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（Ⅱ）

RENESAS

保守／廃止

μPD72123

AGDC Ⅱ

ソフトウエア編

資料番号　IEA-711（第1版）
発行年月　May 1993 P

アプリケーション・ノート（Ⅱ）

NEC

保守／廃止

μPD72123

AGDC Ⅱ

ソフトウエア編

保守／廃止

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，本製品と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。

保守／廃止

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
- ●コンパイルおよびリンク方法
- ●移植方法とテスト・プログラム

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	：気を付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

保守／廃止

目　　次

保守／廃止

図　の　目　次

表　の　目　次

第1章　概　　説

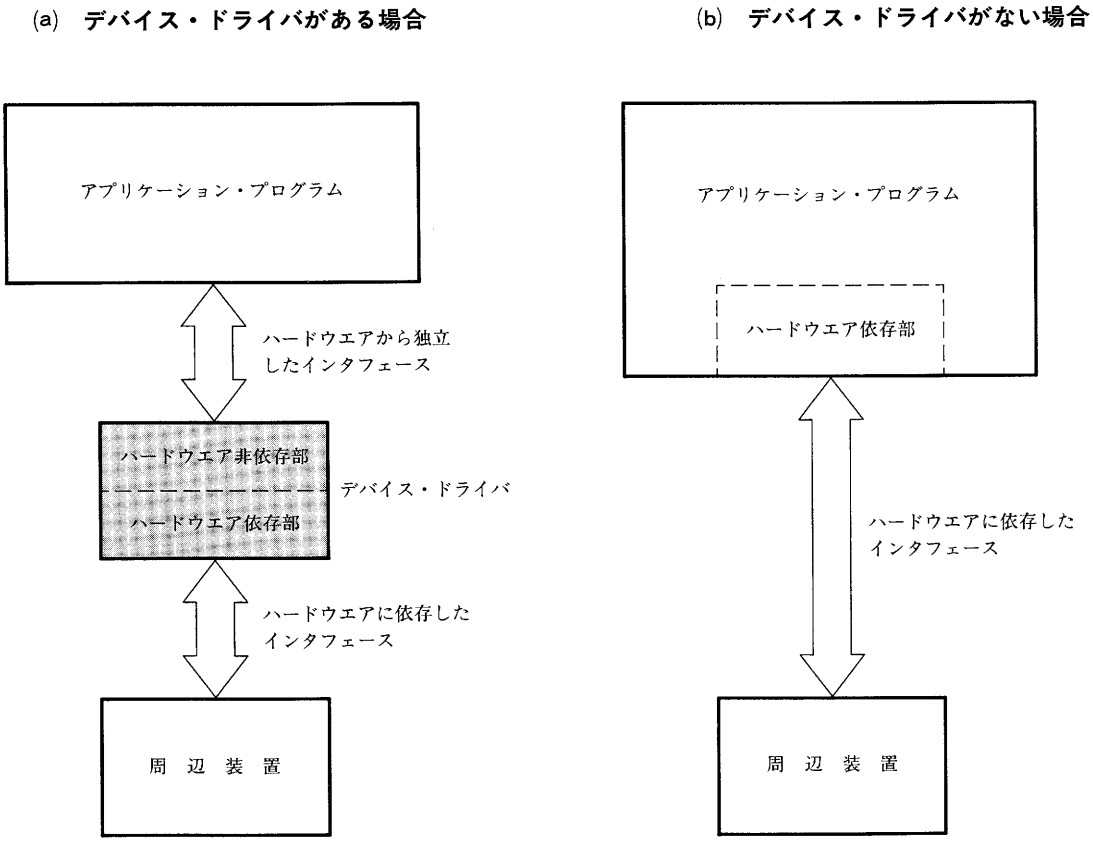
1.1　デバイス・ドライバとは？

業務で使用するプログラム（アプリケーション・プログラム）では，いろいろな周辺装置（ハード・ディスク装置やプリンタ，CRTなど）を動作させる必要があります。このためアプリケーション・プログラムには，以上のような装置を駆動させるためのプログラムを内蔵しています。ただし，このような周辺装置とのインタフェース部分はハードウエアへの依存が高いため，異なるハードウエアを用いようとする場合には，プログラムを書き換える必要が生じます。

アプリケーション・プログラムを図1－1(a)のようにハードウエア依存部と非依存部の一部をひとまとめにして，アプリケーション・プログラムの外に出せれば，アプリケーション・プログラムがハードウエアの影響を受けずにすみます。これを［デバイス・ドライバ］といいます。

［デバイス・ドライバ］とは，アプリケーション・プログラムと装置との橋渡しをする一種のインタフェース・プログラムです。ハードウエア依存部をその中に取り組むことによって，個々のハードウエアの差がアプリケーション・プログラムに影響しないようにしています。

図1－1　デバイス・ドライバの位置付け



1.2 デバイス・ドライバのメリット

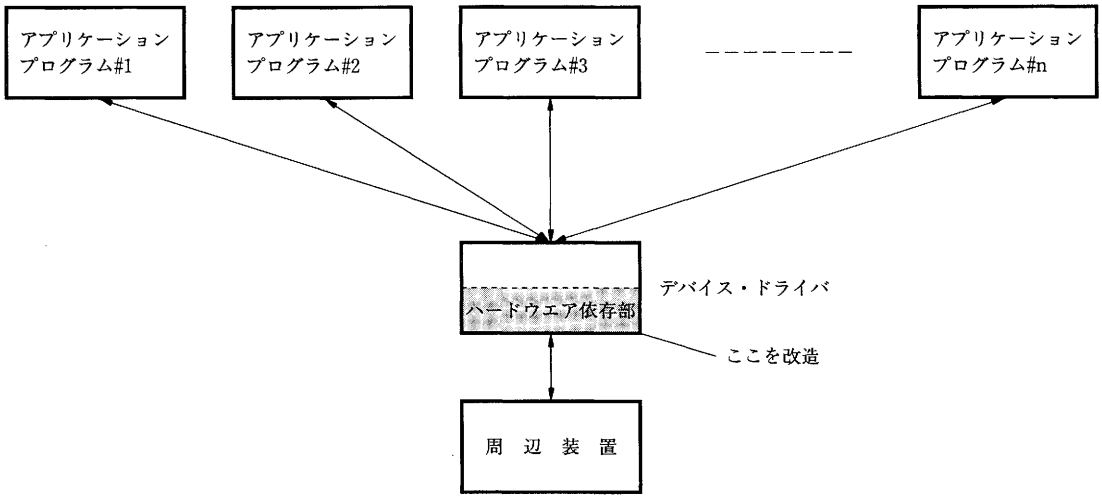
デバイス・ドライバを作るメリットとは，アプリケーション・プログラムをハードウエアから独立させることです。

アプリケーション・プログラムが多数あるものの周辺装置のハードウエアを変更したとすると，図1-2(b)のようにデバイス・ドライバがない場合には，アプリケーション・プログラムを改造しなければなりません。これでは，アプリケーション・プログラム数が増大するほど大変な作業になります。しかし，図1-2(a)のようにデバイス・ドライバがある場合には，デバイス・ドライバのみを改造すればよくなります（アプリケーション・プログラム自体を改造する必要はありません)。

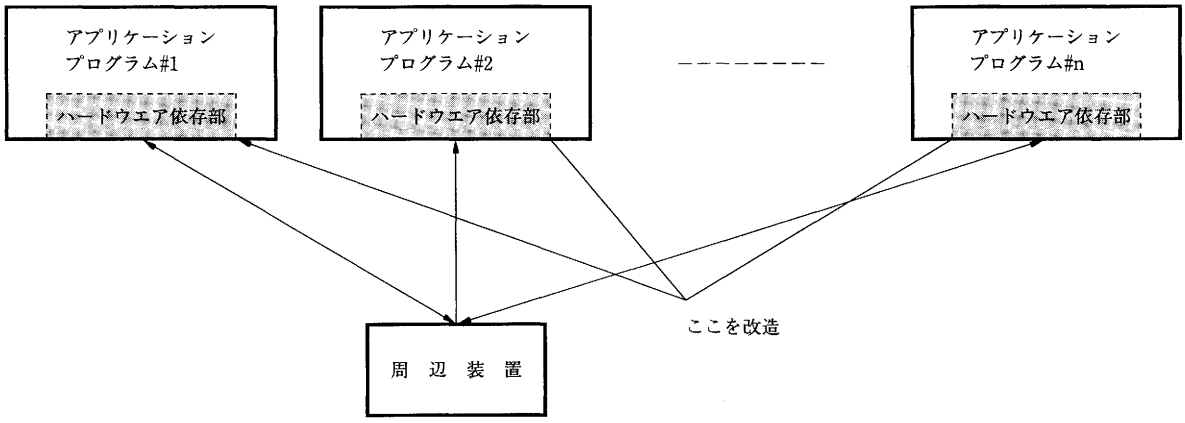
このため，図1-2(b)の場合と比較すると，アプリケーション・プログラムの移植がはるかに容易になります。

図1-2　デバイス・ドライバのメリット

(a)　**デバイス・ドライバがある場合**



(b)　**デバイス・ドライバがない場合**



第２章　ボードの仕様

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。

これによりあと，特に断りのない限り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

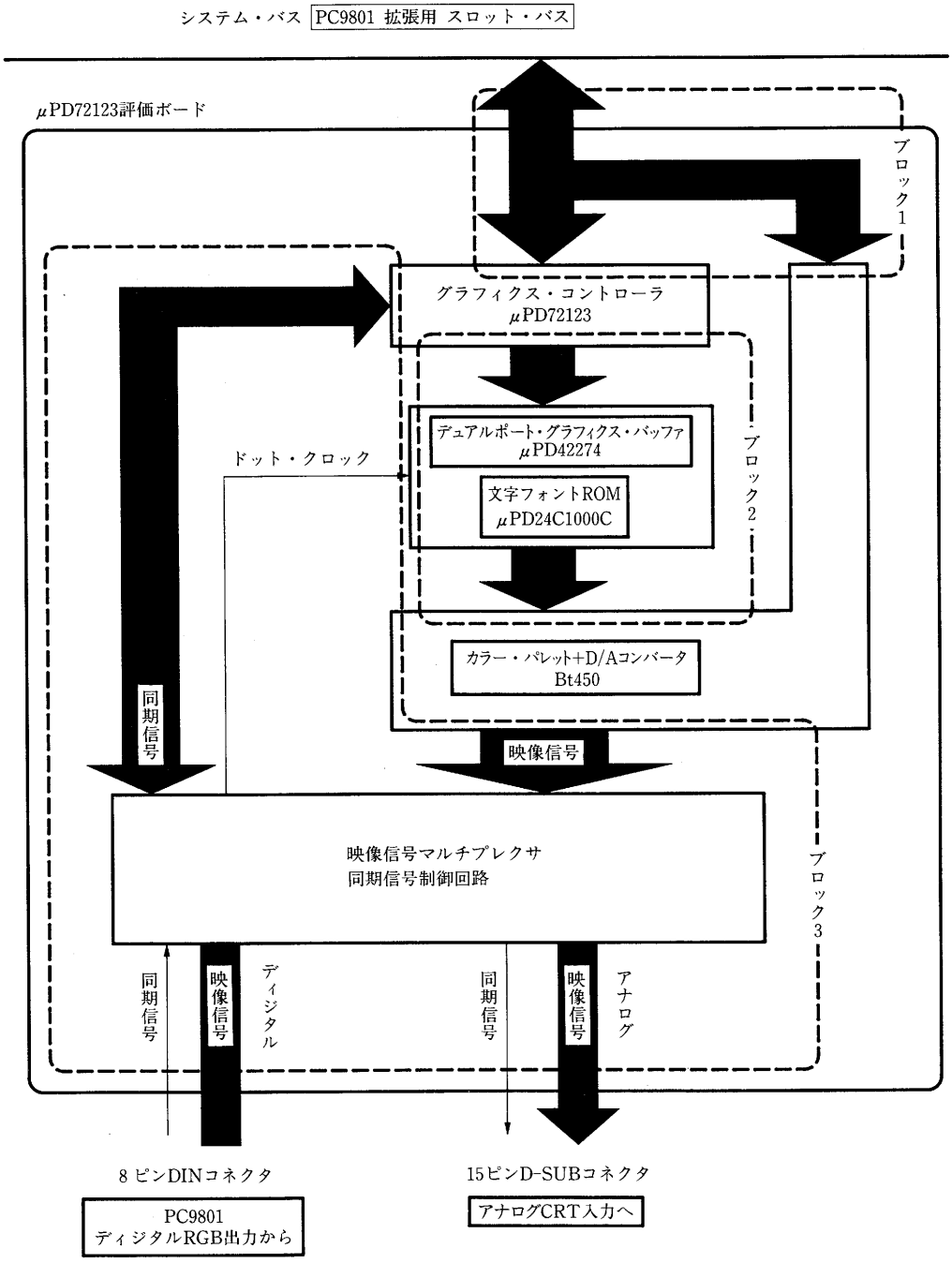
２.１.１　ボード概略図

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・ウインドウ)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・ウインドウ）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・ウインドウ，メモリ・ウインドウは，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。

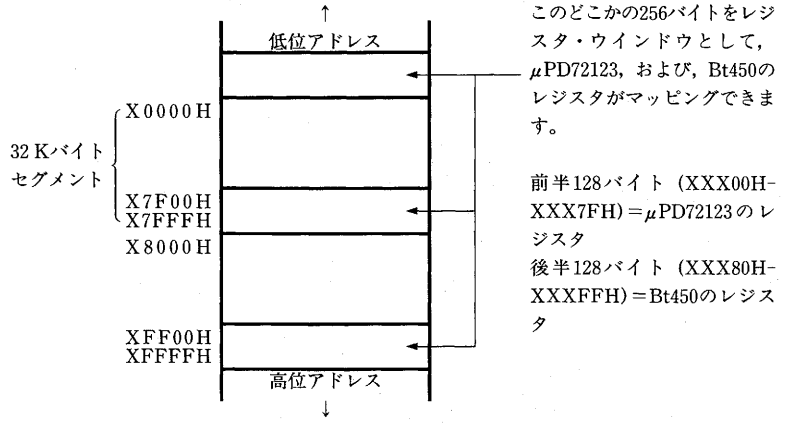
表２－７　00D2H番地（I/O空間）レジスタ

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，表下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。



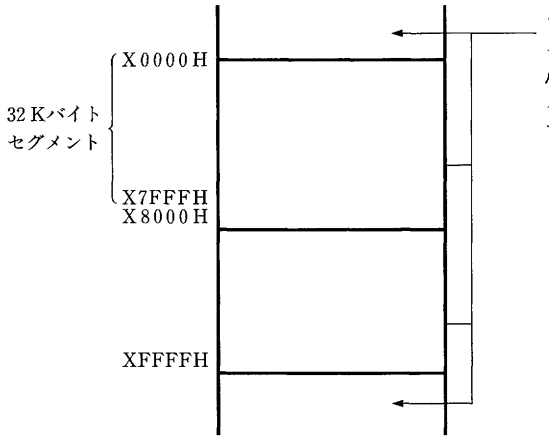
RSEG値	レジスタ・ウインドウ
0H	87F00H-87FFFH
1H	8FF00H-8FFFFH
2H	97F00H-97FFFH
3H	9FF00H-9FFFFH
4H	A7F00H-A7FFFH
5H	AFF00H-AFFFFH
6H	B7F00H-B7FFFH
7H	BFF00H-BFFFFH
8H	CF700H-C7FFFH
9H	CFF00H-CFFFFH
AH	D7F00H-D7FFFH
BH	DFF00H-DFFFFH
CH	E7F00H-E7FFFH
DH	EFF00H-EFFFFH
EH	F7F00H-F7FFFH
FH	FFF00H-FFFFFH

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

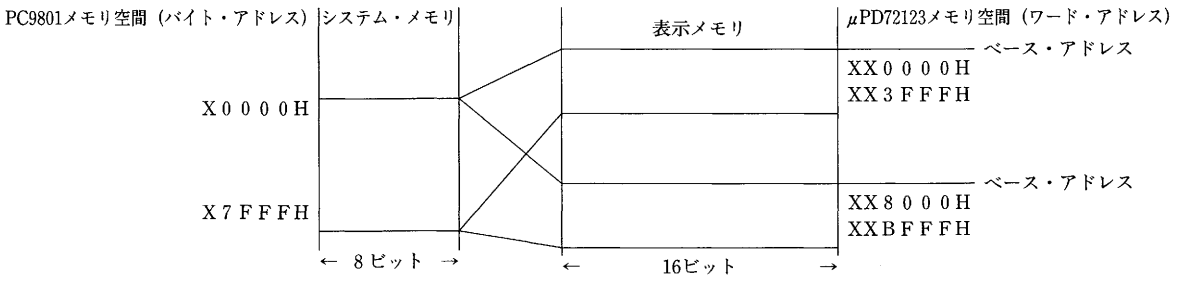
	機　　能
VSEG Video RAM SEGment	μPD72123，および，Bt450のレジスタをPC9801のメモリ空間内のどの領域にマップするかを決定します。CPUが出力するアドレスのビット18-ビット15の値と，VSEGに設定された4ビットとが下記のように比較されます。 アドレス・ビット→19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 アドレス値　→1 Y Y Y Y X X X X X X X X X X X X X X X：don't care　Y：VSEGに設定された4ビット 比較結果が真であり，かつ，00D0H番地レジスタ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



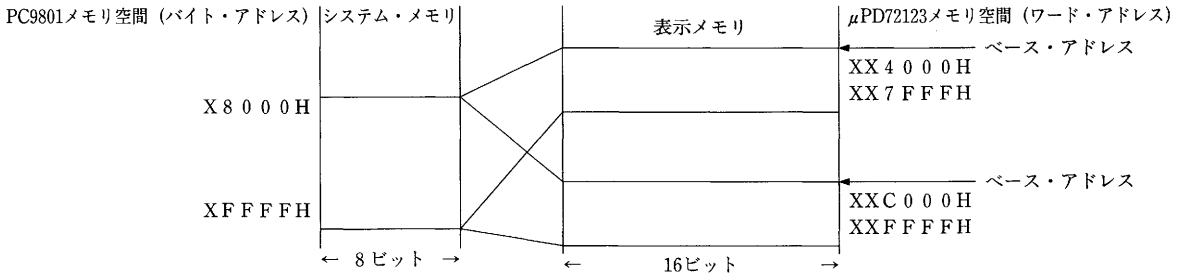
VSEG値	メモリ・ウインドウ
0H	80000H-87FFFH
1H	88000H-8FFFFH
2H	90000H-97FFFH
3H	98000H-9FFFFH
4H	A0000H-A7FFFH
5H	A8000H-AFFFFH
6H	B0000H-B7FFFH
7H	B8000H-BFFFFH
8H	C0000H-C7FFFH
9H	C8000H-CFFFFH
AH	D0000H-D7FFFH
BH	D8000H-DFFFFH
CH	E0000H-E7FFFH
DH	E8000H-EFFFFH
EH	F0000H-F7FFFH
FH	F8000H-FFFFFH

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・ウインドウとした（VSEGに奇数値を設定した）場合】



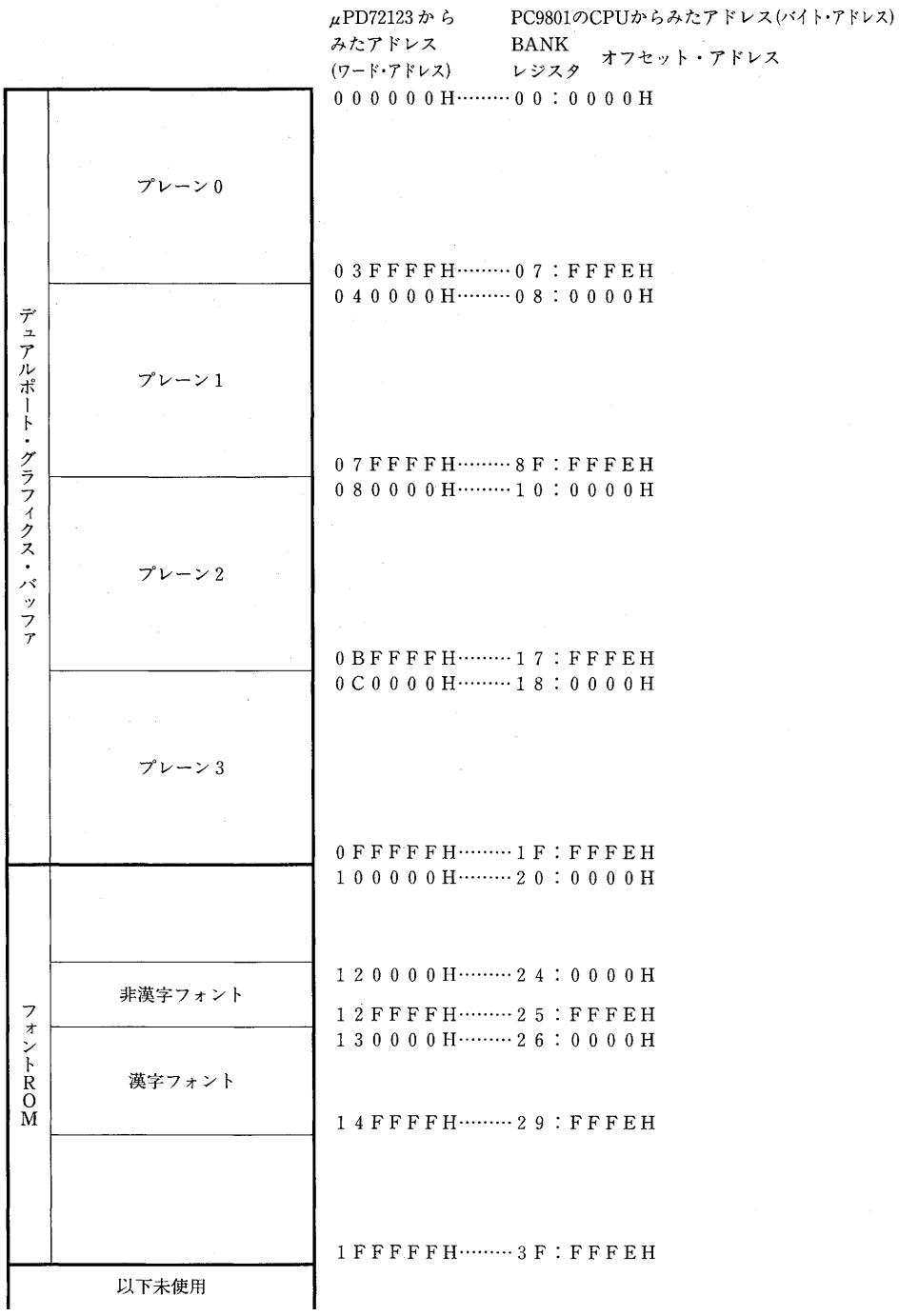
【X0000H-X７FFFHをメモリ・ウインドウとした（VSEGに奇数値を設定した）場合】



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。



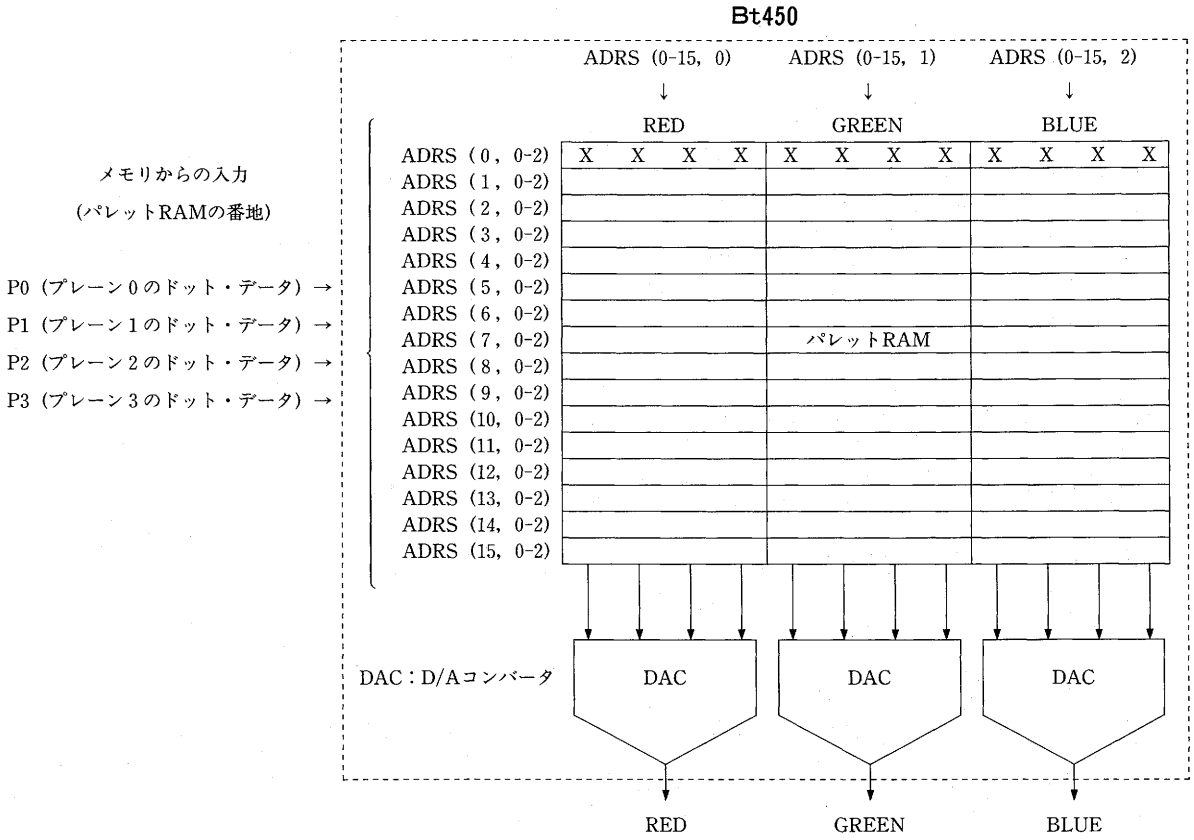
備考　フォント・データのゴーストが読み出される領域

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・ウイ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

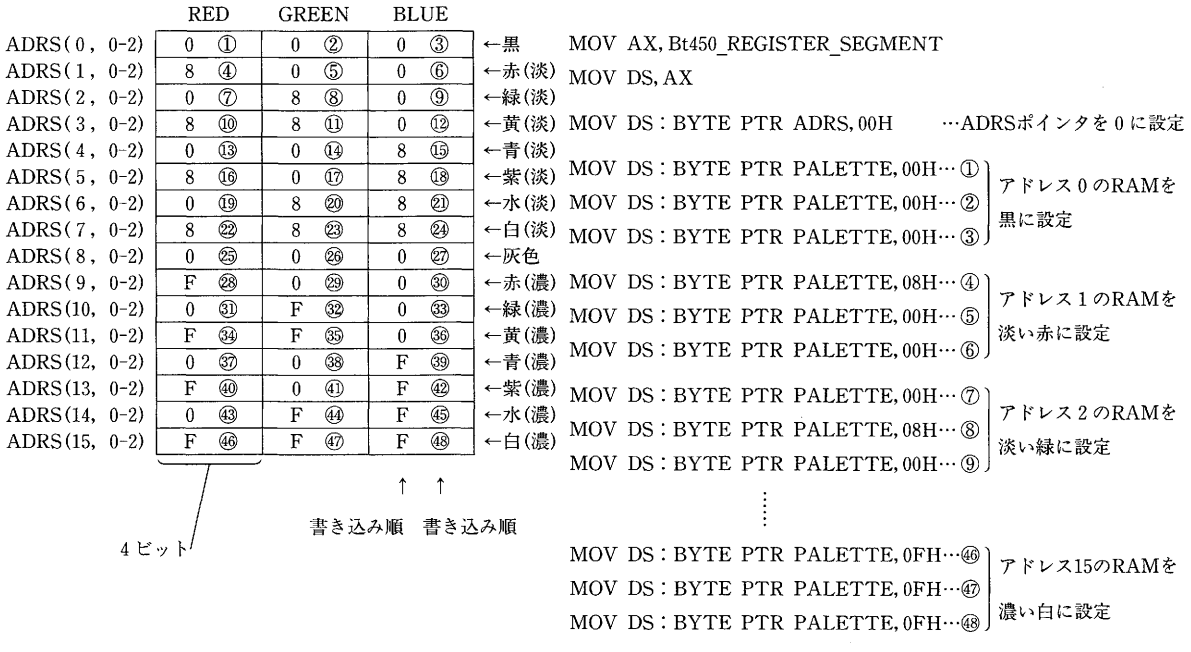
オフセット・アドレス	機　　能	
0000H-007FH	μPD72123のレジスタ群	
0080H	Bt450のアドレス・レジスタADRS （５ビット・レジスタ）	データ・レジスタに設定した値をどの番地のパレットRAMに書き込むかを指定
0082H	Bt450のデータ・レジスタPALETTE （４ビット・レジスタ）	パレットRAM設定値を書き込む

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



保守／廃止

第３章　各関数の説明

３.１　外部開放関数一覧

関数名	機能名
gInit ()	グラフィック・ドライバ初期化
gCreatePage ()	描画専用画面作成
gDeletePage ()	描画専用画面削除
gResetCliparea ()	クリップ・エリア・リセット
gSetCliparea ()	クリップ・エリア・セット
gEnd ()	グラフィクス描画終了
gDrawPage ()	描画対象画面の指定
gDrawPlane ()	描画対象プレーンの指定
gSetBackCode ()	背景色の指定
gSetDrawCode ()	カラー・コードの指定
gSetMode ()	描画モードの指定
gSetLinePtn ()	線種パターン設定
gEntryFillPtn ()	塗りつぶしパターン登録
gSetFillPtn ()	塗りつぶしパターン設定
gSetGpenPtn ()	グラフィクス・ペン・パターン設定
gSetGpenMask ()	クラフィクス・ペン・マスク設定
gSetCharSize ()	文字描画サイズ設定
gSetCharAngle ()	文字回転角の設定
gSetTextAlignment ()	文字列配置の設定
gSetCopyAngle ()	コピー回転角の設定
gClearScreen ()	画面消去
gDot ()	１ドットの点の描画
gGpDot ()	グラフィクス・ペンによる点の描画
gLine ()	１ドット幅の直線
gGpLine ()	グラフィクス・ペンによる直線
gTriangle ()	１ドット幅の三角形の描画
gGpTriangle ()	グラフィクス・ペンによる三角形の描画
gPolygon ()	１ドット幅の多角形の描画
gGpPolygon ()	グラフィクス・ペンによる多角形の描画
gRectangle ()	１ドット幅のボックスの描画
gGpRectangle ()	グラフィクス・ペンによるボックスの描画

保守／廃止

関数名	機能名
gCircle ()	１ドット幅の円の描画
gGpCircle ()	グラフィクス・ペンによる円の描画
gCircArc ()	１ドット幅の円弧の描画
gGpCircArc ()	グラフィクス・ペンによる円弧の描画
gEllipse ()	１ドット幅の楕円の描画
gGpEllipse ()	グラフィクス・ペンによる楕円の描画
gElpsArc ()	１ドット幅の楕円弧の描画
gGpElpsArc ()	グラフィクス・ペンによる楕円弧の描画
gPaint ()	塗りつぶし
gChangeColor ()	塗り直し
gFillTriangle ()	三角形塗りつぶし
gFillRectangle ()	ボックスの塗りつぶし
gFillPolygon ()	多角形の塗りつぶし
gFillCircle ()	円の塗りつぶし
gFillCircArc ()	円弧の塗りつぶし
gFillEllipse ()	楕円の塗りつぶし
gFillElpsArc ()	楕円弧の塗りつぶし
g3opCopy ()	３オペランド・コピー
gCopyScr ()	画面コピー
gGetImage ()	画面情報の読み出し
gPutImage ()	画面情報の書き込み
gPutText ()	文字列の描画
gPutChar ()	文字の描画
gDispPlane ()	表示プレーン指定
gGetColor ()	色情報読み出し
gGetTip ()	最後に描いた弧の範囲
gBitSearch ()	色情報の検索
gPeek ()	表示メモリの読み出し
gPoke ()	表示メモリへの書き込み

保守／廃止

３.２　関数説明

３.２.１　グラフィクス・ドライバ初期化

［シーケンス］

```
int     mode；
int     depth；
gInit（mode，depth）；
error   関数値！＝0
```

［シーケンス］

mode　．．　表示画面分解能

0＝640×400

1＝1024×768

2＝1120×750

depth　．．　画面構成プレーン数

［機　　能］

- グラフィック・ドライバの初期化を行います。
- この関数を，このライブラリに含まれる他のどの関数よりも先に，一度だけ実行してください。この関数よりも先に他の関数を実行させた場合の動作は保証されません。
- 表示画面の分解能を指定された大きさに設定します。表示分解能配置度グラフィック描画終了（gEnd（））を実行しなければ変更することはできません。
- カラー・パレットの値の初期化は行いません（ライブラリでサポートしていませんので，ユーザが自分で行ってください）。

保守／廃止

3.2.2　グラフィクス描画終了

［シーケンス］

gEnd ( ) ;

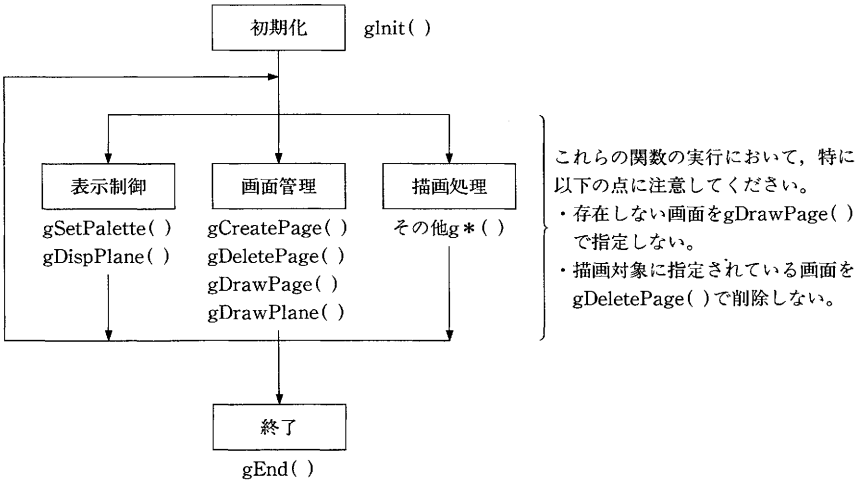
no error status

［機　　能］

●グラフィック・ドライバを以後使用しない場合に実行します。

●この関数を，アプリケーションが終了するまでの間に必ず一度は実行してください。実行しない場合は，システムに戻ってからの正常な動作が保証されません。

関数の組立



保守／廃止

３.２.３　描画専用画面作成

[シーケンス]

```
int     gCreatePage；
int     w，h，d；
int     scrn_no；
scrn_no＝gCreatePage（w，h，d）；
```

[入　　力]

w	. .	画面の幅（ドット数）
h	. .	画面の高さ（ドット数）
d	. .	画面の深さ（プレーン数）

[出　　力]

scrn_no	. .	画面番号

[機　　能]

- ●指定された大きさ（w×h×d）の描画専用の画面を作成します。
  作成できない場合は，画面番号が負の値をとります。
- ●この画面は，描画対象にすることはできますが，表示対象にすることはできません（表示させる場合は，gCopyScr（）を使って表示画面に内容をコピーしてください）。
- ●カレントの描画対象画面は，実行の前後で変化しません。
- ●作成した画面を描画処理の対象にするには，gDrawPage（）を実行します。パラメータとして，この関数値として与えられる画面番号を使用します。
- ●最大32個の描画専用画面を作ることができます。ただし，画面作成のためのメモリの限界に達した場合はこの限りではありません。

保守／廃止

３.２.４　描画専用画面削除

［シーケンス］

int　　scrn_no；

gDeletePage（scrn_no）；

error　　関数値！＝0

［入　　力］

scrn_no　．．　　画面番号

［機　　能］

●指定された番号の画面を削除します。以後，指定された画面に対する描画はいっさい行わないでください。

●現在描画対象になっている画面を削除しないでください。やむをえず削除する場合は，他の描画処理に先駆けてgDrawPage（）で他の存在する画面を，描画処理対象に設定しておいてください。削除したままで描画処理を行った場合の動作はいっさい保証されません。

●指定された画面が存在しない場合は，エラーになります。

保守／廃止

３.２.５　クリップ・エリア・リセット

［シーケンス］

gResetCliparea（）；

no error status

［機　　能］

●現在描画対象になっている画面の全域をクリップ・エリアに設定します。

保守／廃止

3.2.6　クリップ・エリア・セット

[シーケンス]

int　　x0, y0, x1, y1 ;

gSetCliparea (x0, y0, x1, y1) ;

no error status

[入　　力]

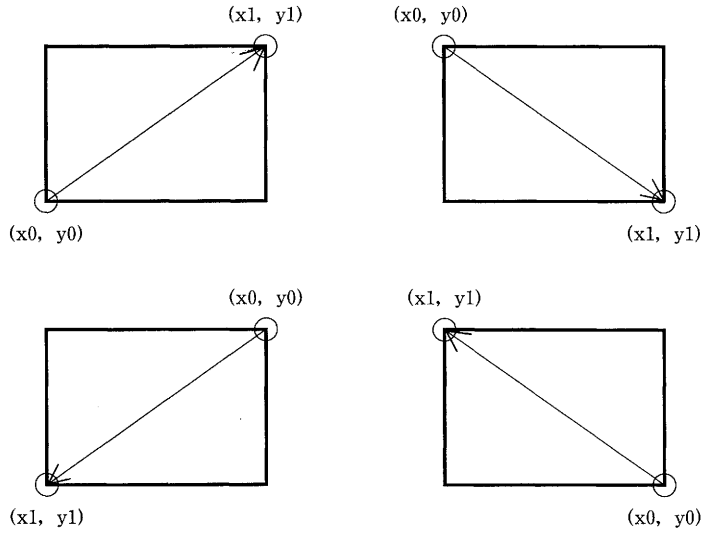
(x0, y0) － (x1, y1) ．．　　クリップ・エリア

[機　　能]

●指定された範囲をクリップ・エリアとして設定します。範囲は現在の描画対象画面の範囲を越えないようにしてください。

● 2 点 (x0, y0) と (x1, y1) の位置関係は任意にとることができます。

クリップエリアの指定形式



保守／廃止

３.２.７　描画対象画面の指定

［シーケンス］

```
int        scrn_no；
gDrawPage（scrn_no）；
error      関数値！＝0
```

［入　　力］

scrn_no　．．画面番号

［機　　能］

- 指定された番号の画面を以後の描画対象として設定します。この関数で設定した画面に対して，図形の描画が行われます。
- クリップ・エリアと描画対象プレーンは，指定した画面に対して最後に設定された範囲が設定されますが，他の描画属性（描画カラーや描画モードなど）は変化しません（切り替える前の状態が継続します)。
- 存在しない画面を指定しないでください。
- エラーが発生した場合は，描画対象画面は変化しません。

保守／廃止

３.２.８　描画対象プレーンの指定

［シーケンス］

```
int       top, count ;
gDrawPlane (top, count) ;
no error status
```

［入　　力］

top　．．　プレーン番号

count　．．　プレーン枚数

［機　　能］

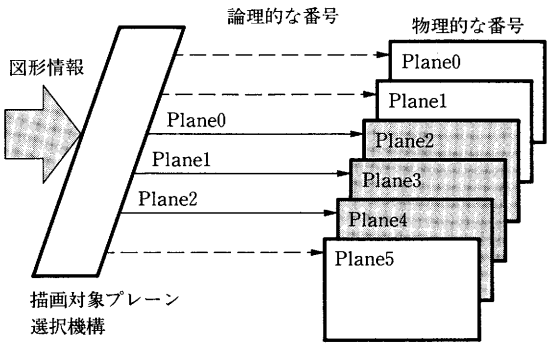
●指定されたプレーン番号から指定された枚数のプレーン数を描画対象に限定します。
指定範囲以外のプレーンは，描画対象から外されます。topで指定された番号のプレーンが画面を構成する先頭プレーン（プレーン０）として扱われるようになり，count枚で構成される画面となります。ここで使用するプレーンの番号は画面を構成するプレーンの先頭を０とする**物理的な番号**です。
以後はtopで指定されたプレーンがプレーン０になります（**論理的なプレーン番号**）。

●この関数実行後は，カラー・コード１で描画するとtopで指定されたプレーンに対して，１による描画が行われる点に注意してください（topの値には左右されません）。

●プレーン番号topとして負の値を指定すると，描画対象プレーンの指定がリセットされます(画面を構成するプレーン全部が描画対象になります)。

●描画対象プレーンは，存在する画面に対して個別に定義されます。したがって，gDrawPage( ) によって画面を切り替えた場合は，その画面に設定されていた描画対象プレーンが選択されます（それまでの画面に設定されていた描画対象プレーンの設定は，次にその画面を選択したときに有効になります)。

描画対象プレーン概念とプレーン番号



※６枚のプレーンで構成される画面のうちプレーン２（物理番号）から３枚のプレーンを描画対象にした場合の例

保守／廃止

３.２.９　背景色の指定

［シーケンス］

```
int        colcode ;
gSetBackCode (colcode) ;
no error status
```

［入　　力］

colcode　．．　背景カラー・コード

［機　　能］

●背景色を指定されたカラー・コードに設定します。
現バージョンでは，画面をクリアするときのカラー・コードとして使用されます。

●カラー・コードは，カラー・パレットに設定された色のインデクスであり，実際にモニタに表現される色とは異なることに注意してください。

保守／廃止

３.２.１０　カラー・コードの指定

［シーケンス］

int　　colcode；

gSetDrawCode（colcode）；

no error status

［入　　力］

colcode　．．　描画カラー・コード

［機　　能］

●描画するときのカラー・コードを設定します。

●おのおのの図形描画における描画カラーとして使用されます。

●描画時には，指定されている描画モードに従って演算が行われます。

●カラー・コードは，カラー・パレットに設定された色のインデクスであり，実際にはモニタに表現される色とは異なることに注意してください。

保守／廃止

３.２.１１　描画モードの指定

［シーケンス］

int　　mode；

gSetMode（mode）；

no error status

［入　　力］

mode　．．　描画モード

b0．．　b3＝カラー・コードのビット値が０のプレーンに行う演算（演算０）

b4．．　b7＝カラー・コードのビット値が１のプレーンに行う演算（演算１）

［機　　能］

●図形描画時に行う演算を設定します。

●描画モードは，２種類の演算コードのコンビネーション・コードになっています。

演算０は，描画カラー・コードのビット値が０のプレーンに対して使用され，演算１は，描画カラー・コードのビット値が１のプレーンに対して使用されます。演算０，演算１に設定する値については，**ユーザーズ・マニュアル**を参照してください。また，「**３.３　演算効果**」も参照してください。

