DVIOUT
DVIPRT

Ver. 2.43.2 Technical Guide

目 次

表 目 次

図 目 次

ファイル例目次

第1章 dviout/dviprtの概要

dviout/dviprtは、TEX[1]のデバイスドライバと呼ばれるものの1つです。これは、DVI[2]ファイルというTEXによるタイプセットの結果得られた出力を解釈し、コンピュータの画面やプリンタに出力するものです。特に、画面へ表示を行なうものをプレビューア、プリンタへ出力するものをプリンタドライバと呼びます。dvioutがプレビューア、dviprtがプリンタドライバに対応します。

TEXそのものは、組版のみを行なうソフトウェアであり、文字などは大きさとして捉え、形には関与しません。プレビューアやプリンタドライバが、文字の形状など情報を収集し、TEXが出力した組版結果を目に見える形にするのです。これらプレビューア/プリンタドライバは、TEXの処理系には含まれていないのが普通です。プログラムの性格上、実行するハードウェアの入出力装置に依存するためです。

事実、dviout/dviprtの場合にも実行するコンピュータの機種に応じて、数種類の実行ファイル[3]を用意しています。ご自分の使用している機種に合わせて、必要な実行ファイルを選んでインストールしてください。

dviout/dviprtのソースファイルをコンパイル可能であれば、コンパイル時にコンパイル・オプションを指定することによって、種々の機能を選択することが可能です (readmesr.docをご覧ください)。

1.1 サポートする機能と対応機種

1.1.1 対応するTEXの種類

- NTT JTEX
- アスキー日本語TEX
- アスキーpTEX[4]

1.1.2 拡張機能 (\special)

- tpic specials
- PostScript specials
- EPS/PS/PBM/GIF形式の画像データ読み込み
- プリンタのページ記述言語の取り込み

1.1.3 コンピュータ

- PC-9801シリーズ、ハイレゾ版PC-9801シリーズ
  - アイ・オー・データ機器のグラフィックアクセラレータボード GA-1024A/1280A対応[5]
- J-3100シリーズ
- AX仕様パソコン
- IBM PC, DOS/V対応機
  - VGA(640×480)

[1] American Mathematical Society の商標です
[2] Device Independent、すなわち特定の機種に依存しない
[3] dviprtは、PC-9801, DOS/V, J-3100の統合版があります
[4] 以下、特に断らない限り、アスキー日本語TEXという表現でアスキーpTEXを含むものとします
[5] canopusのPowerWindow801/801+/928対応のものが、ぱろっと氏により作成されている

  - Super VGA, VESA(800 × 600)
  - Super VGA, VESA(1024 × 768)
  - VESA(1280 × 1024)
- HP100LX/HP200LX
  - CGA(640 × 200)
- PC-100 や FMR50/60/70 対応のものも存在するが、標準パッケージには含まれない

1.1.4 オペレーティング・システム

- MS-DOS Ver.3.1[6]以降
  - EMS メモリーに対応
- MINIX、Linux、386BSD などの OS に移植したものも存在する

1.1.5 プリンタ

- EPSON ESC/P 系、NEC PC-PR 系、NEC NM 系の 24 pin プリンタ
- Canon LIPS III (300dpi)、EPSON ESC/Page (600/300/240dpi) 仕様の LBP
- その他の多数のプリンタに対して、プリンタ定義ファイルを作成することで対応が可能

現在、プリンタ定義ファイルが用意されているプリンタは次の 1.1 のとおりです。

Brother	:	M-1024II P/X(180dpi), HL-8e(300 dpi)
Casio	:	LCS-240(240 dpi)
Canon	:	BJ-10/15(48 pin), BJ-130J/300J(48 pin), LBP-8/LIPS II(240/300 dpi)
Epson	:	ESC/P 8 pin, AP-850/MJ-500(48 pin), AP-900(48 pin), HG-4800(ESC/P 48 pin)
	:	HG-5130(ESC/P 48 pin), LP-2000/3000(300 dpi), LP-7000(240 dpi)
	:	MJ-700V2C/800C/900C/3000CU(720/360dpi)
Fujitsu	:	FMPR-351(48 pin)
Kyocera	:	L-880(300dpi)
HP	:	DeskJet 500J/300J/505J(300 dpi), LaserJet Plus(300 dpi), HP LaserJet 4PJ(600 dpi)
NEC	:	PC-8821(16 pin), PC-PR150V(48 pin), PC406LM(48 dots), PC-PR602(240 dpi)
	:	PC-PR1000/2000/4000(240 dpi), PC-PR2000/4000(400 dpi)
Ricoh	:	LP3320-SP4 mark2/LP2310-SP5(240 dpi)
Sanyo	:	PR-130AX(180 dpi)
Star	:	JT-48(48 pin)
Toshiba	:	PR-48(J31DPR02 180dpi)
Xerox	:	ART2(240 dpi), ART(Hex mode 240 dpi)

表 1.1: プリンタ定義ファイルが用意されているプリンタ

1.1.6 欧文フォントおよび NTT JTEX の和文フォント

- PK, PXL1001, PXL1002, PXL1003 フォーマットのフォント
- これらを束ねたフォントライブラリ
  - アスキー日本語 MicroTEX の FAR (フォントアーカイブ) ファイル
  - emTEX の FLI (フォントライブラリ) ファイル

[6]アプリケーションが使えるフリーエリアが大きくとれるほど快適に運用できるため、MS-DOS Ver 5.0 以降が便利

  - 独自の GTH ファイル、PKD ファイル
- TrueType Font
- 不足しているフォントの`METAFONT` による自動生成機能あり

1.1.7 和文フォント

アスキー日本語 TEX および NTT JTEX ともに、

- アスキー日本語 TEX の JXL4 フォーマットのフォント
- 和文代替フォントとして
  - パソコン内蔵の 16 × 16 ドット[7]のシステムフォント
  - ビットマップ形式のファイル、および、それを PK/JXL4 フォーマットと同様な圧縮法で圧縮したもの
  - (株) ツァイトの「書体倶楽部」/「JG Font」アウトラインフォント
  - Microsoft Windows 3.1 および Windows95 標準の TrueType Font
  - Canon LIPS III や Epson ESC/Page の LBP 内蔵のスケーラブルフォント

1.1.8 出力

- ディスプレイ (`dviout`)
  - GA-1024A/1280A 対応版では、Microsoft Windows 標準の BMP ファイル出力が可能
- プリンタ (`dviprt`)
- `dviprt` では標準出力やファイルに次の形式で出力可能
  - プリンタに送るバイナリコード
  - raw PBM 画像データ[8]
  - EPS 形式画像データ
  - G3 FAX 用の画像データ
  - デバッグ用の 16 進ダンプ

1.2 フォントファイルの指定

TEX の生成した DVI ファイルにおいて使われているフォントに対し、それを出力するには、ディスプレイやプリンタの解像度に応じたビットマップのフォントデータが必要です。さらにそのフォントデータを、`dviout`/`dviprt` が正しく見つけることができるように、設定を行なう必要があります。

そのために、決められた形式に従ってフォントファイル名を環境変数`TEXPK` で指定します。また、和文フォントとしてビットマップフォントを利用する場合は、そのファイル名を環境変数`TEXKNJ` にて指定する必要があります。

これらが正しく設定されていないと、`dviout`/`dviprt` を使うことができません。多くの場合「フォントファイルが見つからない」というエラーになりますが、このときはどのようなフォントファイルを探して見つからなかったかが表示されますので、それを参考にして環境変数`TEXPK` を設定し直してください。

また、環境変数`TEXKNJ` が正しく設定されていないと、ディスク上のフォントファイルが利用できないため、常にコンピュータのシステムフォントが使われることがあります (オプション `-f` や、`-f=1` をつけ

[7] J-3100 や PC-9801 ハイレゾモードのときは 24 × 24, DOS/V においてはそのとき使用されているサイズ
[8] Portable Bitmap

てdviout/dviprt を起動することにより、どのようなフォントが「使われるのか」、あるいは「使われたのか」がわかります。cf. 第 4 章 p.27)。

フォントファイルの指定を行なう環境変数TEXPK、TEXKNJ の内容は、dviout.par, dviprt.par などのパラメータファイルに記述することも可能です (cf. 第 2 章 p.10)。これは、環境変数領域 (メモリー) を節約できるため、極めて有効です。

1.3 オプション・パラメータの指定

dviout/dviprt は、多くのパラメータを持っていますが、指定するものすべてをコマンド行に記述する必要はなく、常用するオプション・パラメータはファイルdviout.par/dviprt.par に、dviprt での出力ページの指定は**ページ指定ファイル**に書いておくことができます[9]。

1.4 dviout.par とdviprt.par (パラメータファイル)

パラメータファイルの存在するディレクトリは、環境変数TEXCFG で指定します。ですから、その環境変数の内容を変更することによって、利用形態に応じた各種の指定を簡単に切り替えることが可能です (cf. 第 2 章 p.10)。

また、dviout/dviprt は、そのプログラム名をdviout.exe/dviprt.exe 以外に変更すると、変更後のプログラム名の最初の 3 文字がdvi である場合に限り、そのプログラム名の拡張子を .par に変えたものをオプション・パラメータを記述してあるファイルとみなし、通常のdviout.par/dviprt.par の場合と同じように検索します。つまり、その名前のファイルが存在すれば読み込みますが、存在しなければ読み込みを行ないません。

たとえば、dviout.exe をdviouts.exe というファイル名にして実行すると、dviouts.par が用いられdviout.par は使用されません。

1.5 ページ指定ファイル

出力するページ番号、空のページを出力する指示、ページ出力後の一時停止の制御をファイル内に記述し、dviprt の起動時にそのファイル名を記述することで、印刷ページの制御が可能となっています (cf. 第 6 章 p.56)。

[9]DVI ファイル名の指定とdviout での出力ページの指定は、コマンドラインでしか使えない

第2章 オプション概観

2.1 起動画面 (dviout)

PC-9801 用のdviout の起動画面です。

```
TeX DVI PREVIEWER PC-9801 Ver.2.43.2
Copyright(C) 1988-89 by TSG, 90-96 by SHIMA,
             1991-95 changed by Akiii, hero.h, Matsuda, Naochan!, Oh-Yeah?, OkI
                SOLITON, sempa, T.Minagawa, Tomiie, T.Uchiyama, Yakumo
                Yoshizawa

usage : dviout [option(s)] dvifile [page(s) eg. 6-8 3+]

 -mag= : mag step(0-9), mag(500-)
 -half : mag step(half) [デフォルト -]
 -br=  : 展開フォント      バッファサイズ (byte 単位)
 -bf=  : フォントファイル バッファサイズ (byte 単位)
 -bb=  : 展開ビットマップ バッファサイズ (byte 単位)
 -bw=  : ワーク用          バッファサイズ (byte 単位)
 -bv=  : イメージ表示での保存ページ数 (EMS のときは負数)
 -EMS= : EMS の使用ページ数 (デフォルト 全部)
 -f=   : 情報表示 (0-3)
 -wait=: エラー時 [0: 停止 1: キー入力を待つ (デフォルト) 2-3: 無視]
 -dpi= : dpi 設定
 -DPI= : dpi(縦方向)
 -e=   : 拡大率 (1000 倍)
 -page : オフセットページ指定 [デフォルト -]
 -X=   : 横方向位置調整 (ドット単位)
 -Y=   : 縦方向位置調整 (ドット単位)
 -w=   : 横方向の幅調整 (ドット単位)
 -h=   : 縦方向の幅調整 (ドット単位)
 -tpic=: tpic special[0:ignore 1:Bezier(デフォルト) 2:spline]
 -GS=  : Ghostscript[0:Off 1:On(default) 2:verbose 3:PBM 4:exact PBM 5:gssub]
 -GIF  : pbmraw の代わりに gifmono を使う [デフォルト -]
 -gsx= : Ghostscript(デフォルト gs.exe)
 -gdat=: pbm/gif ファイルのディレクトリ
 -gbox : 取り込み画像の枠表示 [デフォルト -]
 -gsize: 取り込み画像サイズ :mag 込み [デフォルト -]
 -u=   : ページの切り替え・スピード (0,1,2,...)
 -s=   : 半画面スクロール・スピード (0,1,2,...)
 -sh=  : 横方向スクロール・スピード (0,1,2,...)
 -t=   : 縦方向スクロール・スピード (-16,-15,..,0,1,2,...)
 -R=   : 表示カラー変更 (<RGBrgb> デフォルト白黒反転)
 -B=   : ページ・レジューム
 -r=   : フォントキャッシュのセーブ
 -L    : 長いフォント名の中央省略 [デフォルト -]
 -c    : フォントの整合性チェック [デフォルト -]
 -F=   : 代替フォント指定
 -A=   : フォントサーチ範囲拡大
 -K=   : 倍角漢字指定
 -G=   : ゴシック漢字指定
 -g    : 横書き用漢字フォントを使用 (pTeX 縦書きモード) [デフォルト -]
 -JC=  : JIS 文字コード変換
 -romf=: 内蔵漢字フォントを使用するドット数 (<max>[:<min>])
 -k=   : -romf= から除くドット数 (<dot>:<dot>:...)
 -varf : フォントをスケーラブルにする [デフォルト -]
```

```
-base : ベースラインを引く [デフォルト -]
-box  : 文字の代わりに箱で組版 [デフォルト -]
-gen= : 不足フォントの自動生成
-vfn= : 書体倶楽部/JG/和文 TrueType フォント
-ttf= : 欧文 TrueType フォント [0:Off(default) <dot>:Draw sw]
-nttF=: NTTjTeX での非標準和文フォントの代替フォント指定
-ntt  : NTTjTeX の和文フォントをアスキー日本語 TeX 用フォントで代用 [デフォルト -]
-S=   : 漢字スケール (1000 倍) 標準値 952
-J=   : 漢字ベースライン (1000 倍) 標準値 0
-W=   : V横幅 (ドット単位)
-H=   : V縦幅 (ドット単位)
-P=   : V横位置 (ドット単位)
-Q=   : V縦位置 (ドット単位)
-C    : Vセンタリング [デフォルト -]
-M    : 展開ビットマップを -W, -H で指定したサイズに [デフォルト -]
-E=   : 終了状況

-newsize: 以下のサイズ・オプションを使用 [デフォルト +]
-PW=  : 用紙の横幅 [デフォルト 210mm(A4)]
-PH=  : 用紙の縦幅 [デフォルト 297mm(A4)]
-y=   : 用紙の選択 [A3|A4|A5|B4|B5|H(葉書)[p|l]]
-TM=  : 用紙上端の印字不可能域の幅
-BM=  : 用紙下端の印字不可能域の幅
-LM=  : 用紙左端の印字不可能域の幅
-RM=  : 用紙右端の印字不可能域の幅
-MW=  : 印字可能域の最大横幅
-MH=  : 印字可能域の最大縦幅
-OX=  : 横方向の移動
-OY=  : 縦方向の移動
-V    : 用紙サイズの縦横を逆にする [デフォルト -]
-esize: -e= の影響を受ける [デフォルト -]
-HC   : 水平センタリング [デフォルト -]
-VC   : 垂直センタリング [デフォルト -]
-HS=  : 水平センタリング後の移動
-VS=  : 垂直センタリング後の移動

-=    : パラメータファイル指定
```

上記オプションは、 parameter file <dviout.par> で指定可能

2.2 起動画面 (dviprt)

PC-9801, DOS/V, J-3100 統合版のdviprt の起動画面です。

```
TeX PRINTER DRIVER PC98(+hires)/DOSV/J3100 Ver.2.43.2
Copyright(C) 1988-89 by TSG, 90-96 by SHIMA,
             1991-95 changed by Akiii, hero.h, Matsuda, Naochan!, Oh-Yeah?, OkI
                SADA, SOLITON, sempa, T.Minagawa, Tomiie, T.Uchiyama, Yakumo
                Yoshizawa

usage : dviprt [option(s)] dvifile [page(s) eg. -4 7 . 9 12-14 20- @file]

 -mag= : mag step(0-9), mag(500-)
 -half : mag step(half) [デフォルト -]
 -br=  : 展開フォント      バッファサイズ (byte 単位)
 -bf=  : フォントファイル バッファサイズ (byte 単位)
 -bb=  : 展開ビットマップ バッファサイズ (byte 単位)
 -bw=  : ワーク用          バッファサイズ (byte 単位)
 -EMS= : EMS の使用ページ数 (デフォルト 全部)
 -f=   : 情報表示 (0-3)
 -wait=: エラー時 [0: 停止 1: キー入力を待つ (デフォルト) 2-3: 無視]
 -dpi= : dpi 設定
 -DPI= : dpi(縦方向)
 -e=   : 拡大率 (1000 倍)
 -page : オフセットページ指定 [デフォルト -]
 -nf=  : 改ページコード不使用 (袋とじは 2)
         LBP では、袋とじの右ページの開始位置
 -o=   : 出力ページ (r:reverse, e:even, o:odd)
 -V    : 90 度回転 [デフォルト -]
 -X=   : 横方向位置調整    (ドット単位)
 -Y=   : 縦方向位置調整    (ドット単位)
 -w=   : 横方向の幅調整    (ドット単位)
 -h=   : 縦方向の幅調整    (ドット単位)
 -W=   : 用紙の印字可能横幅 (ドット単位)
 -H=   : 用紙の印字可能縦幅 (ドット単位)
 -M    : 展開ビットマップを用紙印字可能サイズに [デフォルト -]
 -T    : 単票用紙上空き修正 [デフォルト -]
 -C    : センタリング [デフォルト -]
 -tpic=: tpic special[0:ignore 1:Bezier(デフォルト) 2:spline]
 -GS=  : Ghostscript[0:Off 1:On(default) 2:verbose 3:PBM 4:exact PBM 5:gssub]
 -GIF  : pbmraw の代わりに gifmono を使う [デフォルト -]
 -gsx= : Ghostscript(デフォルト gs.exe)
 -gdat=: pbm/gif ファイルのディレクトリ
 -gbox : 取り込み画像の枠表示 [デフォルト -]
 -gsize: 取り込み画像サイズ :mag 込み [デフォルト -]
 -r=   : フォントキャッシュのセーブ
 -L    : 長いフォント名の中央省略 [デフォルト -]
 -c    : フォントの整合性チェック [デフォルト -]
 -F=   : 代替フォント指定
 -A=   : フォントサーチ範囲拡大
 -K=   : 倍角漢字指定
 -G=   : ゴシック漢字指定
 -S=   : 漢字スケール (1000 倍) 標準値 952
 -g    : 横書き用漢字フォントを使用 (pTeX 縦書きモード) [デフォルト -]
 -JC=  : JIS 文字コード変換
 -romf=: 内蔵漢字フォントを使用するドット数 (<max>[:<min>])
 -k=   : -romf= から除くドット数 (<dot>:<dot>:...)
 -varf : フォントをスケーラブルにする [デフォルト -]
 -base : ベースラインを引く [デフォルト -]
 -box  : 文字の代わりに箱で組版 [デフォルト -]
```

```
-gen= : 不足フォントの自動生成
-vfn= : 書体倶楽部/JG/和文 TrueType フォント
-ttf= : 欧文 TrueType フォント [0:Off(default) <dot>:Draw sw]
-nttF=: NTTjTeX での非標準和文フォントの代替フォント指定
-ntt  : NTTjTeX の和文フォントをアスキー日本語 TeX 用フォントで代用 [デフォルト -]
-J=   : 漢字ベースライン (1000 倍) 標準値 0
-y=   : 用紙の選択 [A3|A4|A5|B4|B5|H(葉書)|F<width>[:<height>]|T<number>
        LIPS3   :[p(用紙縦)|l(用紙横)[0(自動)|1(手差し)|2(上段)|3(下段)]]
        ESC/Page:[p(用紙縦)|l(用紙横)[1(標準給紙)|2(オプション給紙)[:<ビン番号>]]]]
-lc=  : LBP でのコピー枚数
-R    : LBP 内蔵, 書体倶楽部のフォントを 90 度回転 [デフォルト -]
-O=   : 出力ファイル (!!!は頁番号)
-s=   : 速度調整 (0,...,10000)
-Z    : １頁毎に一旦停止 [デフォルト -]
-I=   : プリンタ初期化コード記述ファイル

-newsize: 以下のサイズ・オプションを使用 [デフォルト +]
-PW=  : 用紙の横幅 [デフォルト 210mm(A4)]
-PH=  : 用紙の縦幅 [デフォルト 297mm(A4)]
-TM=  : 用紙上端の印字不可能域の幅
-BM=  : 用紙下端の印字不可能域の幅
-LM=  : 用紙左端の印字不可能域の幅
-RM=  : 用紙右端の印字不可能域の幅
-MW=  : 印字可能域の最大横幅
-MH=  : 印字可能域の最大縦幅
-OX=  : 横方向の移動
-OY=  : 縦方向の移動
-HC   : 水平センタリング [デフォルト -]
-VC   : 垂直センタリング [デフォルト -]
-HS=  : 水平センタリング後の移動
-VS=  : 垂直センタリング後の移動
-PF=  : 給紙位置 [1: 左端揃 (デフォルト) 2: 中央]

-=    : パラメータファイル指定
-FAX= : G3 FAX 形式
-comp=: データ圧縮 [0: 無 1:PCL mode 1 2:PCL mode 2]
-p=   : プリンタの種類 e:ESC/P p:PC-PR n:NM o:other(default prtctl.cfg)
       l:LIPS3 m:ESC/Page [v: 字体縦横比 k: 和文外字 d:download_size r:real_size
       m:minimal_count f:font_def D:dpi j:baseline s:size_adj E:edge
       u:minimal_unit c:compress o:other
-P=   : プリンタポートの指定, 0 for BIOS(default), [B]1〜[B]4 for LPT1:〜LPT4:
-hard=: 機種 [0:PC9801 1:DOS/V 2:J3100]
```

上記オプションは、 parameter file <dviprt.par> で指定可能

2.3 機能別オプション一覧

2.1 に、dviout/dviprt のオプション一覧表を示します。

機能	オプションと解説ページ
欧文フォント	-L p20 -gen p47
和文フォント	-k p34 -S p24 -J p25 -vfn p36 p42
ゴシック文字	-G p32
縦書き TEX	-g p126
NTT JTEX における和文フォント	-ntt p25 -nttF p26
代替フォントの指定	-F p20 -A p20 -JC p126
欧文 TrueTypeFont の指定	-ttf p19
フォントのチェック	-c p27 -f p27 -p p29
倍角・拡大/縮小フォントの指定	-K p33 -romf p34 -k p34 -varf p35
dpi と mag	-dpi p60 -DPI p60 -mag p61 -half p61 -e p61 -esize p62
フォントキャッシュのセーブ	-r p21
新サイズオプションの使用/不使用	-newsize p63
用紙のサイズと給紙指定	-PW p66 -PH p66 -y p66 p92
給紙位置	-PF p66
プリンタ固有の印字可能域の指定	-TM p64 -BM p64 -LM p64 -RM p64 -MW p65 -MH p65 -V p56
印字原点	-OX p66 -OY p66
センタリング	-C p70 -HC p67 -VC p67 -HS p67 -VS p67
印字/表示可能な最大幅・高さ	-W p69 -H p69
DVI の原点、左と上のマージン	-X p69 -Y p69
右、下のマージン	-w p69 -h p69 -M p70
イメージ表示	-W p73 -H p73 -P p73 -Q p73 -C p73 -view p74
改ページと袋とじ印字	-T p55 -nf p55
ランドスケープ印字/表示	-V p56 p65
ページの指定	-page p57 -o p57 + p56
画面反転・画面カラー	-R p78
スクロール/出力のスピード調整	-s p71 p94 -sh p71 -t p71 -u p71
プリンタの設定	-p p82 p87 -I p83 -P p87
コピー部数指定	-lc p93
ページ・レジューム機能	-B p58
フォントの回転	-R p126 -RV p27
オプション出力	-comp p86 -base p127 -box p127 -FAX p85
バッファサイズの指定	-br p52 -bf p52 -bb p51 -bv p53 -bw p53
EMS の使用	-EMS p53
出力ファイルの指定	-O p93 -bmp p75
停止、終了	-Z p127 -E p127 -wait p128
tpic specials	-tpic p114
PostScript specials	-GS p117 -GIF p116 -gsx p117 -gbox p118 -gdat p116 -gsize p118
パラメータファイル	-= p11
統合版dviprt での機種指定	-hard p86
PC-9801 以外の機種への補足	-reso p76 p77 -chip p77 -sws p78 -R p80 -E p59 p78 -P p87

表 2.1: オプション一覧

2.4　オプション・パラメータの指定法

オプションには、そのサブパラメータを

1. -dpi, -X, -lc のように数字で指定する
2. -PW, -OX のように長さで指定する
3. -F, -K, -r のように文字列で指定する
4. -L, -g のように ON/OFF を指定する

の 4 種類があります。「長さ」とは、単位を伴った数字のことです。

サブパラメータは、オプションを表わす文字の後に「=」または、「:」をつけて指示します (どちらを用いてもかまいません)。

1. -dpi=300, -X=-10, -lc=5
2. -PW=18.2cm, -OX=100mm
3. -F:cmr10, -K=;48;64;, -r=a:\tmp\dviout.$$$
4. -L=+, -g=-

2.4.1　オプション区切り文字の省略

オプションを表わす文字に続く「=」または「:」は、さらにその後のパラメータが英字でない場合には、-dpi300, -L+ のように省略することができます。

2.4.2　トグルスイッチ

ON, OFF のスイッチになっているオプションは、-L のように「+」,「-」を付けないと、指定する毎に ON, OFF が入れ代わります。このようなスイッチを**トグルスイッチ**と呼びます (「+」,「-」を指定した場合には、「+」で ON,「-」で OFF となります)。以下のオプションの解説にて「+」または「-」を指定するオプション (例えば-L[+|-] のように書かれているもの) は、このトグルスイッチであることを意味します。

パラメータなしでdviout/dviprt を起動すると、デフォルトのトグルスイッチの状態が表示されますが、パラメータのみを指定し (例えば-f)、DVI ファイルを指定しないで起動すると、次項、次々項に示すようにファイルやコマンドラインからパラメータを読み込んだ後の状態が示されます。これで各トグルスイッチの設定状況を把握できます。

2.4.3　dviout.par, dviprt.par

オプションはファイルdviout.par またはdviprt.par に書くこともできます。こういったオプションを記述するファイルをパラメータファイルといい、

1. 環境変数TEXCFG で指定したディレクトリ
2. カレントディレクトリ
3. 環境変数PATH で指定されたディレクトリ

の順で探されます。

また、環境変数TEXPK、TEXKNJ などに設定する内容をこれらのファイルdviout.par/dviprt.par の中に記述することもできます。

これは、例えば

```
TEXPK=c:\font\^d\^s.pk;c:\font\jfms\^s.tfm
```

のように書きます。これらを環境変数とファイルの両方で指定した場合は、環境変数による指定が優先されます。通常の設定はファイルに指定しておき、一時的に設定を変える場合に環境変数を用いると良いでしょう。

パラメータファイルに、記号「#」が書かれていると、その行の「#」から行末までの部分は無視されるので、コメントなどを書いておくことができます。

2.4.4 パラメータファイルの読み込み指定 [オプション-=]

また、-=*file_name* オプションを使うと、前項に示したデフォルトのパラメータファイルの他に、任意の名前のファイルにオプションを記述し、パラメータファイルとして読み込むことができます。このオプション -= は、コマンドライン上で複数回指定することができますし、標準のdviout.par/dviprt.par やコマンドライン上の -= で指定したファイル中から、さらにもう一段階だけ -= でファイルを読み込むことができます (標準のパラメータファイルやコマンドライン上の -= で読み込むファイル内にも複数回の -= を記述することはできますが、ネストの深さは最初のファイルも含めて 2 段階までです)。ファイル内に記述したパラメータと同じ種類のパラメータを、起動時にコマンドライン上で指定した場合には、コマンドライン上のパラメータ (その中ではより後で指定したもの) が有効となります。

-=*file_name* としたとき、

1. ファイル名が絶対パス名なら、そのファイルを使う
2. そうでなければ、まず、カレントディレクトリを探す
3. 見つからなければ、環境変数TEXCFG で指定されたディレクトリを探す

と動作します。なお、このように探しても見つからない場合で、指定したファイル名に拡張子がついていなければ、「.par」を補って再び上記の順に探します。

2.4.5 制限

一般に、1 つのオプション・パラメータにつき 4K bytes までの長さが可能です。

2.4.6　指定法による区別

これらのオプションは、サブパラメータの指定方法によって、以下の 4 つに分けられます (本節の最初を参照)。複数のタイプに属するオプションも存在します。

2.4.6.1　数値指定

数値で指定するもの (8 進数は「0」を、16 進数は「0x」を先頭につけて示す)

-A	§4.2.5 p20	-bb	§5.1.1 p51	-bf	§5.1.3 p52
-br	§5.1.2 p52	-bv	§5.1.5 p53	-bw	§5.1.4 p53
-chip	§9.4.3.1 p77	-comp	§10.1.6 p86	-DPI	§7.2 p60
-dpi	§7.1 p60	-e	§7.4 p61	-E	§9.4.3.4 p78, §13.5.2 p127
-EMS	§5.2 p53	-f	§4.6.2 p27	-GS	§12.2.5 p117
-H	§8.2.2 p69, §9.3.1.2 p73	-h	§8.2.4 p69	-J	§4.4.4 p25
-lc	§10.2.5 p93	-mag	§7.3 p61	-nf	§6.1 p55
-P	§9.3.1.2 p73, §10.1.9 p87	-PF	§8.1.3.5 p66	-Q	§9.3.1.2 p73
-reso	§9.4 p74, §9.4.3.2 p77	-S	§4.4.3 p24	-s	§9.2 p71, §10.4 p94
-sh	§9.2 p71	-t	§9.2 p71	-tpic	§12.1 p114
-ttf	§4.2.3 p19	-u	§9.2 p71	-W	§8.2.1 p69, §9.3.1.2 p73
-w	§8.2.4 p69	-wait	§13.7 p128	-X	§8.2.3 p69
-Y	§8.2.3 p69				

2.4.6.2　長さ指定

長さで指定するもの (cf. 第 8 章 p.64)

-BM	§8.1.3.1 p64	-HS	§8.1.7 p67	-LM	§8.1.3.1 p64
-MH	§8.1.3.2 p65	-MW	§8.1.3.2 p65	-nf	§6.1 p55
-OX	§8.1.4 p66	-OY	§8.1.4 p66	-PH	§8.1.5 p66
-PW	§8.1.5 p66	-RM	§8.1.3.1 p64	-TM	§8.1.3.1 p64
-VS	§8.1.7 p67				

2.4.6.3　文字列指定

文字列で指定するもの (指定方法は、オプションによって異なります)

-B	§6.5 p58	-bmp	§9.4.1.1 p75	-F	§4.2.6 p20
-FAX	§10.1.5 p85	-G	§4.7.5 p32	-gdat	§12.2.3 p116
-gen	§4.12 p47	-gsx	§12.2.4 p117	-hard	§10.1.7 p86
-I	§10.1.2 p83	-JC	§13.1 p126	-K	§4.7.6 p33
-k	§4.7.9 p34	-nttF	§4.5.1.1 p26	-O	§10.3 p93
-o	§6.4.2 p57	-p	§4.6.4 p29, §10.1.1 p82	-R	§9.5 p78
-r	§4.3 p21	-romf	§4.7.8 p34	-y	§8.1.5 p66, §10.2.4 p92
-vfn	§4.8.1.2 p36				

2.4.6.4 トグルスイッチ

ON/OFF を「+」,「-」で指定するもの

-B	§6.5 p58	-base	§13.4.1 p127	-box	§13.4.2 p127
-C	§8.2.6 p70, §9.3.1.2 p73	-c	§4.6.1 p27	-esize	§7.4.3 p62
-g	§13.2 p126	-gbox	§12.2.6 p118	-gsize	§12.2.6 p118
-GIF	§12.2.2 p116	-HC	§8.1.6 p67	-half	§7.3 p61
-L	§4.2.4 p20	-M	§8.2.5 p70	-newsize	§8.1.1 p63
-nf	§6.1 p55	-ntt	§4.5.1 p25	-page	§6.4.1 p57
-R	§13.3 p126	-RV	§4.5.2 p27	-sws	§9.4.3.3 p78
-T	§6.2 p55	-V	§6.3 p56, §8.1.3.4 p65	-varf	§4.7.10 p35
-VC	§8.1.6 p67	-vfn	§4.8.1.2 p36, §4.8.2.1 p42	-view	§9.3.4.1 p74
-Z	§13.5.1 p127				

第3章 環境変数と参照ファイル

3.1 環境変数

dviout/dviprt が参照する環境変数は、実行ファイルなどを探すときに一般的に用いられるPATH のほか、次のものがあります。

TEXCFG

dviout/dviprt が参照するパラメータファイルやプリンタ定義ファイルなどのあるディレクトリを指定します (cf. 第 2 章 p.10, 第 10 章 p.83, 第 4 章 p.36)。

TEXPK

dviout/dviprt が使うフォントのファイルを特別な意味を持つパラメータ^g, ^s, ^d, ^l, ^f を用いて指定します。またここで、^ の代わりに % を使ってもかまいません。詳しくは、第 4 章 p.16 以降を参照してください。

TEXKNJ

dviout/dviprt が使う和文フォントのビットマップフォントを指定します (cf. 第 4 章 p.31)。

TEXPKD

PKD ファイルで指定された PK ファイルのあるディレクトリを指定します (cf. 第 4 章 p.18)。

TEXFLI

emTEX の FLI ファイル名の可変部分を指定します (cf. 第 4 章 p.44)。

TEXFONTS

欧文 TrueType Font を使うときに、TFM ファイルを置いているディレクトリを指定します (cf. 第 4 章 p.19)。通常、TEX システムもこの環境変数を同様の意味に用いますので、共通に使えます。

TEXVFD

NTT jTEX の和文フォントの指定に関連しますが、標準機能では使用しません (cf. 第 4 章 p.27)。

上記の環境変数のうちTEXCFG, TEXFONTS 以外は、環境変数として設定する代わりに、パラメータを記述しておくdviout.par/dviprt.par などのファイルに直接TEXPK=... などと書いておくことができます。両方で指定した場合には、環境変数の方が優先されます。

3.2 参照ファイル

dviout/dviprt が参照するファイルで基本的なものは、拡張子を標準的に .par としているオプションを記述したパラメータファイル、拡張子が .vfn となっている「書体倶楽部」/「JG Font」/和文の「TrueType Font」を用いるときに参照するベクトルフォント定義ファイル、拡張子が .cfg となっているプリンタ定義ファイル、dviout のページ指定をファイルから行なうためのページ指定ファイル、オプション-gen による不足フォントの自動生成に用いるテンプレートファイル、和文の TrueType Font を使用するときに使用する TrueType インデクスファイルがあります。

dviout.par, dviprt.par

dviout やdviprt のオプション・パラメータを設定するファイル。通常、TEXCFG でこれらの存在するディレクトリ名を指定します (cf. 第 2 章 p.10)。

dviout.vfn, dviprt.vfn

dviout やdviprt で「書体倶楽部」/「JG Font」/和文の「TrueType Font」を使用する場合に各種設定を行ないます (cf. 第 4 章 p.36)。拡張子は .vfn で固定ですが、ベース名は-vfn= で変更できます。

????.cfg, prtctl.cfg

dviprt で、標準的にサポートしているプリンタ以外のプリンタの場合に指定するプリンタ定義ファイルです (例えば、AP900.cfg など)。-p=o????.cfg のように、dviprt のパラメータで指定します (拡張子 .cfg は省略でき、さらに-p=o とだけ書いた場合には、-p=oprtctl.cfg と指定したのと同じ意味になります)。-p= で、ディレクトリ名をつけずにファイル名を指定した場合は、まずTEXCFG で指定されているディレクトリから、そのファイルが探されます (cf. 第 10 章 p.83)。

ページ指定ファイル

印刷するページ番号などを制御するためにdviprt のみが参照するファイルです。特に標準的な名前はなく、通常コマンドライン上でページ指定を書く代わりに「@」に続けてファイル名を指定します (cf. 第 6 章 p.56)。

テンプレート ファイル

オプション-gen 指定時には不足のフォントが発見されると、自動的に METAFONT を起動して、そのフォントを作成します。このときに参照する必要手順などを書いたファイルがテンプレート ファイルです。特に標準的な名前はなく、-gen= に続けてファイル名を指定します (cf. 第 4 章 p.47)。

TrueType インデクスファイル

dviout/dviprt が和文代替フォントとして TrueType Font を使用するときに参照するファイルで、コードの変換テーブルなどが入っています。拡張子は .tti ですが、ベクトルフォント定義ファイル中に拡張子なしで指定されます (cf. 第 4 章 p.41)。

第4章 フォント

4.1 フォント概観

欧文フォントとしては、最も一般的な PK[1]フォーマットのフォントのほか、PXL1001, 1002, 1003 フォーマットのフォント、TrueType Font をサポートしています。

これらのうち、TrueType Font 以外は gather.exe を使って GTH ファイルというアーカイブ (これは、dviout/dviprt 独自のもの) にまとめることでディスクスペースを節約することができます。

大きなサイズの PK フォントに対しては、PKD ファイルというdviout/dviprt 独自の中間ファイルを作成して間接的にアクセスすることで参照効率をあげることができます。

dviout/dviprtがサポートするアーカイブの形式としては、上にあげた GTH ファイルの他に、アスキー日本語 MicroTEX の FAR ファイルや、emTEX[2]の FLI ファイルがあります。

一方、和文フォントとしては、アスキー版の日本語 TEX で使われている JXL4 フォーマットの和文フォント (通常min10.pxl などというファイル名) をサポートしており、PK フォントなどと全く同様に使用できます。ただし、JXL4 フォーマットのフォントは、GTH、FAR、FLI ファイルの中に入れて使うことはできません。

この JXL4 フォーマットのフォントは、アスキー日本語 TEX で日本語を使用する上で、必要となるフォントですが、一般には入手が困難であるためdviout/dviprt では、ビットマップ形式のフォントや和文の TrueType Font など多くのフォントを代替として使用することができます。

NTT 版の JTEX で使用される和文フォントは、アスキー版の日本語 TEX の和文フォントのように拡張したフォーマットを用いていません。通常の欧文フォントと同じ形式のファイルを複数用いて、1 つの和文フォントを表わす方式を採っています。従って、dviout/dviprt に対する指定の仕方や扱いも欧文フォントとまったく同じです。(ファイル容量は大きくなりますので、解像度が高いものや、多くのフォントを用いるときは、PKD ファイルを使用すると効果的でしょう)。

また、オプションの指定により NTT JTEX で使用する和文フォントをアスキー日本語 TEX で使用するフォントにマッピングを行ない、代替フォントによる表示や印刷も可能となっています。

4.2 使用フォントの指定

欧文の PK フォント、欧文の TrueType Font、PXL の 1001、1002、1003、JXL4 フォントを使用するためには、そのパス名を含むファイル名のテンプレート (フォント指定と呼びます) を、変数TEXPK に設定します (ここでいう欧文フォントには NTT JTEX 用和文フォントも含まれます)。いくつかのフォーマットのフォントを混在可能することも可能です。TEXPK は環境変数に指定するか、パラメータファイルに記述します。

4.2.1 変数TEXPK の設定

4.1 p.17の記法を用いたフォント指定でフォントのファイル名を、(環境) 変数TEXPK に指定します。dviout/dviprt は、DVI ファイルに書かれたフォント名や解像度などの情報をもとに、このフォント指定にマッチするフォントを探して用いることになります。

[1]Packed
[2]ドイツの Eberhard Matters 氏が MS-DOS, OS/2 に移植した TEX

^g	アーカイブファイル名 (cf. 本章 p.43)
^s	フォント名
^d	解像度の dpi 値
^l	解像度の dpi 値の 5 倍
^f	FLI ファイル名の可変部 (cf. 本章 p.44)

表 4.1: 変数TEXPK で用いることができる記法

フォントの指定を「;」で区切って並べることで、複数の場所に入れてあるフォントを使用することも可能です。前に指定したものから順に調べて、最初にマッチしたものが使われます。また、「^」の代わりに、「%」を使うこともできます。さらに、ディレクトリの区切りの記号「\」を並べて「\\」のようにすると、kpathsea ライブラリ風に再帰的にフォントが検索されます。

例 :

c:\font\^d\ というディレクトリに PK フォントを置いたなら、

```
set TEXPK=c:\font\^d\^s.pk
```

のように指定します。このとき、cmr10 というフォント名で、180dpi の解像度をもつデザインサイズ (10pt) どおりのフォントならば、テンプレートに従ってc:\font\180\cmr10.pk というフォントが検索されることになります。

これは、

```
set TEXPK=c:\font\%d\%s.pk
```

と書くことも可能ですが、バッチファイルの中では、

```
set TEXPK=c:\font\%%d\%%s.pk
```

のように「%」を重ねて指定する必要がありますのでご注意ください。

c:\font\^d\ の他に、c:\font\^dpk\ というディレクトリにも同様のファイル名でフォントを入れていて、この順に検索したいときは

```
set TEXPK=c:\font\^d\^s.pk;c:\font\^dpk\^s.pk
```

とします。

```
set TEXPK=c:\ptex\texmf\fonts\\^dpk\^s.pk
```

とすると、「\\」の部分を存在するディレクトリの列に置き換えながらフォントを探していきます。ただし、置き換えるディレクトリの列は最長 3 つまで（ams\euler\pk など）で、一つの指定に「\\」は一個所だけ許されます。この機能を多用すると、フォンと検索の速度が落ちることがあるので、よく使うフォントは TEXPK の先頭近くに「\\」を含めずに指定しておくとよいでしょう。

ここでは、PK フォントについて示しましたが、他のフォント形式についてもまったく同様です (ただし、欧文 TrueType Font については、この後の記述も参照してください)。

4.2.1.1 パラメータファイルでの指定

環境変数の代わりに、パラメータファイル (デフォルトでは、dviout.par またはdviprt.par) に、他に必要な指定と並べて

```
TEXPK=c:\font\^d\^s.pk
```

のように書いておくことができます。

パラメータファイルと環境変数の両方で指定した場合には、環境変数のみが解釈されます。通常はパラメータファイルに記述したものを使用し、一時的に別のフォントを使いたいときに環境変数を用いるとよいでしょう。

4.2.2 PKD ファイル

64K byte 以上のサイズの大きな PK ファイルを使う場合や、PK ファイルを多数使う DVI ファイル (例えば、NTT 版の jTEX でタイプセットしたもの) を扱うと、フォントファイルが Font file buffer にロードできなかったり、flush の頻度が高くなったりするため、効率と速度が低下する可能性も高くなります。このようなときは、pktopkd.exe を使って、PKD ファイルを作成して、それを仲介して PK ファイルをアクセスすることで問題を回避することができます。

この PKD ファイルも、GTH ファイル (cf. 本章 p.43) の中に入れておくことができます。

pktopkd.exe で変換可能な PK ファイルは、その文字コードが 1 byte で表わされるものからなり、かつ、ファイルサイズが 8M byte 以下のものです。作成される PKD ファイルのサイズは、900 byte 以下と、小さくなっています。

4.2.2.1 pktopkd.exe の実行と環境変数TEXPKD

```
c:\font\360\cmr10.pk
```

というフォントファイルから PKD ファイルを作成するには

```
cd c:\font
pktopkd c:360\cmr10.pk
```

のようにします。このとき、cmr10.pkd というファイルがカレントディレクトリに作成され、そのファイルの中にpktopkd.exe のパラメータで指定したファイル名からドライブ名c: を削除した360\cmr10.pk が記録されます。

PKD ファイルを仲介とした PK ファイルを使うには、(環境) 変数TEXPK に PKD ファイルを (追加) 指定すると同時に、実際の PK ファイル (PKD ファイルではありません) の場所を示すための (環境) 変数TEXPKD を設定しなくてはなりません。

例えば、c:\font\pk360.gth というアーカイブファイルのなかにこのcmr10.pkd を入れた場合は、

```
TEXPK=c:\font\pk^d^g^s.pkd
```

というように (環境) 変数TEXPK に (追加) 指定します (cmr10.pkd のファイル名の拡張子などを変更しても構いませんが、その場合にはその変更したファイル名をTEXPK に指定してください)。

実際のファイルは、環境変数TEXPKD とcmr10.pkd に記録されたパス名をつなげたものがフルパス名と解釈されます。今の例では、

```
set TEXPKD=c:\font
```

とすれば、cmr10.pkd を仲介として、c:\font\360\cmr10.pk がアクセスされます。

```
cd c:\font
set TEXPKD=c:
```

としても同様にアクセスできますが、c: ドライブのカレントディレクトリを変更するとアクセスできなくなりますから、必ず絶対パスで指定しなくてはなりません。また、このTEXPKD もdviout.par やdviprt.par などのパラメータファイルに書くことができます。

