, y-maßstab)

Parameter:

x, y: real\*4, Bildgröße in Einheiten

dateiname: character\*255, Dateiname (mit/ohne Pfad)

x-maßstab, y-maßstab: real\*4, Umrechnungsfaktor für Einheiten

Sämtliche Koordinaten werden zur Umrechnung in Bildpunkte mit diesen Werten multipliziert.

Bei unterschiedlichen Maßstäben in x und y ist Rotation nicht möglich.

#### **Ausgabedatei öffnen (ohne Hintergrundbild, 256 Farben)**

call qqopenco (x, y, dateiname, x-maßstab, y-maßstab) (256 Farben)

Parameter:

x, y: real\*4, Bildgröße in Einheiten

dateiname: character\*255, Dateiname (mit/ohne Pfad)

x-maßstab, y-maßstab: real\*4, Umrechnungsfaktor für Einheiten

Sämtliche Koordinaten werden zur Umrechnung in Bildpunkte mit diesen Werten multipliziert.

Bei unterschiedlichen Maßstäben in x und y ist Rotation nicht möglich.

## **Ausgabedatei öffnen (mit Hintergrundbild)**

oder qqopenin (namein, nameout, x-maßstab, y-maßstab)

Parameter:

namein, nameout: character\*255, Dateiname (mit/ohne Pfad)

Das Bild "namein" wird als Hintergrund bei der Erstellung des Bildes "nameout" verwendet. Es können Bitmaps mit 2 bzw 256 Farben oder true color (24 oder 32 Bits) benutzt werden.

x-maßstab, y-maßstab: real\*4, Umrechnungsfaktor für Einheiten

Sämtliche Koordinaten werden zur Umrechnung in Bildpunkte mit diesen Werten multipliziert.

Bei unterschiedlichen Maßstäben in x und y ist Rotation nicht möglich.

Die Bildgröße und die Farbanzahl werden automatisch vom eingelesenen Bild übernommen.

Bei 256 Farben wird die Farbtabelle des eingelesenen Bildes verwendet.

### Arbeitsweise bei Bildimport

#### Monochrom (1 Bit)

Das geöffnete Bild wird komplett eingelesen und als Bildhintergrund verwendet. Es kann beliebig übermalt werden.

#### 256 Farben

Das Bild wird als Hintergrund verwendet und kann beliebig übermalt werden. Die Farbtabelle wird aus dem eingelesenen Bild übernommen. Da die Farbnummern dabei nicht bekannt sind, ist die Verwendung von Farbdefinitionen mittels Farbnummer nicht mehr sinnvoll. Ein Überschreiben von verwendeten Farben führt dazu, dass sich die entsprechende Farbe im gesamten Bild entsprechend ändert.

Aus diesem Grund wurde die Farbdefinition ohne Farbnummer eingeführt. Dies geschieht mittels "call qqdcolor", wobei für die Stiftfarbe die Farbnummer mit -1 und für die Füllfarbe die Farbnummer mit -2 und für Outline die Farbnummer mit -3 angegeben wird. Die Farbnummer wird intern innerhalb der nicht verwendeten Farben zugewiesen. Sie wird nicht zurückübermittelt. Eine erneute Zuweisung derselben Farbe verwendet intern automatisch dieselbe Farbnummer.

Die Farbe 0 wird als transparent benutzt und kann zum Löschen innerhalb der Übermalung verwendet werden. Um weiss zu malen, muss die Farbe Weiss erst definiert werden. Dies geschieht durch "call qqdcolor (-1, 255, 255, 255)" für die Stiftfarbe, "call qqdcolor (-2, 255, 255, 255)" für die Füllfarbe. oder "call qqdcolor (-3, 255, 255, 255)" für die Farbe der Outlines.

#### Echtfarben (24 und 32 Bit)

Echtfarbenbilder werden nicht eingelesen. Stattdessen wird ein leeres Feld in der Größe des Bildes geöffnet. Dieses dient als Arbeitsfläche und ist wie eine darüber liegende Ebene zu verstehen. Als Farbtabelle wird die 256-Farben-Farbtabelle verwendet. Die Zuweisung der Farben erfolgt wie bei 256 Farben ohne eingelesenem Bild.

