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。

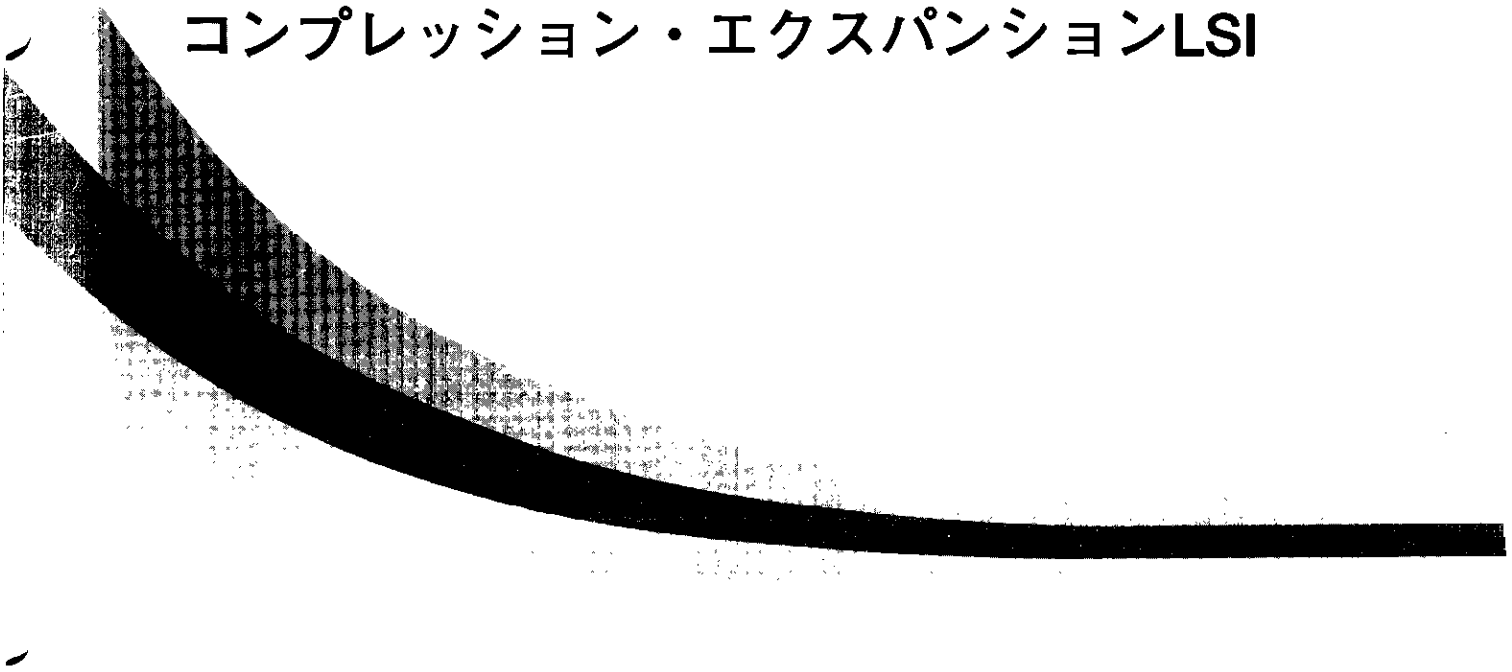
注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

アプリケーション・ノート

RENESAS

μPD72187A

アドバンスト・バイレベル・イメージ・コンプレッション・エクスパンションLSI



資料番号　S12027JJ2V0AN00（第2版）
発行年月　March 1997　NS

目次要約

CMOSデバイスの一般的注意事項

①静電気対策（MOS全般）

注意 MOSデバイス取り扱いの際は静電気防止を心がけてください。

MOSデバイスは強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には，NECが出荷梱包に使用している導電性のトレーやマガジン・ケース，または導電性の緩衝材，金属ケースなどを利用し，組み立て工程にはアースを施してください。プラスチック板上に放置したり，端子を触ったりしないでください。

また，MOSデバイスを実装したボードについても同様の扱いをしてください。

②未使用入力の処理（CMOS特有）

注意 CMOSデバイスの入力レベルは固定してください。

バイポーラやNMOSのデバイスと異なり，CMOSデバイスの入力に何も接続しない状態で動作させると，ノイズなどに起因する中間レベル入力が生じ，内部で貫通電流が流れて誤動作を引き起こす恐れがあります。プルアップかプルダウンによって入力レベルを固定してください。また，未使用端子が出力となる可能性（タイミングは規定しません）を考慮すると，個別に抵抗を介してV_{DD}またはGNDに接続することが有効です。

