NEC μPD7220/72120 Related Public Documents

- ISSCC (International Solid-State Circuit Conference)
- μPD7220A User's Manual
- Nikkei Electronics Magazine (μPD7220)
- Transistor Gijutsu (Technology) Magazine (μPD7220)
- Transistor Gijutsu (Technology) Magazine (μPD7220A)
- Nikkei Electronics Magazine (μPD72120)
- μPD72120 User's Manual

Go to https://www.oguchi-rd.com/LSI%20products.php to get more detailed NEC μPD7220/72120 related information such as;

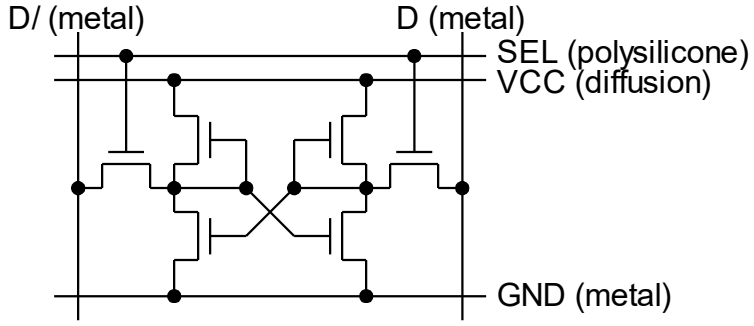
“Logic Schematics”, “Design Notes”, “Evaluation Board Schematics”, “Evaluation Software”, “Silicon Die Photos”, “Newspaper”, “Magazine”, and so forth.

Go to https://www.oguchi-rd.com/patents.php to get patent information including NEC μPD7220/72120 related patents.

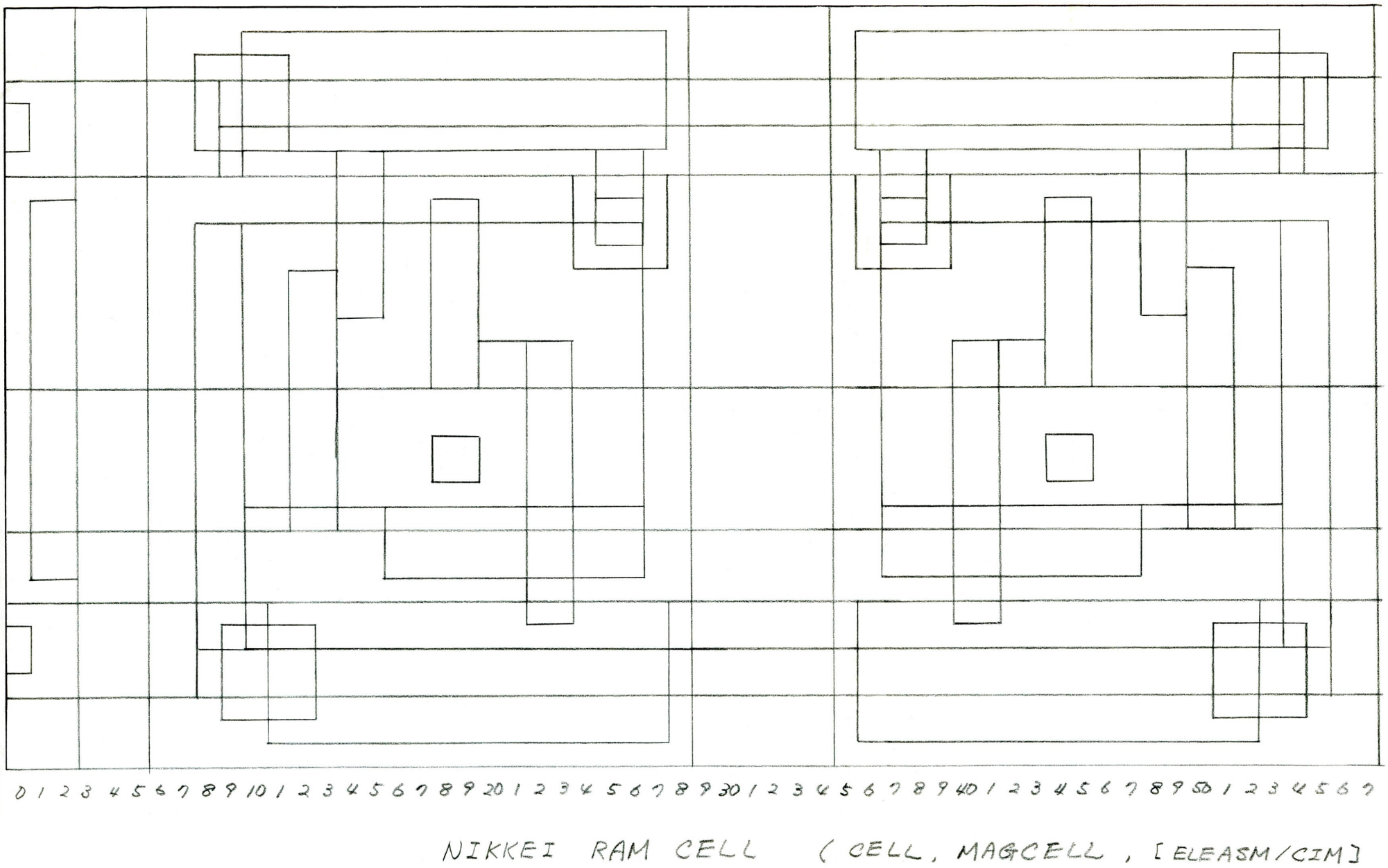
Photo mask layout design example of six transistor SRAM using N channel silicon gate E/D (Enhancement-mode for gate, Depletion-mode for load) MOS transistors for front cover of this issue

I created this graphics image in one day using μPD7220 demo system controlled by Tandy TRS-80 personal computer I privately purchased in May, 1978, brought into laboratory in NEC for decorating front cover sheet of this issue of Nikkei Electronics magazine.

Red (赤)：diffusion layer (拡散層), Green (緑)：poly-silicon layer (ポリシリ層), Blue (青)：Metal (aluminum) layer (アルミ層), Contacts between metal and diffusion layer, Contacts between poly-silicon and diffusion layer



The thesis was related to computer color graphics. Therefore, it achieved distinction of printing by colors for the first time. I specified the colors and created front page image based upon the deputy editor-in-chief Syozo Watanabe's (Nabe-cho) request. Photographer Gengo Aida from Nippon Keizai Shinbun newspaper, came in twice to take a photo of the front cover using 35mm camera once and using 6x9 cm Brownie large camera next day.



Original SRAM cell mask layout I designed before manual digitizing

昭和56年10月12日発行（隔週月曜日発行）第275号　昭和46年5月1日第三種郵便物認可

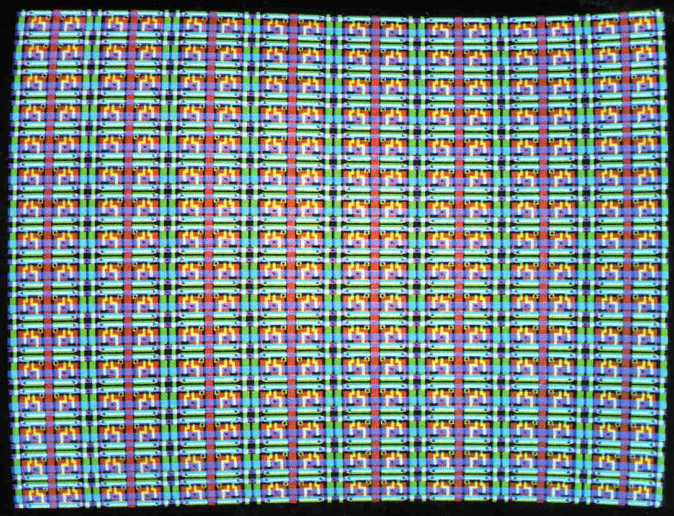
日経エレクトロニクス

NIKKEI ELECTRONICS 1981 10-12

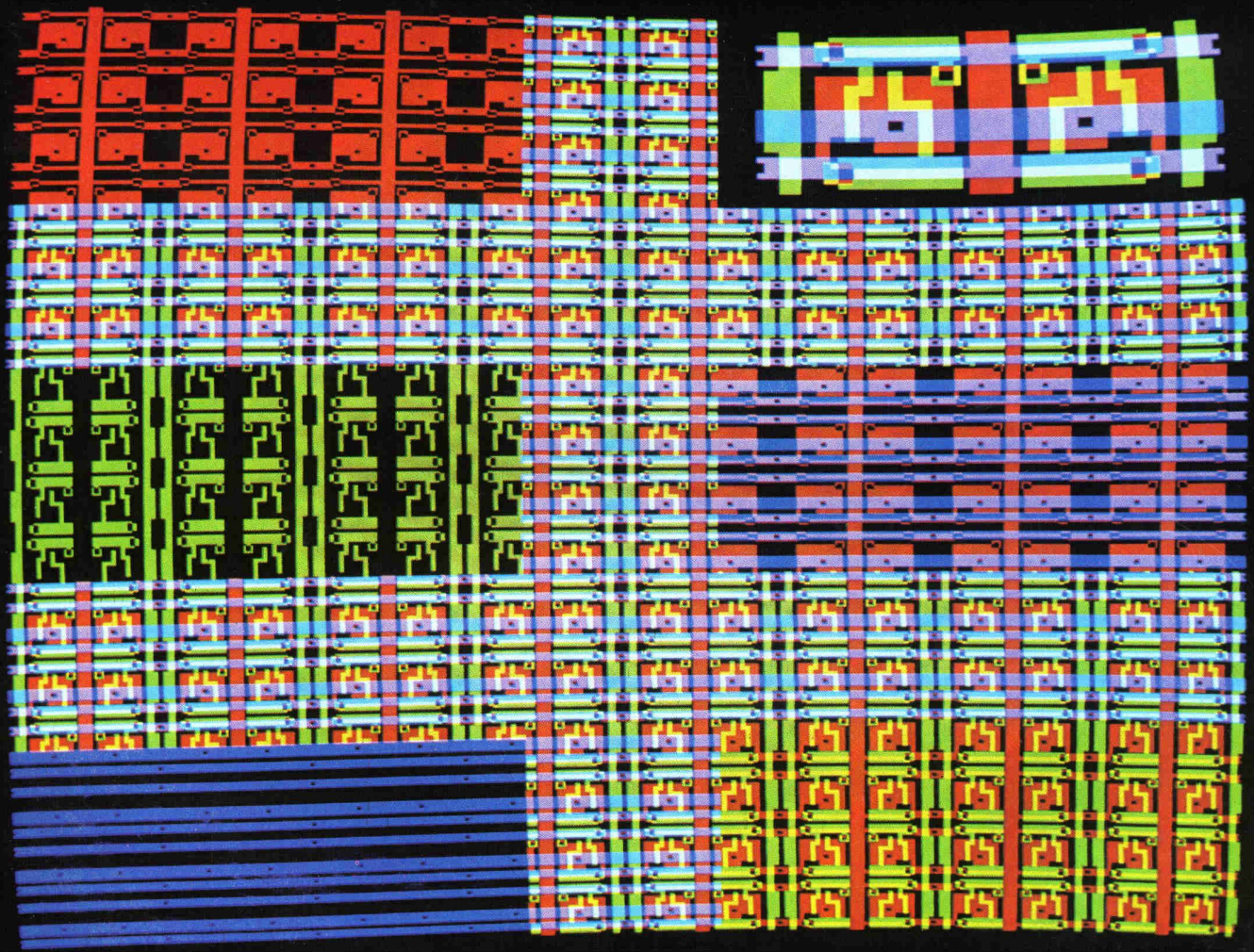
16ビット・マイクロコンピュータ用OS

小型で簡便な各種センサの開発

電圧共振型DC-DCコンバータ



1ドットを800 nsで
描画できるグラフィック・コントローラLSI



日経エレクトロニクス
NIKKEI ELECTRONICS

1ドットを800 nsで描画できるラスタ走査CRT用グラフィック・コントローラLSI

要点　文字表示制御機能と，ドット単位のグラフィック描画機能とを1チップ上に集積した，ラスタ走査CRT用のグラフィック・ディスプレイ・コントローラ(GDC)の概要を紹介している。GDCは1万3000素子を集積したMOS LSIだが，新開発した描画アルゴリズムにより，加減算だけで描画アドレスを算出できる。このため，クロック周波数5 MHzの時，1ドットを800 nsで描画できる。GDCが制御する最大容量256 K語×16ビットの映像メモリーは，システム・バスから分離されているため，装置のCPUがコマンド/パラメータ作成中でも，GDCは並列に描画を実行できる。　(本誌)

小口 哲司
日本電気　集積回路事業部
マイクロコンピュータ・システム部

文字や図形を表示するグラフィック装置には，電子線走査方式と表示管面構造との組み合わせによって，次の3種類のCRTが使用されている。

(1) ランダム走査リフレッシュ型
(2) ランダム走査蓄積管型
(3) ラスタ走査リフレッシュ型

ランダム走査型CRTは，任意の点から任意の方向に任意の点まで走査できるので，開始点，終了点などのベクタ情報を記憶するメモリーのみを持てばよい。一方，ラスタ走査型CRTは，一定の周期と間隔でCRT上を電子線が移動する方式で，テレビジョン放送受信用として一般家庭に普及している。後者は，前者に比べて多色化，多階調化などが容易にできる。反面，1画面分以上の画素を記憶する大容量メモリーが要るため，コスト高になるのが難点だった。しかし，記憶素子の価格が集積度の向上によって大幅に低下してきたため，ラスタ走査型CRTを用いたグラフィック装置は急速に普及しつつある。