Erst bei "call qqclose" werden die Daten gerendert. Das Bild wird durch die Fläche überlagert, wobei die Farbnummer 0 wieder Transparenz bedeutet.

Die Arbeitsfläche kann mittels "call qqldown (y)" (layer down) nach unten verschoben werden, wodurch unten ein weisser Rand ausserhalb des eingelesenen Bildes entsteht. Dieser kann z.B. zum Beschriften von Fotos verwendet werden, wenn die Beschriftung nicht das vorhandene Bild übermalen soll.

## **Bildbeschneidung übermitteln**

call qqborder (switch, x1, y1, x2, y2)

Parameter:

switch: logical\*4, true=an, false=off

x1, y1: real\*4, linke untere Ecke

x1, y1: real\*4, rechte obere Ecke

Ab diesem Aufruf wird nur der im Ausschnitt liegende Teil gezeichnet.

Bei switch=off werden die Koordinaten nicht berücksichtigt,

Vertauschte Koordinaten werden gedreht,

identische Koordinaten führen zu switch=off.

Switch=off setzt die Koordinaten auf Bildgröße

## **Nullpunktverschiebung**

call qqpoorig (x, y)

Parameter:

x, y: real\*4, Verschiebewerte in Einheiten

Nullpunktverschiebung um x und y Einheiten. Dieser Aufruf eignet sich besonders für die Ausgabe wiederkehrender Zeichnungsaufrufe in Schleife.

## **Stiftfarbe definieren**

call qqcolor (farbnummer)

Parameter:

farbnummer: integer\*4, Strichfarbe. 0=weiss, 1=schwarz (bei 2 Farben) bzw. 0..255 bei Graustufen oder 256 Farben.

Mit dieser Farbe werden Linien, Texte und Zahlen gezeichnet.

Achtung: Bei eingelesenen Bildern mit 256 Farben wird die Farbtabelle des eingelesenen Bildes verwendet. Hier ist der Aufruf *Eigene Farbe verwenden*: call qqdcolor zu benutzen.

## **Füllfarbe definieren**

call qqfillc (farbnummer)

Parameter:

farbnummer: integer\*4, Polygon-Füllfarbe. 0=weiss, 1=schwarz (bei 2 Farben) bzw. 0..255 bei Graustufen oder 256 Farben, -1=keine Füllung.

Mit dieser Farbe werden Polygone gefüllt. Farbe 0 füllt Polygone weiß. Soll ein Polygon nicht gefüllt werden, ist -1 anzugeben.

Achtung: Bei eingelesenen Bildern mit 256 Farben wird die Farbtabelle des eingelesenen Bildes verwendet. Hier ist der Aufruf *Eigene Farbe verwenden*: call qqdcolor zu benutzen.

### **Eigene Farbe definieren**

call qqdcolor (farbnummer, rotwert, grünwert, blauwert)

Parameter:

farbnummer: integer\*4, 0=weiss, 1=schwarz (bei 2 Farben), -1 bis -3=beliebig bzw. 0..255 bei Graustufen oder 256 Farben,

rotwert, grünwert, blauwert: integer\*4, Wertebereich 0..255

Mit -1 wird die Stiftfarbe, mit -2 die Füllfarbe und mit -3 die Outline-Farbe auf einer bisher nicht verwendeten Farbnummer definiert. Dies ist bei Verwendung vorhandener Bilder wichtig, damit keine Farbveränderungen vorhandener Pixel stattfinden. Die Farbe wird automatisch gesetzt, also ab Aufruf zum Zeichnen verwendet.

Im Testmodus können weiss und schwarz nicht geändert werden.

### **Strichstärke definieren**

call qqthickn (strichstärke)

Parameter:

Strichstärke: real\*4, Strichstärke in Einheiten

Die Strichstärke wirkt sich nur auf Linien aus, nicht auf Text oder Zahlen.