保守／廃止

３.２.１２　線種パターン設定

［シーケンス］

int　　type；

voit ＊linestyle；

gSetLineptn (type, linestyle)；

no error status

［入　　力］

type　　．．　ライン・スタイル種

0＝実線

1＝16ビット長ライン・スタイル

2＝32ビット長ライン・スタイル

linestyle　．．　ライン・スタイル定義データ

［機　　能］

●直線，円楕円，円弧などの直線による図形における線のスタイルを設定します。

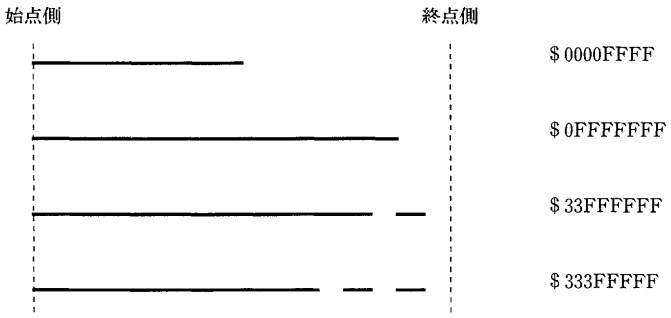
●ライン・スタイルは，ライン・スタイル種の値によって定義形式が異なります。

●実線の場合，ライン・スタイル定義データは省略できます（指定しても無視されます）。

●16ビット長ライン・スタイルの場合，ライン・スタイル定義データはunsigned shortで定義します。

●32ビット長ライン・スタイルの場合，ライン・スタイル定義データはunsigned longで定義します。

ライン・スタイル例（32ビット長）



保守／廃止

３.２.１３　塗りつぶしパターン登録

［シーケンス］

```
int     id ;
int     type ;
int     n ;
void    *pattern ;
gEntryFillPth (id, type, n, pattern) ;
error   関数値！＝0
```

［入　　力］

id	. .	パターン番号
type	. .	パターン種
		0＝ベタ
		1＝単色16ビット幅パターン
		2＝カラー16ドット幅パターン
		6＝カラー32ドット幅パターン
n	. .	パターン・ライン数
pattern	. .	パターン定義データ

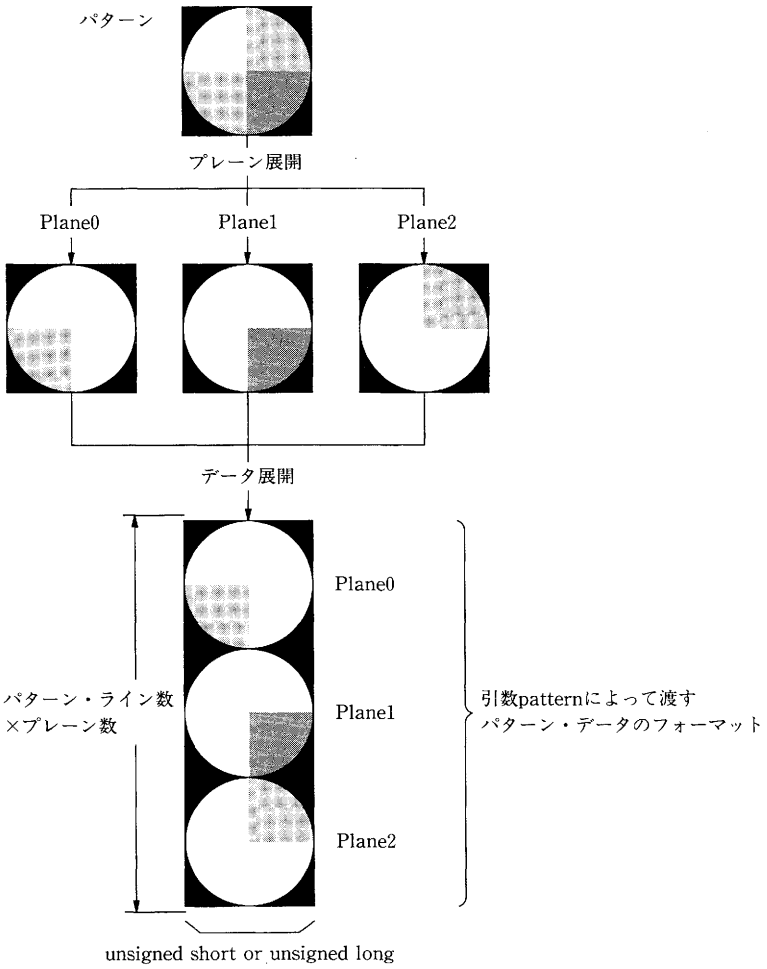
［機　　能］

- 塗りつぶしで使用するパターンの登録を行います。初期状態では，すべてベタになっています。
- パターンの定義形式はパターンの種類によって異なります。
- ベタの場合は，パターン定義形式データは省略できます（指定しても無視されます）。
- 単色16ドット幅パターンの場合は，unsigned shortの配列として定義します。配列要素の数は，指定するライン数以上でなければいけません。
- カラー16ドット幅パターンの場合は，unsigned shortの配列として定義します。配列要素の数は，指定するライン数に描画対象画面の描画対象プレーン数を乗じた値でなければいけません。
- カラー32ドット幅パターンの場合は，unsigned longの配列として定義します。配列要素の数は，指定するライン数に描画対象画面の描画対象プレーン数を乗じた値でなければなりません。
- パターン番号は，0から始まる整数値で，どの番号から使い始めるかはユーザに一任されます。
- パターン１個あたりの総容量数は12 KB，登録可能なパターン数は48個となっています。この制限を越える内容は指定しないでください。

保守／廃止

●この関数を実行しただけでは，塗りつぶしのパターンの内容は変化しません。実際に描画するときに使うパターンを変更するには，gSetFillPtn（ ）を実行します。

カラー・パターンと定義データ



３.２.１４　塗りつぶしパターン設定

［シーケンス］

```
int     id ;
gSetFillPtn (id) ;
no error status
```

［入　　力］

id　．．　パターン番号

［機　　能］

●指定された番号のパターンを塗りつぶしのパターンとします。

●パターン番号の範囲は０から47までです。

保守／廃止

３.２.１５　グラフィクス・ペン・パターン設定

［シーケンス］

unsigned short ＊pattern；

gSetGpenPtn（pattern）；

no error status；

［入　　力］

pattern　．．　グラフィクス・ペン・パターン定義データ

［機　　能］

●グラフィクス・ペン・パターンを設定します。通常は，あわせてグラフィクス・ペン・マスクの設定も行います。

●ペン・パターンの大きさは16×16ドット固定です。データは，unsigned shortの配列として定義します（要素数16）。

保守／廃止

３.２.１６　グラフィクス・ペン・マスク設定

［シーケンス］

unsigned short ＊pattern；

gSetGpenMask（pattern）；

no error status

［入　　力］

pattern　．．　グラフィクス・ペン・マスク定義データ

［機　　能］

●グラフィクス・ペン・マスクを設定します。通常は，あわせてグラフィクス・ペン・パターンの設定も行います。

●ペン・マスクの大きさは16×16ドット固定です。データはunsigned shortの配列として定義します（要素数16）。

保守／廃止

３.２.１７　文字描画サイズ設定

［シーケンス］

```
int     w, h
gSetCharSize (w, h) ;
no error status
```

［入　　力］

w　．．　文字の幅（ドット数）

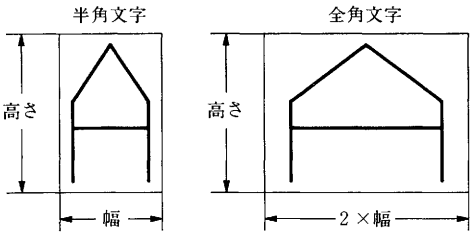
h　．．　文字の高さ（ドット数）

［機　　能］

●文字，または文字列を描画する際の文字の大きさを設定します。

●文字が回転しない状態での幅と高さを指定します。回転した場合は見かけの大きさが同じになるように変形されます。したがって，実際に文字を構成するドット数はここで指定した値とは違ってきます。

●幅は，半角文字を描画するときの大きさを指定します。全角文字は，この倍の値をとります。



保守／廃止

３.２.１８　文字回転角の設定

［シーケンス］

```
int     h_dx, h_dy ;
int     v_dx, v_dy ;
gSetCharAngle (h_dx, h_dy, v_dx, v_dy) ;
error     関数値！＝0
```

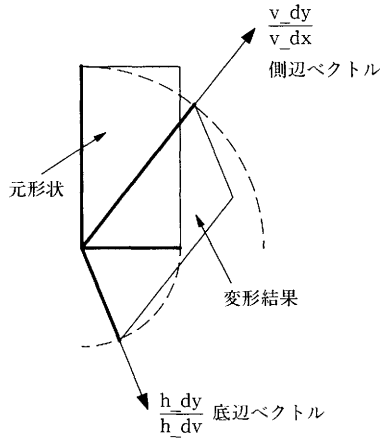
［入　　力］

(h_dx, h_dy)　　．．　底辺回転ベクトル

(v_dx, v_dy)　　．．　側辺回転ベクトル

［機　　能］

- 文字の回転具合を設定します。
- 回転は文字の底辺と側辺について個別に指定します（文字をデザインする空間を表す長方形の下辺を「底辺」，左辺を「側辺」とします)。
- 回転ベクトルは文字のデザイン空間の左下隅を原点とした座標系で定義します。
- 回転ベクトルの長さを０にした場合は，エラーになります。エラーになった場合は，それまでの回転角指定は変化しません。
- ベクトルの大きさは回転状況に関係しません。
  たとえば，ベクトル（1，0）と（100，0）とは同じ回転を意味します。
- コピーにおける回転角とは独立しています。



保守／廃止

３.２.１９　文字列配置の設定

［シーケンス］

```
int     vx, vy ;
int     displacement ;
int     full_adjust ;
gSetTextAlignment (x, y, displacement, full_adjust) ;
no error status
```

［入　　力］

(vx, vy)　　. .　アライメント・ベクトル

displacement　. .　隣接文字間隔（ドット）

full_adjust　全／半文字間隔調整指定

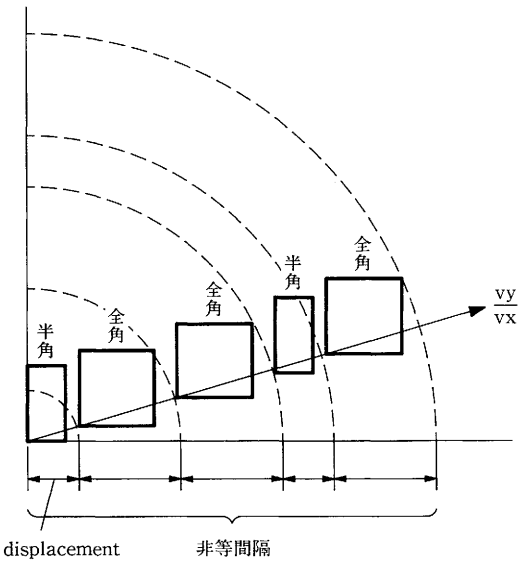
［機　　能］

- 文字列を描画するときの文字の配置を決定します。
- アライメント・ベクトルは，文字列を構成する文字の配置を決定するもので，文字列描画関数gPutText（ ）で指定された描画位置を起点とします。文字列を構成する文字は，デザイン空間の左下隅がアライメント・ベクトル上に位置するように配置されます。
- アライメント・ベクトルがどのように設定されても文字の回転角度はgSetCharAngle（ ）で指定された値をとります（アライメント・ベクトルとは何の関係も持ちません）。
- 隣接文字間隔は，指定された文字列中の隣接する文字の間隔を決定します。直前に描画された文字から次に描画される文字までの距離を定義したもので，アライメント・ベクトルの向きに関係なく一定です。半角文字を描画したときの値を設定します。
- 全／半文字間隔調整指定は０か０以外の値で示されるフラグです。０が指定された場合，隣接文字間隔は文字の全角／半角に関係なく指定値をとります。０以外の値が指定された場合は，全角文字を描画したあとの文字間隔は指定値の２倍の値をとります。

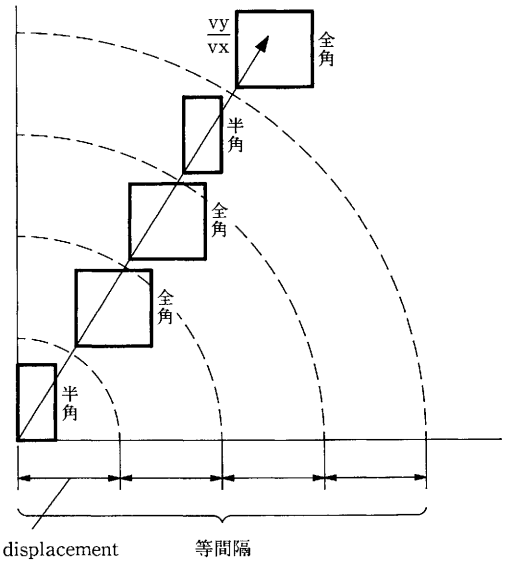
  通常アライメント・ベクトルが，x軸に近い傾きをとっている場合は０以外の値を指定し，y軸に近い傾きをとっている場合は０を指定します。ただし，描画目的によってはこの限りではありません。
- 文字はアライメント・ベクトルの範囲を越えて位置することはできません。文字の描画位置がアライメント・ベクトルの範囲を越えた場合，それ以降の文字は描画されません。

保守／廃止

全／半文字間隔調整指定≠0



全／半文字間隔調整指定＝0



保守／廃止

3.2.20　コピー回転角の設定

[シーケンス]

```
int     h_dx, h_dy ;
int     v_dx, v_dy ;
gSetCopyAngle (h_dx, h_dy, v_dx, v_dy) ;
error     関数値！＝0
```

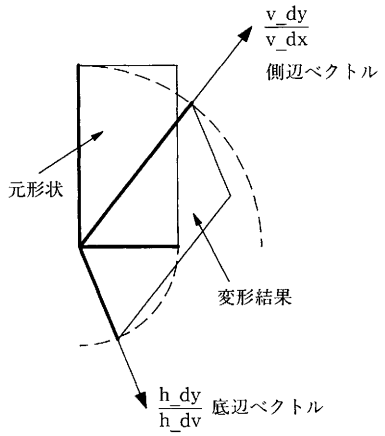
[入　　力]

(h_dx, h_dy)　　. .　　底辺回転ベクトル

(v_dx, v_dy)　　. .　　側辺回転ベクトル

[機　　能]

- コピーする場合の転送源範囲の転送先での回転具合を設定します。
- 回転は回転源の底辺と側辺について個別に指定します（転送源範囲を表す長方形の下辺を「底辺」，左辺を「側辺」とします)。
- 回転ベクトルは，転送源範囲の左下隅を原点とした座標系で定義します。
- 回転ベクトルの長さを 0 にした場合はエラーになります。エラーになった場合はそれまでの回転角指定は変化しません。
- ベクトルの大きさは回転状況に関係しません。たとえば，ベクトル（1，0）と（100，0）とは同じ回転を意味します。
- 文字の回転角指定とは独立しています。



保守／廃止

３.２.２１　画面消去

［シーケンス］

gClearScreen（）；

no error status

［機　　能］

●設定されている背景カラー・コードで画面をクリアします。

●クリップ・エリアが設定されている場合は，クリップ・エリア内部をクリアします（クリップ・エリアを定義したときに，指定された２点はクリアされる範囲に含まれます）。

●描画モードは結果に関与しません。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.２２　１ドットの点の描画

［シーケンス］

int　　x, y ;

gDot (x, y) ;

no error status

［入　　力］

(x, y)　　. .　　描画位置

［機　　能］

●指定位置にドットを描画します。

●描画結果は設定されている描画カラー・コードと描画モードに従います。

●指定位置がクリップ・エリア外部の場合は何も描画されません。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.２３　グラフィクス・ペンによる点の描画

［シーケンス］

int　　x，y；

gGpDot（x，y）；

no error status

［入　　力］

（x，y）　　．．　　描画位置

［機　　能］

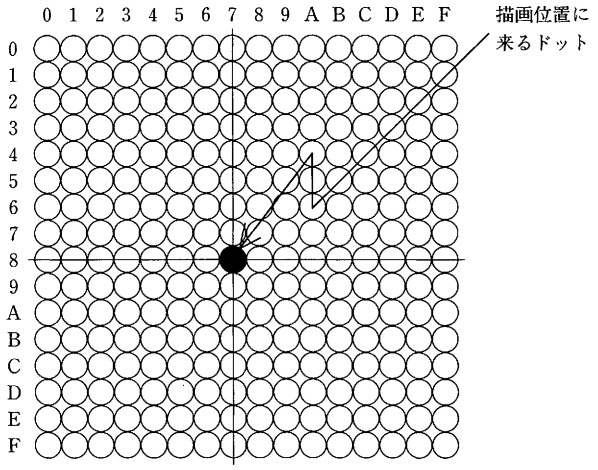
●指定位置にグラフィクス・ペンによるドットを描画します（グラフィクス・ペン・パターン，およびグラフィクス・ペン・マスクを描画します）。

●描画結果は設定されている描画カラー・コードと描画モードに従います。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

ペンおよびマスクと描画位置の関係



保守／廃止

3.2.24　１ドット幅の直線

［シーケンス］

int　　x0, y0, x1, y1 ;

gLine (x0, y0, x1, y1) ;

no error status

［入　　力］

(x0, y0)　　. .　　始点位置

(x1, y1)　　. .　　終点位置

［機　　能］

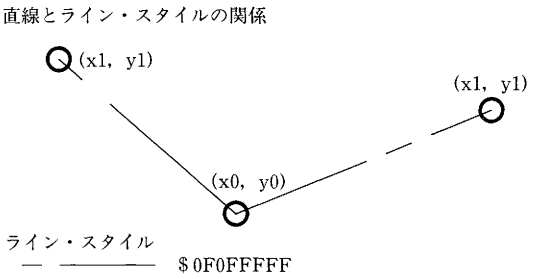
●指定された始点と終点を結ぶ直線を描画します。

●描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。

●連続して描画を行っても，ライン・スタイルの継続は行われません。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。



保守／廃止

3.2.25　グラフィクス・ペンによる直線

［シーケンス］

```
int     x0, y0, x1, y1, ;
gGpLine (x0, y0, x1, y1) ;
no error status
```

［入　　力］

(x0, y0)　　. .　　始点位置

(x1, y1)　　. .　　終点位置

［機　　能］

- 指定された始点と終点を結ぶ直線をグラフィクス・ペンを使って描画します。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 連続して描画を行ってもライン・スタイルの継続は行われません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.26　１ドット幅の三角形の描画

［シーケンス］

```
struct {
        int     px ;
        int     py ;
}       vertex [3] ;
gTriangle (vertex) ;
no error status
```

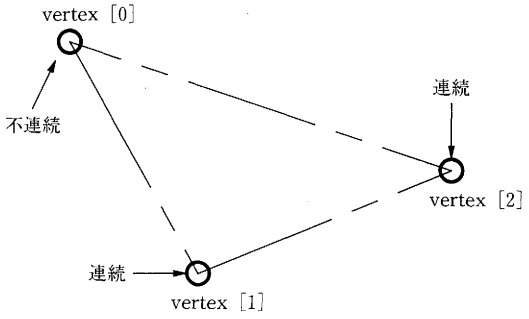
［入　　力］

vertex　．．　三角形の頂点座標

［機　　能］

- 指定された３頂点によって定義される三角形を描画します。
- vertex［0］が描画開始位置になります（終了位置でもあります）。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- ３辺ライン・スタイルは描画開始位置から終了位置までの間で連続していますが，前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

直線とライン・スタイルの関係



ライン・スタイル

— ———— $0F0FFFFF

保守／廃止

3.2.27　グラフィクス・ペンによる三角形の描画

[シーケンス]

```
struct {
        int     px;
        int     py;
}       vertex [3];
gGpTriangle (vertex);
no error status
```

[入　　力]

vertex　．．　三角形の頂点座標

[機　　能]

- 指定された3頂点によって定義される三角形をグラフィクス・ペンを使って描画します。
- vertex [0] が描画開始位置になります（終了位置でもあります）。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 3辺のライン・スタイルに連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.２８　１ドット幅の多角形の描画

［シーケンス］

```
struct {
        int     px ;
        int     py ;
}       vertex [n] ;
int     n ;
gPolygon (n, vertex) ;
no error status
```

［入　　力］

n　　．．　多角形を構成する頂点の数

vertex　．．　頂点座標

［機　　能］

- 指定された頂点によって定義される多角形を描画します。
- vertex［0］が描画開始位置になります（終了位置でもあります)。
- 描画結果は,設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 各辺のライン・スタイルは描画開始位置から終了位置までの間で連続していますが，前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.29　グラフィクス・ペンによる多角形の描画

[シーケンス]

```
struct {
        int     px;
        int     py;
}       vertex [n];
int     n;
gGpPolygon (n, vertex);
no error status
```

[入　　力]

n　　．．　多角形を構成する頂点の数

vertex　．．　頂点座標

[機　　能]

- 指定された頂点によって定義される多角形をグラフィクス・ペンを使って描画します。
- vertex [0] が描画開始位置になります（終了位置でもあります）。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 各辺のライン・スタイルに連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.３０　１ドット幅のボックスの描画

[シーケンス]

int　　x0, y0, x1, y1 ;

gRectangle (x0, y0, x1, y1) ;

no error status

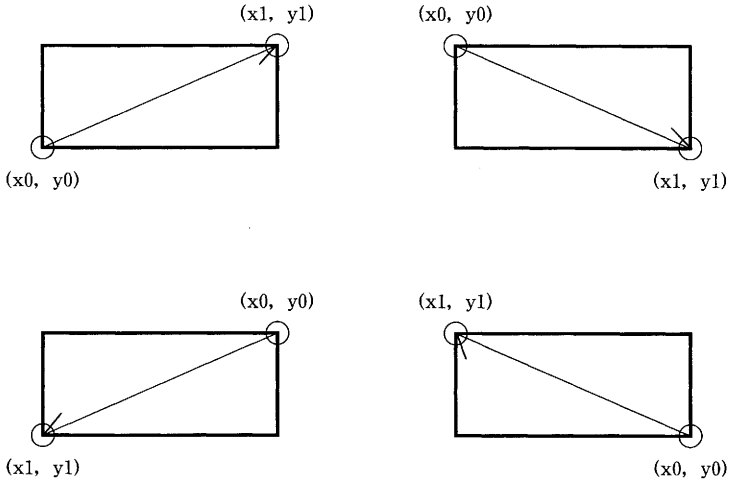
[入　　力]

(x0, y0) − (x1, y1) ;　．．　　ボックスの対角線位置

[機　　能]

- 指定された対角線を持つボックスを描画します。
- 描画結果は設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 各辺のライン・スタイルは，描画開始位置から終了位置までの間で連続していますが，前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

ボックスの定義



保守／廃止

3.2.31　グラフィクス・ペンによるボックスの描画

［シーケンス］

```
int     x0, y0, x1, y1 ;
gGpRectangle (x0, y0, x1, y1) ;
no error status
```

［入　　力］

(x0, y0) − (x1, y1) ;　．．　　ボックスの対角線位置

［機　　能］

- 指定された対角線を持つボックスをグラフィクス・ペンを使って描画します。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 各辺のライン・スタイルの連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.３２　１ドット幅の円の描画

［シーケンス］

int　　x, y, r ;

gCircle (x, y, r) ;

no error status

［入　　力］

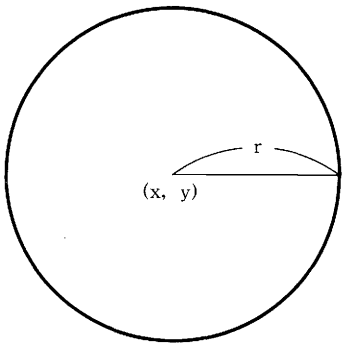
(x, y)　. .　中心位置

r　　. .　半径

［機　　能］

- 指定された中心と半径を持つ円を描画します。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

円の定義



保守／廃止

３.２.３３　グラフィクス・ペンによる円の描画

［シーケンス］

```
int     x, y, r ;
gGpCircle (x, y, r) ;
no error status
```

［入　　力］

(x, y)　．．　中心位置

r　　．．　半径

［機　　能］

- 指定された中心と半径を持つ円をグラフィクス・ペンを使って描画します。
- 描画結果は設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.３４　１ドット幅の円弧の描画

［シーケンス］

```
int     x, y, r ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gCircArc (x, y, r, xs, ys, xe, ye, dir, type) ;
no error status
```

［入　　力］

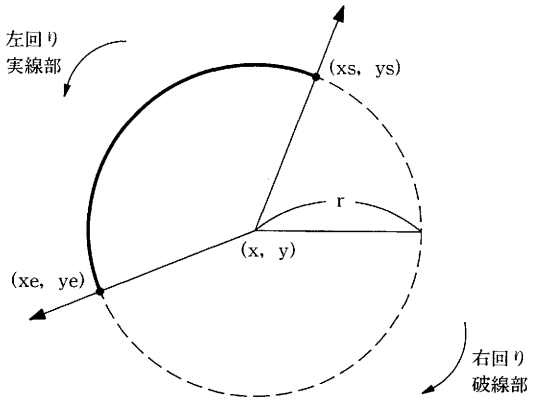
(x, y)	. .	中心位置
r	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝弧
		1＝扇
		2＝弦

［機　　能］

- 指定された中心と半径を持つ円に属する弧／扇／弦を描画します。
- 描画開始位置と終了位置は，おのおののベクトルの円との交点位置になります。
- 弧／扇／弦は描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- 弧／扇／弦のいずれを描画するかは弧の種類によって決定されます。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

円弧の定義



保守／廃止

3.2.35　グラフィクス・ペンによる円弧の描画

[シーケンス]

```
int     x, y, r ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gGpCircArc (x, y, r, xs, ys, xe, ye, dir, type) ;
no error status
```

[入　　力]

(x, y)	. .	中心位置
r	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝弧
		1＝扇
		2＝弦

[機　　能]

- 指定された中心と半径を持つ円に属する弧／扇／弦をグラフィクス・ペンを使って描画します。
- 描画開始位置と終了位置は，おのおののベクトルのもととなる円との交点位置になります。
- 弧／扇／弦は，描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- 弧／扇／弦のいずれを描画するかは弧の種類によって決定されます。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.36　１ドット幅の楕円の描画

[シーケンス]

```
int     x, y, ;
int     rx, ry ;
gEllipse (x, y, rx, ry) ;
no error status
```

[入　　力]

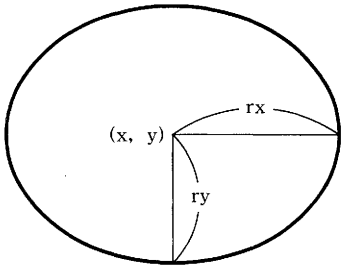
(x, y)　　. .　　中心位置

rx, ry　　. .　　半径

[機　　能]

- 指定された中心と半径を持つ楕円を描画します。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

楕円の定義



保守／廃止

3.2.37　グラフィクス・ペンによる楕円の描画

[シーケンス]

```
int     x, y, ;
int     rx, ry ;
gGpEllipse (x, y, rx, ry) ;
no error status
```

[入　　力]

(x, y)　　. .　　中心位置

rx, ry　　. .　　半径

[機　　能]

- 指定された中心と半径を持つ楕円をグラフィクス・ペンを使って描画します。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.３８　１ドット幅の楕円弧の描画

［シーケンス］

```
int     x, y, ;
int     rx, ry ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gElpsArc (x, y, rx, ry, xs, ys, xe, ye, dir, type) ;
no error status
```

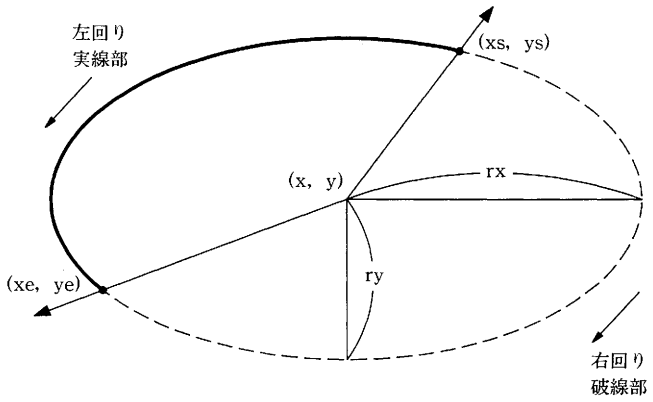
［入　　力］

(x, y)	. .	中心位置
rx, ry	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝弧
		1＝扇
		2＝弦

［機　　能］

- ●指定された中心と半径を持つ円に属する弧／扇／弦を描画します。
- ●描画開始位置と終了位置はおのおののベクトルのもととなる楕円との交点位置になります。
- ●弧／扇／弦は描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- ●弧／扇／弦のいずれを描画するかは弧の種類によって決定されます。
- ●描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- ●前後に描画される線形図形との連続性はありません。
- ●クリップ・エリアによるクリップが行われます。
- ●描画対象画面の描画対象プレーンに対して処理を行います。

楕円弧の定義



保守／廃止

３.２.３９　グラフィクス・ペンによる楕円弧の描画

［シーケンス］

```
int     x, y ;
int     rx, ry ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gGpElpsArc (x, y, rx, ry, xs, ys, xe, ye, dir, type) ;
no error status
```

［入　　力］

(x, y,)	. .	中心位置
rx, ry	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝弧
		1＝扇
		2＝弦

［機　　能］

- 指定された中心と半径を持つ楕円に属する弧／扇／弦をグラフィクス・ペンを使って描画します。
- 描画開始位置と終了位置は，おのおののベクトルのもととなる円との交点位置になります。
- 弧／扇／弦は描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- 弧／扇／弦のいずれを描画するかは，弧の種類によって決定されます。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- 前後に描画される線形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.40　塗りつぶし

［シーケンス］

```
int     x, y ;
int     border ;
gPaint (x, y, border) ;
no error status
```

［入　　力］

(x, y)　．．　塗りつぶし位置

border　．．　境界色コード

［機　　能］

●指定された位置を含む閉領域を塗りつぶします。

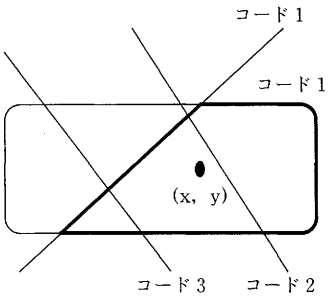
●閉領域は指定境界色コードを境界として定義します。

●描画結果は，設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。

●クリップ・エリアを越えては塗りつぶしは行われません。
クリップ・エリアは境界として扱われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

閉領域定義



コード１を境界とすれば閉領域は太い線で示された範囲になる

保守／廃止

3.2.41　塗り直し

[シーケンス]

int　　x, y;

gChangeColor (x, y);

no error status

[入　　力]

(x, y)　. . 塗りつぶし位置

[機　　能]

●指定された位置を含む閉領域を塗りつぶします。

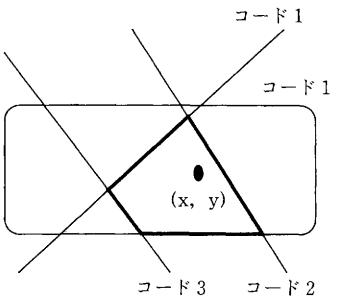
●閉領域は指定位置にあるカラー・コード以外のカラー・コードを境界として定義されます。

●描画結果は設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。

●クリップ・エリアを越えて塗りつぶしは行われません。
クリップ・エリアは境界として扱われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

閉領域定義



閉領域は太い線で示された範囲になる

保守／廃止

３.２.４２　三角形の塗りつぶし

［シーケンス］

```
struct {
        int     px ;
        int     py ;
}       vertex [3] ;
gFillTriangle (vertex) ;
no error status
```

［入　　力］

vertex　．．三角形の頂点座標

［機　　能］

- 指定された３頂点によって定義される三角形を塗りつぶします。
- 描画結果は設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.43　ボックスの塗りつぶし

［シーケンス］

```
int     x0, y0, x1, y1 ;
gFillRectangle (x0, y0, x1, y1) ;
no error status
```

［入　　力］

(x0, y0) − (x1, y1)　．．　ボックスの対角線位置

［機　　能］

- 指定された対角線を持つボックスを塗りつぶします。
- 描画結果は設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.44　多角形の塗りつぶし

[シーケンス]

```
struct {
        int     px;
        int     py;
}       vertex [n];
int     n;
gFillPolygon (n, vertex);
no error status
```

[入　　力]

n　　．．　多角形を構成する頂点の数

vertex　．．　頂点座標

[機　　能]

- 指定された頂点によって定義される多角形を塗りつぶします。
- 描画結果は設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

３.２.45　円の塗りつぶし

［シーケンス］

```
int     x, y, r ;
gFillCircle (x, y, r) ;
no error status
```

［入　　力］

(x, y)　．．　中心位置

r　　．．　半径

［機　　能］

●指定された中心と半径を持つ円を塗りつぶします。

●描画結果は設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.46　円弧の塗りつぶし

[シーケンス]

```
int     x, y, r ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gFillCircArc (x, y, r, xs, ys, xe, ye, dir, type) ;
no error status
```

[入　　力]

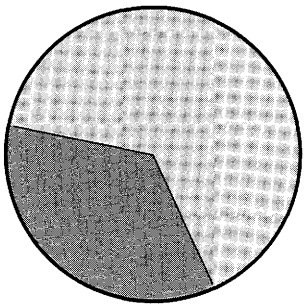
(x, y)	. .	中心位置
r	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝扇
		1＝扇
		2＝弦

[機　　能]

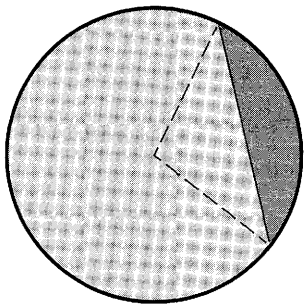
- 指定された中心と半径を持つ円に属する扇／弦を描画します。
- 描画開始位置と終了位置は，おのおののベクトルのもととなる円との交点位置になります。
- 扇／弦は描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- 扇／弦のいずれを描画するかは弧の種類によって決定されます。
- 描画結果は，設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。
- 前後に描画される扇形図形との連続性はありません。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

扇と弦

扇形



弦形



保守／廃止

３.２.47　楕円の塗りつぶし

［シーケンス］

int　　x, y;

gFillEllipse (x, y, rx, ry);

no error status

［入　　力］

(x, y)　. .　　中心位置

rx, ry　. .　　半径

［機　　能］

●指定された中心と半径を持つ楕円を塗りつぶします。

●描画結果は，設定されているフィル・パターンと描画カラー・コード，および描画モードに従います。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.48　楕円弧の塗りつぶし

［シーケンス］

```
int     x, y ;
int     rx, ry ;
int     xs, ys ;
int     xe, ye ;
int     dir, type ;
gElpsArc (x, y, rx, ry, xs, ys, xe, ye, dir, type) ;
no error status
```

［入　　力］

(x, y,)	. .	中心位置
rx, ry	. .	半径
(xs, ys)	. .	描画開始位置ベクトル
(xe, ye)	. .	描画終了位置ベクトル
dir	. .	弧の定義方向
		0＝左回り
		1＝右回り
type	. .	弧の種類
		0＝楕扇
		1＝楕扇
		2＝弦

［機　　能］

- 指定された中心と半径を持つ楕円に属する扇／弦を塗りつぶします。
- 描画開始位置と終了位置は，おのおののベクトルのもととなる円との交点位置になります。
- 扇／弦は描画開始位置から指定方向回りに描画終了位置までの範囲をとります。
- 扇／弦のいずれを描画するかは，弧の種類によって決定されます。
- 描画結果は，設定されているライン・スタイルと描画カラー・コード，および描画モードに従います。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

3.2.49　３オペランド・コピー

[シーケンス]

```
int     sscr ;
int     sx, sy ;
int     mscr ;
int     mx, my ;
int     dx, dy ;
int     xc, yc ;
int     type, p ;
g3opCopy (sscr, sx, sy, mscr, mx, my, dx, dy, xc, yc, type, p) ;
no error status
```

[入　　力]

sscr	. .	転送源画面番号
(sx, sy)	. .	転送領域位置（左下隅）
mscr	. .	マスク画面番号
(mx, my)	. .	マスク領域位置（左下隅）
(dx, dy)	. .	転送先領域位置（左下隅）
(xc, yc)	. .	転送領域の大きさ（ドット数）
type	. .	マクス・ビット値 (0or1)
p	. .	プレーン転送形式 負＝Multi→Multi 正＝Single→Multi（プレーン番号）

[機　　能]

- 指定された画面を転送源とマスクにして，３オペランド･コピーを行います。コピー先は，現在の描画対象画面です。
- マスクに使用される画面と転送源の画面は同じ大きさにしてください。
- マスク領域位置は，ワード境界を指定してください。
- マスク画面のプレーン０がマスクとして使用されます（他のプレーン内容は，コピーの実行に関与しません)。
- マスク・ビット値が０の場合は，マスクのビット値が０の部分に対応する転送源領域がコピーされます。１の場合は，ビット値が１の部分に対応する転送源領域がコピーされます。
- プレーン転送形式としてMulti→Multiを使用する場合は，転送源画面の描画対象プレーン数を転送先画面の描画対象プレーン数以上にしてください。
- コピー回転角指定は無視されます（等倍矩形コピーのみ行います)。

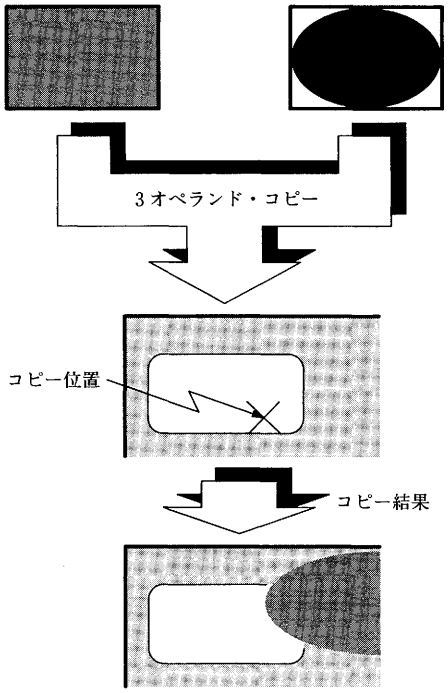
保守／廃止

●設定されている描画カラーと描画モードに従って，演算が行われます。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

マクスの効果



保守／廃止

3.2.50　画面コピー

[シーケンス]

```
int     sscr ;
int     sx, sy ;
int     sw, sh
int     dx, dy ;
int     dw, dh ;
int     p ;
gCopyScr (sscr, sx, sy, sw, sh, dx, dy, dw, dh, p) ;
no error status
```

[入　　力]

sscr	. .	転送源画面番号
(sx, sy)	. .	転送領域位置（左下隅）
(sw, sh)	. .	転送源領域の大きさ（ドット数）
(dx, dy)	. .	転送先領域位置（左下隅）
(dw, dh)	. .	転送先領域の大きさ（ドット数）
p	. .	プレーン転送形式 負＝Multi→Multi 正＝Single→Multi（プレーン番号）

[機　　能]

●指定された転送源範囲をコピーします。

転送先は画面および転送範囲ともに転送源と異なるものを指定できます。転送先範囲が転送源範囲と異なる大きさの場合は，拡大／縮小が行われます。

また，コピー回転角指定に従って回転が行われます。

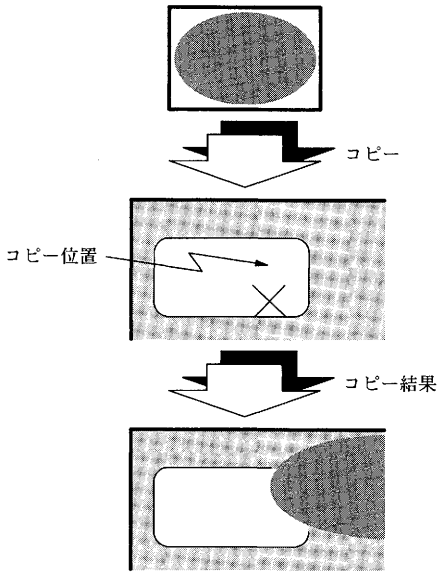
●設定されている描画カラーと描画モードに従って演算が行われます。

●プレーン転送形式として，Multi→Multiを使用する場合は，転送源画面の描画対象プレーン数を転送先画面の描画対象プレーン数以上にしてください。

●クリップ・エリアによるクリップが行われます。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止



保守／廃止

３.２.５１　画面情報の読み込み

［シーケンス］

```
int              x, y ;
int              xc, yc ;
unsigned short   *buf ;
gGetImage (x, y, xc, yc, buf) ;
no error status
```

［入　　力］

(x, y)　．．　読み込み位置（読み込み範囲の左下隅）

(xc, yc)　．．　読み込み範囲の大きさ（ドット数）

［出　　力］

buf　．．　読み込み結果

［機　　能］

●指定された範囲の内容を指定されたバッファに読み込みます。

●バッファに読み込まれる結果は次の構造をとります。

unsigned short buf［yc］［p］［(xc+15)／16］;

pは読み込みを行った画面の描画対象プレーン数です。

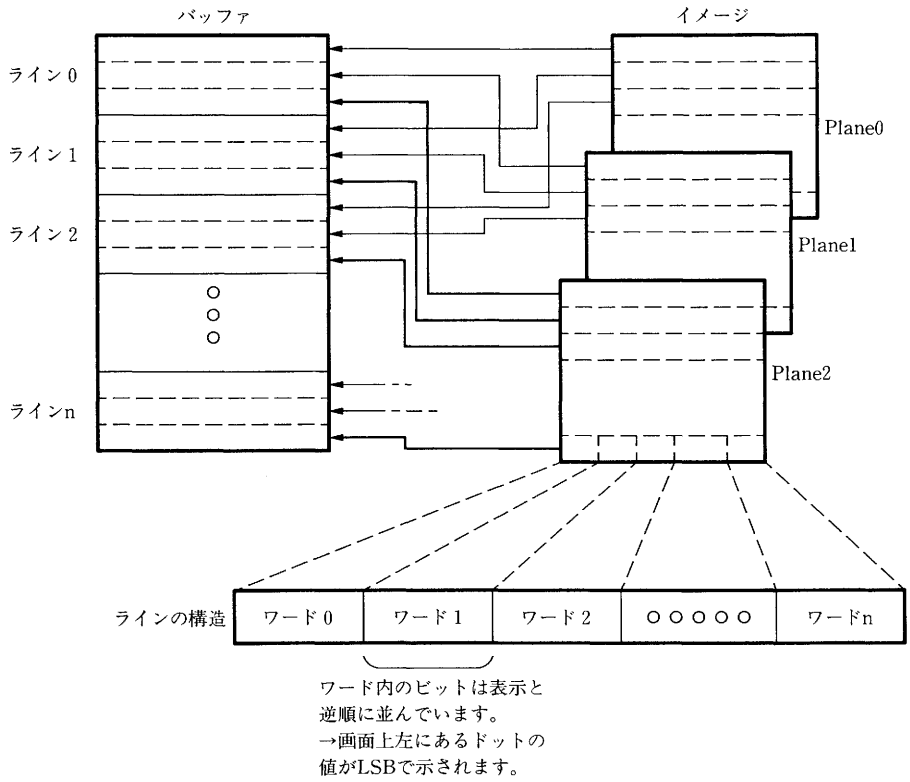
●読み込み範囲がクリップ・エリアを越えないように指定してください。

●描画カラーや描画モードは処理に関与しません。

●描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

イメージとバッファの関係



保守／廃止

３.２.５２　画面情報の書き込み

［シーケンス］

```
int              x, y ;
int              xc, yc ;
unsigned short   *buf ;
int              pitch ;
unsigned long    disp ;
int              p ;
gPutImage (x, y, xc, yc, buf, pitch, disp, p) ;
no error status
```

