お客様各位

カタログ等資料中の旧社名の扱いについて

2010年4月1日を以ってNECエレクトロニクス株式会社及び株式会社ルネサステクノロジが合併し、両社の全ての事業が当社に承継されております。従いまして、本資料中には旧社名での表記が残っておりますが、当社の資料として有効ですので、ご理解の程宜しくお願い申し上げます。

ルネサスエレクトロニクス ホームページ（http://www.renesas.com）

2010年4月1日
ルネサスエレクトロニクス株式会社

【発行】ルネサスエレクトロニクス株式会社（http://www.renesas.com）

【問い合わせ先】http://japan.renesas.com/inquiry

ご注意書き

1. 本資料に記載されている内容は本資料発行時点のものであり、予告なく変更することがあります。当社製品のご購入およびご使用にあたりましては、事前に当社営業窓口で最新の情報をご確認いただきますとともに、当社ホームページなどを通じて公開される情報に常にご注意ください。
2. 本資料に記載された当社製品および技術情報の使用に関連し発生した第三者の特許権、著作権その他の知的財産権の侵害等に関し、当社は、一切その責任を負いません。当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
3. 当社製品を改造、改変、複製等しないでください。
4. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。お客様の機器の設計において、回路、ソフトウェアおよびこれらに関連する情報を使用する場合には、お客様の責任において行ってください。これらの使用に起因しお客様または第三者に生じた損害に関し、当社は、一切その責任を負いません。
5. 輸出に際しては、「外国為替及び外国貿易法」その他輸出関連法令を遵守し、かかる法令の定めるところにより必要な手続を行ってください。本資料に記載されている当社製品および技術を大量破壊兵器の開発等の目的、軍事利用の目的その他軍事用途の目的で使用しないでください。また、当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器に使用することができません。
6. 本資料に記載されている情報は、正確を期すため慎重に作成したものですが、誤りがないことを保証するものではありません。万一、本資料に記載されている情報の誤りに起因する損害がお客様に生じた場合においても、当社は、一切その責任を負いません。
7. 当社は、当社製品の品質水準を「標準水準」、「高品質水準」および「特定水準」に分類しております。また、各品質水準は、以下に示す用途に製品が使われることを意図しておりますので、当社製品の品質水準をご確認ください。お客様は、当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途に当社製品を使用することができません。また、お客様は、当社の文書による事前の承諾を得ることなく、意図されていない用途に当社製品を使用することができません。当社の文書による事前の承諾を得ることなく、「特定水準」に分類された用途または意図されていない用途に当社製品を使用したことによりお客様または第三者に生じた損害等に関し、当社は、一切その責任を負いません。なお、当社製品のデータ・シート、データ・ブック等の資料で特に品質水準の表示がない場合は、標準水準製品であることを表します。

   標準水準： コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット

   高品質水準： 輸送機器（自動車、電車、船舶等）、交通用信号機器、防災・防犯装置、各種安全装置、生命維持を目的として設計されていない医療機器（厚生労働省定義の管理医療機器に相当）

   特定水準： 航空機器、航空宇宙機器、海底中継機器、原子力制御システム、生命維持のための医療機器（生命維持装置、人体に埋め込み使用するもの、治療行為（患部切り出し等）を行うもの、その他直接人命に影響を与えるもの）（厚生労働省定義の高度管理医療機器に相当）またはシステム等
8. 本資料に記載された当社製品のご使用につき、特に、最大定格、動作電源電圧範囲、放熱特性、実装条件その他諸条件につきましては、当社保証範囲内でご使用ください。当社保証範囲を超えて当社製品をご使用された場合の故障および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めておりますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は耐放射線設計については行っておりません。当社製品の故障または誤動作が生じた場合も、人身事故、火災事故、社会的損害などを生じさせないようお客様の責任において冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、機器またはシステムとしての出荷保証をお願いいたします。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様が製造された最終の機器・システムとしての安全検証をお願いいたします。
10. 当社製品の環境適合性等、詳細につきましては製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。お客様がかかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを固くお断りいたします。
12. 本資料に関する詳細についてのお問い合わせその他お気付きの点等がございましたら当社営業窓口までご照会ください。

注 1. 本資料において使用されている「当社」とは、ルネサスエレクトロニクス株式会社およびルネサスエレクトロニクス株式会社がその総株主の議決権の過半数を直接または間接に保有する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

アプリケーション・ノート（Ⅲ）

RENESAS

保守／廃止

μPD72123

AGDC Ⅱ

ウィンドウ編

資料番号　IEA-712（第1版）
発行年月　September 1993 P

アプリケーション・ノート（Ⅲ）

NEC

保守／廃止

μPD72123

AGDC Ⅱ

ウィンドウ編

保守／廃止

○本資料の内容は，後日変更する場合があります。
○文書による当社の承諾なしに本資料の転載複製を禁じます。
○この製品を使用したことにより，第三者の工業所有権等にかかわる問題が発生した場合，当社製品の構造製法に直接かかわるもの以外につきましては，当社はその責を負いませんのでご了承ください。
○当社は，航空宇宙機器，海底中継器，原子力制御システム，生命維持のための医療用機器などに推奨できる製品を標準的には用意しておりません。当社製品をこれらの用途にご使用をお考えのお客様，および，『標準』品質水準品を当社が意図した用途以外にご使用をお考えのお客様は，事前に販売窓口までご連絡頂きますようお願い致します。
　当社推奨の用途例
　　標準：コンピュータ，OA機器，通信機器，計測機器，工作機械，産業用ロボット，AV機器，家電等
　　特別：輸送機器（列車，自動車等），交通信号機器，防災／防犯装置等
○この製品は耐放射線設計をしておりません。

M7　92.6

本資料に掲載の応用回路および回路定数は，例示的に示したものであり，量産設計を対象とするものではありません。

なお，本製品を他の部品と組み合わせて実現するアプリケーション機能の一部が，米国CADTRAK社の米国特許 4,197,590 およびRe.31,200 等ならびにそれらの対応各国特許に関係するおそれがあります。このような特許は，他のグラフィック表示コントローラを用いても，あるいはディスクリート回路を用いても問題になり得るもので，本製品単独では解決できませんので，お客様の責任において対応策をご検討の上，アプリケーション・システムを設計していただきますようお願いいたします。

保守／廃止

巻末にアンケート・コーナを設けております。このドキュメントに対するご意見をお気軽にお寄せください。

はじめに

対象者　このマニュアルは，μPD72123の機能を理解し，それを用いたアプリケーション・プログラムを作成するユーザのエンジニアを対象とします。

目　的　このマニュアルは，μPD72123を用いて描画を行うために必要な関数を収集したアプリケーション・ノートです。次の構成におけるソフトウエア機能をユーザに理解していただくことを目的とします。

構　成　このマニュアルは，大きく分けて以下の内容で構成しております。

- 概　説
- ボードの仕様
- 各関数の説明
- コンパイル／リンク方法

読み方　このマニュアルの読者は，マイクロコンピュータの一般的知識，およびC言語に関する基礎知識を必要とします。

一通りμPD72123のソフトウエア機能を理解しようとするとき

→目次に従って読んでください。

凡　例

データ表記の重み	：左が上位桁，右が下位桁
アクティブ・ロウの表記	：$\overline{\times\times\times}$（端子，信号名称の上に上線）
注	：本文中に付けた注の説明
注意	：気をつけて読んでいただきたい内容
備考	：本文中の補足説明
数の表記	：２進数…××××または××××B
	10進数…××××
	16進数…××××H

関連資料　μPD72123に関する資料

- データ・シート（IC-7967）
- ユーザーズ・マニュアル（IEU-758）
- アプリケーション・ノート（ハードウエア編）（IEA-678）

保守／廃止

保守／廃止

目　次　要　約

保守／廃止

目　　次

表　の　目　次

保守／廃止

第１章　概　　説

１.１　本アプリケーションの概要

本アプリケーション・プログラムは，μPD72123をウィンドウ・システムに適用したアプリケーションを示したものです。

本アプリケーションでは，表示されているウィンドウの移動，リサイズ，スクロール，90°回転(ローテーション)，拡大，縮小を行うことが可能です。

このようなウィンドウのアプリケーションに要求されることは，BitBLT演算の高速性です。

本アプリケーションでは，μPD72123の強力なコピー・コマンドを用いることにより，その高速性を実現しています。

本アプリケーションは，すべてC言語を用いて書かれています。

１.２　表示画面の説明

本アプリケーションを実行すると，付録Bに示すような画面がCRTに表示されます。

表示画面は，以下の３つの項目から構成されています。

- 「Job Menu」　　：ワーク・スクリーンに表示されているウィンドウの動作を選択します。
- 「Source Screen」：源画像データを表示します（源画像1/8縮小表示)。
- 「Work Screen」　：ワーク用エリアで２つのウィンドウが表示されます。
  ジョブ・メニューで選択されたイベントの処理を行います。

次に，項目の詳細を説明します。

１.２.１　「Job Menu」

「Job Menu」は，以下の８つのイベントから成り立っています。

【EXIT】【PRIORITY】【MOVE】【RESIZE】【SCROLL】【ROTATION】【EXLARGEMENT】【SHRINK】

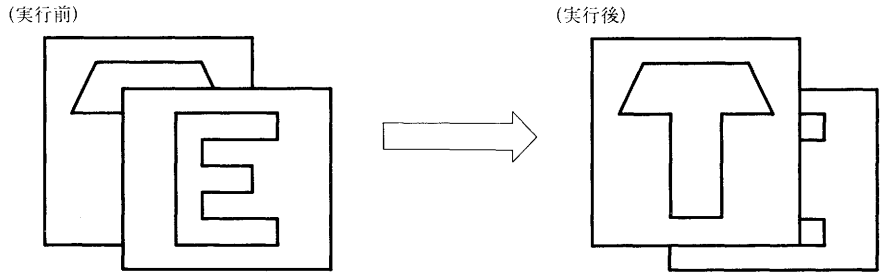
マウスでこれらのイベント画面のいずれかをクリックすることにより，そのクリックしたイベントがアクティブになります。

(1) EXIT

本アプリケーション・プログラムを終了します。

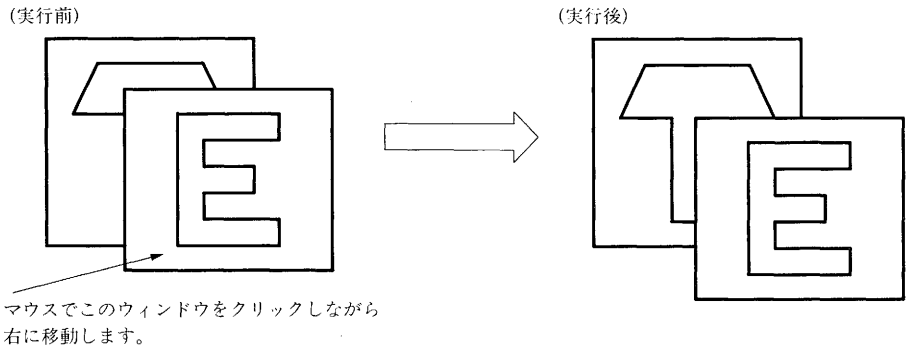
(2) PRIORITY

ワーク・スクリーンに表示されている2つのウィンドウが重なったときに，表示する優先順位を切り替えます。



(3) MOVE

マウスによってクリックされたウィンドウを移動します。

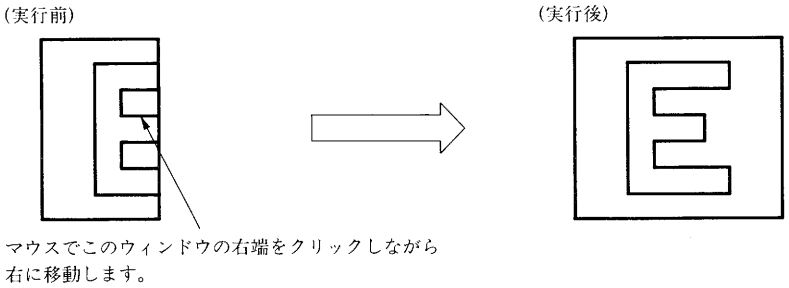


●移動させたいウィンドウに，マウス・カーサを合わせて，マウスの左ボタンをクリックしたままマウスを移動させます。

保守／廃止

(4) RESIZE

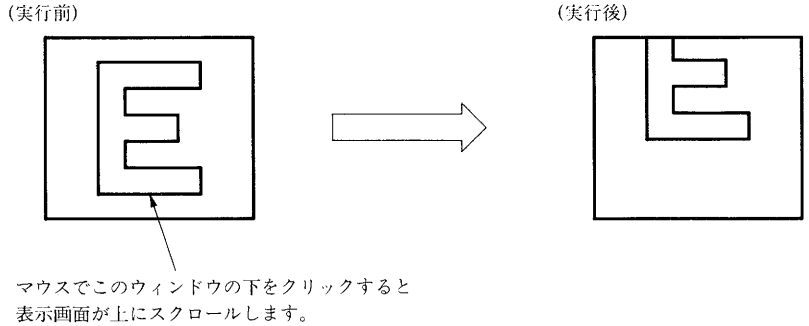
マウスによってクリックされたウィンドウの大きさを変更します。

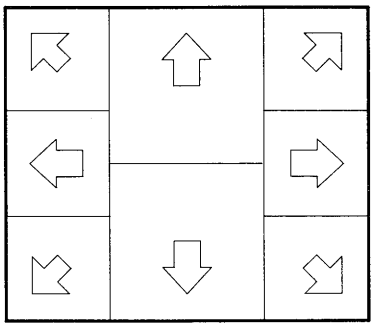


- 大きさを変更したいウィンドウの端にマウス・カーサを合わせて，マウスの左ボタンをクリックしたままマウスを移動します。
- 大きさが決定したらマウスの左ボタンを離します。

(5) SCROLL

ウィンドウの表示内容をスクロールさせます。





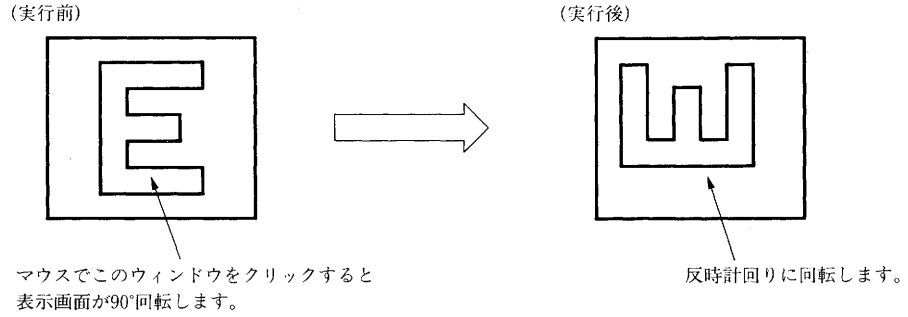
ウィンドウ内のクリック位置とスクロール方向の関係を左図に示します。

スクロールさせたいウィンドウ内の箇所をクリックします。
クリックされている間，スクロールを実行します。

保守／廃止

(6) ROTATION

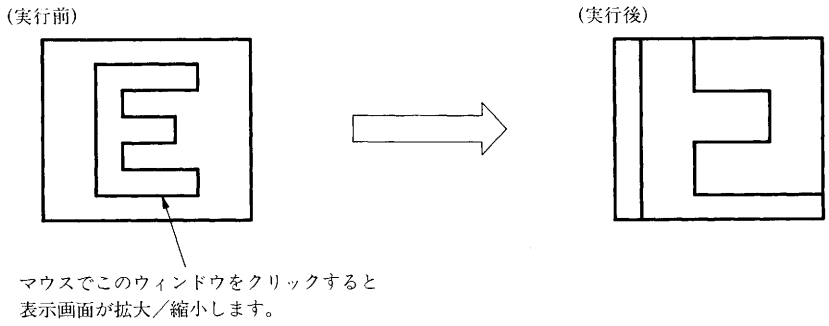
マウスによってクリックされたウィンドウの表示を90°回転します。



●回転させたいウィンドウにマウス・カーサを合わせて，マウスに左ボタンをクリックします。
クリックを行うごとにウィンドウの表示が90°回転します。

(7) ENLARGEMENT/SHRINK

マウスによってクリックされたウィンドウの表示が拡大／縮小します。



1.2.2 「Source Screen」

「Source Screen」は，画像ファイル・ローディング・エリアの全領域を，1/8に縮小してコピーしたものを表示しています。

この画面は，プログラム起動時に一度コピーされます。

表示されている画面の大きさは，240×150ドットです。

1.2.3 「Work Screen」

「Work Screen」は，1.2.1「Job Menu」において，マウスでクリックされてアクティブとなったイベントの処理を行う画面です。

プログラムを起動すると，この画面に2つのウィンドウが表示されます。

「Work Screen」の大きさは，312×356ドットです。この範囲内で，2つのウィンドウの処理を行うことになります。

1.3 画像データ

画像データは，スキャナより読み取って作成してください。

画像データのサイズは，1920×1200ドットで読み取ってください。

画像データを読み取る際には，「プレーン型」でデータを読み取ってください。

“R”，“G”，“B”のプレーンごとにデータを取り込んでください。

ファイル名は，“DEMO.IM”としてください。

第２章　ボード仕様

２.１　ハードウエア概要

μPD72123では，機能と性能評価を目的として評価ボードを設計しています。このボードを例にとり，ハードウエア設計の考え方と一例を説明します。

これよりあと，特に断りのないかぎりμPD72123評価ボードは“ボード”と記述します。

表２－１にボードの機能一覧を示します。

表２－１　機能一覧

機　　能	概　　要
表示メモリ	●RAM ２Mバイト（デュアルポート・グラフィクス・バッファ使用） ●フォントROM １Mバイト（未使用領域あり）
表示領域	●640×400ドット×4プレーン
表示色	●4096色から選んだ16色
ホスト・インタフェース	●NEC PC-98をホストとして，ホストのメモリ空間にAGDCのレジスタ群をマッピング
CRT	●解像度640（水平）×400（垂直）ドットに対応 ●ドット・クロック　21 MHz
クロック	●CLK：9 MHz ●SCLK：5.25 MHz
その他	●スレーブ・モードの評価可能

保守／廃止

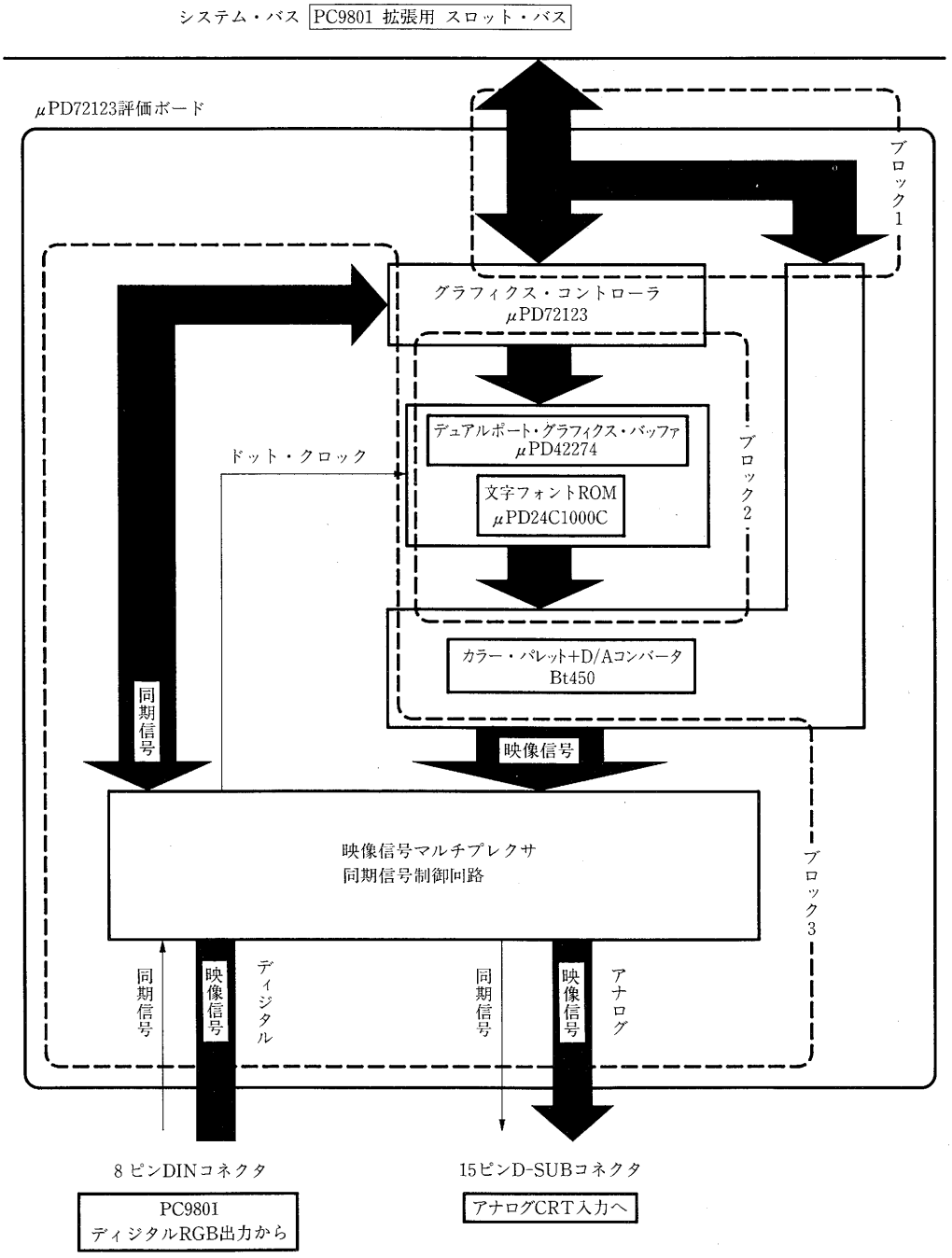
２．１．１　ボード概略図

ボードを以下のように３つのブロックに分けた概略図を示します。

ブロック１：ホスト・インタフェース回路（PC9801拡張用スロット・バス↔μPD72123，Bt450）

ブロック２：メモリ・インタフェース回路（μPD72123↔デュアルポート・グラフィクス・バッファ／フォントROM↔Bt450）

ブロック３：CRTインタフェース回路　（PC9801ディジタルRGB用CRTコネクタ，μPD72123，Bt450↔アナログRGB用CRTコネクタ）



2.1.2　ホスト・マシン

このボードは，当社のパーソナル・コンピュータPC9801本体の拡張用スロットに差し込む形態のアドオン・ボードです。以下に対象となるホスト・マシンを示します。

5インチ・フロッピィ・ディスクを搭載したマシン　：PC9801Vm/VX/RX/RA

3.5インチ・フロッピィ・ディスクを搭載したマシン：PC9801UX/UV/EX/ES

注意1.　PC9801Vmをホストとする場合，使用するソフトウエアによっては主記憶容量を640 Kバイトにする拡張メモリ・ボードが必要です。

2.　PC9801XL/XL²/RLをノーマル・モードで動作させる場合，理論上ホスト・マシンとなりえます。ただし，動作確認は行っておりません。

2.1.3　表示用CRT

PC9801用の標準的なアナログ・カラーCRTを使用します。以下に対象となるCRT（水平同期周波数24 kHz）を示します。

N5913/N5924

PC-TV451/PC-TV453　など

2.1.4　表示解像度と表示色

表示解像度はPC9801の標準解像度（640×400ドット表示）です。また，表示色は「4096色中16色同時表示」とします。

「4096色中16色同時表示」を実現するために米Brooktree社の製品であるBt450を使用します。Bt450は，DAC（ディジタル／アナログ・コンバータ）内蔵のカラー・パレットです。この製品の機能は，**「2.2.3　Bt450の機能」**を参照してください。

2.1.5　デュアルポート・グラフィクス・バッファとカラー・パレット

表示メモリは，当社のデュアルポート・グラフィクス・バッファμPD42274-10（100 nsアクセス品）を使用します。μPD42274は，256 Kビット×4 ビット構成の1 MデュアルポートDRAMです。

μPD72123は，16ビット・バス・インタフェースですので，1メモリ・プレーンあたり最低4個のμPD42274が必要です。これは，640×400ドットの画面16枚分のビット容量になります。また，表示色としては16（$=2^4$）色同時表示ですので，4メモリ・プレーン必要です。したがって，合計16個のμPD42274をボード上に搭載します。

2.2　ソフトウエア（ボードのドライバ）を設計するための情報

2.2.1　PC9801から見たI/Oマップおよびメモリ・マップ

ボードは，PC9801のCPUから見た以下の空間を占有します。

表2-2　ボードが占有するI/O空間およびメモリ空間

	条　　件	アドレス	容量(バイト)
I/O空間		00D0H	1
		00D2H	1
メモリ空間	μPD72123，Bt450に対してのみ読み書きする場合	X7F00H-X7FFFH または XFF00H-XFFFFH	256
	μPD72123，Bt450，および，μPD72123が制御している表示メモリに対して読み書きする場合	X0000H-X7FFFH または X8000H-XFFFFH	32 K

備考　Xは8H，9H，AH，BH，CH，DH，EH，FHのいずれかの値を示します。この値は00D2H番地のレジスタで設定します。

(1)　I/O空間

I/O空間内の00D0H，00D2Hの2バイトにはレジスタがマップされています。これは，μPD72123およびBt450のレジスタを，PC9801のCPUのメモリ空間内にマップするかどうかを決定するためのレジスタです。I/O空間内の00D0H，00D2H番地を使用する他のボードと共存(共に拡張用スロットに挿入した状態にすること）はできません。

(2)　メモリ空間

μPD72123およびBt450のレジスタは，「メモリ・マップトI/O」にしてあります。80000H番地以上，FFFFFH番地以下の空間中の一定領域を，μPD72123およびBt450のインタフェース領域(レジスタ・ウィンドウ)として割り当ててください。どの領域にμPD72123およびBt450のレジスタをマップするかは，00D2H番地のレジスタでプログラマブルに設定できます(**「2.2.1(2)(b)　00D2H番地のレジスタ」**参照)。

μPD72123に搭載されている表示メモリは，PC9801のCPUの空間上にマップすることができます。この場合，80000H番地以上，FFFFFH番地以下の空間中の一定領域を表示メモリのインタフェース領域（メモリ・ウィンドウ）として割り当ててください。CPUの空間内のどの領域に表示メモリをマップするかは，00D2H番地のレジスタでプログラマブルに設定できます。

また，表示メモリ空間のどの領域をCPUの空間内にマップするかは，μPD72123のBANKレジスタでプログラマブルに設定できます。

なお，レジスタ・ウィンドウ，メモリ・ウィンドウは，PC9801のリセットによりクローズされます。したがって，電源投入後，ボードはスリープ状態（メモリ・マップに関してソフトウエア的に見れば，ボードは拡張バス・スロットに挿入されていないのと同じ状態）になります。

(a)　00D0H番地のレジスタ

ボードのアクティブ／インアクティブは，00D0H番地（I/O空間）のレジスタで制御します。このレジスタの機能詳細を示します（MASTER，IRE，DMEについては，**表2－4，表2－5，表2－6**を参照）。

表2－3　00D0H番地（I/O空間）レジスタ

		ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0
00D0H番地レジスタ	リード時	×	×	×	×	×	×	×	MASTER
	ライト時	×	×	×	×	×	×	DME	IRE

×：don't care（読み出し時不定です。書き込み時は，0を書き込んでください。）

表2－4　MASTERの機能

	読み出し値	機　　能
MASTER	0	評価ボードのディジタルRGB入力コネクタにPC9801からのケーブルが接続されています（+12 Vが供給されています）。このため，μPD72123はスレーブ・モードで動作します。
	1	評価ボードのディジタルRGB入力コネクタにPC9801からのケーブルが接続されていません（+12 Vが供給されていません）。このため，μPD72123はマスタ・モードで動作します。

表2－5　IREの機能

	書き込み値	機　　能
IRE Internal Register Enable	0	μPD72123のチップ・セレクト信号$\overline{\text{CSIR}}$，および，Bt450のチップ・セレクト信号$\overline{\text{CS}}$を常にインアクティブとします（μPD72123，Bt450のレジスタをCPUの空間上にメモリ・マップしません）。
	1	00D2H番地のレジスタで定義された番地（メモリ内）をアクセスした場合において，μPD72123のチップ・セレクト信号$\overline{\text{CSIR}}$，および，Bt450のチップ・セレクト信号$\overline{\text{CS}}$をアクティブとします（μPD72123，Bt450のレジスタをCPUの空間上にメモリ・マップします）。

表2－6　DMEの機能

	書き込み値	機　　能
DME Display Memory Enable	0	μPD72123のチップ・セレクト信号$\overline{\text{CSDM}}$を常にインアクティブとします（表示メモリ（μPD72123管理下のメモリ）をCPUの空間上にメモリ・マップしません）。
	1	00D2H番地のレジスタで定義された番地（メモリ内）をアクセスした場合において，μPD72123のチップ・セレクト信号$\overline{\text{CSDM}}$をアクティブとします（表示メモリ（μPD72123管理下のメモリ）をCPUの空間上にメモリ・マップします）。

IRE，DMEの各フラグは，次のような場合に０にリセットされます。

- PC9801に電源を投入する。
- PC9801のリセット・ボタンを押す。

これは，PC9801のリセットによってボードがスリープ状態になることを意味します。

(b)　00D2H番地のレジスタ

00D2H番地(I/O空間)のレジスタは，μPD72123，Bt450のレジスタおよび表示メモリ(μPD72123が管理するメモリ)をPC9801のメモリ空間内のどこにマップするかを制御します。このレジスタの機能詳細を示します。

表2－7　00D2H番地（I/O空間）レジスタ

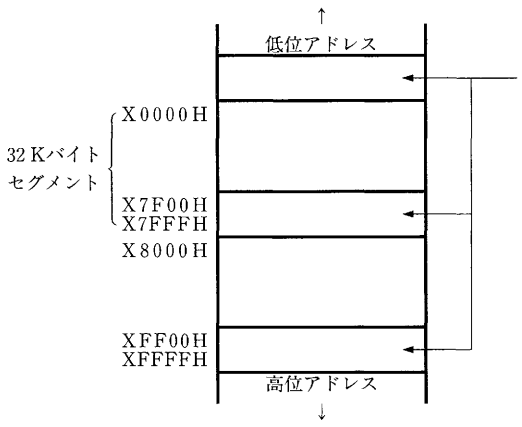
00D2H番地レジスタ（ライト・オンリ）		ビット7	ビット6	ビット5	ビット4	ビット3	ビット2	ビット1	ビット0
	リード時	← 不　定 →							
	ライト時	← VSEG →				← RSEG →			

以下の表にRSEGとVSEGの機能を説明します。なお，表の次の図は表の内容をわかりやすく示したものです。

表2－8　RSEGの機能

	機　　能
RSEG Register SEGment	μPD72123，および，Bt450のレジスタをPC9801のメモリ空間内のどの領域にマップするかを決定します。CPUが出力するアドレスのビット18-ビット15の値と，RSEGに設定された4ビットとが下記のように比較されます。 アドレス・ビット→19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 アドレス値　→1 Y Y Y Y 1 1 1 1 1 1 1 0 X X X X X X X X：don't care　Y：RSEGに設定された4ビット 比較結果が真であり，かつ，00D0H番地レジスタのIREフラグが1であれば，μPD72123のチップ・セレクト信号$\overline{\text{CSIR}}$がアクティブとなります。 アドレス・ビット→19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 アドレス値　→1 Y Y Y Y 1 1 1 1 1 1 1 1 X X X X X X X X：don't care　Y：RSEGに設定された4ビット 比較結果が真であり，かつ，00D0H番地レジスタのIREフラグが1であれば，Bt450のチップ・セレクト信号$\overline{\text{CS}}$がアクティブとなります。

保守／廃止



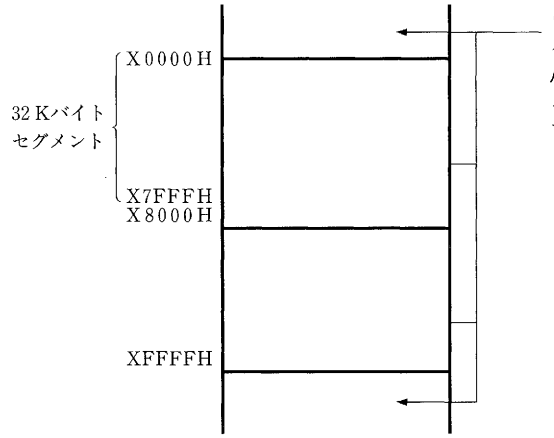
RSEG値	レジスタ・ウィンドウ
0H	87F00H-87FFFH
1H	8FF00H-8FFFFH
2H	97F00H-97FFFH
3H	9FF00H-9FFFFH
4H	A7F00H-A7FFFH
5H	AFF00H-AFFFFH
6H	B7F00H-B7FFFH
7H	BFF00H-BFFFFH
8H	C7F00H-C7FFFH
9H	CFF00H-CFFFFH
AH	D7F00H-D7FFFH
BH	DFF00H-DFFFFH
CH	E7F00H-E7FFFH
DH	EFF00H-EFFFFH
EH	F7F00H-F7FFFH
FH	FFF00H-FFFFFH

備考1.　Bt450のレジスタは，実際には２バイトであり，XXX80HとXXX82Hにマップされます。

2.　Xは8H，9H，AH，BH，CH，DH，EH，FHのいずれかの値を示します。

表２－９　VSEGの機能

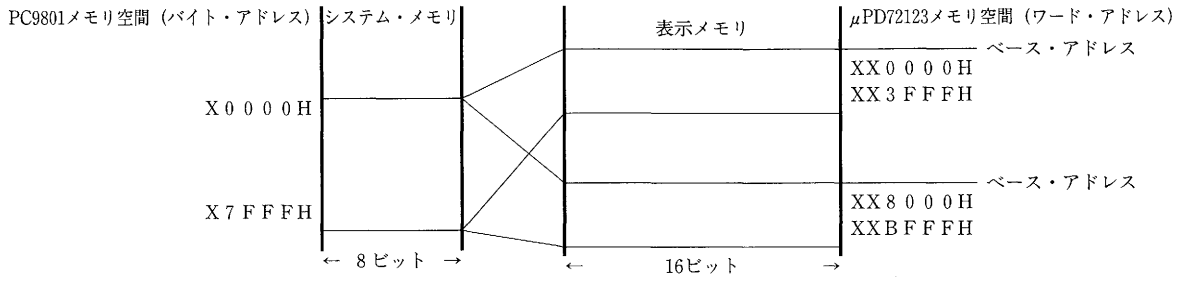
	機　　能
VSEG Video RAM SEGment	μPD72123，および，Bt450のレジスタをPC9801のメモリ空間内のどの領域にマップするかを決定します。CPUが出力するアドレスのビット18-ビット15の値と，VSEGに設定された４ビットとが下記のように比較されます。 アドレス・ビット→19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 アドレス値　　→1 Y Y Y Y X X X X X X X X X X X X X X X：don't care　　Y：VSEGに設定された４ビット 比較結果が真であり，かつ，00D0H番地レジスタのDMEフラグが１であれば，μPD72123のチップ・セレクト信号$\overline{\text{CSDM}}$がアクティブとなります。



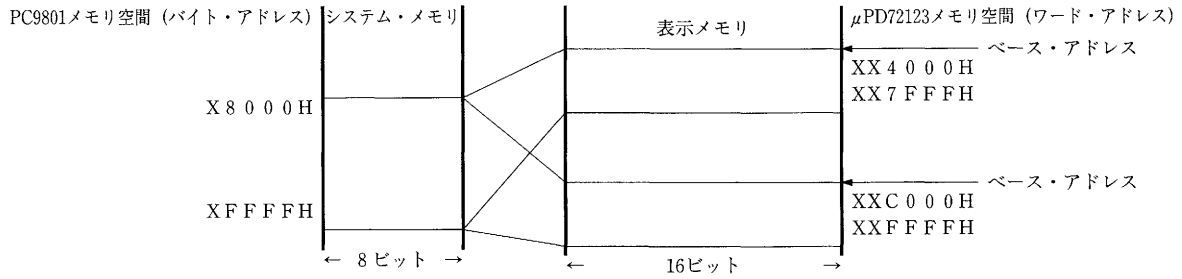
VSEG値	メモリ・ウィンドウ
0H	80000H-87FFFH
1H	88000H-8FFFFH
2H	90000H-97FFFH
3H	98000H-9FFFFH
4H	A0000H-A7FFFH
5H	A8000H-AFFFFH
6H	B0000H-B7FFFH
7H	B8000H-BFFFFH
8H	C0000H-C7FFFH
9H	C8000H-CFFFFH
AH	D0000H-D7FFFH
BH	D8000H-DFFFFH
CH	E0000H-E7FFFH
DH	E8000H-EFFFFH
EH	F0000H-F7FFFH
FH	F8000H-FFFFFH

