

NEC μ PD7220/72120 Related Public Documents

- [ISSCC](#) (International Solid-State Circuit Conference)
- [\$\mu\$ PD7220A User's Manual](#)
- Nikkei Electronics Magazine ([\$\mu\$ PD7220](#))
- Transistor Gijutsu (Technology) Magazine ([\$\mu\$ PD7220](#))
- Transistor Gijutsu (Technology) Magazine ([\$\mu\$ PD7220A](#))
- Nikkei Electronics Magazine ([\$\mu\$ PD72120](#))
- [\$\mu\$ PD72120 User's Manual](#)

Go to <https://www.oguchi-rd.com/LSI%20products.php> to get more detailed NEC μ PD7220/72120 related information such as;

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

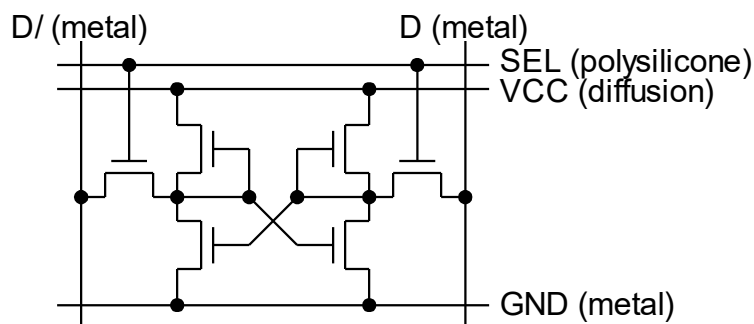
Go to <https://www.oguchi-rd.com/patents.php> to get patent information including NEC μ PD7220/72120 related patents.



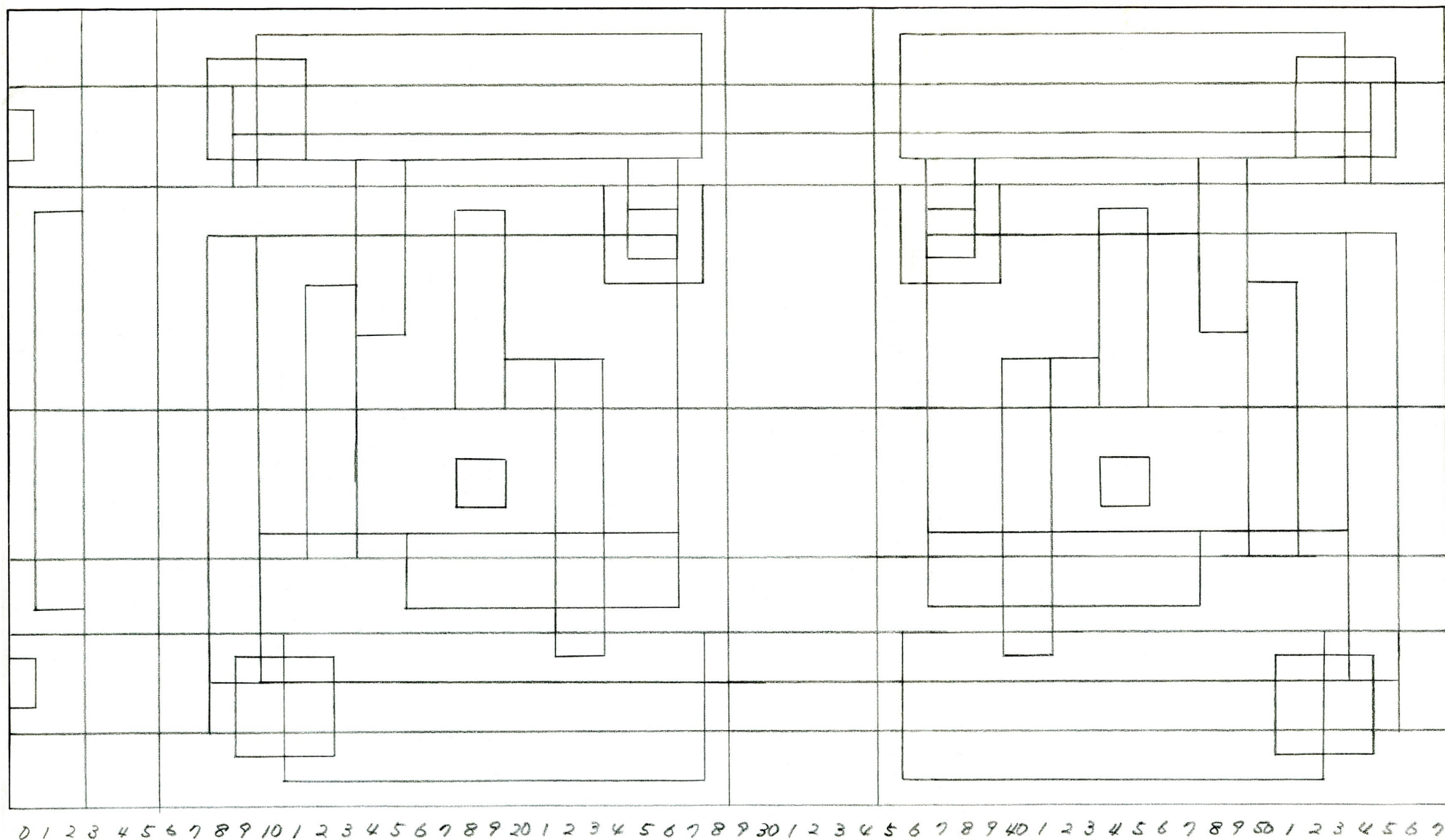
**Photo mask layout design example of six transistor SRAM using N channel silicon gate E/D
(Enhancement-mode for gate, Depletion-mode for load) MOS transistors for front cover of this issue**

I created this graphics image in one day using μ PD7220 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.



NIKKEI RAM CELL (CELL, MAGCELL, [ELEASM/CIM])