ラスタ走査型CRT用の，文字表示制御だけを行うCRTコントローラ(CRTC)は，既に多数発表されている。しかし，最近では，文字単位の表示制御だけではなく，1ドット単位で，細かく表示/描画するような制御機能が，オフィス・コンピュータ用のCRT端末や通信端末装置にも求められつつある。更に，文字表示についても，漢字表示やワード・プロセサ端末装置として，従来型CRTCにはない，より高度な機能，設計自由度の高さが求められている。

これらの要求に応えるため，高度な文字表示制御機能，および1ドット単位で高速にグラフィック描画する機能を，1チップに集積したグラフィック・ディスプレイ・コントローラLSI(GDC: graphic display controller。以下，このLSIを単にGDCと呼ぶ)を開発した。このGDCは，従来の高度なグラフィック装置に採用されているような，ビット・スライスCPUやTTL(transistor transistor logic)，PLA(programmable logic array)などで構成されるグラフィック表示/描画回路を，そのまま集積回路化したのではない。バイポーラの乗算器などを使わなくても高速動作できるような描画アルゴリズムを新たに開発し，MOS LSI化したものである。LSI上には，グラフィック機能のほか，文字表示制御機能も集積してある。このため，簡単な文字表示制御から高級なカラー・グラフィック表示/描画制御まで，広範囲に使える。更に，表示機能を使用せずに表示とは無関係な大容量メモリーの制御用としても利用できる(ここで，「表示」とはGDCが映像メモリーの内容を読み出す操作，「描画」とは映像メモリーに書き込む(書き換える)操作を言う。以下も同じ)。

どのような描画方向に対する直線描画であろうと，円弧描画であろうと，GDCの描画速度はドットの修正時間を含み，一律に800 ns/ドット（クロック周波数5 MHz時）である。これは，既存のグラフィック装置の描画速度よりもかなり速い。例えば，描画用プロセサとしてバイポーラ・ビット・スライスCPU 4個構成の16ビットCPUとワード長56ビットのROM（read only memory），RAM（random access memory）とで構成される高級なグラフィック装置が発表されている。[1] その装置の描画速度は1.12 μs/ドットであり，GDCが達成した描画速度には及ばない。更に，従来装置はソフトウェアで描画処理を行うので，描画種類（円，直線など）や描画方向が異なると，描画速度が大きく変動してしまう。

映像メモリーをシステム・バスから切り離す

GDCは，最大256 K語×16ビット（1024×1024ドットのメモリー・プレーンであれば4枚分）の大容量映像メモリーを自分自身で直接制御する。この映像メモリーは，装置のCPUが制御するシステム・バスから分離されている（図1）。つまり，映像メモリーに対して読み出し/修正変更/書き込みを行うのは，CPUではなく，GDC自身である。このように，映像メモリーをシステムの主記憶から分離した構造にできるため，従来型CRTC（別掲「従来のCRT制御用LSI」，pp. 190-191参照）のように，システム・バスの衝突に対して注意を払う必要はない。GDCがシステム・バスの使用権を要求するのは，DMA（direct memory access）転送要求時だけである。そのため，映像メモリーに対してGDCが描画処理を実行中であっても，CPUは別の処理，例えば次の描画のためのコマンド作成に専念できる。

一般に，座標入力から表示までのグラフィック装置の動作は，次のように区分できる。

① ディジタイザやあらかじめ作成されているリストなどから座標データを取り出す（**座標入力**）。この時CPU自身が座標を発生することもある。

② 座標データを基にして変換処理などを行い，よりマイクロなコマンド/パラメータを作成する（**描画前処理**）。

③ コマンド/パラメータを基にして描画アドレス計算を行う（**描画処理**）。

④ 映像メモリーにアドレスを供給し，データの修正・変更動作を行う（**描画実行**）。

⑤ 常時，表示アドレス演算をし，映像メモリーへアドレスを供給し表示データを読み出す（**表示**）。

⑥ 常時，同期信号を発生しモニタ・テレビに供給すると共に表示/描画のタイミング制御をする（**同期信号発生**）。

今回製品化したGDCは，このうち，③，④，⑤，⑥の処理，制御をCPUの関与なしに行う。従来の装置では，③，④をCPUが処理しているため，描画速度が遅い。

LSIの内部構成

図2(a)は，GDCのチップ写真，(b)はそのブロック図である。図2(b)に示すように，GDCは入出力FIFO（first in first out），コマンド制御ROM，データRAM，同期信号発生部，表示アドレス制御部，描画アドレス制御部，映像メモリー・データ制御部，それにCPUインタフェース/DMA制御部で構成される。各部は，同図のように9ビットと8ビット幅の2種類の内部データ・バスで結ばれている。表示アドレス制御部や描画アドレ

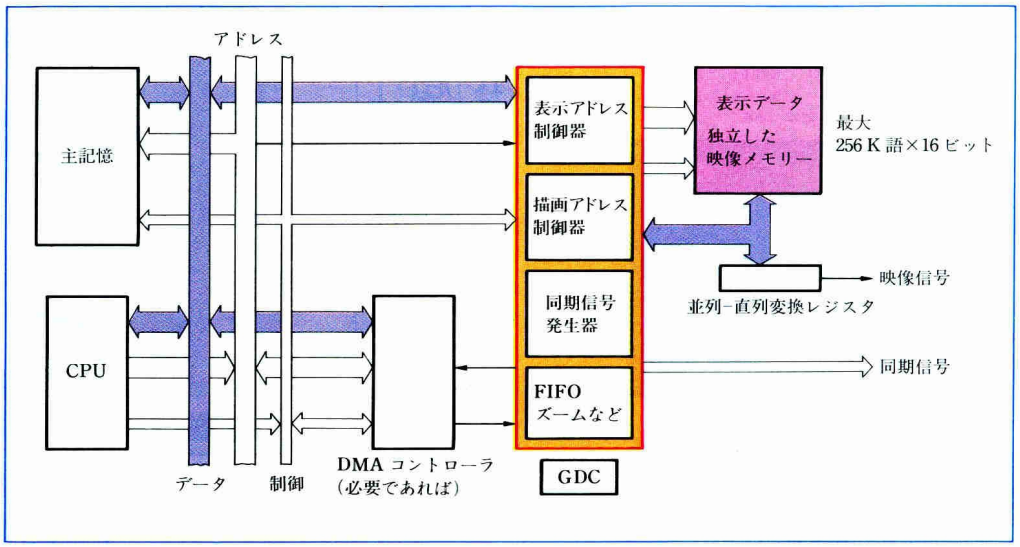


図1 GDCのシステム・ブロック図。GDCおよび映像メモリーはブラック・ボックス化され，独立して動作するI/O機器としてとらえることができ，拡張性に富む

ス制御部などは，互いに関連を保ちながらも独立に並行動作する必要があるため，それぞれ固有のカウンタ，演算器，シフタを持たせている。以下，各部の構成と機能の概略を紹介しておく。

▶入出力 FIFO：16語×9ビットの容量を持ち，CPUから送られてくるコマンド/パラメータを一時記憶する。逆にCPUが映像メモリーやGDCの内部レジスタの内容を読み出す場合も，それらのデータはこのFIFOを経由して読み出される。つまり，DMA転送時を除けば，CPUとGDC（および映像メモリー）間のデータ転送は，すべてこの双方向FIFOを介して行われる。GDCが描画を実行中でも，このFIFOがいっぱいでなければ，CPUはGDCにコマンド/パラメータを送れる。なお，FIFOの9ビット目には，データ・バス上の8ビット・コードがコマンドかパラメータかを示すビットが記憶される。

▶コマンド制御 ROM：128語×14ビット構成で，コマンドの解釈や，表示領域切り替え時に表示アドレス制御部から発生される内部割り込みの処理，データRAMの制御などを行う。

▶描画アドレス制御部：18ビットの描画アドレス・レジスタEADや，グラフィック文字描画時に1～16倍の係数を設定する

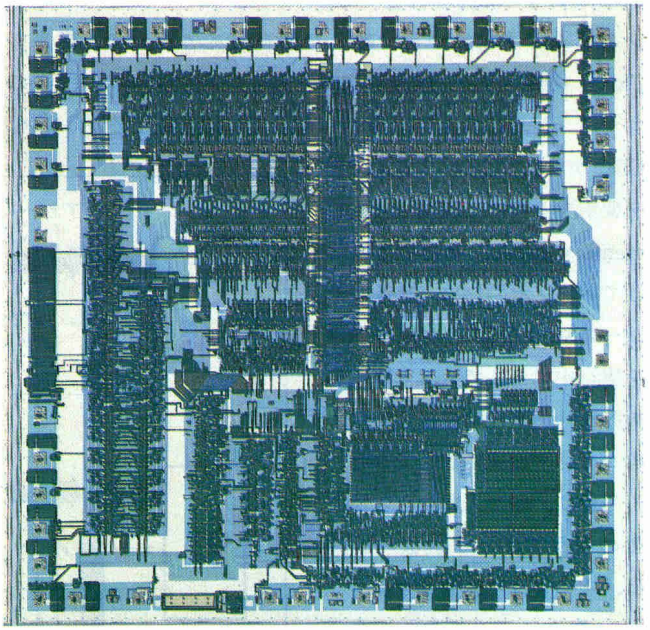


図2(a)　GDCチップ。ゲート長3 μmのnチャネル・シリコン・ゲートE/D MOSトランジスタ1万3000個を搭載する。チップ・サイズは5.29×5.39 mm。高速並列演算を実行するためハード・ワイアド論理を用いる。規則性のある繰り返し回路にまとめ，マスク・レイアウトを容易にしている

ための拡大描画用レジスタ ZW，更に描画パラメータを記憶する 5 本の 14 ビット・レジスタ DC，D，D_1，D_2，DM を持つ。別掲記事「描画アルゴリズムの改善」, pp. 202-205 で述べるような描画アルゴリズムに従って，所定の演算を行い，描画（ワード）アドレス EAD およびドット・アドレス dAD を生成する。整数加減算回路を 3 組（AU_2～AU_4）持ち，前記の描画アルゴリズムに基づく演算と EAD，dAD の生成を 4 クロックで並行に行う。

▶表示アドレス制御部：18 ビットの表示アドレス・レジスタ DAD，8 ビットのリフレッシュ・アドレス・レジスタ REF を持ち，表示アドレス，リフレッシュ・アドレスを生成する。また，1～16 倍の拡大表示係数を設定するためのレジスタ ZR を持つ。

▶同期信号発生部：各種のパラメータ・レジスタ（図 2(b)の同ブロックに示す）の設定値に基づいて，水平・垂直同期信号や表示消去信号などを発生する。

▶映像メモリー・データ制御部：グラフィック描画に必要なドット単位の修正を行う。つまり，映像メモリーから読み出したデータと，直線，破線などの線種を記憶する 16 ビットのレジスタ PTN，および dAD の内容を比較し，後述するような方法で映像メモリーの内容をビット単位で修正する。

▶データ RAM：16 語×8 ビット構成で，表示開始アドレス SAD や表示領域設定ライン数 SL，描画時に使用される 16 ビットの線種データ，8 バイト構成のグラフィックス文字描画用ドット情報を一時的に記憶する（図 3）。これらの値の設定は，CPU

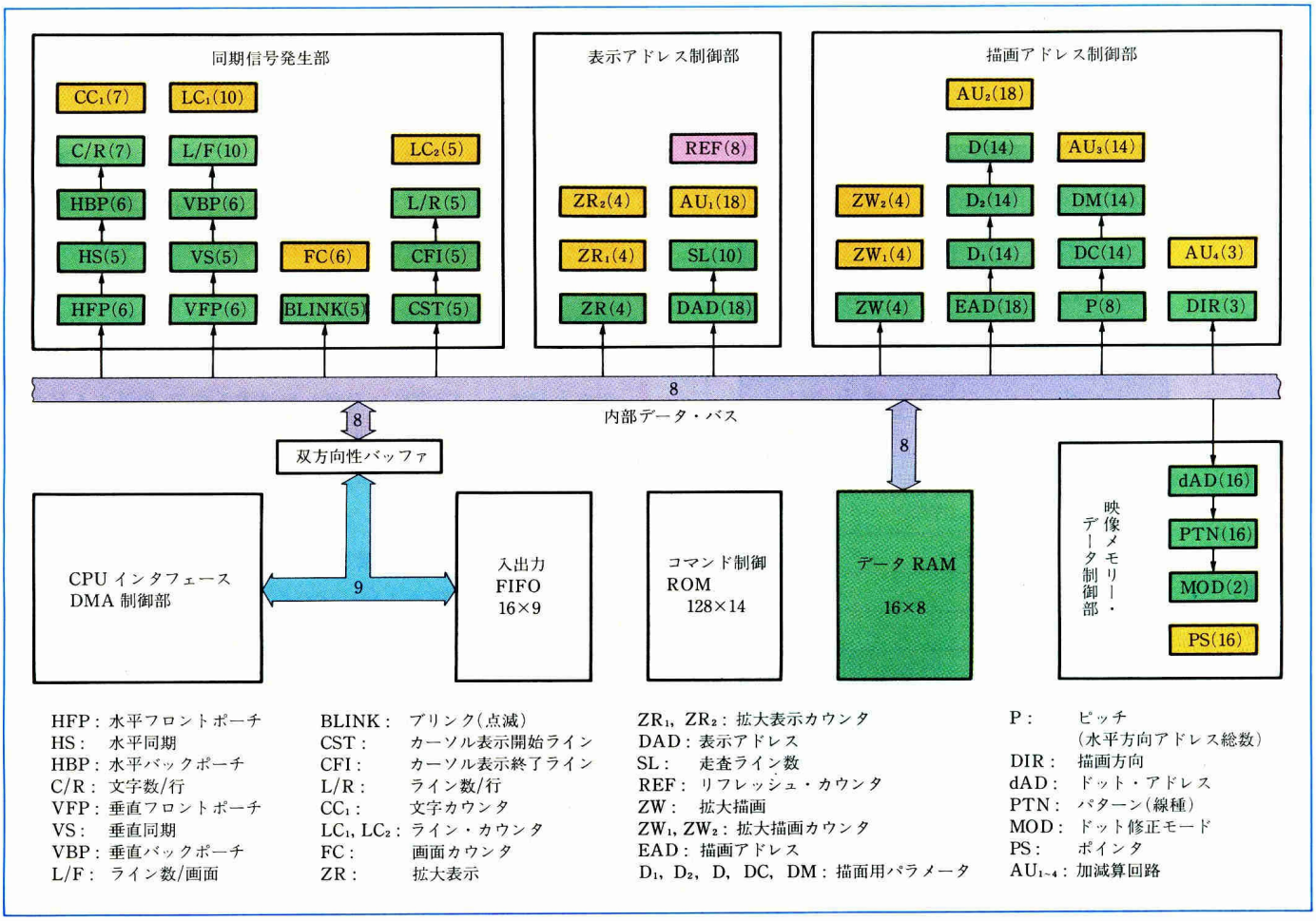


図 2(b) GDC のレジスタ・レベル・ブロック図。従来型 CRTC は同期信号発生部と表示アドレス制御部の一部しか集積していない。■部レジスタはプログラムで任意値に設定できる。各レジスタ群は▭で示す専用のカウンタや演算器を持ち，内部データ・バスとは別経路で並列動作する

