U.D.C. 681.327.13.072.2

ワークステーションにおけるビットマップ技術

Bit Map Display Technology for Workstation

福永　泰*　*Yasushi Fukunaga*
藤田　良*　*Ryô Fujita*
平沢宏太郎**　*Kôtarô Hirasawa*
仙田修一***　*Shûichi Senda*
西田健彦***　*Takehiko Nishida*

インタラクティブに処理を遂行するワークステーションでは，ユーザーに理解しやすい画面を出力するためビットマップ制御という新しい技術を用いた出力方式を採用することが一般的になってきた。

従来方式とビットマップ方式との差異は任意の位置への高速な文字表示とマルチウインドウ制御にある。そこで，まず，高速な文字表示を実現するハードウェア技術を紹介し，これをカラーのビットマップ方式に拡張するため新たに開発したハードウェア技術について詳述する。

一方，マルチウインドウ制御に関しては，従来のイメージで画面を管理する方法ではなく，コードで画面を管理する方式を提言し，このとき，必要となる四つのハードウェア機能について考察した。

1　緒　　言

ワークステーションでは，対話的に仕事を処理する環境をユーザーに提供するため，マンマシンインタフェースが重要な要素となる。中でもディスプレイを用いた出力技術については，従来の文字出力だけでなく，文字・図形・イメージが自由に融合した形で出力できるビットマップ技術を実現することで，より自然に計算機とインタフェースすることができる[1),2)]。

ここでは，**2**章でビットマップ技術を紹介し，**3**章で，特にカラー表示への適用例について概要を説明する。また，**4**章でマルチウインドウ制御を柔軟に行なうためのハードウェアの機能について考察する。

2　ビットマップ表示技術

従来の端末やパーソナルコンピュータの表示装置でのキャラクタの表示位置は，**図1**(a)に示すように，画面をあらかじめ決められた枠に分割し，各枠内に1文字ずつ表示する方式が主体であった。このため，文字と図形を合わせて表示しようとすると，

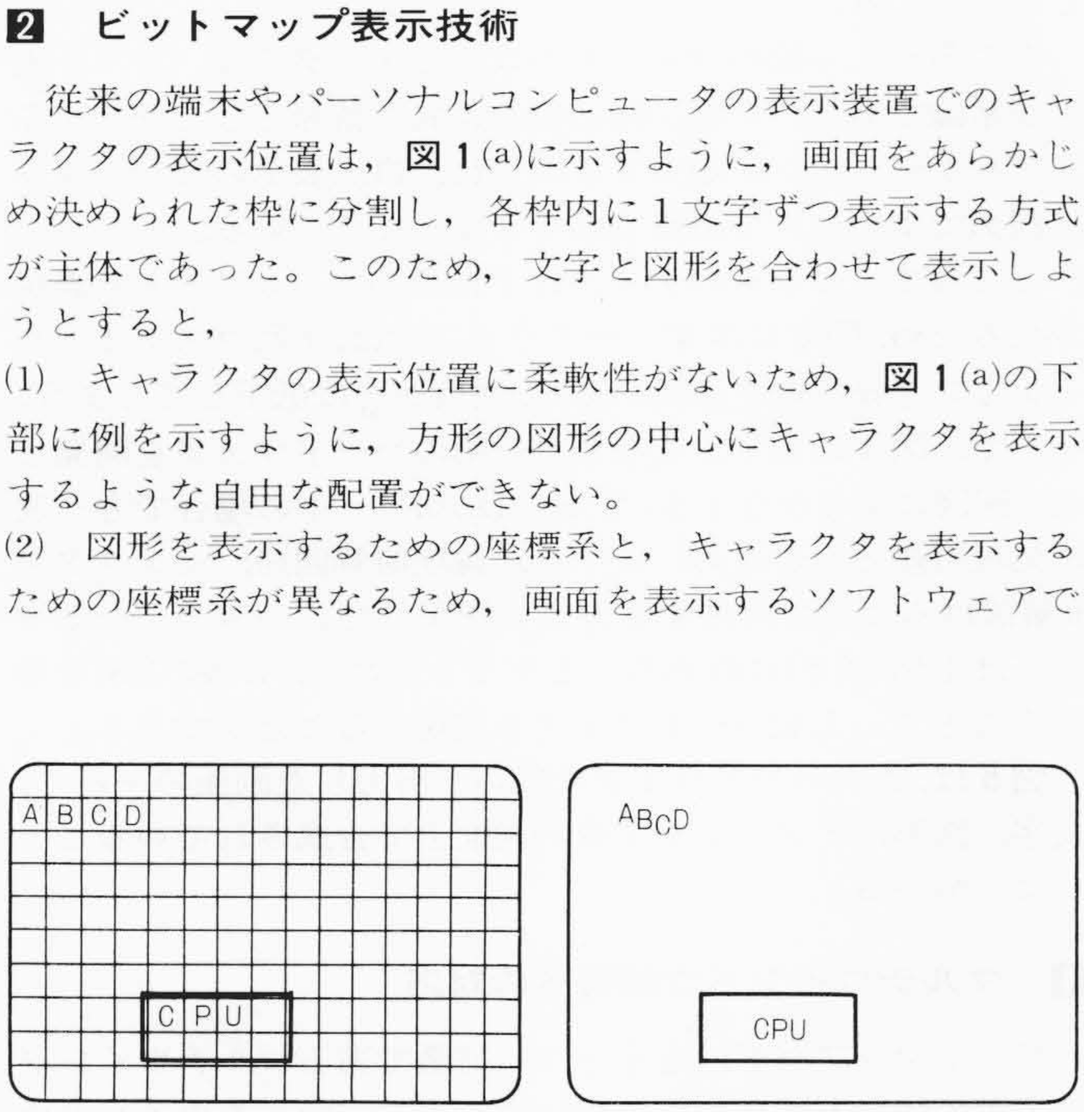
(1)　キャラクタの表示位置に柔軟性がないため，**図1**(a)の下部に例を示すように，方形の図形の中心にキャラクタを表示するような自由な配置ができない。