資料中に「未使用端子の処理」について記載のある製品については，その内容を守ってください。

③初期化以前の状態（MOS全般）

注意 電源投入時，MOSデバイスの初期状態は不定です。

分子レベルのイオン注入量等で特性が決定するため，初期状態は製造工程の管理外です。電源投入時の端子の出力状態や入出力設定，レジスタ内容などは保証しておりません。ただし，リセット動作やモード設定で定義している項目については，これらの動作ののちに保証の対象となります。

リセット機能を持つデバイスの電源投入後は，まずリセット動作を実行してください。

T.82, T.85 に関わる特許権について

ITU-T 勧告 T.82, T.85 に準拠したシステムについては，複数の特許権が存在しております。
これらの特許権に関する必要な権利処理は，お客様の方にてご対応いただきますようお願いいたします。
当社は，これらの特許権に関して一切責任を負いかねますのでご了承ください。

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。

本版で改訂または追加された主な箇所

箇　所	内　容
p.21 ～ p.38	プログラムを大幅に変更したため，第３章　基本関数の内容を変更。 章タイトルは，第３章　制御シーケンス・プログラムに変更。
p.77 ～ p.128	付録Ｃ　基本関数のプログラム・リストの内容を大幅に変更。 章タイトルは，付録Ｃ　制御シーケンス・プログラムのプログラム・リストに変更

本文欄外の★印は，本版で改訂された主な箇所を示しています。

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

はじめに

対象者 このアプリケーション・ノートは，２値画像データ圧縮／伸長用プロセッサμPD72187Aの機能を理解し，それを用いたファクシミリやプリンタなどのアプリケーション・システムを設計するユーザを対象とします。

目　的 このアプリケーション・ノートは，次の構成に示すμPD72187Aの持つハードウエア，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。

μPD72187A（アドバンスト・バイレベル・イメージ・コンプレッション・エクスパンションLSI）は，２値画像データを効果的に伝送，蓄積するために，２値画像が本来持っている統計的性質を利用し，２値画像データを符号化（圧縮）したり，復号化（伸長）したりします。符号化，および復号化の方式については，その国際標準がITU-Tにより定められており，μPD72187Aはこの国際標準に従って符号化／復号化を短時間で効率的に行います。

このアプリケーション・ノートでは，μPD72187Aを使用し，設計した回路について説明します。

構　成 このアプリケーション・ノートは，大きく分けて次の内容で構成されています。

- ●機能仕様
- ●回路の説明
- ●制御シーケンス・プログラム

読み方 このアプリケーション・ノートの読者は，論理回路やマイクロコンピュータに関する一般知識，およびＣ言語に関する基礎知識を必要とします。

一通りμPD72187Aのハードウエア，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。

特に断りのないかぎり，本文中ではμPD71071をDMA（ダイレクト・メモリ・アクセス）コントローラ，μPD71054をPTC（プログラマブル・タイマ／カウンタ），μPD71055をPIU（パラレル・インタフェース・ユニット）と記述します。

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，信号の名称に上線）

メモリ・マップのアドレス　：上部ー上位，下部ー下位

注　：本文中につけた注の説明

注意　：気をつけて読んでいただきたい内容

備考　：本文中の補足説明

数の表記　：２進数…××××または××××B

10進数…××××

16進数…××××H

関連資料　データ・シート　：S10962J

ユーザーズ・マニュアル　：S11056J

注　意　このアプリケーション・ノートに使用できるホスト・マシンは，PC-9801 VX以降のノーマル・モード，Cバス搭載機です。

目　次

図の目次

表の目次

第１章　機能仕様

ここでは，このアプリケーション・ノートで設計した回路について，その概要を説明します。

１．１　機能概要

このボードは，ホストとして当社のパーソナル・コンピュータ PC-9800 シリーズ（１．３．３　ホスト・マシンへの対応参照）を使用し，その拡張スロットに差し込めるようなボードの形にしてあります。

ホストの指示により，ボード上の μPD72187A は画像メモリに蓄えられた２値画像データの符号化（圧縮），および符号メモリに蓄えられた符号データの復号化（伸長）を行います。

画像メモリは 1M ワードとしています。基本的には μPD72187A が管理しますが，ボード上の DMA コントローラを通じてホスト CPU からのアクセスも可能です。

符号メモリは 256K ワードとしています。これは，ボード上の DMA コントローラを使用してホスト CPU が管理します。μPD72187A はいっさい管理しません。

このボードは，ホストからは I/O として認識され，DMA コントローラを通して画像，符号メモリのリード／ライト，DMA コントローラのレジスタのリード／ライトなどを行います。