がSCROLLコマンドによって行う。実際の表示/描画処理時には，表示アドレス制御部などの独立した閉ループ回路内で処理するため，これらのデータは前もって図2(b)中のDAD，SL，PTNなどのレジスタへ転送される。

▶CPUインタフェース・DMA制御部：一般の8ビットまたは16ビット・プロセサと接続するためのインタフェース部。8ビットの双方向性データ・バス（$DB_{0\sim7}$），アドレス・ライン（A_0），書き込み信号（$\overline{WR}$），読み出し信号（$\overline{RD}$）などがシステム・バスから供給される（図4）。また，DMA制御部は，映像メモリーと主記憶間のDMA転送を制御する。映像メモリーのアドレス不連続部を含む矩形領域データについて，主記憶との間で双方向のメモリー-メモリー間DMA転送を可能にする。

映像メモリーに対する描画を4クロックで行う

GDCの内部回路は，外付け回路から供給する単相クロックと同一周波数の2相クロックを内部で発生し，そのクロックに同期して動作する。クロック周波数は，ラスタの1水平走査内表示区間の表示時間と，その時の映像メモリーへのアクセス数（表示ドット数に依存する）によって決定される。図4に示したように，GDCは映像メモリーに対して1回のアクセス（2クロック）で16ビット並列に読み出す。したがって，例えば，水平走

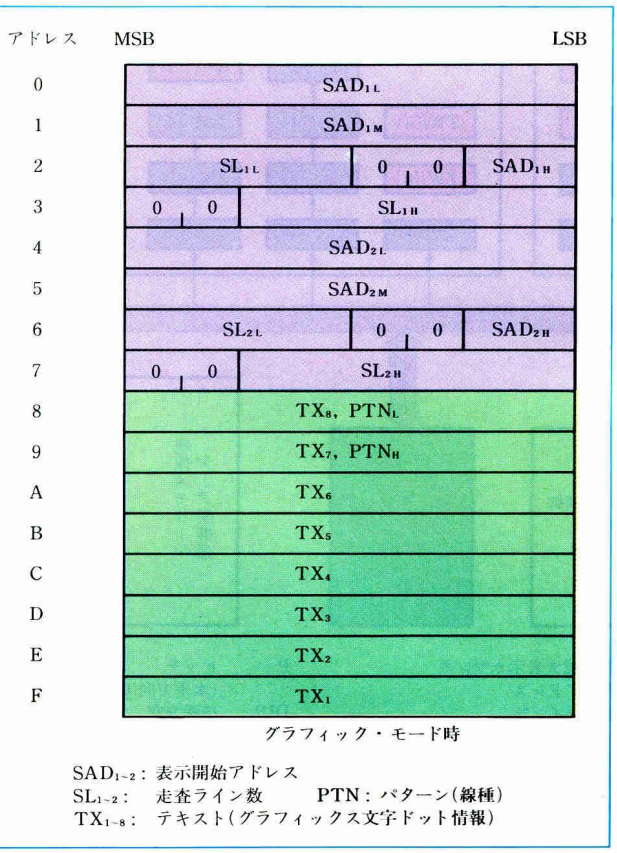


図3 内蔵データRAMマップ。コマンドによって任意なアドレスからデータを設定できる。下位8バイトは表示用，上位8バイトは描画時，線種情報記憶用または文字ドット情報記憶用として使用する

従来のCRT制御用LSI

従来から使用されているCRT制御用LSI（CRTC）は，文字表示制御を主目的として設計されている。これらは，大別して，①DMA転送・行バッファ型と，②ビデオRAM型とに分類できる。両型式共に，表示データを格納するメモリーはCPUが直接制御する主記憶の一部にマッピングされる（図A）。

DMA転送・行バッファ型CRTCは，最大80バイト程度の文字コードと最大20バイト程度の文字属性制御コードを蓄積する行バッファを2系統持つ。各々，現在行表示用とDMA転送される次行表示データ格納用として使用され，行表示の終了ごとに，その用途が反転する。短所は，行バッファ容量の上限がCRTCによって決まってしまい，拡張できないことである。このため，文字属性制御や1行中に表示できる文字数などが制限されてしまい，装置設計上の自由度に欠ける。しかし，表示内容を主記憶からCRTCにDMA転送するので，後述するビデオRAM型CRTCの場合とは異なり，バスの衝突や表示中のフラッシュの発生は起こらない。ただし，表示内容に変化がなくても1行の表示終了ごとに，必ずDMA転送を行う必要がある。1行分のバッファしか内蔵していないからである。DMA転送のたびにCPUを停止させねばならず，キーボード制御や通信制御などのCPU本来の処理能力が低下する。1行に80文字を表示し，文字属性を付加した場合，DMA転送によってCPUが停止する比率は30～40%にも達する。グラフィック表示制御用として，このDMA転送方式は適さない。文字表示制御時には，1行（例えば16ライン）走査期間内に最大100バイト程度のDMA転送を行えばよいが，グラフィック表示制御時には，1ライン走査期間内に1ライン分の表示データをすべて転送せねばならない。例えば，1水平走査時間が64 μsのCRTを使用し，1ラインに512ドット（64バイト）を表示する場合，1Mバイ

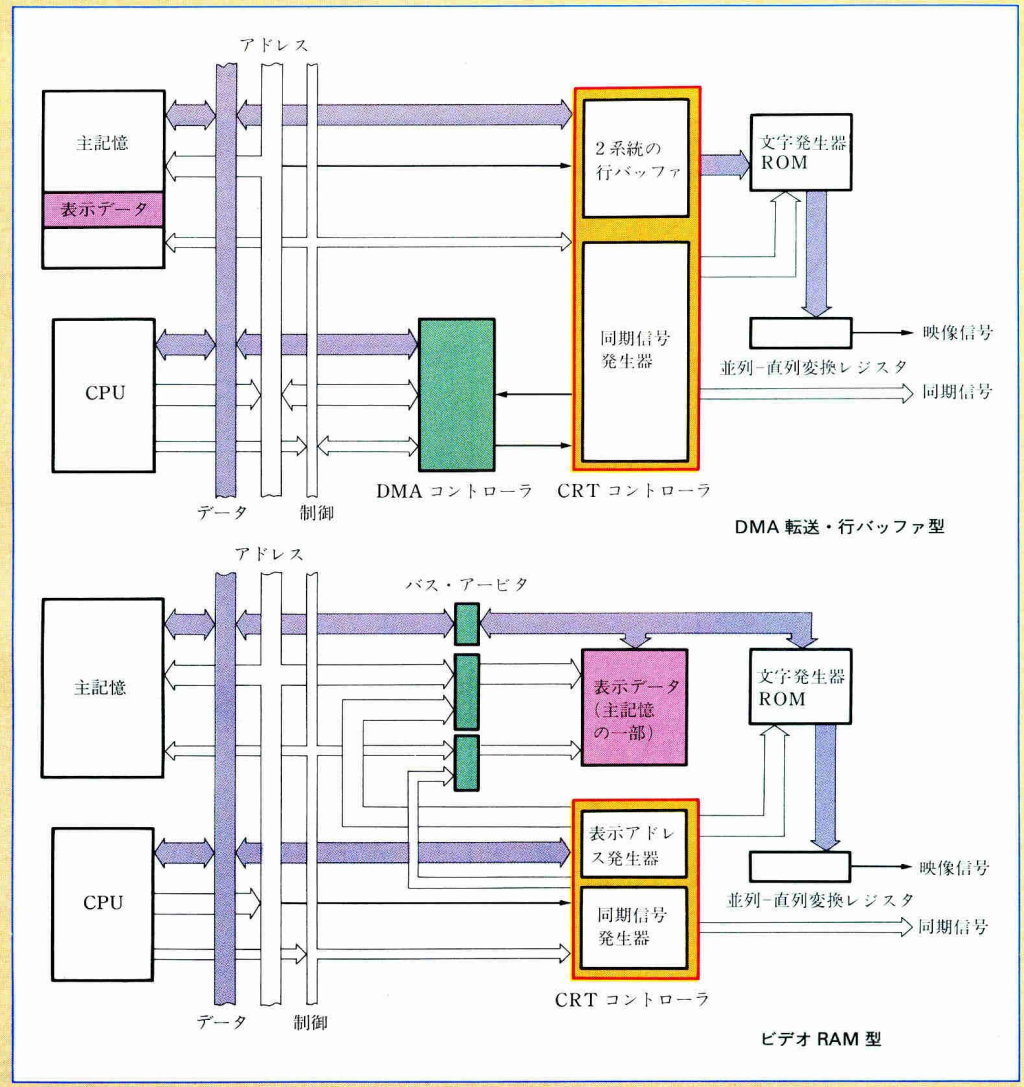
査周波数が31.5 kHzのCRTに1024×1024ドットのインタレース表示を行う時，クロック周波数は約5 MHzになる。

映像メモリーのアドレス・サイクルは，表示時およびリフレッシュ時には2クロック，描画時にはリード/モディファイ/ライト(R/M/W)動作を行うために4クロックに固定されている(図5)。映像メモリーの内容をGDCを経由してCPU側へ読み出す場合も，GDCは描画時と同様，映像メモリーに対してR/M/W動作を行うため，4クロックかかる。

描画時のR/M/W動作は次のような段階を踏む。まず，描画アドレス(EAD=18ビット)が16ビットのアドレス/データ多重化出力端子$AD_{0\sim15}$およびアドレス出力専用端子A_{16}，A_{17}から出力される。サイクルの最初の1クロック(E_1)で出力され

ト/秒の高速で連続したDMA転送を行う必要がある。この時，表示期間中，CPUは完全に停止してしまう。より高解像度のCRTを使用した時には，DMA転送速度を更に高速にせねばならず，実際的ではない。表示速度はDMA転送速度よりも高速だからである。

ビデオRAM型CRTCは，表示データが格納されているメモリーのアドレスを表示期間中に供給する。メモリー内容を変更する時には，CPUが供給するアドレスとデータがメモリーに接続される。このようにアドレスおよびデータ・バスをCPUと表示系とで共有しているので，バスの衝突を防止するための処置が必要となる。装置設計時のソフトウェアおよびハードウェア設計上の負担が大きい。また，表示にフラッシュを生じさせずにメモリー内容の変更を行う場合には，特に，バス・アービトレーションに費やす時間的損失が大きく，データ転送機能が低下してしまう。表示メモリーは主記憶上にマップされるが，ハードウェア構成上は主記憶とは独立した小規模なメモリー構成としなければならない(図A参照)ことも，メモリー素子が安価で高集積化されつつある現在，装置設計者を悩ませる一因ともなっている。



図A 従来のCRT制御用LSIのシステム・ブロック図。両方式共に，表示データは主記憶中にマップされる。システム構成上の自由度や拡張性に乏しい。本文で紹介するGDCの場合は■部回路が不要となる

たアドレスはトライステート出力状態を持つオクタル・ラッチ回路に $\overline{RAS}$ (row address strobe) 出力が高レベルの時にラッチされる(図4参照)。この $\overline{RAS}$ 信号は，一般的な16ピン・ダイナミックRAMに対する行・列アドレス切り替え信号の基準タイミング信号としても用いられ，外付け遅延回路によって発生タイミングがダイナミックRAMにとって最適となるように調整した後，ダイナミックRAMの $\overline{RAS}$，$\overline{CAS}$ (column address strobe) 端子に接続する。更に，アドレス・ラッチの出力を切り替え，GDCが供給する16ビットのアドレス信号を8ビットに時分割された行・列アドレスとしてダイナミックRAMに接続する。E_2 から E_3 にかける1クロック間に，$\overline{DBIN}$ (data bus in) 信号を出力し，映像メモリーから読み出された16ビット・データをバス・バッファを介して映像メモリー・バスに載せ，一括してGDC内に読み込む (read)。E_3 の1クロックで，ドット修正モード定義情報(図2中のMOD)に従って，読み込まれたデータに修正を加える(modify)。E_4 においてGDCから修正後データを出力し，$\overline{DBIN}$ 信号を外付け回路によって遅延させて生成するダイナミックRAM書き込み信号 ($\overline{WE}$) の立ち

