計算言語学 17－2
（1979. 2. 23）

# 漢字ドット・フォントからベクター・フォントへの変換

間下 浩之
日本アイ・ビー・エム(株)東京サイエンティフィック・センター

## はじめに

漢字仮名混り文の処理が各所で行なわれるようになり，各種の漢字入出力装置[1][2]が出まわってきた。東京サイエンティフィック・センターでは計算機の適用分野の拡大を目的に，日本語処理，画像処理，図形処理等の研究を行なっているが，特に日本語処理のためには漢字入出力の段階での融通性が必須となる。漢字を取扱う適用分野，たとえば情報検索システム，高級編集システム等，における会話型システムにとっては表示装置が最大の問題となり，以下の要件があげられる。

①漢字・画像・図形を同時に表示したい。
②拡大・縮小・回転を容易に自由に行ないたい。

これらの要件を満足して画像・図形を容易に表示可能な映像表示装置（IBM 3250映像表示装置[3]，3277 GRAPHIC ATTACHMENT＋映像表示装置[4]等）に同時に漢字等の文字も表示することを考えた。②の要件により字体発生方式はドット方式と比較してストローク方式（ベクターによりデータを持つ）の方が良い。しかしハードウェアによる文字発生は以下の理由で困難なため，ソフトウェアで補うことにした。

1)文字によってデータ量が異なるデータである。[5]
2)表示制御部の論理が複雑になる。[5]

一般にドット方式はかなり普及しているため，数多くの字体が存在するが，ベクター方式による字体（ベクター・フォント（ベクター字体））はほとんど手にはいらない状態である。本報告はドット字体からベクター字体作成のための変換を行なったので，その方法について述べる。また字体としては18×18骨組型字体，32×32明朝体について実験を行なったのでその差異についても述べる。

## 1．18×18字体による実験

### 1.1 前提

① 字体は骨組型で厚みを持っていないものとする。すなわち，ゴチック体や明朝体でなく，表示装置用として使用するような18×18の文字である。

② 結果のデータはX，Y座標の相対番地（X，Y＝0～17）で表わす。始点・終点の対で情報をとるとデータ量が多くなるため，前の線分の終点と次の線分の始点が同じ場合は片方が省略できるようにする。ただしデータ中に制御文字（CC）を挿入した場合は，ブランク・ベクターが引かれると考える。すなわち図1の文字は右のようなデータになる。これによりデータ量は最小となる。

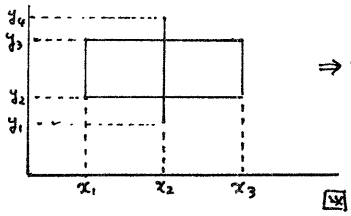



図1

⇒ $x_1\ y_3\ x_1\ y_2\ x_3\ y_2\ x_3\ y_3\ x_1\ y_3$ CC $x_2\ y_4\ x_2\ y_1$

このためにデータは筆順とは異なる。

## 1.2 走査の方法

①処理1

図2のような2次元の対象にたいして、縦，横，ななめ（45度）の4通りの走査を行なう。4通りの走査によって得られた線分の始点と終点の値を表1に挿入する。表1の左側の数字は18×18の2次元配列の番地を表わす。表の5列のうち左4列は4つの各方向によって2どの番地と結ばれたかを示す。4通りの走査を方向性を持たせないように互いの番地に同じ情報を持たせるようにする。たとえば番地1は①の走査で番地5と結ばれ，番地5は④の走査で番地1と結ばれている。一番右の列は結ばれた線分の本数である。始点、終点の決定については後の規則の項でふれる。

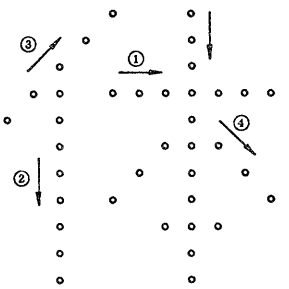


図 2

②処理2

表1の性質として"数が奇数である番地は，必ず始点か終点になりうる。"ということがいえる。処理2ではこの表を図1に示したようなベクター のデータにすることが役割である。処理を2回のパスに分け1回目には数が奇数である番地をすべて捜す。はじめに番地3が見つかったら、その数から1を引き4つの列のうちデータのある番地37へ飛ぶ。番地37で同じ処理が行なわれ，数がなくなる番地まで進む。その後は番地4から同様の処理が行なわれる。この操作を最後まで行なうと必ず表1は数が偶数ばかりの番地だけになる。2回目のパスは数が1以上であるものすべてに関して1回目と同じことを行なう。この操作で結果のデータ量は最小となる。