表１－１　機能概要

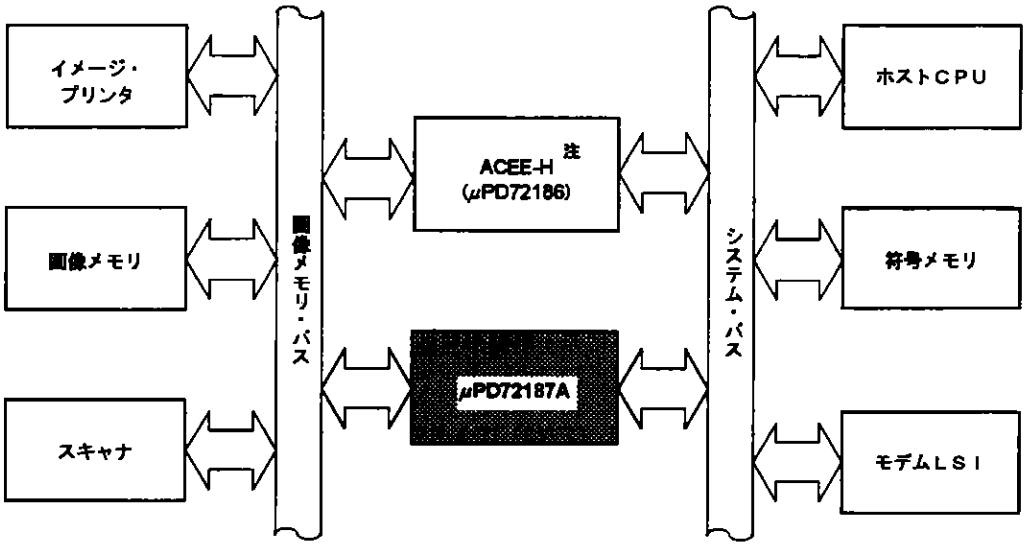
機　能	概　要
画像メモリ	SRAM:1M ワード ホストのメモリ領域から独立しており，基本的には μPD72187A が管理しますが，ボード上の DMA コントローラを通じてホスト CPU からも管理が可能です。
符号メモリ	SRAM:256K ワード ホストのメモリ領域から独立しており，ボード上の DMA コントローラを通じてホスト CPU が管理します。
ホスト・インタフェース	ホストからは I/O として認識されます。また，ホスト CPU は以下の動作を行うことができます。 ・μPD72187A のレジスタの読み書き ・DMA コントローラのレジスタの読み書き ・符号メモリの読み書き（DMA コントローラ経由） ・画像メモリの読み書き（DMA コントローラ経由） ・PTC レジスタの読み書き ・PIU レジスタの読み書き
動作クロック	μPD72187A　：20MHz DMA コントローラ　：8MHz PTC　：8MHz（カウンタ標準クロック）
タイマ	μPD72187A による符号化／復号化の処理時間の測定が可能です。

＜主要デバイス＞

μPD72187A		×1
μPD71071	DMA コントローラ	×1
μPD71054	PTC（プログラマブル・タイマ／カウンタ）	×1
μPD71055	PIU（パラレル・インタフェース・ユニット）	×1
μPD434008	SRAM	×6

１.２　システム構成例

図１－１　システム構成例



注　ACEE-H(μPD72186）は，MH,MR,MMR 方式に対応した２値画像圧縮／伸長用プロセッサです。

1.3　ホスト・インタフェース

1.3.1　I/O アドレスの割り付け

この回路はホスト側から見ると（１）～（５）に示すようなレジスタの集合に見えます。

（１）μPD72187A のレジスタ
（２）μPD71055（システム・レジスタ）の I/O マップ
（３）μPD71071（DMA コントローラ）のレジスタ
（４）μPD71054（タイマ）のレジスタ
（５）リセット・レジスタ（74ACT74）

なお，表中のＸ，Ｙについては，以下に示す制御が可能です。

Ｘ…ボード上のスイッチ（SW3）により任意に設定可能です。
Ｙ…ボード上のスイッチ（SW2）により任意に設定可能です。

（１）μPD72187A のレジスタ

JP5 の設定により μPD72187A のレジスタを直接 I/O マップに割り付ける「直接マッピング方式」とアドレス・レジスタを使用した「間接マッピング方式」とを選択できます。

表１－２　μPD72187A のレジスタ（直接マッピング方式）(1/2)

