```
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 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(?)
.code
  mov esi, offset prtMargins
  lea edi, psd.rtMargin
  m2m ecx, 4
  push ecx
  push esi
  push edi
  .if dword ptr [esi+8]           ; take the right margin as flag
                      mov psd.Flags, PSD_INHUNDREDTHSOFMILLIMETERS or
PSD_MARGINS
                      rep movsd
  .else
                      mov psd.Flags, PSD_INHUNDREDTHSOFMILLIMETERS or
PSD_DEFAULTMINMARGINS
  .endif
  invoke PageSetupDlg, addr psd                         ; get a printer
device context
  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
       mov esi, rv(GlobalLock, psd.hDevNames)
       push rv(GlobalLock, psd.hDevMode)
       mov edx, esi
       movzx eax, word ptr [esi.DEVNAMES.wOutputOffset]
       add edx, eax
       push edx
       mov edx, esi
       movzx eax, word ptr [esi.DEVNAMES.wDeviceOffset]
       add edx, eax
       push edx
       mov edx, esi
       movzx eax, word ptr [esi.DEVNAMES.wDriverOffset]
       add edx, eax
       push edx
       call CreateDC     ;  hPrDC=CreateDC(lpszDriver, lpszDevice, lpszOutput,
pDeviceMode)
       mov hPrDC, eax
       push eax
       invoke GlobalUnlock, psd.hDevNames
       invoke GlobalUnlock, psd.hDevMode
       pop eax
       mov docInfo.cbSize, sizeof DOCINFO
       mov docInfo.lpszDocName, chr$("TinyRtf")
       invoke StartDoc, hPrDC, addr docInfo          ; start a print job
```

```
        .if eax==SP_ERROR
            invoke DeleteDC, hPrDC
            jmp PriError
        .endif
;       invoke SendMessage, hEdit, EM_SETTARGETDEVICE, hPrDC, cxPhys
        ; not needed
        m2m fr.hdc, hPrDC
        m2m fr.hdcTarget, hPrDC

        mov fr.rc.left, twips(psd.rtMargin.left)
        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
        add fr.rc.bottom, twips(psd.ptPaperSize.y)
        sub fr.rc.bottom, twips(psd.rtMargin.bottom)

;       Get the current selection into a CHARRANGE
        invoke SendMessage, hEdit, 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, hEdit, EM_SETSEL, 0, -1
            invoke SendMessage, hEdit, 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
            invoke SendMessage, hEdit, EM_FORMATRANGE, 1, addr fr
            .Break .if eax<=fr.chrg.cpMin
            .Break .if eax>=fr.chrg.cpMax
            mov fr.chrg.cpMin, eax
            invoke EndPage, hPrDC
            mov fSuccess, eax
        .Until sdword ptr eax<=0
        invoke SendMessage, hEdit, EM_FORMATRANGE, 0, 0          ; free the
cache, important
        .if fSuccess>0
            invoke EndDoc, hPrDC
        .else
            invoke AbortDoc, hPrDC
        .endif
        invoke DeleteDC, hPrDC
        invoke SendMessage, hEdit, 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

ClearLocVars proc          ; put "call ClearLocals" as first instruction after
LOCALS - eax unchanged on exit
  push eax                 ; do not use with uses esi etc - push them
manually behind the call!
  lea eax, [esp+8]         ; pushed eax and ret address
  mov esp, ebp             ; base page of calling procedure
  align 4                  ; 74 instead of 123 cycles on Celeron M, no effect
on P4
@@:
  push 0                   ; 120 bytes: 196 cycles on P4
  cmp esp, eax             ; rep stosd??
  ja @B
  sub esp, 8               ; 19 bytes with align 4
  pop eax
  ret
ClearLocVars endp
```