### **Ausgabegröße definieren**

call qqdpi (xdpi, ydpi)

Parameter:

Pixels pro Zoll in x und y: real\*4

Der Wert hat keinen Einfluß auf die erstellte Bitmap.

Es ist nur ein Wert im Bitmap-Kopf.

### **Outline-Farbe definieren**

call qqoutcolor (farbnummer)

Parameter:

farbnummer: integer\*4, Outlinefarbe. 0=weiss, 1=schwarz (bei 2 Farben) bzw. 0..255 bei Graustufen oder 256 Farben.

Achtung: Bei eingelesenen Bildern mit 256 Farben wird die Farbtabelle des eingelesenen Bildes verwendet. Hier ist der Aufruf *Eigene Farbe verwenden*: call qqdcolor zu benutzen.

### **Outline-Strichstärke definieren**

call qqoutdef (strichstärke)

Parameter:

strichstärke: real\*4, Outline-Strichstärke in Prozent der normalen Strichstärke

Ab diesem Aufruf werden alle Linien als Outline mit der definierten Outline-Strichstärke und Outline-Farbe gezeichnet. Die Stiftfarbe und Linienstärke der Linie selbst ist unverändert.

Die Funktion wird beendet durch Wechsel der Stiftfarbe oder durch den folgenden Aufruf

## **Outline-Ende**

call qqoutend

Beendet das Zeichnen von Linien als Outline.

## **Rotation**

call qqrotate (x, y, w)

Parameter:

x, y: real\*4, Drehpunkt-Koordinaten in Einheiten

w: real\*4, Rotationswinkel in Grad

Die Rotation wirkt sich auf die ihm folgenden Aufrufe aus. Mehrere Rotationsaufrufe addieren sich nicht. Um die Rotation wieder auszuschalten, muss als Winkel 0. eingegeben werden.

Rotation und Verzerrung schließen sich aus.

## **Zeichnende Aufrufe**

### **Zeichne einen Punkt**

call qqpoint (x, y)

Parameter:

x, y: real\*4, Punktkoordinaten in Einheiten

### **Zeichne eine Linie vom zuletzt erreichten Punkt nach x, y**

call qqline (x, y)

Parameter:

x, y: real\*4, Zielpunkt-Koordinaten in Einheiten

### **Zeichne ein Polygon, wahlweise weiß, schwarz oder nicht gefüllt.**

call qqpoly (npolygon, polygon)

Parameter:

npolygon: integer\*4, Anzahl der Polygonpunkt-Paare (x, y)

polygon: real\*4, Punktkoordinaten in Einheiten

Die Polygonkoordinaten x, y werden jeweils paarweise in einem Array übergeben.

Polygonzüge müssen geschlossen sein, um gefüllt werden zu können.

### **Zeichne einen Bogen vom zuletzt erreichten Punkt nach x, y**

call qqbow (x, y)

Parameter:

x, y: real\*4, Zielpunkt-Koordinaten in Einheiten

Der Bogen wird in Verlängerung der zuletzt gezeichneten Linie zum Punkt x, y gezeichnet. War der letzte Befehl ein Point, wird der Bogen in Verlängerung der gedachten Linie der letzten beiden erreichten Punkte gezeichnet.

Mit diesem Aufruf kann kein Vollkreis gezeichnet werden. Für einen Vollkreis müssen zwei Halbkreise nacheinander gezeichnet werden.

### **Zeichne eine Bezier-Kurve vom zuletzt erreichten Punkt nach x, y**

call qqcurve (x, y, xnext, ynext)

Parameter:

x, y: real\*4, Zielpunkt-Koordinaten in Einheiten

xnext, ynext: real\*4, Koordinaten in Einheiten

Der Bogen wird in Verlängerung der zuletzt gezeichneten Linie (x-1, y-1 nach x, y) zum Punkt x, y gezeichnet. War der letzte Befehl ein Point, wird der Bogen in Verlängerung der gedachten Linie der letzten beiden erreichten Punkte gezeichnet.

xnext/ynext geben einen Punkt auf der gedachten geraden Verlängerung der Kurve an.