(2)　図形を表示するための座標系と，キャラクタを表示するための座標系が異なるため，画面を表示するソフトウェアでは二つの座標系を意識しなければならない。

という問題があった。これに対し，**図1**(b)に示すように，文字も画面上の任意の位置に置けるようにすれば，上記のような問題を解消することが可能である。従来のグラフィックシステムではこのような表示が実現されているが，文字の表示を線分列によって行なうため，表示速度が遅いという欠点があった。特にテキスト列を扱うようなソフトウェアを実行する場合問題となる。

そこで**図1**(b)に示す表示を，従来のキャラクタ展開とほぼ同じ速度で実現可能なビットマップ表示技術が開発された。こうした表示技術を用いたハードウェア上に，良好なソフトウェアを開発すれば，従来技術では考えられない新しいマンマシンインタフェースを提供することができる。

図1(b)に示す表示が可能な制御系をユーザーに提供するためには，表示画面上の一画素ずつをアドレスできることが必要で，このためには，次に示す機能が不可欠となる。



(a) 従来技術　　(b) ビットマップ技術

図1　従来技術とビットマップ技術の比較　　従来の表示技術とビットマップによる表示技術の差異を例で示す。ビットマップ技術では，文字を任意の位置に表示できるという特徴をもっている。

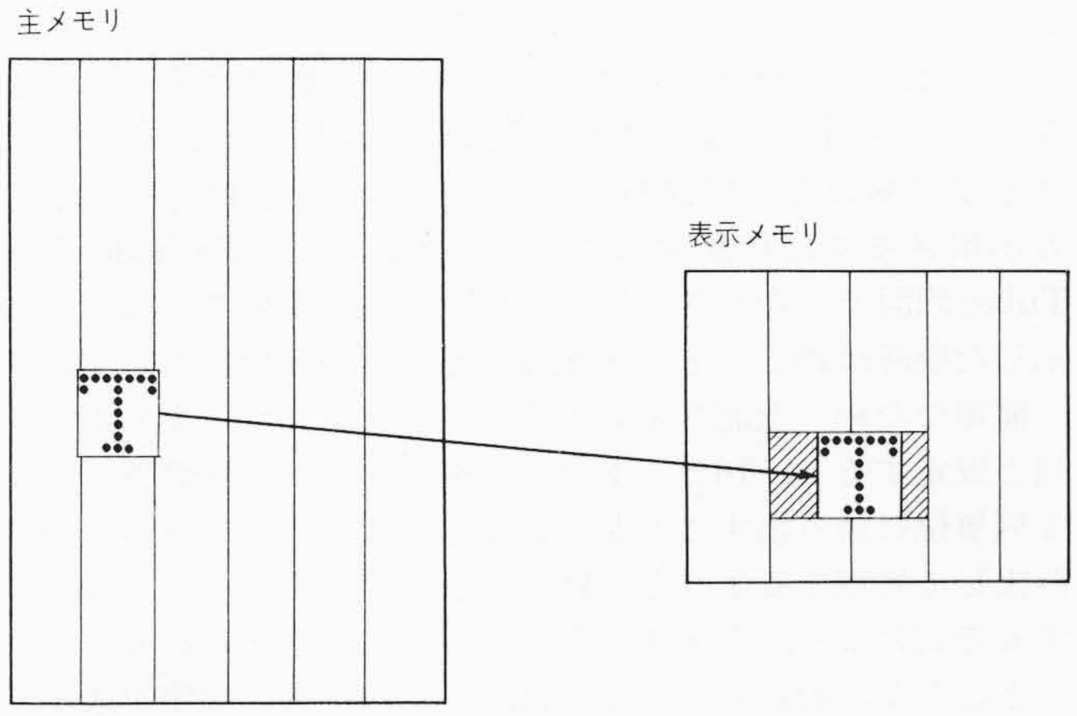


図2　キャラクタ展開の方式　　ビットマップ表示技術でのキャラクタ展開方式を示す。主メモリ上にあるフォントデータ(Tという文字)を表示メモリ上に移動することでキャラクタ展開を行なうが，このとき，メモリのワード境界(図の縦線)を越えて移動できるようにすることで，任意の位置に文字の展開が行なえる。

* 日立製作所日立研究所　** 日立製作所日立研究所 工学博士　*** 日立製作所大みか工場

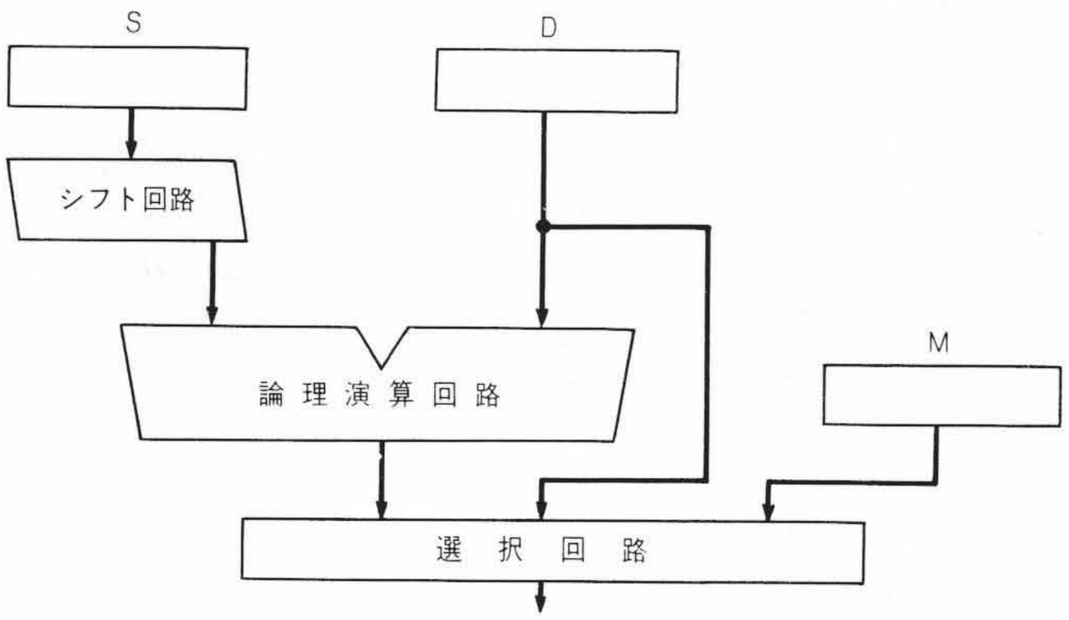


図3 ラスタオペレーションのハードウェア ビットマップ制御に必要なラスタオペレーション用ハードウェアを示す。元のデータSをシフトした後で書き込み先のデータDと論理演算し，更に演算結果を選ぶか，書き込み先のデータDを選ぶかをマスクデータMで選択できるようにすることで，図2のキャラクタ展開を高速に行なえる。