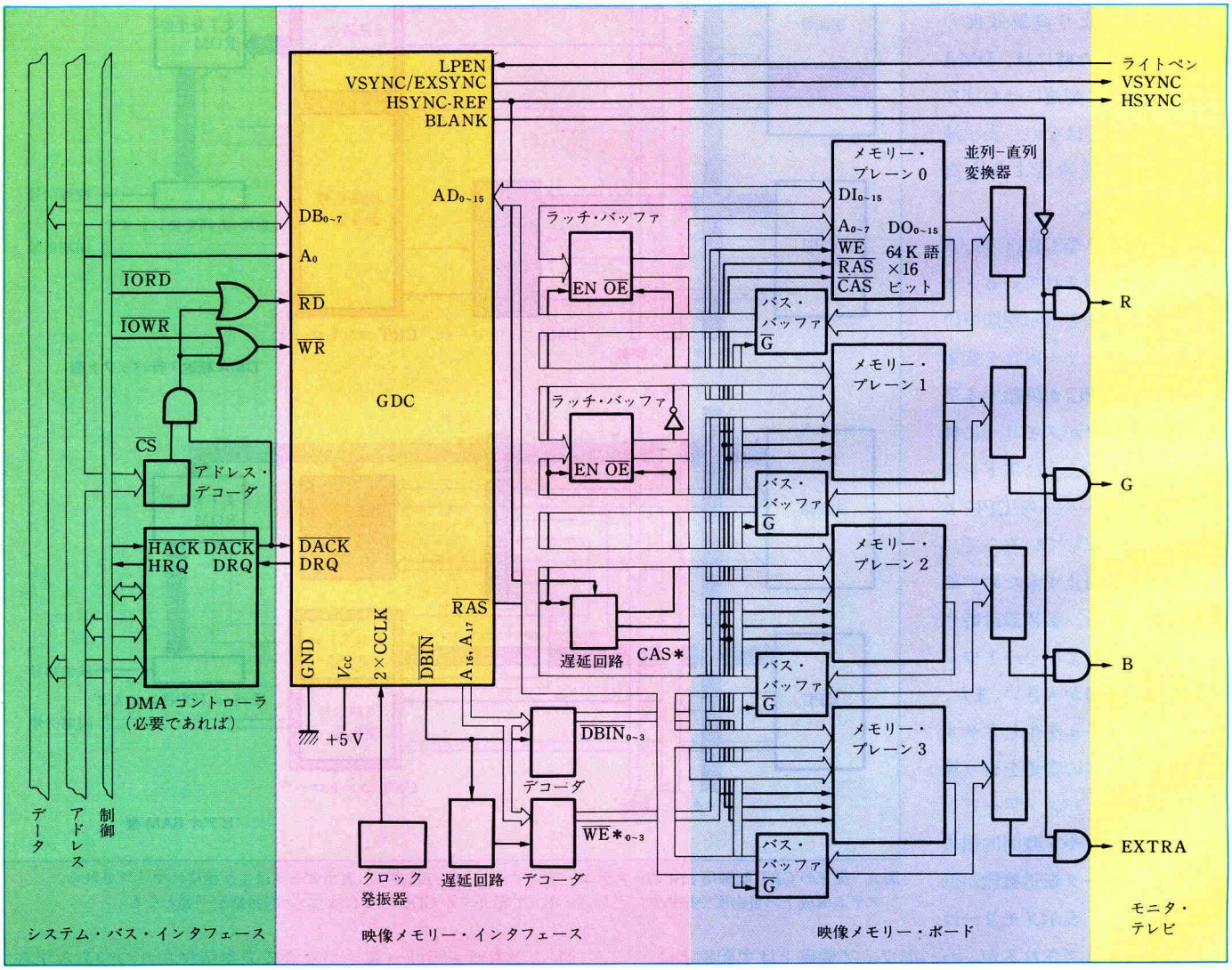


図4　インタフェース・ブロック図。1024×1024ドット16色グラフィック表示を簡単なインタフェースで実現できる。64Kダイナミック RAM 64個で構成する映像メモリー部を含め1枚のプリント基板上に実装できる

下がり時に映像メモリーに書き込む(write)。アドレスは，$\overline{RAS}$ および $\overline{CAS}$ の立ち下がり時にダイナミック RAM に取り込まれる。

1ドット単位の描画/修正が可能

映像メモリー・データの修正は，映像メモリー・データ制御部において図6のようにして行っている。16ビットのポインタ(PS)は16ビットのパターン情報(PTN)のうち1ビットのみを指し示し，初期化動作時や直線，円弧などの描画終了時には最下位ビットのみが“1”となる。1ドットの描画後，上位方向に1ビットだけシフトし，PTN に対する指示位置を更新する。PS によって指示された PTN の1ビットのみを EXOR (exclusive OR)，AND, OR などのドット修正ゲートの一方の入力として使用する。一方，16ビットのドット・アドレス(dAD)は16ビットの修正前データから修正を加えるべきビットを抽出し，修正ゲートに接続する。dAD が“0”であるビットに対応する修正前データはドット修正の対象とはならず，単に，GDC 内を循環する。ドット修正モードは REPLACE, COMPLEMENT, CLEAR, SET の4種あり，2ビットの MOD レジスタに修正の種別を格納しておく。図6に示すように REPLACE 時には修正前データは無視される。

REPLACE 以外の修正モードでは，パターン参照内容(PTN レジスタ中の PS で指される内容)が“1”であるドットについてのみ修正を加え，“0”の場合には，単にメモリー内容を循環して再書き込みをする(図7参照)。この修正をドット単位の論理演算としてとらえるならば，パターンが“1”であるドットについて，各々，排他的論理和(EXOR)，否定論理積(NAND)，論理和(OR)として動作する。異なる修正モードを組み合わせて使用したり，パターン内容を反転して設定すると，多彩な修正結果を得ることができる。

描画サイクル中であっても，表示アドレス制御部において表示およびリフレッシュのアドレス計算を行っている。表示サイクルには，表示アドレスのみが映像メモリーに供給される。図4の例では，1枚当たり16個の64Kビット・ダイナミック RAM で構成されるメモリー・プレーン4枚に搭載されるすべてのダ

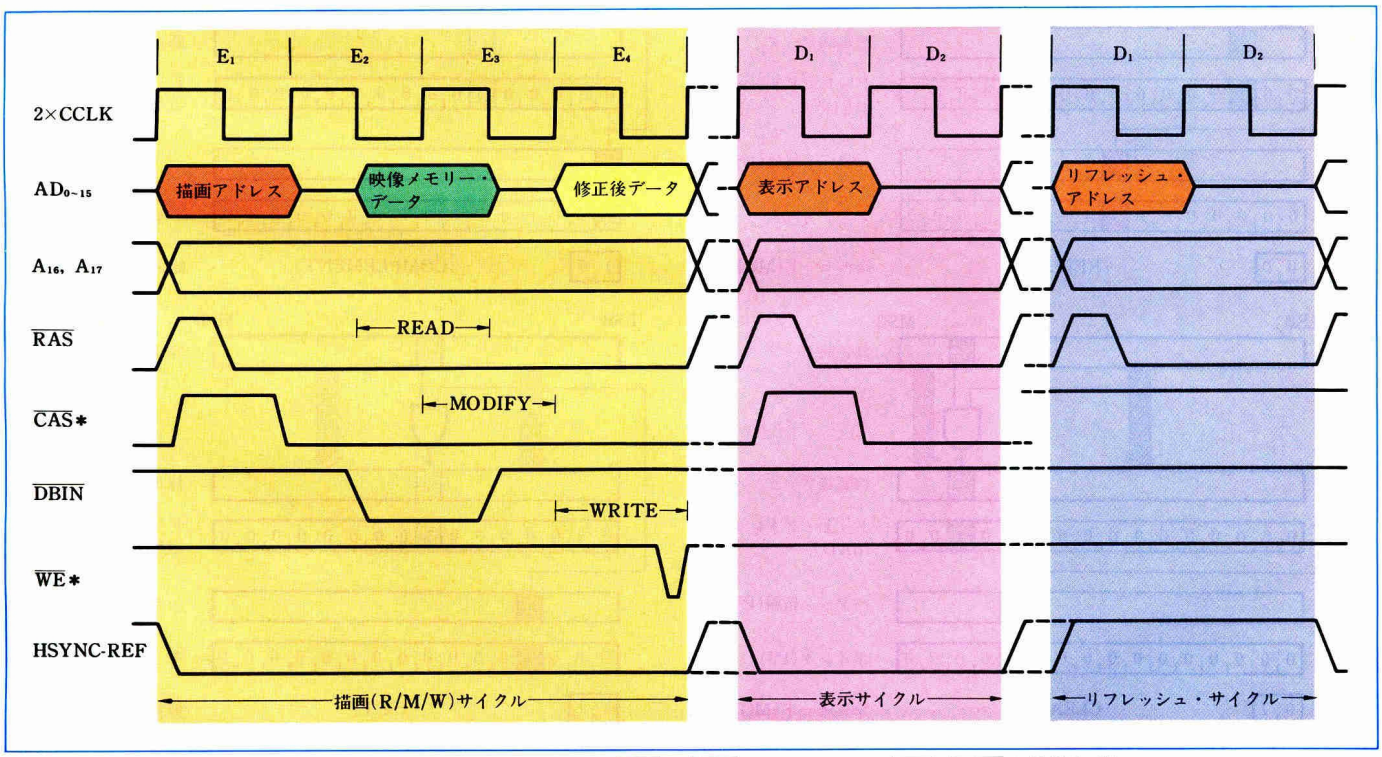


図5 映像メモリー制御タイミング。3種のアドレス・サイクルを持つ。$\overline{CAS}$*，$\overline{RAS}$* は各々基準信号 $\overline{RAS}$，$\overline{DBIN}$ を外付け回路によって遅延し，最適なタイミングに発生させる。リフレッシュは HSYNC 時に行う

イナミックRAMに対して同一の$\overline{RAS}$, $\overline{CAS}$信号が接続される。したがって，各メモリー・プレーンのダイナミックRAM出力DOから同時に16ビットずつの表示データが並列-直列変換レジスタに送られ，映像直列信号を得る。描画サイクル時には$\overline{DBIN}$と$\overline{WE}$信号を，この例ではメモリー・バンク切り替え用信号として使用されるアドレス信号A_{16}, A_{17}によってデコードし，メモリー・プレーン選択を行うが，表示時には各メモリー・プレーンから同時に表示データを取り出す構成とする。1個のGDCで，1024×1024ドットであれば，メモリー・プレーン4枚，16階調（色）を，512×512ドットならば，メモリー・プレーン16枚，64K階調を取ることができる。後述するGDCの並列動作機能を用いれば，メモリー・プレーンの増設は制限なく行える。

映像メモリーとしてダイナミックRAMを用いる設定を後述するSYNCコマンドによって行うと，HSYNC時にリフレッシュ・アドレスを$AD_{0\sim15}$の下位8ビットから出力する。この時，アドレス・ラッチの下位8ビットの出力のみを有効とし，$\overline{CAS}$信号に立ち下がり状態が生じないように外付け回路で制御する（図4）。RASオンリー・リフレッシュが可能となる。表示期間中には連続したアドレスが映像メモリーに与えられるので，リフレッシュは特に行う必要はないと考えがちだが，拡大表示時やインタレース走査時，映像メモリーの一部のみを表示する時などに，ダイナミックRAMが定めるリフレッシュ周期規格を満足しなくなる。

表示域と映像メモリーの水平方向の大きさを独立に定義

CPUがGDCに送出する映像メモリー・アドレスは絶対番地で与える。X, Yの座標による方が，より直接的であるように

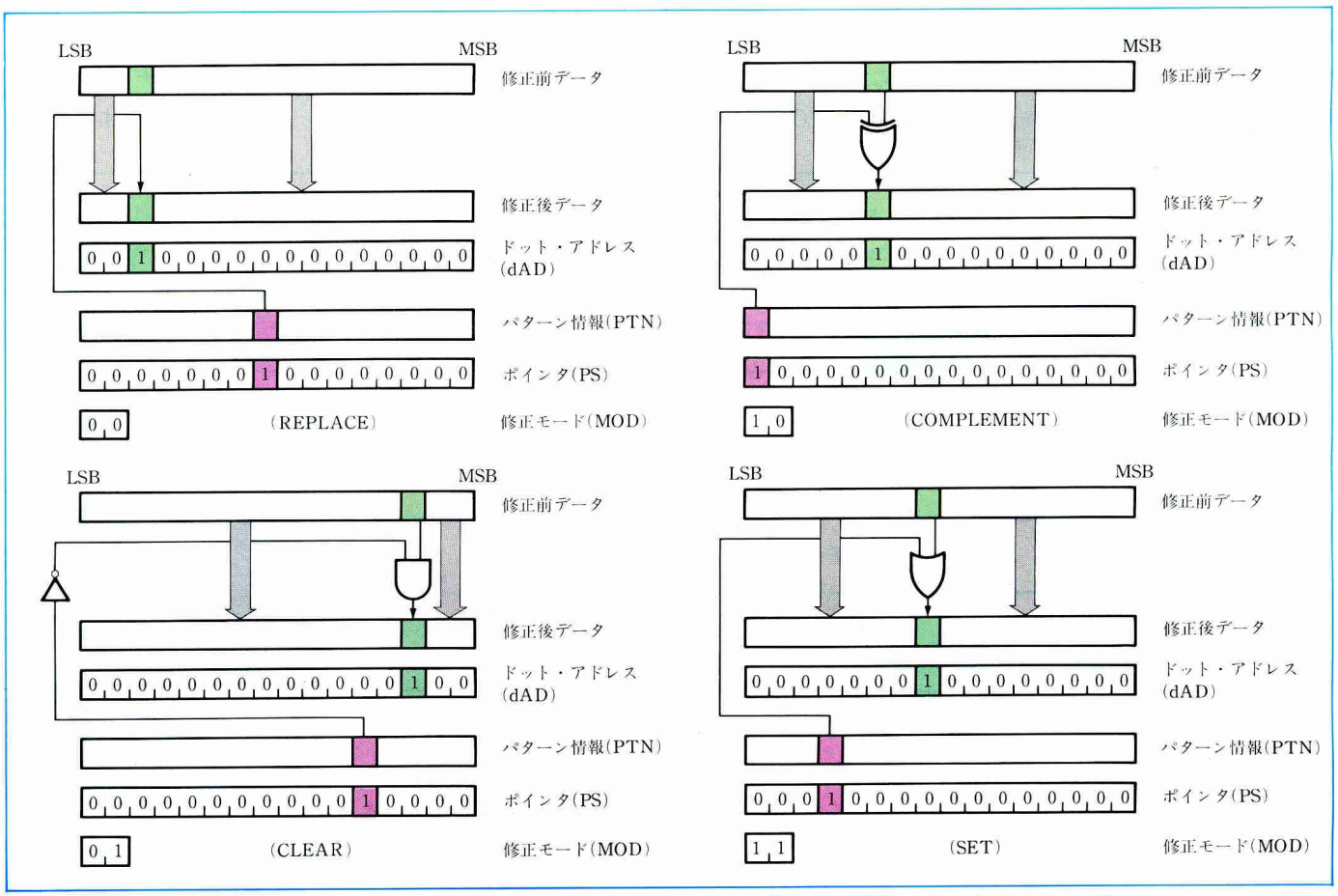


図6 グラフィック・モード時のドット修正法。描画時，GDCに読み込まれた16ビットの映像メモリー内容に対して修正を加える。修正用ゲート群は各ビットに1個割り当てているのでdADをすべて“1”とすると全16ビットに対して同時に修正を加え得る

