23.6.1.1 Druck-Projekt

Das Text-Druck-Projekt, ermöglicht es Ihnen, einfache Text-Dokumente auf Druckern direkt auszudrucken oder in eine PDF- oder PostScript-Datei zu drucken.

23.6.1.1.1 Vorbemerkungen - Hinweise

Um in Gambas ein Dokument zu drucken, müssen Sie:

- zuerst ein neues Printer-Objekt erzeugen,
- dann die Methode Printer.Configure() aufrufen, die einen Dialog zur Druckerkonfiguration angezeigt,
- danach den Druckvorgang starten, indem Sie die Methode Printer.Print() aufrufen und
- anschließend das Dokument Seite für Seite mit Paint zeichnen.

Der Druckvorgang wird in einer lokalen Ereignisschleife ausgeführt und die Print-Methode kehrt erst zurück, wenn der Druckvorgang abgeschlossen oder abgebrochen wurde. Während dieser lokalen Ereignisschleife werden nacheinander die folgenden Ereignisse ausgelöst:

- Printer.Begin(),
- Printer.Paginate(),
- Printer.Draw() und
- Printer.End().

Nur das Draw-Ereignis müssen Sie zwingend implementieren. Alle anderen Ereignisse sind optional.

Ereignis Begin()
Dieses Ereignis wird ausgelöst, wenn der Druckvorgang beginnt. Sie können dort die Eigenschaft Printer.Count definieren, wenn Sie genau wissen, wie viele Seiten Sie zu drucken haben. Sonst müssen Sie das Ereignis Paginate() implementieren, um Ihr Dokument zu layouten und die Seitenzahl zu berechnen. Standardmäßig wird nur eine Seite gedruckt. Andere wichtige Parameter wie das Papierformat und die Auflösung können ebenfalls in diesem Ereignis festgelegt werden.

Ereignis Paginieren()
Dieses Ereignis wird ausgelöst, damit Sie Ihr Dokument im Hintergrund mit einer Seitenzahl versehen können. Wenn Sie dieses Ereignis behandeln, dann wird der Event-Handler immer wieder aufgerufen, bis Sie die Eigenschaft Count explizit definieren. Die Berechnung der Anzahl der Seiten hängt davon ab, was Sie genau zeichnen. Wenn Sie Text zeichnen, so können Sie die Methoden Paint.TextSize() oder Paint.RichTextSize() zur Berechnung der Größe der Elemente wie die Texthöhe verwenden.

Beachten Sie: Auch das Ereignis Paginate() hat nichts mit dem eigentlichen Zeichnen zu tun. Das geschieht ausschließlich im Draw-Ereignis, dass für jede Seite einmal aufgerufen wird.

Ereignis Draw()
Dieses Ereignis wird für jede Seite, die gedruckt werden muss, einmal aufgerufen. Das komplette Zeichnen erfolgt mit der Klasse Paint. Die aktuelle Seitenzahl wird von der Eigenschaft Printer.Page zurückgegeben. Jedes Mal, wenn das Draw-Ereignis ausgelöst wird, wird eine neue Seite auf dem Drucker angelegt – außer der ersten, die wird automatisch erzeugt.

Ereignis End()
Dieses Ereignis wird ausgelöst, wenn der Druckvorgang beendet ist. Die Methode Paint.Begin() wird automatisch kurz vor dem Printer.Begin-Ereignis und die Methode Paint.End() kurz nach dem Printer.End-Ereignis aufgerufen. Deshalb müssen sie diese nicht explizit aufrufen.

Ränder
Das Zeichnen einer Seite erfolgt innerhalb eines Rechtecks, dessen Ursprung (0, 0) ist und die Größen Paint.Width sowie Paint.Height besitzt. Diese Größen gelten für den Fall, wenn die Eigenschaft Printer.FullPage auf True gesetzt wurde. Sie sollten immer mit dieser Einstellung drucken. Im folgenden Text-Druck-Projekt wird Ihnen gezeigt, wie Sie eine Textseite mit frei definierten Rändern für alle vier Seitenränder versehen und mit Hilfe des Papierformats das Rechteck festlegen, innerhalb dessen gedruckt wird.