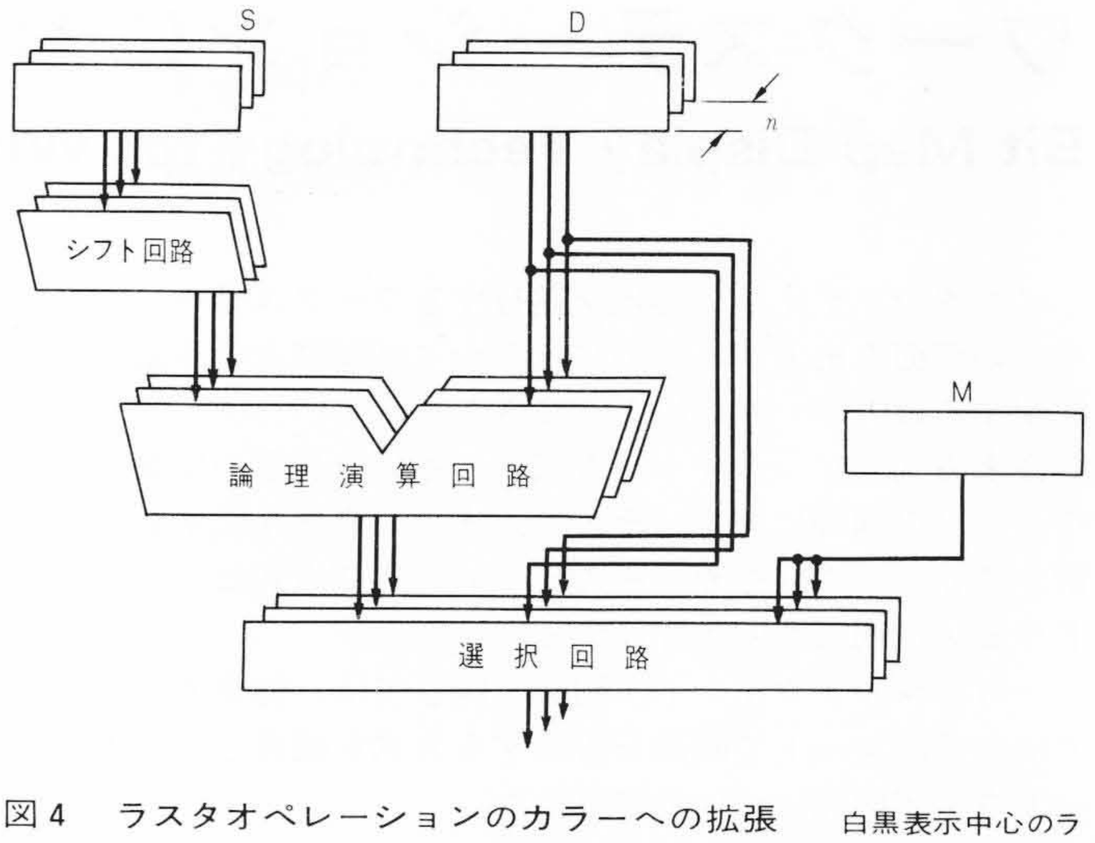


図4 ラスタオペレーションのカラーへの拡張 白黒表示中心のラスタオペレーションのカラーへの拡張用ハードウェアの構成を示す。カラーの色数だけ，図3のハードウェアを保有することで，カラー表示のときも同じ性能で文字展開ができる。

一般に，キャラクタのフォントデータは**図2**の左に示すように，ドットイメージでメモリ上に記憶されている(ここでは文字'T'を例にとって説明している)。これを同図の右に示す表示用メモリに移動することにより，CRT(Cathode Ray Tube)画面上に表示させることができる。同図でメモリ内に示した縦線は16ビットのメモリ読出し幅の境界線を示す。

簡単なため，上記フォントデータを16×16ビットのドット列と仮定すると，16ビットのバス構成をもつ処理装置では，1列単位に読み出すことができる。そこで，1ビットずつ読み出して処理するよりも，16ビットのデータをそのまま移動する方式により，表示速度を上げる方法が考えられる。

ところが，前に述べたように，任意の位置に文字を表示できるようにするためには，読み出した16ビットのデータを，任意のビット数だけシフトして表示用のメモリに書き込む必要がある。更に，書き込み先の16ビットのデータをすべて書き換えてしまうと，**図2**の斜線で示すような展開されない部分まで変更されるため，その部分を書き換えないようマスクする機能も必要となる。

以上をまとめると，**図3**に示すハードウェアがビットマップ制御にとって必要不可欠なものとなる。これは，

(1) 書き込むデータSを任意のビット数シフトする機能

(2) シフトされたデータと，書き込み先のデータDの論理演算を行なう機能

(3) 演算結果を書き込むか，データDを書き込むかをマスクデータMによって選択する機能

の三つによって構成される。一般にこうした処理は，ラスタオペレーションとかBIT BLTという名前で呼ばれており，ビットマップ技術を代表する処理として知られている。

例えば，このハードウェアを用いることで次のような処理を実現できる。

(1) EOR(排他的論理和)で2回演算すると，元のデータに戻るという性質がある。これを利用してカーソルを表示したり，消去したりすることができる。

(2) 文字表示時に，下にある図を消さずに表示する，あるいは消して表示する，という二つのモードを演算の種類を変えることでユーザーに提供することができる。

このように，基本的な機能であるラスタオペレーションを駆使することで，新しいマンマシンインタフェースを実現できる。

3 ビットマップ制御のカラーへの拡張

世の中で知られているビットマップ表示の多くは，白黒を中心としたものである。これは，**図3**で示したラスタオペレーションが各画素単位の移動という一種のイメージデータの処理であり，カラーのようにイメージ情報が増加すると，処理しなければならないデータ量が比例して増大するため，マンマシンインタラクションにとって十分な性能が出せないことに起因する。