360*.pk というワイルドカードで指定される全てを PKD ファイルに変換するには、以下のようにコマンドを実行します (PKD ファイルのフォーマットについては、本章 p.45 を参照してください)。

```
for %%1 in (360\*.pk) do pktopkd %%1
```

4.2.3 欧文 TrueType Font [オプション-ttf]

通常の PK フォントファイルなどと同様 (拡張子が .ttf となっている) TrueType Font の在処をTEXPK に記述し、オプション

-ttf=*dot*

を指定することにより、CM-TrueType Font などの欧文の TrueType Font が使用できます。

TEXPK には従来の PK フォントなども同時に記述できます。

```
TEXPK=a:\windows\system\^s.ttf;c:\font\^d\^s.pk
```

のように TrueType Font を先に記述すると良いでしょう (TrueType Font が存在するものはそれを用い、ない場合には PK フォントを探します)。

この-ttf= において *dot* に正の数値が指定されたとき、TEXPK のフォント指定のうち拡張子が .ttf であるものに DVI ファイルから得られるフォントがマッチすると、欧文 TrueType Font が使用されます。

dot はフォント描画モードを決めるパラメータでdviout.vfn/dviprt.vfn のDraw sw (cf. 本章 p.39) の数値に相当します。例えば-ttf=40 とすると、40 ドット以下の文字は「精細モード」、41 ドット以上のフォントは「標準モード」によってラスタライズされます。ただし *dot* と比較するのは、個々の文字の大きさではなく、そのフォント固有のデザインサイズをドット数に換算した値です。この値は-f=1 オプション (cf. 本章 p.27) によって確認できます。

なお-ttf=0 (デフォルト) とすると欧文 TrueType Font の使用を禁止します。この場合TEXPK の記述にマッチする*.ttf が存在しても単に無視されます。

欧文の TrueType Font (foo.ttf) を使う場合、水平方向の metric (文字送り幅) を得るために環境変数TEXFONTS から対応する TFM ファイル (foo.tfm) を探します。この環境変数TEXFONTS は、TEX の処理系がタイプセットを行なうときに TFM ファイルを読むときの検索パスとして標準的に参照するもので、例えば

```
set TEXFONTS=c:\font\tfm;c:\font\jfms
```

などと設定されるものです。

この例では、cmr10.tfm などの TFM ファイルをc:\font\tfm に入れて、min10.tfm などの JFM ファイルをc:\font\jfms に入れているものとしています。拡張子が .tfm のファイルを 2 つのディレクトリに分けて入れてある理由については、和文代替フォントの項を参照してください (cf. 本章 p.23)。

なお、ディレクトリの区切りとして、\の代わりに\\を用いると kpathsea 風に再帰的に TFM ファイルが検索されます。たとえば

```
set TEXFONTS=c:\ptex\texmf\fonts\\tfm
```

とすると、\\の部分は、存在するディレクトリの列に置き換えながら TFM ファイルを探していきます。ただし、置き換えるディレクトリの列は最長 3 つまでで、一つの指定につき一個所のみ許されます。\\の代わりに、//も使うことができます。

該当する TFM ファイルが見つからないと警告を発して`foo.ttf` の metrics 情報のみを使って出力を続けますが、この場合、TrueType Font の metrics 情報の精度の問題から文字の位置などがずれてしまうことがあります。

4.2.4 長いフォント名の中央省略 [オプション-L]

DVI ファイルから得られるフォント名が 8 文字以上の長いものであるときは、デフォルトでは先頭の 8 文字を取って ^s に代入しますが、

```
-L[+|-]
```

のオプションが ON になっているとき (すなわち、-L+ のとき) は、先頭と末尾の 4 文字づつを取って、8 文字にして ^s に代入します。-L はトグルスイッチです。

このオプションは、LaTeX の picture 環境で用いられるフォントや NTT JTeX の和文フォントのように長いフォント名のファイルを MS-DOS のようなファイル名に長さの制限のある OS 上で用いるときに必要となります。ただし、後述の FLI ライブラリなどを用いて、長いフォント名をそのまま使用できる場合には、-L- のままとしておいてください。-L+ とすると逆にフォントのファイルが探せなくなります。

4.2.5 フォントのサーチ範囲の拡大 [オプション-A]

```
-A=number
```

のようにオプション-A の後に数字 *number* を指定することで、(環境) 変数TEXPK の ^d,^l に代入される数字の範囲を変更することができます。具体的には、-A で指定された値に 3 を加えた種類の値がフォントの検索に使用されることになります。例えば、デフォルトの状態 -A=0 では、各フォントに対し計算によって得られる値と、その上下 1 ずつずれた値の 3 種類の値が、^d や^l に代入されて探されることになります。一方、-A=3 とすると、3 + 3 で 6 つ (計算された値とその上に 3 つ、下に 2 つ) の値で、計算された値に近いものから順に検索が行なわれます。

4.2.6 代替フォントの指定 [オプション-F]

必要なフォントが無いときには、代替として使用するフォントをオプション-F で指定することができます。指定の仕方は一般的には次の通りです。

```
-F=sfn1.sres1=ofn1.ores1;sfn2.sres2=ofn2.ores2...
```

sfn_i, ofn_i フォント名、(環境) 変数TEXPK の ^s に対応
$sres_i$, $ores_i$ 解像度を表わす数字、TEXPK の ^d または ^l に対応

多くの代替フォントがある場合など、コマンドラインに書けないときは、パラメータファイルで指定することもできます。

4.2.6.1 代替フォント選択処理の動作

必要とされる見つからないフォントが $ofn_i.ores_i$ にマッチすれば、それを $sfn_i.sres_i$ に置き換えて ($i = 1, 2, \ldots$ の順に、i が等しい時は、(環境) 変数TEXPK で指定された順に) 探していきます。
より詳しく言えば次のようになります。いま、^s, ^d, ^l に代入されるものを、S, D, L と書くと、

> ofn_i が S に等しい場合 (ただし、ofn_i が空の時は等しいとみなす)、さらに $ores_i$ が D か L に等しい ($ores_i$ が空の時は等しいとみなす)
>
> ⇓
>
> sfn_i が空でなければ、S を sfn_i に置き換えて (空ならば、S のまま)、さらに $sres_i$ が空でなければ、D と L 共に $sres_i$ に置き換えて (空ならば、そのままで) 探します。

各項は、「. ; =」を区切り記号として、4 つのデータからなっていますが、途中で切れていれば、その後は空であると解釈されます。以下に例をいくつかあげておきましょう。

例 :

-F=cmr10

> -F=cmr10.=.; と同じで、見つからないフォントがあれば、フォント名を、cmr10 に置き換えて探します。

-F=.320

> -F=.320=.; と同じで、見つからないフォントがあれば、^d, ^l を 320 に置き換えて探します。

-F=.320=.330;cmr10.340=cmr12.350;.340=.350

> 見つからないフォントがあって、その ^d または ^l が 330 なら、それを 320 に置き換えて探し、それで見つからないフォントで、^d または ^l が 350 で、^s がcmr12 のものがあれば、^d, ^l を 340 に、^s をcmr10 に置き換えて探し、さらに、それでも見つからない場合に、^d または ^l が 350 ならば、それを 340 に置き換えて探します。

-F=cmr10.=.320

> 見つからないフォントで、^d か ^l が 320 のものは、ファイル名をcmr10 に変えて探します。

フォント名には、「. = ;」が含まれていないと仮定します。また、区切り記号「. = ;」には意味上の区別はなく、どれを書いても同じ結果ですが、上で説明したように書いた方が分かりやすいので、そのようにすることを勧めます。

4.3 フォントキャッシュのセーブ [オプション-r]

-r=*save_file_name*

によってフォントキャッシュ[3]をディスク上にセーブし、次回の起動時に読み込んで立ち上げを高速化します。

オプションには -r=a:\tmp\dviout.$$$ のように、-r= の後にセーブ/ロードするファイル名を指定します。

このオプションは、RAM ディスクはあるが、大きなディスクキャッシュを持たないシステムで、ディスク上のフォントファイルを使う場合に特に有効です。

4.3.1　注意事項

4.3.1.1　セーブを行なわない場合

セーブはdviout/dviprt の終了時に行なわれますが、フォントキャッシュがフラッシュされた場合には、セーブを行ないません。また、セーブのときと同じ状態で起動された場合でないと、データのロードを行ないません。ロードを行なったあと、新たにフォントがロードされなかった場合は、終了時のセーブを省略し、新たにセーブを行ないません。

EMS と conventional memory の両者をフォントキャッシュのバッファに用いている場合は、まず conventional memory から使われますが、このときはバッファの使用が EMS にまでおよぶとセーブが行なわれません。

また、EMS のみをフォントキャッシュのバッファとしている場合は、そのバッファの使用が 64K byte 以上になるとやはりセーブを行ないません。

4.3.1.2　キャッシュデータが破壊されている場合

セーブされたフォントキャッシュのデータが壊れていると、起動時にエラーになったり、プログラムがハングアップしてしまうことがあります。このときは、-r オプションをはずすか、あるいは壊れたファイルを消去してください。

4.4　和文代替フォント

アスキー日本語 TEX で標準的に和文フォントとして用いるのは、JXL4 フォーマットのフォントですが、この形式のフォントをさまざまな解像度でそろえるのは事実上困難なため、dviout/dviprt では、システムフォント (PC-9801 では ROM として内蔵される漢字フォント、DOS/V 等では DOS のシステムが使用する和文のフォントを指す)、通常のビットマップ形式のフォント、そのビットマップ形式のフォントを PK フォントや JXL4 フォントと同様の方法で圧縮したフォント (これはdviout/dviprt 独自の形式となります)、少数の文字を扱うのに便利なやはり独自形式の KG 和文フォントを和文代替フォントとして使用できます。また、書体倶楽部/JG Font/和文 TrueType Font や、(dviprt では)LIPS III や ESC/Page 仕様の LBP に内蔵のスケーラブルフォントも代替の和文フォントとして用いることができます。どのフォントを使用する場合にも、最適のサイズや種類 (書体) のものを自動的に選択して使用します。

システムフォントは本章 p.33、ビットマップフォントは本章 p.31 を、書体倶楽部/JG Font は本章 p.35、和文 TrueType Font は本章 p.41、LBP の内蔵フォントについては第 10 章 p.87 を参照してください。

また、dviout/dviprt で使われる、圧縮していない和文ビットマップフォントのフォーマットについては、本章 p.45 を参照してください。

[3]Expanded Font buffer などの使用した文字のビットマップデータをメモリー内に一時保存したもの

これらの和文代替フォントを使用する場合には、漢字フォントの情報を与える JFM ファイル[4](拡張子は、.tfm となっている) が必要です。その JFM ファイルからサイズなどの情報を得た上で、その情報に従って、上記の和文代替フォントを配置します。

また、NTT JTEX では既に述べたように、和文フォントとして欧文フォントと同じ形式のフォントを用いているため、通常はそのためのフォントを用意する必要がありますが、オプション -ntt を用いると、DVI ファイルに記録されたこれらのフォントの名前を、アスキー日本語 TEX で使用する和文フォントの名前にマッピングすることができます。通常dviout/dviprt で使用できる種々の形式の和文フォントが NTT JTEX でも使用できることになります。このオプションを有効にした場合には、NTT JTEX で作成した DVI ファイルの和文フォントであっても以下の記述が適応されます (cf. 本章 p.25)。

4.4.1 和文フォントの選択順序

dviout/dviprt での和文フォントの選択は 4.2 の順序で行なわれます。ただし、書体倶楽部/JG Font/和文 TrueType Font、LBP 内蔵フォントでは、種類 (書体) ごとに、特定のサイズ (ドット数) のフォントに関してそのフォントを使用しないという指定を行なうことができます。

1 JXL4 フォーマットのフォント
2 「書体倶楽部」/「JG Font」/和文「TrueType Font」(-vfn= の指定による)
3 LBP 内蔵フォント (-p= の指定による)
4 システムフォントを拡大/縮小したもの (-romf=, -k= の指定による)
5 ディスク上のビットマップフォント
6 システムフォント

表 4.2: 和文フォントの選択順序

4.4.2 TEXPK への JFM ファイルの登録

アスキー日本語 TEX で、JXL4 フォーマット以外の和文フォント (和文代替フォント) を使用するときには、フォントの metric 情報などを得るため JFM ファイル (拡張子は .tfm となっている) が必要となります。min5.tfm, goth5.tfm, nmin5.tfm, ngoth5.tfm などのファイルがそれで、これらはアスキー日本語 TEX が DVI ファイルを作るときに参照します。これらの JFM ファイルも、(環境) 変数TEXPK で指定することによって、dviout/dviprt から参照できるようになります。JXL4 フォーマットのフォントと和文代替フォントを混用する場合は、JXL4 フォーマットのフォント指定よりも後に、JFM ファイルのフォント指定をおくようにしてください。

dviout/dviprt では、欧文の TFM[5]ファイルと JFM ファイルとの判別を高速に行なうために、**(環境) 変数TEXPK のフォント指定の中に欧文のフォントのための TFM ファイルにマッチするものを書いてはいけない**仕様になっています。このため、もしフォントの検索の過程でフォント指定のいずれかが、欧文の TFM ファイルにマッチしてしまうと、JFM ファイルであるかのように扱ってエラーとなります。ですから、欧文の TFM ファイルとは別のディレクトリにmin5.tfm などの JFM ファイルを移して、それを指定するようにしてください。

[4]欧文の TFM ファイルに相当し、和文代替フォントを使用する場合には、フォントグリフファイルから十分な metric 情報が得られないために必要となる
[5]TEXfont metric

例 :

c:\font\jfms に、min5.tfm などをコピーしておいて

```
set TEXPK=c:\font\^d\^s.pk;c:\font\jfms\^s.tfm
```

などとバッチファイルに書いておきます。このとき、c:\font\jfms にcmr5.tfm のような (JFM ファイルでない) 通常の TFM ファイルがあるとエラーが生じます。

4.4.2.1 JFM ファイルを GTH ファイルにまとめる

gather.exe で JFM ファイルをjfm.gth といった GTH ファイル (cf. 本章 p.43) にまとめることも可能です。例えば、まとめたものをc:\font に入れる場合には

```
set TEXPK=c:\font\^d\^s.pk;c:\font\jfm^g^s.tfm
```

のように指定します。

4.4.3 和文代替フォントの横幅 [オプション-S]

-S=w

のように、オプション-S で数字 w を指定すると、dviout/dviprt は、JFM ファイルで与えられる漢字「亜」の文字送り幅 (2 つの文字を連続して配置する時の reference point の標準の進み幅) の w/1000 倍にできるだけ近いドット幅の和文代替フォントを探そうとします。このとき、-S で指定された値が 1000 以下の時には、JFM ファイルで与えられるフォントの文字送り幅が求める幅に最も近いフォントから探し、1000 以上の時は、-S で決まる文字送り幅以下のドット幅のフォントを順次探して、いずれかのフォントが見つかればそれを用います。見つからないときは、求めるドット数にもっとも近く横幅が 16 ドット (ハイレゾモードでは 24 ドット) 未満のものを探します。それでも見つからなければ、その時使用しているドット幅のシステムフォントを用います。-S は、デフォルトは 952 で、JFM ファイルにあるフォントの標準の横幅の 952/1000 倍のドット幅を意味します。

4.4.3.1 和文代替フォントの位置補正

和文代替フォントは一定の文字送り幅のフォントとして扱われ、JFM ファイルを読んで漢字の「亜」の文字送り幅、あるいはそれに最も近いドットサイズのフォントが選択されますが、JFM ファイルに書かれてある文字送り幅は文字によって一般に異なるので、以下のように文字の位置の補正を行ないます。

JIS コードで

(A) 2121 − 2125, 212b, 212c, 216b, 216c, 216d: 、。, . ゛ ゜ ° ′ ″

(B) 214b, 214d, 214f, 2151, 2153, 2155, 2157, 2159, 215b: ） 〕 ］ ｝ 〉 》 」 』 】

(C) 214a, 214c, 214e, 2150, 2152, 2154, 2156, 2158, 215a: （ 〔 ［ ｛ 〈 《 「 『 【

(D) 上記以外

と分けて、(C) の場合は reference point を

|代替フォントの文字の文字送り幅| − |JFM ファイルから読んだ文字送り幅|

だけ、(D) の場合はその半分だけ左に移動して配置しています。(A) と (B) の場合は、reference point を補正しません。

なお、縦書きの場合にも、上記の「左」を「上」と読み変えて同様に処理を行ないます。

4.4.4 和文代替フォントのベースライン位置 [オプション-J]

```
-J=u
```

のように-Jで数字 u を指定すると、JFM ファイルから計算した値よりもフォントの高さの $u/1000$ 倍だけベースラインの下に来るように位置を調整します。デフォルトの値は 0 です。

4.5 NTT jTEX における和文フォント

NTT jTEX では、和文フォントとして欧文のフォントと同じ形式のフォントファイルを使用します。一般に和文フォントでは、欧文フォントと比べてより多くの文字を用意する必要があるため、サブフォントと呼ばれる 33 のフォントファイル群で 1 つの和文フォントを表現します。

dviout/dviprt では、NTT jTEX, JaWaTEX, ezjTEX で使用するこのような形式の和文フォントは、欧文フォントと同じ方法で利用することができます。さらに、アスキー日本語 TEX が使用する和文フォントの形式にマッピングを行ない、アスキー日本語 TEX で使用できる種々のフォントを代替フォントとして使うこともできます。これには、2 つのやり方があります。

4.5.1 アスキー日本語 TEX 和文フォントで置き換える方法 [オプション-ntt]

最初の方法は

```
-ntt[+|-]
```

を ON にするものです。この指定によって、dviout/dviprt がサポートするアスキー日本語 TEX 用の和文フォントを、NTT jTEX でタイプセットされた DVI ファイルからも利用できるようになります。これは、デフォルトで組み込まれている機能です[6]。

この場合、通常のアスキー日本語 TEX の場合のように JFM ファイル min10.tfm, goth10.tfm を欧文の TFM ファイルと別の所において、それをTEXPK の ^s.tfm で指定します。

また、NTT jTEX や ezjTEX ではフォント名が 8 文字を越えるため、-L+ (長いフォント名の中央省略) オプションが必須となります。dviout.par/dviprt.par に-L+ オプションを入れておくことをお勧めします (cf. 本章 p.20)。

NTT jTEX の和文フォントでは、明朝に dmj*、ゴシックに dgj* というファイル名が、JaWaTEX では、明朝に mj*、ゴシックに bj* が使われています。

オプション-ntt+ を指定すると、dviout/dviprt はdmj* あるいは mj* という名前のフォントをアスキー日本語 TEX のmin10 というフォントに、dgj* または bj* を、アスキー日本語 TEX のgoth10 というフォントに置き換えた上で、アスキー日本語 TEX を使用している場合と同じようにTEXPK や後述のTEXKNJ(cf. 本章 p.31) を参照して、使用する和文フォントを決定します。従って、dmj* あるいは mj* のようなフォントを用意する必要はなく、アスキー日本語 TEX を使用している場合と同様に、JXL4 フォントやシステムフォ

[6]以下の -ntt+ の機能の説明は、八雲氏のnttrpl.doc より引用

ント、knjtopk.exe で圧縮されたフォント、書体倶楽部/JG Font/TrueType Font、LBP 内蔵のスケーラブルフォントなどの和文フォントを使うことができます。

例 :

```
set TEXPK=c:\font\^d\^s.pk;c:\font\jfms\^s.tfm
```

としてあると、-ntt+ のもとでdmj*10 というフォントが探されると、上記のTEXPK の ^s には、min10 が代入されることになります。

4.5.1.1 和文フォントの追加定義 [オプション-nttF]

NTT JTEX において \jfont によって、新たに和文フォントを追加定義した場合には、追加した和文フォント名をdviout/dviprt にも教える必要があります。これには、以下のようにオプション-nttF を用います。

```
-nttF=<jfm_file_1>=<追加_font_1>;<jfm_file_2>=<追加_font_2>;...
```

このオプションの場合、ポイント数 (10 や 12 など) や解像度 (118 や 360 など) を含まない名前を指定することに注意してください。

例えば、LATEX 文書中に

```
\jfont\xmouhitu=mou10
```

のようにしてmou10 という和文フォント名を追加定義し、書体倶楽部フォントの毛筆体 (mouhitu.vf1, mouhitu.vf2) を使った出力を得る場合を考えます。(\jfont の説明は省略します。詳しく知りたい方は、『LATEX 自由自在』磯崎秀樹著サイエンス社刊の p.214 等をご覧ください。)

必要な手順は次のようになります。

1. TFM ファイルを用意する。
   用意する TFM ファイルは、GO32 版 NTT JTEX の場合 *jsy*.tfm と *jka*.tfm、(mou10 の場合は、moujsy10.tfm とmoujka10.tfm) です。
   moujsy10.tfm はdmjsy10.tfm をコピーして、moujka10.tfm はdmjka10.tfm をコピーして作ってください。
2. LATEX 文書をタイプセットする。
3. 代替用の JFM ファイル (ここでは仮にztmoumaa.tfm とします) を用意する。
   min10.tfm をコピーしてztmoumaa.tfm を作ってください。
4. dviout.vfn/dviprt.vfn に、フォント名mouhitu と作成した JFM ファイルztmoumaa.tfm を登録します。
   登録方法はアスキー日本語 TEX の場合と同じです。詳しくは本章 p.35 をお読みください。
5. dviout/dviprt を、-ntt+ および-nttF オプションを使って起動します。例えば、
   ```
   dviprt -ntt+ -nttF=ztmoumaa=mou -vfn+ foo.dvi
   ```
   のように起動します。
   JFM ファイル名と追加和文フォント名の間は「=」で区切ります。また、「;」で区切ることによって複数の置き換えを指定できます (「.」も区切り記号として使えますが、混乱を避けるため使わない方が良いでしょう)。

4.5.2 VFD ファイルによる方法

もう一つの方法は、NTT jTEX, JaWaTEX, ezjTEX の和文フォントとして書体倶楽部フォントを使用できるようにするものです。

これは、標準で提供されるdviout/dviprt には組み込まれていませんので、-DVFD を指定してコンパイルし直す必要があります (先の方法とこの方法の両方を同時に組み込むこともできます)。コンパイルについては、ソースアーカイブ中のreadmesr.doc をご覧ください。

この方法で書体倶楽部フォントを使用する場合には、vfntovfd.exe を使って書体倶楽部フォントからVFD ファイルを作成しておく必要があります。

まず、書体倶楽部形式のフォントがあるディレクトリを、環境変数TEXVFD に設定します (この変数もTEXPK などと同様に、パラメータファイル内に記述することができます)。さらに VFD ファイルを、

```
TEXPK=c:\font\dpi^d^g^s.pk;b:\vector\^s.vfd
```

のように環境変数や、パラメータファイルの中で指定することにより NTT jTEX で書体倶楽部フォントが使用可能になります。

このとき書体を 90 度回転するオプション-RV が使えます。

また、アスキー日本語 TEX の場合と同様、-vfn オプションによりdviout.vfn/dviprt.vfn を用いたベクトルフォントの調整も可能です。

詳しくは、八雲氏のnttvfn05.lzh というファイルに含まれるnttvfn.doc や、吉田氏のvfnb03.lzh というファイルに含まれるvfnpatch.doc を別途入手してご覧ください。

4.6 フォントのチェック

4.6.1 フォントの整合性チェック [オプション-c]

オプション

```
-c[+|-]
```

を ON にすると、DVI ファイルに書かれているフォントと、実際にロードしたフォントとの整合性を、チェックサムのデータを用いて調べます。不整合があると

```
Warning: check sum doesn't match in Font 35: dvi(37CDA321) font(37CDA721)
```

のように表示されます。Font の後の数字は、DVI ファイルの中でのフォント番号です。フォントキャッシュが効いていると、フォントをファイルからロードしない可能性がありますから、dviout.par などでフォントキャッシュを指定している場合に整合性のチェックをするためには、コマンドラインで、パラメータなしの-r オプションを指定してください。このエラーは、TEX が使用した TFM ファイルと、ロードした実際のフォントのデータとがマッチしていないことを意味します。その 2 つのファイルのバージョンが異なっているとき発生するエラーです。

4.6.2 フォント情報の表示 [オプション-f]

dviout/dviprt にオプション-f を指定することで、DVI ファイルに含まれるフォントの情報から実際に使用するフォントの情報に変換し、それを表示します。

4.6.2.1 ファイルへの出力

ここで表示される情報は標準出力になされますので、例えば

```
A>dviout -f=1 foo.dvi  > foo.dat
```

とすれば、ファイルfoo.dat に書き込まれます。

4.6.2.2 使用フォントの表示

```
-f=1
```

を指定すると、dviout/dviprt の実行が終了したとき、使用されたフォントファイル名が表示されます。

```
70: c:\font\jfm.gth goth10.tfm c:\font\jfont\kanji32.pk(35x37)G PK/H
68: c:\font\pk216.gth eufm10.216  PK
67: c:\font\850.far cmbx10  PXL1003
49: c:\font\1200\min10.pxl + JXL
41: c:\font\jfm.gth goth10.tfm c:\font\jfont\kanji14.pk(29x28)2G PK
40: c:\font\jfm.gth min10.tfm (16x15) ROM
31: c:\font\pk180.gth cmr10.180  ?
 0: c:\font\cmr10.pkd + PKD
```

最初の数字はフォント番号。次が、フォントファイル名で、GTH ファイルか、FAR ファイルの場合には、その中のフォントファイル名が続けて表示されます。それが和文フォントの JFM ファイルであれば、さらに続けて、実際の和文フォントデータのファイルが示されます。

和文フォントのときの「( )」内の数字は、dviout/dviprt が最適と判断した文字サイズの横と縦のドット数で、その横のドット数に近い和文フォントを探したことを意味し、その後に 「2」が続くときには倍角にしたこと (cf. 本章 p.33) を、「G」が続くときにはゴシックを合成したこと (cf. 本章 p.32) を意味します。

最後にフォントの形式が記号で示されます。フォントの形式が PKD、VFD、JXL、欧文 TrueType Font の 4 つの場合には、フォント形式の前に 「+」または 「-」が付きます。「+」は、そのフォントファイルが、dviout/dviprt 終了時にオープンされていたことを、「-」は逆にクローズされたことを表わします。フォント形式が 「?」となっているものは、dviout/dviprt が処理した部分にそのフォントが不要だったか、あるいはフォントキャッシュを使ったため、新たにそのフォントをロードする必要がなかったことを意味します。

4.6.2.3 検索フォントと状況の表示

サブパラメータなしで

```
-f
```

と指定した場合は、ロードすべきフォントと、初期化後のdviout/dviprt の状況のみが表示され終了します。ロードすべきフォントの表示は、

```
91 cmssbx10.293(1467): c:\font\pk294.gth cmbx10.294
90   cminch.118( 590): c:\font\590.far cminch
```

```
70   goth10.170( 849): c:\font\jfm.gth goth10.tfm
67   cmbx10.170( 849): c:\font\850.far cmbx10
63   goth10.141( 707): c:\font\jfm.gth goth10.tfm
40    min10.118( 590): c:\font\jfm.gth min10.tfm
```

というような形式です。最初の数字は、DVI ファイルの中で使われているフォント番号、次が ^s.^d(^l) を表わしています。「:」の後は、ディスク上に見つかったフォントファイルで、GTH ファイル、あるいは FAR ファイルのときは、その中の対応するファイル (欧文なら PK、PXL1001, PXL1002, PXL1003, PKD ファイル、和文代替フォントなら JFM ファイル) が続けて表示されます。この例では、91 番のフォントに対して代替フォントが使われることが分かります。

4.6.2.4 オプション-f=2, -f=3

一方、-f=2 とすると実行後に flush の回数だけを表示し、-f=3 とすると情報が出力されなくなります。

4.6.3 chkfont.exe による情報表示

chkfont.exe を用いることによって、DVI ファイルや、GTH ファイル、TFM ファイル、あるいはフォントファイル内のフォントの情報を得ることができます。詳しくは、chkfont.doc をご覧ください。

4.6.4 外字作成のヒント [オプション-p のサブパラメータg]

dviprt のオプション-p は使用するプリンタを指定するものですが、-p=l や-p=m でさらにサブパラメータg を指定すると、印字を行なわずに使用される和文フォントの情報が得られます。

```
-p=l;g
```

この指定をすると、-p=l や-p=m のもつ LBP 出力の意味が失われるので、LBP への出力を想定していない場合にも使うことができます。

出力先は

1. -O を指定しなかった場合
   すべての使われている和文フォントの文字情報を標準出力に出力します。
2. -O[=*file*] と指定した場合
   gaiji.1 に対応する和文フォントの文字情報を標準出力、または *file* に出力します。

gaiji.1 という付属のフォントファイルは、TEXKNJ (cf. 本章 p.31) の最後に指定しておきます。このパラメータは、この目的以外に使わないので-= によるファルで追加指定するとよいでしょう。たとえば以下をそのファイルに書いておきます。

```
TEXKNJ=c:\font\jfont\^s^d.pk;c:\font\jfont\kanji^d.pk;c:\font\gaiji.1
```

サブパラメータg を指定した場合は、-p=l や-p=m の LBP 対応は意味を失い (ダミーです) プリンタへの出力は行なわれませんが、fmin= などの指定は、DVI で使われているフォントに対し、どのようなフォントファイルを対応させるかと関連しているので意味があります。

どのようなフォントにgaiji.1 が選択されることになるかにご注意ください (cf. 本章 p.23, 本章 p.24, 第 10 章 p.87)。

出力例は

```
# c:\font\jfm.gth min10.tfm c:\font\gaiji.1
#  40:0:39x37(32x32)
2122    2
2126    2
213C   10
....
# c:\font\jfm.gth goth10.tfm Gothic-Medium.J83
#  41:3:39x37(32x32)
2126    2
213C    5
213F    1
....
```

「#」で始まる 2 行がフォントに関する情報で、1 行目は-f=1 のパラメータで表示されるものに対応しています (フォントファイル、LBP のフォント etc.)。2 行目は

```
# <font_code>:<font_type>:<width>x<height>(<w_width>x<w_width>)
```

となっていて、最初の<font_code> は、DVI ファイルでのフォント番号 (-f=1 で表示されるものと同じ) です。<width>x<height> は、固定サイズフォントとして最適のドットサイズとdviprt が判断したサイズで、最後の<w_width>x<w_height> は、それに対応してTEXKNJ などに基づいて検索し (実際にdviprt が) 使うはずのファイルのフォントサイズです。

<font_type> の値は

- 0: KG 和文フォント (gaiji.1 もこの形式)
- 1: Huge format PK 和文フォント
- 2: 上記でない PK 和文フォント
- 3: 圧縮していない和文ビットマップフォント
- 4: LBP 内蔵のスケーラブルフォント
- 5: 和文 TrueType フォント
- 6: JG Font
- 7: 書体倶楽部フォント
- 8: システムフォント
- 9: 欧文 TrueType フォント
- 10: JXL4 フォント

となっています (将来変更の可能性があるが、0, 1, 2, 3 などは変更されないであろう)。

「#」のない行は、

```
<code> <number>
```

となっていて、<code> は、そのフォントで使われた文字の JIS コードを 4 桁の 16 進数で表示したもの、<number> は、出現回数 (ただし、256 以上の場合は 256 と表示) です。<font_code> の小さなフォントから<code> の小さい順に出力されます。

また、文字の種類が多くて (3000 程度まで問題なし) すべて出力できなかった時は、最初に

```
## Over
```

という行が出力されます。

なお、gaiji.1 は、1 dot×1 dot のサイズの文字で収録文字数 0 個の KG 和文フォントファイルです。

DVI ファイルで使用されている和文フォントの情報を上記オプションを使って出力し、そのあと必要な文字データを作成してからdviout/dviprt を再度起動する、という 2 パス形式での実行がこのオプションにより可能になっています。特殊なサイズの文字や装飾のついた文字を含む DVI ファイルを処理するのに便利です。-p=lg や-p=mg による出力データを読み込んで KG フォーマットのフォントを作成するプログラムの元になるソース mkgaiji.c が提供されています。

```
                <<< MaKe GAIJI for dviout/dviprt v2.39.1- >>>
                        Ver 0.1, 1993 written by SHIMA

Usage: mkgaiji [-m] [-d=<dir>] [-f=<par>] [[<] file]

       -m      : merge FONT
       -d=<dir>: directory where the FONT are made
       -f=<font>[.<dot>]=<type>;<flag>;<num>;<num>;...;<font>[.<dot>]=...

        <font> : substring of TFM name (min, min8, goth,...)
        <dot>  : width indicated by dviout/dviprt (<dot>-<dot> is valid)
        <type> : - or an integer (Skip or Mincho, Gothic, Kaisho, Gyousho,...)
        <flag> : a sequence of 0 or 1   (Outline, Shadow, Italic,...)
        <num>  : a certain number       (any parameters, dot,...)

Ex.    mkgaiji -d=b:\font -f=goth=1;min.50=2;01001;21;18;min=-
```

dviprt との組み合わせ例 :

file.dvi で使われているgoth* フォントで、横幅が 60 dot 以上のものは、mkgaiji.exe によってタイプ 4 のフォントを 01 という attribute で必要な文字のみ作成して、フォントファイルc:\font\goth<dot>.kg に収めて (既に存在していれば、存在しない必要な文字のみ作成して merge する)、印字する。

```
A>dviprt -=gaiji.par -O file| mkgaiji -m -d=c:\font -f=goth.60-=4;01;goth=-
A>dviprt file
```

gaiji.par の中身は

```
-p=l;g;goth=80;0
TEXPK=c:\font\gaiji.1
```

4.7 システムフォントと和文ビットマップフォント

dviout/dviprt では、JXL4 フォントが存在しない場合に、システムフォント (PC-9801 の ROM フォントや DOS/V の V-Text ドライバが使用する和文フォント) やビットマップのフォントを和文代替フォントとして使用できます。

4.7.1 和文ビットマップフォントの指定 (環境変数TEXKNJ)

ディスク上の和文ビットマップフォントを利用するには、(環境) 変数TEXKNJ でそのフォントを指定します。ファイル名には各フォントの横のドット数が含まれていなければなりません。TEXKNJ では、そのドット数を ^d で指定します。縦も同じドット数である必要があります。ただし、後述の PK 圧縮したものや、KG 和文フォントでは縦横のドット数が同じである必要がなく、そのドット数がファイル名に含まれる ^d に対応する数とも一致する必要がありませんが、フォントの選択自体は、そのファイル名に基づいて行なわれます。指定の仕方は、例えば

```
set TEXKNJ=c:\font\jfont\kanji^d;c:\font\jfont\jisfont.^d
```

のようにします。

TEXKNJ の中に ^s があると、それは対応する JFM ファイルのファイル名から拡張子 (3 文字でなければならない) を除いた部分の、さらに後ろから数字を除いた文字列に置き代わります。例えば、goth5.tfm, goth10.tfm などのときは、^s には、goth が代入されますので、

```
set TEXKNJ=c:\font\jfont\^s^d.pk;c:\font\jfont\kanji^d.pk
```

のように指定します。これは、例えばc:\font\jfont というディレクトリにgoth24.pk, kanji24.pk, kanji32.pk, kanji48.pk などの和文フォントが存在するときの指定です。

4.7.2 入手可能な和文ビットマップフォント

現在、JIS の 24 × 24、16 × 16、tachibana14 × 14 のフォントが、フリーで手に入れることが可能です。また、京都大学教育ソフト研究開発クラブの開発したフリーウェア LABOsystem123 には、32 × 32、48 × 48 のフォントが存在します。これを、dviout/dviprt で使えるように変換するバッチファイルが、labofont.lzh に入っています。

4.7.3 和文ビットマップフォントの圧縮 (knjtopk.exe)

圧縮されていない和文ビットマップフォントは、付属のプログラムknjtopk.exe で PK フォントの標準圧縮、または JXL4 フォントで用いられる拡張圧縮と同様の方式で圧縮をすることができます。圧縮したフォントファイルは、もとのものと全く同様に使うことができます。また、圧縮されたフォントでは、縦と横のドット数が異なるものも用いることができます。圧縮の仕方は

```
knjtopk c:\font\jfont\kanji.24 temp.24 24 24 H
```

のようにします。c:\font\jfont\kanji.24 がオリジナルのフォントで、temp.24 という PK 圧縮されたフォントが作成されます。パラメータの 2 つの数字のうち最初の 24 は横のドット数の指定、後の 24 は縦のドット数の指定です。縦のドット数と横のドット数が等しいときは、縦のドット数の指定を省略できます。オプションH は拡張圧縮を意味しますが、ドット数の小さいフォントの場合、このオプションを付けない方が圧縮率が高くなることがあります。

また、オプションf を指定すると、圧縮途中の様子が表示されますが、極端に処理スピードが落ちます。オプションr を指定することにより、圧縮したフォントファイルをもとにもどすこともできます。

4.7.4 和文ビットマップフォントの変換 (knjfont.exe)

他の和文フォントのファイルからの変換に役立つknjfont.exe というプログラムが付属しています。詳しくは、knjfont.doc をご覧ください。

4.7.5 ゴシック文字の合成 [オプション-G]

代替フォントを使用する場合、デフォルトで DVI ファイルから得られるフォント名にgoth が含まれている和文フォントに対し、ゴシック文字を合成して用います。ただし、対応するフォントファイル名にも同じ文字列goth が含まれている場合には、この合成を行ないません。例えば

```
set TEXKNJ=c:\font\jfont\^s.^d;c:\font\jfont\kanji.^d
```

と設定してあったとしましょう。フォント名goth10のフォントで32×32ドットのものが必要な場合、goth.32という名前のフォントが、c:\font\jfontに存在するときには、そのフォントをそのまま加工せずに使います。goth.32というフォントが見つからない場合で、kanji.32という名前のフォントがc:\font\jfontで見つかれば、そのフォントのビットを1 dotずらして重ねあわせることによってゴシック(強調文字)文字を合成して用います。両者とも存在しなければ、別の近いドット数のものを探します。

フォント名にgoth以外の文字列が含まれているフォントに対してこのゴシック文字機能の合成を行ないたいときには、オプション-Gに続けて、そのフォント名を表わす文字列を書きます(例えば、

```
-G=min
```

のように)。このとき、新しく指定したフォント名に対してのみ、ゴシックの合成を行なうことができます(複数のフォント名を同時に指定することはできません)。

```
-G
```

のようにフォント名を指定せずに用いると、どのような名前のフォントに対しても、ゴシック文字機能による合成を行なわないようにできます。

また、書体倶楽部/JG Font/和文のTrueType Font/LBPの内蔵フォントを用いる場合には、ゴシック文字を別途指定できます(ゴシック用のフォントを指定したり、擬似的に合成することが指定できる)ので、そちらの項を参照してください。この場合も、-Gで指定した文字を含むフォント名(デフォルトはgoth)のフォントが、ゴシックのフォントとみなされます。

4.7.6 倍角文字の合成 [オプション-K]

和文代替フォントのうち倍角文字(縦、横ともに2倍なので、4倍角とも呼ばれる)を合成して使うものは、

```
-K=;num_of_dots1;num_of_dots2; ... ;num_of_dotsn;
```

のようにオプション-Kを用いて指定することができます。JXL4フォーマットの和文フォントのように代替フォントでないものは、このオプションで指定できません。*num_of_dotsn*には、倍角として使用可能な横幅のドット数を「;」で挟んで以下の例のように書きます(各ドット数は18以上でなくてはなりませんが、複数指定可です)。

```
-K=;32;48;
```

これは、横幅が16ドット、または24ドットの和文フォントが存在すれば、倍角にして、32ドット、48ドットとして使用してもよいことを指示するものです。実際にどのフォントを使うかは、-Sの解説(cf. 本章p.24)で述べたような優先順位で定まりますが、32ドットのフォントの方が、16ドットを倍角にして32ドットにしたものより優先されます。

4.7.7 システムフォントのサイズ

dviout/dviprtでは、オペレーティングシステムが標準的にサポートしている和文フォントを使用できます。たとえば、PC-9801ではROMに内蔵されたフォントです。以下のサイズの和文フォントが使用できます。

PC-9801 ノーマル : 16×16
PC-9801 ハイレゾ : 24×24
DOS/V : 16×16, 24×24
J-3100 : 16×16, 24×24
AX : 16×16

ただし、DOS/V では、-romf= のパラメータを指定し、$font.sys または Jfont.sys で /24=ON として 24dot のフォントが組み込まれている場合にのみ 24 × 24 の和文フォントを使うことができます。また、システムに和文フォントを組み込んでいないときに和文フォントが含まれている DVI ファイルを扱うには -romf=0 を指定する必要があります。

4.7.8 システムフォントの縮小/拡大/切り替え [オプション -romf]

-romf=*max_dots*[:*min_dots*[:*min24_dots*]]

とすることにより、システムフォントを縮小または拡大して合成した和文フォントを使用することができます。

これは、指定された *max_dots* ドット以下、*min_dots* ドット以上のドットサイズのフォントに対してのみ実行されます。速度は、ほんの少し低下します。

DOS/V 版、J-3100 版では、24dot のシステムフォントを使うか、16dot のシステムフォントを使うかの境目を *min24_dots*(=24dot のシステムフォントを使う最低サイズ) で指定できます。

デフォルトは、-romf=-1:0 (DOS/V 版では -romf=-1:0:10000、J-3100 版では -romf=-1:0:20) であり、-varf+ を指定しない限り、システムフォントの縮小/拡大機能を使用しないことを意味します。ただし、-romf=0 は特別の意味があり、システムフォントの不使用を意味します。

例えば、-romf=15 とすることにより、6pt などの極小文字に対しても、印刷イメージにより近い出力が dviout で得られます。

また、-romf=10000 とすると、10000 ドット以下の和文フォント (すなわち事実上すべての和文フォント) に、16 × 16 ドット (または、24 × 24 ドット) のシステムフォントを拡大または縮小したものを用いるようになります。この場合は、和文のビットマップフォントが不要になります。文字の品質はあまり良くありませんが、dviout で使用すると便利でしょう。

4.7.9 システムフォントの使用限定 [オプション -k]

このシステムフォントの縮小/拡大機能を使うときに、ある特定のドット数の和文フォントについては、システムフォントの縮小/拡大したもの以外の (例えば、PK 形式で圧縮した) フォントを使いたいときは、そのドット数を

-k=*dots*[;*dots*;*dots*...]