Koordinaten
Anstelle der Zeichnungskoordinaten in Dots können Sie auch Koordinaten in Millimetern verwenden. Dafür müssen Sie die Methode Paint.Scale(Sx, Sy) verwenden, um das Koordinatensystem so zu skalieren, dass jede von Ihnen angegebene Koordinate dann die Einheit Millimeter hat. Mit Hilfe der Methode Paint.Translate(Tx, Ty) können Sie zusätzlich den Koordinatenursprung des Koordinatensystems nach Bedarf verschieben. Die x-Achse zeigt nach rechts, aber die y-Achse nach unten!

In den folgenden Quelltext-Abschnitten wird Ihnen gezeigt, wie Sie drei geometrische Formen (Rechteck, Kreis und Linie) und Text mit einem festgelegten Font auf einem Blatt DIN A4 zeichnen und alle Koordinaten in Millimetern angeben:

```
' Gambas class file

Private CurPrinter As Printer
Private LeftMargin As Integer = 20 '-- Dimensions in millimetres
Private TopMargin As Integer = 20
Private Rightmargin As Integer = 15
Private BottomMargin As Integer = 20

Public Sub Form_Open()
    CurPrinter = New Printer As "CurPrinter"
    CurPrinter.Paper = Printer.A4
    CurPrinter.FullPage = True
    CurPrinter.GrayScale = False
    CurPrinter.Resolution = 300
End

Public Sub btnPrint_Click()
    If CurPrinter.Configure() = False Then
       CurPrinter.Print()
    Endif
End

Public Sub CurPrinter_Begin()

    Paint.Scale(Paint.Width / CurPrinter.PaperWidth, Paint.Height / CurPrinter.PaperHeight)
    Paint.Translate(LeftMargin, TopMargin)
    Paint.FontScale = (1 / (Paint.Height / CurPrinter.PaperHeight))

End

Public Sub CurPrinter_Draw()

    Dim curX, curY, curW, curH As Float
    Dim curPrintH, curPrintW As Float
    Dim curText As String

    Paint.AntiAlias = True
    Paint.Font = Font["Monospace,10"]
'-- Print Str(Paint.Font.Size * 0.353) & " mm"

    curPrintH = CurPrinter.PaperHeight - TopMargin - BottomMargin
    curPrintW = CurPrinter.PaperWidth - LeftMargin – RightMargin

'-- Drawing a red rectangle
    curW = CurPrinter.PaperWidth - LeftMargin - RightMargin
    Paint.DrawRect(0, 0, curW, 100, Color.Red, 0.1)

'-- Drawing a black circle
    Paint.Arc(curW / 2, 50, 40, 0, 2 * Pi, False)
    Paint.Stroke()

'-- Drawing text
    curText = "»»» Drucken mit Gambas"
    curX = 0
    curY = 170
    curH =  Paint.Font.Size * 0.353
    curW = CurPrinter.PaperWidth - LeftMargin - RightMargin
    Paint.DrawText(curText, curX, curY, curW, curH, Align.Center)

'-- Draw a thin blue line
    Paint.Brush = Paint.Color(Color.Blue)   '-- Line color
    Paint.MoveTo(0, curPrintH)              '-- Starting point of the line
    Paint.LineTo(curPrintW, curPrintH)      '-- End point of the line
    Paint.LineWidth = 0.1                   '-- Line width
    Paint.Stroke()                          '-- Paint the line

End

Public Sub CurPrinter_End()
End
```

Druckerdialog
Der Aufruf des Druckerdialog und der Start des Drucks

```
If CurPrinter.Configure() = False Then
   CurPrinter.Print()
Endif
```

aus Gambas heraus zeigen unter QT einige Besonderheiten, die Sie beachten müssen:

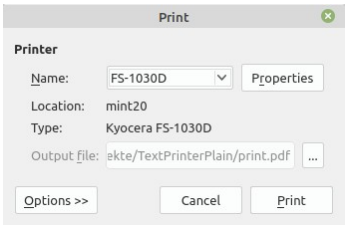

Abbildung 23.6.1.1.1: Drucker-Dialog

- Die Einstellung Properties> Pages> Orientation wird übernommen und beim Drucken realisiert. Das Paper-Format wird nicht übernommen.
- Im Gegensatz dazu werden die Einstellungen Properties> Pages> Margins beim Drucken ignoriert, weil es offenbar keine Kommunikation zwischen dem Druckdialog und der Printer-Klasse gibt.
- Die Auflösung (Einheit dpi) können Sie beim direkten Ausdruck unter Properties> Advanced> General> Resolution einstellen. Diese Einstellung im Drucker-Dialog ändert aber die Festlegung Printer.Resolution = xy im Quelltext nach den Erfahrungen der Autoren nicht!

Schriftgrößen
Die Einheit der Schriftgröße, wie sie von der Eigenschaft Paint.Font.Size zurückgegeben wird, ist der typografische Punkt. Ein typografischer Punkt ist 1/72 eines Zolls (0,353 mm) groß. Wenn also eine 10-Punkt-Schrift eine gute Größe für das Zeichnen von Text auf dem Bildschirm ist, kann sie für den Ausdruck auf Papier zu groß sein: Da die Auflösung des Druckers viel größer ist als die des Bildschirms, druckt man normalerweise alles kleiner.

Wenn Sie Methoden wie Paint.Scale(...), Paint.Translate(...) oder Paint.Rotate(...) verwenden, wird die Schriftgröße entsprechend der Paint-Matrix verändert:

```
[1]   Public Sub CurPrinter_Draw()
[2]
[3]       Paint.Font = Font["Monospace,10"]
[4]       ...
[5]   End
[6]
[7]   Public Sub CurPrinter_Begin()
[8]
[9]      Paint.Scale(Paint.Width / CurPrinter.PaperWidth, Paint.Height / CurPrinter.PaperHeight)
[10]     Paint.Translate(LeftMargin, TopMargin)
[11]     Paint.FontScale = (1 / (Paint.Height / CurPrinter.PaperHeight))
[12]
[13]  End
```

Die Anweisungen in den Zeilen 3 und 11 stellen sicher, dass Sie genau mit dem eingestellten Font drucken!

23.6.1.1.2 Text-Druck-Projekt

Das Projekt, das auf Ideen des Autors Claus Dietrich basiert, ermöglicht es Ihnen, einfache Text-Dokumente auf Druckern direkt auszudrucken oder den Inhalt der Text-Dokumente in eine PDF- oder PostScript-Datei zu drucken und zu speichern.

Eingesetzt werden wesentliche Eigenschaften, Methoden und Ereignisse der Klasse Printer (gb.qt4).

- In einem Datei-Auswahldialog können Sie die zu druckende Datei auswählen und den Text in einem Text-Editor anzeigen:

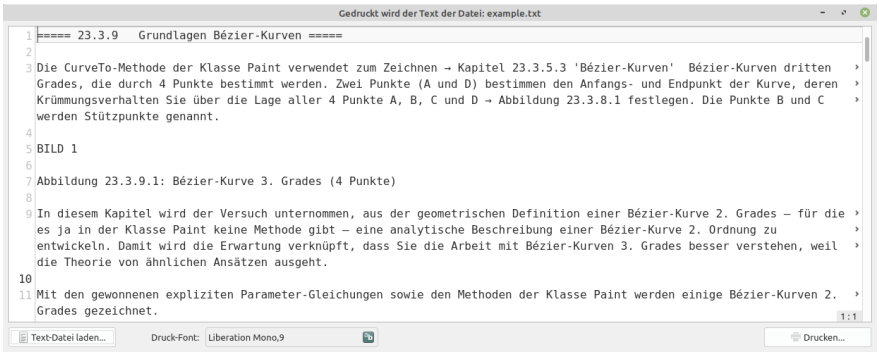

Abbildung 23.6.1.1.2: Programm-Oberfläche

- Den Text können Sie vor dem Ausdruck noch ändern. Die Änderungen werden in der ausgewählten Text-Datei abgespeichert.
- Es wird auf eine DIN A4-Seite gedruckt. Das Seitenformat kann nur im Quelltext geändert werden.
- Die Texträder sind mit festen Werten vorgegeben, die sich als praktisch erwiesen haben.
- Den vor-eingestellten Druck-Font jedoch können Sie in einem Font-Auswahldialog ändern.
- Fließtextabschnitte im Text werden in Abhängigkeit von den Seitenrändern und dem festgelegten Druck-Font umgebrochen. Eine Silbentrennung erfolgt nicht.
- Die Seitennummerierung erfolgt nicht mit Hilfe des Paginate-Ereignisses, sondern über die Bestimmung der Gesamtzahl der zu druckenden Seiten vor dem Ausdruck.
- Eine Druck-Seite enthält jeweils eine Kopfzeile mit dem Namen der Druck-Datei und dem Druck-Datum, dem nach einer Trennlinie der Seiten-Textinhalt folgt. Jede Druck-Seite besitzt eine Fußzeile, in welcher die aktuelle Seitenzahl und die Anzahl aller Druck-Seiten unter einer Linie im Format 'Seite x von y' angezeigt wird.