［入　　力］

(x, y)	. .	書き込み位置（書き込み範囲の左下隅）
(xc, yc)	. .	書き込み範囲の大きさ（ドット数）
buf	. .	書き込みデータ
pitch	. .	書き込みデータのピッチ（ワード数）
disp	. .	書き込みデータのプレーン間隔（ワード数）
p	. .	転送形式 負＝Multi→Multi 正＝Single→Multi（プレーン番号）

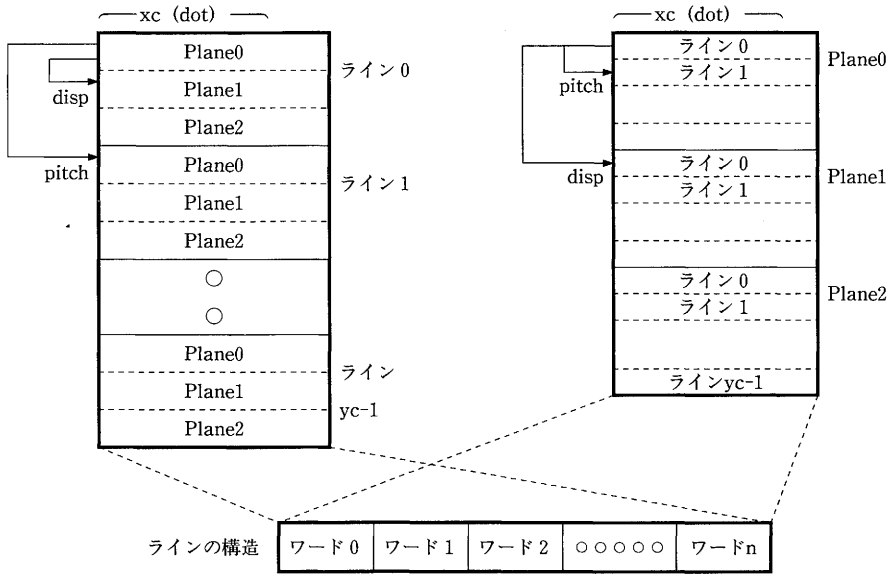
［機　　能］

- 与えられた書き込みデータを，画面上の指定範囲に転送します。
- 書き込むデータの並びに対応して，ピッチとプレーン間隔が指定できます。
- 転送時に回転や拡大／縮小は行われません。
- 設定されている描画カラーと描画モードに従って演算が行われます。
- プレーン転送形式としてMulti→Multiを使用する場合は，転送源画面の描画対象プレーン数を転送先画面の描画対象プレーン数以上にしてください。
- クリップ・エリアによるクリップが行われます。
- 描画対象画面の描画対象プレーンに対して処理を行います。

保守／廃止

バッファの代表的な展開例

ライン展開　　プレーン展開



備考　ライン展開／プレーン展開のどちらのデータ構造でも可能です。

保守／廃止

3.2.53　文字の描画

［シーケンス］

```
int     x, y ;
unsigned short code ;
gPutChar (x, y, code) ;
no error status
```

［入　　力］

(x, y)　. .　描画位置

code　. .　描画する文字

［機　　能］

●指定位置に指定された文字（JISコード）を描画します。
文字の回転，およびサイズ指定に従って回転と拡大／縮小が行われます。

● 2バイト（全角）文字は上位バイトに区コード（0x21..），下位バイトに点コード（0x21..）を設定します。1バイト（半角）文字は上位バイトに0x00を設定し，下位バイトに文字コード（0x00.. 0xff）を設定します。

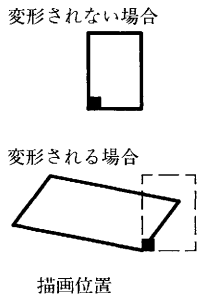
●描画位置には，文字のデザイン範囲の左下隅がきます。

● 8区から5区までの文字はサポートされていません（与えられたのと異なる文字が描画されることがあります）。

●設定されている描画カラーと描画モードに従って演算が行われます。

●クリップ・エリアによるクリップが行われます。

描画位置と文字の関係



保守／廃止

3.2.54　文字列の描画

［シーケンス］

```
int     x, y ;
unsigned char *text ;
gPutText (x, y, text) ;
no error status
```

［入　　力］

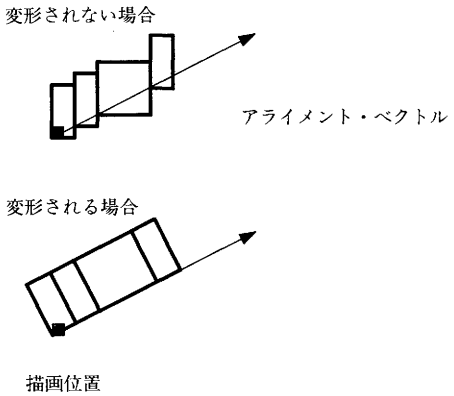
(x, y)　. .　描画位置

text　. .　描画するテキスト

［機　　能］

- 指定位置より，すでに指定されているテキスト（シフトJISコード）の配置と文字回転，およびサイズ指定に従って文字列を描画します。
- **描画位置には，テキストの先頭文字の左下隅がきます。**
- ８区から５区までの文字はサポートされていません（与えられたのと異なる文字が描画されることがあります）。
- 設定されている描画カラーと描画モードに従って演算が行われます。
- クリップ・エリアによるクリップが行われます。
- 描画位置がテキストのアライメント・ベクトルからはみ出る場合は，それ以降の文字は描画されません。

描画位置と文字列の関係



保守／廃止

3.2.55　表示プレーン指定

［シーケンス］

unsigned short plane_mask；

gDispPlane（plane_mask）；

no error status

［入　　力］

plane_mask　．．　表示プレーン・マスク

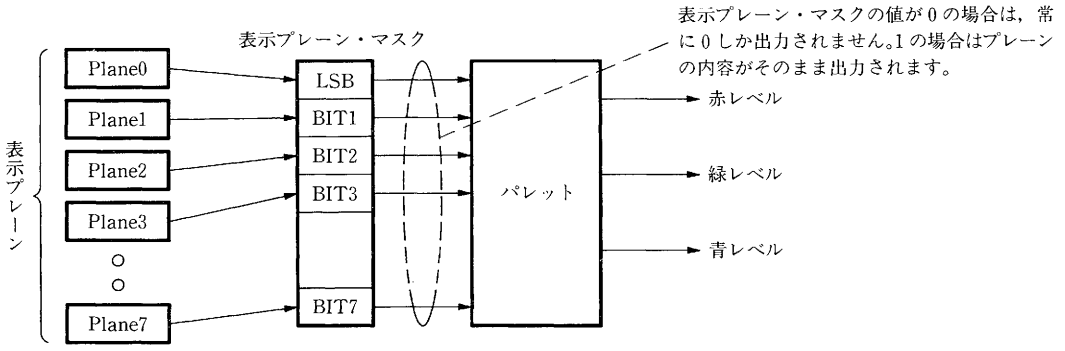
［機　　能］

●表示プレーン・マスクで表示許可されたプレーンを表示させます。

プレーン・マスクは，ビットごとに表示画面を構成するプレーンに対応します。LSBがプレーン０に対応し，以降ビット１がプレーン１，ビット２がプレーン２と対応していきます（このプレーンの番号は物理的な番号です：描画対象プレーンの設定状態とは無関係に，画面を構成する最初のプレーンがプレーン０になります)。ビットの値が１の場合が表示許可になります。

●すべてのプレーンを表示の対象にする場合は０を設定します。

表示プレーン・マスクと表示の関係



保守／廃止

3.2.56　色情報読み出し

［シーケンス］

```
int     gGetColor ();
int     x, y;
int     code;
code=gGetColor (x, y);
```

［入　　力］

(x, y) . .　　読み出すドット位置

［出　　力］

code　. .　　ドット・カラー・コード

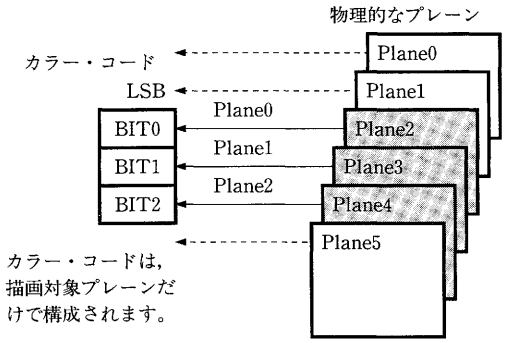
［機　　能］

●指定された位置にあるドットのカラー・コードを与えます。

●描画対象プレーンが限定されている場合は，限定された範囲内でのローカルなカラー・コードを作成します。たとえば，第２プレーンから３プレーンが描画対象になっている場合，与えられるカラー・コードは０から７までの範囲になります。

●クリップ・エリア外を指定してはいけません。

描画対象プレーンとカラー・コード



※６枚のプレーンで構成される画面のうち，プレーン２(物理番号)から３枚のプレーンを描画対象にした場合の例

保守／廃止

３.２.５７　最後に描いた弧の範囲

［シーケンス］

```
int     x0, y0, x1, y1;
gGetTip (&x0, &y0, &x1, &y1);
no error status
```

［出　　力］

(x0, y0)　．．　描画開始位置

(x1, y1)　．．　描画終了位置

［機　　能］

●最後に実行された弧描画機能の描画開始位置と描画終了位置を与えます。

●対象となる描画機能は以下のとおりです。

gCircArc ( ), gGpCircArc ( ), gFillCircArc ( ),

gElpsArc ( ), gGpElpsArc ( ), gFillElpsArc ( )

上記関数を実行しないでこの関数を実行した場合，与えられる結果は意味を持ちません。

保守／廃止

3.2.58　色情報の検索

［シーケンス］

```
int     x, y, dir ;
int     col, dat ;
int     dx, dy ;
gBitSearch (x, y, dir, col, dat, &dx, &dy)
;
error   関数値！＝0
        最初から条件が満たされています。
```

［入　　力］

(x, y)　．．　検索開始位置

dir　．．　検索方向

0＝左方向

1＝右方向

col　．．　検索する色

dat　．．　検索方法

0＝検索する色が途切れる位置を探します。

1＝検索する色が現れる位置を探します。

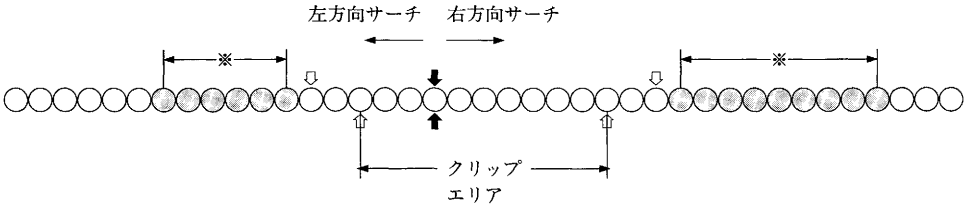
［出　　力］

(dx, dy)　．．　検索された位置

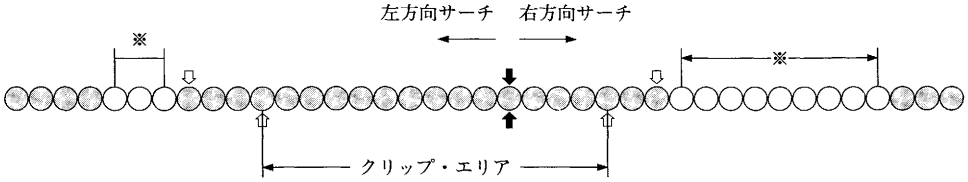
［機　　能］

- 指定位置より指定方向に条件を満たすドットを探し，検索方向に対して一つ手前のドット位置を与えます。
- 検索開始位置がすでに条件を満たしている場合はエラーになります。
- 検索中にクリップ・エリアを出る場合は，クリップ・エリアの境界位置が結果として与えられます。
- クリップ・エリア外を検索開始位置として指定してはいけません。

検索する色が現れる位置を捜す場合の例



検索する色が途切れる位置を捜す場合の例



※の範囲に検索開始位置を置くとエラーになります。

◎　検索する色のドット
○　検索する色以外の色のドット
⬇　検索開始位置
⇩　検索終了位置

備考　検索途中でクリップ・エリアの境界になった場合，その位置が検索結果となります。

保守／廃止

3.2.59　表示メモリの読み出し

［シーケンス］

```
unsigned short gPeek ( ) ;
unsigned long addr ;
unsigned short data ;
data＝gPeek (addr) ;
```

［入　　力］

addr　．．　読み出すメモリのアドレス
　　バイト・アドレス（偶数のみ）

［出　　力］

data　．．　読み出されたデータ

［機　　能］

●グラフィック・メモリ上の指定アドレスのデータを与えます。
●アドレスはバイト・アドレスですが，偶数のみを指定してください。
●アドレスは0から始まります。

保守／廃止

３.２.６０　表示メモリへの書き込み

［シーケンス］

```
unsigned long addr ;
unsigned short data ;
gPoke (addr, data) ;
```

［入　　力］

addr　．．　書き込むメモリのアドレス
　　バイト・アドレス（偶数のみ）
data　．．　書き込むデータ

［機　　能］

●グラフィック・メモリ上の指定アドレスにワード・データを書き込みます。
●アドレスはバイト・アドレスですが，偶数のみを指定してください。
●アドレスは０から始まります。
●無意味に書き込みを行うと，システムの情報を破壊するので注意してください。

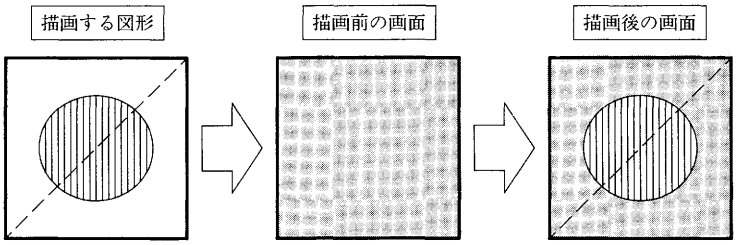
保守／廃止

３.３　演算効果

演算は16種類あり，描画コードのビット値が０のプレーンと１のプレーンとで異なる演算を指定できます。組み合わせで考えると256種類の演算が可能です。

３.３.１　Stencil演算

PSET（置き換え）演算による演算効果例



Stencil演算コード

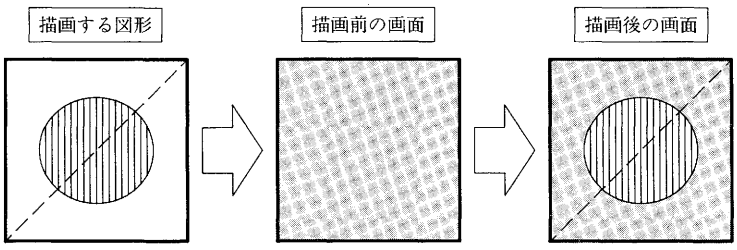
演算種	コード
PSET	0xC9
OR	0xC7
AND	0x79
XOR	0x47
NOT	0x9c

描画する図形が線形図形（[直線] のような塗りつぶさない図形）の場合は，実部の部分だけが所定の演算で描画されます。塗りつぶし図形の場合は，塗りつぶしパターンの１で定義された箇所だけが所定の演算で描画されます（線形図形の虚部や塗りつぶし図形における塗りつぶしパターンが０で定義された箇所が透明である扱いを受ける演算）。

図では，演算種がPSET（置き換え）の場合が示されています。他の演算種を用いた場合は，図形が描画された（透明として扱われない）部分の描画前の内容と描画使用としている内容との間で所定の演算が行われます。

３.３.２　Fill演算

PSET（置き換え）演算による演算効果例



Stencil演算コード

演算種	コード
PSET	0x02
OR	0xC7
AND	0x82
XOR	0x47
NOT	0x13

描画する図形が，線形図形（直線のような塗りつぶさない図形）の場合は，実部の部分が所定の演算で描画されます。虚部の部分は，カラー・コード０が無条件に設定されます。塗りつぶし図形の場合は，塗りつぶしパターンの１で定義された箇所が所定の演算で描画され，０で定義された部分は，そのまま０が設定される演算です。

図では，演算種がPSET（置き換え）の場合が示されています。他の演算種を用いた場合は，図形が描画された（０が無条件設定されない）部分で描画前の内容と描画しようとしている内容との間で所定の演算が行われます。

保守／廃止

第４章　コンパイルおよびリンク方法

４.１　開発環境

OS　　：MS-DOS V3.0以降
C　　：MS-C　　V4.0以降
その他：LINK.EXE，LIB.EXE

４.２　ライブラリ構成

AGDCデバイス・ドライバは複数のオブジェクト・ファイル（.OJB）から構成されます。リンク時の指定を簡略化するためには，これらのオブジェクトをライブラリにします。たとえば，ライブラリ名称を“HYPER.LIB”とすると，このライブラリには次のオブジェクトが含まれます。

gmain.obj，gsub.obj，gio.obj

これらのオブジェクト・ファイルはすべて同じモデル（スモール，ミディアム，ラージ）でコンパイルされていなければならないことに注意してください。

４.３　コンパイル方法

依存インクルード・ファイル	GIO.H，GIOSYS.H
付加スイッチ	/J,/A $1
	$1 はモデル・タイプです。
コンパイル・オプション	LARGE_DATA　　ラージ，ミディアム・モデルで指定します。
	LARGE_COCDE　　ラージ，コンパクト・モデルで指定します。
	実際に指定するときは-Dスイッチを頭につけてください。
オブジェクトとソースの対応	GMAIN.OBJ<---GMAIN.C
	GSUB.OBJ　<---GSUB.C
	GIO.OBJ　　<---GIO.C

保守／廃止

4.4　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合わせをしてください。
- LINK.EXEのスイッチとして／NOIを指定します。

4.5　そ の 他

ユーザ・プログラムをコンパイルする場合には以下の点に注意してください。

- GIO.H，GIOSYS.Hをユーザ・プログラムからインクルードして中の定義を使用しないでください。
- 代表的な演算と描画色コードの定義はファイル“AGDCCOL.H”，“AGDCOP.H”で行われていますが，この内容はユーザの目的にあわせて自由に編集してください。

第５章　移植方法とテスト・プログラム

５.１　移植の際の修正部分

ユーザにあった移植を行う場合には，次の箇所を変更してください。以下の説明は，行番号を含めて付録Aのリストに基づいています。

(1) GIOSYS.H；

20～115行　：I/Oアクセス，メモリ構成などに関する定義

(2) GMAIN.C ；

10～38行　：CRT同期信号設定値，表示制御など

146～161行 ：メモリ・セグメント設定

167～181行 ：画面分解能設定

備考　その他，PC-98に依存する部分があるかもしれません。

５.２　デモ・プログラム

付録A.8にウインドウのデモ･プログラムを記載しています。アプリケーションの試作用としてご使用ください。

以下にデモ・プログラムのコンパイル時に必要なものを示します。

- デモ・プログラムのソース
- agdccop.h
- agdccol.h
- shyper.lib（μPD72123のデバイス・ドライバ・ライブラリ）
  （メモリ・モデルによってs，m，c，lのいずれか）
- その他，標準ヘッダ・ファイル，標準ライブラリ

注意　ライブラリとデモ・プログラムのメモリ・モデルを合わせてください。

(1) 使用方法

以下に使用方法を説明します。

- スペース・キーで動作が変わります。
- ESCキーで終了します（画面クリアは行いません）。

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。

- [GIO.H]
- [GIOSYS.H]
- [AGDCCOL.H]
- [AGDCOP.H]
- [GMAIN.C]
- [GIO.C]
- [GSUB.C]
- [WINDOW.C]

保守／廃止

A.1　[GIO.H]

保守／廃止

GIO.H

```
00001 /*********************************/
00002 /*      @(#) gio.h 2.0           */
00003 /*********************************/
00004
00005 /*
00006  * NEC AGDC library header file
00007  */
00008
00009 /*
00010  * コピー属性
00011  */
00012 #define TYPEBITS        0xc000  /*==== コピータイプ ====*/
00013 #define NORM_COPY       0x0000  /* 単純コピー             */
00014 #define FRES_COPY       0x4000  /* 任意角回転コピー       */
00015 #define SLANT_COPY      0x8000  /* 傾斜コピー             */
00016                                                           /*====== 付加属性 ======*/
00017 #define ESE_BIT         0x0001  /* 転送源開始位置 右下    */
00018 #define REV_BIT         0x0002  /* 鏡像反転               */
00019 #define ROT_BIT         0x0004  /* 上下反転               */
00020 #define D90_BIT         0x0008  /* ９０度回転コピー       */
00021 #define NO_FS           0x0010  /* ドット補正をしない     */
00022 #define NO_MAG          0x0020  /* 拡大／縮小をしない     */
00023 #define SR_MULTI        0x0040  /* マルチソース */
00024 #define FAST_BIT        0x0080  /* 高速指定 */
00025
00026 /*
00027  * 一般的な描画モード
00028  */
00029 #define PSET    0x02
00030 #define XOR             0x47
00031 #define AND             0x89
00032 #define OR              0xc7
00033 #define PRESET  0x27
00034
00035 /*
00036  * 描画プレーンの指定（カラー・パレット指定が可能なときはパレット番号になる）
00037  */
00038 #define P_BLACK         0
00039 #define P_RED           1
00040 #define P_GREEN         2
00041 #define P_YELLOW        3
00042 #define P_BLUE          4
00043 #define P_MAGENTA       5
00044 #define P_CYAN          6
00045 #define P_WHITE         7
00046 #define P_ALL           0xff
00047
00048
00049 /* μＰＤ７２１２０レジスタ定義 */
00050 #define R_EADORGL               0x00
00051 #define R_EADORGH_B             0x02
00052 #define R_EADORGH               0x02
00053 #define R_dADORG_B              0x03
00054 #define R_EAD1L                 0x04
00055 #define R_EAD1H_B               0x06
00056 #define R_dAD1_B                0x07
00057 #define R_EAD2L                 0x08
00058 #define R_EAD2H_B               0x0a
00059 #define R_dAD2_B                0x0b
00060 #define R_PDISPSL               0x0c
00061 #define R_PDISPSH_B             0x0e
00062 #define R_PDISPDL               0x10
00063 #define R_PDISPDH_B             0x12
00064 #define R_PMAX                  0x14
00065 #define R_MOD10                 0x16
00066 #define R_PTNPL                 0x18
00067 #define R_PTNPH                 0x1a
00068 #define R_STACKL                0x1c
00069 #define R_STACKH_B              0x1e
```

1

保守／廃止

GIO.H

```
00070
00071  #define R_EADORGSL              0x20
00072  #define R_EADORGSH_B    0x22
00073  #define R_EADORGSH              0x22
00074  #define R_dADORGS_B             0x23
00075  #define R_ORGSEL_BP             0x0c
00076
00077  #define R_EAD3L                 0x24
00078  #define R_EAD3H_B               0x26
00079
00080  #define R_CTRL2_B               0x3b
00081  #define         B_RVS           0x20            /* RVS of CTRL2 on */
00082  #define         B_VBIE          0x10            /* VBIE of CTRL2 on */
00083  #define         B_DWAITE        0x08            /* DWAIT of CTRL2 on */
00084  #define         B_DRWMD         0x04            /* DRWMD of CTRL2 on */
00085  #define         B_BSYMD         0x02            /* BSYMD of CTRL2 on */
00086  #define         B_BANKX         0x01            /* BANKX of CTRL2 on */
00087  #define R_BANK_B                0x3c
00088  #define R_CTRL_B                0x3d
00089  #define         B_DBIE          0x80            /* DBIE of CTRL on */
00090  #define         B_PBIE          0x40            /* PBIE of CTRL on */
00091  #define         B_CLPIE         0x20            /* CLPIE of CTRL on */
00092  #define         B_ABORT         0x02            /* ABORT of CTRL on */
00093  #define         B_RESET         0x01            /* RESET of CTRL on */
00094  #define R_STATUS                0x3c
00095  #define R_PGPORT                0x3e
00096  #define R_X                             0x40
00097  #define R_Y                             0x42
00098  #define R_DX                    0x44
00099  #define R_DY                    0x46
00100  #define R_XS                    0x48
00101  #define R_YS                    0x4a
00102  #define R_XE                    0x4c
00103  #define R_YE                    0x4e
00104  #define R_XC                    0x50
00105  #define R_YC                    0x52
00106  #define R_DH                    0x54
00107  #define R_DV                    0x56
00108  #define R_PITCHS                0x58
00109  #define R_PITCHD                0x5a
00110  #define R_STMAX                 0x5c
00111  #define R_PLANES                0x5e
00112  #define R_PTNCNT                0x60
00113  #define R_XCLMIN                0x62
00114  #define R_YCLMIN                0x64
00115  #define R_XCLMAX                0x66
00116  #define R_YCLMAX                0x68
00117  #define R_PTNH                  0x6a
00118  #define R_MAG_HV_B              0x6c
00119  #define R_CLIP_B                0x6d
00120  #define R_COMMAND               0x6e
00121  #define R_DISPLAY_CTRL  0x70
00122  #define R_DISP_PITCH    0x72
00123  #define R_DADL                  0x74
00124  #define R_DADH_WC               0x76
00125  #define R_GCSRX                 0x78
00126  #define R_GCSRYS                0x7a
00127  #define R_GCSRYE                0x7c
00128  #define R_SYNC                  0x7e
```

2

保守／廃止

A.2　[GIOSYS.H]

保守／廃止

GIOSYS.H

```
00001 /*********************************/
00002 /*      @(#) giosys.h 2.0        */
00003 /*********************************/
00004
00005 #define VOID    /**/
00006 #define GLOBAL  /**/
00007 #define REG     register
00008 #define u_char  unsigned char
00009 #define FLAG    unsigned char
00010 #define u_int   unsigned int
00011 #define u_short unsigned short
00012 #define u_long  unsigned long
00013
00014 #define L_regW(x)       ((u_int)((x) & 0x0000ffffL))
00015 #define H_regW(x)       ((u_int)((u_long)(x) >> 16))
00016 #define abs(i)          (((i) < 0) ? -(i) : (i))
00017
00018 #define SWAP(x,y)       { int i; i = x; x = y; y = i; }
00019
00020 /* PC-9801シリーズタイプコード */
00021 #define TYPE_NORM               0x10    /* 1 = Nomal mode  0 = High reso mode   */
00022 #define TYPE_VX                 0x20    /* 1 = PC-9801VX   0 = PC-98XL          */
00023
00024 /* Ｉ／Ｏアドレス */
00025 #define MEMMAP_PORT             0xD0    /* メモリマップ制御ポート */
00026 #define MEMSEG_PORT             0xD2    /* メモリマップセグメント指定ポート */
00027
00028 /* Ｉ／Ｏレジスタ内容 */
00029 #define IRSEL                   0x01    /* ＡＧＤＣ，パレットレジスタマップ指定 */
00030 #define DMSEL                   0x02    /* ＡＧＤＣメモリマップ指定 */
00031 #define PDSEL                   0x10    /* 表示イネーブル指定 */
00032 #define VSEG_MASK               0xF0    /* ＡＧＤＣメモリマップセグメント指定 */
00033 #define VSEG_LSB                4               /* ＡＧＤＣメモリマップセグメント指定
      */
00034 #define RSEG_MASK               0x0F    /* ＡＧＤＣ，パレットレジスタセグメント指定 *
      /
00035 #define RSEG_LSB                0               /* ＡＧＤＣ，パレットレジスタセグメン
      ト指定 */
00036
00037 /* 表示制御レジスタ オフセットアドレス */
00038 #define PIXEL_ADDR              0x80    /* Color Look-up Table(Pixel Address) */
00039 #define COLOR_VALUE             0x82    /* Color Look-up Table(Color Value) */
00040
00041 /* Ｉ／Ｏウィンドウ制御データ */
00042 #define IO_WINDOW               0x80    /* メモリマップドＩ／Ｏアドレス(PC-98XL HR) *
      /
00043 #define EX_ENABLE               0x10    /* 拡張メモリ許可 (PC-98XL HR) */
00044
00045 /*
00046  * 画面（表示，作業）メモリ構成
00047  */
00048 #define DSP_PITCH       0x040000L       /* disp memory plane's pitch(word) */
00049 #define DSP_TOP         0x000000L       /* 表示メモリ先頭アドレス(AGDCアドレス)*/
00050 #define DSP_SIZ         0x100000L       /* 表示メモリサイズ（ワード数）  */
00051 #define DRWSC_TOP       0x004000L       /* 描画専用メモリ先頭アドレス(AGDCアドレス) *
      /
00052 #define DRWSC_SIZ       0x03C000L       /* 描画専用メモリサイズ（ワード数）*/
00053 #define WRKSC_TOP       0x044000L       /* ワーク画面先頭アドレス(AGDCアドレス) */
00054 #define WRKSC_SIZ       0x03C000L       /* ワーク画面メモリサイズ（ワード数）  */
00055 #define KANJI_TOP       0x100000L;      /* 漢字フォントデータ先頭アドレス(AGDCアドレ
      ス)*/
00056 #define MAXPLANE        4                       /* Max plane number */
00057 #define STACK_ADDR      0x0fffffL       /* スタック領域アドレス(未使用) */
00058 #define STACK_MAX       0x3000          /* 6word(単位)（大きさ）*/
00059
00060 /*<<< AGDC CONFIGURATION VALUE >>>*/
00061
00062 /*      フィルパターン設定領域 */
00063 #define HPADDR_WH       0x08            /* Pattern Address */
```

1

保守／廃止

GIOSYS.H

```
00064 #define HPADDR_WL       0x4000          /* ( AGDC Adr = 044000H ) */
00065 #define HPADDR_BH       0x10            /* Pattern Address */
00066 #define HPADDR_BL       0x8000          /* ( CPU  Adr = 088000H ) */
00067
00068 /* ペンマスク設定領域 */
00069 #define HGMADDR_WH      0x08            /* Pattern Address */
00070 #define HGMADDR_WL      0x5000          /* ( AGDC Adr = 045000H ) */
00071 #define HGMADDR_BH      0x10            /* Pattern Address */
00072 #define HGMADDR_BL      0xA000          /* ( CPU  Adr = 08A000H ) */
00073
00074 /* ペンパターン設定領域 */
00075 #define HGPADDR_WH      0x08            /* Pattern Address */
00076 #define HGPADDR_WL      0x5100          /* ( AGDC Adr = 045100H ) */
00077 #define HGPADDR_BH      0x10            /* Pattern Address */
00078 #define HGPADDR_BL      0xA200          /* ( CPU  Adr = 08A200H ) */
00079
00080 /* 文字に関する定義 */
00081 #define CHR_SIZAH       8               /* 横サイズ （ＡＮＫ）     */
00082 #define CHR_SIZKH       16              /* 横サイズ （ 漢字 ）     */
00083 #define CHR_SIZHSP      9               /* 横サイズ（ＡＮＫ余白こみ） */
00084 #define CHR_SIZV        16              /* 縦サイズ */
00085 #define CHR_SIZVSP      20              /* 縦サイズ（余白こみ） */
00086 #define FONTPITCH       1               /* フォントピッチ（ワード） */
00087 #define FONTDISP        16L             /* フォント間隔（ワード） */
00088 #define SIZEMAX 384
00089
00090 /* ボードモード選択 */
00091 #define SEL_IR  outp(MEMMAP_PORT, PDSEL|IRSEL)
00092 #define SEL_CR  outp(MEMMAP_PORT, PDSEL|IRSEL)
00093
00094 /*********************************************************
00095  *      _v_get_reg : 内部レジスタ読み込み（ワード）              *
00096  *********************************************************/
00097 #define _v_get_reg(r)   (*(int far *)(_QQZZ_reg | (r)))
00098
00099 /*********************************************************
00100  *      _v_get_reg : 内部レジスタ読み込み（バイト）              *
00101  *********************************************************/
00102 #define _v_get_regb(r)  (*(char far *)(_QQZZ_reg | (r)))
00103
00104 /*********************************************************
00105  *      _v_put_reg : 内部レジスタ書き込み（ワード）              *
00106  *********************************************************/
00107 #define _v_put_reg(r, d) (*(int  far *)(_QQZZ_reg | (r))=(d))
00108
00109 /*********************************************************
00110  *      _v_put_regb : 内部レジスタ書き込み（バイト）             *
00111  *********************************************************/
00112 #define _v_put_regb(r, d) (*(char far *)(_QQZZ_reg | (r))=(d))
00113
00114 #define MAX_CRTMODE     2               /* MAX 画面モード数       */
00115 #define FPTNMAX         48              /* 塗りつぶしパターン最大登録数 */
00116
00117 /*********************************************************
00118  * gramadr : グラフィックメモリアドレス計算                  *
00119  *********************************************************/
00120 #define gramadr(a)      ((unsigned short far *)(_QQZZ_gmem | (unsigned long)(a)))
00121 #define l_bank(a)       ((unsigned short)((a) >> 16))
00122 #define l_offset(a)     ((unsigned short)((a) & 0x0000ffffL))
00123
00124 /*
00125  * ＣＲＴ同期信号設定値
00126  */
00127 typedef struct                  /* SYNC Parameter                       */
00128 {                                       /* ============================ */
00129         u_int           hs;             /* H Sync
              */
00130         u_int           hbp;    /* H Back Porch                         */
00131         u_int           hh;             /* HBP to Center of HS          */
```

GIOSYS.H

```
00132           u_int           hd;             /* H Display
    */
00133           u_int           hfp;    /* H Front Porch                        */
00134           u_int           vs;             /* V Sync
                */
00135           u_int           vbp;    /* V Back Porch                         */
00136           u_int           lf;             /* Line/Field
    */
00137           u_int           vfp;    /* V Front Porch                        */
00138   } PARAsync;
00139
00140   /*
00141    * 表示制御, メモリ制御レジスタ設定値
00142    */
00143   typedef struct                          /* Display Register Parameter   */
00144   {

00145           u_int           cntl;           /* DISPLAY_CTRL                 (70H)   */
00146           u_int           pich;           /* AC + DISPLAY_PITCH   (72H)   */
00147           u_int           dadl;           /* DAD(low)                             (74H)
                */
00148           u_int           dadh;           /* WC + DAD(high)               (76H)   */
00149   } PARAdisp;
00150
00151   /*
00152    * 塗りつぶしパターン登録テーブル
00153    */
00154   typedef struct {
00155           int     Ptn_type;                       /* 塗りつぶしパターンタイプ        */
00156           u_int   Ptn_length;             /* 塗りつぶしパターン数            */
00157           u_short *Ptn_top;               /* 塗りつぶしパターン              */
00158   }       FPTNTBL;
00159
00160   /*
00161    * 画面管理テーブル
00162    */
00163
00164   #define SCRMAX  32                      /* 画面最大数（3以上） */
00165
00166   typedef struct {
00167           u_char  scr_status;             /* 使用状態 */
00168   #define         READY           0x01    /*       未使用 */
00169   #define         USED            0x02    /*       使用中 */
00170   #define         DELETED         0x03    /*       仮削除 */
00171   #define         PARMANENT       0x80    /*       パーマネント */
00172           u_short scr_width;              /* 幅   （ドット） */
00173           u_short scr_height;             /* 高さ（ドット） */
00174           u_short scr_depth;              /* 深さ（ドット） */
00175           u_short scr_planetop;   /* 描画対象プレーン先頭番号 */
00176           u_short scr_planecnt;   /* 描画対象プレーン枚数 */
00177           u_short scr_pitch;              /* ピッチ（ワード） */
00178           u_long  scr_displace;   /* プレーン間隔（ワード） */
00179           short   scr_org_x;              /* 原点x座標 */
00180           short   scr_org_y;              /* 原点y座標 */
00181           short   scr_cmode;              /* クリップモード */
00182           short   scr_cx0;                /* クリップエリア（x最小値） */
00183           short   scr_cx1;                /* クリップエリア（x最大値） */
00184           short   scr_cy0;                /* クリップエリア（y最小値） */
00185           short   scr_cy1;                /* クリップエリア（y最大値） */
00186           u_long  scr_top;                /* 先頭アドレス（AGDCアドレス） */
00187           u_long  scr_length;             /* 占有空間（ワード） */
00188           int     scr_idno;                       /* 対応する画面番号テーブルの番号 */
00189   }       SCRTBL;
00190
00191   /*
00192    * 多角形の座標値
00193    */
00194   struct  v_xy {
00195           int             x;
```

3

保守／廃止

GIOSYS.H

```
00196           int             y;
00197   };
```

A.3　[AGDCCOL.H]

保守／廃止

AGDCCOL.H

```
00001 /*********************************/
00002 /*       描画プレーンの指定        */
00003 /*********************************/
00004
00005 #define P_BLACK          0
00006 #define P_RED            1
00007 #define P_GREEN          2
00008 #define P_YELLOW         3
00009 #define P_BLUE           4
00010 #define P_MAGENTA        5
00011 #define P_CYAN           6
00012 #define P_WHITE          7
00013 #define P_ALL            (~0)
```

1

A.4　[AGDCOP.H]

保守／廃止

AGDCOP.H

```
00001 /************************/
00002 /*      AGDC演算定義      */
00003 /************************/
00004
00005 #define ST_PSET          0xC9            /* Stencil PSET */
00006 #define ST_OR            0xC7            /* Stencil OR */
00007 #define ST_AND           0x79            /* Stencil AND */
00008 #define ST_XOR           0x47            /* Stencil XOR */
00009 #define ST_NOT           0x9c            /* Stencil NOT */
00010 #define FL_PSET          0x02            /* Fill PSET */
00011 #define FL_OR            0xC7            /* Fill OR */
00012 #define FL_AND           0x82            /* Fill AND */
00013 #define FL_XOR           0x47            /* Fill XOR */
00014 #define FL_NOT           0x13            /* Fill NOT */
```

1

A.5　[GMAIN.C]

保守／廃止

GMAIN.C

```
00001 /*********************************/
00002 /*      @(#)' gmain.c 3.2        */
00003 /*********************************/
00004
00005 #include <ctype.h>
00006 #include <dos.h>
00007 #include "giosys.h"
00008 #include "gio.h"
00009
00010 /* ＣＲＴ同期信号設定値 */
00011 static  PARAsync  _QQ_sync[MAX_CRTMODE+1] = {   /* CRT parameter */
00012         { 7 ,  8,0x34,0x4f,   8,   8, 0x19,0x190,  7 }, /*  640*400 */
00013         { 4 , 11,  28,  63,   9,   5,   22,  384,  9 }, /* 1024*768 */
00014         { 4 ,  8,  34,  69,   6,   5,   22,  375,  9 }  /* 1120*750 */
00015 };
00016
00017 /* 表示制御, メモリ制御レジスタ設定値 */
00018 static  PARAdisp  _QQ_disp[MAX_CRTMODE+1] = {   /* Display parameter */
00019     /* DISPLAY_CTRL , DISP_PITCH , DAD_L , DAD_H_WC     /*    CRT   */
00020 #define B_SVS   (1<<0)  /* スレーブ同期, 水平／垂直カウンタ初期化タイミング */
00021 #define B_SPST  (0<<1)  /* 同期パラメタ設定フラグ */
00022 #define B_LFI   (0<<2)  /* 総ライン数／フレーム（インターレース） */
00023 #define B_SD    (0<<3)  /* BLANK端子出力状態 */
00024 #define B_MS    (0<<4)  /* マスター／スレーブ指定 */
00025 #define B_MASK  (0<<5)  /* 同期信号マスク（B_MSの内容によって意味が変わる）*/
00026 #define B_TCCL  (1<<6)  /* 表示サイクルカウンタ初期化タイミング */
00027 #define B_FCCL  (0<<7)  /* フィールドカウンタ初期化タイミング */
00028 #define B_SC    (0<<8)  /* データトランスファモード時の内部動作定義 */
00029 #define B_RE    (1<<9)  /* リフレッシュ動作の有無 */
00030 #define B_IN    (0<<10) /* インターレース／ノンインターレース指定 */
00031 #define B_DAD_  (7<<11) /* 表示アドレス更新形態 */
00032 #define B_DTT   (0<<14) /* データ転送タイミング信号出力タイミング */
00033 #define B_DTM   (1<<15) /* 表示サイクル発生タイミング */
00034 #define CTRL_DATA       B_SVS|B_SPST|B_LFI|B_SD|B_MS|B_MASK|B_TCCL|B_FCCL|B_SC|B_RE|B
      _IN|B_DAD_|B_DTT|B_DTM
00035         { CTRL_DATA,    0x28,     0x0000,   0x2700 },   /*  640*400 */
00036         { 0x8741,       0x40,     0x0000,   0x3f00 },   /* 1024*768 */
00037         { 0x8741,       0x46,     0x0000,   0x4500 }    /* 1120*750 */
00038 };
00039
00040 /* システム情報 */
00041 GLOBAL  u_char  _QQZZ_PCtype    = 0;    /* PC-98 type */
00042 GLOBAL  u_int   _QQZZ_AGDC_SEG  = 0;    /* AGDCレジスタセグメント */
00043 GLOBAL  u_int   _QQZZ_GMEM_SEG  = 0;    /* グラフィックメモリセグメント */
00044 GLOBAL  u_long  _QQZZ_reg;                              /* AGDCレジスタアドレスオフセ
      ット */
00045 GLOBAL  u_long  _QQZZ_gmem;                             /* グラフィックメモリアドレス
      オフセット */
00046 GLOBAL  u_char  _QQZZ_seg_base  = 0x00; /* メモリセグメント選択値 */
00047
00048 /* ＡＧＤＣ制御情報 */
00049 GLOBAL  u_char  _QQZZ_ctrl;                             /* CTRLレジスタデフォルト値 *
      /
00050 GLOBAL  u_char  _QQZZ_ctrl2;                    /* CTRL2レジスタデフォルト値 */
00051
00052 /* 画面管理情報 */
00053 GLOBAL  int     _QQZZ_DSPscr;                           /* 表示画面の枚数 */
00054 GLOBAL  int     _QQZZ_MAXscr;                           /* 全画面の枚数（後で削除） *
      /
00055 GLOBAL  SCRTBL  _QQZZ_screen[SCRMAX];   /* 画面管理テーブル */
00056 GLOBAL  int     _QQZZ_scrnid[SCRMAX];           /* 画面番号管理テーブル */
00057 GLOBAL  int     _QQZZ_WRK0scr;                          /* 作業画面０画面番号（マスク
      専用） */
00058 GLOBAL  int     _QQZZ_WRK1scr;                          /* 作業画面１画面番号 */
00059 GLOBAL  int     _QQZZ_DRWscrtop;                        /* 描画専用画面の最小画面番号
       */
00060 GLOBAL  u_long  _QQZZ_LEFTgram;                 /* 描画専用画面用メモリの残容量(ﾊﾞｲﾄ)
      */
00061 GLOBAL  u_long  _QQZZ_FREEgram;                 /* 描画専用画面用メモリの空き領域先頭
```