のように指定します。ここで指定したドット数は、-romf= で設定した範囲から除かれます。オプション -romf を使用していない場合にも、16 ドット幅の和文フォントは、パソコンのシステムフォントを用います (PC-9801 ハイレゾモードでは 24 ドット幅フォントになります) が、-k=16 (ハイレゾモードでは、-k=24) を指定すると、システムフォントを用いません。ただし、他のフォントが見つからなかった場合は、システムフォントを用いることになります (システムフォント不使用は、-romf=0 です)。

PC-9801 では起動前の画面に漢字が表示されていると、システムフォントを用いるときにその漢字の部分がちらつきますが異常ではありません。

区切りの文字「;」の代わりに、数字以外の文字を使うことが可能です。

例 :

例えば、以下のようにします。

```
-romf=10000 -k=24;32;48
```

4.7.10 和文ビットマップフォントの縮小/拡大 [オプション-varf]

```
-varf[+|-]
```

のスイッチを ON にすることにより、システムフォントとディスク上の和文ビットマップフォント (独自方式で圧縮したものも含む) の両方に無段階の縮小/拡大を行なうことで、スケーラブルフォントとして用いることができます。

基礎になるフォントは、このオプションをつけなかった場合に用いられるフォントです。

フォントの JFM ファイルを読んで、それに該当する縦横のサイズになるように縦方向と横方向を個別に拡大/縮小するので、平体や長体の JFM フォントにも対応できます。

4.7.10.1 すべてのフォントの縮小/拡大

この-varf+ オプションとサイズ縮小/拡大オプション-e を併用すると、欧文フォントや JXL4 フォントを含むすべてのフォントの縮小/拡大が行なわれます (cf. 第 7 章 p.61)。

基礎となるフォントは、このオプションをつけなかった場合に用いられるフォントです。

4.8 スケーラブルフォント

dviout/dviprt では、和文代替フォントとして、(株) ツァイトの書体倶楽部/JG Font や Microsoft Windows 3.1 や Windows95 の TrueType Font を使用できます。

これらのフォントを利用するには、ベクトルフォント定義ファイルと呼ばれるファイルに必要情報を記述し、オプション-vfn+ を指定します。

4.8.1 書体倶楽部/JG Font

ベクトルフォントである「書体倶楽部」のフォントを使えば、最適な大きさの和文フォントを適宜作成するため、本来の大きさの和文フォントによる、美しい出力を得ることができます。ただし、ベクトルフォントの展開に (かなり) 時間がかかることや、フォントを購入しなければならないという欠点もあります。

「書体倶楽部」のフォントは、線分でつないで作られるフォントですが、同じく (株) ツァイトのワードプロセッサ JG Ver.3.0 からは、「JG Font」と呼ばれる 3 次ベジェのアウトラインフォントを使用するようになりました。dviout/dviprt でもこの JG Font を、従来の「書体倶楽部」のフォントと同じように扱うことができます[7](以下このドキュメントでは、「書体倶楽部」フォントと「JG Font」を合わせて、ベクトルフォントと呼ぶこととします)。

[7]Ver.2.43 現在、JG Font の出力は書体倶楽部フォント程美しくはありません。

なお、吉澤氏によって、ビットマップフォントから、「書体倶楽部」形式のベクトルフォントへの変換を行なう便利なユーティリティーが作成されています。

4.8.1.1 ベクトルフォント定義ファイル (dviout.vfn, dviprt.vfn)

ベクトルフォントである書体倶楽部/JG Font を使うには、環境変数TEXCFG で設定されたディレクトリに、ベクトルフォント定義ファイル (dviout.vfn とdviprt.vfn) をコピーし、それらのファイルの中身を自分の環境に合わせて書き直します (dviout.vfn とdviprt.vfn は、環境変数TEXCFG, PATH の順に探されます)。

4.8.1.2 ベクトルフォントの利用 [オプション-vfn]

ベクトルフォント定義ファイルを書き直したなら、dviout/dviprt に

```
-vfn+
```

オプションを指定することで、和文フォントとして書体倶楽部のフォントや JG Font が使われるようになります (第一水準のフォントしかない場合でも使えます)。

```
-vfn-
```

はベクトルフォントを使用しないことを表わします。一方

```
set vfn=name
```

のように環境変数を設定すると、dviout.vfn, dviprt.vfn の代わりに *name*.vfn を使うように指定できます。これはオプション-vfn を用いて

```
-vfn=name
```

とすることもできます (このときには必然的に -vfn+ の意味も表わします)。

和文の TrueType Font を用いるときにもこのオプションを指定します (cf. 本章 p.42)。

> (以下のベクトルフォントについての記述は、sempa 氏・ーみー氏によって Ver.2.38 用に書かれたものを、JG Font 対応 (Ver.2.39 以降) に際して Naochan!氏が修正したものの一部をもとに、とがし氏が LaTeX 化したものです。)

4.8.1.3 ベクトルフォント定義ファイルの設定

ベクトルフォント定義ファイルの内容は 4.1 p.37 のようになっています。

普通に使うのであれば何も考えず、 4.1 p.37 の “Vector Font Name” の部分を、自分の環境に合わせて書き換えさえすれば、そのままベクトルフォントを使うことができます。実際に期待したベクトルフォントが使われていることを確認するには、起動時のオプションで-f=1 を指定してください。終了時に表示される使用フォントのリストに、設定したベクトルフォント名があげられるはずです。“Vec(??)” という表示がベクトルフォントであることを示します。

なお、パスの区切り記号には「\」の他に「/」を使うこともできます。

```
#########################################################
#      DVIOUT, DVIPRT 用ベクトルフォント定義ファイル     #
#########################################################
#
#   #で始まる行はコメント行です。不用であれば削除してください。
#
%version = 2
#  vfn ファイルの整合性チェック用です。
#
%vfont_list
#  ベクトルフォント定義部
#    1) Font No. の指定 [1-5]
#    2) ベクトルフォントのパスを記述する。拡張子を付けてはいけない
#    3) w_adj フォントサイズの幅の補正
#        200 - 2000(基準 1000)
#    4) h_adj フォントサイズの高さの補正
#        200 - 2000(基準 1000)
#    5) 第 2 水準の文字を使わないのであれば、no vf2 に 1 を設定する
#
#    ','が項目間区切りになるので、忘れないように!
#
# Vec Font No.   Vector Font Name                  w_adj h_adj no vf2
#-------------+-----------------------------------+-----+-----+------
 1,            b:\vector\明朝,                     1000, 1000,   0
 2,            b:\vector\ゴシック,                 1000, 1000,   0
 3,
 4,
 5,
#
%jfm_list
#  JFM ファイル定義部  使用するフォントのパラメータを記述する。
#
  ----- 途中省略 -----
#
#         Vec  長           thin
#  JFM    Font ／ 斜 Draw   xfat
#  Name    No. 平     sw    yfat    novec  f_goth  rot
#--------+----+--+--+----+--------+-------+------+-----
 min,      1,  a, a, n40, 100;0;0,      ,     ,     0
 goth,     2,  a, a, n40, 100;0;0,      ,    1,     0
 tmin,     1,  a, a, n40, 100;0;0,      ,     ,     0
 tgoth,    2,  a, a, n40, 100;0;0,      ,    1,     0
#
#end of dviout.vfn
```

ファイル例 4.1: ベクトルフォント定義ファイル

4.8.1.4 ベクトルフォント定義ファイルの書式

以上の設定で最低限の環境は整いますが、以下の説明を参考にベクトルフォント定義ファイルの変更を行なえば、フォントの微妙な調整や多彩なフォントの使用が可能になります。

バージョン番号の記述

dviout/dviprt 本体と、ベクトルフォント定義ファイルの組み合せの整合性をチェックするため、定義ファイルの中のどこかにバージョン番号を記述する必要があります。現在の定義ファイルのバージョンは「2」ですので、

```
%version = 2
```

のように記述してください。ベクトルフォント定義ファイルのバージョンが変わった時 (その時には書式が変わっているはずです) には、それに合わせて変更する必要があります。

ベクトルフォントの定義

行頭に “%vfont_list” を記述すると、その行からファイルの終端、もしくは次の定義部までがベクトルフォントの定義部となります。

Vec Font No	ベクトルフォント番号
Vector Font Name	ベクトルフォントの名前
[w_adj]	フォントの幅補正係数
[h_adj]	フォントの高さ補正係数
[no vf2]	第二水準文字使用不使用のスイッチ

ここで各項目は「,」で区切って、1 行に 1 フォントずつ記述してください。また「[]」でくくった項目は省略可能です。各々の項目の意味は以下のとおりです。

1. Vec Font No (ベクトルフォント番号)
   ベクトルフォントは最大 5 種類の登録が可能です。1 から 5 の数字で記述してください。なおそれぞれのフォントに対し、書体倶楽部では第一水準 (.vf1)、第二水準 (.vf2) の 2 つのファイル、JG Font では、非漢字 (.fn0)、第一水準 (.fn1)、第二水準 (.fn2) の 3 つのファイルがあるので、最大 10 個ないし 15 個のベクトルフォントファイルを使用することになります。
2. Vector Font Name (ベクトルフォントの名前)
   使用するベクトルフォントのフルパスを記述します。ただし拡張子 (.vf1, .vf2 / .fn0, .fn1, .fn2) は付けません。
3. w_adj h_adj (フォントサイズの補正)
   書体倶楽部シリーズのフォントの中には、一部若干小さめにデザインされたものがあります。そこで出力されるフォントの大きさを幅と高さについて、各々“w_adj” の項と “h_adj” の項とで調整してください。どちらも基準を 1000(デフォルト値) として 200 から 2000 の範囲で指定できます。例えば「書体倶楽部」の「教科書体」では大体 1200 程度が良いようです。その他のフォントについては、実際に出力してみて調整してください。
4. no vf2 (第二水準文字不使用の指示)
   通常ベクトルフォントは、各々のフォントに対して、第一水準 (.vf1)、第二水準 (.vf2) の両方のフォントを用意しておく必要があります (JG Font の場合は、第一水準 (.fn1)、第二水準 (.fn2) の他に、かな・記号類の入った .fn0 のフォントが必要になります。)。
   しかし、第二水準の文字を使わないのであれば、“no vf2” の項に ‘1’ を設定することにより、第一水準のフォントだけ使用するようになります。この場合、万一第二水準の文字が使われていると警告がでます。この警告を無視すると、その文字は黒く塗りつぶした box で出力されます。なおデフォルトは ‘0’(第二水準を使用) です。

JFM ファイルの定義

ベクトルフォント定義ファイル中で、行頭に “%jfm_list” を記述すると、その行からファイルの終り、または次の定義部までが JFM ファイルの定義部となります。この部分で、TEX で使われているフォント名 (すなわち JFM ファイルの名前) と、実際に使用するベクトルフォントを対応させます。またその他に、フォントの変形やフォント展開の調整パラメータなどの指定も行ないます。

JFM Name	JFM ファイルの名前
Vec Font No	ベクトルフォント番号
[長/平]	長体/平体の指定
[斜]	斜体の指定
[Draw sw]	フォント描画方法の指定
[thin]	標準モード用描画調整パラメータ
[xfat]	精細モード用描画調整パラメータ
[yfat]	精細モード用描画調整パラメータ
[novec]	ベクトルフォント不使用のドットサイズ
[f_goth]	擬似ゴシックの指定
[rot]	フォント回転の指定

ここでも各項目は「,」(thin,xfat,yfat の間は「;」) で区切って、1 行に 1 フォントずつ記述してください。また「[]」でくくった項目は省略可能です。各々の項目の意味は以下のとおりです。

1. JFM Name (JFM ファイルの名前)
   和文フォントに対応する JFM ファイル名から、拡張子と末尾の数字を除いたものを記述します。例えばmin10 を使う場合には、対応する JFM はmin10.tfm となるので、“.tfm” と “10” を除いた “min” を指定します。なお、ここで “min” を 1 つ定義すれば、“min5” から “min20” まですべてにマッチします。
2. Vec Font No (ベクトルフォント番号)
   上で指定した JFM に対して、どのベクトルフォントを使って出力するかを、ベクトルフォント番号 (ベクトルフォント定義部で定義した番号) で指定します。
3. Draw sw (フォント描画方法の指定)
   フォント描画方法は、「f」、「o」、「n」の 3 種類のいずれかで指定します。各スイッチは次のような意味を持ちます。

   f (fill edge) 中身を塗りつぶす描画方法
   o (trace outline) 縁をなぞる描画方法
   n (f + o) 縁をなぞって中身を塗りつぶす

   これらの描画を、「標準モード」と呼びます。
   一方、「f」もしくは「n」の後ろに数値を指定すると、その数値以下のドット数 (幅) では、「精細モード」になります。「f」、「n」どちらを指定しても、「精細モード」になれば描画されるフォントは同じです。例えば「n40」と記述すれば、0 → 40 ドットまでのフォントは「精細モード」で、41 ドット以上のフォントは、「標準モード」の「縁をなぞって中身を塗りつぶす描画方法」になります。
   「標準モード」と「精細モード」の使い分けについては、本章 p.40 の「描画モード」を参考にしてください。
4. 長/平、斜体 (フォントアレンジの指定)
   これら項目は、アレンジフォントを使用する場合のみ変更してください。それ以外はどちらの項にも「a」を指定します。
   「長/平」の項には、‘a，b，c，d，e，g，h，i，j’ のいずれか、「斜体」の項には、‘a，b，c，d，e，f，g，h，i，j，k，l，m’ のいずれかが指定可能です。各パラメータの意味と使い方については、arrange.doc をお読みください。
5. thin，xfat，yfat (描画調整パラメータ)
   この 3 つのパラメータは「;」で区切って指定します。

(a) `thin`
描画方法のスイッチに「`f`」もしくは「`n`」が指定され、かつ「標準モード」で描画される場合に有効となるパラメータで、塗りつぶしの際の濃度を指定します。デフォルトは '100' で、この時完全に黒く塗りつぶされます。値を小さくしていくと段々と薄くなります[8]。

(b) `xfat, yfat`
「精細モード」で描画される場合に有効となるパラメータです。精細モードでは、フォントが比較的細く描画されますが、あまりに細いと感じる場合には、'`xfat`" と "`yfat`" に数値を指定すれば、指定したドット数だけ、それぞれ横方向と縦方向に線を太くできます。デフォルトはどちらも '0' です。

6. `novec` (ベクトルフォント不使用のドットサイズ)
あるドット数ではベクトルフォントを使いたくない、といった場合などには、この項目にそのドット数を記述してください。「`;`」で区切って複数指定可能です。
例えば、

> 22 ドットのフォントはベクトルフォントを使わずに、24 ドットのビットマップフォントを使いたい。また 30 ドットのフォントには 32 ドットのビットマップフォントを使いたい

といった場合には、「`22;30`」と指定してください。なおここで指定する数値は、あくまでも要求されるジャストサイズのドット数であることに注意してください。

7. `f_goth` (擬似ゴシックの指定)
`dviout`/`dviprt` には、ゴシック体のフォントがない場合を想定して、他のフォントを加工して擬似ゴシックを生成する機能があります。この機能を制御するためのパラメータです。次の 3 つの数値で指定してください。デフォルトは '0' です。

> **0** "`goth`" および `-G` で指定のフォントを擬似ゴシックにする
> **1** 擬似ゴシックの処理を強制的に禁止
> **2** 強制的に擬似ゴシックの処理を行なう

"`goth`" に「書体倶楽部」の「細ゴシック」などを使う場合などは、何も指定しないと、折角の細いゴシックが擬似ゴシック処理されて太くなってしまうので、'1' を指定をすると良いでしょう。

8. `rot` (フォント回転の指定)
グラフの縦軸に使う場合などを対象に考えられたオプションです。次の 4 つの数値で指定してください。デフォルトは '0' です。

> **0** 上向き
> **1** 右向き
> **2** 下向き
> **3** 左向き

4.8.1.5 ベクトルフォントの描画モード

ベクトルフォントの描画には、2 つのモードが用意されています。これは前述したとおり、「標準モード」と「精細モード」の 2 つで、各々フォントの展開処理が若干異なっています。この違いは結果として、出力されるフォントの質と速度に現れます。では、それぞれのモードについてもう少し詳しく説明します。

1. 「標準モード」

[8] 上記の説明は、標準の "TPIC による描画機能" を使った場合にあてはまります。コンパイルの際に「`VFTPIC`」をはずすか、もしくは「`NOTPIC`」を定義した場合には、"`thin`" パラメータの意味が変わり、フォントの線の太さを表わすようになります (デフォルトは '50')。

TPIC の機能を使って展開するモードです。このモードでは、フォントの展開に要する時間は、比較的短くなります (それでも、ベクトルフォントを使用しない場合に較べると、桁違いに遅くなります。)。

2. 「精細モード」
   残念ながら、標準モードは速い反面、展開されるフォントの線幅が若干太めになります。解像度の十分高いデバイスではあまり気にならないようですが、そうでなければ、小さなフォントは潰れてしまいます。そのような時に「精細モード」を使用すれば、フォントが細めに展開され、小さなドット数でも潰れにくくなります。しかし、「標準モード」に比べて処理時間は長くなってしまいます。

この 2 つのモードのどちらを選択するかは、処理時間と出力の美しさを天秤にかけて判断する必要が有りますが、例えば「明朝体」では、最低「n40」、つまり 40 ドット以下は「精細モード」とするのが良いのではないでしょうか。

4.8.2 和文 TrueType Font

dviout/dviprt では、Microsoft Windows 3.1 や Windows95 が標準としてサポートしている和文 TrueType Font を使用することができます。

4.8.2.1 準備と使用法

まず ttindex.exe を用いて、たとえば

```
ttindex a:\windows\system\msmincho.ttf
```

とすると、a:\windows\system にある TrueType Font ファイル msmincho.ttf から、拡張子を .tti に変えた msmincho.tti というファイルが同じディレクトリに作成されるはずです。

TrueType インデクスファイル (*.tti) は、TrueType Font (*.ttf) をアクセスする際に利用する各種情報、コード変換テーブルなど保持しており、約 17K byte のサイズになります。

Windows95 に付属している TrueType Font の msmincho.ttc および msgothic.ttc にも対応しています (ttindex.exe Ver.0.4 より)。この場合は、

```
ttindex a:\windows\system\msmincho.ttc
```

などとして windows\fonts に、TrueType Font の拡張子を .tti と変えたファイルを作成してください。通常 windows\fonts は、hidden 属性になっているようです。

このようにして作成した *.tti ファイルは、必ず元の *.ttf と同じディレクトリに置いてください。次に、たとえば 4.2 のように dviout.vfn, dviprt.vfn に利用するフォントのパスを記述してください。

```
# Vec Font No.   Vector Font Name                 w_adj h_adj no vf2
#-------------+---------------------------------+-----+-----+------
 1,            a:\windows\system\msmincho,        1000, 1000,   0
 2,            a:\windows\system\msgothic,        1000, 1000,   0
 3,
 4,
 5,
```

ファイル例 4.2: TrueType Font 用ベクトルフォント定義ファイル

もちろん書体倶楽部/JG Font も同時に指定できます。Vector Font Name 以外のパラメータの意味は書体倶楽部/JG Font と同じです。ただし TrueType Font では no vf2 は無視されます。

以上の設定によって、書体倶楽部/JG Font のときと同じく `-vfn+` によって TrueType Font の利用が可能になります。

4.8.2.2 問題点

TrueType Font のアウトラインは 2 次 B スプライン曲線で構成されています。現在の処理では 2 次 B スプライン曲線を直線近似し、書体倶楽部フォントの展開ルーチンに渡しています。このため書体倶楽部フォントと比較してアウトラインを構成する線分の数が多くなり、展開速度が低下します (速度低下には `*.tti` ファイルなどに対するディスクアクセスによるものも含まれます)。

また、現状では TrueType の composite glyph 形式のグリフデータには対応していません。composite glyph は複数のグリフデータの組み合わせ、変形によって一つの文字を構成する方式で、英文フォントのアクセント付き文字などで使われています。msmincho, msgothic は composite glyph 形式のグリフデータを含んでいないようですが、別の和文 TrueType Font では漢字の偏や旁のデータを composite glyph 形式で持っている可能性も考えられます。現在 composite glyph 形式を含む和文 TrueType Font は `ttindex.exe` でチェックして使えないようにしています。

さらに、TrueType の特徴でもある hint 情報は一切利用していませんので、低解像度における文字品質は MS Windows と比較して明らかに劣ります。

4.9 巨大フォント

`dviout`/`dviprt` では、ある文字を使用すると、そのビットマップデータをメモリー内の Expanded font buffer (cf. 第 5 章 p.52) に保存し、再度現れた場合にそれを用いることによって高速化を実現しています。一方、メモリー内に保存される文字データの 1 文字のサイズは、約 64K byte (700×700dot 程度) まで、としています。それ以上のサイズの文字が使われることは少ないと思われますが、その場合には「raw PBM ファイルの形でディスク上に展開して、再度読み込む」という方法で処理します。

4.9.1 ディスク上に展開されるフォント

ディスク上での展開がサポートされているフォントは、PXL1003、PK、PKD の欧文フォント、および JXL4、PK 形式で圧縮された和文フォント、システムフォント、KG 和文フォント、書体倶楽部/JG Font/TrueType Font です。

圧縮されたフォントでは 1 文字のデータサイズが 64K byte 以内にである必要があります。また、展開後のフォントの横方向のサイズは 8000dot 以内でなければなりません (縦方向は、特に制限無し)。ただし、pTEX 縦書きモードでは、巨大な JXL4 形式のフォントはサポートされていません。

これらのフォントで、展開後のサイズが 64K byte 以上になる文字、あるいは拡大/縮小の処理があってそれがメモリー内で行なえない大きなサイズの文字の場合にディスク上での展開が用いられます。

具体的には、`-gdat` (cf. 第 12 章 p.116) が設定してあれば、そのディレクトリに、設定してなければカレントディレクトリに、raw PBM 形式 (cf. 第 12 章 p.124) の `bf$.pbm` というファイルとして巨大文字が展開されます。

なお、ディスク上で展開を行った場合で拡大/縮小を伴うときは、スムージングの処理が省略されます。書体倶楽部/JG Font/TrueType Font の巨大フォントは、512×512dot のサイズに展開したものから、スムージングの処理なしに拡大したものとなります。

4.9.2 LBP のスケーラブル和文フォントの場合

LBP 内蔵のスケーラブル和文フォントの場合は、巨大なフォントも問題なく扱えます。LIPS III の場合は、テキストモードでは縦横のサイズが 420 dot までという制限がありますが、それを越える場合はベクトルモードに移って対応しています。なお、1 ページを複数枚の用紙に分割して印字するような場合は、`-p=` のサブパラメータ`E` を指定してください (cf. 第 10 章 p.91)。

4.10 フォントのアーカイブ

欧文のフォントファイルは種類が多いため、解像度の異なるものをそろえると、ファイルの数が大変多くなります。(ハードディスク上では、どんなに小さなサイズのファイルでも、一定のサイズを占有してしまいますので) それらのファイルをハードディスクにインストールするには、かなりのディスクスペースを必要とすることになります。このため、それらを (例えば解像度毎に)1 つのファイルにまとめて扱う、ということが考えられます。

`dviout/dviprt` は、そのために独自ファイル形式である GTH 形式のファイルをサポートしていて、PK, PXL1001, PXL1002, PXL1003, PKD, JFM のファイルをまとめることができます。

同様なものとして、アスキー日本語 MicroTEX で使われていた FAR ファイル、emTEX の FLI ファイルがあり、`dviout/dviprt` ではこれらの形式もサポートしています。

4.10.1 GTH ファイル (`gather.exe`)

`gather.exe` を用いると、複数の PK フォントや PXL フォントなどをひとまとめにした GTH ファイルをつくることができます (拡張子も含めて、同一の名前のものは不可)。

例えば、ディレクトリ`c:\font\118` に 118dpi の欧文の PK フォントが`cmr10.pk`, `cmr8.pk` などという名前で入っているときは、

```
gather -a dpi118 c:\font\118\*.pk
```

とすることにより、`dpi118.gth` という GTH ファイルにまとめることができ、ディスクスペースの節約に役立ちます。GTH ファイルには、収録したファイルのパス名のうち、ドライブ名、ディレクトリ名を除いたファイル名のみが記録されます。

GTH ファイルには、PK, PXL1001, PXL1002, PXL1003, PKD, JFM ファイルを混在して入れておくことができ、`dviout/dviprt` から直接読むことができます。

GTH ファイルは `^g` を用いて`TEXPK` に指定します。

```
set TEXPK=c:\font\dpi^d^g^s.pk
```

とすると、`c:\font\dpi118.gth` という GTH ファイルの中の、例えば `cmr10.pk` というように解釈されることになります (`^g` の前のファイル名の拡張子のデフォルトは `.gth` です)。通常どおり、「`;`」で区切って他のフォント指定と併用できます。前から見ていって、最初に見つかったものが採用されます :

```
TEXPK=c:\font\^d\^s.pk;c:\myfont\dpi^d\^s.pk;c:\font\dpi^d^g^s.pk
```

`gather.exe` でオプション・パラメータ`-v`, `-u`, `-e`, `-d` などを指定することにより、GTH ファイルに収録されたファイル名を表示したり、GTH ファイルに対して指定したファイルの追加、取り出し、削除ができます。ファイル名の指定は複数並べて書くことができ、ワイルドカードが有効です。例えば次のように指定します。

```
gather -ub c:\font\dpi118 c:cmb*.pk c:line*.pk
```

詳しくは、gather.doc を参照してください。

4.10.2 FAR ファイル

アスキーの日本語 MicroTEX で使われていた FAR ファイルをサポートしています。GTH ファイルと同様 ^g を使って

```
set TEXPK=c:\font\^l.far^g^s
```

というように指定します。

4.10.3 FLI ファイル

dviout/dviprt は、emTEX の FLI 形式のフォントライブラリに対応しています。FLI 形式のフォントライブラリは、lj_0.fli, lj_h.fli,..., lj_5b.fli, lj_sli.fli などの名前になっています。(環境) 変数TEXPK ではこの可変部分を ^f で表現し、^f を置換する文字列を環境変数TEXFLI に列挙して指定します。

例えば上記のファイルがc:\font\ というディレクトリ上にあるときには、次のように設定します。

```
set TEXPK=c:\font\lj_^f.fli^g^s;...(その他の検索パス)
set TEXFLI=0;h;1;2;3;4;5a;5b;sli
```

^f の含まれるフォント指定だけを FLI 形式のライブラリと見なしますので、必ず 1 個だけ ^f が含まれるように設定してください。

なお、-L オプションは、デフォルトの-L- のままにしておいてください。-L+ の場合には、フォント名の中央省略が行なわれますので注意が必要です。

また、TEXFLI の設定もdviout.par/dviprt.par などのパラメータファイルで指定することが可能で、^f, ^g, ^d, ^l などは %f, %g, %d, %l と表記することもできます。

FLI 形式フォントライブラリでは、フォントがどのファイルにあるかが名前から判断できないため、検索が遅くなる可能性があります (一度読んだファイルの情報は、フォントの検索が終わるまで保持しておき、この問題を回避しています。ただし near ヒープを使うため、あまりたくさんのファイル名の組み合わせがあるとエラーになることがあります)。

4.11 フォントのフォーマット

通常 TEX の欧文のフォントには、METAFONT プログラムを使って生成されたものが用いられます。

cmr10 (Computer Modern Roman 10 point) で、300dpi (300dot/inch) のものをMETAFONT で作成すると、cmt10.tfm というフォントメトリックファイルと、cmr10.300gf (DOS では、cmr10.300) という GF[9]フォントファイルが作成されます。前者は、解像度 (今の場合は 300dpi) には依らないファイルで、文字送り幅 (各文字に対し、その文字を置いたときの次の文字の標準の位置) や、「f と f とが隣り合うと ff となる」などという情報が書かれています。後者は、各文字のビットマップデータが記録されたファイルで、解像度毎に異なったファイルになります。

[9]generic font

dviout/dviprt のような TEX のデバイスドライバは、この GF フォントではなく、ある種の圧縮を施してサイズを小さくした PK フォントファイルを用いるのが通例です。GF フォントから PK フォントへの変換には、gftopk というプログラムが用いられます。

なお、メモリー上の制限から大きなサイズ (64K byte 以上) の PK フォントが扱えなかったり、効率が悪くなることがあります。これを解消するためdviout/dviprt は独自の PKD ファイルをサポートしています。これは、pktopkd.exe を使って PK ファイルから作成されるもので、各文字のビットマップデータの PK ファイルにおける位置が書かれているものです。

4.11.1 PKD ファイルのフォーマット

ヘッダの部分は、PK ファイルの形式と類似しています。

```
247, 63, k[1], min[2], max[2], name[k-4], ds[4], cs[4], hppp[4], vppp[4]
```

最初の 2 byte は、247, 63 で、これは、PKD ファイルの識別に使います。min[2], max[2] は、登録された文字コードの最大のものと、最小のものです。name[k-4] は、記録されたパス名で、終わりは 0 が入っています。ds, cs, hppp, vppp は、デザインサイズ、チェックサム、横方向の 1 ポイントあたりのドット数、縦方向の 1 ポイントあたりのドット数が、PK ファイルからそのままコピーされて入っています。

このヘッダの後には、各文字データのある PK ファイルにおける先頭からの位置が、3 byte 整数で、min から、max まで順に並んでいます。ただし、対応する文字データが PK ファイルにない場合は、0 が記録されています。数は、全て上位バイトから下位バイトへ、という順に並んでいます。

4.11.2 PXL1001, PXL1002, PXL1003

dviout/dviprt では、欧文フォントに PK フォントの他、PXL1001, PXL1002, PXL1003 の形式のフォントも扱うことができます。PXL1001 形式と PXL1002 形式は非圧縮ですが、前者は幅を word(TEX では 4byte) 単位で、後者は byte 単位で切り上げて格納しています。PXL1003 は、アスキー日本語 MicroTEX 独自のもので PK フォントと同様な圧縮法を取ったものですが、各フォントに対する情報は、他の PXL 形式に合わせたものです。

4.11.3 JXL4

アスキー日本語 TEX では、和文フォントに対して、欧文のフォントメトリック・ファイルを拡張した和文のフォントメトリック・ファイルである JFM ファイル (min10.tfm, goth10.tfm など) と、各文字の圧縮されたビットマップデータを格納した JXL4 形式のフォントファイルが使われます。

4.11.4 和文ビットマップフォントのフォーマット

dviout/dviprt ではシステムフォントなどの他、フリーで使用可能なビットマップフォントをサポートしています。そのフォーマットは以下のとおり極めて単純なものです。

和文フォントの 1 文字のサイズを、$n \times n$ としましょう。m を、$(n+7)/8$ を越えない最大の整数とします。1 文字のデータは、文字のビットマップの最上位列の左から右へ、各列 m バイトで、順に下の行へと $m \times n$ バイトの通常の形式です。横幅のドット数が 8 で割り切れない

ときは、各列の最後のバイトの余った下位ビットには 0 を入れておきます。上位ビットが左側になります。

フォントファイルにはこれが JIS コード順に並んでいます。JIS コード J の文字のデータは、$J = [J_1, J_2]$ (J_1 は上位、J_2 は下位バイト) とすると、ファイルの先頭から

$$((J_1 - 33) \times 94 + J_2 - 33) \times m \times n$$

バイトをスキップした後になければなりません。

上の形式の和文ビットマップフォントは、解像度が大きくなるとファイルサイズが大きくなりますが、dviout/dviprt では、これを PK ファイルと同様な方法で圧縮した独自の PK 和文フォントをサポートしています。そのための変換プログラムは knjtopk.exe です。

4.11.5　KG 和文フォントのフォーマット

一方、少数の文字を登録して使うのに便利な独自の KG 和文フォントというものをサポートしています。圧縮をしていないビットマップデータが収められていますが、ファイルの先頭には収められている文字の情報が書かれています。

最初の 32 byte はヘッダで、圧縮された和文フォントファイルと同様

```
0, 0, 247, id[1], width[2], height[2], byte_width[2], dummy[22]
```

となっています。ただし、id は「G」となって、他の圧縮フォントと区別されます。

width は、フォントのドット単位での幅、height は、ドット単位での高さ、byte_width は、byte 単位での幅で、それぞれ下位、高位の順に 2 byte で書かれています。dummy は使われていませんが、0 になっています。

そのあと続くのは、入っている文字の JIS コードの表で

```
total[2], code[total][2]
```

となっています。total は、入っている文字数で、code[1],...,code[total] は、収録順に JIS コードが書かれています (width[2] などと同じ 2 byte 整数)。

その後続けて、隙間無く、圧縮していない文字データが最後まで並んでいます (その 1 文字のデータの形は、圧縮していない和文フォントのものと同一形式です)。

1. code[1],...,code[total] は、小さい順に並んでいなくてもよい
2. 1 文字のデータサイズは、(width+7)/8×height で、64K byte 以上の巨大フォントでもよい
3. JIS code が *code* の文字があるかどうかは、code[] を見なくてはならないが、code[j] が *code* に等しいとすると、j 番目に収められているので、先頭から (width+7)/8×height×j + 2×total + 34 byte のところからデータが存在する。

4.11.6　和文ビットマップフォントの圧縮フォーマット

knjtopk.exe を使って圧縮された和文ビットマップフォントには H をつけて圧縮したかどうかによって 2 種のフォーマットのものがあります。どちらも先頭の 32 byte は

```
0, 0, 247, id[1], width[2], height[2], byte_width[2], dummy[22]
```

となっており、先頭の 4 byte は、ファイルの識別に使われます。H をつけた場合に作成されるフォーマットを Huge format と呼び、後者を Small format と呼ぶことにすると、id は、前者が「H」、後者が 'K' となります。

width は、フォントのドット単位での幅、height は、ドット単位での高さ、byte_width は、byte 単位での幅で、それぞれ下位、高位の順に 2 byte で書かれています。dummy は使われていませんが、0 になっています。

先頭の 32 byte に続いて、各文字の圧縮データの位置などの情報が、Small format では 3 byte で、Huge format では 5 byte で表現されたものが、圧縮前の和文ビットマップフォントに並べられた文字順に格納文字数だけ並んでいます。Small format では、それに続いて、Huge format では 5 byte を飛ばした後に、各文字の圧縮データが同様の順序で隙間なく並べられています。

圧縮は、Small format のときは欧文フォントの PK フォーマットと同様な方法が、Huge format のときは和文フォントの JXL4 フォーマット同様な方法が用いられます (アスキー出版技術部責任編集『日本語 TEX テクニカルブック I』にこれらの圧縮の詳細が書かれていますが、若干のミスがあるので注意してください)。

各文字に関する位置などの情報は、Small format では以下のようになります。

```
ad1[1], ad2[1], ad3[1]
```

ad1 & 0x80 が ON のときは圧縮せずに元の和文ビットマップフォントのままで、OFF のときは欧文の PK フォントの圧縮方法で圧縮された形で格納され、後者では ad1 & 0x40 の ON/OFF で、文字の最左上のビットの ON/OFF を示します。実際のデータの格納位置の先頭は、ad1 & 0x3f, ad2, ad3 で与えられます (上位バイト→下位バイトの順で、22 bit)。圧縮のときの dyn_f は 12 に固定されています。

Huge format では

```
flag[1], address[4]
```

であり、ここで flag は JXL4 フォーマットのときの flag と同じですが、flag.s は 0 になっています。ただし、dyn_f = flag & 0x0f の値が 15 の場合は、圧縮しないもとのままのデータであることを示します (width が 8 の倍数以外であれば、dyn_f が 14 の場合と異なります)。address[4] は、データの格納アドレスで (下位バイト→上位バイトの順で、32 bit)、データサイズは次のデータの格納アドレスとの差で分かります (最後のデータのサイズを得るために、もう 1 文字分の 5 byte がついています)。

Small format では、各文字の圧縮データのアドレスが 22 bit で表現されるため、圧縮後のファイルサイズが 4M byte 以下である必要があります。

極端に小さなサイズのフォントの場合以外では、Huge format にした方が圧縮率が高いでしょう。

4.12 不足フォントの自動生成機能 [オプション-gen]

dviout/dviprt が必要なフォントを見つけられなかった場合に、dviout/dviprt の中から子プロセスで METAFONT を起動し、不足しているフォントを自動生成することができます。

基本的なフォント作成手順/環境は、あらかじめテンプレートファイルに設定しておく必要があります。dviout/dviprt は、テンプレートファイルの設定に従って必要なフォントの名前やサイズなどのデータを挿入したバッチファイルを作成し、実行します。当然、メタフォントが既にインストールされていて、正常に動作している環境が必要です。

また、METAFONT ソースが見つからない、あるいは設定パラメータにミスがあった場合など、自動生成できなかったフォントについては、必要なルーチンをまとめて別のバッチファイルに出力して手作業で必要なフォントの生成ができるようにしています。

-gen=*template_file*

というオプションでテンプレートファイルを指定すると、この機能が ON になります。

4.12.1 テンプレートファイル

template というテンプレートファイルのサンプルを付けてありますので、それを見ながら読んでください (cf. 4.3 p.49)。書式は以下の通りです。

1. 1 文字目が 「#」で始まる行はコメント文であり、バッチファイルには出力されません。ただし、1 つだけ例外があります。2 文字目が 「!」の場合は、そのまま出力されます。

   例 :

   # この文は出力されない。

   #! この行は出力される。

2. テンプレートファイルは、4 つの領域からなります。各領域は、各々1 文字目が 「%」で始まる、3 種類の行

   %1st

   %2nd

   %3rd

   で区切られます。
3. %1st の行より前は、特殊なパラメータを設定する領域です。この領域で設定できるパラメータは 4 つあります。

   auto=no

   子プロセスで自動的にフォント生成を行なうのを禁止し、gen_font で指定した名前のバッチファイルのみ生成します。

   extra_size=300

   dviout の実行で足りないフォントが見つかった時に、プリンタ用のフォントも作りたい場合に、プリンタの dpi 値を設定します。この値は、5 の ^e の計算に使用されます。

   gen_tmp=

   子プロセスに用いるバッチファイルの名前をフルパス名で指定します。gen_tmp.bat がデフォルトです。

   gen_font=

   子プロセスでは生成できなかったフォントを作成するためのバッチファイル名をフルパス名で設定します。デフォルトは、gen_font.bat です。

   mode_name=

   dpi 値と METAFONT の mode 名の対応関係を設定します。書式は、

   mode_name=<dpi 値>:<mode 名>

   です。例えば、mode_name=300:CanonCX は、300dpi の場合は CanonCX という mode 名を対応させることを表わします。この mode_name の行は、テンプレートファイルの中に何回でも書けますが、extra_size= の行よりは後に書いてください。

```
######################################################################
#
mode_name=118:bitgraph
mode_name=300:CanonCX
#
%1st
#
echo off
rem
#
%2nd
#
virmf &cmplain \mode:=^n; mag:=^m; input ^s
gftopk ^s.^dgf ^s.pk
rem
#
%3rd
#
echo Done!
######################################################################
```

ファイル例 4.3: テンプレートファイル

4. 2 番目の%1st と%2nd の間には、echo off のようなバッチファイルの前処理コマンドを入れておきます。また、4 番目の%3rd 以降には、エラー処理などの後処理コマンドを入れておきます。
5. 3 番目の、%2nd と %3rd の間の領域が、実際にフォントを作成するためのルーチンです。この領域では、^ + 英小文字 (例^s) を使うことで、必要な情報を挿入することができます。
   - ^s は、見つからなかったフォント名に置換されます。
   - ^m は、そのフォントの倍率に置換されます。
   - ^d は、そのフォントのサイズ (dpi 値 × 倍率) に置換されます。
   - ^D は、dpi 値 に置換されます。
   - ^n は、dpi 値に対応した mode 名に置換されます。
   - ^N は、extra_size= で設定された dpi 値に対応する mode 名に置換されます。
   - ^e は、1 番目の領域で設定された dpi 値 × 倍率に置換されます。例えば、dviout の実行で、足りないフォントが見つかった時に、ついでにプリンタ用のフォントも作ってしまいたい場合に、1 番目の領域で extra_size=360 のように書いてプリンタの dpi 値を設定しておくわけです。

   ^を含む文字列をバッチファイルに出力したい場合は、^^と 2 つ続けてください。例えば、^^s とすれば、置換は行なわれず、^s という文字列が出力されます。

4.12.2 簡単な例

4.3 に簡単なテンプレートファイルの例を挙げておきます。

cmr10 scaled 5000 というフォントが見つからなかった場合、次のような内容のgen_tmp.bat が出力され、子プロセスで実行されます。

```
echo off
rem
virmf &cmplain \mode:=bitgraph; mag:=5.000000; input cmr10
gftopk cmr10.590gf cmr10.pk
rem
echo Done!
```

子プロセスでcmr10 scaled 5000およびcmtt10 scaled 2488というフォントの生成に失敗した場合には、次のような内容のgen_font.batが出力されます。

```
echo off
rem
virmf &cmplain \mode:=bitgraph; mag:=5.000000; input cmr10
gftopk cmr10.590gf cmr10.pk
rem
virmf &cmplain \mode:=bitgraph; mag:=2.488000; input cmtt10
gftopk cmtt10.294gf cmtt10.pk
rem
echo Done!
```

第5章 メモリー設定

5.1 バッファサイズの指定

dviout/dviprt は、 5.1 の 5 種類のバッファを使用し、それに対応するオプションで、それぞれのバッファ量を操作することができます。

	オプション	条件	デフォルト	
			dviout	dviprt
Bit Map buffer	-bb		0	0
Expanded font buffer	-br		80000	80000
Font file buffer	-bf	PC-9801 ノーマル	65520	98288
		PC-9801 ハイレゾ	65520	65520
		統合版		65520
		その他	65520	100000
Working buffer	-bw	-vfn 指定時	20481	0
		その他	0	
View Image buffer	-bv	PC-9801	6	–
		その他	−1	

表 5.1: バッファ一覧とデフォルト容量

5.1.1 ビットマップバッファ [オプション-bb]

-bb=*size*

1 ページ、またはそれを分割して、ビットマップの形で展開するためのバッファ(Bit map buffer) の大きさを指定します。このバッファが小さいと、1 ページが分割されて展開されます。

5.1.1.1 -bb=0 の意味

ただし、デフォルトの-bb=0 は特別の意味を持ち、メモリーを最大限活用して、dviout ではページが分割しないように、dviprt では-bb と-br に対応するバッファが最も大きくなるよう自動調整されますので、通常はデフォルトのまま使うのがよいでしょう。-bb=0 の意味は、詳しくは次の通りです :

dviout の場合

dviout では、-br, -bf で指定された大きさのバッファを割り当てた残りのメモリーのうち、1 ページが分割されないだけのメモリーを-bb に対応するバッファに確保します。足りなければ、残りのメモリーを全てこのバッファに割り当てます。さらに残ったメモリーがあり、-br の値が正の場合はそれをすべて-br に対応するバッファに、-br の値が負または 0 で-bw の値が −1 の場合は残りをすべて-bw に対応するバッファに割り当てます。

dviprt の場合

dviprt では、最低 80000byte を確保し、-br, -bf で指定された大きさのバッファを割り当てて余裕ができれば、-br, -bb に対応するバッファに、等分して割り当てます (ただし、-br の値が 0 または負の場合は、-bb にのみ割り当てます)。1 ページを分割せずに展開できるサイズ以上になった場合は、余りを

1. -br に対応するバッファが EMS に取れたときは-bw に対応するバッファに
2. そうでないときは-br に対応するバッファに

割り当てます。

このとき、-bb と-br に対応するバッファへのメモリーの割り当ては-br の値で調節します。例えば-p=l オプションを指定して、フォントのダウンロード機能と LBP の内蔵フォントを用いる場合は -br に対応するバッファのサイズをあまり大きくとる必要はないので、-br の値を小さくとるか負の値にすると、より効率的になることが考えられますが、メモリーの空き具合によっても異なります。

5.1.2 展開フォントバッファ [オプション-br]

-br=*size*

展開された文字に関するサイズなどのデータとその文字のビットマップデータのためのバッファ(Expanded font buffer)。1 文字ずつ、その文字が使われるときに初めて (フォントファイルのなどのデータより) 展開され、このバッファに格納されてからビットマップバッファに転送されます。以降、同じ文字を使うときは、このバッファから読みます。ただし、バッファが一杯になったらバッファをフラッシュ(クリアー) します。

このバッファは、EMS が存在すれば、そこから 64K byte 単位で、最大 32M byte まで割り当てることができます (EMS からは、デフォルトで可能な限り多くを割り当てます)。conventional memory に確保されたものと合わせて使うことができます。

-br=0 の場合 *size*=0 は、EMS のみを使うことを意味します。高速の EMS メモリーが十分にある場合は、この指定-br=0 がよいでしょう。

-br= 正の場合 *size* は、conventional memory から確保されるバッファの最低サイズです。

-br= 負の場合 *size* が負の場合は、丁度 $-size$ の大きさのバッファを conventional memory から確保することを意味します。

ただし、-bb がデフォルトの-bb=0 になっている場合は、*size* の正負に応じて意味が異なるので、前項を参照してください。

br=80000 が初期値です。

5.1.3 フォントファイルバッファ [オプション-bf]

-bf=*size*

フォントのファイルを格納しておくバッファ(Font file buffer)。バッファが一杯になったら、バッファをフラッシュします。いわゆるディスクキャッシュのような働きをします。デフォルトでは、PC-9801 においては表 GVRAM を使い、dviout では 65520 byte、dviprt では 98288 byte となります。

5.1.4 ワークエリア [オプション-bw]

```
-bw=size
```

conventional memory 上に確保されるワークエリア (Working buffer) のサイズを指定するもので、デフォルトは 0 です (EMS があれば、それもワークエリアに用いられます)。ただし、-vfn が設定されている場合は、このバッファを TrueType Font や書体倶楽部フォントの展開に使うため、デフォルトで約 20K byte 確保します。

例えば、pTEX での文字列の 90 度回転にワークエリアを使っています。EMS がないと 128×128 dot 以上のフォントが回転できませんが (EMS があれば、512×512dot まで可能)、例えば 32K byte 確保すれば、512×512 dot まで対応できます。

今後の拡張機能によっては、これを使うものが増える可能性があります。

5.1.5 表示イメージバッファ [オプション-bv]

```
-bv=page
```

このオプションによって、イメージ表示用のバッファ (View Image buffer) のページ数を指定します。

デフォルトでは、PC-9801 の normal mode のときは裏 GVRAM を用い、PC9801 の hireso mode の時は使用されない Red/Green/Level の 3 プレーンを使うことで、ともに 6 ページ分を確保します。それ以外の機種では *page*=−1 となります。

page が、正の値のとき、保存のためバッファを conventional memory に確保します。*page* が負のとき、−*page* というページ数に対応するバッファを EMS に確保します。

page には、−16 以上 16 以下が指定可能です。

5.1.6 制限

-bb, -br, -bf は、メモリーサイズに応じてのみ制限を受けます。

例 :

例えば、

```
-br=150000 -bf=100000
```

のように指定します。

プログラム終了後に、バッファの使用状態が表示されます (cf. 第 13 章 p.128)。高速に動かすためには、flush が 0 で、split が 1 であることが望まれます。必要に応じて、各バッファのサイズを変更します。フリーのメインメモリーが大きければバッファも大きく取れます。dviprt の場合は、たとえ split が増えても-bf や-br に対応したバッファがフラッシュしないようにするとよいでしょう。

