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.

第8章

グラフィック・ディスプレイ・コントローラ

グラフィック・ディスプレイ・コントローラは，文字表示制御だけでなく，直線，円弧などを高速に描画する機能をもっています．したがって，文字表示制御用コントローラCRTCとは，機能的に別物と考えてよさそうです．本章では，パソコン用のグラフィック表示ボードを作り，塗りつぶしなどのソフトウェアを紹介します．

小口　哲司
南野　宏樹
樋口三左男

CRT制御用LSIの比較

マイクロプロセッサやCRT制御用LSIの出現によって，CRT端末装置はそれまで使用されていたテレタイプライタに代わり，5，6年前から急速に普及しました．当初は，簡単な文字制御表示だけを目的とするCRTCが主でしたが，最近では文字単位制御だけではなく，従来TTLを数百個も使用しなければ実現できなかったドット単位での複雑な表示/描画制御を実行できる，グラフィック制御用LSIが発売されるようになりました．

CRTCは大まかに分けてつぎの3種類の用途・機能別分類ができます．

①文字表示用（文字制御の延長であるセミグラフィックも含む）
②ゲーム/アニメーション用
③グラフィックス表示/描画用

さらに，主記憶と映像記憶部との位置関係，すなわち，CRTCをシステムに組み込んだときのシステム構成によって分類すると，つぎの3種類があります．

①DMA転送・行バッファ型
②ビデオRAM型
③独立メモリ型

代表的なCRTCを，上記分類にしたがって**表1**にまとめてみました．グラフィックス制御では文字制御と比較して，同一の情報を伝達するために数10倍もの映像データ・アクセスを必要とします．したがって，CRTCが直接，映像データを取り扱えるような独立メモリ構成でないと，システム・パフォーマンスを向上できないことが傾向としてつかめます．

DMA転送型は，現在行表示用とDMA転送データ蓄積用の2系統の行バッファをもち，1行表示終了ごとにその用途を反転しています．1行分のバッファしか内蔵していないため，1行表示期間内に，次行に表示すべき1行分の表示データの転送を終了する必要があります．グラフィック制御時には，1ライン表示期間内に1ライン分のデータ転送を終了しなければならず，グラフィック用として，このシステム構成は適しません．

ビデオRAM型は，CPUとCRTCが同一の映像メモリを共有するため，CPUが映像メモリをアクセスする(描画)タイミングとCRTCからアドレスを供給し表示するタイミングとを制御するバス・アービタを必要とします．また，CPUに対して長期間にわたってWAITをかけたくない場合には，高速なスタティックRAMを使用して，描画サイクルをスチールする必要があります[6]．ただし，このような回路を付加したとしても性能の向上を大きく望めません．16ビットCPUも含め汎用CPUは，当然のことながら，グラフィック制御専用として設計されていないからです．直線描画において，1ドットを描画するために，以下のような煩雑な処理を実行しなければなりません．

①描画すべきワード・アドレスおよびドット位置を算出する
②描画すべきアドレスの映像メモリ・データを読み出す
③映像メモリ・データとドット位置とを比較し，そのドット位置に対して論理演算（修正）を加える
④描画すべきアドレスを発生し，修正済みデータを書き込む

単に，アキュムレータの内容をメモリに書き込むだけでも2μs（4MHz Z80の場合では4μs）以上を費やします．したがって，上記①～④

<表1>　代表的CRTCの分類

用途／機能別分類	品　名	システム構成
文字表示用（6847はセミグラフィック機能内蔵）	インテル　8275／8276	DMA転送・行バッファ
	日本電気　μPD3301	DMA転送・行バッファ
	日　立　HD46505	ビデオRAM
	三　菱　6847	ビデオRAM
	S M C　9007	DMA／ビデオRAM両用
ゲーム用	T　I　9918	独立メモリ
	日本電気　μPD777	シングル・チップ・マイコン
グラフィックス用	トムソン　EF9365	独立メモリ
	日本電気　μPD7220	独立メモリ

を処理し1ドットを描画するとき，高速描画が可能な場合を想定しても，20μs 程度を要します．

この値は，1水平走査期間内に，2～3ドットしか描画できないことを意味します．描画専用LSIを使用すると，連続的に（例えば800nsごとに）描画を続行できます．

▶ゲーム用LSI

ゲーム用LSIは，直線/円弧などを描画する機能はありませんが，表示セルの表示開始タイミングを1ドットの細かい刻みで制御し，映像メモリ内容の書き換えを行わずに，表示セルを上下左右にスムースに移動させることができます．すなわち，表示開始座標や表示セル名を変更するだけで，表示セルの移動や変更を瞬時にして行えます[7]．

ただし，操作性や融通性に難点があり，1走査線内における表示セル個数の上限など使用制限が厳しいことから，汎用性に欠ける面が多分にあります．グラフィック制御用LSIで高速な再描画により移動物体を形成し，同程度以上の機能を得ることは容易です．この場合には，上記使用制限はありません．

▶グラフィック用LSI

機能比較図中の項目のほかに，μPD7220にありEF9365にはない主要機能として，以下を掲げることができます．

⑧FIFO内蔵　⑨双方向DMA転送　⑩映像メモリ内容のREAD/WRITE　⑪スクロール　⑫画面分割　⑬描画タイミング選択　⑭外部同期による並列動作　⑮文字表示機能

グラフィック・ディスプレイ・コントローラ（GDC）μPD7220は，文字表示制御だけではなく，直線，円弧などを高速に描画するグラフィック表示/描画機能を内蔵した周辺装置制御用LSIです．数々のシステム的，性能的に優れた特徴があり[1][2]，従来より使用されている文字表示制御用コントローラ(CRTC)とは，機能的に別物と考えてよいでしょう．

なお，動作スピード区分として，3種が用意されています（表4）．

<表2>　ゲーム用LSIの比較

	TI　9918	日電　μPD777，778
① 解像度	256×192 ドット	75×60 ドット
② 移動表示セルの大きさ	8×8，16×16 ドット	7×7 ドット
③ 移動セル使用個数 (a) 画面内 (b) 一定走線内 (c) 同一座標内	 32個 4個 2個	 25個 11個 5個
④ 表示優先順位属性	有	有
⑤ 繰り返し表示属性	無	有(X, Y, XY 方向)
⑥ ライン・バッファ	無	有(12×6ビット2系統)
⑦ カラー	6色＋白＋黒	6色＋オレンジ＋ブルーシアン
⑧ 背景	2色ビット・マップ・イメージ	単色（全背景同一色）
⑨ 映像出力	複合映像信号（NTSC）	VIDEO, R-Y, B-Y, CROMA. モジュレータ外付け(NTSC)

<表3>　グラフィック用LSIの比較

	トムソン　EF9365	日電　μPD7220
① 解像度	32K×8ビット 256×256，512×512ドット	256K×16ビット 水平方向32ドット単位で プログラマブル．垂直は実装メモリ容量に依存する．
② 描画種類	直線，グラフィクス文字	直線，グラフィクス文字， 四辺形，円，弧，塗りつぶし
③ ドット修正機能	なし（外付回路による）	REPLACE,COMPLEMENT, SET, CLEAR の4種内蔵．
④ 描画速度	1.1μs/ドット	0.8μs/ドット（5 MHz 時）
⑤ 直線線種，グラフィックス文字ドット情報	線種：4種に固定．ドット情報：96文字分ROMにより固定．	16ビット線種パターン，8×8ドット情報ともにプログラマブル
⑥ 同期信号発生	PAL方式に固定．	プログラマブル．(インターレース可)
⑦ 拡大機能	文字描画時のみ1～16倍	文字描画時および表示時1～16の整数倍

<表4>　μPD7220のスピード区分

品　名	f max(MHz)
μPD7220	4
μPD7220-1	5
μPD7220-2	5.5

▒グラフィック・ディスプレイ・コントローラμPD7220の機能とハードウェア

●μPD7220機能の概要

μPD7220にはつぎの3種類の表示/描画動作モードがあります．文字表示からグラフィック表示描画まで，また，大容量メモリ制御用としても使用できます．

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

表5に，μPD7220の特徴をまとめます．

また，図1に端子接続図，図2に内部ブロック図，表6に端子機能を示します．

●拡張ボードのハードウェア

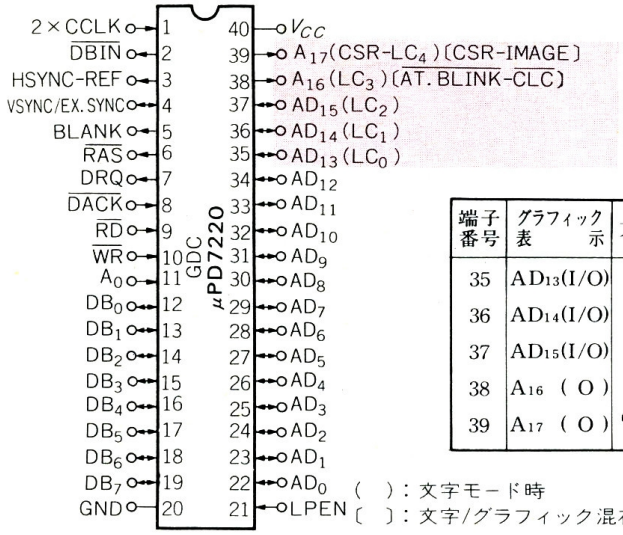
μPD7220を使用した表示装置では，従来型CRTCのように，大規模な記憶容量をもつ映像メモリを，主記憶上にマッピングする必要はありません．すなわち，映像メモリのアドレス/データ・バスはCPUと分離されたシステム構成（グラフィック表示サブシステム）であるため（図3），CPUとのインターフェース回路が簡略化できるからです．

ソフトウェア作成の見地からは，

＜表5＞　μPD7220の機能

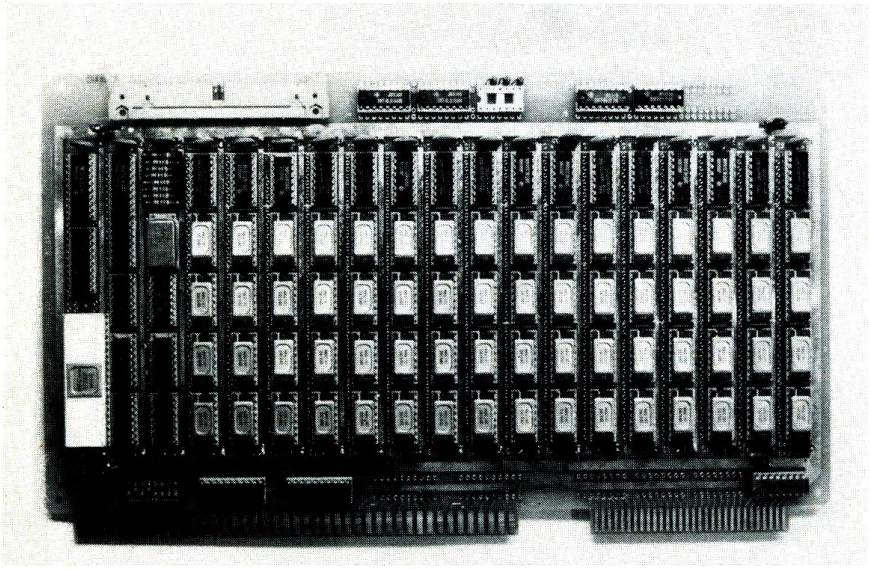
①直線，四辺形，円，弧，グラフィクス文字，短形領域の塗りつぶしを800ns／ドット（5MHz時）で高速描画する．
②グラフィック・モード時，256K×16ビット（4Mビット）の大容量メモリの直接制御
③映像メモリを主記憶から分離し，映像処理系をブラック・ボックス化したシステム構成
④CPUインターフェース部に，CPU－映像処理系間の処理速度調整用16バイトFIFOバッファ内蔵
⑤CPU側における描画前処理と同時並行して描画する処理形態
⑥主記憶－映像メモリ間の双方向DMA転送を4クロック／バイトの高速で実行
⑦映像メモリ内容に対するドット単位修正機能を4種内蔵
⑧映像メモリをダイナミックRAM（DRAM）で構成したときの容易なインターフェース
⑨1から16の整数倍の拡大表示，拡大描画機能
⑩縦横斜め全方向へのスクロール（パニング）制御機能
⑪映像メモリ領域と表示領域との独立した領域設定機能
⑫外部同期機能を用いた多数個並列動作
⑬プログラマブルな同期信号発生．インターレース走査可能
⑭直線，破線などの線種，グラフィクス文字描画時のドット構成はプログラマブル
⑮描画タイミング制御（フラッシュレスとフラッシュ描画）の選択可
⑯ビット・マスク操作による可変ドット位置に対する連続描画
⑰表示速度に対するクロック周波数の3段階の選択可能
⑱高級な文字表示制御機能

＜図1＞　μPD7220の端子接続図



端子番号	グラフィック表示	文字表示	文字/グラフィック混在表示
35	AD13(I/O)	LC0(O)	AD13 (I/O)
36	AD14(I/O)	LC1(O)	AD14 (I/O)
37	AD15(I/O)	LC2(O)	AD15 (I/O)
38	A16 (O)	LC3(O)	AT.BLINK-CLC (O)
39	A17 (O)	CSR-LC4 (O)	CSR-IMAGE (O)

（　）：文字モード時
〔　〕：文字/グラフィック混在モード時



＜写真1＞　グラフィック表示拡張ボード

各ユーティリティ・ソフトウェアを主記憶の用途別使用制限（ユーザーズ・エリアの制約など）に無関係に作成できます．したがって，汎用性，蓄積継続性，拡張性の高いソフトウェア作成が可能となります．

のちほど製作するグラフィック表示拡張ボードは，グラフィック表示/描画の基本機能だけを最小構成の外付け回路により実現することを目的としています．したがって，文字表示機能やDMA転送機能などは付加していません．ただし，μPD7220 1個で直接制御できる最大値である，256KWの映像メモリを64K DRAM（4164D-3）64個によってフル実装しています（**図4，写真1**）．

外観上，単なるメモリ拡張ボードのように見えますが，このボードにCPUのシステム・バスとカラーCRTを接続するだけで，カラー・グラフィック機能拡張ボードとして動作します．16ビットCPUを使用した最新のビデオRAM型グラフィック用ボードが文献(8)に紹介されています．これは映像メモリ容量が1/4以下であるにもかかわらず，非常に多くの外付け回路を必要としています．

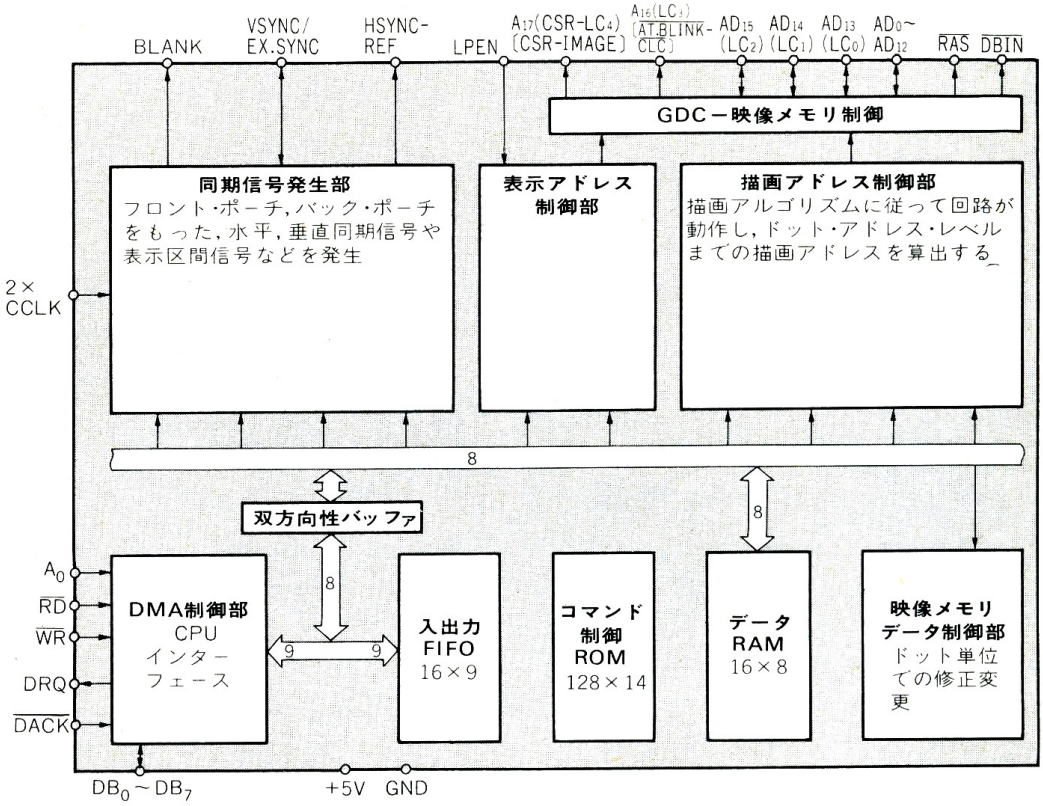
また，日本楽器社のホーム・コンピュータYISには，グラフィック制御用として，本ボードと同一形状のボード2枚に，ゲート・アレイ，PAL（プログラマブル・アレイ・ロジック）を含むTTLがぎっしり詰め込まれています．