23.6.1.1.3 Text-Druck-Projekt: Quelltext

Der Quelltext wird komplett angegeben. Wichtige Prozeduren werden anschließend kommentiert.

```
[1]  ' Gambas class file
[2]
[3]  Private $PrintFont As Font
[4]  Private $sFileName As String
[5]  Private $sCurText As String
[6]  Private $sCurFilePath As String
[7]  Private hPrinter As Printer
[8]  Private PageTexts As New String[]
[9]
[10] '-- All 6 of the following values have been tested in practice
[11] Private LeftMargin As Integer = 20 ' In mm
[12] Private Rightmargin As Integer = 15
[13] Private TopMargin As Integer = 15
[14] Private BottomMargin As Integer = 15
[15] Private HeaderMargin As Integer = 10
[16] Private FooterMargin As Integer = 10
[17] '-----------------------------------
[18]
[19] Private LMargin As Integer  '-- In dots
[20] Private TMargin As Integer
[21] Private RMargin As Integer
[22] Private BMargin As Integer
[23] Private HMargin As Integer
[24] Private FMargin As Integer
[25]
[26]
[27] Public Sub Form_Open()
[28]
[29]     Dim sMessage As String
[30]
[31]     If Printer.List.Count = 0 Then ' True
[32]        sMessage = "<b><font size='+1', color='DarkRed'>"
[33]        sMessage &= ("PRINT MODE")
[34]        sMessage &= "</b></font><hr>"
[35]        sMessage &= "    " & ("No installed printer was detected.") & "<br>"
[36]        sMessage &= "    " & ("Printing to file only.")
[37]        Message.Error(sMessage)
[38]        cboxPrintToFile.Value = True
[39]        cboxPrintToFile.Enabled = False
[40]     Endif
[41]
[42]     $PrintFont = Font["Liberation Mono,9"] '-- Default setting
```

```
[43]     bboxSetPrintFont.Text = $PrintFont.ToString()
[44]
[45]     txtEditor.Font = Font["Monospace,12"]
[46]     txtEditor.Wrap = True
[47]     txtEditor.ShowLineNumber = True
[48]     txtEditor.ShowBraces = True
[49]     txtEditor.ShowCurrent = True
[50]     txtEditor.ShowPosition = True
[51]
[52] End
[53]
[54] Public Sub btnOpenText_Click()
[55]
[56]     Dim sFilter As String
[57]
[58]     lblFinish.Text = ""
[59]
[60]     Dialog.Title = ("Select a plain-text file")
[61]     sFilter = "*.txt;*.html;*.sql;*.sh;*.js;*.css;*.c;*.c++;*.xml;*.class;*.webpage;*.gbs"
[62]     Dialog.Filter = [sFilter, ("Text files"), "*", ("All files")]
[63]     If Dialog.OpenFile(False) Then Return
[64]     $sCurFilePath = Dialog.Path
[65]     txtEditor.Text = File.Load(Dialog.Path)
[66]     $sCurText = txtEditor.Text
[67]     $sFileName = File.Name(Dialog.Path)
[68]     FMain.Caption = ("The text of the file is printed:") & " " & $sFileName
[69]
[70] End
[71]
[72] Public Sub txtEditor_Change()
[73]     $sCurText = txtEditor.Text
[74]     txtEditor.Save($sCurFilePath)
[75] End
[76]
[77] Public Sub bboxSetPrintFont_Click()
[78]
[79]     Dialog.Title = ("Select a print font")
[80]     Dialog.Font = Font["Liberation Mono,9"] '-- Default setting
[81]     If Dialog.SelectFont() Then Return
[82]
[83]     $PrintFont = Dialog.Font
[84]     bboxSetPrintFont.Text = Dialog.Font.ToString()
[85]
[86] End
[87]
[88] Public Sub btnPrintText_Click()
[89]
[90]     If $sCurText Then
[91]        lblFinish.Text = ""
[92]        With hPrinter = New Printer As "CurPrinter"
[93]             .Paper = Printer.A4
[94]             .Resolution = 300
[95]             .FullPage = True
[96]        End With
[97]
[98]        If cboxPrintToFile.Value Then
[99]           Dialog.Title = ("Printing to file ...")
[100]            Dialog.Filter = ["*.pdf", "PDF file", "*.ps", "Post script file"]
[101]            Dialog.AutoExt = True
[102]            If Dialog.SaveFile() Then Return
[103]            hPrinter.OutputFile = Dialog.Path
[104]            hPrinter.Print()
[105]         Else
[106]            If hPrinter.Configure() = False Then
[107]               hPrinter.Print()
[108]            Endif
[109]         Endif
[110]      Endif
[111]
[112]  End
[113]
[114]
[115]  Public Sub CurPrinter_Begin()
[116]
[117]      Dim sFitPageText, sTextToPrint, sFullFitText As String
[118]      Dim hPaintExtents As PaintExtents
[119]      Dim iPages, curW, curH As Integer
[120]
[121]      lblFinish.Text = ""
[122]      Paint.Font = $PrintFont
[123]
[124]      LMargin = Paint.Width / hPrinter.PaperWidth * LeftMargin
[125]      TMargin = Paint.Width / hPrinter.PaperWidth * TopMargin
[126]      RMargin = Paint.Width / hPrinter.PaperWidth * RightMargin
[127]      BMargin = Paint.Width / hPrinter.PaperWidth * BottomMargin
[128]
```

```
[129]       HMargin = Paint.Width / hPrinter.PaperWidth * HeaderMargin
[130]       FMargin = Paint.Width / hPrinter.PaperWidth * FooterMargin
[131]
[132]   '-- Determine the number of pages - only possible via RichTextExtends
[133]   '-- TextExtends cannot be used
[134]   '-- Take not of the passed text, which converted to Rich-Text