注意　Xは8H，9H，AH，BH，CH，DH，EH，FHのいずれかの値を示します。

【X0000H-X7FFFHをメモリ・ウィンドウとした（VSEGに偶数値を設定した）場合】



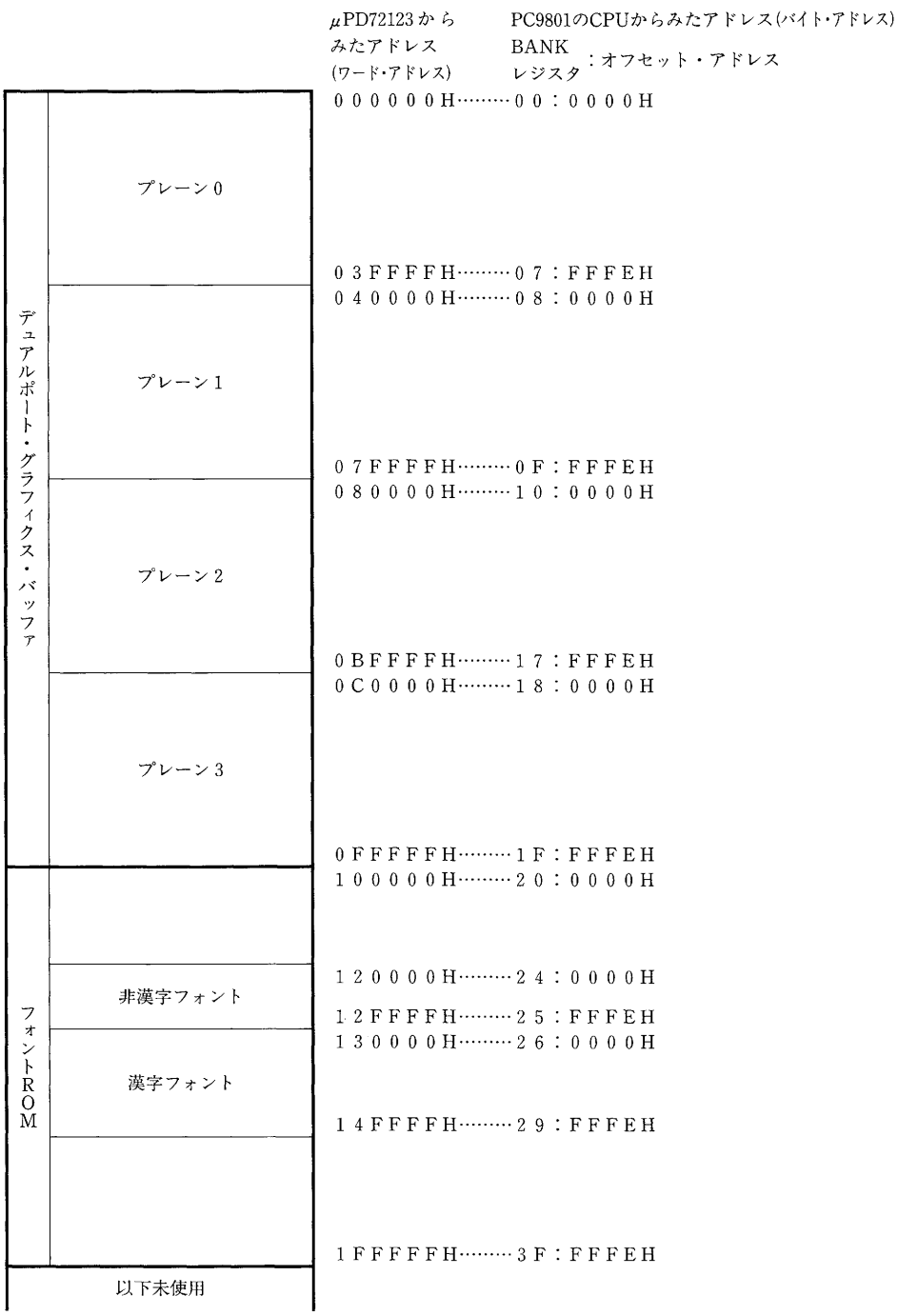
【X0000H-X7FFFHをメモリ・ウィンドウとした（VSEGに奇数値を設定した）場合】



注意　ベース・アドレスは，μPD72123内部のBANKレジスタ（８ビット）とCTRL2レジスタのBANKXフラグ（１ビット）との合計９ビットで決定されます。

２.２.２　μPD72123から見たメモリ・マップ

μPD72123から見た表示メモリ・マップを以下に示します。



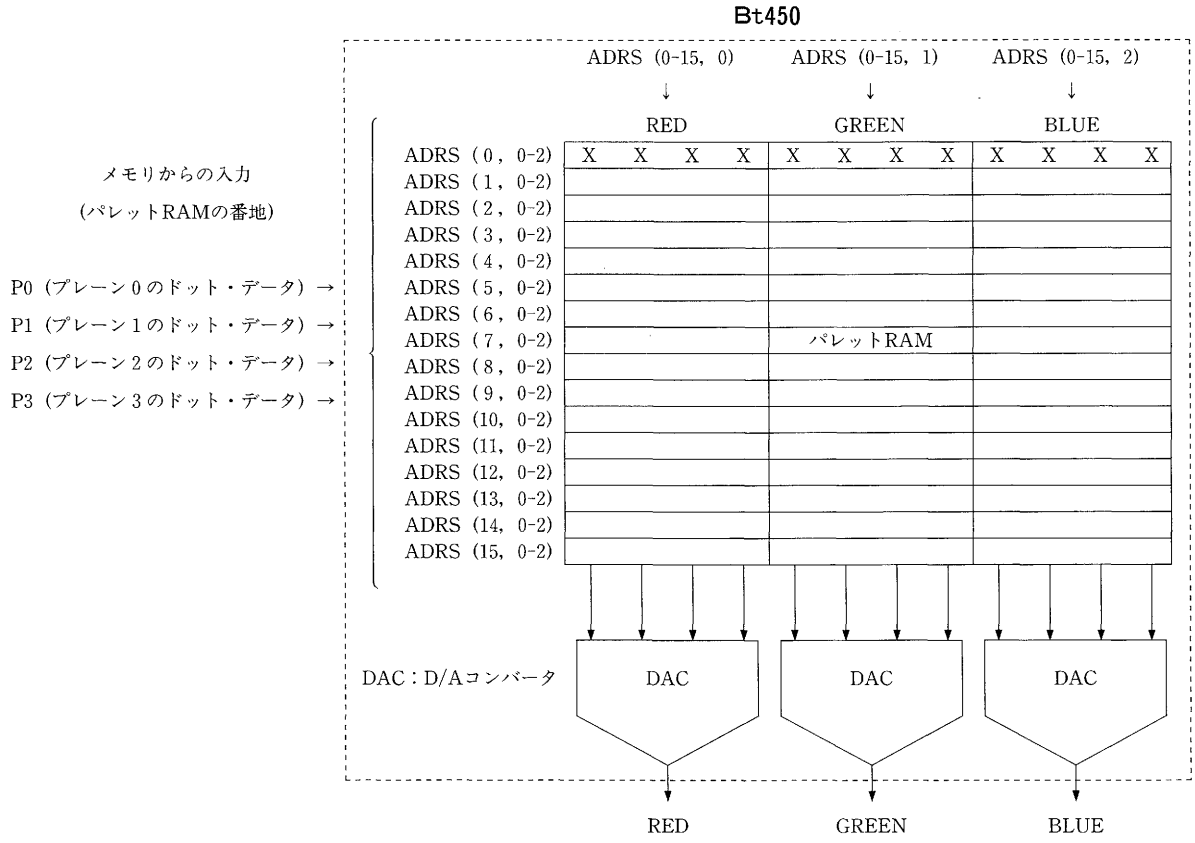
備考　フォント・データのゴーストが読み出される領域

2.2.3　Bt450の機能

Bt450は，米Brooktree社の商品であり，「カラー・パレット＋D/Aコンバータ」機能を持つチップです。12ビット×16ワードのパレットRAMを内蔵しているため，4096色中16色同時表示が可能です。

Bt450内部は図2－1のようになっています。

図2－1　Bt450の内部機能ブロック



Bt450は1ピクセルあたり4ビット（プレーン0からプレーン3まで）の情報が入力されると，この情報をインデクス（パレットRAMの参照アドレス）として使用します。インデクスによってアドレスされたRAMの内容が12ビットの色情報（RGB各4ビット・データ）としてD/Aコンバータに入力されます。3台のD/Aコンバータは入力された4ビット・データ（0から15までの16レベル）をアナログに変換し，それぞれ赤，青，緑の輝度情報として出力します。

次にPC9801がBt450に対してパレット値を設定する方法について述べます。

PC9801から見た場合，Bt450は２つのレジスタを持っているように見えます。前述の方法により，レジスタ・ウィンドウをオープンにした場合，オフセット・アドレス0080H番地と0082H番地に各１バイトずつレジスタがマップされます。

表２－10　Bt450のレジスタ

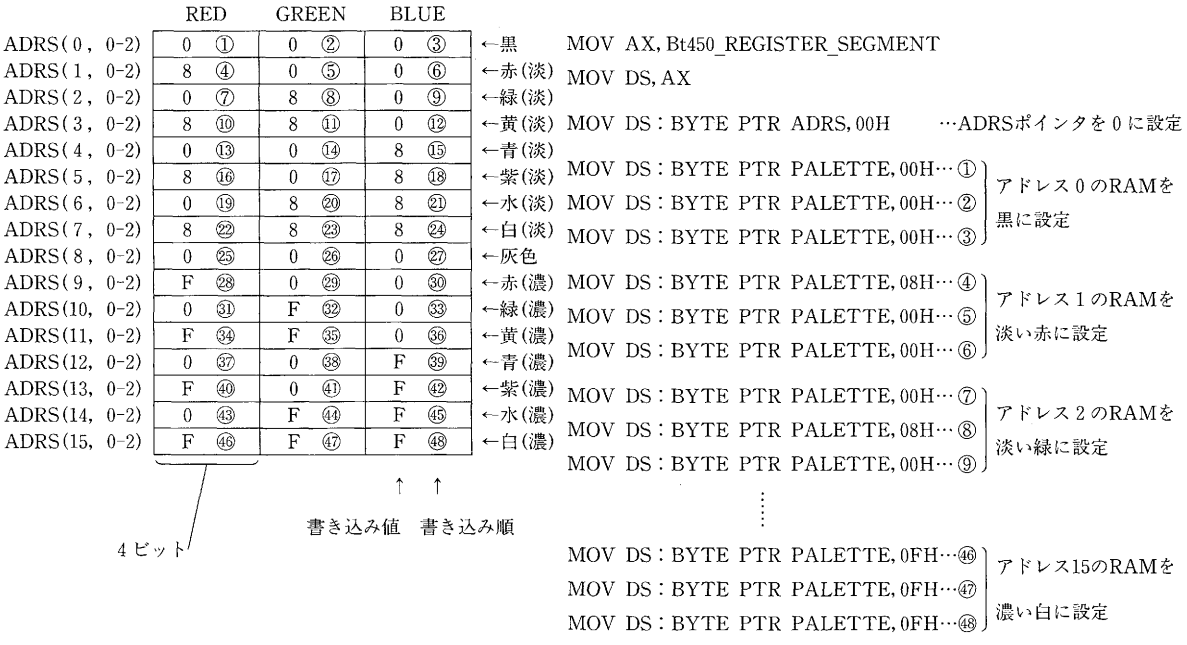
オフセット・アドレス	機　　能	
0000H-007FH	μPD72123のレジスタ群	
0080H	Bt450のアドレス・レジスタADRS（５ビット・レジスタ）	データ・レジスタに設定した値をどの番地のパレットRAMに書き込むかを指定
0082H	Bt450のデータ・レジスタPALETTE（４ビット・レジスタ）	パレットRAM設定値を書き込む

Bt450のデータ・バスはPC9801のデータ・バスの下位５ビットに接続しています。ADRSにはオート・インクリメント（PALETTEレジスタに対するライト動作によりADRSレジスタの内容が+1される）の機能があります。以下に設定例を示します。

【設定例】

パレットRAMに下記の設定をするためには，下記の①から順番にデータ・レジスタに書き込みます。

図２－２　Bt450のレジスタ設定例



第３章　各関数の説明

３.１　デモ・メイン関数（demo.c）

３.１.１　関数一覧

No.	ラベル名	機　　能
P-1	main ()	メイン関数
P-2	sys_init	システムの初期化
P-3	menu_pro	メニューのクリック処理
P-4	source_pro	ソース・スクリーンのクリック処理
P-5	work_pro	ワーク・スクリーンのクリック処理
P-6	priority_pro	ウィンドウのプライオリティ処理
P-7	move_pro	ウィンドウのMOVE処理
P-8	resize_pro	ウィンドウのRESIZE処理
P-9	scroll_pro	ウィンドウのSCROLL処理
P-10	rotation_pro	ウィンドウの90°回転処理
P-11	enlarge_pro	ウィンドウの拡大処理
P-12	shrink_pro	ウィンドウの縮小処理
P-13	load_pro	画像ファイルのローディング
P-14	filechk_pro	画像ファイルのチェック
P-15	put_pro	画像ファイルのセット
P-16	source_copy	ソース・スクリーンのコピー
P-17	screen_pro	表示エリア作成
P-18	title_dsp	タイトルの作成
P-19	work_dsp	ワーク・スクリーンの作成
P-20	source_dsp	ソース・スクリーンの作成
P-21	menu_dsp	メニュー・スクリーンの作成
P-22	cur_setting	カーサの設定
P-23	cur_input	カーサの入力
P-24	cur_locate	カーサの位置設定
P-25	chk_menu_no	メニュー番号の取得
P-26	chk_win_position	ウィンドウ内のクリック位置の取得
P-27	chk_win_line	ウィンドウ内のクリック枠の取得
P-28	view_area	ウィンドウの表示領域

３．１．２　各関数の説明

以下に，メイン部の各関数について説明します。

(1)　main ()

【機　　能】

コマンド起動時の呼び出し関数です。以下のことを実行します。

a)　PC-9801側の表示を初期化

b)　システム初期化関数の呼び出し

c)　マウスの状態を取得して以下の処理を行います。

- ●メニューがクリックされたとき　　：menu_pro () を実行
- ●ワーク・スクリーンがクリックされたとき：work_pro () を実行
- ●ソース・スクリーンがクリックされたとき：source_pro () を実行

d)　イベントが［EXIT］の場合は，ディスプレイ表示をOFFにしてEXITします。

【関　数　名】

main

【書　　式】

int main (argc, argv)

【パラメータ】

int argc　　：コマンド起動パラメータ数

uchar ＊argv []　：パラメータ・ポインタ

【戻　り　値】

なし

(2) sys_init

【機　　能】

a) 各構造体の初期化を行います。

b) 表示画面を作成して，ウィンドウのセットアップを行います。

c) ウィンドウに表示するための画像ファイルのロードとセットを行います。

d) マウスの初期化を行います。

【関 数 名】

sys_init

【書　　式】

int sys_init (agrc, agrv)

【パラメータ】

int agrc　　：システム・パラメータ

uchar *argv []　：システム・パラメータ・ポインタ

【戻 り 値】

−1：エラー

0：OK

(3) menu_pro

【機　　能】

メニュー・スクリーンをクリックしたときの処理をします。

このアプリケーションでは，ウィンドウのMOVE，SCROLL，RESIZE，ROTATION，ENLARGEMENT，SHRINK，PRIORITYの処理を行うことができます。その際のメニュー画面の制御を以下のように行います。

a)　前回アクティブであったイベントの枠を元の表示に戻し，今回クリックされたイベントの枠をアクティブ状態にします。

【関　数　名】

menu_pro

【書　　式】

void menu_pro（m）

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(4)　source_pro

【機　　能】

ソース・スクリーンをクリックしたときの処理をします。

マウス型構造体で示される座標（m→x，m→y）をソース・エリアの座標に変換して，高プライオリティ状態にあるウィンドウの表示内容を変更します。

【関　数　名】

source_pro

【書　　式】

void source_pro（m）

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(5)　work_pro

【機　　能】

マウスによってクリックされたワーク・スクリーンを処理します。

イベントの状態によりそれぞれの処理をします。

【関　数　名】

work_pro

【書　　式】

void work_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

なし

(6) priority_pro

【機　　能】

ウィンドウの表示優先順位を入れ替えます。

【関 数 名】

priority_pro

【書　　式】

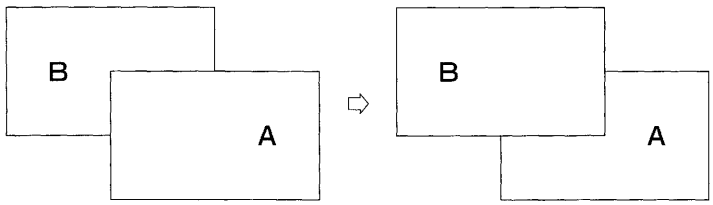
void priority_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし



(7)　move_pro

【機　　能】

マウスによってクリックされたウィンドウの表示位置を移動させます。

【関　数　名】

move_pro

【書　　式】

void move_pro（m）

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(8) resize_pro

【機　　能】

マウスによってクリックされたウィンドウの大きさを変更します。

【関 数 名】

resize_pro

【書　　式】

void resize_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(9)　scroll_pro

【機　　能】

マウスによってクリックされたウィンドウの表示内容をスクロールさせます。

【関　数　名】

scroll_pro

【書　　式】

void scroll_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

なし

保守／廃止

⑽ rotation_pro

【機　　能】

マウスでクリックされたウィンドウの表示内容を90°回転して表示します。

【関 数 名】

rotation_pro

【書　　式】

void rotation_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(11)　enlarge_pro

【機　　能】

マウスによってクリックされたウィンドウの表示内容を拡大させます。

【関　数　名】

enlarge_pro

【書　　式】

void enlarge_pro（m）

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(12) shrink_pro

【機　　能】

マウスによってクリックされたウィンドウの表示内容を縮小させます。

【関 数 名】

shrink_pro

【書　　式】

void shrink_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

⑬ load_pro

【機　　能】

画像ファイルをロードして，ソース・エリアにセットします。

【関 数 名】

load_pro

【書　　式】

int load_pro（fname）

【パラメータ】

uchar ＊fname：画像ファイル名

【戻 り 値】

0：エラー

1：正　常

保守／廃止

⑭ filechk_pro

【機　　能】

画像ファイルのヘッダー部を読み込み，データ形式をチェックします。

【関 数 名】

filechk_pro

【書　　式】

int filechk_pro (fd)

【パラメータ】

int fd：ファイル・ディスクリプタ

【戻 り 値】

0：正　常

−1：エラー

保守／廃止

⑮ put_pro

【機　　能】

画像ファイルをPUTコマンドを使用してソース・エリアにセットします。

【関 数 名】

put_pro

【書　　式】

int put_pro (fd)

【パラメータ】

int fd：ファイル・ディスクリプタ

【戻 り 値】

0：エラー

1：正　常

保守／廃止

(16) source_copy

【機　　能】

ソース・エリアの内容をソース・スクリーンへ表示させます。

【関 数 名】

source_copy

【書　　式】

void source_copy ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(17) screen_pro

【機　　能】

オン・スクリーンの画面を作成します。

【関 数 名】

screen_pro

【書　　式】

void screen_pro (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(18) title_dsp

【機　　能】

本ウィンドウ・プログラムのタイトルを作成表示します。

【関 数 名】

title_dsp

【書　　式】

void title_dsp ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(19) work_dsp

【機　　能】

ワーク・スクリーンを作成表示します。

【関 数 名】

work_dsp

【書　　式】

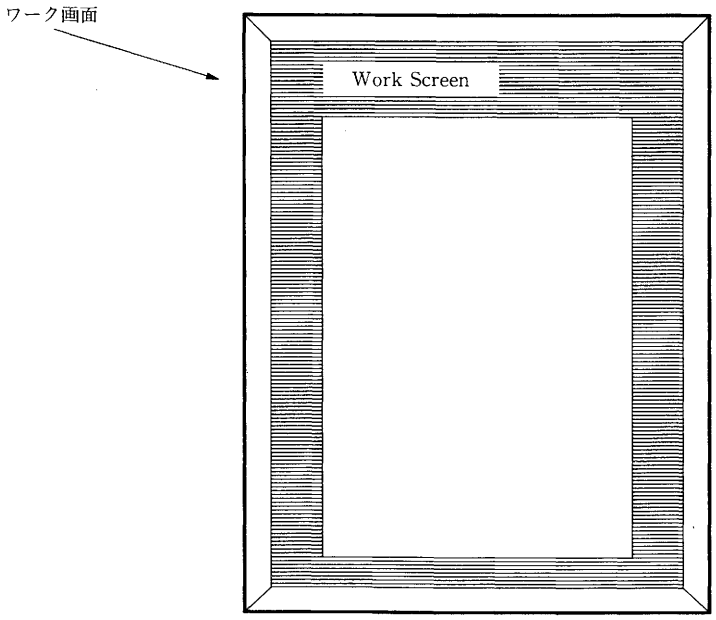
void work_dsp ()

【パラメータ】

なし

【戻 り 値】

なし



保守／廃止

(20) source_dsp

【機　　能】

ソース・スクリーンを作成表示します。

【関 数 名】

source_dsp

【書　　式】

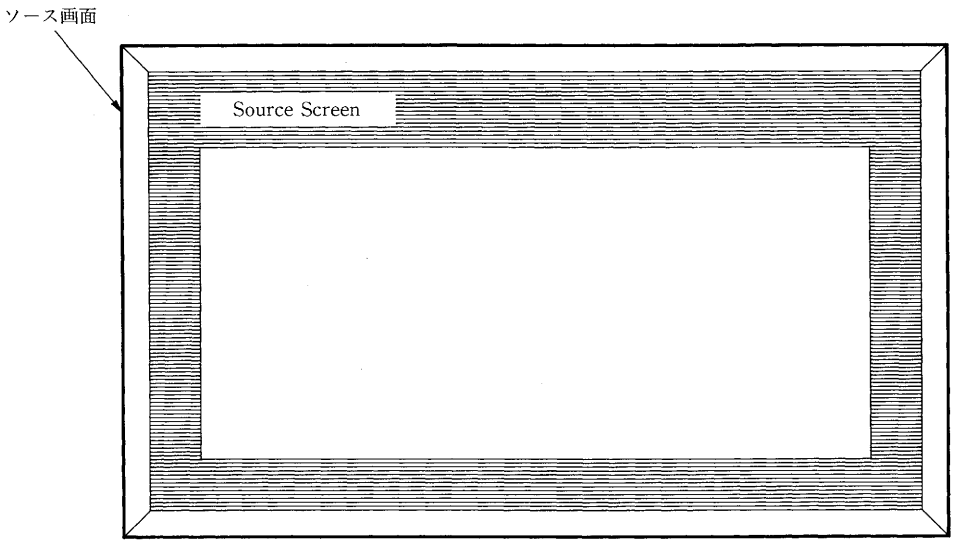
void source_dsp ()

【パラメータ】

なし

【戻 り 値】

なし



(21) menu_dsp

【機　　能】

メニュー・スクリーンを作成します。

【関 数 名】

menu_dsp

【書　　式】

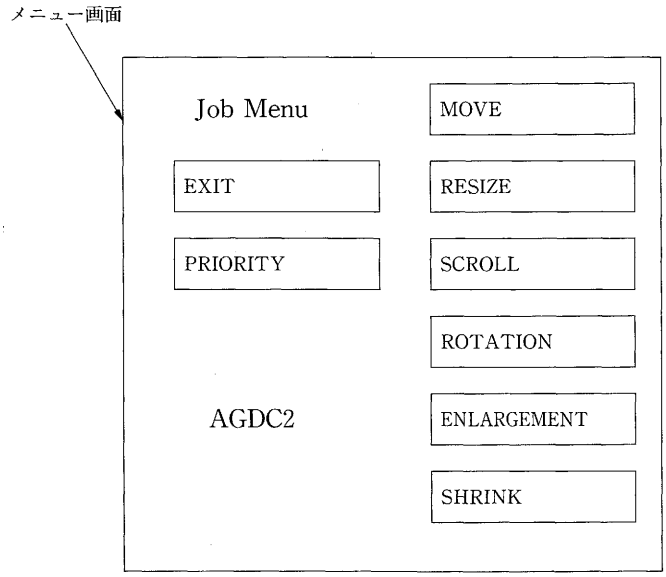
void menu_dsp (m)

【パラメータ】

Mouse *m：マウス型構造体ポインタ

【戻 り 値】

なし



保守／廃止

(22) cur_setting

【機　　能】

マウス・カーサの形状をセットします。

【関 数 名】

cur_setting

【書　　式】

int cur_setting ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(23) cur_input

【機　　能】

マウスの状態を調べます。

【関　数　名】

cur_input

【書　　式】

int cur_input (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

0：マウスのクリック・ボタンは押されていない

1：各スクリーンをクリック中

−1：意味のない場所をクリック中

保守／廃止

(24) cur_locate

【機　　能】

マウス・カーサの位置を移動させます。

【関 数 名】

cur_locate

【書　　式】

void cur_locate (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

⑵5) chk_menu_no

【機　　能】

マウス・カーサの位置が，メニューの各要素上にあるかを調べます。

【関　数　名】

chk_menu_no

【書　　式】

void chk_menu_no (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻　り　値】

menu≠0：メニューの番号

menu＝0：該当なし

保守／廃止

(26) chk_win_position

【機　　能】

ウィンドウのどの位置でクリックされているかを調べます。

【関 数 名】

chk_win_position

【書　　式】

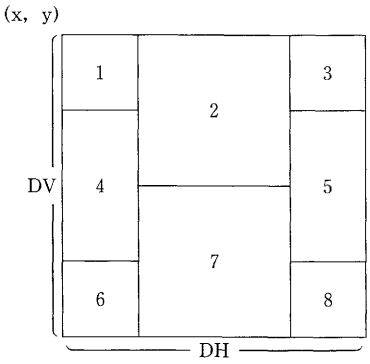
int chk_win_position (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

−1：該当なし



保守／廃止

(27) chk_win_line

【機　　能】

マウスによってクリックされている場所が，ウィンドウ中のどの辺に位置しているかを調べます。

【関 数 名】

chk_win_line

【書　　式】

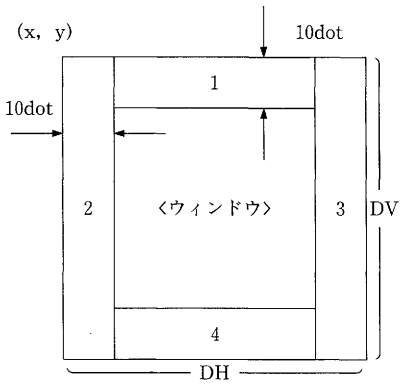
int chk_win_line (m)

【パラメータ】

Mouse ＊m：マウス型構造体ポインタ

【戻 り 値】

−1：該当なし



保守／廃止

㉘ view_area

【機　　能】

指定ウィンドウの表示領域を示す枠を表します。

【関 数 名】

view_area

【書　　式】

viod view_area (no)

【パラメータ】

int no：ウィンドウ番号

【戻 り 値】

なし

保守／廃止

３.２　ウィンドウ制御系関数（win.c）

３.２.１　関数一覧

No.	ラベル名	機　　能
W-1	Wdisplay	ディスプレイ表示の開始
W-2	Woffdisplay	ディスプレイ表示の終了
W-3	Wsetup	ウィンドウのセットアップ
W-4	Wopen	ウィンドウのオープン
W-5	Wchgpriority	ウィンドウのプライオリティの入れ替え
W-6	Whighpriority	ウィンドウのプライオリティの設定（高）
W-7	Wlowpriority	ウィンドウのプライオリティの設定（低）
W-8	Wgetpriority	ウィンドウのプライオリティ状態の取得
W-9	Wmove	ウィンドウの表示位置変更
W-10	Wresize	ウィンドウ・サイズの変更
W-11	Wscroll	ウィンドウのスクロール
W-12	Wrotation	ウィンドウの回転処理
W-13	Wenlarge	ウィンドウの拡大処理
W-14	Wshrink	ウィンドウの縮小処理
W-15	Wsrcchk	ウィンドウのソース座標領域のチェック
W-16	Wclick	ウィンドウ・クリックのチェック

保守／廃止

３.２.２　各関数の説明

以下に，ウィンドウ関係の各関数について説明します。

(1)　Wdisplay

【機　　能】

ディスプレイの表示をONにします。

【関　数　名】

Wdisplay

【書　　式】

void Wdisplay (D)

【パラメータ】

Display ＊D：ディスプレイ型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(2) Woffdisplay

【機　　能】

μPD72123の表示をOFFにします。

【関 数 名】

Woffdisplay

【書　　式】

void Woffdisplay ()

【パラメータ】

なし

【戻 り 値】

なし

(3) Wsetup

【機　　能】

指定パラメータによるウィンドウを定義します。

ワーク・エリアに，Wopenのときに転送源となるデータをコピーしておきます。

【関 数 名】

Wsetup

【書　　式】

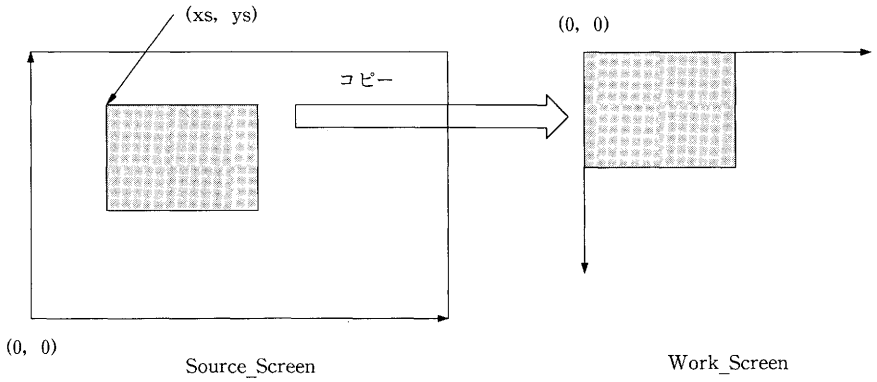
int Wsetup (W)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

−1：error（ウィンドウ定義に失敗）



ワーク・エリアに，Wopenのときに転送源となるデータをコピーしておきます。

保守／廃止

(4) Wopen

【機　　能】

指定ウィンドウを開きます。

【関 数 名】

Wopen

【書　　式】

void Wopen (W)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(5) Wchgpriority

【機　　能】

ウィンドウの表示優先順位を入れ替えます。

【関 数 名】

Wchgpriority

【書　　式】

void Wchgpriority (w1, w2)

【パラメータ】

Window ＊w1：ウィンドウ型構造体ポインタ

Window ＊w2：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(6) Whighpriority

【機　　能】

指定ウィンドウの表示優先順位を高くします。

【関 数 名】

Whighpriority

【書　　式】

void Whighpriority (w)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(7)　Wlowpriority

【機　　能】

指定ウィンドウの表示優先順位を低くします。

【関　数　名】

Wlowpriority

【書　　式】

void Wlowpriority (w)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻　り　値】

なし

保守／廃止

(8) Wgetpriority

【機　　能】

プライオリティ・フラグにより，ウィンドウの番号を返します。

【関 数 名】

Wgetpriority

【書　　式】

int Wgetpriority (pri, cpy)

【パラメータ】

Copy ＊cpy：コピー型構造体ポインタ

int pri　　：プライオリティ・フラグを示します。

【戻 り 値】

なし

保守／廃止

(9)　Wmove

【機　　能】

指定ウィンドウの表示位置を変更します。

【関　数　名】

Wmove

【書　　式】

void Wmove (w, mx, my)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

int　mx　　：水平方向の移動量

int　my　　：垂直方向の移動量

【戻　り　値】

なし

保守／廃止

⑽ Wresize

【機　　能】

指定ウィンドウの大きさを変更します。

【関 数 名】

Wresize

【書　　式】

int Wresize (w, mode, m)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

int mode　：リサイズする辺の種類

int m　　：大きさの変更量

【戻 り 値】

大きさの変更量 (dot)

保守／廃止

(11) Wscroll

【機　　能】

ウィンドウの表示領域を移動させます。

【関 数 名】

Wscroll

【書　　式】

int Wscroll (w, dx, dy)

【パラメータ】

Window *w：ウィンドウ型構造体ポインタ

int dx　　：X方向の移動量

int dy　　：Y方向の移動量

【戻 り 値】

なし

保守／廃止

(12) Wrotation

【機　　能】

ウィンドウ内の表示データを90°回転させます。

【関 数 名】

Wrotation

【書　　式】

void Wrotation (w)

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(13) Wenlarge

【機　　能】

ウィンドウ内の表示データを拡大します。

【関 数 名】

Wenlarge

【書　　式】

void Wenlarge（w）

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

⑭ Wshrink

【機　　能】

ウィンドウ内の表示データを縮小します。

【関 数 名】

Wshrink

【書　　式】

void Wshrink（w）

【パラメータ】

Window ＊w：ウィンドウ型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(15) Wsrcchk

【機　　能】

ソース・エリアの範囲をチェックします。

【関 数 名】

Wsrcchk

【書　　式】

int Wsrcchk (w, mode, m)

【パラメータ】

Window *w：ウィンドウ型構造体ポインタ

int mode　：辺の種類

int m　　：移動量

【戻 り 値】

移動可能量 (dot)

保守／廃止

⑯　Wclick

【機　　能】

現在指定されているカーサの位置がウィンドウ上であるかをチェックします。

【関　数　名】

Wclick

【書　　式】

int Wclick（m）

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻　り　値】

0：カーサがウィンドウ上にない

1：カーサがウィンドウ上にある

３.３　マウス系関数（mouse.c）

３.３.１　関数一覧

No.	ラベル名	機　　能
M-1	Minit	マウスの初期化
M-2	Mstatus	マウス状態の取得
M-3	Mlocate	マウス・カーサの位置設定
M-4	Mclickoff	クリックのリリース・チェック
M-5	Mbell	クリック音
M-6	Mclose	マウスのオフ状態

保守／廃止

３.３.２　各関数の説明

以下にマウス関係の各関数について説明します。

(1) Minit

【機　　能】

マウスを初期化します。マウス・ドライバが組み込み済みかどうかをチェックし，ミッキー・ドッド比，マウス形状を設定します。

【関　数　名】

Minit

【書　　式】

int Minit (m)

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻　り　値】

－1：マウス・ドライバが組み込まれていない

0：マウス・ドライバが組み込まれている

保守／廃止

(2) Mstatus

【機　　能】

マウスの状態を調べて，マウス構造体にセットします。

【関 数 名】

Mstatus

【書　　式】

int Mstatus (m)

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻 り 値】

0：クリック・ボタンが押されていない

1：クリック・ボタンが押されている

保守／廃止

(3) Mlocate

【機　　能】

マウス構造体で示される位置へカーサを移動します。

【関 数 名】

Mlocate

【書　　式】

void Mlocate (m)

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(4) Mclickoff

【機　　能】

クリック・ボタンが離されるまで，ループを実行します。

【関 数 名】

Mclickoff

【書　　式】

void Mclickoff (m)

【パラメータ】

Mouse　＊m：マウス型構造体ポインタ

【戻 り 値】

なし

保守／廃止

(5) Mbell

【機　　能】

マウス用のクリック音を出します。

【関 数 名】

Mbell

【書　　式】

void Mbell ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(6) Mclose

【機　　能】

マウス・カーサの表示を消します。

【関 数 名】

Mclose

【書　　式】

void Mclose ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

３.４　μPD72123制御系関数

３.４.１　関数一覧

No.	ラベル名	機　　能
G-1	GenbReg	μPD72123レジスタ・アクセスの許可
G-2	GdsbReg	μPD72123レジスタ・アクセスの禁止
G-3	GmapReg	マップ・レジスタ（D0H）の設定
G-4	GPalette	カラー・パレット・レジスタ（Bt450）の設定
G-5	GInit	μPD72123の初期化
G-6	GDispOn	ストップ・ディスプレイの解除
G-7	GDispOff	ストップ・ディスプレイの実行
G-8	GColor	論理演算の設定
G-9	GPmax	PMAXレジスタの設定
G-10	GsetMag	MAGH/MAGVレジスタのセット
G-11	GsetClip	クリッピング・エリア設定
G-12	GnonClip	クリッピング・エリア解除
G-13	GorgSrc	ソース側（転送源）座標設定
G-14	GorgDst	デスティネーション側（転送先）座標設定
G-15	GsetCmd	コマンド発行の処理
G-16	GDPbsy	DPBSYフラグのチェック
G-17	GPgport	PGPORTへの書き込み
G-18	GreadCol	色情報の読み出し
G-19	GPset	ドット描画
G-20	GLine	ラインの描画
G-21	GLineD2	ラインの描画
G-22	GBox	BOXの描画
G-23	GCircle	円描画
G-24	GElps	楕円描画
G-25	GCsec	扇型描画
G-26	GCarc	円弧描画
G-27	GEarc	楕円弧描画
G-28	GEsec	楕扇型描画
G-29	GCseg	弦形描画
G-30	GEseg	楕弦形描画
G-31	GgCircle	円描画（グラフィクス・ペン）
G-32	GgElps	楕円描画（グラフィクス・ペン）
G-33	GgLine	直線描画（グラフィクス・ペン）

保守／廃止

No.	ラベル名	機　　能
G-34	GgCarc	円弧描画（グラフィクス・ペン）
G-35	GgEarc	楕円弧描画（グラフィクス・ペン）
G-36	GBoxFill	BOX塗りつぶし
G-37	GBoxClr	BOXのクリア
G-38	GCls	クリア・スクリーン
G-39	GtileFill	BOXタイリング・フィル
G-40	GPaint	ペイント・コマンド
G-41	GtilePaint	ペイント・コマンド（タイリング・パターン）
G-42	GCrlFill	円塗りつぶし
G-43	GtriFill	三角形塗りつぶし
G-44	GtraFill	台形塗りつぶし
G-45	GelpsFill	楕円塗りつぶし
G-46	GFCopy	コピー処理（高速コピー）
G-47	GSLCopy	コピー処理（傾斜コピー）
G-48	GCopy	コピー処理（単純コピー）
G-49	G90Copy	コピー・コマンド（90°回転）
G-50	G180Copy	コピー・コマンド（180°回転）
G-51	G270Copy	コピー・コマンド（270°回転）
G-52	GFRCopy	任意角回転・拡大／縮小コピー
G-53	G3oPCopy	３オペランド・コピー
G-54	GPutA	PUT_Aコマンド
G-55	GPutC	PUT_Cコマンド
G-56	GPrint	文字列表示処理
G-57	GPutch	JISコードの全角文字表示
G-58	Glogo	“AGDCII”アウトラインフォント表示処理
G-59	Gflame	Boxフレームの表示
G-60	sjtoj	シフトJIS→JISコードへの変換
G-61	rnd	乱数の発生
G-62	wait	ウエイト
G-63	GcalES	レンジ変換
G-64	GSetTile	タイリング・パターンの設定

３.４.２　各関数の説明

以下に，μPD72123制御の各関数について説明します。

(1) GenbReg

【機　　能】

μPD72123の内部レジスタへのアクセスを可能にします。

【関　数　名】

GenbReg

【書　　式】

void GenbReg（）

【パラメータ】

なし

【戻　り　値】

なし

保守／廃止

(2) GdsbReg

【機　　能】

μPD72123の内部レジスタへのアクセスを禁止します。

【関 数 名】

GdsbReg

【書　　式】

void GdsbReg ()