Original SRAM cell mask layout I designed before manual digitizing

日経エレクトロニクス

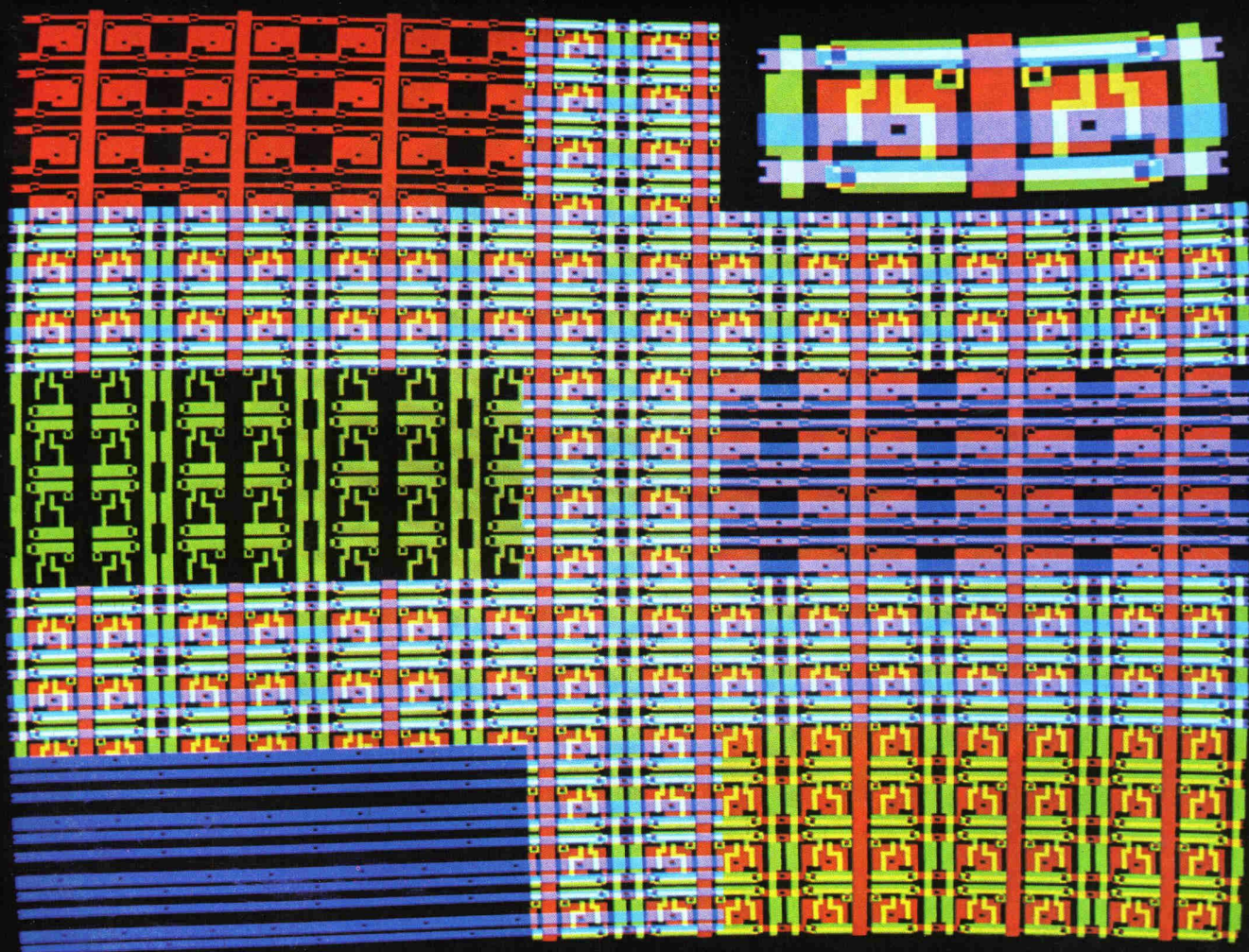
NIKKEI ELECTRONICS

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



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するため、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・プロセッサ端末装置として、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ずと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かなり速い。例えば、描画用プロセッサとしてバイポーラ・ビット・スライスCPU4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⁹⁾。その装置の描画速度は1.12 μ s/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 \times 16ビット(1024 \times 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・バスの使用权を要求するのは、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

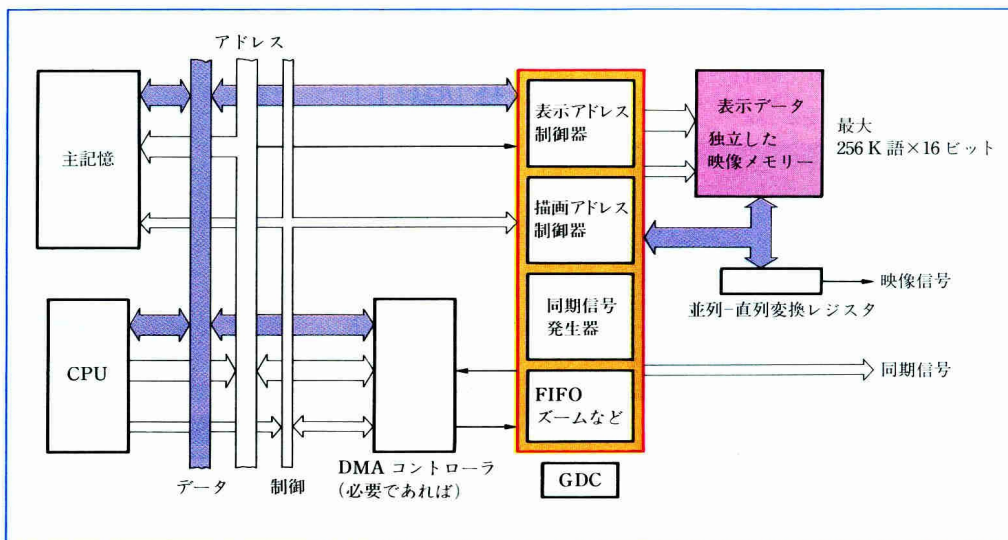


図1 GDCのシステム・ブロック図。GDCおよび映像メモリーはブラック・ボックス化され、独立して動作するI/O機器としてとらえることができ、拡張性に富む

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、シフトを持たせている。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 がいっぱいであれば、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

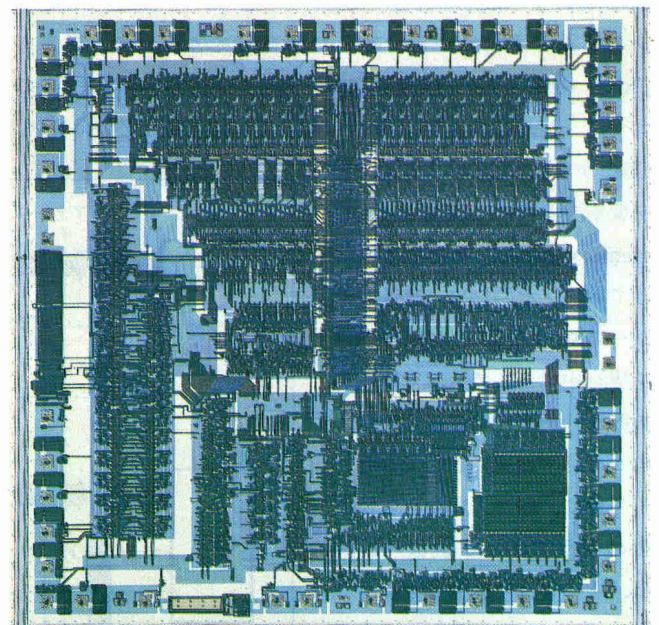
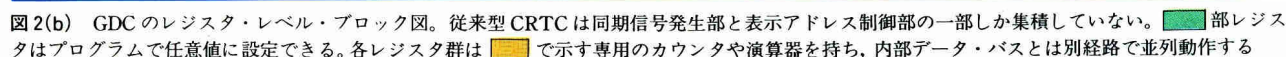


図2(a) GDC チップ。ゲート長 $3\mu\text{m}$ の n チャンネル・シリコン・ゲート E/D MOS トランジスタ 1 万 3000 個を搭載する。チップ・サイズは $5.29 \times 5.39\text{ mm}$ 。高速並列演算を実行するためハード・ワイアド論理を用いる。規則性のある繰り返し回路にまとめ、マスク・レイアウトを容易にしている