I/O アドレス	リード／ライト	機　　能
X0Y0H	R/W	コントロール・レジスタ読み出し／書き込み
X0Y1H	－	使用禁止
X0Y2H	R	ステータス・レジスタＬ読み出し
X0Y3H	R	ステータス・レジスタＨ読み出し
X0Y4H	R/W	モード・レジスタＬ読み出し／書き込み
X0Y5H	R/W	モード・レジスタＨ読み出し／書き込み
X0Y6H	W	画素数設定レジスタＬ書き込み
X0Y7H	W	画素数設定レジスタＨ書き込み
X0Y8H	W	ライン数設定レジスタＬ書き込み
X0Y9H	W	ライン数設定レジスタＨ書き込み
X0YAH	R/W	ビット・プレーン・ライン数設定レジスタＬ読み出し／書き込み
X0YBH	R/W	ビット・プレーン・ライン数設定レジスタＨ読み出し／書き込み
X0YCH	W	ストライプ・レジスタＬ書き込み
X0YDH	W	ストライプ・レジスタＨ書き込み
X0YEH	R/W	AT レジスタＬ読み出し／書き込み
X0YFH	W	AT レジスタＨ書き込み

表１－２　μPD72187A のレジスタ（直接マッピング方式）(2/2)

I/O アドレス	リード／ライト	機　　能
X1Y0H	R/W	AT ライン・レジスタ L 読み出し／書き込み
X1Y1H	W	AT ライン・レジスタ H 書き込み
X1Y2H	W	画像データ開始アドレス・レジスタ L 書き込み
X1Y3H	W	画像データ開始アドレス・レジスタ M 書き込み
X1Y4H	W	画像データ開始アドレス・レジスタ H 書き込み
X1Y5H	－	使用禁止
X1Y6H	R/W	画像データ終了アドレス・レジスタ L 読み出し／書き込み
X1Y7H	R/W	画像データ終了アドレス・レジスタ M 読み出し／書き込み
X1Y8H	R/W	画像データ終了アドレス・レジスタ H 読み出し／書き込み
X1Y9H	－	使用禁止
X1YAH	R/W	データ・バッファ・レジスタ L 読み出し／書き込み
X1YBH	R/W	データ・バッファ・レジスタ H 読み出し／書き込み
X1YCH	R/W	ホスト・バス・モード・レジスタ読み出し／書き込み
X1YDH	－	使用禁止
X1YEH	R/W	マーカ・コード・バッファ・レジスタ読み出し／書き込み
X1YFH	－	使用禁止
X2Y0H	R	マーカ・コード・ウインドウ・レジスタ L 読み出し
X2Y1H	R	マーカ・コード・ウインドウ・レジスタ H 読み出し
X2Y2H	R/W	コマンド・レジスタ１読み出し／書き込み
X2Y3H	－	使用禁止
X2Y4H	R/W	コマンド・レジスタ２読み出し／書き込み
X2Y5H	－	使用禁止
X2Y6H	R/W	割り込みステータス・レジスタ読み出し／書き込み
X2Y7H	－	使用禁止
X2Y8H	R/W	ライン・メモリ・レジスタ読み出し／書き込み
X2Y9H	－	使用禁止
X2YAH	R/W	マスク・レジスタ読み出し／書き込み
X2YBH	－	使用禁止
X2YCH	R/W	TP レジスタ読み出し／書き込み
X2YDH	－	使用禁止

表１－３　μPD72187A のレジスタ（間接マッピング方式）(1/2)

アドレス・レジスタ	I/O アドレス	リード／ライト	機　能
00H	X0Y0H	R/W	コントロール・レジスタ読み出し／書き込み
	X0Y1H	−	使用禁止
	X0Y2H	R	ステータス・レジスタ L 読み出し
	X0Y3H	R	ステータス・レジスタ H 読み出し
	X0Y4H	R/W	モード・レジスタ L 読み出し／書き込み
	X0Y5H	R/W	モード・レジスタ H 読み出し／書き込み
	X0Y6H	W	画素数設定レジスタ L 書き込み
	X0Y7H	W	画素数設定レジスタ H 書き込み
	X0Y8H	W	ライン数設定レジスタ L 書き込み
	X0Y9H	W	ライン数設定レジスタ H 書き込み
	X0YAH	R/W	ビット・プレーン・ライン数設定レジスタ L 読み出し／書き込み
	X0YBH	R/W	ビット・プレーン・ライン数設定レジスタ H 読み出し／書き込み
	X0YCH	W	ストライプ・レジスタ L 書き込み
	X0YDH	W	ストライプ・レジスタ H 書き込み
	X0YEH	R/W	AT レジスタ L 読み出し／書き込み
	X0YFH	W	AT レジスタ H 書き込み
01H	X1Y0H	R/W	AT ライン・レジスタ L 読み出し／書き込み
	X1Y1H	W	AT ライン・レジスタ H 書き込み
	X1Y2H	W	画像データ開始アドレス・レジスタ L 書き込み
	X1Y3H	W	画像データ開始アドレス・レジスタ M 書き込み
	X1Y4H	W	画像データ開始アドレス・レジスタ H 書き込み
	X1Y5H	−	使用禁止
	X1Y6H	R/W	画像データ終了アドレス・レジスタ L 読み出し／書き込み
	X1Y7H	R/W	画像データ終了アドレス・レジスタ M 読み出し／書き込み
	X1Y8H	R/W	画像データ終了アドレス・レジスタ H 読み出し／書き込み
	X1Y9H	−	使用禁止
	X1YAH	R/W	データ・バッファ・レジスタ L 読み出し／書き込み
	X1YBH	R/W	データ・バッファ・レジスタ H 読み出し／書き込み
	X1YCH	R/W	ホスト・バス・モード・レジスタ読み出し／書き込み
	X1YDH	−	使用禁止
	X1YEH	R/W	マーカ・コード・バッファ・レジスタ読み出し／書き込み
	X1YFH	−	使用禁止