ところが，一般のパーソナルコンピュータでは，カラー表示が常識であり，今後，特に色の差異を情報として使用するアプリケーションが増大するにつれ，白黒表示のビットマップ制御でよいという時代ではなくなってきている。

ところで，カラーのデータは1画素当たりnビットで表現されるため，**図3**に示すハードウェアでは，n回ラスタオペレーションを繰り返す必要があり，速度が$\frac{1}{n}$に低下してしまう。そこで，**図4**に示すようなカラー用のハードウェアを開発した。同図に明らかなように，SやDがnビットの場合でも一度に処理可能とするため，シフタ，論理演算回路，セレクタをn層設けることで対応する方式を採用した。各層のハードウェアは1個のLSIに収めることができるため，n個のLSIを用いることで，上記ハードウェアを実現することができる。

図5は，このハードウェアを用いて出力した画面の一例で，文字，図形，イメージが1枚の画面上で合成されていることがよく分かる。

4 マルチウインドウ制御への対応

ビットマップ制御のもう一つの特徴であるマルチウインドウ表示の制御方式について次に説明する。マルチウインドウ表示の制御方法としては，**図6**に示すような3方式で対応することが考えられる。

(a)の方式は，**図6**の例に示すように，全画面データをメインメモリ上にすべてイメージでもつ方式である。マルチウインドウ表示を変更する場合は，イメージデータの必要部分をメインメモリから表示メモリにラスタオペレーションで転送すればよい。

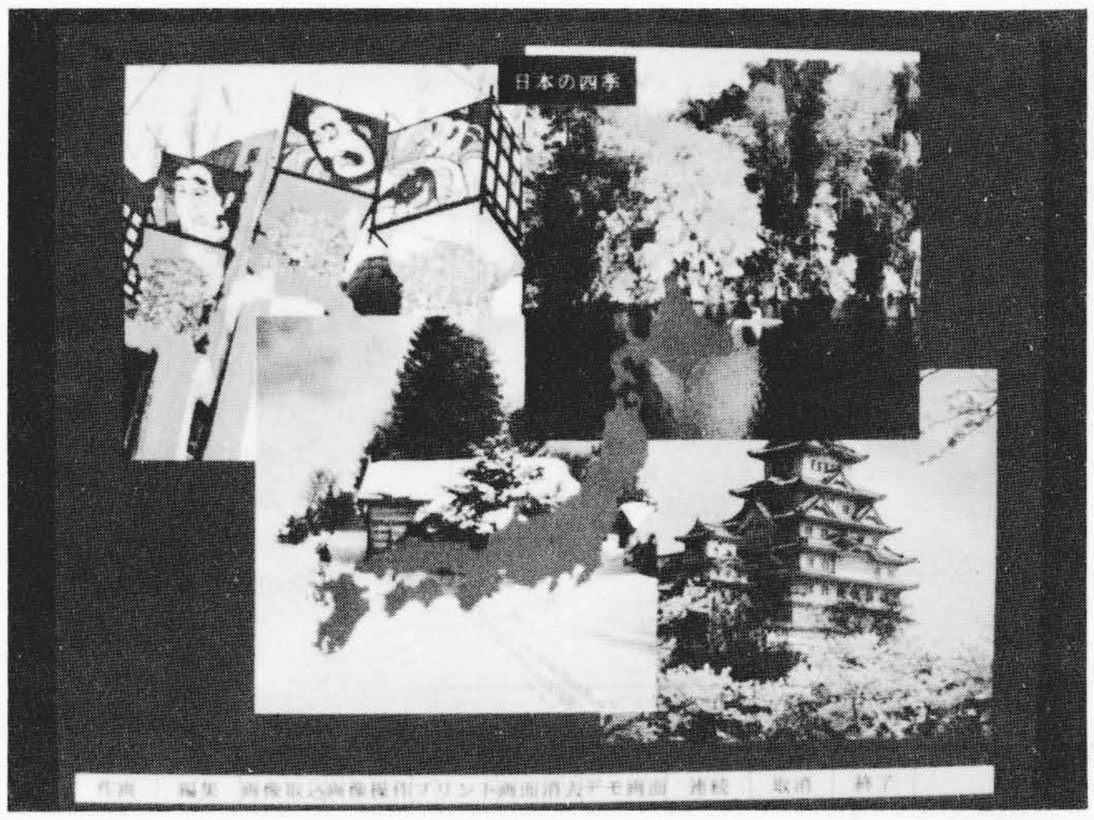


図5　表示画面例　　文字，図形，画像を合わせて一画面上に表示した例を示す。メニューや標題は文字で，日本地図は図形で，4枚の写真は画像で表示されている。

本方式では，メインメモリ上にすべてのイメージデータをもつ必要があり，特にカラー表示の場合は，イメージデータがプレーン数に比例して増大するため問題が大きい。

(b)の隠れている部分だけメインメモリ上にイメージデータでもつ方式は，ウインドウの変更時に，隠れてしまうイメージデータを表示メモリからメインメモリへ転送し，一方，隠れている部分が表示される場合は，これをメインメモリから表示メモリへ転送して制御する方式である。