▶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。

▶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ビ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



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 (DB₀₋₇)、アドレス・ライン (A₀)、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、映像メモリと主記憶間の DMA 転送を制御する。映像メモリ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

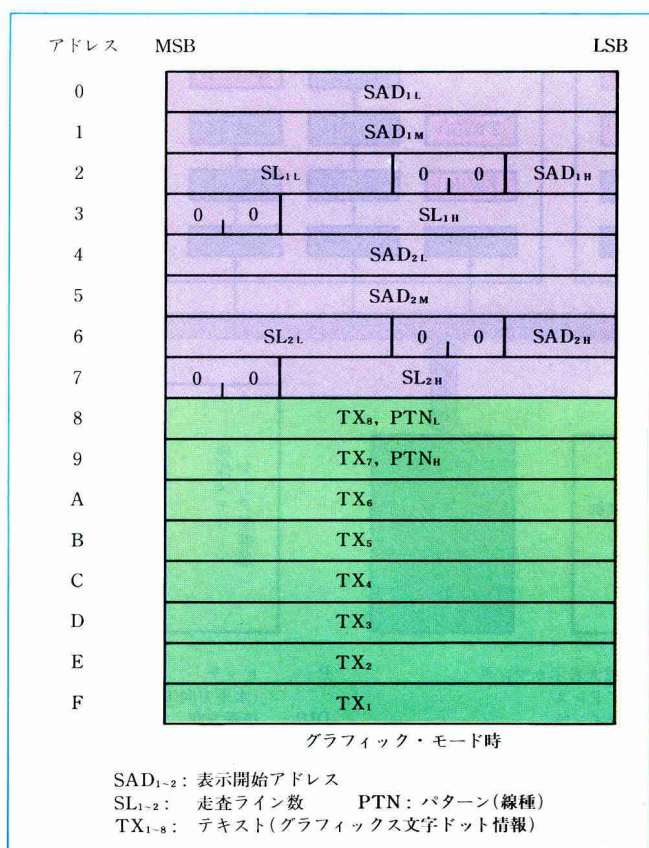


図 3 内蔵データ RAM マップ。コマンドによって任意なアドレスからデータを設定できる。下位 8 バイトは表示用、上位 8 バイトは描画時、線種情報記憶用または文字ドット情報記憶用として使用する

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

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、① DMA 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う、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 μ s 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、1 M バイ

画周波数が 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дрес/データ多重化出力端子 AD₀₋₁₅ およびアドレス出力専用端子 A₁₆、A₁₇ から出力される。サイクルの最初の 1 クロック (E₁) 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дрес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ため、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

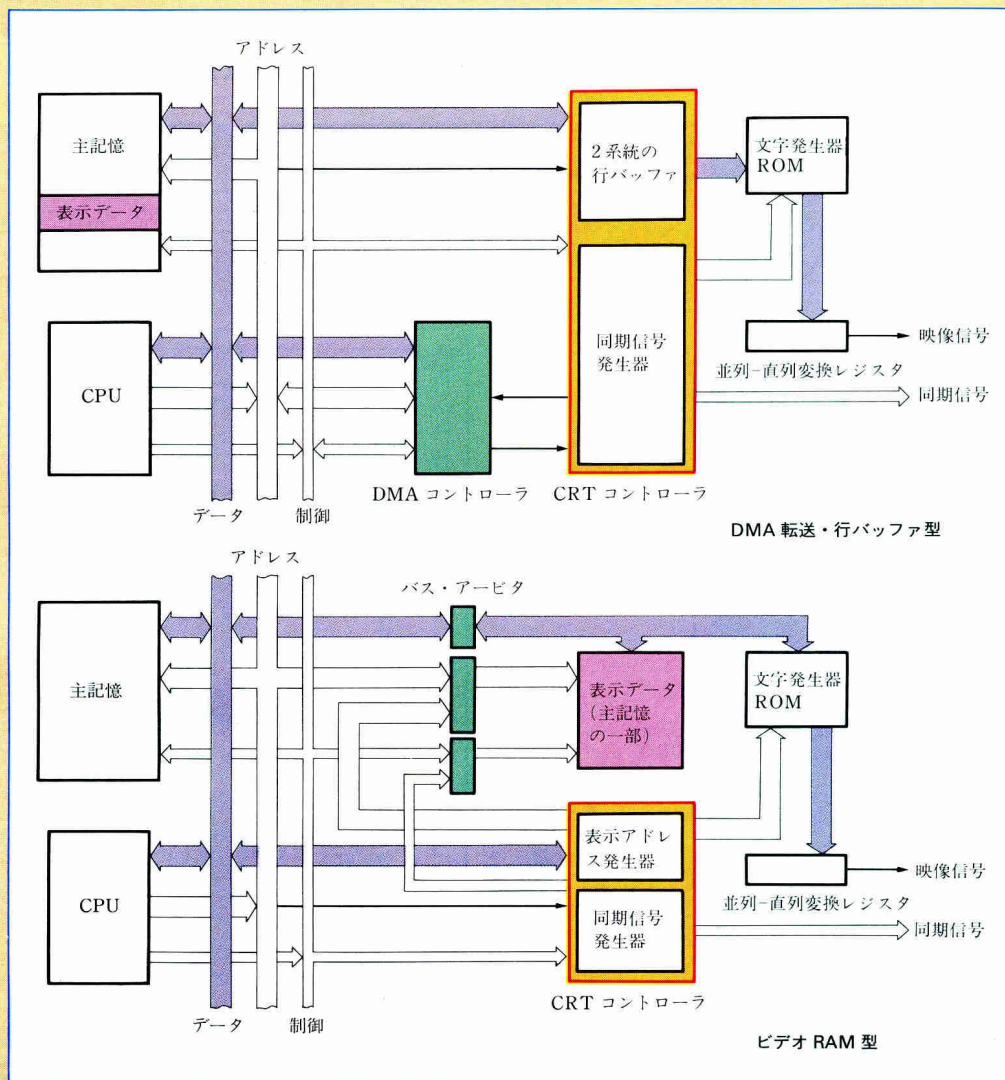


図 A 従来の CRT 制御用 LSI のシステム・ブロック図。両方式共に、表示データは主記憶中にマップされる。システム構成上の自由度や拡張性に乏しい。本文で紹介する GDC の場合は 部回路が不要となる

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。

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 $\overline{\text{RAS}}$ (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 $\overline{\text{RAS}}$ 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 $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ (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 にかかる1クロック間に、 $\overline{\text{DBIN}}$ (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、 $\overline{\text{DBIN}}$ 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 ($\overline{\text{WE}}$) の立ち