### **Zeichne eine Bezier-Kurve (1)**

call qqbezier (w1, r1, x1, y1, w2, r2, x2, y2)

Parameter: alle real\*4, Radien und Koordinaten in Einheiten, Winkel in Grad

w1, w2, r1, r2: Winkel und Radius bestimmen den Abstand der Kontrollpunkte von den Anfangs- und Endpunkten

x1, y1, x2, y2: Koordinaten des Anfangs- und Endpunktes

### **Zeichne eine Bezier-Kurve (2)**

call qqbezie2 (x1, y1, x1', y1', x2', y2', x2, y2)

Parameter: alle real\*4, Koordinaten in Einheiten.

x1, y1, x2, y2: Koordinaten des Anfangs- und Endpunktes

x1', y1', x2', y2': Koordinaten der Kontrollpunkte.

### **Zeichne eine Bezier-Kurve (3)**

call qqbezien (x, y, nxy)

Parameter:

x, y: Feld, real, Koordinaten der Stützpunkte der Kurve.

nxy: integer, Anzahl Koordinatenpaare in x/y

Bei Bezier-Kurven werden die angegebenen Stützpunkte nicht eingehalten. Ausgenommen sind nur der Start- und Endpunkt.

Dieser Aufruf ist mit Vorsicht zu verwenden. Bezier-Kurven mit mehr als vier Stützpunkten neigen dazu, irgendwann (in Abhängigkeit der Parameter) auszubrechen und undefinierte Verläufe zu nehmen. Derartige Ausreißer stellen also keinen Programmfehler dar. Demzufolge ist hier auch keine Korrektur des Programmes möglich. Eine Garantie für diesen Aufruf ist ausgeschlossen.

### **Zeichne eine Kurve durch vorgegebene Punkte**

call qqcotton (x, y, nxy)

Parameter:

x, y: Feld, real, Koordinaten der Stützpunkte der Kurve.

nxy: integer, Anzahl Koordinatenpaare in x/y

Im Gegensatz zu Bezier-Kurven werden hier alle angegebenen Stützpunkte eingehalten.

## Move

call qqmove (x, y)

Parameter:

x, y: real\*4, Zielpunkt-Koordinaten in Einheiten

wie qqpoint, ohne zu zeichnen. Dieser Aufruf macht nur Sinn, um die Koordinaten für einen Bogen zu definieren, der ohne eine gezeichnete Linie starten soll.

Beispiel: call qqmove (x1, y1), call qqpoint (x2, y2), call qqbow (x3, y3) ist wie call qqpoint (x1, y1), call qqline (x2, y2), call qqbow (x3, y3) ohne Linie von x1/y1 nach x2/y2.

qqmove darf anderweitig nicht verwendet werden. Die nächste Linie startet sonst nicht mit gerundetem Anfang.

## Zeichne einen Text

call qqfont1 (x, y, schriftgröße, text)

Parameter:

x, y: real\*4, Koordinaten des linken Anfangs der Schriftgrundlinie in Einheiten

schriftgröße: real\*4, Schriftgröße in Einheiten

text: character, der auszugebende Text

Der Text muss mit einer 0 abgeschlossen werden.

Beispiel: call qqfont1 (x, y, 30., text//char (0))

Die resultierende Schriftgröße (Schriftgröße mal Maßstab) sollte nicht unter 12 Punkten liegen. Beste Ergebnisse ab 30 Punkten.

## Zeichne eine Festkomma-Zahl

call qqifont1 (x, y, schriftgröße, zahl, länge)

Parameter:

x, y: real\*4, Koordinaten des linken Anfangs der Schriftgrundlinie in Einheiten

schriftgröße: real\*4, Schriftgröße in Einheiten

zahl: integer\*4, die auszugebende Zahl

länge: integer\*4, Anzahl der zu zeichnenden Ziffern

Die resultierende Schriftgröße (Schriftgröße mal Maßstab) sollte nicht unter 12 Punkten liegen. Beste Ergebnisse ab 30 Punkten.