【パラメータ】

なし

【戻 り 値】

なし

(3) GmapReg

【機　　能】

μPD72123の内部レジスタ群のマッピング・レジスタ群を設定します。同時に，カラー・パレット・レジスタ（Bt450のレジスタ）のマッピングも行います。

【関 数 名】

GmapReg

【書　　式】

void GmapReg ()

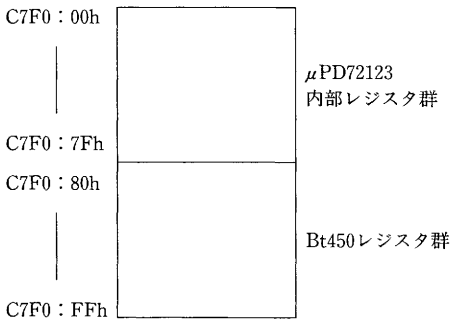
【パラメータ】

なし

【戻 り 値】

なし

〈PC9801側から見たメモリマップ〉



保守／廃止

(4) GPalette

【機　　能】

カラー・パレット・レジスタ（Bt450）の設定を行います。

【関 数 名】

GPalette

【書　　式】

void GPalette（buf）

【パラメータ】

Palette　＊buf：パレット型構造体

【戻 り 値】

なし

保守／廃止

(5) GInit

【機　　能】

μPD72123の初期化を行います。

同時に，マスタもしくはスレーブ動作の設定も行います。

【関 数 名】

GInit

【書　　式】

void GInit（dsp)

【パラメータ】

Display　＊dsp：ディスプレイ型構造体

【戻 り 値】

なし

保守／廃止

(6) GDispOn

【機　　能】

グラフィクス・ディスプレイ（CRT）の表示をONにします。

【関 数 名】

GDispOn

【書　　式】

void GDispOn ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(7) GDispOff

【機　　能】

グラフィクス・ディスプレイ（CRT）の表示をOFFにします。

【関 数 名】

GDispOff

【書　　式】

void GDispOff ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(8) GColor

【機　　能】

描画を行うための，論理演算の種類を設定します。

【関 数 名】

GColor

【書　　式】

void GColor (mod, color)

【パラメータ】

uint mod ：MOD0，MOD1レジスタへの設定値を指定

uint color：PLANESレジスタへの設定値を指定

【戻 り 値】

なし

【備　　考】

mod，colorで使用する値は，“const.h”で定義されています。

保守／廃止

(9) GPmax

【機　　能】

PMAXレジスタに値を設定します。

【関 数 名】

GPmax

【書　　式】

void GPmax（max）

【パラメータ】

int max：PMAXレジスタへの設定値を指定

【戻 り 値】

なし

保守／廃止

⑽　GsetMag

【機　　能】

拡大／縮小率を指定するMAGH，MAGVレジスタに値を設定します。

【関　数　名】

GsetMag

【書　　式】

void GsetMag（magh，magv）

【パラメータ】

int magh：水平方向の拡大／縮小率を設定

int magv：垂直方向の拡大／縮小率を設定

【戻　り　値】

なし

【備　　考】

設定する値は，“const.h”で定義されています。

保守／廃止

(11) GsetClip

【機　　能】

描画領域のクリッピング領域を設定します。

【関 数 名】

GsetClip

【書　　式】

void GsetClip（tbl）

【パラメータ】

Clip　＊tbl：CLIP型構造体ポインタ

【戻 り 値】

なし

保守／廃止

⑿ GnonClip

【機　　能】

クリッピング領域を無効にします。

【関 数 名】

GnonClip

【書　　式】

void GnonClip ()

【パラメータ】

なし

【戻 り 値】

なし

(13) GorgSrc

【機　　能】

転送源の座標系を設定します。Coodinate型構造体で指定します。

【関 数 名】

GorgSrc

【書　　式】

void GorgSrc (buf)

【パラメータ】

Coodinate ＊buf：Coodinate型構造体

【戻 り 値】

なし

保守／廃止

⑭ GorgDst

【機　　能】

転送先の座標系を設定します。Coodinate型構造体で指定します。

【関 数 名】

GorgDst

【書　　式】

void GorgDst（buf）

【パラメータ】

Coodinate ＊buf：Coodinate型構造体

【戻 り 値】

なし

保守／廃止

(15) GsetCmd

【機　　能】

μPD72123に対してコマンドを発行します。

【関 数 名】

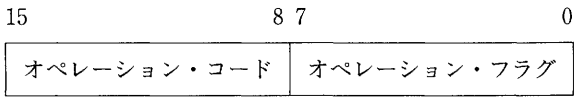
GsetCmd

【書　　式】

int GsetCmd (code)

【パラメータ】

uint code：コマンド・コード



【戻 り 値】

0：正常

−1：エラー

【動　　作】

a)　ステータス・レジスタの値を読み込んで，DPERR/PPERRフラグが，インアクティブであることを確認します。

b)　プリプロセッサ／描画プロセッサが処理実行中でないことを確認します。
(ステータス・レジスタの値を読み込んで，PPBSY/DPBSYフラグがインアクティブであることを確認します。)

c)　ステータス・レジスタの値を読み込んで，PPBSY/DPBSYフラグが，インアクティブであることを確認したあと，コマンド・コードとオペレーション・フラグを書き込みます。

d)　ステータス・レジスタの値を読み込んで，PPBSYフラグが，インアクティブになるまで待ちます。

保守／廃止

⑯ GDPbsy

【機　　能】

μPD72123がアイドル状態になるまで待ちます。

【関 数 名】

GDPbsy

【書　　式】

void GDPbsy ()

【パラメータ】

なし

【戻 り 値】

なし

保守／廃止

(17) GPgport

【機　　能】

μPD72123のPGPORTレジスタへデータを書き込みます。

【関 数 名】

GPgport

【書　　式】

uint GPgport (data)

【パラメータ】

uint data：PGPORTへのデータ書き込み

【戻 り 値】

−1：DPERR/PPERRが発生

0：書き込み終了

1：DPBSY/PPBSY＝0（アイドル状態）

保守／廃止

⑱ GreadCol

【機　　能】

色情報を読み出します。

【関 数 名】

GreadCol

【書　　式】

uint GreadCol（x，y）

【パラメータ】

int x，y：色情報読み出し座標

【戻 り 値】

色情報

保守／廃止

⑲ GPset

【機　　能】

点描画コマンドを実行します。

この際，μPD72123に対して発行しているコマンドは，A_DOT_Mコマンドです。

【関　数　名】

GPset

【書　　式】

void GPset（x，y）

【パラメータ】

int x：x座標

int y：y座標

【戻　り　値】

なし

保守／廃止

⑳ GLine

【機　　能】

始点（x，y）から終点（xe，ye）までの直線を描画します。

なお，この関数では線種は実線固定で，描画終了点まで描画します。

また，線幅／線種の拡大／縮小も行いません。

【関　数　名】

GLine

【書　　式】

void GLine（x，y，xe，ye）

【パラメータ】

int x，y　：描画開始点座標

int xe，ye：描画終了点座標

【戻　り　値】

なし

保守／廃止

(21) GLineD2

【機　　能】

ポリライン描画を行う場合に使用します。

直線の終点を与えるだけで，連続して直線を描画することができます。

【関　数　名】

GLineD2

【書　　式】

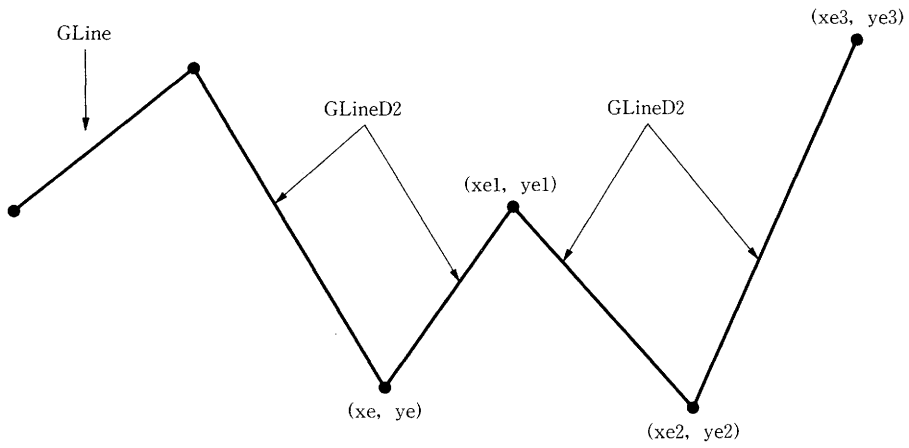
void GLineD2（xe，ye）

【パラメータ】

int xe：描画終了点座標

【戻　り　値】

なし



(22) GBox

【機　　能】

長方形描画を行います。

ただし，線種は実線のみで，線幅は１ドット幅固定です。

【関　数　名】

GBox

【書　　式】

void GBox（x，y，xe，ye）

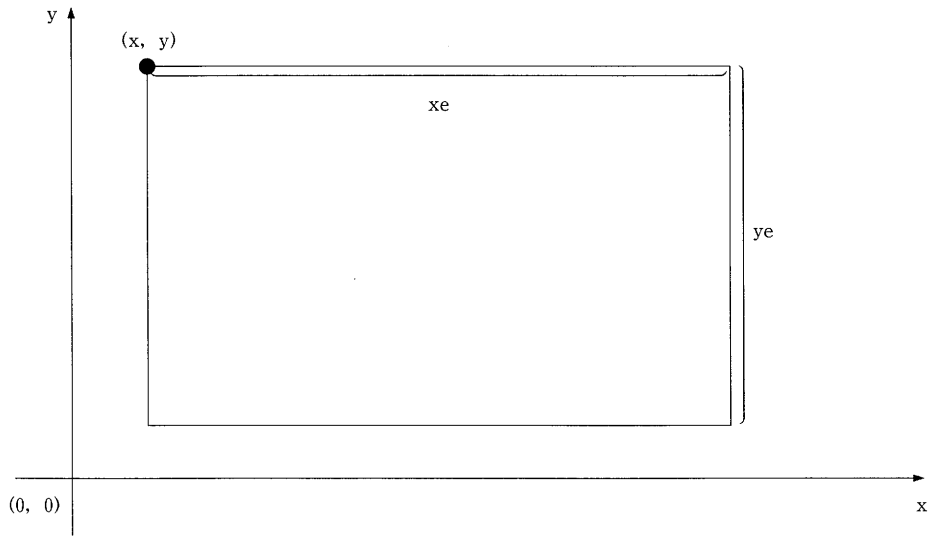
【パラメータ】

int x，y　：描画開始点座標

int xe，ye：横辺，縦辺の長さを指定

【戻　り　値】

なし



この関数では，常に左上が開始点となるような長方形描画を行います。

保守／廃止

(23) GCircle

【機　　能】

円描画コマンドを実行します。

ただし，線種は実線のみです。

【関 数 名】

GCircle

【書　　式】

void GCircle (x, y, r)

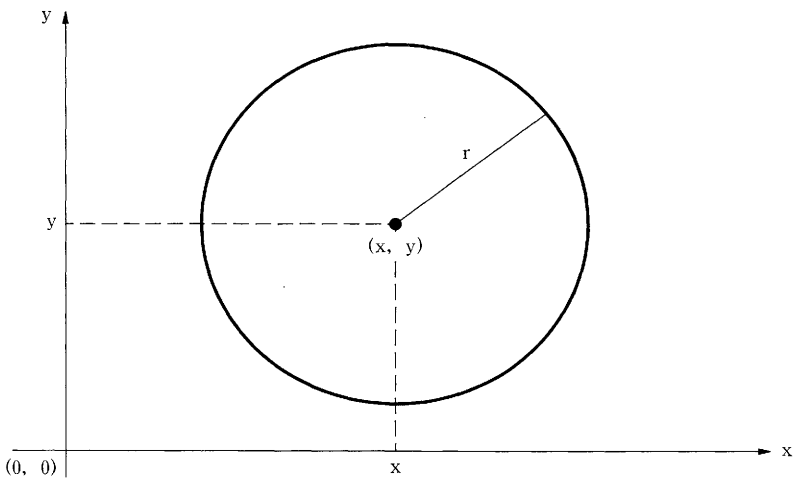
【パラメータ】

int x, y：円の中心点座標

int r　　：円の半径

【戻 り 値】

なし



保守／廃止

㉔ GElps

【機　　能】

楕円描画を実行します。

ただし，線種は実線のみです。

【関 数 名】

GElps

【書　　式】

void GElps (xc, yc, dy, dh, dv)

【パラメータ】

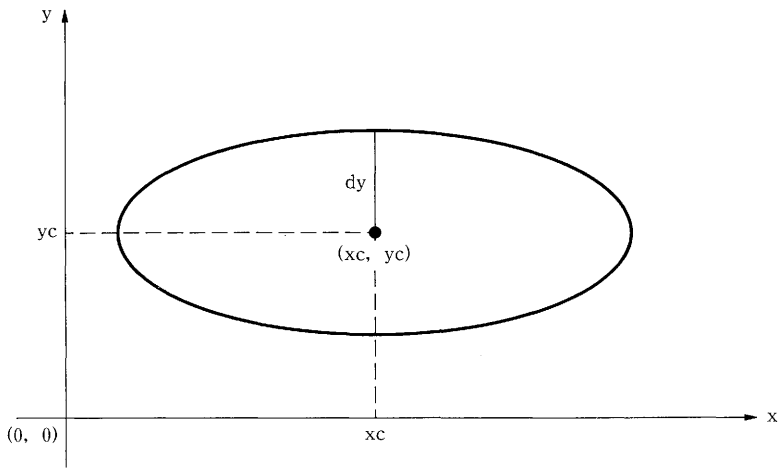
int xc, yc：楕円の中心点

int dy　　：Y軸方向の半径

int dh, dv：楕円の比率値

【戻 り 値】

なし



保守／廃止

(25) GCsec

【機　　能】

扇型の描画を行います。

ただし，この関数では時計回りに描画を行います。線種は実線のみです。

【関 数 名】

GCsec

【書　　式】

void GCsec (xc, yc, xs, ys, xe, ye, dx)

【パラメータ】

int xc, yc：扇型の中心点座標

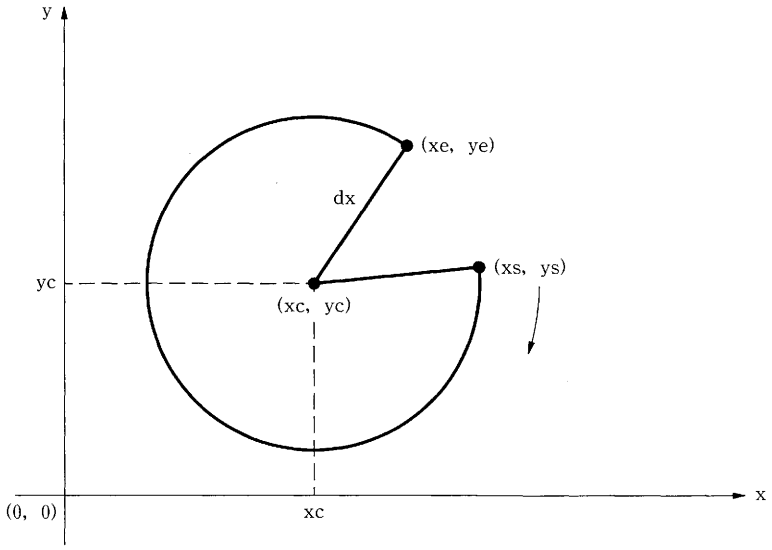
int xs, ys：弧描画開始点

int xe, ye：弧描画終了点

int dx　　：半径

【戻 り 値】

なし



保守／廃止

(26) GCarc

【機　　能】

円弧描画を実行します。

ただし，この関数では時計回りに描画を行います。線種は実線のみです。

また，描画終了点も描画します。

【関 数 名】

GCarc

【書　　式】

void GCarc (xc, yc, xs, ys, xe, ye, dx)

【パラメータ】

int xc, yc：円弧の中心点座標

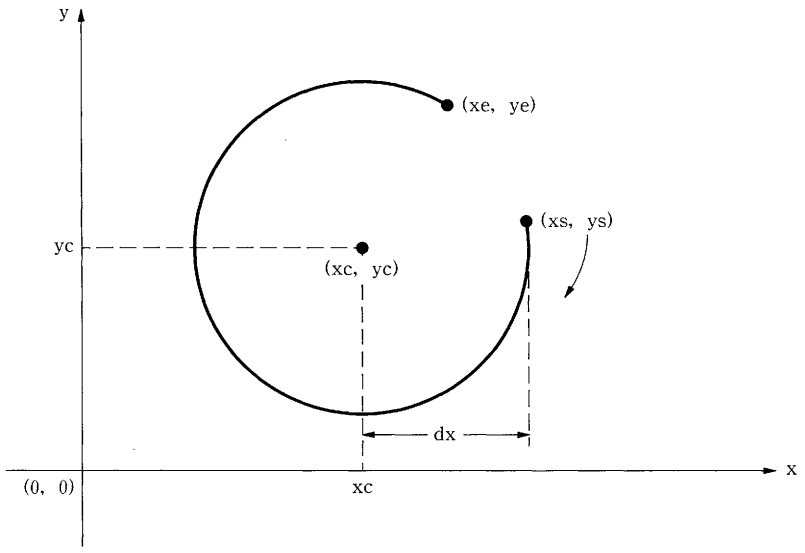
int xs, ys：弧描画開始点

int xe, ye：弧描画終了点

int dx　　：半径

【戻 り 値】

なし



保守／廃止

(27) GEarc

【機　　能】

楕円弧の描画を実行します。

ただし，この関数では時計回りに描画を行います。線種は実線のみです。

また，描画終了点も描画します。

【関 数 名】

GEarc

【書　　式】

void GEarc (xc, yc, xs, ys, xe, ye, dx, dy, dh, dv)

【パラメータ】

int xc, yc：楕円弧の中心点座標

int xs, ys：弧描画開始点

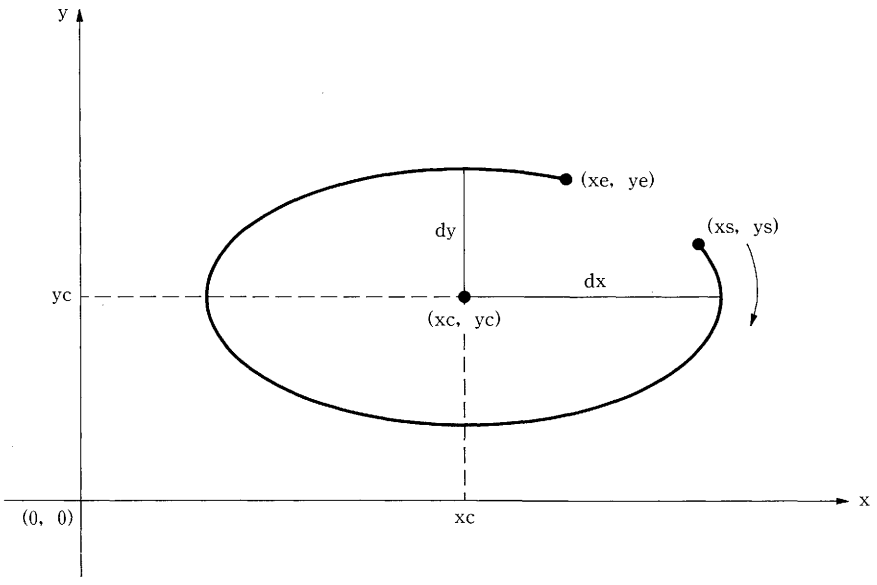
int xe, ye：弧描画終了点

int dx, dy：半径

int dh, dv：$dx^2/dy^2 = dh/dv$を満たすdh, dvを与えます。

【戻 り 値】

なし



保守／廃止

(28) GEsec

【機　　能】

楕扇形の描画を実行します。

ただし，この関数では時計回りに描画を行います。線種は実線のみです。

また，描画終了点も描画します。

【関　数　名】

GEsec

【書　　式】

void GEsec (xc, yc, xs, ys, xe, ye, dx, dy, dh, dv)

【パラメータ】

int xc, yc：楕円弧の中心点座標

int xs, ys：弧描画開始点

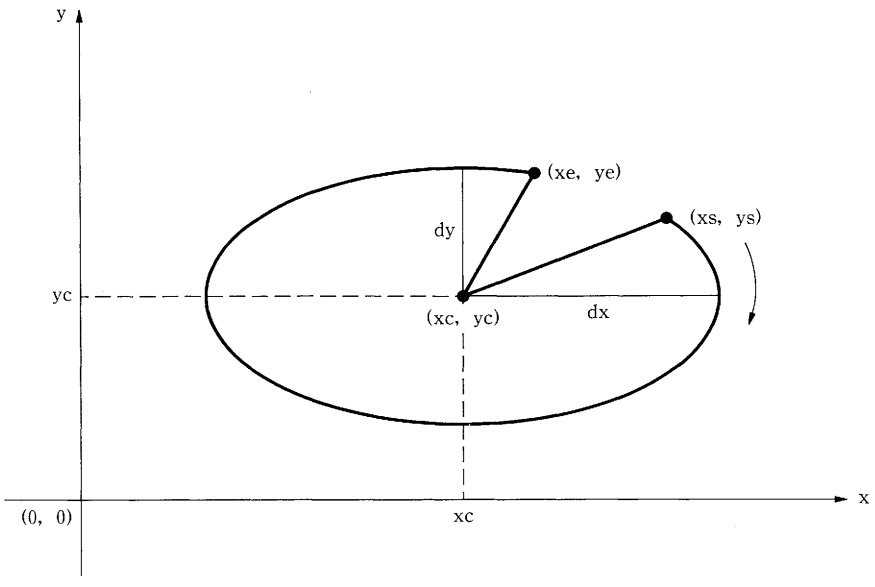
int xe, ye：弧描画終了点

int dx, dy：半径

int dh, dv：$dx^2/dy^2 = dh/dv$を満たすdh, dvを与えます。

【戻　り　値】

なし



保守／廃止

(29) GCseg

【機　　能】

弦形の描画を実行します。

ただし，この関数では時計回りに描画を行います。線種は実線のみです。

また，描画終了点も描画します。

【関　数　名】

GCseg

【書　　式】

void GCseg (xc, yc, xs, ys, xe, ye, dx)

【パラメータ】

int xc, yc：弦形の中心点座標

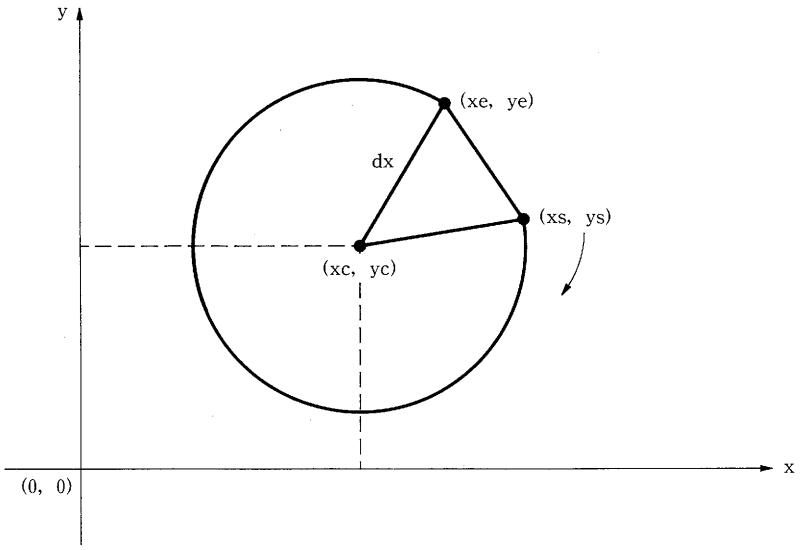
int xs, ys：弧描画開始点

int xe, ye：弧描画終了点

int dx　　：半径

【戻　り　値】

なし



保守／廃止

(30) GEseg

【機　　能】

楕弦形の描画を実行します。

この関数では時計回りに描画を行います。

【関 数 名】

GEseg

【書　　式】

void GEseg (xc, yc, xs, ys, xe, ye, dx, dy, dh, dv)

【パラメータ】

int xc, yc：楕円弧の中心点座標

int xs, ys：弧描画開始点

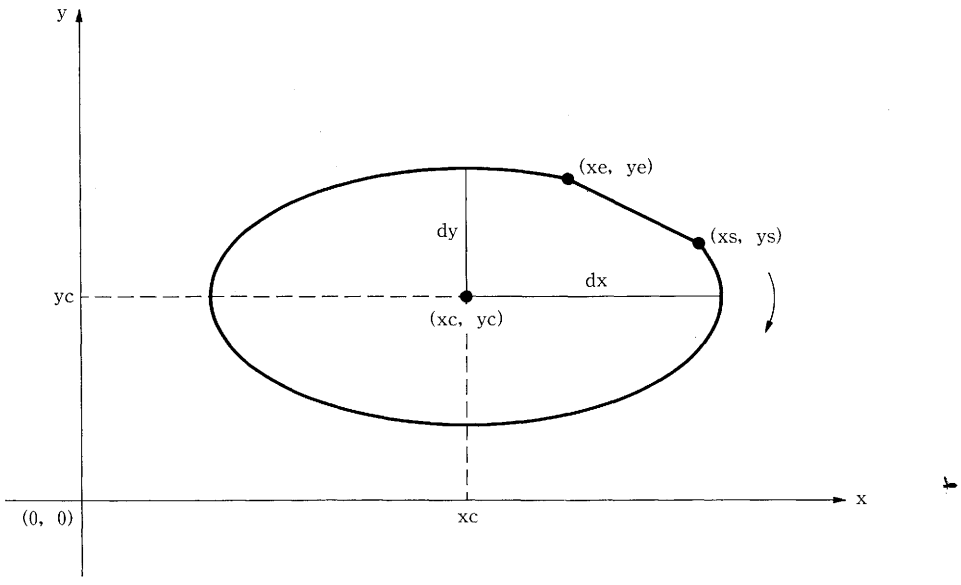
int xe, ye：弧描画終了点

int dx, dy：半径

int dh, dv：$dx^2/dy^2 = dh/dv$を満たすdh, dvを与えます

【戻 り 値】

なし



保守／廃止

(31) GgCircle

【機　　能】

グラフィクス・ペンによる円描画を実行します。

この関数では，時計回りに描画を実行します。

【関 数 名】

GgCircle

【書　　式】

void GgCircle (x, y, r, ead3, ptnp)

【パラメータ】

int x, y：円の中心点座標

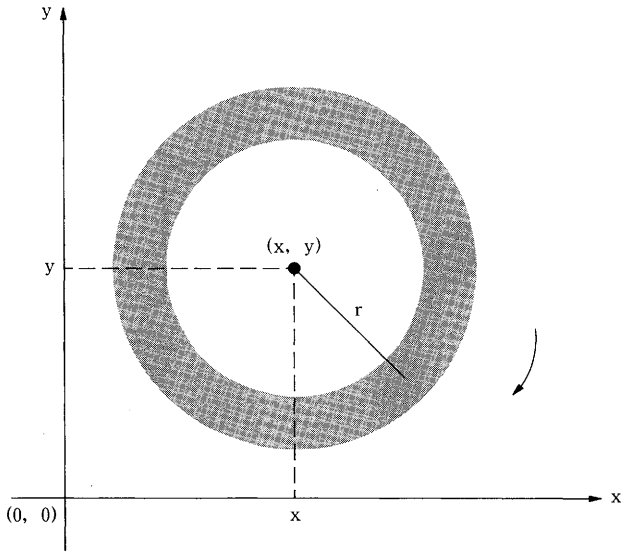
int r　　：半径

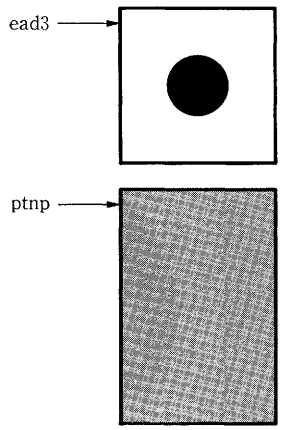
ulong ead3：ペン先形状を設定している先頭アドレス

ulong ptnp：フィル・パターンを格納している先頭アドレス

【戻 り 値】

なし





保守／廃止

(32) GgElps

【機　　能】

グラフィクス・ペンによる楕円描画を実行します。

この関数では，時計回りに描画を実行します。

【関 数 名】

GgElps

【書　　式】

void GgElps (xc, yc, dy, dh, dv, ead3, ptnp)

【パラメータ】

int xc, yc：楕円の中心点座標

int dy　　：半径

int dh, dv：楕円の横辺／縦辺の比

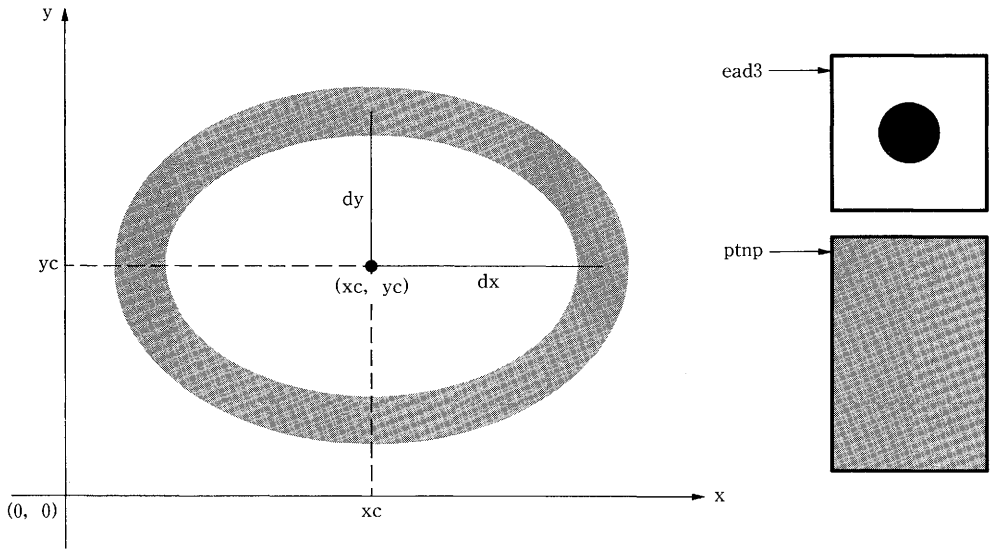
ulong ead3　：ペン先形状を設定している先頭アドレス

ulong ptnp　：フィル・パターンを格納している先頭アドレス

($dx^2/dy^2 = dh/dv$を満たすように設定)

【戻 り 値】

なし



保守／廃止

(33) GgLine

【機　　能】

グラフィクス・ペンによる直線の描画を実行します。

【関 数 名】

GgLine

【書　　式】

void Ggline (x, y, xe, ye, ead3, ptnp)

【パラメータ】

int x, y　：描画開始点座標

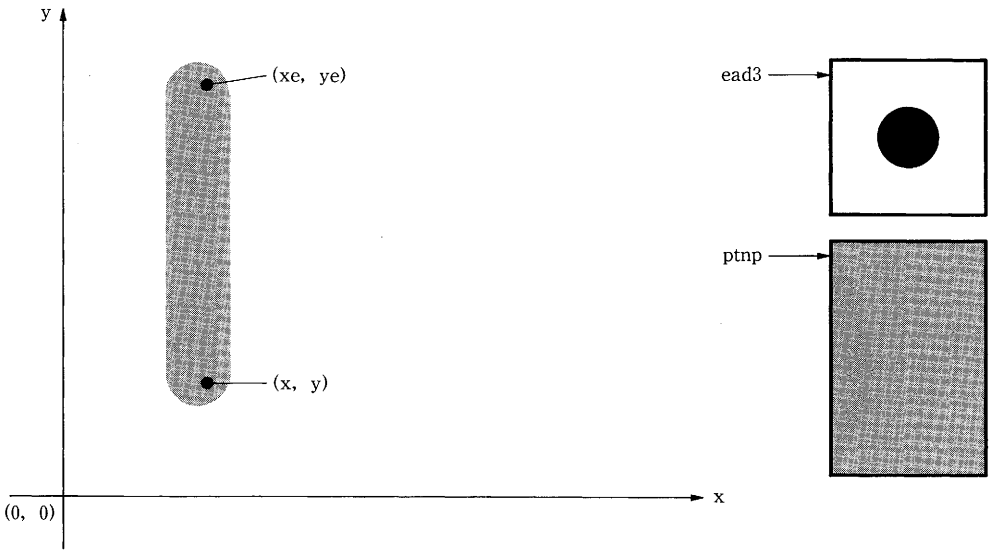
int xe, ye：描画終了点座標

ulong ead3　：ペン先形状を設定している先頭アドレス

ulong ptnp　：フィル・パターンを格納している先頭アドレス

【戻 り 値】

なし



保守／廃止

㉞ GgCarc

【機　　能】

グラフィクス・ペンによる円弧描画を実行します。

【関 数 名】

GgCarc

【書　　式】

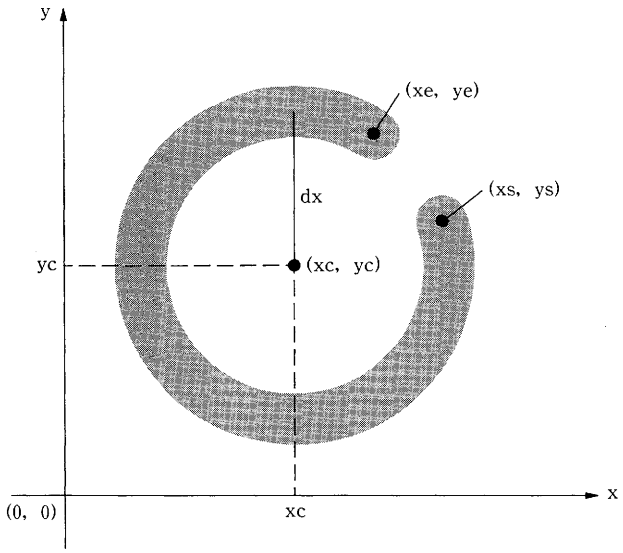
void GgCarc (xc, yc, dx, xs, ys, xe, ye, ead3, ptnp)

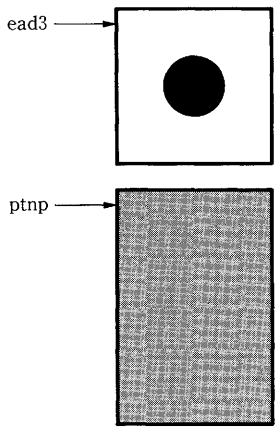
【パラメータ】

int xc, yc：円弧の中心点座標
int dx　　：半径
int xs, ys：描画開始点
int xe, ye：描画終了点
ulong ead3　：ペン先形状を設定している先頭アドレス
ulong ptnp　：フィル・パターンを格納している先頭アドレス

【戻 り 値】

なし





保守／廃止

㉟ GgEarc

【機　　能】

グラフィクス・ペンによる楕円弧の描画を実行します。

この関数では，時計回りに描画を行い，描画終了点まで描画します。

【関 数 名】

GgEarc

【書　　式】

void GgEarc (xc, yc, dx, dy, dh, dv, xs, ys, xe, ye, ead3, ptnp)