1

保守／廃止

GMAIN.C

```
00061 */
00062
00063 /* カレント画面情報 */
00064 GLOBAL  SCRTBL  *_QQZZ_CURscreen;                /* 描画対象画面管理テーブル */
00065 GLOBAL  int     _QQZZ_CURscr;                          /* 描画対象画面番号 */
00066
00067 /* 表示管理情報 */
00068 GLOBAL  u_int   _QQZZ_Bcolor = 0;                /* 背景色（初期値＝０：黒）          */
00069
00070 /* 描画修飾情報 */
00071 GLOBAL  u_int   _QQZZ_ColorCode = 0;     /* カラーコード */
00072 GLOBAL  u_int   _QQZZ_DrawMode = PSET;  /* 描画モード */
00073 GLOBAL  u_char  _QQZZ_LinePL   = 0;              /* 0: １６ビット，1: ３２ビット線種
      */
00074 GLOBAL  u_int   _QQZZ_LinePtnL = 0xffff;/* 線種パターン（下位１６ビット）*/
00075 GLOBAL  u_int   _QQZZ_LinePtnH = 0xffff;/* 線種パターン（上位１６ビット）*/
00076 GLOBAL  u_char  _QQZZ_LineWid   = 0x0;  /* 線幅 */
00077 GLOBAL  char    _QQZZ_LinePtnEF = 0;     /* 線種パターン拡大フラグ */
00078 GLOBAL  FPTNTBL _QQZZ_Fptnlist[FPTNMAX];/* 塗りつぶしパターン登録リスト */
00079 GLOBAL  u_char  _QQZZ_FptnType = 0;             /* 塗りつぶし種別*/
00080 GLOBAL  u_int   _QQZZ_Fptnwidth= 1;             /* 塗りつぶしパターン幅 */
00081 GLOBAL  u_int   _QQZZ_Fptncnt  = 0xffff;/* 塗りつぶしパターン法    */
00082 GLOBAL  u_int   _QQZZ_MAGHV  = 0xff;   /* 水平，垂直方向 拡大／縮小係数 */
00083 GLOBAL  u_int   _QQZZ_charenlarge_H = 0;/* 文字描画拡大／縮小フラグ(水平)*/
00084 GLOBAL  u_int   _QQZZ_charenlarge_V = 0;/* 文字描画拡大／縮小フラグ(垂直)*/
00085 GLOBAL  u_int   _QQZZ_charmag_H = 0xF;  /* 文字描画拡大／縮小率(水平) */
00086 GLOBAL  u_int   _QQZZ_charmag_V = 0xF;  /* 文字描画拡大／縮小率(垂直) */
00087 GLOBAL  u_int   _QQZZ_charsize_H = CHR_SIZAH;/* 文字描画幅（指定値） */
00088 GLOBAL  u_int   _QQZZ_charsize_V = CHR_SIZV;/* 文字描画高さ（指定値） */
00089 GLOBAL  int     _QQZZ_charangle_HX = CHR_SIZAH; /* 文字底辺回転角（ｘ成分） */
00090 GLOBAL  int     _QQZZ_charangle_HY = 0;          /* 文字底辺回転角（ｙ成分） */
00091 GLOBAL  int     _QQZZ_charangle_VX = 0;          /* 文字側辺回転角（ｘ成分） */
00092 GLOBAL  int     _QQZZ_charangle_VY = CHR_SIZV;/* 文字側辺回転角（ｙ成分） */
00093 GLOBAL  int     _QQZZ_chartransf = 0;            /* 変形文字描画 */
00094 GLOBAL  int     _QQZZ_chardoff_x = 0;            /* 文字描画オフセット（ｘ） */
00095 GLOBAL  int     _QQZZ_chardoff_y = CHR_SIZV;/* 文字描画オフセット（ｙ） */
00096 GLOBAL  int     _QQZZ_alignment_x = CHR_SIZKH;/* 文字配置ライン（ｘ成分）*/
00097 GLOBAL  int     _QQZZ_alignment_y = CHR_SIZV;/* 文字配置ライン（ｙ成分）*/
00098 GLOBAL  int     _QQZZ_chardistance = CHR_SIZAH+1;/* 文字配置間隔 */
00099 GLOBAL  char    _QQZZ_halfadjust = 0;  /* 半角調整指定 */
00100 GLOBAL  int     _QQZZ_copyangle_HX = CHR_SIZKH;/* コピー底辺回転角(x成分) */
00101 GLOBAL  int     _QQZZ_copyangle_HY = 0;          /* コピー底辺回転角(y成分) */
00102 GLOBAL  int     _QQZZ_copyangle_VX = 0;          /* コピー側辺回転角(x成分) */
00103 GLOBAL  int     _QQZZ_copyangle_VY = CHR_SIZV;/* コピー側辺回転角(y成分) */
00104 GLOBAL  int     _QQZZ_copytransf = 0;            /* 変形コピーフラグ */
00105 GLOBAL  u_int   _QQZZ_Gpen_ptncnt = 16; /* ペンパターンのライン数 */
00106 GLOBAL  u_int   _QQZZ_Gpen_mskcnt = 16; /* ペンマスクのライン数 */
00107 GLOBAL  int     _QQZZ_Tip_sx;                          /* 最後に描いた弧の始点ｘ座標
       */
00108 GLOBAL  int     _QQZZ_Tip_sy;                          /* 最後に描いた弧の始点ｙ座標
       */
00109 GLOBAL  int     _QQZZ_Tip_ex;                          /* 最後に描いた弧の終点ｘ座標
       */
00110 GLOBAL  int     _QQZZ_Tip_ey;                          /* 最後に描いた弧の終点ｙ座標
       */
00111
00112 /* 外部参照関数 */
00113 extern  int             _QQZZ_copy_AC();
00114
```

2

保守／廃止

GMAIN.C

```
00114
00115 /*****************************************************************
00116  *                                                               *
00117  * PC-9801シリーズのタイプを識別する                             *
00118  *                                                               *
00119  *****************************************************************/
00120 static u_char
00121 _QQ_pc_type()
00122 {
00123         return(TYPE_NORM | TYPE_VX);
00124 }
00125
```

GMAIN.C

```
00125
00126 /*************************************************************
00127  *                                                           *
00128  * グラフィックドライバ初期化                                *
00129  *                                                           *
00130  *************************************************************/
00131 GLOBAL int
00132 gInit(crt_mode, plane)
00133 int             crt_mode;       /* 画面モード               */
00134 int             plane;          /* 描画対象プレーン数       */
00135 {
00136 static  FLAG    FirstTime = 1;
00137         REG int i;
00138         int   .  wd, hi;
00139
00140         if (crt_mode < 0 || MAX_CRTMODE < crt_mode) {
00141                 return(-1);
00142         }
00143         if (plane < 1 || MAXPLANE < plane) {
00144                 return(-2);
00145         }
00146         _QQZZ_PCtype = _QQ_pc_type();
00147         switch ( _QQZZ_PCtype ) {
00148          default :                      /* PC-98XL High resolution */
00149                 /* FALL INTO... */
00150          case TYPE_VX|TYPE_NORM :       /* PC-9801 512KB  */
00151                 _QQZZ_seg_base  = 0x00;         /* メモリセグメント選択値 */
00152                 _QQZZ_AGDC_SEG = 0x87F0;        /* AGDCレジスタセグメント */
00153                 _QQZZ_GMEM_SEG = 0x8000;        /* グラフィックメモリセグメント */
00154                 break;
00155          case TYPE_VX   :               /* PC-9801 640KB  */
00156          case TYPE_NORM :               /* PC-98XL Normal */
00157                 _QQZZ_seg_base  = 0x88;         /* メモリセグメント選択値 */
00158                 _QQZZ_AGDC_SEG = 0xc7f0;        /* AGDCレジスタセグメント */
00159                 _QQZZ_GMEM_SEG = 0xc000;        /* グラフィックメモリセグメント */
00160                 break;
00161         }
00162         outp(MEMSEG_PORT, _QQZZ_seg_base);
00163         _QQZZ_reg = (u_long)_QQZZ_AGDC_SEG << 16;
00164         _QQZZ_gmem= (u_long)_QQZZ_GMEM_SEG << 16;
00165         SEL_IR;                                 /* 内部レジスタ・アクセス許可 */
00166
00167         /* 画面分解能設定 */
00168         switch ( crt_mode ) {
00169          case 0  :
00170                 wd =  640;
00171                 hi =  400;
00172                 break;
00173          case 1  :
00174                 wd = 1024;
00175                 hi =  768;
00176                 break;
00177          case 2  :
00178                 wd = 1120;
00179                 hi =  750;
00180                 break;
00181         }
00182         /* 画面管理テーブル初期化 */
00183         for(i = 0; i < SCRMAX; i += 1) {
00184                 _QQZZ_screen[i].scr_status = READY;
00185                 _QQZZ_scrnid[i] = -1;
00186         }
00187         /* 表示画面確保 */
00188         _QQZZ_DSPscr = _QQ_AGDCinit(wd, hi, plane, _QQZZ_screen,
00189                                 &_QQ_disp[crt_mode], &_QQ_sync[crt_mode]);
00190         for(i = 0; i < _QQZZ_DSPscr; i += 1) {
00191                 _QQZZ_scrnid[i] = (int)i;
00192         }
00193         _QQZZ_MAXscr = _QQZZ_DSPscr;
```

4

保守／廃止

GMAIN.C

```
00194           _QQZZ_DRWscrtop = _QQZZ_DSPscr;
00195
00196           /* 作業画面確保 */
00197           _QQZZ_LEFTgram = 2*WRKSC_SIZ;    /* 作業画面用メモリの残容量(ﾊﾞｲﾄ)*/
00198           _QQZZ_FREEgram = 2*WRKSC_TOP;    /* 作業画面用メモリの空き領域先頭*/
00199           _QQZZ_WRK0scr = v_makescreen(wd,hi,1, _QQZZ_screen, _QQZZ_scrnid);
00200           _QQZZ_screen[_QQZZ_WRK0scr].scr_status |= PARMANENT;
00201           _QQZZ_DRWscrtop += 1;
00202           _QQZZ_WRK1scr = v_makescreen(wd,hi,plane, _QQZZ_screen, _QQZZ_scrnid);
00203           _QQZZ_screen[_QQZZ_WRK1scr].scr_status |= PARMANENT;
00204           _QQZZ_DRWscrtop += 1;
00205           _QQZZ_LEFTgram = 2*DRWSC_SIZ;    /* 描画専用画面用メモリの残容量(ﾊﾞｲﾄ)*/
00206           _QQZZ_FREEgram = 2*DRWSC_TOP;    /* 描画専用画面用メモリの空き領域先頭*/
00207
00208           v_act_page(0);
00209           v_initclip();
00210           v_lineptn(0);
00211           v_fillptn(0);
00212           if (FirstTime) {
00213                   FirstTime = 0;
00214                   v_clear(_QQZZ_Bcolor);          /* 画面消去 */
00215           }
00216           /* 文字列描画配置ラインの初期化 */
00217           _QQZZ_alignment_x = _QQZZ_CURscreen->scr_width-1;
00218           _QQZZ_alignment_y = 0;
00219
00220           return(_QQZZ_MAXscr);
00221   }
00222
```

GMAIN.C

```
00222
00223 /*****************************************************************
00224  *                                                               *
00225  * ＡＧＤＣ初期化                                              *
00226  *                                                               *
00227  *****************************************************************/
00228 static int
00229 _QQ_AGDCinit(w, h, d, screen_info, disp_para, sync_para)
00230 int     w, h, d;               /* 表示画面の大きさ */
00231 SCRTBL  *screen_info;
00232 PARAsync *sync_para;
00233 PARAdisp *disp_para;
00234 {
00235 #define SCR(s)  (screen_info+(s))
00236         int     pitch;
00237         REG int clk;
00238
00239         SEL_IR;
00240         /* ＡＧＤＣ制御レジスタデフォルト値設定 */
00241         _QQZZ_ctrl = 0x00;
00242         _QQZZ_ctrl2= B_DWAITE | B_DRWMD | B_BSYMD;
00243
00244         /* ＡＧＤＣリセット */
00245         _v_put_regb(R_CTRL_B, _QQZZ_ctrl | B_RESET | B_ABORT);  /* Reset & Abort */
00246         _v_put_regb(R_CTRL_B, _QQZZ_ctrl);
00247         _v_put_regb(R_CTRL2_B, _QQZZ_ctrl2);
00248         _v_put_regb(R_BANK_B, 0);
00249
00250         /* 表示同期設定 */
00251         _v_set_disp(disp_para, sync_para);
00252
00253         /* 画面管理情報設定 */
00254         SCR(0)->scr_status = USED | PARMANENT;  /* 使用状態 */
00255         SCR(0)->scr_width = w;                  /* 幅   (ドット) */
00256         SCR(0)->scr_height = h;                 /* 高さ (ドット) */
00257         SCR(0)->scr_depth = d;                  /* 深さ (ドット) */
00258         SCR(0)->scr_planetop = 0;               /* 描画対象ﾌﾟﾚｰﾝ先頭 */
00259         SCR(0)->scr_planecnt = d;               /* 描画対象ﾌﾟﾚｰﾝ枚数 */
00260         SCR(0)->scr_org_x = 0;                  /* 原点ｘ座標 */
00261         SCR(0)->scr_org_y = 0;                  /* 原点ｙ座標 */
00262         SCR(0)->scr_cmode = 0;                  /* クリップモード */
00263         SCR(0)->scr_cx0 = 0;
00264         SCR(0)->scr_cx1 = SCR(0)->scr_width-1;
00265         SCR(0)->scr_cy0 = 0;
00266         SCR(0)->scr_cy1 = SCR(0)->scr_height-1;
00267         SCR(0)->scr_pitch = (SCR(0)->scr_width+15)/16;
00268         SCR(0)->scr_top = DSP_TOP;
00269         SCR(0)->scr_displace = DSP_PITCH;
00270         SCR(0)->scr_length = SCR(0)->scr_displace*SCR(0)->scr_depth;
00271
00272         /* スタック設定 */
00273         _v_put_reg(R_STACKL, (u_short)(STACK_ADDR & 0xffff));
00274         _v_put_regb(R_STACKH_B, (u_char)(STACK_ADDR >> 16));
00275         _v_put_reg(R_STMAX, STACK_MAX);
00276
00277         return(1);              /* 作成した画面の枚数 */
00278 #undef  SCR
00279 }
00280
```

GMAIN.C

```
00280
00281 /*****************************************************************
00282  *                                                               *
00283  * 描画専用画面作成                                              *
00284  *                                                               *
00285  *****************************************************************/
00286 GLOBAL int
00287 gCreatePage(w, h, d)
00288 int     w, h, d;
00289 {
00290         int     scrn_no;
00291
00292         /* 画面作成 */
00293         scrn_no = v_makescreen(w, h, d, _QQZZ_screen, _QQZZ_scrnid);
00294         /* 描画対象画面の再設定 */
00295         v_act_page(_QQZZ_CURscr);
00296
00297         return(scrn_no);
00298 }
00299
```

保守／廃止

GMAIN.C

```
00299
00300 /*****************************************************************
00301  *                                                               *
00302  * 新規描画専用画面確保                                          *
00303  *                                                               *
00304  *****************************************************************/
00305 static int
00306 v_makescreen(wd, hi, dp, screen_info, screen_id)
00307 int     wd, hi, dp;
00308 SCRTBL  *screen_info;
00309 int     *screen_id;
00310 {
00311 #define SCR(s)  (screen_info+(s))
00312         int     scrninf_no;     /* 使用される画面管理情報テーブルのの番号 */
00313         int     scrn_no;        /* 使用される画面番号テーブルの番号 */
00314         int     sweep;
00315         u_short pitch;
00316         u_long  scrn_size;
00317         u_long  v_sweepgram();
00318
00319         pitch = (wd+15)/16;
00320         scrn_size = (u_long)dp*hi*pitch;
00321
00322         /* 画面管理テーブルの空きを探す */
00323         for(sweep = 0;;) {
00324                 /* 未使用領域の大きさを調べる */
00325                 if(_QQZZ_LEFTgram < scrn_size) {
00326                         if(sweep == 0) {
00327                                 u_long  new_freetop;
00328                                 /* 空きが小さいのでメモリを整理する */
00329                                 new_freetop = v_sweepgram(screen_info, screen_id);
00330                                 if(new_freetop == 0) {
00331                                         /* 管理テーブルに空きがない場合 */
00332                                         /* end for debug */
00333                                         return(-1);
00334                                 }
00335                                 _QQZZ_LEFTgram += _QQZZ_FREEgram-new_freetop;
00336                                 _QQZZ_FREEgram = new_freetop;
00337                                 continue;
00338                         } else {
00339                                 /* 空き容量が足らない */
00340                                 return(-1);
00341                         }
00342                 }
00343                 /* 割り当てるテーブルを探す（画面番号決定） */
00344                 scrninf_no = _QQZZ_MAXscr;
00345                 if(scrninf_no < SCRMAX) {
00346                         /* 見つかった場合 */
00347                         break;
00348                 } else {
00349                         /* 空いているテーブルがない場合 */
00350                         if(sweep == 0) {
00351                                 /* 管理テーブルの整理を行う */
00352                                 u_long  new_freetop;
00353                                 new_freetop = v_sweepgram(screen_info, screen_id);
00354                                 if(new_freetop == 0) {
00355                                         /* 管理テーブルに空きがない場合 */
00356                                         return(-1);
00357                                 }
00358                                 _QQZZ_LEFTgram += _QQZZ_FREEgram-new_freetop;
00359                                 _QQZZ_FREEgram = new_freetop;
00360                                 continue;
00361                         } else {
00362                                 /* 画面の枚数が多すぎる */
00363                                 return(-1);
00364                         }
00365                 }
00366         }
00367
```

8

保守／廃止

GMAIN.C

```
00368         /* 画面情報作成 */
00369         SCR(scrninf_no)->scr_status = USED;
00370         SCR(scrninf_no)->scr_width = wd;
00371         SCR(scrninf_no)->scr_height = hi;
00372         SCR(scrninf_no)->scr_depth = dp;
00373         SCR(scrninf_no)->scr_planetop = 0;
00374         SCR(scrninf_no)->scr_planecnt = dp;
00375         SCR(scrninf_no)->scr_pitch = pitch;
00376         SCR(scrninf_no)->scr_displace = (u_long)hi*pitch;
00377         SCR(scrninf_no)->scr_org_x = 0;
00378         SCR(scrninf_no)->scr_org_y = 0;
00379         SCR(scrninf_no)->scr_cmode = 0;
00380         SCR(scrninf_no)->scr_cx0 = 0;
00381         SCR(scrninf_no)->scr_cx1 = SCR(scrninf_no)->scr_width-1;
00382         SCR(scrninf_no)->scr_cy0 = 0;
00383         SCR(scrninf_no)->scr_cy1 = SCR(scrninf_no)->scr_height-1;
00384         SCR(scrninf_no)->scr_top = _QQZZ_FREEgram;
00385         SCR(scrninf_no)->scr_length = (u_long)dp*hi*pitch;
00386
00387         /* 画面管理状況の更新 */
00388         _QQZZ_LEFTgram -= SCR(scrninf_no)->scr_length;
00389         _QQZZ_FREEgram += SCR(scrninf_no)->scr_length;
00390         _QQZZ_MAXscr += 1;
00391
00392         /* 使用する画面番号テーブルの決定 */
00393         for(scrn_no = _QQZZ_DRWscrtop; scrn_no < SCRMAX; scrn_no += 1) {
00394                 if(*(screen_id+scrn_no) < 0) {
00395                         break;
00396                 }
00397         }
00398         *(screen_id+scrn_no) = scrninf_no;
00399         SCR(scrninf_no)->scr_idno = scrn_no;
00400
00401         return(scrn_no);
00402 #undef  SCR
00403 }
00404
```

GMAIN.C

```
00404
00405 /*****************************************************************
00406  *                                                               *
00407  * 描画専用画面削除                                              *
00408  *                                                               *
00409  *****************************************************************/
00410 GLOBAL int
00411 gDeletePage(no)
00412 int     no;
00413 {
00414         if(no == _QQZZ_CURscr) {
00415                 /* カレントは削除できない */
00416                 return(-1);
00417         }
00418         return(v_deltscreen(no));
00419 }
00420
```

GMAIN.C

```
00420
00421  /*****************************************************************
00422   *                                                               *
00423   * 描画専用画面削除                                              *
00424   *                                                               *
00425   *****************************************************************/
00426  static int
00427  v_deltscreen(scrn_no)
00428  int     scrn_no;
00429  {
00430          SCRTBL  *deltscrn;
00431
00432          if(_QQZZ_scrnid[scrn_no] < 0) {
00433                  /* 画面が存在しない場合 */
00434                  return(-1);
00435          }
00436
00437          /* 仮削除 */
00438          if(_QQZZ_screen[_QQZZ_scrnid[scrn_no]].scr_status & PARMANENT) {
00439                  /* パーマネント属性の画面は削除できない */
00440                  return(-1);
00441          } else {
00442                  _QQZZ_screen[_QQZZ_scrnid[scrn_no]].scr_status = DELETED;
00443          }
00444  }
00445
```

保守／廃止

GMAIN.C

```
00445
00446  /*************************************************************
00447   *                                                           *
00448   * グラフィックメモリのスイープ                              *
00449   *                                                           *
00450   *************************************************************/
00451  static u_long
00452  v_sweepgram(screen_info, screen_id)
00453  SCRTBL  *screen_info;
00454  int     *screen_id;
00455  {
00456  #define SCR(s)          (screen_info+(s))
00457  #define isdeleted(s)    (SCR(s)->scr_status == DELETED)
00458          int     srchtop;                /* 最後に調べた管理情報 */
00459          int     freetop;                /* 仮削除されている管理情報の先頭 */
00460          int     movetop;                /* 移動する管理情報の先頭 */
00461          int     moveend;                /* 移動する管理情報の最後+1 */
00462          int     deltcnt;                /* 削除される管理情報の総数 */
00463          int     leftcnt;                /* 残った管理情報の総数 */
00464          int     sweeptop;               /* グラフィックメモリスイープを
00465                                             開始するテーブルの番号 */
00466          int     clearno;
00467          u_long  freetopadr;             /* 仮削除されているGRAMの先頭 */
00468          u_long  movetopadr;             /* 移動するGRAMの先頭 */
00469          u_long  moveendadr;             /* 移動するGRAMの最後+1 */
00470          u_long  movesize;               /* 移動するGRAMの量(BYTE) */
00471
00472          /* 画面管理情報テーブルのスイープ */
00473          /* 仮削除されている管理情報を探す */
00474          freetopadr = 0;
00475          for(srchtop = _QQZZ_DRWscrtop; srchtop < _QQZZ_MAXscr; srchtop += 1) {
00476                  if(isdeleted(srchtop)) {
00477                          freetopadr = SCR(srchtop)->scr_top;
00478                          break;
00479                  }
00480          }
00481          if(freetopadr == 0) {
00482                  /* 画面管理情報テーブルに空きがない */
00483                  return(0);
00484          }
00485          leftcnt = 0;
00486          deltcnt = 0;
00487          sweeptop = freetop = srchtop;
00488          while(srchtop < _QQZZ_MAXscr) {
00489                  int     move;
00490                  int     movecnt;
00491                  /* 移動するべき管理情報の先頭を探す */
00492                  for(; srchtop < _QQZZ_MAXscr; srchtop += 1){
00493                          if(isdeleted(srchtop)) {
00494                                  *(screen_id+SCR(srchtop)->scr_idno) = -1;
00495                          } else {
00496                                  break;
00497                          }
00498                  }
00499                  movetop = srchtop;
00500                  if(movetop < _QQZZ_MAXscr) {
00501                          /* 移動すべき管理情報の最後を探す */
00502                          for(srchtop += 1; srchtop < _QQZZ_MAXscr; srchtop += 1) {
00503                                  if(isdeleted(srchtop)) {
00504                                          break;
00505                                  }
00506                          }
00507                          moveend = srchtop;
00508                          movecnt = moveend-movetop;
00509                          /* テーブル内容の移動 */
00510                          memcpy(SCR(freetop),SCR(movetop),movecnt*sizeof(SCRTBL));
00511                          freetop += movecnt;
00512                          leftcnt += movecnt;
00513                  }
```

12

保守／廃止

GMAIN.C

```
00514           }
00515           /* 使われなくなった画面管理情報のクリア */
00516           for(srchtop = freetop; srchtop < _QQZZ_MAXscr; srchtop += 1) {
00517                   SCR(srchtop)->scr_status = READY;
00518           }
00519           /* 画面番号管理テーブルの調整 */
00520           deltcnt = (_QQZZ_MAXscr-_QQZZ_DRWscrtop)-leftcnt;
00521           _QQZZ_MAXscr -= deltcnt;
00522           for(srchtop = _QQZZ_DRWscrtop; srchtop < _QQZZ_MAXscr; srchtop += 1) {
00523                   *(screen_id+SCR(srchtop)->scr_idno) = srchtop;
00524           }
00525           /* グラフィックメモリのスイープ */
00526           for(;sweeptop < _QQZZ_MAXscr; sweeptop += 1) {
00527                   /* グラフィックメモリの移動 */
00528                   _QQZZ_movescr(freetopadr, SCR(sweeptop));
00529                   SCR(sweeptop)->scr_top = freetopadr;
00530                   freetopadr += SCR(sweeptop)->scr_length;
00531           }
00532   #undef  SCR
00533
00534           return(freetopadr);
00535   }
00536
```

保守／廃止

GMAIN.C

```
00536
00537 /*****************************************************************
00538  *                                                               *
00539  * 画面移動                                                      *
00540  *                                                               *
00541  *****************************************************************/
00542 GLOBAL VOID
00543 _QQZZ_movescr(destadr, src_scrninfo)
00544 u_long  destadr;
00545 SCRTBL  *src_scrninfo;
00546 {
00547         u_long  org_off;
00548
00549         _QQZZ_statbusy(3);              /* Wait until(PPBUSY==0) */
00550
00551         org_off = (src_scrninfo->scr_height-1-src_scrninfo->scr_org_y)
00552                 *src_scrninfo->scr_pitch+src_scrninfo->scr_org_x/16;
00553         /* 転送元のパラメタ設定 */
00554         _v_put_reg(R_EADORGSL, L_regW(src_scrninfo->scr_top+org_off));
00555         _v_put_regb(R_EADORGSH_B, H_regW(src_scrninfo->scr_top+org_off));
00556         _v_put_regb(R_dADORGS_B, src_scrninfo->scr_org_x & 0xf);
00557         _v_put_reg(R_EAD2L, L_regW(src_scrninfo->scr_top));
00558         _v_put_regb(R_EAD2H_B, H_regW(src_scrninfo->scr_top));
00559         _v_put_regb(R_dAD2_B, 0);
00560         _v_put_reg(R_PITCHS, src_scrninfo->scr_pitch);
00561         _v_put_reg(R_PDISPSL, L_regW(src_scrninfo->scr_displace));
00562         _v_put_regb(R_PDISPSH_B, H_regW(src_scrninfo->scr_displace));
00563         /* 転送先側のパラメタ設定 */
00564         _v_put_reg(R_EADORGL, L_regW(destadr+org_off));
00565         _v_put_regb(R_EADORGH_B, H_regW(destadr+org_off));
00566         _v_put_regb(R_dADORG_B, src_scrninfo->scr_org_x & 0xf);
00567         _v_put_reg(R_EAD1L, L_regW(destadr));
00568         _v_put_regb(R_EAD1H_B, H_regW(destadr));
00569         _v_put_regb(R_dAD1_B, 0);
00570         _v_put_reg(R_PITCHD, src_scrninfo->scr_pitch);
00571         _v_put_reg(R_PDISPDL, L_regW(src_scrninfo->scr_displace));
00572         _v_put_regb(R_PDISPDH_B, H_regW(src_scrninfo->scr_displace));
00573         /* クリップ解除 */
00574         _v_put_reg(R_CLIP_B, 0);
00575         _v_put_reg(R_XCLMIN, 0);
00576         _v_put_reg(R_YCLMIN, 0);
00577         _v_put_reg(R_XCLMAX, src_scrninfo->scr_width-1);
00578         _v_put_reg(R_YCLMAX, src_scrninfo->scr_height-1);
00579         /* その他実行に必要なパラメタの設定 */
00580         _v_put_reg(R_DH, src_scrninfo->scr_width-1);
00581         _v_put_reg(R_DV, src_scrninfo->scr_height-1);
00582         _v_put_reg(R_PMAX, 1 << (src_scrninfo->scr_depth-1));
00583         _v_put_reg(R_PLANES, 0xffff);           /* 描画モード */
00584         _v_put_reg(R_MOD10, 0x00);              /* MOD1,MOD0 */
00585         _v_put_regb(R_MAG_HV_B, 0xFF);          /* MAGV,MAGH */
00586         /* コピー実行
00587                 修飾なし,(高速)マルチtoマルチ */
00588         _v_put_reg(R_COMMAND, 0x800c);
00589 }
00590
```

GMAIN.C

```
00590
00591  /*****************************************************************
00592   *                                                               *
00593   * クリップエリアリセット                                        *
00594   *                                                               *
00595   *****************************************************************/
00596  GLOBAL VOID
00597  gResetCliparea()
00598  {
00599          v_clipmode(0);
00600          v_cliparea(0, 0,
00601                  _QQZZ_CURscreen->scr_width-1, _QQZZ_CURscreen->scr_height-1);
00602  }
00603
```

GMAIN.C

```
00603
00604 /*****************************************************************
00605  *                                                               *
00606  * クリップエリアセット                                          *
00607  *                                                               *
00608  *****************************************************************/
00609 GLOBAL VOID
00610 gSetCliparea(x0, y0, x1, y1)
00611 {
00612         v_clipmode(0);
00613         v_cliparea(x0, y0, x1, y1);
00614 }
00615
```

GMAIN.C

```
00615
00616  /*****************************************************************
00617   *                                                               *
00618   * クリップエリア初期化                                          *
00619   *                                                               *
00620   *****************************************************************/
00621  GLOBAL VOID
00622  v_initclip()
00623  {
00624          int     scrn_no;
00625          SCRTBL  *scrn;
00626
00627          _QQZZ_CURscreen->scr_cmode = 0;
00628          _QQZZ_CURscreen->scr_cx0 = 0;
00629          _QQZZ_CURscreen->scr_cx1 = _QQZZ_CURscreen->scr_width-1;
00630          _QQZZ_CURscreen->scr_cy0 = 0;
00631          _QQZZ_CURscreen->scr_cy1 = _QQZZ_CURscreen->scr_height-1;
00632          v_clipmode(_QQZZ_CURscreen->scr_cmode);
00633          v_cliparea(_QQZZ_CURscreen->scr_cx0, _QQZZ_CURscreen->scr_cy0,
00634                     _QQZZ_CURscreen->scr_cx1, _QQZZ_CURscreen->scr_cy1);
00635  }
00636
```

GMAIN.C

```
00636
00637  /*****************************************************************
00638   *                                                               *
00639   * クリップモード設定                                          *
00640   *                                                               *
00641   *****************************************************************/
00642  GLOBAL VOID
00643  v_clipmode(n)
00644  int     n;
00645  {
00646          _QQZZ_CURscreen->scr_cmode = n;
00647          if(n == 1) {
00648                  _QQZZ_CURscreen->scr_cx0 = 0;
00649                  _QQZZ_CURscreen->scr_cx1 = _QQZZ_CURscreen->scr_width-1;
00650                  _QQZZ_CURscreen->scr_cy0 = 0;
00651                  _QQZZ_CURscreen->scr_cy1 = _QQZZ_CURscreen->scr_height-1;
00652          }
00653
00654          _v_clipmode(n);
00655  }
00656
```

GMAIN.C

```
00656
00657  /*****************************************************************
00658   *                                                               *
00659   * クリップエリア設定                                            *
00660   *                                                               *
00661   *****************************************************************/
00662  GLOBAL VOID
00663  v_cliparea(x0, y0, x1, y1)
00664  int     x0, y0, x1, y1;
00665  {
00666          _QQZZ_CURscreen->scr_cx0 = x0;
00667          _QQZZ_CURscreen->scr_cx1 = x1;
00668          _QQZZ_CURscreen->scr_cy0 = y0;
00669          _QQZZ_CURscreen->scr_cy1 = y1;
00670
00671          _v_cliparea(x0, y0, x1, y1);
00672  }
00673
```

GMAIN.C

```
00673
00674 /*****************************************************************
00675  *                                                               *
00676  * 表示画面制御                                                  *
00677  *                                                               *
00678  *****************************************************************/
00679 VOID    _v_set_disp(dispp, syncp)
00680 PARAdisp        *dispp;
00681 PARAsync        *syncp;
00682 {
00683         _v_put_reg(R_DISPLAY_CTRL, dispp->cntl & ~0x02);
00684         _v_put_reg(R_DISPLAY_CTRL, dispp->cntl | 0x02); /* ＳＹＮＣ書込み許可 */
00685         _v_put_reg(R_SYNC, syncp->hs);
00686         _v_put_reg(R_SYNC, syncp->hbp);
00687         _v_put_reg(R_SYNC, syncp->hh);
00688         _v_put_reg(R_SYNC, syncp->hd);
00689         _v_put_reg(R_SYNC, syncp->hfp);
00690         _v_put_reg(R_SYNC, syncp->vs);
00691         _v_put_reg(R_SYNC, syncp->vbp);
00692         _v_put_reg(R_SYNC, syncp->lf);
00693         _v_put_reg(R_SYNC, syncp->vfp);
00694         _v_put_reg(R_DISPLAY_CTRL, dispp->cntl); /* 禁止 */
00695         _v_put_reg(R_DISP_PITCH, dispp->pich);
00696         _v_put_reg(R_DADL, dispp->dadl);
00697         _v_put_reg(R_DADH_WC, dispp->dadh);
00698         _v_put_reg(R_GCSRX,  0x00);             /* Cursor cannot use */
00699         _v_put_reg(R_GCSRYS, 0x00);
00700         _v_put_reg(R_GCSRYE, 0x00);
00701 }
00702
```

GMAIN.C

```
00702
00703  /*****************************************************************
00704   *                                                               *
00705   * グラフィックス描画終了                                        *
00706   *                                                               *
00707   *****************************************************************/
00708  GLOBAL
00709  VOID    gEnd()
00710  {
00711          _QQZZ_statbusy(3);      /* Wait until(DPBUSY|PPBUSY==0) */
00712
00713          if ( !_QQZZ_PCtype ) {
00714                  outp(0x91, EX_ENABLE);
00715                  outp(0x93, EX_ENABLE);
00716          }
00717          outp(MEMMAP_PORT, 0);
00718  }
00719
```

保守／廃止

GMAIN.C

```
00719
00720  /*****************************************************************
00721   *                                                               *
00722   * 描画対象画面の指定                                            *
00723   *                                                               *
00724   *****************************************************************/
00725  GLOBAL int
00726  gDrawPage(n)
00727  int     n;              /* 描画対象画面番号 */
00728  {
00729          return(v_act_page(n));
00730  }
00731
```

保守／廃止

GMAIN.C

```
00731
00732  /*****************************************************************
00733   *                                                               *
00734   * 描画対象プレーンの指定                                        *
00735   *                                                               *
00736   *****************************************************************/
00737  GLOBAL int
00738  gDrawPlane(top, count)
00739  int     top;            /* 描画対象プレーン先頭番号 */
00740  int     count;          /* 描画対象プレーン枚数 */
00741  {
00742          if(top < 0) {
00743                  top = 0;
00744                  count = _QQZZ_CURscreen->scr_depth;
00745          }
00746          return(v_plane(top, count));
00747  }
00748
```

GMAIN.C

```
00748
00749 /*****************************************************************
00750  *                                                               *
00751  * 画面アドレスの計算                                            *
00752  *                                                               *
00753  *****************************************************************/
00754 GLOBAL int
00755 _QQ_get_scraddr(n, widp, adrp)
00756 int             n;
00757 u_long          *widp, *adrp;
00758 {
00759         u_long  wid, adr;
00760         int     scrn_no;
00761         SCRTBL  *scrn;
00762
00763         scrn_no = _QQZZ_scrnid[n];
00764         if(scrn_no < 0) {
00765                 return(scrn_no);
00766         }
00767         scrn = &_QQZZ_screen[scrn_no];
00768
00769         *widp = scrn->scr_displace;
00770         *adrp = scrn->scr_top
00771                 +scrn->scr_planetop*scrn->scr_displace;
00772
00773         return(scrn_no);
00774 }
00775
```

GMAIN.C

```
00775
00776  /*****************************************************************
00777   *                                                               *
00778   * 描画（アクティブ）画面の指定                                  *
00779   *                                                               *
00780   *****************************************************************/
00781  GLOBAL int
00782  v_act_page(n)
00783  int             n;              /* 描画対象画面番号       */
00784  {
00785          u_long  wid, adr;
00786          u_long  org;
00787          int     scrn_no;
00788          SCRTBL  *scrn;
00789
00790          scrn_no = _QQ_get_scraddr(n, &wid, &adr);
00791          if(scrn_no < 0) {
00792                  return(-1);
00793          }
00794          _QQZZ_CURscr = n;
00795          _QQZZ_CURscreen = scrn = &_QQZZ_screen[scrn_no];
00796          org = scrn->scr_pitch*((scrn->scr_height-1)-scrn->scr_org_y)
00797                          +scrn->scr_org_x/16;
00798          _QQZZ_statbusy(1);      /* Wait until(PPBUSY==0) */
00799          /* 描画対象画面に関するパラメタの設定 */
00800          _v_put_reg(R_PDISPDL, L_regW(wid));
00801          _v_put_regb(R_PDISPDH_B, H_regW(wid));
00802          _v_put_reg(R_PITCHD, scrn->scr_pitch);
00803          _v_put_reg(R_EADORGL, L_regW(adr+org));
00804          _v_put_regb(R_EADORGH_B, H_regW(adr+org));
00805          _v_put_regb(R_dADORG_B, scrn->scr_org_x & 0xF);
00806          /*          */
00807          _v_put_reg(R_PDISPSL, L_regW(wid));
00808          _v_put_regb(R_PDISPSH_B, H_regW(wid));
00809          _v_put_reg(R_PITCHS, scrn->scr_pitch);
00810          _v_put_reg(R_EADORGSL, L_regW(adr+org));
00811          _v_put_regb(R_EADORGSH_B, H_regW(adr+org));
00812          _v_put_regb(R_dADORGS_B, scrn->scr_org_x & 0xF);
00813          /*          */
00814          _v_put_reg(R_PMAX, 0x1 << (scrn->scr_planecnt-1));
00815          /* クリップエリアの設定 */
00816          _v_clipmode(_QQZZ_CURscreen->scr_cmode);
00817          _v_cliparea(_QQZZ_CURscreen->scr_cx0,_QQZZ_screen->scr_cy0,
00818                          _QQZZ_CURscreen->scr_cx1,_QQZZ_CURscreen->scr_cy1);
00819
00820          return(0);
00821  }
00822
```

GMAIN.C

```
00822
00823  /*****************************************************************
00824   *                                                               *
00825   * 描画対象プレーン指定                                          *
00826   *                                                               *
00827   *****************************************************************/
00828  GLOBAL VOID
00829  v_plane(top, cnt)
00830  int     top;            /* 描画対象プレーン先頭番号 */
00831  int     cnt;            /* 描画対象プレーン枚数 */
00832  {
00833          u_long  adr, org;
00834
00835          if (top < 0 || top+cnt > _QQZZ_CURscreen->scr_depth) {
00836                  return;
00837          }
00838          _QQZZ_CURscreen->scr_planetop = top;
00839          _QQZZ_CURscreen->scr_planecnt = cnt;
00840          org = _QQZZ_CURscreen->scr_pitch*((_QQZZ_CURscreen->scr_height-1)
00841                  -_QQZZ_CURscreen->scr_org_y)+_QQZZ_CURscreen->scr_org_x/16;
00842          adr = _QQZZ_CURscreen->scr_top+top*_QQZZ_CURscreen->scr_displace;
00843          /* 描画対象プレーン数理及び原点の設定 */
00844          _v_put_reg(R_EADORGL, L_regW(adr+org));
00845          _v_put_regb(R_EADORGH_B, H_regW(adr+org));
00846          _v_put_regb(R_dADORG_B, _QQZZ_CURscreen->scr_org_x & 0xF);
00847          _v_put_reg(R_EADORGSL, L_regW(adr+org));
00848          _v_put_regb(R_EADORGSH_B, H_regW(adr+org));
00849          _v_put_regb(R_dADORGS_B, _QQZZ_CURscreen->scr_org_x & 0xF);
00850          _v_put_reg(R_PMAX, 0x1 << (cnt-1));
00851  }
00852
```

保守／廃止

GMAIN.C

```
00852
00853  /*****************************************************************
00854   *                                                               *
00855   * 背景色の指定                                                  *
00856   *                                                               *
00857   *****************************************************************/
00858  GLOBAL  VOID
00859  gSetBackCode(n)
00860  int     n;
00861  {
00862          v_back_color(n);
00863  }
00864
```

GMAIN.C

```
00864
00865  /*****************************************************************
00866   *                                                               *
00867   * 背景色の指定                                                  *
00868   *                                                               *
00869   *****************************************************************/
00870  GLOBAL VOID
00871  v_back_color(n)
00872  u_int           n;              /* 背景色（パレット番号）  */
00873  {
00874          _QQZZ_Bcolor = n;
00875  }
00876
```

保守／廃止

GMAIN.C

```
00876
00877 /*****************************************************************
00878  *                                                               *
00879  * カラーコードの指定                                            *
00880  *                                                               *
00881  *****************************************************************/
00882 GLOBAL  VOID
00883 gSetDrawCode(c)
00884 u_int   c;
00885 {
00886         _QQZZ_ColorCode = c;
00887 }
00888
```

GMAIN.C

```
00888
00889 /*****************************************************************
00890  *                                                               *
00891  * 描画モードの指定                                              *
00892  *                                                               *
00893  *****************************************************************/
00894 GLOBAL  VOID
00895 gSetMode(m)
00896 u_int   m;
00897 {
00898         _QQZZ_DrawMode = m;
00899 }
00900
```

GMAIN.C

```
00900
00901 /*****************************************************************
00902  *                                                               *
00903  * 線種パターン設定                                              *
00904  *                                                               *
00905  *****************************************************************/
00906 GLOBAL
00907 VOID    gSetLinePtn(type, ptnp)
00908 int             type;           /* 線種パターンタイプ     */
00909 u_short         *ptnp;          /* 線種パターン           */
00910 {
00911         v_lineptn(type, ptnp);
00912 }
00913
```