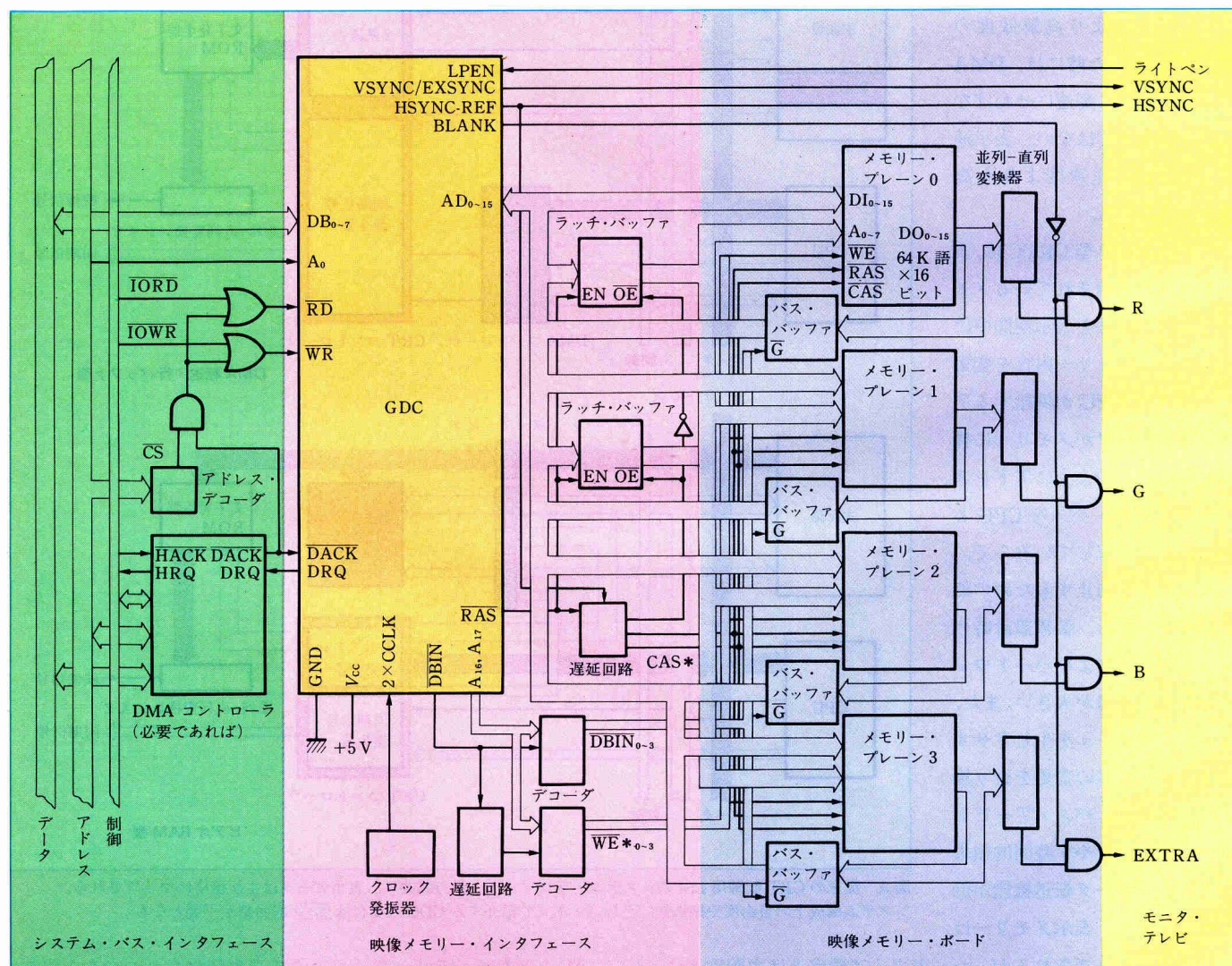


図4 インタフェース・ブロック図。1024×1024ドット16色グラフィック表示を簡単なインタフェースで実現できる。64Kダイナミック RAM 64個で構成する映像メモリー部を含め1枚のプリント基板上に実装できる

下がり時に映像メモリーに書き込む(write)。アドレスは、 $\overline{\text{RAS}}$ および $\overline{\text{CAS}}$ 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内に循環する。ドット修正モードはREPLACE, COMPLE-

MENT, 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

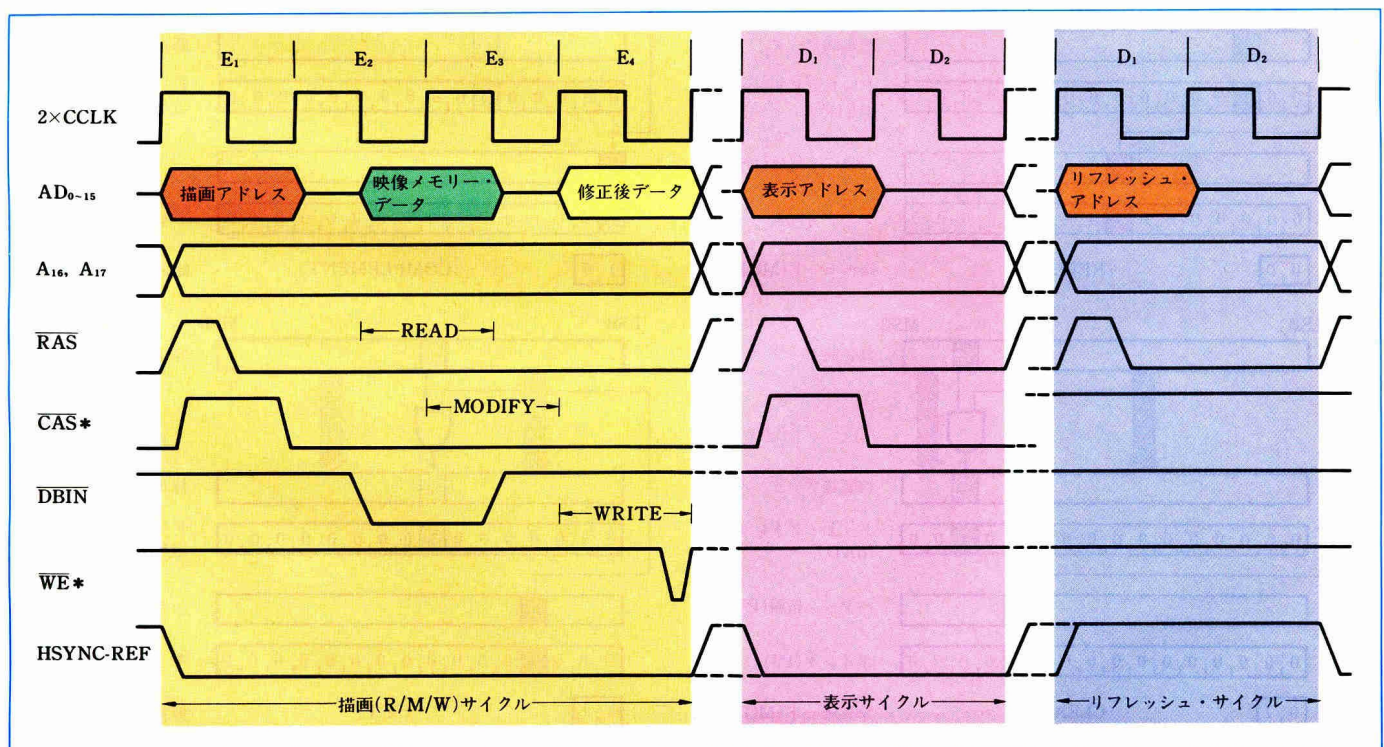


図5 映像メモリー制御タイミング。3種のアドレス・サイクルを持つ。 $\overline{\text{CAS}}^*$ 、 $\overline{\text{RAS}}^*$ は各々基準信号 $\overline{\text{RAS}}$ 、 $\overline{\text{DBIN}}$ を外付け回路によって遅延し、最適なタイミングに発生させる。リフレッシュはHSYNC時に行う