【パラメータ】

int xc, yc：楕円弧の中心点座標

int dx, dy：半径

int dh, dv：楕円弧の横辺／縦辺の比

int xs, ys：描画開始点

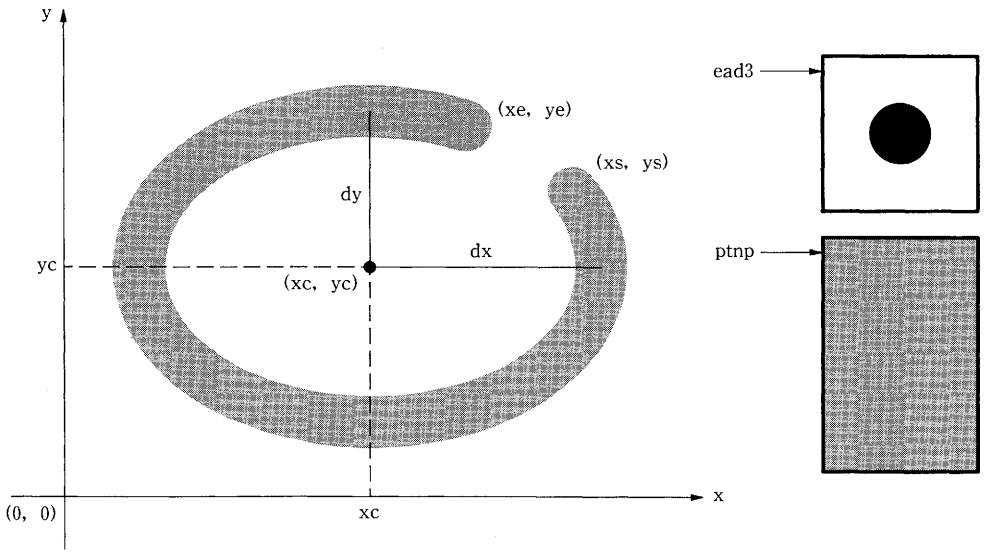
int xe, ye：描画終了点

ulong ead3　：ペン先形状を設定している先頭アドレス

ulong ptnp　：フィル・パターンを格納している先頭アドレス

【戻 り 値】

なし



保守／廃止

(36) GBoxFill

【機　　能】

長方形を塗りつぶします。

この関数では，PTNCNTレジスタはffffhにします。

描画開始点（x，y）は，左上の座標を与えてください。

【関　数　名】

GBoxFill

【書　　式】

void GBoxFill（x，y，dx，dy）

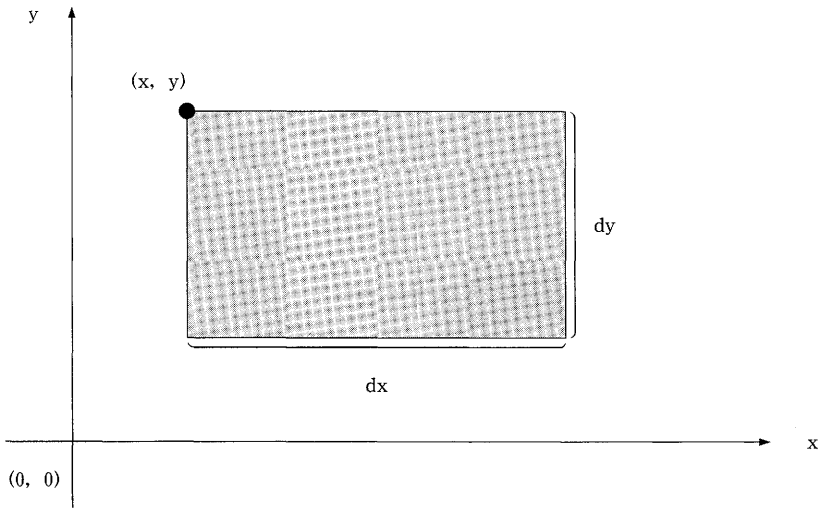
【パラメータ】

int x，y　：描画開始点

int dx，dy：横辺／縦辺の長さ

【戻　り　値】

なし



保守／廃止

(37) GBoxClr

【機　　能】

長方形の領域を“0”クリアします。

この関数では，PTNCNTレジスタの値を0000hにして実行します。

DYレジスタには-dyとして設定されますので注意してください。

【関 数 名】

GBoxC1r

【書　　式】

void GBoxC1r（x，y，dx，dy）

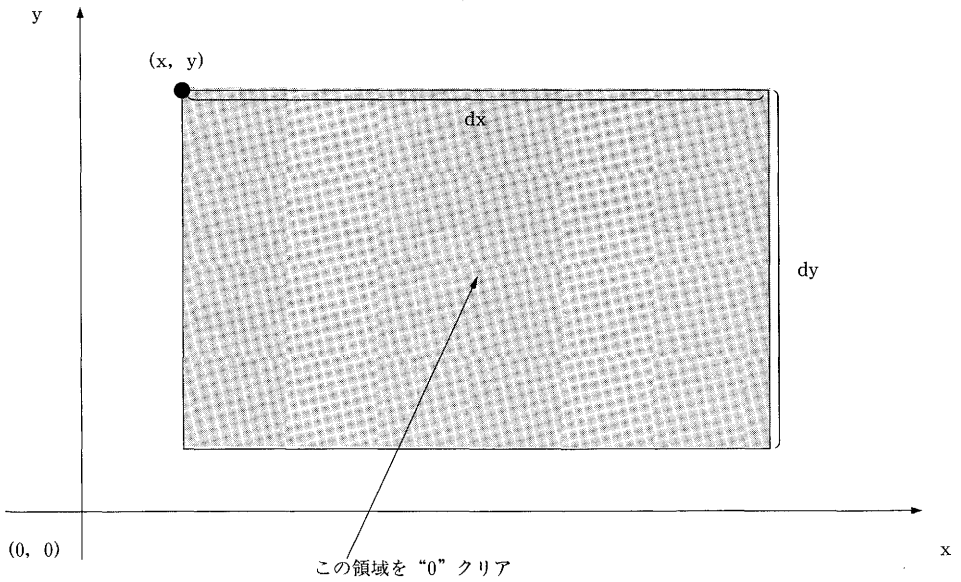
【パラメータ】

int x，y　：描画開始点座標

int dx，dy：横辺，縦辺の長さを指定

【戻 り 値】

なし



保守／廃止

㊳ GCls

【機　　能】

標準オン・スクリーン・エリアを全プレーンにわたって“0”クリアします。

【関 数 名】

GCls

【書　　式】

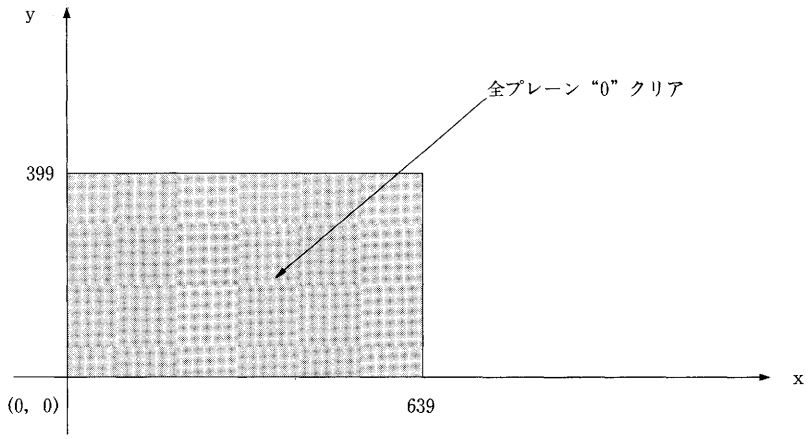
void GCls ()

【パラメータ】

なし

【戻 り 値】

なし



保守／廃止

㊴ GtileFill

【機　　能】

指定された長方形内を，メモリ上にあらかじめ定義されているタイリング・パターンで塗りつぶします。

【関 数 名】

GtileFill

【書　　式】

void GtileFill (x, y, dx, dy, t)

【パラメータ】

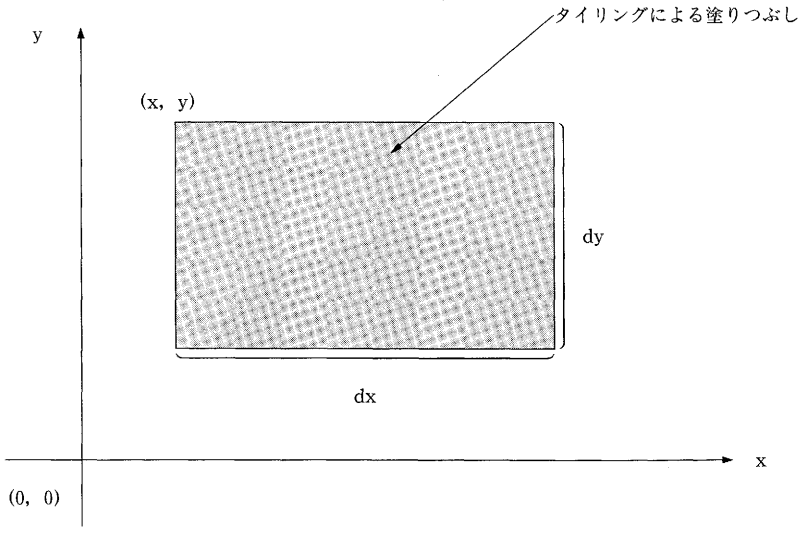
int x, y　：長方形の左上の座標

int dx, dy：横辺／縦辺の長さ

Tilling *t　：タイリング・パターン構造体ポインタ

【戻 り 値】

なし



保守／廃止

(40) GPaint

【機　　能】

境界検索開始点（x，y）の色以外の色を境界色として塗りつぶします。

論理演算は，GColor（）であらかじめ指定しておきます。

【関 数 名】

GPaint

【書　　式】

void GPaint（x，y，col）

【パラメータ】

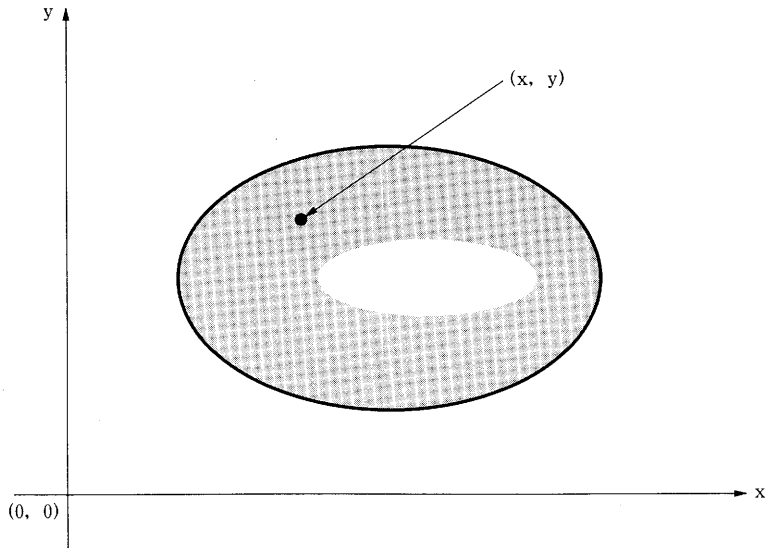
int x，y：境界点検索開始点座標

uint col　：境界色の指定[注]

注　この関数では，境界検索開始点（x，y）の色以外の色を境界色として塗りつぶしますので，この設定は無意味となります。

【戻 り 値】

なし



保守／廃止

(41) GtilePaint

【機　　能】

あらかじめ定義してあるタイリング・パターンで，任意閉領域を塗りつぶします。

【関 数 名】

GtilePaint

【書　　式】

void GtilePaint (x, y, col, t)

【パラメータ】

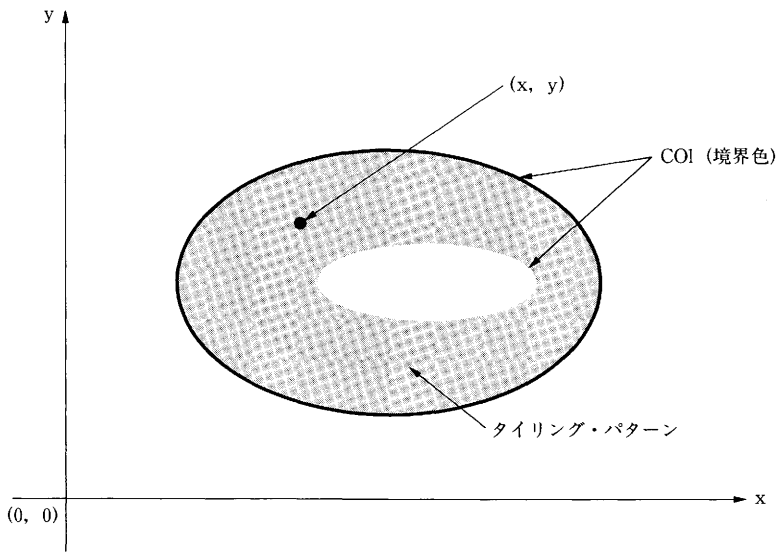
int x, y：境界点検索開始点の座標

uint col　：境界色を指定

Tilling *t　：タイリング・パターン構造体ポインタ

【戻 り 値】

なし



保守／廃止

(42) GCrlFill

【機　　能】

円を塗りつぶします。

【関 数 名】

GCrlFill

【書　　式】

void GCrlFill (xc, yc, dx)

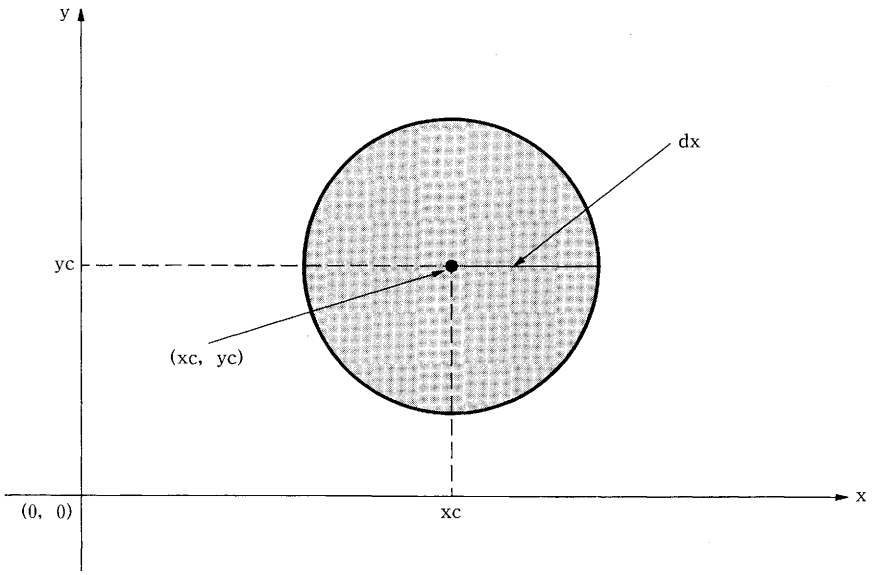
【パラメータ】

int xc, yc：円の中心点座標

int dx　　：半径

【戻 り 値】

なし



保守／廃止

(43) GtriFill

【機　　能】

三角形を塗りつぶします。

【関 数 名】

GtriFill

【書　　式】

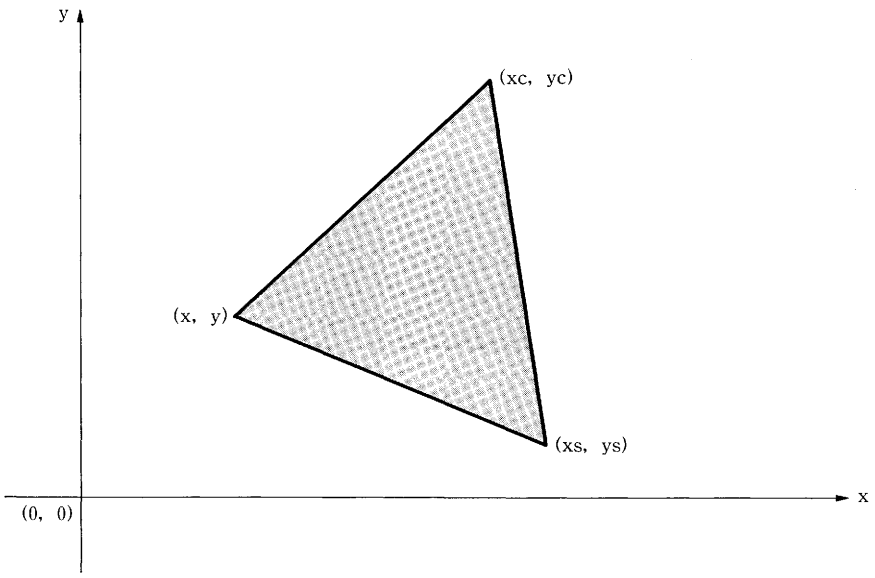
void GtriFill (x, y, xs, ys, xc, yc)

【パラメータ】

int x, y　：
int xs, ys：　これらの座標で示される三角形内を塗りつぶす
int xc, yc：

【戻 り 値】

なし



保守／廃止

(44) GtraFill

【機　　能】

台形を塗りつぶします。

【関 数 名】

GtraFill

【書　　式】

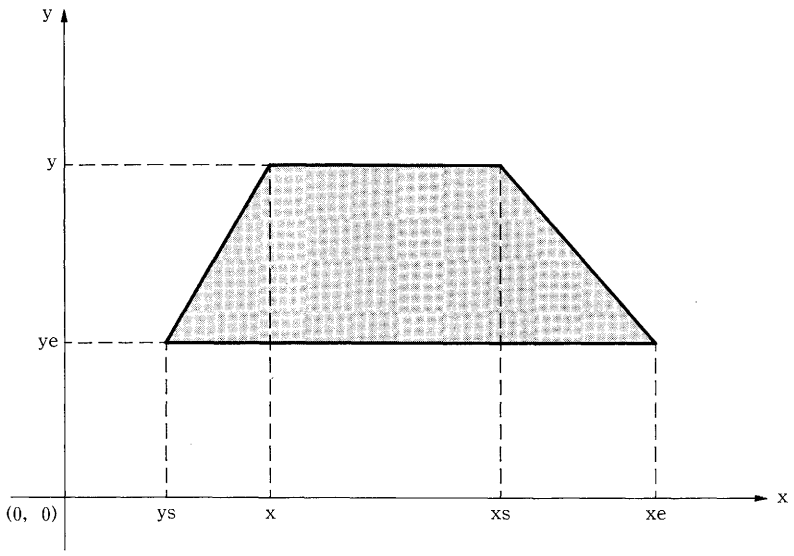
void GtraFill (x, y, xs, ys, xe, ye)

【パラメータ】

int x, y　：
int xs, ys：　各座標を指定
int xe, ye：

【戻 り 値】

なし



保守／廃止

(45) GelpsFill

【機　　能】

楕円を塗りつぶします。

【関 数 名】

GelpsFill

【書　　式】

void GelpsFill (xc, yc, dy, dh, dv)

【パラメータ】

int xc, yc：楕円の中心点座標

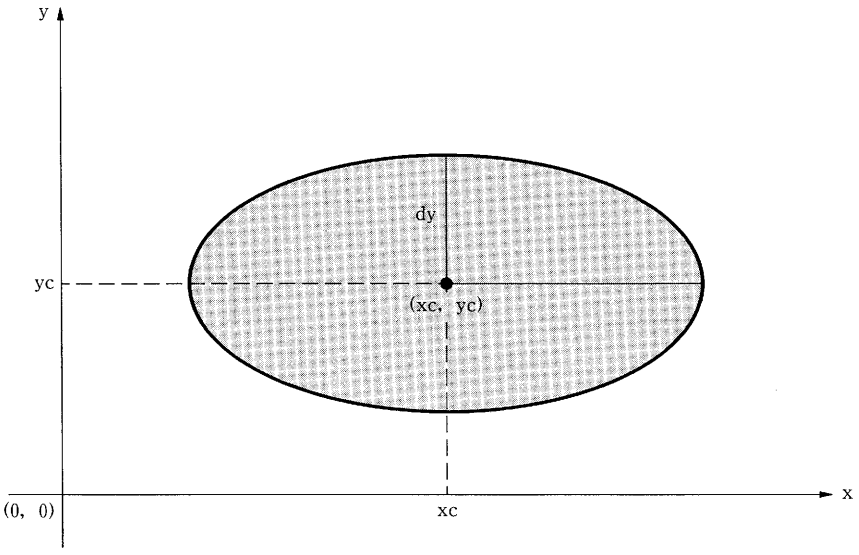
int dy　　：y方向の半径

int dh, dv：縦／横の比率

(dx²/dy²＝dh/dvを満たすように設定)

【戻 り 値】

なし



保守／廃止

(46) GFCopy

【機　　能】

高速コピーを実行します。

a) 座標系の指定は，c→modeで指定します。

b) 実際のコピー・エリアの座標または絶対番地を指定します。

c) コピーする領域の大きさを指定します。

【関 数 名】

GFCopy

【書　　式】

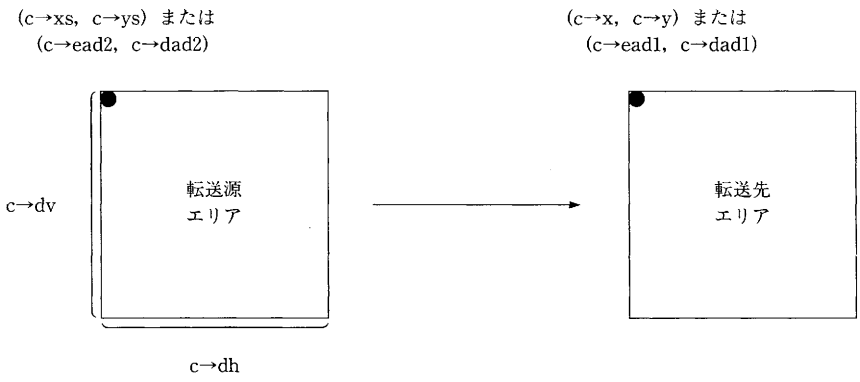
void GFCopy (c)

【パラメータ】

copy ＊c：コピー型構造体ポインタ

【戻 り 値】

なし



保守／廃止

(47) GSLCopy

【機　　能】

傾斜コピーを実行します。

a) 座標系，領域の指定等は，GFCopy () と同じです。

b) 傾斜させるドット数をs1で指定します。

【関 数 名】

GSLCopy

【書　　式】

void GSLCopy (c, s1)

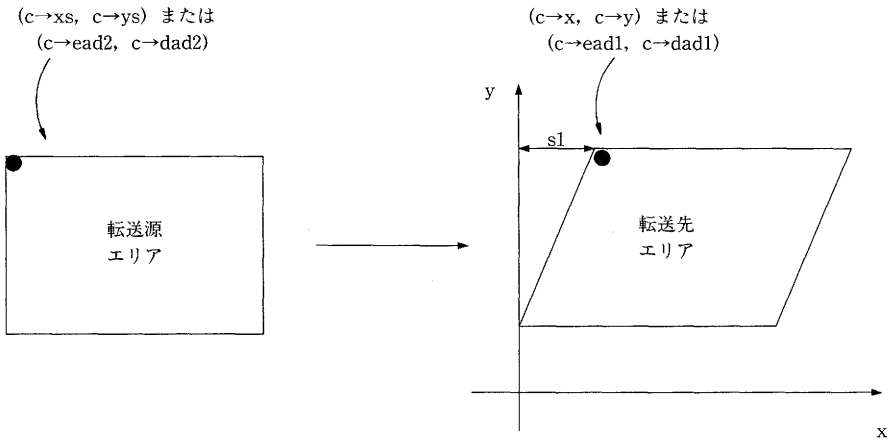
【パラメータ】

Copy *c：コピー型構造体ポインタ

int s1　　：傾斜ドット数

【戻 り 値】

なし



保守／廃止

(48) GCopy

【機　　能】

単純コピーを実行します。

a)　座標系，領域の指定等は，GFCopy () と同じです。

【関 数 名】

GCopy

【書　　式】

void GCopy (c, magh, magv)

【パラメータ】

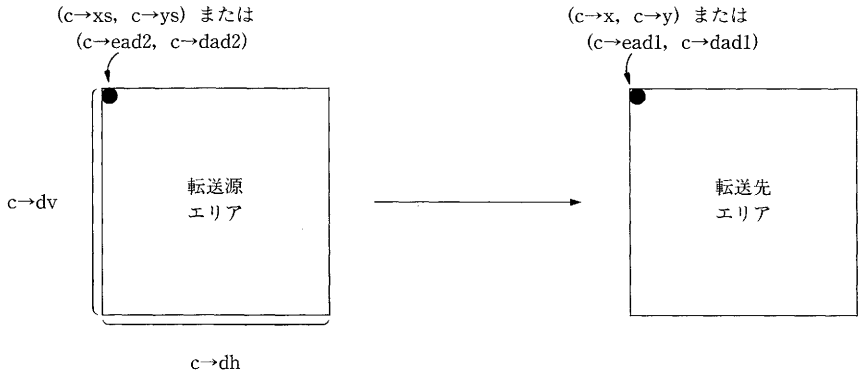
Copy *c：コピー型構造体ポインタ

int magh：x軸方向の拡大／縮小フラグ

int magv：y軸方向の拡大／縮小フラグ

【戻 り 値】

なし



保守／廃止

(49) G90Copy

【機　　能】

90°回転コピーを実行します（回転方向は反時計回り）。

a) 座標系，領域の指定等は，GFCopy () と同じです。

b) 転送先エリアの拡大／縮小率は，magh，magvで指定します。

（フラグは，“const.h”で定義されています。）

【関 数 名】

G90Copy

【書　　式】

void G90Copy (c, magh, magv)

【パラメータ】

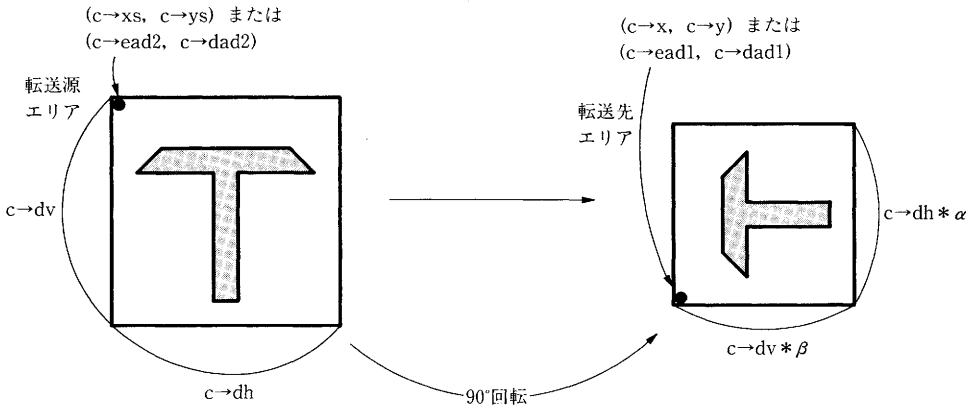
Copy ＊c：コピー型構造体ポインタ

int magh：x軸方向の拡大／縮小フラグ

int magv：y軸方向の拡大／縮小フラグ

【戻 り 値】

なし



保守／廃止

(50) G180Copy

【機　　能】

180°回転コピーを実行します。

【関 数 名】

G180Copy

【書　　式】

void G180Copy (c, magh, magv)

【パラメータ】

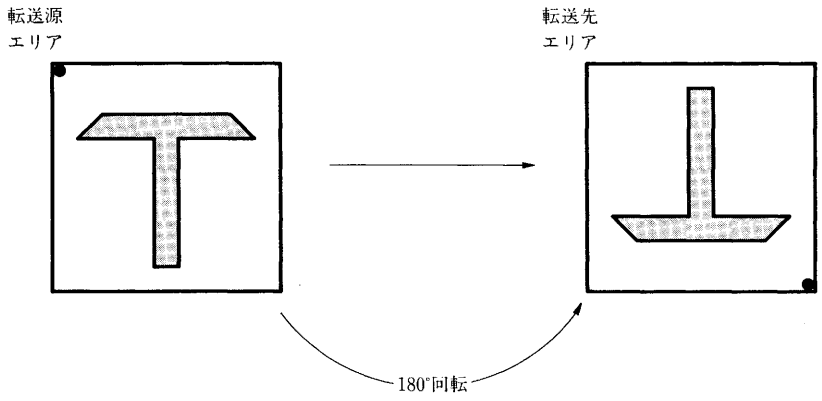
Copy *c：コピー型構造体ポインタ

int magh：x方向の拡大／縮小フラグ

int magv：y方向の拡大／縮小フラグ

【戻 り 値】

なし



保守／廃止

(51) G270Copy

【機　　能】

270°回転コピーを実行します。

【関 数 名】

G270Copy

【書　　式】

void G270Copy (c, magh, magv)

【パラメータ】

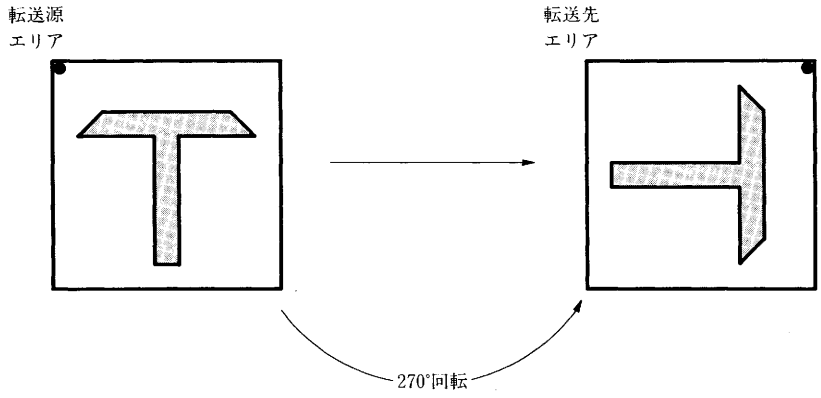
Copy *c：コピー型構造体ポインタ

int magh：x方向の拡大／縮小フラグ

int magv：y方向の拡大／縮小フラグ

【戻 り 値】

なし



保守／廃止

(52) GFRCopy

【機　　能】

任意角回転コピーを行います。

a) 座標系，コピー領域の大きさ，拡大／縮小の指定は，GFCopyと同じです。

【関 数 名】

GFRCopy

【書　　式】

void GFRCopy (c, dx, dy, xe, ye, magh, magv)

【パラメータ】

Copy *c ：コピー型構造体ポインタ

int dx, dy：横辺の傾き

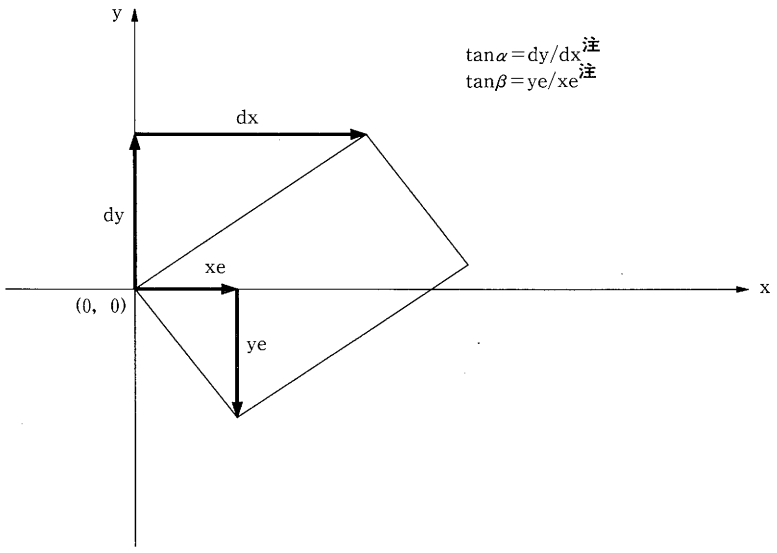
int xe, ye：縦辺の傾き

int magh ：x軸方向の拡大／縮小フラグ

int magv ：y軸方向の拡大／縮小フラグ

【戻 り 値】

なし



注　レジスタに設定する値は，ベクトルの向きも考慮して設定してください。

保守／廃止

(53) G3opCopy

【機　　能】

３オペランド・コピーを実行します。

（マスク領域（c→ead3）を指定する以外は，パラメータ設定はGFCopy（）と同じです。）

【関 数 名】

G3opCopy

【書　　式】

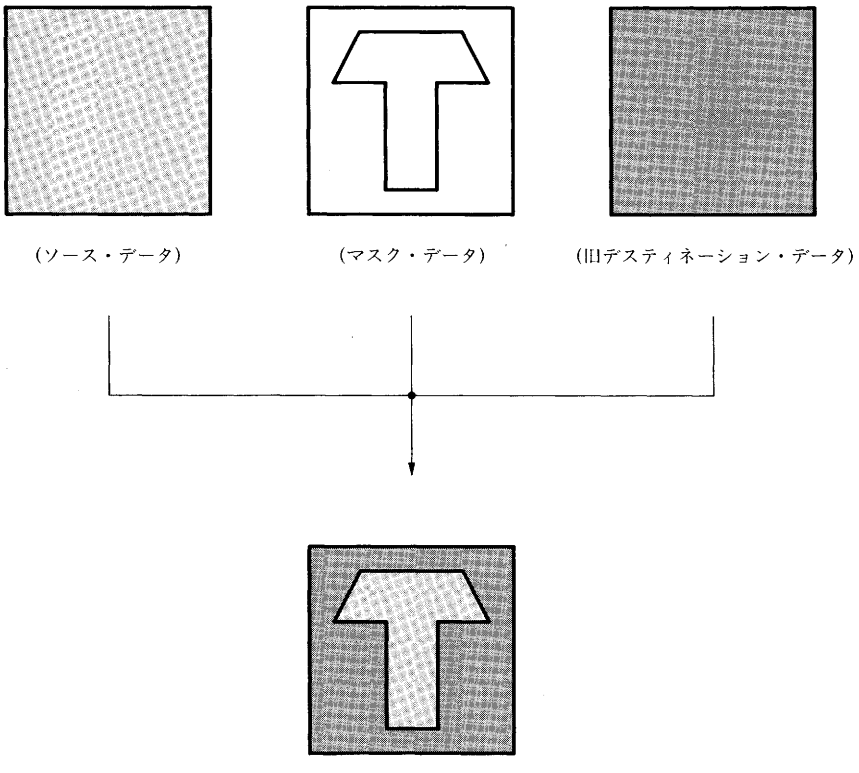
void G3opCopy（c）

【パラメータ】

Copy ＊c：コピー型構造体ポインタ

【戻 り 値】

なし



保守／廃止

(54) GPutA

【機　　能】

システム・メモリから表示メモリへのビット・ブロック転送コマンドの発行を行います。

（本関数はコマンドの発行だけを行うので，実際にデータを転送するのは本関数の呼び出し後に行います）。

【関 数 名】

GPutA

【書　　式】

void GPutA（adr，dh，dv）

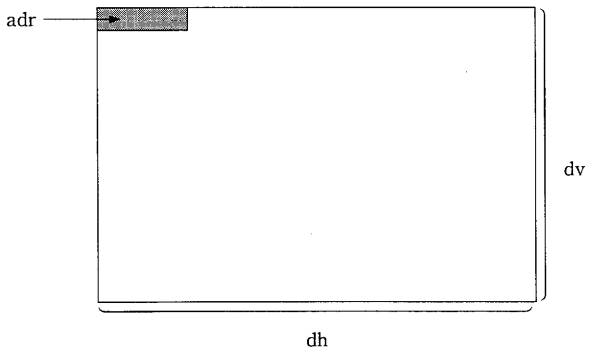
【パラメータ】

ulong adr　：転送先の先頭アドレス

int dh，dv：転送先の矩形領域の大きさを示す

【戻 り 値】

なし



保守／廃止

(55) GPutC

【機　　能】

システム・メモリから表示メモリへのビット・ブロック転送コマンドの発行を行います。

（転送先のエリアの先頭を座標で指定する以外は，GPutA（）と同じです。）

【関 数 名】

GPutC

【書　　式】

void GPutC（x，y，dh，dv）

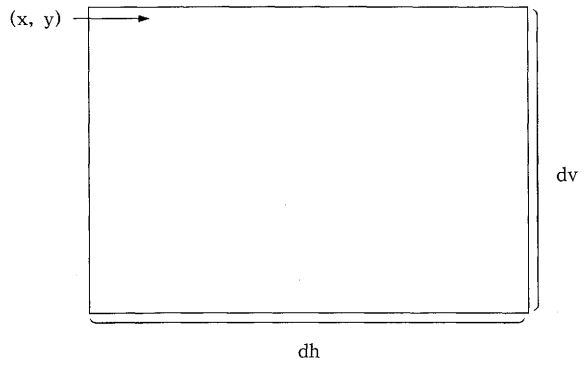
【パラメータ】

int x，y　：転送先の先頭座標を指定

int dh，dv：転送先の矩形領域の大きさを示す

【戻 り 値】

なし



保守／廃止

(56) GPrint

【機　　能】

表示メモリに文字列を描画します。

a) 文字列の内容はASCIIコードまたはシフトJISコードで指定します。
文字列の終りは，必ずヌルコード（00h）とします。

b) μPD72123のコピー・コマンドを使用して，μPD72123のアドレス空間に割り付けられている漢字ROMより，漢字フォントを表示メモリ上にコピーします。

【関　数　名】

GPrint

【書　　式】

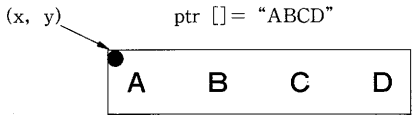
void GPrint（x，y，ptr，f）

【パラメータ】

int x，y　：描画開始座標
uchar ＊ptr：文字列配列のポインタ
Font ＊f　：フォント型構造体ポインタ

【戻　り　値】

なし



保守／廃止

(57) GPutch

【機　　能】

指定された文字を描画します。文字列の内容は，JISコードで与えます。

（与えられたJISコードから，該当する漢字フォントROMのアドレスを算出して描画領域にコピーします。）

【関 数 名】

GPutch

【書　　式】

void GPutch（x，y，code）

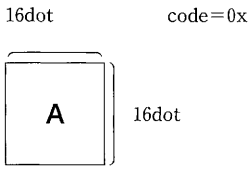
【パラメータ】

int x，y ：描画開始座標

uint code：表示JISコード

【戻 り 値】

なし



保守／廃止

(58) Glogo

【機　　能】

“AGDCII”のロゴのアウトラインを描画します。

(GBoxFill () とGCrac () を用いてアウトラインを作ります。)

【関 数 名】

Glogo

【書　　式】

void Glogo (x, y, es, col)

【パラメータ】

int x, y：描画領域の左下の座標を指定

int es　：大きさ

int col　：色指定

【戻 り 値】

なし

保守／廃止

(59) Gflame

【機　　能】

フレームの枠を表示します。

（指定されたBOXの外側に２ドット幅の枠を描画します。）

【関 数 名】

Gflame

【書　　式】

void Gflame (flg, x, y, dx, dy)

【パラメータ】

int flg　　：フレームの表示モード

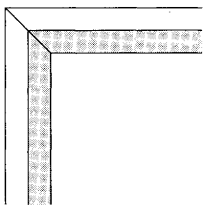
int x, y　：フレームの左上座標

int dx, dy：フレームの大きさ

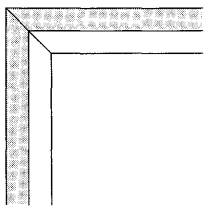
【戻 り 値】

なし

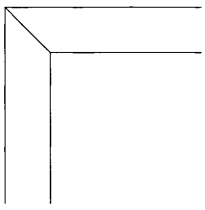
〈flg＝ON〉



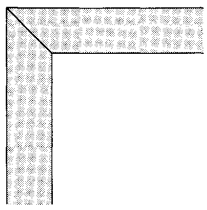
〈flg＝OFF〉



〈flg＝PULL〉



〈flg＝PUSH〉



保守／廃止

⑹ sjtoj

【機　　能】

与えられたシフトJISコードをJISコードに変換します。

【関 数 名】

sjtoj

【書　　式】

uint sjtoj (k)

【パラメータ】

uint k：シフトJISコード

【戻 り 値】

JISコード

保守／廃止

(61) rnd

【機　　能】

mxで指定された範囲内で乱数を発生させます。

【関 数 名】

rnd

【書　　式】

int rnd (mx)

【パラメータ】

int mx：乱数の発生範囲

【戻 り 値】

なし

保守／廃止

(62) wait

【機　　能】

ソフトウエア／タイマです。

【関 数 名】

wait

【書　　式】

void wait (msec)

【パラメータ】

uint msec：ウエイト・カウント数

【戻 り 値】

なし

保守／廃止

⑹ GcalES

【機　　能】

拡大／縮小フラグで示される倍率を，被変換データに乗じます。

【関 数 名】

GcalES

【書　　式】

int GcalES (es_flag, d)