▶ハードウェア回路の作成

このボードは，インテル社が提唱するシステム・バス(マルチバス)に直接接続できる各種システム制御用ボード（インテルSBCなど）と同一寸法でつくりました．システム・バスは50ピン・フラット・ケーブル・コネクタを介して接続しています．μPD7220をI/O装置として定義しているため，各8本のアドレス/データ・バスと2本のI/O制御線を接続するだけです．したがって，ホスト・マシンで使用するCPUはど

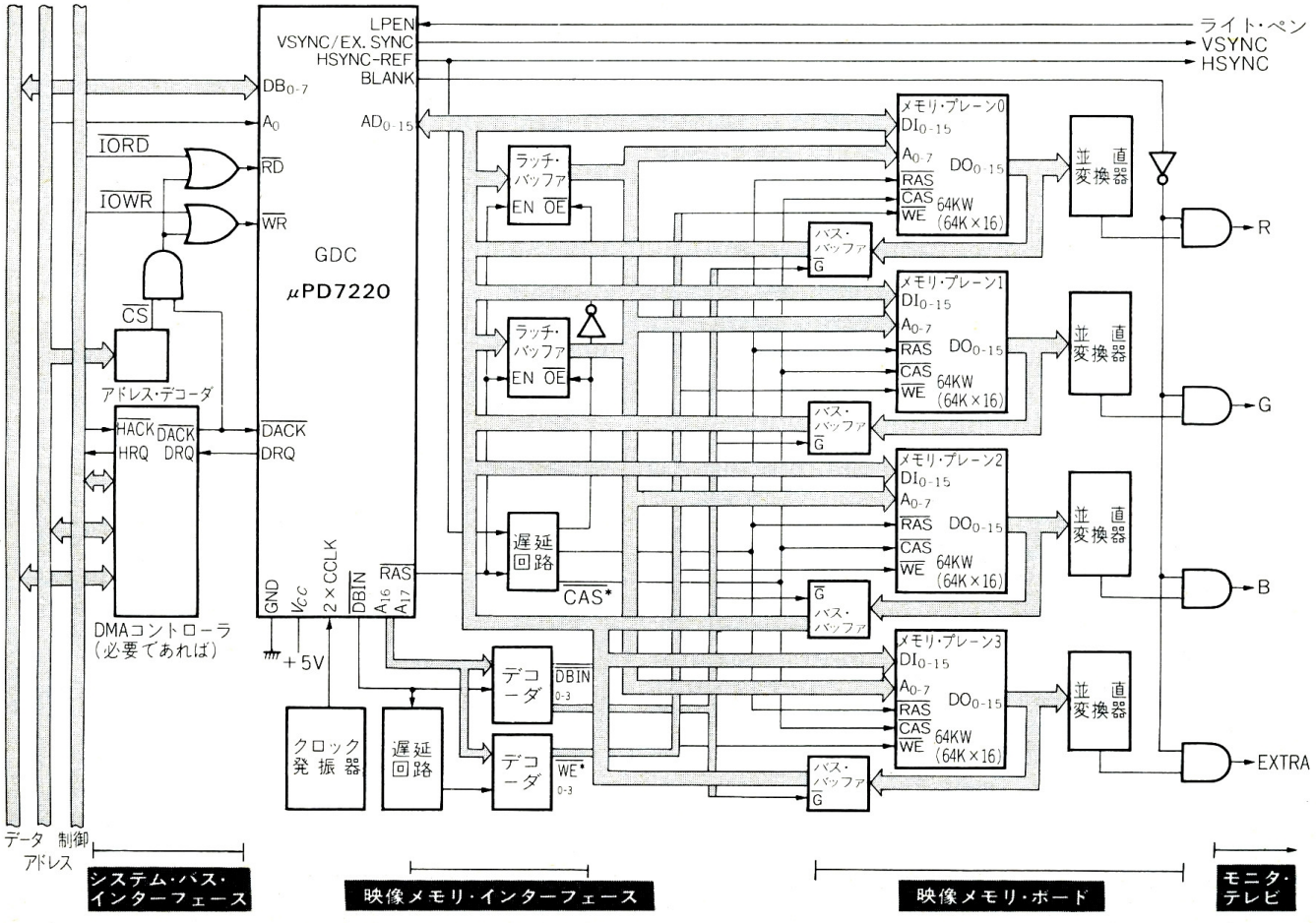
ういうものでもかまいません．既存のパーソナル・コンピュータはすべて接続できるようになっています．後述する塗りつぶしのソフトウェアを作るときには，ラジオシャック社のTRS-80をホスト・マシンとして使用しました．

本回路では，1水平走査時間が31.5kHzの高解像度カラー・モニタTVを使用し，512×512ドット・ノンインタレースと1024×1024ドット・インタレースによるグラフィッ



＜図2＞
μPD7220の内部ブロック図

＜図3＞　グラフィック・ディスプレイ・システム構成図



＜表6＞　μPD7220の端子機能

電源・グラウンド端子	V_{CC}	＋5Vの電源を接続
	GND	0Vの接地用端子
クロック端子	2×CCLK （2×Character Clock）	単相クロックを供給．μPD7220内部において，この単相クロックと同一周波数の2相クロックを作成し，その2相クロックに同期して内部回路が動作する． クロックの周波数は，水平方向の表示文字数（表示ドット数）と水平表示時間との関係で決まる．
システム・バス接続用端子	DB_0～DB_7 （CPU Data Bus 0～7）	双方向性のデータ・バスで，8ビットまたは16ビットの標準的なマイクロプロセッサに接続．
	$\overline{RD}$ （CPU Read Strobe）	CPUがμPD7220からデータまたはステータス・フラグを読み出すとき，低レベル信号を供給する．
	$\overline{WR}$ （CPU Write Strobe）	CPUがμPD7220にコマンドまたはパラメータを書き込むとき，低レベル信号を供給．
	A_0 （CPU Address Bus 0）	通常，CPUのアドレス・ライン最下位ビットが接続され，$\overline{RD}$，$\overline{WR}$信号との組み合わせにより，以下のようにデータ・バス信号の種類を選択する． A_0 / $\overline{RD}$ / $\overline{WR}$ / 機能 0 / 0 / 1 / ステータス・フラグの読み出し 1 / 0 / 1 / データ(μPD7220内FIFO)の読み出し 0 / 1 / 0 / パラメータの書き込み 1 / 1 / 0 / コマンドの書き込み μPD7220はCS（チップ・セレクト）端子をもっていないので，外部回路によってCSおよび$\overline{DACK}$の信号を$\overline{RD}$および$\overline{WR}$信号に組み合わす必要がある．
DMA制御用端子	DRQ （DMA Request）	DMA要求出力で，DMAコントローラのDRQ入力に接続．DMARまたはDMAWコマンドの実行によって活性化される．
	$\overline{DACK}$ （DMA Acknowledge）	DMA転送中であることを示す入力で，DMAコントローラ$\overline{DACK}$出力が接続される．転送バイト数のカウントや転送開始のタイミング指示用として使用する．
映像メモリ制御用端子	AD_0～AD_{15} （Display Memory Address/Data Bus 0～15）	双方向性のアドレス／データ・バスで，映像メモリに接続．アドレスとデータが時分割されており，$\overline{RAS}$出力の高レベル期間にアドレスを外部回路にラッチする． リフレッシュ時，リフレッシュ・アドレスは下位8ビットに出力される．
	A_{16}，A_{17} （Display Memory Address 16，17）	グラフィック表示モード時，アドレス上位2ビットが出力され，映像メモリに接続．フローティング出力状態は生じない．
	RAS （Row Address Strobe）	以下の3種の用途がある． ① アドレス多重化を行っている16ピン・ダイナミックRAMに対する$\overline{RAS}$，$\overline{CAS}$信号の基本タイミング信号． ② 高レベル時，AD_0～AD_{15}に対するアドレス・ラッチ・タイミング信号． ③ 映像メモリのアドレス・サイクルの弁別用タイミング信号． より具体的には，映像メモリから並列に読み出された映像を直列信号に変換するレジスタの，LOADクロック発生タイミング選択用として使用する．
	$\overline{DBIN}$ （Data Bus In）	映像メモリに対するRead/Modify/Write（R/M/W）実行時にのみ出力され，映像メモリの出力を映像メモリのデータ・バスに乗せるために使用する．この信号を外部回路によって遅延させ，映像メモリに対する書き込みタイミング信号（ME）としても使用する．
同期信号発生用端子	HSYNC-REF （Horizontal Sync-Refresh Timing）	水平同期信号としてモニタ・テレビに接続．ダイナミックRAM制御モードを選択した場合には，リフレッシュ・アドレスがAD_0～AD_7に出力されていることを示すタイミング信号としても使用する．
	VSYNC （Vertical SYNC）	μPD7220がマスタ動作時に，垂直同期信号としてモニタ・テレビに接続．
	EX，SYNC （External SYNC）	μPD7220がスレーブ動作時に，外部同期入力端子として使用する． 入力信号の“H”→“L”への変化時に，μPD7220内部の同期信号発生回路が初期化される．
表示制御用端子	BLANK （Blanking Signal）	表示消去信号として使用．以下の状態のときに出力される． ① 水平帰線または垂直帰線区間 ② RESET，STOPなどの表示停止コマンド実行時からSTARTコマンド実行時までの区間 ③ 映像メモリに対するR/M/W実行区間
	LPEN （Light-Pen Strobe）	ライトペンが光入力を検出したとき高レベルの信号を入力する．そのときにμPD7220内にラッチされた表示アドレスをLPENコマンドによって，CPUは読み出すことができる．
	LC_0～LC_3 （Line Count 0～3）	文字モード時，μPD7220に内蔵されている5ビットのライン・カウンタのうち下位4ビットが出力される．文字発生用ROMのアドレスに接続する．
	CSR-LC4 （Cursor-Line Count 4）	文字モード時，カーサ表示出力と内蔵ライン・カウンタの最上位ビットを時分割して出力する．表示期間中にはカーサ表示信号を出力．LC_4の信号はBLANK出力信号の立ち下がり時に外部フリップフロップに記憶させる．
	$\overline{AT\cdot BLINK\text{-}CLC}$ （Attribute Blink-Clear Line Counter）	文字／グラフィック混在モード時，ブリンキングする文字属性に供給するブリンキング・タイミング信号と外部回路によって構成するライン・カウンタに対するクリア信号を時分割して出力する．表示期間中にはブリンキング・タイミング信号を出力する．外付けライン・カウンタはHSYNC-REF出力信号の立ち下がり時にカウントすると同時に，$\overline{CLC}$信号の抽出を行う．
	CSR-IMAGE （Cursor-Image）	文字／グラフィック混在モード時，カーサ表示出力とイメージ信号出力を時分割して出力．表示期間中にはカーサ表示信号を出力する．イメージ信号は，文字表示領域とイメージ表示領域の切り替えを外部表示制御回路に指示する目的の信号で，BLANK出力信号の立ち下がり時に，外部フリップフロップに記憶させる．

ク表示の2種をハードウェア上，切り替え可能にしてあります．この画面構成はソフトウェアによって任意値に変更可能です．

いずれの場合にも μPD7220には5MHzのクロックを供給します．**表5**⑰に示したように，表示速度から算出したクロック周波数が規格よりも高すぎる場合には算出値の1/2にでき(“DAD+2” 設定)，逆に低すぎる場合（256×256ドットなど低解像度への応用）には算出値の2倍の周波数のクロックを選択できるように（“IM”設定）設計されています．すなわち，低解像度から高解像度への種々の応用を想定して，表示速度と描画速度との比率を1対1，1対2，2対1に変化させて対応づけができます．後述する映像メモリのデータ・バス幅を可変とする方法を用いると，この比率をさらに大きく取ることができます．

▶制御信号のタイミング

本ボードでも，512×512ドット・ノンインタレース時には“IM”設定を行い，通常使用時におけるクロック周波数の算出値2.5MHzの倍の5MHzで動作させています．その結果，2.5MHzのクロックを供給した場合と比較して，2倍の描画速度が得られることになります．

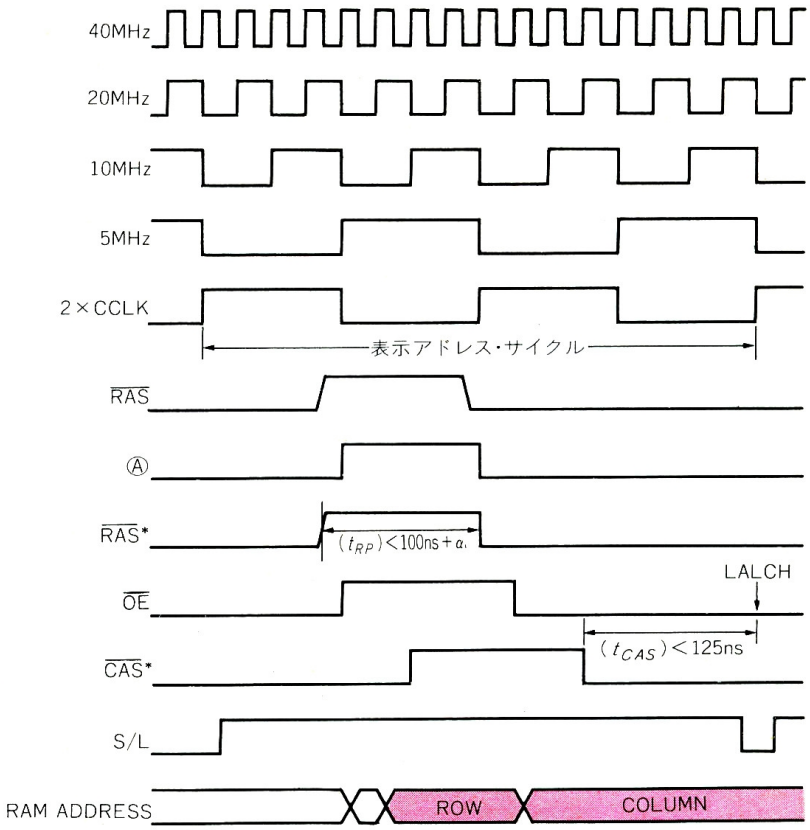
グラフィック・モード時は，18ビットのアドレス線と16ビットのデータ線を直接制御し，下位16ビットのアドレス/データは時分割多重化信号として入出力します．さらに，標準的な16ピンDRAMは，行/列アドレスに時分割されたアドレス信号を必要とします．このため，映像メモリ制御基本タイミング信号 $\overline{\text{RAS}}$ を出力し，この信号を基にして外付け遅延回路などにより，以下の制御信号を作ることができます．

①アドレス・ラッチ信号

②DRAMに供給する行/列アドレス選択信号（$\overline{\text{RAS}}$*，$\overline{\text{CAS}}$*）

③アドレス・ラッチの出力タイミング選択信号（$\overline{\text{OE}}$）

<図5> クロック，映像メモリ制御信号のタイミング・チャート



＊は外付け回路によって作成した信号

<表7> μPD4164のスピード区分，スイッチングの改善

	4164-2→4164-20		4164-3→4164-15	
t_{RAS} (ns)	200	200	150	150
t_{CAS} (ns)	135	100	100	75