表１－３　μPD72187A のレジスタ（間接マッピング方式）(2/2)

アドレス・レジスタ	I/O アドレス	リード／ライト	機　能
02H	X2Y0H	R	マーカ・コード・ウインドウ・レジスタ L 読み出し
	X2Y1H	R	マーカ・コード・ウインドウ・レジスタ H 読み出し
	X2Y2H	R/W	コマンド・レジスタ１読み出し／書き込み
	X2Y3H	−	使用禁止
	X2Y4H	R/W	コマンド・レジスタ２読み出し／書き込み
	X2Y5H	−	使用禁止
	X2Y6H	R/W	割り込みステータス・レジスタ読み出し／書き込み
	X2Y7H	−	使用禁止
	X2Y8H	R/W	ライン・メモリ・レジスタ読み出し／書き込み
	X2Y9H	−	使用禁止
	X2YAH	R/W	マスク・レジスタ読み出し／書き込み
	X2YBH	−	使用禁止
	X2YCH	R/W	TP レジスタ読み出し／書き込み
	X2YDH	−	使用禁止

（２）μPD71055（システム・レジスタ）の I/O マップ

表１－４　システム・レジスタの I/O マップ

I/O アドレス	リード／ライト	機　能
X3Y0H	R	ポート０読み出し（アドレス・レジスタ）
	W	ポート０書き込み（アドレス・レジスタ）
X3Y2H	R	ポート１読み出し（転送方向レジスタ）
	W	ポート１書き込み（転送方向レジスタ）
X3Y4H	R	ポート２読み出し
	W	ポート２書き込み
X3Y6H	R	使用禁止
	W	コマンド・レジスタ書き込み

（３）μPD71071（DMA コントローラ）のレジスタ

表１－５　μPD71071 のレジスタ

I/O アドレス	リード／ライト	機　能
X4Y0H	R	使用禁止
	W	イニシャライズ書き込み
X4Y1H	R	チャネル・レジスタ読み出し
	W	チャネル・レジスタ書き込み
X4Y2H	R	カウント・レジスタＬ読み出し
	W	カウント・レジスタＬ書き込み
X4Y3H	R	カウント・レジスタＨ読み出し
	W	カウント・レジスタＨ書き込み
X4Y4H	R	アドレス・レジスタＬ読み出し
	W	アドレス・レジスタＬ書き込み
X4Y5H	R	アドレス・レジスタＭ読み出し
	W	アドレス・レジスタＭ書き込み
X4Y6H	R	アドレス・レジスタＨ読み出し
	W	アドレス・レジスタＨ書き込み
X4Y7H	R	使用禁止
	W	使用禁止
X4Y8H	R	デバイス・コントロール・レジスタＬ読み出し
	W	デバイス・コントロール・レジスタＬ書き込み
X4Y9H	R	デバイス・コントロール・レジスタＨ読み出し
	W	デバイス・コントロール・レジスタＨ書き込み
X4YAH	R	モード・コントロール・レジスタ読み出し
	W	モード・コントロール・レジスタ書き込み
X4YBH	R	ステータス・レジスタ読み出し
	W	使用禁止
X4YCH	R	テンポラリ・レジスタＬ読み出し
	W	使用禁止
X4YDH	R	テンポラリ・レジスタＨ読み出し
	W	使用禁止
X4YEH	R	リクエスト・レジスタ読み出し
	W	リクエスト・レジスタ書き込み
X4YFH	R	マスク・レジスタ読み出し
	W	マスク・レジスタ書き込み