見えるが，汎用性や拡張性を損なってしまう。座標原点位置の定義や，X，Y 座標の変化域など応用装置にとっては予期し得ない不確定要素が数多くあり，ハードウェアとの一体性が欠如するためである。アドレスを絶対番地で与えると映像メモリー構成を自由に選択でき，表示とは無関係な分野での大容量メモリー制御用としての応用も開ける。

水平方向の表示域と映像メモリーの大きさとを，独立に最大255アドレスまでコマンドによって定義できる。このため，表示域やメモリーの大きさを，512×512や1024×1024のように256または512ドット単位で定義する必要はない。水平方向に対しては2アドレス32ドット単位で255アドレス4064ドットまで

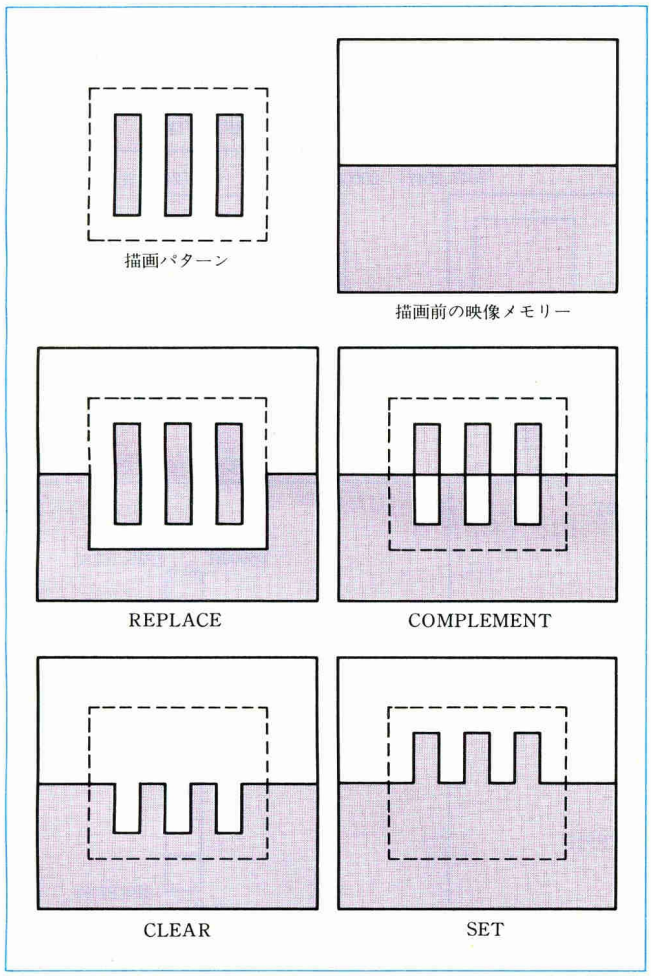


図7 4種類のドット修正モード。プログラムで任意に設定できる描画パターンに従って，4種のドット修正モードを選択できる。EXOR，NAND，ORによるドット単位論理演算としてとらえ得る

設定でき，垂直方向のドット数は映像メモリーの記憶容量によって決定される。図8に示すような種々のメモリー構成，表示域定義ができる。例えば，図8(b)のように，水平方向に4064ドットを定義した時，垂直方向には最大1028ドットを定義できる。一般に使用されるCRTの縦横比（アスペクト・レシオ）は約3:4である。このCRTに512×512の表示を行うと1ドット

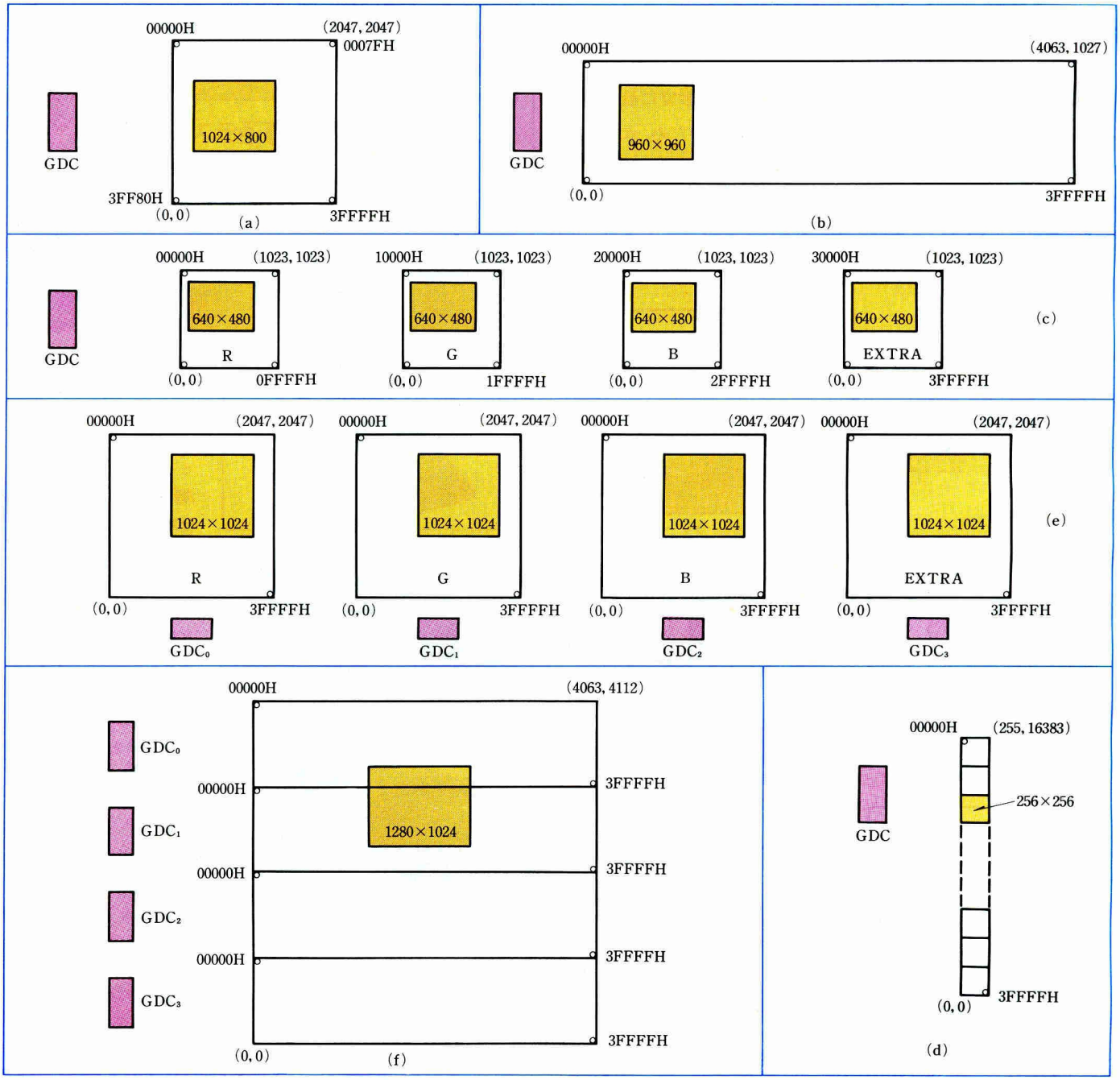


図8 映像メモリー構成と表示域の設定例。映像メモリー領域と表示領域を独立にコマンドで設定できるので装置設計上，融通性がある。GDCの並列動作によって，更に拡張性が増す

が正方形とならず，円が楕円として表示される。この時，表示域定義を縦横比に合わせて，例えば480×640ドットとすれば，円は円として表示できる。表示開始アドレスを変更するコマンドにより，表示域を全方向にドット単位でパニングし，図における非表示域を表示することもできる。X，Y座標値と絶対番地との変換は映像メモリー構成に従ってCPUでのソフトウェアによって次のように簡単な操作で行える。

図8(a)のように，座標原点（0, 0）をメモリー・マップ左下に取り，X，Y座標最大値をX_{MAX}，Y_{MAX}とする。マップ左上を0番地とし左から右へ走査され，アドレスが“1”ずつ増加する。

水平方向アドレス総数Pは

$$P=\mathrm{INT}\{(X_{MAX}+1)/16\}$$

ある点（X, Y）のワード・アドレス（EAD）およびドット・アドレス（dAD）は

$$\mathrm{EAD}=P*(Y_{MAX}-Y)+\mathrm{INT}(X/16)$$

$$\mathrm{dAD}=\mathrm{REM}(X/16)$$

ここで，INTは整数部，REMは剰余部を示す。アドレスは加減算およびシフト操作により容易に導出できる。INT(X/16)はXの4ビット右シフトにより，REM(X/16)はXの下位4ビットを残し，上位ビットを“0”にすれば求められる。更に，Pの値が“2のべき乗”であれば$P*(Y_{MAX}-Y)$は（$Y_{MAX}-Y$）の左シフト操作によって得られる。

21種類のコマンドを解釈する

GDCは，表1に示す21種のコマンドを解釈する。各コマンドは8ビットで構成され，必要に応じてパラメータを付随させて使用する。標準的な8ビットまたは16ビットのプロセサで制御できる。CPUが供給するアドレス信号（図4中のA_0）は，書き込み信号（$\overline{WR}$），読み出し信号（$\overline{RD}$）との組み合わせによって，データ・バス信号（$DB_{0\sim7}$）の種別を表2のように定義する。コマンド/パラメータ転送時およびFIFO内容読み出し時には，CPUは前もって$A_0=0$，$\overline{RD}=0$，$\overline{WR}=1$によってデータ・バス上に出てくるステータス・フラグを検出し，FIFOの状態を把握している必要がある。8種のステータス・フラグが用意されている（表2）。コマンド/パラメータ転送には次の方法があり，用途に応じて選択する。

① FIFO FULLが“0”であることを確認して転送する。

② FIFO EMPTYが“1”であることを確認して，最大16バ

イトを連続転送する。この時，主記憶-GDC 間 DMA 転送によって高速転送してもよい。

FIFO 内容読み出し時には，次のようにして行う。

① DATA READY が“1”であることを確認して読み出す。

② FIFO FULL が“1”であることを確認して，16 バイトを連続的に読み出す。この時，GDC-主記憶間 DMA 転送によって高速読み出ししてもよい。

FIFO は CPU から見て，読み出し/書き込み兼用として動作する。初期化動作以後，書き込み専用として動作し，描画アドレス制御部や制御 ROM が個々の処理を実行中である時にはコマンド/パラメータを一時的に蓄積する。上記処理が終了すると即座に FIFO 内容は取り出され，制御 ROM はその解釈処理を開始する。READ などの読み出しコマンドが FIFO から取り出され，解読されると，CPU からなんらかのコマンドが発行され

表1 コマンド・リストと動作の内容。21 種のコマンドを持ち，可変バイト数のパラメータを付加して使用する。いったん FIFO に蓄積された後，制御 ROM を駆動して諸制御機能を達成する

コマンド	動作内容
動作制御	
RESET	初期化動作
SYNC	動作モード，同期信号波形の定義
MASTER/SLAVE	マスタ動作，スレーブ動作の選択
表示制御	
START	表示の開始の指示
STOP	表示の停止の指示
ZOOM	拡大表示係数，拡大描画係数の設定
SCROLL	表示開始アドレス，表示領域の設定
CSRFORM	文字表示時のカーソル形状などの設定
PITCH	映像メモリー水平方向ワード数の設定
LPEN	ライトペン・アドレスの読み出しの指示
描画制御	
VECTW	描画に必要な各種パラメータの設定
VECTE	直線，四辺形，円弧描画の実行の指示
TEXTW	グラフィックス・テキスト・コード設定
TEXTE	グラフィックス・テキスト描画実行指示
CSRW	描画アドレスの設定
CSRR	描画アドレスの読み出しの指示
MASK	マスク・レジスタ値の設定
映像メモリー制御	
WRITE	パラメータの映像メモリーへの書き込み準備
READ	映像メモリー・データの読み出しの指示
DMAW	映像メモリーへの DMA 転送開始の指示
DMAR	映像メモリーからの DMA 転送開始の指示

るまで，FIFOはCPU側から見て読み出し専用として動作する。つまり，映像メモリー内容や内蔵レジスタの内容を連続的に蓄積する。同時に，最初に蓄積されたデータはFIFOから取り出され，GDC上のレジスタ（図2には示していない）にセットされる。それによって，ステータス・フラグ(DATA READY)が“1”になる。前述のように，ステータス・フラグ読み出し条件を満足する信号，すなわち，$A_0=0$，$\overline{RD}=0$，$\overline{WR}=1$を(GDCが非選択時に)外付け回路によって与えると，ステータス・フラグは$DB_{0\sim7}$端子から静的な出力信号として取り出せる。タイマ割り込み(VSYNC)，ライトペン検出割り込み(LPEN)，FIFO BUSY（FIFO FULL）などのハードウェア・ハンドシェーク用信号として使用できる。