[135]       curW = Paint.Width - LMargin - RMargin
[136]       hPaintExtents = Paint.RichTextExtents(Replace($sCurText, "\n", "<br>"), curW)
[137]       curH = Paint.Height - TMargin - HMargin - BMargin - FMargin
[138]       iPages = Ceil(hPaintExtents.Height / curH) + 1 '-- Add one page for safety
[139]
[140]   '-- Generate a text which fits into the available width. Take note, that this text may have
[141]   '-- a lot of additional line breaks (LF)! The height is bigger than the number of expected
[142]   '-- pages to that the text fully fits into the avalable space. With this we get the full
[143]   '-- text with all addtional line-breaks.
[144]
[145]       curW = Paint.W - LMargin - RMargin
[146]       curH = (Paint.H - TMargin - HMargin - BMargin - FMargin) * iPages
[147]       sFullFitText = Paint.TrimText($sCurText, curW, curH)
[148]       sTextToPrint = sFullFitText
[149]
[150]   '-- Split the full text into separate pages
[151]       PageTexts.Clear()
[152]       Do
[153]       '-- Get the part of the text which fits in to one page
[154]           curW = Paint.W - LMargin - RMargin
[155]           curH = Paint.H - TMargin - HMargin - BMargin - FMargin
[156]           sFitPageText = Paint.TrimText(sTextToPrint, CurW, CurH)
[157]       '-- Add the text the PageTexts-Array
[158]           PageTexts.Add(sFitPageText)
[159]       '-- Cut the last page out of the full text
[160]           sTextToPrint = String.Right$(sTextToPrint, -String.Len(sFitPageText))
[161]       '-- Repeat this until nothing is left to print
[162]       Loop Until sTextToPrint = ""
[163]
[164]   '-- Set the total number of pages
[165]       hPrinter.Count = PageTexts.Count
[166]
[167]   End
[168]
[169]   Public Sub CurPrinter_Draw()
[170]
[171]       Dim curX, curY, curW, curH As Integer
[172]       Dim sPrintText As String
[173]
[174]   '-- HEADER-TEXT
[175]       Paint.Font = Font["Liberation Mono,9"]
[176]       curX = LMargin
[177]       curY = TMargin
[178]       curW = Paint.W - LMargin - RMargin
[179]       curH = HMargin
[180]       sPrintText = $sFileName
[181]       Paint.DrawText(sPrintText, curX, curY, curW, curH, Align.TopLeft)
[182]       sPrintText = Format(Now, "dddd - dd.mm.yyyy")
[183]       Paint.DrawText(sPrintText, curX, curY, curW, curH, Align.TopRight)
[184]
[185]   '-- HEADER-LINE
[186]       curX = LMargin
[187]       curY = TMargin + (Paint.Width / hPrinter.PaperWidth) * 3.8 '-- mm
[188]       curW = Paint.W - RMargin
[189]       Paint.MoveTo(curX, curY)    '-- Starting point of the line
[190]       Paint.LineTo(curW, curY)    '-- End point of the line
[191]       Paint.LineWidth = 0.1       '-- Line width
[192]       Paint.Stroke()              '-- Paint the line
[193]
[194]   '-- CONTENT
[195]   '-- If the text fits the requested rectangle, it is returned as is.
[196]   '-- Otherwise it is trimmed and followed by an ellipsis character so that it fits.
[197]   '-- Remove "…" if found at the end of a page
[198]       If String.Right$(PageTexts[hPrinter.Page - 1], 1) = "…" Then
[199]          PageTexts[hPrinter.Page - 1] = String.Left$(PageTexts[hPrinter.Page - 1], -1)
[200]       Endif
[201]       Paint.Font = $PrintFont
[202]       curX = LMargin
[203]       curY = TMargin + HMargin
[204]       curW = Paint.W - LMargin - RMargin
[205]       curH = Paint.H - TMargin - HMargin - BMargin - FMargin
[206]       sPrintText = PageTexts[hPrinter.Page - 1]
[207]       Paint.DrawText(sPrintText, curX, curY, curW, curH)
[208]
[209]   '-- FOOTER-LINE
[210]       curX = LMargin
[211]       curY = Paint.H - BMargin - (Paint.Width / hPrinter.PaperWidth) * 4.0 '-- mm
[212]       curW = Paint.W - RMargin
[213]       Paint.MoveTo(curX, curY)    '-- Starting point of the line
[214]       Paint.LineTo(curW, curY)    '-- End point of the line
```

```
[215]      Paint.LineWidth = 0.1       '-- Line width
[216]      Paint.Stroke()              '-- Paint the line
[217]
[218]  '-- FOOTER-TEXT
[219]      Paint.Font = Font["Liberation Mono,9"]
[220]      curX = LMargin
[221]      curY = Paint.H - FMargin - BMargin
[222]      curW = Paint.W - LMargin - RMargin
[223]      curH = HMargin
[224]
[225]      sPrintText = ("Page") & " " & hPrinter.Page & " " & ("of") & " " & hPrinter.Count
[226]      Paint.DrawText(sPrintText, curX, curY, curW, curH, Align.BottomRight)
[227]
[228]  End
[229]
[230]  Public Sub CurPrinter_End()
[231]      lblFinish.Alignment = Align.Center
[232]      lblFinish.Text = "        " & ("The text has been printed successfully!")
[233]  End
[234]
[235]  Public Sub Form_Close()
[236]
[237]      Dim hWindow As Window
[238]
[239]  '-- Close all open windows
[240]      For Each hWindow In Windows
[241]          hWindow.Close()
[242]      Next
[243]
[244]  End
```