| | → | ↓ | ↗ | ↘ | 数 |
|---|---|---|---|---|---|
| 1 | 5 | | | 324 | 2 |
| 2 | | | | | |
| 3 | | | 37 | | 1 |
| 4 | | | | | |
| 5 | 1 | | | | |
| | | | | | |
| | | | | | |
| 322 | 324 | | | | 1 |
| 323 | | | | | |
| 324 | 322 | 288 | | 1 | 3 |

表 1

## 1.3 実験1

処理1の始点・終点の決定において何の規則ももうけないと図3のように引かなくてもよい線分をよけいに引くことになる。

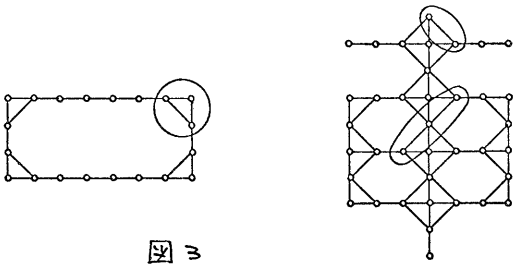


図 3

これを防ぐために簡単な基本規則をもうける。ここである線分の両はじの点を端点，それ以外の点を通過点と呼ぶ。

**基本規則1**

ある線分を考えた場合，それに所属するすべての点が他の線分の通過点であるならば，その線分は引かない。

この規則に従って簡単な実験を行なった結果以下の問題が生じた。

図4がこれを表わしているが、引かなければいけない線分が別な線分（この場合はななめの線分）と干渉を起こして引かなくなってしまっている。漢字にはこのようなパターンが数多く出現する。この点を考慮して基本規則を多少修正したものが基本規則1′である。

図4

基本規則 1′

基本規則1の条件を長さ1のものだけに適用する。（長さ1とは端点どうしを結ぶ線分を表わし、今後長さmとはm＋1の点を包含する線分を表わす）。ただしななめの線分に関してはこれをあてはめない。