（４）　μPD71054（タイマ）のレジスタ

表１－６　μPD71054 のレジスタ

I/O アドレス	リード／ライト	機　能
X5Y0H	R	カウンタ#0 読み出し
	W	カウンタ#0 書き込み
X5Y2H	R	カウンタ#1 読み出し
	W	カウンタ#1 書き込み
X5Y4H	R	カウンタ#2 読み出し
	W	カウンタ#2 書き込み
X5Y6H	R	使用禁止
	W	コントロール・ワード・レジスタ書き込み

（５）　リセット・レジスタ（74ACT74）

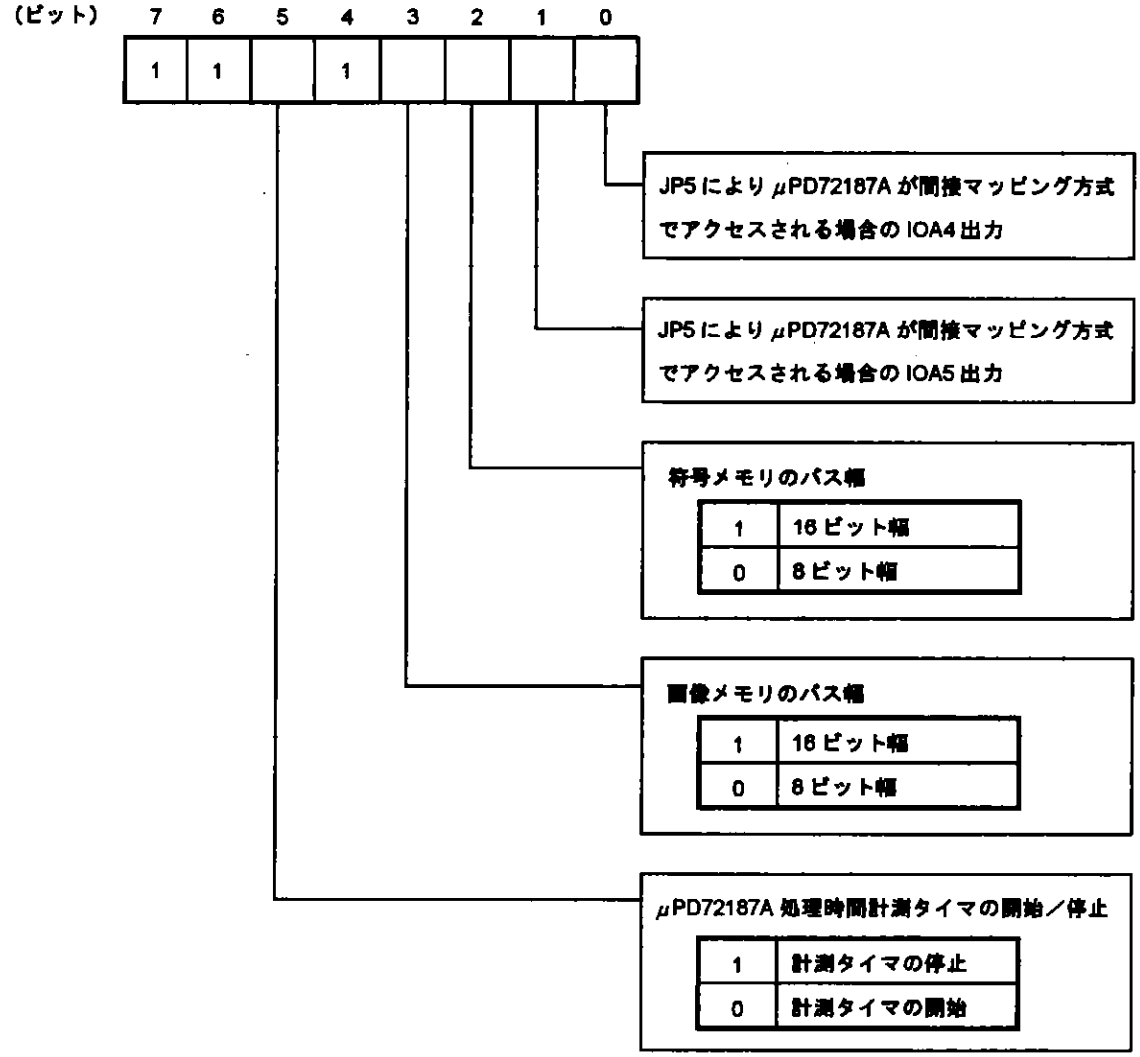
表１－７　リセット・レジスタの I/O マップ

I/O アドレス	リード／ライト	機　能
X6Y0H	R	使用禁止
	W	μPD72187A ボード，リセット ON
X6Y2H	R	使用禁止
	W	μPD72187A ボード，リセット OFF

