```
twips MACRO arg
  push arg
  call twipsP
  EXITM <eax>
ENDM
;       567 twips per cm: A4 = 21 X 29,7 cm
;       twips = 21*567 X 29.7X567 = 11907 * 16839,9

PrintRTF proc
LOCAL fSuccess:SDWORD
LOCAL ptrScaling
LOCAL hPrDC:DWORD
LOCAL OldSel:CHARRANGE
LOCAL psd:PAGESETUPDLG
LOCAL docInfo:DOCINFO
LOCAL fr:FORMATRANGE
  call ClearLocVars                                   ; clear all structure elements
  push edi
  push esi
  mov psd.lStructSize, sizeof PAGESETUPDLG
.data?
  prtMargins        dd 4 dup(?)             ; filled with prtL etc IniFile$ settings
  dmScaleDef        dd ?
.code
  mov esi, offset prtMargins
  lea edi, psd.rtMargin
  m2m ecx, 4
  push ecx
  push esi
  push edi
  mov psd.Flags, PSD_INHUNDREDTHSOFMILLIMETERS or PSD_MARGINS
  rep movsd
  m2m psd.hwndOwner, hOwn
  invoke PageSetupDlg, addr psd                       ; get a printer and page settings
  pop esi               ; the unchanged order is
  pop edi               ; intentional: we swap the pointers
  pop ecx
  .if eax==0
        invoke CommDlgExtendedError
        test eax, eax
        jne PriError
  .else
        rep movsd
        invoke GlobalLock, psd.hDevMode
        push eax
        mov edx, dmScaleDef             ; ---------- optional scaling of output, prtScale in IniFile$ ----------
        mov ptrScaling, edx             ; dmScaleDef could be e.g. 70; no action if it is zero
        test edx, edx
        .if !Zero?
            mov [eax.DEVMODE.dmScale], dx
        .endif
        mov esi, rv(GlobalLock, psd.hDevNames)
        movzx eax, word ptr [esi.DEVNAMES.wOutputOffset]
        add eax, esi
        push eax
        movzx eax, word ptr [esi.DEVNAMES.wDeviceOffset]
        add eax, esi
        push eax
        movzx eax, word ptr [esi.DEVNAMES.wDriverOffset]
        add eax, esi
        push eax
        call CreateDC     ;  hPrDC=CreateDC(lpszDriver, lpszDevice, lpszOutput, pDeviceMode)
        mov hPrDC, eax
        invoke GlobalUnlock, psd.hDevNames
        invoke GlobalUnlock, psd.hDevMode
        invoke GlobalFree, psd.hDevNames
        invoke GlobalFree, psd.hDevMode
```

```
        mov docInfo.cbSize, sizeof DOCINFO
        mov docInfo.lpszDocName, FileBody$  ; chr$("TinyRtf")
        invoke StartDoc, hPrDC, addr docInfo            ; start a print job
        .if eax==SP_ERROR
              invoke DeleteDC, hPrDC
              jmp PriError
        .endif
;       invoke SendMessage, hRE, EM_SETTARGETDEVICE, hPrDC, cxPhys ; not very useful
        m2m fr.hdc, hPrDC
        m2m fr.hdcTarget, hPrDC

;       mov cxPhys, rv(GetDeviceCaps, hPrDC, PHYSICALWIDTH) yields 4958, a factor 2.x too small
;       mov cyPhys, rv(GetDeviceCaps, hPrDC, PHYSICALHEIGHT) yields 7017 - expected 16840 for A4

        mov fr.rc.left, twips(psd.rtMargin.left)   ; psd:PAGESETUPDLG
        neg eax
        mov fr.rc.right, eax
        add fr.rc.right, twips(psd.ptPaperSize.x)
        sub fr.rc.right, twips(psd.rtMargin.right)

        mov fr.rc.top, twips(psd.rtMargin.top)
        neg eax
        mov fr.rc.bottom, eax           ; bottom=height-top

        push psd.ptPaperSize.y
        call twipsP
        mov ecx, ptrScaling             ; ---------- optional scaling ----------
        test ecx, ecx
        .if !Zero?                      ; this hack will work with the Adobe PDF "printer",
              cdq                       ; but not with real printers that do not allow scaling
              div ecx
              imul eax, eax, 100      ; if scale=25%, take height, divide by 25 and multiply with 100
        .endif
        add fr.rc.bottom, eax           ; twips(psd.ptPaperSize.y)
;       deb 1, "Test", eax, ptrScaling, fr.rc.bottom
        sub fr.rc.bottom, twips(psd.rtMargin.bottom)

;       Get the current selection into a CHARRANGE
        invoke SendMessage, hRE, EM_EXGETSEL, 0, addr fr.chrg
        mov eax, fr.chrg.cpMax
        mov edx, fr.chrg.cpMin
        mov OldSel.cpMax, eax
        mov OldSel.cpMin, edx
        sub eax, edx
        .if sdword ptr eax<=127         ; User has not selected a lot of text, therefore print all pages
              invoke SendMessage, hRE, EM_SETSEL, 0, -1
              invoke SendMessage, hRE, EM_EXGETSEL, 0, addr fr.chrg
        .endif

;       Use GDI to print successive pages
        .Repeat
              invoke StartPage, hPrDC
              mov fSuccess, eax
              .Break .if sdword ptr eax<=0
              push fr.rc.bottom
              invoke SendMessage, hRE, EM_FORMATRANGE, 1, addr fr
              pop fr.rc.bottom
;             The rc.bottom member may be changed after the message is sent. If it is changed, it must indicate the
;             largest rectangle that can fit within the bounds of the original rectangle and still contain the specified
;             text without printing partial lines. It may be necessary to reset this value after each page is printed.
;             These dimensions are given in TWIPS.  (MS Support)
              .Break .if eax<=fr.chrg.cpMin ; relevant for an empty doc with hidden chars in front
              .Break .if eax>=fr.chrg.cpMax
              mov fr.chrg.cpMin, eax
              invoke EndPage, hPrDC
              mov fSuccess, eax
        .Until sdword ptr eax<=0
        invoke SendMessage, hRE, EM_FORMATRANGE, 0, 0 ; free the cache, important
        .if fSuccess>0
```

```
            invoke EndDoc, hPrDC
        .else
            invoke AbortDoc, hPrDC
        .endif
        invoke DeleteDC, hPrDC
        invoke SendMessage, hRE, EM_EXSETSEL, 0, addr OldSel            ; restore old selection
  .endif
;  mov eax, fSuccess
@@:
  pop esi
  pop edi
  ret
PriError:
  MsgBox 0, "Printing problem", 0, MB_OK
  jmp @B
PrintRTF endp


twipsP proc
.data
  tw2cm    REAL4 0.567
.code
  ffree st(7)
  ffree st(7)
  fld tw2cm
  fild dword ptr [esp+4]
  fmul
  fistp dword ptr [esp+4]
  pop edx
  pop eax
  jmp edx
twipsP endp
```