国立国語研究所、現代雑誌90種用字用語調査における上位2000字（頻度順）について、基本規則1′を適用して行なった実験結果は以下の通りである。

| | | |
|---|---|---|
| 成功 | 1346字 | 67.3% |
| 失敗 | 654字 | 32.7% |

失敗例を図5に示す。

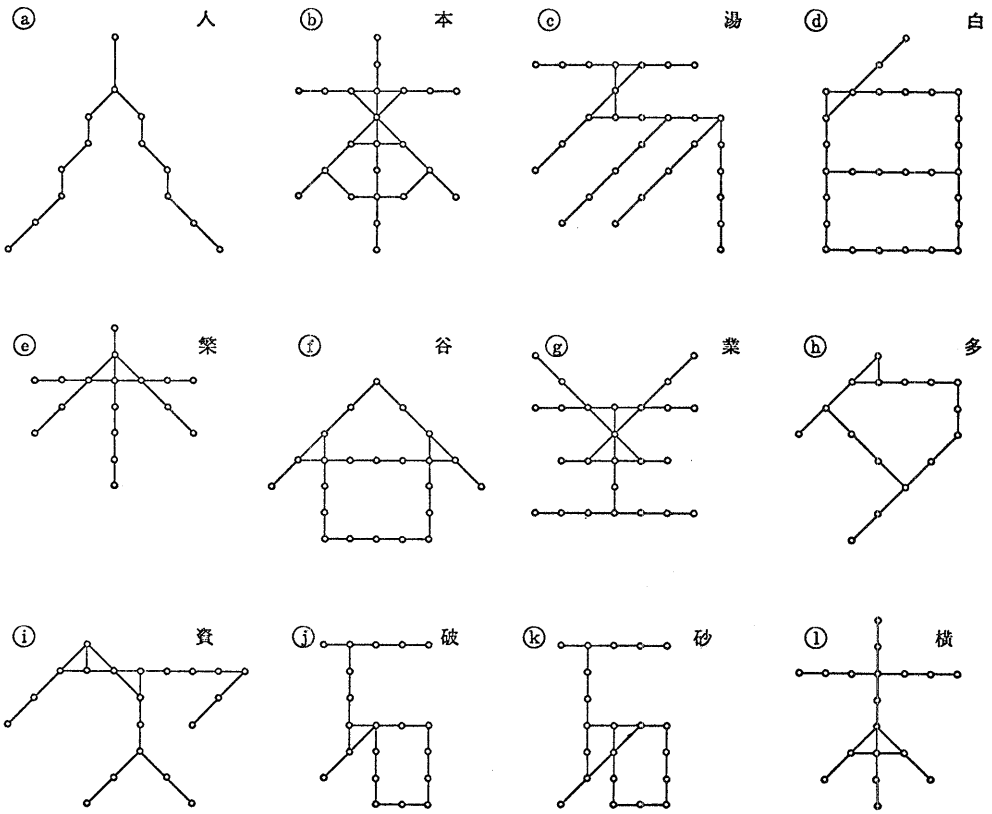



図5

結果の考察

なな め の線分が問題になると思われるが，ⓑ，ⓒ，ⓕ，ⓗ，ⓙ，ⓚは縦，横の線分が問題となった。ⓐに関しては当然予想される結果であるが，この形を持つ字が1%程度あった。

## 1.4 実験2

実際に使う立場として，標準の大きさの0.5倍から1.2倍程度まではこれらの欠点が目立たないが，それ以上に拡大した場合にこの成功率では問題がある。そこで成功率90%以上に向上させるために以下の特別な追加規則をもうけて実験してみることにした。

追加規則

1．縦，横用

①長さ2で，すべての点が他の線分の通過点であり，真中の点を通る他の線分がその点によって長さ2以上ずつに分割されている場合，この線分は引かない。図5のⓑ，ⓚの修正である。

②つぎのパターンを持ったものは端点をずらす。図5のⓕの修正である。

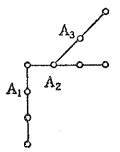
ⓐ長さ1で1つの端点が他の線分の通過点で，もう1つの端点が別の長さ2以上の線分の端点のつぎの2点が他の線分の通過点である場合は除く。次の図の様な場合が考えられる。図5のⓗ，ⓚの修正である。

例外を考慮した場合　　例外を考慮しなかった場合

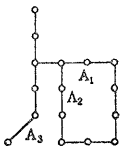
（ここではななめの線を考慮しない）

2．ななめ用

①線分Xの端点$A_1$が他の線分の通過点であり，端点のとなりの点$A_2$が長さ3以上の他の線分の端点又は他の線分の通過点である場合，線分Xの端点は$A_2$となる。ただし$A_2$が他の線分の通過点であり$A_3$が長さ3以上の他の線分の端点であるときには，線分Xの端点は$A_3$となる。




ⓑ，ⓓ，ⓔ，ⓘの修正




ⓔ，ⓚの修正

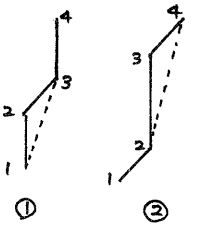
② 長さ1で1つの端点が他の線分の通過点で，もう1つの端点が他の線分の端点で$A_1$あるいは$A_2$に点があるとき，この線分は引かない。


ただしこの規則は必ず①の規則の後に適用した方が有効である。②は①，②とあてはめると正常になる。

3．特別用

④のために特別に作られた規則である。処理1，処理2が終了した後に，制御文字から制御文字までの線分の数が3以上のものを検査する。その時にX，Yそれぞれの差分をとり，それらが以下のパターンになっているものは調整を行う。

| | $X_1Y_1X_2Y_2X_3Y_3X_4Y_4$ データ | | | | | | 差分 |
|---|---|---|---|---|---|---|---|
| | $X_1-X_2$ | $Y_1-Y_2$ | $X_2-X_3$ | $Y_2-Y_3$ | $X_3-X_4$ | $Y_3-Y_4$ | |
| ① | 0 | m | 1 | 1 | 0 | m | |
| ② | 1 | 1 | 0 | m | 1 | 1 | |




1の部分は−1でもよい。①の場合は$X_2Y_2$を，②の場合は$X_3Y_3$を削除してそれぞれ$X_1Y_1$と$X_3Y_3$を，$X_2Y_2$と$X_4Y_4$を連結する。