SYNCコマンドによって動作モードを選ぶ

SYNCコマンドの構成を図9に示す。このSYNCコマンドの第1パラメータ中のCHRおよびGのビット操作によって，次の3種の動作モードのうちの任意の1種を選択する。

(1) グラフィック・モード

(2) 文字/グラフィック混在モード

(3) 文字モード

動作モードの選択によって，描画時の描画アドレス計算およ

表2 データ・バス信号内容とステータス・フラグ。4種の8ビット構成のデータ・バス信号を読み出し/書き込みする。下位3ビットのステータス・フラグをコマンド・レベルでのCPU-GDC間データ転送時に使用する

データ・バス信号内容		
	$\overline{RD}=0$	$\overline{WR}=0$
$A_0=0$	ステータス・フラグ読み出し	パラメータ書き込み
$A_0=1$	FIFOデータ読み出し	コマンド書き込み

ステータス・フラグ		
DB	名称	状態
0	DATA READY	読み出しデータのセット終了
1	FIFO FULL	FIFO内容満杯
2	FIFO EMPTY	FIFO内容空白
3	DRAWING	描画実行中
4	DMA EXECUTE	DMA転送中
5	VERTICAL SYNC	垂直同期信号期間
6	HORIZONTAL BLANK	水平消去期間
7	LIGHT PEN DETECT	ライトペン光検出終了

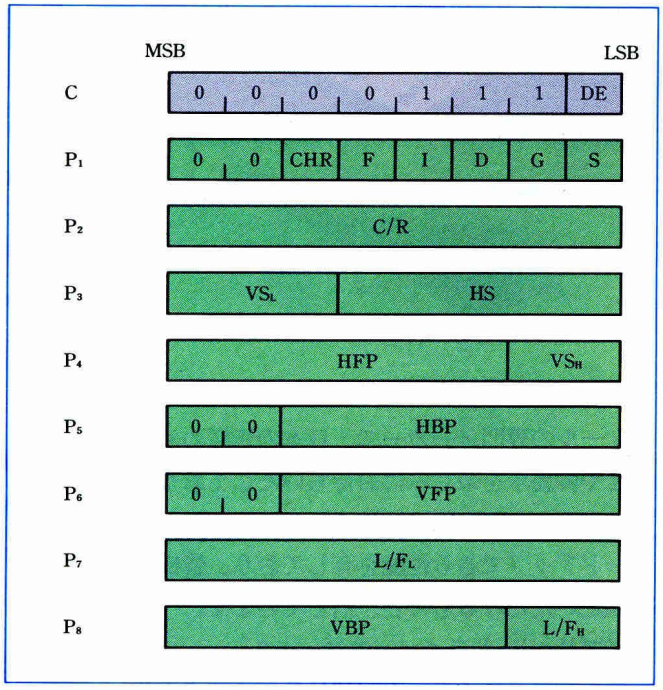


図 9　SYNC コマンド。コマンド/パラメータは 8 ビット構成であり，コマンドは処理の実行や方向付けを行う。パラメータはレジスタ値の設定などに使われる

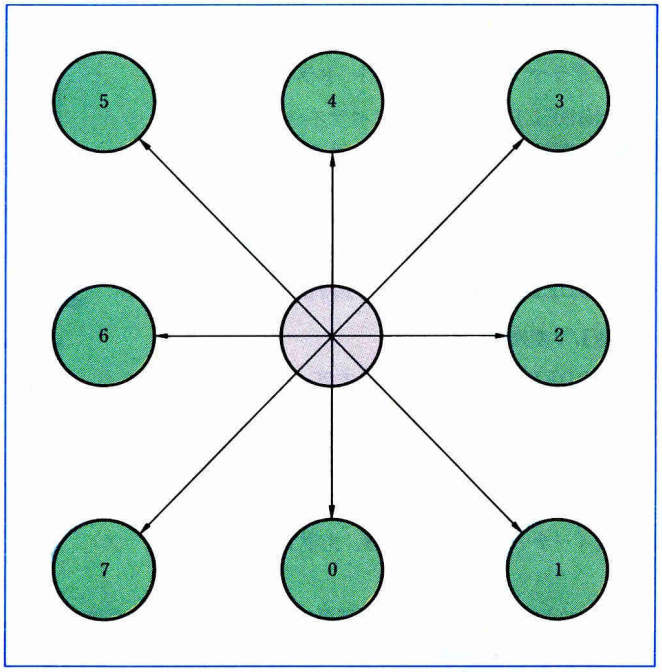


図 10　描画方向の定義。45 度単位で 8 通りの描画開始時基線方向を定義する。GDC 内での描画アドレス演算時，描画方向変化が検出されると一時的に方向値を変化させて，アドレス算出する

び表示データの修正法や $AD_{0\sim15}$，A_{16}，A_{17} の端子機能に変化を与える。表示アドレス制御部や同期信号発生部の動作は不変である。グラフィック・モードではドット単位描画制御を行うため，描画ワード・アドレス EAD だけでなくドット・アドレス dAD をも算出する必要がある。GDC のように 16 ビットすべての映像メモリー内容をチップ内に読み込み，修正を加える形を採ると，ドット・アドレスは外部に出力されないので，装置設計者はドット・アドレスの役割について，意識しなくてもよい。

描画アドレス演算は図 10 の描画方向指示に従って，表 3 のようにして行われる。方向（DIR）が“3”，すなわち，左下から右上 45 度方向に描画方向が設定されている時，ワード・アドレスを水平方向総アドレス数だけ減算し，ドット・アドレス・レジスタの内容を左方向（上位方向）に 1 ビット回転する。これは，表示時，1 本上の走査線でドット位置が右に 1 ドット移動した

表 3　描画方向に基づく描画アドレス演算。グラフィック描画時にのみ行うドット・アドレス（dAD）演算は左右方向への回転動作によって達成する。MSB または LSB が“1”の時，描画ワード・アドレス EAD 演算が追加される

DIR	グラフィック描画時	文字描画時
0	EAD+P⟶EAD	EAD+P⟶EAD
1	EAD+P⟶EAD dAD(MSB)=1: EAD+1⟶EAD dAD⟶LR	EAD+P+1⟶EAD
2	dAD(MSB)=1: EAD+1⟶EAD dAD⟶LR	EAD+1⟶EAD
3	EAD−P⟶EAD dAD(MSB)=1: EAD+1⟶EAD dAD⟶LR	EAD−P+1⟶EAD
4	EAD−P⟶EAD	EAD−P⟶EAD
5	EAD−P⟶EAD dAD(LSB)=1: EAD−1⟶EAD dAD⟶RR	EAD−P−1⟶EAD
6	dAD(LSB)=1: EAD−1⟶EAD dAD⟶RR	EAD−1⟶EAD
7	EAD+P⟶EAD dAD(LSB)=1: EAD−1⟶EAD dAD⟶RR	EAD+P−1⟶EAD

注）EAD：ワード・アドレス　　LR：左方向回転
　　dAD：ドット・アドレス　　RR：右方向回転
　　P：水平方向総アドレス数

点のアドレスを意味する。CPUが描画開始時の方向を設定した後，描画中にはGDCが描画アルゴリズムに従ってドットごとに描画方向を算出する。図11の直線描画例では，CPUは方向“2”を与える。第3点に対するアドレス演算時，一時的に“3”に変更され45度右上の点をアドレスする。

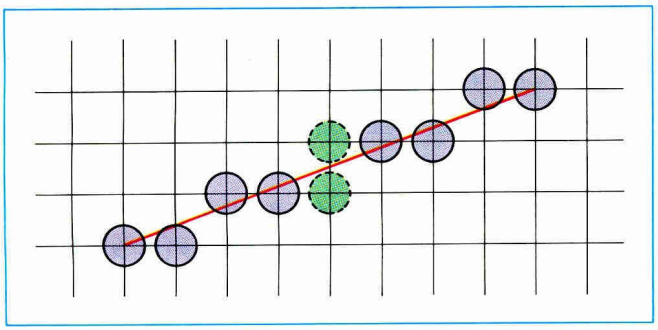


図11 直線描画の例。X方向に座標をドット単位で移動する。Y座標小数部が四捨五入条件を満足すると，Y座標を増加する。描画方向を180度変更すると，境界点（●のドット）でY座標位置の再現性がないが，ソフトウェアによって回避できる

文字モード時あるいは文字/グラフィック混在モード時でグラフィックス描画を行わない時，ドット・アドレス（dAD）はビット・マスク用として動作し，ポインタ（PS）の機能は無視される。ドット単位のアドレス演算は不要であり，パターン・レジスタ（PTN）に格納する文字コードをそのまま映像メモリーに送出すればよいからである。ただし，REPLACEなど4種のドット修正機能は生かされている。文字描画時には，図12に示すように，dADが“1”のビットに対応する修正前データ（映像メモリー内容）とPTNデータに対して各修正を実行する。dAD内容はMASKコマンドによって，図12のように任意値

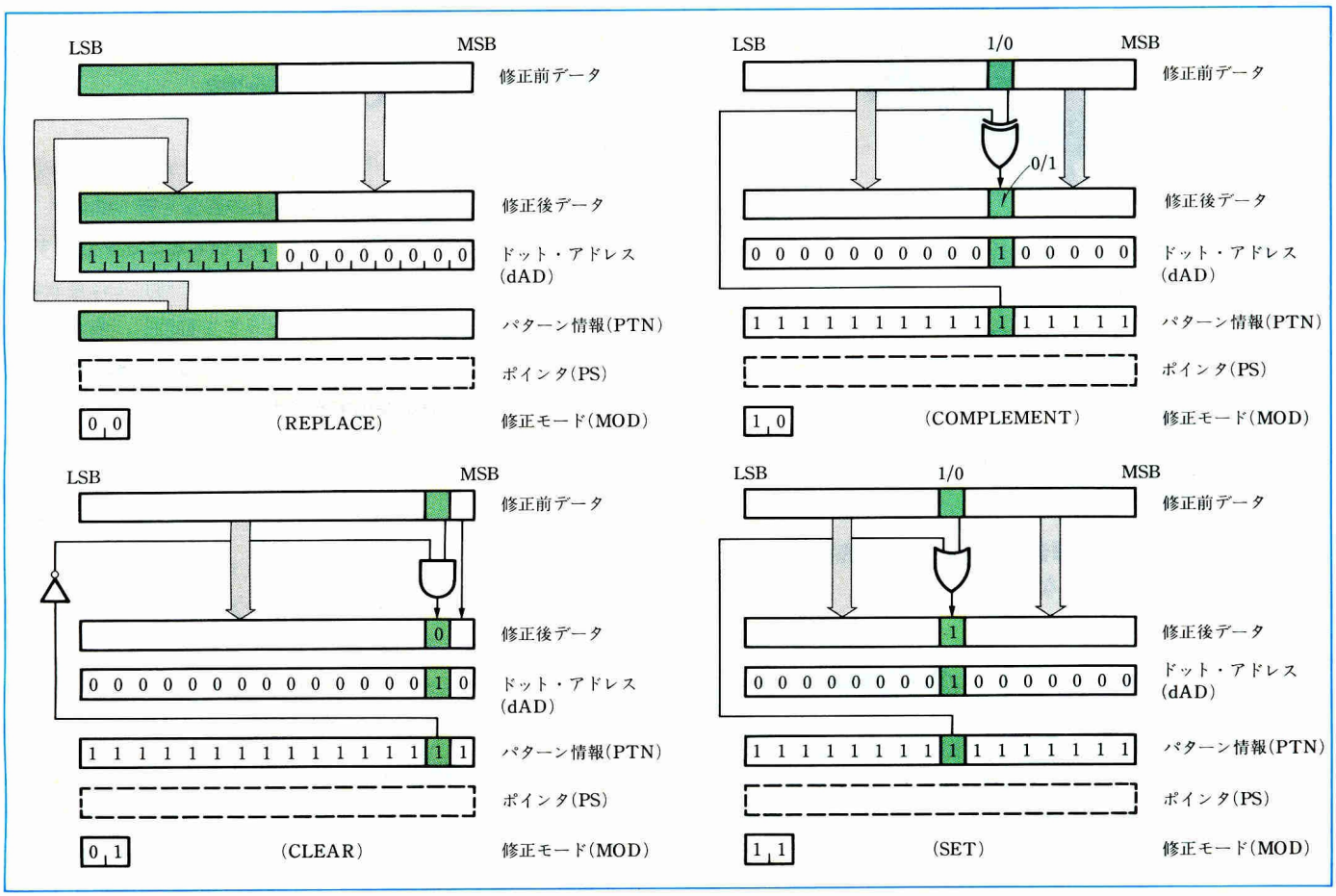


図12 文字モード時のドット修正法。ドット・アドレス・レジスタdADが“1”のビットに対応するデータにのみ修正を加える。ポインタの機能は無視される

に設定できる。特に，文字モードでは文字属性設定ビットに対するビット・マスク操作や上位/下位バイトの判別などに，また，グラフィック・モード時には，全ビットを“1”とし16ドット同時描画を行う映像メモリーの高速クリアなどに有効である。グラフィック・モード時であっても，dADのMSB（most significant bit）とLSB（least significant bit）が常に“1”であるため，ワード単位で描画アドレス計算が行われる（表3参照）。

SYNCコマンドの第1パラメータでは動作モード設定のほかに，Dはリフレッシュ有無指定，Iはインタレース走査指定，Fは描画タイミングの選択を行う。Fを“1”に設定すると，水平および垂直の表示消去期間にのみ描画動作を起こすので，表示にフラッシュを生じない。ただし，表示期間にも描画を行う“F＝0”の場合に比べて，平均化した描画速度は約1/5程度となる。第2パラメータ以降の設定でフロントポーチ，バックポーチ