1.3.2　システム設定用レジスタ

（１）I/O アドレス　X3Y0H　番地

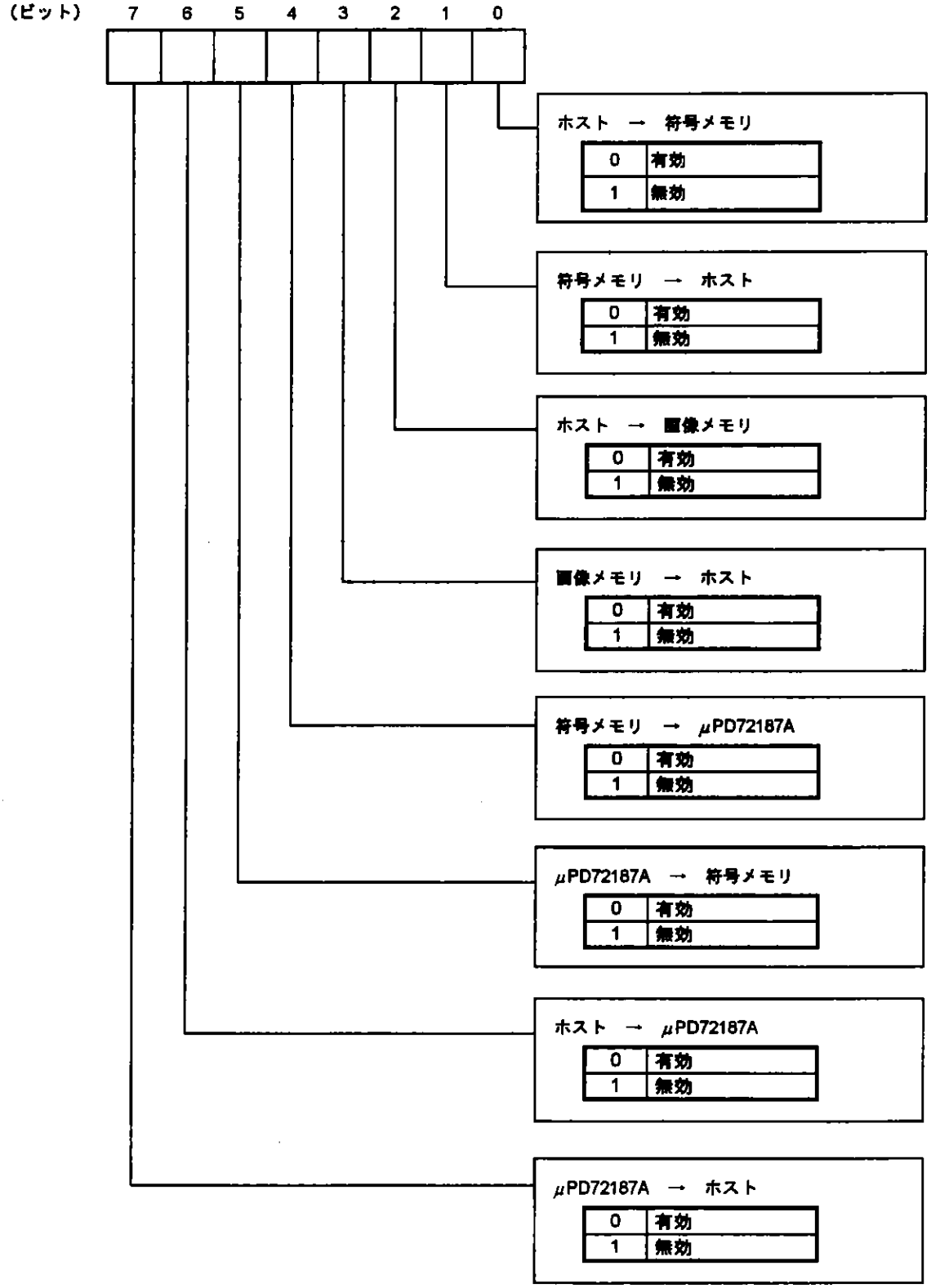
ポート０　：アドレス・レジスタ（出力）



（２）I/O アドレス　X3Y2H　番地

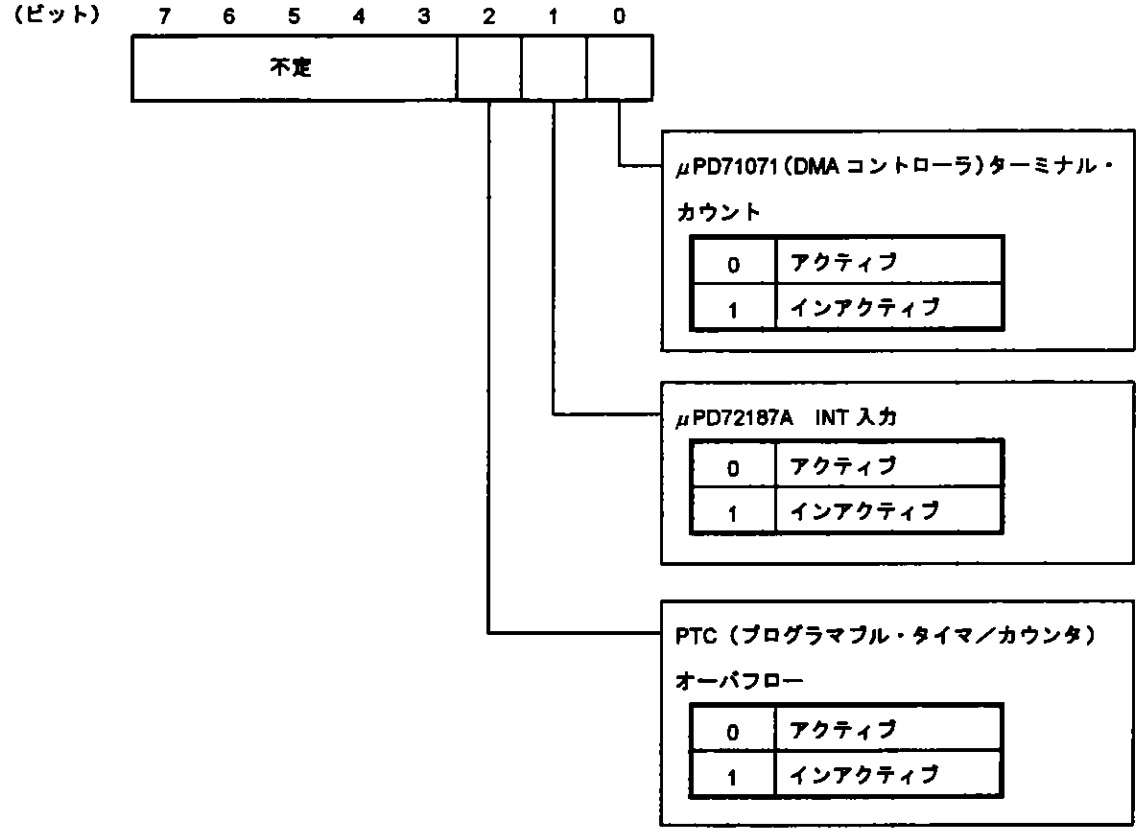
ポート１　：転送方向・レジスタ（出力）