④映像直列信号を発生する並列－直列変換レジスタの映像メモリ内容のロード・タイミング信号（S/L）

図5に，上記タイミングの相互関係を示します．

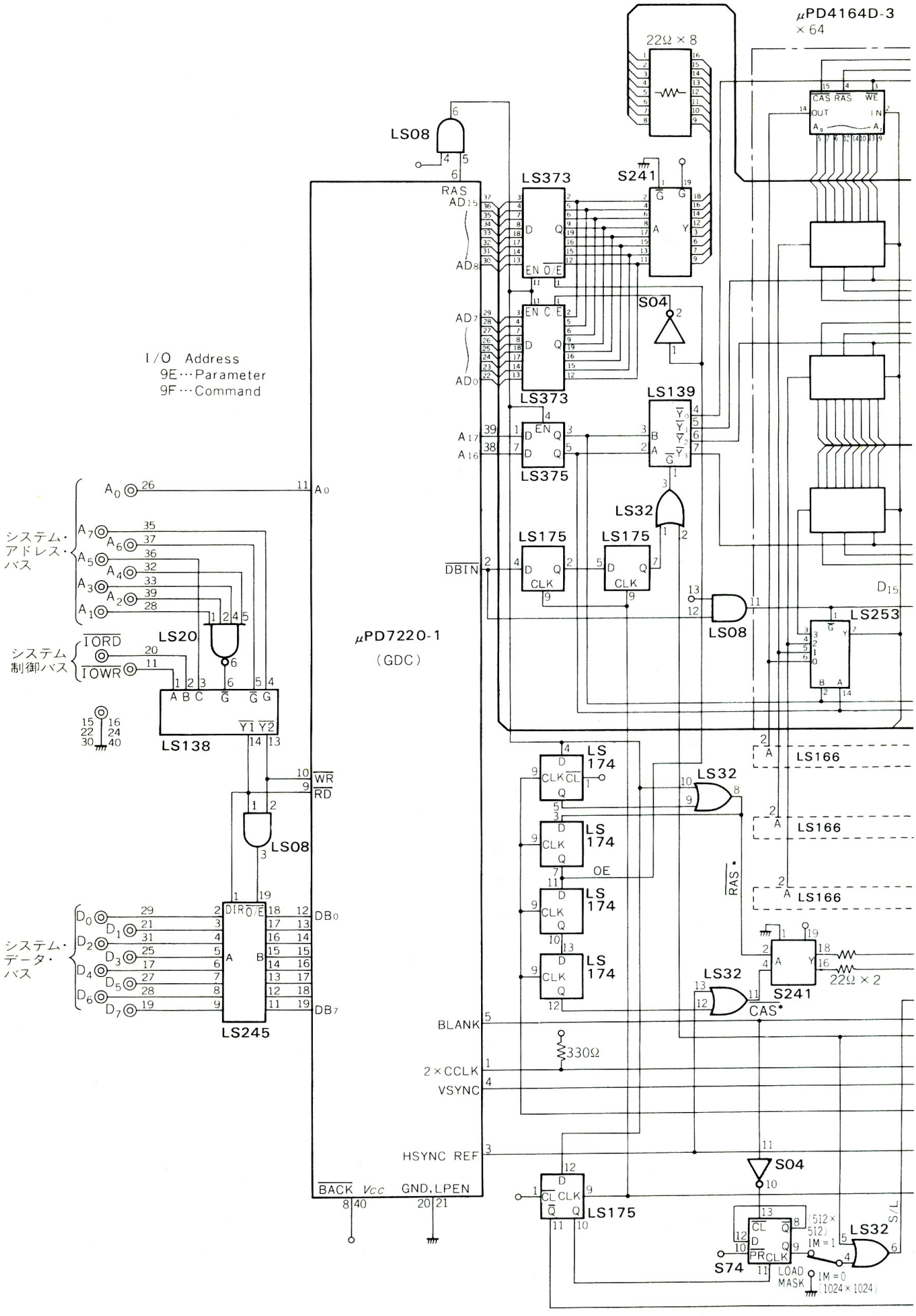
μPD7220のクロック周波数が高くなると，次第にDRAMに供給する $\overline{\text{RAS}}$*，$\overline{\text{CAS}}$* 信号の規格，とくに，$\overline{\text{RAS}}$* のプリチャージ幅（t_{RP}）と $\overline{\text{CAS}}$* 立ち下がりからのデータ出力遅延（t_{CAS}）が問題となってきます．μPD7220は半クロック幅の $\overline{\text{RAS}}$ 規準信号（5MHzで100ns幅）を出力するため，t_{RP} 規格を満足するように，外部ゲートによって信号幅を広げる外付け回路が必要となります．

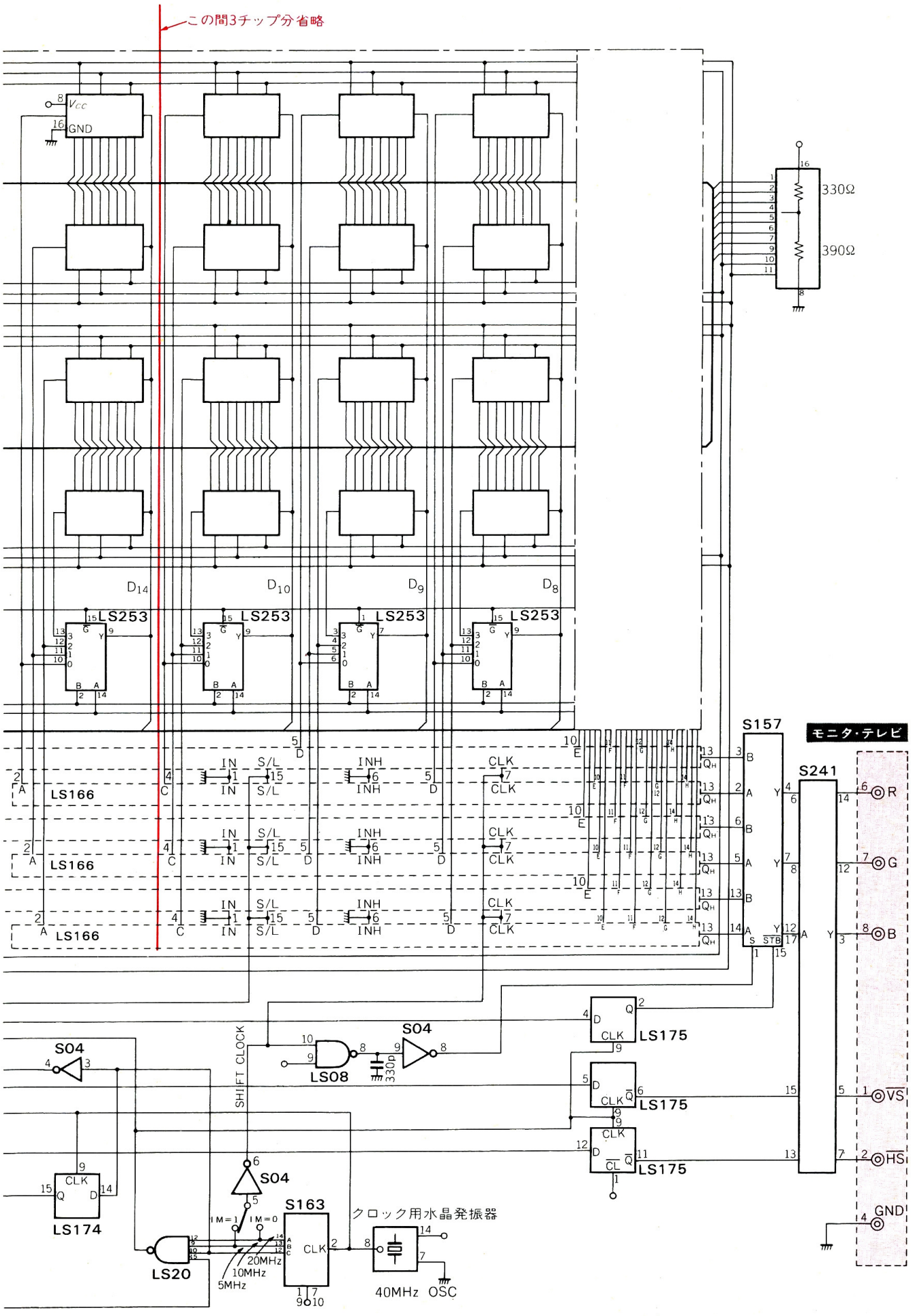
記憶素子のスピード区分表示に，意味合いをもったものも出されています．μPD4164の場合も $\overline{\text{RAS}}$ 立ち下がりからのデータ出力遅延 t_{RAS} を基に，スピード区分および電気的スペックが**表7**のように移行します．t_{CAS} 規格が大幅に改善されるため，本ボードへの応用においてもスイッチング余裕が生じ，外付け回路の削減が可能となるでしょう．

μPD7220は，CPUからあらかじめ与えられるコマンド/パラメータ(以下，単にC/Pと略す)に従い，表示アドレス（DAD），描画ワード・アドレス(EAD)，“DRAM”設定したときアドレス/データ出力の下位8ビットにHSYNC（水平同期）発生時に出力されるリフレッシュ・アドレス(RAD)の3種のアドレスを映像メモリに供給します．表示およびリフレッシュのアドレス・サイクルは2クロック，描画時は4クロックです（**図6**）．

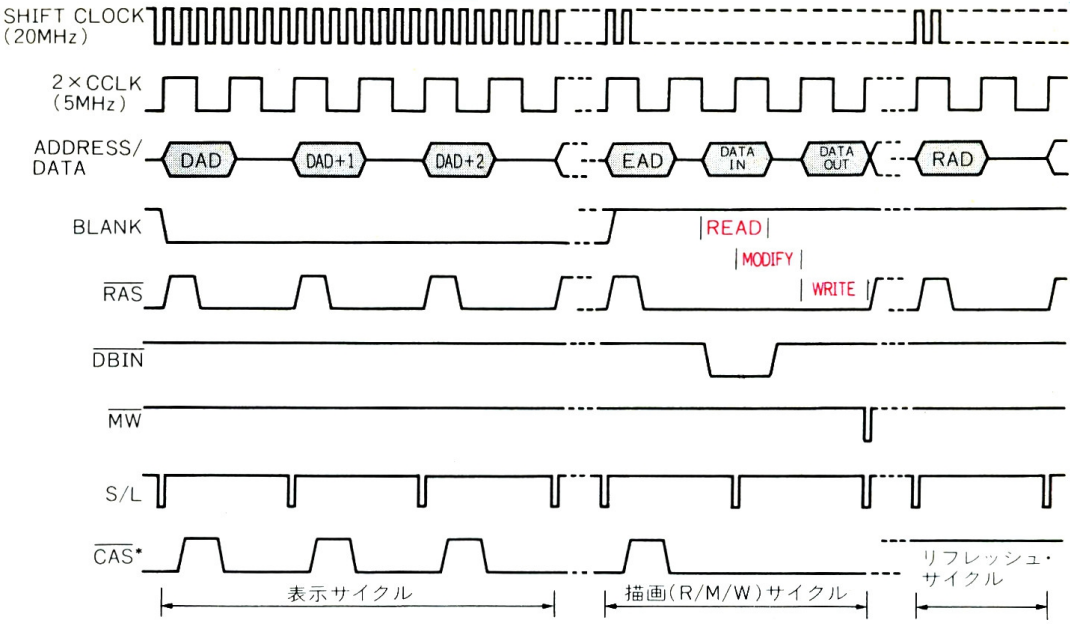
描画は，1ワード・アドレスに格

<図4> μPD7220（GDC）グラフィック表示用拡張ボード（256KW×16，1024×1024×4）回路図




この間3チップ分省略
V_{CC}
GND
330Ω
390Ω
D_{14}
D_{10}
D_9
D_8
LS253
LS166
IN
S/L
INH
CLK
S157
S241
モニタ・テレビ
R
G
B
VS
HS
GND
SHIFT CLOCK
S04
LS08
330p
LS175
LS174
LS20
S163
IM=1
IM=0
20MHz
10MHz
5MHz
クロック用水晶発振器
40MHz OSC

<図6>
映像メモリ・サイクル（IM="0"）

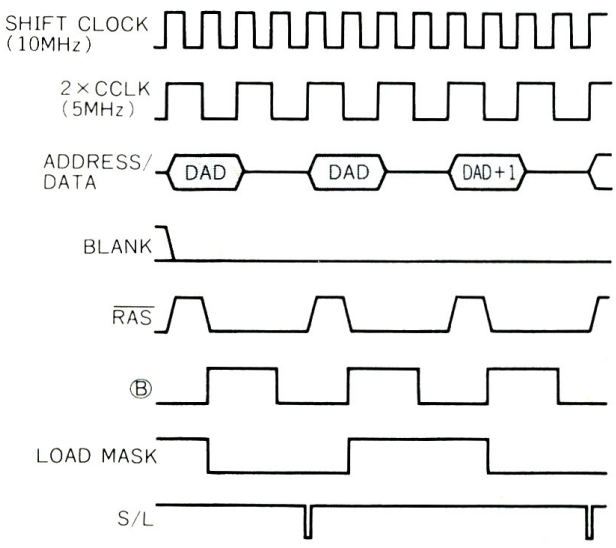


納されている映像メモリ内容16ビットのデータをμPD7220内に書き込み(READ)，ドット修正位置情報やパターン情報との比較修正（MODIFY)後，映像メモリに16ビット同時に再書き込み（WRITE)することによってなされます．書き込みタイミング信号（$\overline{MW}$）はμPD7220が出力するDBIN信号を遅延させて発生します．リフレッシュは，HSYNC信号によって$\overline{CAS}$*信号を高レベルに保持し，RASオンリ・リフレッシュを実行します．

本ボードでは並列－直列交換シフトレジスタのシフト・クロックの周波数を，ドット・クロックの1/2とするように回路を構成しています．1024×1024ドット表示の場合，通常の回路構成であると約40MHzのシフト・クロックが必要となり，LS166の最大動作周波数を超えてしまうからです．さらに，映像メモリ内容読み出し時のメモリ・プレーン選択は，LS253を使用することにより簡易化しています．

図7に，“IM”設定をしたときの表示アドレスの変化を示します．“IM”設定をしない場合（**図6**）にくらべ，μPD7220は同一の表示アドレスを連続した2表示アドレス・サイクルにわたって出力します．並列－直列変換レジスタのロード・クロックの発生周期を2倍とし，そのシフト・クロック周波数を1/2とすることにより，相対的に，表示速度を描画速度の1/2に落とす（逆説的には，描画速度を表示速度の2倍にする）ことができます．“IM”設定や拡大表示設定をした場合でも，描画やリフレッシュ時のアドレス・サイクルには変化はありません．

<図7>
映像メモリ・サイクル（IM="1"）



●ソフトウェアの概要

μPD7220はCPUが作成したC/Pを受け取り解釈します．以後，CPU側処理とは無関係に描画などの機能を達成します．

μPD7220は**21種の8ビット構成コマンド**（**表8**），およびそれに付随するパラメータを解釈でき，各々のコマンドは動作制御，表示制御，描画制御，映像メモリ制御の4種に大別できます．

CPUがμPD7220に送出するC/Pは，いったん，μPD7220が内蔵する16バイトのFIFOに格納されます．このFIFOは，μPD7220が映像メモリに対する描画を実行中であるときなどには，μPD7220の内部回路への読み出し動作を停止します．

このFIFOを内蔵したことにより，CPUはμPD7220の描画動作と並行してC/P作成処理などを実行し，表示/描画のタイミングとは無関係に，いつでもC/PをμPD7220に転送し，FIFOに蓄積できます．したがって，ソフトウェア作成上，割り込み信号などによって表示/描画タイミングを把握しておく負担はあり

＜表8＞　μPD7220のコマンド

動作制御

コマンド	動　作　内　容
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

コマンド	動　作　内　容
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	文字表示時のカーサ形状等の設定
PITCH	映像メモリ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指示

描画制御

コマンド	動　作　内　容
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

映像メモリ制御

コマンド	動　作　内　容
WRITE	パラメータの映像メモリへの書き込み準備
READ	映像メモリ・データの読み出しの指示
DMAW	映像メモリへのDMA転送開始の指示
DMAR	映像メモリからのDMA転送開始の指示

ません．

CPUはFIFOの状態をステータス・フラグ（表9）によりチェックすることができます．FIFO関係のステータス・フラグとして，“FIFO-FULL”と“FIFO-EMPTY”の2種が用意されています．

C/P送出の際のフラグ・チェックの方法としてつぎの2種があります．

①FIFO-FULL＝“0”であることを確認して送出する

②FIFO-EMPTY＝“1”であることを確認して，16バイトまで連続転送する

また，CPUはμPD7220を経由して映像メモリ内容などを読み出すことができます．この場合にも，読み出しデータは，いったん，FIFOに記憶されます．この場合のフラグ・チェックはつぎのようにします．

①DATA-READY＝“1”であることを確認して読み出す

②FIFO-FULL＝“1”であることを確認して，16バイトを連続して読み出す

しかし，μPD7220にC/Pを送出したり，データを読み出すたびにステータス・フラグをいちいちチェックすることは，システム性能を劣化させることになりかねません．これを回避するための方策を，実例に基づきつつ以下に示します．

＜表9＞　ステータス・フラグ

フラグ名称	読み出し端子	機　能
DATA READY	DB_0	μPD7220がREADなどの読み出しコマンド実行後，読み出しデータが読み出し可能な状態になったことを示す
FIFO FULL	DB_1	FIFOがデータで満たされたことを示す
FIFO EMPTY	DB_2	FIFO内容が空白であることを示す
DRAWING	DB_3	描画中であることを示す
DMA EXECUTE	DB_4	DMA転送を続行中であることを示す
VERTICAL SYNC	DB_5	垂直同期信号（VSYNC）が発生していることを示す
HORIZONTAL BLANK	DB_6	水平消去信号（HBLANK）が発生していることを示す
LIGHT PEN DETECT	DB_7	ライト・ペン信号によるアドレスの検出がなされたことを示す

◉基本描画制御ソフトの具体例

LSIというハードウェアが供給されても，ソフトウェアがなければ，それを動作させることはできません．さらに，μPD7220の描画機能を十分に引き出すには，μPD7220の動作を熟知したうえで，ソフトウェアを

＜表10＞　コマンド/パラメータ(C/P)解釈速度

COMMAND	CMD.TRANSLATION	PARAM.TRANSLATION	COMMAND	CMD.TRANSLATION	PARAM.TRANSLATION
SYNC	6　(1.2)	2　(0.4)	TEXTE	16　(3.2)	
MASTER／SLAVE	12　(2.4)		CSRW	6　(1.2)	(*1)
START (6BH)	12　(2.4)		CSRR	14　(2.8)	
START (0DH)	6　(1.2)		MASK	10　(2.0)	2　(0.4)
STOP	6　(1.2)		WRITE：W	12　(2.4)	(*2)
ZOOM	10　(2.0)	2　(0.4)	WRITE：HB	12　(2.4)	8　(1.6)
SCROLL	10　(2.0)	4　(0.8)	WRITE：LB	14　(2.8)	8　(1.6)
CSRFORM	10　(2.0)	2　(0.4)	READ：W	14　(2.8)	
PITCH	10　(2.0)	2　(0.4)	READ：HB	12　(2.4)	
LPEN	12　(2.4)		READ：LB	14　(2.8)	
VECTW	10　(2.0)	2　(0.4)	DMAW	SAME AS WRITE COMMAND	
VECTE	18　(3.6)		DMAR	SAME AS READ COMMAND	
TEXTW	10　(2.0)	4　(0.8)			