5.2 EMS の利用 [オプション-EMS]

dviout/dviprt は、EMS が存在する場合はそれを活用することができます。

EMS のメモリーは、展開されたフォント用のバッファ (Expanded font buffer) と、dviout におけるイメージ表示画面の保存、さらに今後の機能追加のためのワークエリアとして用いること (これについては、work_buf.doc を参照) ができます。使用する場合は、ワークエリア (Working buffer) として、最小 4 ページが確保されます。

-EMS=*page*

によって、dviout/dviprt で使用できる EMS の最大ページ数を指定します。

2048 がデフォルトで、32M byte まで (従って通常の場合、空いている EMS をすべて) 使用します。EMS が存在しなければ使用しません。

0 を指定すると、EMS が存在しても使用しません。

確保された EMS メモリーはワークエリアとして 4 ページとり、また、イメージ表示のバッファに指定されただけ確保し、残りを-br に対応するバッファに割り当てます。

第6章 印刷スタイルと出力ページ指定

6.1 改ページと袋とじ印字 [オプション-nf]

通常はページの下部に空白があっても、その部分の改行を省略して、改ページコードをプリンタに送ります。オプション・パラメータ

```
-nf
```

は、これを変更するものですが、-p=l や-p=m を指定したページプリンタとそれ以外とでは異なる意味をもちます。

ページプリンタ以外の場合は、-nf+ の指定によって、改行のコードを送り、改ページコードを送らないようにします。

また

```
-nf=2
```

とすると、印刷開始ページから数えた奇数番目のページ印字後には改行のコードを送り、偶数番目のページ印字後は改ページのコードを送るようになります。-V (cf. 本章 p.56) を併用してページサイズを調整することにより、袋とじ印字ができます (プリンタの行送りの誤差による「ずれ」は多少起こります)。

一方、-p=l や-p=m を指定したページプリンタの場合は、

```
-nf=[num_of_dots|長さ]
```

で袋とじ印字が指定できます。

すなわち、印刷を開始したページから数えて、奇数番目のページ印字後には改ページを行なわずに、*num_of_dots* で示された dot 数だけ右にずらした位置に偶数番目のページを印字します。

また、dot 数でなく新サイズオプションの「長さ」で指定することもできます。これにより、紙を横置き指定にして使えば袋とじになります。

例えば LIPS III の 300dpi の解像度のプリンタで LaTeX の A5j で作った文章を A4 で袋とじにするには

```
A>dviprt -y=A5P/A4L -nf=1754 dvifile
```

あるいは、

```
A>dviprt -y=A5P/A4L -nf=148.5mm dvifile
```

のように-nf= に A4 用紙の長辺の長さの半分を指定します。

A4 版で作成したtexjman.dvi に対し

```
A>dviprt -varf+ -e=707 -OX=-7.44mm -OY=-7.44mm -y=A5P/A4L -nf=148.5mm texjman
```

とすると A4 横置きの縮小・袋とじによる印刷ができます (cf. 第 7 章 p.61, 第 10 章 p.92)。

6.2 ページ最初のCR/LF制御 [オプション-T]

単票用紙による印刷の場合、プリンタによっては 1 ページ目、あるいは 2 ページ以降のページ上部の空きが無視されることがあります。

これによって不具合が起こるときには、

```
-T[+|-]
```

オプションを ON にすることにより解消できることもあります。

ページの最初のCR/LF を無視するプリンタがあるので、このオプションによってページの最初に空のビットイメージ印字を行なうコードを送っています。このオプションを指定してもうまくいかない場合は、プリンタ定義ファイルを作成して対処ください (escp_24.src の最後の部分を参照)。

6.3　ランドスケープ印字　　[オプション-V]

オプション

```
-V[+|-]
```

を ON にした場合、90 度回転した形での印字が可能です。-V はトグルスイッチです。これは使用頻度の観点から、高速化があまり考慮されていません。ただし、-p=l や-p=m を指定した LBP での印字の場合は、プリンタ側での 90 度回転を行ないますので高速です。

90 度回転出力を指定し、-H (新サイズオプションのときは-PW) で用紙の縦の長さの半分を与え、さらに-nf=2 を指定すると袋とじスタイルで印字されます (用紙の半分の大きさの DVI ファイルを出力するとき)。

このとき袋とじスタイルにするには、-H による指定ではなくて、-nf の後の数字または「長さ」で指定します。

また、オプション-y のサブパラメータL (-V+ に対応)、P (-V- に対応) によっても同様の結果が得られます (ただし、必ずA4 などの用紙の指定をすることになります)。

6.4　ページの指定

出力するページを指定をするときは、起動時のパラメータの最後で DVI ファイル名の指定よりもさらにあとに、

```
start_page-end_page
```

のように書きます。

dviout/dviprt 起動時のパラメータのうち、DVI ファイル名とこのページの指定だけは、パラメータファイルに書くことができません (dviprt に対するページ指定は、後述のようにページ指定ファイルに書くことができますが、DVI ファイル名とdviout に対するページ指定は、常にコマンドライン上から指定することになります)。

ページとページの区切り文字は、「-」です。「10-11」だとか、「-21」や「2-」といった指定も可能で、空白で区切って、複数並べて書いてもかまいません。(複数並べた、1 つ 1 つをブロックと呼び、dviout では [RETURN] により次のブロックへ移動します。前のブロックにもどることはできません。)

dviout で、数字の区切り記号として、「-」の代わりに「+」を使うと、*start_page* よりも前のページにまで戻ってプレビューすることができます。

dviprt では、「.」、「Z+」、「Z-」、「Z」を指定することができ、「.」は空のページの出力を、「Z+」、「Z-」、「Z」はそれ以降、ページ印字後一旦停止するかどうかのフラグをそれぞれ ON, OFF 反転します (cf. 第 13 章 p.127)。

さらにdviprt では

```
@page_file
```

と指定することにより、出力ページの指定をファイル *page_file* から読み込むことができます (このファイルを「ページ指定ファイル」と呼びます)。ファイル中では、コマンドラインと同様なページ指定ができるほか、「#」で始められた行は行全体がコメントとして無視されます。例えば、袋とじ印字の機能と組み合わせて (cf. 本章 p.55)

```
# 両面袋とじ
4 Z+ 1 Z 2-3 8 Z 5 Z 6-7 . Z 9 Z 10-11
```

のようにファイルに記述します。なお、ページ指定のファイル中で別のファイルを読み込むことはできません。

6.4.1 DVI に書き込まれたページノンブルの使用 [オプション-page]

通常ページ指定に記述するページ番号は、最初のページが 1 ページ、次のページが 2 ページというように、物理的に何枚目かという数字で指定しますが、

```
-page[+|-]
```

を ON にすると、DVI ファイルの中に書いてある論理的なページ番号であるページノンブル (すなわち DVI に書き込まれた\count0 の値) を用いて指定することができます。TEX のソースファイルの中で、ページを指定したり変更しているときに意味を持ちます。

同一のページ番号があるときは、最初に見つかった方が対象になります (通常は先頭から探しますが、dviprt でオプション-o=r を指定すると後ろから探すことができます)。

6.4.2 dviprt での出力ページ制御 [オプション-o]

dviprt では、パラメータ

```
-o=[r][e|o]
```

で出力するページを制御でき、r e o はそれぞれ、逆順の出力、偶数ページのみの出力、奇数ページのみの出力を意味します。

o と e は、同時には指定できません。例えば、-o=re と指定すると、偶数ページのみが逆順に出力されます。用紙の両面に印刷する場合に使うと便利です。

この偶数、奇数に関しては、パラメータ-page の影響を受けません。

6.4.3 ページ指定のマクロ

さらに複雑なページ制御をdviprt で行なうには、ページ指定のためのマクロを使うことができ、以降のページ指定に有効です。それはM で始めた

```
Minitialize;difference;pages  start_page-end_page
```

の形式です。マクロ文は、255byte まで許されます。

このマクロは、次に新たなM で始まるマクロが現れるまで有効です。また、M のみ (続くマクロ文を書かないもの) は、以降のマクロの指定を解除することを意味します。

マクロ文中では、v, w, x, y, z の 5 つの変数を使うことができます。

pages は、「式」または「Z+」,「Z-」,「Z」,「.」を「,」で区切って並べたもので、z > 0 が成立すれば「式」の値のページの出力などを行ない、終了後は *difference* に従って変数の値を変え、さらに z の値を 1 減らしますが、z > 0 ならば再び *pages* に従ったページ出力を行なう、と繰り返します。

ここで、「式」とは、変数や数を「+」または「-」で結合したもので、x+5, x+v-3 などのようなものです。

「式」の値が *start_page* と *end_page* の間に入っていなければ、通常は空白ページが出力されますが、*pages* の「式」のいくつかを[ ]で囲んでブロックを作っておくと、その中に *start_page* と *end_page* の間のページ出力がない場合、そのブロックの出力を無視します。ただし、[ ] のネスティングはできません。

また、*initialize* と *difference* は、「代入文」を並べたもので、「代入文」は

変数 += 式　　変数-= 式　　変数 *= 式　　変数/= 式　　変数&= 式

の何れかの形のものです。

変数は、v = w = 0、x は *start_page*、y は *end_page*、z は *end_page*−*start_page*+1 が代入された後、*initialize* によって初期値が定まります。

ただし、-page+ が指定されていれば (cf. 本章 p.57)、最初からの通しページ番号に変換してから x, y, z へ代入されます。

また、*start_page* あるいは *end_page* が指定されていなければ、それぞれ、最初のページ、最後のページと解釈されます。

C 言語の形式で言えば

```
v = w = 0;
x = start_page;
y = end_page;
z = y - x + 1;
for(initialize; z > 0; difference, z--){
    pages の出力;
}
```

となります。

たとえば、-o=re は

```
Mv+=y,v&=1,y-=v,z-=v;y-=2,z-=1;y
```

と、両面印刷の袋とじは

```
Mz+=3,z/=4;x+=4;x+3,Z+,x,Z,[x+1,Z+,x+2,Z]
```

とマクロで記述できます ([の前の , は省略できます)。

後者を p4to1 というファイル名のファイルに書いて TEXCFG で指定されたディレクトリに入れておけば

```
@p4to1 15-50
```

というように dviprt の起動時の最後のパラメータでページ指定ができます。

6.5　dviout のページ・レジューム機能　[オプション-B]

dviout 専用のオプションです。前回、最後に表示したページから表示を再開することができます。

```
-B-, -B=-
```

ページ・レジューム機能を使用しない。

```
-B+, -B=+, -B
```

ページ・レジューム情報の記録先をメモリーとする。(-B としてもトグル動作しません。)

```
-B=file_name
```

file_name で指定したファイルをページ・レジューム情報の記録先とする。*file_name* はフルパス名で記述する。

ページ・レジューム情報の記録先をメモリーとした場合は、コンピュータを起動した直後や、他のプログラムを動かした後などの 1 回目は通常この機能は働きません。

6.5.1 メモリー上のページ・レジューム情報の記録位置

6.5.1.1 PC-9801 版

ノーマルモードでは GVRAM と TEXT VRAM に、ハイレゾモードでは GVRAM に書き込んでいます。

6.5.1.2 DOS/V 版

0040:00f0 からの 5 バイトにレジューム情報を書き込んでいます。

AT の純正の BIOS では0040:00f0-0040:00ff の 16 バイトは ICA(Inter Communication Area) としてアプリケーション間交信用のエリアとして予約されています。ただ、PS/2 の BIOS には、このエリアはありません。

DOS/V のdviout ではこの 16 バイトのうち最初の 5 バイトを使用しています。

この領域は、現在全く使用されていないので最近のサードパーティの BIOS では無視されている可能性もあります。メモリーをレジューム情報の記憶先とする場合には、このことにご注意ください。

6.5.1.3 J-3100 版 [J-3100 版dviout でのオプション-E]

使用する機種によって、以下のように指定し、アドレスも異なります。

```
-E=n
```

```
n = 0:  レジュームアドレス = 0040:00f0 (DynaBook V 以外)
    1:          〃         = 0040:00eb (DynaBook V)
```

6.5.1.4 HP100LX 版

レジューム用のワークエリアとして、9000:FAB0 から 5 バイト使用しています。この領域は、Low Battery 等の表示をシステムが行う際、該当箇所の VRAM を退避するための領域のほぼ中央にあたります。ROM のバージョンの差により、多少領域が移動しても弊害はないものと思われます。ただし、当然のことながら、Low Battery 等の表示が行われると、レジューム情報は消滅してしまいます。

第7章 解像度とマグニフィケーション

7.1 解像度の指定 [オプション-dpi]

```
-dpi=resolution
```

フォントの解像度を dpi (1 インチあたりのドット数) で指定します。

dviout ではデフォルトの値がありますし、dviprt でも各プリンタの定義ファイルでデフォルトの解像度を指定しますので、通常は、ユーザが-dpi オプションを指定する必要はありません。

なお、正常に印字するためには、-dpi がプリンタに合わせて正しく設定されていると同時に、この解像度に合わせたフォントがインストールされている必要があります。

7.1.1 dviout の場合

dviout のデフォルトの解像度は、118dpi です。

7.1.2 dviprt の場合

dviprt では、-p オプションで指定するプリンタによってデフォルトの解像度が異なります。

オプション	プリンタ	デフォルトの解像度
-p=p	NEC の PC-PR 系 24 ドット	160
-p=n	NEC の NM 系 24 ドット	180
-p=e	EPSON の ESC/P24 ドット	180
-p=l	LIPS III	300
-p=m	ESC/Page	300
	(無指定の場合は-p=e と解釈されます)	

表 7.1: プリンタ毎のデフォルトの解像度

また、-p=o<プリンタ定義ファイル>とした場合は、それぞれのプリンタ定義ファイルに記述されている dpi 数がデフォルトになります。ただしこの場合、プリンタ定義ファイルがdviout/dviprt Ver.2.34 以降で提供されたoptcfg.exe Ver.2.1 以降で作成されている必要があります。それ以前のバージョンのoptcfg.exe で作成したプリンタ定義ファイルも使用できますが、その場合はデフォルトは 180 になります。

7.2 縦方向の dpi の指定 [オプション-DPI]

横方向と縦方向の dpi が異なるときは、縦方向の dpi を、

```
-DPI=resolution
```

で指定します。

横方向は、-dpi で指定したもの、またはデフォルト値が使われます。このオプションを指定するときには、縦方向と横方向の dpi が指定したものとなっているフォントを用意しておく必要があります。

7.3 マグニフィケーションの指定 [オプション-mag, -half]

マグニフィケーションに関しては TEX と同様に 1.2 のべきで指定することができます。また、マグニフィケーションの値を与えることもできます。指定には-mag または-half+ を用います。

```
-mag=magstep
-mag=magnification
-half[+|-]
```

magstep が 1 のときは、1.2 倍の解像度のフォントが使われ、2 の時は 1.2^2 倍のもの、3 の時は 1.2^3 倍のもの、というように解釈されます。*magstep* には 0 から 9 までの整数が指定できます。magstep を half にするときは、-half+ を指定することで $1.2^{1/2}$ 倍の解像度となります。-mag や-half を指定しなかった場合は、DVI ファイルに書かれている magstep の値を使います。一方、*magnification* に 500 以上の値を指定したときは、*magnification*/1000 倍と解釈されます。すなわち、-mag=1 と -mag=1200 とは同じことを意味します。

例えば、-mag=1 -dpi=160 と指定したときは、通常 160 × 1.2=192dpi のフォントを使います (実際には、magstep0 の 192dpi フォントと magstep1 の 160dpi フォントは同じではありませんから、この記述は正確ではありませんが、PK フォントのパス名や、GTH ファイル名に 192 が含まれている PK フォントファイルが、(環境) 変数TEXPK に従って探されます)。

7.4 擬似的縮小/拡大 [オプション-e]

オプション

```
-e=number
```

により、縦横とも *number*/1000 倍のサイズに変えることができます。-mag では、フォント自体も拡大されたものを使いますが、-e を用いる場合は、字間の空白を調節することによってページのサイズを変えます。*number* の値は、300 から 4000 の間で指定します。すなわち 3/10 から 4 倍までが指定できますが、*number* に標準の 1000 から余り離れた値を指定すると、歪みが大きくなります。

7.4.1 和文代替フォントとの関係

和文代替フォント (すなわちシステムフォント、ディスク上のビットマップフォント、書体倶楽部フォント、LBP 内蔵のスケーラブルフォントなど) は、-e オプションの影響を受けます。例えば、-e=500 とすると縦横の大きさが半分のサイズの和文フォントを最適のフォントとみなしてそれを探します。従って、-e=500 -romf=10000 のようにシステムフォントの縮小/拡大のオプションが同時に指定されているとシステムフォントの漢字が-e=500 に適応したサイズになるようスケール変換されて用いられます (cf. 第 4 章 p.34)。

7.4.2　全フォントへの影響

さらに、-varf(cf. 第 4 章 p.35) オプションと併用すると、欧文フォントと JXL4 フォントを含む和文のすべてのフォントがスケール変換されて用いられます。従って、dviout で-e=500 -varf+ と指定すると、縦横それぞれ半分に縮小された画面を見ることができます。

7.4.3　新サイズ「長さ」との関係と影響　　[オプション-esize]

新サイズオプションの「長さ」は、通常この-e オプションの影響を受けないため、画面上の枠の大きさなどに変化はありませんが、dviout でオプション

```
-esize[+|-]
```

を ON としたときには、新サイズオプションの「長さ」にも影響を及ぼし、用紙枠の大きさや、仮想紙面サイズが-e= オプションの値に従って変化し ます。例えばページが大きいためにdviout において画面が分割されてしまう場合に、このオプションを -e=800 -esize+ のように指定することによって分割を避けることができます。

第8章 位置・サイズの指定

表示/印字位置の調整やその範囲を調整するオプションには、従来からの DVI ファイルに書かれているページサイズを基準にする方法と、用紙サイズを基準にする方法との 2 通りあり、後者を新サイズオプションと呼ぶことにします。2 つのうちの一方のみが有効になります。

dviout/dviprt では、1 ページのビットマップデータをメモリー内に確保した領域 (Bit map buffer) に展開してから画面表示や印字出力を行ないますが、従来はこのメモリー内に確保する領域のサイズ (縦横の長さ) を、DVI ファイルに記述されたサイズを元に決めていました。DVI ファイルによっては、書かれてあるページサイズをはみ出して文字などを配置する命令が書かれていることがあり、この場合 Bit map buffer からはみ出た部分はカットされて出力されません。こういった場合には、オプション -X, -Y, -w, -h を用いて Bit map buffer のサイズを増やしたり、DVI 原点の位置をずらしたりすることで対応してきました。

一方、Ver.2.39 で導入された「新サイズオプション」では、Bit map buffer のサイズを DVI ファイルに記述されたページサイズではなく、用紙サイズを元に決めます。これを用いると、ページの上下左右が切れることは、ほとんどなくなると思われます (新サイズオプションの元で、後述のようにパラメータを正しく設定していて、なお端が切れる場合には実際の表示/印字可能範囲外にデータが置かれてしまったことを意味します)。

ただし、新サイズオプションを用いない場合に比べて、より大きな Bit map buffer が必要となることが多いので、dviout でもページが分割される可能性が増えます。また、A4 縦以外の用紙サイズでタイプセットされた DVI ファイルを dviout/dviprt で用いる時は、それに応じた用紙サイズのパラメータを指定しなくてはなりません。

8.1 用紙サイズ基準の新サイズオプション

-PW, -PH, -TM, -BM, -LM, -RM, -MW, -MH, -OX, -OY, -HC, -VC, -HS, -VS, -PF があります。
新サイズオプションでは

- サイズの値は基本的にドット数ではなくて実寸で指定します。
- 原則として、DVI ファイルの原点と実際の印刷の原点を一致させます。
- dviout でのイメージ表示の枠は、メモリー内に確保された 1 ページのサイズを表わします。
- 新サイズオプションが使われた場合、従来のサイズ関係のオプション -W, -H, -X, -Y, -w, -h, -P, -Q, -C, -M は (指定されていても) 総て無視されます。
- dviout の新サイズオプションで確保されるバッファのデフォルトは (A4 用紙縦サイズなので)、

  ```
  -X118 -Y118 -W976 -H1380 -M+ -newsize-
  ```

  と同値です。

8.1.1 新サイズオプションの使用/不使用 [オプション-newsize]

```
-newsize[+|-]
```

は、トグルスイッチ型オプションで、新サイズオプションを使用する/しないを切り替えます。デフォルトは ON で、使用することを表わします。

8.1.2　長さの単位

新サイズオプションの大半は、パラメータとして「長さ」を与える必要がありますが、これは次のいずれかの単位を付けた数字で指定します。単位を付けなかった場合には、エラーとなります。

1. mm：ミリメートル
2. cm：センチメートル
3. in：インチ (1in = 25.4mm)
4. pt：ポイント (1in = 72.27pt)
5. dot/*number*dpi：特定の dpi におけるドット数。*number* の部分に、dpi 値を指定する。
   例 :-MW=1800dot/360dpi：360dpi のプリンタでの 1800 ドット分の長さ (= 5in)

マイナスの値は、「-」を数字の前に付けることによって表わします。

8.1.3　プリンタ個別の指定

ここに示す指定は、用紙に対する値なので、-V+ や、-y= オプションのランドスケープ印字指定とは独立しています。LBP の場合などを除き、通常同じ値を使用します。

8.1.3.1　印字不可能域の指定　　[**オプション**-TM, -BM, -LM, -RM]

あらかじめ、用紙の上下左右の縁にあって、その**プリンタでは印字できない部分の長さを指定**しておきます :

-TM= 長さ：用紙上端の印字不可能域の幅
-BM= 長さ：用紙下端の印字不可能域の幅
-LM= 長さ：用紙左端の印字不可能域の幅
-RM= 長さ：用紙右端の印字不可能域の幅

以上のデフォルト値は、すべて0mm となっています。ただし、-p=l または-p=m による LIPS III または、ESC/Page モードでの LBP 印字を指定した時のデフォルトは、-TM=10.3mm, -LM=5mm, -RM=5mm, -BM=5mm です (さらに、ランドスケープ印字を指定すると、-TM=5mm, -RM=10.3mm に変わります)。この10.3mm は、5mm + 63dot/300dpi にあたります。

これらのオプションは、マージンを表わすものではありませんので注意してください。

-V+、あるいは-y のサブパラメータL によってランドスケープ印字を指定するときには、-LM, -TM, -PW (ただし、LIPS III または、ESC/Page モードでの印字のときは、-TM, -RM) が正確に設定されていないと印字位置がずれます。

-OX, -OY を変更しても印字位置の調整を行なうことができますが、-LM, -TM が設定されていない状態で、印字位置がずれている場合には、-OX, -OY を指定しないで、まずこの-LM, -TM を調整して、DVI 原点が用紙の左上端から右下に 1inch ずつ入ったところになるようにしてください。

このオプションにより印字位置の調整を行なう場合、

- **印字位置が右にずれている場合、-LM の設定値を増やす**。
- **印字位置が下にずれている場合、-TM の設定値を増やす**。

具体的な数値は、プリンタのマニュアルに印字できない領域の幅として載っている場合が多いので、これを参照してください。

なお、-LM (左印字不可域の幅) は、通常デフォルトの -PF=1 (左端を揃えた給紙) の場合にのみ指定し、-PF=2 (中央を揃えた給紙) の場合は指定しないのが標準です (指定すると正確に中央にそろいません)。

また-MW (印字可能域の最大横幅) を設定することにより-RM (右印字不可領域の幅) は-PW と-LM から自動的に判断されるので、通常-RM の指定は不要です (PW - LM - MW が負の時はRM = 0 となって用紙の右端まで印字可能になり、そうでない場合はRM = PW - LM - MW となって、プリンタの印字可能最大幅まで印字できます)。

LBP などのように、用紙のサイズによらず機械的制約か何かの理由で、常に右端が印字できない時だけ-RM を設定する必要があります。ただし、LBP でランドスケープ印字をする可能性があって、LBP 印字のランドスケープ印字の場合のデフォルトの10.3mm と異なるなら、必ず設定してください。

8.1.3.2 印字可能な最大サイズの指定 [オプション-MW, -MH]

その**プリンタで印字可能な領域の最大サイズを指定**します :

-MW= 長さ : 印字可能域の最大横幅 (Maximum printable Width)
-MH= 長さ : 印字可能域の最大縦幅 (Maximum printable Height)

以上、指定がない時は用紙サイズとマージンから計算します。

通常 (-PF=1 で、回転した印字でない場合)Bit map buffer の横のサイズは、PW - LM - RM を基準にしますが、-MW が設定してあれば、それを越えていないかどうかチェックされます。越えていれば、RM を増やすことによって対処されます。縦方向も同様です。連続用紙に印字できるプリンタの場合は-MH は設定する必要は無いでしょう。

8.1.3.3 BJ-10v の時の例

```
-TM=8.5mm -BM=12.7mm -LM=3.4mm -MW=203.2mm
```

-RM は0mm、-MH は無限大なので指定不要。

上記のような各オプションとプリンタ依存事項は、bj10v.par といったファイルに入れておき、-= で読み込むとよいでしょう。

8.1.3.4 dviout での指定

以上をdviout に対して指定すると、印字可能域に枠線が表示され、印刷した時に端が欠けないかどうかの確認に使えます。

また、dviout で、

-V+

や-y オプションのランドスケープパラメータ (L) を指定することによって、用紙のサイズの縦横を入れ替えることができ、dviprt でのオプション-V+ (cf. 第 6 章 p.56) やオプション-y (cf. 第 10 章 p.92) でランドスケープ印字を選んだ場合と整合がとれているので、dviprt と同じパラメータファイルでプレビューすることができます。

8.1.3.5　給紙位置 (dviprt のみ)

プリンタの給紙位置については、主に次の二つがあります。

1. 印字可能域の左端と用紙の左端を揃える場合
2. 印字可能域の中央に用紙を置く場合

通常は前者ですが、それ以外では以下の-PF による指定をします (デフォルトは、-PF=1)。

-PF=[1|2] 1: 左端を揃えた給紙 (paper feed position = left) 2: 中央を揃えた給紙 (paper feed position = center)

-PF=2 はセンタリングを行なうということではありませんのでご注意ください。-PF=2 を指定した時には、印字可能域の最大横幅を基準に中央をそろえますので、-MW を正しく指定してください。

8.1.4　印字原点　[オプション-OX, -OY]

DVI ファイルでは、通例、用紙左上を基準として (1inch, 1inch) = (25.4mm, 25.4mm) の位置に原点があります。基本的には、(-LM, -TM が正しく設定してあれば) この原点が実際の用紙の上でも (1inch, 1inch) の場所に来るように印字されます。

しかし、何らかの理由で原点をずらしたい時は、次のオプションを指定します。

-OX= 長さ : 横方向の移動 -OY= 長さ : 縦方向の移動

デフォルトの値は、どちらも0mm です。

指定した場合は、DVI ファイルの原点を指定方向に動かします。

8.1.5　用紙のサイズ　[オプション-PW, -PH, -y]

表示および印字の用紙サイズを指定します。

-PW= 長さ : 用紙の横幅 (Paper Width) -PH= 長さ : 用紙の長さ (Paper Height)

デフォルトの値は、次の例にある A4 用紙のサイズです。

-y=A3|A4|A5|B4|B5|H

を指定すると、それぞれの値は以下のものがデフォルト値になります (cf. 第 10 章 p.92)。

A3 用紙の時　: -PW=297mm -PH=420mm
A4 用紙の時　: -PW=210mm -PH=297mm
A5 用紙の時　: -PW=148mm -PH=210mm
B4 用紙の時　: -PW=257mm -PH=364mm
B5 用紙の時　: -PW=182mm -PH=257mm
H(はがき) の時 : -PW=100mm -PH=148mm

ここに無い用紙サイズをよく使う場合は、例えばUSletter.parという名前で-PW=8.5in -PH=11inといった内容のファイルを作っておいて、-= オプションを用いて-=USletter.parのように指定すれば、用紙サイズを簡単に切り替えできます。

8.1.6 センタリング [オプション-HC, -VC]

DVI ファイルから読み取ったテキスト幅、高さを文書のサイズであるとみなし、新サイズオプションで指定された用紙サイズ (PW, PH) を元にして、用紙の中央にセンタリングを行ないます。

-HC[+|-]：水平センタリング (horizontal centering)
-VC[+|-]：垂直センタリング (vertical centering)

縦と横のセンタリングは独立していますので、どちらか一方だけを指定することもできます。
このオプションを指定した場合は、-OX, -OY の値は無視されます。
デフォルトではどちらも OFF です。

8.1.7 センタリング位置の調整 [オプション-HS, -VS]

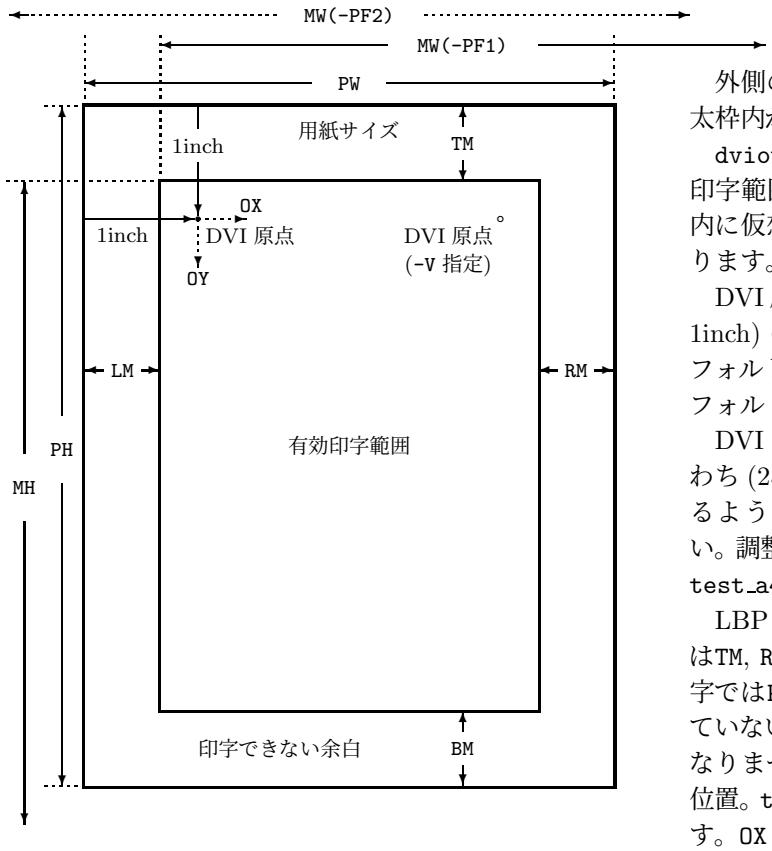
次のオプションは、上記のセンタリングを行なった後で印刷位置を調整したい場合に指定します。指定する値は、センタリング後の位置に対する相対的な移動量です。
センタリングを指定していない場合には無視されます。

-HS= 長さ：水平センタリング後の移動 (horizontal shift after centering)
-VS= 長さ：垂直センタリング後の移動 (vertical shift after centering)

デフォルトではどちらも0mmとなっています。

8.1.8 新サイズオプションの図示と概略

用紙の大きさを指定してタイプセットした DVI ファイルを正しい位置に印字するには、プリンタ固有のパラメータをdviprtに伝えなければなりません。例えば、余白無しに左端から印字することのできないプリンタがほとんどですから、用紙上でプリンタが印字可能な最左上端と用紙の左上端のずれを知って TEX の DVI ファイルの原点の位置が用紙の正しい位置に来るようにする必要があります。
また、-HC オプションなどで用紙の中央に正しくセンタリングするには、-PW, -LM を正しく設定して、dviout/dviprt に知らせる必要があります。
さらに、プリンタで印字可能な最大幅を設定しておけば、これを越えないよう調整がなされます。
新サイズオプションでは、解像度に依存しない「長さ」でパラメータを指定できるので、dvioutとdviprtで共通の値を指定すればよく、dviout で用紙に対する印字結果を正確にプレビューすることができます。
8.1 p.68 に、用紙に対して、それぞれのオプションが意味する「長さ」を図示しました。用紙が基準ですので、-V+ オプションなどを指定しても用紙に対する-PW, -LM などの意味は変わらないことに注意してください (このときは、印字のデータが 90 度回転しているので、DVI 原点は元の用紙で右上に変わり、-OX, -OY も 90 度回転して考えます)。
図の有効印字範囲に収まらないデータ (文字や図形) はカットされます。テキストのヘッダやフッタも含



外側の太枠が用紙サイズ、内側の太枠内が有効印字範囲です。

dviout/dviprt では、左図の有効印字範囲の大きさを元に、メモリー内に仮想紙面 (Bit map buffer) を作ります。

DVI 原点は、用紙の端から (1inch, 1inch) の位置となるのが TEX のデフォルトです。dviout/dviprt のデフォルトでもあります。

DVI 原点が (1inch, 1inch)、すなわち (25.4mm, 25.4mm) の位置になるようにLM, TM を設定してください。調整には、付属のtest_org.dvi, test_a4.dvi を用いると便利です。

LBP でのランドスケープ印字ではTM, RM が、-V による 90 度回転印字ではPW, LM, TM が正しく設定されていないと DVI 原点が正しい位置になりません (DVI 原点は図の白丸の位置。test_org.dvi が調整に使えます。OX とOY の方向も回転します)。

図 8.1: 用紙に対して新サイズオプションが意味する「長さ」

めてこの有効印字範囲内に収まっている必要があります (富樫氏のtestpage.sty を使うとチェックに役立ちます)。

デフォルトの-PF=1 では、MW, MH は、有効印字範囲がプリンタの印字限界を越えていないかどうかのチェックに使われます。越えていた場合は、RM やBM を増やして有効印字範囲が自動的に減らされます。

-PF=2 では、LM=0mm が標準で、このときは用紙がプリンタの印字可能限界の幅 (MW) の中央にあると考えた出力を行ないます (仮想紙面は、有効印字範囲から左にMW(-PF2) の左向き矢印の先まで拡大されます)。

-HC, -VC では、DVI ファイルに書かれているページの大きさに基づいて (上図の有効印字範囲の大きさとは独立ですが、この中に収まることが期待されます) 用紙の中央にページが配置されるように (OX, OY が) 調整されます。このとき、PW, PH, LM, TM, HS, VS が参照されます。

8.2　DVI ファイルに書かれた情報を基準とするオプション

DVI ファイルに記述されたページサイズを元に調整するオプションは-W, -H, -X, -Y, -w, -h, -P,-Q, -C, -M です。

一般に数は、10 進数、16 進数 (最初が「0X」か「0x」)、8 進数 (最初が「0」、以下が数字) のいずれで指定しても OK です。数値は、-dpi で指定した値でなく実際に使うディスプレイやプリンタの解像度に従ったドット数で幅を設定してください。

dviout では、-W, -H の大きさで枠が表示され、印刷イメージを知ることができます。

8.2.1 印字/表示可能な最大幅 [オプション-W]

-W=*num_of_dots*

で印字/表示可能な最大幅 (dviprt では、用紙の幅から、プリンタで印字不可能な左右の余白の幅を除いたものに対応) が定義されます。ページの大きさがこれを越えるかどうかチェックが行なわれ、越えるときはエラーメッセージが画面に出て、無視して実行するかどうか聞かれます。続行すると、設定された横幅からはみでた部分はカットして印字/表示します。ドット単位の大きさです。

-W=1800

のように書きます。特に制限を設けないときには、-W=-1 とします。

デフォルトでは、8inch の幅に対応するドット数になっています。dviout ではデフォルトが-W=944 となり、dviprt では 180dpi の場合-W=1440, 360dpi の場合-W=2880 となります。ただし、-p=l や-p=m を指定した場合は、-W=2360 がデフォルトとなります。

8.2.2 印字/表示可能な最大高さ [オプション-H]

印字/表示可能な最大高さ (dviprt では、用紙の縦幅からプリンタで印字不可能な上下の余白の長さを除いたものに対応) は、

-H=*num_of_dots*

で指定できます。

制限を設けないときは-H=-1 とします。

デフォルトでは、11inch の幅に対応するドット数になっています。すなわち、dviout ではデフォルトが-H=1298 となり、dviprt では 180dpi の場合-H=1980, 360dpi の場合-H=3960 となります。ただし、-p=l や-p=m を指定した場合は、-H=3387 がデフォルトとなります。

8.2.3 DVI の原点、左と上のマージン [オプション-X, -Y]

新サイズオプションを用いないdviout/dviprt では、デフォルトで DVI の原点が画面ないしプリンタの印字領域の左上隅になります (TEX では、左上隅から、右へ 1inch、下へ 1inch の所を DVI の原点にするのが通常ですのでご注意ください)。適切なマージンを設定するには、

-X=*num_of_h_dots* -Y=*num_of_v_dots*

を用いて、ドット単位で左マージンと上マージンを調整します。-X や-Y には、-Y=-30 のように負の数も指定できます。用紙の 1 ページより原稿が少し長く入りきらないが原稿の上部が空いているような時に、負の数を指定すると 1 ページにおさめることができます。

8.2.4 右マージン、下マージン [オプション-w, -h]

DVI ファイルで指定してあるページの大きさをはみ出して文字を置く命令が書かれている場合は、

-w=*num_of_dots* -h=*num_of_dots*

によって、それぞれ、右にはみ出ている長さ、下にはみ出ている長さをカバーするようにページサイズを拡大します。単位はドットです。

例えば、下に 500 ドットはみ出している場合、-h=500 とします。500 より大きい値でもかまいません。左、または上にはみ出ているときは、それぞれ、-X, -Y を用いれば OK です。

ページサイズからはみ出た部分は、何も指定しないと欠けて出力されます。

8.2.5　印字範囲の自動調整　[オプション-M]

トグルスイッチ、

```
-M[+|-]
```

のオプションを ON にすると、-W, -H で指定された用紙の印字可能範囲いっぱいになるよう自動的に-w, -h が調整されます。

8.2.6　センタリング　[オプション-C]

dviprt でトグルスイッチ-C[+|-] を ON にすると、印字すべき部分が紙の中央に来るように左と上のマージンを決めてくれます。これを有効にするには、用紙の横幅と縦幅が指定されていなければなりません (cf. オプション-W -H)。-W と-H の一方のみが指定されているときは、そちらのみが調整されます (横幅や縦幅の指定を解除するには、-1 という値をいれます)。

-X と-Y の値を適当に決めて、実際に用紙の中央に来るように調整しておけば、同じ状況でそのプリンタを使う限り、同一の-W -H -X -Y の値で、異なったサイズの原稿 (magstep や、dpi が異なってもよい) でも用紙が同じなら、いつも中央に来るように印字されます。-w と-h の値は影響しません。

dviout でこの-C を指定しているときは、-X, -Y, -w, -h の値は、枠に対する本文の表示位置を変えません。この場合は、-X, -Y に応じた値を-P, -Q で指定することによって、dviprt での印刷イメージを知ることができます (cf. 第 9 章 p.73)。

第9章 dviout の操作

9.1 キー操作

キー操作	機能
[H], [HELP]	キー操作の一覧表示
[→] [←] [↑] [↓]	上下左右にスクロール
[SHIFT] + [→] [←] [↑] [↓]	半画面切替
[SPACE]	次の部分へ*1
[BS]	前の部分へ*1
[P], [B]	前ページへ*2
[N]	次ページへ*2
[G], [M]	指定ページへ*3
[RETURN]	起動時のパラメータで与えた次のブロックへ*4
[V]	1/4 もしくは 1/8 縮小イメージ表示*5
[C]	1/8 縮小イメージ表示
[R]	白黒反転*6
[ESC] ,[Q]	終了

表 9.1: dviout のキー操作

dviout のキー操作を 9.1 にまとめておきました。以下は、 9.1 の補足説明です。説明番号は、表中の番号に対応します。

∗1 ページ分割されている場合は前/次の分割部分、分割されていない場合は前/次のページに移ります。
∗2 このキーに先だって数字を入力しておくと、その数だけページをジャンプします。3 桁以上の数字を入力した場合は、最後の 3 桁が有効です。ページを移ると、そのページの先頭を表示します。
∗3 指定ページ番号を入力してから、このキーを押してください。3 桁以上の数字を入力した場合は、最後の 3 桁が有効です。ページを移ると、そのページの先頭を表示します。
∗4 次のブロック (cf. 第 6 章 p.56) の指定がないときは終了します。
∗5 もとの画面に戻るには、任意のキー ([X] のように機能の無いキーの方が無難) を押してください。イメージ表示中にさらに [V] を押すと、以前のイメージ画面を見ることができます。また、2 ページをイメージ表示中に [M] を押すと、右側にイメージ表示されたページにジャンプします。
∗6 -R= オプションにて、白黒反転、または画面の色設定ができます。この色設定をしている場合は、そこで設定された色と通常の画面との切り替えを行ないます (GA-1280A/1024A 版では、背景色と前景色が切り替わります)。

9.2 スクロール/キー反応のスピード 調整 [オプション -t, -sh, -s, -u]

9.2 p.72 に示すように、パラメータに指定する数 (num) を大きくするに従って、スクロールスピードなどが遅くなります。

どのオプションも 0 以上の数 (5000 以下が許されますが、2 から 200 程度が適当でしょう) が設定できま

-t=*num*	縦方向スクロール	低速化 (デフォルト 0)
-sh=*num*	横方向スクロール	低速化 (デフォルト 0)
-s=*num*	半画面スクロール	低速化 (デフォルト 0)
-u=*num*	ページ切り替え	低速化 (デフォルト 0)

表 9.2: スクロール・スピードの調整

すが、-t のみ負の値 (−16 以上) も許されます (GA-1280A/1024A 版では、-t は −128 以上、-sh は −16 以上が指定可能)。使用するコンピュータの速度に応じて適当な値を設定してください。

なお、オプション-sh は [R] による画面反転のキー反応速度調整、オプション-u はイメージ表示画面やヘルプ画面表示後の次のキーを受けつけるまでの時間調整と連動しています。

9.3 イメージ表示

プレビューアで表示中、[V] を押すと、そのページを 1/4、または 1/8 に縮小したイメージ表示が得られます。もとの表示に戻すには、どれか他のキーを押します。ただし、1 ページが分割されているとき (つまり、ページの表示が [2-1] のようになっているとき) は、この機能は使えません。また、1/8 に縮小しても画面に入りきらないときにも、この機能は使えません。1/4 の縮小 1 ページが画面に入りきれば、1/4 縮小イメージ表示になります。この場合でも、[V] の代わりに [C] を押すことにより、1/8 に縮小した印刷イメージ表示が得られます。

イメージ表示の 1 ページが画面の半分の幅に納まれば、2 回目以降のイメージ表示のときに、以前のイメージ表示の部分と合わせて、2 ページが同時にイメージ表示されます。これは、1/4 表示か 1/8 表示のどちらか一方で、1/4 で 2 ページ表示が可能なときは 1/4 表示でのみ行なえます。

イメージ画面が表示されている状態で、さらに [V] を押すと、より以前のイメージ表示を見ることができます。保存して見ることができるページ数は、PC-9801 の場合、デフォルト状態で 6 ページです。ただし、裏 GVRAM を使わないときは、この数は少なくなります。また、2 ページ同時表示状態で [M] を押すと、右側に表示されたページにジャンプすることができます。

オプション-bv でページ数を指定すると、最大 16 ページまで保存できます (cf. 第 5 章 p.53)。PC-9801 以外の機種のデフォルトは、EMS を使用する場合 1 ページ、使用しない場合はページ保存を行なわないようになっています。

9.3.1 枠の表示

イメージ表示画面では、その時の状態に応じて枠が表示されます。この枠のサイズや表示される条件は、新サイズオプションを使う場合 (-newsize+) と使わない場合 (-newsize-) では異なります。

9.3.1.1 新サイズオプションを使用した場合

枠は表示/印字可能範囲を表わします。-PW, -PH 以外の、-LM, -TM などを設定しなければ、それらはデフォルトの0mm で、枠は用紙サイズとなり、用紙に印字されるべき位置に正しく表示されます。一方、-LM, -TM, -RM, -BM などもdviprt と同じ長さに設定すれば、枠は実際の印字可能領域を表わし、印字すべき部分が印字可能領域に入っているかどうかのチェックができます。

9.3.1.2 新サイズオプションを使わなかった場合

-W -H で、用紙の幅と高さを、プレビューアの解像度に合わせて設定しておけば、枠が表示され、印刷イメージがわかります。デフォルトでは-W=944 -H=1298 となっていて、118dpi のとき横 8inch、縦 11inch に対応します。