イナミック RAM に対して同一の $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ 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 $\overline{\text{DBIN}}$ と $\overline{\text{WE}}$ 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、64 K 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くと、HSYNC 時にリフレッシュ・アドレスを $AD_{0 \sim 15}$ 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、 $\overline{\text{CAS}}$ 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

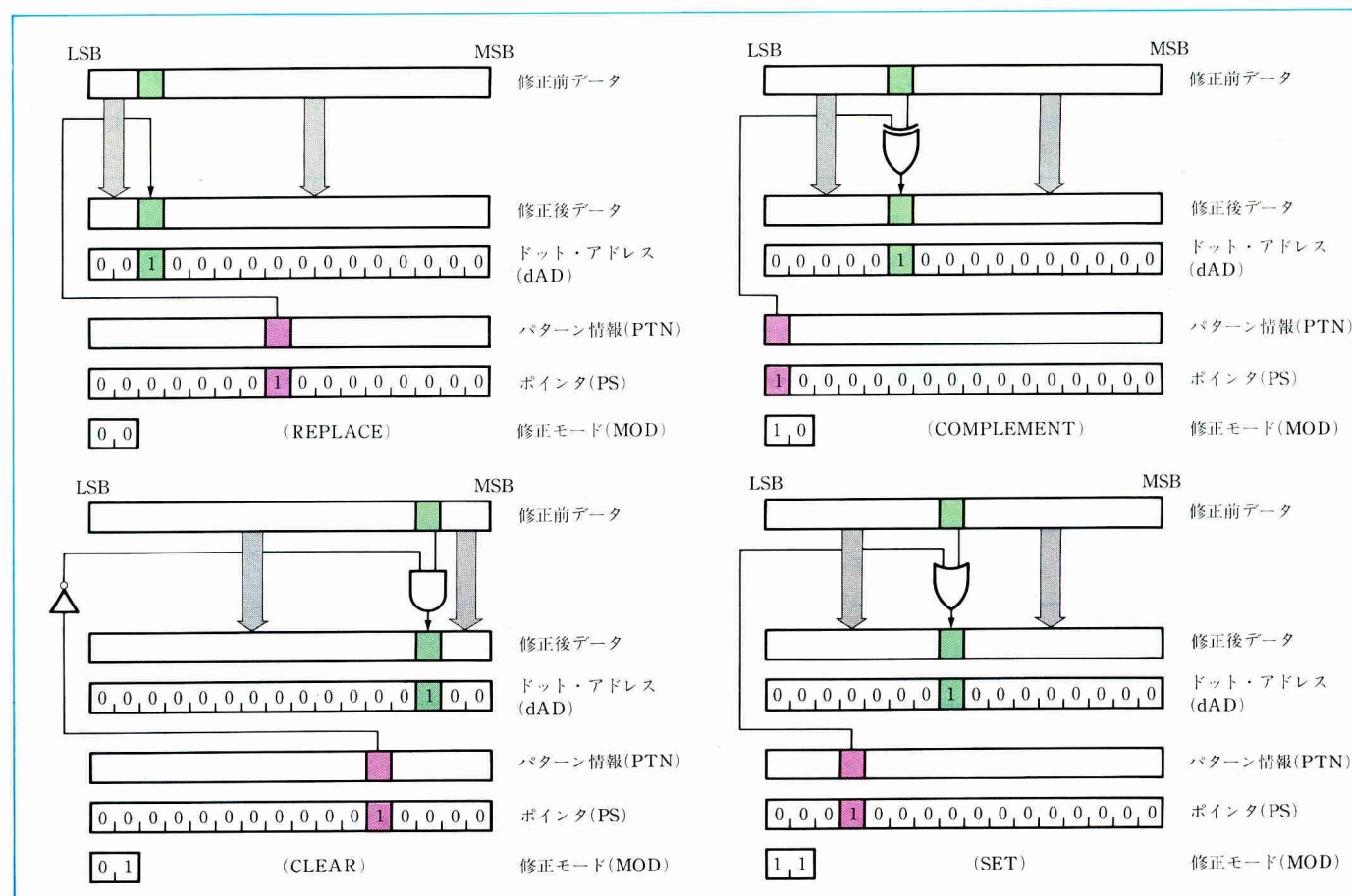


図6 グラフィック・モード時のドット修正法。描画時、GDC に読み込まれた 16 ビットの映像メモリー内容に対して修正を加える。修正用ゲート群は各ビットに 1 個割り当てているので dAD をすべて “1” とすると全 16 ビットに対して同時に修正を加え得る

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

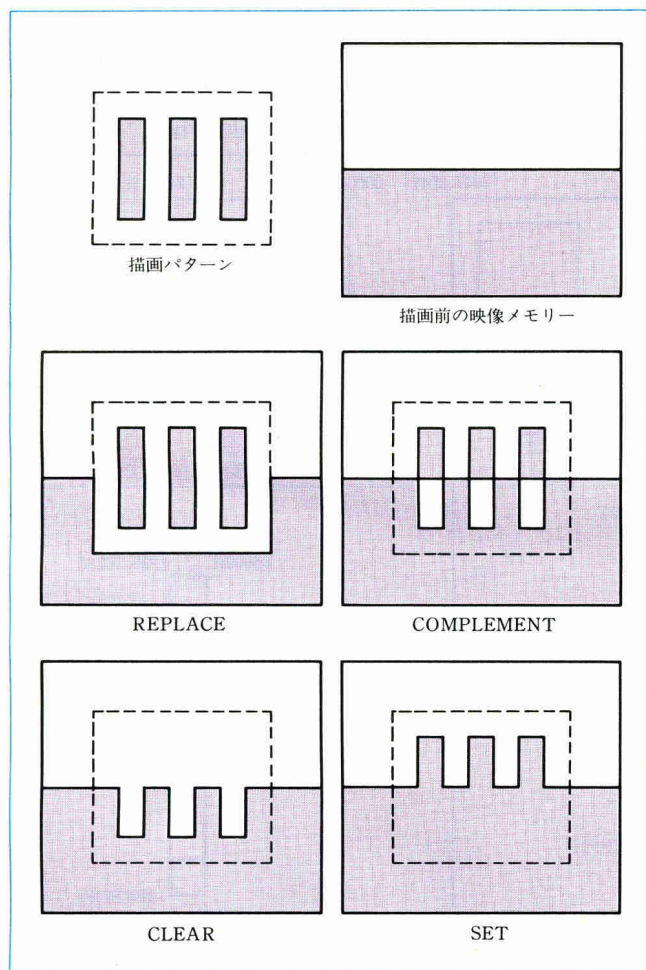


図7 4種類のドット修正モード。プログラムで任意に設定できる描画パターンに従って、4種のドット修正モードを選択できる。EXOR, NAND, ORによるドット単位論理演算としてとらえ得る