UNIT：CLOKS　　(　) DENOTES μs at 5 MHz
(*1) P1，P2：2 (0.4)，P3：4～64 (0.8～13)　　(*2) P1：2 (0.4)，P2：4 (0.8)

作成しなければなりません．そのため，ハードウェア装置設計者自身が，IOCS（入出力制御ソフト)を設計するという能動的な傾向が強まってきつつあります．

μPD7220 初期化および映像メモリ・クリア・ルーチン(CLMEM)，直線(LINE)，四辺形(REC)，円弧(ARCCRL）の各描画ルーチンのリストを稿末に示します．

初期化ルーチンにおいて，μPD7220に与えるパラメータ値は各装置ごとに特定できるため，ROM 領域に格納しておきます．

RESET コマンド発行後，順次，所要のC/Pを μPD7220にブロック転送します．このとき，CPUはFIFO の状態をいちいちチェックする必要はありません．μPD7220 内におけるC/P解釈速度は，CPUのC/P転送速度と比較して，十分高速だからです（表10)．

前もって，実線/破線などの線種やReplace/Complement などのドット修正モードは μPD7220に設定済みであるものとして，直線などのサブルーチンは記述してあります．

直線サブルーチンでは，始点/終点の座標およびカラー情報，四辺形では，対角線上の2点の座標およびカラー情報，円弧では中心点座標と開始角/終了角およびカラー情報を各レジスタに設定したのち，サブルーチンをコールするだけで各種描画を実行します．

各描画サブルーチンでは，C/P送出開始時にFIFO-EMPTY＝“1”であることをチェックしたのちは，FIFO 状態検出をせず，無条件に16バイトを μPD7220 に送出しています．16バイトを超える場合には，FIFO-FULL＝“0” を検出しています．

描画アドレス（EAD）などのCPUへの読み出しの場合には，DATA-READY＝“1”であることを1回だけ検出したのち，連続してデータを読み出しています．SINや乗算フローも簡潔にまとめられており，約1Kバイトでサブルーチンを構成できます．

▒ソフトウェアの具体例

●任意閉曲線内の任意パターンによる高速万能型塗りつぶし

μPD7220は45度単位の傾きをもった四辺形内の塗りつぶしであれば，任意なパターンで高速に塗りつぶし描画を実行する機能を内蔵しています．バー・グラフ作成やCAD関係処理に対しては，この機能だけで十分です．しかし，パイ・チャート(円グラフ)作成や任意閉領域内塗りつぶしのように，より複雑な面に対する描画はμPD7220単体では不可能です．CPUによるソフトウェア介在をとくに頻繁に必要とします．

すでに，この種のソフトウェアは作成されていますが，描画速度の点で見劣りがします．μPD7220のハードウェア動作の特質を十分に生かし，ハードウェアにマッチした塗りつぶしユーティリティを新規に作成しましたので，以下に説明します．

▶塗りつぶし描画の実行

このプログラムは，Z80のアセンブリ言語で記述しており，実質的なプログラム容量は1.2Kバイトです(**リスト**参照)．

円内部を塗りつぶすだけであれば，あらかじめ用意されているテーブルを参照しながら，疑似的に塗りつぶす方法もありますが，任意閉領域に対する塗りつぶしとなると，この方法は用いることができません．さらに，この方法では，円描画や塗りつぶしが粗雑なものとなることは明白です．

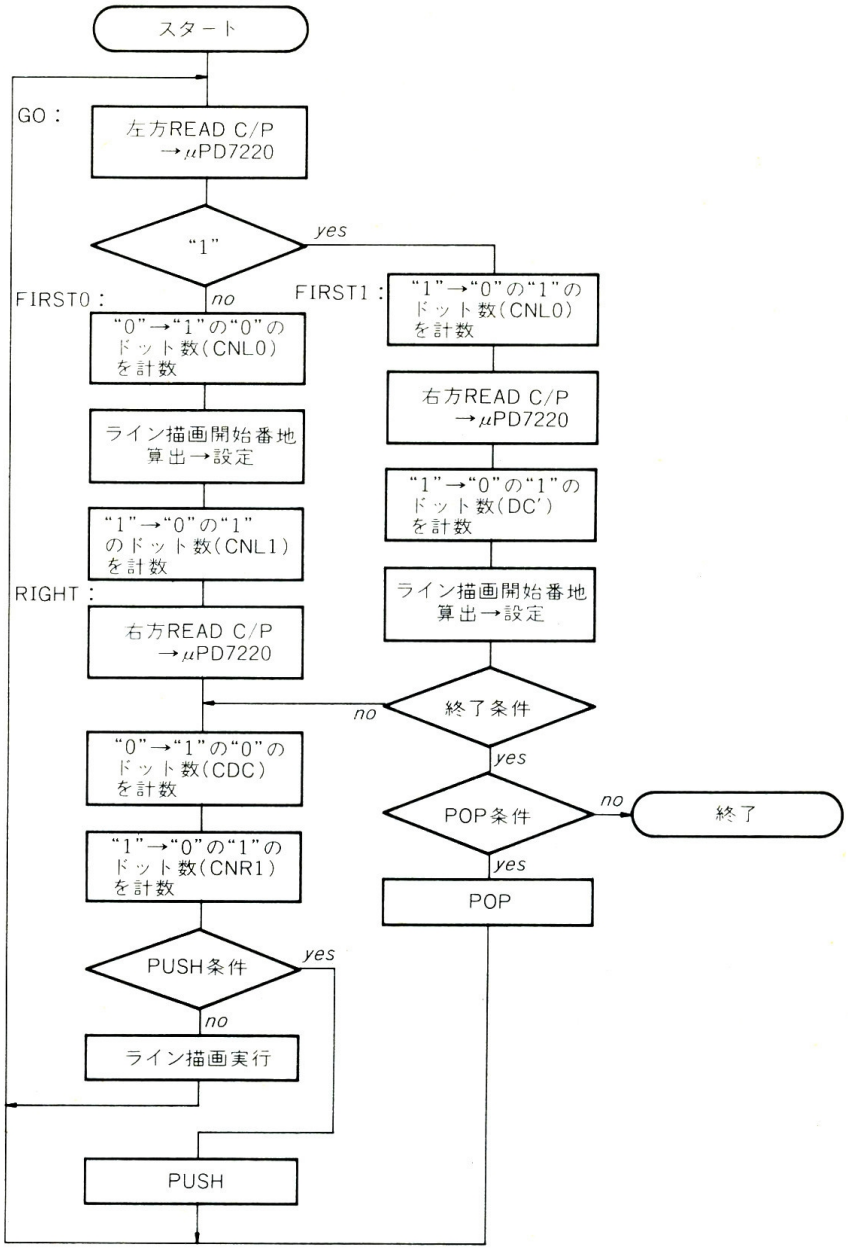
このプログラムでは，映像メモリ内容をμPD7220を経由してCPUが読み出し，境界点を検出後，直線描画を繰り返し実行することによって塗りつぶしを行っています(**図8**)．

実際には，より精細な処理が必要となりますが，これらについては，そのつど記述します．**図9**に実行評価結果を示します．

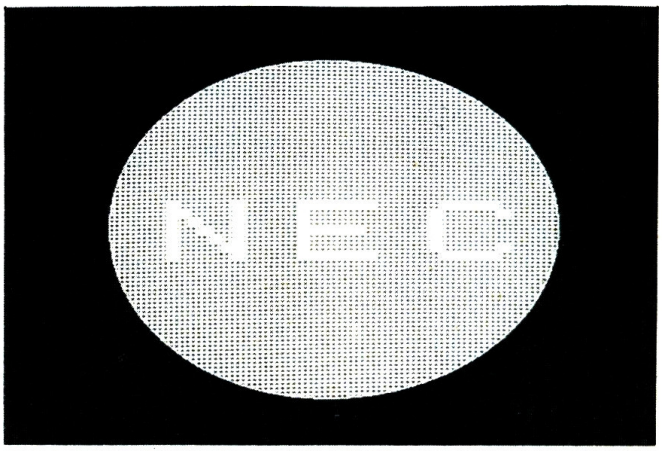
写真2は第⑤例の実行結果です．③例にある程度の塗りつぶしドット数をもつ領域に対する塗りつぶしの使用頻度が高いことを考慮すると，瞬時にして，塗りつぶし描画を終了することがわかります．μPD7220のクロック周波数を2.5MHzから5MHzとしても，描画時間には変動がありません．このことは，つぎのような理由によります．

①CPUにおける境界点サーチ処理は，μPD7220による描画処理と比

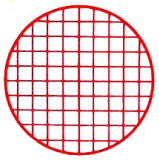
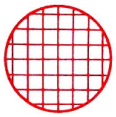
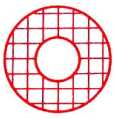
<図8> 塗りつぶしサブルーチンの概略フロー



<写真2>
塗りつぶし描画例



＜図9＞
塗りつぶし描画実行結果

	AREA-FILLING EXAMPLE	FLASH	FLASHLESS
①	$r=240$ 	1.4sec	3.0sec
②	$r=160$ 	0.7sec	1.4sec
③	$r=60$ 	0.1sec	0.2sec
④	$r_1=60$ $r_2=160$ 	1.0sec	1.6sec
⑤	$r=160$ 	1.6sec	2.2sec

CPU：Z80　　$f=1.78$MHz
μPD7220：　$f=2.5$MHz or 5MHz

較して遅い
②CPU処理とμPD7220の処理とが並列動作している
③このため，μPD7220による描画処理時間は無視できる
④CPU処理時間だけが描画に費やす時間的要素となる

この傾向は，③例から⑤例にかけて，さらに顕著となります．境界点の検出が複雑になるからです．

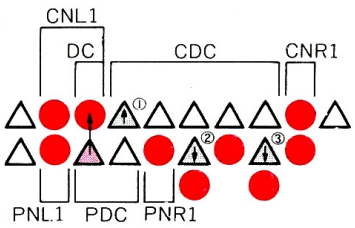
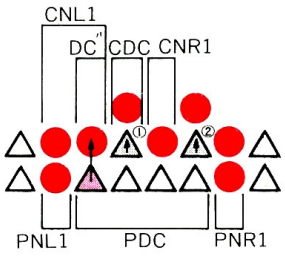
以上の結果から，
①RGB各プレーンに同一のC/Pを順次送出し，カラー表示を行う場合であっても塗りつぶし時間には大きな変動がないこと
②ソフトウェア作成の巧拙によって，システム・パフォーマンスが左右されること
がわかります．

CPUは1.78MHzで動作させています．4MHzにすれば塗りつぶし時間は図6の値の1/2以下となるでしょう．8086などの16ビットCPUを用いれば，さらに高速化できることは述べるまでもありません．このプログラムでは，特殊な演算専用LSI（8087など）は使用していません．

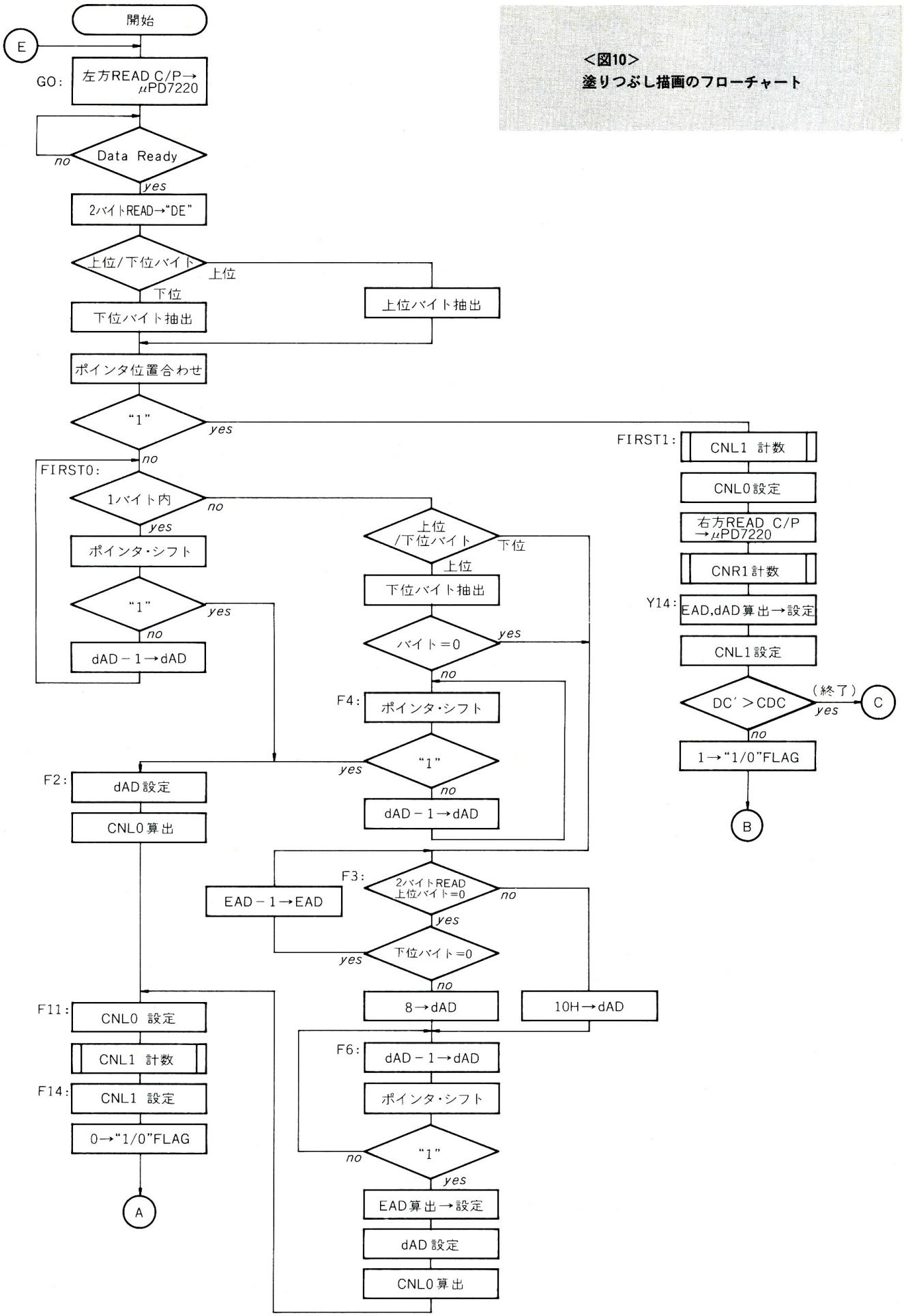
▶複雑な図形の塗りつぶし

境界点の検出はつぎのようにして行います．まず，サーチ開始点を設定します．サーチ開始点から左方への映像メモリ内容の読み出しは，μPD7220にREADコマンドを発行することにより実行します．描画方向＝6，読み出しワード数＝40H（水平方向のアドレス総数）をVECTWコマンドで，サーチ開始絶対番地をCSRWコマンドで，また，ワード単位でアドレスを移動させるため全ビットにマスクをかけた後，READ：Wコマンドを発行します．

＜図11＞　サーチ開始点のドットが"1"のときの凸凹発生検出

	(a)	(b)
MODEL		
CONDITION	(DC′+CDC)＞(PDC+PNR1)	(PDC)＞(DC′+CDC+CNR1)
ADDRESS	EAD±P+(dAD+PDC+PNR1−DC′)	EAD+(dAD+CDC+CNR1)
UP/DOWN	$\overline{\text{UP}}/\overline{\text{DUWN}}$→UP/DOWN	STAY SAME VALUE
OTHERS	(DC′+CDC)−(PDC+PNR1)→PDC	PDC−(DC′+CDC+CNR1)→PDC
	0→DC′/CNL0	0→DC′/CNL0
	PNR1→CNL1	CNR1→CNL1
	1→"1/0"FLAG	1→"1/0"FLAG
	1→"MORE"FLAG	1→"MORE"FLAG

CNL0：現時点における左方向の0の総数　PNL1：境界点サーチ前の左方向の1の総数　EAD：描画のワード・アドレス
CNL1：現時点における左方向の1の総数　PNR0：境界点サーチ前の右方向の0の総数　PDC：境界サーチ前の境界内の0の総数
CNR0：現時点における右方向の0の総数　PNR1：境界点サーチ前の右方向の1の総数　P　：水平方向のワード数
CNR1：現時点における右方向の1の総数　DC′：境界点サーチ前のその点から右方向へ1であるドットの総数　○：1　△：0
CDC：境界内（閉領域内）の0の総数
PNL0：境界点サーチ前の左方向の0の総数　dAD：ドット・アドレス