描画アルゴリズムの改善

ラスタ走査型CRTを用いたグラフィック装置で直線や円弧などを描画することは，1画面分以上の容量を持つ映像メモリーに対して，ビット単位でデータ修正を加えることを意味する。水平方向に対してはビット単位，垂直方向に対しては走査線単位で区切られた格子の交点上に，ドットを描画し，表示の場合は，その映像メモリーを読み出して表示する。元来，直線や円弧などは連続的に値が変化するアナログ量なので，それを格子の交点上のドットとして近似し量子化するために，描画装置はなんらかの処置をする必要がある（本文中の図11参照）。これを達成する一番簡単な方法は，あらかじめ座標変位テーブルを多数用意しておき，描画の種類に従ってテーブル参照の方法を変更しながら描画する近似的な手法である。しかし，描画結果に部分的な歪が生ずる場合が多く，好ましくない。

描画精度を向上させるには，方程式の解を求める方法が一番確実である。通常，水平または垂直方向について最小ドット単位刻みで座標を移動しながら，もう一方の座標値を導出する。図11の例では，X軸方向に座標を“1”ずつ移動しながらY座標を算出する。式を解いた結果得られた座標値は実数なので，小数部を四捨五入することにより整数座標位置に近似する。誤差は±0.5座標以内に収まる。したがって，整数座標格子に対す

を持つ水平/垂直同期信号幅および表示期間を定義する。これらの値は水平関係を2クロック，垂直関係を1走査線単位で定義できるので，NTSC(National Television System Committee)のテレビジョン放送基準に合致した同期信号を生成できる。MASTER/SLAVEコマンドによってGDCをスレーブ動作設定した場合，外部同期信号を受け付け，その降下時に同期信号発生部を初期化する機能を持つ。このため，多数個のGDCを同一システム内で互いに同期が取れた状態で並列動作することや，テレビ・カメラなどの映像機器から出力される映像信号にGDCが制御する映像信号を重畳することなどが可能である。

描画の実際

図13は，512×512ドットの解像度で直線および四辺形を，白を含む7色表示した例である。この描画装置の映像メモリー構成は図14のようになっている。四辺形部は，対角線上の点(0,

る近似としては誤差がない。図11の例では，Y座標の変位は“0”または“±1”なので，描画開始時の座標が既知であり，その値が記憶されているならば，算出される整数部データは使用する必要性がない。

座標変化が生じる状態の検出だけが行えれば，あらかじめ記憶しておく描画開始時の座標値を±1することにより描画点の座標を算出できる。換言すれば，小数部演算の結果のみに注目し，その結果と“1/2”との大小比較を行えば，整数部は容易に導出できる。ただし，方程式をそのままの形で演算したのでは，実数乗除算や三角関数演算を実行せねばならない。更に，これらの演算は1ドット描画ごとに実行するので，たとえ高速のCPUや演算専用LSIを用いたとしても，描画速度の飛躍的な向上は期待できない。原点に立ち戻って，描画アルゴリズムを考え直す必要がある。

整数加減算だけで描画点を算出

そこでGDCでは，方程式の解のうち小数部の結果のみに注目しつつ描画点を導出する方法を等価的に実行する方式を採っている。座標が変化した時の小数部データの変位がどの程度の大きさになるかをGDCが容易に判別できるように，CPUは後述する描画用パラメータを1回の描画の実行に先立って，1回だけ算出する。パラメータはGDCに送出され，GDCはこのパラメータを用い整数加減算処理だけで連続的に描画点を算出す

0)，(511, 511) を結ぶ四辺形から X，Y 座標を“16”ずつ移動した16個の四辺形と，(255, 0)，(255, 511) を結ぶ45度傾斜四辺形から同様に X，Y 座標を移動した傾斜四辺形で構成されている。順次，色を変化させながら描画した。白の描画時にはR，G，B各メモリー・プレーンに対して，同一座標異描画アドレスに各々同一の描画を行う。総計4万1280ドットを描画している。直線部は，(0, 0)，(511, 511) を結ぶ直線から始め，同様に座標と色を変化させ32本の直線描画を行い，総描画ドット数は2万6624ドットである。0.1秒程度で瞬間的にこれらの描画をすべて終了する。円弧描画においても，この描画速度は変わらない。描画基線の傾きが45度単位であれば，四辺形および矩形領域に対してドット単位連続描画を行うグラフィックス文字描画を，一括して高速描画する機能を持つ。45度以外の場合は直線描画の集合として実現する。文字をドットで表現する方法として，直線描画によって文字形成するストローク法がある。この方法は描画単位がドットではなく直線であるため細かいアク

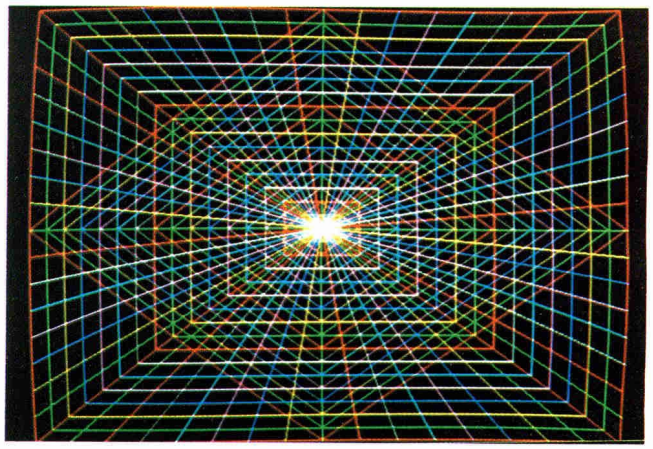


図13　白を含む7色で直線，四辺形を描画した例

る。

2点間の直線補間を行う直線描画の場合には，次のようなアルゴリズムに従って描画用パラメータを決定している。2点間の X および Y 座標の変位を，それぞれ，ΔX，ΔY とすると，直線の方程式は次式で表せる。

$$(\Delta Y/\Delta X)*N=I+F \qquad (1)$$

ここで

N：X 座標値。“1”から始まり“1”ずつ変化する。

I：Y 座標の整数部。描画開始点からの変位を示し，開始点では“0”。

F：Y 座標の小数部。

小数部のみに注目して式(1)を変形し，四捨五入の条件を加味すると式(2)が得られる。

$$F=(\Delta Y/\Delta X)*N-I\geqq 1/2 \qquad (1)'$$

$$2*N*\Delta Y-(2*I+1)*\Delta X\geqq 0 \qquad (2)$$

N を“1”ずつ変化させながら式(2)の正負判定を行えば，Y 座標の変化点が得られる。式(2)は実数を含まず整数（ΔX，ΔY も整数）だけで表現されるので，このままの形で直接演算しても構わないが，X 座標変化時の式(2)の差分，つまり，$N=n$ の時と，$N=n+1$ の時の式(2)の値の差が既知ならば，整数加減算のみによって式(2)の結果が算出できる。すなわち，ある点の式(2)の値と，その差分が与えられれば，次の描画点における式(2)の値が得られる。したがって，CPUは次の(a)～(d)で示す4種の描画用パラメータを演算生成し，GDCに送出する。

(a) 第1回目の描画アドレス計算に必要な式(2)の初期値データ。

(b) 式(2)が正の時，次の描画点における式(2)を導出するための差分データ

(c) 式(2)が負の時，次の描画点における式(2)を導出するための差分データ

(d) 最大描画ドット数

ここで，(d)は描画終了点の検出のために使用する。(a)は，式(2)において，$N=1$，$I=0$ の場合に相当する。したがって，

$$2*\Delta Y-\Delta X \qquad (a)$$

式(2)が正の時，次回の描画点における式(2)は N および I をともに“1”だけ加算したものとなる（直線描画の場合は，傾きが0～45度の範囲内に入るように取り扱うため）。したがって，その差分は

$$2*(\Delta Y-\Delta X) \qquad (b)$$

負の場合，N のみを“1”だけ加算したものとなる。したがって，

$$2*\Delta Y \qquad (c)$$

最大描画ドット数は，

$$\Delta X \qquad (d)$$

で求められる。

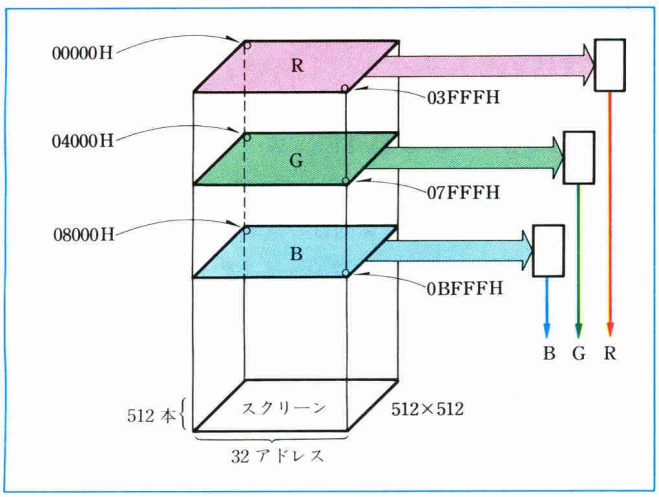


図14 512×512ドットの7色グラフィック表示時の映像メモリー・プレーン構成例。表示時には0Hから3FFFHのアドレスを与え，R，G，Bの3枚のプレーンから同時に映像出力を得る

図11において，左下から右上の点へ描画する場合と，逆方向への描画の場合とが考えられる。四捨五入法に基づく描画を行うので，小数部演算結果が境界値の“1/2”とならなければ，描画開始点がどちらにあろうとドット描画位置には再現性がある。しかし，描画進行方向に依存した描画座標の移動を行うので，境界値を持つドットがあると図の破線で示したドットのように位置ずれを生じる。同一図形のドット反転修正モードでの再描画による図形の選択的消去が不完全となる。しかし，描画方向の選択に制限を加え，描画開始点を一定とする単純なソフトウェア処理を施せば，この欠陥は除去できる。

円弧描画は，1/8弧を基本描画単位にしている。全円を描画するには，1/8弧を8回描く。直線描画の場合と同様，小数部の四捨五入法を採用している。詳細は述べないが，半径をrとした時，描画ドット総数，描画開始点からのマスキング・ドット数および$(r-1)$，$2(r-1)$，-1の5種のパラメータを作成し，GDCに設定する。そうすると，直線描画の時と同様，GDCは加減算操作だけで1/8弧を描くためのアドレスを計算できる。CPUでのパラメータ算出の際に，1～2回乗除算が必要になるが，1回の円弧描画について一度だけでよい。GDC内での描画アドレス計算は直線描画の場合と同一なので，描画速度は桁違いに速くなる。なお，CPUからGDCへ渡す描画パラメータは，本文中の表4にまとめてある。

セントをつけにくく，描画品質が良くない。GDC を使用して文字表現する時，ストローク方式を併用し文字を回転させてもよい。直線や円弧は，どのような方向に対しても描画できる（図15）。描画速度は描画方向に全く依存しない。

GDC に描画実行させるには，CPU のソフトウェアやディジタイザなどで座標を発生させ，この値を基にして CPU は，GDC が解釈できるレベルの，よりマイクロなコマンド/パラメータを作成する（これを描画前処理と呼ぶ）必要がある。次のような段階を経る。① 実線，破線などの線種を選択する。② REPLACE などのドット修正モードを選択する。③ 座標-絶対アドレス変換を行い，描画開始アドレスを算出する。④ 直線，円弧などの描画種類や描画方向を図 15，表 4 に従って決定し，描画パラメータを算出する。⑤ 描画実行の指示をする。VECTE，TEXTE，WRITE コマンドに続くパラメータ入力，の 3 種の描画実行コマンドを持つ。

図 13 の描画実行時，表 5 のコマンド/パラメータを CPU は GDC に送出している。まず，TEXTW で実線（FFFF），WRITE で COMPLEMENT 修正を指定している。これらの指定は変更がある時のみ行えばよい。2 回目の描画では特に設定していない。R（レッド）のメモリー・プレーン上の表示画面左下隅に定義した座標（0, 0）に相当する描画開始絶対ワード・アドレス“3FE0”とドット・アドレス“0”を CSRW で設定した後，図 10，図 15，表 4 に従って，VECTW により描画方向と描画パラメータを設定する。1 回目の描画時の方向は“3”であるが，2 回目は“2”となる。描画パラメータの負数は“2 の補数”表現する。描画実行コマンド実行前に送出するコマンドの送出順には制限はない。描画実行時に所期の値が設定されていればよい。表 5 に示すように 1 回の描画に要するコマンド/パラメータ総数は，直線描画時 15，四辺形描画時には 17 で済む。内蔵 FIFO の容量は 16 バイトであるので，描画実行中に次の描画用として送出されるコマンド/パラメータをほとんどすべて蓄積できる。したがって，描画を間断なく実行する。

図 16 は，メモリー-メモリー間 DMA 転送およびグラフィックス文字拡大描画の例である。中央部の 64×32 ドット矩形領域を選択的に映像メモリーから主記憶へ，各メモリー・プレーンごとに 1 回，計 3 回の DMAR コマンド発行によって DMA 転送し，そのパターンを左右下隅の 2 カ所に計 6 回の DMAW コマンドによって，主記憶から DMA 転送した。4 クロック/バ