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

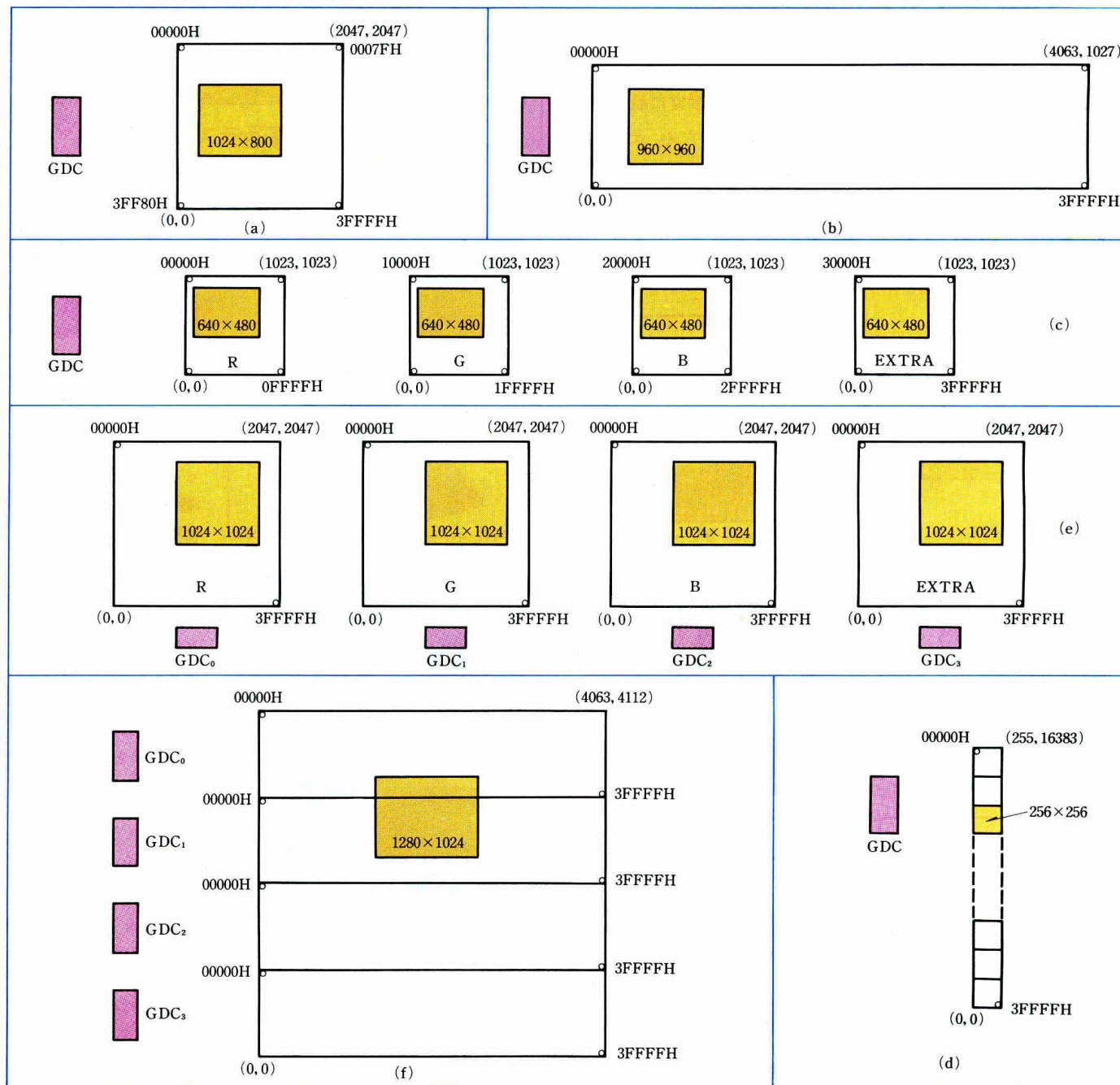


図8 映像メモリ構成と表示域の設定例。映像メモリ領域と表示領域を独立にコマンドで設定できるので装置設計上、融通性がある。GDCの並列動作によって、更に拡張性が増す

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

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

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

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

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

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 $P * (Y_{\text{MAX}} - Y)$ は $(Y_{\text{MAX}} - Y)$ 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 ($\overline{\text{WR}}$)、読み出し信号 ($\overline{\text{RD}}$) との組み合わせによって、データ・バス信号 (DB_{0-7}) 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 $A_0=0$ 、 $\overline{\text{RD}}=0$ 、 $\overline{\text{WR}}=1$ 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 SYNC MASTER/SLAVE	初期化動作 動作モード、同期信号波形の定義 マスタ動作、スレーブ動作の選択
表示制御 START STOP ZOOM SCROLL CSRFORM PITCH LPEN	表示の開始の指示 表示の停止の指示 拡大表示係数、拡大描画係数の設定 表示開始アドレス、表示領域の設定 文字表示時のカーソル形状などの設定 映像メモリー水平方向ワード数の設定 ライトペン・アドレスの読み出しの指示
描画制御 VECTW VECTE TEXTW TEXTE CSRW CSRR MASK	描画に必要な各種パラメータの設定 直線、四辺形、円弧描画の実行の指示 グラフィックス・テキスト・コード設定 グラフィックス・テキスト描画実行指示 描画アドレスの設定 描画アドレスの読み出しの指示 マスク・レジスタ値の設定
映像メモリー制御 WRITE READ DMAW DMAR	パラメータの映像メモリーへの書き込み準備 映像メモリー・データの読み出しの指示 映像メモリーへの DMA 転送開始の指示 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DB₀₋₇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・ハンドシェイ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