オプション-P, -Q は、オプション-C を指定したときのみ意味を持ちます。dviprt で-C を指定しているとき、dviout でも-C を指定し、dviprt の-X と-Y に応じた値を-P, -Q に設定すれば、用紙に対する印刷イメージがわかります。このときのdviout の-X と-Y の値は、左、または上が欠けているとき (枠からはみ出ているという意味ではありません) に設定します。dviout で-C を指定しているときは、-X -Y -w -h の値は、枠に対する本文の表示位置を変えません。

ただし、GA-1280A/1024A 版では、新サイズオプションを使用しないときには枠を表示しません。

9.3.2 イメージ表示の詳細

PC-9801 ノーマル版では [V], [C] によるレイアウト表示の仕様は以下のようになっています。

まず、状態を以下のように定義します。

- **(0-0)** 1 ページが split しているか、1/8 × 1/8 縮小イメージが画面に入らない
- **(4-4)** (0-0) でなく、左半画面に 1/4 × 1/4 縮小イメージが入る
- **(4-8)** (0-0) でなく、画面に 1/4 × 1/4 縮小イメージが入るが、左半画面には入らない
- **(8-8)** (0-0), (4-4), (4-8) でなく、1/8 × 1/8 縮小イメージが左半画面に入る
- **(8-0)** (0-0), (4-4), (4-8) でなく、1/8 × 1/8 縮小イメージが全画面に入るが半画面に入らない

このとき、

[V]
- **(4-4)** 1/4 × 1/4 縮小イメージ表示、以前に保存したものがあれば画面右側に表示し、現在の縮小イメージを保存
- **(4-8)** 1/4 × 1/4 縮小イメージ表示
- **(8-8)** 1/8 × 1/8 縮小イメージ表示、以前に保存したものがあれば画面右側に表示し、現在の縮小イメージを保存
- **(8-0)** 1/8 × 1/8 縮小イメージ表示
- **(0-0)** 無視 (縮小イメージ表示不可)

[C]
- **(4-4)** 1/8 × 1/8 縮小イメージ表示
- **(4-8)** 1/8 × 1/8 縮小イメージ表示、以前に保存したものがあれば画面右側に表示し、現在の縮小イメージを保存
- **(8-8)** 1/8 × 1/8 縮小イメージ表示、以前に保存したものがあれば画面右側に表示し、現在の縮小イメージを保存
- **(8-0)** 1/8 × 1/8 縮小イメージ表示
- **(0-0)** 無視 (縮小イメージ表示不可)

となります。

以前に保存したものが画面右側に表示されている状態で、さらに [V] を押すと、画面右側の表示がより以前に保存されたものに変わります。

9.3.3 イメージ表示の解像度による違い

使用する機種によってデフォルトで保存するページ数が異なるほか、解像度によって表示の縮小率が異なります。

PC-9801 のハイレゾや GA-1280A/1024A 版、DOS/V 版 Super VGA などのハイレゾモードでは、[C], [V] キーによるイメージ表示の縮小率は、1/8, 1/4 から、それぞれ 1/4, 1/2 になります。

9.3.4 GA-1280A/1024A 版

イメージ表示中に [K] を押すと、イメージ表示専用モードになります。また、[S] を押すと、現在表示中のページを MS Windows 標準の BMP ファイル形式で出力することができます (cf. 本章 p.75)。

イメージ表示専用モードのときは、(イメージ表示でない) 通常の表示のときと同様、[V] や [C] などの他、数字やページ移動のキーが有効で、イメージ表示画面のページ移動ができます。ただし、以下の相違があります。

- 上下左右にスクロールするためのキーは無視されます。
- 画面左上のページ番号の右側に -> が表示されているときは、[V] は -> が表示されていないイメージ表示への移行のキーを意味し、また [S] と [K] は無視されます。
- 画面左上のページ番号の右側に -> が表示されていないときは、[V] は可能なら右側に表示されている以前のイメージ表示 (これは Gray Scale とはならない) のスクロールになります。[K] は、イメージ表示専用モードと通常のモードとのトグルスイッチとして作用します。機能が割り当てられていないキーを押すことにより-> を表示させることができます。

9.3.4.1 イメージ表示専用モード [オプション-view]

デフォルトをイメージ表示専用モードとするには、

```
-view+
```

とオプションで指定します。

9.4 画面モード・解像度の指定 [オプション-reso]

GA-1280A/1024A 版, J-3100 版, DOS/V 版では、解像度などの画面モードを指定できます。

9.4.1 GA-1280A/1024A 版

アイ・オー・データ機器製の PC-9801 用グラフィック・アクセラレータ GA-1280A/1024A に対応し、高解像度の表示を行ないます。この版は、PC-9801 の元来のグラフィック画面には対応していません。

また、1/2 倍の縮小イメージ表示画面において、Gray Scale (白黒の明るさの階調表示を利用して、縮小画面を見やすくする機能) をサポートしています。

```
-reso=n
```

n= **0** デフォルト、すなわちgainitで指定した解像度で 2 色。MS Windows の DOS Windows から起動した場合は、Windows での解像度を用います。

-1 上記と同じですが Gray Scale 表示をサポートします。

8 640 × 480 2 色

4 640 × 480 Gray Scale

12 800 × 600 2 色

10 800 × 600 Gray Scale

7 1024 × 768 2 色

3 1024 × 768 Gray Scale

19 1280 × 960 2 色 (GA-1280A のみ)

17 1280 × 960 Gray Scale(GA-1280A のみ)

23 1280 × 1024 2 色 (GA-1280A のみ)

21 1280 × 1024 Gray Scale(GA-1280A のみ)

1600 × 1024 ドットのモードはサポートしていません。

使用する場合にはグラフィック・アクセラレータ付属のインタフェースプログラムgainit.exeをあらかじめ常駐させておきます。

-reso=に指定する値はアイ・オー・データ機器のgalib.docに記述のある関数GAinit()の引数と同じものです。上記の Gray Scale で指定する値は、gainit.exeで 16 色モードを指定する値に対応しています。実際の色は、-R=で指定できます (cf. 本章 p.80)。

また、従来のdviout に比べて同一のパラメータではスクロール速度が遅くなるので-t=の値は −128 まで、-sh=の値は −16 までサポートしています。大きな負の値にするほど、縦、および横のスクロール速度が速くなります。

なお、NDHD-2 (ノーマルディスプレイ使用) のモードでは、-reso=で解像度を変更できません。

デフォルトでは、起動前と同一解像度の 2 色モードになりますが、-reso=で強制的に他の多色モードにできます。また、gainitで/w=F0と指定した場合や、ROM エリアに空きがない場合は、GVRAM のウィンドウアドレスが 0xf00000 になります (これは、dviout 終了時に seg: F000 と表示されることで確認できます)。

多色モード、あるいはウィンドウアドレスが 0xf00000 の場合は、dviout の表示/スクロール速度が遅くなります。

-reso=の値を 0 または −1 以外に指定して解像度をgainitでの設定と変えると、interlaced 表示になってしまうようです。

9.4.1.1 BMP ファイル出力 [オプション-bmp]

縮小イメージ表示中に [S] キーを押すことにより、

-bmp=*file_name*

で指定したファイルに、Microsoft Windows で使われている BMP ファイル形式で、現在のページ内容を出力できます。ページが分割されているときは、縮小イメージ表示にすることはできず、BMP ファイルへの出力もできません。*file_name* のデフォルトは “tmp%04d.bmp” で、%04d はページ番号を表わす 4 桁の数字で置き換えられます (桁数が 4 桁に満たないときには左側に 0 を並べて 4 桁とします)。コマンドライン

上で指定するには

dvioout "-bmp=tmp%04d.bmp" *dvi_file*

のように指定します (「%」が command.com に解釈されないように、括っておく必要があります)。C における printf(*file_name*, *page*) の出力の形ですので、パスを指定する \ は \\ のように書き、例えば、次のようになります。

例 :

```
-bmp=b:\\bmp\\temp.bmp
"-bmp=b:\\bmp\\tmp.%03d"
```

9.4.2 J-3100 版

表示解像度の指定ができます。

```
-reso=n
```

n=**0** DCGA 640 × 400 (デフォルト)
1 VGA 640 × 480

9.4.3 DOS/V 版

DOS/V 版 dvioutは VGA の 640 × 480 に加えて、Super VGA の 800 × 600、1024 × 768、1280 × 1024 の解像度で表示が可能です。

dvioutはビデオカードに搭載されているビデオ BIOS を利用して画面モードを設定します。dvioutで使用できる画面モードを 9.3 に示します。

メーカー	コントローラ	水平解像度	垂直解像度	色数	画面モード番号
IBM	VGA	640	480	16	0x12
TsengLab	ET4000	800	600	16	0x29
WD/Paradise	WD90C00	800	600	16	0x58
Trident	T8900	800	600	16	0x5B
S3	86C80X,9XX	800	600	16	0x6A
ATI	MACH32	800	600	16	0x54
VESA		800	600	16	0x102
TsengLab	ET4000	1024	768	16	0x37
WD/Paradise	WD90C00	1024	768	16	0x5D
Trident	T8900	1024	768	16	0x5F
S3	86C80X,9XX	1024	768	16	0x104
ATI	MACH32	1024	768	16	0x55
VESA		1024	768	16	0x104
VESA		1280	1024	16	0x106

表 9.3: DOS/V 版dviout で使用される画面モード

1024 × 768 および 1280 × 1024 の表示解像度では [C], [V] キーによるイメージ表示の縮小率は、それぞれ 1/8, 1/4 から 1/4, 1/2 に変更されています。

DOS/V 版特有の起動オプションには、以下のものがあります。

9.4.3.1 ビデオコントローラの指定 [オプション-chip]

```
-chip=n
```

n=**0** VGA (デフォルト)
1 ET4000
2 86C80X,9XX
3 T8900
4 WD90C00
5 WD100X
6 MACH32
9 VESA 対応動作

VRAM のバンク切換やハードウェアスクロールの制御方法は Super VGA コントローラによって異なります。これらを正常に行なわせる為、Super VGA カードで dviout を実行する場合は使用されているビデオコントローラに合わせて本オプションを必ず指定してください。

VESA BIOS Extension(VBE) に準拠したビデオカードであれば、VESA 対応動作 (-chip=9) を指定する事でコントローラの種類を問わず dviout を動作させる事ができます。

上記のパラメータリストに無いコントローラでは、VGA(-chip=0) を指定してください。この場合、1024 × 768 以上の解像度での表示やハードウェアスクロール機能 (コントローラに依る) が使用できない等の制限があります。

MACH32(-chip=6) を指定した場合と VESA 対応動作 (-chip=9) の場合には、強制的にソフトウェアスクロールが使われます。これは、MACH32 のハードウェアスクロール機能に未対応なのと、VBE に dviout で利用可能なハードウェアスクロールのファンクションが用意されていない為です。

dviout はページ番号およびヘルプを DOS 経由で出力していますが、実装されているビデオ BIOS によっては文字の表示が不完全になったり、全く表示されない場合があります。

9.4.3.2 表示解像度の指定

```
-reso=n
```

n=**0** 640 × 480 (デフォルト)
1 800 × 600
2 1024 × 768
3 1280 × 1024

1280×1024 の指定は、VESA 対応動作時のみ有効です。また、VESA 対応動作を指定した場合、640×480 は無効になります。

9.4.3.3 ソフトウェアスクロールの指定 [オプション-sws]

```
-sws[+|-]
```

dviout は、通常ハードウェアスクロールを行ないますが、本オプションを指定するとソフトウェアスクロールを行ないます。-chip オプションを指定してもスクロールが正常に行なえない場合は、本オプションを指定してください。

9.4.3.4 画面モード番号の直接指定 [DOS/V 版dviout でのオプション-E]

```
-E=mode (mode= 画面モード番号)
```

通常は-reso および-chip オプションからコントローラ・メーカー標準の画面モード番号を判定しますが、「私のビデオカードは ET4000 だけど、1024 × 768 モードの番号は0x66 だよ。」という場合、オプションで

```
-chip=1 -reso=2 -E=0x66
```

と強制指定できます。

具体的な数字 (*mode*) については、お手持ちのグラフィックカードのマニュアルをご覧ください。

なお、数字は普通は 10 進数で指定しますが、先頭に「0x」を付けることにより、16 進数で指定することもできます。

Diamond Stealth ボードでは、画面の初期化がうまくいかないため、これを解決するための常駐プログラムfix104.com (hero.h 氏作) が、fix104.lzh の中に提供されています。必要な方はご利用ください。

また、標準の$disp.sys の代わりに、h.murata 氏の HiText ドライバー$disph.sys を組み込んである場合は、-E0 と指定するだけで構いません。

9.4.3.5 使用可能な画面モードオプションの組み合わせ

現在dviout で使用可能な-chip, -reso, -E, -sws の組み合わせは、 9.4 p.79 の通りです。

9.4.4 HP100LX 版

HP100LX 版 dviout は、画面モードは CGA で、解像度は 640 × 200 に固定されています。

9.5 画面反転・画面カラー [dviout でのオプション-R]

画面の白黒を反転するには、

```
-R
```

と指定します (-R+ でも同様です。一方-R- により、反転を行なわないように指定できます)。

また、

```
-R= 色指定
```

コントローラ	水平解像度	垂直解像度	-chip	-reso	-E	-sws
VGA	640	480	0	0	0x12	+/-
SVGA 非特定	800	600	0	1	0x6A	+/-
ET4000	640	480	1	0	0x12	-
ET4000	800	600	1	1	0x29	+/-
ET4000	1024	768	1	2	0x37	+/-
86C80X/9XX	800	600	2	1	0x6A	+/-
86C80X/9XX	1024	768	2	2	0x104	+/-
T8900	800	600	3	1	0x5B	+/-
T8900	1024	768	3	2	0x5F	+/-
WD90CXX	800	600	4	1	0x58	+/-
WD90CXX	1024	768	4	2	0x5D	+/-
WD100X	800	600	5	1	0x58	+/-
WD100X	1024	768	5	2	0x5D	+/-
MACH8/32	800	600	6	1	0x54	+
MACH8/32	1024	768	6	2	0x55	+
VESA	800	600	9	1	0x102	+
VESA	1024	768	9	2	0x104	+
VESA	1280	1024	9	3	0x106	+

表 9.4: 使用可能な画面モード オプションの組み合わせ

によって画面カラーの設定ができます。色指定の仕方は、以下に示すように機種によって異なります。

9.5.1 PC-9801 版 (アナログディスプレイ使用の場合)

以下のように設定します。

```
-R=<RGB><rgb>
```

前景色

R 赤輝度レベル (0〜F)
G 緑 〃
B 青 〃

背景色

r 赤輝度レベル (0〜F)
g 緑 〃
b 青 〃

色指定は前景色、背景色とも必須で省略することはできません。-R= の次に 3 原色の強度を指定する 6 桁の 16 進数がなければ無視され (このときは、-R を指定したとみなされます)、0-9, a-f, A-F 以外の文字は 0 と見なされます。

例 :

-R=f82132 文字色の強度は赤 15、緑 8、青 2（オレンジになる）
背景色の強度は赤 1、緑 3、青 2（深緑になる）
-R=000aaa 明るいグレーバックのリバース
-R=000fff 通常のリバース

-R+ による画面の白黒反転では、画面上のページ番号表示がほとんど見えなくなって不便ですので、上記のようにグレーバックのリバースなどを用いるとよいでしょう (最適な値は、ディスプレイによっても異なります)。

9.5.2 GA-1280A/1024A 版

```
-R=<RRGGBB><rrggbb>
```

前景色

RR 赤輝度レベル (00〜FF)
GG 緑 〃
BB 青 〃

背景色

rr 赤輝度レベル (00〜FF)
gg 緑 〃
bb 青 〃

または、

```
-R=<RRGGBB><rrggbb><RRGGBB><RRGGBB><RRGGBB><RRGGBB><RRGGBB>
```

と設定します。

後者の形式の最初の部分は、前者の形式の前景色と背景色の指定と同じです。Gray Scale 表示は 5 段階階調で表示されますが、後者の残り 5 つの<RRGGBB> で、その色調を指定できます。色指定をした場合でも、[R] キーによる反転トグルは有効です。

また、輝度を表わす 16 進 2 桁の文字間に 16 進を表わす文字以外の区切り文字を入れることができます。例えば、

例 :

```
-R=00000;f0f0f0
```

と指定します。デフォルトは

```
-R=ffffff;000000;ffffff;c0c0c0;808080;404040;000000
```

となっています。

9.5.3 J-3100 版, DOS/V 版

以下のように設定します。

```
-R=<RRGGBB><rrggbb>
```

前景色

RR 赤輝度レベル (00〜3F)
GG 緑 〃
BB 青 〃

背景色

rr 赤輝度レベル (00〜3F)
gg 緑 〃
bb 青 〃

例 :

-R=0000103F3F3F 白地に暗めの青で表示

色指定をした場合、キーによる反転トグルは無効になります。色指定をしなかった (-R のみ) 場合は、通常通り画面の白黒反転を行ないます。

J-3100 版では、本オプションは-reso=1 を指定した場合にのみ有効となります。

9.5.4 AX 版

-R 起動オプション、[R] 操作キーによるdviout の画面反転機能はサポート されていません。

第10章 dviprtでの印字

10.1 プリンタの指定

10.1.1 標準対応プリンタの指定

各種のプリンタをサポートしていますが、利用するプリンタに合わせ、正しくオプションを指定する必要があります。-p オプションで直接対応しているプリンタは 10.1 の通りです。

-p=e : ESC/P 系の 24pin プリンタ (デフォルト)
-p=p : PC-PR 系の 24pin プリンタ
-p=n : NM 系の 24pin プリンタ
-p=l : Canon LIPS III レーザー・ビーム・プリンタ
-p=m : EPSON ESC/Page レーザー・ビーム・プリンタ

表 10.1: -p オプションで直接対応しているプリンタ

直接対応していないプリンタは、-p=o でそのプリンタの定義ファイルを指定します。
レーザー・ビーム・プリンタについては、本章 p.87 を参照してください。また、-p= の指定を複数回行なった場合には、-p=o による指定も含めて、最後に指定したものが有効になります。

10.1.1.1 その他のプリンタへの対応

初期化に特別なコードが必要なプリンタや、特殊なプリンタのときは、そのコントロールの仕方を書いたプリンタ定義ファイルを作って、

-p=o*config_file_name*

のようにパラメータで指定して、それを読み込むことで対応しています。

プリンタ定義ファイルの作成

ここで用いるプリンタ定義ファイルは、決められた形式のソースファイルを書き、optcfg.exe で変換することにより作成することができます。例えばescp_24.src は、EPSON ESC/P の 24pin プリンタ用のプリンタ定義のソースファイルで、

```
A>optcfg escp_24
```

とすると、プリンタ定義ファイルescp_24.cfg が作成されます。この詳細については、プリンタ定義ファイルの記述 (第 11 章 p.95) を参照してください。また、-O オプション (cf. 本章 p.93) を使えば、ファイルに対してdviprt の出力を行なうことができますので、プリンタ定義ファイルのデバッグに役立つでしょう。
なお、

```
A>optcfg -r escp_24
```

とすると、escp_24.cfg がソーステキストに逆変換され、その結果を標準出力に出力します。標準出力の代わりにesc_24.src といったファイルに落とすには、

```
A>optcfg -r escp_24 > escp_24.src
```

などとします。

プリンタ定義ファイルの指定

プリンタ定義ファイル c:\tex\driver\escp_48.cfg は、-p=oc:\tex\driver\escp_48 のようにパラメータで指定してください。最後の .cfg は省略して構いません。

また、プリンタ定義ファイルを環境変数TEXCFG で指定したディレクトリに置いている場合であれば、-p=oepson_16 というようにドライブ名とディレクトリ名を省略することが可能です。ドライブ名とディレクトリ名を両方とも省略した場合には、環境変数TEXCFG に設定されたディレクトリで指定した名前のプリンタ定義ファイルを探し、みつからなければ、環境変数PATH に設定されたディレクトリも探します。

単に、-p=o と指定したときは、TEXCFG, PATH に設定されたディレクトリでprtctl.cfg というファイルが探されます。

プリンタ定義ファイルが起動時に正しく読み込まれると、そのソースファイルのname の所に書いたメッセージが表示されます。

10.1.2 初期化コード の追加 [オプション-I]

プリンタに最初に送る標準のコードの直後に追加して送るコードを、ファイルに書いておくことも可能です。プリンタ定義ファイルに書いた場合と異なり、コンピュータ内部のメモリーを消費しないので長大なコードを送ることができます。

-I=*init_code_file*

によって、そのファイル名 *init_code_file* を指定します。例えば、-p=l オプションで LIPS III 対応プリンタを使う場合では、飾り文字属性設定命令や、コメント表示命令などに使うことが考えられます。送出するコードは *init_code_file* の中に記述します。

ファイル中の通常の (アスキーコードが 16 進数で 20 のスペースより大きな) 文字はそのまま送られ、スペース、タブ、改行などは無視されます。

ただし「\」は次のような特別の意味をもちます。

- ∗1 \\ これ以降その行の最後まで無視される (コメント等を書くときに使います)
- ∗2 \w 出力のページ幅 (dot 単位)
- ∗3 \W 出力のページ幅 (byte 単位)
- ∗4 \h 出力のページ長 (dot 単位)
- ∗5 \H 出力のページ長 (byte 単位)
- ∗6 \r 横方向解像度 (dpi)
- ∗7 \R 縦方向解像度 (DPI)
- ∗8 \p 通しページ番号
- ∗9 \f DVI ファイルのファイル名
- ∗10 \?? 上の 1–9 以外では、続く 2 文字で 16 進数を表わすとみなされ、そのコードがそのまま送られる (例\0D \0A)

この 2 から 8 までは、例えば、\rb のようにその直後にオプション文字b B H のいずれか 1 つをつけることができ、それぞれ binary で下位上位の順、binary で上位下位の順、4 桁の 16 進数、の形式ででその数を出力します。それ以外のときは、通常の 10 進数で出力します。

\p は、-O=*file_name* (cf. 本章 p.93) で、ページ毎に異なったファイルに出力する場合に意味を持ちます。

10.1.3 raw PBM 形式の出力

例えば、 10.1 p.84 のような内容のファイルdvi2pbm.ini は、raw PBM 形式のヘッダ用です (cf. 第 12 章 p.123)。

```
P 4 \0A
#Image\20 generated\20 by\20 dviprt\20 (\f:\r x \R )\0A
\w \20 \h \0A
```

ファイル例 10.1: dvi2pbm.ini

```
name                : PBM image format
upper_position      : LEFT_IS_HIGH
pins                : 8
minimal_unit        : 10000
maximal_unit        : 10000
bit_image_mode      :
normal_mode         :
send_bit_image      :
after_bit_image     :
skip_spaces         :
line_feed           :
form_feed           :
dpi                 : 118
```

ファイル例 10.2: dvi2pbm.src

10.2 の内容のdvi2pbm.src から生成したdvi2pbm.cfg を-p=o オプションで指定し、コマンドラインで

```
dviprt -p=odvi2pbm -I=dvi2pbm.ini -O=output_file ... dvi_file page
```

とすれば、指定したページの raw PBM 形式のビットイメージ出力が得られます。複数ページを出力するには-O で、ページ毎に異なるファイルに出力する指定を用いるとよいでしょう (cf. 本章 p.93)。

付属のdvi2pbm.par を用いれば

```
dviprt -=dvi2pbm -O=output_file ... dvi_file page
```

とすることができます。118dpi 以外にするときは、-dpi を指定します。

また、新仕様のdvi2pbm.cfg は、dvi2pbm.ini の機能を兼ねたものなので後者を指定する必要はありません (指定してはいけない)。

ビットイメージ出力のために切り取るサイズを-PW=, -PH= で指定し (必要なら、-LM=0mm -TM=0mm とし)、切り取る左上端の位置を-OX= -OY= で調整します。横サイズ、縦サイズは 8dot 単位に切り上げられます。

raw PBM 形式のかわりに monochrome GIF 形式の出力を得る方法は、次々項の「G3 FAX 形式」の最後をご覧ください。

10.1.4 EPS 形式の出力

raw PBM 形式の出力と同様、Encapsulated PostScript(EPS) 形式で出力することができます。実際のビットイメージは、16 進数表記のテキスト形式となり、EPS フォーマットに従ったヘッダがつきます。

そのためには、付属のepsimage.cfgを用います。使い方はepsimage.srcをご覧ください。

10.1.5 G3 FAX形式の出力 [オプション-FAX]

オプション-FAXによって、G3 FAXのデータ圧縮形式 (T.4勧告1次元符号化方式) での出力を行なうことができます。-FAXを指定すると、デフォルトで-dpi=200となりA4版のfine modeで、MEGASOFTのStarfaxに対応するヘッダを付けてFAX Modemで転送可能な形でデータを出力します。この出力先はオプション-O=(cf. 本章p.93)によってディスク上のファイルにすることができます。

```
-FAX=[Fwidth:[max_height]]
```

*width, max_height*は縦と横のドット数で、これを超える部分はカットして出力されます。ページの幅(-PW=などで指定したもの)がこれより小さい場合は、横方向にセンタリングされます。一方、ページの縦の長さがここで指定したものに満たなくても長さを合わせるための空の行の出力は行ないません。よって、*max_height*を十分長く取っておけば、ページの長さの分だけが出力されます。

デフォルトは、*width*=1728, *max_height*=2290です。*width*=2048とした場合、デフォルトではB4版のfine modeでのMEGASOFTのStarfaxに対応するヘッダがつけられます。

各ページの出力は、FAXの規格にあった形でなされますが、ページの切れ目やヘッダ、フッタに付けるものは、-p=o*config_file_name*や、-I=*init_code_file*によって自由に変更できます。これらのデフォルトは付属のstarfax.srcにあるもので10.3のようになっています。bit_image_modeにはヘッダとして出力されるコードが書かれています。また、10.3のform_feedとnormal_modeに定義されている9 byteの文字列は、G3 FAXの規格によるページの終わりを表わすコード (RTC: ReTurn to Control) です。

```
name            : Starfax format
upper_position  : LEFT_IS_HIGH
pins            : 8
minimal_unit    : 10000
maximal_unit    : 10000
encode          : FAX
bit_image_mode  :   SF \x01 \x00 \x00 \x00 \x00 \x00
                  \x00 \x40 \x00 \x00 \x00 \x00 \x00 \x00
normal_mode     : \x00 \x08 \x80 \x00 \x08 \x80 \x00 \x08 \x80
send_bit_image  :
after_bit_image :
skip_spaces     :
line_feed       :
form_feed       : \x00 \x08 \x80 \x00 \x08 \x80 \x00 \x08 \x80
dpi             : 200
```

ファイル例 10.3: starfax.src

この-FAX=を指定した場合には、upper_position , minimal_unit , maximal_unit , send_bit_image , after_bit_image , skip_spaces , line_feedの設定は無視されます (upper_position, minimal_unit, maximal_unitについては、省略できませんので、上記のstarfax.srcのように適当な値を記述します)。

200dpi系のフォントがない場合は、-dpi=180として180dpiのフォントを使うか、さらに-mag=1を指定して216dpiあるいは、-halfを指定して197dpiとすると、180dpiシリーズのフォントが使用できます。

また、付属のg3fax.cfgは、ヘッダなしに出力するもので、例えば

```
dviprt -dpi180 -p=og3fax -O=file_name.!!! dvifile
```

とすれば、-FAX= のフラグが ON となり、G3 FAX のデータがページ毎に別のファイルとして出力されます。

なお、逆に G3 FAX 形式の圧縮ファイルを dviout/dviprt で扱うには、付属の fax2pbm.exe を用います。詳しくは fax2pbm.doc をご覧ください。

この fax2pbm.exe を用いると、dviprt が G3 FAX 形式で出力したファイルを monochrome GIF 形式のファイルに変換できますので、DVI ファイルの各ページを monochrome GIF ファイルにすることができます。それには、dviprt には -FAX=F*width* オプションを、fax2pbm.exe には、-i -g オプションを指定してください。

10.1.6 データ圧縮出力 [オプション-comp]

G3 FAX 形式の圧縮や LIPS III 出力での圧縮 (cf. 本章 p.90) がありますが、さらに、HP DeskJet などに採用されている PCL データ圧縮のうち mode 1 と mode 2 に対応しています。この圧縮は

```
-comp=n
```

n=**0** 無圧縮
1 PCL mode 1 圧縮
2 PCL mode 2 圧縮

により指定します。

1 ラスタ行単位で圧縮、出力を行なうため、LEFT_IS_HIGH 型、LEFT_IS_LOW 型のプリンタ (cf. 96) にしか対応していません。

mode 1 圧縮は、単純なバイト単位の run length 圧縮で、mode 2 圧縮は、run length 圧縮と無圧縮のうち効率のいいほうを選ぶものです。

HP DeskJet の場合、圧縮後のデータサイズをプリンタに送る必要がありますが、これは新仕様のプリンタ定義ファイルを用いることによって可能となっています。また、この新しいプリンタ定義ファイルには、この -comp オプションの指定に相当する記述を入れることが可能です (encode)。

DJ505J をご使用の方は、付属のプリンタ定義ファイルが利用できます。このファイルは新しい仕様に従ったプリンタ定義ファイルで、mode 1 圧縮用が dj505j1.cfg、mode 2 圧縮用が dj505j2.cfg です。

10.1.7 機種の指定 [-hard]

PC-9801, DOS/V, J-3100 用の統合版 dviprt では、以下のようにして使用機種を指定します。通常（標準エラー出力がリダイレクトされていないとき）はこの指定がなくても、いずれの機種であるか自動的に判断されます。

```
-hard=n
```

n=**0** PC-9801
1 DOS/V
2 J-3100

10.1.8 PC-9801 ハイレゾモードでの変更点

PC-9801 ハイレゾモードではフルセントロニクスプリンタ・インターフェイスに対応しています。プリンタの電源 OFF や不接続、OFF LINE、ペーパーエラーを dviprt が検出すると、警告メッセージを出力して印刷を中止するかを聞いてきます。中止しない場合、プリンタに起きている問題点を取り除くと自動的にそれを検出して印刷を再開します。

10.1.9 プリンタ I/O ポートの指定 [DOS/V 版 dviprt でのオプション -P]

```
-P=[B]n
```

このオプションを指定すると、dviprt は論理デバイス LPT1: から LPT:4 を指定して、BIOS を経由せずにプリンタ出力を行ないます。

DOS/V では、システムに付属するプリンタドライバ (\$PRNESCP.SYS 等) を組み込んである場合、プリンタ BIOS を使って出力すると、印字データが漢字コード変換されてしまって正しく出力することができません。これを避けるため、プリンタ I/O ポートに直接データを出力する本オプションを用意しました。これには、I/O のウェイトを指定するすることもできます。

例えば、-P=1 とすると LPT1: の論理デバイスに割り当てられた I/O ポートへ直接印字データを送出します。n は 0 から 4 までを指定でき、1 から 4 は LPT1: から LPT4: までに対応します。デフォルトは -P=0 で、この場合はプリンタ BIOS を使います。ウェイトの指定は、-P=1;10, -P=1;100 のように「;」の後の数字で行ないます。デフォルトは 0 で、ノーウェイトを意味します。なお、B を指定せずに本オプションを使った場合、ネットワークプリンタを使うことはできません。

ネットワーク上のプリンタに出力するときは、BIOS 経由を指定する B オプションを付加してください。例えば、-P=B3 とすると BIOS 経由で LPT3: に出力します。

10.2 レーザー・ビーム・プリンタでの印刷

オプションのパラメータ -p=l によって Canon の LIPS III を使用する LBP[1] に、-p=m によって EPSON の ESC/Page を使用する LBP にそれぞれ対応し、欧文フォントのダウンロード機能と和文スケーラブルフォント機能を使うことができます。これにより TEX の印字を高速に行なうことができます。

スケーラブルフォントは、デフォルトでは、「明朝体」と「角ゴシック体」を用いますが、LIPS III の場合には「角ゴシック体」が使用できる状態にない場合もあり、このときには和文フォントは全て「明朝体」となります。

パラメータ -p=l には、以下のようにサブパラメータ v, d, r, k, m, f, D, o, j, s, u, c, E, g を ; または : で区切って指定することができます (サブパラメータ g については第 4 章 p.29 をご覧ください)。

10.2.1 LIPS III の場合

デフォルトは、

```
-p=lv1000;m2;d512;j79;s1047;D300
```

で、新サイズオプションの下では、印字位置が TEX 本来のものに調整されます。

[1] 以下では、レーザービームプリンタを LBP と呼ぶ

新サイズオプションを使わない場合は

```
-X=240 -Y=114
```

という位置調整を行なうと、印字位置が TEX の本来のもの (すなわち、原点が用紙の左上端から、右、下にそれぞれ 1inch 入ったところ) になります。右や下が欠けて印字されてしまうときは、-w や -h に適当な数字 (例えば 10) を指定して、印字範囲を拡大しておきます (これを指定しても印字位置が変わってしまうことはありません)。なお、センタリングを行なうときは、-W と -H に正しい値が設定されていれば (デフォルトでは、A4 用紙に適合するように設定されている)、

```
-X=0 -Y=0 -C
```

とすればよいでしょう。

v サブパラメータv に続く数字で、和文スケーラブルフォントの縦横の比率を変えることができます。与えた数字の 1000 分の 1 倍に縦のサイズを拡大、または縮小します。デフォルトは 1000 ですが、200 以上 5000 までのみ許され、それ以外では 1000 と解釈されます。ただし、0 のみは特別で LBP 内蔵のスケーラブルフォントを用いないことを意味します。

d サブパラメータd に続く数字で、フォントのダウンロードの許容容量を Kbyte 単位で指定します。デフォルトは 512 です。0 はフォントのダウンロードを行なわないことを意味します。

サブパラメータk を指定しない場合、フォントのダウンロードは、1 バイト系の文字に対してのみ行なわれます (NTT 版 JTEX の和文フォントは 1 バイト系なので、オプション -ntt で 2 バイト系に変換しない限りはダウンロードの対象となります)。ダウンロードは、指定された印字対象範囲での使用総数が多いフォントから優先して、指定回数以上使用された文字に対し、許容容量内で行なわれます。ただし、ダウンロードされた欧文フォントセットの総数が、「32 から使用するスケーラブルフォントの総数を引いた数」に達すると、それ以上のダウンロードは行なわれません (このとき、コードが 128 以上の文字を含むフォントは 2 と数えます)。

r d で用いられるダウンロードサイズの計算は、デフォルトでは各フォントのセルサイズ (そのフォントに含まれるダウンロード文字の最大の横のドットサイズと最大の縦のドットサイズ) を基準に行ないますが、r の指定により、実際の転送データ量 (各文字毎のサイズを基準とする) に変更されます。セルサイズ基準の方が、大きな値となります。

k プリンタ内蔵の和文スケーラブルフォント以外の和文フォントを使う場合に、それを 1 バイト系の欧文フォントに混ぜてダウンロードすることを指定します。例えば、ゴシックに書体倶楽部のフォントを使用する場合などに有効ですが、プリンタ内蔵以外の和文フォントを使わない場合は指定しても無意味です。

このサブパラメータk を指定すると、フォントの変換テーブル用に EMS から 16K byte(それが存在しない場合は、Conventional memory から 16K byte) が使用されます。

また、k に続く 2 つの数字で和文フォントのダウンロードをコントロールできます。

k[=*max_char*[;*skip*]]

これらは、*max_char* (デフォルトは 63) より多くの文字をダウンロードした欧文フォント、および使用頻度の多いほうから数えて *skip*(デフォルトは 4) 番目以下の欧文フォントには、和文フォントを混ぜないことを意味します。後者、または、前者と後者の数字を省略すると、それぞれデフォルトの値となります。

特に、*max_char* に 0、あるいは *skip* に十分大きな数字を指定すると、欧文フォントをダウンロードした後、欧文フォントに埋めこまないで和文フォントのみをダウンロードすることになります。

フォントのダウンロードが行なわれると

```
<0:80><30:75><25:49><31:30+97>...
```

のように表示されます。< > 内の最初の数字は、ダウンロードされたフォントの番号 (`-f=1` オプションで表示される) で、次の数字はそのフォント内のダウンロードされた文字数です。「`+`」の後の数字は、ダウンロードされた和文フォントの文字数を表わします。

フォントのダウンロード機能を用いる場合、フォントキャッシュの指定`-r` は無効になります。

m サブパラメータ`m` では、指定した印字対象範囲で`m` に続けて指定した数以上出現した 1 バイト系の文字をダウンロードすることを意味します。この数のデフォルトは 2 です。ただしその文字の属するフォントセットの文字の出現総数が 4 未満の場合あるいは、縦または横のドットサイズが 128 を越える文字はダウンロードしません。

f サブパラメータ`f` でスケーラブルフォントの指定を行ないます。これは、複数指定可能で順に解釈されます。指定されなかったものはデフォルトの指定を使います。指定のフォーマットは以下のとおりです。

`f`*name*`=`*shotai*`[G][H][F][S][O][;`*vh_ratio*`[;`*weight*`[;`*dots*`;`*dots*`;...]]]`

name

TEX におけるフォント名から (後ろに数字 — 通常ポイント数 — があるときは) 後ろの数字を取り去ったもの (`^s` に代入されるもの) で、`min`, `goth` など

shotai

LIPS III における書体番号で、明朝体 80、角ゴシック体 81、丸ゴシック体 82、教科書体 83、楷書体 84 など

さらに、オプションとして、

- G : 擬似ゴシックを指定するとき指定 (1 dot 幅を増やす)
- H : 擬似ゴシックを指定するとき指定 (2 dot 幅を増やす)
- F : Fill pattern を指定するとき指定
- O : Outline font を指定するとき指定
- S : Shadow 印字を指定するとき指定

`G`, `H` は同時に指定できませんが、他は重複して指定できます。

vh_ratio

字体の縦横比の 1000 倍を指定。ただし、0 は、そこで指定されているフォントで *dots* に該当しないものに対し (1 は、そこで指定されているフォントで *dots* に該当するものに対し)、スケーラブルフォントを使わないことを意味します (この場合は、*shotai* などには適当な数を書いておきます)。それ以外は次以下に`f` の指定があればそれで判断され、なければデフォルトになります。

weight

デフォルト 22(明朝 22, 角ゴシック 22, 丸ゴシック 2)

dots

ここでの指定をスキップし次の指定を見る境界となるドットサイズを指定 (-f=1 で「( )」内に表示される数、cf. 第 4 章 p.27) *dots*-*dots* のように範囲による指定を含めることができます。

このプリンタ内蔵フォントの指定において、*shotai* の後の任意の「;」以降を省略することができます (デフォルトが用いられる)。例えば、

```
fgoth=82;1000;2;fmin=80;1000;22;16-24;32
fgoth=80H;1000;22;44;fgoth=80OS;2000
```

D サブパラメータD でプリンタの解像度を dpi 単位で指定します。-dpi は、使用するフォントの解像度なので、例えば、-dpi=240 -p=lD300 とすると、80%に縮小された印刷ができます (当然対応する欧文フォントが必要です。また、-D を指定すると-dpi のデフォルト値も同じ値に変更されます)。

o サブパラメータo でパラメータファイルを指定できます。これを指定するときは、-p=l のサブパラメータの最後に書きます。optcfg.exe で作成したものを指定できます。これを用いる場合、dviprt は通常のプリンタ初期化を行なった後に (この時、-y で指定された用紙選択も行なって) 指定されたパラメータファイルに記述された初期化コードをプリンタに出力しますので、-y による用紙選択が無効になることがあるので注意してください。また、-I による初期化ファイルの差し込みはさらにこの後行なわれます。それぞれのコード出力順序に注意してください。

j オプション-J (cf. 第 4 章 p.25) は LBP 内蔵フォントにも影響しますが、サブパラメータj は、LBP 専用のベースライン調整パラメータです。あるベースライン上に普通に LBP 内蔵漢字フォントをおいた時、そのフォントの高さの <num>%がベースラインより下に来るように LBP が設計されている場合、j<num>*10 をサブパラメータに指定すれば、LBP 内蔵フォントはベースライン上に乗るようになります。LIPS III では 79 がデフォルトです。

s サブパラメータs は、LBP 専用の文字拡大/縮小調整パラメータです。LIPS III の和文フォントはデザインの関係上 10pt の幅を要求しても、実際に印刷される文字の幅はその 95.5%程度であるため、これを補正するものです。1000 で拡大縮小なし、2000 で 2 倍の大きさになります。サイズが変わっても文字の左右中心線の位置は変わらないように印字位置を補正するようになっています。デフォルトは 1047 です。これは vfn や pk とのバランスをとるため、オプション-S (cf. 第 4 章 p.24) による拡大縮小の影響も受けます。例えば、-S=500 -p=ls2000 とすれば、LBP 内蔵フォントは一切拡大縮小されません。

u サブパラメータu は通常使う必要はありません。Canon の LBP では、EPSF や TPIC、書体倶楽部などを多用した文書を印刷する場合、印刷結果に横方向に白すじが入ったり、サービスマン・コールを起こして LBP がダウンする場合がまれにあります。こういう場合にこのオプションを指定すると問題なく印刷できる場合が有ります。引数には整数を指定しますが、これは具体的にはプリンタ CFG ファイルの minimal unit や pins に当たる部分を変更します。デフォルトは minimal unit=1, pins=8 ですが、引数に n を指定した場合 n の絶対値を minimal unit に、n が負であれば pins を 24 に変更します。問題が起きた場合にu10 やu-10 等を行なうと効果的があると思われます。

c これは LIPS III でのラスタイメージデータ転送で、データ圧縮を行なうためのフラグです。通常はほとんど効果がありませんが、サブパラメータu を指定した場合に役に立ちます。たとえば、-p=lc;u-10 等と指定します。ESC/Page プリンタの場合は、この指定は無視されます。

E LIPS III では「内蔵スケーラブルフォントやダウンロードフォントの文字が、用紙の境界にかかると印字できない」という仕様のようです (ESC/Page では、印字可能ですが、dviout/dviprt 側の対応が不十分なので、境界で分割される文字を印字する場合は、このオプションを使ってください)。例えばposter.tex などを使って、1 ページを複数枚の用紙に分けて印字する場合にこのような問題が生じます。サブパラメータE を指定した場合、用紙の境界にかかる可能性のある文字に対して、LBP 内蔵和文スケーラブルフォントの印字にはベクトルモードに移ることにより、そのほかのフォントでは用紙に入る部分をビットマップで転送することにより、対応します。ベクトルモードでは文字修飾などに一部制限が出るうえ印刷コードも冗長になるので通常は使用しない方が賢明です。

上記のj, s の補正値は OkI (CBH33984@PCVAN) 氏の発言により、以下のようにしてデフォルトが決められています。

> LIPS III で漢字を 100mm×100mm で 2000 種ぐらい重ねうちしてその結果できる四角形の外寸を求めました。明朝体については、一部の例外を除き幅 95.5mm・高さ 87.3mm・深さ 7.9mm となりました。よって LIPS III の漢字の幅は文字ピッチの 95.5%程度であろうと思われます。測定誤差もありますが、この値はmin10.tfm から得られる 95.2%にほぼ近く、JFM 通りに漢字サイズ/ピッチをセットしても LIPS III の元来の漢字フォントのデザインから大きく違わないことになります。角ゴシックでは 96mm/100mm となり 96%ちょうどです。まあ、明朝と角ゴシック間のバランスもあるので、角ゴシックについての 0.5%の違いはここでは考慮しないことにしました。よって LIPS III 漢字のサイズ/ピッチは JFM をそのまま使って良いのではないかと判断し、以上の実測から LIPS III 明朝フォントの幅 95.5mm よりs のデフォルトは $1/95.5 = 1.047(1047)$ とし、j のデフォルトは 79 としました。

10.2.2 LIPS III の内蔵フォントにおける制限について

LIPS III 内蔵和文フォントについては、次のような場合に機能が制限されます。

- フォントサイズが 420×420 dot(約 3.5cm) 以上の場合
- 拡張 tpic specials のrt によって、90 度単位以外の回転を行なった場合
- 用紙の端で切れる文字をサブパラメータE を使って印字した場合

この時、サブパラメータf で指定された、書体指定での Fill pattern(F), Shadow(S) や *weight* 等の指定は無効となります。

10.2.3 ESC/Page 対応の LBP(LP-9000/7000/3000/2000 など) の場合

LIPS III の場合と大体同じなので相違点のみ記します。まず、-p=m のサブパラメータについてです。デフォルトは、

```
-dpi=300
-p=mv1000;m2;d512;j90;s1000;D300
```

ですが、-p=mD600 を指定すると、上記のデフォルトのD300 がD600 に変わって、LP-9000(600dpi) に適合します。ただし、-p=mD240 を指定したときのみ、デフォルトのj90 も変わって

```
-dpi=240
-p=mv1000;m2;d512;j200;s1000;D240
```