GMAIN.C

```
00913
00914 /*****************************************************************
00915  *                                                               *
00916  * 線種パターン設定                                              *
00917  *                                                               *
00918  *****************************************************************/
00919 GLOBAL VOID
00920 v_lineptn(type, ptnp)
00921 int             type;
00922 u_short *ptnp;
00923 {
00924 REG     u_short         *p;
00925 static  u_short         l = 0xffff;
00926
00927         p = ptnp;
00928         switch ( type ) {
00929          case 0 :               /* 実線 */
00930                 p = &l;
00931                 /* FALL INTO... */
00932          case 1 :               /* １６ビット線種パターン */
00933                 _QQZZ_LinePL   = 0;
00934                 break;
00935          case 2 :               /* ３２ビット線種パターン */
00936                 _QQZZ_LinePL   = 1;
00937                 _QQZZ_LinePtnH = p[1];
00938         }
00939         _QQZZ_LinePtnL = p[0];
00940 }
00941
```

GMAIN.C

```
00941
00942 /*****************************************************************
00943  *                                                               *
00944  * 線幅設定                                                      *
00945  *                                                               *
00946  *****************************************************************/
00947 GLOBAL VOID
00948 v_linewid(wid, esl)
00949 int             wid;
00950 int             esl;
00951 {
00952 REG     u_int   es;
00953
00954         if ((es = esl) < 0) {
00955                 _QQZZ_LinePtnEF = 1; es = -es;
00956         } else {
00957                 _QQZZ_LinePtnEF = 0;
00958         }
00959         _QQZZ_LineWid = ((es & 0xf) << 4) | (wid & 0xf);
00960 }
00961
```

GMAIN.C

```
00961
00962 /*****************************************************************
00963  *                                                               *
00964  * 塗りつぶしパターン登録                                        *
00965  *                                                               *
00966  *****************************************************************/
00967 GLOBAL
00968 gEntryFillPtn(id, type, n, ptnp)
00969 int             id;
00970 int             type;
00971 u_int   n;
00972 u_short *ptnp;
00973 {
00974         if(id < FPTNMAX) {
00975                 _QQZZ_Fptnlist[id].Ptn_type = type;
00976                 if(type) {
00977                         _QQZZ_Fptnlist[id].Ptn_length = n;
00978                         _QQZZ_Fptnlist[id].Ptn_top = ptnp;
00979                 }
00980                 return(0);
00981         } else {
00982                 return(-1);
00983         }
00984 }
00985
```

保守／廃止

GMAIN.C

```
00985
00986 /*****************************************************************
00987  *                                                               *
00988  * 塗りつぶしパターン設定                                        *
00989  *                                                               *
00990  *****************************************************************/
00991 GLOBAL
00992 gSetFillPtn(id)
00993 int             id;
00994 {
00995         v_fillptn(_QQZZ_Fptnlist[id].Ptn_type, _QQZZ_Fptnlist[id].Ptn_length, _QQZZ_F
      ptnlist[id].Ptn_top);
00996 }
00997
```

保守／廃止

GMAIN.C

```
00997
00998  /*****************************************************************
00999   *                                                               *
01000   * 塗りつぶしパターン設定                                        *
01001   *                                                               *
01002   *****************************************************************/
01003  GLOBAL VOID
01004  v_fillptn(type, n, ptnp)
01005  int             type;
01006  u_int   n;
01007  u_short *ptnp;
01008  {
01009
01010          switch ( type ) {
01011           case 0 :                /* べた塗り */
01012                   _QQZZ_FptnType = 0;
01013                   _QQZZ_Fptncnt  = 0xffff;
01014                   _QQZZ_Fptnwidth= 1;
01015                   return;
01016           case 1 :                /* 単色塗りつぶし */
01017                   _QQZZ_FptnType = 1;
01018                   _QQZZ_Fptncnt  = n;
01019                   _QQZZ_Fptnwidth= 16;
01020                   break;
01021           case 2 :                /* カラー塗りつぶし */
01022                   _QQZZ_FptnType = 2;
01023                   _QQZZ_Fptncnt  = n;
01024                   _QQZZ_Fptnwidth= 16;
01025                   n *= _QQZZ_CURscreen->scr_depth;
01026                   break;
01027           case 6 :                /* ３２ビットパターン */
01028                   _QQZZ_FptnType = 6;
01029                   _QQZZ_Fptncnt  = n;
01030                   _QQZZ_Fptnwidth= 32;
01031                   n *= 2*_QQZZ_CURscreen->scr_depth;
01032                   break;
01033          }
01034          if ((type != 6) && (n == 1)) {
01035                   _QQZZ_FptnType = 0;
01036                   _QQZZ_Fptncnt  = *ptnp;
01037          } else {
01038                   _QQZZ_ptnp(ptnp, n);
01039          }
01040  }
01041
```

GMAIN.C

```
01041
01042 /*****************************************************************
01043  *                                                               *
01044  * グラフィックスペンパターン設定                                *
01045  *                                                               *
01046  *****************************************************************/
01047 GLOBAL VOID
01048 gSetGpenPtn(ptnp)
01049 u_short         *ptnp;
01050 {
01051         _QQZZ_Gpen_ptncnt  = 16;
01052         _QQZZ_Gpen_ptnp(ptnp, _QQZZ_Gpen_ptncnt);
01053 }
01054
```

GMAIN.C

```
01054
01055 /*****************************************************************
01056  *                                                               *
01057  * グラフィックスペンマスク設定                                  *
01058  *                                                               *
01059  *****************************************************************/
01060 GLOBAL VOID
01061 gSetGpenMask(ptnp)
01062 u_short         *ptnp;
01063 {
01064         _QQZZ_Gpen_mskcnt  = 16;
01065         _QQZZ_Gpen_mask(ptnp, _QQZZ_Gpen_mskcnt);
01066 }
01067
```

保守／廃止

GMAIN.C

```
01067
01068 /*****************************************************************
01069  *                                                               *
01070  * 文字描画サイズ設定                                          *
01071  *                                                               *
01072  *****************************************************************/
01073 GLOBAL int
01074 gSetCharSize(wd, hi)
01075 u_int   wd, hi;
01076 {
01077         _QQZZ_charsize_H = wd;  /* 文字描画幅（指定値） */
01078         _QQZZ_charsize_V = hi;  /* 文字描画高さ（指定値） */
01079
01080         /* 水平方向倍率の計算と設定 */
01081         _QQZZ_charenlarge_H = _QQ_calcmagcode(CHR_SIZAH, wd,
01082                                         _QQZZ_charangle_HX, _QQZZ_charangle_HY,
01083                                         &_QQZZ_charmag_H);
01084         /* 垂直方向倍率の計算と設定 */
01085         _QQZZ_charenlarge_V = _QQ_calcmagcode(CHR_SIZV, hi,
01086                                         _QQZZ_charangle_VX, _QQZZ_charangle_VY,
01087                                         &_QQZZ_charmag_V);
01088
01089         /* 文字描画オフセットの計算 */
01090         _QQ_calcchardoff(&_QQZZ_chardoff_x, &_QQZZ_chardoff_y);
01091
01092         return(0);
01093 }
01094
```

GMAIN.C

```
01094
01095 /*****************************************************************
01096  *                                                               *
01097  * 文字回転角の設定                                              *
01098  *                                                               *
01099  *****************************************************************/
01100 GLOBAL int
01101 gSetCharAngle(h_dx, h_dy, v_dx, v_dy)
01102 int             h_dx, h_dy;
01103 int             v_dx, v_dy;
01104 {
01105         if((h_dx == 0 && h_dy == 0) || (v_dx == 0 && v_dy == 0)) {
01106                 return(-1);
01107         }
01108
01109         _QQZZ_charangle_HX = h_dx;      /* 文字底辺回転角（x成分） */
01110         _QQZZ_charangle_HY = h_dy;      /* 文字底辺回転角（y成分） */
01111         _QQZZ_charangle_VX = v_dx;      /* 文字側辺回転角（x成分） */
01112         _QQZZ_charangle_VY = v_dy;      /* 文字側辺回転角（y成分） */
01113
01114         /* 変形指定の設定 */
01115         if(h_dx >= 0 && h_dy == 0 && v_dx == 0 && v_dy >= 0) {
01116                 /* 変形されない場合 */
01117                 _QQZZ_chartransf = 0;
01118         } else {
01119                 /* 変形される場合 */
01120                 _QQZZ_chartransf = 0;
01121         }
01122
01123         /* 水平方向倍率の計算と設定 */
01124         _QQZZ_charenlarge_H = _QQ_calcmagcode(CHR_SIZAH, _QQZZ_charsize_H,
01125                                         h_dx, h_dy, &_QQZZ_charmag_H);
01126         /* 垂直方向倍率の計算と設定 */
01127         _QQZZ_charenlarge_V = _QQ_calcmagcode(CHR_SIZV, _QQZZ_charsize_V,
01128                                         v_dx, v_dy, &_QQZZ_charmag_V);
01129
01130         /* 文字描画オフセットの計算 */
01131         _QQ_calcchardoff(&_QQZZ_chardoff_x, &_QQZZ_chardoff_y);
01132
01133         return(0);
01134 /*
01135         底辺・・文字のアウトラインボックスの下辺
01136         側辺・・文字のアウトラインボックスの左辺
01137         ベクトルは文字の左下隅を原点とする
01138 */
01139 }
01140
```

保守／廃止

GMAIN.C

```
01140
01141 /*****************************************************************
01142  *                                                               *
01143  * 文字列配置の設定                                              *
01144  *                                                               *
01145  *****************************************************************/
01146 GLOBAL VOID
01147 gSetTextAlignment(x, y, dist, half)
01148 int     x, y;
01149 int     dist;
01150 int     half;
01151 {
01152         _QQZZ_alignment_x = x;
01153         _QQZZ_alignment_y = y;
01154         _QQZZ_chardistance = dist;
01155         _QQZZ_halfadjust = half;
01156
01157 }
01158
```

GMAIN.C

```
01158
01159 /*****************************************************************
01160  *                                                               *
01161  * 拡大／縮小率コード計算                                        *
01162  *                                                               *
01163  *****************************************************************/
01164 static int
01165 _QQ_calcmagcode(sw, dw, rx, ry, mag)
01166 int     sw, dw;         /* ソースサイズ(sw), デストサイズ(dw) */
01167 int     rx, ry;         /* 回転ベクトル */
01168 int     *mag;           /* 拡大／縮小コード(0..F) */
01169 {
01170 extern  unsigned short _QQZZ_lsqrtr();
01171         register long   length;
01172         int             enlarge;
01173
01174         if(rx != 0 && ry != 0) {
01175                 /* 拡大／縮小倍率補正 */
01176                 length = _QQZZ_lsqrtr((long)rx*rx+(long)ry*ry);
01177                 if(abs(rx) >= abs(ry)) {
01178                         dw = (int)(((long)dw*abs(rx))/length);
01179                 } else {
01180                         dw = (int)(((long)dw*abs(ry))/length);
01181                 }
01182         }
01183         /* 倍率の計算 */
01184         if(dw >= sw) {
01185                 /* 拡大の場合 */
01186                 enlarge = 0;
01187                 *mag = (16*sw)/dw-1+(((16*sw)%dw) ? 1:0);
01188         } else {
01189                 /* 縮小の場合 */
01190                 enlarge = 0;
01191                 *mag = (16*dw)/sw-1;
01192         }
01193         if(*mag > 15) {
01194                 *mag = 15;
01195         } else if(*mag < 0) {
01196                 *mag = 0;
01197         }
01198
01199         return(enlarge);
01200 }
01201
```

GMAIN.C

```
01201
01202  /*****************************************************************
01203   *                                                               *
01204   * コピー回転角の設定                                            *
01205   *                                                               *
01206   *****************************************************************/
01207  GLOBAL int
01208  gSetCopyAngle(h_dx, h_dy, v_dx, v_dy)
01209  int             h_dx, h_dy;
01210  int             v_dx, v_dy;
01211  {
01212          if((h_dx == 0 && h_dy == 0) || (v_dx == 0 && v_dy == 0)) {
01213                  return(-1);
01214          }
01215
01216          _QQZZ_copyangle_HX = h_dx;      /* コピー底辺回転角（x成分） */
01217          _QQZZ_copyangle_HY = h_dy;      /* コピー底辺回転角（y成分） */
01218          _QQZZ_copyangle_VX = v_dx;      /* コピー側辺回転角（x成分） */
01219          _QQZZ_copyangle_VY = v_dy;      /* コピー側辺回転角（y成分） */
01220
01221          /* 変形指定の設定 */
01222          if(h_dx >= 0 && h_dy == 0 && v_dx == 0 && v_dy >= 0) {
01223                  /* 変形されない場合 */
01224                  _QQZZ_copytransf = 0;
01225          } else {
01226                  /* 変形される場合 */
01227                  _QQZZ_copytransf = 0;
01228          }
01229
01230          return(0);
01231  }
01232
```

GMAIN.C

```
01232
01233 /*****************************************************************
01234  *                                                               *
01235  * 画面消去                                                      *
01236  *                                                               *
01237  *****************************************************************/
01238 GLOBAL VOID
01239 gClearScreen()
01240 {
01241         v_clear(_QQZZ_Bcolor);
01242 }
01243
```

保守／廃止

GMAIN.C

```
01243
01244 /*****************************************************************
01245  *                                                               *
01246  * 1ドットの点の描画                                             *
01247  *                                                               *
01248  *****************************************************************/
01249 GLOBAL VOID
01250 gDot(x, y)
01251 int             x, y;
01252 {
01253         v_pset(x, y, _QQZZ_ColorCode, _QQZZ_DrawMode);
01254 }
01255
```

GMAIN.C

```
01255
01256 /*****************************************************************
01257  *                                                               *
01258  * グラフィックスペンによる点の描画                              *
01259  *                                                               *
01260  *****************************************************************/
01261 GLOBAL VOID
01262 gGpDot(x, y)
01263 int             x, y;
01264 {
01265         v_gpen_pset(x, y, x+1, y, _QQZZ_ColorCode, _QQZZ_DrawMode);
01266 }
01267
```

保守／廃止

GMAIN.C

```
01267
01268 /*****************************************************************
01269  *                                                               *
01270  * 1ドット幅の直線                                               *
01271  *                                                               *
01272  *****************************************************************/
01273 GLOBAL VOID
01274 gLine(xs, ys, xe, ye)
01275 int     xs, ys, xe, ye;
01276 {
01277         v_line(xs, ys, xe, ye, _QQZZ_ColorCode, _QQZZ_DrawMode, 1, 1);
01278 }
01279
```

GMAIN.C

```
01279
01280 /*****************************************************************
01281  *                                                               *
01282  * グラフィックスペンによる直線                                  *
01283  *                                                               *
01284  *****************************************************************/
01285 GLOBAL VOID
01286 gGpLine(xs, ys, xe, ye)
01287 int     xs, ys, xe, ye;
01288 {
01289         v_gpen_line(xs, ys, xe, ye, _QQZZ_ColorCode, _QQZZ_DrawMode, 1);
01290 }
01291
```

保守／廃止

GMAIN.C

```
01291
01292 /*****************************************************************
01293  *                                                               *
01294  * 1ドット幅の三角形の描画                                       *
01295  *                                                               *
01296  *****************************************************************/
01297 GLOBAL VOID
01298 gTriangle(pp)
01299 struct v_xy     *pp;            /* 頂点座標（3個）*/
01300 {
01301         v_line((pp+0)->x, (pp+0)->y, (pp+1)->x, (pp+1)->y,
01302                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0, 1);
01303         v_line((pp+1)->x, (pp+1)->y, (pp+2)->x, (pp+2)->y,
01304                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0, 0);
01305         v_line((pp+2)->x, (pp+2)->y, (pp+0)->x, (pp+0)->y,
01306                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0, 0);
01307 }
01308
```

保守／廃止

GMAIN.C

```
01308
01309 /*****************************************************************
01310  *                                                               *
01311  * グラフィックスペンによる三角形の描画                          *
01312  *                                                               *
01313  *****************************************************************/
01314 GLOBAL VOID
01315 gGpTriangle(pp)
01316 struct v_xy     *pp;            /* 頂点座標（3個）*/
01317 {
01318         v_gpen_line((pp+0)->x, (pp+0)->y, (pp+1)->x, (pp+1)->y,
01319                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01320         v_gpen_line((pp+1)->x, (pp+1)->y, (pp+2)->x, (pp+2)->y,
01321                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01322         v_gpen_line((pp+2)->x, (pp+2)->y, (pp+0)->x, (pp+0)->y,
01323                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01324 }
01325
```

保守／廃止

GMAIN.C

```
01325
01326 /*****************************************************************
01327  *                                                               *
01328  * 1ドット幅の多角形の描画                                       *
01329  *                                                               *
01330  *****************************************************************/
01331 GLOBAL VOID
01332 gPolygon(n, pp)
01333 int             n;
01334 struct v_xy     *pp;
01335 {
01336         register int    i;
01337
01338         if(n == 1) {
01339                 v_pset((pp+0)->x, (pp+0)->y, (pp+0)->x+1, (pp+0)->y,
01340                                         _QQZZ_ColorCode, _QQZZ_DrawMode);
01341         } else if(n == 2) {
01342                 v_line((pp+0)->x, (pp+0)->y, (pp+1)->x, (pp+1)->y,
01343                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 1, 1);
01344         } else if(n >= 3) {
01345                 for(i = 0; i < n-1; i += 1) {
01346                         v_line((pp+i)->x, (pp+i)->y, (pp+i+1)->x, (pp+i+1)->y,
01347                                         _QQZZ_ColorCode, _QQZZ_DrawMode,
01348                                         0, (i == 0) ? 1 : 0);
01349                 }
01350                 v_line((pp+i)->x, (pp+i)->y, (pp+0)->x, (pp+0)->y,
01351                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0, 0);
01352         }
01353 }
01354
```

GMAIN.C

```
01354
01355 /*****************************************************************
01356  *                                                               *
01357  * グラフィックスペンによる多角形の描画                          *
01358  *                                                               *
01359  *****************************************************************/
01360 GLOBAL VOID
01361 gGpPolygon(n, pp)
01362 int             n;
01363 struct v_xy     *pp;
01364 {
01365         register int    i;
01366
01367         if(n == 1) {
01368                 v_gpen_pset((pp+0)->x, (pp+0)->y, (pp+0)->x+1, (pp+0)->y,
01369                                 _QQZZ_ColorCode, _QQZZ_DrawMode);
01370         } else if(n == 2) {
01371                 v_gpen_line((pp+0)->x, (pp+0)->y, (pp+1)->x, (pp+1)->y,
01372                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 1);
01373         } else if(n >= 3) {
01374                 for(i = 0; i < n-1; i += 1) {
01375                         v_gpen_line((pp+i)->x, (pp+i)->y,
01376                                 (pp+i+1)->x, (pp+i+1)->y,
01377                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01378                 }
01379                 v_gpen_line((pp+i)->x, (pp+i)->y, (pp+0)->x, (pp+0)->y,
01380                                 _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01381         }
01382 }
01383
```

GMAIN.C

```
01383
01384 /*****************************************************************
01385  *                                                               *
01386  * 1ドット幅のボックスの描画                                     *
01387  *                                                               *
01388  *****************************************************************/
01389 GLOBAL VOID
01390 gRectangle(xs, ys, xe, ye)
01391 int     xs, ys, xe, ye;
01392 {
01393         v_box(xs, ys, xe, ye, _QQZZ_ColorCode, _QQZZ_DrawMode);
01394 }
01395
```

GMAIN.C

```
01395
01396 /*****************************************************************
01397  *                                                               *
01398  * グラフィックスペンによるボックスの描画                        *
01399  *                                                               *
01400  *****************************************************************/
01401 GLOBAL VOID
01402 gGpRectangle(xs, ys, xe, ye)
01403 int     xs, ys, xe, ye;
01404 {
01405         v_gpen_line(xs, ys, xe, ys, _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01406         v_gpen_line(xe, ys, xe, ye, _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01407         v_gpen_line(xe, ye, xs, ye, _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01408         v_gpen_line(xs, ye, xs, ys, _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01409 }
01410
```

GMAIN.C

```
01410
01411 /*****************************************************************
01412  *                                                               *
01413  * 1ドット幅の円の描画                                           *
01414  *                                                               *
01415  *****************************************************************/
01416 GLOBAL VOID
01417 gCircle(x, y, r)
01418 int     x, y, r;
01419 {
01420         v_circle(x, y, r, _QQZZ_ColorCode, _QQZZ_DrawMode);
01421 }
01422
```

保守／廃止

GMAIN.C

```
01422
01423 /*****************************************************************
01424  *                                                               *
01425  * グラフィックスペンによる円の描画                              *
01426  *                                                               *
01427  *****************************************************************/
01428 GLOBAL VOID
01429 gGpCircle(x, y, r)
01430 int     x, y, r;
01431 {
01432         v_gpen_circle(x, y, r, _QQZZ_ColorCode, _QQZZ_DrawMode);
01433 }
01434
```

保守／廃止

GMAIN.C

```
01434
01435 /*****************************************************************
01436  *                                                               *
01437  * 1ドット幅の円弧の描画                                        *
01438  *                                                               *
01439  *****************************************************************/
01440 GLOBAL VOID
01441 gCircArc(x, y, r, xs, ys, xe, ye, dir, type)
01442 int     x, y, r, xs, ys, xe, ye, dir, type;
01443 {
01444         v_circ_arc(x,y,r,_QQZZ_ColorCode,_QQZZ_DrawMode,xs,ys,xe,ye,dir,type);
01445         _QQ_arc_se(&_QQZZ_Tip_sx, &_QQZZ_Tip_sy, &_QQZZ_Tip_ex, &_QQZZ_Tip_ey);
01446 }
01447
```

GMAIN.C

```
01447
01448 /*************************************************************
01449  *                                                           *
01450  * グラフィックスペンによる円弧の描画                        *
01451  *                                                           *
01452  *************************************************************/
01453 GLOBAL VOID
01454 gGpCircArc(x, y, r, xs, ys, xe, ye, dir, type)
01455 int     x, y, r, xs, ys, xe, ye, dir, type;
01456 {
01457         int     xsp, ysp, xep, yep;
01458         switch(type) {
01459                 case 0: /* 弧 */
01460                         v_gpen_circ_arc(x,y, r, _QQZZ_ColorCode,_QQZZ_DrawMode,
01461                                                         xs,ys, xe,ye, dir, 1);
01462                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01463                         break;
01464                 case 1: /* 扇 */
01465                         v_gpen_circ_arc(x,y, r, _QQZZ_ColorCode,_QQZZ_DrawMode,
01466                                                         xs,ys, xe,ye, dir, 0);
01467                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01468                         v_gpen_line(xep, yep, x, y,
01469                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01470                         v_gpen_line(x, y, xsp, ysp,
01471                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01472                         break;
01473                 case 2: /* 弦 */
01474                         v_gpen_circ_arc(x,y, r, _QQZZ_ColorCode,_QQZZ_DrawMode,
01475                                                         xs,ys, xe,ye, dir, 0);
01476                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01477                         v_gpen_line(xep, yep, xsp, ysp,
01478                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01479                         break;
01480         }
01481         _QQZZ_Tip_sx = xsp;
01482         _QQZZ_Tip_sy = ysp;
01483         _QQZZ_Tip_ex = xep;
01484         _QQZZ_Tip_ey = yep;
01485 }
01486
```

保守／廃止

GMAIN.C

```
01486
01487 /*****************************************************************
01488  *                                                               *
01489  * 楕円用DH,DV計算                                              *
01490  *                                                               *
01491  *****************************************************************/
01492 static VOID
01493 calDH_DV(rx, ry, dh, dv)
01494 int     rx, ry;
01495 int     *dh, *dv;
01496 {
01497 static  u_long  upper0_mask[] = {
01498                 0xffffffffL,    0xfffffffeL,    0xfffffffcL,    0xfffffff8L,
01499                 0xfffffff0L,    0xffffffe0L,    0xffffffc0L,    0xffffff80L,
01500                 0xffffff00L,    0xfffffe00L,    0xfffffc00L,    0xfffff800L,
01501                 0xfffff000L,    0xffffe000L,    0xfffc000L,     0xffff8000L,
01502                 0xffff0000L,    0xfffe0000L,    0xfffc0000L,    0xfff80000L,
01503                 0xfff00000L,    0xffe00000L,    0xffc00000L,    0xff800000L,
01504                 0xff000000L,    0xfe000000L,    0xfc000000L,    0xf8000000L,
01505                 0xf0000000L,    0xe0000000L,    0xc0000000L,    0x80000000L,
01506         };
01507         register int    mask_no;
01508         register int    shift_cnt;
01509         u_long  t_dh, t_dv;
01510
01511         t_dh = (u_long)rx*(u_long)rx;
01512         t_dv = (u_long)ry*(u_long)ry;
01513         for(mask_no = 0; mask_no < 32; mask_no += 1) {
01514                 register u_long mask;
01515                 mask = upper0_mask[mask_no];
01516                 if(((t_dh & mask)==0) && ((t_dv & mask)==0)) {
01517                         break;
01518                 }
01519         }
01520         if(shift_cnt > 8) {
01521                 shift_cnt = mask_no-8;
01522                 *dh = (int)(t_dh >> shift_cnt);
01523                 *dv = (int)(t_dv >> shift_cnt);
01524         } else {
01525                 *dh = (int)t_dh;
01526                 *dv = (int)t_dv;
01527         }
01528 }
01529
```

GMAIN.C

```
01529
01530 /*****************************************************************
01531  *                                                               *
01532  * 1ドット幅の楕円の描画                                        *
01533  *                                                               *
01534  *****************************************************************/
01535 GLOBAL VOID
01536 gEllipse(x, y, rx, ry)
01537 int     x, y, rx, ry;
01538 {
01539         int     dh, dv;
01540         calDH_DV(rx, ry, &dh, &dv);
01541         v_ellipse(x, y, rx, ry, dh, dv, _QQZZ_ColorCode, _QQZZ_DrawMode);
01542 }
01543
```

保守／廃止

GMAIN.C

```
01543
01544 /*****************************************************************
01545  *                                                               *
01546  * グラフィックスペンによる楕円の描画                            *
01547  *                                                               *
01548  *****************************************************************/
01549 GLOBAL VOID
01550 gGpEllipse(x, y, rx, ry)
01551 int     x, y, rx, ry;
01552 {
01553         int     dh, dv;
01554         calDH_DV(rx, ry, &dh, &dv);
01555         v_gpen_ellipse(x, y, rx, ry, dh, dv, _QQZZ_ColorCode, _QQZZ_DrawMode);
01556 }
01557
```

GMAIN.C

```
01557
01558 /*****************************************************************
01559  *                                                               *
01560  * 1ドット幅の楕円弧の描画                                      *
01561  *                                                               *
01562  *****************************************************************/
01563 GLOBAL VOID
01564 gElpsArc(x, y, rx, ry, xs, ys, xe, ye, dir, type)
01565 int     x, y, rx, ry, xs, ys, xe, ye, dir, type;
01566 {
01567         int     dh, dv;
01568         calDH_DV(rx, ry, &dh, &dv);
01569         v_elps_arc(x,y, rx,ry, dh,dv, _QQZZ_ColorCode, _QQZZ_DrawMode,
01570                                                 xs,ys, xe,ye, dir, type);
01571         _QQ_arc_se(&_QQZZ_Tip_sx, &_QQZZ_Tip_sy, &_QQZZ_Tip_ex, &_QQZZ_Tip_ey);
01572 }
01573
```

62

保守／廃止

GMAIN.C

```
01573
01574 /*************************************************************
01575  *                                                           *
01576  * グラフィックスペンによる楕円弧の描画                      *
01577  *                                                           *
01578  *************************************************************/
01579 GLOBAL VOID
01580 gGpElpsArc(x, y, rx, ry, xs, ys, xe, ye, dir, type)
01581 int     x, y, rx, ry, xs, ys, xe, ye, dir, type;
01582 {
01583         int     dh, dv;
01584         int     xsp, ysp, xep, yep;
01585
01586         calDH_DV(rx, ry, &dh, &dv);
01587         switch(type) {
01588                 case 0: /* 弧 */
01589                         v_gpen_elps_arc(x,y, rx,ry, dh,dv,
01590                                         _QQZZ_ColorCode, _QQZZ_DrawMode,
01591                                         xs,ys, xe,ye, dir, 1);
01592                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01593                         break;
01594                 case 1: /* 扇 */
01595                         v_gpen_elps_arc(x,y, rx,ry, dh,dv,
01596                                         _QQZZ_ColorCode, _QQZZ_DrawMode,
01597                                         xs,ys, xe,ye, dir, 0);
01598                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01599                         v_gpen_line(xep, yep, x, y,
01600                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01601                         v_gpen_line(x, y, xsp, ysp,
01602                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01603                         break;
01604                 case 2: /* 弦 */
01605                         v_gpen_elps_arc(x,y, rx,ry, dh,dv,
01606                                         _QQZZ_ColorCode, _QQZZ_DrawMode,
01607                                         xs,ys, xe,ye, dir, 0);
01608                         _QQ_arc_se(&xsp, &ysp, &xep, &yep);
01609                         v_gpen_line(xep, yep, xsp, ysp,
01610                                         _QQZZ_ColorCode, _QQZZ_DrawMode, 0);
01611                         break;
01612         }
01613         _QQZZ_Tip_sx = xsp;
01614         _QQZZ_Tip_sy = ysp;
01615         _QQZZ_Tip_ex = xep;
01616         _QQZZ_Tip_ey = yep;
01617 }
01618
```

GMAIN.C

```
01618
01619 /*****************************************************************
01620  *                                                               *
01621  * 塗りつぶし                                                    *
01622  *                                                               *
01623  *****************************************************************/
01624 GLOBAL VOID
01625 gPaint(x, y, border)
01626 int     x, y, border;
01627 {
01628         v_paint(x, y, _QQZZ_ColorCode, border);
01629 }
01630
```

保守／廃止

GMAIN.C

```
01630
01631  /*****************************************************************
01632   *                                                               *
01633   * 塗り直し                                                      *
01634   *                                                               *
01635   *****************************************************************/
01636  GLOBAL VOID
01637  gChangeColor(x, y)
01638  int      x, y;
01639  {
01640          v_n_paint(x, y, _QQZZ_ColorCode);
01641  }
01642
```

保守／廃止

GMAIN.C

```
01642
01643 /*****************************************************************
01644  *                                                               *
01645  * 三角形塗りつぶし                                              *
01646  *                                                               *
01647  *****************************************************************/
01648 GLOBAL VOID
01649 gFillTriangle(pp)
01650 struct v_xy     *pp;
01651 {
01652         v_tri_fill((pp+0)->x, (pp+0)->y, (pp+1)->x, (pp+1)->y,
01653                 (pp+2)->x, (pp+2)->y,_QQZZ_ColorCode, _QQZZ_DrawMode);
01654 }
01655
```

保守／廃止

GMAIN.C

```
01655
01656 /*****************************************************************
01657  *                                                               *
01658  * ボックスの塗りつぶし                                          *
01659  *                                                               *
01660  *****************************************************************/
01661 GLOBAL VOID
01662 gFillRectangle(x1, y1, x2, y2)
01663 int     x1, y1, x2, y2;
01664 {
01665         v_boxf(x1, y1, x2, y2, _QQZZ_ColorCode, _QQZZ_DrawMode);
01666 }
01667
```

GMAIN.C

```
01667
01668 /*****************************************************************
01669  *                                                               *
01670  * 多角形の塗りつぶし                                            *
01671  *                                                               *
01672  *****************************************************************/
01673 GLOBAL VOID
01674 gFillPolygon(n, pp)
01675 int             n;
01676 struct v_xy     *pp;
01677 {
01678         v_polygon_fill(n, pp, _QQZZ_ColorCode, _QQZZ_DrawMode);
01679 }
01680
```

保守／廃止

GMAIN.C

```
01680
01681 /*****************************************************************
01682  *                                                               *
01683  * 円の塗りつぶし                                                *
01684  *                                                               *
01685  *****************************************************************/
01686 GLOBAL VOID
01687 gFillCircle(x, y, r)
01688 int     x, y, r;
01689 {
01690         v_circ_fill(x,y, r, _QQZZ_ColorCode, _QQZZ_DrawMode);
01691 }
01692
```

保守／廃止

GMAIN.C

```
01692
01693  /*****************************************************************
01694   *                                                               *
01695   * 円弧の塗りつぶし                                              *
01696   *                                                               *
01697   *****************************************************************/
01698  GLOBAL VOID
01699  gFillCircArc(x, y, r, xs, ys, xe, ye, dir, type)
01700  int      x, y, r, xs, ys, xe, ye, dir, type;
01701  {
01702          v_circ_arc_fill(x,y, r, _QQZZ_ColorCode, _QQZZ_DrawMode, xs,ys, xe,ye, dir, t
       ype);
01703          _QQ_arc_se(&_QQZZ_Tip_sx, &_QQZZ_Tip_sy, &_QQZZ_Tip_ex, &_QQZZ_Tip_ey);
01704  }
01705
```

GMAIN.C

```
01705
01706 /*****************************************************************
01707  *                                                               *
01708  * 楕円の塗りつぶし                                              *
01709  *                                                               *
01710  *****************************************************************/
01711 GLOBAL VOID
01712 gFillEllipse(x, y, rx, ry)
01713 int     x, y, rx, ry;
01714 {
01715         int     dh,dv;
01716         calDH_DV(rx, ry, &dh, &dv);
01717         v_elps_fill(x,y, rx, ry, dh, dv, _QQZZ_ColorCode, _QQZZ_DrawMode);
01718 }
01719
```

GMAIN.C

```
01719
01720 /*****************************************************************
01721  *                                                               *
01722  * 楕円弧の塗りつぶし                                            *
01723  *                                                               *
01724  *****************************************************************/
01725 GLOBAL VOID
01726 gFillElpsArc(x, y, rx, ry, xs, ys, xe, ye, dir, type)
01727 int     x, y, rx, ry, xs, ys, xe, ye, dir, type;
01728 {
01729         int     dh, dv;
01730         calDH_DV(rx, ry, &dh, &dv);
01731         v_elps_arc_fill(x,y, rx,ry, dh,dv, _QQZZ_ColorCode, _QQZZ_DrawMode,
01732                                                 xs,ys,xe,ye, dir, type);
01733         _QQ_arc_se(&_QQZZ_Tip_sx, &_QQZZ_Tip_sy, &_QQZZ_Tip_ex, &_QQZZ_Tip_ey);
01734 }
01735
```

GMAIN.C

```
01735
01736 /*****************************************************************
01737  *                                                               *
01738  * ３オペランドコピー                                            *
01739  *                                                               *
01740  *****************************************************************/
01741 GLOBAL VOID
01742 g3opCopy(sscr, sx, sy, mscr, mx, my, dx, dy, xc, yc, type, p)
01743 int             sscr;
01744 int             sx, sy;
01745 int             mscr;
01746 int             mx, my;
01747 int             dx, dy;
01748 int             xc, yc;
01749 int             type;
01750 int             p;
01751 {
01752         _QQZZ_3ope_copy_AC(sscr,sx,sy+(yc-1), mscr,mx,my+(yc-1), dx,dy+(yc-1),
01753                         xc,yc, type,_QQZZ_ColorCode,_QQZZ_DrawMode, p);
01754 }
01755
```

保守／廃止

GMAIN.C

```
01755
01756 /*****************************************************************
01757  *                                                               *
01758  * 画面コピー                                                    *
01759  *                                                               *
01760  *****************************************************************/
01761 gCopyScr(sscr, sx, sy, sw, sh, dx, dy, dw, dh, p)
01762 int             sscr;
01763 int             sx, sy;
01764 int             sw, sh;
01765 int             dx, dy;
01766 int             dw, dh;
01767 int             p;
01768 {
01769         SCRTBL  *ss;
01770
01771         ss = &_QQZZ_screen[_QQZZ_scrnid[sscr]];
01772         if(_QQZZ_CURscr != sscr) {
01773                 /* 転送元画面と転送先画面が異なる場合 */
01774                 v_2scrnmightycopy(sscr, sx, sy, sw, sh, dx, dy, dw, dh,
01775                                 _QQZZ_ColorCode, _QQZZ_DrawMode, p);
01776                 return;
01777         }
01778
01779         /* 転送元画面と転送先画面が同じ場合 */
01780         if(sw == dw && sh == dh && !_QQZZ_copytransf) {
01781                 /* 拡大／縮小及び変形が行われない場合 */
01782                 v_scrnstratecopy(sscr, sx, sy, sw, sh, dx, dy,
01783                                 _QQZZ_ColorCode, _QQZZ_DrawMode, p);
01784                 return;
01785         }
01786         /* 転送元画面と転送先画面が同じで拡大／縮小または変形が行われる場合 */
01787         {
01788                 int     oldscrn_no;     /* カレントだった画面の番号 */
01789                 int     enlarge_h, enlarge_v;
01790                 int     mag_h, mag_v;
01791                 int     doff_hx, doff_hy, doff_vx, doff_vy;
01792                 int     doff_cx, doff_cy;
01793                 int     copyr_l, copyr_r, copyr_u, copyr_d;
01794                 int     fsx, fsy, fsw, fsh;     /* 最終的なコピー範囲 */
01795                 oldscrn_no = _QQZZ_CURscr;
01796                 v_act_page(_QQZZ_WRK1scr);
01797                 if(p >= 0) {
01798                         _v_put_reg(R_PMAX, 1);
01799                 }
01800                 /* 転送先のコピー範囲の計算 */
01801                 enlarge_h = _QQ_calcmagcode(sw, dw,
01802                                 _QQZZ_copyangle_HX,_QQZZ_copyangle_HY,&mag_h);
01803                 _QQ_calccopydoff(enlarge_h, mag_h, sw, _QQZZ_copytransf,
01804                                 _QQZZ_copyangle_HX, _QQZZ_copyangle_HY,
01805                                 &doff_hx, &doff_hy);
01806                 enlarge_v = _QQ_calcmagcode(sh, dh,
01807                                 _QQZZ_copyangle_VX,_QQZZ_copyangle_VY,&mag_v);
01808                 _QQ_calccopydoff(enlarge_v, mag_v, sh, _QQZZ_copytransf,
01809                                 _QQZZ_copyangle_VX, _QQZZ_copyangle_VY,
01810                                 &doff_vx, &doff_vy);
01811                 doff_cx = doff_hx+doff_vx;
01812                 doff_cy = doff_hy+doff_vy;
01813                 copyr_l = (doff_hx>doff_vx) ? doff_vx:doff_hx;
01814                 copyr_l = (copyr_l>doff_cx) ? doff_cx:copyr_l;
01815                 copyr_l = (copyr_l<0) ? copyr_l+dx:dx;
01816                 copyr_d = (doff_hy>doff_vy) ? doff_vy:doff_hy;
01817                 copyr_d = (copyr_d>doff_cy) ? doff_cy:copyr_d;
01818                 copyr_d = (copyr_d<0) ? copyr_d+dy:dy;
01819                 copyr_r = (doff_hx>doff_vx) ? doff_hx:doff_vx;
01820                 copyr_r = (copyr_r<doff_cx) ? doff_cx:copyr_r;
01821                 copyr_r = (copyr_r<0) ? dx:copyr_r+dx;
01822                 copyr_u = (doff_hy>doff_vy) ? doff_hy:doff_vy;
01823                 copyr_u = (copyr_u<doff_cy) ? doff_cy:copyr_u;
```

保守／廃止

GMAIN.C

```
01824                   copyr_u = (copyr_u<0) ? dy:copyr_u+dy;
01825                   /* コピー結果を作る */
01826                   v_2scrnmightycopy(oldscrn_no, sx, sy, sw, sh, dx, dy, dw, dh,
01827                                   0x00, 0x00, p);
01828                   if(_QQZZ_copyangle_HX*_QQZZ_copyangle_HY ||
01829                           _QQZZ_copyangle_VX*_QQZZ_copyangle_VY) {
01830                           /* コピー結果が矩形にならない変形が行われる場合 */
01831                           int     copyr_lw, copyr_rw;
01832                           copyr_lw= copyr_l & (~0xf);
01833                           copyr_rw= (copyr_r & (~0xf))+15;
01834                           /* コピーマスクを作る */
01835                           v_act_page(_QQZZ_WRK0scr);
01836                           v_boxf(copyr_lw,copyr_d, copyr_rw,copyr_u, 1, 0x22);
01837                           v_2scrnmightycopy(oldscrn_no, sx, sy, sw, sh, dx, dy,
01838                                           dw, dh, 0x00, 0x33, 0);
01839                           v_act_page(oldscrn_no);
01840                           fsx = copyr_lw;
01841                           fsy = copyr_u;
01842                           fsw = copyr_rw-copyr_lw+1;
01843                           fsh = copyr_u-copyr_d+1;
01844                           _QQZZ_3ope_copy_AC(_QQZZ_WRK1scr, fsx, fsy,
01845                                           _QQZZ_WRK0scr, fsx, fsy,
01846                                           fsx, fsy, fsw, fsh, 1,
01847                                           _QQZZ_ColorCode, _QQZZ_DrawMode, p);
01848                   } else {
01849                           /* 変形されても結果が矩形になる場合 */
01850                           fsx = copyr_l;
01851                           fsy = copyr_d;
01852                           fsw = copyr_r-copyr_l+1;
01853                           fsh = copyr_u-copyr_d+1;
01854                           v_act_page(oldscrn_no);
01855                           v_scrnstratecopy(_QQZZ_WRK1scr, fsx, fsy, fsw, fsh,
01856                                           fsx, fsy,
01857                                           _QQZZ_ColorCode, _QQZZ_DrawMode, p);
01858                   }
01859           }
01860           return;
01861   }
01862
```

保守／廃止

GMAIN.C

```
01862
01863 /*************************************************************
01864  *                                                           *
01865  * ２画面間コピー(All mighty)                                *
01866  *                                                           *
01867  *************************************************************/
01868 GLOBAL VOID
01869 v_2scrnmightycopy(sscr, sx, sy, sw, sh, dx, dy, dw, dh, c, mod, single)
01870 int             sscr, sx, sy;
01871 int             sw, sh;
01872 int             dx, dy;
01873 int             dw, dh;
01874 int             c, mod;
01875 int             single;
01876 {
01877         int     enlarge_H, enlarge_V;
01878         int     mag_H, mag_V;
01879         int     doff_x, doff_y;
01880
01881         u_int   flag;
01882         int     arg[4];
01883         int     px, py;                 /* AGDCに与える描画位置 */
01884         int     halfwidth;
01885         int     wd, hi;
01886         u_long  top;
01887         int     boff;
01888         int     pitch;
01889         u_long  disp;
01890         int     src_no;
01891         SCRTBL  *src;
01892
01893         /* 垂直方向倍率の計算と設定 */
01894         enlarge_V = _QQ_calcmagcode(sh, dh,
01895                         _QQZZ_copyangle_VX, _QQZZ_copyangle_VY, &mag_V);
01896         /* 水平方向倍率の計算と設定 */
01897         enlarge_H = _QQ_calcmagcode(sw, dw,
01898                         _QQZZ_copyangle_HX, _QQZZ_copyangle_HY, &mag_H);
01899
01900         /* 描画オフセットの計算 */
01901         _QQ_calccopydoff(enlarge_V, mag_V, sh, _QQZZ_copytransf,
01902                                 _QQZZ_copyangle_VX, _QQZZ_copyangle_VY,
01903                                 &doff_x, &doff_y);
01904         /* コピーコマンド設定 */
01905         flag = FRES_COPY;
01906         arg[0] =  _QQZZ_copyangle_HX;
01907         arg[1] =  _QQZZ_copyangle_HY;
01908         arg[2] = -_QQZZ_copyangle_VX;
01909         arg[3] = -_QQZZ_copyangle_VY;
01910         if(!_QQZZ_copytransf) {
01911                 /* 変形されない場合 */
01912                 flag = NORM_COPY;
01913         }
01914         if(single < 0) {
01915                 /* マルチソース指定 */
01916                 flag |= SR_MULTI;
01917         }
01918
01919         /* 描画位置計算 */
01920         px = dx+doff_x;
01921         py = dy+doff_y;
01922
01923         /* 拡大／縮小の設定 */
01924         if(mag_H == 0xf && mag_V == 0xf) {
01925                 flag |= NO_MAG;
01926         }
01927
01928         /* 転送源情報作成 */
01929         src_no = _QQ_get_scraddr(sscr, &disp, &top);
01930         if(src_no < 0) {
```