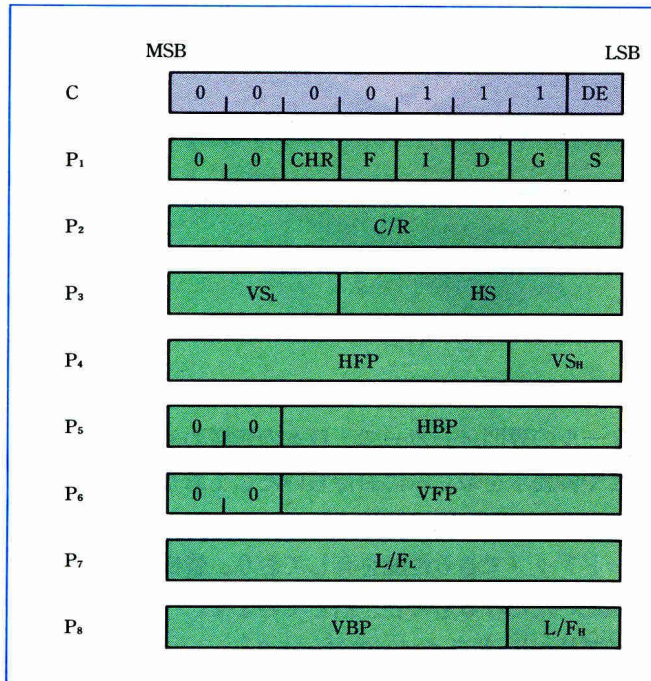


図9 SYNC コマンド。コマンド/パラメータは8ビット構成であり、コマンドは処理の実行や方向付けを行う。パラメータはレジスタ値の設定などに使われる

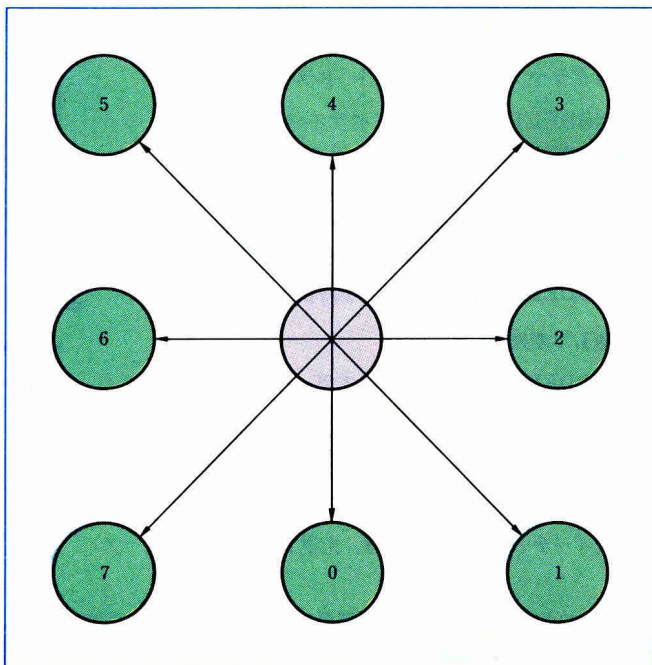


図10 描画方向の定義。45度単位で8通りの描画開始時基線方向を定義する。GDC 内での描画アドレス演算時、描画方向変化が検出されると一時的に方向値を変化させて、アドレス算出する

び表示データの修正法や AD_{0-15} , 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 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
dAD: ドット・アドレス
P: 水平方向総アドレス数

LR: 左方向回転
RR: 右方向回転

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

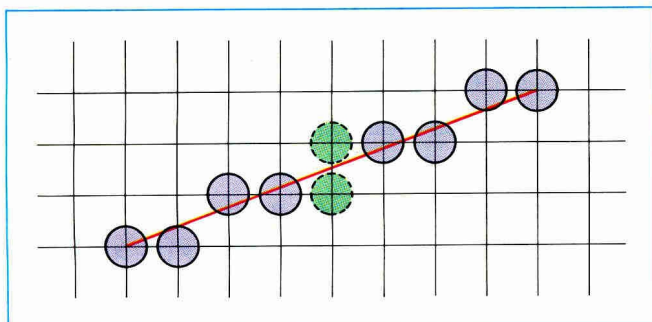


図 11 直線描画の例。X 方向に座標をドット単位で移動する。Y 座標小数部が四捨五入条件を満足すると、Y 座標を増加する。描画方向を 180 度変更すると、境界点 (● のドット) で Y 座標位置の再現性がないが、ソフトウェアによって回避できる

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。

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一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

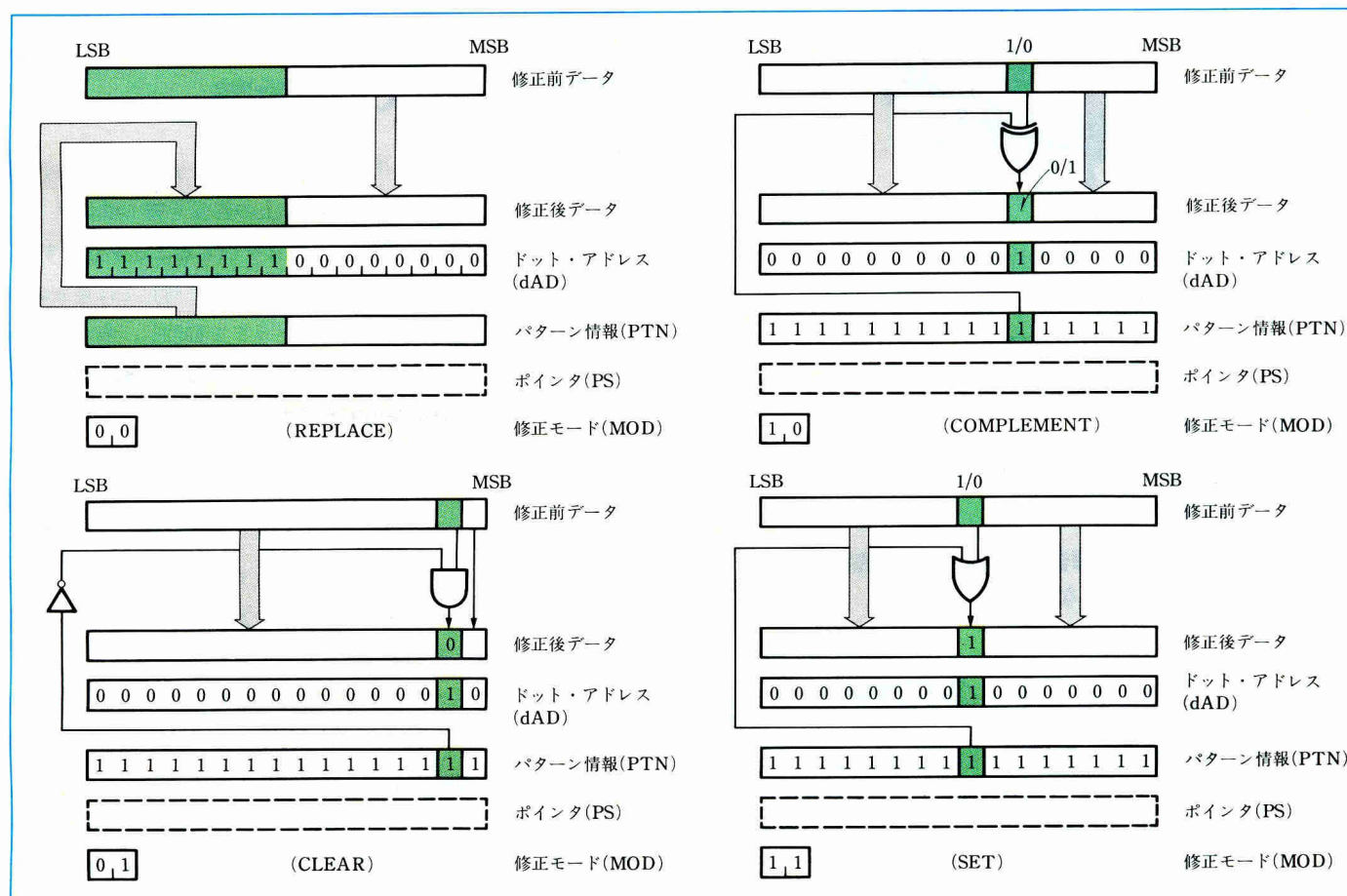


図 12 文字モード時のドット修正法。ドット・アドレス・レジスタ dAD が“1”のビットに対応するデータにのみ修正を加える。ポインタの機能は無視される

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用している。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