この場合は，オーバラップ制御が複雑になるという問題がある。

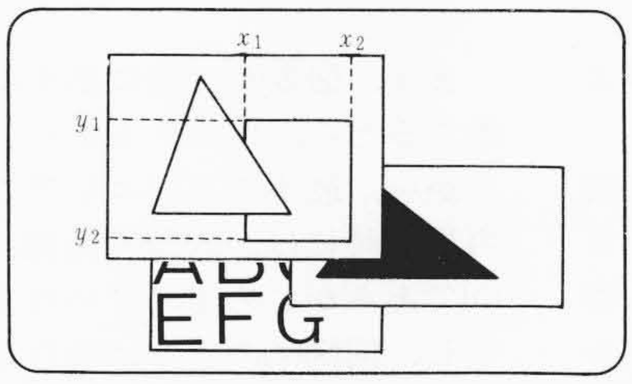
(c)の文字や図形をコードでもつ方式は，**図6**に示すように，画面1の四角形，三角形については頂点座標(xi, yi)で管理し，画面2の文字については文字コードで管理する方式である(画像データはそのままの形で処理される)。この方式はコード情報からイメージ情報への展開，すなわちベクトルの描画や文字のフォント展開の速度を高速にする必要はあるが，画面の管理がやりやすい方法である[3]。

そこで，今回の表示プロセッサでは文字や図形を展開するハードウェアを付加することで高速化に対応し，(c)の方式を採用することにした。次にこの方法による具体的なマルチウインドウの制御例と，本方式を実現する上での高速化ハードウェアについて説明する。

4.1　表示制御例

図7を用いて，マルチウインドウの表示制御の代表例を説明する。メインメモリ上には，三つの画面のコード情報が区分されて置かれ，CRTの画面上には同図(a)に示すような三つのウインドウで表示されているとする。

画面2の優先度を上げ，**図7**(b)の形にする場合は，画面1によって隠れている画面2の部分〔同図(b)の破線内ⓘ〕の再展開を行なえばよい。このためには，展開エリアをⓘとして，



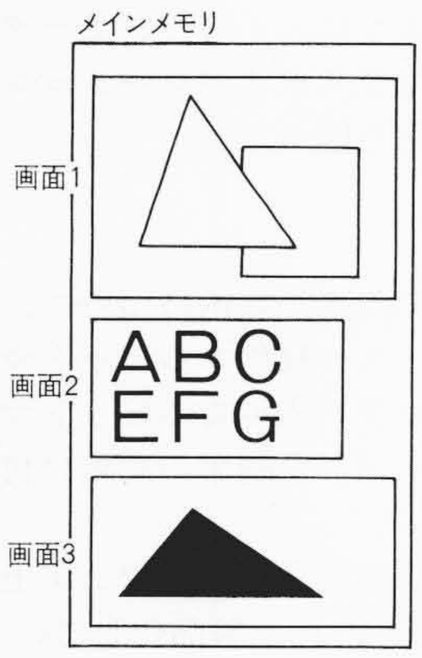
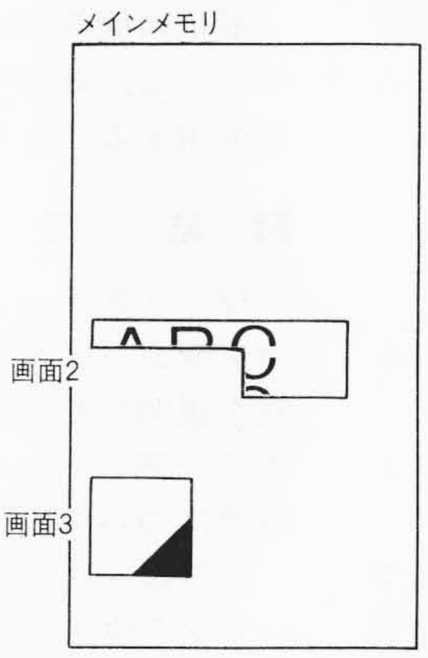
	(a) イメージ管理	(b) 隠れ面イメージ管理	(c) コード管理
メインメモリ内画面情報			
長　所	● マルチウインドウ管理をすべてメインメモリ→表示メモリへのイメージ移動で管理	● マルチウインドウ管理をすべてメインメモリ⇄表示メモリ間のイメージ移動で管理	● CRT表示画面分の表示メモリだけがイメージ情報 ● ウインドウ内情報の変更はコード情報の変更だけ
短　所	● 大容量のイメージメモリを要する。 ● ウインドウ内情報の変更はコード，イメージ情報の両者に関係する。	● 大容量のイメージメモリを要する。 ● 重なり具合の複雑化に伴い，管理複雑化。	● コード情報から表示メモリ内イメージ情報への高速展開を要する。