このような追加規則によって同じ2000字を実験した結果は以下の通りである。

| | | |
|---|---|---|
| 成功 | 1936字 | 96.8% |
| 失敗 | 64字 | 3.2% |

失敗例を図6に示す。

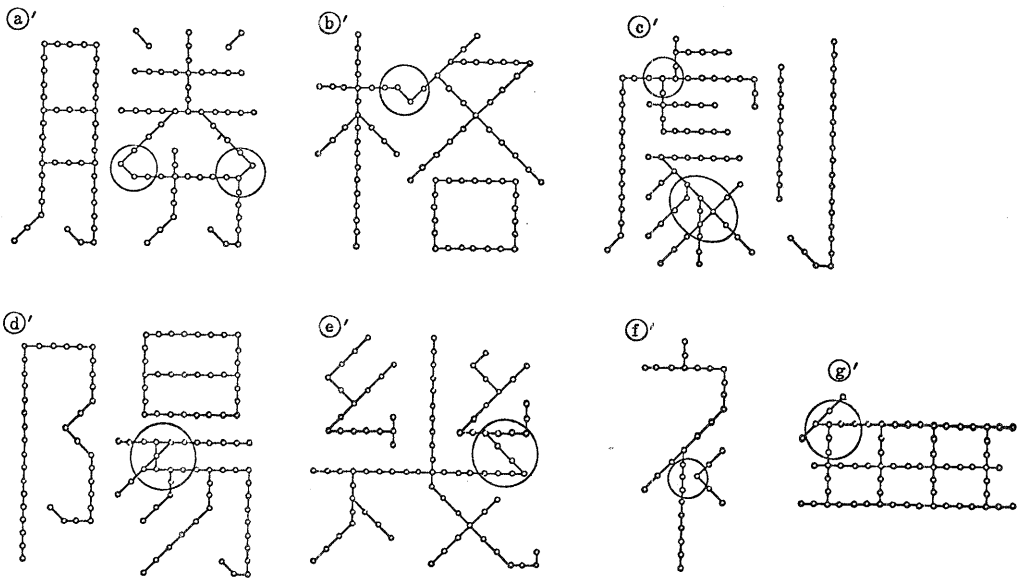



図 6

結果の考察

ⓐ′, ⓑ′, ⓒ′, ⓔ′に見られるように，失敗の大部分は連結してはいけないものを連結している現象である。これらは小さな範囲しかチェックしない規則では解決できず，困難な問題である。ⓐは追加規則1－①の拡張で解決できるかもしれない。ⓕ′, ⓖ′は追加規則1－②によって引き起こされた現象である。正解は片方が延長されて連結しなければならないが，厳密に見なければ問題ないかもしれない。

## 2．32×32字体による実験

### 2.1 前提

① 字体はEP刷装置等にも使用可能な32×32ドット・マトリックス明朝体である。

② 結果は18×18の場合と同様のデータ形にする。

### 2.2 走査の方法

18×18と同じ方法で実験をすると，厚みを持つ字体のためまわりぶらにぎこちなさが目立つ。それを解消するため1.2における4方向に30度，60度という角度の方向4つを追加してみた。図7のように8方向の走査を行なう。よって1.2の処理②における表1は4列追加されることになる。当然のことながら行は（32×32－18×18＝700）行追加となる。

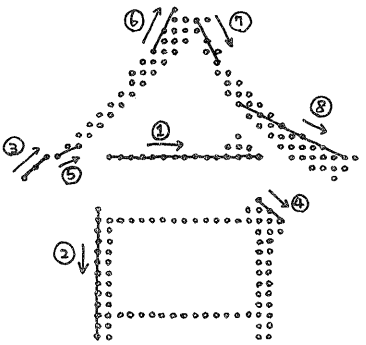



図7

| | → | ↓ | ↗ | ↘ | → | ↗ | ↓ | ↘ | 数 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 5 | | | 6 | | 8 | | | 3 |
| 2 | | | | | | | 5 | | 1 |
| 3 | | | 5 | | | | 2 | | 2 |
| 4 | 150 | | | 100 | | | | | 2 |
| 5 | 1 | | 3 | | | | | | 2 |
| 6 | | | | 1 | | | | | 1 |
| 1022 | | | | 1024 | | | | | 1 |
| 1023 | | 1000 | | | | | | | 1 |
| 1024 | | | | 1022 | | | | | 1 |