がデフォルトとなり、LP-7000(240dpi) に適合します。

shotai

ESC/Page における書体番号は

9 : 角ゴシック体漢字
10 : 明朝体漢字
66 : 丸ゴシック体漢字
67 : 教科書体漢字
68 : 楷書体漢字
69 : 行書体漢字

です。そのあとの、サブパラメータは、

G : 擬似ゴシック を指定するとき指定 (文字線幅設定 1 を送っています)
H : 擬似ゴシック を指定するとき指定 (文字線幅設定 3 を送っています)
F : 20%スクリーンパターン を指定するとき指定
O : Outline font を指定するとき指定
S : Shadow 印字 を指定するとき指定

G, H は同時に指定できませんが、他は重複して指定できます。

ESC/Page での 擬似ゴシック は、プリンターに文字線幅設定 1 または 3 を送って実現しています。このコマンドによって実現される印字結果はプリンターの機種により異なります。

10.2.4 用紙サイズと給紙の指定 [dviprt でのオプション-y]

オプション

-y=*youshi*[P|L[*kyushi*[:*bin*]]][/][*youshi*[P|L[*kyushi*[:*bin*]]]]

で LIPS III や ESC/Page プリンタに用紙の選択を指示します。また、-y のあとのサブパラメータは、大文字・小文字の区別をしませんので、どちらをつかってもかまいません。

youshi や *kyushi* を 2 回指定した場合は、前半が DVI ファイルを展開する紙面サイズの指定で、後半が LBP への給紙の指定になります。袋とじ印字をする場合、このように 2 回の指定すると便利です (cf. 第 6 章 p.55)。

-y=*youshi*[P|L[*kyushi*[:*bin*]]]

のように、「/」を含め後半を省略した場合には、前半と同じものが指定されたことになりますので、通常は-y=A5P1 のように前半のみの指定で十分です。

「/」を指定すると、後半の部分で LBP への給紙の指定が決まります。後半を省略しない場合には、「/」を省略することができます。特に後半を省略して

-y=*youshi*[P|L[*kyushi*[:*bin*]]]/

「/」のみ指定すると、DVI ファイルを展開する紙面サイズの指定のみを行ない、LBP に用紙選択のコマンドを送らないことを意味します。

また、-V (cf. 第 6 章 p.56) の指定の影響を受けるため、-V -y=A4B4 は、-y=A4LB4L と同じことを意味します。

-p=l や-p=m を指定していない場合やdvioutでも、前半部分のパラメータ *youshi* と P または L の部分は意味をもち、A3, A4, A5, B4, B5 または、はがきサイズの用紙を指定した時、新サイズオプションの-PW, -PH のデフォルトがそれぞれの用紙サイズに設定されます (cf. -PW, -PH の項、第 8 章 p.66)。-p=l や-p=m の LBP を指定した場合でなければ、定型用紙サイズ指定以外のサブパラメータは (-y=F も) すべて無視されます。

youshi

A3, A4, A5, B4, B5, H, F*width*[:*height*], T*number* のいずれかを指定します。それぞれ順に、A3, A4, A5, B4, B5, はがきサイズ、フリーサイズを意味します (A3, B4 は、該当の大きさの用紙をサポートしているプリンタにのみ使用できます)。F の場合、パラメータとして幅と、必要ならば「:」に続けて高さをドット単位で指定します。T の場合には、パラメータとしてペーパー指定番号 (cf. LBP のマニュアル) を直接書きます。

P|L

P を指定するとデフォルトのポートレイト (用紙縦置き) を、L を指定するとランドスケープ (用紙横置き) をプリンタに指示します。

kyushi

そのあとのサブパラメータ *kyushi* は給紙方法の指示で、以下のように数字 0, 1, 2, 3 で指定します。

	LIPS III	**ESC/Page**
0 :	カセット給紙	指定不可
1 :	手差し給紙	標準給紙装置
2 :	上段カセット (旧機種はサポートされない)	オプション給紙装置
3 :	下段カセット (旧機種はサポートされない)	指定不可

ESC/Page においてはさらに、ビン番号を「:」の後に続けて数字で指定することができます。ただし、LP7000 の ESC/Page では上記のオプション給紙装置のコマンドがサポートされていません。LIPS III や ESC/Page で給紙方法を省略した場合、フロントパネルでの設定に従います。

10.2.5 コピー部数の指定 [オプション-lc]

また、-p=l や-p=e により出力する場合には、

```
-lc=number
```

によって、コピー部数を指定することができます。デフォルトでは-lc=1 で、1 部のみの出力です。

10.3 ファイルへの出力 [オプション-O]

dviprt では、

```
-O=[file_name]
```

によって指定したファイル名 *file_name* に出力を行ないます。ファイル名の代わりに、デバイス名を書くと、そのデバイスに出力されます。-O= に続けてファイル名を書かないと、出力データが 16 進数で標準出力 (通常は画面) に出力され、プリンタの設定のデバッグに利用できます。

ファイル名 *file_name* に文字列「!!!」が含まれていると、ページ毎に異なったファイルに出力することができます。このとき (含まれている最後の)「!!!」は、通しのページ番号 (001, 002, ...) に置き換わります。通しページ番号が、999 になるとそれ以上はファイルを変更せずに続けて同じファイルに出力されます。なお、各出力ファイルの先頭には、初期化のコード (プリンタ定義ファイルの bit_image_mode で定義されたものと-I オプション (cf. 本章 p.83) によるもの) が付加されます。

なお、ファイルのデータをコピーコマンドなどでプリンタに転送するときは、raw モードにしないと正しく転送できません (PC-9801 では、[CTRL]+[F·4] で graph モードに切り替えます)。

10.4 出力スピードの調整

[dviprt でのオプション-s]

-s=*number*

の *number* を設定することにより、印字のためのデータ出力速度を遅くすることができます。*number* のデフォルトは 0 で最速です。10000 程度の値にまで大きくすることができます。特に、NetWare などの LAN での dviprt の利用を行なう場合、出力速度を落とす必要性からこのオプションがつけられました。

第11章 プリンタ定義ファイルの記述

11.1 プリンタ定義ファイル

プリンタ定義ファイルは、

- プリンタが要求するビットイメージデータの形式
- プリンタの印字命令のうちdviprtが使用するプリンタの機能に当たるもの

などを定義しておくためのファイルで、この情報をもとにdviprtはプリンタコードを生成し出力します。

dviprtが直接使用するプリンタ定義ファイルは、拡張子が.cfgのもので、これは人が読み書きしやすい形ではなくコンピュータよりの形 (バイナリ形式) をしています。人が読み書きしやすいプリンタ定義ファイルの形式としてテキスト形式のものが用意されており、この形式のファイルの拡張子は慣例的に.srcとしています。dviprtは、バイナリ形式の*.cfgしか解釈できませんので、テキスト形式の定義ファイル*.srcをなんらかの手段でバイナリ形式に変換する必要があります。この変換を行なうプログラムがoptcfg.exeです。

これから説明するプリンタ定義ファイルの説明はテキスト形式の定義ファイル*.srcに関するものです。従って新たにプリンタ定義ファイルを作成したり既存のものを修正する場合にはテキスト形式の*.srcに対して作業を行ない、dviprtで使うには一度optcfg.exeを使って*.cfgに変換してください。

dviprtの配布には多くのプリンタ定義ファイルが付属しています。各種の新しいプリンタは既存のプリンタのプリンタ命令と同じ命令、またはそれを拡張したもの、を使用していることが多いのでプリンタ定義ファイルを一から作成しなければならないことはまれです。まずは似たプリンタ、例えば旧機種や別メーカの製品に機能拡張を施しただけのプリンタならばその元になった機種のための設定ファイルがないかどうかを確認してそれをそのまま、またはわずかな修正で使用できないか検討してみてください。

またこの説明で具体的なイメージがつかめない場合には配布に含まれているものを例としてご覧になることをお勧めします。

なお、プリンタ定義ファイルの仕様がdviprtバージョン 2.42で拡張されました。ここでの説明はこの拡張された仕様について述べてあります。ここで説明するテキスト形式のプリンタ定義ファイルをバイナリ形式に変換するにはoptcfg.exeのバージョン 3.0 以降が必要です。旧仕様との関係については本章p.112をご覧ください。

11.2 説明中の表記に関して

この説明では以下のような表記を用います。

- 出力されるプリンタコード
  実際に出力されるプリンタコードはイタリック体を使って 16 進数表記 (16 進ダンプ) することとします。

  例 :

  FF FD 23 45 0D 0A

- 数値
  説明中の数値は上記のようなプリンタの出力コードを表わす場合を除いて特に断らない限り 10 進数表記です。説明の都合上 n 進数表記の数を用いる場合には、$(Number)_n$のようにその数を括弧で囲ん

み、基数を下付き文字で添えて表わします。

例 :

8 進数表記の 123

$(123)_8$

- プリンタ定義ファイル中に実際に記述する文字列
  プリンタ定義ファイルに実際に書く内容は、そのままをタイプライタ体で記述することとします。

  例 :

  ```
  maximal_unit  : 360
  ```

11.3　項目の詳細

プリンタ定義ファイルの全項目の意味を詳しく述べます。説明では一部、dviprt のプリンタコード生成についての理解を前提としている部分があります。dviprt のプリンタコード生成手順に関する説明 (本章 p.107) を参考にしてください。初めて目を通す場合には、各項目に設定する値の型さえわかっていただければ結構です。各項目は、整数型、選択型、文字列型、プリンタコード型、のいずれかの型を持っています。

- `name`

  プリンタの名前 (文字列) です。出力されるプリンタコードには影響を及ぼしません。dviprt 起動時にこの名前をディスプレイに表示しますので、どのプリンタ用のプリンタ定義ファイルなのか判別できるような名前を登録しておくのがよいでしょう。

- `upper_position`

  プリンタのピン配列などを選択し、指定します。

  *1 ピン配列の指定
  ビットイメージをプリンタに出力する際、イメージデータを並び替えてから送る必要があります。これは、プログラム内部ではビットイメージを上から下、左から右の順番で行優先で保持しているのに対して、プリンタが要求するイメージデータの順番が必ずしもそうなっていない為です。
  dviprt ではこの順番を出力データ中のドットの順番と印字結果との対応によって次の 4 種類に区分しています。これらの中の一つをこの項目で指定します。

  (a) `HIGH_BIT`
  11.1 p.97 のように、イメージデータを列優先で出力します。この種類に該当するものとして ESC/P 系のプリンタがあります。

  (b) `LOW_BIT`
  `HIGH_BIT` と基本的には同じですが、縦方向の並びが 8 ドット毎に (すなわち 1 バイトのデータの中のビットの並びが) `HIGH_BIT` とは逆の順序になっています (cf.　11.2 p.97)。この種類には PC-PR 系が該当します。

  (c) `LEFT_IS_HIGH`
  イメージデータを行優先で出力します (cf.　11.3 p.97)。HP DeskJet やほとんどのページプリンタがこの種類に該当します。

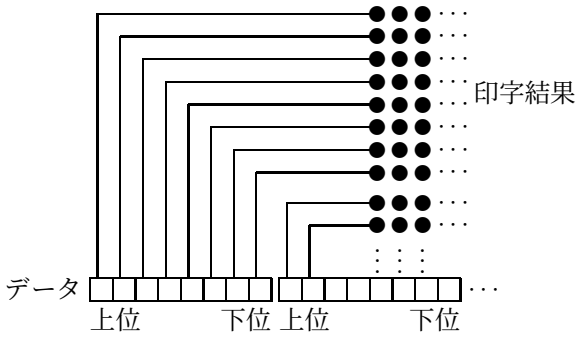


図 11.1: HIGH_BIT 型のプリンタ

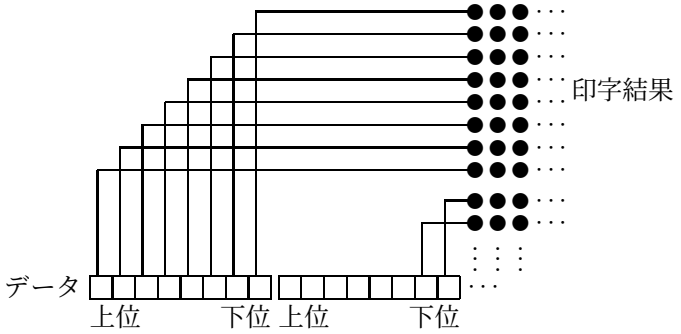


図 11.2: LOW_BIT 型のプリンタ

(d) LEFT_IS_LOW

LEFT_IS_HIGH と基本的に同じですが、イメージデータ中のビットの並びが 8 ドット毎に LEFT_IS_HIGH とは逆の順序になっています。これに該当するプリンタは今のところ存在しないようです。

∗2 NON_MOVING

イメージデータを印字後、次回の印字位置が印字した次の位置に移動せず元の位置のままであるようなプリンタ (ページプリンタに多いようです) では、さらに NON_MOVING を記述します (cf. 11.4 p.98)。

もし NON_MOVING に該当するプリンタで upper_position に NON_MOVING を指定し忘れると、重ね刷りされる、印字位置がずれる、などの不具合が生じます。

- pins

プリンタが一回の印字で印字する縦のドット数です。必ず 8 の倍数で指定します。

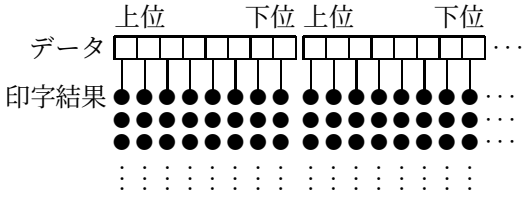


図 11.3: LEFT_IS_HIGH 型のプリンタ

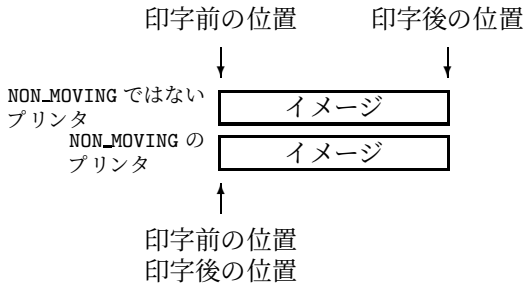


図 11.4: NON_MOVING

- minimal_unit

  ビットイメージを扱う際の最小単位の横幅です (cf. 11.5 p.107)。バイト単位 (ドット数/8) で指定します。dviprt は、DVI ファイルをビットマップイメージに変換した結果のうち、幅がここで指定されたバイト数、高さが pins で指定された範囲のドット数の矩形が空白であれば、ビットイメージの印字命令の代わりに横方向の移動命令 (skip_spaces。後述) を使います。従ってこの minimal_unit の値が小さいほど skip_spaces が使われる可能性が増すことになります。

  この項目を省略した場合、minimal_unit は、出力コードの大きさが最小になるであろう値に設定されます。

- maximal_unit

  イメージデータを出力するときにプリンタが一度に扱うことが出来る最大の横幅です。バイト単位で指定します。プリンタの最大印字可能幅を指定するものではありませんので注意してください。

  印字データの横幅や横方向の移動量がこの値よりも大きい場合にはこの値以下の大きさに分割されて出力されます。

  この項目が省略された場合、max_unit は十分に大きな値となります。多くのプリンタではプリンタの最大印字可能幅かそれ以上の値なので、通常は指定しないか、十分大きな値 (たとえば、10000) にしておきます。

- bit_image_mode

  印字データを出力する前に最初に一度だけ出力するプリンタ初期化コードです。つまりdviprt を起動後に初めて出力されるプリンタコードになります。通常、改行幅の変更、1 ページ目の給紙などを行ないます。

- normal_mode

  全ページを出力後、すなわち一番最後に出力されるプリンタコードです。プリンタのバッファのフラッシュ・最終ページの排紙を行なうコード、および bit_image_mode で変更した内容を元に戻すためのコード、などを指定します。

  最終ページのみが出力されない場合、この項目でバッファのフラッシュや排紙のためのプリンタコードを指定し忘れていることが原因の一つとして考えられます。

- send_bit_image

ビットイメージの印字命令を出力する毎にデータの先頭に付加して出力するプリンタコードです。後述する bit_row_header および after_bit_image と組み合わせて一つのビットイメージの印字命令を形成します (cf. 11.7 p.110)。

- bit_row_header

  upper_position の項目に LEFT_IS_HIGH か LEFT_IS_LOW が指定された場合に有効です。

  1 ラスタ行 (縦方向 1 ドット) のデータの先頭にこのコードが付加されます (cf. 11.8 p.110)。この項目が省略された場合には何も出力されません。

- after_bit_image

  各イメージデータを出力する毎に、ここに定義されるデータをイメージデータの末尾に付加します (cf. 11.7 p.110)。この項目が省略された場合には何も出力されません。

- skip_spaces

  印字位置を水平方向に移動する命令。

- line_feed

  プリンタのバッファのフラッシュ、行頭への移動、pins ドット下への移動、を行なうためのプリンタコード。dviprt は印字位置を縦方向に移動するのにこのコードを使いますので、ここで定義されたコードによって、丁度 pins ドット下の行頭に移動することが必要です。プリンタの改行コードを使う場合には “改行幅 =pins ドット分の長さ” になるようにあらかじめ bit_image_mode で改行幅を変更しておかなければなりません (通常、プリンタの改行幅の初期値は “プリンタフォントの高さ $+\alpha$” です)。

  この項目が省略された場合には ASCII コードの「復帰 + 改行」(*0D 0A*) が用いられます。

- form_feed

  プリンタのバッファのフラッシュおよび改ページ (現ページの排紙と次ページの給紙) を行なうためのプリンタコード。この項目が省略された場合には ASCII コードの「復帰 + 改ページ」(*0D 0C*) が用いられます。

  ページの最終行のみが出力されない場合、この項目でプリンタのバッファのフラッシュを行なうコードを指定し忘れていることが原因の一つとして考えられます。

- dpi

  プリンタのデフォルトの解像度です。dviprt の起動オプション -dpi や -DPI が指定されない場合、ここで記述された解像度が用いられます。y_dpi を指定しなかった場合は、縦横が同じ解像度となります。

  整数で指定します。省略された場合には 180dpi が指定されたものとみなされます。

- y_dpi 縦横の解像度が違う場合に、縦の解像度を指定します。dviprt の起動オプション -DPI に対応しますが、起動オプションが指定された場合は、起動オプションの方が優先されます。

- encode

  プリンタに送るビットイメージデータをどのように符号化するかを指定します。符号化とは、ここでは元の形に一意的に復元できるような形にデータを変換することをさします。例えばデータ圧縮などです。

ここに指定される内容と同様のことが起動オプションによっても可能な場合 (例えば-FAX や-comp) もありますが、プリンタの制御コードとビットイメージの符号化の方法との間には密接な関係がある場合が多いためプリンタ定義の中で記述できるようになっています。

この項目では以下の指定が可能です。

- HEX データを ASCII テキストによる 16 進数表現に変換します。16 進数ダンプを想像していただくとよいでしょう。
- FAX G3 Fax エンコード。起動オプションの-FAX と等価。ならべて-FAX オプションの引数を書いておくことも可能。
- PCL1 HP PCL mode 1 圧縮。起動オプション-comp=1 と等価。
- PCL2 HP PCL mode 2 圧縮。起動オプション-comp=2 と等価。

11.4 書式

11.4.1 コメント

以下のような行はコメント行として無視されます。

- 文字 「;」(セミコロン) で始まる行。「;」の前に空白文字を先行させることも出来ます。
- 文字 a–z, _ (アンダースコア) 以外の文字で始まる行。

ただし、後述するプリンタコード型の項目中の「継続行」に限っては、後者の場合はあてはまりません。

11.4.2 各項目共通の書式

各項目は、

項目名 : 値

のように、項目名とその値とを「:」(コロン) で区切って書きます。項目名は必ず第 1 カラムから書きはじめます。空白文字が先行した場合コメント行とみなされてしまいますので注意してください。「:」の前後の空白はあってもなくても構いません。

11.4.3 項目の型別の書式

11.4.3.1 文字列型の項目

文字列を指定します。この項目には name が該当します。

例 :

```
name            : BJ-10v 48 pin
name            : HP DeskJet 505J (mode 2 compression)
```

11.4.3.2 整数型の項目

pins、minimal_unit、maximal_unit、dpi、y_dpi が該当します。数値を 10 進表記で記述します。

例 :

```
pins      : 48
```

このタイプの項目の値は以下に示すような範囲内の非負の整数です。

pins	8 の倍数かつ $8 \leq$ pins ≤ 128
minimal_unit	1 以上
maximal_unit	1 以上
dpi	1 以上
y_dpi	1 以上

11.4.3.3 選択型

upper_position、encode が該当します。

あらかじめ決められた数種類の選択肢の中から一つを選び、記述します。付加的な情報をさらに付け加えることができる項目もあります。

このタイプの各項目の選択肢は以下のようになっています。選択肢の意味については各項目の説明をご覧ください。付加情報を記述する場合には同じ行に空白文字で区切ってならべて書きます。

項目名	選択肢	付加情報
upper_position	HIGH_BIT	選択肢によらず
	LOW_BIT	NON_MOVING
	LEFT_IS_HIGH	を指定可能。
	LEFT_IS_LOW	
encode	FAX	-FAX オプションの引数
	HEX	
	PCL1	
	PCL2	

例 :

```
upper_position   : LEFT_IS_HIGH NON_MOVING
encode           : FAX 1728;2280
```

11.4.3.4 プリンタコード 型の項目

この型には

bit_image_mode, normal_mode, send_bit_image, bit_row_header, after_bit_image, skip_spaces, line_feed, form_feed

の各項目が該当します。

このタイプの項目は、dviprt が要求するプリンタの動作を具体的にどのようなプリンタ命令を使ってプリンタに行なわせるか、ということを定義しています。

この項目の値は、次の 2 種類の要素を出力する順番に並べたものからなっています。

- そのまま出力するコード。

- 書式指定による、dviprt の内部変数の値やそれを使って様々な演算を施した結果の出力。例えば、水平スキップの移動量を 2 バイトのバイナリ表記で出力させたい、というときなどに使います。

プリンタコードの定義中ではスペースやタブなどの空白文字は単なる区切り文字として扱われ全て無視されます。従って空白文字そのものを記述したい場合には後述するエスケープシーケンスを用いて表現する必要があります。

コードの指定

そのまま出力するプリンタコードは以下のように記述します。

- アルファベット、数字、'\' を除いた記号類など、空白文字以外の ASCII テキスト文字はそのまま記述します。

  例 :

  ```
  normal_mode    : showpage
  ```

- 上記以外のものは、以下のようなエスケープシーケンスを用いて表わします。エスケープシーケンスは文字 \で始めます。

  `\`　*backslash* または *yen*。'\' そのもの (ASCII コード *5C*)。
  `\s`　または
  `\SP`　*SPace character*。スペース文字 (ASCII コード *20*)。
  `\e`　または
  `\ESC`　*ESCape character*。エスケープ文字 (ASCII コード *1B*)。
  `\n`　*New line, line feed*。改行 (ASCII コード *0A*)。
  `\t`　*horizontal Tab*。水平タブ (ASCII コード *09*)。
  `\r`　*carriage Return*。復帰 (ASCII コード *0D*)。
  `\f`　*Form feed*。改ページ (ASCII コード *0C*)。
  `\v`　*Vertical tab*。垂直タブ (ASCII コード *0B*)。
  `\"`　*double quotation*。文字「"」(ASCII コード *22*)。
  `\0`*ooo*　*Octal number*。8 進数で表わされたコード。「\0」(オー 0 ではなくゼロ 0) で始めます。*o* には 0 以上 7 以下の数字が入ります。例えば $(33)_8$は \033 のように表わします。1 バイトのコードを表わす数ですのでとりうる値の範囲は 0 以上 255 以下、すなわち \0000 から \0377 です。桁数は可変です。
  `\`*ddd*　*Decimal number*。10 進数で表わされたコード。*d* には 0 以上 9 以下の数字が入ります。ただし最上位桁の数字は 8 進数表記と区別するために 0 以外の数字で始めます。例えば $(72)_{10}$は \72 のように表わします。とりうる値の範囲や桁数は 8 進数と同様です。
  `\x`*hh*　または
  `\X`*hh*　*heXadecimal number*。16 進数で表わされたコード。*h* には 0 以上 9 以下の数字、または a から f までのアルファベット (大文字でも可) が入ります。例えば $(1A)_{16}$は \x1a のように表わします。とりうる値の範囲や桁数は 8 進数と同様です。

  これらのエスケープシーケンスを使って例えば次のように記述します。

  例 :

  ```
  bit_image_mode    : \e 3 \24
  line_feed         : \r \n
  ```

エスケープシーケンスにエスケープシーケンスではない表記を続ける場合には 1 つ以上の空白文字で区切ってください。逆の場合やエスケープシーケンス同士を続ける場合には空白文字は必要有りません。

例 :

```
bit_image_mode   : \e3\24   これはエラー
line_feed        : \r\n     これは可
```

書式指定

書式指定は一般に以下のように書式と式を「,」(コンマ) で区切って記述します。

書式, 式

この「式」の値を「書式」に従って出力する、ということになります。この指定の中に空白文字を含めてはいけません。

従来の*.src との互換性を考慮して、「, 式」の部分が省略された場合には「横方向の送り幅」(後述する内部変数 d の値) を「式」の値とみなします。

- 書式新仕様のプリンタ定義ファイルには以下の書式があります。この中で n はその桁数を表わし、$1 \leq n \leq 7$、または $n = ?$ とします。$n = ?$ の場合可変長の桁数を表わし、数の表現に必要な最小の桁数で出力されます。可変長の桁数は ASCII テキスト形式の書式においてのみ使用可能です。

  \b$n$ バイナリ 下位 → 上位バイト

  \B$n$ バイナリ 上位 → 下位バイト

  \o$n$ 8 進 ASCII テキスト

  \d$n$ 10 進 ASCII テキスト

  \h$n$ 16 進 ASCII テキスト (小文字)

  \H$n$ 16 進 ASCII テキスト (大文字)

  上記の ASCII テキスト表記 (8 進、10 進 16 進) の書式では桁数の指定に「I」を続けることによって最下位桁の文字コードに$(16)_{10}$すなわち$(10)_{16}$を加えるようにできます。これは桁数指定が可変長 (「?」) の場合に最下位桁を表わすいわばデリミタのような役割をします。

  例えば、「\d?I」というフォーマットで$(1000)_{10}$を表現した場合、出力されるコードは 16 進表記で *31 30 30 40* です。

  バイナリ表記のフォーマットの場合にはこの書式オプション I は無視されます。

  \st 文字列を式の値の回数だけ出力。

  この書式\st だけは他のものと比べて特殊なもので以下のような形をしています。

  \st, 式,"文字列"

  「書式」から最初の「"」までには空白文字を含めてはいけませが「"」で囲まれた部分には改行文字以外の空白文字を含めることができます。「文字列」の長さは 255 バイト以内です。「"」で囲まれた部分は、そのまま出力するコードの指定と同じ方法で記述します。つまり空白文字は全て無視されエスケープシーケンスが使えます。文字「"」そのものを文字列に含めるには「\"」と記述してください。

  この書式 \st では式の「値そのもの」を出力する代わりに式の値を回数の指定とみなしてその回数だけ「文字列」をくり返し出力します。

- 式

  式は以下のような定数、内部変数、演算子、および括弧という要素からなります。

  - 定数

    0 以上$(65535)_{10}$以下、つまり$(FFFF)_{16}$以下の非負の整数です。表記の桁数は可変です。

    `0`*oooooo*

    8 進表記の整数。「`0`」で始めます。例えば$(13)_8$は「`013`」のように記述します。

    ddddd

    10 進表記の整数。「`0`」以外の数字で始めます。例えば$(1234)_{10}$は「`1234`」のように記述します。

    `x`*hhhh* または`X`*hhhh*

    16 進表記の整数。例えば$(7FFC)_{16}$は「`x7ffc`」のように記述します。「`x`」一文字だけ書いた場合、後述の「水平方向の絶対位置」とみなされてしまうので注意してください。

  - 内部変数

    `w` ページ幅 (dots)。

    `h` ページ高 (dots)。

    `r` 横の解像度 (dpi)。

    `R` 縦の解像度 (dpi)。

    `p` 通しページ。始めのページが 1 です。値は `form_feed` を出力した後で更新されます。

    `v` pins の項目の値を 8 で割ったもの。

    `c` `constant` の項目で定義された値。旧仕様との互換性のために用意されています。新たにプリンタ定義ファイルを作成するときには使わないようにしてください。

    `s` 出力するイメージデータそのもののバイト数。`send_bit_image`, `bit_row_header` および `after_bit_image` の定義内でのみ使える変数です。
    データ圧縮や HEX MODE によるイメージデータの加工を指定している場合にはその加工後のデータサイズです。`send_bit_image` と `after_bit_image` の定義の中では縦 `pins` ドット分のデータのバイト数、`bit_row_header` ではその 1 ラスタ行分のデータのバイト数です。

    `d` 横方向の送り。`send_bit_image`, `bit_row_header`, `after_bit_image`, `skip_spaces` の定義内でのみ使える変数です。旧仕様で書式指定を使って出力されていた値はこの変数の値に該当します。

    `x` 水平方向の絶対的な印字位置 (dots)。ページ左端のドットの位置が 0 です。定義中で使われているプリンタコード型の項目値を出力する直前の印字位置を指しています。従って、`send_bit_image`, `bit_row_header`, `after_bit_image`, `skip_spaces` といった `x` の値が変更される可能性がある項目の定義中で印字位置が移動した後の値が欲しければ `x` の代わりに後述する式による表現を用いて `x+d` と表現してください。例えば ESC/P 24 pin のプリンタで、横方向のスキップを相対位置指定でなく絶対位置指定によって行なうには `skip_spaces` を以下のように定義します。

    例 :

    ```
    skip_spaces      : \e $ \b2,x+d
    ```

    `x` の値は `after_bit_image` や `skip_spaces` を出力した後に更新され `line_feed` を出力した後で 0 にリセットされます。ただし、`NON_MOVING` タイプのプリンタの場合は、

after_bit_image の後での更新は行なわれず skip_spaces を出力したあとにのみ更新されます。従っていずれの場合も x の表わす位置は実際のプリンタの状態に一致することになります。

y 垂直方向の絶対的な印字位置 (dots)。ページ最上部のドットの位置が 0 です。x 同様、コードを出力する直前の位置を指しています。値は line_feed を一回出力するごとに更新され、form_feed を出力した直後に 0 にリセットされます。

– 演算子

以下の二項演算子があります。

	算術演算		ビット演算
+	加算	|	論理和
-	減算	&	論理積
*	乗算	^	排他的論理和
/	除算	>	右シフト
%	剰余	<	左シフト

– 開き括弧「(」と閉じ括弧「)」

これらを組み合わせて「式」を形成します。式は、通常の中置記法で記述します。ただし、演算子の間に「優先順位はなく」、左側から順に解釈されます。例えば、以下のようなものが「式」の例です。

例 :

```
p
(w+7)/8
x+d/2
```

最後の例は、慣例的には x + (d/2) の意味になりますが、ここでは、(x + d)/2 と解釈されます。前者のように解釈させたいのであれば、括弧を用いて x+(d/2) と明示的に演算順序を指定するか、d/2+x としてください。

演算は、アプリケーション内部では符号なしの 2byte で計算されます。従って、2byte 符号なし整数で表わされる範囲からはみ出すような演算についてはその結果は保証されません。

以下は「書式, 式」の例です。

例 :

```
\b2,s
\d?,(w+7)/8
```

プリンタコード 型の項目の実際

これまでに説明してきた書式指定出力とコード出力を組み合わせて、実際にどのように項目の記述を行なうかを説明します。

書式指定とコード指定は以下のように出力する順番にならべて書くだけです。

例 :

```
send_bit_image  : \ESC Z \b2
```

この例では

1B 5A

というコードに続いて、これから送るビットイメージの横幅 (ドット単位) を下位 → 上位の順にバイナリ形式で送ります。例えば幅が 280 ドットならば数値は、

18 01

というコードで出力されます。
次の例は 1 ラスタ行ぶんのデータサイズをバイト単位で出力する例です。

例 :

```
bit_row_header  : \ESC _ W \b2,s
```

この例では

1B 5F 57

に続いてイメージデータのバイト数を 2 バイトのバイナリ形式で下位バイト → 上位バイトの順に出力します。例えばデータのバイト数が 35 バイトならば数値は、

23 00

というコードで出力されます。
次のものは 2 つ以上の書式が同時に指定された場合の例です。

例 :

```
send_bit_image  : \x9b \d?,s ; \d?,d/8 .r
```

`d?,s` はイメージデータの全バイト数を不定長の 10 進数 ASCII テキスト表現で送信することを表わしています。`\d?,d/8` はイメージデータの横幅をバイト単位の 10 進数 ASCII テキスト表現で送信することを表わしています。従って、`send_bit_image` のコードとしてプリンタへ送られるコードは、例えばこれから送るイメージデータの横幅が 280 ドットでイメージデータに対して圧縮を施したりしなければ、

9B 32 38 30 3B 33 35 2E 72

となります。
コードを何も設定しない (空のコードを設定する) 場合には項目名 +':' のみを記述します。

例 :

```
skip_spaces   :
```

プリンタコードの定義が長い場合には複数行に分けて記述することが出来ます。後続する行の行頭を空白文字ではじめてください。

例 :

```
bit_image_mode  : sttjob\SP c\x0d \x0a
                  sttpage\SP p\SP a4\x0d \x0a
                  unitdef\SP i\x0d \x0a
                  trnsl\SP 0\SP 81840\x0d \x0a
```

11.5 dviprt の出力コード生成手順

11.5.1 文書の構成

DVI ファイルはdviprt 内部でビットマップイメージに変換され、これがさらにプリンタコードへと変換されるわけですが、この、ビットイメージ → プリンタコード変換の観点から見た「文書の構成」について考えてみます。

縦 pins ドット、横数バイト (8 の倍数ドット) のブロックを「ユニット」と呼ぶことにします。dviprt がプリンタコードを生成する際の基本単位はこのユニットです。

dviprt がビットイメージを扱うときの最小単位は幅が minimal_unit バイトのユニットです。この最小単位のユニットがページの幅の分だけ横に並んだものが「行」、その「行」がページの高さの分だけ縦に並んだものが「ページ」です。ページが、縦 pins ドット間隔、横 minimal_unit 間隔の格子で区切られているところを想像していただけばよいでしょう (cf. 11.5)。この一つ一つのマス目を、左上から順に (11.5 の番号のように) 処理していきます。なお、minimal_unit よりもページの幅のほうが狭ければ行全体を最小単位のユニットとし、同様に行頭から minimal_unit 幅で行を分割していって行末に minimal_unit よりも小さな幅のイメージしか残らなかった場合 (つまりページの横幅が minimal_unit で割り切れない場合) にはその残り全部を 1 つのユニットとします。

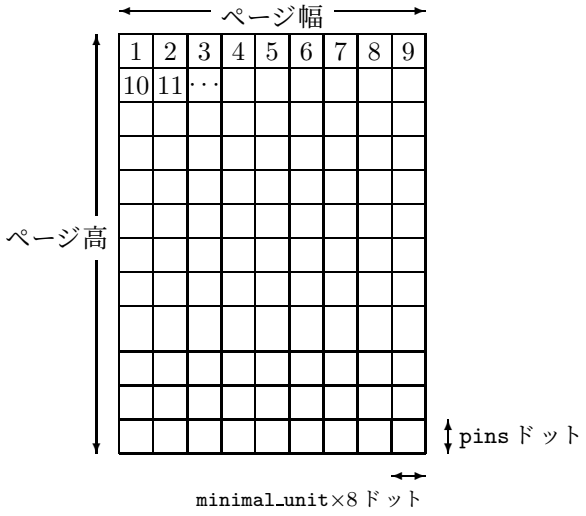


図 11.5: 文書の構成

11.5.2 出力コードの生成

11.5.2.1 一般的な手順

dviprt は出力コードのサイズを最適化するために次のような条件を満たすプリンタコードを生成します。

- 空白の部分があればビットマップの印字命令の代わりに水平スキップ命令 (横方向の移動命令) を使う。
- 行全体が空白であれば縦方向の移動命令 (改行命令) を使う。
- 行末の水平スキップは無駄であるので行なわない。
- 同様にページ末尾の改行は無駄であるので行なわない。

これらのことを実現するため文書は次に示す手順で検査され出力されます。この説明の中で特に大事なのは bit_image_mode、send_bit_image などのプリンタコード型の項目の値がどのタイミングで出力されるのかということと、minimal_unit や maximal_unit などの項目の値が出力されるビットイメージデータにどう影響するのかということです。これらの点に特に注意して説明を読んでください。

説明中に現れる $HSKIP$、$VSKIP$、$DSIZE$、はそれぞれその段階で未出力の、横方向の方向のスキップ幅、改行数、ビットイメージの幅、です。「幅」は 8 ドット単位の値です。

ビットイメージ化された文書は次のような手順でプリンタコードに変換されます。

∗1 プリンタの初期化
bit_image_mode を出力します。

∗2 各ページの出力
次の手順によって各ページを出力します。

(a) dviprt 内部のページ単位の情報を初期化
未出力の縦方向のスキップ量 (すなわち改行数) を 0 に初期化します。以下の説明ではこの量を $VSKIP$ と表わすことにします。

(b) ページ上部から順に各行を出力。
各行の出力手順については後述します。

(c) 改ページ
最終ページでなければ form_feed を出力。

∗3 プリンタの復帰処理
normal_mode を出力します。

各行は、以下の手順で出力されます。

∗1 dviprt 内部の行単位の情報を初期化
未出力の印字ユニットの横幅、横方向のスキップ量、をそれぞれ 0 に初期化します。以下の説明では、これらの値をそれぞれ $DSIZE$、$HSKIP$ と表わすことにします。

∗2 空白の検査、ビットイメージの出力、空白部分のスキップ
次の手順を行の左端から順に右端 (行末) まで繰り返す。

(a) 行の左側から順に、幅 minimal_unit のユニットを取り出す (cf. 11.5 p.107)。minimal_unit よりも少ないデータしか残っていなければ、その残り全てを取り出す。

(b) ビットイメージとして出力するデータのバッファリング
取り出したデータに空白以外のデータが含まれていれば未出力の印字ユニットにこれを加え、加えたデータの横幅だけ $DSIZE$ を増やす。出力はまだ行なわない (手順 2c または 3 でまとめて出力する)。

(c) ビットイメージの出力
取り出したデータが全て空白データであった場合、もし $DSIZE > 0$ であれば、次の手順により未出力の印字ユニット (cf. 11.6 p.109) を出力する。

i. 垂直方向の移動
$VSKIP$ の分だけ line_feed を出力後 $VSKIP$ を 0 にリセットする。

ii. 水平方向の移動
$HSKIP$ の分だけ横方向にスキップ (すなわち skip_spaces を出力) し、$HSKIP$ を 0 にリセット。$HSKIP >$ maximal_unit ならば maximal_unit 以下の量に分割して出力する。

iii. NON_MOVING の処理
upper_position の項目に NON_MOVING の記述があれば $HSKIP$ に $DSIZE$ の値を代入。

iv. ビットイメージの出力

以下の手順を $DSIZE > 0$ の間繰り返すことによって未出力の印字ユニットを出力する。この手順によりビットイメージが maximal_unit 以下の幅に分割されて出力されることになる。

A. 幅 $DSIZE$ の印字ユニットのうち幅 maximal_unit のユニットを取り出す。
$DSIZE <$ maximal_unit であれば残りのデータ全てを取り出す。

B. send_bit_image を出力。

C. 取り出したユニットを upper_position に設定された内容に従って加工し、出力する (cf. 11.7 p.110, 11.8 p.110)。

D. after_bit_image を出力。

E. 出力したユニットの幅だけ $DSIZE$ の値を減らす。

v. 取り出した空白データの幅だけ $HSKIP$ を増やす。

∗3 未出力のビットイメージデータの出力

行末までこれらの処理を繰り返した後、未出力の印字ユニットが残っていれば手順 2c と同じ手順で出力する。

∗4 改行

$VSKIP$ を 1 増やす。改行コードの出力はまだ行なわない (手順 2(c)i で行なう)。

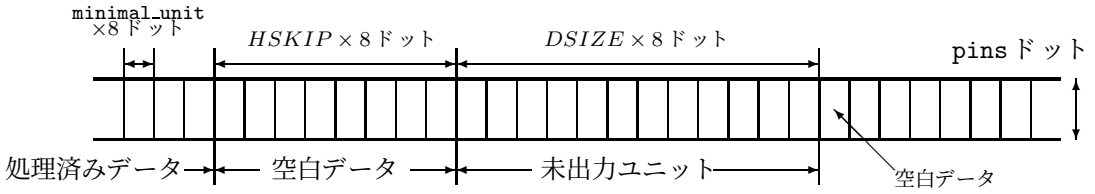


図 11.6: 印字ユニット

ただし、手順 2b で行なわれる空白のチェックは、skip_spaces が空文字列でない場合 (すなわち何等かのコードが指定されている場合) にのみ行なわれます。当然ですが、skip_spaces の命令が無ければ横方向のスキップが行なえないからです。結果として空白データも印字ユニットの一部として出力されることになります。

また、upper_position が LEFT_IS_HIGH、または LEFT_IS_LOW のもので、かつ bit_row_header が指定されている場合には、縦 1 ドット分のデータ毎に、そのデータの先頭に bit_row_header を付加して出力します (cf. 11.8 p.110)。

改行や水平スキップは印字データを出力する直前にのみ出力されることになりますからこの手順は次のように考えることも出来ます。

- 空白データは出力しない。
- 従って印字命令には印字位置を合わせるための移動命令 (水平移動や改行、改ページ) を先行させる。

11.6 プリンタ定義ファイルの記述に関する注意点

ここではプリンタ定義ファイルを新たに記述する際の注意事項を例を挙げて述べます。

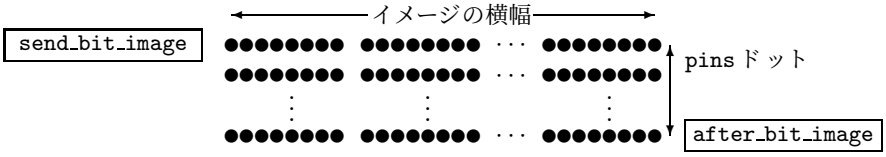


図 11.7: 出力コード

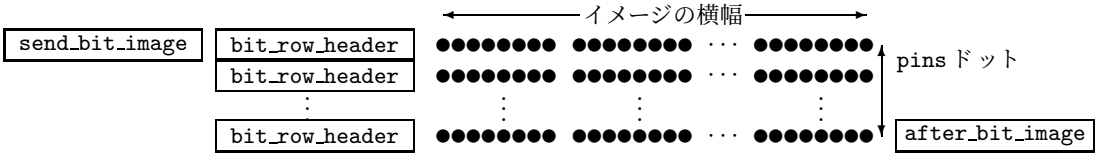


図 11.8: bit_row_header

11.6.1 maximal_unit

この項目はプリンタの印字可能領域の最大幅を差すものではありません。しばしば混同されるようですので注意してください。印字可能領域の最大幅は、dviprt のコマンドラインオプションのうち新サイズオプションであれば-MW、そうでなければ-W を用いて指定します。

maximal_unit は、一度にプリンタに送ることが出来る横方向の送り幅 (バイト単位) です。横方向の送り幅とは前述したようにビットイメージの横幅や横方向のスキップの移動幅のことです。ここでは例として横 1000 ドットのイメージをプリンタに送る場合を考えます。

- 一度に幅 400 ドットのイメージまで扱えるプリンタの場合
  イメージを 400, 400, 200(ドット) の幅に分割して、計 3 回に分けてプリンタに送信する必要があります (cf. 11.9 p.111)。
  この 400, 400, 200 ドット幅のイメージデータそれぞれに対して send_bit_image, after_bit_image, bit_row_header が付加されて出力されることになります。
  このようなプリンタでは、

  例 :

  ```
  maximal_unit    : 50
  ```

  と指定しておきます。バイト単位であることに注意してください。
- 一度に十分広い幅のイメージを扱えるプリンタの場合
  十分に広い幅のイメージ、つまりプリンタの最大印字可能幅以上の幅のイメージ、を一度に扱えるプリンタでは maximal_unit は十分に (必要以上にでも可) 大きく指定しておきます。現在あるプリンタのほとんどはこの条件を満たします。例えば次のように設定しておきます。

  例 :

  ```
  maximal_unit    : 10000
  ```

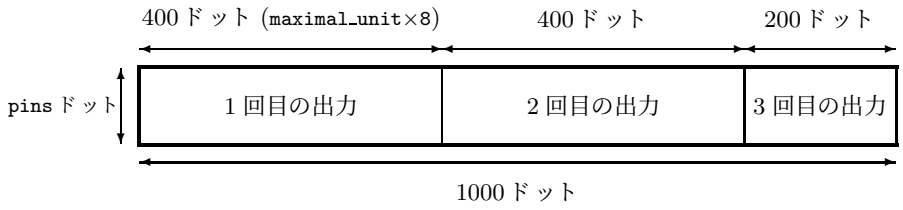


図 11.9: 一度に小さいイメージしか扱えないプリンタ

11.6.2 minimal_unit

skip_spaces が空のとき、空白データかどうかのチェックは行なわれませんから minimal_unit の値は結果として出力コードに影響を与えません。このことを利用して、PBM フォーマットでの出力など空白のイメージデータをスキップ命令や改行で置き換えるべきでない場合に対応できます。