【パラメータ】

int es_flag：拡大／縮小フラグ

int d　　：被変換データ

【戻 り 値】

なし

例 es_flagが－1 (ES15_16) のとき，拡大率は15/16となります。

戻り値＝d＊15/16

保守／廃止

(64) GSetTile

【機　　能】

タイリング・パターンの設定を行います。

a) タイリング・パターンの格納プレーン数をt→planeで指定します。

b) タイリング・パターンの格納座標系をt→orgで指定します。

c) タイリング・パターンでの描画論理演算をt→color，t→modで指定します。

d) システム・メモリ側にあるタイリング・パターンの先頭アドレスをt→ptrに指定します。

e) GPutA () を使用して，d) で指定されたデータを表示メモリ上のタイリング・パターン格納エリアに転送します。

【関　数　名】

GSetTile

【書　　式】

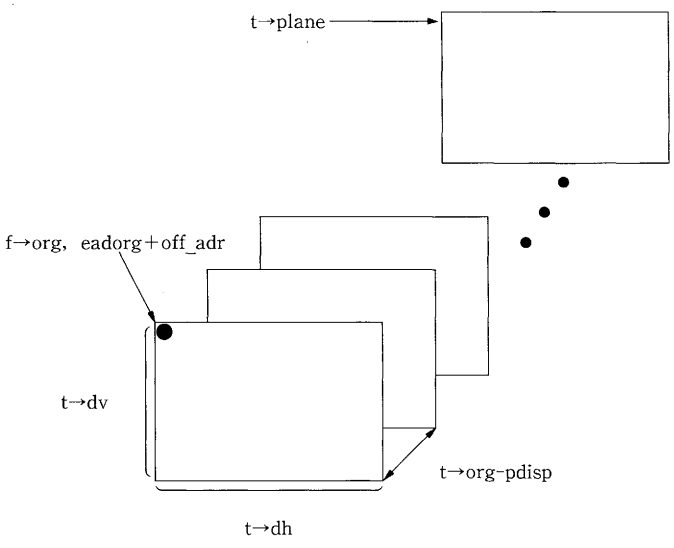
void GSetTile (t)

【パラメータ】

Tilling *t：タイリング・パターン構造体ポインタ

【戻　り　値】

なし



保守／廃止

第４章　コンパイル／リンク方法

４.１　開発環境

OS：MS-DOS　　V3.0以降
C　：MS-C　　V4.0

４.２　コンパイル方法

保存インクルード・ファイル　　const.h, tile.h, menu.h, file.h, struct.h
付加スイッチ　　/AL, /Zp, /J
/AL：ラージ・メモリ・モデル。コードおよびデータの制限はありません。
/Zp　：指定されたバイト境界上で構造体メンバをパックします。
/J　　：char型のディフォールトを，符号つきから符号なしに変更します。

４.３　リンク方法

ユーザ・プログラムとドライバ・ライブラリをリンクする場合には，以下の点に注意してください。

- 標準ライブラリとライブラリおよびユーザ・プログラムのモデルを合わせてください。
- LINK.EXEのスイッチとして/NOIを指定します。

保守／廃止

付録Ａ　リ　ス　ト

ここでは、以下のプログラム・リストを表示します。

- ●CONST.H
- ●STRUCT.H
- ●TILE.H
- ●MENU.H
- ●FILE.H
- ●AGDCMAP.H
- ●DEMO.C
- ●WIN.C
- ●MOUSE.C
- ●GIO.C

A.1　CONST.H

CONST.H

```
00001 /*****************************************************************************
00002  *
00003  *      const.h
00004  *
00005  *              定数定義用インクルードファイル      (Ver1.0)
00006  *
00007  *****************************************************************************/
00008
00009 /*
00010  * 変数型  定義
00011  */
00012 typedef         unsigned char  uchar;
00013 typedef         unsigned int   uint;
00014 typedef         unsigned long  ulong;
00015 /*
00016  * 定数の定義
00017  */
00018 #define         ERROR   -1
00019 #define         OK       0
00020 #define         BREAK    1
00021 #define         NUL      0
00022
00023 #define         ON       1
00024 #define         OFF      0
00025 #define         PULL     2
00026 #define         PUSH     3
00027 /*
00028  *  クリッピング  モード
00029  */
00030 #define         INSIDE   0
00031 #define         OUTSIDE  2
00032 #define         NON      1
00033 /*
00034  *  レクタングルの辺
00035  */
00036 #define         LINE_TOP    1           /* ｳｴ   */
00037 #define         LINE_LEFT   2           /* ﾋﾀﾞﾘ */
00038 #define         LINE_RIGHT  3           /* ﾐｷﾞ  */
00039 #define         LINE_BUTTOM 4           /* ｼﾀ   */
00040 /*
00041  *  コピー  コード
00042  */
00043 #define         COPY_AA     0x7800      /* Address => Address     */
00044 #define         COPY_AC     0x8000      /* Address => Coodinete   */
00045 #define         COPY_CA     0x7c00      /* Coodonate => Adress    */
00046 #define         COPY_CC     0x8400      /* Coodinate => Coodinate */
00047 /*
00048  *  プレーン展開の種類
00049  */
00050 #define         S_M         0x0008      /* ｼﾝｸﾞﾙ => ﾏﾙﾁ   */
00051 #define         M_S         0x0004      /* ﾏﾙﾁ   => ｼﾝｸﾞﾙ */
00052 #define         M_M         0x000c      /* ﾏﾙﾁ   => ﾏﾙﾁ   */
00053 /*
00054  *  回転モード
00055  */
00056 #define         ROT_0       0           /*   0°  ｶｲﾃﾝ */
00057 #define         ROT_90      1           /*  90°   "   */
00058 #define         ROT_180     2           /* 180°   "   */
00059 #define         ROT_270     3           /* 270°   "   */
00060 /*
00061  *  拡大・縮小フラグ
00062  */
00063 #define         ES16_16     0           /* 16 / 16 (  x1.0   ) */
00064 #define         ES16_15     1           /* 16 / 15 (  x1.066 ) */
00065 #define         ES16_14     2           /* 16 / 14 (  x1.143 ) */
00066 #define         ES16_13     3           /* 16 / 13 (  x1.231 ) */
00067 #define         ES16_12     4           /* 16 / 12 (  x1.333 ) */
00068 #define         ES16_11     5           /* 16 / 11 (  x1.455 ) */
00069 #define         ES16_10     6           /* 16 / 10 (  x1.6   ) */
```

1

CONST.H

```
00070  #define          ES16_9      7           /* 16 /  9 (  x1.778 ) */
00071  #define          ES16_8      8           /* 16 /  8 (  x2.0   ) */
00072  #define          ES16_7      9           /* 16 /  7 (  x2.286 ) */
00073  #define          ES16_6     10           /* 16 /  6 (  x2.667 ) */
00074  #define          ES16_5     11           /* 16 /  5 (  x3.2   ) */
00075  #define          ES16_4     12           /* 16 /  4 (  x4.0   ) */
00076  #define          ES16_3     13           /* 16 /  3 (  x5.333 ) */
00077  #define          ES16_2     14           /* 16 /  2 (  x8.0   ) */
00078  #define          ES16_1     15           /* 16 /  1 (  x16.0  ) */
00079
00080  #define          ES15_16    -1           /* 15 / 16 (  x0.9375 ) */
00081  #define          ES14_16    -2           /* 14 / 16 (  x0.875  ) */
00082  #define          ES13_16    -3           /* 13 / 16 (  x0.8125 ) */
00083  #define          ES12_16    -4           /* 12 / 16 (  x0.75   ) */
00084  #define          ES11_16    -5           /* 11 / 16 (  x0.6875 ) */
00085  #define          ES10_16    -6           /* 10 / 16 (  x0.625  ) */
00086  #define          ES9_16     -7           /*  9 / 16 (  x0.5625 ) */
00087  #define          ES8_16     -8           /*  8 / 16 (  x0.5    ) */
00088  #define          ES7_16     -9           /*  7 / 16 (  x0.4375 ) */
00089  #define          ES6_16    -10           /*  6 / 16 (  x0.375  ) */
00090  #define          ES5_16    -11           /*  5 / 16 (  x0.3125 ) */
00091  #define          ES4_16    -12           /*  4 / 16 (  x0.25   ) */
00092  #define          ES3_16    -13           /*  3 / 16 (  x0.1875 ) */
00093  #define          ES2_16    -14           /*  2 / 16 (  x0.125  ) */
00094  #define          ES1_16    -15           /*  1 / 16 (  x0.0625 ) */
00095  /*
00096   *  プライオリティ・フラグ
00097   */
00098
00099  #define          LOW_PRI     0
00100  #define          HIGH_PRI    1
00101  /*
00102   *  クリック  コード
00103   */
00104
00105  #define          NON_CLICK   0           /* ﾉﾝ ｸﾘｯｸ    */
00106  #define          MENU_CLICK  1           /* ﾒﾆｭｰ ｽｸﾘｰﾝ */
00107  #define          SRC_CLICK   2           /* ｿｰｽ ｸﾘｰﾝ    */
00108  #define          WRK_CLICK   3           /* ﾜｰｸ ｽｸﾘｰﾝ  */
00109  /*
00110   *  画像ファイルのドットサイズ
00111   */
00112  #define          SRC_XSIZE   1920
00113  #define          SRC_YSIZE   1200
00114  /*
00115   *  標準座標系の定義
00116   */
00117  #define          STD_ORG     0x3e581
00118  #define          STD_ORGD    0
00119  #define          STD_PDISP   0x400001
00120  #define          STD_PITCH   0x281
00121  /*
00122   * カラー  コード----------------------  PLANE  0  1  2  3 -----
00123   *                                              R  G  B  I
00124   */
00125  #define      BLACK       0               /*     0  0  0  0   */
00126  #define      RED         1               /*     1  0  0  0   */
00127  #define      GREEN       2               /*     0  1  0  0   */
00128  #define      YELLOW      3               /*     1  1  0  0   */
00129  #define      BLUE        4               /*     0  0  1  0   */
00130  #define      MAGENTA     5               /*     1  0  1  0   */
00131  #define      CYAN        6               /*     0  1  1  0   */
00132  #define      WHITE       7               /*     1  1  1  0   */
00133  #define      ITEN        8               /*     0  0  0  1   */
00134  #define      RGB         7               /*     1  1  1  0   */
00135  /*
00136   * 論理演算 ---------------------------------- MOD1 ------ MOD0------
00137   *                                  PLANES =   ON          OFF
00138   */
```

2

保守／廃止

CONST.H

```
00139 #define     OR          0xc7          /* D <- D+S      D <- D    */
00140 #define     AND         0x87          /* D <- D･S      D <- D    */
00141 #define     REP         0xc9          /* D <- D+S      D <- D･/S */
00142 #define     SET         0x02          /* D <- S        D <- 0    */
00143 #define     NOR         0x02          /* D <- S        D <- 0    */
00144 #define     XOR         0x47          /* D <- D￣S     D <- D    */
00145 #define     REV         0x67          /* D <- ￣D      D <- D    */
00146 #define     PUT         0x07          /* D <- S        D <- D    */
```

A.2　STRUCT.H

保守／廃止

STRUCT.H

```
00001 /*****************************************************************************
00002  *
00003  *      struct.h
00004  *
00005  *              構造体定義用インクルードファイル    (Ver1.0)
00006  *
00007  *****************************************************************************/
00008
00009 /*
00010  * マウス型構造体
00011  */
00012 typedef struct
00013 {
00014         int     lclick  ;       /* 左クリック状態  */
00015         int     rclick  ;       /* 右クリック状態  */
00016         int     x       ;       /* カーソル位置(X) */
00017         int     y       ;       /* カーソル位置(Y) */
00018         int     dx      ;       /* X方向移動量     */
00019         int     dy      ;       /* Y方向移動量     */
00020         int     event   ;       /* イベント        */
00021         int     mode    ;       /* クリック  モード*/
00022         int     menu    ;       /* メニュー        */
00023         int     win     ;       /* ウィンドウ番号  */
00024 } Mouse;
00025 /*
00026  * ディスプレイ型構造体
00027  */
00028 typedef struct
00029 {
00030         ulong   dad     ;       /* 表示開始アドレス    */
00031         ulong   dp_pitch;       /* 1ラインのワード数 */
00032         uint    wc      ;       /* ライン間のワード数 */
00033 } Display;
00034 /*
00035  * 座標系型構造体
00036  */
00037 typedef struct
00038 {
00039         ulong   eadorg  ;       /* 座標原点のアドレス    */
00040         uchar   dadorg  ;       /* 座標原点のドット位置 */
00041         uint    pitch   ;       /* 水平方向のワード数    */
00042         ulong   pdisp   ;       /* プレーン間の変位量    */
00043         uint    pmax    ;       /* 最大プレーン          */
00044 } Coodinate;
00045 /*
00046  * クリッピング型構造体
00047  */
00048 typedef struct
00049 {
00050         uint    clip_mode;      /* クリッピング  モード  */
00051         int     xmin    ;       /* 最小 X座標            */
00052         int     ymin    ;       /* 最小 Y座標            */
00053         int     xmax    ;       /* 最大 X座標            */
00054         int     ymax    ;       /* 最大 Y座標            */
00055 } Clip;
00056 /*
00057  * コピー型構造体
00058  */
00059 typedef struct
00060 {
00061         uint       mode ;       /* 転送位置の指定       */
00062         int        x    ;       /* 転送先の開始座標  X  */
00063         int        y    ;       /*                   Y  */
00064         int        xs   ;       /* 転送源の開始座標  X  */
00065         int        ys   ;       /*                   Y  */
00066         int        dh   ;       /* 横サイズ             */
00067         int        dv   ;       /* 縦サイズ             */
00068         ulong      ead1 ;       /* 絶対番地指定時の転送先アドレス    */
00069         uchar      dad1 ;       /*                          ドット位置 */
```

STRUCT.H

```
00070         ulong       ead2 ;        /* 絶対番地指定時の転送源アドレス    */
00071         uchar       dad2 ;        /*                          ドット位置 */
00072         ulong       ead3 ;        /* マスク・データの開始アドレス      */
00073 } Copy;
00074 /*
00075  * ウィンドウ型構造体
00076  */
00077 typedef struct
00078 {
00079         int          no  ;        /* ウィンドウ番号             */
00080         int          pri ;        /* プライオリティ  フラグ */
00081         int          es  ;        /* 拡大・縮小  フラグ     */
00082         int          rot ;        /* 回転  フラグ           */
00083         Coodinate    src ;        /* ソースエリアの座標系   */
00084         Coodinate    dst ;        /* 表示エリア  の座標系   */
00085         Coodinate    wrk ;        /* ワークエリアの座標系   */
00086         Copy         copy;        /* ウィンドウ位置、サイズ */
00087         Clip         dstclip;     /* 表示エリア  の領域     */
00088         Clip         srcclip;     /* ソースエリアの領域     */
00089 } Window;
00090 /*
00091  * パレット型構造体
00092  */
00093 typedef struct
00094 {
00095         uchar   red[16]  ;        /* 赤の発色量  */
00096         uchar   green[16];        /* 緑の発色量  */
00097         uchar   blue[16] ;        /* 青の発色量  */
00098 } Palette;
00099 /*
00100  * タイリング型構造体
00101  */
00102 typedef struct
00103 {
00104         int             plane   ;       /* プレーン数  */
00105         uint far        *ptr    ;       /* データ配列のポインタ */
00106         Coodinate       org     ;       /* 座標系              */
00107         int             dh      ;       /* ライン間のワード数 */
00108         int             dv      ;       /* ライン間のワード数 */
00109         uchar           mod     ;
00110         uint            color   ;
00111         ulong           off_adr ;
00112 } Tilling;
00113 /*
00114  * フォント型構造体
00115  */
00116 typedef struct
00117 {
00118 int             cnt;            /* 重ね書き数   */
00119 int             magh;           /* 倍率         */
00120 int             magv;           /* 倍率         */
00121 int             slant;          /* 傾き         */
00122 int             face;           /* 傾き         */
00123 } Font;
00124
00125
```

2

A.3 TILE.H

保守／廃止

TILE.H

```
00001 /*****************************************************************************
00002  *
00003  *      tile.h
00004  *
00005  *              タイリング設定用インクルードファイル          (Ver1.0)
00006  *
00007  *****************************************************************************/
00008
00009 uint    logo[32][6] =
00010 {
00011 {       0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00012 {       0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00013 {       0x0000,0x0030,0x0000,0x0000,0x0000,0x0000 },
00014 {       0x0180,0x0038,0x03ff,0xfffe,0x0000,0x7f00 },
00015 {       0x03c0,0x003c,0x01ff,0xfffe,0x0003,0xff80 },
00016 {       0x07e0,0x003e,0x00ff,0xfffc,0x001f,0xff80 },
00017 {       0x0ff0,0x007f,0x007f,0xc07c,0x007f,0xffc0 },
00018 {       0x1ff8,0x007c,0x007f,0xc03c,0x00fe,0x1fc0 },
00019 {       0x3ffc,0x0078,0x007f,0x803c,0x01f8,0x0fe0 },
00020 {       0x1ffe,0x0070,0x007f,0x801c,0x03f0,0x0fe0 },
00021 {       0x0fff,0x0060,0x00ff,0x8018,0x07e0,0x07e0 },
00022 {       0x0fff,0x80e0,0x00ff,0x8008,0x0fc0,0x0380 },
00023 {       0x0fff,0xc0e0,0x00ff,0x8300,0x1fc0,0x0000 },
00024 {       0x0fff,0xe0c0,0x00ff,0x0700,0x1f80,0x0000 },
00025 {       0x0fff,0xf0c0,0x00ff,0x0e00,0x3f80,0x0000 },
00026 {       0x0eff,0xf9c0,0x01ff,0x3e00,0x3f80,0x0000 },
00027 {       0x0c7f,0xfdc0,0x03ff,0xfe00,0x3f80,0x0000 },
00028 {       0x1c3f,0xfd80,0x07ff,0xfe00,0x3f80,0x0000 },
00029 {       0x1c3f,0xff80,0x03fe,0xfe00,0x3f80,0x0000 },
00030 {       0x1c1f,0xff80,0x01fe,0x1c00,0x3fc0,0x0000 },
00031 {       0x1c0f,0xff80,0x03fe,0x0c00,0x3fe0,0x0080 },
00032 {       0x1807,0xff80,0x03fe,0x0c00,0x3fe0,0x0380 },
00033 {       0x3803,0xff00,0x03fe,0x0c20,0x3ff8,0x0780 },
00034 {       0x3801,0xff00,0x03fc,0x0060,0x3ffe,0x3f00 },
00035 {       0x3800,0xff00,0x03fc,0x0060,0x1fff,0xfe00 },
00036 {       0x3e00,0x7f00,0x07fc,0x00c0,0x1fff,0xfe00 },
00037 {       0x3f00,0x3e00,0x07fc,0x01c0,0x0fff,0xfc00 },
00038 {       0x7f00,0x1e00,0x07f8,0x07c0,0x07ff,0xfc00 },
00039 {       0x7f00,0x1e00,0x0fff,0xffc0,0x03ff,0xf800 },
00040 {       0x7c00,0x0e00,0x1fff,0xff80,0x01ff,0xf000 },
00041 {       0x7000,0x0c00,0x7fff,0xff00,0x001f,0xf000 },
00042 {       0x0000,0x0000,0x0000,0x0000,0x0000,0xf000 }
00043 };
00044 static uint cur_tbl[2][16]=
00045 {
00046 { 0x0000,0xe000,0xf800,0x7e00,0x7f80,0x3fe0,0x3ff8,0x1ffe,
00047   0x1fff,0x0f00,0x0f00,0x0700,0x0700,0x0300,0x0300,0x0100  },
00048 { 0x0000,0x0000,0x3800,0x3c00,0x1e0e,0x0f1e,0x079c,0x0bf8,
00049   0x0df8,0x36f8,0x1b78,0x2cff,0x35ff,0x3bff,0x07ff,0x038f  }
00050 };
00051 static uint tile[6][3][16] =
00052 {
00053 /* tiling no 0 */
00054 {
00055 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00056   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00057 { 0xaaaa,0x5555,0xaaaa,0x5555,0xaaaa,0x5555,0xaaaa,0x5555,
00058   0xaaaa,0x5555,0xaaaa,0x5555,0xaaaa,0x5555,0xaaaa,0x5555 },
00059 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00060   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00061 },
00062 /* tiling no 1 */
00063 {
00064 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00065   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00066 { 0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,
00067   0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,0x0000,0xffff },
00068 { 0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,
00069   0x0000,0xffff,0x0000,0xffff,0x0000,0xffff,0x0000,0xffff },
```

TILE.H

```
00070 },
00071 /* tiling no 2 */
00072 {
00073 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00074   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00075 { 0xcccc,0xcccc,0x3333,0x3333,0xcccc,0xcccc,0x3333,0x3333,
00076   0xcccc,0xcccc,0x3333,0x3333,0xcccc,0xcccc,0x3333,0x3333,},
00077 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00078   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00079 },
00080 /* tiling no 3 */
00081 {
00082 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00083   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00084 { 0x4444,0x8888,0x1111,0x2222,0x4444,0x8888,0x1111,0x2222,
00085   0x4444,0x8888,0x1111,0x2222,0x4444,0x8888,0x1111,0x2222 },
00086 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00087   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00088 },
00089 /* tiling no 4 */
00090 {
00091 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00092   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00093 { 0x8888,0x0000,0x2222,0x0000,0x8888,0x0000,0x2222,0x0000,
00094   0x8888,0x0000,0x2222,0x0000,0x8888,0x0000,0x2222,0x0000,},
00095 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00096   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00097 },
00098 /* tiling no 5 */
00099 {
00100 { 0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,
00101   0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000 },
00102 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00103   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00104 { 0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,
00105   0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff,0xffff },
00106 }
00107 };
00108
```

2

A.4　MENU.H

保守／廃止

MENU.H

```
00001 /*****************************************************************************
00002  *
00003  *      menu.h
00004  *
00005  *              メニュー定義用インクルードファイル (Ver1.0)
00006  *
00007  *****************************************************************************/
00008
00009 #define MENU_MAX 8
00010 struct menu_tpl
00011 {
00012 int         no   ;      /* menu no.          */
00013 int         x    ;      /* start x position */
00014 int         y .  ;      /* start y           */
00015 int         dx   ;      /* width             */
00016 int         dy   ;      /* length            */
00017 int         len  ;      /* string length     */
00018 uchar   str[30] ;      /* memu string       */
00019 };
00020 struct menu_tpl menu[9] =
00021 {
00022 { 0,      0,  0,  0,  0,   0,""},
00023 { 1,     36,314,104, 18,   4,"EXIT"},
00024 { 2,     36,290,104, 18,   8,"PRIORITY"},
00025 { 3,    160,338,104, 18,   4,"MOVE"},
00026 { 4,    160,314,104, 18,   6,"RESIZE"},
00027 { 5,    160,290,104, 18,   6,"SCROLL"},
00028 { 6,    160,266,104, 18,   8,"ROTATION"},
00029 { 7,    160,242,104, 18,  11,"ENLARGEMENT"},
00030 { 8,    160,218,104, 18,   6,"SHRINK"}
00031 };
00032 #define     EXIT 1
00033 #define     PRIORITY 2
00034 #define     MOVE 3
00035 #define     RESIZE 4
00036 #define     SCROLL 5
00037 #define     ROTATION 6
00038 #define     ENLARGE 7
00039 #define     SHRINK 8
00040
```

A.5　FILE.H

保守／廃止

FILE.H

```
00001 /*****************************************************************************
00002  *
00003  *      file.h
00004  *
00005  *              画像ファイル定義用インクルードファイル       (Ver1.0)
00006  *
00007  *****************************************************************************/
00008 /*
00009  *  画像ファイル処理用変数エリア
00010  */
00011 #define        HED_SIZE 512
00012
00013 uchar   scan_color             ;/* 色指定                          */
00014 uchar   scan_pic               ;/* データ構成                      */
00015 uchar   scan_bit               ;/* 色指定                          */
00016 uchar   scan_dither            ;/* 色指定                          */
00017 uint    scan_xsize             ;/* 色指定                          */
00018 uint    scan_ysize             ;/* 色指定                          */
00019
00020 struct  file_tpl
00021         {
00022         uchar   fname[12]       ;/* file name                        */
00023         uchar   r1              ;
00024         uchar   data[6]         ;/* pixcel/plane                     */
00025         uchar   r2              ;
00026         uchar   bit             ;/* 階調数                          */
00027         uchar   r3              ;
00028         uchar   xsize[4]        ;/* 主走査サイズ                    */
00029         uchar   r4              ;
00030         uchar   ysize[4]        ;/* 副走査サイズ                    */
00031         uchar   r5              ;
00032         uchar   color[5]        ;/* カラー指定                      */
00033         uchar   r6              ;
00034         uint    dtl             ;/* data length                      */
00035         uchar   reserved[472]   ;
00036         }file_hed ;
```

A.6　AGDCMAP.H

保守／廃止

AGDCMAP.H

```
00001  /*****************************************************************************
00002   *
00003   *       agdcmap.h
00004   *
00005   *               ＡＧＤＣ２マッピング用インクルードファイル　(Ver1.0)
00006   *
00007   *****************************************************************************/
00008
00009  /* AGDC2 BOARD ADDRESS PORT DEFINE */
00010
00011  #define BASE_MEM 0xc0000000L
00012  #define BASE_PTR 0xc7f00000L
00013  #define BASE_PAL 0x87f80000L
00014  #define VRAM_ADR 0xa0000000L
00015
00016  uint    *DISP_CTRL = ( uint  far *)(BASE_PTR + 0x00701);
00017  uint    *DP_PITCH  = ( uint  far *)(BASE_PTR + 0x00721);
00018  ulong   *DAD       = ( ulong far *)(BASE_PTR + 0x00741);
00019  uchar   *WC        = ( uchar far *)(BASE_PTR + 0x00771);
00020  uint    *GCSRX     = ( uint  far *)(BASE_PTR + 0x00781);
00021  uint    *GCSRYS    = ( uint  far *)(BASE_PTR + 0x007a1);
00022  uint    *GCSRYE    = ( uint  far *)(BASE_PTR + 0x007c1);
00023
00024  uint    *HS        = ( uint  far *)(BASE_PTR + 0x007e1);
00025  uint    *HBP       = ( uint  far *)(BASE_PTR + 0x007e1);
00026  uint    *HH        = ( uint  far *)(BASE_PTR + 0x007e1);
00027  uint    *HD        = ( uint  far *)(BASE_PTR + 0x007e1);
00028  uint    *HFP       = ( uint  far *)(BASE_PTR + 0x007e1);
00029  uint    *VS        = ( uint  far *)(BASE_PTR + 0x007e1);
00030  uint    *VBP       = ( uint  far *)(BASE_PTR + 0x007e1);
00031  uint    *LF        = ( uint  far *)(BASE_PTR + 0x007e1);
00032  uint    *VFP       = ( uint  far *)(BASE_PTR + 0x007e1);
00033
00034  ulong   *EADORG    = ( ulong far *)(BASE_PTR + 0x00001);
00035  uchar   *DADORG    = ( uchar far *)(BASE_PTR + 0x00031);
00036  ulong   *EADORGS   = ( ulong far *)(BASE_PTR + 0x00201);
00037  char    *DADORGS   = ( uchar far *)(BASE_PTR + 0x00231);
00038  uint    *PITCHD    = ( uint  far *)(BASE_PTR + 0x005a1);
00039  uint    *PITCHS    = ( uint  far *)(BASE_PTR + 0x00581);
00040  uint    *PLANES    = ( uint  far *)(BASE_PTR + 0x005e1);
00041  uchar   *MOD       = ( uchar far *)(BASE_PTR + 0x00161);
00042  ulong   *PTNP      = ( ulong far *)(BASE_PTR + 0x00181);
00043
00044  uint    *PTNCNT    = ( uint  far *)(BASE_PTR + 0x00601);
00045  uint    *PTNH      = ( uint  far *)(BASE_PTR + 0x006a1);
00046  ulong   *STACK     = ( ulong far *)(BASE_PTR + 0x001c1);
00047  uint    *STMAX     = ( uint  far *)(BASE_PTR + 0x005c1);
00048
00049  ulong   *PDISPS    = ( ulong far *)(BASE_PTR + 0x000c1);
00050  ulong   *PDISPD    = ( ulong far *)(BASE_PTR + 0x00101);
00051  uint    *PMAX      = ( uint  far *)(BASE_PTR + 0x00141);
00052
00053  uchar   *CLIP      = ( uchar far *)(BASE_PTR + 0x006d1);
00054  uchar   *MAG       = ( uchar far *)(BASE_PTR + 0x006c1);
00055  uint    *XCLMIN    = ( uint  far *)(BASE_PTR + 0x00621);
00056  uint    *YCLMIN    = ( uint  far *)(BASE_PTR + 0x00641);
00057  uint    *XCLMAX    = ( uint  far *)(BASE_PTR + 0x00661);
00058  uint    *YCLMAX    = ( uint  far *)(BASE_PTR + 0x00681);
00059
00060  ulong   *EAD1      = ( ulong far *)(BASE_PTR + 0x00041);
00061  ulong   *EAD2      = ( ulong far *)(BASE_PTR + 0x00081);
00062  ulong   *EAD3      = ( ulong far *)(BASE_PTR + 0x00241);
00063  uchar   *DAD1      = ( uchar far *)(BASE_PTR + 0x00071);
00064  uchar   *DAD2      = ( uchar far *)(BASE_PTR + 0x000b1);
00065
00066  uint    *X         = ( int  far *)(BASE_PTR + 0x00401);
00067  uint    *Y         = ( int  far *)(BASE_PTR + 0x00421);
00068  uint    *DX        = ( int  far *)(BASE_PTR + 0x00441);
00069  uint    *DY        = ( int  far *)(BASE_PTR + 0x00461);
```

AGDCMAP.H

```
00070 uint    *XS         = ( int  far *)(BASE_PTR + 0x00481);
00071 uint    *YS         = ( int  far *)(BASE_PTR + 0x004a1);
00072 uint    *XE         = ( int  far *)(BASE_PTR + 0x004c1);
00073 uint    *YE         = ( int  far *)(BASE_PTR + 0x004e1);
00074 uint    *XC         = ( int  far *)(BASE_PTR + 0x00501);
00075 uint    *YC         = ( int  far *)(BASE_PTR + 0x00521);
00076 uint    *DH         = ( uint  far *)(BASE_PTR + 0x00541);
00077 uint    *DV         = ( uint  far *)(BASE_PTR + 0x00561);
00078
00079 uint    *COMMAND    = ( uint  far *)(BASE_PTR + 0x006e1);
00080 uchar   *CTRL2      = ( uchar far *)(BASE_PTR + 0x003b1);
00081 uchar   *BANK       = ( uchar far *)(BASE_PTR + 0x003c1);
00082 uchar   *CTRL       = ( uchar far *)(BASE_PTR + 0x003d1);
00083 uint    *STATUS     = ( uint  far *)(BASE_PTR + 0x003c1);
00084 uint    *PGPORT     = ( uint  far *)(BASE_PTR + 0x003e1);
00085
```

2

A.7　DEMO.C

保守／廃止

DEMO.C

```
00001 /*****************************************************************************
00002  *
00003  *      μＰＤ７２１２３ デモンストレーション・プログラム
00004  *
00005  *              ( メイン処理 )
00006  *
00007  *      ｺﾝﾊﾟｲﾙ･ｽｲｯﾁ : /AL /J /Zp
00008  *      ﾘﾝｸ ﾌｧｲﾙ   : Win.Obj + Gio + Mouse
00009  *
00010  *****************************************************************************/
00011
00012 #include <stdio.h>
00013 #include <fcntl.h>
00014 #include <sys¥types.h>
00015 #include <sys¥stat.h>
00016
00017 #include "const.h"
00018 #include "struct.h"
00019 #include "tile.h"
00020 #include "menu.h"
00021 #include "file.h"
00022
00023 Coodinate scaner,on_screen,work1,work2,cur_screen;
00024 Coodinate tile_org;
00025 Mouse   M;
00026 Window  W[2];
00027 Clip    work_scrn,src_scrn,menu_scrn,src_area,dst_area;
00028 Display dsp;
00029 Font    fontZen,fontHan,fontSlt;
00030 Tilling pat[6];
00031
00032 main(argc,argv)
00033         int      argc;
00034         uchar   *argv[];
00035 {
00036         printf("¥x1b[>1h¥x1b[2J¥x1b[>5h");
00037         if ( sys_init(argc,argv) == ERROR )
00038         {
00039                 WoffDisplay();
00040                 printf("¥x1b[>1l¥x1b[>5l");
00041                 exit(-1);
00042         }
00043         while(M.event != EXIT)
00044         {
00045                 if ( cur_input(&M) != NON_CLICK)
00046                 {
00047                         cur_locate(&M);
00048                         switch( M.mode )
00049                         {
00050                                 case MENU_CLICK:
00051                                                         menu_pro(&M);
00052                                                         break;
00053                                 case WRK_CLICK:
00054                                                         work_pro(&M);
00055                                                         break;
00056                                 case SRC_CLICK :
00057                                                         source_pro(&M);
00058                                                         break;
00059                         }
00060                 }
00061                 cur_locate(&M);
00062         }
00063         WoffDisplay();
00064         printf("¥x1b[>1l¥x1b[>5l");
00065 }
00066 /*
```

1

DEMO.C

```
00066 */
00067 /*
00068  * システム初期化
00069  */
00070
00071 sys_init(argc,argv)
00072 int     argc;
00073 uchar   *argv[];
00074 {
00075 uchar   fname[30];
00076 int     i;
00077
00078 /* ディスプレイ構造体  設定 */
00079
00080         dsp.dad         = 0;
00081         dsp.dp_pitch    = 0x281;
00082         dsp.wc          = 0x27;
00083
00084         Wdisplay(&dsp);                 /* 表示許可  */
00085
00086 /* 座標系設定 */
00087         scaner.eadorg = 0x32ca81;       /* 源画像エリア  */
00088         scaner.dadorg = 0;
00089         scaner.pitch  = 1920 / 16;
00090         scaner.pdisp  = STD_PDISP;
00091         scaner.pmax   = 4;
00092
00093         on_screen.eadorg = 0x3e581;     /* オン・スクリーン用 */
00094         on_screen.dadorg = 0;
00095         on_screen.pitch  = 640 / 16;
00096         on_screen.pdisp  = STD_PDISP;
00097         on_screen.pmax   = 4;
00098
00099         work1.eadorg = 0x7d281;         /* ワーク・バッファ１ */
00100         work1.dadorg = 0;
00101         work1.pitch  = 640 / 16;
00102         work1.pdisp  = STD_PDISP;
00103         work1.pmax   = 4;
00104
00105         work2.eadorg = 0xbba81;         /* ワーク・バッファ２ */
00106         work2.dadorg = 0;
00107         work2.pitch  = 640 / 16;
00108         work2.pdisp  = STD_PDISP;
00109         work2.pmax   = 4;
00110
00111         cur_screen.eadorg = 0xc3e581;   /* カーソル・スクリーン */
00112         cur_screen.dadorg = 0;
00113         cur_screen.pitch  = 640 / 16;
00114         cur_screen.pdisp  = STD_PDISP;
00115         cur_screen.pmax   = 1;
00116 /* クリッピング構造体定義 */
00117         work_scrn.clip_mode = INSIDE;
00118         work_scrn.xmin      = 314;
00119         work_scrn.xmax      = 314+312;
00120         work_scrn.ymin      = 14;
00121         work_scrn.ymax      = 14+356;
00122
00123         src_scrn.clip_mode  = INSIDE;
00124         src_scrn.xmin       = 10;
00125         src_scrn.xmax       = 10+240;
00126         src_scrn.ymin       = 10;
00127         src_scrn.ymax       = 10+150;
00128
00129         menu_scrn.clip_mode = INSIDE;
00130         menu_scrn.xmin      = 10;
00131         menu_scrn.xmax      = 10+270;
00132         menu_scrn.ymin      = 140;
00133         menu_scrn.ymax      = 203+140;
00134
```

DEMO.C

```
00135         src_area.clip_mode = INSIDE;
00136         src_area.xmin      =  0;
00137         src_area.xmax      =  SRC_XSIZE-1;
00138         src_area.ymin      =  0;
00139         src_area.ymax      =  SRC_YSIZE-1;
00140
00141         dst_area.clip_mode = INSIDE;
00142         dst_area.xmin      =  0;
00143         dst_area.xmax      =  640-1;
00144         dst_area.ymin      =  0;
00145         dst_area.ymax      =  400-1;
00146 /* ウィンドウ定義 */
00147         W[0].no          = 0;
00148         W[0].pri         = HIGH_PRI;
00149         W[0].rot         = ROT_0;
00150         W[0].es          = ES16_16;
00151         W[0].src         = scaner;                /* ウィンドウ［0］       */
00152         W[0].wrk         = work1;
00153         W[0].dst         = on_screen;
00154         W[0].srcclip     = src_area;
00155         W[0].dstclip     = work_scrn;
00156         W[0].copy.mode   = COPY_CC+M_M;
00157         W[0].copy.x      = 400;
00158         W[0].copy.y      = 300;
00159         W[0].copy.xs     = 100;
00160         W[0].copy.ys     = 400;
00161         W[0].copy.dh     = 149;
00162         W[0].copy.dv     = 199;
00163
00164         W[1].no          = 1;
00165         W[1].pri         = LOW_PRI;
00166         W[1].rot         = ROT_90;
00167         W[1].es          = ES16_10;
00168         W[1].src         = scaner;                /* ウィンドウ［1］       */
00169         W[1].wrk         = work2;
00170         W[1].dst         = on_screen;
00171         W[1].srcclip     = src_area;
00172         W[1].dstclip     = work_scrn;
00173         W[1].copy.mode   = COPY_CC+M_M;
00174         W[1].copy.x      = 320;
00175         W[1].copy.y      = 350;
00176         W[1].copy.xs     = 1500;
00177         W[1].copy.ys     = 99;
00178         W[1].copy.dh     = 99;
00179         W[1].copy.dv     = 99;
00180 /* タイリングパターンの座標系設定 */
00181         tile_org.eadorg = 0x32d201;
00182         tile_org.dadorg = 0;
00183         tile_org.pitch  = 0x11;
00184         tile_org.pdisp  = 0x400001;
00185         tile_org.pmax   = 4;
00186 /* タイリングパターンの設定 */
00187         for (i=0;i<6;i++)
00188         {
00189                 pat[i].mod        = SET;
00190                 pat[i].color      = RGB;
00191                 pat[i].org        = tile_org;
00192                 pat[i].ptr        = (uint far *)&tile[i][0][0];
00193                 pat[i].dh         = 16*16-1;
00194                 pat[i].dv         = 0;
00195                 pat[i].off_adr    = (ulong)(i*40);
00196                 pat[i].plane      = 3;
00197                 GsetTile(&pat[i]);
00198         }
00199         pat[5].mod   = 0x74;
00200         pat[5].color = ITEN;
00201 /* フォント型の定義 */
00202         fontHan.cnt     = 2;            /* 半角 */
00203         fontHan.magh    = ES8_16;
```