Kommentar

- In den Zeilen 31 bis 50 wird zuerst geprüft, ob ein Drucker im System installiert ist, dann wird auch der Vorgabe-Font für den Font-Auswahldialog festgelegt. Abschließend werden wesentliche Eigenschaften des Editors für die Anzeige des Inhaltes der ausgewählten Textdatei bestimmt.
- In einem Datei-Auswahldialog in den Zeilen 60 bis 68 können Sie die zu druckende Datei auswählen.
- In einem Font-Auswahldialog können Sie in den Zeilen 77 bis 86 den eingestellten Druck-Font ändern.
- In den Zeilen 88 bis 112 wird zuerst ein neues Printer-Objekt erzeugt und konfiguriert. Wenn der Ausdruck in eine Datei aktiviert wurde, dann wird der Text in eine Datei 'gedruckt'. Sonst wird ein Dialog zum Konfigurieren eines Druckers aufgerufen und abschließend der Ausdruck mit der Methode hPrinter.Print() gestartet. Die Methode gibt *False* zurück, wenn der Ausdruck erfolgreich war.
- In der Ereignisbehandlung CurPrinter_Begin() in den Zeilen 115 bis 167 werden zuerst die Ränder von der Einheit Millimeter in XY umgerechnet (Zeilen 124 bis 133). Dann wird die Anzahl der zu druckenden Seiten in Abhängigkeit von der bedruckbaren Breite und dem festgelegten Druck-Font berechnet.
- In der Ereignisbehandlung curPrinter_Draw() in den Zeilen 169 bis 228 werden die einzelnen Seiten gezeichnet. Wer mit Paint arbeitet wird feststellen: Die dedizierte Festlegung von curX, curY, curW, curH oder curText vor jedem Zeichnen – in erster Linie von Text – ist ein guter Weg, um sich genau das Rechteck vorzustellen, das den Container bildet, in dem man den Text zeichnen möchte.
- Zuerst werden in den Zeilen 174 bis 192 auf jeder Seite eine Kopfzeile mit dem Namen der Druck-Datei (linksbündig) und dem Druck-Datum (rechtsbündig) sowie einer Trennlinie darunter gezeichnet.
- Dann folgt der Seitentext mit jeweils geänderter Seitenzahl, der in den Zeilen 194 bis 207 gezeichnet wird.
- Abschließend werden in den Zeilen 209 bis 226 auf jeder Seite eine Fußzeile mit der aktuellen Seitenzahl und der Anzahl aller Druck-Seiten im Format 'Seite x von y' gezeichnet, die rechtsbündig unter einer Trennlinie steht.