似たような設定に minimal_unit を十分に大きく設定しておく、というものがあります。例えば、

例 :

```
minimal_unit    : 10000
```

のようにする場合です。この場合行全体が最小単位として扱われますので、

- 行全体が空白の場合
  この行のデータは改行命令に置き換えられます。
- 行に空白以外の部分がある場合
  行全体をビットイメージとして出力します。

つまり、空行を改行命令に置き換えるということは行なわれますが空白のユニットをスキップ命令に置き換える、ということは行なわれません。

以下では minimal_unit の値と出力コードのサイズの関係について述べます。従って skip_spaces に何等かの命令が定義されていて空白データのチェックが行なわれる場合についてのみ考えます。

minimai_unit は、空白データかどうかを調べる際の最小の単位ですから、この値が小さいほど、ビットイメージの印字命令の代わりにスキップ命令が出力される頻度が増すことは前に述べました。

一行の印字を考えた場合、出力されるプリンタコードはスキップ命令とビットイメージの印字命令が交互に並んでいます (maximal_unit が小さい場合には必ずしもそうはなりませんがここでは考えません)。

ここで出力コード全体のバイト数に大きく影響するのがこの印字命令のうちイメージデータを除いた部分、すなわち skip_spaces、send_bit_image、bit_row_header および after_bit_image のバイト数です。スキップ命令が多く使われるほど出力されるイメージデータそのものの大きさは小さくなりますが、その代わり印字命令とスキップ命令の切替時に出力されるコード、つまり上記の 4 つの項目で定義されるプリンタコードが出力される頻度が増します。

例として pins = 48 のときにイメージ 8 ドット、空白 8 ドット、イメージ 8 ドット、というビットイメージを印字する場合を考えます。

skip_spaces が 2 バイト、send_bit_image が 10 バイト、bit_row_header と after_bit_image が 0 バイトのとき出力されるプリンタコードのバイト数は、

- minimal_unit = 1 のとき

$$(10 + 48) \times 2 + 2 = 118 \text{ バイト}$$

- minimal_unit = 10 のとき

$$10 + 3 \times 48 = 154 \text{ バイト}$$

と、minimal_unit が小さいほうが有利です。

一方、これとは異なって skip_spaces が 2 バイト、send_bit_image が 5 バイト、bit_row_header が 4 バイト、after_bit_image は 0 バイトのときには、

- minimal_unit = 1 のとき

$$(5 + 4 \times 48 + 48) \times 2 + 2 = 492 \text{ バイト}$$

- minimal_unit = 10 のとき

$$5 + 4 \times 48 + 3 \times 48 = 341 \text{ バイト}$$

と、逆に大きくなってしまいます。これは bit_row_header が大きく影響しているためです。

一般には、bit_row_header が定義してある場合や send_bit_image と after_bit_image のバイト数の合計が pins に比べて極端に大きい場合には minimal_unit を 1 よりも少し大き目に定義しておくほうが出力コードの大きさは小さくなります。出力コードのサイズが最適になるような minimal_unit の値のおおよその目安は length (strings) をstrings のバイト数として、

$$\{\text{length}(\texttt{send\_bit\_image}) + \text{length}(\texttt{after\_bit\_image}) + \text{length}(\texttt{skip\_spaces})\}/\texttt{pins} + \text{length}(\texttt{bit\_row\_header})$$

と表わされます。

以上のことは出力データのサイズのみに注目した場合の最適値について述べたものです。実際にはプリンタの機械的動作の速度、印字結果の品質、プリンタ命令の精度なども考慮すべき要素です。これらとのかねあいで最も適した値を選んでください。

11.6.3　トップマージンが無視される場合の対処

ページのトップマージンが無視される場合にはページの先頭で出力されるプリンタコード、すなわち bit_image_mode と form_feed に対し以下のような何種類かの変更を試してみてください。

- 改ページコードを加える。
- 横 1 ドット幅の空白のビットイメージやタブやスペースなどの空白文字の印字を行なうコードをつけ加えておく。

ただし、このときこのコードによって印字位置が移動してしまったのを元に戻すために復帰コード (同じ行の行頭に移動する命令 (多くのプリンタでは *0D*。絶対位置指定での水平位置 0 への移動でも可) をその直後に追加してください。

11.6.4　旧仕様との互換性

現在のプリンタ定義ファイルの仕様は旧仕様の上位互換になっており、新しいoptcfg.exe でも旧仕様のプリンタ定義ファイルはそのまま処理できます。しかし、拡張によって旧仕様のいくつかの機能は必要なくなったり、より一般的な項目・表現によって置き換えが可能になったりしています。

ここではそのような機能が現仕様においてどのように置き換えることができるのかということを述べます。

11.6.4.1 HEX_MODE

従来 upper_position の項目で HEX_MODE を指定することによりビットイメージデータを 16 進テキストで出力できていましたが、これは項目 encode で HEX を指定することと等価です。

11.6.4.2 constant

項目 constant を使った数値の出力については書式指定の「式」を使って同じ結果を得ることができますのでこの説明では触れていません。optcfg.exe やdviprt は constant を使ったプリンタ定義ファイルも受け付けますが、新たにプリンタ定義ファイルを作成する場合には constant は使わないようにしてください。

11.6.4.3 書式指定

プリンタコード型の項目の書式指定には従来 D、DD、DDD、T、M、という書式オプションが用意されていました。現在の仕様でもこれらは引き続き使用可能ですが、これは旧仕様に従って記述された*.src との互換性のためです。現在の仕様では式による表現に置き換えることが出来ますので新しくプリンタ定義ファイルを記述する際にはそちらを使用するようにしてください。

これらの書式オプションはそれぞれ以下のような式による表現で代替えが可能です。

従来	新仕様
D	d/2 または d>1
DD	d/4 または d>2
DDD	d/8 または d>3
T	d*“pins の値/8” または d*v または s
M	d*“constant の値” または d*c

実際、optcfg.exe でこれらの書式オプションを含んだ*.src を処理すると自動的に等価な「式」に変換します。どのような式に変換されるかはoptcfg.exe のオプション-r を使って*.cfg を逆変換することによって知ることが出来ます。

第12章 拡張機能(\special)

TEX には、TEX 本体が解釈を行なわずに、指定された文字列をそのままドライバに渡す \special というプリミティブ命令が用意されています。dviout/dviprt は、この \special 命令を用いたいくつかの拡張を行なっています。

12.1 tpic specials

[**オプション**-tpic]

dviout/dviprt は、tpic specials の命令を含んだ DVI ファイルを表示・印刷する機能を持っています。この機能のスイッチは-tpic です。

```
-tpic=switch
```

tpic specials のコマンドを無視するには、-tpic=0 とします。tpic specials の命令を多く含んでいる関係で、表示・印刷に時間がかかる文書を校正するときに使うと便利です。

デフォルトは-tpic=1 で、tpic specials のsp コマンドで描写する曲線に Bezier 曲線を使用します。

-tpic=2 とすると、Bezier 曲線の代わりに spline 曲線を用います。[1]

使用できる通常の tpic specials コマンド：

pn, pa, fp, ip, da, dt, sp, ar, ia, sh, wh, bk

12.1.1 注意事項

tpic specials の詳細については、付属のtpicdoc.tex(LATEX 文書) をご覧ください (なお、この文書の表示・印字には-tpic=2 を指定してください)。ここでは、若干の注意事項だけを述べておきます。

12.1.1.1 ページをまたがる線幅の指定

dviout/dviprt は、ページ毎に処理を行なうのでページをスキップして表示/印刷すると以前のページの線幅が生かされずデフォルトの線幅になることがあります。デフォルト以外の線幅を複数ページにわたって用いるときは、各ページの最初か一連の tpic specials の直前にpn コマンドを書くのが無難です。

12.1.1.2 LBP 印字でのシェーディング処理

sh によるシェーディングで、閉じた図形の塗り潰しが行なえます。例えば白で塗り潰すと、図形の下にあったものはテキストも含めて総て消去することを意味します。LBP で印字する場合にダウンロードを行なった文字や LBP 内蔵フォントは、このシェーディング処理で消去されませんので注意が必要です。これが問題となるときには、欧文フォントに対しては、ダウンロード機能を OFF に、和文フォントに対しては、LBP 内蔵フォントを用いないようにしてください。(具体的には、オプション-p のサブパラメータv, d に 0 を指定します。cf. 第 10 章 p.88)

[1] Ver 2.38.3 までは spline 曲線、Ver 2.39 では Bezier 曲線を固定的に用いていた

12.1.2 tpic specials の拡張

dviout/dviprt の tpic specials は、本来の tpic specials と比べて、以下のように拡張が行なわれています。

12.1.2.1 da コマンド の拡張

gnuplot などで関数グラフをかくのに便利なように、da コマンドを拡張し、複数の引き数を取れるようにしました。

引き数は交互に実線部分、空白部分の長さをインチ単位の実数で指定します。この場合、破線はこの指定パターンのくり返しで描かれ、パスの各点を「連続に」通過します。すなわち、各点上に実線部分がのるとは限りません。例えば

```
\special{da 0.1}    % o___   ___    ___o___   ___
\special{da 0.1 0.1}% o___   ___   ___ o ___   ___
\special{da 0.1 0.01 0.01 0.01}          % 一点鎖線
\special{da 0.1 0.01 0.01 0.01 0.01 0.01}% 二点鎖線
```

12.1.2.2 rt コマンド の増設

関数グラフの Y 軸キャプションをかくのに便利なように、rt コマンドを設けました。

rt x y f で、以後の全てのテキストおよびグラフィックス出力を (x, y) の回りに角度f だけ回転させます。他の tpic コマンドの場合と同様、x, y はミリインチ単位の整数、f は時計回りが正のラジアン単位。回転は相対的ではなく、常にビットマップの絶対座標に対して行なわれ、f=0 でリセットされます。例えば

```
\begin{picture}(500,500)
\put(100,250){\special{rt 0 0 4.71239}\makebox(0,0){Leftside}%
\special{rt 0 0 0}}
\put(250,250){\special{rt 0 0 3.14159}\makebox(0,0){Upsidedown}%
\special{rt 0 0 0}}
\put(400,250){\special{rt 0 0 1.57080}\makebox(0,0){Rightside}%
\special{rt 0 0 0}}
\end{picture}
```

12.1.2.3 Bz コマンド の増設

これは、sp コンドで描写する曲線を Bezier 曲線から spline 曲線へ、あるいはその逆への切り替えをするコマンドです。

```
\special{Bz 1} で以降の \special{sp} をベジェに
\special{Bz 0} で以降の \special{sp} をスプラインに
```

となります。デフォルトは、Bezier 曲線を用いますが、オプション-tpic=2 で spline 曲線に変えることもできます。

12.2 PostScript コードの取り込み

LaTeX のソースで、(NTT jTeX で主に使われる)eclepsf.sty や、(アスキー日本語 TeX で主に使われる)epsbox.sty を使ってPostScript[2]コードを記述することができます。

これらは \special 命令によって記述され、DVI ファイルの中に存在することになります。

GNU の Ghostscript[3]は、PostScript コードを解釈して、ビットマップに展開して出力する機能を持つもので、DOS 上の DOS Extender(go32) の環境に移植されています。

dviout/dviprt では、上記の \special 命令をさらに拡張したものをサポートしており、これを DVI ファイル内で見つけると子プロセスとして Ghostscript を呼び出します。呼び出された Ghostscript は、\special 命令に含まれていたPostScript のコードや EPSF[4]を解釈して画像データファイルとして出力します。dviout/dviprt へ処理が戻ったときに、そのデータを読み出して表示することになります。

以上の動作はオプション-GS によって制御されます。

12.2.1 epsbox.sty との関連

\special コマンド内のpostscriptbox コマンドの処理部分はjdvi2kps に基づいています。これは、epsbox.sty と組み合せて使用するjdvi2kps と整合性を保つためです。そのため、epsbox.c 内の一部の関数は松下電器産業 (株) 情報システム研究所に著作権がありますので、その取り扱いはjdvi2kps の copyright ファイルに記された内容に従わなければなりません。なお、営利目的でない限り、今回の形でdviout/dviprt に取り込まれたものの再配布は OK(連絡は不要) との了解をいただいています。

12.2.2 画像データファイルの切り替え [オプション-GIF]

dviout/dviprt が扱う画像データファイルは、raw PBM[5]形式か monochrome GIF[6]形式を選択することができます。すなわち

```
-GIF[+|-]
```

を ON にすると、Ghostscript を用いてdviout/dviprt が生成し、利用する画像ファイルを、raw PBM から monochrome GIF に変更します。GIF ファイルは圧縮操作が施されているので、同じ画像の PBM ファイルと比較して一般にファイルサイズがかなり小さく抑えられます。

dviout/dviprt が読み込めるようにPostScript ファイルから作成する画像データのファイル名は、元のファイルの拡張子のみを .pbm または、.gif に変更したものとなります。

なお、Ghostscript の起動に先立って、同じデータから作成された画像データファイルが既に存在しているかどうかのチェックを行ない、存在していれば、Ghostscript を再度実行しないで直接その画像データを用います。

12.2.3 画像データファイルのディレクトリ指定 [オプション-gdat]

dviout/dviprt が扱う PBM または GIF ファイルが置かれるディレクトリは、デフォルトではカレントディレクトリですが

[2]PostScript は、Adobe Systems の商標です
[3]淺山氏によってdviprt 用のプリンタ定義ファイルをサポートする汎用ドライバが開発され、それ組み込んだものがあります
[4]Encapsulated PostScript ファイル
[5]Portable Bitmap
[6]Graphic Interchange Format、CompuServe Incorporated が著作権を持っており、GIF は商標です

```
-gdat=path
```

によって変更することができます。例えば

```
-gdat=b:\gdat\pbm118
```

のように指定します。

12.2.4 起動コマンドの指定 [オプション-gsx]

Ghostscript は、gs.exe というファイル名で、パスの通ったディレクトリに存在すると仮定されますが、パスを含む起動コマンド名は、

```
-gsx=path
```

により変更できます。起動にパラメータが必要なときは、スペースの代わりに ^ を区切りに使い

```
-gsx=go32.exe^gs241^-q
```

のように指定します。

Ghostscript は、多くのメモリーを必要とするので、快適に動かすためには、メモリー設定の章を参照して、EMS やバッファに関するオプションの設定を工夫してください (cf. 第 5 章 p.51)。

12.2.5 動作モード指定 [オプション-GS]

```
-GS=mode
```

mode を 1 から 5 の番号で指定することによって、PostScript 関連の動作モードを指定します。

***mode*=0:** EPS/PS ファイル、PS コード、PBM ファイル、GIF ファイル非対応です。すなわちこれらに関連した \special 命令を無視します。ただし、-gbox+ は有効です (cf. 本章 p.118)。

***mode*=1:** EPS/PS ファイル、PS コード、PBM ファイル、GIF ファイルに関連した \special 命令に対応します (デフォルト)。

Ghostscript を呼び出す必要があって、それが実行できないときは終了します。

***mode*=2:** EPS/PS ファイル、PS コード、PBM ファイル、GIF ファイルに関連した \special 命令に対応します。

Ghostscript を呼び出す必要があって、それが実行できないときも、スキップして続行します。また、Ghostscript からのメッセージが-GS=1 のときよりも詳しく表示されます。

***mode*=3:** Ghostscript は起動しません。対応する PBM(または、GIF) ファイルが存在していればそれを用い、存在しなければスキップします。

この場合、必要な画像データファイルとドットサイズが異なっていても、適当に拡大/縮小のスケール変換をしますので、例えばdviprt 用に作成した画像データファイルをdviout で縮小して見ることができます。

横のサイズは、バイト幅 (8 ドット単位) に切り上げで直したときに同じならば等しいと見なします。

***mode*=4:** Ghostscript は起動しません。サイズも含めて対応する PBM(または、GIF) ファイルが存在していればそれを用い、存在しなければ、スキップします。

***mode*=5:** Ghostscript は起動しません。*mode*=4 と同様ですが、必要な画像データファイルが存在しないときそれに対して EPS/PS ファイルから (dviout/dviprt が取り込む)PBM または GIF ファイルへの変換を行なうバッチファイル gs_exec.bat と PS ファイル gs_exec$.ps を作成します。付属の gssub.exe が必要です。

既に gs_exec.bat や gs_exec$.ps が存在する場合には、以前の内容の末尾に追加しますので、必要に応じて起動前に gs_exec.bat と gs_exec$.ps とを消去しておいてください。

作成されるバッチファイルは、必要な PBM(または GIF) ファイルのうち存在しないものだけを変換しますので、このバッチファイルを起動すれば次回から dviout/dviprt は PBM(GIF) ファイルを直接読むことができます。

12.2.6 取り込み画像の外枠表示 [オプション -gbox]

また、

```
-gbox[+|-]
```

によって、取込画像の外枠を表示することができます。このオプションは、-GS=0 でも有効です。

12.2.7 取り込み画像のサイズについて [オプション -gsize]

EPS/PS 画像の取り込みのサイズは、PostScript specials のパラメータで指定されます。DVI ファイルの magnification の値が 1000 でない場合 (例えば、TEX のソースに

```
\mag=\magstep1
```

などと書かれていた場合)、PostScript specials に書かれた取り込みサイズをこの magnification を行った後のサイズとする場合には、

```
-gsize[+|-]
```

のスイッチを ON にします。

特に、magnification が 1000 以外の DVI ファイルで epsbox.sty などを用いた画像取り込みを行なうときは、-gsize+ を指定してください。

epsbox.sty を使うと、magnification に応じてスケール変換した PostScript specials が生成されます。これと dviout/dviprt におけるスケール変換とで、二重にスケール変換されて正しいサイズでなくなることを防ぐことができます。

12.2.8 動作の流れ

PostScript のコードが書かれた PostScript specials や EPS ファイルを読み込む PostScript specials が入った DVI ファイルを dviout/dviprt で扱うには、次のようにします。

∗1 Ghostscript が、パスの取ったディレクトリに入っていて、gs でコマンドラインから起動できる状態になっているとしましょう。また、Ghostscript は、pbmraw または gifmono 出力がサポートされてい

るものとします。現在 DOS Extender(GO32) 上で動作するものが提供されています。

*2 使われる EPSF は、カレントディレクトリ、DVI ファイルのあるディレクトリ、-gdat= で指定したディレクトリ、のいずれかに置いておきます (この順で探されます)。
一時的な PS ファイルは、カレントディレクトリに作られます。PBM(GIF) ファイルは、-gdat= で指定したディレクトリ (指定されていなければ、カレントディレクトリ) に作られます。

*3 Ghostscript は、Working buffer を開放し、それで足りなければ Bit map buffer も開放してメモリーを確保した後で起動されます。Bit map buffer を開放する際、Expanded font buffer が EMS に存在して、まだ使われていない部分が充分あれば、Bit map buffer の内容をそこに一時的に避難します。
避難できなかった場合は、Ghostscript から戻って来たとき、Bit map buffer の内容が破壊されているので、再びそのページの最初からやり直します。

*4 dviout/dviprt が EPS/PS ファイルを処理するに先立って、既に PBM(GIF) ファイルが作成済みかどうかチェックされます。これは、-gdat= で指定したディレクトリ (指定していなければ、カレントディレクトリ) を探して、見つかった場合は Ghostscript を呼び出さずに、既に作成されているの PBM(GIF) ファイルを使います。
従って前項にあるように Bit map buffer が避難できなかった場合でも、再び当該箇所に戻って来たときは、必要な PBM(GIF) ファイルができているので正しく処理されます。Ghostscript の処理速度の方が dviout/dviprt の処理速度より遅いので、これでもほとんど不便はないと思われます。

*5 上記で、対応する正しい PBM(GIF) ファイルかどうかは、ファイル名、タイムスタンプ、ビットマップの縦横のドットサイズ、の 3 点を調べて判断されます (-GS=3 のときは、拡大/縮小を行なってサイズを合わせるので、サイズのチェックはしません)。
よって、EPSF を後から更新したり、異なる解像度で dviout/dviprt を起動した場合なども間違いなく処理されます。
PBM(GIF) ファイルの名前は、EPSF の対応する拡張子 (通常 .ps) を、.pbm (GIF ファイルの場合は .gif) に変えたものです。対応する PBM(GIF) ファイルでなくても同じ名前のファイルが存在すれば、Ghostscript の処理で上書きされることになります。

*6 以上のことを考慮すると、例えば VCPI 規格の EMS を使っている場合、dviout/dviprt が使う EMS と Ghostscript が使う EMS をうまく分けておけば、都合がよいことが分かります。
また、PBM(GIF) ファイルのサイズは、大きくなる可能性がありますし、Ghostscript もディスクスワップの可能性がありますので、ディスクの空き領域に注意してください。例えば、EMS や、ハードディスク (eg. b:) が十分に空いていれば、

```
-br=0 -EMS=58 -gdat=b:\gdat\pbm118
```

などという指定がよいでしょう。そうでない場合は、-br に負の適当な値 (-br=-40000 など) を設定してください。

12.2.8.1 表示される文字列の意味

Ghostscript が起動されると

```
Calling gs.exe to make golfer.pbm[0:184480]
No Conventional Memory Space.
Calling gs.exe to make golfer.pbm[1:356112]
```

のような表示がなされます。[ ] の中の最初の数字は

0: Working buffer のみを開放した起動

1: Working buffer と、Bit map buffer の両方を開放した起動

の意味で、次の数字は、Ghostscript が使用可能な conventional memory のサイズです。

また上の`No Conventional Memory Space.` は、Ghostscript のエラーメッセージです。

12.2.8.2 Ghostscript 起動時に使われるメモリーのサイズ

Ghostscript の起動に使われる conventional memory は、`dviout`/`dviprt` 終了時に表示される

```
Remaining memory far + Working buffer + Bit map buffer
```

です。

Expanded font buffer を (PC-9801 以外では Font file buffer も)conventional memory から大きく取らないようにします (cf. 第 5 章 p.52)。すなわち EMS が存在しない場合は、`-br=0` とできないので負の値を設定します (例えば、`-br=-20000`)。

Ghostscript にも充分にメモリーを割かないと、速度が遅くなります。また、`-bv` に対応するバッファは、conventional memory に取らないようにしてください。

メモリーがどうしても不足する場合は、`-GS=3`, `-GS=5` などのオプションを活用してください (この後の PBM、GIF ファイル作成の項参照)。例えば、

∗1 `-GS=5` で`dviout`/`dviprt` を起動して、`gs_exec.bat` を作成
∗2 `gs_exec.bat` で Ghostscript を起動して、必要な PBM(GIF) ファイルを作成
∗3 再度`dviout`/`dviprt` を起動すると (`-GS=5` のままでもよい)、必要な PBM(GIF) ファイルがあるので画像が取り込める

12.2.8.3 解像度の違いと画像ファイル

`dviout` と`dviprt` が画像ファイルを作るときには通常、同じ規則によってファイル名を付けるので、解像度が異なるファイルが同じ名前でできることになります。`dviout` と`dviprt` を交互に利用する場合には、実行のたびごとに新しい画像ファイルが必要となり、そのために Ghostscript を起動しようとします。

これを避けるため、`-gdat=` を指定して異なったディレクトリに PBM(GIF) ファイルを作るとよいでしょう。

また、画像ファイルの大きさが大きくなりがちな`dviprt` のみ`-GIF+` を指定して、圧縮率の高い GIF ファイルを使用するのもよいでしょう。

あるいは、`-GS=3` を指定すると、解像度が異なっていても PBM(GIF) ファイルのデータを拡大/縮小して用いることができます。例えば、プリンタ用の PBM(GIF) ファイルのみ作成しておき、それを`dviout` でも用いることができます。

12.2.8.4 テンポラリな画像ファイル

テンポラリな PBM(GIF) ファイルを必要とする部分があれば、`dviout`/`dviprt` を起動する毎に、それに対して Ghostscript が呼び出されて PBM(GIF) ファイルが作成されます。テンポラリな PBM(GIF) ファイルは 1 ページ内のみで管理されます。

`dviout`/`dviprt` がテンポラリな画像ファイルを作成するのは、以下のいずれかの場合です。

∗1 DVI ファイル中の `\special` で直接PostScript コードが書き込まれている場合の画像データ
∗2 同一ページ中で、同じ EPS/PS ファイルから異なった画像ファイルを生成する場合

*3 同一ページ中で、30 個以上の異なった EPS/PS ファイルを処理する場合
*4 左右反転、上下反転などの処理を必要とする画像データの場合

12.2.9 PBM または GIF ファイルの作成について

最後に、gs_exec.bat を使った PBM または GIF ファイルの作成について解説します。

-GS=5 のオプションをつけ、gssub.exe をパスの通ったディレクトリに入れておくと、必要な *.ps → *.pbm の変換を行なうバッチファイルgs_exec.bat と PS ファイルgs_exec$.ps が作成されます。

-GIF+ オプションを指定したときは、*.ps → *.gif の変換となります。

例えば、

```
Calling gssub.exe to make g2.pbm[0:184384]
GSSUB: Wrote on ‘gs_exec$.ps’.
GSSUB: Execute ‘gs_exec.bat’ to make ‘sample.gif’.
```

のように表示され PS ファイルgs_exec$.ps とバッチファイルgs_exec.bat が作成されます。

同名のPostScript ファイル (上の場合g2.ps) に対する同一の PBM/GIF ファイルの作成の部分が既にgs_exec.bat に存在すれば

```
GSSUB: The same codes are found in the output file ‘gs_exec$.ps’. Skip.
```

と表示されて追加は行なわれませんが、同名のPostScript ファイルでバッチファイルの内容が異なっている場合は

```
Warning: The same PS file is used in a different way
```

と表示されます。Ghostscript がgs で起動できる状態でgs_exec.bat を実行すれば、必要とされた総ての PBM(GIF) ファイルへの変換が行なわれます。特に、Ver 2.5 以降の Ghostscript の場合は

```
gs_exec page
```

によって、指定したページに必要な PBM(GIF) ファイルのみの作成ができます。

このバッチファイルで作成された PBM(GIF) ファイルのうちdviout/dviprt で認識されるのは、1 つのPostScript ファイルに対し、拡張子を .pbm(.gif) に変えたもの 1 つのみです。

```
Temporary PS file: tmpps.ps
IGNORE ? (y/n)
```

と表示された場合は、DVI ファイルの中の \special 命令で直接PostScript コードが書かれている部分の処理を表わしています。このときに作られる PBM(GIF) ファイルは、dviout/dviprt ではテンポラリなファイルとして扱われます。従ってこのときは作成されたバッチファイルを実行して PBM(GIF) ファイルを作成しても、dviout/dviprt では、そのままでは利用できません。

なお Ghostscript が Ver.2.5 以前のものであるとgs_exec.bat を実行したとき、いくつかのエラーメッセージが表示されますが、変換が正常に行なわれていれば無視してください。gs_exec.bat での処理の都合上、0 でない終了値 (255) を返しているためです。メッセージで表示される終了値が 255 であれば変換は正常に行なわれています。

12.2.10 PostScript specials の仕様

PostScript のファイルを読み込んだり、PostScript のコードを直接書くために dviout/dviprt がサポートしている special 命令について以下に述べます。

∗1 EPSF(Encapsulated PostScript File) の読み込み

epsbox.sty の出力する

```
\special{postscriptbox{width pt}{height pt}{epsf_file}}
```

のサポート他、\special の中身が

```
epsfile=epsf_file [hsize=width_len] [vsize=height_len]
epsfile=epsf_file [hscale=w] [vscale=h]
epsfile=epsf_file [scale=s]
```

という形式をサポートしました。これは、(ecl)epsf.sty で作られる DVI ファイルの他、TEX のテキスト中に \special を直接書くことを考慮しています。
また、上のいずれにもさらに

```
[hoffset=h_off] [voffset=v_off] [truesize]
```

を付けることができます。
例えば、以下のように書きます。

```
\special{epsfile=tiger.ps hsize=5.2cm vsize=7.5cm}
\special{epsfile=tiger.ps hsize=300dot/118dpi}
\special{epsfile=tiger.ps scale=0.8}
```

eps_file は、EPSF のファイル名です。*width_len*, *height_len*, *h_off*, *v_off* は、新サイズオプションの「長さ」、または、単位を付けないときは PostScript での単位 (1 inch = 72 単位) で与えることができます。
このとき、dviout/dviprt は EPSF にかかれたサイズが、*hsize*×*vsize* になるよう拡大/縮小した画像を取り込みます (実際は、Ghostscript を起動して、拡大/縮小した画像ファイルを作成します。ただし、-GS=3 オプションが指定されていて、画像ファイルが存在する場合は、その画像データを用いて、dviout/dviprt 内部で拡大/縮小の処理を行ないます。従って、前者に比べて画質が劣ります)。
なお、EPS ファイルには、%%BoundingBox: というキーワードがあって、それに続いて、サイズが書かれており、それがこの処理で参照されます。
width_len が負のときは、左右反転を *height_len* が負のときは上下反転を意味します。

- サイズは DVI ファイルに書かれた magnification の値に応じて調整されます。同様に -mag= や -e= を指定すれば、それに応じてサイズも変換されます。ただし、truesize が指定してあれば、\mag, -mag=, -e= などの影響を受けません。
- hsize, vsize の一方のみを指定したときは、EPSF に書かれたサイズをもとに、相似変換されます。何も指定されていない場合は、EPSF に書かれたサイズに従います。
- vscale, hscale が指定されていると、それに応じて EPSF に書かれた縦/横のサイズがスケール変換されます。一方のみ指定した場合は、他方は 1 となります。scale を指定すると、縦横共にその値でスケール変換されます。
- eclepsf.sty で処理すると、hsize, vsize の両方が単位のつかない数で与えられます。サイズに負の値が指定でき、例えば左右/上下反転 (裏返し) になります。

- hoffset, voffset で、EPS ファイルから取り出すときの位置を、水平/垂直方向にずらすことができます。

*2 PSF の読み込み

\special の中身が

psfile=*psf_file* [hsize=*width_len*] [vsize=*height_len*]
[[[hscale=*w*] [vscale=*h*]] | [scale=*s*]]
[hoffset=*h_off*] [voffset=*v_off*] [truesize]

というものを設けました。

デフォルトは、

s = 1, *h_off* = *v_off* = 0, *width_len* = *height_len* = 1in

です。*width_len*, *height_len* は、取り込みのウィンドウ・サイズの指定です。hsize, vsize の一方のみを指定したときは、それをもとに相似変換されます。また、hoffset, voffset, truesize の意味は、epsfile= の場合と同じです。

*3 PostScript コードの処理

\special で、最初が「PS:」や、「"␣」(後者は " + 半角空白) で始まると、以降はPostScript のコードとみなされ、テンポラリファイルにそのコードを書き込んで Ghostscript に渡されて処理されます。

12.3 画像データの取り込み

dviout/dviprt は、画像データをファイルから読み込んで、要求されるサイズに直して貼り込む機能を持っています。

サポートしているのは、raw PBM(Portable Bitmap) 形式と、monochrome GIF 形式のファイルです (拡張子が、前者は .pbm、後者は .gif となっていることが要請されます)。なお、dviprt を使って raw PBM 形式でのページ出力も可能ですが、これについては-I オプションの項 (cf. 第 10 章 p.84) を参照してください。

また、下記の機能を使って LaTeX で画像データを取り込むための便利なマクロpbmf.sty が、八雲氏により作成されています。EPSF 取り込みの LaTeX マクロ(ecl)epsf.sty と同様なものですが、詳しくはpbmf.doc をご覧ください。

PBM ファイルの画像データの取り込みは、\special で

pbmfile=*pbm_file* [hsize=*width_len*] [vsize=*height_len*] [truesize]
pbmfile=*pbm_file* [[[hscale=*w*] [vscale=*h*]]|[scale=*s*]] [truesize]

と指定します。*pbm_file* が、取り込む画像データのファイル名です (ただし、拡張子は .pbm に置き換えられますので、拡張子を省略することができます)。

hscale, vscale, scale のいずれかを指定すると、PBM ファイルの画像の大きさを 72dot/inch と考えて拡大/縮小変換します。hscale または、vscale のみ指定した場合は、他方は 1 とみなします。

上の指定をしなくて、hsize, vsize を指定したときは、そのサイズになるようスケール変換します。この 2 つの一方のみを指定した場合は、指定したサイズになるよう相似変換します。

hsize, vsize, hscale, vscale の何れかが指定してあれば\mag, -mag=, -e= に応じてさらにスケール変換されますが、PostScript specials のときと同様truesize を指定すると、\mag, -mag=, -e= などの影響を受けません。

これらの指定がなかった場合は、そのまま拡大/縮小せずに取り込みます。

なお、-DTILE オプションを付けてdviout/dviprt をコンパイルし直すと、さらにtile のキーワードをつけることができ、タイリングが指定できます。

例えば、以下のように TEX のテキスト中に書きます。

```
\special{pbmfile=golfer.pbm}
\special{pbmfile=golfer.pbm vsize=10.5cm}
\special{pbmfile=back.pbm tile hsize=100dot/300dpi vsize=80dot/300dpi}
```

上記でpmbfile= をgiffile= に置き換えると、monochrome GIF ファイルを読み込むことできます (ファイル名の拡張子は、.gif に置き換えます)。他のパラメータ (hsize やhscale など) の指定の仕方と意味は raw PBM ファイルの場合と同じです。

例えばアスキー日本語 LATEX でtiger.gif を取り込むには、以下のように書きます。

```
\documentstyle{jarticle}
\begin{document}
Tiger: height 10cm
\vspace{0.5cm}

\special{giffile=tiger vsize=10cm}
\vspace{11cm}

Tiger: width 8cm, height 7cm
\vspace{0.5cm}

\special{giffile=tiger hsize=8cm vsize=7cm}
\end{document}
```

このとき画面には

```
Figure (432,445) -> (451,465)
Figure (432,445) -> (372,325)
```

などとと表示されますが、1 行目は「ドットサイズが 432×445 の画像データファイルを 451×465 のサイズに変形して取り込んだ」ことを意味します。

この表示は標準出力になされますので、リダイレクトでファイルに書くことができます。

12.3.1 raw PBM ファイルのフォーマット

raw PBM ファイルのフォーマットは以下のようになっています。

```
P 4       0x0a                                  : 3 byte, 最初の"P4"は、ID
#<string> 0x0a                                  : 任意バイト
#<string> 0x0a                                  : 任意バイト
........
<width_by_dots> 0x20 <height_by_dots> 0x0a      : 横と縦のドットサイズ
........                                         : binary data
```

- P4, <width_by_dots>, <height_by_dots> の次にくる 1 byte は、Whitespace でも構いません (0x20, 0x0a, 0x0d, 0x09 のいずれか)。<width_by_dots>, <height_by_dots> は、通常の 10 進数です。
- 2 行目以降に続く「#」で始まる行は、コメント行とみなされます。#<string> は、

  ```
  # Image generated by Ghostscript (device=pbmraw)
  ```

というようなものです。

- 画像の binary data は、`width_by_byte = (<width_by_dots>+7)/8` byte とおくと

  第 1 列 `width_by_byte` byte, 第 2 列 `width_by_byte` byte,...

  の形式の `width_by_byte`×`<height_by_dots>` byte の data です。最初のバイトの再上位ビットが、左上端にあたります。

12.4 ページ記述言語の埋め込み

`dviprt` で、*file_name* で与えられたファイルの内容をバイナリでそのままプリンタに送る `\special` コマンド

```
\special{lipsfile=file_name}
\special{escpagefile=fil_ename}
\special{pdlfile=file_name}
```

をサポートしています。`lipsfile=` は、`-p=l` を指定した時に、`escpagefile=` は、`-p=m` を指定した時に、`pdlfile=` は、`dviprt` でいつでも有効ですが、それ以外では無視されます。*file_name* に拡張子がないときは、それぞれ拡張子 `.lp3`, `.epg`, `.pdl` が補われます。詳しくは、`pdldoc.tex` を参照してください。

12.5 ファイルの取り込み

tpic specials などを書いたファイルを

```
\special{file=file_name}
```

と指定して読み込むことができます。ファイルには複数の `\special{...}` を書いておくことができます。

- ファイルのパス名のサーチは、`epsfile=` などのときと同じ (カレントディレクトリ、DVI ファイルのあるディレクトリ、`-gdat` で指定したディレクトリの順) です。
- ファイル中に「`%`」があると、`0x0a` の改行コード まで無視されます。
- ファイル中の `\special{...}` の中の `{...}` の部分のみ解釈され、他の部分は無視されます。
- この `{...}` の中に `{ }` のネストがあれば正しく解釈されます。
- `{...}` の中の文字で、コード `0x20` 以下の文字は空白に置き換えられます。
- 1 つの `{...}` の中は、最大 1000 byte まで許されます。

第13章 その他の拡張機能

この章では、第 12 章 p.114 で述べた \special を使用した拡張機能以外の拡張機能について、まとめて解説します。

13.1 JIS コード変換 [オプション-JC]

```
-JC=number:new_code=old_code[y|t]:...:new_code=old_code[y|t]
```

number には、変換する文字の (定義の) 総数を 10 進数で与えます。*new_code*, *old_code* は 4 桁の 16 進数で、JIS_code が *old_code* の文字を *new_code* に変換します。デフォルトでは、縦書文字、横書文字の両者ともに変換しますが、「y」を指定すると横書文字のみ、「t」を指定すると縦書文字のみ変換します。先頭から探して、最初に見つかった変換だけが有効です。

13.2 縦書き TEX での横書きフォントの利用 [オプション-g]

dviout/dviprt は、縦書きが扱えるアスキーの pTEX に対応しています。

```
-g[+|-]
```

を ON にすると、縦書きモードにおいても、一部の文字を回転したり位置の補正を行なって、横書き用漢字フォントを使用します。

具体的には、横書き用の (これまで使っていた) 漢字フォントでは、カッコなどの向きが左右を向き、小さい「ゃゅょっ」や句読点などが左下にあるのでこのまま使うと格好悪くなります。それ故に、フォントの回転や位置補正をすることによって、比較的正しい出力が得られるようにします。

> この機能をオプションにした (デフォルトにしない) のは、将来 (誰かが) 縦書き用にデザインした漢字フォントを作ってくれることを期待しているからです (特に、カナ文字などは、縦書きにすると左右にばらつきますので)。
>
> (この段落、Naochan!氏の238_ptex.doc より)

詳しくは、238_ptex.doc をご覧ください。なお、Naochan!.tex と、Naochan!.dvi は、Naochan!氏によって作成された pTEX のサンプルです (このサンプルの著作権は、Naochan!氏にあります)。

13.3 フォントの回転 [dviprt でのオプション-R]

dviprt では、

```
-R[+|-]
```

を ON にすると、オプション-p=l で使われる LIPS III、オプション-p=m で使われる ESC/Page 内蔵の和文スケーラブルフォント、および -vfn オプションが ON のときに使われるフォントを 90 度回転して用いることができます。

13.4 オプション出力

13.4.1 ベースラインの表示 [オプション-base]

```
-base[+|-]
```

を ON にすると、文字のベースラインの位置に「線」を表示することができます。元来は、pIEX 対応に関連して文字位置の補正のデバッグ用に作られたものです。

13.4.2 文字の箱表示 [オプション-box]

```
-box[+|-]
```

これも、-base と同様デバッグ用に作られたものですが、文字の代わりに「箱」を書きます。

-p=l, -p=m オプションを使っている場合には、「箱」の中を網掛けすることによって、以下のように使用されているフォント種類が識別できるようになっています。

PK/JXL/-vfn のフォント：BOX
システムフォント　　：BOX + 網掛け (グレイスケール)
ダウンロードフォント　：BOX + 濃い網掛け

13.5 停止、終了

13.5.1 dviprt の停止と終了 [オプション-Z]

dviprt を実行途中で一旦停止したいときは、[ESC] キーを押してください。

Abort(A) or End after flush(E) or Continue(C)?

と表示されます。[A], [E], [C] の何れかのキーを入力すると

A: 直ちにdviprt から抜けだす
E: その行の印字データを出力後にdviprt を終了
C: 印字データの出力を続行

となります。[A] の入力では、ラスターデータの転送の途中でも中断しますので、これで都合の悪いプリンタの場合は [E] を入力してください。

dviprt でのパラメータ

```
-Z[+|-]
```

が ON になると、1 ページ印字ごとに一旦停止します。任意のキーを押すと続行しますが、[ESC] を押すと以降の印字を取り止めます。

[STOP] キーで、中断することはできません。

13.5.2 PC-9801 版dviout の終了時の画面モード指定 [オプション-E]

dviout を終了したとき、16 色グラフィックモードになるかどうかは、デフォルトでは、ディップスイッチで拡張グラフィックモードを指定していたかどうかで決まります。強制的に 16 色モードにしたい場合は

```
-E=16
```

を、8 色モードにしたい場合は、

```
-E=8
```

をパラメータに指定してください。

13.5.3　dviout 終了時の情報

終了時には、例えば、

```
Saved Font Cache
Files open              : flush  0, total      3 files
Bit map buffer          : flush  0, total 124416 bytes, used 124416 split 1
Font file buffer        : flush  0, total  65536 bytes, used  41160 in GVRAM
Expanded font buffer    : flush  0, total  40000 bytes, used 112131 #1245 EMS
View Image buffer       : number10, total 160000 bytes              in EMS
Working buffer          : EMS    7  +     194417 bytes (EMS: total 285 pages)
Remaining memory        :           local  37568 bytes, far     224 bytes
```

のように使用されたバッファの状況が表示されます。Expanded font buffer の最後の「#」の後の数字は、新たに展開した文字の数です。Font Cache をロードして、それで足りた場合は 0 となります。

13.6　メッセージ表示を消す

バッファの使用状況などの表示は標準出力に対してなされます。従って、この表示を省略したい場合は、起動コマンドラインの最後に、> NULL を付けてヌルデバイスにリダイレクトしてください。

13.7　ユーザのキー入力の要求　　[オプション-wait]

dviout/dviprt が致命的でないエラーや警告時の処理は、オプション

```
-wait=mode
```

で変更できます。*mode* の値により

- 0: 無条件に処理停止
- 1: ユーザのキー入力によって、続行または処理停止 (デフォルト)
- 2: エラーメッセージを表示して続行
- 3: エラーメッセージの表示なしに続行

となります。

第14章 ユーティリティー・プログラム

dviout/dviprt に付属のユーティリティー・プログラムには以下のものがあります。

gather.exe

PK, PXL1001, PXL1002, PXL1003, JFM, PKD, VFD などのフォントファイルを束ねて GTH ファイルを作成するものです。束ねたファイルから特定のフォントを取り出すこともできます。

GTH ファイルはdviout/dviprt から直接読むことができるので、これを用いると、ディスクスペースの節約に役立ちます。任意のファイルを束ねることができ、メモなどをいっしょに入れておくことが可能です。詳しくは、gather.doc をご覧ください。

optcfg.exe

dviprt は、プリンタ定義ファイルを用いることで種々のプリンタに対応していますが、そのソースファイルをdviprt が読み込める形式に変更するプログラムです。逆方向の変換を行なうこともできます。ソースファイルの書き方は、プリンタ定義ファイルの記述 (cf. 第 11 章 p.95) を参照してください。

knjtopk.exe

和文のビットマップ形式のフォントファイルを通常の PK フォントや JXL4 フォントと同様な方法で圧縮して、ファイルサイズを小さくするものです。圧縮されたものもdviout/dviprt から読むことができます。逆方向の変換もできます。

knjfont.exe

一般のビットマップ形式の和文のフォントファイルを、dviout/dviprt で扱えるフォントファイル形式に変換するためのユーティリティーです。詳しくは、knjfont.doc をご覧ください。

pktopkd.exe

大きなサイズの PK ファイルから PKD ファイルを作成して、dviout/dviprt が効率よく扱えるようにします。詳しくは、第 4 章 p.18 を参照してください。

chkfont.exe

DVI, TFM, JFM, GF, PK, PKD, PXL, GTH, FAR ファイルから主にフォントの情報を読んで表示します。詳しくは、chkfont.doc を参照してください。

gssub.exe

PostScript specials を使用する文書を扱うときに、Ghostscript を起動して、PostScript コードからdviout/dviprt が扱える画像データに変換するためのバッチファイルを作成します。このためには、dviout/dviprt 起動時に-GS=5 オプションを指定します。詳しくは、PostScript の項を参照してください (cf. 第 12 章 p.121)。

fax2pbm.exe

G3 FAX で使われる T.4 勧告 1 次元符号化方式の圧縮ファイルを、ページ別の複数の raw PBM ファイル、または monochrome GIF ファイルに変換するプログラムです。これにより G3 FAX のファイルがdviout/dviprt で処理できるようになります。詳しくは、fax2pbm.doc をご覧ください。

`mkgaiji.exe`

`dviprt` の`-p=l`, `-p=m` のサブパラメータ`g` によって、まず必要な和文文字情報を出力し、それらの文字を作成した後、`dviout`/`dviprt` を起動する、という 2 パス形式の動作を行なうときの和文フォント作成のプログラムです (cf. 第 4 章 p.29)。提供されるのは、文字データ作成部分を除いた「文字情報析部分」のソースファイルです。個々のハードウエア、ソフトウエア、フォントデータに基づいてフォントを作成したり、特別なサイズや飾り文字などを使う場合の利用が考えられます。

`ttindex.exe`

`dviout`/`dviprt` で和文代替フォントとして Microsoft Windows 標準の TrueType Font を使用するために参照する TrueType インデクスファイル `*.tti` を造ります。詳しくは、`ttfont.doc` をご覧ください。

第15章 エラーとその対策

15.1 初めて使うときに起こしやすいエラー

ILLEGAL_ARGS: 環境変数 または パラメータ・ファイルで TEXPK を設定して下さい

環境変数、またはパラメータファイルでTEXPKに、正しく「フォント指定」を設定してください (cf. 第1章 p.3, 第4章 p.16, 第4章 p.23)。

NO_FONT: *

DVI ファイルで使用されているフォントが見つかりません。探されたフォントの情報が表示されるので (環境) 変数TEXPK などの設定を確認してください。本当に所有していないフォントが使用されている場合は、-F (cf. 第4章 p.20) で代替フォントを指定することで、取り敢えず表示/印刷を行なうことができます。-f (cf. 第4章 p.27) オプションをつけて起動して、使われるフォントを詳しくチェックするとよいでしょう。chkfont.exe (cf. 第14章 p.129) も役立ちます。また、METAFONT が使用可能であれば、-gen (cf. 第4章 p.47) により、不足フォントを自動作成することも可能です。

Unpack_Error: Illegal font ID:c:/font/118/cmr10.tfm(..)