<図10>
塗りつぶし描画のフローチャート

これらのC/P発行の間，FIFOステータス・フラグのチェックは必要ありません．送出するC/P総数が11バイトであり，μPD7220が描画実行中であっても，FIFOがあふれることはないからです．また，読み出し開始アドレスは主記憶内に格納しておき，ブロック転送命令によってμPD7220へ移送しています．

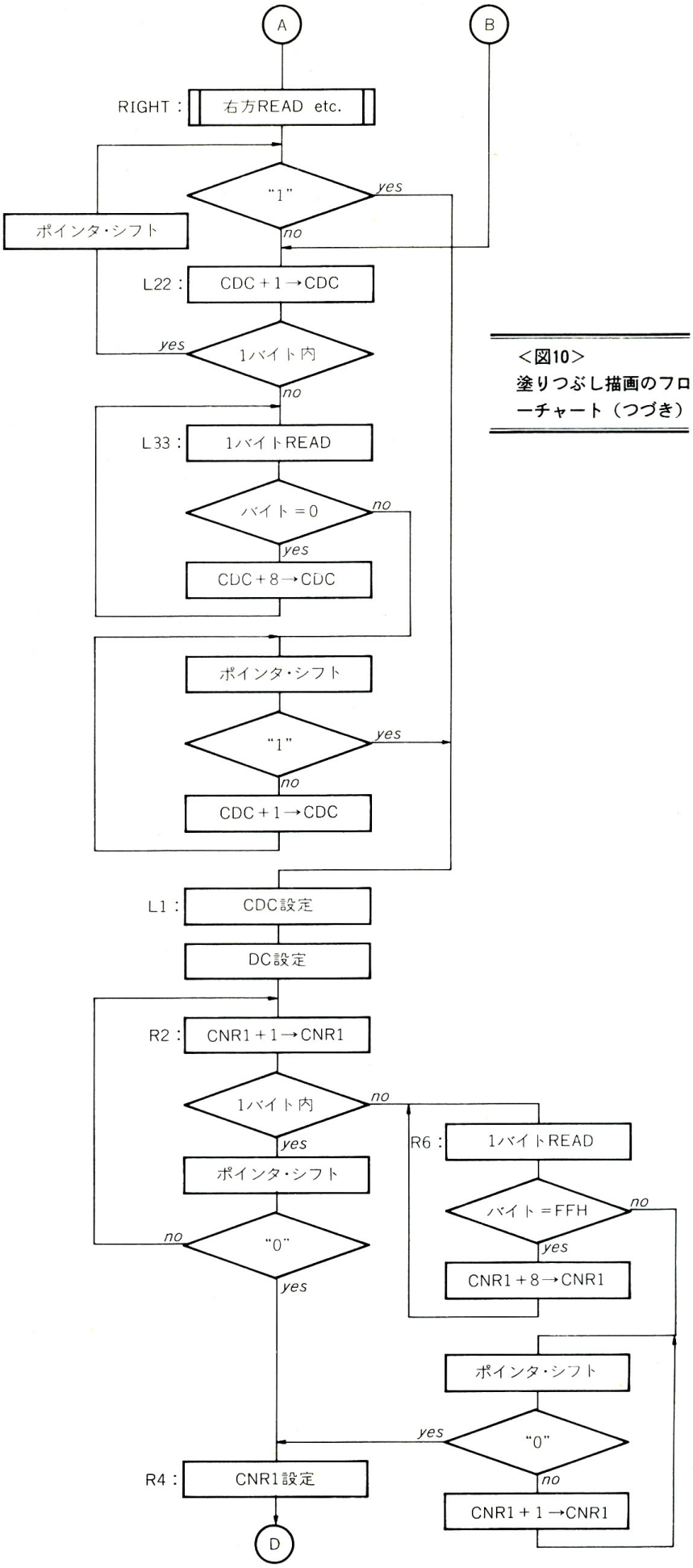
READコマンド発行後，ステータス・フラグDATA-READYが“1”となることを1回だけ確認します．“1”となった時点では，少なくとも2バイトが映像メモリから読み出されてFIFOに蓄積されています．したがって，CPUは直後に2バイトを連続して読み出し可能です．さらに，μPD7220は映像メモリに対するアクセスが可能なタイミングを判別し，自動的にアドレスを更新しながらデータをFIFOに蓄積しています．したがって，以後，データを読み出すたびにステータス・フラグを逐一チェックする必要もありません．

境界点を検出するフローは，さほど難解ではないので説明を省略します．**図10**を参照してください．

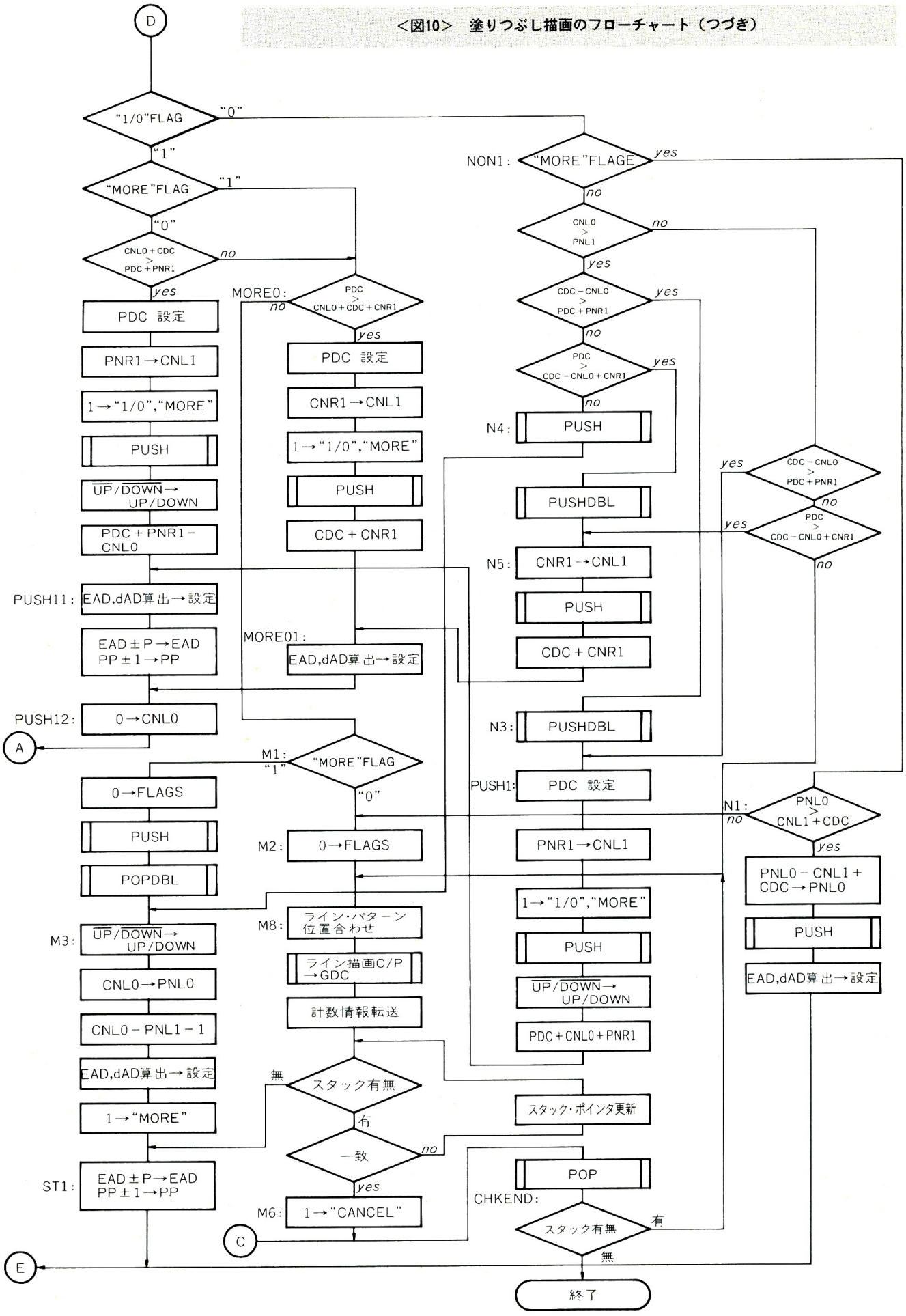
塗りつぶし終了の状態検出や凸凹の状態を検出する目的で，8種類の16ビット汎用レジスタを用います．レジスタ値を確定するため，閉領域を構成する左右の“1”のドット数についても算出します(**図11**)．この塗りつぶしアルゴリズムでは，**境界点サーチ開始点のドットが“1”であるか“0”であるかが重要な因子**となります(**図11**，**図12**)．

各図とも，△点から右方への直線描画による塗りつぶしはすでに終了しており，ふたたび，上方の開始点から境界点サーチが始まるものと仮定します．サーチ開始時には，PNL1，PDC，PNR1は算出設定済みの状態にあります．

サーチ開始点ドットが“1”の場合(**図11**)，(**a**)と(**b**)の2種の凸凹発生が考えられます．凸凹発生の検出がなされたならば，描画情報を一



<図10>
塗りつぶし描画のフローチャート（つづき）


D
"1/0"FLAG
"0"
"1"
"MORE"FLAG
"1"
"0"
CNL0+CDC > PDC+PNR1
no
yes
PDC 設定
PNR1→CNL1
1→"1/0","MORE"
PUSH
UP/DOWN→ UP/DOWN
PDC+PNR1−CNL0
PUSH11:
EAD,dAD算出→設定
EAD±P→EAD
PP±1→PP
PUSH12:
0→CNL0
A
MORE0:
PDC > CNL0+CDC+CNR1
no
yes
PDC 設定
CNR1→CNL1
1→"1/0","MORE"
PUSH
CDC+CNR1
MORE01:
EAD,dAD算出→設定
M1:
"MORE"FLAG
"1"
"0"
0→FLAGS
PUSH
POPDBL
M2:
0→FLAGS
M3:
UP/DOWN→ UP/DOWN
CNL0→PNL0
CNL0−PNL1−1
EAD,dAD算出→設定
1→"MORE"
ST1:
EAD±P→EAD
PP±1→PP
M8:
ライン・パターン位置合わせ
ライン描画C/P→GDC
計数情報転送
スタック有無
無
有
一致
no
yes
M6:
1→"CANCEL"
C
E
NON1:
"MORE"FLAGE
yes
no
CNL0 > PNL1
no
yes
CDC−CNL0 > PDC+PNR1
yes
no
PDC > CDC−CNL0+CNR1
yes
no
N4:
PUSH
PUSHDBL
N5:
CNR1→CNL1
PUSH
CDC+CNR1
N3:
PUSHDBL
PUSH1:
PDC 設定
PNR1→CNL1
1→"1/0","MORE"
PUSH
UP/DOWN→ UP/DOWN
PDC+CNL0+PNR1
スタック・ポインタ更新
POP
CHKEND:
スタック有無
有
無
終了
CDC−CNL0 > PDC+PNR1
yes
no
PDC > CDC−CNL0+CNR1
yes
no
N1:
PNL0 > CNL1+CDC
no
yes
PNL0−CNL1+CDC→PNL0
PUSH
EAD,dAD算出→設定

時スタックし凸凹発生位置に対する境界点サーチを開始します．これを怠たると，塗りつぶしがなされない領域が発生します．

（a）の状態は，

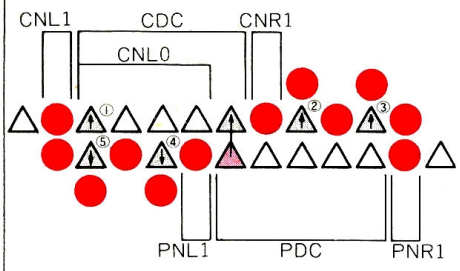
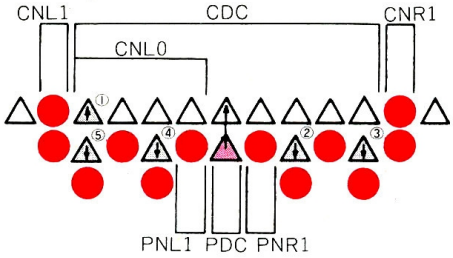
（DC′＋CDC）＞（PDC＋PNR1）

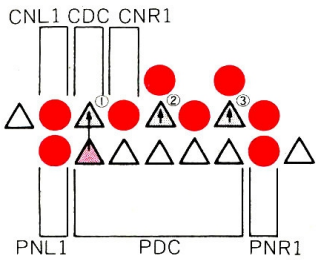
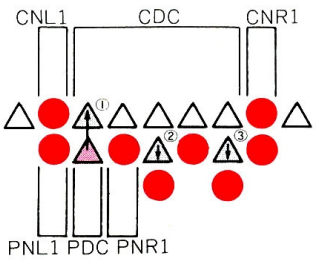
によって検出できます．このとき，すでに導出済みであるワード/ドット・アドレス，描画ドット数など10バイト構成の描画開始点①における塗りつぶし情報をスタックします．

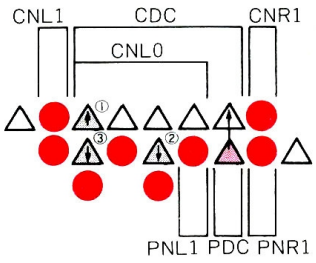
つぎに，点②のサーチ開始番地を次式によって算出します．

EAD±P＋（dAD＋PDC＋PNR1－DC′）

＜図12＞　サーチ開始点のドットが“0”のときの凸凹発生検出

	(a)	(b)
MODEL		
CONDITION	CNL0＞PWL1 AND PDC＞(CDC－CNL0＋CNR1)	CNL0＞PNL1 AND (CDC－CNL0)＞(PDC＋PNR1)
ADDRESS	EAD＋(dAD＋CDC＋CNR1)	EAD±P＋(dAD＋CNL0＋PDC＋PNR1)
UP/DOWN	STAY SAME VALUE	$\overline{UP/DOWN}$→UP/DOWN
OTHERS	PDC－(CDC－CNL0＋CNR1)→PDC	(CDC－CNL0)－(PDC＋PNR1)→PDC
	0→DC /CNL0	0→DC′/CNL0
	CNR1→CNL1	PNR1→CNL1
	1→“1/0”FLAG	1→“1/0”FLAG
	1→“MORE”FLAG	1→“MORE”FLAG
	1→“MORE*”FLAG	1→“MORE*”FLAG

	(c)	(d)
MODEL		
CONDITION	PDC＞(CDC－CNL0＋CNR1) CNL0＞PNL1 IS UNCONDITIONAL	(CDC－CNL0)＞(PDC＋PNR1) CNL0＞PNL1 IS UNCONDITIONAL
PROCESS	SAME AS (a). “MORE *”IS NOT SET.	SAME AS (b). “MORE *”IS NOT SET.