Die gezeichnete Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird die Länge automatisch erhöht. Dieser Wert wird nicht zurückgegeben.

## Zeichne eine Gleitkomma-Zahl einfacher Genauigkeit

call qqrfont1 (x, y, schriftgröße, zahl, länge, nks)

Parameter:

x, y: real\*4, Koordinaten des linken Anfangs der Schriftgrundlinie in Einheiten

schriftgröße: real\*4, Schriftgröße in Einheiten

zahl: real\*4, die auszugebende Zahl

länge: integer\*4, Anzahl der zu zeichnenden Ziffern

nks: integer, Anzahl der Nachkommastellen

Die resultierende Schriftgröße (Schriftgröße mal Maßstab) sollte nicht unter 12 Punkten liegen. Beste Ergebnisse ab 30 Punkten.

Die gezeichnete Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird zunächst die Zahl der Nachkommastellen reduziert. Reicht das nicht aus, wird die Länge automatisch erhöht. Dieser Wert wird nicht zurückgegeben.

### **Zeichne eine Gleitkomma-Zahl doppelter Genauigkeit**

call qqdfont1 (x, y, schriftgröße, zahl, länge, nks)

Parameter:

x, y: real\*4, Koordinaten des linken Anfangs der Schriftgrundlinie in Einheiten

schriftgröße: real\*4, Schriftgröße in Einheiten

zahl: real\*8, die auszugebende Zahl

länge: integer\*4, Anzahl der zu zeichnenden Ziffern

nks: integer, Anzahl der Nachkommastellen

Die resultierende Schriftgröße (Schriftgröße mal Maßstab) sollte nicht unter 12 Punkten liegen. Beste Ergebnisse ab 30 Punkten.

Die gezeichnete Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird zunächst die Zahl der Nachkommastellen reduziert. Reicht das nicht aus, wird die Länge automatisch erhöht. Dieser Wert wird nicht zurückgegeben.

### **Zeichne einen Kreis oder Kreisbogen**

call qqcircle (w1, w2, x, y, r1, r2)

Parameter:

w1, w2: real\*4, Start- und Endwinkel in Grad, Vollkreis: (0., 360.)

x, y: real\*4, Mittelpunkt-Koordinaten in Einheiten

r1, r2: real\*4, Radien in Einheiten. r1=rechts, r2=oben.

### **Abschluß. Die Datei wird ausgegeben.**

call qqclose

Dieser Aufruf schließt auch alle definierenden Aufrufe ab (z.B. Outline) bzw. setzt die entsprechenden Werte zurück.

## Andere Aufrufe

### **Festkommazahl in Text konvertieren**

call qqiconv (text, nz, zahl, länge)

Parameter:

text: character, Bereich zur Aufnahme der in Zeichen umgewandelten Zahl

nz: integer, Anzahl Stellen, die zur Aufnahme der Zahl tatsächlich verwendet wurden

zahl: integer, die zu konvertierende Zahl

länge: integer\*4, Mindestanzahl Zeichen, die nach text übertragen werden sollen

Die konvertierte Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird die Länge automatisch erhöht. Dieser Wert wird in nz zurückgegeben.

Die Länge von text muss ausreichend sein, um die größtmögliche Zahl aufzunehmen. Dabei ist das Vorzeichen mit zu berücksichtigen.

Soll die Zahl in jedem Fall linksbündig ausgegeben werden, so ist länge mit 1 zu definieren.

### **Gleitkommazahl einfacher Genauigkeit in Text konvertieren**

call qqrconv (text, nz, zahl, länge, nks)

Parameter:

text: character, Bereich zur Aufnahme der in Zeichen umgewandelten Zahl

nz: integer, Anzahl Stellen, die zur Aufnahme der Zahl tatsächlich verwendet wurden

zahl: real\*4, die zu konvertierende Zahl

länge: integer\*4, Mindestanzahl Zeichen, die nach text übertragen werden sollen

nks: integer, Anzahl der Nachkommastellen

Die konvertierte Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird zunächst die Zahl der Nachkommastellen reduziert. Reicht das nicht aus, wird die Länge automatisch erhöht. Dieser Wert wird in nz zurückgegeben.