図6　マルチウインドウ制御方式の比較
マルチウインドウの各種制御方式を比較し，その長所，短所を示す。

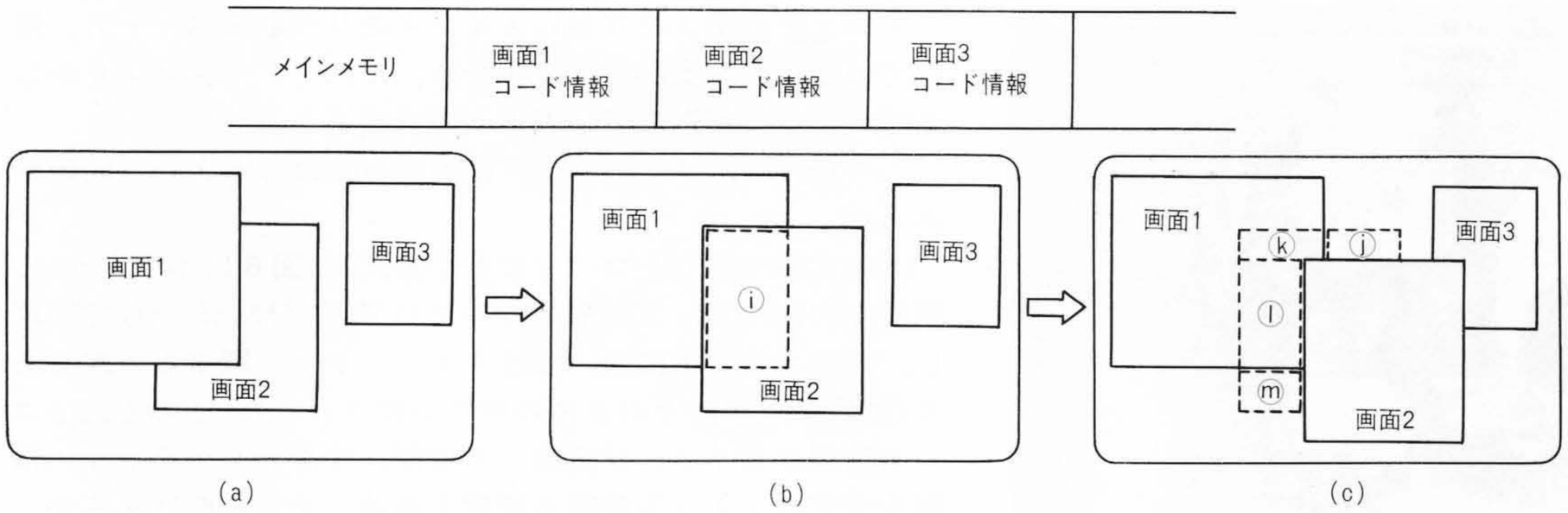


図7 表示制御手順
マルチウインドウの表示制御を，ウインドウの優先度変更(b)，ウインドウの移動(c)について示す。画面の内容を変更する場合は，変更される方形エリア内だけメインメモリ上のコード情報の展開が行なわれる。