	(e)	
MODEL		
CONDITION	CNL0＞PNL1 OTHER CONDITIONS ARE NOT DETECTED	“1/0”FLAG＝0 AND “MORE”FLAG＝1 PNL0＞(CNL1＋CDC)
ADDRESS	EAD±P＋(dAD＋CNL0－PNL1－1)	EAD－(CNL1＋1)
UP/DOWN	$\overline{UP/DOWN}$→UP/DOWN	STAY SAME VALUE
OTHERS	CNL0－PNL1→PNL0	PNL0－(CNL1＋CDC)→PNL0
	0→“1/0”FLAG	0→“1/0”FLAG
	1→“MORE”FLAG	1→“MORE”FLAG

ここで，括孤外はワード・アドレス，括孤内はドット・アドレスです．また，EAD/dADはそれぞれ点①のワード/ドットのアドレスを示し，Pは水平方向のアドレス総数です．境界点サーチの上下移動方向を示すフラグUP/DOWNが“0”のときPを加算し，“1”のときには減算します．

図11(a)の場合には，Pを加算します．このとき，点③のように凸凹が2箇所以上発生している場合が想定されます．したがって，

(DC′+CDC)−(PDC+PNR1)
⟶PDC

を実行することにより，この状態の検出を可能とします．また，

PNR1⟶CNL1

として，CNL1の算出を省略します．

以下，図11(b)の場合も同様に処理します．

境界点サーチ開始点が“0”である場合には，処理はさらに複雑になります．開始点からみて左右同時に凸凹箇所が発生する場合が考えられるからです（図12）．まず，

(CNL0)>(PNL1)

の判定をします．さらに(a)，(b)の条件が取れたとき，MORE*フラグをセットし，描画開始点①の描画情報を異なる2種のスタック領域にスタックします．MORE*フラグは，スタックを1回以上たてるときに使います．

図12(a)の例では，その後，点②と③の描画情報をスタックしたのち第2のスタック領域にPUSHしてある点①の情報をPOPし，描画開始点④を算出します．点④における描画情報算出後の凸凹条件判定では，図12(e)右側の条件が取れるので点④の情報もスタックします．点⑤から塗りつぶし描画を実行し，つぎつぎにスタック内容をPOPしながら④③②①の順に塗りつぶします．

単に，円や任意四辺形内部の塗りつぶしだけを行う場合には，このような複雑な処理は不要です．したがって，描画速度も高速化できます．

上記以外の凸凹発生判定やその後の処理については，図やフローチャートを参照してください．リストについては省略します．

▶直線描画と塗りつぶし

つぎに，μPD7220が関与する直線描画について説明します．塗りつぶしパターンは8×8ドット構成のユーザが任意に定義可能な領域(TEXT)に格納してあります．塗りつぶしの上下移動方向とテキスト・ポインタを格納するレジスタ(DIRTEXT)値を基にして，パターン情報/バイトを抽出します．さらに，描画開始点のドット・アドレスを基準としてパターン情報の描画開始ドット位置を決定し，TEXTWコマンドによってパターン情報を設定します．

描画方向が水平/垂直の方向に限定されている場合には，VECTWコマンドに付随するパラメータ送出バイト数を一般的な直線描画の場合と比較して，4バイト分を削減できます．パラメータ“DC”に総描画ドット数，“D”に“−1”を設定するだけで，直線描画を正常に実行します．

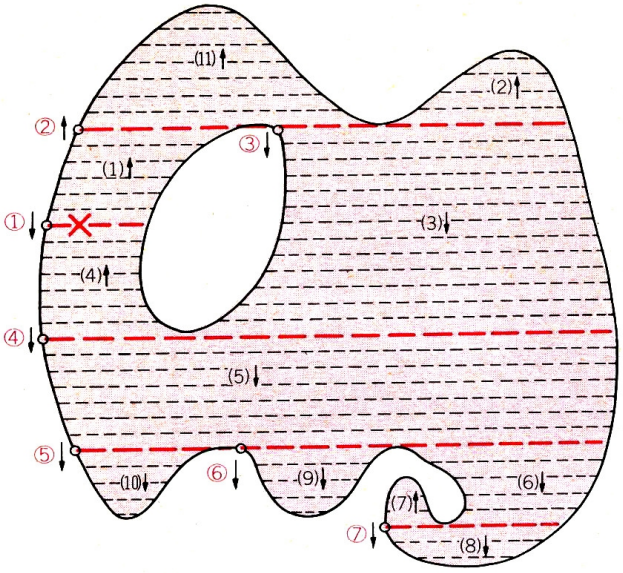
これらのC/P送出のときにも，ステータス・フラグのチェックを省略しています．READコマンド発行後に塗りつぶし描画関連C/Pを送出することになるからです．

次に，実際の塗りつぶし描画時，どのような順番で描画し，描画情報をPUSH/POPするかについて触れます．図13を例に取ります．

描画開始点（×印）左方の境界点①における塗りつぶし描画情報をサーチし，上下方向情報（矢印）とともにPUSHします．領域(1)を上方向に描画し，②，③をPUSHし，領域(2)を描画します．③をPOPして領域(3)を，④をPUSHして領域(4)を塗ります．

本プログラムでは，塗りつぶし描画終了後，全スタック内容と描画アドレスとの一致比較を行います．一致した場合には，CANCELフラグをそのスタック内容に付加します．

<図13>
スタック点と塗りつぶし順序



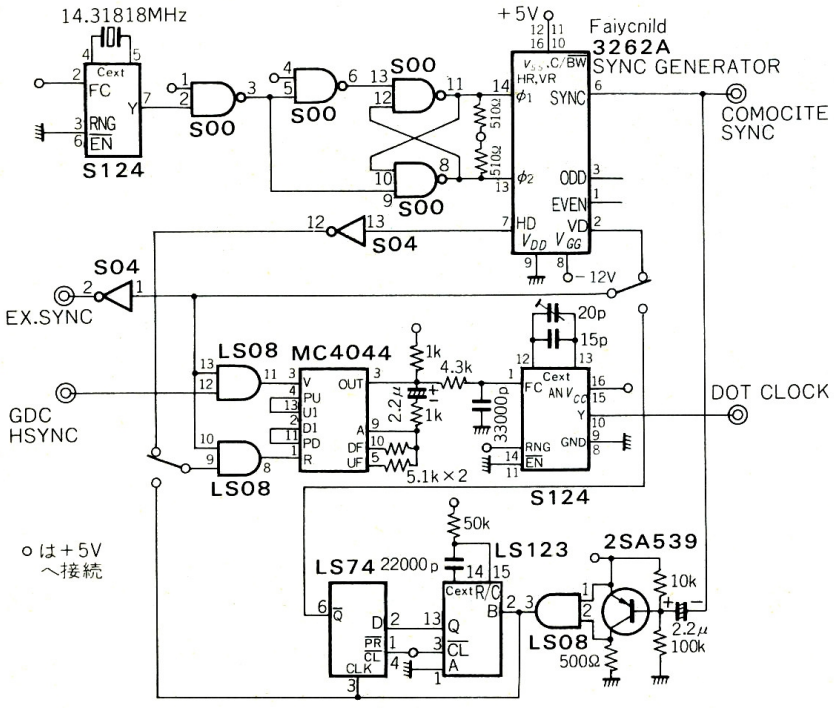
POP時，CANCELフラグ有無チェックを行い，同一のアドレスからの塗りつぶしを禁止します．このような状態は，閉領域内に二重に閉領域が設定されている場合に発生します．

このため，領域(4)の描画後，スタックされている①にはCANCELフラグが付与されます．④をPOPし領域(5)を描画，⑤⑥をPUSHし領域(6)を描画，⑦をPUSHし領域(7)，⑦をPOPし領域(8)，⑥をPOPし領域(9)，⑤をPOPし領域(10)，②をPOPし領域(11)を描画します．

つぎに，①をPOPしたとき，CANCEL＝“1”のため，この描画情報を無視します．ここで，スタック内容が皆無となり塗りつぶしを終了します．

前述のボードにおいて，表示用として使用していない第4のプレーンを閉曲線データ格納用として使用すれば，表示上，閉曲線によって明確に区分されていない領域に対して塗りつぶし描画動作を起こすことができます．さらに，塗りつぶしパターンを任意なドット構成に変化させたとしても，塗りつぶし描画の実行時間は変動しません．同一閉領域の各色プレーン描画ごとに塗りつぶしパターンを変更すると多彩な模様をつけることができます．

<図14> NTSC信号とμPD7220との周期調整回路



●NTSC映像信号との同期

μPD7220は，外部信号によって，内蔵する同期信号発生回路を初期化する機能をもっています．μPD7220をMASTER/SLAVEコマンドによって，スレーブ動作設定すると，EX．SYNC入力端子に接続された信号の立ち下がり時に初期化動作を起こします．2個以上のμPD7220を並列動作させるとき，1個のマスタμPD7220のVSYNC出力を他のスレーブμPD7220のEX．SYNCに接続します．

この外部同期機能は，一般のCRTCやNTSC映像信号（PAL/SECAMでも可）を扱う装置との同期合わせについても有効に動作します．

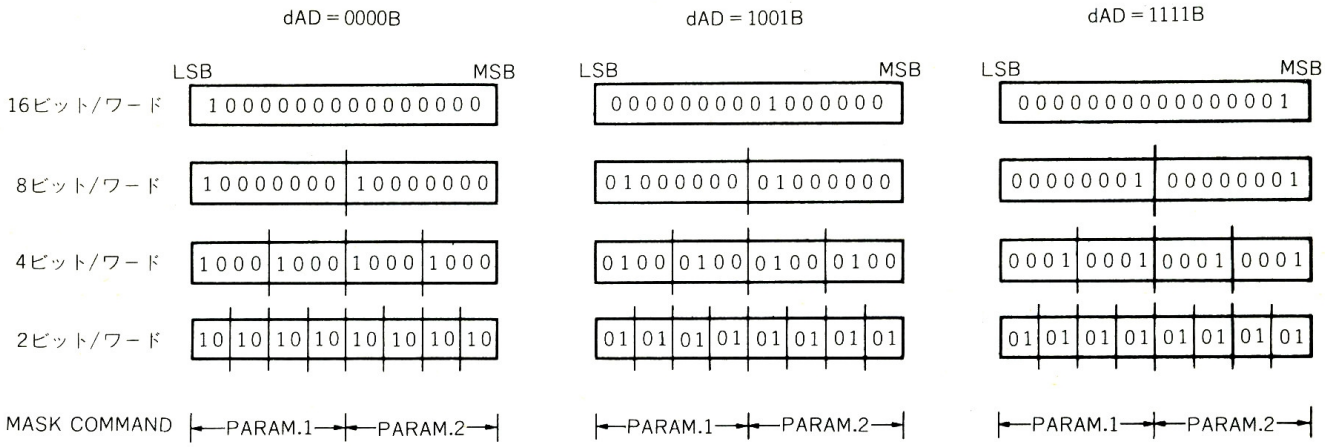
NTSC映像信号と日電PC-8001との同期については，すでに，紹介されています(5)．

NTSC映像信号との同期が取れれば，μPD7220によって描画作成した文字図形表示データをテレビ・カメラなどを仲介せず，直接，VTRに録画したり，逆に，テレビ・カメラやビデオ・ディスクなどが供給する映像信号に重畳させることが容易になります．

μPD7220を使用した装置では，垂直関係同期に関しては，上記した外部同期機能によって達成します．1水平走査期間内の同期ずれ補正は，PLL(フェーズ・ロックト・ループ)を活用します（図14）．

積分器（アクティブ・ローパス・フィルタ）を構成するための，チャージ・ポンプを内蔵した位相検波器MC4044によって，NTSC信号中の同期信号とμPD7220が出力するHSYNCとの立ち下がり位相差を検出

＜図15＞　映像メモリ語長短縮時のMASKコマンドによるドット・アドレス設定例



し，電圧制御発振器（VCO）S124を駆動します．VCOのクロックを分周してμPD7220に供給します．VD（垂直ドライブ）期間には，9H間にわたり1/2H周期の等価パルスなどが含まれるため，位相比較を禁止します．

図14では，同期信号発生器（SG；3262A）からHD，VD信号が直接取り出せる場合と，コンポジットSYNC（複合同期）信号として与えられる場合との2種について示してあります．

2SA539は，コンポジット信号から同期信号だけを分離するスライサです．LS123とLS74とによって，VD信号をディジタル的に分離しています．同期が取れかかっている状態（**写真3**），取れた状態（**写真4**），第2フィールドの同期状態（**写真5**）を参照してください．

◉描画速度を2倍に高める動作設定法

ハードウェアの項でも触れましたが，以下のソフト/ハード的処置により，描画速度を倍にすることができます．

①SCROLLコマンドにより，IM＝“1”に設定する

②描画時，VECTWコマンドにより，DGD＝“1”に設定する

③通常時の倍のPITCH値をPITCHコマンドにより設定する

④並直変換レジスタのロード・クロック・マスキング回路を使用する（**図4**）．

逆に，DAD+2＝“1”に設定すると，μPD7220は表示アドレス・サイクルごとに，アドレスを＋2します．アドレス最下位ビットをメモリ・プレーン切り替え用として用い，32ビットを同時に読み出したり，アドレス最下位ビットを外付け回路によって1クロックごとに反転させ，表示アドレス・サイクルを1クロックに短縮して使用できます．

これらの特殊機能により，ドット・クロックが80MHzを超えるような高解像度グラフィックスへの応用に対してもμPD7220を利用できます．

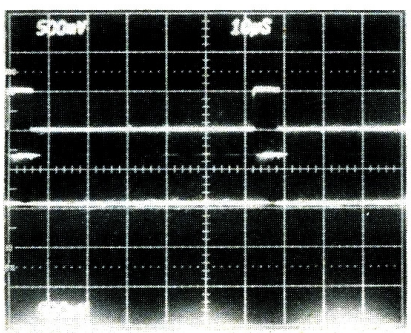
◉映像メモリ・データ・バス幅の選択

μPD7220はグラフィック・モードおよび文字/グラフィック混在モード時，映像メモリに対しデータ・バス16ビットを接続し，1語16ビットの映像メモリを直接制御します．映像メモリの1語長は標準的な16ビットのほか，8/4/2ビットとして使用でき，幅広い応用に対処できます．とくに，Low-endグラフィックスへの応用では，つぎのような好結果をもたらします．

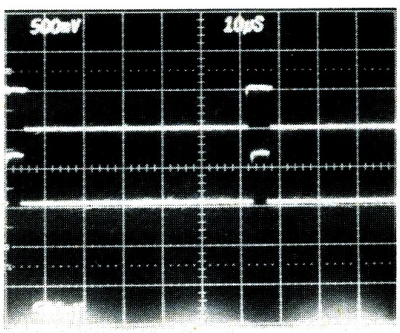
(1)記憶素子数や映像メモリ制御用外付け回路を大幅に削減できる．

(2)μPD7220に供給するクロック周波数を高くでき，描画速度が増大する．

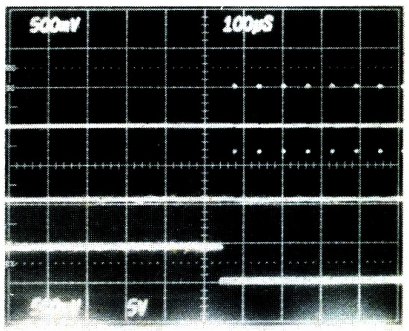
256×256ドット程度のビデオ・ゲームなどでよく使用される低解像度グラフィックスを，1語16ビット構成で実現しようとすると，μPD7220



＜写真3＞　NTSC信号とμPD7220のHSYNC



＜写真4＞　NTSC信号とμPD7220のHSYNC



＜写真5＞　NTSC信号とμPD7220のHSYNC（第2フィールドの同期状態）

のクロック周波数が1MHzを割ります．さらに，RGB 3枚のメモリ・プレーンをもつとき，1ビット出力のDRAMを使用すると，48個の記憶素子を実装しなければなりません．しかし，1語4ビット構成とすれば，描画速度は4倍となり，記憶素子数は12個で済みます．4ビット入出力の記憶素子を使用すれば，各色プレーンに1個ずつ，計3個で構成できることになります．

1語長を短縮するとき問題となるのは，描画時におけるドット・アドレスの処置です．μPD7220は本来，1語16ビットに対する制御を行うように設計されており，ドット・アドレス・レジスタの最上位または最下位ビットの状態をみて，ワード・アドレスの演算の必要性をチェックしています．

したがって，語長を短縮したときにも，このチェックがμPD7220内部で等価的に実行できれば，正常に描画動作を実行します．

描画開始ドット・アドレスの設定は，CSRWコマンドによるのではなく，MASKコマンドによって直接的に行います（図15）．

語長短縮を行った場合であっても，WRITEコマンドやDMA転送を使用できます．ワード単位書き込みのほかにバイト単位での書き込みが許されているからです．

◍その他

μPD7220をフラッシュレス描画設定すると，水平/垂直のブランキング期間にのみ描画動作を起動します．

不定領域に対する塗りつぶしのように，CPUの処理時間比率が大きい場合には，フラッシュ描画設定時と比較して，総描画時間には大きな差異を生じません．しかし，連続的に描画実行する場合には差異が出てきます．

描画速度を極限にまで高めて使用したい応用では，μPD7220 2個構成による，サイクル・スチール表示/描画が効果を発揮します．クロック周波数をできるだけ低くした表示専用μPD7220と，フラッシュ描画設定し，クロック周波数を高くした描画専用μPD7220を用い，共有する映像メモリのアドレス/データをスイッチしながら使用します．

文字表示の場合には，1ドットの表示画面上での縦横化を1対1に取る必然性はありません．

このため，グラフィック表示においても1対1の縦横比となっていないパソコンが一部に見受けられます．円を円として描画あるいは表示できないため，惰円を描画して表示上，円として見せる方策が取られていますが，プリンタで印字したとき真円にはなりません．これでは真のグラフィックスであるとはいえません．

CADなどで使用される距離直視用点格子描画や惰円，二次曲線などの描画は，μPD7720を使用することにより簡略化できます．

CPUによって直接，映像メモリに1ドットを描画する場合には，描画すべきワードおよびドットのアドレスを算出し，ある1ドットに対してR/M/Wを実行する必要があります．

μPD7220を用いたときには，あるドットから8方向への描画位置変化だけをとらえ，VECTWコマンドによりその描画方向を設定し，描画実行コマンドVECTEを発行するだけで描画します．

また，点格子描画のように，規則性のあるドット描画の場合には，CPUにドット描画位置を算出させず，μPD7220に直線描画をさせ，ドット描画と描画アドレスの更新とを同時に行うソフトを構成すれば，良好な描画結果を得ることができます．

▒おわりに

CRT端末装置の将来動向はつぎのようになると推測されます．

①高級仕様化志向とLow-end志向への分化

②映像信号制御回路など周辺回路のゲート・アレイ化

③VIDEOTEX，TELETEXT表示制御などへの特殊用途化

1985年頃には，ビット・マップ・メモリに文字をドット単位描画する文字表示装置が普及し，一般化することが予想されます．文字のマルチ・フォント化，マルチ・スペース化，グラフ表示や手書き文字との混在などへの発展性を秘めています．

Low-end志向のCRTCは，不要な回路の集積化を嫌い，ユーザの用途に合致したものとするため，専用LSIではなく，ゲート・アレイによって供給されることになるでしょう．