Die Länge von text muss ausreichend sein, um die größtmögliche Zahl aufzunehmen. Dabei sind der Dezimalpunkt und das Vorzeichen mit zu berücksichtigen.

### **Gleitkommazahl doppelter Genauigkeit in Text konvertieren**

call qqdconv (text, nz, zahl, länge, nks)

Parameter:

text: character, Bereich zur Aufnahme der in Zeichen umgewandelten Zahl

nz: integer, Anzahl Stellen, die zur Aufnahme der Zahl tatsächlich verwendet wurden

zahl: real\*8, die zu konvertierende Zahl

länge: integer\*4, Mindestanzahl Zeichen, die nach text übertragen werden sollen

nks: integer, Anzahl der Nachkommastellen

Die konvertierte Zahl wird bei Bedarf links mit Leerzeichen aufgefüllt. Wenn die Zahl nicht in die definierte Länge paßt, wird zunächst die Zahl der Nachkommastellen reduziert. Reicht das nicht aus, wird die Länge automatisch erhöht. Dieser Wert wird in nz zurückgegeben.

Die Länge von text muss ausreichend sein, um die größtmögliche Zahl aufzunehmen. Dabei sind der Dezimalpunkt und das Vorzeichen mit zu berücksichtigen.

### **(Quasi-) Zufallszahl ermitteln**

n=qqrandom (i)

Parameter:

i: integer\*4, 0: nächste, von der vorherigen abweichende Zahl

-1: Start mit quasi-Zufallszahl (immer dieselbe)

-2: Start mit echter Zufallszahl.

## Aufruf der Modulbibliothek aus C/C++-Programmen:

Um Fortran-Module aus C/C++ erfolgreich aufrufen zu können, muss die Fortran-Laufzeitbibliothek zunächst initialisiert werden. Dazu rufen Sie die Runtime-Bibliothek folgendermaßen auf:

```
void g95_runtime_start(int argc, char *argv[ ]);
```

Die Aufrufargumente sind `argc=0` und `argv=NULL`.

Bei Programmbeendigung muss die Laufzeit-Umgebung wieder geschlossen werden. Das geschieht durch

```
void g95_runtime_stop();
```

Um die Module aus C-Programmen aufzurufen, muss den Modulnamen ein Unterstrich angefügt werden.

### Beispielprogramm:

```
extern "C" void qqopen_(float*,float*,char*,float*,float*);
extern "C" void qqpoint_(float*,float*);
extern "C" void qqline_(float*,float*);
extern "C" void qqclose_();
extern "C" void g95_runtime_start(int argc, char *argv[]);
extern "C" void g95_runtime_stop();
int main()
{
float ix, iy, scalex, scaley;
char outfile[14], *openarg;
int i;

i=0;
openarg=0;
g95_runtime_start(i, &openarg);
ix=1201.; iy=1001.;
scalex=1.; scaley=1.;
qqopen_(&ix,&iy,".\\qqtest.bmp\0",&scalex,&scaley);
ix=10.;iy=10.;
qqpoint_(&ix,&iy);
ix=1190.; iy=990.;
qqline_(&ix,&iy);
qqclose_();
g95_runtime_stop();
return 0;
}
```

### Einzubindende Bibliotheken:

```
/pfad/qqmodlib.a
/F/G95/lib/gcc-lib/i686-pc-mingw32/4.0.3/libf95.a
/C/Dev-Cpp/lib/libws2_32.a
```

Alle Ausgaben nach stdout sind in einem Modul zusammengefasst, das Sie beliebig anpassen können. Damit haben Sie Einfluß darauf, auf welche Weise eventuelle Fehlermeldungen ausgegeben werden.

```
#include <stdio.h>
extern "C" void qqprint_ (char cformat[])
{
printf("%s\n",cformat);
return;
}
```

Falls erforderlich, finden Sie weitere Informationen in der G95-Beschreibung.