3

DEMO.C

```
00204           fontHan.magv    = ES16_16;
00205           fontHan.slant   = 0;
00206           fontHan.face    = 8;
00207
00208           fontZen.cnt     = 3;            /* 全角 */
00209           fontZen.magh    = ES16_16;
00210           fontZen.magv    = ES16_16;
00211           fontZen.slant   = 0;
00212           fontZen.face    = 17;
00213
00214           fontSlt.cnt     = 3;            /* 傾斜 */
00215           fontSlt.magh    = ES16_16;
00216           fontSlt.magv    = ES16_8;
00217           fontSlt.slant   = 10;
00218           fontSlt.face    = 18;
00219
00220           M.x     = 300;
00221           M.y     = 200;
00222           M.event = MOVE;
00223           Minit(&M);
00224
00225           screen_pro(&M);
00226           strcpy(fname,"demo.im");
00227           if ( argc != 0)
00228           {
00229                   if ( strnicmp(argv[1],"-L",2) == 0 )
00230                   {
00231                           strcpy(fname,argv[1]+2);
00232                   }
00233                   if ( strnicmp(argv[1],"-N",2) == 0 )
00234                   {
00235                           strcpy(fname,"");
00236                   }
00237           }
00238           if ( strcmpi(fname,"") != 0 )
00239           {
00240                   if ( load_pro(fname)==0)
00241                   {
00242                           printf("¥n画像ファイルをロード出来ません　!¥n");
00243                           return(-1);
00244                   }
00245           }
00246           source_copy();
00247           cur_setting();
00248           Wsetup(&W[0]);
00249           Wsetup(&W[1]);
00250           Wopen(&W[0]);
00251           Wopen(&W[1]);
00252   }
00253   /*
```

DEMO.C

```
00253 */
00254 /*
00255  * メニュークリック処理
00256  */
00257 menu_pro(m)
00258 Mouse *m;
00259 {
00260 int     x,y,dh,dv,i,j;
00261
00262         if ( m->menu ==  0 )
00263         {
00264                 MclickOff(m);
00265         }
00266         i = m->event;
00267         j = m->menu;
00268         GorgDst(&on_screen);
00269
00270         x=menu[i].x;
00271         y=menu[i].y;
00272         dh=menu[i].dx;
00273         dv=menu[i].dy;
00274         Gflame(PUSH,x,y,dh,dv);
00275         Gcolor(REP,CYAN);
00276         Gprint(x+3,y-2,&menu[i].str[0],&fontHan);
00277
00278         x=menu[j].x;
00279         y=menu[j].y;
00280         dh=menu[j].dx;
00281         dv=menu[j].dy;
00282         Gflame(PULL,x,y,dh,dv);
00283         Gcolor(REP,RED);
00284         Gprint(x+3,y-2,&menu[j].str[0],&fontHan);
00285         Mbell();
00286         MclickOff(m);
00287         if ( j == PRIORITY )
00288         {
00289                 priority_pro(m);
00290
00291                 x=menu[j].x; y=menu[j].y; dh=menu[j].dx; dv=menu[j].dy;
00292                 Gflame(PUSH,x,y,dh,dv);
00293                 Gcolor(REP,BLACK);
00294                 Gprint(x+3,y-2,&menu[j].str[0],&fontHan);
00295
00296                 x=menu[i].x; y=menu[i].y; dh=menu[i].dx; dv=menu[i].dy;
00297                 Gflame(PULL,x,y,dh,dv);
00298                 Gcolor(REP,WHITE);
00299                 Gprint(x+3,y-2,&menu[i].str[0],&fontHan);
00300
00301                 return;
00302         }
00303         m->event = m->menu;
00304         cur_locate(m);
00305 }
00306 /*
```

保守／廃止

DEMO.C

```
00306 */
00307 /*
00308  * ワーク・スクリーン・クリック処理
00309  */
00310 work_pro(m)
00311 Mouse  *m;
00312 {
00313         if ( Wclick(m) == 0 )
00314         {
00315                 MclickOff(m);
00316                 return;
00317         }
00318         Mbell();
00319         switch(m->event)
00320         {
00321                 case MOVE     : move_pro(m);            break;
00322                 case RESIZE   : resize_pro(m);          break;
00323                 case SCROLL   : scroll_pro(m);          break;
00324                 case ROTATION : rotation_pro(m);        break;
00325                 case ENLARGE  : enlarge_pro(m);         break;
00326                 case SHRINK   : shrink_pro(m);          break;
00327
00328                 default :       MclickOff(m);   break;
00329         }
00330 }
00331 /*
```

DEMO.C

```
00331 */
00332 /*
00333  * ソース・スクリーン・クリック処理
00334  */
00335 source_pro(m)
00336 Mouse *m;
00337 {
00338 int     x,y,xs,ys,dh,dv,no;
00339 Copy    src;
00340
00341         no = WgetPriority(HIGH_PRI,&src);
00342         xs  = (m->x-src_scrn.xmin) * 8;
00343         ys  = (m->y-src_scrn.ymin) * 8;
00344         if ( W[no].rot == ROT_90 || W[no].rot == ROT_270 )
00345         {
00346                 dh = W[no].copy.dv;
00347                 dv = W[no].copy.dh;
00348         }
00349         else
00350         {
00351                 dh = W[no].copy.dh;
00352                 dv = W[no].copy.dv;
00353         }
00354         dh = GcalES(-W[no].es,dh);
00355         dv = GcalES(-W[no].es,dv);
00356
00357         if ( xs    < src_area.xmin ) xs = src_area.xmin;
00358         if ( xs+dh > src_area.xmax ) xs = src_area.xmax-dh;
00359         if ( ys-dv < src_area.ymin ) ys = dv;
00360         if ( ys    > src_area.ymax ) ys = src_area.ymax;
00361
00362         W[no].copy.xs = xs;
00363         W[no].copy.ys = ys;
00364
00365         Wsetup(&W[no]);
00366         Wopen(&W[no]);
00367         view_area(no);
00368         Mbell();
00369         MclickOff(m);
00370 }
00371 /*
```

7

保守／廃止

DEMO.C

```
00371 */
00372 /*
00373  * ＰＲＩＯＲＩＴＹ処理　（ウィンドウの優先順位変更　）
00374  */
00375 priority_pro(m)
00376 Mouse *m;
00377 {
00378         Wchgpriority(&W[0],&W[1]);
00379         MclickOff(m);
00380 }
00381 /*
```

保守／廃止

DEMO.C

```
00381 */
00382 /*
00383  *      MOVE処理　（ウィンドウの移動　）
00384  */
00385 move_pro(m)
00386 Mouse  *m;
00387 {
00388 int     xoff,yoff,i;
00389         i = m->win;
00390         xoff = m->x - W[i].copy.x;
00391         yoff = m->y - W[i].copy.y;
00392         while(Mstatus(m))
00393         {
00394                 if( m->dx == 0 && m->dy==0)
00395                 {
00396                         cur_locate(m);
00397                         continue;
00398                 }
00399                 Wmove(&W[i],m->dx,m->dy);
00400                 m->x = W[i].copy.x+xoff;
00401                 m->y = W[i].copy.y+yoff;
00402                 cur_locate(m);
00403         }
00404         cur_locate(m);
00405 }
00406 /*
```

DEMO.C

```
00406 */
00407 /*
00408  *      ＲＥＳＩＺＥ処理（ウィンドウ・サイズ変更 ）
00409  */
00410 resize_pro(m)
00411 Mouse *m;
00412 {
00413 int     dx,dy,r,mx,my,mode;
00414         mx=m->x;
00415         my=m->y;
00416         if ((mode=chk_win_line(m))== 0 )
00417         {
00418                         MclickOff(m);
00419                         return;
00420         }
00421         while(Mstatus(m))
00422         {
00423                 if ( mode == LINE_LEFT || mode==LINE_RIGHT )
00424                 {
00425                         if ( m->dx != 0 )
00426                         {
00427                                 r = Wresize(&W[m->win],mode,m->dx);
00428                                 view_area(m->win);
00429                                 mx+=r;
00430                         }
00431                 }
00432                 else
00433                 {
00434                         if ( m->dy != 0 )
00435                         {
00436                                 r = Wresize(&W[m->win],mode,m->dy);
00437                                 view_area(m->win);
00438                                 my+=r;
00439                         }
00440                 }
00441                 m->x = mx;
00442                 m->y = my;
00443                 cur_locate(m);
00444         }
00445 }
00446 /*
```

DEMO.C

```
00446 */
00447 /*
00448  *      ＳＣＲＯＬＬ処理 （ウィインドウ・スクロール ）
00449  */
00450 scroll_pro(m)
00451 Mouse *m;
00452 {
00453 int     dx,dy,r;
00454 Mouse   Mdumy;
00455         r = chk_win_position(m);
00456         switch(r)
00457         {
00458         case 1 : dx = -6 ; dy =  6 ; break;
00459         case 2 : dx =  0 ; dy =  6 ; break;
00460         case 3 : dx =  6 ; dy =  6 ; break;
00461         case 4 : dx = -6 ; dy =  0 ; break;
00462         case 5 : dx =  6 ; dy =  0 ; break;
00463         case 6 : dx = -6 ; dy = -6 ; break;
00464         case 7 : dx =  0 ; dy = -6 ; break;
00465         case 8 : dx =  6 ; dy = -6 ; break;
00466         }
00467         while(Mstatus(&Mdumy))
00468         {
00469                 Wscroll(&W[m->win],dx,dy);
00470                 view_area(m->win);
00471         }
00472         cur_locate(m);
00473 }
00474 /*
```

保守／廃止

DEMO.C

```
00474 */
00475 /*
00476  * ROTATION処理
00477  */
00478 rotation_pro(m)
00479 Mouse *m;
00480 {
00481         Wrotation(&W[m->win]);
00482         view_area(m->win);
00483         MclickOff(m);
00484 }
00485 /*
```

DEMO.C

```
00485 */
00486 /*
00487  * ＳＨＲＩＮＫ処理 （ウィンドウ縮小表示）
00488  */
00489 shrink_pro(m)
00490 Mouse *m;
00491 {
00492         Wshrink(&W[m->win]);
00493         view_area(m->win);
00494         MclickOff(m);
00495 }
00496 /*
```

DEMO.C

```
00496 */
00497 /*
00498  * ＥＮＬＡＲＧＥＭＥＮＴ処理
00499  */
00500 enlarge_pro(m)
00501 Mouse *m;
00502 {
00503         Wenlarge(&W[m->win]);
00504         view_area(m->win);
00505         MclickOff(m);
00506 }
00507 /*
```

DEMO.C

```
00507 */
00508 /*
00509  *  画像ファイル・ロード処理
00510  */
00511 load_pro(fname )
00512 uchar   *fname;
00513 {
00514 int     fd;
00515
00516         fd =  open(fname , O_BINARY | O_RDWR ,S_IREAD | S_IWRITE );
00517         if ( fd == ERROR )
00518         {
00519                 printf("¥nFILE OPEN ERROR !");
00520                 return(NUL);
00521         }
00522         if ( filechk_pro( fd ) == ERROR )
00523         {
00524                 printf("¥nFILE TYPE MISMATCH !");
00525                 close( fd );
00526                 return( NUL );
00527         }
00528         if ( put_pro(fd) == NUL )
00529         {
00530                 printf("¥nFILE PUT ERROR !");
00531                 close( fd );
00532                 return( NUL );
00533         }
00534         if ( close(fd)==ERROR )
00535         {
00536                 printf("¥nFILE CLOSE ERROR !");
00537                 return( NUL );
00538         }
00539         return(1);
00540 }
00541 /*
```

DEMO.C

```
00541 */
00542 /*
00543  *画像ファイル  チェック処理
00544  */
00545 filechk_pro( fd )
00546 int     fd;
00547 {
00548 uint    r;
00549
00550 /* ファイル・ヘッダー読込み  */
00551
00552         r = read(fd,file_hed.fname,HED_SIZE);
00553         if ( r == ERROR )
00554         {
00555                 printf("¥nファイル・ヘッダー読込み時にエラーが発生しました");
00556                 return(ERROR);
00557         }
00558
00559 /*  ファイルの種類のチェック */
00560
00561         if ( memcmp(&file_hed.data[0],"PLANE ",6) != 0 )
00562         {
00563                 printf("¥nファイルの種類が違います (pixcel/plane)");
00564                 return(ERROR);
00565         }
00566         if ( memcmp(&file_hed.bit,"1",1) != 0 )
00567         {
00568                 printf("¥n階調指定が正しくありません");
00569                 return(ERROR);
00570         }
00571         if ( sscanf(file_hed.xsize,"%04d",&scan_xsize) != 1 )
00572         {
00573                 printf("¥n主走査サイズが正しくありません");
00574                 return(ERROR);
00575         }
00576         if ( scan_xsize > SRC_XSIZE )
00577         {
00578                 printf("¥n主走査サイズが正しくありません");
00579                 return(ERROR);
00580         }
00581         if ( sscanf(file_hed.ysize,"%04d",&scan_ysize) != 1 )
00582         {
00583                 printf("¥n副走査サイズが正しくありません");
00584                 return(ERROR);
00585         }
00586         if ( scan_ysize > SRC_YSIZE )
00587         {
00588                 printf("¥n走査サイズが正しくありません");
00589                 return(ERROR);
00590         }
00591         return( OK );
00592 }
00593 /*
```

DEMO.C

```
00593 */
00594 /*
00595  *ソース画像表示処理
00596  */
00597 put_pro(fd)
00598 int     fd;
00599 {
00600 uchar   buf[500];
00601 uint    data[500];
00602 int     r,i,j,k,x,y,dh,dv;
00603 ulong   adr;
00604
00605         GorgDst(&scaner);
00606         x = src_area.xmin;
00607         y = src_area.ymax;
00608         dh = src_area.xmax-src_area.xmin;
00609         dv = src_area.ymax-src_area.ymin;
00610         Gcolor(SET,BLACK);
00611         GBoxFill(x,y,dh,dv);
00612         adr = 0xf9b01;
00613         for ( k=0;k<3;k++)
00614         {
00615                 GorgDst(&scaner);
00616                 Gpmax(1);
00617                 Gcolor(PUT,RGB);
00618                 GnonClip();
00619                 GPutA(adr,dh,dv);
00620                 for ( i = 0; i < dv+1 ; i++)
00621                 {
00622                         if ( read(fd,buf,(dh+1)/8) != (dh+1)/8 )
00623                                 return(NUL);
00624                         swab(buf,data,240);
00625                         for ( j=0;j<((dh+1)/16);j++)
00626                         {
00627                                 r=GPgport(data[j]);
00628                                 if ( r == BREAK ) break;
00629                                 if ( r == ERROR ) return(ERROR);
00630                         }
00631                 }
00632                 adr = adr + scaner.pdisp;
00633         }
00634         return(1);
00635 }
00636 /*
```

DEMO.C

```
00636 */
00637 /*
00638  *表示画面作成処理
00639  */
00640 screen_pro(m)
00641 Mouse   *m;
00642 {
00643         GorgDst(&on_screen);
00644         GtileFill(0,399,639,399,&pat[0]);
00645         title_dsp();
00646         work_dsp();
00647         source_dsp();
00648         menu_dsp(m);
00649 }
00650 /*
```

DEMO.C

```
00650 */
00651 /*
00652  * タイトル表示
00653  */
00654 title_dsp()
00655 {
00656 uint    *ptr;
00657 int     i,r;
00658 int     x,y,dh,dv;
00659         x  = 5;
00660         y  = 395;
00661         dh = 290;
00662         dv = 38;
00663         GorgDst(&on_screen);
00664         GtileFill(x,y,dh,dv,&pat[1]);
00665         Gflame(PULL,x,y,dh,dv);
00666
00667         Gcolor(REP,BLACK);
00668         Gprint(11,392,"μＰＤ７２１２３",&fontZen);
00669         Gcolor(REP,WHITE);
00670         Gprint(12,392,"μＰＤ７２１２３",&fontZen);
00671         Gcolor(REP,BLACK);
00672         Gprint(20,373,"ＤＥＭＯＮＳＴＲＡＴＩＯＮ　ＰＲＯＧＲＡＭ",&fontHan);
00673
00674         ptr = (uint *)logo;
00675         GorgDst(&on_screen);
00676         Gcolor(REP,RED);
00677         GnonClip();
00678         GPutC(195,393,95,31);
00679         for ( i=0; i < 6*32 ; i++)
00680         {
00681                 r=GPgport(*ptr++);
00682                 if ( r == BREAK ) break;
00683                 if ( r == ERROR ) return(ERROR);
00684         }
00685 }
00686 /*
```

DEMO.C

```
00686 */
00687 /*
00688  *ワーク　スクリーン表示
00689  */
00690 work_dsp()
00691 {
00692 uint    *ptr ;
00693 int     x,y,dh,dv;
00694
00695         GorgDst(&on_screen);
00696         x  = work_scrn.xmin;
00697         y  = work_scrn.ymax;
00698         dh = work_scrn.xmax-work_scrn.xmin;
00699         dv = work_scrn.ymax-work_scrn.ymin;
00700
00701         Gflame(PULL,x-7,y+20,dh+14,dv+25);
00702         GtileFill(  x-7,y+20,dh+14,dv+25,&pat[1]);
00703         Gcolor(REP,BLACK);
00704         Gprint(x+10,y+19,"Ｗｏｒｋ　Ｓｃｒｅｅｎ",&fontHan);
00705         Gcolor(REP,WHITE);
00706         Gprint(x+11,y+18,"Ｗｏｒｋ　Ｓｃｒｅｅｎ",&fontHan);
00707         Gcolor(SET,BLUE);
00708         GBoxFill(x,y,dh,dv);
00709         Gcolor(SET,WHITE);
00710         Glogo(x+10,y-200,8,WHITE);
00711 }
00712 /*
```

DEMO.C

```
00712 */
00713 /*
00714  *ソース　スクリーン表示
00715  */
00716 source_dsp()
00717 {
00718 int     x,y,dh,dv;
00719
00720         x  = src_scrn.xmin;
00721         y  = src_scrn.ymax;
00722         dh = src_scrn.xmax-src_scrn.xmin;
00723         dv = src_scrn.ymax-src_scrn.ymin;
00724         GorgDst(&on_screen);
00725         Gflame(PULL,x-7,y+20,dh+14,dv+25);
00726         GtileFill(  x-7,y+20,dh+14,dv+25,&pat[1]);
00727         Gcolor(REP,BLACK);
00728         Gprint(x+10,y+19,"Ｓｏｕｒｃｅ　Ｓｃｒｅｅｎ",&fontHan);
00729         Gcolor(REP,WHITE);
00730         Gprint(x+11,y+18,"Ｓｏｕｒｃｅ　Ｓｃｒｅｅｎ",&fontHan);
00731         Gcolor(SET,BLUE);
00732         GBoxFill(x,y,dh-1,dv-1);
00733         Gcolor(REP,RED);
00734         Gprint(x+35,y-50,"ＮＯＷ　ＬＯＡＤＩＮＧ　！",&fontHan);
00735 }
00736 /*
```

DEMO.C

```
00736 */
00737 /*
00738  * メニュー表示
00739  */
00740 menu_dsp(m)
00741 Mouse *m;
00742 {
00743 uint    *ptr;
00744 int     i,x,y,dh,dv;
00745
00746         GorgDst(&on_screen);
00747         Gflame(OFF,10,348,270,155);
00748         Gcolor(REP,BLACK);
00749         Gprint(30,343,"Ｊｏｂ　Ｍｅｎｕ",&fontHan);
00750         for ( i=1;i<=MENU_MAX;i++)
00751         {
00752                 Gcolor(SET,BLUE);
00753                 GBoxFill(menu[i].x,menu[i].y,menu[i].dx,menu[i].dy);
00754                 Gflame(PUSH,menu[i].x,menu[i].y,menu[i].dx,menu[i].dy);
00755                 Gcolor(REP,CYAN);
00756                 Gprint(menu[i].x+3,menu[i].y-2,&menu[i].str[0],&fontHan);
00757         }
00758         i=m->event;
00759         x=menu[i].x;
00760         y=menu[i].y;
00761         dh=menu[i].dx;
00762         dv=menu[i].dy;
00763         Gflame(PULL,x,y,dh,dv);
00764         Gcolor(REP,WHITE);
00765         Gprint(x+3,y-2,&menu[i].str[0],&fontHan);
00766         Gcolor(REP,BLUE);
00767         Gprint(40,250,"ＡＧＤＣ２",&fontSlt);
00768 }
00769 /*
```

DEMO.C

```
00769 */
00770 /*
00771  *      メニューエリアのチェック
00772  */
00773 chk_menu_no(m)
00774 Mouse *m;
00775 {
00776 int     i,x,y,dh,dv;
00777         for ( i=1;i<= MENU_MAX;i++)
00778         {
00779                 if ( menu[i].x < m->x && m->x < (menu[i].x+menu[i].dx) &&
00780                      m->y > (menu[i].y-menu[i].dy) && m->y < menu[i].y )
00781                 {
00782                         if ( m->menu != 0 && m->menu != i)
00783                         {
00784                                 x  = menu[m->menu].x;
00785                                 y  = menu[m->menu].y;
00786                                 dh = menu[m->menu].dx;
00787                                 dv = menu[m->menu].dy;
00788                                 GorgDst(&on_screen);
00789                                 GtileFill(x,y,dh,dv,&pat[5]);
00790                         }
00791                         if ( m->menu != i )
00792                         {
00793                                 m->menu = i;
00794                                 x  = menu[m->menu].x;
00795                                 y  = menu[m->menu].y;
00796                                 dh = menu[m->menu].dx;
00797                                 dv = menu[m->menu].dy;
00798                                 GorgDst(&on_screen);
00799                                 GtileFill(x,y,dh,dv,&pat[5]);
00800                         }
00801                         return;
00802                 }
00803         }
00804         if ( m->menu != 0 )
00805         {
00806                 x  = menu[m->menu].x;
00807                 y  = menu[m->menu].y;
00808                 dh = menu[m->menu].dx;
00809                 dv = menu[m->menu].dy;
00810                 GorgDst(&on_screen);
00811                 GtileFill(x,y,dh,dv,&pat[5]);
00812         }
00813         m->menu = 0;
00814 }
00815
00816 /*
```

DEMO.C

```
00816 */
00817 /*
00818  * マウス状態入力
00819  */
00820 cur_input(m)
00821 Mouse   *m;
00822 {
00823 int     r;
00824
00825         r = Mstatus(m);
00826         chk_menu_no(m);
00827         if ( r == 0 )
00828         {
00829                 m->mode = NON_CLICK;
00830                 return(NON_CLICK);
00831         }
00832         if ( m->menu != 0 )
00833         {
00834                 m->mode = MENU_CLICK;
00835                 return(1);
00836         }
00837         /* source screen area */
00838         if ( src_scrn.xmin <= m->x && src_scrn.xmax >= m->x &&
00839              src_scrn.ymin <= m->y && src_scrn.ymax >= m->y )
00840         {
00841                 m->mode = SRC_CLICK;
00842                 return(1);
00843         }
00844         /* work screen area */
00845         if ( work_scrn.xmin <= m->x && work_scrn.xmax >= m->x &&
00846              work_scrn.ymin <= m->y && work_scrn.ymax >= m->y )
00847         {
00848                 m->mode = WRK_CLICK;
00849                 return(1);
00850         }
00851         return(-1);
00852 }
00853 /*
```

保守／廃止

DEMO.C

```
00853 */
00854 /*
00855  * マウスカーソル形状の設定
00856  */
00857 cur_setting()
00858 {
00859 uint    *ptr;.
00860 int     i,r;
00861         ptr = (uint *)&cur_tbl[1][0];
00862         GorgDst(&tile_org);
00863         Gpmax(1);
00864         Gcolor(SET,RED);
00865         GnonClip();
00866         GPutC(0,0,15,15);
00867         for ( i=0; i < 16 ; i++)
00868         {
00869                 r=GPgport(*ptr++);
00870                 if ( r == BREAK ) break;
00871                 if ( r == ERROR ) return(ERROR);
00872         }
00873         ptr = (uint *)&cur_tbl[0][0];
00874         GPutC(0,-16,15,15);
00875         for ( i=0; i < 16 ; i++)
00876         {
00877                 r=GPgport(*ptr++);
00878                 if ( r == BREAK ) break;
00879                 if ( r == ERROR ) return(ERROR);
00880         }
00881 }
00882 /*
```

DEMO.C

```
00882 */
00883 /*
00884  * マウス  カーソル位置の設定
00885  */
00886 static int ox,oy;
00887 cur_locate(m)
00888 Mouse   *m;
00889 {
00890 Copy    c;
00891         if ( m->x == ox && m->y == oy) return;
00892         Mlocate(m);
00893         if( ox != m->x || oy != m->y )
00894         {
00895                 GorgDst(&cur_screen);
00896                 GnonClip();
00897                 Gcolor(SET,RED);
00898                 GBoxClr(ox,oy,15,15);
00899                 GorgSrc(&tile_org);
00900                 GorgDst(&cur_screen);
00901                 c.x = m->x;
00902                 c.y = m->y;
00903                 c.xs = 0;
00904                 c.ys = 0;
00905                 c.dh = 15;
00906                 c.dv = 15;
00907                 c.mode = COPY_CC+S_M;
00908                 if ( work_scrn.xmin <= m->x && work_scrn.xmax >= m->x &&
00909                      work_scrn.ymin <= m->y && work_scrn.ymax >= m->y )
00910                 {
00911                         c.ys -= 16;
00912                 }
00913                 Gcolor(SET,RED);
00914                 GCopy(&c,0,0);
00915         }
00916         ox=m->x;
00917         oy=m->y;
00918 }
00919 /*
```

DEMO.C

```
00919 */
00920 /*
00921  * ソース・スクリーンの画像表示
00922  */
00923 source_copy()
00924 {
00925 Copy c;
00926         GorgSrc(&scaner);
00927         GorgDst(&on_screen);
00928         c.xs = 0;
00929         c.ys = SRC_YSIZE-1;
00930         c.x  = src_scrn.xmin;
00931         c.y  = src_scrn.ymax;
00932         c.dh = SRC_XSIZE-1;
00933         c.dv = SRC_YSIZE-1;
00934         c.mode = COPY_CC+M_M;
00935         Gcolor(SET,RGB);
00936         Gcopy(&c,ES2_16,ES2_16);
00937         view_area(0);
00938         view_area(1);
00939 }
00940 /*
```

DEMO.C

```
00940 */
00941 /*
00942  * ウィンドウのクリック位置チェック
00943  */
00944 chk_win_position(m)
00945 Mouse *m;
00946 {
00947 int     x,y,xs,ys,xe,ye,dh,dv,no,mx,my;
00948
00949         no = m->win;
00950         x  = m->x;
00951         y  = m->y;
00952         xs = W[no].copy.x;
00953         ys = W[no].copy.y;
00954         dh = W[no].copy.dh;
00955         dv = W[no].copy.dv;
00956         xe = xs + dh;
00957         ye = ys - dv;
00958
00959         if ( xs<=x && x<=(xs+dh/4) && (ys-dv/4)<=y && y<=ys )
00960         {
00961                 return(1);
00962         }
00963         if ((xs+dh/4)<=x && x<=(xe-dh/4) && (ys-dv/2)<=y && y<=ys )
00964         {
00965                 return(2);
00966         }
00967         if ( (xe-dh/4)<=x && x<=xe && (ys-dv/4) <= y && y <= ys )
00968         {
00969                 return(3);
00970         }
00971         if ( xs<=x && x<=(xs+dh/4) && (ys-dv/4*3) <= y && y <= (ys-dv/4))
00972         {
00973                 return(4);
00974         }
00975         if ( (xe-dh/4)<=x && x<=xe && (ye+dv/4) <= y && y <= (ys-dv/4))
00976         {
00977                 return(5);
00978         }
00979         if ( xs<=x && x<=(xs+dh/4) &&  ye <= y && y <= (ye+dv/4))
00980         {
00981                 return(6);
00982         }
00983         if ( (xs+dh/4)<=x && x<=(xe-dh/4) && ye <= y && y <= (ye+dv/2))
00984         {
00985                 return(7);
00986         }
00987         if ( (xe-dh/4)<=x && x<=xe &&  ye <= y && y <= (ye+dv/4))
00988         {
00989                 return(8);
00990         }
00991         return(-1);
00992 }
00993 /*
```

DEMO.C

```
00993 */
00994 /*
00995  * ウィンドウのクリック位置チェック
00996  */
00997
00998 chk_win_line(m)
00999 Mouse *m;
01000 {
01001 int     x,y,xs,ys,xe,ye,dh,dv,no,mx,my;
01002
01003         no = m->win;
01004         x  = m->x;
01005         y  = m->y;
01006         xs = W[no].copy.x;
01007         ys = W[no].copy.y;
01008         dh = W[no].copy.dh;
01009         dv = W[no].copy.dv;
01010         xe = xs + dh;
01011         ye = ys - dv;
01012
01013         if ( xs <= x && x <= (xs+10) && ye <= y && y <= ys )
01014         {
01015                 return(LINE_LEFT);
01016         }
01017         if ( xe-10 <= x && x <= xe && ye <= y && y <= ys )
01018         {
01019                 return(LINE_RIGHT);
01020         }
01021         if ( xs <= x && x <= xe && ys - 10 <= y && y <= ys )
01022         {
01023                 return(LINE_TOP);
01024         }
01025         if ( xs <= x && x <= xe  && ye <= y && y <= ye+10 )
01026         {
01027                 return(LINE_BUTTOM);
01028         }
01029         return(-1);
01030 }
01031 /*
```

保守／廃止

DEMO.C

```
01031 */
01032 /*
01033  * ウィンドウの表示エリアの枠
01034  */
01035 view_area(no)
01036 int     no;
01037 {
01038 static int x[2]={0,0};
01039 static int y[2]={0,0};
01040 static int dx[2]={0,0};
01041 static int dy[2]={0,0};
01042         GorgDst(&on_screen);
01043         GsetClip(&src_scrn);
01044         Gcolor(XOR,RGB);
01045         GBox(x[no],y[no],dx[no],dy[no]);
01046
01047         x[no] =  W[no].copy.xs / 8 + src_scrn.xmin;
01048         y[no] =  W[no].copy.ys / 8 + src_scrn.ymin;
01049         if ( W[no].rot == 1 || W[no].rot == 3 )
01050         {
01051                 dy[no] = W[no].copy.dh  ;
01052                 dx[no] = W[no].copy.dv  ;
01053         }
01054         else
01055         {
01056                 dx[no] = W[no].copy.dh ;
01057                 dy[no] = W[no].copy.dv ;
01058         }
01059         dx[no] = GcalES(-W[no].es,dx[no]) / 8;
01060         dy[no] = GcalES(-W[no].es,dy[no]) / 8;
01061         GsetClip(&src_scrn);
01062         Gcolor(XOR,RGB);
01063         GBox(x[no],y[no],dx[no],dy[no]);
01064         GnonClip();
01065 }
```

A.8　WIN.C

保守／廃止

WIN.C

```
00001 /*****************************************************************************
00002  *
00003  *       win.c
00004  *
00005  *       ウィンドウ制御系関数
00006  *
00007  *****************************************************************************/
00008
00009 #include <stdio.h>
00010 #include "const.h"
00011 #include "struct.h"
00012
00013 Window  *lwp,*hwp;
00014 Clip    lclip,hclip;
00015 /*
00016  * ディスプレイ表示許可
00017  */
00018 Wdisplay(D)
00019 Display *D;
00020 {
00021         GInit(D);
00022         GCls();
00023         GDispOn(0L);
00024 }
00025 /*
00026  * ディスプレイ表示禁止
00027  */
00028 Woffdisplay()
00029 {
00030         GDispOff();
00031         GdsbReg();
00032 }
00033 /*
```

1

WIN.C

```
00033 */
00034 /*
00035  * ウィンドウ定義（セットアップ）
00036  */
00037 Wsetup(w)
00038 Window  *w;
00039 {
00040 Copy c;
00041         GorgSrc(&w->src);
00042         GorgDst(&w->wrk);
00043         c.x = 0;
00044         c.y = 0;
00045         c.xs = w->copy.xs;
00046         c.ys = w->copy.ys;
00047         if ( w->rot == 1 || w->rot == 3 )
00048         {
00049                 c.dh = GcalES(-w->es,w->copy.dv);
00050                 c.dv = GcalES(-w->es,w->copy.dh);
00051         }
00052         else
00053         {
00054                 c.dh = GcalES(-w->es,w->copy.dh);
00055                 c.dv = GcalES(-w->es,w->copy.dv);
00056         }
00057         c.dh++;
00058         c.dv++;
00059         c.mode = COPY_CC+M_M;
00060         GnonClip();
00061         Gcolor(PUT,RGB);
00062         switch(w->rot)
00063         {
00064                 case ROT_0   :
00065                                 Gcopy(&c,w->es,w->es);
00066                                 break;
00067                 case ROT_90  :
00068                                 c.y=-(w->copy.dv);
00069                                 G90copy(&c,w->es,w->es);
00070                                 break;
00071                 case ROT_180 :
00072                                 c.x=w->copy.dh;
00073                                 c.y=-(w->copy.dv);
00074                                 G180copy(&c,w->es,w->es);
00075                                 break;
00076                 case ROT_270 :
00077                                 c.x=w->copy.dh;
00078                                 G270copy(&c,w->es,w->es);
00079                                 break;
00080         }
00081         if ( w->pri == HIGH_PRI )
00082         {
00083                 Whighpriority(w);
00084         }
00085         else
00086         {
00087                 Wlowpriority(w);
00088         }
00089 }
00090 /*
```

2

保守／廃止

WIN.C

```
00090 */
00091 /*
00092  * ウィンドウ・表示
00093  */
00094 Wopen(w)
00095 Window  *w;
00096 {
00097 Copy c;
00098 Clip    clp;
00099
00100         Gorgsrc(&w->wrk);
00101         Gorgdst(&w->dst);
00102         c.x  = w->copy.x;
00103         c.y  = w->copy.y;
00104         c.xs = 0;
00105         c.ys = 0;
00106         c.dh = w->copy.dh;
00107         c.dv = w->copy.dv;
00108         c.mode = COPY_CC+M_M;
00109         if ( w->pri == HIGH_PRI)
00110         {
00111                 GnonClip();
00112         }
00113         else
00114         {
00115                 GsetClip(&hclip);
00116         }
00117         Gcolor(PUT,RGB);
00118         Gcopy(&c,0,0);
00119 }
00120 /*
```

3

保守／廃止

WIN.C

```
00120 */
00121 /*
00122  * ウィンドウ・位置変更
00123  */
00124 Wmove(w,mx,my)
00125 Window  *w;
00126 int     mx;  /* x-position */
00127 int     my;  /* y-      ""  */
00128 {
00129 int     x,y,dx,dy,dh,dv,xe,ye;
00130 int     wx,wy;
00131 Copy    cp;
00132 Clip    cl;
00133
00134         x = w->copy.x;
00135         y = w->copy.y;
00136         dh = w->copy.dh;
00137         dv = w->copy.dv;
00138         wx = x+mx;
00139         wy = y+my;
00140         if ( wx     < w->dstclip.xmin )         wx = w->dstclip.xmin;
00141         if ( wx+dh  > w->dstclip.xmax )         wx = w->dstclip.xmax-dh;
00142         if ((wy-dv) < w->dstclip.ymin )         wy = w->dstclip.ymin+dv;
00143         if ( wy     > w->dstclip.ymax )         wy = w->dstclip.ymax;
00144         dx = wx-x;
00145         dy = wy-y;
00146         xe = x+dh;
00147         ye = y-dv;
00148
00149         if ( w->pri == HIGH_PRI )
00150         {
00151                 GsetClip(&lclip);
00152                 GorgDst(&w->dst);
00153                 Gcolor(SET,BLUE);
00154                 if ( dx != 0 )
00155                 {
00156                         if ( dx > 0 )   GBoxFill(x,   y,  dx,  (dv));
00157                         else            GBoxFill(xe,  y,  dx,  (dv));
00158                 }
00159                 if ( dy != 0 )
00160                 {
00161                         if ( dy > 0 )   GBoxFill(x,   ye, dh,   -dy   );
00162                         else            GBoxFill(x,   y,  dh,   -dy   );
00163                 }
00164                 w->copy.x = wx;
00165                 w->copy.y = wy;
00166                 Whighpriority(w);
00167                 Wopen(w);
00168                 Wopen(lwp);
00169         }
00170         else
00171         {
00172                 GsetClip(&hclip);
00173                 GorgDst(&w->dst);
00174                 GColor(SET,BLUE);
00175                 if ( dx != 0 )
00176                 {
00177                         if ( dx > 0 )   GBoxFill(x,   y,  dx,  (dv));
00178                         else            GBoxFill(xe,  y,  dx,  (dv));
00179                 }
00180                 if ( dy != 0 )
00181                 {
00182                         if ( dy > 0 )   GBoxFill(x,   ye, dh,   -dy   );
00183                         else            GBoxFill(x,   y,  dh,   -dy   );
00184                 }
00185                 w->copy.x = wx;
00186                 w->copy.y = wy;
00187                 Wlowpriority(w);
00188                 Wopen(w);
```

WIN.C

```
00189                   Wopen(hwp);
00190           }
00191   }
00192   /*
```