VIDEOTEX用LSIは，電話回線などを介して商店や各家庭のパソコンに接続され，手書き文字サインの照合や電子郵便などの各種情報の授受を迅速に行うメディアの進展を加速する素子となることが予想されます．

◆参考文献◆

(1) 日本電気；μPD7220 GDCユーザーズ・マニュアル，IEM-734

(2) 小口哲司；1ドットを800ns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1981年10月12日，p.186

(3) 日本電気；μPD7220D開発速報，ID5977

(4) 井上壽雄；インターフェース/ペリフェラル回路，エレクトロニクス，昭和57年1月～5月号

(5) 釣譲一，谷内実則；パソコン画像をVTRで録画する，インターフェース1982年2月号，p.212

(6) D. L. Ruhberg；Low Cost Components Make Color Graphics Practical, Digital Design, June 1982, p.74

(7) Steve Ciarcia；High-Resolution Sprite-Oriented Color Graphics, Byte, Aug. 1982, p.57

(8) Ron Shinn；Color graphics board eliminate wait state, Electronic Design, Sep.16 1982, p.247

＜リスト＞　基本描画サブルーチン

```
;
;DRAWING SUBROUTINE EXAMPLE FOR "GDC"
; WRITTEN BY T.OGUCHI
;
        ;
        ;SYSTEM EQUATE
        ;
GDCCMD  EQU     9FH             ;GDC COMMAND ADDRESS
GDCPAR  EQU     9EH             ;GDC PARAMETER ADDRESS
VECTE   EQU     6CH             ;VECTE COMMAND CODE
VECTWC  EQU     4CH             ;VECTW COMMAND CODE
TEXTE   EQU     68H             ;TEXTE COMMAND CODE
START   EQU     6BH             ;START COMMAND CODE
WRITEC  EQU     22H             ;WRITE WORD COMMAND CODE (RESET)
        ;
; INITIALIZE GDC AND CLEAR DISPLAY MEMORY
        ;
CLMEM:  LD      HL,CPINIT
        LD      DE,SYNC
        LD      BC,36H
        LDIR                    ;ALLOCATE C/P BYTES
        XOR     A
        OUT     (GDCCMD),A      ;<RESET>
        LD      BC,900H+GDCCMD
        LD      HL,SYNC
        CALL    SEND            ;<SYNC>,<P1-P8>
        LD      BC,200H+GDCCMD
        CALL    SEND            ;<PITCH>,<P1>
        LD      BC,200H+GDCCMD
        CALL    SEND            ;<ZOOM>,<P1>
        LD      BC,900H+GDCCMD
        CALL    SEND            ;<SCROLL>,<P1-P8>
        LD      BC,400H+GDCCMD
        CALL    SEND            ;<CSRFORM>,<P1-P3>
        ;CLEAR DISPLAY MEMORY
        LD      BC,400H+GDCCMD
        CALL    SEND            ;<CSRW>,<P1-P3>
        LD      BC,600H+GDCCMD
        CALL    SEND            ;<VECTW>,<P1-P5>
        LD      HL,MASK
        LD      BC,300H+GDCCMD
        CALL    SEND            ;<MASK>,<P1-P2>
        LD      BC,900H+GDCCMD
        CALL    SEND            ;<TEXTW>,<P1-P8>
        LD      A,WRITEC
        OUT     (GDCCMD),A      ;<WRITE:W(RESET)>
        LD      A,TEXTE
        OUT     (GDCCMD),A      ;<TEXTE>
        LD      A,START
        OUT     (GDCCMD),A      ;<START>
        RET
        ;
; DRAWING SUBROUTINE FOR "LINE"
;   ## INPUT DATA -- (X1),(Y1),,(X2),(Y2),,(COLOR)
        ;
LINE:   CALL    CSR             ;SET CURSOR ADDRESS
LINE1:  LD      HL,(X2)         ;GET SECOND COORDINATE X
        LD      DE,(X1)         ;GET FIRST COORDINATE X
        LD      A,8             ;SET "LINE" AND INITIAL "DIR"
L1:     OR      A
        SBC     HL,DE           ;CALCULATE DX
        JR      NC,L2           ;CHECK IF X2<X1
        ADD     HL,DE           ;RETURN X2
        EX      DE,HL           ;EXCHANGE X2,X1
        OR      7               ;CHANGE "DIR"
        SBC     HL,DE           ;CALCULATE DX
L2:     PUSH    HL              ;PUSH DX (ABSOLUTE VALUE)
        LD      HL,(Y2)         ;GET SECOND COORDINATE Y
        LD      DE,(Y1)         ;GET FIRST COORDINATE Y
L3:     XOR     3               ;CHANGE "DIR"
        SBC     HL,DE
        JR      NC,L4           ;CHECK IF Y2<Y1
        ADD     HL,DE           ;RETURN Y2
        EX      DE,HL           ;EXCHANGE Y2,Y1
        JR      L3
        ;
L4:     EX      DE,HL           ;SAVE DY (ABSOLUTE VALUE) TO 'DE'
        POP     HL              ;POP DX
        PUSH    HL              ;PUSH DX
        LD      BC,0FFFFH
        SBC     HL,DE
        JR      C,L5            ;CHECK IF DX>DY
        XOR     1               ;CHANGE "DIR"
        ADD     HL,BC
        JR      C,L5            ;CHECK IF DX-DY=0
        OR      1               ;CHANGE "DIR"
L5:     POP     HL              ;POP DX
        PUSH    HL              ;PUSH DX
        ADD     HL,BC
        JR      C,L6            ;CHECK IF DX=0
        INC     A               ;CHANGE "DIR"
        AND     0FCH            ;CHANGE "DIR"
L6:     LD      H,D
        LD      L,E             ;LOAD DY TO 'HL'
        ADD     HL,BC
        JR      C,L7            ;CHECK IF DY=0
        INC     A               ;CHANGE "DIR"
        AND     0FEH            ;CHANGE "DIR"
L7:     LD      (VECTW+1),A     ;SET "DIR"
        POP     HL              ;POP DX
        DEC     A
        BIT     1,A
        JR      NZ,L8           ;CHECK IF NEEDS EXCHANGE
        EX      DE,HL           ;EXCHANGE DX,DY
L8:     INC     A               ;RETURN "DIR"
        EX      AF,AF           ;SAVE "DIR"
        SLA     L
        RL      H
        PUSH    HL              ;PUSH 2*DY
        CALL    SCSRW           ;<CSRW>,<P1-P5>
        LD      A,VECTWC        ;SET "VECTW" COMMAND CODE
        OUT     (GDCCMD),A      ;<VECTW>
        EX      AF,AF'          ;RETURN "DIR"
        OUT     (GDCPAR),A      ;<P1>
        OUT     (C),E           ;<P2>
        SET     6,D             ;SET "DGD" (IF NEEDED)
        OUT     (C),D           ;<P3>
        LD      (VECTW+2),DE    ;STORE "DC" FOR COLOR
        RES     6,D             ;RESET "DGD" (IF NEEDED)
        POP     HL              ;POP 2*DY
        PUSH    HL              ;PUSH 2*DY
        OR      A
        SBC     HL,DE           ;2*DY-DX--->'HL'
        OUT     (C),L           ;<P4>
        OUT     (C),H           ;<P5>
        LD      (VECTW+4),HL    ;STORE "D" FOR COLOR
        OR      A
        SBC     HL,DE           ;2*DY-2*DX--->'HL'
        OUT     (C),L           ;<P6>
        OUT     (C),H           ;<P7>
        LD      (VECTW+6),HL    ;STORE "D2" FOR COLOR
        POP     HL              ;2*DY--->'HL'
        OUT     (C),L           ;<P8>
        OUT     (C),H           ;<P9>
        LD      (VECTW+8),HL    ;STORE "D1" FOR COLOR
L9:     LD      A,VECTE         ;SET "VECTE" COMMAND CODE
        OUT     (GDCCMD),A      ;<VECTE>
        CALL    CSR12           ;CHECK AND SET "EAD" FOR COLOR
        RET     C               ;RETURN IF END
        CALL    SCSRW           ;<CSRW>,<P1-P3>
        LD      BC,0A00H+GDCCMD ;SET NUMBER OF COMMAND/PARAMETER BYTES
        CALL    SEND            ;<VECTW>,<P1-P9>
        JR      L9              ;REDRAW FOR COLOR
        ;
; TRANSFER C/P TO GDC
        ;
SCSRW:  LD      HL,CSRW         ;SET "CSRW" START ADDRESS
        LD      BC,0400H+GDCCMD ;SET NUMBER OF C/P BYTES
SENDEM: IN      A,(GDCPAR)      ;GET STATUS FLAGS
        BIT     2,A
        JR      Z,SENDEM        ;CHECK IF FIFO-EMPTY=1
SEND:   OUTI                    ;TRANSFER COMMAND BYTE
        DEC     C               ;DECREMENT I/O ADDRESS
        OTIR                    ;TRANSFER WHOLE PARAMETER BYTES
        RET
        ;
; TRANSLATE COORDINATE-X,Y TO "EAD" WITH COLOR CONTROL
        ;
CSR:    LD      HL,(X1)         ;GET COORDINATE X
CSR10:  LD      DE,(Y1)         ;GET COORDINATE Y
CSR11:  LD      A,E
        CPL
        LD      E,A
        LD      A,D
        CPL
        LD      D,A             ;COMPLEMENT 'DE'
        SRL     H
        RR      L
        RRA
        SRL     H
        RR      L
        RRA
        SRL     D
        RR      E
        RR      L
        RRA
        SRL     D
        RR      E
        RR      L
        RRA
        LD      (CSRW+3),A      ;SET DOT ADDRESS
        LD      H,E
        LD      (CSRW+1),HL     ;SET "EAD"
        LD      A,8
        LD      (COLOR1),A      ;INITIALIZE COLOR POINTER
CSR12:  LD      A,(COLOR1)      ;LOAD CP.
        LD      B,A             ;SAVE CP.
        LD      A,(COLOR)       ;LOAD COLOR
        LD      C,A             ;SAVE COLOR
CSR13:  SRL     B               ;SHIFT POINTER
        JR      C,CSR15         ;CHECK IF END
        AND     B               ;COMPARE COLOR,CP.
        JR      NZ,CSR14        ;CHECK IF COINCIDENCE
        LD      A,C             ;LOAD COLOR
        JR      CSR13           ;RESTART COMPARISON
        ;
CSR14:  LD      A,B             ;LOAD CP.
        LD      (COLOR1),A      ;STORE CP.
        SRL     B               ;SHIFT CP. FOR "EAD" CALCULATION
        LD      A,(CSRW+3)      ;LOAD "EADH"
        AND     0F0H            ;CLEAR LOWER 4 BITS
        ADD     A,B
        LD      (CSRW+3),A      ;STORE "EADH"
        RET
        ;
CSR15:  LD      A,8
        LD      (COLOR1),A      ;INITIALIZE COLOR POINTER
        RET
        ;
; DRAWING SUBROUTINE FOR "RECTANGLE"
;   ## INPUT DATA -- (X1),(Y1),,(X2),(Y2),,(COLOR)
        ;
REC:    CALL    CSR             ;SET CURSOR ADDRESS
        LD      HL,(X2)         ;GET SECOND COORDINATE X
        LD      DE,(X1)         ;GET FIRST COORDINATE X
        XOR     A               ;INITIALIZE "DIR"
        SBC     HL,DE           ;CALCULATE DX
        JR      NC,R1           ;CHECK IF X2<X1
        ADD     HL,DE           ;RETURN X2
        EX      DE,HL           ;EXCHANGE X1,X2
        XOR     6               ;CHANGE "DIR"
```

```
        SBC     HL,DE           ;CALCULATE DX
R1:     PUSH    HL              ;PUSH DX (ABSOLUTE VALUE)
        LD      HL,(Y2)         ;GET SECOND COORDINATE Y
        LD      DE,(Y1)         ;GET FIRST COORDINATE Y
R2:     XOR     2               ;CHANGE "DIR"
        SBC     HL,DE           ;CALCULATE DY
        JR      NC,R3           ;CHECK IF Y2<Y1
        ADD     HL,DE           ;RETURN Y2
        EX      DE,HL           ;EXCHANGE Y1,Y2
        JR      R2
        ;
R3:     LD      BC,0
        ADC     HL,BC
        JP      Z,LINE1         ;CHECK IF DY=0 THEN EXIT TO LINE
        EX      DE,HL           ;SAVE DY TO 'DE'
        POP     HL              ;POP DX
        ADC     HL,BC
        JP      Z,LINE1         ;CHECK IF DX=0 THEN EXIT TO LINE
        BIT     1,A
        JR      NZ,R4           ;CHECK IF NEEDS EXCHANGE
        EX      DE,HL           ;EXCHANGE DX,DY
R4:     OR      40H             ;SET "RECTANGLE"
        LD      (VECTW+1),A     ;SET "DIR"
        EX      AF,AF'          ;SAVE "DIR"
        PUSH    HL              ;PUSH DX
        CALL    SCSRW           ;<CSRW>,<P1-P3>
        LD      A,VECTWC        ;LOAD "VECTW" COMMAND CODE
        OUT     (GDCCMD),A      ;<VECTW>
        EX      AF,AF'          ;RETURN "DIR"
        OUT     (GDCPAR),A      ;<P1>
        LD      HL,4003H        ;SET "DC","DGD" (IF NEEDED)
        OUT     (C),L           ;<P2>
        OUT     (C),H           ;<P3>
        LD      (VECTW+2),HL    ;STORE "DC" FOR COLOR
        POP     HL              ;POP DX
        OUT     (C),L           ;<P4>
        OUT     (C),H           ;<P5>
        LD      (VECTW+4),HL    ;STORE "D" FOR COLOR
        OUT     (C),E           ;<P6>
        OUT     (C),D           ;<P7>
        LD      (VECTW+6),DE    ;STORE "D2" FOR COLOR
        LD      DE,0FFFFH       ;LOAD "D1=-1"
        OUT     (C),E           ;<P8>
        OUT     (C),D           ;<P9>
        LD      (VECTW+8),DE    ;STORE "D1" FOR COLOR
        OUT     (C),L           ;<P10>
        OUT     (C),H           ;<P11>
        LD      (VECTW+10),HL   ;STORE "DM" FOR COLOR
R5:     IN      A,(GDCPAR)      ;GET STATUS FLAGS
        BIT     1,A
        JR      NZ,R5           ;CHECK IF FIFO-FULL=0
        LD      A,VECTE         ;LOAD "VECTE" COMMAND CODE
        OUT     (GDCCMD),A      ;<VECTE>
        CALL    CSR12           ;CHECK AND SET "EAD" FOR COLOR
        RET     C               ;RETURN IF END
        CALL    SCSRW           ;<CSRW>,<P1-P3>
        LD      BC,0C00H+GDCCMD ;SET NUMBER OF C/P BYTES
        CALL    SEND            ;<VECTW>,<P1-P11>
        JR      R5              ;REDRAW FOR COLOR
        ;
; DRAWING SUBROUTINE FOR "CIRCLE"
;   ## INPUT DATA -- (X2),(Y2),,(RR),,(COLOR)
        ;
CIRCLE: LD      A,22H           ;SET "CIRCLE","DIR"
        LD      (GR+1),A        ;SAVE INITIAL "DIR"
        LD      (VECTW+1),A     ;STORE "DIR"
        CALL    CDEF1           ;SET "EAD"
        LD      HL,0            ;LOAD "DM=0"
        LD      (VECTW+10),HL   ;STORE "DM=0"
C10:    LD      DE,0B504H       ;LOAD 2**(1/2)/2 IN HEX
        LD      BC,(RR)         ;LOAD RADIUS (RR)
        CALL    MULTI           ;(RR)*2**(1/2)/2
C11:    SET     6,H             ;SET "DGD" (IF NEEDED)
        LD      (VECTW+2),HL    ;STORE "DC"
        LD      HL,(RR)         ;LOAD (RR)
        DEC     HL              ;DECREMENT (RR)
        LD      (VECTW+4),HL    ;STORE "D=RR-1"
        SLA     L
        RL      H               ;GET 2*(RR-1)
        LD      (VECTW+6),HL    ;STORE "D2=2*(RR-1)"
        LD      HL,0FFFFH       ;LOAD "D1=-1"
        LD      (VECTW+8),HL    ;STORE "D1=-1"
C12:    CALL    SCSRW           ;<CSRW>,<P1-P3>
        LD      BC,0C00H+GDCCMD ;SET NUMBER OF C/P BYTES
        CALL    SEND            ;<VECTW>,<P1-P11>
C13:    IN      A,(GDCPAR)      ;GET STATUS FLAGS
        BIT     1,A
        JR      NZ,C13          ;CHECK IF FIFO-FULL=0
        LD      A,VECTE         ;LOAD "VECTE" COMMAND CODE
        OUT     (GDCCMD),A      ;<VECTE>
C14:    CALL    CSR12           ;CHECK AND SET "EAD"
        JR      NC,C12          ;REDRAW FOR COLOR
        LD      A,(VECTW+1)     ;LOAD "DIR"
        ADD     A,3             ;CHANGE "DIR"