GMAIN.C

```
01931                   return(-1);
01932           }
01933           src = &_QQZZ_screen[src_no];
01934           pitch = src->scr_pitch;
01935           if(single >= 0) {
01936                   top += (u_long)src->scr_displace*single;
01937           }
01938           top += (u_long)(src->scr_height-1-(sy+sh-1))*src->scr_pitch+sx/16;
01939           boff = sx & 0xf;
01940
01941           /* コピー */
01942           _QQ_mightycopy(top, boff, sw, sh, pitch, disp, px, py,
01943                           enlarge_H, mag_H, enlarge_V, mag_V, c, mod, flag, arg);
01944
01945           return(0);
01946   }
01947
```

保守／廃止

GMAIN.C

```
01947
01948 /*****************************************************************
01949  *                                                               *
01950  * 単純画面コピー(Strate & Fast)                                 *
01951  *                                                               *
01952  *****************************************************************/
01953 GLOBAL VOID
01954 v_scrnstratecopy(sscr, sx, sy, sw, sh, dx, dy, c, mod, single)
01955 int     sscr, sx, sy;   /* 転送源画面番号と転送開始位置（左下隅） */
01956 int     sw, sh;         /* 転送源サイズ */
01957 int     dx, dy;         /* 転送先位置（左下隅） */
01958 int     single;         /* プレーン転送形式 負=M->M,正（プレーン番号）=S->M */
01959 int     c, mod;         /* カラー及び演算指定 */
01960 {
01961         u_int   flag;
01962         int     px, py;                 /* AGDCに与える描画位置 */
01963         int     wd, hi;
01964         u_long  top;
01965         int     boff;
01966         int     pitch;
01967         u_long  disp;
01968         int     src_no;
01969         SCRTBL  *src;
01970
01971         /* コピーコマンド設定 */
01972         flag = NORM_COPY | NO_MAG | FAST_BIT;
01973         if(single < 0) {
01974                 /* マルチソース指定 */
01975                 flag |= SR_MULTI;
01976         }
01977
01978         /* 転送源情報作成 */
01979         src_no = _QQ_get_scraddr(sscr, &disp, &top);
01980         if(src_no < 0) {
01981                 return(-1);
01982         }
01983         src = &_QQZZ_screen[src_no];
01984         pitch = src->scr_pitch;
01985         if(single >= 0) {
01986                 top += (u_long)src->scr_displace*single;
01987         }
01988
01989         if(sscr == _QQZZ_CURscr && (sy > dy || (sy == dy && sx < dx))){
01990                 /* 右下側から転送する場合 */
01991                 flag |= ESE_BIT | ROT_BIT;
01992                 top += (u_long)(src->scr_height-1-sy)*src->scr_pitch
01993                         +(sx+sw-1)/16;
01994                 boff = (sx+sw-1) & 0xf;
01995                 px = dx+sw-1;
01996                 py = dy;
01997         } else {
01998                 /* 左上側から転送する場合 */
01999                 top += (u_long)(src->scr_height-1-(sy+sh-1))*src->scr_pitch
02000                         +sx/16;
02001                 boff = sx & 0xf;
02002                 px = dx;
02003                 py = dy+sh-1;
02004         }
02005
02006         /* コピー */
02007         _QQ_stratecopy(top, boff, sw, sh, pitch, disp, px, py, c, mod, flag);
02008
02009         return(0);
02010 }
02011
```

GMAIN.C

```
02011
02012 /*****************************************************************
02013  *                                                               *
02014  * 画面情報の読み込み                                            *
02015  *                                                               *
02016  *****************************************************************/
02017 GLOBAL VOID
02018 gGetImage(x, y, xc, yc, buf)
02019 int             x, y;
02020 int             xc, yc;
02021 u_short *buf;
02022 {
02023         int     pitch;
02024         u_long  disp;
02025
02026         if (xc <= 0 || yc <= 0) {
02027                 return(-1);
02028         }
02029         disp = (u_long)((xc+15)/16);
02030         pitch = _QQZZ_screen->scr_planecnt*(int)disp;
02031         _QQZZ_get(x, y+(yc-1), xc,yc, buf, pitch, disp);
02032 }
02033
```

GMAIN.C

```
02033
02034 /*****************************************************************
02035  *                                                               *
02036  * 画面情報の書き込み                                            *
02037  *                                                               *
02038  *****************************************************************/
02039 GLOBAL VOID
02040 gPutImage(x, y, xc, yc, buf, pitch, disp, p)
02041 int             x, y;
02042 int             xc, yc;
02043 int             *buf;
02044 int             pitch;
02045 u_long  disp;
02046 int             p;
02047 {
02048         int     dp;
02049
02050         if(xc <= 0 || yc <= 0) {
02051                 return(-1);
02052         }
02053         if(p < 0) {
02054                 /* M->M */
02055                 dp = _QQZZ_CURscreen->scr_planecnt;
02056         } else {
02057                 /* S->M */
02058                 dp = 1;
02059         }
02060         _QQZZ_put(x, y+(yc-1), xc, yc, dp, buf, pitch, disp,
02061                         _QQZZ_ColorCode, _QQZZ_DrawMode, p);
02062 }
02063
```

保守／廃止

GMAIN.C

```
02063
02064 /*****************************************************************
02065  *                                                               *
02066  * 文字列の描画                                                  *
02067  *                                                               *
02068  *****************************************************************/
02069 gPutText(x, y, text)
02070 int             x, y;
02071 u_char  *text;
02072 {
02073         u_int   flag;
02074         int     arg[4];
02075         int     halfwidth;
02076         int     wd, hi;
02077         u_long  fonttop;
02078         int     pitch;
02079         u_long  disp;
02080         u_short code;
02081         u_char  *tp;
02082         int     px, py;                 /* 描画位置 */
02083         u_short limitdist;              /* 描画限界距離 */
02084         u_short drawdist;               /* 描画距離 */
02085         u_short lsqrt1();
02086         u_int   shift2jis();
02087
02088         /* コピーコマンド設定 */
02089         flag = FRES_COPY;
02090         arg[0] =  _QQZZ_charangle_HX;
02091         arg[1] =  _QQZZ_charangle_HY;
02092         arg[2] = -_QQZZ_charangle_VX;
02093         arg[3] = -_QQZZ_charangle_VY;
02094         if(!_QQZZ_chartransf) {
02095                 /* 変形されない場合 */
02096                 flag = NORM_COPY;
02097         }
02098
02099         /* 拡大／縮小の設定 */
02100         if(_QQZZ_charmag_H == 0xf && _QQZZ_charmag_V == 0xf) {
02101                 flag |= NO_MAG;
02102         }
02103
02104 #define sqare(i)        ((long)(i)*(long)(i))
02105 #define dposx(p)        ((p)+_QQZZ_chardoff_x)
02106 #define dposy(p)        ((p)+_QQZZ_chardoff_y)
02107         tp = text;
02108         drawdist = 0;
02109         limitdist = _QQZZ_lsqrtr(sqare(_QQZZ_alignment_x)
02110                         +sqare(_QQZZ_alignment_y));
02111         while ((code = *tp++) != '¥0' && drawdist <= limitdist) {
02112                 /* 描画する文字のフォント情報を得る */
02113                 if (iskanji(code)) {
02114                         code <<= 8;
02115                         code |= *tp++;
02116                         code = shift2jis(code);
02117                 }
02118                 halfwidth = _QQ_code2addr(code, &wd, &hi, &fonttop, &pitch);
02119                 /* 描画位置計算 */
02120                 _QQZZ_pipoint_all(drawdist,_QQZZ_alignment_x,_QQZZ_alignment_y,
02121                                                         &px, &py);
02122                 /* 描画 */
02123                 _QQ_mightycopy(fonttop, 0, wd, hi, pitch, FONTDISP,
02124                                 dposx(x+px), dposy(y+py),
02125                                 _QQZZ_charenlarge_H, _QQZZ_charmag_H,
02126                                 _QQZZ_charenlarge_V, _QQZZ_charmag_V,
02127                                 _QQZZ_ColorCode, _QQZZ_DrawMode, flag, arg);
02128                 /* 描画距離の更新 */
02129                 if(_QQZZ_halfadjust) {
02130                         if(halfwidth) {
02131                                 drawdist += _QQZZ_chardistance;
```

81

GMAIN.C

```
02132                           } else {
02133                                   drawdist += 2*_QQZZ_chardistance;
02134                           }
02135                   } else {
02136                           drawdist += 2*_QQZZ_chardistance;
02137                   }
02138           }
02139   }
02140
```

GMAIN.C

```
02140
02141 /*****************************************************************
02142  *                                                               *
02143  * 文字の描画                                                    *
02144  *                                                               *
02145  *****************************************************************/
02146 gPutChar(x, y, code)
02147 int             x, y;
02148 u_short code;
02149 {
02150         u_int   flag;
02151         int     arg[4];
02152         int     px, py;                 /* AGDCに与える描画位置 */
02153         int     halfwidth;
02154         int     wd, hi;
02155         u_long  fonttop;
02156         int     pitch;
02157
02158         /* コピーコマンド設定 */
02159         flag = FRES_COPY;
02160         arg[0] =  _QQZZ_charangle_HX;
02161         arg[1] =  _QQZZ_charangle_HY;
02162         arg[2] = -_QQZZ_charangle_VX;
02163         arg[3] = -_QQZZ_charangle_VY;
02164         if(!_QQZZ_chartransf) {
02165                 /* 変形されない場合 */
02166                 flag = NORM_COPY;
02167         }
02168
02169         /* 描画位置計算 */
02170         px = x+_QQZZ_chardoff_x;
02171         py = y+_QQZZ_chardoff_y;
02172
02173         /* 拡大／縮小の設定 */
02174         if(_QQZZ_charmag_H == 0xf && _QQZZ_charmag_V == 0xf) {
02175                 flag |= NO_MAG;
02176         }
02177
02178         /* 描画 */
02179         halfwidth = _QQ_code2addr(code, &wd, &hi, &fonttop, &pitch);
02180         _QQ_mightycopy(fonttop, 0, wd, hi, pitch, FONTDISP, px, py,
02181                                 _QQZZ_charenlarge_H, _QQZZ_charmag_H,
02182                                 _QQZZ_charenlarge_V, _QQZZ_charmag_V,
02183                                 _QQZZ_ColorCode, _QQZZ_DrawMode, flag, arg);
02184 }
02185
```

GMAIN.C

```
02185
02186 /*****************************************************************
02187  *                                                               *
02188  * 文字描画オフセット計算                                        *
02189  *                                                               *
02190  *****************************************************************/
02191 static VOID
02192 _QQ_calcchardoff(doffx, doffy)
02193 int     *doffx, *doffy;
02194 {
02195         int     ph_hi, ph_wd;
02196
02197         /* 側辺ベクトルと底辺ベクトルの計算 */
02198         ph_hi = _QQ_calcdrawsize(_QQZZ_charenlarge_V,_QQZZ_charmag_V,CHR_SIZV);
02199         if(_QQZZ_chartransf) {
02200                 _QQ_calctransformoff(_QQZZ_charangle_VX, _QQZZ_charangle_VY,
02201                                                 ph_hi, doffx, doffy);
02202         } else {
02203                 *doffx = 0;
02204                 *doffy = (_QQZZ_charangle_VY >= 0) ? ph_hi : -ph_hi;
02205         }
02206 }
02207
```

保守／廃止

GMAIN.C

```
02207
02208 /*****************************************************************
02209  *                                                               *
02210  * コピー開始位置オフセット計算                                  *
02211  *                                                               *
02212  *****************************************************************/
02213 static VOID
02214 _QQ_calccopydoff(enlarge, mag, length, transf, vx, vy, doffx, doffy)
02215 int             *doffx, *doffy;
02216 int             enlarge, mag;
02217 int             length;
02218 int             transf, vx, vy;
02219 {
02220         int     ph_hi, ph_wd;
02221
02222         /* 側辺ベクトルと底辺ベクトルの計算 */
02223         ph_hi = _QQ_calcdrawsize(enlarge, mag, length);
02224         if(transf) {
02225                 _QQ_calctransformoff(vx, vy, ph_hi, doffx, doffy);
02226         } else {
02227                 *doffx = (vx) ? ((vx > 0) ? ph_hi : -ph_hi) : 0;
02228                 *doffy = (vy) ? ((vy > 0) ? ph_hi : -ph_hi) : 0;
02229         }
02230 }
02231
```

GMAIN.C

```
02231
02232 /*****************************************************************
02233  *                                                               *
02234  * 描画サイズ計算                                                *
02235  *                                                               *
02236  *****************************************************************/
02237 static int
02238 _QQ_calcdrawsize(large, mag, orgsize)
02239 int             large;
02240 int             mag;
02241 int             orgsize;
02242 {
02243         int     destsize;
02244
02245         /* 側辺ベクトルと底辺ベクトルの計算 */
02246         destsize = (large) ? (orgsize*16+(mag+1)/2)/(mag+1)-1
02247                            : (orgsize*(mag+1)+(16/2))/16-1;
02248         return(destsize);
02249 }
02250
```

保守／廃止

GMAIN.C

```
02250
02251 /*************************************************************
02252  *                                                           *
02253  * 変形コピーにおける描画オフセット計算                      *
02254  *                                                           *
02255  *************************************************************/
02256 static VOID
02257 _QQ_calctransformoff(vx, vy, size, doffx, doffy)
02258 int             vx, vy;
02259 int             size;
02260 int             *doffx, *doffy;
02261 {
02262 #define sign(i)         (((i) < 0) ? -1 : 1)
02263 #define fy(x,u,v)       ((short)(((long)(x)*(v)+(u)/2)/(u)))
02264 #define fx(y,u,v)       ((short)(((long)(y)*(u)+(v)/2)/(v)))
02265         int     abs_vx, abs_vy;
02266         int     sign_angle;
02267         abs_vx = abs(vx);
02268         abs_vy = abs(vy);
02269         sign_angle = sign(vx)*sign(vy);
02270         if(abs_vx >= abs_vy) {
02271                 *doffx = (vx >= 0) ? size : -size;
02272                 *doffy = sign(*doffx)*sign_angle*fy(abs(*doffx),abs_vx,abs_vy);
02273         } else {
02274                 *doffy = (vy >= 0) ? size : -size;
02275                 *doffx = sign(*doffy)*sign_angle*fx(abs(*doffy),abs_vx,abs_vy);
02276         }
02277 }
02278
```

GMAIN.C

```
02278
02279 /*****************************************************************
02280  *                                                               *
02281  * 色情報読みだし                                                *
02282  *                                                               *
02283  *****************************************************************/
02284 GLOBAL int
02285 gGetColor(x, y)
02286 int             x, y;
02287 {
02288         if(x < _QQZZ_CURscreen->scr_cx0 || x > _QQZZ_CURscreen->scr_cx1) {
02289                 return(-1);
02290         }
02291         if(y < _QQZZ_CURscreen->scr_cy0 || y > _QQZZ_CURscreen->scr_cy1) {
02292                 return(-1);
02293         }
02294         return(v_point(x, y));
02295 }
02296
```

保守／廃止

GMAIN.C

```
02296
02297 /*****************************************************************
02298  *                                                               *
02299  * 最後に描いた弧の範囲                                          *
02300  *                                                               *
02301  *****************************************************************/
02302 GLOBAL VOID
02303 gGetTip(x0, y0, x1, y1)
02304 int     *x0, *y0, *x1, *y1;
02305 {
02306         *x0 = _QQZZ_Tip_sx;             /* 最後に描いた弧の始点ｘ座標 */
02307         *y0 = _QQZZ_Tip_sy;             /* 最後に描いた弧の始点ｙ座標 */
02308         *x1 = _QQZZ_Tip_ex;             /* 最後に描いた弧の終点ｘ座標 */
02309         *y1 = _QQZZ_Tip_ey;             /* 最後に描いた弧の終点ｙ座標 */
02310 }
02311
```

GMAIN.C

```
02311
02312 /*****************************************************************
02313  *                                                               *
02314  * 色情報の検索                                                  *
02315  *                                                               *
02316  *****************************************************************/
02317 GLOBAL
02318 gBitSearch(x, y, dir, col, dat, dx, dy)
02319 int             x, y, dir, col, dat;
02320 int             *dx, *dy;
02321 {
02322         return(v_bitsearch(x, y, dir, col, dat, dx, dy));
02323 }
02324
```

GMAIN.C

```
02324
02325 /*****************************************************************
02326  *                                                               *
02327  * 表示メモリの読みだし                                          *
02328  *                                                               *
02329  *****************************************************************/
02330 GLOBAL u_short
02331 gPeek(adr)
02332 u_long  adr;
02333 {
02334         u_short bank, offset;
02335
02336         bank = (u_short)(adr >> 16);
02337         offset = (u_short)(adr & 0x0000FFFFL);
02338
02339         return(v_peek(bank, offset));
02340 }
02341
```

GMAIN.C

```
02341
02342 /*****************************************************************
02343  *                                                               *
02344  * 表示メモリへの書き込み                                        *
02345  *                                                               *
02346  *****************************************************************/
02347 GLOBAL VOID
02348 gPoke(adr, data)
02349 u_long  adr;
02350 u_short data;
02351 {
02352         u_short bank, offset;
02353
02354         bank = (u_short)(adr >> 16);
02355         offset = (u_short)(adr & 0x0000FFFFL);
02356
02357         v_poke(bank, offset, data);
02358 }
```

保守／廃止

A.6　[GIO.C]

GIO.C

```
00001 /*********************************/
00002 /*      @(#) gio.asm 3.2         */
00003 /*********************************/
00004
00005 #include <ctype.h>
00006 #include <dos.h>
00007 #include "giosys.h"
00008 #include "gio.h"
00009
00010 /*      <<< EXTERNAL DATA AREA >>> */
00011 extern  u_char  _QQZZ_ctrl;
00012
00013 extern  u_char  _QQZZ_PCtype;
00014 extern  u_int   _QQZZ_AGDC_SEG;
00015 extern  u_char  _QQZZ_LinePL;
00016 extern  u_int   _QQZZ_LinePtnL;
00017 extern  u_int   _QQZZ_LinePtnH;
00018 extern  u_char  _QQZZ_LineWid;
00019 extern  char    _QQZZ_LinePtnEF;
00020 extern  u_char  _QQZZ_FptnType;
00021 extern  u_int   _QQZZ_Fptncnt;
00022
00023 extern  u_int   _QQZZ_DrawMode;
00024 extern  u_int   _QQZZ_ColorCode;
00025
00026 extern  u_long  _QQZZ_reg;
00027 extern  u_long  _QQZZ_gmem;
00028 extern  u_char  _QQZZ_seg_base;
00029
00030 /* パターンアドレス（ＡＧＤＣアドレス） */
00031 static u_int    QQ_FillPtnp_H;              /* カレントフィルパターン(High) */
00032 static u_int    QQ_FillPtnp_L;              /* カレントフィルパターン(Low) */
00033 static u_int    QQ_GpenPtnp_H;              /* カレントペンパターン(High) */
00034 static u_int    QQ_GpenPtnp_L;              /* カレントペンパターン(Low) */
00035 static u_int    QQ_GpenMask_H;              /* カレントペンマスク(High) */
00036 static u_int    QQ_GpenMask_L;              /* カレントペンマスク(Low) */
00037
```

保守／廃止

GIO.C

```
00037
00038 /*****************************************************************
00039  *                                                               *
00040  * AGDCステイタス待ち                                            *
00041  *                                                               *
00042  *****************************************************************/
00043 _QQZZ_statbusy(mask)
00044 int             mask;
00045 {
00046         while((_v_get_reg(R_STATUS) & mask) != 0);
00047 }
00048
```

GIO.C

```
00048
00049 /*****************************************************************
00050  *                                                               *
00051  * クリップモード設定                                            *
00052  *                                                               *
00053  *****************************************************************/
00054 _v_clipmode(n)
00055 int             n;
00056 {
00057         /* プリプロセッサ待ち */
00058         while((_v_get_reg(R_STATUS) & 1) != 0);
00059
00060         /* クリップモード設定 */
00061         _v_put_regb(R_CLIP_B, n & 3);
00062 }
00063
```

GIO.C

```
00063
00064 /*****************************************************************
00065  *                                                               *
00066  * クリップエリア設定                                          *
00067  *                                                               *
00068  *****************************************************************/
00069 _v_cliparea(x0, y0, x1, y1)
00070 int             x0, y0, x1, y1;
00071 {
00072         /* プリプロセッサ待ち */
00073         while((_v_get_reg(R_STATUS) & 1) != 0);
00074
00075         /* クリップエリアの正規化 */
00076         if(x0 > x1) {
00077                 int             temp;
00078                 temp = x0;
00079                 x0 = x1;
00080                 x1 = temp;
00081         }
00082         if(y0 > y1) {
00083                 int             temp;
00084                 temp = y0;
00085                 y0 = y1;
00086                 y1 = temp;
00087         }
00088         /* クリップエリア設定 */
00089         _v_put_reg(R_XCLMIN, x0);
00090         _v_put_reg(R_XCLMAX, x1);
00091         _v_put_reg(R_YCLMIN, y0);
00092         _v_put_reg(R_YCLMAX, y1);
00093 }
00094
```

保守／廃止

GIO.C

```
00094
00095 /*****************************************************************
00096  *                                                               *
00097  * フィルパターンデータ転送(to AGDC RAM)                        *
00098  *                                                               *
00099  *****************************************************************/
00100 _QQZZ_ptnp(ptnp, len)
00101 u_short *ptnp;
00102 int             len;
00103 {
00104 #ifndef LARGE_DATA                      /* for small model */
00105         struct SREGS    segreg;
00106 #endif
00107         unsigned short far      *adr;
00108         unsigned short far      *v_setmem();
00109         u_char  dm_sel;
00110         u_short patad_h;                /* フィルパターン保存エリア上位アドレス */
00111         u_short patad_l;                /* フィルパターン保存エリア下位アドレス */
00112
00113         /* パターンアドレス設定 */
00114         QQ_FillPtnp_H = HPADDR_WH;
00115         QQ_FillPtnp_L = HPADDR_WL;
00116         patad_h = HPADDR_BH;
00117         patad_l = HPADDR_BL;
00118         adr = v_setmem(patad_h, patad_l);
00119
00120         /* フィルパターン転送 */
00121 #ifdef  LARGE_DATA              /* for large model */
00122         memcpy(adr, ptnp, 2*len);
00123 #else                                   /* for small model */
00124         segread(&segreg);
00125         movedata(segreg.ds, ptnp, (u_short)((unsigned long)adr >> 16), (u_short)adr,
      2*len);
00126 #endif
00127         SEL_IR;                         /* Enable <IRSEL> */
00128 }
00129
```

保守／廃止

GIO.C

```
00129
00130  /*****************************************************************
00131   *                                                               *
00132   * ペンパターンデータ転送(to AGDC RAM)                          *
00133   *                                                               *
00134   *****************************************************************/
00135  _QQZZ_Gpen_ptnp(ptnp, len)
00136  u_short *ptnp;
00137  int             len;
00138  {
00139  #ifndef LARGE_DATA                      /* for small model */
00140          struct SREGS    segreg;
00141  #endif
00142          unsigned short far      *adr;
00143          unsigned short far      *v_setmem();
00144          u_char  dm_sel;
00145          u_short patad_h;                /* ペンパターン保存エリア上位アドレス */
00146          u_short patad_l;                /* ペンパターン保存エリア下位アドレス */
00147
00148          /* パターンアドレス設定 */
00149          QQ_GpenPtnp_H = HGPADDR_WH;
00150          QQ_GpenPtnp_L = HGPADDR_WL;
00151          patad_h = HGPADDR_BH;
00152          patad_l = HGPADDR_BL;
00153          adr = v_setmem(patad_h, patad_l);
00154
00155          /* ペンパターン転送 */
00156  #ifdef  LARGE_DATA              /* for large model */
00157          memcpy(adr, ptnp, 2*len);
00158  #else                                   /* for small model */
00159          segread(&segreg);
00160          movedata(segreg.ds, ptnp, (u_short)((unsigned long)adr >> 16), (u_short)adr,
      2*len);
00161  #endif
00162
00163          SEL_IR;                         /* Enable <IRSEL> */
00164  }
00165
```

保守／廃止

GIO.C

```
00165
00166 /*****************************************************************
00167  *                                                               *
00168  * ペンマスクデータ転送(to AGDC RAM)                             *
00169  *                                                               *
00170  *****************************************************************/
00171 _QQZZ_Gpen_mask(ptnp, len)
00172 u_short *ptnp;
00173 int             len;
00174 {
00175 #ifndef LARGE_DATA                      /* for small model */
00176         struct SREGS    segreg;
00177 #endif
00178         unsigned short far      *adr;
00179         unsigned short far      *v_setmem();
00180         u_char  dm_sel;
00181         u_short patad_h;                /* ペンパターン保存エリア上位アドレス */
00182         u_short patad_l;                /* ペンパターン保存エリア下位アドレス */
00183
00184         /* パターンアドレス設定 */
00185         QQ_GpenMask_H = HGMADDR_WH;
00186         QQ_GpenMask_L = HGMADDR_WL;
00187         patad_h = HGMADDR_BH;
00188         patad_l = HGMADDR_BL;
00189         adr = v_setmem(patad_h, patad_l);
00190
00191         /* ペンパターン転送 */
00192 #ifdef  LARGE_DATA              /* for large model */
00193         memcpy(adr, ptnp, 2*len);
00194 #else                                   /* for small model */
00195         segread(&segreg);
00196         movedata(segreg.ds, ptnp, (u_short)((unsigned long)adr >> 16), (u_short)adr,
    2*len);
00197 #endif
00198
00199         SEL_IR;                         /* Enable <IRSEL> */
00200 }
00201
```

保守／廃止

GIO.C

```
00201
00202 /*****************************************************************
00203  *                                                               *
00204  * ドットカラー読み込み                                          *
00205  *                                                               *
00206  *****************************************************************/
00207 GLOBAL int
00208 v_point(x, y)
00209 int             x, y;
00210 {
00211         /* プリプロセッサ待ち */
00212         while((_v_get_reg(R_STATUS) & 1) != 0);
00213
00214         /* 読み込み位置設定 */
00215         _v_put_reg(R_X, x);
00216         _v_put_reg(R_Y, y);
00217
00218         /* 読み込みコマンド発行 */
00219         _v_put_reg(R_COMMAND, 0x9c00);          /* <READ_COL> */
00220
00221         /* プリプロセッサ待ち */
00222         while((_v_get_reg(R_STATUS) & 1) != 0);
00223
00224         /* 結果を得る */
00225         return(_v_get_reg(R_DX) & 0xff);
00226 }
00227
```

保守／廃止

GIO.C

```
00227
00228 /*****************************************************************
00229  *                                                               *
00230  * 直線系図形のラインスタイル設定                                *
00231  *                                                               *
00232  *****************************************************************/
00233 static u_short
00234 QQ_set_linestyle(c_cmd)
00235 u_short c_cmd;
00236 {
00237         /* 下位１６ビットパターンの設定 */
00238         _v_put_reg(R_PTNCNT, _QQZZ_LinePtnL);
00239
00240         /* 上位１６ビットパターンの設定 */
00241         if(_QQZZ_LinePL == 1) {
00242                 /* ラインスタイルが３２ビット長の場合
00243                         上位１６ビットパターンを設定する */
00244                 _v_put_reg(R_DH, _QQZZ_LinePtnH);
00245                 return(c_cmd | 0x44);           /* PL=1,IP=1 */
00246         } else {
00247                 /* ラインスタイルが１６ビット長の場合
00248                         上位１６ビットパターンをクリアする */
00249                 _v_put_reg(R_DH, 0x0000);
00250                 return(c_cmd);
00251         }
00252 }
00253
```

保守／廃止

GIO.C

```
00253
00254 /*****************************************************************
00255  *                                                               *
00256  * 円，楕円系図形のラインスタイル設定                            *
00257  *                                                               *
00258  *****************************************************************/
00259 static u_short
00260 QQ_set_arcstyle(c_cmd)
00261 u_short c_cmd;
00262 {
00263         /* 下位１６ビットパターンの設定 */
00264         _v_put_reg(R_PTNCNT, _QQZZ_LinePtnL);
00265
00266         /* 上位１６ビットパターンの設定 */
00267         if(_QQZZ_LinePL == 1) {
00268                 /* ラインスタイルが３２ビット長の場合
00269                         上位１６ビットパターンを設定する */
00270                 _v_put_reg(R_PTNH, _QQZZ_LinePtnH);
00271                 return(c_cmd | 0x44);           /* PL=1,IP=1 */
00272         } else {
00273                 /* ラインスタイルが１６ビット長の場合
00274                         上位１６ビットパターンをクリアする */
00275                 _v_put_reg(R_PTNH, 0x0000);
00276                 return(c_cmd);
00277         }
00278 }
00279
```

保守／廃止

GIO.C

```
00279
00280 /*****************************************************************
00281  *                                                               *
00282  * 線種の拡大／縮小設定                                          *
00283  *                                                               *
00284  *****************************************************************/
00285 static u_short
00286 QQ_set_linemag(c_cmd)
00287 u_short c_cmd;
00288 {
00289         u_char  mag_hv;                 /* レジスタに設定する拡大／縮小指定コード */
00290         u_short flags;                  /* 拡大／縮小設定にともなうフラグ */
00291
00292         flags = 0x00;
00293         mag_hv = 0xff;
00294
00295         if(_QQZZ_LineWid != 0) {
00296                 if(_QQZZ_LineWid & 0x0f) {
00297                         /* 線幅拡大の場合 */
00298                         flags = 0x60;                   /* ES=1,IP=1 */
00299                 }
00300                 if(_QQZZ_LineWid & 0xf0) {
00301                         /* 線パターンの拡大／縮小が行われる場合 */
00302                         mag_hv = _QQZZ_LineWid-0x10;
00303                 } else {
00304                         /* 線パターンの拡大／縮小がない場合 */
00305                         mag_hv |= 0xf0;                 /* 等倍コードを設定する */
00306                 }
00307                 if(_QQZZ_LinePtnEF != 1) {
00308                         flags |= 0x62;                  /* ESH=1, IP=1 */
00309                 }
00310         }
00311
00312         /* 拡大／縮小コードの設定 */
00313         _v_put_regb(R_MAG_HV_B, mag_hv);
00314
00315         return(c_cmd | flags);
00316 }
00317
```

保守／廃止

GIO.C

```
00317
00318 /*****************************************************************
00319  *                                                               *
00320  * AGDCによる直線描画                                            *
00321  *                                                               *
00322  *****************************************************************/
00323 v_line(x0, y0, x1, y1, c, mode, wep, ip)
00324 int             x0, y0, x1, y1;
00325 int             c, mode, wep, ip;
00326 {
00327         u_short command;                /* 発行するコマンド */
00328
00329         /* プリプロセッサ待ち */
00330         while((_v_get_reg(R_STATUS) & 1) != 0);
00331
00332         /* 描画位置パラメタ設定 */
00333         _v_put_reg(R_X, x0);
00334         _v_put_reg(R_Y, y0);
00335         _v_put_reg(R_XE, x1);
00336         _v_put_reg(R_YE, y1);
00337         /* カラーコードと演算指定の設定 */
00338         _v_put_reg(R_PLANES, c);
00339         _v_put_reg(R_MOD10, mode);
00340
00341         /* ラインスタイル設定 */
00342         command = QQ_set_linestyle(0x1400);
00343         /* 終点描画指定をコマンドコードに加える */
00344         command |= (wep & 1);
00345         /* 線種拡大／縮小設定 */
00346         command |= QQ_set_linemag(command);
00347
00348         /* コマンド発行 */
00349         _v_put_reg(R_COMMAND, command);
00350 }
00351
```

保守／廃止

GIO.C

```
00351
00352  /*****************************************************************
00353   *                                                               *
00354   * 矩形データのセット                                          *
00355   *                                                               *
00356   *****************************************************************/
00357  QQ_set_box(x0, y0, x1, y1, c, mode)
00358  int              x0, y0, x1, y1;
00359  int              c, mode;
00360  {
00361          _v_put_reg(R_X, x0);                    /* 開始位置 X */
00362          _v_put_reg(R_Y, y0);                    /* 開始位置 Y */
00363          _v_put_reg(R_XS, x1);                   /* 終了位置 X */
00364          _v_put_reg(R_YS, y1);                   /* 終了位置 Y */
00365          _v_put_reg(R_PLANES, c);                /* カラーコード */
00366          _v_put_reg(R_MOD10, mode);              /* 描画モード */
00367  }
00368
```

保守／廃止

GIO.C

```
00368
00369 /*****************************************************************
00370  *                                                               *
00371  * AGDCによる長方形の描画                                        *
00372  *                                                               *
00373  *****************************************************************/
00374 v_box(x0, y0, x1, y1, c, mode)
00375 int             x0, y0, x1, y1;
00376 int             c, mode;
00377 {
00378         u_short command;                /* 発行するコマンド */
00379
00380         /* プリプロセッサ待ち */
00381         while((_v_get_reg(R_STATUS) & 1) != 0);
00382
00383         /* 矩形領域共通パラメタの設定 */
00384         QQ_set_box(x0, y0, x1, y1, c, mode);
00385
00386         /* ラインスタイル設定 */
00387         command = QQ_set_linestyle(0x4800);     /* <A_REC> */
00388         /* 線種拡大／縮小設定 */
00389         command = QQ_set_linemag(command);
00390
00391         /* コマンド発行 */
00392         _v_put_reg(R_COMMAND, command);
00393 }
00394
```

保守／廃止

GIO.C

```
00394
00395 /*****************************************************************
00396  *                                                               *
00397  * フィルパターンデータ・セット                                  *
00398  *                                                               *
00399  *****************************************************************/
00400 static u_short
00401 QQ_PtrnSet(c_cmd)
00402 u_short c_cmd;
00403 {
00404         u_short flags;                  /* フィルパターン操作に関与するフラグ */
00405
00406         /* パターン格納アドレスの設定 */
00407         _v_put_reg(R_PTNPH, QQ_FillPtnp_H);
00408         _v_put_reg(R_PTNPL, QQ_FillPtnp_L);
00409         /* パターン長の設定 */
00410         _v_put_reg(R_PTNCNT, _QQZZ_Fptncnt);
00411
00412         if(_QQZZ_FptnType == 0) {
00413                 /* １６ビット長ラインパターンの場合 */
00414                 flags = 0x10;                                          /* TL=0,SS=1
      (PTNCNT) */
00415         } else if(_QQZZ_FptnType == 1) {
00416                 /* １６ビット長シングルプレーンパターンの場合 */
00417                 flags = 0x90;                                          /* TL=1,SS=1
      (Single) */
00418         } else if(_QQZZ_FptnType == 2) {
00419                 /* １６ビット長マルチプレーンパターンの場合 */
00420                 flags = 0x80;                                          /* TL=1,SS=0
      (Multi) */
00421                 _v_put_regb(R_PDISPSH_B, 0x00);
00422                 _v_put_reg(R_PDISPSL, _QQZZ_Fptncnt);
00423         } else {
00424                 /* ３２ビット長パターンの場合 */
00425                 flags = 0x81;
00426                 _v_put_regb(R_PDISPSH_B, 0x00);
00427                 _v_put_reg(R_PDISPSL, 2*_QQZZ_Fptncnt);
00428         }
00429
00430         return(c_cmd | flags);
00431 }
00432
```

GIO.C

```
00432
00433  /*****************************************************************
00434   *                                                               *
00435   * AGDCによる塗りつぶし長方形の描画                              *
00436   *                                                               *
00437   *****************************************************************/
00438  v_boxf(x0, y0, x1, y1, c, mode)
00439  int             x0, y0, x1, y1;
00440  int             c, mode;
00441  {
00442          u_short command;                /* 発行するコマンド */
00443
00444          /* プリプロセッサ待ち */
00445          while((_v_get_reg(R_STATUS) & 1) != 0);
00446
00447          /* 矩形領域共通パラメタの設定 */
00448          QQ_set_box(x0, y0, x1, y1, c, mode);
00449
00450          /* フィルパターンパラメタの設定 */
00451          command = QQ_PtrnSet(0x8c2c);           /* <A_REC_FILL_C> */
00452
00453          /* コマンド発行 */
00454          _v_put_reg(R_COMMAND, command);
00455  }
00456
```

GIO.C

```
00456
00457 /*****************************************************************
00458  *                                                               *
00459  * AGDCによる塗りつぶし三角形の描画                              *
00460  *                                                               *
00461  *****************************************************************/
00462 v_tri_fill(x0, y0, x1, y1, x2, y2, c, mode)
00463 int             x0, y0, x1, y1, x2, y2;
00464 int             c, mode;
00465 {
00466         u_short command;                /* 発行するコマンド */
00467
00468         /* プリプロセッサ待ち */
00469         while((_v_get_reg(R_STATUS) & 1) != 0);
00470
00471         /* 頂点座標パラメタの設定 */
00472         _v_put_reg(R_X, x0);
00473         _v_put_reg(R_Y, y0);
00474         _v_put_reg(R_XS, x1);
00475         _v_put_reg(R_YS, y1);
00476         _v_put_reg(R_XC, x2);
00477         _v_put_reg(R_YC, y2);
00478         /* カラーコードと演算の設定 */
00479         _v_put_reg(R_PLANES, c);
00480         _v_put_reg(R_MOD10, mode);
00481
00482         /* フィルパターンパラメタの設定 */
00483         command = QQ_PtrnSet(0x6c2c);           /* <A_TRI_FILL> */
00484
00485         /* コマンド発行 */
00486         _v_put_reg(R_COMMAND, command);
00487 }
00488
```

保守／廃止

GIO.C

```
00488
00489 /*****************************************************************
00490  *                                                               *
00491  * 円データのセット                                              *
00492  *                                                               *
00493  *****************************************************************/
00494 QQ_set_circle(x, y, r, c, mode)
00495 int             x, y, r, c, mode;
00496 {
00497         _v_put_reg(R_XC, x);                            /* 中心点 X */
00498         _v_put_reg(R_YC, y);                            /* 中心点 Y */
00499         _v_put_reg(R_DX, r);                            /* 半径 */
00500         _v_put_reg(R_PLANES, c);                        /* カラーコード */
00501         _v_put_reg(R_MOD10, mode);                      /* 描画モード */
00502 }
00503
```

GIO.C

```
00503
00504 /*****************************************************************
00505  *                                                               *
00506  * AGDCによる円の描画                                            *
00507  *                                                               *
00508  *****************************************************************/
00509 v_circle(x, y, r, c, mode)
00510 int             x, y, r;
00511 int             c, mode;
00512 {
00513         u_short command;                /* 発行するコマンド */
00514
00515         /* プリプロセッサ待ち */
00516         while((_v_get_reg(R_STATUS) & 1) != 0);
00517
00518         /* 円描画パラメタの設定 */
00519         QQ_set_circle(x, y, r, c, mode);
00520
00521         /* ラインスタイルパラメタの設定 */
00522         command = QQ_set_arcstyle(0x5000);      /* <CRL> */
00523
00524         /* コマンド発行 */
00525         _v_put_reg(R_COMMAND, command);
00526 }
00527
```

保守／廃止

GIO.C

```
00527
00528 /*****************************************************************
00529  *                                                               *
00530  * AGDCによる円弧の描画                                          *
00531  *                                                               *
00532  *****************************************************************/
00533 v_circ_arc(x, y, r, c, mode, xs, ys, xe, ye, dir, type)
00534 int             x, y, r;
00535 int             xs, ys, xe, ye;
00536 int             dir, type;
00537 int             c, mode;
00538 {
00539         u_short command;                /* 発行するコマンド */
00540
00541         /* プリプロセッサ待ち */
00542         while((_v_get_reg(R_STATUS) & 1) != 0);
00543
00544         /* 円描画パラメタの設定 */
00545         QQ_set_circle(x, y, r, c, mode);
00546
00547         /* 円弧描画に必要なパラメタの設定 */
00548         _v_put_reg(R_XS, xs);
00549         _v_put_reg(R_YS, ys);
00550         _v_put_reg(R_XE, xe);
00551         _v_put_reg(R_YE, ye);
00552
00553         /* 弧, 扇, 弦の選択パラメタ設定
00554                 基本コマンドコードが異なる */
00555         if(type == 2) {
00556                 command = 0x5a40;                               /* <CSEG> */
00557         } else if(type == 1) {
00558                 command = 0x5840;                               /* <CSEC> */
00559         } else {
00560                 command = 0x5441;                               /* <CARC><WEP> */
00561         }
00562         /* ラインスタイルパラメタの設定 */
00563         command = QQ_set_arcstyle(command);
00564
00565         /* 弧の描画方向を基本コマンドコードに加える */
00566         if(dir == 1) {
00567                 command |= 0x80;                                /* <CF> */
00568         }
00569
00570         /* コマンド発行 */
00571         _v_put_reg(R_COMMAND, command);
00572 }
00573
```

GIO.C

```
00573
00574 /*****************************************************************
00575  *                                                               *
00576  * AGDCによる塗りつぶし円の描画                                  *
00577  *                                                               *
00578  *****************************************************************/
00579 v_circ_fill(x, y, r, c, mode)
00580 int             x, y, r;
00581 int             c, mode;
00582 {
00583         u_short command;                /* 発行するコマンド */
00584
00585         /* プリプロセッサ待ち */
00586         while((_v_get_reg(R_STATUS) & 1) != 0);
00587
00588         /* 円描画パラメタの設定 */
00589         QQ_set_circle(x, y, r, c, mode);
00590
00591         /* フィルパターンの設定 */
00592         command = QQ_PtrnSet(0x502c);           /* <CRL_FILL> */
00593
00594         /* コマンド発行 */
00595         _v_put_reg(R_COMMAND, command);
00596 }
00597
```