保守／廃止

WIN.C

```
00192 */
00193 /*
00194  * ウィンドウ・プライオリティ変更
00195  */
00196 Wchgpriority(w1,w2)
00197 Window  *w1,*w2;
00198 {
00199         if ( w1->pri == HIGH_PRI )
00200         {
00201                 Wlowpriority(w1);
00202                 Whighpriority(w2);
00203         }
00204         else
00205         {
00206                 Wlowpriority(w2);
00207                 Whighpriority(w1);
00208         }
00209         Wopen(w1);
00210         Wopen(w2);
00211 }
00212 /*
00213  * ウィンドウ・プライオリティ設定（ＨＩＧＨ）
00214  */
00215 WhighPriority(w)
00216 Window  *w;
00217 {
00218         w->pri = HIGH_PRI;
00219         hwp = w;
00220         hclip.clip_mode = OUTSIDE;
00221         hclip.xmin = w->copy.x;
00222         hclip.xmax = w->copy.x+w->copy.dh;
00223         hclip.ymin = w->copy.y-w->copy.dv;
00224         hclip.ymax = w->copy.y;
00225 }
00226 /*
00227  * ウィンドウ・プライオリティ設定（ＬＯＷ）
00228  */
00229 WlowPriority(w)
00230 Window  *w;
00231 {
00232         w->pri = LOW_PRI;
00233         lwp =w;
00234         lclip.clip_mode = OUTSIDE;
00235         lclip.xmin = w->copy.x;
00236         lclip.xmax = w->copy.x + w->copy.dh;
00237         lclip.ymin = w->copy.y - w->copy.dv;
00238         lclip.ymax = w->copy.y;
00239 }
00240 /*
```

保守／廃止

WIN.C

```
00240 */
00241 /*
00242  * ウィンドウ・スクロール
00243  */
00244 Wscroll(w,dx,dy)
00245 Window  *w;
00246 int     dx;  /* X dot */
00247 int     dy;  /* y dot */
00248 {
00249 int     xs,ys,dh,dv;
00250         xs = w->copy.xs;
00251         ys = w->copy.ys;
00252         if ( w->rot == ROT_90 || w->rot == ROT_270 )
00253         {
00254                 dv = GcalES(-w->es,w->copy.dh);
00255                 dh = GcalES(-w->es,w->copy.dv);
00256         }
00257         else
00258         {
00259                 dv = GcalES(-w->es,w->copy.dv);
00260                 dh = GcalES(-w->es,w->copy.dh);
00261         }
00262         switch(w->rot)
00263         {
00264                 case ROT_0   : xs += dx; ys += dy;      break;
00265                 case ROT_90  : xs += dy; ys -= dx;      break;
00266                 case ROT_180 : xs -= dx; ys -= dy;      break;
00267                 case ROT_270 : xs +=-dy; ys += dx;      break;
00268         }
00269         if ( xs    < w->srcclip.xmin ) xs = w->srcclip.xmin;
00270         if ( xs+dh > w->srcclip.xmax ) xs = w->srcclip.xmax-dh;
00271         if ( ys-dv < w->srcclip.ymin ) ys = w->srcclip.ymin+dv;
00272         if ( ys    > w->srcclip.ymax ) ys = w->srcclip.ymax;
00273         w->copy.xs = xs;
00274         w->copy.ys = ys;
00275         Wsetup(w);
00276         Wopen(w);
00277 }
00278 /*
```

WIN.C

```
00278 */
00279 /*
00280  * ウィンドウ・拡大
00281  */
00282 Wenlarge(w)
00283 Window  *w;
00284 {
00285 int     xs,ys,dh,dv;
00286         xs = w->copy.xs;
00287         ys = w->copy.ys;
00288         if ( w->es >= 15 ) return;
00289         w->es++;
00290         if ( w->rot == 1 || w->rot == 3 )
00291         {
00292                 dh = GcalES(-w->es,w->copy.dv);
00293                 dv = GcalES(-w->es,w->copy.dh);
00294         }
00295         else
00296         {
00297                 dh = GcalES(-w->es,w->copy.dh);
00298                 dv = GcalES(-w->es,w->copy.dv);
00299         }
00300         if ( xs + dh >= w->srcclip.xmax || ys-dv <= w->srcclip.ymin )
00301         {
00302                 w->es--;
00303                 return;
00304         }
00305         Wsetup(w);
00306         Wopen(w);
00307 }
00308 /*
```

8

保守／廃止

WIN.C

```
00308 */
00309 /*
00310  * ウィンドウ・縮小
00311  */
00312 Wshrink(w)
00313 Window  *w;
00314 {
00315 int     xs,ys,dh,dv;
00316         xs = w->copy.xs;
00317         ys = w->copy.ys;
00318         if ( w->es <= -15 ) return;
00319         w->es--;
00320         if ( w->rot == ROT_90 || w->rot == 270 )
00321         {
00322                 dh = GcalES(-w->es,w->copy.dv);
00323                 dv = GcalES(-w->es,w->copy.dh);
00324         }
00325         else
00326         {
00327                 dh = GcalES(-w->es,w->copy.dh);
00328                 dv = GcalES(-w->es,w->copy.dv);
00329         }
00330         if ( xs + dh >= w->srcclip.xmax || ys-dv <= w->srcclip.ymin )
00331         {
00332                 w->es++;
00333                 return;
00334         }
00335         Wsetup(w);
00336         Wopen(w);
00337 }
00338 /*
```

保守／廃止

WIN.C

```
00338 */
00339 /*
00340  * ウィンドウ・回転
00341  */
00342 Wrotation(w)
00343 Window  *w;
00344 {
00345 int     xs,ys,dh,dv,sdh,sdv;
00346         xs = w->copy.xs;
00347         ys = w->copy.ys;
00348         dh = w->copy.dh;
00349         dv = w->copy.dv;
00350
00351         if ( w->rot == ROT_90 || w->rot == ROT_270)
00352         {
00353                 sdh = GcalES(-w->es,dh);
00354                 sdv = GcalES(-w->es,dv);
00355         }
00356         else
00357         {
00358                 sdv = GcalES(-w->es,dh);
00359                 sdh = GcalES(-w->es,dv);
00360         }
00361         xs += (sdh-sdv)/2;
00362         ys += (sdh-sdv)/2;
00363
00364         if ( xs     < w->srcclip.xmin ) return(-1);
00365         if ( xs+sdh > w->srcclip.xmax ) return(-1);
00366         if ( ys     > w->srcclip.ymax ) return(-1);
00367         if ( ys-sdv < w->srcclip.ymin ) return(-1);
00368         w->rot++;
00369         if ( w->rot >= 4 ) w->rot =0;
00370         w->copy.xs = xs;
00371         w->copy.ys = ys;
00372         Wsetup(w);
00373         Wopen(w);
00374 }
00375 /*
```

保守／廃止

WIN.C

```
00375 */
00376 /*
00377  * ウィンドウ・サイズ変更
00378  */
00379 Wresize(w,mode,m)
00380 Window  *w;
00381 int     mode;
00382 int     m;
00383 {
00384 int     x,y,dh,dv,xs,ys,sm,sdh,sdv,xe,ye;
00385
00386         if ( m == 0 ) return(0);
00387         x = w->copy.x;
00388         y = w->copy.y;
00389         dh = w->copy.dh;
00390         dv = w->copy.dv;
00391         xe = x+dh;
00392         ye = y-dv;
00393
00394 /* WORK CLIP AREA CHECK */
00395         switch(mode)
00396         {
00397         case LINE_LEFT :
00398                         if (x+m < w->dstclip.xmin ) m = w->dstclip.xmin-x;
00399                         if (x+m > xe-32           ) m = xe-32-x;
00400                         break;
00401         case LINE_TOP  :
00402                         if ( y+m > w->dstclip.ymax ) m = w->dstclip.ymax-y;
00403                         if (y+m<ye+32 )              m=ye+32-y;
00404                         break;
00405         case LINE_RIGHT :
00406                         if (xe+m > w->dstclip.xmax ) m = w->dstclip.xmax-xe;
00407                         if (xe+m < x+32            ) m = x+32-xe;
00408                         break;
00409         case LINE_BUTTOM :
00410                         if (ye+m > y-32            ) m = y-32-ye;
00411                         if (ye+m < w->dstclip.ymin ) m = w->dstclip.ymin-ye;
00412                         break;
00413         }
00414 /* SOURCE CLIP AREA CHECK */
00415         switch(mode)
00416         {
00417         case LINE_LEFT :
00418                         if ( w->rot == ROT_0   ) m = Wsrcchk(w,LINE_LEFT,   m);
00419                         if ( w->rot == ROT_90  ) m =-Wsrcchk(w,LINE_TOP,    -m);
00420                         if ( w->rot == ROT_180 ) m =-Wsrcchk(w,LINE_RIGHT,  -m);
00421                         if ( w->rot == ROT_270 ) m = Wsrcchk(w,LINE_BUTTOM, m);
00422                         break;
00423         case LINE_TOP :
00424                         if ( w->rot == ROT_0   ) m = Wsrcchk(w,LINE_TOP,    m);
00425                         if ( w->rot == ROT_90  ) m = Wsrcchk(w,LINE_RIGHT,  m);
00426                         if ( w->rot == ROT_180 ) m =-Wsrcchk(w,LINE_BUTTOM,-m);
00427                         if ( w->rot == ROT_270 ) m =-Wsrcchk(w,LINE_LEFT,   -m);
00428                         break;
00429         case LINE_RIGHT :
00430                         if ( w->rot == ROT_0   ) m = Wsrcchk(w,LINE_RIGHT,  m);
00431                         if ( w->rot == ROT_90  ) m =-Wsrcchk(w,LINE_BUTTOM,-m);
00432                         if ( w->rot == ROT_180 ) m =-Wsrcchk(w,LINE_LEFT,   -m);
00433                         if ( w->rot == ROT_270 ) m = Wsrcchk(w,LINE_TOP,    m);
00434                         break;
00435         case LINE_BUTTOM :
00436                         if ( w->rot == ROT_0   ) m = Wsrcchk(w,LINE_BUTTOM, m);
00437                         if ( w->rot == ROT_90  ) m = Wsrcchk(w,LINE_LEFT,   m);
00438                         if ( w->rot == ROT_180 ) m =-Wsrcchk(w,LINE_TOP,    -m);
00439                         if ( w->rot == ROT_270 ) m =-Wsrcchk(w,LINE_RIGHT,  -m);
00440                         break;
00441         }
00442         if ( m == 0 ) return(0);
00443
```

WIN.C

```
00444         if ( w->pri == HIGH_PRI )
00445                 GsetClip(&lclip);
00446         else
00447                 GsetClip(&hclip);
00448         GorgDst(&w->dst);
00449         Gcolor(SET,BLUE);
00450
00451 /* CLEAR PROCESS */
00452         switch( mode )
00453         {
00454         case LINE_LEFT :
00455                         if ( m > 0 )    GBoxFill(x,y,m,dv);
00456                         w->copy.x  +=  m;
00457                         w->copy.dh += -m;
00458                         break;
00459         case LINE_TOP :
00460                         if ( m < 0 )    GboxFill(x,y,dh,-m);
00461                         w->copy.y  += m;
00462                         w->copy.dv += m;
00463                         break;
00464         case LINE_RIGHT:
00465                         if ( m < 0 )    GboxFill(xe,y,m,dv);
00466                         w->copy.dh += m;
00467                         break;
00468         case LINE_BUTTOM:
00469                         if ( m > 0 )    GboxFill(x,ye,dh,-m);
00470                         w->copy.dv += -m;
00471                         break;
00472         }
00473         Wsetup(w);
00474         Wopen(lwp);
00475         Wopen(hwp);
00476         return(m);
00477 }
00478 /*
```

保守／廃止

WIN.C

```
00478 */
00479 /*
00480  * ソース・エリアの範囲チェック
00481  */
00482 Wsrcchk(w,mode,m)
00483 Window  *w;
00484 int     mode;
00485 int     m;
00486 {
00487 int     xs,ys,dh,dv,xe,ye;
00488
00489         xs = w->copy.xs;
00490         ys = w->copy.ys;
00491         if ( w->rot == ROT_0 || w->rot == ROT_180 )
00492         {
00493                 dh = GcalES(-w->es,w->copy.dh);
00494                 dv = GcalES(-w->es,w->copy.dv);
00495         }
00496         else
00497         {
00498                 dh = GcalES(-w->es,w->copy.dv);
00499                 dv = GcalES(-w->es,w->copy.dh);
00500         }
00501         xe = xs+dh;
00502         ye = ys-dv;
00503         m = GcalES(-w->es,m);
00504         switch(mode )
00505         {
00506         case LINE_LEFT :
00507                         if ( xs + m < w->srcclip.xmin ) m = w->srcclip.xmin-xs;
00508                         if ( xs + m > xe )              m = xe-xs;
00509                         w->copy.xs += m;
00510                         break;
00511         case LINE_TOP  :
00512                         if ( ys + m > w->srcclip.ymax ) m = w->srcclip.ymax-ys;
00513                         if ( ys + m < ye )              m = ye-ys;
00514                         w->copy.ys += m;
00515                         break;
00516         case LINE_RIGHT :
00517                         if ( xe + m > w->srcclip.xmax ) m = w->srcclip.xmax-xe;
00518                         if ( xe + m < xs )              m = xs-xe;
00519                         break;
00520         case LINE_BUTTOM :
00521                         if ( ye + m < w->srcclip.ymin ) m = w->srcclip.ymin-ye;
00522                         if ( ye + m > ys )              m = ys-ye;
00523                         break;
00524         }
00525         m = GcalES(w->es,m);
00526         return(m);
00527 }
00528 /*
```

WIN.C

```
00528 */
00529 /*
00530  * ウィンドウ・クリック  チェック
00531  */
00532 Wclick(m)
00533 Mouse   *m;
00534 {
00535         if ( hwp->copy.x <= m->x && (hwp->copy.x+hwp->copy.dh) >= m->x &&
00536              (hwp->copy.y-hwp->copy.dv) <= m->y && hwp->copy.y >= m->y )
00537         {
00538                 m->win = hwp->no;
00539                 return(1);
00540         }
00541         if ( lwp->copy.x <= m->x && (lwp->copy.x+lwp->copy.dh) >= m->x &&
00542              (lwp->copy.y-lwp->copy.dv) <= m->y && lwp->copy.y >= m->y )
00543         {
00544                 m->win = lwp->no;
00545                 return(1);
00546         }
00547         return(0);
00548 }
00549 /*
```

WIN.C

```
00549 */
00550 /*
00551  * ウィンドウ・プライオリティ　チェック
00552  */
00553 WgetPriority(pri,cpy)
00554 int     pri;
00555 Copy    *cpy;
00556 {
00557         if ( pri == HIGH_PRI )
00558         {
00559                 cpy->mode = hwp->copy.mode;
00560                 cpy->x    = hwp->copy.x;
00561                 cpy->y    = hwp->copy.y;
00562                 cpy->xs   = hwp->copy.xs;
00563                 cpy->ys   = hwp->copy.ys;
00564                 cpy->dh   = hwp->copy.dh;
00565                 cpy->dv   = hwp->copy.dv;
00566                 return(hwp->no);
00567         }
00568         else
00569         {
00570                 cpy->mode = lwp->copy.mode;
00571                 cpy->x    = lwp->copy.x;
00572                 cpy->y    = lwp->copy.y;
00573                 cpy->xs   = lwp->copy.xs;
00574                 cpy->ys   = lwp->copy.ys;
00575                 cpy->dh   = lwp->copy.dh;
00576                 cpy->dv   = lwp->copy.dv;
00577                 return(lwp->no);
00578         }
00579 }
```

A.9　MOUSE.C

保守／廃止

MOUSE.C

```
00001 /*****************************************************************************
00002  *
00003  *      mouse.c
00004  *
00005  *              マウス制御関数
00006  *
00007  *****************************************************************************/
00008
00009 #include <stdio.h>
00010 #include <dos.h>
00011 #include <sys¥types.h>
00012 #include <sys¥stat.h>
00013 #include <io.h>
00014 #include <conio.h>
00015 #include <memory.h>
00016
00017 #include"const.h"
00018 #include"struct.h"
00019
00020 static union REGS inregs,outregs;
00021 static struct SREGS segregs;
00022
00023 static uchar    m_style[64];
00024 static int ox,oy;
00025
00026 /*
00027  *      マウスの初期化
00028  */
00029 Minit(m)
00030 Mouse   *m;
00031 {
00032         inregs.x.ax=0;
00033         int86(51,&inregs,&outregs);
00034         if (outregs.x.ax == 0)
00035         {
00036                 printf("¥nマウス・ドライバが組み込まれていません");
00037                 exit(-1);
00038         }
00039                                         /* ミッキードット比設定 */
00040         inregs.x.ax=15;
00041         inregs.x.cx=6;
00042         inregs.x.dx=6;
00043         int86(51,&inregs,&outregs);
00044
00045         memset(m_style,0x00,64);        /* マウス形状 */
00046         inregs.x.ax=9;
00047         inregs.x.bx=7;
00048         inregs.x.cx=15;
00049         segread(&segregs);
00050         segregs.es=segregs.ds;
00051         inregs.x.dx=(int)m_style;
00052         int86x(51,&inregs,&outregs,&segregs);
00053                                         /* 移動範囲指定 */
00054         inregs.x.ax=16;
00055         inregs.x.cx=0;
00056         inregs.x.dx=640-16;
00057         int86(51,&inregs,&outregs);
00058
00059         inregs.x.ax=17;
00060         inregs.x.cx=0;
00061         inregs.x.dx=399-16;
00062         int86(51,&inregs,&outregs);
00063
00064         inregs.x.ax=1;                  /* マウス・カーソル表示 */
00065         int86(51,&inregs,&outregs);
00066
00067         Mlocate(m);
00068         ox=m->x;
00069         oy=m->y;
```

1

保守／廃止

MOUSE.C

```
00070
00071           outp(0x3fdb,830 % 256);                 /* クリック音の設定 */
00072           outp(0x3fdb,830 / 256);                 /* クリック音の設定 */
00073
00074           return(0);
00075   }
00076   /*
00077    *      マウスカーソルの非表示
00078    */
00079   Mclose()
00080   {
00081           inregs.x.ax=2;
00082           int86(51,&inregs,&outregs);
00083   }
00084   /*
00085    *      マウスの状態取得
00086    */
00087   Mstatus(m)
00088   Mouse *m;
00089   {
00090           inregs.x.ax=3;
00091           int86(51,&inregs,&outregs);
00092           m->rclick = outregs.x.bx;               /* 右ボタン */
00093           m->lclick = outregs.x.ax;               /* 左ボタン */
00094           m->x      = outregs.x.cx;               /* x座標   */
00095           m->y      = outregs.x.dx;               /* y座標   */
00096           m->y      = (400-m->y);
00097
00098           if ( m->x > 640-16 )    m->x = 640-16;
00099           if ( m->y < 16 )        m->y = 16;
00100
00101           m->dx     = m->x-ox;
00102           m->dy     = m->y-oy;
00103           ox = m->x;
00104           oy = m->y;
00105           if ( m->rclick == 0 )
00106                   return(0);
00107           else
00108                   return(1);
00109   }
00110   /*
00111    *      マウス位置の設定
00112    */
00113   Mlocate(m)
00114   Mouse   *m;
00115   {
00116           inregs.x.ax=4;
00117           inregs.x.cx=m->x;               /* x座標   */
00118           inregs.x.dx=(400-m->y);         /* y座標   */
00119           int86(51,&inregs,&outregs);
00120           ox = m->x;
00121           oy = m->y;
00122   }
00123   /*
00124    *      クリック・ボタンのリリース  チェック
00125    */
00126   MclickOff(m)
00127   Mouse   *m;
00128   {
00129           Mlocate(m);
00130           while(1)
00131           {
00132                   inregs.x.ax=3;
00133                   int86(51,&inregs,&outregs);
00134                   if ( outregs.x.bx == 0 )
00135                   {
00136                           break;
00137                   }
00138           }
```

2

MOUSE.C

```
00139           Mlocate(m);
00140   }
00141   /*
00142    * マウス・クリック音
00143    */
00144   Mbell()
00145   {
00146   int     i;
00147           inregs.h.ah=0x17;
00148           int86(0x18,&inregs,&outregs);
00149           for(i=0;i<300;i++)
00150           {
00151                   i++;
00152                   i--;
00153           }
00154           inregs.h.ah=0x18;
00155           int86(0x18,&inregs,&outregs);
00156   }
00157
```

保守／廃止

A.10　GIO.C

保守／廃止

GIO.C

```
00001 /*****************************************************************************
00002  *
00003  *      gio.c
00004  *
00005  *              ＡＧＤＣ２制御系関数
00006  *
00007  *****************************************************************************/
00008
00009 #include <stdio.h>
00010 #include <dos.h>
00011 #include <memory.h>
00012 #include <io.h>
00013 #include <conio.h>
00014 #include <ctype.h>
00015
00016 #define VSEG 8
00017 #define RSEG 8
00018
00019 #include"const.h"
00020 #include"agdcmap.h"
00021 #include"struct.h"
00022
00023 static union REGS inregs,outregs;
00024 static struct SREGS segregs;
00025
00026 Palette pal_data =
00027 {
00028 /*0--1--2--3--4--5--6--7--8--9-10-11-12-13-14-15-----------*/
00029 {00,15,00,15,00,15,00,15,15,15,15,15,15,15,15,15}, /* RED   */
00030 {00,00,15,15,00,00,12,15,15,15,15,15,15,15,15,15}, /* GREEN */
00031 {00,00,00,00,13,15,15,15,15,15,15,15,15,15,15,15}  /* BLUE  */
00032 };
00033
00034 static uint     dctrl;
00035 static uchar    wcl;
00036 static int      rndseed = 0;
00037 /*
```

保守／廃止

GIO.C

```
00037 */
00038 /*************************************************************************
00039  *       ハード・ウエアの制御
00040  *************************************************************************/
00041 /*
00042  * ＡＧＤＣ２アクセス許可
00043  */
00044 GenbReg()
00045 {
00046         outp(0xd0,0x11);
00047 }
00048 /*
00049  * ＡＧＤＣ２アクセス禁止
00050  */
00051 GdsbReg()
00052 {
00053         outp(0xd0,0x12);
00054 }
00055 /*
00056  * ＡＧＤＣ２マッピングレジスタ設定
00057  */
00058 GmapReg()
00059 {
00060         outp(0xd2,0x88);
00061 }
00062 /*
00063  * カラーパレット設定
00064  */
00065
00066 GPalette(buf)
00067 Palette *buf;
00068 {
00069 int     i;
00070 uchar   far *ptr  = (uchar far *)BASE_PAL;
00071
00072         GenbReg();
00073         ( ulong )ptr = ( ulong )ptr | (( ulong )RSEG << 27 );
00074         *ptr = 0x00;
00075         ptr  = ptr + 2;
00076         for ( i=0;i<16;i++)
00077         {
00078                 *ptr = buf->red[i];
00079                 *ptr = buf->green[i];
00080                 *ptr = buf->blue[i];
00081         }
00082 }
00083 /*
```

2

保守／廃止

GIO.C

```
00083 */
00084 /*************************************************************************
00085  *      ＡＧＤＣレジスタの設定，コマンド発行関連
00086  *************************************************************************/
00087 /*
00088  * ＡＧＤＣ２の初期化
00089  */
00090 void GInit(dsp)
00091 Display *dsp;
00092 {
00093 uint    master = 0x0010;
00094         GmapReg();
00095         GenbReg();
00096         *CTRL      = 0x03;
00097         *CTRL2     = 0x2e;
00098         *DISP_CTRL = 0xba02 | master ;
00099         *HS        = 0x0007;
00100         *HBP       = 0x0009;
00101         *HH        = 0x0023;
00102         *HD        = 0x004f;
00103         *HFP       = 0x0007;
00104         *VS        = 0x0008;
00105         *VBP       = 0x0019;
00106         *LF        = 0x0190;
00107         *VFP       = 0x0007;
00108         *DISP_CTRL = 0xba00 | master ;
00109         dctrl      =  0xba00 | master ;
00110
00111         *DP_PITCH  = dsp->dp_pitch;
00112         *DAD       = dsp->dad;
00113         wcl        = (uchar)(dsp->wc & 0xff);
00114         *WC        = wcl;
00115         *GCSRX     = 0x2000;
00116         *GCSRYS    = 0x0000;
00117         *GCSRYE    = 0x0000;
00118
00119         *BANK   = 00;
00120         *EADORG = 0x3E58;
00121         *EADORGS= 0x7CD8;
00122         *EAD1   = 000000;
00123         *EAD2   = 000000;
00124         *EAD3   = 000000;
00125         *PITCHS = 0x028;
00126         *PITCHD = 0x028;
00127         *PDISPS = 0x040000;
00128         *PDISPD = 0x040000;
00129         *PMAX   = 4;
00130         *PLANES = 0000;
00131         *MOD    = 0;
00132         *PTNP   = 000000;
00133         *PTNCNT = 0xffff;
00134         *PTNH   = 0x0000;
00135         *STACK  = 0x0ffffff1;
00136         *STMAX  = 0x07FF;
00137         *XCLMIN = 0;
00138         *XCLMAX = 0;
00139         *YCLMIN = 0;
00140         *YCLMAX = 0;
00141         *MAG    = 0x0ff;
00142         *CLIP   = NON;
00143         *X      = 0000;
00144         *Y      = 0000;
00145         *DX     = 0000;
00146         *DY     = 0000;
00147         *XS     = 0000;
00148         *YS     = 0000;
00149         *XE     = 0000;
00150         *YE     = 0000;
00151         *XC     = 0000;
```

保守／廃止

GIO.C

```
00152         *YC     = 0000;
00153         *DH     = 0000;
00154         *DV     = 0000;
00155         if ((inp(0xd0) & 0x01) != 0 )
00156                 master = 0x10;
00157         else
00158                 master = 0x41;
00159         dctrl      =  0xba00 | master ;
00160         *DISP_CTRL = 0xba00 | master ;
00161
00162         Gpalette(&pal_data);
00163 }
00164 /*
00165  * 表示許可
00166  */
00167 void GDispOn(dad)
00168 ulong   dad;
00169 {
00170         dad        &= 0x0ffffff1;
00171         dctrl      &= 0xfff7;
00172         while(1)
00173         {
00174                 if ( (*STATUS & 0x0010) != 0) break;
00175         }
00176         *DAD        = dad;
00177         *WC         = wcl;
00178         *DISP_CTRL  = dctrl;
00179
00180 }
00181 /*
00182  * 表示禁止
00183  */
00184 void GDispOff()
00185 {
00186         dctrl      |= 0x0008;
00187         *DISP_CTRL  = dctrl;
00188 }
00189 /*
00190  * 論理演算設定
00191  */
00192 GColor(mod,color)
00193 uint    mod;
00194 uint    color;
00195 {
00196         *MOD = mod;
00197         *PLANES = color;
00198 }
00199 /*
00200  *  ＰＭＡＸ設定
00201  */
00202 GPmax(max)
00203 int     max;
00204 {
00205         *PMAX    = max;
00206 }
00207 /*
00208  * 拡大／縮小率設定
00209  */
00210 GsetMag(magh,magv)
00211 int     magh;
00212 int     magv;
00213 {
00214         if ( magh < 0 ) magh = -magh;
00215         if ( magv < 0 ) magv = -magv;
00216
00217         *MAG = (uchar)((15-magh)*16 + (15-magv));
00218 }
00219 /*
00220  * クリッピング・エリア設定
```

4

保守／廃止

GIO.C

```
00221  */
00222  GsetClip(tbl)
00223  Clip    *tbl;
00224  {
00225          *CLIP   = tbl->clip_mode;
00226          *XCLMIN = tbl->xmin;
00227          *XCLMAX = tbl->xmax;
00228          *YCLMIN = tbl->ymin;
00229          *YCLMAX = tbl->ymax;
00230  }
00231  /*
00232   * クリッピング解除
00233   */
00234  GnonClip()
00235  {
00236          *CLIP   = NON;
00237  }
00238
00239  /*
00240   * 転送元座標系設定
00241   */
00242  GorgSrc( buf )
00243  Coodinate       *buf;
00244  {
00245          *EADORGS = buf->eadorg;
00246          *DADORGS = buf->dadorg | 0x10;
00247          *PITCHS  = buf->pitch;
00248          *PDISPS  = buf->pdisp;
00249  }
00250  /*
00251   * 転送先（描画）座標系設定
00252   */
00253  GorgDst( buf )
00254  Coodinate       *buf;
00255  {
00256          *EADORG = buf->eadorg;
00257          *DADORG = buf->dadorg;
00258          *PITCHD = buf->pitch;
00259          *PDISPD = buf->pdisp;
00260          *PMAX   = buf->pmax;
00261  }
00262  /*
00263   * ＡＧＤＣ２コマンド実行
00264   */
00265  GsetCmd(code)
00266  uint    code;
00267  {
00268  uint    str;
00269          str = *STATUS;
00270          if ((str & 0x0c)!= 0 )
00271          {
00272                  return(-1);
00273          }
00274          while(1)
00275          {
00276                  if ( (str & 0x0003 ) == 0 )
00277                  {
00278                          break;
00279                  }
00280                  str = *STATUS;
00281          }
00282          *COMMAND = code;
00283          while(1)
00284          {
00285                  str = *STATUS;
00286                  if ( (str & 0x0001 ) == 0 )
00287                  {
00288                          break;
00289                  }
```

保守／廃止

GIO.C

```
00290           }
00291           return(OK);
00292   }
00293   /*
00294    * ＤＰＢＳＹチェック
00295    */
00296   GDPbsy()
00297   {
00298   uint    str;
00299           while(1)
00300           {
00301                   str = *STATUS;
00302                   if ( (str & 0x0003 ) == 0 )
00303                   {
00304                           break;
00305                   }
00306           }
00307   }
00308   /*
00309    *  ＰＧＰＯＲＴ書き込み
00310    */
00311   GPgport(data)
00312   uint    data;
00313   {
00314   uint    str;
00315
00316           while(1)
00317           {
00318                   str = *STATUS;
00319                   if ( ( str & 0x0080 ) != 0 )
00320                   {
00321                           break;
00322                   }
00323                   if ( ( str & 0x000c ) != 0 )
00324                   {
00325                           return(ERROR);
00326                   }
00327                   if ( ( str & 0x0003 ) == 0 )
00328                   {
00329                           return(BREAK);
00330                   }
00331           }
00332           *PGPORT = data;
00333           return(OK);
00334   }
00335   /*
```

保守／廃止

GIO.C

```
00335 */
00336 /*************************************************************************
00337  *       線図形描画コマンド
00338  *************************************************************************/
00339 /*
00340  * 色情報読みだし
00341  */
00342 uint GreadCol(x,y)
00343         int     x;
00344         int     y;
00345 {
00346         *X=x;
00347         *Y=y;
00348         GsetCmd(0x9c00);
00349         return(*DX);
00350 }
00351 /*
00352  * 点描画
00353  */
00354 void GPset(x,y)
00355         int     x;
00356         int     y;
00357 {
00358         *X=x;
00359         *Y=y;
00360         *PTNCNT = 0xffff;
00361         GsetCmd(0x0c40);
00362 }
00363 /*
00364  * 直線描画
00365  */
00366 void GLine(x,y,xe,ye)
00367         int     x;
00368         int     y;
00369         int     xe;
00370         int     ye;
00371 {
00372         *X=x;
00373         *Y=y;
00374         *XE=xe;
00375         *YE=ye;
00376         *PTNCNT = 0xffff;
00377         GsetCmd(0x1441);
00378 }
00379 /*
00380  * 直線描画（連続）
00381  */
00382 void GLineD2(xe,ye)
00383         int     xe;
00384         int     ye;
00385 {
00386         *XE=xe;
00387         *YE=ye;
00388         *PTNCNT = 0xffff;
00389         GsetCmd(0x2841);
00390 }
00391 /*
00392  * ＢＯＸ描画
00393  */
00394 void GBox( x, y , xe,ye)
00395         int     x;
00396         int     y;
00397         int     xe;
00398         int     ye;
00399 {
00400         *X      = x;
00401         *Y      = y;
00402         *DX     = xe;
00403         *DY     = -ye;
```

保守／廃止

GIO.C

```
00404           *MAG    = 0xff;
00405           *PTNCNT = 0xffff;
00406           GsetCmd(0x4c40);
00407   }
00408   /*
00409    * 円描画
00410    */
00411   void GCircle(x,y,r)
00412           int     x;
00413           int     y;
00414           int     r;
00415   {
00416           *XC=x;
00417           *YC=y;
00418           *DX=r;
00419           *PTNCNT = 0xffff;
00420           GsetCmd(0x5040);
00421   }
00422
00423   /*
00424    * 楕円描画
00425    */
00426   void GElps(xc,yc,dy,dh,dv)
00427           int     xc;
00428           int     yc;
00429           int     dy;
00430           int     dh;
00431           int     dv;
00432   {
00433           *XC=xc;
00434           *YC=yc;
00435           *DY=dy;
00436           *DH=dh;
00437           *DV=dv;
00438           *PTNCNT = 0xffff;
00439           GsetCmd(0x5c40);
00440   }
00441   /*
00442    * 扇型描画
00443    */
00444   void GCsec(xc,yc,xs,ys,xe,ye,dx)
00445   int     xc,yc,xs,ys,xe,ye,dx;
00446   {
00447           *XC       = xc;
00448           *YC       = yc;
00449           *XS       = xs;
00450           *YS       = ys;
00451           *XE       = xe;
00452           *YE       = ye;
00453           *DX       = dx;
00454           *PTNCNT   = 0xffff;
00455           GsetCmd(0x58c0);
00456   }
00457   /*
00458    * 円弧描画
00459    */
00460   void GCarc(xc,yc,dx,xs,ys,xe,ye)
00461           int     xc,yc,dx,xs,ys,xe,ye;
00462   {
00463           *XC=xc;
00464           *YC=yc;
00465           *DX=dx;
00466           *XS=xs;
00467           *YS=ys;
00468           *XE=xe;
00469           *YE=ye;
00470           *PTNCNT = 0xffff;
00471           GsetCmd(0x54c1);
00472   }
```

保守／廃止

GIO.C

```
00473 /*
00474  * 楕円弧描画
00475  */
00476 void GEarc(xc,yc,dx,dy,dh,dv,xs,ys,xe,ye)
00477         int     xc,yc,dx,dy,dh,dv,xs,ys,xe,ye;
00478 {
00479         *XC=xc;
00480         *YC=yc;
00481         *DX=dx;
00482         *DY=dy;
00483         *DH=dh;
00484         *DV=dv;
00485         *XS=xs;
00486         *YS=ys;
00487         *XE=xe;
00488         *YE=ye;
00489         *PTNCNT = 0xffff;
00490         GsetCmd(0x60c1);
00491 }
00492 /*
00493  * 楕扇型描画
00494  */
00495 void GEsec(xc,yc,dx,dy,dh,dv,xs,ys,xe,ye)
00496         int     xc,yc,dx,dy,dh,dv,xs,ys,xe,ye;
00497 {
00498         *XC=xc;
00499         *YC=yc;
00500         *DX=dx;
00501         *DY=dy;
00502         *DH=dh;
00503         *DV=dv;
00504         *XS=xs;
00505         *YS=ys;
00506         *XE=xe;
00507         *YE=ye;
00508         *PTNCNT = 0xffff;
00509         GsetCmd(0x64c1);
00510 }
00511 /*
00512  * 弦型描画
00513  */
00514 void GCseg(xc,yc,dx,xs,ys,xe,ye)
00515         int     xc,yc,dx,xs,ys,xe,ye;
00516 {
00517         *XC=xc;
00518         *YC=yc;
00519         *DX=dx;
00520         *XS=xs;
00521         *YS=ys;
00522         *XE=xe;
00523         *YE=ye;
00524         *PTNCNT = 0xffff;
00525         GsetCmd(0x5ac1);
00526 }
00527 /*
00528  * 楕弦型描画
00529  */
00530 void GEseg(xc,yc,dx,dy,dh,dv,xs,ys,xe,ye)
00531         int     xc,yc,dx,dy,dh,dv,xs,ys,xe,ye;
00532 {
00533         *XC=xc;
00534         *YC=yc;
00535         *DX=dx;
00536         *DY=dy;
00537         *DH=dh;
00538         *DV=dv;
00539         *XS=xs;
00540         *YS=ys;
00541         *XE=xe;
```

保守／廃止

GIO.C

```
00542           *YE=ye;
00543           *PTNCNT = 0xffff;
00544           GsetCmd(0x65c1);
00545   }
00546
00547   /*
```

保守／廃止

GIO.C

```
00547 */
00548 /*************************************************************************
00549  *       グラフィックス・ペン図形描画コマンド
00550  *************************************************************************/
00551
00552 /*
00553  * 円描画（グラフィックス・ペン）
00554  */
00555 void GgCircle(x,y,r,ead3,ptnp)
00556         int     x;
00557         int     y;
00558         int     r;
00559         ulong   ead3;
00560         ulong   ptnp;
00561 {
00562         *XC     = x;
00563         *YC     = y;
00564         *DX     = r;
00565         *EAD3   = ead3;
00566         *PTNP   = ptnp;
00567         GsetCmd(0xc866);
00568 }
00569 /*
00570  * 楕円描画（グラフィックス・ペン）
00571  */
00572 void GgElps(xc,yc,dy,dh,dv,ead3,ptnp)
00573         int     xc;
00574         int     yc;
00575         int     dy;
00576         int     dh;
00577         int     dv;
00578         ulong   ead3;
00579         ulong   ptnp;
00580 {
00581         *XC     = xc;
00582         *YC     = yc;
00583         *DY     = dy;
00584         *DH     = dh;
00585         *DV     = dv;
00586         *EAD3   = ead3;
00587         *PTNP   = ptnp;
00588         GsetCmd(0xcc66);
00589 }
00590 /*
00591  * 直線描画（グラフィックス・ペン）
00592  */
00593 void GgLine(x,y,xe,ye,ead3,ptnp)
00594         int     x;
00595         int     y;
00596         int     xe;
00597         int     ye;
00598         ulong   ead3;
00599         ulong   ptnp;
00600 {
00601         *X=x;
00602         *Y=y;
00603         *XE=xe;
00604         *YE=ye;
00605         *EAD3   = ead3;
00606         *PTNP   = ptnp;
00607         GsetCmd(0xc067);
00608 }
00609 /*
00610  * 円弧描画（グラフィックス・ペン）
00611  */
00612 void GgCarc(xc,yc,dx,xs,ys,xe,ye,ead3,ptnp)
00613         int     xc,yc,dx,xs,ys,xe,ye;
00614         ulong   ead3;
00615         ulong   ptnp;
```