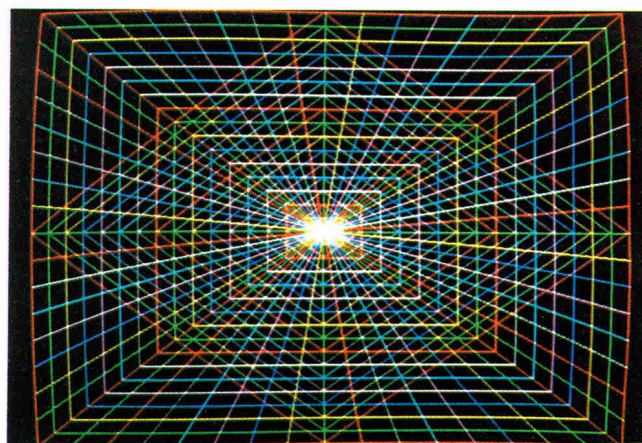


図 13 白を含む 7 色で直線, 四辺形を描画した例

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。

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

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。

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

$$2 * N * \Delta Y - (2 * I + 1) * \Delta X \geq 0 \quad (2)$$

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,

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

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

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

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,

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

最大描画ドット数は,

$$\Delta X \quad (d)$$

で求められる。

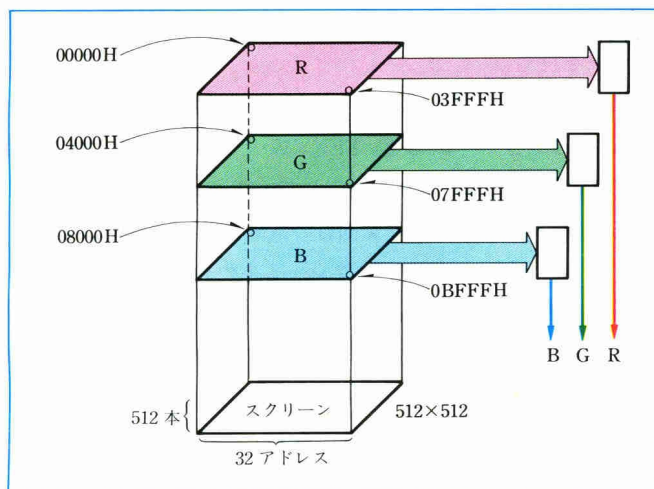


図 14 512×512 ドットの 7 色グラフィック表示時の映像メモリー・プレーン構成例。表示時には 0H から 3FFFH のアドレスを与え、R、G、B の 3 枚のプレーンから同時に映像出力を得る

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で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ル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

DIR	直線	円弧	グラフィックス文字	傾斜グラフィックス文字	四辺形	DMA
0						
1						
2						
3						
4						
5						
6						
7						

● : 描画始点 → : 描画方向 ■ (緑) : 定義域 ■ (紫) : 描画域

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	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 \uparrow : 切り上げ
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	—	—	—	—	—	

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	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	DF	01
	VECTE	6C												

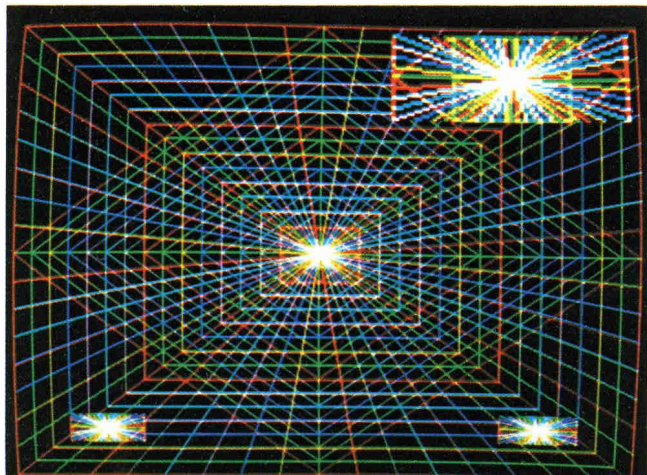


図 16 図 13 の一部をメモリー-メモリー間 DMA 転送し、拡大描画した例

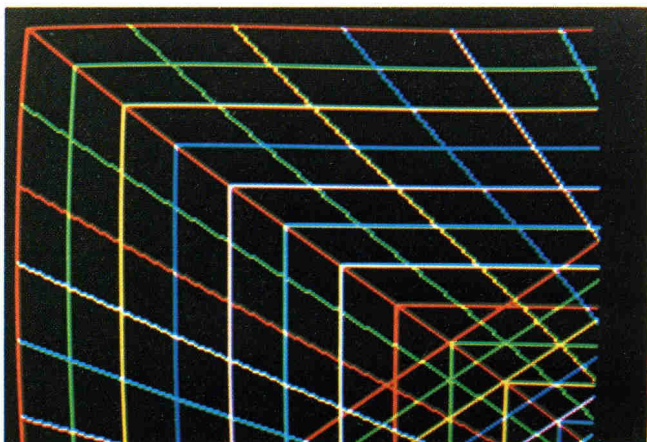


図 17 図 13 の一部の 3 倍拡大表示例

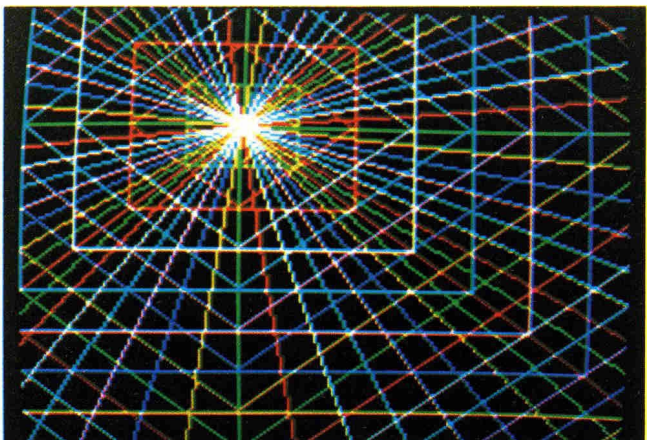


図 18 パニングにより図 13 の中央部を表示した例

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・プロセッ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・メモリの制御用、ファクシミリ受信用バッファ・メモリ制御用など、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, 鶴野「グラフィック制御用LSI」, 『NEC技報』, pp. 58-63, 1981年6月. ●