        AND     27H             ;CHANGE "DIR"
        LD      (VECTW+1),A     ;STORE "DIR"
        LD      HL,(GR+1)       ;LOAD INITIAL "DIR" TO 'L'
        CP      L               ;COMPARE "DIR"
        RET     Z               ;RETURN IF END
        BIT     0,A
        JR      NZ,C14          ;REDRAW FOR ALL OCTANTS
        CALL    CDEF1           ;RENEW "EAD"
        JR      C12             ;REDRAW FOR ALL OCTANTS
        ;
;FOCUS INTO ONE OF OCTANT AND SET "DIR"
;   ## INPUT DATA -- INTEGRAL DEGREE: 'HL'
;   ## OUTPUT DATA -- FOCUSED DEGREE WITHIN 45: 'HL', DIR: 'A'
        ;
FOCUS:  LD      B,3             ;LOAD SUBTRACT VALUE
        LD      A,0FFH          ;LOAD INITIAL "DIR"
        LD      DE,2DH          ;LOAD 45 DEGREES IN HEX
FOCUS1: SUB     B               ;CHANGE "DIR"
        SBC     HL,DE           ;THETA-45
        JR      Z,FOCUS2        ;CHECK IF THETA=0
        JR      NC,FOCUS1       ;CHECK IF THETA<0
FOCUS2: ADD     HL,DE           ;THETA+45
        AND     7               ;EXTRACT LOWER 3 BITS
        OR      20H             ;SET "CIRCLE"
        BIT     0,A
        RET     Z               ;CHECK IF END
        EX      DE,HL
        SBC     HL,DE           ;CALCULATE 45-THETA
        RET
;   ## INPUT DATA -- INTEGRAL DEGREE:'HL'
;   ## OUTPUT DATA -- SIN(THETA):'HL'
        ;
FOCSIN: CALL    FOCUS           ;FOCUS AND SET "DIR"
        LD      (GR+1),A        ;SAVE "DIR"
SIN:    LD      A,L
        OR      H
        RET     Z               ;RETURN IF DEGREE=0
        PUSH    AF              ;PUSH DEGREE
        ;
        ;CONVERT DEGREE TO RADIAN
        ;
        LD      C,L
        LD      B,H             ;LOAD 'HL' TO 'BC'
        LD      DE,(SINK)       ;LOAD (PI/180) IN HEX
        LD      HL,0
        LD      A,6             ;SET LOOP COUNTS
SIN2:   SRL     B
        RR      C               ;SHIFT MULTIPLIER
        JR      NC,SIN3         ;CHECK IF LSB=0
        ADD     HL,DE           ;ADD
SIN3:   DEC     A               ;DECREMENT LOOP COUNTS
        JR      Z,SIN4          ;CHECK IF END
        RR      H
        RR      L               ;SHIFT RESULT
        JR      SIN2            ;RESTART MULTIPLICATION
        ;
        ;END OF CONVERSION
        ;SIN(THETA :RADIAN)
        ;
SIN4:   POP     AF              ;POP DEGREE
        CP      5
        JR      C,SIN5          ;CHECK IF LESS THAN 4 DEGREES
        PUSH    AF              ;PUSH DEGREE
        EX      DE,HL
        LD      C,E
        LD      B,D             ;X TO 'BC','DE'
        CALL    MULTI           ;X**2 ---> 'HL', X ---> 'DE'
        EX      DE,HL
        POP     AF              ;POP DEGREE
        PUSH    HL              ;SAVE X
        CP      18H
        JR      C,SIN1          ;CHECK IF LESS THAN 23 DEGREES
        PUSH    DE              ;SAVE X**2
        LD      BC,(SINK+4)     ;LOAD 1/20 IN HEX
        CALL    MULSUB          ;1-X**2/20 ---> 'BC'
        POP     DE              ;POP X**2
        CALL    MULTI           ;X**2*(1-X**2/20) ---> 'HL'
        EX      DE,HL
SIN1:   LD      BC,(SINK+2)     ;LOAD 1/6 IN HEX
        CALL    MULSUB          ;1-X**2*(1-X**2/20)/6 ---> 'BC'
        POP     DE              ;RETURN X
        CALL    MULTI           ;X*(1-X**2*(1-X**2/20)/6) --->'HL'
        ;
        ;END OF SIN(THETA) CALCULATION
        ;
SIN5:   EX      DE,HL
        LD      BC,(RR)         ;LOAD RADIUS
        ;
; CALCULATE X*Y (BOTH DECIMAL FRACTION)
;   ## INPUT DATA -- MULTIPLICAND X:'DE' , MULTIPLIER Y:'BC'
;   ## OUTPUT DATA -- X*Y:'HL'
        ;
MULTI:  LD      HL,0
        XOR     A               ;CLEAR SHIFT COUNTS
MULTI1: EX      AF,AF'          ;SAVE SC.
        LD      A,C
        OR      B
        JR      Z,MULTI2        ;CHECK IF Y=0 (END OF CALCULATION)
        EX      AF,AF'          ;RETURN SC.
        INC     A               ;INCREMENT SC.
        SRL     B
        RR      C               ;SHIFT Y
        JR      NC,MULTI3       ;CHECK IF LSB=0
        ADD     HL,DE           ;ADD
MULTI3: RR      H
        RR      L               ;SHIFT X*Y
        JR      MULTI1          ;RESTART MULTIPLICATION
        ;
MULTI2: EX      AF,AF'          ;RETURN SC.
        NEG
        AND     0FH             ;CALCULATE SC.
        RET     Z               ;CHECK IF END
        LD      B,A             ;LOAD SC. TO 'B'
MULTI4: SRL     H
        RR      L
        DJNZ    MULTI4          ;SHIFT RESULT SC. TIMES
        RET     NC              ;RETURN IF NO ROUND CONDITION
        INC     HL              ;ROUND OFF
        RET
        ;
;CALCULATE (1-X*Y)
        ;
MULSUB: CALL    MULTI           ;X*Y
        EX      DE,HL
        LD      HL,0
        OR      A
        SBC     HL,DE           ;1-X*Y
        LD      C,L
        LD      B,H             ;LOAD 1-X*Y TO 'BC'
        RET
        ;
;DRAWING SUBROUTINE FOR "ARC/CIRCLE"
;   ## INPUT DATA -- (X2),(Y2),,(RR),,(T1),(T2),,(COLOR)
```

```
ARCCRL: LD      HL,(T1)         ;GET (T1)
        LD      DE,(T2)         ;GET (T2)
        OR      A
        SBC     HL,DE           ;(T1)-(T2)
        JP      Z,CIRCLE        ;CHECK IF (T1)=(T2)
        EX      DE,HL
        CALL    FOCSIN          ;FOCUS (T2),CALCULATE RR*SIN(T2)
        LD      (T2R),HL        ;STORE (RR)*SIN(T2)
        LD      HL,(T1)         ;GET (T1)
        CALL    FOCUS           ;FOCUS (T1)
        LD      (GR),A          ;SAVE "DIR(T1)"
        CALL    SIN             ;CALCULATE RR*SIN(T1)
        LD      (T1R),HL        ;STORE (RR)*SIN(T1)
        LD      HL,(GR)         ;LOAD "DIR(T1)","DIR(T2)"
        LD      A,H
        CP      L
        JR      Z,ACRL1         ;CHECK IF WITHIN SAME OCTANT
        BIT     0,A
        JR      Z,ACRL2
ACRL3:  LD      HL,(T2R)        ;LOAD "DM=RR*SIN(T2)"
        CALL    CDEF13          ;SET "EAD"
        CALL    C10             ;DRAW ARC
ACRL4:  LD      HL,(GR)         ;LOAD "DIR(T1)","DIR(T2)"
        CP      L
        JR      Z,ACRL5         ;CHECK IF DIR(T1)=DIR(T2)
        LD      B,A
        LD      A,L
        LD      HL,0
        LD      (VECTW+10),HL   ;STORE "DM"
        CALL    CDEF11          ;SET "EAD"
        CALL    C12             ;DRAW ARC
ACRL5:  LD      HL,0
        BIT     0,A
        JR      NZ,ACRL12
        LD      HL,(T1R)        ;LOAD "DM=RR*SIN(T1)"
        CALL    CDEF13          ;SET "EAD"
        JP      C12             ;DRAW ARC
        ;
;DEFINE DRAWING START POINT
        ;
CDEF13: LD      (VECTW+10),HL   ;STORE "DM"
CDEF12: LD      A,(GR+1)        ;LOAD "DIR(T2)"
        LD      B,A             ;SAVE "DIR(T2)"
        ADD     A,0BH           ;CHANGE "DIR(T2)"
        AND     27H             ;CHANGE "DIR(T2)"
CDEF11: LD      (GR+1),A        ;STORE "DIR(T2)"
        LD      A,B
CDEF10: LD      (VECTW+1),A
CDEF:   BIT     0,A
        JR      Z,CDEF1         ;CHECK IF NEEDS "DIR"-3
        ADD     A,0DH           ;CHANGE "DIR"
        AND     27H             ;CHANGE "DIR"
CDEF1:  LD      DE,(RR)
        LD      HL,(Y2)
        CP      22H
        JR      Z,CDEF3         ;CHECK IF "DIR"=2
        CP      26H
        JR      Z,CDEF4         ;CHECK IF "DIR"=6
        LD      HL,(X2)
        CP      20H
        JR      NZ,CDEF5        ;CHECK IF "DIR"=0
        SBC     HL,DE           ;X2-RR ---> 'HL'
CDEF7:  LD      DE,(Y2)         ;Y2 ---> 'DE'
        JP      CSR11
        ;
CDEF5:  ADD     HL,DE           ;X2+RR ---> 'HL'
        JR      CDEF7
        ;
CDEF4:  ADD     HL,DE           ;Y2+RR
CDEF6:  EX      DE,HL           ;Y2+-RR ---> 'DE'
        LD      HL,(X2)         ;X2 ---> 'HL'
        JP      CSR11
        ;
CDEF3:  SBC     HL,DE           ;Y2-RR
        JR      CDEF6
        ;
ACRL1:  LD      HL,(T1R)        ;LOAD RR*SIN(T1)
        LD      DE,(T2R)        ;LOAD RR*SIN(T2)
        OR      A
        SBC     HL,DE
        JR      C,ACRL10        ;CHECK IF RR*SIN(T1)<RR*SIN(T2)
        BIT     0,A
        JR      NZ,ACRL11
ACRL2:  LD      HL,0            ;LOAD "DM=0"
        CALL    CDEF13          ;SET "EAD"
        LD      HL,(T2R)        ;LOAD "DC=RR*SIN(T2)"
        CALL    C11             ;DRAW ARC
        LD      DE,0B504H       ;LOAD 2**(1/2)/2
        LD      BC,(RR)         ;LOAD (RR)
        CALL    MULTI           ;(RR)*2**(1/2)/2
        SET     6,H             ;SET "DGD" (IF NEEDED)
        LD      (VECTW+2),HL    ;STORE "DC"
        LD      A,(GR+1)        ;LOAD "DIR(T2)"
        JP      ACRL4
        ;
ACRL11: EX      DE,HL           ;LOAD "DM=RR*SIN(T2)"
ACRL12: CALL    CDEF13          ;SET "EAD"
        LD      HL,(T1R)        ;LOAD "DC=RR*SIN(T1)"
        JP      C11             ;DRAW ARC
        ;
ACRL10: BIT     0,A
        JP      NZ,ACRL3
        LD      HL,(T1R)        ;LOAD "DM=RR*SIN(T1)"
        CALL    CDEF13          ;SET "EAD"
        LD      HL,(T2R)        ;LOAD "DC=RR*SIN(T2)"
        JP      C11             ;DRAW ARC
        ;
;DRAW LINE FROM POINT ON ARC TO POINT AT CENTER
        ;
PIL:    LD      A,(COLOR)       ;LOAD COLOR
        PUSH    AF              ;PUSH COLOR
        LD      A,1
        LD      (COLOR),A       ;SET 1 TO COLOR
        LD      HL,(T2)         ;LOAD (T2)
        CALL    FOCSIN          ;RR*SIN(T2)
        LD      A,L
        OR      H
        JR      Z,PIL1          ;CHECK IF RR*SIN(T2)=0
        PUSH    HL              ;PUSH RR*SIN(T2)
        CALL    CDEF13          ;SET "EAD"
        POP     HL              ;POP RR*SIN(T2)
        DEC     HL              ;LOAD "RR*SIN(T2)-1"
        CALL    C11             ;SIMULATE DRAWING ARC TO GET "EAD"
PIL2:   IN      A,(GDCPAR)      ;GET STATUS FLAGS
        BIT     1,A
        JR      NZ,PIL2         ;CHECK IF FIFO-FULL=0
        LD      BC,0E0000H+GDCCMD;SET "CSRR" COMMAND CODE
        OUT     (C),B           ;<CSRR>
        XOR     A
        LD      H,A
        LD      D,A             ;CLEAR 'H','D'
PIL3:   IN      A,(GDCPAR)      ;GET STATUS FLAGS
        RRCA
        JR      NC,PIL3         ;CHECK IF DATA-READY=1
        IN      L,(C)           ;(EADL)
        IN      E,(C)           ;(EADH)
        IN      A,(GDCCMD)      ;(DADL)
        IN      A,(GDCCMD)      ;(DADH)
        LD      B,7             ;PRESET DOT ADDRESS
        OR      A
        JR      NZ,PIL4         ;CHECK IF DADL=0
        LD      B,0FH           ;PRESET DOT ADDRESS
        IN      A,(GDCCMD)      ;(DADH)
        ;
        ;CONVERT "EAD" TO COORDINATE
        ;
PIL4:   RLCA
        JR      C,PIL5
        DJNZ    PIL4
PIL5:   LD      A,L
        OR      A
        RLA
        RL      E
        RL      D
        RLA
        RL      E
        RL      D
        RLA
        RL      H
        RLA
        RL      H
        ADD     A,B
        LD      L,A
        LD      (X1),HL         ;SAVE (X1)
        LD      A,E
        CPL
        LD      L,A
        LD      A,D
        CPL
        AND     3
        LD      H,A
        LD      (Y1),HL         ;SAVE (Y1)
        ;
        ;END OF CONVERSION
        ;
        POP     AF              ;POP COLOR
        LD      (COLOR),A       ;RETURN COLOR
        JP      LINE            ;DRAW LINE
        ;
PIL1:   CALL    CDEF12          ;SET "EAD"
        CALL    SCSRW           ;<CSRW>,<P1-P3>
        JR      PIL2            ;START CURSOR READ
        ;
; COMMANDS AND PARAMETERS FOR INITIALIZATION
        ;
CPINIT: DB      0EH,6,3EH,0A5H  ;<SYNC>,<P1-P3>
        DB      18H,6,0FH,0     ;<P4-P7>
        DB      92H,47H,80H,46H ;<P8>,<PITCH>,<P1>,<ZOOM>
        DB      0,70H,0,0       ;<P1>,<SCROLL>,<P1-P2>
        DB      10H,60H,0,0     ;<P3-P6>
        DB      0,60H,4BH,0     ;<P7-P8>,<CSRFORM>,<P1>
        DB      40H,0,49H,0     ;<P2-P3>,<CSRW>,<P1>
        DB      0,0,4CH,10H     ;<P2-P3>,<VECTW>,<P1>
        DB      40H,40H,0,10H   ;<P2-P5>
        DB      0,0,0,0         ;<P6-P9>
        DB      0,0,4AH,0FFH    ;<P10-P11>,<MASK>,<P1>
        DB      0FFH,78H,0FFH,0FFH      ;<P2>,<TEXTW>,<P1-P2>
        DB      0FFH,0FFH,0FFH,0FFH     ;<P3-P6>
        DB      0FFH,0FFH       ;<P7-P8>
        ;
;CONSTANT FOR SIN(THETA) CALCULATION
        ;
SINK:   DB      0FAH,8EH,0AAH,2AH
        DB      0CCH,0CH
        ;
X1:     DS      2               ;FIRST COORDINATE X
Y1:     DS      2               ;FIRST COORDINATE Y
X2:     DS      2               ;SECOND COORDINATE X
Y2:     DS      2               ;SECOND COORDINATE Y
RR:     DS      2               ;RADIUS
COLOR:  DS      1               ;COLOR (MSB XXXXXBGR LSB)
T1:     DS      2               ;START ARGUMENT (INTEGRAL DEGREE)
T2:     DS      2               ;END ARGUMENT (INTEGRAL DEGREE)
GR:     DS      2               ;GENERAL REGISTER
T1R:    DS      2               ;RR*SIN(T1)
T2R:    DS      2               ;RR*SIN(T2)
COLOR1: DS      1               ;COLOR POINTER BUFFER
SYNC:   DS      9               ;<SYNC>,<P1-P8>
PITCH:  DS      2               ;<PITCH>,<P1>
ZOOM:   DS      2               ;<ZOOM>,<P1>
SCROLL: DS      9               ;<SCROLL>,<P1-P8>
CSRFOR: DS      4               ;<CSRFORM>,<P1-P3>
CSRW:   DS      4               ;<CSRW>,<P1-P3>
VECTW:  DS      12              ;<VECTW>,<P1-P11>
MASK:   DS      3               ;<MASK>,<P1-P2>
TEXTW:  DS      9               ;<TEXTW>,<P1-P8>
        ;
        END
```