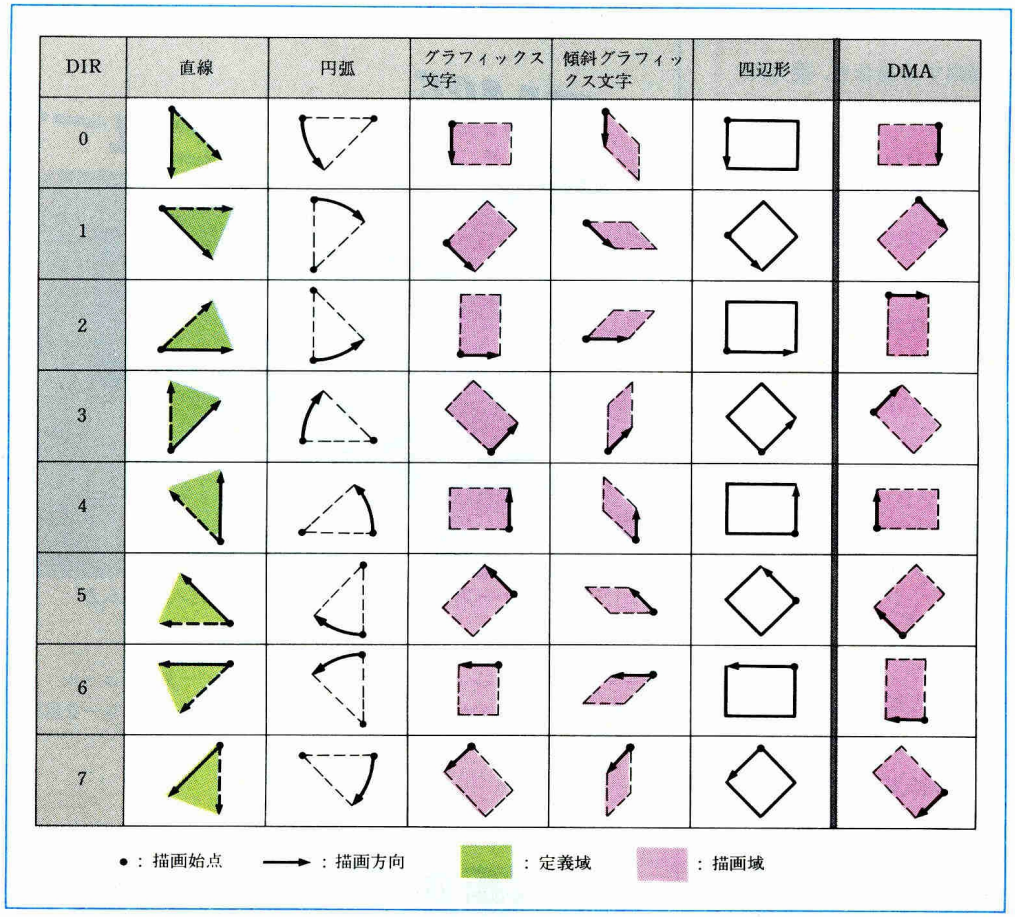


図 15　描画開始時の描画方向選択。直線描画では 1/8 領域内に含まれる直線を 1 個の描画方向で表現する。全円描画の場合には描画始点と描画方向を変化させ，1/8 弧を 8 回描画して形成する

イトの速度で，バイト/ワード変換を実時間で行いつつ，連続的なメモリー-メモリー間バースト DMA 転送を実現する。バスの分離がなされていない場合のメモリー間 DMA 転送の場合と比較して，2 倍以上高速である。同一図形の描画が多い時には，異なる映像メモリーに図形セルを描画し，任意個所への DMA 転送を繰り返してもよい。前記パターンを 3 倍に拡大し

表 4　描画パラメータ設定値。描画アルゴリズムに従って必要とされる描画パラメータを設定する。最大 5 種 10 バイトのパラメータを作成し GDC に送出するだけで，GDC は描画を実行する

	DC	D	D_2	D_1	DM
初期値	0	8	8	−1	−1
直線	ΔX	$2\|\Delta Y\|-\|\Delta X\|$	$2\|\Delta Y\|-2\|\Delta X\|$	$2\|\Delta Y\|$	——
円	$(r/\sqrt{2})\uparrow$	$r-1$	$2(r-1)$	−1	0
弧	N	$r-1$	$2(r-1)$	−1	M
四辺形	3	A^{*1}	B^{*1}	−1	A^{*1}
8×8 ドット グラフィックス文字	7	——	——	——	——
8×8 ドット以外の グラフィックス文字，塗りつぶし	B^{*2}	A^{*2}	——	——	——
DMA	B^{*3}	A^{*3}	$(A^{*3}/2)$	——	——
READ/WRITE	D^{*1}	——	——	——	——
1 文字 1 ドット	——	——	——	——	——

注）
ΔX：　X 座標変位
ΔY：　Y 座標変位
r：　半径
N：　描画総ドット数
M：　マスキング・ドット数
A^{*1}，A^{*2}：　第 1 描画方向ドット数
B^{*1}，B^{*2}：　第 2 描画方向ドット数
A^{*3}：　第 1 DMA 方向バイト数
B^{*3}：　第 2 DMA 方向バイト数
D^{*1}：　R/W バイト数
↑：　切り上げ

表 5　図 13 の描画実行時，CPU が GDC に送出するコマンド/パラメータの実例。直線部，四辺形部共に描画開始時と 2 回目の描画時の場合について示した

●直線描画

	コマンド名	C	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
(0, 0) ↓ (511, 511)	TEXTW	78	FF	FF									
	WRITE	21											
	CSRW	49	E0	3F	00								
	VECTW	4C	0B	FF	01	FF	01	00	00	FE	03		
	VECTE	6C											
(0, 16) ↓ (511, 495)	CSRW	49	E0	7B	00								
	VECTW	4C	0A	FF	01	7F	01	80	FF	7E	03		
	VECTE	6C											

●四辺形描画

	コマンド名	C	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}
(0, 0) , (511, 511)	TEXTW	78	FF	FF									
	WRITE	21											
	CSRW	49	E0	3F	00								
	VECTW	4C	42	03	00	FF	01	FF	01	FF	FF	FF	01
	VECTE	6C											
(16, 16) , (495, 495)	CSRW	49	E1	7D	00								
	VECTW	4C	42	03	00	DF	01	DF	01	FF	FF	DF	01
	VECTE	6C											

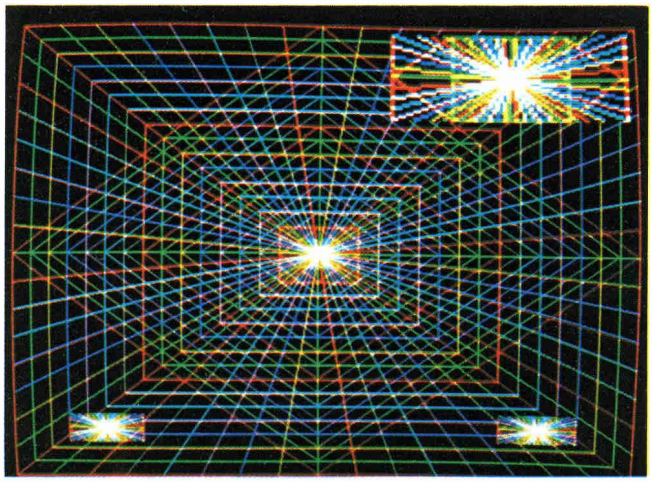


図16　図13の一部をメモリー-メモリー間DMA転送し，拡大描画した例

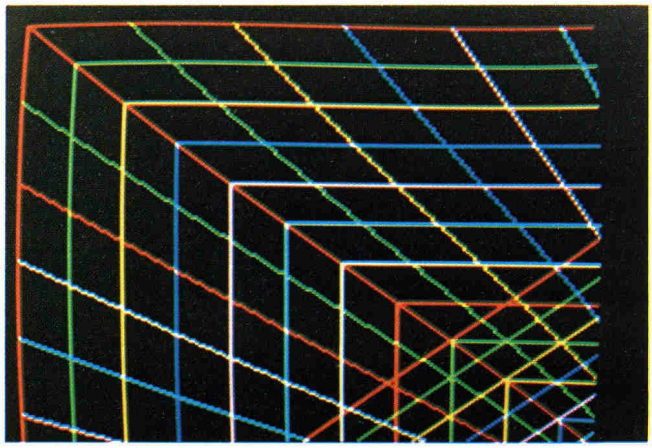


図17　図13の一部の3倍拡大表示例

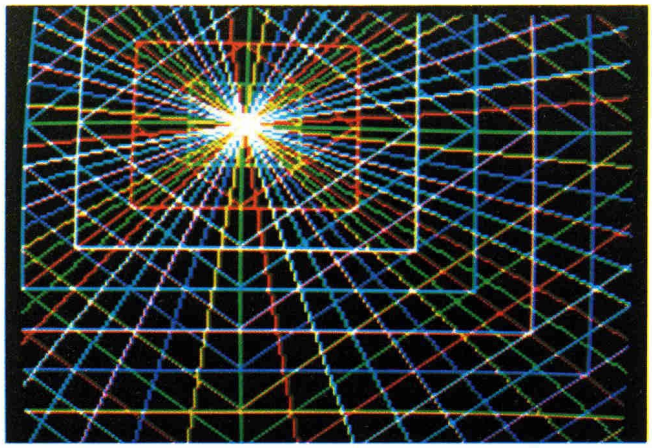


図18　パニングにより図13の中央部を表示した例

て図16の右上に描画した。グラフィックス文字描画は，ドット単位連続描画による任意のドット構成，文字外形，文字幅を持つ文字描画や矩形領域塗りつぶし描画だけにとどまらず，このような画像データの拡大描画にも応用できる。

描画時以外に表示時にも拡大制御機能を持つ。各々，拡大係数として1～16の整数をZOOMコマンドで設定できる。図17では，表示開始アドレスを変更せずに図13の左上部を3倍に拡大して表示した。拡大描画とは異なり，映像メモリー内容には変更がない。表示のタイミング制御のみを行うだけなので，瞬時に拡大できる。表示をこのように拡大すると不可視領域が生ずる。この不可視領域を表示するため表示領域を上下左右にドット単位で移動するパニングをSCROLLコマンドによって実現できる。図18は，パニングの結果，中央部を表示した例である。拡大表示時のほかに，表示領域より映像メモリー領域を大きく定義した場合にもパニングが応用できる。

描画実行に先立ち，CPUが算出しVECTWコマンドによってGDCに与える，描画パラメータの各種描画における値を表4に示す。直線，円などのグラフィックス描画だけでなく，DMA転送数や映像メモリー内容の読み出し/書き込み転送数制御にも使用する。このように描画パラメータ算出は簡略化されており，短時間で求められる。演算結果はGDCが描画処理実行中であってもFIFOに転送し蓄積できる。したがって，CPUによる描画前処理とGDCによる描画アドレス演算および映像メモリーに対する実質的な描画の，従来3段階に分けて実行していた処理を分散同時処理でき，高速で高性能な描画機能を持つ。高速のCPUや演算専用LSIを用いると，描画前処理はより高速化でき，高性能なグラフィック装置を構成できる。現用装置と比較する時，描画機能のみを取り上げても飛躍的な改善がなされており，直線描画で10倍，円描画で1000倍以上の描画速度を実現した。

グラフィック表示以外への応用範囲も広い

同期信号や表示アドレスの発生のみを行う従来型CRTCとは異なり，高速グラフィックス描画，多数個の並列動作，大容量メモリー制御，ダイナミックRAMに対するリフレッシュ，DMA転送などの諸機能を持っているため，表示とは無関係な分野への応用もあり，応用範囲は極めて広い。

カナや英数字だけでなく，漢字表示機能を持つ表示装置が普

及してきた。英数字などと同様に，漢字の情報処理コードが日本工業標準規格（JIS）によって決められている。[3] 7ビット・コード2バイトで構成される14ビットであり，3000字以上を指定している。カラー表示など種々の文字属性を1字ごとに付加した漢字表示を行うには，文字属性制御ビットを含めると16ビットを超える情報が必要で，制御が複雑となる。従来のDMA転送型CRTCでは，内蔵行バッファの記憶容量および構成から文字コードとして使用できるビット数は最大8に固定されており，漢字表示制御用としては使用できなかった。GDCが制御する映像メモリーの1アドレスには最大16ビットの情報を格納するので，漢字表示時には少なくとも2個のアドレスにまたがって文字属性を含む漢字1字分の情報を設定し，表示サイクルにおいてはその情報を同時にアクセスせねばならない。GDCは表示サイクル時の表示アドレス進行を通常の“1”ではなく，“2”によるインクリメント形態を採る機能を持つ。したがって，1表示サイクルに連続した2個のアドレスの表示データを同時に取り出すことが容易である。各々のアドレスに文字コードと文字属性コードを分割して設定してもよい。この機能は，漢字表示への応用のほかに，GDCのクロック周波数を規格以上に高くせずに，1行中の表示文字数を100字以上必要とする高解像度文字表示や1024×1024ドット以上のノーインタレース高解像度グラフィック表示への応用を可能にする。

このほか，GDCは，表示結果をハードコピーする際の，グラフィック装置とプリンタとのインタフェース設計を容易にし，また，図表の作成を含むワード・プロセサへの応用に対しても大きな威力を発揮するだろう。また，フロッピー・ディスク用バッファ・メモリーの制御用，ファクシミリ受信用バッファ・メモリー制御用など，多分野への応用が考えられている。

参考文献

1) Johnson, S., “Fast Raster-Scan System Displays Graphics and Images,” *Electronic Design,* vol. 28, no. 10, pp. 205-208, May 10, 1980.
2) Oguchi, T., Higuchi, M., Kamaya, M., Uno, T. and Suzuki, M., “A Single-Chip Display Controller,” *Digest of Technical Papers, 1981 IEEE International Solid-State Circuits Conference,* pp. 170-171, Feb. 1981.
3) 日本規格協会「JIS C 6226」，『JISハンドブック　情報処理 1981』，pp. 126-185，1981年4月.
4) 小口，樋口，鎌谷，鵜野「グラフィック制御用LSI」，『NEC技報』，pp. 58-63，1981年6月.

●