GIO.C

```
00597
00598 /*****************************************************************
00599  *                                                               *
00600  * 楕円データのセット                                            *
00601  *                                                               *
00602  *****************************************************************/
00603 QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode)
00604 int             x, y, rx, ry, dh, dv, c, mode;
00605 {
00606         _v_put_reg(R_XC, x);                            /* 中心点 X */
00607         _v_put_reg(R_YC, y);                            /* 中心点 Y */
00608         _v_put_reg(R_DX, rx);                           /* X方向半径 */
00609         _v_put_reg(R_DY, ry);                           /* Y方向半径 */
00610         _v_put_reg(R_DH, dh);                           /* 半径の２乗比 X */
00611         _v_put_reg(R_DV, dv);                           /* 半径の２乗比 Y */
00612         _v_put_reg(R_PLANES, c);                        /* カラーコード */
00613         _v_put_reg(R_MOD10, mode);                      /* 描画モード */
00614 }
00615
```

保守／廃止

GIO.C

```
00615
00616 /*****************************************************************
00617  *                                                               *
00618  * AGDCによる楕円の描画                                          *
00619  *                                                               *
00620  *****************************************************************/
00621 v_ellipse(x, y, rx, ry, dh, dv, c, mode)
00622 int             x, y, rx, ry, dh, dv;
00623 int             c, mode;
00624 {
00625         u_short command;                /* 発行するコマンド */
00626
00627         /* プリプロセッサ待ち */
00628         while((_v_get_reg(R_STATUS) & 1) != 0);
00629
00630         /* 楕円描画パラメタの設定 */
00631         QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode);
00632
00633         /* ラインスタイルパラメタの設定 */
00634         command = QQ_set_arcstyle(0x5c00);              /* <ELPS> */
00635
00636         /* コマンド発行 */
00637         _v_put_reg(R_COMMAND, command);
00638 }
00639
```

保守／廃止

GIO.C

```
00639
00640 /*****************************************************************
00641  *                                                               *
00642  * AGDCによる楕円弧の描画                                        *
00643  *                                                               *
00644  *****************************************************************/
00645 v_elps_arc(x, y, rx, ry, dh, dv, c, mode, xs, ys, xe, ye, dir, type)
00646 int             x, y, rx, ry, dh, dv;
00647 int             xs, ys, xe, ye;
00648 int             dir, type;
00649 int             c, mode;
00650 {
00651         u_short command;                /* 発行するコマンド */
00652
00653         /* プリプロセッサ待ち */
00654         while((_v_get_reg(R_STATUS) & 1) != 0);
00655
00656         /* 楕円描画パラメタの設定 */
00657         QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode);
00658
00659         /* 楕円弧を描画するために必要なパラメタの設定 */
00660         _v_put_reg(R_XS, xs);                           /* 始点 X */
00661         _v_put_reg(R_YS, ys);                           /* 始点 Y */
00662         _v_put_reg(R_XE, xe);                           /* 終点 X */
00663         _v_put_reg(R_YE, ye);                           /* 終点 Y */
00664
00665         /* 弧, 扇, 弦の選択パラメタ設定
00666                 基本コマンドコードが異なる */
00667         if(type == 2) {
00668                 command = 0x6540;                               /* <ESEG> */
00669         } else if(type == 1) {
00670                 command = 0x6440;                               /* <ESEC> */
00671         } else {
00672                 command = 0x6041;                               /* <EARC><WEP> */
00673         }
00674
00675         /* ラインスタイルパラメタの設定 */
00676         command = QQ_set_arcstyle(command);
00677
00678         /* 弧の描画方向を基本コマンドコードに加える */
00679         if(dir == 1) {
00680                 command |= 0x80;                                /* <CF> */
00681         }
00682
00683         /* コマンド発行 */
00684         _v_put_reg(R_COMMAND, command);
00685 }
00686
```

保守／廃止

GIO.C

```
00686
00687 /*****************************************************************
00688  *                                                               *
00689  * AGDCによる塗りつぶし楕円の描画                                *
00690  *                                                               *
00691  *****************************************************************/
00692 v_elps_fill(x, y, rx, ry, dh, dv, c, mode)
00693 int             x, y, rx, ry, dh, dv;
00694 int             c, mode;
00695 {
00696         u_short command;                /* 発行するコマンド */
00697
00698         /* プリプロセッサ待ち */
00699         while((_v_get_reg(R_STATUS) & 1) != 0);
00700
00701         /* 楕円描画パラメタの設定 */
00702         QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode);
00703
00704         /* フィルパターンパラメタの設定 */
00705         command = QQ_PtrnSet(0x5c2c);           /* <ELPS_FILL> */
00706
00707         /* コマンド発行 */
00708         _v_put_reg(R_COMMAND, command);
00709 }
00710
```

保守／廃止

GIO.C

```
00710
00711 /*************************************************************
00712  *                                                           *
00713  * AGDCによる塗りつぶし塗りつぶし台形の描画                  *
00714  *                                                           *
00715  *************************************************************/
00716 _QQZZ_trapz_fill(x0, x1, x2, x3, y0, y1, wbs, c, mode)
00717 int             x0, x1, x2, x3, y0, y1;
00718 int             wbs, c, mode;
00719 {
00720         u_short command;                /* 発行するコマンド */
00721
00722         /* プリプロセッサ待ち */
00723         while((_v_get_reg(R_STATUS) & 1) != 0);
00724
00725         /* 描画パラメタの設定 */
00726         _v_put_reg(R_X, x0);                            /* 第１ｘ座標 */
00727         _v_put_reg(R_XS, x1);                           /* 第２ｘ座標 */
00728         _v_put_reg(R_YS, x2);                           /* 第３ｘ座標 */
00729         _v_put_reg(R_XE, x3);                           /* 第４ｘ座標 */
00730         _v_put_reg(R_Y, y0);                            /* 第１ｙ座標 */
00731         _v_put_reg(R_YE, y1);                           /* 第２ｙ座標 */
00732         _v_put_reg(R_PLANES, c);                        /* カラーコード */
00733         _v_put_reg(R_MOD10, mode);                      /* 演算指定 */
00734
00735         /* コマンドコード作成 */
00736         if(wbs) {
00737                 command = 0x702c;                               /* <A_TRA_FILL(WBS=1)
    > */
00738         } else {
00739                 command = 0x700c;                               /* <A_TRA_FILL(WBS=0)
    > */
00740         }
00741
00742         /* フィルパターンパラメタ設定 */
00743         command = QQ_PtrnSet(command);
00744
00745         /* コマンド発行 */
00746         _v_put_reg(R_COMMAND, command);
00747 }
00748
```

GIO.C

```
00748
00749  /*****************************************************************
00750   *                                                               *
00751   * 塗りつぶし処理                                                *
00752   *                                                               *
00753   *****************************************************************/
00754  QQ_paint(cmd, x, y, c, border)
00755  int             x, y, c, border;
00756  u_short cmd;
00757  {
00758          u_short command;                /* 発行するコマンド */
00759
00760          /* プリプロセッサおよび描画プロセッサ待ち */
00761          while((_v_get_reg(R_STATUS) & 3) != 0);
00762
00763          /* パラメタ設定 */
00764          _v_put_reg(R_X, x);                                     /* 中心座標 X */
00765          _v_put_reg(R_Y, y);                                     /* 中心座標 Y */
00766          _v_put_reg(R_PLANES, c);                        /* カラーコード */
00767          _v_put_reg(R_DX, border);                       /* 境界色コード */
00768          _v_put_reg(R_MAG_HV_B, 0xFF);
00769          _v_put_reg(R_MOD10, 0x02);                      /* MOD1=0,MOD0=2 */
00770
00771          /* フィルパターンパラメタの設定 */
00772          command = QQ_PtrnSet(cmd);
00773          /* コマンド発行 */
00774          _v_put_reg(R_COMMAND, command);
00775  }
00776
```

保守／廃止

GIO.C

```
00776
00777 /*****************************************************************
00778  *                                                               *
00779  * AGDCによる境界色指定閉領域塗りつぶし                          *
00780  *                                                               *
00781  *****************************************************************/
00782 v_paint(x, y, c, border)
00783 int             x, y, c, border;
00784 {
00785         QQ_paint(0x6820, x, y, c, border);
00786 }
00787
```

GIO.C

```
00787
00788 /*****************************************************************
00789  *                                                               *
00790  * AGDCによる境界色指定閉領域塗りつぶし                          *
00791  *                                                               *
00792  *****************************************************************/
00793 v_n_paint(x, y, c)
00794 int             x, y, c;
00795 {
00796         QQ_paint(0x6824, x, y, c, 0);
00797 }
00798
```

GIO.C

```
00798
00799 /*****************************************************************
00800  *                                                               *
00801  * AGDCによる点の描画                                            *
00802  *                                                               *
00803  *****************************************************************/
00804 v_pset(x, y, c, mode)
00805 int             x, y;
00806 int             c, mode;
00807 {
00808         u_short command;                /* 発行するコマンド */
00809
00810         /* プリプロセッサ待ち */
00811         while((_v_get_reg(R_STATUS) & 1) != 0);
00812
00813         /* 描画パラメタの設定 */
00814         _v_put_reg(R_X, x);                                     /* 描画位置 X */
00815         _v_put_reg(R_Y, y);                                     /* 描画位置 Y */
00816         _v_put_reg(R_PLANES, c);                        /* カラーコード */
00817         _v_put_reg(R_MOD10, mode);                      /* 描画モード */
00818         _v_put_reg(R_PTNCNT, 0xffff);
00819
00820         /* コマンド発行 */
00821         _v_put_reg(R_COMMAND, 0x0C00);          /* <A_DOT_M> */
00822 }
00823
```

保守／廃止

GIO.C

```
00823
00824 /*****************************************************************
00825  *                                                               *
00826  * ペンパターンとマスクのセット                                  *
00827  *                                                               *
00828  *****************************************************************/
00829 GLOBAL u_short
00830 QQ_Gpen_PtrnSet(c_cmd)
00831 u_short c_cmd;
00832 {
00833         /* ペンパターンポインタの設定 */
00834         _v_put_reg(R_PTNPH, QQ_GpenPtnp_H);
00835         _v_put_reg(R_PTNPL, QQ_GpenPtnp_L);
00836
00837         /* ペンマスクポインタの設定 */
00838         _v_put_regb(R_EAD3H_B, QQ_GpenMask_H);
00839         _v_put_reg(R_EAD3L, QQ_GpenMask_L);
00840
00841         return(c_cmd);
00842 }
00843
```

保守／廃止

GIO.C

```
00843
00844 /*****************************************************************
00845  *                                                               *
00846  * AGDCによるグラフィックスペンでの点の描画                      *
00847  *                                                               *
00848  *****************************************************************/
00849 v_gpen_pset(x, y, c, mode)
00850 int             x, y;
00851 int             c, mode;
00852 {
00853         u_short command;                /* 発行するコマンド */
00854
00855         /* プリプロセッサ待ち */
00856         while((_v_get_reg(R_STATUS) & 1) != 0);
00857
00858         /* 描画パラメタの設定 */
00859         _v_put_reg(R_X, x);                                     /* 開始位置 X */
00860         _v_put_reg(R_XE, x+1);                          /* 終了位置 X */
00861         _v_put_reg(R_Y, y);                                     /* 開始位置 Y */
00862         _v_put_reg(R_YE, y);                            /* 終了位置 Y */
00863         _v_put_reg(R_PLANES, c);                        /* カラーコード */
00864         _v_put_reg(R_MOD10, mode);                      /* 描画モード */
00865         _v_put_reg(R_PTNCNT, 0xffff);           /* PTNCNT */
00866         _v_put_regb(R_MAG_HV_B, 0x00);          /* =等倍 */
00867
00868         /* グラフィックスペンパラメタ設定 */
00869         command = QQ_Gpen_PtrnSet(0xc066);
00870
00871         /* コマンド発行 */
00872         _v_put_reg(R_COMMAND, command);
00873 }
00874
```

保守／廃止

GIO.C

```
00874
00875 /*****************************************************************
00876  *                                                               *
00877  * AGDCによるグラフィックスペンでの直線描画                      *
00878  *                                                               *
00879  *****************************************************************/
00880 v_gpen_line(x0, y0, x1, y1, c, mode, wep)
00881 int             x0, y0, x1, y1;
00882 int             c, mode, wep;
00883 {
00884         u_short command;                /* 発行するコマンド */
00885
00886         /* プリプロセッサ待ち */
00887         while((_v_get_reg(R_STATUS) & 1) != 0);
00888
00889         /* 描画パラメタの設定 */
00890         _v_put_reg(R_X, x0);                                /* 開始位置 X */
00891         _v_put_reg(R_Y, y0);                                /* 開始位置 Y */
00892         _v_put_reg(R_XE, x1);                               /* 終了位置 X */
00893         _v_put_reg(R_YE, y1);                               /* 終了位置 Y */
00894         _v_put_reg(R_PLANES, c);                            /* カラーコード */
00895         _v_put_reg(R_MOD10, mode);                          /* 描画モード */
00896
00897         /* ペンパターン設定 */
00898         command = QQ_Gpen_PtrnSet(0xc066);      /* <A_LINE_M1> */
00899
00900         /* 終点描画指定を基本コマンドコードに付加する */
00901         command |= (wep & 1);
00902
00903         /* ラインスタイルパラメタ設定 */
00904         command = QQ_set_linestyle(command);
00905
00906         /* 線種拡大／縮小パラメタ設定 */
00907         command = QQ_set_linemag(command);
00908
00909         /* コマンド発行 */
00910         _v_put_reg(R_COMMAND, command);
00911 }
00912
```

保守／廃止

GIO.C

```
00912
00913 /*****************************************************************
00914  *                                                               *
00915  * AGDCによるグラフィックスペンでの円の描画                      *
00916  *                                                               *
00917  *****************************************************************/
00918 v_gpen_circle(x, y, r, c, mode)
00919 int             x, y, r;
00920 int             c, mode;
00921 {
00922         u_short command;                /* 発行するコマンド */
00923
00924         /* プリプロセッサ待ち */
00925         while((_v_get_reg(R_STATUS) & 1) != 0);
00926
00927         /* 円描画パラメタの設定 */
00928         QQ_set_circle(x, y, r, c, mode);
00929
00930         /* ペンパターンパラメタの設定 */
00931         command = QQ_Gpen_PtrnSet(0xc866);
00932
00933         /* 線種拡大／縮小パラメタの設定 */
00934         command = QQ_set_linemag(command);
00935
00936         /* ラインスタイルパラメタの設定 */
00937         command = QQ_set_arcstyle(command);
00938
00939         /* コマンド発行 */
00940         _v_put_reg(R_COMMAND, command);
00941 }
00942
```

GIO.C

```
00942
00943 /*****************************************************************
00944  *                                                               *
00945  * AGDCによるグラフィックスペンでの円弧の描画                    *
00946  *                                                               *
00947  *****************************************************************/
00948 v_gpen_circ_arc(x, y, r, c, mode, xs, ys, xe, ye, dir, wep)
00949 int             x, y, r;
00950 int             xs, ys, xe, ye;
00951 int             dir, wep;
00952 int             c, mode;
00953 {
00954         u_short command;                /* 発行するコマンド */
00955
00956         /* プリプロセッサ待ち */
00957         while((_v_get_reg(R_STATUS) & 1) != 0);
00958
00959         /* 円描画パラメタの設定 */
00960         QQ_set_circle(x, y, r, c, mode);
00961
00962         /* 円弧描画に必要なパラメタの設定 */
00963         _v_put_reg(R_XS, xs);                           /* 始点 X */
00964         _v_put_reg(R_YS, ys);                           /* 始点 Y */
00965         _v_put_reg(R_XE, xe);                           /* 終点 X */
00966         _v_put_reg(R_YE, ye);                           /* 終点 Y */
00967
00968         /* グラフィックスペンパラメタの設定 */
00969         command = QQ_Gpen_PtrnSet(0xd066);
00970
00971         /* 線種拡大／縮小パラメタの設定 */
00972         command = QQ_set_linemag(command);
00973
00974         /* 終点描画指定を基本コマンドコードに付加する */
00975         command |= (wep & 1);
00976
00977         /* ラインスタイルパラメタの設定 */
00978         command = QQ_set_arcstyle(command);
00979
00980         /* 弧の描画方向を基本コマンドコードに付加する */
00981         if(dir) {
00982                 command |= 0x0080;
00983         }
00984
00985         /* コマンド発行 */
00986         _v_put_reg(R_COMMAND, command);
00987 }
00988
```

保守／廃止

GIO.C

```
00988
00989 /*****************************************************************
00990  *                                                               *
00991  * AGDCによるグラフィックスペンでの楕円の描画                    *
00992  *                                                               *
00993  *****************************************************************/
00994 v_gpen_ellipse(x, y, rx, ry, dh, dv, c, mode)
00995 int             x, y, rx, ry, dh, dv;
00996 int             c, mode;
00997 {
00998         u_short command;                /* 発行するコマンド */
00999
01000         /* プリプロセッサ待ち */
01001         while((_v_get_reg(R_STATUS) & 1) != 0);
01002
01003         /* 楕円描画パラメタの設定 */
01004         QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode);
01005
01006         /* グラフィックスペンパターンパラメタの設定 */
01007         command = QQ_Gpen_PtrnSet(0xcc66);
01008
01009         /* 線種拡大／縮小パラメタの設定 */
01010         command = QQ_set_linemag(command);
01011
01012         /* ラインスタイルパラメタの設定 */
01013         command = QQ_set_arcstyle(command);
01014
01015         /* コマンド発行 */
01016         _v_put_reg(R_COMMAND, command);
01017 }
01018
```

保守／廃止

GIO.C

```
01018
01019 /*****************************************************************
01020  *                                                               *
01021  * AGDCによる楕円弧の描画                                        *
01022  *                                                               *
01023  *****************************************************************/
01024 v_gpen_elps_arc(x, y, rx, ry, dh, dv, c, mode, xs,ys, xe,ye, dir, wep)
01025 int             x, y, rx, ry, dh, dv;
01026 int             xs, ys, xe, ye;
01027 int             dir, wep;
01028 int             c, mode;
01029 {
01030         u_short command;                /* 発行するコマンド */
01031
01032         /* プリプロセッサ待ち */
01033         while((_v_get_reg(R_STATUS) & 1) != 0);
01034
01035         /* 楕円描画パラメタの設定 */
01036         QQ_set_ellipse(x, y, rx, ry, dh, dv, c, mode);
01037
01038         /* 円弧描画に必要なパラメタの設定 */
01039         _v_put_reg(R_XS, xs);                           /* 始点 X */
01040         _v_put_reg(R_YS, ys);                           /* 始点 Y */
01041         _v_put_reg(R_XE, xe);                           /* 終点 X */
01042         _v_put_reg(R_YE, ye);                           /* 終点 Y */
01043
01044         /* グラフィックスペンパターンパラメタの設定 */
01045         command = QQ_Gpen_PtrnSet(0xd466);
01046
01047         /* 線種拡大／縮小パラメタの設定 */
01048         command = QQ_set_linemag(command);
01049
01050         /* ラインスタイルパラメタの設定 */
01051         command = QQ_set_arcstyle(command);
01052
01053         /* 終点描画指定を基本コマンドコードに付加する */
01054         command |= (wep & 1);
01055
01056         /* 弧の描画方向を基本コマンドコードに付加する */
01057         if(dir) {
01058                 command |= 0x0080;
01059         }
01060
01061         /* コマンド発行 */
01062         _v_put_reg(R_COMMAND, command);
01063 }
01064
```

保守／廃止

GIO.C

```
01064
01065 /*****************************************************************
01066  *                                                               *
01067  * AGDCによるコピーコマンド実行（ＡＣタイプ）                    *
01068  *                                                               *
01069  *****************************************************************/
01070 _QQZZ_copy_AC(cmd, mag, c, mode)
01071 u_short cmd;
01072 int             mag, c, mode;
01073 {
01074         u_short command;                /* 発行するコマンド */
01075
01076         /* プリプロセッサ待ち */
01077         while((_v_get_reg(R_STATUS) & 1) != 0);
01078
01079         /* コピーパラメタの設定 */
01080         _v_put_reg(R_MOD10, mode);                      /* 描画モード */
01081         _v_put_reg(R_PLANES, c);                        /* カラーコード */
01082         _v_put_regb(R_MAG_HV_B, mag);           /* 拡大／縮小 */
01083
01084         /* コマンド発行 */
01085         _v_put_reg(R_COMMAND, cmd | 0x8000);
01086 }
01087
```

GIO.C

```
01087
01088 /*****************************************************************
01089  *                                                               *
01090  * AGDCメモリの読みだし                                          *
01091  *                                                               *
01092  *****************************************************************/
01093 GLOBAL u_short
01094 v_peek(bank, offset)
01095 int             bank, offset;
01096 {
01097         u_short data;
01098         unsigned short far      *v_setmem();
01099
01100         data = *v_setmem(bank, offset);
01101
01102         SEL_IR;                         /* IRSEL */
01103
01104         return(data);
01105 }
01106
```

GIO.C

```
01106
01107 /*****************************************************************
01108  *                                                               *
01109  * AGDCメモリへの書き込み                                        *
01110  *                                                               *
01111  *****************************************************************/
01112 GLOBAL VOID
01113 v_poke(bank, offset, data)
01114 u_short bank, offset;
01115 u_short data;
01116 {
01117         unsigned short far      *v_setmem();
01118
01119         *v_setmem(bank, offset) = data;
01120
01121         /* IRSEL */
01122         SEL_IR;
01123
01124         return;
01125 }
01126
```

GIO.C

```
01126
01127 /*****************************************************************
01128  *                                                               *
01129  * AGDCメモリアクセス準備                                        *
01130  *                                                               *
01131  *****************************************************************/
01132 unsigned short far*
01133 v_setmem(bank, offset)
01134 u_short bank, offset;
01135 {
01136         /* プリプロセッサ待ち */
01137         while((_v_get_reg(R_STATUS) & 1) != 0);
01138
01139         {
01140                 _v_put_regb(R_BANK_B, bank);
01141                 if(offset > 0x7fff) {
01142                         outp(MEMSEG_PORT, _QQZZ_seg_base+0x10);
01143                 } else {
01144                         outp(MEMSEG_PORT, _QQZZ_seg_base);
01145                 }
01146                 outp(MEMMAP_PORT, PDSEL | DMSEL);
01147                 return(gramadr(offset));
01148         }
01149 }
```

A.7　[GSUB.C]

保守／廃止

GSUB.C

```
00001 /*********************************/
00002 /*      @(#) gsub.c 3.2          */
00003 /*********************************/
00004
00005 #include <ctype.h>
00006 #include <dos.h>
00007 #include "giosys.h"
00008 #include "gio.h"
00009
00010 extern  u_char  _QQZZ_ctrl;
00011 extern  u_char  _QQZZ_ctrl2;
00012
00013 extern  int     _QQZZ_WRK0scr;
00014 extern  int     _QQZZ_WRK1scr;
00015 #define WORKSCR0        (_QQZZ_WRK0scr) /* ワーク画面0（マスク作成専用） */
00016 #define WORKSCR1        (_QQZZ_WRK1scr) /* ワーク画面1（表示画面と同容量） */
00017
00018 extern  SCRTBL  *_QQZZ_CURscreen;       /* カレント画面の管理情報 */
00019 extern  SCRTBL  _QQZZ_screen[];         /* 画面管理テーブル */
00020 extern  int             _QQZZ_CURscr;
00021 extern  long    _QQZZ_reg;
00022
00023 extern  u_char  _QQZZ_LinePL;           /* 0:１６ビット,1:３２ビット線種 */
00024 extern  u_int   _QQZZ_LinePtnL;         /* 線種パターン（下位１６ビット） */
00025 extern  u_char  _QQZZ_FptnType;         /* 塗りつぶし種別 */
00026 extern  u_int   _QQZZ_Fptncnt;          /* 塗りつぶしパターン法    */
00027 extern  u_int   _QQZZ_Fptnwidth;        /* 塗りつぶしパターンの幅 */
00028 extern  u_int   _QQZZ_ColorCode;        /* カラーコード */
00029 extern  u_int   _QQZZ_DrawMode;         /* 描画モード */
00030
00031 /* _QQ_save_*()で保存される情報 */
00032 static  u_char  s_lpl, s_type;                  /* 線種パターン */
00033 static  u_int   s_lpL, s_cnt, s_wid;            /* 塗りつぶしパターン */
00034 static  short   s_cmod;                         /* クリップモード */
00035 static  short   s_cx0, s_cy0, s_cx1, s_cy1;     /* クリップエリア */
00036 static  int     s_plntop, s_plncnt;             /* 描画対象プレーン */
00037 static  int     s_wscr;                         /* 描画対象画面番号 */
00038
00039
```

保守／廃止

GSUB.C

```
00039
00040 /*****************************************************************
00041  *                                                               *
00042  * 線種，塗りつぶしパターンの保存                                *
00043  *                                                               *
00044  *****************************************************************/
00045 static VOID
00046 _QQ_save_ptn()
00047 {
00048         s_lpl  = _QQZZ_LinePL;
00049         s_lpL  = _QQZZ_LinePtnL;
00050         s_type = _QQZZ_FptnType;
00051         s_cnt  = _QQZZ_Fptncnt;
00052         s_wid  = _QQZZ_Fptnwidth;
00053 }
00054
```

GSUB.C

```
00054
00055 /*****************************************************************
00056  *                                                               *
00057  * 線種，塗りつぶしパターンの復元                                *
00058  *                                                               *
00059  *****************************************************************/
00060 static VOID
00061 _QQ_recover_ptn()
00062 {
00063         _QQZZ_LinePL   = s_lpl;
00064         _QQZZ_LinePtnL = s_lpL;
00065         _QQZZ_FptnType = s_type;
00066         _QQZZ_Fptncnt  = s_cnt;
00067         _QQZZ_Fptnwidth= s_wid;
00068 }
00069
```

保守／廃止

GSUB.C

```
00069
00070 /*****************************************************************
00071  *                                                               *
00072  * クリップ情報の保存                                            *
00073  *                                                               *
00074  *****************************************************************/
00075 static VOID
00076 _QQ_save_clip()
00077 {
00078         s_cmod = _QQZZ_CURscreen->scr_cmode;
00079         s_cx0  = _QQZZ_CURscreen->scr_cx0;
00080         s_cy0  = _QQZZ_CURscreen->scr_cy0;
00081         s_cx1  = _QQZZ_CURscreen->scr_cx1;
00082         s_cy1  = _QQZZ_CURscreen->scr_cy1;
00083 }
00084
```

保守／廃止

GSUB.C

```
00084
00085 /*****************************************************************
00086  *                                                               *
00087  * クリップ情報の復元                                            *
00088  *                                                               *
00089  *****************************************************************/
00090 static VOID
00091 _QQ_recover_clip()
00092 {
00093         v_clipmode(s_cmod);
00094         v_cliparea(s_cx0, s_cy0, s_cx1, s_cy1);
00095 }
00096
```

GSUB.C

```
00096
00097  /*****************************************************************
00098   *                                                               *
00099   * プレーン情報の保存                                            *
00100   *                                                               *
00101   *****************************************************************/
00102  static VOID
00103  _QQ_save_plane()
00104  {
00105          s_plntop = _QQZZ_CURscreen->scr_planetop;
00106          s_plncnt = _QQZZ_CURscreen->scr_planecnt;
00107  }
00108
```

GSUB.C

```
00108
00109 /*****************************************************************
00110  *                                                               *
00111  * プレーン情報の復元                                            *
00112  *                                                               *
00113  *****************************************************************/
00114 static VOID
00115 _QQ_recover_plane()
00116 {
00117         v_plane(s_plntop, s_plncnt);
00118 }
00119
```

保守／廃止

GSUB.C

```
00119
00120 /*****************************************************************
00121  *                                                               *
00122  * 描画画面選択の保存                                            *
00123  *                                                               *
00124  *****************************************************************/
00125 static VOID
00126 _QQ_save_wscr()
00127 {
00128         s_wscr = _QQZZ_CURscr;
00129 }
00130
```

GSUB.C

```
00130
00131  /*****************************************************************
00132   *                                                               *
00133   * プレーン情報の復元                                            *
00134   *                                                               *
00135   *****************************************************************/
00136  static VOID
00137  _QQ_recover_wscr()
00138  {
00139          v_act_page(s_wscr);
00140  }
00141
```

保守／廃止

GSUB.C

```
00141
00142 /*****************************************************************
00143  *                                                               *
00144  * 多角形の塗りつぶし出力                                        *
00145  *                                                               *
00146  *****************************************************************/
00147 GLOBAL VOID
00148 v_polygon_fill(n, dcp, c, mod)
00149 int             c;
00150 int             mod;
00151 int             n;
00152 struct v_xy     *dcp;
00153 {
00154 REG     struct v_xy     *dp;
00155         int     i, pc, flg;
00156         int     x0, y0, x1, y1, area_w, area_h;
00157         int     ox, oy;
00158         unsigned long   mask_top, patn_top;
00159         unsigned short  mask_pitch, patn_pitch;
00160         unsigned long   mask_disp, patn_disp;
00161         int     t_planetop, t_planecnt;
00162         int     t_cmode, t_cx0, t_cy0, t_cx1, t_cy1;
00163
00164         /* ターゲット画面番号の退避 */
00165         _QQ_save_wscr();
00166         _QQ_save_clip();
00167         _QQ_save_ptn();
00168         _QQ_save_plane();
00169
00170         /* ワーク画面0にマスクパターンを作る */
00171         v_act_page(WORKSCR0);
00172         v_plane(0, 1);
00173         _QQ_poly_rect(n, dcp, &x0, &y0, &x1, &y1);
00174         x0 &=  0x0f;    /* フィルパターンの位置を合わせるため */
00175         area_w = x1-x0+1;
00176         area_h = y1-y0+1;
00177         v_fillptn(0);
00178         v_boxf(x0, y0, x1, y1, 0, 0x22);
00179         _QQ_polyfill_mask(n, dcp, x0, y0, x1, y1);
00180         _QQ_recover_ptn();
00181
00182         /* ワーク画面1にフィルパターンを作る */
00183         v_act_page(WORKSCR1);
00184         _QQ_recover_clip();
00185         _QQ_recover_plane();
00186         v_boxf(x0, y0, x1, y1, _QQZZ_ColorCode, _QQZZ_DrawMode);
00187
00188         /* ターゲット画面に結果を移す */
00189         _QQ_recover_wscr();
00190
00191         _QQZZ_3ope_copy_AC(WORKSCR1, x0, y1, WORKSCR0,
00192                                                 x0, y1, x0, y1, area_w, area_h, 1, c,
 mod, -1);
00193 }
00194
```

保守／廃止

GSUB.C

```
00194
00195  /*****************************************************************
00196   *                                                               *
00197   * 多角形描画範囲判定                                            *
00198   *                                                               *
00199   *****************************************************************/
00200  static VOID
00201  _QQ_poly_rect(n, dcp, x0, y0, x1, y1)
00202  int     n;
00203  struct v_xy     *dcp;
00204  int     *x0, *y0, *x1, *y1;
00205  {
00206  register int    min_x, max_x, min_y, max_y;
00207  register int    i;
00208          min_x = max_x = (dcp+0)->x;
00209          min_y = max_y = (dcp+0)->y;
00210          for(i = 1; i < n; i += 1) {
00211                  if((dcp+i)->x < min_x) {
00212                          min_x = (dcp+i)->x;
00213                  } else if((dcp+i)->x > max_x) {
00214                          max_x = (dcp+i)->x;
00215                  }
00216                  if((dcp+i)->y < min_y) {
00217                          min_y = (dcp+i)->y;
00218                  } else if((dcp+i)->y > max_y) {
00219                          max_y = (dcp+i)->y;
00220                  }
00221          }
00222          *x0 = min_x;
00223          *y0 = min_y;
00224          *x1 = max_x;
00225          *y1 = max_y;
00226  }
00227
```

保守／廃止

GSUB.C

```
00227
00228 /*****************************************************************
00229  *                                                               *
00230  * 多角形マスク作成                                              *
00231  *                                                               *
00232  *****************************************************************/
00233 static VOID
00234 _QQ_polyfill_mask(n, dcp, x0, y0, x1, y1)
00235 int             n;
00236 struct v_xy     *dcp;
00237 int             x0, y0, x1, y1;         /* 多角形描画範囲 */
00238 {
00239 #define SAMEDIR(a,b)    (((a) > 0 && (b) > 0) || ((a) < 0 && (b) < 0))
00240 #define NEXTSUFFIX(n,s) ((s) == (n)-1 ? 0 : (s)+1)
00241 #define PREVSUFFIX(n,s) ((s) == 0 ? (n)-1 : (s)-1)
00242 REG     struct v_xy     *sp, *ep;
00243         int             i;
00244         int             c = 1;
00245         int             wbs;
00246
00247         v_lineptn(0);
00248         v_fillptn(0);
00249
00250         for(i = 0; i < n; i += 1) {
00251                 int     dir;    /* 辺の垂直方向の向き */
00252                 int     ei;
00253                 sp = (dcp+i);
00254                 ep = (dcp+(ei = NEXTSUFFIX(n, i)));
00255                 dir = ep->y-sp->y;
00256                 if(dir > 0) {
00257                         REG struct v_xy *pp;
00258                         int     prevdir;
00259                         int     pi;
00260                         pi = i;
00261                         do {
00262                                 pp = (dcp+(pi = PREVSUFFIX(n, pi)));
00263                                 prevdir = sp->y-pp->y;
00264                         } while(prevdir == 0);
00265                         if(SAMEDIR(dir, prevdir)) {
00266                                 wbs = 0;
00267                         } else {
00268                                 wbs = 1;
00269                         }
00270                         _QQZZ_trapz_fill(x0, ep->x, x0, sp->x, ep->y, sp->y, wbs, c,
    XOR);
00271                 } else if(dir < 0) {
00272                         REG struct v_xy *ap;
00273                         int     afterdir;
00274                         int     ai;
00275                         ai = ei;
00276                         do {
00277                                 ap = (dcp+(ai = NEXTSUFFIX(n, ai)));
00278                                 afterdir = ap->y-ep->y;
00279                         } while(afterdir == 0);
00280                         if(SAMEDIR(dir, afterdir)) {
00281                                 wbs = 0;
00282                         } else {
00283                                 wbs = 1;
00284                         }
00285                         _QQZZ_trapz_fill(x0, sp->x, x0, ep->x, sp->y, ep->y, wbs, c,
    XOR);
00286                 }
00287         }
00288         /* 輪郭の描画 */
00289         for (sp = dcp, ep = dcp+1, i = 0; i < n-1; sp++, ep++, i += 1) {
00290                 v_line(sp->x, sp->y, ep->x, ep->y, c, PSET, 1, 1);
00291         }
00292         v_line(sp->x, sp->y, dcp->x, dcp->y, c, PSET, 1, 1);
00293 }
00294
```

保守／廃止

GSUB.C

```
00294
00295 /*****************************************************************
00296  *                                                               *
00297  * 3オペランドコピー                                            *
00298  *                                                               *
00299  *****************************************************************/
00300 GLOBAL int
00301 _QQZZ_3ope_copy_AC(sscr, sx, sy, mscr, mx, my, dx, dy, xc, yc, t, c, mod, p)
00302 int     sscr;           /* 転送源画面番号 */
00303 int     mscr;           /* マスク画面番号 */
00304 int     sx, sy;         /* 転送元の位置（左上隅） */
00305 int     mx, my;         /* マスクパターン先頭位置（左上隅：ワード境界） */
00306 int     dx, dy;         /* 転送先の位置（左上隅） */
00307 int     xc, yc;         /* 転送範囲の大きさ（ドット数） */
00308 int     t;                      /* マスクビット指定 */
00309 int     c;                      /* プレーンと演算の関係 */
00310 int     mod;            /* 演算 */
00311 int     p;                      /* プレーン転送形式 負=M->M,正（プレーン番号）=S->M */
00312 {
00313         u_long  mask_disp, mask_top;
00314         u_long  patn_disp, patn_top;
00315         u_short mask_pitch, patn_pitch;
00316         u_short mask_boff, patn_boff;
00317         int     sscr_no, mscr_no;
00318         SCRTBL  *sscrn, *mscrn;
00319         u_short cmd;
00320 #define sbx     sx
00321 #define sby     (sscrn->scr_height-1-sy)
00322 #define mbx     mx
00323 #define mby     (mscrn->scr_height-1-my)
00324         mscr_no = _QQ_get_scraddr(mscr, &mask_disp, &mask_top);
00325         sscr_no = _QQ_get_scraddr(sscr, &patn_disp, &patn_top);
00326         if(mscr_no < 0 || sscr_no < 0) {
00327                 return(-1);
00328         }
00329         mscrn = &_QQZZ_screen[mscr_no];
00330         sscrn = &_QQZZ_screen[sscr_no];
00331         mask_pitch = mscrn->scr_pitch;
00332         patn_pitch = sscrn->scr_pitch;
00333         if(p >= 0) {
00334                 patn_top += (u_long)sscrn->scr_displace*p;
00335         }
00336         mask_top += (u_long)mby*mask_pitch +(mbx >> 4);
00337         patn_top += (u_long)sby*patn_pitch +(sbx >> 4);
00338         mask_boff = mbx & 0xf;
00339         patn_boff = sbx & 0xf;
00340
00341         _QQZZ_statbusy(1);              /* Wait until(PPBUSY==0) */
00342
00343         _v_put_reg(R_EAD2L, L_regW(patn_top));  /* 転送源adr（下位ワード）*/
00344         _v_put_reg(R_EAD2H_B, H_regW(patn_top));/* 転送源adr（上位ワード）*/
00345         _v_put_regb(R_dAD2_B, patn_boff);               /* 転送源ワード内オフセット */
00346         _v_put_reg(R_PITCHS, patn_pitch);               /* 転送源ピッチ */
00347         _v_put_reg(R_PDISPSL, L_regW(patn_disp));/* 転送源プレーン間隔（下位ワード）*/
00348         _v_put_regb(R_PDISPSH_B, H_regW(patn_disp));/* 転送源プレーン間隔（上位ワード）*/
00349         _v_put_reg(R_DH, xc-1);                                         /* 転送領域の幅 */
00350         _v_put_reg(R_DV, yc-1);                                         /* 転送領域の高さ */
00351         _v_put_reg(R_EAD3L, L_regW(mask_top));  /* マスクadr（下位ワード） */
00352         _v_put_regb(R_EAD3H_B, H_regW(mask_top));/* マスクadr（上位ワード） */
00353         _v_put_reg(R_X, dx);                                            /* 転送先X座標 */
00354         _v_put_reg(R_Y, dy);                                            /* 転送先Y座標 */
00355         _v_put_reg(R_PLANES, c);                                        /* 描画モード */
00356         _v_put_reg(R_MOD10, mod);                                       /* MOD1,MOD0 */
00357         _v_put_regb(R_MAG_HV_B, 0xFF);                  /* MAGV,MAGH */
00358
```

GSUB.C

```
00359           if(t == 0) {
00360                   cmd = 0x8051;
00361           } else {
00362                   cmd = 0x8071;
00363           }
00364           if(p < 0) {
00365                   /* Multi->Multi */
00366                   cmd |= 0x000c;
00367           } else {
00368                   /* Single->Multi */
00369                   cmd |= 0x0008;
00370           }
00371           _v_put_reg(R_COMMAND, cmd);
00372
00373           return(0);
00374   }
00375
```

保守／廃止

GSUB.C

```
00375
00376  /*****************************************************************
00377   *                                                               *
00378   * 扇形, 弦形の塗りつぶし出力                                    *
00379   *                                                               *
00380   *****************************************************************/
00381  GLOBAL VOID
00382  v_circ_arc_fill(cx,cy, r, c, mod, sx,sy, ex,ey, dir, type)
00383  int             cx,cy, r, c, mod, sx,sy, ex,ey, dir, type;
00384  {
00385  #define max(a,b)        ( (a) < (b) ? (b) : (a) )
00386  #define min(a,b)        ( (a) > (b) ? (b) : (a) )
00387  #define CHK_PAINT(x, y) {if (v_point((x),(y))==0) v_paint((x),(y),0,0);}
00388          int             lx, by, rx, ty; /* 境界座標 */
00389          int             dx, dy;         /* 転送先開始座標 */
00390          int             mx,my, xx,yy, flg;
00391          REG int         xc, yc;
00392
00393          /* 描画画面の退避 */
00394          _QQ_save_wscr();
00395
00396          /* マスク作成 */
00397          _QQ_save_ptn();
00398          _QQ_save_clip();
00399          _QQ_save_plane();
00400          v_act_page(WORKSCR0);
00401          v_fillptn(0);
00402          v_lineptn(0);
00403          v_plane(0, 1);
00404          v_clipmode(0);
00405          ++r;
00406          lx = cx - r;
00407          rx = cx + r;
00408          ty = cy + r;
00409          by = cy - r;
00410          if(_QQZZ_FptnType == 0) {
00411                  xx = _QQZZ_CURscreen->scr_width/2 & ~0xF;
00412                  yy = _QQZZ_CURscreen->scr_height/2;
00413          } else {
00414                  xx = _QQZZ_Fptnwidth*((_QQZZ_CURscreen->scr_width/2)
00415                          /_QQZZ_Fptnwidth)+(cx%_QQZZ_Fptnwidth);
00416                  yy = _QQZZ_Fptncnt*((_QQZZ_CURscreen->scr_height/2)
00417                          /_QQZZ_Fptncnt)+(cy%_QQZZ_Fptncnt);
00418          }
00419          dx = _QQZZ_CURscreen->scr_width -1;
00420          dy = _QQZZ_CURscreen->scr_height-1;
00421          if (r >= xx || r >= yy) {
00422                  flg = 1;
00423                  lx = max(lx, 0); rx = min(rx, dx);
00424                  by = max(by, 0); ty = min(ty, dy);
00425                  dx = lx; dy = ty ;
00426                  xx = cx; yy = cy;
00427          } else {
00428                  flg = 0;
00429                  dx = lx; mx = xx-cx; lx+= mx; rx+= mx;
00430                  dy = ty; my = yy-cy; ty+= my; by+= my;
00431                  sx+= mx; sy+= my   ; ex+= mx; ey+= my;
00432          }
00433          --r;
00434          dx -= lx & 0xF;
00435          lx &= ~0xF;
00436          xc = rx - lx + 1;
00437          yc = ty - by + 1;
00438          v_cliparea(lx,by, rx,ty);
00439          v_boxf(lx, by, rx, ty, 1, PSET);
00440          v_circ_arc (xx,yy, r, 0, PSET, sx,sy, ex,ey, dir, (type)?type:1);
00441          if ( flg ) {
00442                  CHK_PAINT(lx,by); CHK_PAINT(lx,ty);
00443                  CHK_PAINT(rx,by); CHK_PAINT(rx,ty);
```

15

保守／廃止

GSUB.C

```
00444           } else {
00445                   v_paint(lx, by, 0, 0);
00446           }
00447           v_circ_arc (xx,yy, r, 1, PSET, sx,sy, ex,ey, dir, (type)?type:1);
00448           _QQ_recover_ptn();
00449
00450           /* フィルパターン作成 */
00451           v_act_page(WORKSCR1);
00452           _QQ_recover_clip();
00453           _QQ_recover_plane();
00454           v_boxf(lx,by, rx,ty, c, PSET);
00455
00456           /* マスクとフィルパターンの合成出力 */
00457           _QQ_recover_wscr();
00458           _QQZZ_3ope_copy_AC(WORKSCR1,lx,ty,WORKSCR0,lx,ty,dx,dy,xc,yc,1,c,mod,-1);
00459
00460           return(0);
00461   }
00462
```