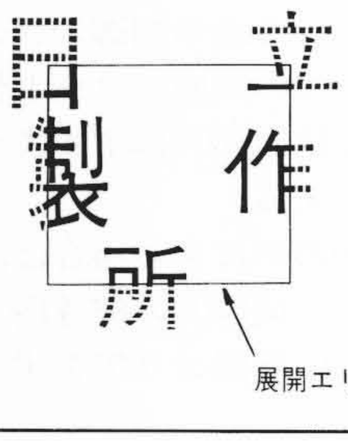
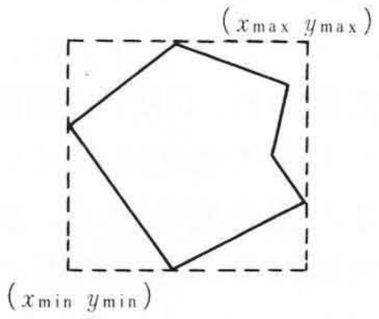
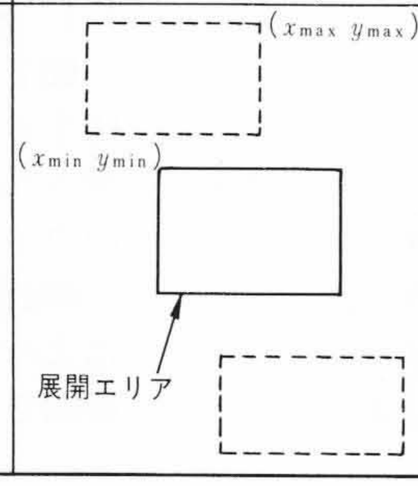
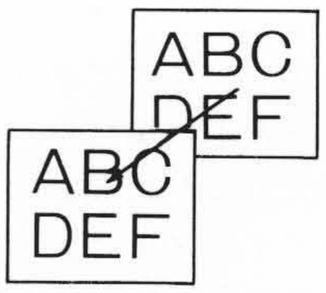
	(a)	(b)	(c)	(d)
項 目	完全な切出し処理	方形エリア展開の高速化		方形エリアの移動
内 容	文字フォントの切出し処理	エリア管理情報の算出	エリア管理情報と展開エリアの重なりチェック	カラーラスタオペレーション
例				

図8 マルチウインドウ制御機構 マルチウインドウ制御のため導入した四つのハードウェアの機能について説明する。

メインメモリの画面2のコード情報の再展開を行なえばよい。これによってCRT表示画面は同図(b)の形に変わる。

次に，画面2を移動する場合について，その処理方法を説明する。まず，移動先に画面2をもっていけばよいから，**3**章で説明したラスタオペレーションの機能を使う。更に，移動元の画面2を消去する必要があるため，残されたエリアを**図7**(c)に破線で示す四つの方形エリアⓙ，ⓚ，ⓛ，ⓜに分離する。そして，エリアⓚ，ⓛについては，メインメモリ上の画面1のコード情報の再展開を行ない，ⓙ，ⓜについては背景色で展開する。この操作によって，同図(c)の画面を得ることができる。

4.2 マルチウインドウ制御機構

4.1の表示例で述べた制御手順を実現するには，

(1) 限られた方形エリアだけを高速に再展開する機能

(2) 限られた方形エリアのイメージデータを高速に移動する機能

の二つが必要となる。このために，今回開発したハードウェアに適用した機能の概要をまとめたものが**図8**である。

まず，**図7**(c)のⓚ，①に示したように，同一の画面を二つの方形エリアに分割して再展開しても，二つの方形のつなぎの部分はつながっている必要がある。このため，**図8**(a)に示すように，展開エリアの境界での切出し処理を正確に行なう必要がある。特に文字展開時の切出し処理については，**図8**(a)の例に示すように，文字が展開エリアの境界とどのように交差しようとも正確に展開できる機能を付加した。

また，前記ⓚ，①の展開のように，メインメモリ上の同じ画面コード情報を何回もチェックして，異なった方形エリアに展開する必要がある。このため，**図8**(b)，(c)に示す機能を付加した。

まず，**図8**(b)の例に示すように，画面のコードデータを作成するとき，作成するコードデータが画面上のどの位置にあるかを，最小の方形エリアで求められるようにした。更に，再展開時には，同図(c)の例に示すように，展開エリアと同図(b)であらかじめ求めていた方形エリアの重なり具合をチェックし，同図(c)のように重ならない場合はその部分のコード情報を読み飛ばすことで，すべてのコードデータをチェックしなくてもよいようにした。

最後に**図8**(d)に示すように，イメージデータの高速移動については，**3**章で述べたラスタオペレーションハードウェアを使用することで高速化を図った。

5 結 言

ワークステーションでの表示技術について，特に，新しいマンマシンインタフェースを提供するビットマップ技術の概要を説明した。また，カラー表示でのビットマップ技術の適用の一例と，マルチウインドウ制御のために付加した機能の概要について説明した。

今後は，ここで開発した表示ハードウェア上に，使いやすいソフトウェアの開発を行ない，製品化していく予定である。

参考文献

1) J. Warnock, et al.：A Device Independent Graphics Imaging Model for Use with Raster Devices, Computer Graphics, **16**, 3, 313～319(1982-7)

2) D. E. Lipkie,：Star Graphics：An Object-Oriented Implementation, Computer Graphics, **16**, 3,115～124(1982-7)

3) R. J. Littlefield：Priority Windows：A Device Independent Vector Oriented Approach, Computer Graphics, **18**, 3, 187～193(1984-7)