表2

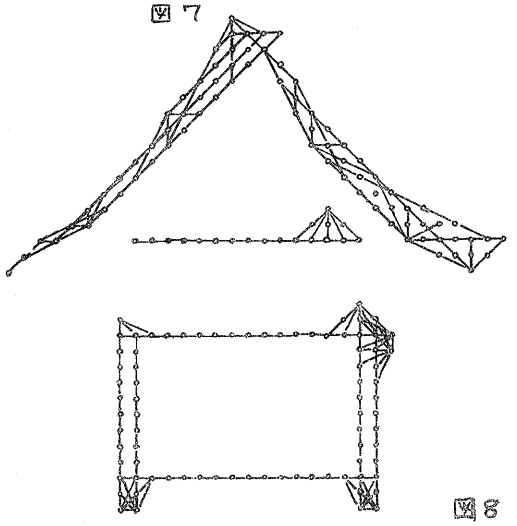


図8

18×18の基本規則1を用いて実験を行なうと図8のようになる。厚みのある字体のため、ふちどりは重要であるが内部は線分を引かなくてもかまわない。図8においてはまだ余分な線分があるため次の規則によってそれらの消去を試みた。

基本規則　2

ある線分を考えた場合、それに所属するすべての点のうち1点以外が他の線分の通過点であるならばその線分は引かないこととする。ただし新たに追加した30度と60度の4方向については、すべての点が通過点である場合に限り線分を引かない。

この規則によって実験した結果を図9に示す。

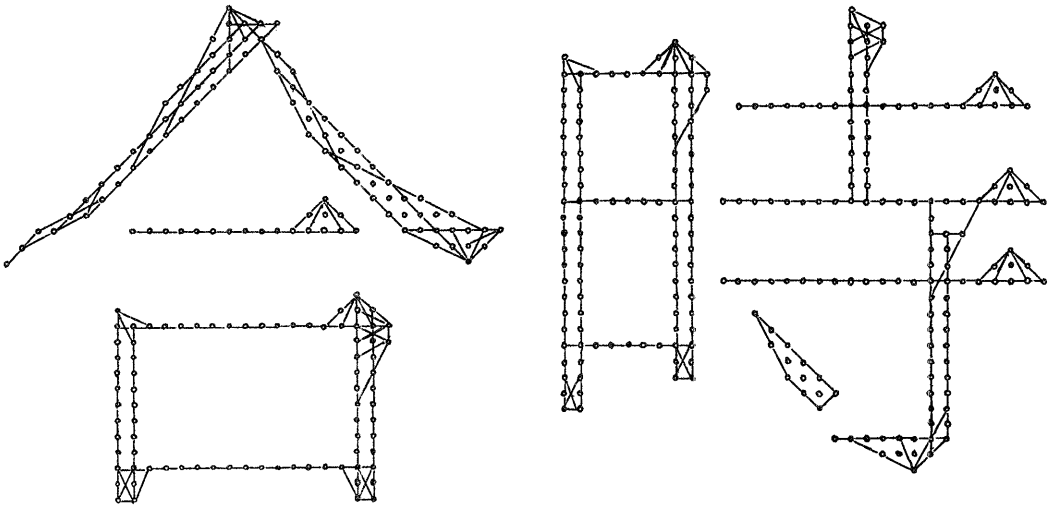


図9

これにより余分な線分はかなり取り除かれる。図8に比べてデータ量としては1～2割程度減少しているように思われる。

このように24×24，32×32，48×48等の字体において、すべてのストロークが厚みを持つならばOCRのような技術によってふちどりを行ない，後で中ぬりをする算法の方が有効かもしれないが，ストロークにより厚みがないものが存在すると論理が複雑になり，上記の規則のような方式が良いと思われる。32×32字体においてもすべての字（所持している漢字）について行なったわけではないので，それほどは，きりしたことはいえない。また，たとえふちどりができたとしても，中ぬりの処理をハードウェアで行なえなければその算法が困難なためうまくいかない。

3．字体編集システム

18×18字体の誤り修正や新たな漢字を追加する場合のために字体編集システムを作成した。このシステムでは図10のような格子状の座標を画面に用意し，使用者が図形を見ながら適当な修正が可能となるようにした。各ストロークは方向性を持つデータとして取扱っているため筆順の変更が可能になる。これによりデータ量は増加するが，他の適用分野（たとえば，電子計算機書道，計算機教育）にも有効なデータとなる。以下にシステムの命令を示す。