保守／廃止

GSUB.C

```
00462
00463 /*****************************************************************
00464  *                                                               *
00465  * 楕扇形，楕弦形の塗りつぶし出力                                *
00466  *                                                               *
00467  *****************************************************************/
00468 GLOBAL VOID
00469 v_elps_arc_fill(cx,cy, xr,yr,dh,dv, c, mod, sx,sy,ex,ey, dir, type)
00470 int             cx,cy, xr,yr,dh,dv, c, mod, sx,sy,ex,ey, dir, type;
00471 {
00472 #define max(a,b)        ( (a) < (b) ? (b) : (a) )
00473 #define min(a,b)        ( (a) > (b) ? (b) : (a) )
00474 #define CHK_PAINT(x, y) {if (v_point((x),(y))==0) v_paint((x),(y),0,0);}
00475         int             lx, by, rx, ty; /* 境界座標 */
00476         int             dx, dy;         /* 転送先開始座標 */
00477         int             f, mx,my, xx,yy, flg;
00478         u_char          s_lpl, s_type;  /* save area */
00479         u_int           s_lpL, s_cnt;
00480         REG int         xc, yc;
00481
00482         /* 描画対象画面の退避 */
00483         _QQ_save_wscr();
00484
00485         /* マスク作成 */
00486         _QQ_save_ptn();
00487         _QQ_save_clip();
00488         _QQ_save_plane();
00489         v_act_page(WORKSCR0);
00490         v_fillptn(0);
00491         v_lineptn(0);
00492         v_plane(0, 1);
00493         v_clipmode(0);
00494         ++xr; ++yr;
00495         lx = cx - xr;
00496         rx = cx + xr;
00497         ty = cy + yr;
00498         by = cy - yr;
00499         if(_QQZZ_FptnType == 0) {
00500                 xx = _QQZZ_CURscreen->scr_width/2 & ~0xF;
00501                 yy = _QQZZ_CURscreen->scr_height/2;
00502         } else {
00503                 xx = _QQZZ_Fptnwidth*((_QQZZ_CURscreen->scr_width/2)
00504                         /_QQZZ_Fptnwidth)+(cx%_QQZZ_Fptnwidth);
00505                 yy = _QQZZ_Fptncnt*((_QQZZ_CURscreen->scr_height/2)
00506                         /_QQZZ_Fptncnt)+(cy%_QQZZ_Fptncnt);
00507         }
00508         dx = _QQZZ_CURscreen->scr_width -1;
00509         dy = _QQZZ_CURscreen->scr_height-1;
00510         if (xr >= xx || yr >= yy) {
00511                 flg = 1;
00512                 lx = max(lx, 0); rx = min(rx, dx);
00513                 by = max(by, 0); ty = min(ty, dy);
00514                 dx = lx; dy = ty;
00515                 xx = cx; yy = cy;
00516         } else {
00517                 flg = 0;
00518                 dx  = lx; mx  = xx-cx; lx += mx; rx += mx;
00519                 dy  = ty; my  = yy-cy; ty += my; by += my;
00520                 sx += mx; sy += my   ; ex += mx; ey += my;
00521         }
00522         --xr; --yr;
00523         dx -= lx & 0xF;
00524         lx &= ~0xF;
00525         xc = rx - lx + 1;
00526         yc = ty - by + 1;
00527         v_cliparea(lx,by, rx,ty);
00528         v_boxf(lx, by, rx, ty, 1, PSET);
00529         v_elps_arc(xx,yy, xr,yr, dh,dv, 0, PSET, sx,sy, ex,ey, dir, (type)?type:1);
00530         if ( flg ) {
```

保守／廃止

GSUB.C

```
00531                   CHK_PAINT(lx,by); CHK_PAINT(lx,ty);
00532                   CHK_PAINT(rx,by); CHK_PAINT(rx,ty);
00533           } else {
00534                   v_paint(lx, by, 0, 0);
00535           }
00536           v_elps_arc(xx,yy, xr,yr, dh,dv, 1, PSET, sx,sy, ex,ey, dir, (type)?type:1);
00537           _QQ_recover_ptn();
00538
00539           /* フィルパターン作成 */
00540           v_act_page(WORKSCR1);
00541           _QQ_recover_clip();
00542           _QQ_recover_plane();
00543           v_boxf(lx,by, rx,ty, c, PSET);
00544
00545           /* マスクとフィルパターンの合成出力 */
00546           _QQ_recover_wscr();
00547           _QQZZ_3ope_copy_AC(WORKSCR1,lx,ty,WORKSCR0,lx,ty,dx,dy,xc,yc,1,c,mod,-1);
00548
00549           return(0);
00550   }
00551
```

保守／廃止

GSUB.C

```
00551
00552 /*****************************************************************
00553  *                                                               *
00554  * ドット検索                                                    *
00555  *                                                               *
00556  *****************************************************************/
00557 GLOBAL int
00558 v_bitsearch(x, y, dir, col, dat, dx, dy)
00559 int     x, y, dir, col, dat;
00560 int     *dx, *dy;
00561 {
00562         unsigned short  cmd;
00563         unsigned short  busy;
00564
00565         _QQZZ_statbusy(1);              /* Wait until(PPBUSY==0) */
00566
00567         _v_put_reg(R_X, x);             /* 検索開始位置x */
00568         _v_put_reg(R_Y, y);             /* 検索開始位置y */
00569         _v_put_reg(R_DX, col);  /* 境界色 */
00570         cmd = 0x3900;
00571         if(dir) {
00572                 cmd |= 0x0020;
00573         }
00574         if(dat) {
00575                 cmd |= 0x0010;
00576         }
00577         _v_put_reg(R_COMMAND, cmd);
00578
00579         do {
00580                 busy = _v_get_reg(R_STATUS);
00581         } while((busy & 0x3) && ((busy & 0xc) == 0));
00582
00583         *dx = _v_get_reg(R_X);
00584         *dy = _v_get_reg(R_Y);
00585
00586         return(busy & 0xc);
00587 }
00588
```

GSUB.C

```
00588
00589 /*****************************************************************
00590  *                                                               *
00591  * 画面消去                                                      *
00592  *                                                               *
00593  *****************************************************************/
00594 GLOBAL VOID
00595 v_clear(c)
00596 int     c;
00597 {
00598         _QQZZ_statbusy(3);      /* Wait until(PPBUSY==0) */
00599         _v_put_reg(R_PTNCNT, 0xffff);
00600         _v_put_reg(R_X, _QQZZ_CURscreen->scr_cx0);
00601         _v_put_reg(R_Y, _QQZZ_CURscreen->scr_cy0);
00602         _v_put_reg(R_XS, _QQZZ_CURscreen->scr_cx1);
00603         _v_put_reg(R_YS, _QQZZ_CURscreen->scr_cy1);
00604         _v_put_reg(R_PLANES, c);
00605         _v_put_reg(R_MOD10, 0x32);
00606         _v_put_reg(R_COMMAND, 0x8c3e);
00607 }
00608
```

保守／廃止

GSUB.C

```
00608
00609 /*****************************************************************
00610  *                                                               *
00611  * 台形塗りつぶし                                                *
00612  *                                                               *
00613  *****************************************************************/
00614 GLOBAL int
00615 v_tra_fill(x1, x2, x3, x4, y1, y2, wbs, c, mode)
00616 int             x1, x2, x3, x4;         /* 台形各点のX座標 */
00617 int             y1, y2;                 /* 上底，下底のY座標 */
00618 int             c;                      /* カラーコード */
00619 int             mode;                   /* 描画モード */
00620 int             wbs;                    /* 下底描画指定 */
00621 {
00622
00623         if (y1 == y2) {         /* this is for AGDC bug */
00624                 int             status;
00625                 if (x1 > x3) x1 = x3;
00626                 if (x2 < x4) x2 = x4;
00627                 if(wbs) {
00628                         return( v_boxf(x1,y1, x2,y1, c, mode) );
00629                 } else {
00630                         return(0);
00631                 }
00632         }
00633         if (y1 < y2) {
00634                 SWAP(y1, y2);
00635                 SWAP(x1, x3);
00636                 SWAP(x2, x4);
00637         }
00638         if (x1 > x2 && x3 > x4){
00639                 SWAP(x1, x2);
00640                 SWAP(x3, x4);
00641         }
00642         return( _QQZZ_trapz_fill(x1,x2,x3,x4, y1,y2, wbs, c, mode) );
00643 }
00644
```

GSUB.C

```
00644
00645 /*************************************************************
00646  *                                                           *
00647  * イメージ転送(GDCRAM->CPURAM)                              *
00648  *                                                           *
00649  *************************************************************/
00650 GLOBAL VOID
00651 _QQZZ_get(x, y, dh, dv, buf, pitch, disp)
00652 int             x, y;
00653 int             dh, dv;
00654 u_short *buf;
00655 int             pitch;
00656 u_long  disp;
00657 {
00658         register u_short        *dt;
00659         register int    pc, lc, dc, dp;
00660         u_short *pltop;
00661         u_short cmd;
00662         int     wc;             /* 転送ワード数／ライン */
00663         int     nextline;
00664         int     nextplane;
00665
00666         pltop = buf;
00667         wc = (dh+15)/16;
00668         nextline = pitch-wc;
00669         nextplane = disp-wc;
00670         dp = _QQZZ_CURscreen->scr_planecnt;
00671
00672         _QQZZ_statbusy(3);              /* Wait until(PPBUSY && DPBUSY==0) */
00673         _v_put_reg(R_PDISPSL, L_regW(_QQZZ_CURscreen->scr_displace));
00674         _v_put_regb(R_PDISPSH_B, H_regW(_QQZZ_CURscreen->scr_displace));
00675         _v_put_reg(R_PITCHS, _QQZZ_CURscreen->scr_pitch);
00676         _v_put_reg(R_X, x);
00677         _v_put_reg(R_Y, y);
00678         _v_put_reg(R_DH, dh-1);
00679         _v_put_reg(R_DV, dv-1);
00680         /* M->M */
00681         _v_put_reg(R_COMMAND, 0x9A0F);
00682         for(lc = 0; lc < dv; lc += 1) {
00683                 dt = pltop;
00684                 for(dc = 0; dc < dp; dc += 1) {
00685                         for(pc = 0; pc < wc; pc += 1) {
00686                                 *dt++ = _v_get_reg(R_PGPORT);
00687                         }
00688                         dt += nextplane;
00689                 }
00690                 pltop += pitch;
00691         }
00692 }
00693
```

GSUB.C

```
00693
00694  /*****************************************************************
00695   *                                                               *
00696   * イメージ転送(CPURAM->GDCRAM)                                  *
00697   *                                                               *
00698   *****************************************************************/
00699  GLOBAL VOID
00700  _QQZZ_put(x, y, dh, dv, dp, buf, pitch, disp, c, mode, tform)
00701  int             x, y;
00702  int             dh, dv;
00703  int             dp;
00704  u_short *buf;
00705  int             pitch;
00706  u_long  disp;
00707  int             c;
00708  int             mode;
00709  int             tform;
00710  {
00711          register u_short        *dt;
00712          register int    pc, lc, dc;
00713          u_short *pltop;
00714          u_short cmd;
00715          int     wc;                     /* 転送ワード数／ライン */
00716          int     nextline;
00717          int     nextplane;
00718
00719          pltop = buf;
00720          if(tform >= 0) {
00721                  pltop += (u_long)disp*tform;
00722                  dp = 1;
00723          }
00724          wc = (dh+15)/16;
00725          nextline = pitch-wc;
00726          nextplane = (int)disp-wc;
00727          _QQZZ_statbusy(3);      /* Wait until(PPBUSY && DPBUSY==0) */
00728          _v_put_reg(R_X, x);
00729          _v_put_reg(R_Y, y);
00730          _v_put_reg(R_DH, dh-1);
00731          _v_put_reg(R_DV, dv-1);
00732          _v_put_reg(R_PLANES, c);
00733          _v_put_reg(R_MOD10, mode);
00734          if(dp < 2) {
00735                  /* S->M */
00736                  _v_put_reg(R_COMMAND, 0x980B);
00737                  dt = pltop;
00738                  for(lc = 0; lc < dv; lc += 1) {
00739                          for(pc = 0; pc < wc; pc += 1) {
00740                                  _v_put_reg(R_PGPORT, *dt++);
00741                          }
00742                          dt += nextline;
00743                  }
00744          } else {
00745                  /* M->M */
00746                  _v_put_reg(R_COMMAND, 0x980F);
00747                  for(lc = 0; lc < dv; lc += 1) {
00748                          dt = pltop;
00749                          for(dc = 0; dc < dp; dc += 1) {
00750                                  for(pc = 0; pc < wc; pc += 1) {
00751                                          _v_put_reg(R_PGPORT, *dt++);
00752                                  }
00753                                  dt += nextplane;
00754                          }
00755                          pltop += pitch;
00756                  }
00757          }
00758  }
00759
```

GSUB.C

```
00759
00760 /*****************************************************************
00761  *                                                               *
00762  * シフトＪＩＳコード -> ＪＩＳコード変換                        *
00763  *                                                               *
00764  *****************************************************************/
00765 u_int
00766 shift2jis(code)
00767 u_int   code;
00768 {
00769 REG     int             ch, cl;
00770
00771         ch = code >> 8;
00772         cl = code & 0xff;
00773         if ( (ch = code >>   8) <  0x81) return(code);
00774         if ( (cl = code & 0xff) >= 0x80) --cl;
00775
00776         if ( ch >= 0xe0) ch -= 0x40;
00777         ch -= 0x81;
00778         ch += ch;
00779         if (cl >= 0x9e) {
00780                 ch++;
00781                 cl -= 0x5e;
00782         }
00783         ch +=.0x21;
00784         cl -= 0x1f;
00785         return( (ch << 8) | cl );
00786 }
00787
```

GSUB.C

```
00787
00788 /*****************************************************************
00789  *                                                               *
00790  * 文字コードアドレス計算                                        *
00791  *                                                               *
00792  *****************************************************************/
00793 GLOBAL int
00794 _QQ_code2addr(code, wd, hi, fonttop, pitch)
00795 u_int   code;
00796 u_long  *fonttop;
00797 u_short *pitch;
00798 u_short *wd, *hi;
00799 {
00800 REG     u_int           u, a1, a2;
00801         u_long          adr, l;
00802         int             halfwidth;
00803
00804         adr = KANJI_TOP;
00805         u   = code;
00806         a1  = u >> 8;
00807         a2  = u & 0xff;
00808         if (a1 == 0) {  /* 半角 */
00809                 if((a2 >= 0x21) && (a2 <= 0x7f)) {
00810                         code = 0x2900 | a2;
00811                 } else if((a2 >= 0xa1) && (a2 <= 0xdf)) {
00812                         code = 0x2a00 | (a2-0x80);
00813                 } else {
00814                         code = 0x2121;
00815                 }
00816                 *wd = CHR_SIZAH;
00817                 halfwidth = 0;
00818         } else {        /* 全角 */
00819                 *wd = CHR_SIZKH;
00820                 halfwidth = 0;
00821         }
00822         *hi = CHR_SIZV;
00823         l = (u_long)code << 4;
00824         *fonttop = adr + l;
00825         *pitch = FONTPITCH;
00826
00827         return(halfwidth);
00828 }
00829
```

保守／廃止

GSUB.C

```
00829
00830 /*************************************************************
00831  *                                                           *
00832  * 多目的コピー (A_COPY_ACタイプ)                            *
00833  *                                                           *
00834  *************************************************************/
00835 GLOBAL VOID
00836 _QQ_mightycopy(top, boff, wd, hi, pitch, disp, x, y,
00837                                 shrinkh, magh, shrinkv, magv, c, mod, flag, args)
00838 u_long  top;
00839 int             boff;
00840 int             wd, hi;
00841 int             pitch;
00842 u_long  disp;
00843 int             x, y;
00844 u_int   shrinkh, shrinkv;
00845 u_int   magh, magv;
00846 u_int   c;
00847 u_int   mod;
00848 REG u_int       flag;
00849 u_int   *args;
00850 {
00851 REG     int     cmd;
00852         u_int   mag;
00853
00854         /* 合成倍率コードの作成 */
00855         mag = ((magh & 0x0F) << 4) | (magv & 0x0F);
00856
00857         /* AGDCステイタスチェック */
00858         _QQZZ_statbusy(1);      /* Wait until(PPBUSY==0) */
00859
00860         if(flag & SR_MULTI) {
00861                 /* マルチソース->マルチデストネーション */
00862                 cmd = 0x0c;
00863         } else {
00864                 /* シングルソース->マルチデストネーション(Default) */
00865                 cmd = 0x08;
00866         }
00867         switch ( flag & TYPEBITS ) {
00868          case FRES_COPY :
00869                 if (args[0]>=0 && args[1]==0 && args[2]==0 && args[3]<=0) {
00870                         goto ES_ONLY;
00871                 }
00872                 _v_put_reg(R_DX, args[0]);
00873                 _v_put_reg(R_DY, args[1]);
00874                 _v_put_reg(R_XE, args[2]);
00875                 _v_put_reg(R_YE, args[3]);
00876                 cmd |= 0x12;
00877                 if(!(flag & NO_MAG) && (mag != 0xff)) {
00878                         if(shrinkh) {
00879                                 cmd |= 0x80;
00880                         }
00881                         if(shrinkv) {
00882                                 cmd |= 0x40;
00883                         }
00884                 }
00885                 if(!(flag & NO_FS)) {
00886                         cmd |= 0x20;
00887                 }
00888                 break;
00889          case SLANT_COPY :
00890                 if(args[0]) {
00891                         _v_put_reg(R_DX, args[0]);
00892                         cmd |= 0x01;
00893                         if(flag & ESE_BIT) {
00894                                 cmd |= 0x80;
00895                         }
00896                         if(flag & REV_BIT) {
00897                                 cmd |= 0x40;
```

保守／廃止

GSUB.C

```
00898                           }
00899                           if(flag & ROT_BIT) {
00900                                   cmd |= 0x20;
00901                           }
00902                           break;
00903                   }
00904                   /* FALL INTO... */
00905           default :
00906   ES_ONLY :
00907                   if (!(flag & NO_MAG) && (mag != 0xff)) {
00908                           cmd |= 0x03;
00909                           if(shrinkh) {
00910                                   cmd |= 0x80;
00911                           }
00912                           if(shrinkv) {
00913                                   cmd |= 0x10;
00914                           }
00915                   } else {
00916                           if(flag & FAST_BIT) {
00917                                   cmd |= 0x02;
00918                           }
00919                           if(flag & D90_BIT) {
00920                                   cmd |= 0x10;
00921                           }
00922                           if(flag & ESE_BIT) {
00923                                   cmd |= 0x80;
00924                           }
00925                   }
00926                   if(flag & REV_BIT) {
00927                           cmd |= 0x40;
00928                   }
00929                   if(flag & ROT_BIT) {
00930                           cmd |= 0x20;
00931                   }
00932                   break;
00933           }
00934
00935           _v_put_reg(R_EAD2L, L_regW(top));
00936           _v_put_regb(R_EAD2H_B, H_regW(top));
00937           _v_put_regb(R_dAD2_B, boff);
00938           _v_put_reg(R_DH, wd-1);
00939           _v_put_reg(R_DV, hi-1);
00940           _v_put_reg(R_PITCHS, pitch);
00941           _v_put_reg(R_PDISPSL, L_regW(disp));
00942           _v_put_regb(R_PDISPSH_B, H_regW(disp));
00943           _v_put_reg(R_X, x);
00944           _v_put_reg(R_Y, y);
00945           _QQZZ_copy_AC(cmd, mag, c, mod);
00946
00947           return;
00948   }
00949
```

27

GSUB.C

```
00949
00950 /*****************************************************************
00951  *                                                               *
00952  * 高速単純コピー (A_COPY_ACタイプ)                              *
00953  *                                                               *
00954  *****************************************************************/
00955 GLOBAL VOID
00956 _QQ_stratecopy(top, boff, wd, hi, pitch, disp, x, y, c, mod, flag)
00957 u_long  top;
00958 int             boff;
00959 int             wd, hi;
00960 int             pitch;
00961 u_long  disp;
00962 int             x, y;
00963 u_int   c;
00964 u_int   mod;
00965 REG u_int       flag;
00966 {
00967 REG     int     cmd;
00968         int     halfwidth;
00969         u_int   mag;
00970
00971         /* AGDCステイタスチェック */
00972         _QQZZ_statbusy(1);      /* Wait until(PPBUSY==0) */
00973
00974         if(flag & SR_MULTI) {
00975                 /* マルチソース->マルチデストネーション */
00976                 cmd = 0x0c;
00977         } else {
00978                 /* シングルソース->マルチデストネーション(Default) */
00979                 cmd = 0x08;
00980         }
00981         if(flag & FAST_BIT) {
00982                 cmd |= 0x02;
00983         }
00984         if(flag & D90_BIT) {
00985                 cmd |= 0x10;
00986         }
00987         if(flag & ESE_BIT) {
00988                 cmd |= 0x80;
00989         }
00990         if(flag & REV_BIT) {
00991                 cmd |= 0x40;
00992         }
00993         if(flag & ROT_BIT) {
00994                 cmd |= 0x20;
00995         }
00996
00997         _v_put_reg(R_EAD2L, L_regW(top));
00998         _v_put_regb(R_EAD2H_B, H_regW(top));
00999         _v_put_regb(R_dAD2_B, boff);
01000         _v_put_reg(R_DH, wd-1);
01001         _v_put_reg(R_DV, hi-1);
01002         _v_put_reg(R_PITCHS, pitch);
01003         _v_put_reg(R_PDISPSL, L_regW(disp));
01004         _v_put_regb(R_PDISPSH_B, H_regW(disp));
01005         _v_put_reg(R_X, x);
01006         _v_put_reg(R_Y, y);
01007         _QQZZ_copy_AC(cmd, mag, c, mod);
01008
01009         return;
01010 }
01011
```

GSUB.C

```
01011
01012 /*****************************************************************
01013  *                                                               *
01014  * 始点, 終点値の取得                                            *
01015  *                                                               *
01016  *****************************************************************/
01017 GLOBAL VOID
01018 _QQ_arc_se(xsp, ysp, xep, yep)
01019 int     *xsp, *ysp, *xep, *yep;
01020 {
01021         _QQZZ_statbusy(1);              /* Wait until(PPBUSY==0) */
01022         *xsp = _v_get_reg(R_XS);
01023         *ysp = _v_get_reg(R_YS);
01024         *xep = _v_get_reg(R_XE);
01025         *yep = _v_get_reg(R_YE);
01026 }
01027
```

GSUB.C

```
01027
01028 /*****************************************************************
01029  *                                                               *
01030  * ３２ビット整数の開平（切捨て版）                              *
01031  *                                                               *
01032  *****************************************************************/
01033 unsigned short
01034 _QQZZ_lsqrtc(value)
01035 unsigned long   value;
01036 {
01037 #define ANSWERDIGIT     16
01038 #define VALUEDIGIT      32
01039 typedef unsigned long   UL;
01040 typedef unsigned short  US;
01041         register UL     value_mask;     /* 被開平値用マスク
01042                                            演算ごとに右に２ビットシフトする */
01043         register UL     compared;       /* 被開平値比較値用レジスタ */
01044         register UL     comparer;       /* 被開平値と比較する途中経過値用
01045                                            演算ごとに右に２ビットシフトする */
01046         register UL     comparer_mask;  /* comparerの値を作るためのマスク
01047                                            演算ごとに右に２ビットシフトする */
01048         register US     answer_mask;    /* 答えを作るためのマスク
01049                                            演算ごとに右に１ビットシフトされる*/
01050         register US     answer;         /* 演算結果を納めるレジスタ */
01051         register short  digit;          /* 処理した桁数 */
01052
01053         value_mask = 0xc0000000L;
01054         answer_mask = 0x8000;
01055         comparer_mask = 0x40000000L;
01056         compared = 0;
01057         comparer = 0;
01058         answer = 0;
01059         for(digit = 0; digit < ANSWERDIGIT; digit += 1) {
01060                 compared |= (value & value_mask);
01061                 if(compared < (comparer | comparer_mask)) {
01062                         /* この桁の答えが0になる場合 */
01063                 } else {
01064                         /* この桁の答えが1になる場合 */
01065                         answer |= answer_mask;
01066                         compared -= (comparer | comparer_mask);
01067                         comparer += (comparer_mask << 1);
01068                 }
01069                 comparer_mask  >>= 2;
01070                 comparer >>= 1;
01071                 value_mask >>= 2;
01072                 answer_mask >>= 1;
01073         }
01074
01075         return(answer);
01076 #undef  ANSWERDIGIT
01077 #undef  VALUEDIGIT
01078 }
01079
```

保守／廃止

GSUB.C

```
01079
01080 /*****************************************************************
01081  *                                                               *
01082  * ３２ビット整数の開平（四捨五入版）                            *
01083  *                                                               *
01084  *****************************************************************/
01085 unsigned short
01086 _QQZZ_lsqrtr(value)
01087 unsigned long   value;
01088 {
01089 #define ANSWERDIGIT     16
01090 #define VALUEDIGIT      32
01091 typedef unsigned long   UL;
01092 typedef unsigned short  US;
01093         register UL     value_mask;     /* 被開平値用マスク
01094                                            演算ごとに右に２ビットシフトする */
01095         register UL     compared;       /* 被開平値比較値用レジスタ */
01096         register UL     comparer;       /* 被開平値と比較する途中経過値用
01097                                            演算ごとに右に２ビットシフトする */
01098         register UL     comparer_mask;  /* comparerの値を作るためのマスク
01099                                            演算ごとに右に２ビットシフトする */
01100         register US     answer_mask;    /* 答えを作るためのマスク
01101                                            演算ごとに右に１ビットシフトされる*/
01102         register US     answer;         /* 演算結果を納めるレジスタ */
01103         register short  digit;          /* 処理した桁数 */
01104
01105         value_mask = 0xc0000000L;
01106         answer_mask = 0x8000;
01107         comparer_mask = 0x40000000L;
01108         compared = 0;
01109         comparer = 0;
01110         answer = 0;
01111         for(digit = 0; digit < ANSWERDIGIT-1; digit += 1) {
01112                 compared |= (value & value_mask);
01113                 if(compared < (comparer | comparer_mask)) {
01114                         /* この桁の答えが0になる場合 */
01115                 } else {
01116                         /* この桁の答えが1になる場合 */
01117                         answer |= answer_mask;
01118                         compared -= (comparer | comparer_mask);
01119                         comparer += (comparer_mask << 1);
01120                 }
01121                 comparer_mask  >>= 2;
01122                 comparer >>= 1;
01123                 value_mask >>= 2;
01124                 answer_mask >>= 1;
01125         }
01126         /* 最後の一桁 */
01127         compared |= (value & value_mask);
01128         if(compared < (comparer | comparer_mask)) {
01129                 /* この桁の答えが0になる場合 */
01130         } else {
01131                 /* この桁の答えが1になる場合 */
01132                 answer |= answer_mask;
01133                 compared -= (comparer | comparer_mask);
01134                 comparer += (comparer_mask << 1);
01135         }
01136         /* 四捨五入判定 */
01137         compared <<= 2;
01138         comparer <<= 1;
01139         if(compared >= (comparer | 1)) {
01140                 /* 繰上がる場合 */
01141                 answer += 1;
01142         }
01143
01144         return(answer);
01145 #undef  ANSWERDIGIT
01146 #undef  VALUEDIGIT
01147 }
01148
```

保守／廃止

GSUB.C

```
01148
01149 /*****************************************************************
01150  *                                                               *
01151  * 円と直線（円の中心からの半直線）との交点計算（直線限定あり）  *
01152  *                                                               *
01153  *****************************************************************/
01154 GLOBAL  VOID
01155 _QQZZ_pipoint(radius, angle_x, angle_y, px, py)
01156 short   radius;
01157 short   angle_x, angle_y;
01158 short   *px, *py;
01159 {
01160         unsigned short  _QQZZ_lsqrtr();
01161         long    scale;
01162         short   base;
01163
01164         base = (short)_QQZZ_lsqrtr((long)angle_x*angle_x+(long)angle_y*angle_y);
01165         scale = (long)radius*angle_x;
01166         *px = (short)(scale/base)+(((2*(scale%base)) > base) ? 1 : 0);
01167         scale = (long)radius*angle_y;
01168         *py = (short)(scale/base)+(((2*(scale%base)) > base) ? 1 : 0);
01169 }
01170
```

GSUB.C

```
01170
01171 /*****************************************************************
01172  *                                                               *
01173  * 円と直線（円の中心からの半直線）との交点計算（直線限定なし）  *
01174  *                                                               *
01175  *****************************************************************/
01176 _QQZZ_pipoint_all(radius, angle_x, angle_y, px, py)
01177 short   radius;
01178 short   angle_x, angle_y;
01179 short   *px, *py;
01180 {
01181         int     abs_px, abs_py;
01182
01183         abs_px = abs(*px);
01184         abs_py = abs(*py);
01185         if(abs_py > abs_px) {
01186                 /* 直線の傾き（絶対値）が1を越える場合 */
01187                 _QQZZ_pipoint(radius, angle_y, angle_x, py, px);
01188         } else {
01189                 /* 直線の傾き（絶対値）が1以下の場合 */
01190                 _QQZZ_pipoint(radius, angle_x, angle_y, px, py);
01191         }
01192
01193         return(0);
01194 }
01195
```

GSUB.C

```
01195
01196 /*****************************************************************
01197  *                                                               *
01198  * vsyncになるのを待つ                                          *
01199  *                                                               *
01200  *****************************************************************/
01201 GLOBAL  int
01202 _QQZZ_catchvsync()
01203 {
01204         int     busy;
01205         do {
01206                 busy = _v_get_reg(R_STATUS);
01207         } while(busy & 0x10);
01208         do {
01209                 busy = _v_get_reg(R_STATUS);
01210         } while((busy & 0x10) == 0);
01211
01212         return(busy);
01213 }
```

A.8　[WINDOW.C]

WINDOW.C

```
00001 #include <stdio.h>
00002 #include <math.h>
00003 #include "agdcop.h"
00004 #include "agdccol.h"
00005
00006 #define LWIDTH  1120
00007 #define LHEIGHT 750
00008 #define LDEPTH  4
00009 #define PWIDTH  640
00010 #define PHEIGHT 400
00011 #define PDEPTH  4
00012 #define PX_MAX  (PWIDTH-1)
00013 #define PY_MAX  (PHEIGHT-1)
00014 #define px(x)   (short)((x)*(long)PWIDTH/LWIDTH)
00015 #define py(y)   (short)((y)*(long)PHEIGHT/LHEIGHT)
00016
00017 #define LIMIT   (LHEIGHT/2)
00018 #define WINDSIZ (4*LIMIT/5)
00019 #define W1_CX   (LWIDTH/4)
00020 #define W1_CY   (LHEIGHT/4)
00021 #define W2_CX   (int)(3*(long)LWIDTH/4)
00022 #define W2_CY   (LHEIGHT/4)
00023 #define W3_CX   (LWIDTH/4)
00024 #define W3_CY   (int)(3*(long)LHEIGHT/4)
00025 #define W4_CX   (int)(3*(long)LWIDTH/4)
00026 #define W4_CY   (int)(3*(long)LHEIGHT/4)
00027
00028 static  int             xx[LIMIT+5], yy[LIMIT+5];
00029 static  int             w_scrn;
00030
00031 main(argc, argv)
00032 int             argc;
00033 char    *argv[];
00034 {
00035         int             win_sw;
00036         int  i, j, k, rn;
00037
00038         if(argc > 1) {
00039                 win_sw = 1 << atoi(argv[1]);
00040         } else {
00041                 win_sw = 0;
00042         }
00043         gInit(0, PDEPTH);
00044         w_scrn = gCreatePage(PWIDTH, PHEIGHT, PDEPTH);
00045         if(w_scrn < 0) {
00046                 printf("[31m ワーク画面が作成できない！[0m¥n");
00047                 exit(-1);
00048         }
00049         rn = LIMIT;
00050         initarc(rn);
00051         gDrawPage(w_scrn);
00052         demoscreen();
00053         gDrawPage(0);
00054
00055         for(;;){
00056                 if(win_sw & 1) wind1();
00057                 if(win_sw & 2) wind2();
00058                 if(win_sw & 4) wind3();
00059                 if(win_sw & 8) wind4();
00060                 i = bdos(6,0xff) & 0xff;
00061                 if(i == '¥033') break;
00062                 if(i == ' '){
00063                         shrink();
00064                 }
00065                 if(i == 0x0d) {
00066                         do {
00067                                 i = bdos(6,0xff) & 0xff;
00068                         } while(i != 0x0d);
00069                 }
```

保守／廃止

WINDOW.C

```
00070                   blinking();
00071           }
00072           gEnd();
00073   }
00074
00075   int     magv_s=16, magv_b=16;           /* 垂直倍率=magv_s/magv_b */
00076   int     magh_s=16, magh_b=16;           /* 水平倍率=magh_s/magh_b */
00077
00078   shrink()
00079   {
00080   static  int     f=0;
00081
00082           gDrawPage(0);
00083           if(f==0){
00084                   gClearScreen();
00085                   magv_s = magh_s = 16;
00086                   magv_b = magh_b = 13;
00087                   f = 1;
00088           }else if(f==1){
00089                   gClearScreen();
00090                   magv_s = magh_s = 10;
00091                   magv_b = magh_b = 16;
00092                   f = 2;
00093           }else{
00094                   gClearScreen();
00095                   magv_s = magh_s = 16;
00096                   magv_b = magh_s = 16;
00097                   f = 0;
00098           }
00099   }
00100
00101   blinking()
00102   {
00103   static  int     c=0;
00104   static  int     n=0;
00105
00106           if(n++ < 8) return;
00107           n = 0;
00108           gDrawPage(w_scrn);
00109           gSetDrawCode(c);
00110           gSetMode(ST_PSET);
00111           gFillCircle(px(320+180),py(200),py(20));
00112           gDrawPage(0);
00113           if(c==1) c = 0;
00114           else    c = 1;
00115   }
00116
00117   wind1()
00118   {
00119   static  int     n=0;
00120           int     dh, dv;
00121           int     sx, sy, dx, dy;
00122
00123           dh = (int)(((long)magh_s*WINDSIZ)/magh_b);
00124           dv = (int)(((long)magv_s*WINDSIZ)/magv_b);
00125           sx = xx[n];
00126           sy = yy[n]-WINDSIZ;
00127           dx = W1_CX-dh/2;
00128           dy = W1_CY-dv/2;
00129
00130           if(++n >= LIMIT) n = 0;
00131           gDrawPage(0);
00132           gSetDrawCode(P_ALL);
00133           gSetMode(FL_PSET);
00134           gCopyScr(w_scrn, px(sx),py(sy), px(WINDSIZ),py(WINDSIZ),
00135                                           px(dx),py(dy), px(dh), py(dv), -1);
00136   }
00137
00138   wind2()
```

保守／廃止

WINDOW.C

```
00139 {
00140 static  int      n=0;
00141         int      dh, dv;
00142         int      sx, sy, dx, dy;
00143
00144         dh = (int)(((long)magh_s*WINDSIZ)/magh_b);
00145         dv = (int)(((long)magv_s*WINDSIZ)/magv_b);
00146         sx = xx[LIMIT-n-1];
00147         sy = yy[LIMIT-n-1]-WINDSIZ;
00148         dx = W2_CX-dh/2;
00149         dy = W2_CY-dv/2;
00150
00151         n += 2;
00152         if(n >= LIMIT) n = 0;
00153         gDrawPage(0);
00154         gSetDrawCode(P_ALL);
00155         gSetMode(FL_PSET);
00156         gCopyScr(w_scrn, px(sx),py(sy), px(WINDSIZ),py(WINDSIZ),
00157                                          px(dx),py(dy), px(dh), py(dv), -1);
00158 }
00159
00160 wind3()
00161 {
00162 static  int      n=0,dir= 7;
00163 static  int      x=200,y=WINDSIZ;
00164         int      dh, dv;
00165         int      sx, sy, dx, dy;
00166
00167         dh = (int)(((long)magh_s*WINDSIZ)/magh_b);
00168         dv = (int)(((long)magv_s*WINDSIZ)/magv_b);
00169         sx = x;
00170         sy = y-WINDSIZ;
00171         dx = W3_CX-dh/2;
00172         dy = W3_CY-dv/2;
00173
00174         gDrawPage(0);
00175         gSetDrawCode(P_ALL);
00176         gSetMode(FL_PSET);
00177         gCopyScr(w_scrn, px(sx),py(sy), px(WINDSIZ),py(WINDSIZ),
00178                                          px(dx),py(dy), px(dh), py(dv), -1);
00179
00180         if(dir > 0){
00181                 if(y+dir > 749){
00182                         dir= -dir;
00183                 }
00184         }else{
00185                 if(y-WINDSIZ+dir <= 0){
00186                         dir= -dir;
00187                 }
00188         }
00189         x += dir; y += dir;
00190 }
00191
00192 wind4()
00193 {
00194 static  int      n=0,dir= 7;
00195 static  int      x=0,y=450;
00196         int      dh, dv;
00197         int      sx, sy, dx, dy;
00198
00199         dh = (int)(((long)magh_s*WINDSIZ)/magh_b);
00200         dv = (int)(((long)magv_s*WINDSIZ)/magv_b);
00201         sx = x;
00202         sy = y-WINDSIZ;
00203         dx = W4_CX-dh/2;
00204         dy = W4_CY-dv/2;
00205
00206         gDrawPage(0);
00207         gSetDrawCode(P_ALL);
```

WINDOW.C

```
00208           gSetMode(FL_PSET);
00209           gCopyScr(w_scrn, px(sx),py(sy), px(WINDSIZ),py(WINDSIZ),
00210                                           px(dx),py(dy), px(dh), py(dv), -1);
00211
00212           if(dir > 0){
00213                   if(x+WINDSIZ+dir > 900){
00214                           dir= -dir;
00215                   }
00216           }else{
00217                   if(x+dir < 0){
00218                           dir= -dir;
00219                   }
00220           }
00221           x += dir;
00222   }
00223
00224   demoscreen()
00225   {
00226           int     i,j,k,c;
00227
00228           gSetDrawCode(4);
00229           gSetMode(ST_PSET);
00230           gFillRectangle(px(0),py(0),px(1119),py(749));
00231           c = 0;
00232           j = 0;
00233           for(i=0;i<1120;i++){
00234                   gSetDrawCode(c);
00235                   gSetMode(ST_PSET);
00236                   gLine(px(i),py(749),px(640),py(i/5));
00237                   if(j++ > 40){
00238                           c++;
00239                           c &= 0x7;
00240                           j=0;
00241                   }
00242           }
00243           for(i=1;i<1000;i+=20){
00244                   gSetDrawCode(5);
00245                   gSetMode(ST_PSET);
00246                   gCircle(px(560),py(375),py(i));
00247           }
00248           demo7();
00249           gSetMode(ST_PSET);
00250           gSetDrawCode(1);
00251           gPutText(px(400),py(746-30*13),"MULTI  WINDOW  DEMO");
00252           gSetDrawCode(2);
00253           gPutText(px(400),py(746-30*14),"MULTI  WINDOW  DEMO");
00254           gSetDrawCode(7);
00255           gPutText(px(400),py(746-30*15),"MULTI  WINDOW  DEMO");
00256   }
00257
00258   initarc(rn)
00259   {
00260           int     i, j, k;
00261           float   h, stp;
00262
00263           j = 0;
00264           stp = 3.14159/(float)rn*2.0;
00265           for(h=0;h<=6.3;h+=stp){
00266                   xx[j] = 200*cos(h)+550;
00267                   yy[j] = 200*sin(h)+540;
00268   #if     0
00269                   /* for debug */
00270                   gSetDrawCode(6);
00271                   gDot(px(xx[j]), py(yy[j]));
00272                   /* end for debug */
00273   #endif
00274                   j++;
00275           }
00276   }
```

保守／廃止

WINDOW.C

```
00277
00278 demo7()
00279 {
00280         int     x1, x2, x3, x4;
00281         int     y1, y2, y3, y4;
00282         int     i, j, k;
00283
00284         for (k=0; k<400; k+=7){
00285                 x1=k+100; y1=0; x2=100+399; y2=k;
00286                 x3=100+399-k; y3=399; x4=100; y4=399-k;
00287                 demosub7(x1,x2,x3,x4,y1,y2,y3,y4);
00288         }
00289         for (k=0; k<200; k+=5){
00290                 x1=k+200; y1=100; x2=200+199; y2=100+k;
00291                 x3=200+199-k; y3=299; x4=200; y4=299-k;
00292                 demosub7(x1,x2,x3,x4,y1,y2,y3,y4);
00293         }
00294         for (k=0; k<100; k+=3){
00295                 x1=k+250; y1=150; x2=250+99; y2=150+k;
00296                 x3=250+99-k; y3=249; x4=250; y4=249-k;
00297                 demosub7(x1,x2,x3,x4,y1,y2,y3,y4);
00298         }
00299 }
00300
00301 demosub7(x1,x2,x3,x4,y1,y2,y3,y4)
00302 {
00303         x1 += 200;
00304         x2 += 200;
00305         x3 += 200;
00306         x4 += 200;
00307
00308         gSetMode(ST_PSET);
00309         gSetDrawCode(3);
00310         gLine(px(x1), py(y1), px(x2), py(y2));
00311         gSetDrawCode(6);
00312         gLine(px(x2), py(y2), px(x3), py(y3));
00313         gSetDrawCode(7);
00314         gLine(px(x3), py(y3), px(x4), py(y4));
00315         gSetDrawCode(1);
00316         gLine(px(x4), py(y4), px(x1), py(y1));
00317 }
```

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（II） ソフトウエア編

（IEA-711（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

札幌支社	(011)231−0161
仙台支社	(022)261−5511
盛岡支店	(0196)51−4344
山形支店	(0236)23−5511
山形支店いわき出張所	(0246)21−5511
長岡支店	(0258)36−2155
水戸支店	(0292)26−1717
横浜支社	(045)324−5511
高崎支店	(0273)26−1255
太田支店	(0276)46−4011
宇都宮支社	(0286)21−2281
小山支店	(0285)24−5011
長野支店	(0262)35−1444
松本支店	(0263)35−1666
諏訪支店	(0266)53−5350
甲府支店	(0552)24−4141
大宮支社	(048)641−1411

立川支社	(0425)26−0911
千葉支社	(043)227−9084
静岡支社	(054)255−2211
沼津支店	(0559)63−4455
浜松支店	(053)452−2711
北陸支社	(0762)23−1621
福井支店	(0776)22−1866
富山支店	(0764)31−8461
京都支社	(075)344−7824
神戸支社	(078)332−3311
中国支社	(082)242−5504
鳥取支店	(0857)27−5311
岡山支店	(086)225−4455
四国支社	(0878)36−1200
新居浜支店	(0897)32−5001
松山支店	(0899)45−4111
九州支社	(092)271−7700
北九州支店	(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)548−7918
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）