(環境) 変数TEXPK では、和文フォントの JFM ファイルを指定できますが、欧文フォントの TFM ファイルがあると、エラーになります。和文フォントの項 (cf. 第4章 p.23) を参照してください。表示メッセージのcmr10.tfm の部分が、他の欧文の TFM ファイルである場合も同じです。

Warning: Cannot find cmr10.tfm for TT Font

欧文の TrueType Font を使用しましたが、(この例の場合にはcmr10 に) 対応する*.tfm ファイルが見つかりませんでした。文字を並べるための情報を*.ttf から得ていますので、文字位置に「ずれ」が生じている可能性があります。TFM ファイル*.tfm のあるディレクトリを環境変数TEXFONTS に設定してください (cf. 第4章 p.19)。

Font_Mismatch: 16: c:/font/jfm.gth MIN10.TFM c:/font/jfont/kanji.24(7423H)

フォントファイルに登録されていないコードの文字が現れたことを示しています。

NO_MEMORY: font file over: *

-bf (cf. 第5章 p.52) でサイズを指定するフォントファイルバッファに、フォントファイルが読み込めなかったことを意味します。大きなサイズの PK フォントファイルを多数使っている場合には、pktopkd.exe で PKD ファイルを作成し、これを使ってアクセスするようにしてください (cf. 第4章 p.18)。

NO_MEMORY: ヒープエリアの不足

書体倶楽部フォントや TrueType Font の展開、tpic specials のコードにはヒープエリアを使っており、また環境変数やフォントファイルのパス名などもヒープエリアに格納しています。従って、複雑な TrueType Font や tpic specials 命令などがあるとこのエラーが生じやすくなります。ワークエリアもフォントの展開に使うことができるので-bw=(cf. 第5章 p.53) の値を増やすと解消できることがあります。

NO_MEMORY: **バッファ領域の不足**

メモリーが足りなくなっています。常駐しているプログラムなどがあれば解除して、フリーなメモリーを確保するか、バッファの大きさを調整してください (cf. 第 5 章 p.51)。

BAD_SYSTEM: **テキストがプリンタのドット幅を超えています**

-W で調整するか、あるいは無視して続行します (このとき、はみでた部分は欠けて印字されます。cf. 第 8 章 p.69)。新サイズオプションを使用しているときには、-PW, -LM, -RM, -MW の項 (cf. 第 8 章 p.66) を参照してください。

BAD_SYSTEM: **テキストの 1 ページの長さが大きすぎます**

-H で調整するか、あるいは無視して続行します。新サイズオプションを使用しているときには、-PH, -TM, -BM, -MH の項 (cf. 第 8 章 p.66) を参照してください。

テキストの上部、あるいは、左部分が欠ける

-X または-Y (cf. 第 8 章 p.69) で調整してください。あるいは、新サイズオプションを使用して-LM, -TM (cf. 第 8 章 p.64) を正しく設定 (必要なら、-OX, -OY も設定) してください。

テキストの下部、あるいは、右部分が欠ける

DVI ファイルに書かれているページサイズをオーバーした部分がある場合に起こります。-w または-h (cf. 第 8 章 p.69) で調整してください。新サイズオプションを使用すれば、ほとんどの場合解決されます。必要に応じて-PW, -PH (cf. 第 8 章 p.66) などを正しく設定してください。

dviout/dviprt **を起動した時にパラメータが解釈されない**

常駐プログラムや DOS の仕様により、dviout/dviprt のコマンドラインでのパラメータの指定に使われる文字「;」や「=」が別の意味に解釈されてしまうことがあります (例えばマルチステートメントの区切りなど)。これらの区切文字は、ほとんどの場合「:」など別の文字に置き換え可能なので、それに変更するか、オプションをファイルに記述して、-= で読み込むようにしてください (cf. 第 2 章 p.11)。

dviout **でスクロールやページ切り替えが遅い、または速すぎる**

-s, -t, -sh, -u (cf. 第 9 章 p.71) などで指定する数値を小さく、または大きくしてください。

dviout **でキーの反応が過敏すぎる**

-sh, -u (cf. 第 9 章 p.71) で大きな数値を指定してください。

紙面における印字位置がおかしい

-X, -Y, -C (cf. 第 8 章 p.69) で調整。新サイズオプションの場合には、-LM, -TM(, -OX, -OY, -HC, -VC, -HS, -VS, -PF2)(cf. 第 8 章 p.64) などに設定する値を調整してください。この位置の調整には、付属の test_org.dvi を用いることができます (cf. 8.1 p.68)。-LM や-TM は、左マージンや上マージンの長さを指定するパラメータではありませんので、注意してください。また、dviout において新サイズオプションを使用している時には、紙面を示す枠は、デフォルトで A4 用紙であり、例えば B5 用紙では-y=B5 (cf. 第 8 章 p.66, 第 10 章 p.92) のように指定する必要があることに注意してください。

横方向、または、縦方向にずれが生じた行が印字される

印字限界幅を越えて印字命令を送るとそれを次行に繰り越すプリンタがあり、そのためこの現象が生じることがあります。-MW, -RM, -LM, -PW が正しく設定してあれば、印字限界幅を超えた印字命令を

送ることはありません (cf. 8.1 p.68)。A4 用紙に対する印字限界のテストには、付属の`test_a4.dvi`が使えます。

ページとページの間で空白ページが出力されてしまう

1 ページの印字限界長を越えて印字命令を送るとそれを次ページに繰り越すプリンタがあり、そのためこの現象が生じることがあります。とくに、プリンタ定義ファイルで `skip_spaces` の項 (cf. 第 11 章 p.95) が空白のプリンタの場合注意が必要です。`-MH`, `-TM`, `-BM`, `-PH` が正しく設定してあれば、印字限界幅を超えた印字命令を送ることはありません (cf. 8.1 p.68)。A4 用紙に対する印字限界のテストには、付属の`test_a4.dvi` が使えます。

和文のゴシック文字が正しく出力されない

ゴシック文字のフォントがない場合には当然正しく出力できませんが、擬似的にドットをずらして強調文字にすることができます。`-G` の項 (cf. 第 4 章 p.32)、書体倶楽部や TrueType Font、あるいは LBP のスケーラブルフォントを用いている場合には、それぞれ対応する項を参照してください (cf. 第 4 章 p.36, 第 4 章 p.41, 第 10 章 p.89)。

欧文の文字と和文の文字の高さが不揃い

和文のフォントに代替フォントを使っている場合は、その代替フォントに応じた調整が必要です。一般にはオプション`-J` (cf. 第 4 章 p.25) で、書体倶楽部/JG Font の時はファイル dviout.vfn, `dviprt.vfn` (cf. 第 4 章 p.36) の中のパラメータで調整します。LIPS III/Esc Page のときは、`-p=` のサブパラメータ`j` でも調整できます。

プレビューアで、縮小イメージ表示が見られない

[V] や [C] キーによる縮小イメージ表示は、画面が分割表示されているときは使うことができません。その状態のときには、画面上のページ番号が `[1]` ではなく、`[1-1]` のように表示されていることで確認できます。対処方法については、次項を参照してください。

プレビューアで、1 ページが分割されてしまう

`-bb=0` のオプションを使うか、`-bb` の値を増やす (cf. 第 5 章 p.51)。メモリーが足りなくなるときは、バッファ (`-br` の値を減らし、`-bf` は設定しない) で調整し、例えば日本語フロントエンドプロセッサなどを解放して (`adddrv`, `deldrv` などを使ったバッチファイルがよいでしょう) メモリーを確保してください。また、大きな magstep を用いない (`-mag=0` などとする) ようにするのも効果的です。EMS があるときは、`-br=0`, `-bb=0`, `-bw=0` (cf. 第 5 章 p.53) とします。ページサイズが大きい文書の場合は、`-e=800 -varf+ -esize+` などとして、縮小表示することでも使用バッファ量を少なくできます (cf. 第 7 章 p.62)。

また、新サイズオプションを使用しないように指定したり (`-newsize-`)、一部機能を削った小サイズの`dviout`(`dviouts.exe`, `dvioutt.exe` など) を使うことも考えられます。

GA-1024A/1280A 用`dviout` でページ番号だけしか表示されない

GA-1024A/1280A を使用するための初期化プログラム`gainit` のパラメータ`\W=` で設定したウィンドウが使用できない場合に、このエラーが起きます。例えば、仮想 86 モードで設定したアドレスに、RAM が割り付けられている場合などです。空いているアドレスを設定するか、`\W=F0` を指定するように変えます。16 色モード (gray scale) にするとエラーが起きないようですが、これは保証できません。

No Conventional Memory Space と表示されGhostscript が起動されない

Ghostscript 起動のために開放可能な conventional memory を増やしてください。各種の常駐プログラムはなるべく解除するようにし、EMS が十分にあるときは-br=0 とし、そうでなければ負の値 (例えば-br=-40000) を指定します (cf. 第 5 章 p.53)。また、-bv (cf. 第 5 章 p.53) に正の値を設定しないほうがよいでしょう。

GO32 の上で動く Ghostscript でstub.exe と結合された実行ファイルのときは、stub.exe を切り離して-gsx=go32^gs のようにすると Ghostscript を起動するのに必要な conventional Memory がより少なくて済みます。

Ghostscript の実行が遅い、あるいは止まってしまう

Ghostscript 実行のために使用する Protect Memory とディスクの空き領域を十分に確保してください。

VCPI 規格の EMS を使用しているときには、-EMS=44 のように適切な正の値をオプションで設定して、Ghostscript を実行するための Protect Memory を十分に確保するようにしてください (cf. 第 5 章 p.53)。またディスクは、作成される画像データと、場合によっては Ghostscript のスワップ領域にも使われますのでご注意ください。うまくいかない場合は-GS=5 (cf. 第 12 章 p.117) オプションを使うこともできます。

PostScript の取り込み画像のサイズがおかしい

DVI ファイルの magnification の値をデフォルトである 1000 以外にして、サイズの変更をした上でepsbox.sty などの EPS ファイル取り込みのマクロを用いた場合、そのマクロとdviout/dviprt とで二重にスケール変換を行なってしまうため、取り込み画像のサイズが正しくなくなることになります。このような場合には、オプション-gsize+ (cf. 第 12 章 p.118) を指定してください。

dviprt で印字したが、トップマージンが空かない

これは、単票用紙の場合に起こることが多く、プリンタ側の問題ですが、-T (cf. 第 6 章 p.55) オプションで解消できる可能性があります。あるいは、プリンタ定義ファイルを作成して対処しますが、それにはescp_24.src の最後の注意の項を参照してください。

dviprt での印字で、漢字やひらがなが極端に小さい

漢字やひらがななど、いわゆる全角文字の部分には、大きさに応じた和文フォントか、書体倶楽部フォントや TrueType Font などの可変サイズのフォントが必要です。

また、TrueType Font などのスケーラブルフォントを使おうとする場合には、その指定を誤るとフォントの存在が認識されないため、システムフォントが使われてしまい、小さくなってしまうことがあります。

ディスク上のフォントの場合は、それが認識されているかどうか、-r -f=1 というオプションをつけて調べて見てください (cf. 第 4 章 p.27)。必要とされる和文フォントのドットサイズと実際に使われたフォントとの関係が分かります。

必要なら-K による倍角指定 (cf. 第 4 章 p.33) や、-varf+ によるフォント拡大/縮小機能 (cf. 第 4 章 p.35) を指定してください (ただし、品質の点では問題が残ります)。和文フォントが代替のスケーラブルフォントの時 (-varf+ 指定の時も含めて)、その大きさは-S (cf. 第 4 章 p.24) で調整できます。

LIPS III で印字すると、横方向に空白の筋が入る

プリンタ側に原因があるようですが、-p=l のサブパラメータu により解消できる場合があります (cf. 第 10 章 p.90)。-p=lc;u-10 などとしてください。-p=lu-10 でもかまいませんが、データ転送量が増えて印字が遅くなる可能性があります。

LBP を使ったとき、紙面の境界付近の文字が印字されない

LBP の内蔵スケーラブルフォントやダウンロードしたフォントの文字が、用紙の境界にかかると印字できない現象です（LBP の仕様)。poster.tex などを使って、いくつかの用紙に分割して印字する場合に起こり得ます。これは、-p=l または-p=m のサブパラメータE を使って -p=l;E または-p=m;E のように指定することにより、対応できます (cf. 第 10 章 p.91)。LIPS III では、このような文字はベクトルモードやビットマップ転送を用いて印字することになるので、通常は指定しない方が無難です。

これでもうまくいかないときは、さらにサブパラメータd0 を使って、-p=l;E;d0 のようにしてみてください。

縦書きで、記号類の位置や方向がおかしい

普通の漢字フォントでは、記号類 (約物) の位置が横書き用に調整されていますので、-g+ で縦書き用に補正して使います (cf. 第 13 章 p.126)。なお、一辺のサイズが 500 ドット (EMS を使わないときは 300 ドット) 以上の巨大な文字では方向の補正ができないことがあります。

和文代替フォントの JG Font による文字がおかしい

和文の代替フォントとしての JG Font 対応は現在のところ暫定的なもので、「文字があまり美しくない」、「文字の下部が欠けることがある」などの不十分な点があります。

-FAX を指定したが、Starfax 形式の出力が得られない

dviprt.par などにプリンタ指定のオプション (例えば、-p=m など) が設定されていると、それが有効になるためデフォルトの Starfax 形式の出力となりません。-p= の設定を削除するか、あるいはコマンドラインから -p=ostarfax と明示的に指定してください。

MS Windows の DOS Window で実行するとエラーが起きる

dvioutは、全画面 DOS Window でないと正常に動かないようです。pif ファイルに設定しておくと良いでしょう。

また、TrueType Font を使用していると Windows 本体との間で共有違反が生じる可能性があります。TrueType Font を read only 属性にしておくと、このエラーは解消できます。

dviprt で直接、あるいはファイル出力を、prn: デバイスに転送したが印字できない

PC-9801 でprn: デバイスにファイルを転送するときは、Graphic モードでないと、そのままの内容が転送されません。[CTRL]+[F·4] が、Graphic モードと日本語モードとの切り替えスイッチになっています。

プリンタによる印字が全くできない

使用するプリンタに合ったオプション-p (cf. 第 10 章 p.82) の指定が必要です。プリンタ定義ファイルの指定やその作成が必要なこともあります。作成済みのものを使用する場合でも、プリンタのディップスイッチやパネルスイッチなどの設定によっては正しく動作しないこともありますのでご注意ください。プリンタのマニュアルも参照してください。

DOS/V の日本語モードのときは、-P (cf. 第 10 章 p.87) の説明を参照して、-P=1 などのようにプリンタポートを設定してください。また、ネットワークプリンタは BIOS 経由出力にしてください。

場合によっては、プリンタ側に 16 進ダンプ機能があれば、それを使って -O (cf. 第 10 章 p.93) による出力と比較してみるとよいでしょう。

プリンタでの印字が遅い

原因がコンピュータ側にあるのか、あるいはプリンタ側にあるのかを確かめるには、dviprt のオプション・パラメータに-O=nul(cf. 第 10 章 p.93) をつけてプリンタ側にデータを送らないものとの速度を比べるとよいでしょう。

これで十分高速になれば、プリンタの処理速度、あるいは、データの転送速度（多くの場合、プリンタ側の受信速度）がネックで遅いと考えられます。プリンタ定義ファイル (cf. 第 11 章 p.95) を作成して、転送データを変更すると速度が向上されるかもしれません。

一方、-O=nul としても遅いときは、高速のコンピュータに変えるのが当然効果的ですが、その前に、各バッファ(cf. 第 5 章 p.51) へのメモリー割り当てがうまくいっているか、フォントに関するバッファの flash が度々起こっていないか (cf. 第 13 章 p.128) などを、チェックしてください。

PC-9801 で日本語の入った文章をプレビューするとき、起動時に画面上の全角文字がちらつくことがありますが、これはエラーではありません (高速化のためです)。

また、PC-9801 版の dvioutでは、表と裏のグラフィック・ビデオラムを使用します。これらを使用するプログラム (特に、メモリー常駐型のもの) と併用する場合はご注意ください。なお、-bv (cf. 第 5 章 p.53) のオプションで、裏のグラフィック・ビデオラムは使用しないように指定できます。

15.2　警告メッセージ一覧

`Warning: bad file` ***cache_file***

フォントキャッシュのファイルが壊れている

`Warning: No memory to read special`

1 つの special コマンドを読み込むだけのメモリーを確保できない (\special{...} の {...} の中身を、ヒープエリアに読み込みます)

`Warning: Unknown special:` *string*

DVI ファイルに解釈できない \special コマンド *string* が存在します

`Warning: No such a page:` *number*

コマンドラインで指定した *number* ページは存在しない

`Warning: can't find parameter file '`*par_file*`'`

-= で指定したパラメータファイル *par_file* が見つからない

`Warning: unable to open` *ps_file*`(PS)`

PostScript ファイル *ps_file* が見つからない

`Warning: Need temporary PBM for PS`

左右/上下反転を行なったり、1 ページ中で多数のPostScript ファイルを使用したなどの理由でテンポラリな PBM ファイルを作成する必要ができた

`Warning: too long special in` *`sp_file`*

`\special{file=...}` で読み込んだファイル *sp_file* 中のある 1 つの special コマンドの長さが大きすぎる (1000byte が限度)

`Warning: PS code exist`

`-GS=5` を指定しているが、DVI ファイル中にPostScript コードがそのまま書かれている

`Warning: Cannot get` *`pbm_file`*

PBM ファイル *pbm_file* が見つからない

`Warning: tmp_buf overflow!:` *`number`* `dots`

画像データファイルのデータの横方向のドット数 *number* が限度を超えている。横のドット数は約 14300 が限度 (拡大/縮小を含むときは処理後との和)

`Warning:` *`gif_file`* `isn't monochrome GIF file.`

gif_file は、monochrome GIF ファイルでない

`Warning: Wrong separator in GIF`

monochrome GIF ファイルが異常 (separator , が存在しない)

`Warning: Not monochrome GIF`

読み込んだファイルが monochrome GIF ファイルでない

`Warning: Not supported interlace mode for` *`gif_file`*

monochrome GIF ファイル *gif_file* が interlace mode になっている

`Warning: No working space for GIF`

monochrome GIF ファイルを処理するためのワーキングエリアがメモリーに確保できない (EMS などを使えば問題ない)

`Warning: Data error in GIF`

monochrome GIF ファイルが異常

`Warning: No memory to resize font`

(`-e=` や`-romf=` を指定したとき) フォントの文字サイズを変更するためのワーキングエリアが確保できない (EMS を使った場合は、512×512 ドットまで)

`Warning: check sum doesn't match in Font` *`code`*`: dvi(`*`sum1`*`) tfm/jfm(`*`sum2`*`)`

(`-c+` オプションを使用したとき) DVI ファイルと使用したフォントのチェックサムが一致していない (DVI ファイルを作成した TEX が使った TFM ファイルが、`TEXPK` で指定して読まれたフォントに対応するものでない。つまりバージョンが異なっている)

`Warning: tpic: visible pattern exceeds max ... v(^^;)v`

tpic specials において描く線の繰り返しのパターンが複雑すぎる

`Warning: tpic: path stack empty ... v(^^;)v`

tpic specials で記憶した通過点以上を処理している (tpic specials の書き方が不正)

`Warning: [vfont] `*`name`*`.vfn error(line:`*`number`*`)`

name.vfn の *number* 行目にエラーがある

`Warning: [vfont] Can't open `*`name`*`.vfn file.`

書体倶楽部/JG Font/TrueType Font の定義ファイルが見つからない

`Warning: [vfont] h-data error.(`*`code`*`) `*`font_file`*

書体倶楽部のフォントファイル *font_file* の *code* の文字のデータが異常

`Warning: [vfont] v-data error.(`*`code`*`) `*`font_file`*

書体倶楽部のフォントファイル *font_file* の *code* の文字のデータが異常

`Warning: [vfont] `*`font_file`*` not found.`

書体倶楽部のフォントファイル *font_file* が見つからない

`Warning: [vfont] vfn file version mismatch. x(`*`num1`*`) -> o(`*`num2`*`)`

書体倶楽部フォントのバージョンと dviout.vfn/dviprt.vfn のバージョンが不整合である

`Warning: [vfont] vfn file error. JFM(`*`name`*`) <-> Vector font No.`*`number`*`.`

書体倶楽部フォントで jfm と vfont[ ] が対応していない

`Warning: [vfont] Novec size is too large.`

書体倶楽部フォントのデータが異常

`Warning: [vfont] 10 bits data error.(`*`code`*`) `*`font`*

書体倶楽部フォントのデータが異常

`Warning: [vfont] expand error X. (`*`number`*`)`

書体倶楽部フォントのデータが異常

`Warning: [vfont] expand error Y. (`*`number`*`)`

書体倶楽部フォントのデータが異常

`Warning: [vfont] JIS level 2 is used. `*`font`*`(`*`code`*`H)`

書体倶楽部フォントで第二水準フォントファイルがないのに、第二水準文字が使われた

`Warning: [vfont] read pointer error.(`*`code`*`) `*`font`*

書体倶楽部フォント *font* の *code* の文字データの位置データが読み込めなかった

`Warning: [JGfont] JIS level 2 is used. `*`font_file`*`(`*`code`*`H)`

JG Font で第二水準フォントファイルがないのに、第二水準文字が使われた

`Warning: [JGfont] 12 bits data error.(`*code*`) `*font*

JG Font ファイルが異常

`Warning: [JGfont] `*font_file*` not found.`

JG Font ファイル *font_file* が見つからない

`Warning: [JGfont] read preamble error.(`*code*`) `*font*

JG Font *font* の *code* の文字データのプリアンブルが読み込めなかった

`Warning: [JGfont] read pointer error.(`*code*`) `*font*

JG Font *font* の *code* の文字の位置データが読み込めなかった

`Warning: Working buffer overflow for vfont`

書体倶楽部/JG Font/TrueType Font 使用のためのワーキングエリアが不足していている (`-bw=` の値を増やしてください。20000 以上、130000 程度まで増やすと効果があります)

`Warning: Cannot find `*name*`.tfm for TT Font`

欧文 TrueType Font の *name*`.ttf` に対応する *name*`.tfm` が見つからない

`Warning: [TTfont] data error.(`*code*`) `*font*

TrueType Font のデータが不正

`Warning: can't open PDLspecial file `*file_name*

\special コマンドの `lipsfile=`, `escpagefile=`, `\pdlfile=` で読み込もうとしたファイルが見つかりません

`Warning: `*PDLspecial*` exist, but not `*name*` printer`

\special コマンドの `lipsfile=` または `escpagefile=` がありましたが、対応する LBP 用の出力になっていません

15.3 エラーメッセージ一覧

`Bad System: No working space (`*number* byte`) to rotate a character`

文字を回転するためのワーキングエリアが足りない

`Bad System: Width Over.`

表示すべき幅が指定した幅より大きすぎる

`Bad System: `**テキストがプリンターのドット幅を超えています.**

印字すべき幅が指定したプリンタの印字可能幅を越えている

`Bad System: `**テキストの１ページの長さが大きすぎます.**

印字すべき 1 ページの長さが指定したプリンタの印字可能な 1 ページの長さを越えている

`Command Error: Negative Pointer(POST)`

DVI ファイルが異常 (ポストアンブルの位置が負の値である)

`Command Error:  No Postamble`

DVI ファイルが異常 (ポストアンブルがない)

`Command Error:  Negative Pointer(Last BOP)`

DVI ファイルが異常 (Last BOP の位置が負の値である)

`Command Error:  Reading Illegal Long`

DVI ファイルが異常 (num, den, mag が正の値でない)

`Command Error:  Reading Illegal Integer`

DVI ファイルが異常 (stack_depth が負、page が 0 または負になっている)

`Command Error:  No BOP command in page` *`number`*

DVI ファイルが異常 (*number* ページに BOP がない)

`Command Error:  No BOP`

DVI ファイルを解釈中 BOP が見つからなくなった (Internal Error)

`Command Error:  code:`*`code`*

DVI ファイルが異常 (使用しているフォントのリストを作成中に *code* のコードが現れた)

`Command Error:  Illegal command` *`code`*

DVI ファイルが異常 (ページ内容を展開中に未定義の *code* のコマンドが現れた)

`Device Fault:  NO GAINIT`

GA-1024A/1280A 対応版で、`gainit.exe` が常駐していない

`Illegal Args:  -y=`*`parameter`*

用紙選択指定のフォーマットが違っている

`Illegal Args:  mag step over`

`-mag=` による値が 0 から 9 までの数、あるいは 500 以上の数でなかった

`Illegal Args:  dpi`

`-dpi=` による dpi の指定が正常でない (0 または負になっている)

`Illegal Args:  -e= should be 300-4000`

`-e=` による拡大率指定の値は 300 以上 4000 以下でなければならない

`Illegal Args:  -p=`*`parameter`*

`-p=` によるプリンタ指定のフォーマットが正しくない

`Illegal Args:  環境変数 または パラメータ・ファイルで %s を設定して下さい.`

```
Example: set TEXPK=b:^d\^s.^d;dpi^d^g^s.pk;b:\font\^l.far^g^s;b:^l\^s.pxl
         ^d: dpi number
         ^l: dpi number x 5
         ^s: font name
         ^g: GTH/far file
```

TEXPK によって使用するフォントの指定をパラメータファイル dviout.par/dviprt.par で指定する場合は、そのディレクトリ名を環境変数TEXCFG で指定すること

`Illegal Args: Path name buffer is out of range`

TEXPK/TEXKNJ によるフォント指定が正しくないため、長すぎるフォントのパス名が生成された

`Illegal Args: %<c> in TEXPK or TEXKNJ`

TEXPK/TEXKNJ の「%」または「^」の次の文字が<c> となっている (d, l, s, g, f 以外は、正しくない)

`Illegal Args: bad parameter -`*`option=parameter`*

オプション指定のパラメータが正しくない *parameter* として数値を要求される場合なのにそうなっていない場合や、長さを指定するところでそのフォーマットが違っていた場合など

`Illegal Args: no match -`*`option`*

存在しないオプション *option* を指定した

`Illegal Args: Nesting of parameter file `*`par_file`*

-= によって読み込むパラメータファイルのネスティングが深すぎる

`Illegal Args: Bad parameter in -JC`

-JC= のパラメータの書き方が間違っている

`Illegal Args: -PF` **の値が異常**

-PF=1 と-PF=2 のみが可能です

`Illegal Args: -p=..v`*`number`*

-p=l または-p=m のサブパラメータv の値が異常

`Illegal Args: too much width -FAX=F`*`width..`*

FAX 出力の最大横幅のドット数 *width* が限度 (12000) を超えている

`Illegal Args: `*`chip, reso`* **の画面モードを決定できません**

DOS/V 版のdviout における-chip=、-reso= の指定が解釈できない

`Illegal Args: SVGA` **チップが指定されていません** **[例** `:-chip=1 (ET4000)]`

DOS/V 版のdviout で SVGA モードを使うときは、-chip= を指定してください

`Illegal Args: -reso=`*`number`*

GA-1024A/1028A 対応版において、-reso= の値が不正である

`File Fault: Not DVI file`

指定したファイルが DVI ファイルでない (先頭の 2byte が、247, 2 となっていない)

`File Fault: Bad page direction `*`page_file`*

起動のパラメータ@*page_file* で指定したページ指定ファイルの記述が異常

`File Fault:  No ID`

指定したファイルが DVI ファイルでないか、壊れている (DVI ファイルの末尾には 2 (pTEX のときは 3) のあと、4byte 以上続く 223 がなければならない)

`File Fault:  fail to read FONT`

フォントファイルを読み込むときにエラーが起きた

`File Fault:  `*`font_file`*` contains JXL4 file`

GTH/FAR/FLI ファイル *font_file* の中に JXL4 フォーマットの和文フォントファイルが含まれている

`File Fault:  JXL4 file `*`font_file`*

JXL4 フォーマットの和文フォントファイル *font_file* が壊れている

`File Fault:  Cannot re_open JXL file `*`font_file`*

JXL4 フォーマットの和文フォントがオープンできなくなった

`File Fault:  `*`pbm_file`*` isn't a pbm file`

pbm_file は raw PBM ファイルでない (先頭の 3byte は P 4 Whitespace でなければならない)

`File Fault:  Maybe old format:`*`cfg_file`*

プリンタ定義ファイル *cfg_file* が異常

`File fault:  Invalid TrueType font `*`font_file`*

font_file は、正常な TrueType Font でない

`File Fault:  Does not seem CFG file for dviprt::`*`cfg_file`*

cfg_file は、プリンタ設定ファイルでない

`File Fault:  Minor version mismatch:`*`cfg_file`*

指定されたプリンタ設定ファイル *cfg_file* は、対応していないバージョンである

`Font Mismatch:  `*`font_code`*

フォント番号 *font_code* のフォントが定義されていない (DVI ファイルが異常)

`Font Mismatch:  `*`font_code`*`:  `*`font_file`*` (`*`char_code`*`H)`

文字コード *char_code* の文字が *font_file* の中に見つからない

`Memory Fault:  Internal error in EMS mapping(`*`page1 page2 page3 page4`*`)`

EMS マッピングでエラーが生じた (割り当てようとしたページは、*page1*, ... Internal Error)

`-r=` で指定したフォントキャッシュ・ファイルが壊れていると、このエラーが生じることがあります (このときは、キャッシュ・ファイルを削除するか、`-r=` オプションをはずす)

`Memory Fault:  Internal error in moving EMS memory(`*`page:off -> page:off`*`)`

EMS におけるメモリーのコピーでエラーが生じた (Internal Error)

`No File: file_name`

DVI ファイル、フォントファイル、ページ指定ファイル、あるいは \special{file=*file_name*} で指定したファイルが見つからない

`No File: VFN file file_name`

VFN ファイル *file_name* が見つからない

`No File: PK file file_name`

PK ファイル *file_name* が見つからない

`No Font: Any of the above.`

フォントファイルが見つからない (このメッセージの上に、探して見つからなかったフォントのパス名がすべて示される)

`No Memory:` **バッファ領域の不足**

メモリーの空きが不十分なため指定したサイズのバッファが確保できない

`No Memory:` **フォントバッファ領域の不足**

メモリーの空きが不十分なため、DVI で使われているフォントの情報のためのバッファが確保できない

`No Memory:` **ヒープエリアの不足**

ヒープエリアのメモリーが足りない (使われているフォントのパス名、書体倶楽部フォントなどの展開、tpic specials の解釈などにヒープエリアが使われる cf. 第 5 章 p.53)

`No Memory: Not enough bitmap buffer`

ビットマップバッファが小さすぎます (このバッファは LBP へのダウンロードフォントを決める際にも用います)

`No Memory: font file over: font_file`

フォントファイルが大きすぎてフォントファイル用のバッファに読み込めない (大きな PK フォントの場合は、PKD ファイルが必要です)

`No Memory: Too big font`

dviout/dviprt で扱えるサイズを越えるサイズの文字があった

`No Memory:` **ハイレゾ用ワークエリアの不足**

メモリーが少なくて PC-9801 ハイレゾ用のワークエリアが確保できない

`No Memory: Working`

LIPSIII でのラスタデータ圧縮のためのワーキングエリアを確保できない (転送の際の縦のドット数の単位を減らす)

`No Memory:` **スタック領域の不足**

DVI ファイルにあるスタック領域がヒープエリアに確保できない

`No Memory: near heap for FLI`

FLI フォントライブラリからフォントをサーチする際のヒープエリアのメモリーが不足

`No Memory: tpic: path stack full ... v(^^;)v`

tpic specials を解釈するのにヒープエリアが足りない (途中の通過点を記録するためのメモリー)

`Program Stop: Making font table`

使用しているフォントを調査中にエラーが起きた (Internal Error)

`Program Stop:` **印刷可能領域がありません：サイズ関係オプション要チェック**

印字領域の横幅または縦長が正になっていません (新サイズオプションの指定を修正してください)

`Program Stop: Cannot write font cache`

`-r=` オプションによるフォントのキャッシュの書き込みに失敗

`Program Stop: Stack Over flow`

DVI ファイルで定義されている以上の STACK を使っている (DVI ファイルが異常)

`Program Stop: Stack Under flow`

DVI ファイルにおける STACK の PUSH と POP の対応が異常

`Program Stop: Twice fnt_def`

DVI ファイルで同じフォントが二重に定義されている (DVI ファイルが異常)

`Program Stop: No such a page : dvifile-page :` *number*

コマンドパラメータで指定した DVI ファイルの (先頭から数えての) *number* ページが存在しない (起動時のページ指定のエラー)

`Program Stop: badly formed PS command:` *string*

PostScript special における `\special {postscriptbox=...}` のフォーマットが正しくない

`Program Stop: Cannot execute` *program*`:`*error_no*

PostScript を処理するため *program* を呼びだそうとしたが、実行できない

`Program Stop: Too many PS codes in a page`

1 ページ中に含まれる PostScript specials の数が多すぎる (990 個が限度。ただし、同じ条件で同一の PS/EPS ファイルを呼び出す場合は、最初の異なった 32 個までは数を加算しない)

`Program Stop: Can't open temporary PS file`

DVI ファイル中の PostScript コード処理のためカレントディレクトリにテンポラリファイルをオープンしようとしたが、できなかった

`Program Stop: Can't open configuration file "`*cfg_file*`"`

プリンタ定義ファイル *cfg_file* がオープンできない

`Program Stop: Can't open "`*out_file*`"`

`-O=` で指定した `dviprt` の出力ファイル *out_file* がオープンできない

`Program Stop: Can't open` *`ini_file`*

`-I=` で指定したプリンタの初期化コード記述ファイルがオープンできない

`Program Stop: Divided by zero in the CFG of printer`

プリンタ設定ファイルの式を評価中に 0 での除算が発生した

`Program Stop: Stack not initialized`

DVI 用のスタックの初期化をしていない (Internal Error)

`Program Stop:` **指定された解像度 (*res*) はサポート されていません**

DOS/V の VESA 対応ビデオボードで解像度 *res* はサポート されていない

`Program Stop: Can't open template file`

フォント自動生成機能で参照する template file が見つかりません

`Program Stop: Illegal statement in template`

フォント自動生成機能で参照した template file の内容が不正です

`Unpack Error: More bits than required`

font_code:*font_file*
フォントファイルのデータが不正 (ディスクのフォントファイルが壊れていたか、メモリーにロードされた後で壊れた)

`Unpack Error: Illegal font ID:` *`font_file`*`(<ID>)`

`Check TEXPK!`
font_file は不適切なファイルである (たとえば、欧文の TFM ファイルは不適切)

`Unpack Error: Too big raster in` *`font_file`*

font_file の中の`dviout`/`dviprt` で扱える以上のサイズの文字を使っている

`Unpack Error: Too big vector in` *`font_file`*

font_file で`dviout`/`dviprt` で扱える以上のサイズの文字を使っている (VFD ファイルの場合)

`Unpack Error: Unexpected` *`code`*`!`

PK フォントが壊れている (文字データをサーチ中に不正なコード *code* を検出した)

第16章 制限

dviout, dviprt はいろいろな意味で制限が少ないのが特徴です。

例えば、1 ページ中の文字や線の数の制限はありません。また、扱える 1 ページの大きさは、事実上制限がないと考えてよいでしょう。

プレビューアでの横スクロールは、6720 ドットまでという制限がありますが、それより大きくても、-X、または、-OX を指定すれば、その右側も見ることができます。この制限は、プリンタドライバにはありません。

DVI ファイルで使用できるフォントの種類の数は最大で 256 ですが、問題なく扱えると思われます。ただし、プログラムが 1 度に同時にオープンするファイルの数の最大値は

2 ＋ 〈実際に使用するディスク上の和文フォントファイルの数〉
＋ 〈PKD ファイルを仲介にしてアクセスする PK ファイルの数〉
＋ 〈使用する書体倶楽部 /JG Font/ 和文 TrueType Font のファイル数〉

となっています。これらのファイルは、できるだけオープンしたままにしますが、その数が多すぎれば、必要に応じてクローズします。デフォルトでは、dviout/dviprt 終了時にその直前にオープンしていたファイルの数が表示されます。

TUG(TEX ユーザーズグループ) によって TEX のデバイスドライバが満たすべき要件について報告がなされていますが、それらはすべてクリアーしていると思われます。

ただし、tpic specials などの拡張機能については、その複雑さなどにつき限界があります。

16.1 処理能力の限度

dviout/dviprt の処理能力の限界の概略をまとめてみました (cf. 16.1 p.147)。

2 byte 整数 (−32768 から 32767 までの整数) のドットサイズで表わされ、その範囲を目安とする限界をPIXEL と書くことにします。

また、local heap のサイズ (35K byte 程度) を目安とする限界をLOCAL で示すことにします。

機能	限界
DVI ファイルのサイズ	特に制限なし
DVI ファイルの総ページ数	10000 ページ
DVI ファイルで使われるフォントの種類	TEX での制限の 256(それ以上も可 ?)
	フォントのパス名は local heap に保存
ページのサイズ	PIXEL
ページの位置補正	PIXEL
描かれる縦線、横線の幅と長さ	PIXEL
1 ページ中の文字や線の数	特に制限なし
1 ページ中の PBM/GIF 画像数	特に制限なし
1 ページ中の PS/EPS ファイル数	1000 回程度まで可 (同一画像ならより多くてよい)
取り込める画像サイズ (拡大/縮小無し)	幅は 14300dot まで、高さはPIXEL
取り込める画像サイズ (拡大/縮小あり)	拡大／縮小の前後の幅の合計が 14300dot まで
文字サイズ (LBP 内蔵和文フォント)	LBP に依存。LIPS III では、50cm 以上でも可
文字サイズ (上以外の一般的上限)	幅は 8000dot、高さはPIXEL
varf+ による文字の拡大/縮小	上と同様
PK 形式欧文フォントの 1 文字のサイズ	圧縮後の 1 文字が 64K byte
PK/PH 和文圧縮フォントの 1 文字のサイズ	同上
JXL4 形式フォントの 1 文字のサイズ	同上
PK 形式欧文フォント	PKD ファイル使用で、ファイルサイズ 8M byte
PK 和文圧縮フォント	ファイルサイズが 4M byte
PH 和文圧縮フォント、JXL4 形式フォント	ファイルサイズは特に制限なし
KG 和文フォント	ファイルサイズは特に制限なし
PXL1001, 1002, 1003 形式フォント	ファイルサイズが 64K byte
pTEX 縦書きの JXL4 フォント	1 文字の幅と高さが 400dot 程度
pTEX 縦書きで-g+ の回転補正文字	幅と高さが 500dot 程度
起動時の各オプションパラメータ	4000byte
1 つの\special{...} の長さ	LOCAL
\special{file=...} のファイル	ファイル中の各 \special の内容が 1000byte
tpic specials の処理用のワーク	LOCAL
-e による拡大/縮小	4 倍から 0.3 倍まで
-mag による拡大/縮小	30 倍から 0.5 倍まで
G3 FAX 形式での圧縮画像出力	幅は 12000dot、高さはPIXEL

表 16.1: 処理能力の限界

第17章 配布

改変を行なわない限りdviout/dviprt の配布は自由です。全てのファイルを一括した配布を望みますが、強制するものではありません。一括配布でない場合にも、この配布の規定は伝えるようにしてください。gather, chkfont, fax2pbm といったユーティリティーは、独立したプログラムですので分離することに問題はありません。

改変した場合は、オリジナルの copyright を明記した上で、改変したことが分かるように明示されていれば、再配布も可能です。(例えば、copyright の表記に自分の名前をつけ加えて、2.43.2A のように特別のバージョン番号を付け加え起動時に表示するなど)

ソースファイルの流用は自由です。ただし、jdvi2kps から引用したepsbox.c の一部を除きます。これにつきましては、ソースのアーカイブに含まれる、該当ファイルのコメントとcopyrigh.t をご覧ください。(dviout/dviprt の一部として、無改造で配布されることに関してはご承諾いただいています。)

第18章 補記

このドキュメントは、tex.doc Ver.2.12 を Y. Kusumi 氏の手により日本語 LaTeX のテキストに変換されたもの (Sep. 3, 1990) とその後の各種拡張をされた方のドキュメントをもとに、SHIMA が作成しました。

Ver.2.35–2.37 においては、sempa 氏により、加筆・修正が行なわれました。

その間の作成に当たって千秋氏、吉村氏の協力を得るとともに、Ver.2.39 および Ver.2.42 においては、とがし氏により若干の加筆と大幅な整理が行なわれました。また、プリンタの定義ファイル記述の部分は淺山氏に、表紙の白抜き文字は、松田氏によるものです。

PC-9801 版dviout/dviprt Ver.2.33 が公開された後、DOS/V や J-3100 などの各機種に対して多くの方々による移植が行なわれ、書体倶楽部フォント対応などの各種機能拡張も行なわれていました。それが、sempa 氏によって、Ver.2.34 の形に統合され、DOS 版の TeX のデバイスドライバとして広く使われるようになりました。

その後も、tpic specials、pTeX、NTT JTeX、PostScript、LBP を含む各種プリンタへの対応などの改良が多くの方々により行なわれていますが、これは sempa 氏の統合があってはじめて可能になったのだ思います。開発者、ユーザを代表して、Ver 2.34–2.38 のお世話をしてくださった sempa 氏に対し、深く感謝いたします。

このような経緯を経て、dvout/dviprt は、対応機種や機能の拡張が行なわれてきたわけですが、そのすべてを SHIMA は把握していませんし、リストアップすることも手に余ります。しかしながら、最後に例としていくつかの拡張を取り上げ、その拡張に最初に取り組まれた方、あるいは、中心的役割をされた方を以下に述べさせていただき、感謝の意を表わしたいと思います。

PC-9801 ハイレゾモード	: OkI
DOS/V	: SOLITON, hero.h
AX	: Akiii
J-3100	: sempa, hero.h, 千秋
HP100LX	: HARUYA
ESC/Page	: 冨家, OkI
LIPS III ESC/Page 統合	: OkI
tpic specials	: Oh-Yeah?
新サイズオプション	: 吉澤
pTeX	: Naochan!
NTT JTeX	: 八雲
書体倶楽部フォント	: 吉田秀樹, Minagawa, 向内
JG Font	: Naochan!
TrueType Font	: 松田
FLI 形式フォント	: 吉崎栄泰
システムフォントの縮小/拡大	: Naochan!, 吉澤, 松田
不足フォント自動生成機能	: 八雲
PostScript	: 内山孝憲
PBM 画像データ	: 八雲
GIF 画像データ	: 淺山
ページ記述言語埋め込み	: 岩井貞之
プリンタ定義ファイルの拡張書式	: 淺山

<< Sep. 22, 1990 – March 23, 1996 by SHIMA >>

索 引