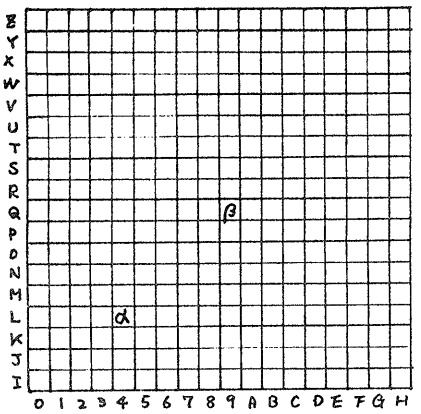



図 10

図の点αはL4ヌは4Lで表わす。αからβへのストロークは4L99，4LQ9，L499,L4Q9のいずれかで表わす。このストロークには方向性を持たせるために始点は始めに終点を後に書き，画面上には終点に矢印が書かれる。編集作業は現在ポインターとしてカレント・ストロークと呼ばれるストロークを対象に行なわれる。カレント・ストロークは二重輝度によって表示される。

命令群

① Bottom
データ中の最後のストロークをカレント・ストロークにする。

② DELete
カレント・ストロークを消去する。

③ Edit <漢字符号>
字体編集モードにはいる。指定された漢字符号のデータがファイルから作業域に読みこまれる。

④ FILE
字体編集モードを終うし作業域の結果をファイルに書きこむ。

⑤ Input
字体入力モードにはいる。今後入力されたストロークはカレント・ストロークの後に順に挿入される。字体編集モードには空白行によってもどる。

⑥ Locate <ストローク>
指定されたストロークをカレント・ストロークにする。

⑦ Move
字体接続モードにはいる。今後既存のストロークを入力することによって，そのストロークをカレント・ストロークの後に続ける。筆順変更などに使用される。

⑧ Next
カレント・ストロークの次のストロークをカレント・ストロークにする。

⑨ Point <現在点> <変更点>
現在点によって指定された点を始終点とするすべてのストロークにたいして，現在点から変更点への移動が行なわれる。

⑩ QUIT
字体編集モードを終うする。ただし作業域の結果はファイルに書きこまれない。

⑪ Replace <ストローク>
カレント・ストロークを消去し指定されたストロークをカレント・ストロークにする。指定されたストロークは既存のものでなくてもよい。

⑫ Shift <水平移動量> <垂直移動量>
字体全体を指定された移動量に従って平行移動する。

⑬ TOP
データ中の最初のストロークをカレント・ストロークにする。
⑭ Up
カレント・ストロークの前のストロークをカレント・ストロークにする。
このような機能によって字体の編集は容易に行なえるため，変換が100%になる必要はない。又、自分の好みの図形，地図マーク等も容易に漢字符号として入力可能なため、他の適用分野においても広い利用が可能である。なお，このような座標指定による編集でなくカーソル入力による編集も現在考慮中である。

## 4. おわりに

18×18字体において7000字を変換して実際に使用しているが，現状では1.2倍以下で使用されることが多くこの程度ではわざわざ編集しなくても何ら問題は起きない。なお7000字のデータ量の平均は，各点及び制御文字を2バイトとして計算すると100バイト弱になる。32×32字体は実験的に1000字程度しか変換していないがおおまかの傾向はつかめた。この2つの字体の実験によって世の中に存在する字体の何割かは同様の規則に従って変換することが可能となるだろう。最初からベクター字体を設計するよりもこの変換によってかなりの労働が削減可能と思われるので，既存の映像表示装置に漢字を表示することも簡単になると思われる。

## 謝辞

漢字字体編集システムを作成していただいた東京サイエンティフィック・センター研究員の宇野栄氏，また議論に参加していただいた諸研究員の方々に感謝いたします。

## 参考文献

[1] 荒井・桑原「実用の輪広がる漢字情報処理」 日経エレクトロニクス pp.76～pp.113 1978 12-11
[2] 菅井「漢字入出力装置開発の現状と展望」 コンピュートピア pp.52－pp.60 1978.9.
[3] 萬地・遠藤・首藤「漢字ディスプレイ技術」 日本語情報処理シンポジウム報告集 情報処理学会 pp.220－pp.227 1978 7.17～20
[4] IBMパンフレット「IBM3250映像表示システム」 N:G518－5083
[5] IBMマニュアル「IBM3277 display station Graphic Attachment」G.I. GA33-3039
[6] 国立国語研究所「現代雑誌九十種の用語用字(2)」 秀英出版