保守／廃止

GIO.C

```
00616 {
00617         *XC=xc;
00618         *YC=yc;
00619         *DX=dx;
00620         *XS=xs;
00621         *YS=ys;
00622         *XE=xe;
00623         *YE=ye;
00624         *EAD3   = ead3;
00625         *PTNP   = ptnp;
00626         GsetCmd(0xd067);
00627 }
00628 /*
00629  * 楕円弧描画（グラフィックス・ペン）
00630  */
00631 void GgEarc(xc,yc,dx,dy,dh,dv,xs,ys,xe,ye,ead3,ptnp)
00632         int     xc,yc,dx,dy,dh,dv,xs,ys,xe,ye;
00633         ulong   ead3;
00634         ulong   ptnp;
00635 {
00636         *XC=xc;
00637         *YC=yc;
00638         *DX=dx;
00639         *DY=dy;
00640         *DH=dh;
00641         *DV=dv;
00642         *XS=xs;
00643         *YS=ys;
00644         *XE=xe;
00645         *YE=ye;
00646         *EAD3   = ead3;
00647         *PTNP   = ptnp;
00648         GsetCmd(0xd467);
00649 }
00650 /*
```

保守／廃止

GIO.C

```
00650 */
00651 /*************************************************************************
00652  *      塗りつぶしコマンド
00653  *************************************************************************/
00654 /*
00655  * ＢＯＸ塗りつぶし
00656  */
00657 void GBoxFill( x,y,dx,dy)
00658         int     x;
00659         int     y;
00660         int     dx;
00661         int     dy;
00662 {
00663         *X      = x;
00664         *Y      = y;
00665         *DX     = dx;
00666         *DY     = -dy;
00667         *PTNCNT = 0xffff;
00668         GsetCmd(0x903c);
00669 }
00670 /*
00671  * ＢＯＸクリア
00672  */
00673 void GBoxClr(x,y,dx,dy)
00674 int     x;
00675 int     y;
00676 int     dx;
00677 int     dy;
00678 {
00679         *X      = x;
00680         *Y      = y;
00681         *DX     = dx;
00682         *DY    = -dy;
00683         *PTNCNT = 0;
00684         GsetCmd(0x903e);
00685 }
00686 /*
00687  * クリアスクリーン
00688  */
00689 void GCls()
00690 {
00691         *PMAX   = 0x8;
00692         *PLANES = 0x0000;
00693         *MOD    = 0x02;
00694         *X      = 0;
00695         *Y      = 0;
00696         *DX     = 639;
00697         *DY     = 399;
00698         GsetCmd(0x903e);
00699 }
00700 /*
00701  * ＢＯＸタイリング・フィル
00702  */
00703 void GtileFill(x,y,dx,dy,t)
00704 int     x;
00705 int     y;
00706 int     dx;
00707 int     dy;
00708 Tilling *t;
00709 {
00710         *X=x;
00711         *Y=y;
00712         *DX=dx;
00713         *DY=-dy;
00714         *MOD = t->mod;
00715         *PLANES = t->color;
00716         *PTNCNT = (t->dh+1)/16;
00717         *PTNP = t->org.eadorg + t->off_adr;
00718         if ( t->plane == 1 )
```

保守／廃止

GIO.C

```
00719           {
00720                   GsetCmd(0x90bc);
00721           }
00722           else
00723           {
00724                   *PDISPS = t->org.pdisp;
00725                   GsetCmd(0x90ac);
00726
00727           }
00728   }
00729   /*
00730    *  ペイント
00731    */
00732   void GPaint(x,y,col)
00733   int     x,y;
00734   uint    col;
00735   {
00736           *X      = x;
00737           *Y      = y;
00738           *DX     = col;
00739           *PTNCNT  = 0xffff;
00740           GsetCmd(0x6834);
00741           GDPbsy();
00742   }
00743   /*
00744    *  ペイント（タイリング）
00745    */
00746   void GtilePaint(x,y,col,t)
00747   int     x,y;
00748   uint    col;
00749   Tilling *t;
00750   {
00751           *X      = x;
00752           *Y      = y;
00753           *DX     = col;
00754           *MOD = t->mod;
00755           *PLANES = t->color;
00756           *PTNCNT = (t->dh+1)/16;
00757           *PTNP = t->org.eadorg + t->off_adr;
00758           if ( t->plane == 1 )
00759           {
00760                   GsetCmd(0x68b0);
00761           }
00762           else
00763           {
00764                   *PDISPS = t->org.pdisp;
00765                   GsetCmd(0x68a0);
00766           }
00767           GDPbsy();
00768   }
00769   /*
00770    * 円塗りつぶし
00771    */
00772   void GCrlFill(xc,yc,dx)
00773           int     xc;
00774           int     yc;
00775           int     dx;
00776   {
00777           *XC     = xc;
00778           *YC     = yc;
00779           *DX     = dx;
00780           *PTNCNT = 0xffff;
00781           GsetCmd(0x503c);
00782   }
00783   /*
00784    * 三角形塗りつぶし
00785    */
00786   void GtriFill(x,y,xs,ys,xc,yc)
00787           int     x,y,xs,ys,xc,yc;
```

保守／廃止

GIO.C

```
00788 {
00789         *X=x;
00790         *Y=y;.
00791         *XS=xs;
00792         *YS=ys;
00793         *XC=xc;
00794         *YC=yc;
00795         *PTNCNT = 0xffff;
00796         GsetCmd(0x6c3c);
00797 }
00798 /*
00799  * 台形塗りつぶし
00800  */
00801 void GtraFill(x,y,xs,ys,xe,ye)
00802 int     x,y,xs,ys,xe,ye;
00803 {
00804         *X=x;
00805         *Y=y;
00806         *XS=xs;
00807         *YS=ys;
00808         *XE=xe;
00809         *YE=ye;
00810         *PTNCNT = 0xffff;
00811         GsetCmd(0x703c);
00812 }
00813 /*
00814  * 楕円塗りつぶし
00815  */
00816 void GelpsFill(xc,yc,dy,dh,dv)
00817 int     xc,yc,dy,dh,dv;
00818 {
00819         *XC=xc;
00820         *YC=yc;
00821         *DY=dy;
00822         *DH=dh;
00823         *DV=dv;
00824         *PTNCNT = 0xffff;
00825         GsetCmd(0x5c3c);
00826 }
00827 /*
```

保守／廃止

GIO.C

```
00827 */
00828 /*************************************************************************
00829  *      コピー コマンド
00830  *************************************************************************/
00831 /*
00832  * ＣＯＰＹ処理（高速）
00833  */
00834 void GFCopy(c)
00835 Copy *c;
00836 {
00837         *X      = c->x;
00838         *Y      = c->y;
00839         *XS     = c->xs;
00840         *YS     = c->ys;
00841         *DH     = c->dh;
00842         *DV     = c->dv;
00843         *EAD1   = c->ead1;
00844         *DAD1   = c->dad1;
00845         *EAD2   = c->ead2;
00846         *DAD2   = c->dad2;
00847         GsetCmd(c->mode & 0xffec | 0x0002);
00848 }
00849 /*
00850  * ＣＯＰＹ処理（傾斜）
00851  */
00852 void GSLCopy(c,sl)
00853 Copy *c;
00854 int     sl;
00855 {
00856         *X      = c->x;
00857         *Y      = c->y;
00858         *XS     = c->xs;
00859         *YS     = c->ys;
00860         *DH     = c->dh;
00861         *DV     = c->dv;
00862         *EAD1   = c->ead1;
00863         *DAD1   = c->dad1;
00864         *EAD2   = c->ead2;
00865         *DAD2   = c->dad2;
00866         *DX     = sl;
00867         GsetCmd(c->mode & 0xffec | 0x0001);
00868 }
00869 /*
00870  * ＣＯＰＹ処理
00871  */
00872 void GCopy(c,magh,magv)
00873 Copy *c;
00874 int     magh;
00875 int     magv;
00876 {
00877 uint    code;
00878         *X      = c->x;
00879         *Y      = c->y;
00880         *XS     = c->xs;
00881         *YS     = c->ys;
00882         *DH     = c->dh;
00883         *DV     = c->dv;
00884         *EAD1   = c->ead1;
00885         *DAD1   = c->dad1;
00886         *EAD2   = c->ead2;
00887         *DAD2   = c->dad2;
00888         code = c->mode;
00889         if ( magh != 0 || magv != 0 )
00890         {
00891                 code = code | 0x03;
00892                 if ( magh > 0 )
00893                         code = code | 0x80;
00894                 if ( magv > 0 )
00895                         code = code | 0x10;
```

保守／廃止

GIO.C

```
00896                   GsetMag(magh,magv);
00897           }
00898           GsetCmd(code);
00899   }
00900   /*
00901    * ９０° ＣＯＰＹ処理
00902    */
00903   void G90Copy(c,magh,magv)
00904           Copy *c;
00905           int     magh;
00906           int     magv;
00907   {
00908   uint    code;
00909           *X  = c->x;
00910           *Y  = c->y;
00911           *XS = c->xs;
00912           *YS = c->ys;
00913           *DV = c->dv;
00914           *DH = c->dh;
00915           *EAD1    = c->ead1;
00916           *DAD1    = c->dad1;
00917           *EAD2    = c->ead2;
00918           *DAD2    = c->dad2;
00919           if ( magh == 0 && magv== 0 )
00920           {
00921                   code = c->mode + 0x10;
00922           }
00923           else
00924           {
00925                   code = c->mode + 0x32;
00926
00927                   *DX  = 0;
00928                   *DY  = 1;
00929                   *XE  = 1;
00930                   *YE  = 0;
00931                   if ( magh > 0 )
00932                           code = code | 0x80;
00933                   if ( magv > 0 )
00934                           code = code | 0x40;
00935                   GsetMag(magh,magv);
00936           }
00937           GsetCmd(code);
00938   }
00939   /*
00940    * ＣＯＰＹ処理
00941    */
00942   void G180Copy(c,magh,magv)
00943           Copy *c;
00944           int     magh;
00945           int     magv;
00946   {
00947   uint    code;
00948           *X       = c->x;
00949           *Y       = c->y;
00950           *XS      = c->xs;
00951           *YS      = c->ys;
00952           *DH      = c->dh;
00953           *DV      = c->dv;
00954           *EAD1    = c->ead1;
00955           *DAD1    = c->dad1;
00956           *EAD2    = c->ead2;
00957           *DAD2    = c->dad2;
00958           if ( magh == 0 && magv== 0 )
00959           {
00960                   code = c->mode | 0x20;
00961           }
00962           else
00963           {
00964                   code = c->mode | 0x32;
```

保守／廃止

GIO.C

```
00965                   *DX  = -1;
00966                   *DY  =  0;
00967                   *XE  =  0;
00968                   *YE  =  1;
00969                   if ( magh > 0 )
00970                           code = code | 0x80;
00971                   if ( magv > 0 )
00972                           code = code | 0x40;
00973                   GsetMag(magh,magv);
00974           }
00975           GsetCmd(code);
00976
00977   }
00978   /*
00979    * ＣＯＰＹ処理
00980    */
00981   void G270Copy(c,magh,magv)
00982   Copy *c;
00983   int     magh;
00984   int     magv;
00985   {
00986   uint    code;
00987           *X      = c->x;
00988           *Y      = c->y;
00989           *XS     = c->xs;
00990           *YS     = c->ys;
00991           *DH     = c->dh;
00992           *DV     = c->dv;
00993           *EAD1   = c->ead1;
00994           *DAD1   = c->dad1;
00995           *EAD2   = c->ead2;
00996           *DAD2   = c->dad2;
00997           if ( magh == 0 && magv== 0 )
00998           {
00999                   code = c->mode | 0x30;
01000           }
01001           else
01002           {
01003                   code = c->mode | 0x32;
01004                   *DX  =  0;
01005                   *DY  = -1;
01006                   *XE  = -1;
01007                   *YE  =  0;
01008                   if ( magh > 0 )
01009                           code = code | 0x80;
01010                   if ( magv > 0 )
01011                           code = code | 0x40;
01012                   GsetMag(magh,magv);
01013           }
01014           GsetCmd(code);
01015   }
01016   /*
01017    * 任意角回転ＣＯＰＹ処理
01018    */
01019   void GFRCopy(c,dx,dy,xe,ye,magh,magv)
01020   Copy *c;
01021   int     dx;
01022   int     dy;
01023   int     xe;
01024   int     ye;
01025   int     magh;
01026   int     magv;
01027   {
01028   uint    code;
01029           *X      = c->x;
01030           *Y      = c->y;
01031           *XS     = c->xs;
01032           *YS     = c->ys;
01033           *DH     = c->dh;
```

GIO.C

```
01034           *DV       = c->dv;
01035           *DX       = dx;
01036           *DY       = dy;
01037           *XE       = xe;
01038           *YE       = ye;
01039           *EAD1     = c->ead1;
01040           *DAD1     = c->dad1;
01041           *EAD2     = c->ead2;
01042           *DAD2     = c->dad2;
01043           code = c->mode | 0x12;
01044           if ( magh != 0)
01045           {
01046                   if ( magh > 0 )
01047                   code = code | 0x80;
01048           }
01049           if ( magv != 0)
01050           {
01051                   if ( magv > 0 )
01052                   code = code | 0x40;
01053           }
01054           GsetMag(magh,magv);
01055           GsetCmd(code);
01056   }
01057   /*
01058    * ＣＯＰＹ処理( ３オペランド・コピー )
01059    */
01060   void G3opCopy(c)
01061   Copy *c;
01062   {
01063   uint      code;
01064           *X        = c->x;
01065           *Y        = c->y;
01066           *XS       = c->xs;
01067           *YS       = c->ys;
01068           *DH       = c->dh;
01069           *DV       = c->dv;
01070           *EAD1     = c->ead1;
01071           *DAD1     = c->dad1;
01072           *EAD2     = c->ead2;
01073           *DAD2     = c->dad2;
01074           *EAD3     = c->ead3;
01075           code      = c->mode;
01076           code      = code | 0x0011;
01077           GsetCmd(code);
01078
01079   }
01080   /*
01081    *  ＰＵＴ＿Ａコマンド
01082    */
01083   void GPutA(adr,dh,dv)
01084   ulong     adr;
01085   int       dh;
01086   int       dv;
01087   {
01088           *EAD1     = adr;
01089           *DAD1     = 0;
01090           *DH       = dh;
01091           *DV       = dv;
01092           GsetCmd(0x940b);
01093   }
01094   /*
01095    *  ＰＵＴ＿Ｃコマンド
01096    */
01097   void GPutC(x,y,dh,dv)
01098   int       x;
01099   int       y;
01100   int       dh;
01101   int       dv;
01102   {
```

保守／廃止

GIO.C

```
01103           *X          = x;
01104           *Y          = y;
01105           *DH         = dh;
01106           *DV         = dv;
01107           GsetCmd(0x980b);
01108   }
01109   /*
```

保守／廃止

GIO.C

```
01109 */
01110 /*************************************************************************
01111  *      文字表示
01112  *************************************************************************/
01113 /*
01114  * 文字列表示
01115  */
01116 void GPrint(x,y,ptr,f)
01117 int     x;
01118 int     y;
01119 uchar   *ptr;
01120 Font    *f;
01121 {
01122 ulong   adr;
01123 uchar   dadr;
01124 uint    code,code1;
01125 int     i;
01126 Copy    c;
01127 uchar   cd;
01128         *PITCHS  = 1;
01129         c.x=x;
01130         c.y=y;
01131         c.dh = 15;
01132         c.dv = 15;
01133         c.mode = COPY_AC+S_M;
01134         while(*ptr)
01135         {
01136                 if ( isprint(*((uint *)ptr)) != 0 )
01137                 {
01138                         code1 =0;
01139                         (uchar)code1 = *ptr;
01140                         code1 += 0x2300;
01141                         ptr++;
01142                 }
01143                 else
01144                 {
01145                         code1 = *((uint *)ptr);
01146                         swab(&code1,&code,2);
01147                         code1 = sjtoj(code);
01148                         ptr++;
01149                         ptr++;
01150                 }
01151                 adr = (ulong)code1;
01152                 adr = adr << 4;
01153                 adr += 0x1000001;
01154                 dadr =0;
01155                 for ( i=0;i<f->cnt;i++)
01156                 {
01157                         c.ead2 = adr;
01158                         c.dad2 = dadr++;
01159                         if ( f->magh == 0 && f->magv == 0 && f->slant == 0)
01160                         {
01161                                 GCopy(&c,0,0);
01162                         }
01163                         else
01164                         {
01165                                 GFRCopy(&c,1,0,-(f->slant),-15,f->magh,f->magv);
01166                         }
01167                 }
01168                 c.x += f->face;
01169         }
01170 }
01171 /*
01172  * 全角１文字表示（ＪＩＳコード）
01173  */
01174 void Gputch(x,y,code)
01175 int     x;
01176 int     y;
01177 uint    code;
```

保守／廃止

GIO.C

```
01178 {
01179 ulong   cg_adr;
01180         *PITCHS  = 1;
01181         *X       = x;
01182         *Y       = y;
01183         *DH      = 15;
01184         *DV      = 15;
01185         *MAG     = 0xff;
01186         *CLIP    = NON;
01187
01188         cg_adr = (ulong)code;
01189         cg_adr = cg_adr << 4;
01190         cg_adr += 0x1000001;
01191
01192         *EAD2 = cg_adr;
01193         *DAD2 = 0;
01194         GsetCmd(0x800b);
01195 }
01196 /*
01197  * アウトラインフォント（ＡＧＤＣ２のロゴ）
01198  */
01199 void Glogo(x,y,es,col)
01200 int     x,y,es,col;
01201 {
01202 int     base;
01203 /* A OUTLINE */
01204         GColor(SET,col);
01205         GBoxFill(x,y+es*6,es*2,es*6);
01206         GBoxFill(x+es*4,y+es*6,es*2,es*6);
01207         GBoxFill(x+es*2,y+es*4,es*2,es*2);
01208         GCarc(x+es*3,y+es*6,es*1,x+es*2,y+es*6,x+es*4,y+es*6);
01209         GCarc(x+es*3,y+es*6,es*3,x+0 ,y+es*6,x+es*6,y+es*6);
01210         GPaint(x+es*4+1,y+es*6+1,col);
01211 /* G OUTLINE */
01212         x=x+es*7;
01213         GBoxfill(x,y+es*6,es*2,es*3);
01214         GBoxFill(x+es*3,y+es*5,es*3,es*2);
01215         GCarc(x+es*3,y+es*6,es*1,x+es*2,y+es*6,x+es*4,y+es*6);
01216         GCarc(x+es*3,y+es*6,es*3,x  ,y+es*6,x+es*6,y+es*6);
01217         GLine(x+es*4,y+es*6,x+es*6,y+es*6);
01218         GPaint(x+es*4+1,y+es*6+1,col);
01219         GCarc(x+es*3,y+es*3,es*1,x+es*4,y+es*3,x+es*2,y+es*3);
01220         GCarc(x+es*3,y+es*3,es*3,x+es*6,y+es*3,x,y+es*3);
01221         GPaint(x+es*4+1,y+es*3-1,col);
01222 /* D OUTLINE */
01223         x=x+es*7;
01224         GBoxFill(x,y+es*9,es*3,es*2);
01225         GBoxFill(x,y+es*2,es*3,es*2);
01226         GBoxFill(x,y+es*7,es*2,es*5);
01227         GBoxFill(x+es*4,y+es*6,es*2,es*3);
01228         GCarc(x+es*3,y+es*6,es*1,x+es*3,y+es*7,x+es*4,y+es*6);
01229         GCarc(x+es*3,y+es*6,es*3,x+es*3,y+es*9,x+es*6,y+es*6);
01230         GPaint(x+es*4+1,y+es*6+1,col);
01231         GCarc(x+es*3,y+es*3,es*1,x+es*4,y+es*3,x+es*3,y+es*2);
01232         GCarc(x+es*3,y+es*3,es*3,x+es*6,y+es*3,x+es*3,y);
01233         GPaint(x+es*4+1,y+es*3-1,col);
01234 /* C OUTLINE */
01235         x=x+es*7;
01236         GBoxFill(x,y+es*6,es*2,es*3);
01237         GBoxFill(x+es*4,y+es*6,es*2,es/2);
01238         GBoxFill(x+es*4,y+es*3,es*2,es/2);
01239         GCarc(x+es*3,y+es*6,es*1,x+es*2,y+es*6,x+es*4,y+es*6);
01240         GCarc(x+es*3,y+es*6,es*3,x ,y+es*6,x+es*6,y+es*6);
01241         GPaint(x+es*4+1,y+es*6+1,col);
01242         GCarc(x+es*3,y+es*3,es*1,x+es*4,y+es*3,x+es*2,y+es*3);
01243         GCarc(x+es*3,y+es*3,es*3,x+es*6,y+es*3,x,y+es*3);
01244         GPaint(x+es*2-1,y+es*3-1,col);
01245 /* 2 OUTLINE */
01246         x=x+es*7;
```

保守／廃止

GIO.C

```
01247           GBoxFill(x,y+es*9,es*6,es*2);
01248           GBoxFill(x,y+es*2,es*6,es*2);
01249           GBoxFill(x+es/2,y+es*7,es*2,es*5);
01250           GBoxFill(x+es*6-es/2,y+es*7,-es*2,es*5);
01251   }
01252   /*
```

GIO.C

```
01252 */
01253 /*************************************************************************
01254  *      その他の関数（コマンドの組合せたものなど）
01255  *************************************************************************/
01256 /*
01257  *フレーム表示
01258  */
01259 void Gflame(flg,x,y,dx,dy)
01260         int     flg;
01261         int     x;
01262         int     y;
01263         int     dx;
01264         int     dy;
01265 {
01266 int     xe,ye;
01267         xe = x + dx;
01268         ye = y - dy;
01269         switch(flg )
01270         {
01271         case PUSH :
01272                 Gcolor(SET,BLACK);
01273                 GLine( x-2      ,y+2    ,xe+1   ,y+2);  /* (1) ｳｴ   ｿﾄ */
01274                 GLine( x-1      ,y+1    ,xe     ,y+1);  /* (2) ｳｴ   ﾅｶ */
01275                 GLine( x-1      ,y+2    ,x-1    ,ye-2); /* (7) ﾋﾀﾞﾘ ﾅｶ */
01276                 GLine( x-2      ,y+2    ,x-2    ,ye-2); /* (8) ﾋﾀﾞﾘ ｿﾄ */
01277                 Gcolor(SET,CYAN);
01278                 GLine( x-1      ,ye-2   ,xe+2   ,ye-2); /* (5) ｼﾀ   ｿﾄ */
01279                 GLine( x        ,ye-1   ,xe+2   ,ye-1); /* (6) ｼﾀ   ﾅｶ */
01280                 GLine( xe+2     ,y+1    ,xe+2   ,ye-2); /* (3) ﾐｷﾞ  ｿﾄ */
01281                 GLine( xe+1     ,y      ,xe+1   ,ye-2); /* (4) ﾐｷﾞ  ﾅｶ */
01282                 break;
01283         case PULL :
01284                 Gcolor(SET,CYAN);
01285                 GLine( x-2      ,y+2    ,xe+1   ,y+2);  /* (1) ｳｴ   ｿﾄ */
01286                 GLine( x-1      ,y+1    ,xe     ,y+1);  /* (2) ｳｴ   ﾅｶ */
01287                 GLine( x-1      ,y+2    ,x-1    ,ye-2); /* (7) ﾋﾀﾞﾘ ﾅｶ */
01288                 GLine( x-2      ,y+2    ,x-2    ,ye-2); /* (8) ﾋﾀﾞﾘ ｿﾄ */
01289                 Gcolor(SET,BLACK);
01290                 GLine( x-1      ,ye-2   ,xe+2   ,ye-2); /* (5) ｼﾀ   ｿﾄ */
01291                 GLine( x        ,ye-1   ,xe+2   ,ye-1); /* (6) ｼﾀ   ﾅｶ */
01292                 GLine( xe+2     ,y+1    ,xe+2   ,ye-2); /* (3) ﾐｷﾞ  ｿﾄ */
01293                 GLine( xe+1     ,y      ,xe+1   ,ye-2); /* (4) ﾐｷﾞ  ﾅｶ */
01294                 break;
01295         case ON :
01296                 Gcolor(SET,CYAN);
01297                 GLine( x-2      ,y+2    ,xe+2   ,y+2);  /* (1) ｳｴ   ｿﾄ */
01298                 GLine( x-2      ,y+2    ,x-2    ,ye-1); /* (8) ﾋﾀﾞﾘ ｿﾄ */
01299                 GLine( xe+1     ,y+1    ,xe+1   ,ye-1); /* (4) ﾐｷﾞ  ﾅｶ */
01300                 GLine( x-1      ,ye-1   ,xe+1   ,ye-1); /* (6) ｼﾀ   ﾅｶ */
01301                 Gcolor(SET,BLACK);
01302                 GLine( x-1      ,y+1    ,xe     ,y+1);  /* (2) ｳｴ   ﾅｶ */
01303                 GLine( x-2      ,ye-2   ,xe+2   ,ye-2); /* (5) ｼﾀ   ｿﾄ */
01304                 GLine( xe+2     ,y+1    ,xe+2   ,ye-2); /* (3) ﾐｷﾞ  ｿﾄ */
01305                 GLine(x-1       ,y+1    ,x-1    ,ye  ); /* (7) ﾋﾀﾞﾘ ﾅｶ */
01306                 break;
01307         case OFF :
01308                 Gcolor(SET,BLACK);
01309                 GLine( x-2      ,y+2    ,xe+2   ,y+2);  /* (1) ｳｴ   ｿﾄ */
01310                 GLine( x-2      ,y+2    ,x-2    ,ye-1); /* (8) ﾋﾀﾞﾘ ｿﾄ */
01311                 GLine( xe+1     ,y+1    ,xe+1   ,ye-1); /* (4) ﾐｷﾞ  ﾅｶ */
01312                 GLine( x-1      ,ye-1   ,xe+1   ,ye-1); /* (6) ｼﾀ   ﾅｶ */
01313                 Gcolor(SET,CYAN);
01314                 GLine( x-1      ,y+1    ,xe     ,y+1);  /* (2) ｳｴ   ﾅｶ */
01315                 GLine( x-2      ,ye-2   ,xe+2   ,ye-2); /* (5) ｼﾀ   ｿﾄ */
01316                 GLine( xe+2     ,y+1    ,xe+2   ,ye-2); /* (3) ﾐｷﾞ  ｿﾄ */
01317                 GLine(x-1       ,y+1    ,x-1    ,ye  ); /* (7) ﾋﾀﾞﾘ ﾅｶ */
01318                 break;
01319         }
01320 }
```

保守／廃止

GIO.C

```
01321 /*
01322  * シフトＪＩＳ－＞ＪＩＳコード変換
01323  */
01324 uint sjtoj(k)
01325 uint    k;
01326 {
01327         uint    k1;
01328         k1=(k>>8) & 0xff;
01329         k &= 0xff;
01330
01331         if ( k1 >= 0xe0 ) k1 -= 0x40;
01332         k1 = (k1-0x81)*2+0x21;
01333
01334         if ( k>0x7f) k--;
01335         if ( k>0x9d)
01336         {
01337                 k1++;
01338                 k-=0x9e-0x21;
01339         }
01340         else
01341         {
01342                 k -= 0x40-0x21;
01343         }
01344         return((k1 << 8 ) + k);
01345 }
01346 int rnd(mx)
01347 /*
01348  * 乱数発生
01349  */
01350 int     mx;
01351 {
01352         rndseed = rndseed * 5 + 1;
01353         mx = rndseed % mx;
01354         if ( mx < 0 )
01355         {
01356                 mx = -mx;
01357         }
01358         return(mx);
01359 }
01360 /*
01361  * ウェイト
01362  */
01363 void wait(msec)
01364 uint    msec;
01365 {
01366 ulong   lng;
01367 int     i,j;
01368         for ( i=0;i<msec;i++)
01369         {
01370                 for (lng=0;lng<2501;lng++)
01371                 {
01372                         j *= 2;
01373                 }
01374         }
01375 }
01376 /*
01377  * レンジ変換（拡大／縮小）
01378  */
01379 int GcalES(es_flag,d)
01380 int     es_flag;
01381 int     d;
01382 {
01383 float f;
01384         if ( es_flag == 0 ) return(d);
01385         f = (float)d;
01386         if ( es_flag > 0 && es_flag < 16 )
01387         {
01388                 f = f * 16.0 / (float)(16-es_flag);
01389         }
```

25

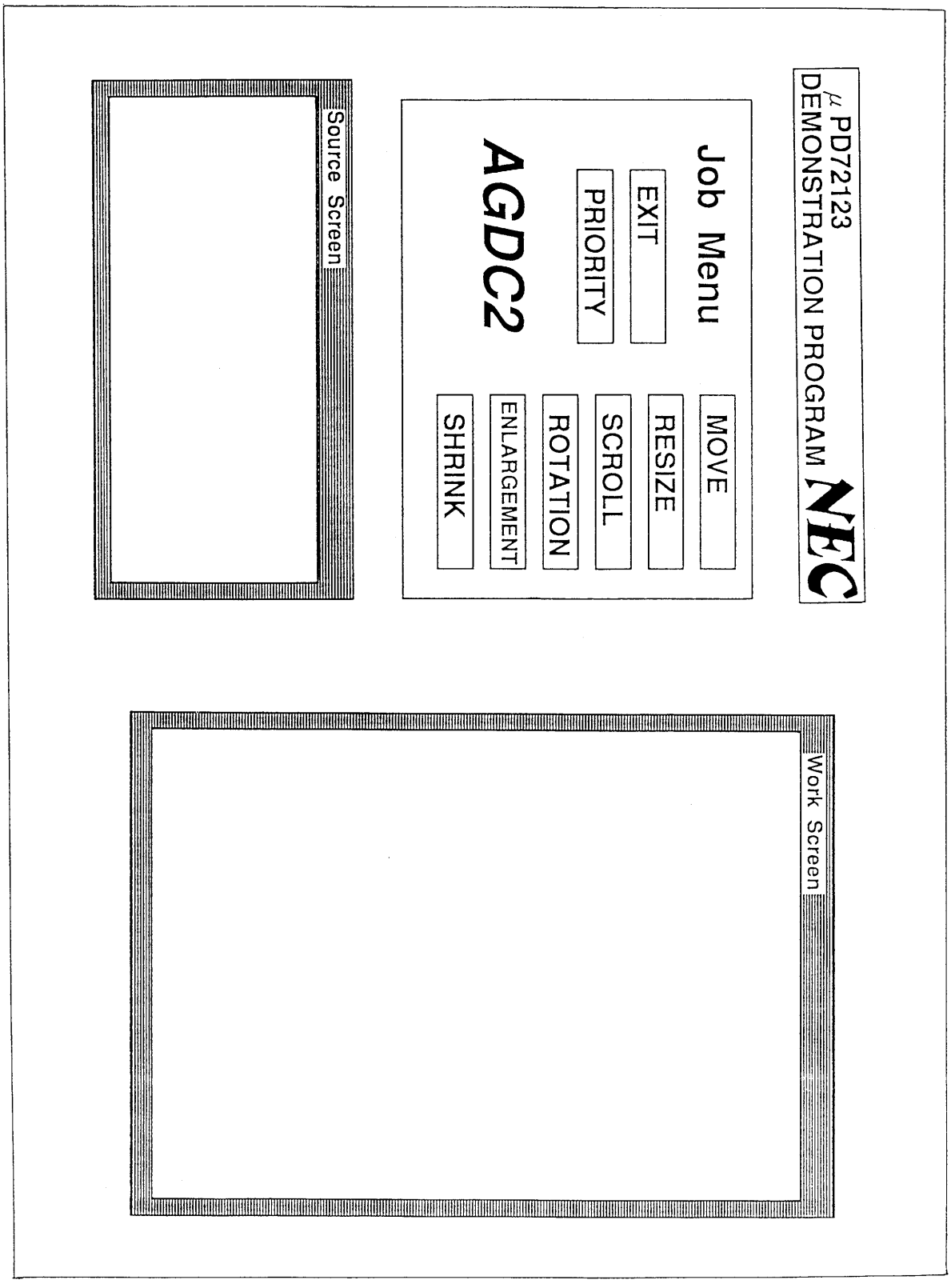
保守／廃止

GIO.C

```
01390           if ( es_flag < 0 && es_flag > -16 )
01391           {
01392                   f = f * (float)(16+es_flag) / 16.0;
01393           }
01394           if ( f < 0.0 )
01395           {
01396                   f -= 0.5;
01397           }
01398           else
01399           {
01400                   f += 0.5;
01401           }
01402   /*
01403           f -= 1;
01404   */
01405           return((int)f);
01406   }
01407   /*
01408    * タイリング・パターン設定
01409    */
01410   void GsetTile(t)
01411   Tilling *t;
01412   {
01413   int     i,j,r,cnt;
01414   uint    buf[500];
01415   ulong   adr;
01416   uint    far *ptr;
01417           ptr = t->ptr;
01418           for ( i=0;i<t->plane;i++)
01419           {
01420                   adr = t->org.eadorg + (ulong)i * t->org.pdisp;
01421                   adr += t->off_adr;
01422                   GorgDst(t->org);
01423                   Gpmax(1);
01424                   Gcolor(SET,RGB);
01425                   GnonClip();
01426                   GPutA(adr,t->dh,t->dv);
01427                   cnt = ((t->dh+1) / 16 ) * (t->dv+1);
01428                   for ( j=0;j<cnt;j++)
01429                   {
01430                           r=GPgport(*ptr++);
01431                           if ( r == BREAK ) break;
01432                           if ( r == ERROR ) return(ERROR);
01433                   }
01434           }
01435   }
```

保守／廃止

付録Ｂ　見本画面



保守／廃止

保守／廃止

アンケート記入のお願い

お手数ですが，このドキュメントに対するご意見をお寄せください。今後のドキュメント作成の参考にさせていただきます。

［ドキュメント名］ μPD72123 アプリケーション・ノート（III） ウィンドウ編

（IEA-712（第1版））

［お名前など］（さしつかえのない範囲で）

御社名（学校名，その他） （ ）
ご住所 （ ）
お電話番号 （ ）
お仕事の内容 （ ）
お名前 （ ）

1．ご評価（各欄に○をご記入ください）

項　　目	大変良い	良　い	普　通	悪　い	大変悪い
全体の構成					
説明内容					
用語解説					
調べやすさ					
デザイン，字の大きさなど					
そ の 他（ ）					
（ ）					

2．わかりやすい所（第　章，第　章，第　章，第　章，その他 ）

理由［ ］

3．わかりにくい所（第　章，第　章，第　章，第　章，その他 ）

理由［ ］

4．ご意見，ご要望

5．このドキュメントをお届けしたのは

NEC 販売員，特約店販売員， NEC 半応技本部員，その他（ ）

ご協力ありがとうございました。

下記あてに FAX で送信いただくか，最寄りの販売員にコピーをお渡しください。

NEC 半導体応用技術本部インフォメーションセンター
FAX：(044)548-7900

キリトリ

保守／廃止

保守／廃止

お問い合わせは，最寄りのNECへ

本　　社	〒108-01 東京都港区芝五丁目7番1号（NEC本社ビル）	
コンシューマ半導体販売事業部 OA半導体販売事業部 インダストリ半導体販売事業部	〒108-01 東京都港区芝五丁目7番1号（NEC本社ビル）	東京(03)3454-1111
中部支社 半導体販売部	〒460 名古屋市中区栄四丁目14番5号（松下中日ビル）	名古屋(052)242-2755
関西支社 半導体販売部	〒540 大阪市中央区城見一丁目4番24号（NEC関西ビル）	大阪(06)945-3178 大阪(06)945-3200 大阪(06)945-3208

北海道支社	札幌	(011)231-0161
東北支社	仙台	(022)261-5511
岩手支店	盛岡	(0196)51-4344
山形支店	山形	(0236)23-5511
郡山支店	郡山	(0249)23-5511
いわき支店	いわき	(0246)21-5511
長岡支店	長岡	(0258)36-2155
水戸支店	水戸	(0292)26-1717
神奈川支社	横浜	(045)324-5511
群馬支店	高崎	(0273)26-1255
太田支店	太田	(0276)46-4011
宇都宮支店	宇都宮	(0286)21-2281
小山支店	小山	(0285)24-5011
長野支社	長野	(0262)35-1444
松本支店	松本	(0263)35-1666
上諏訪支店	諏訪	(0266)53-5350
甲府支店	甲府	(0552)24-4141
埼玉支社	大宮	(048)641-1411
立川支社	立川	(0425)26-5981
千葉支社	千葉	(043)238-8116
静岡支社	静岡	(054)255-2211
沼津支店	沼津	(0559)63-4455
浜松支店	浜松	(053)452-2711
北陸支社	金沢	(0762)23-1621
福井支店	福井	(0776)22-1866
富山支店	富山	(0764)31-8461
京都支社	京都	(075)344-7824
神戸支社	神戸	(078)332-3311
中国支社	広島	(082)242-5504
鳥取支店	鳥取	(0857)27-5311
岡山支店	岡山	(086)225-4455
四国支社	高松	(0878)36-1200
新居浜支店	新居浜	(0897)32-5001
松山支店	松山	(0899)45-4111
九州支社	福岡	(092)271-7700
北九州支店	北九州	(093)541-2887

（技術お問い合わせ先）

OA半導体販売事業部 OAシステム技術部	〒210 川崎市幸区塚越三丁目484番地	川崎(044)548-7921
半導体応用技術本部 中部応用システム技術部	〒460 名古屋市中区栄四丁目14番5号（松下中日ビル）	名古屋(052)242-2762
半導体応用技術本部 西日本応用システム技術部	〒540 大阪市中央区城見一丁目4番24号（NEC関西ビル）	大阪(06)945-3383

半導体応用技術本部
インフォメーションセンター
FAX(044)548-7900
（FAXで対応させていただいております）

A93.5