外部 DMA コントローラの転送方向を決定する。



（3）I/O アドレス　X3Y4H　番地

ポート2　：（入力）



1.3.3　ホスト・マシンへの対応

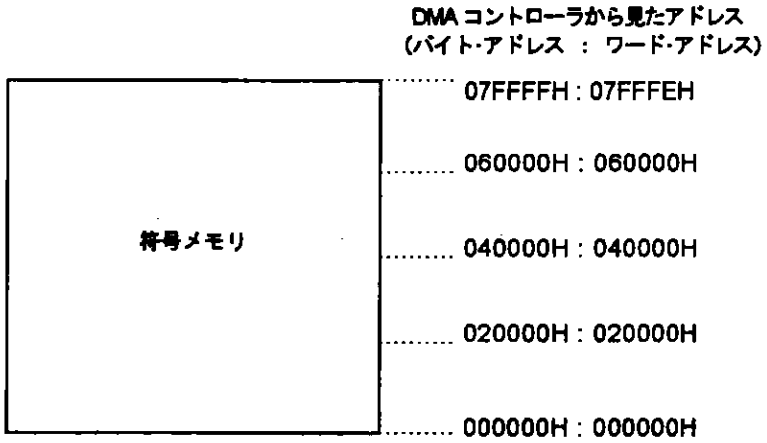
この回路では，PC-9801VX 以降のノーマル・モード，C バス搭載機に対応しています。

1.4　画像／符号メモリ・マップ

1.4.1　画像／符号メモリ・マップ

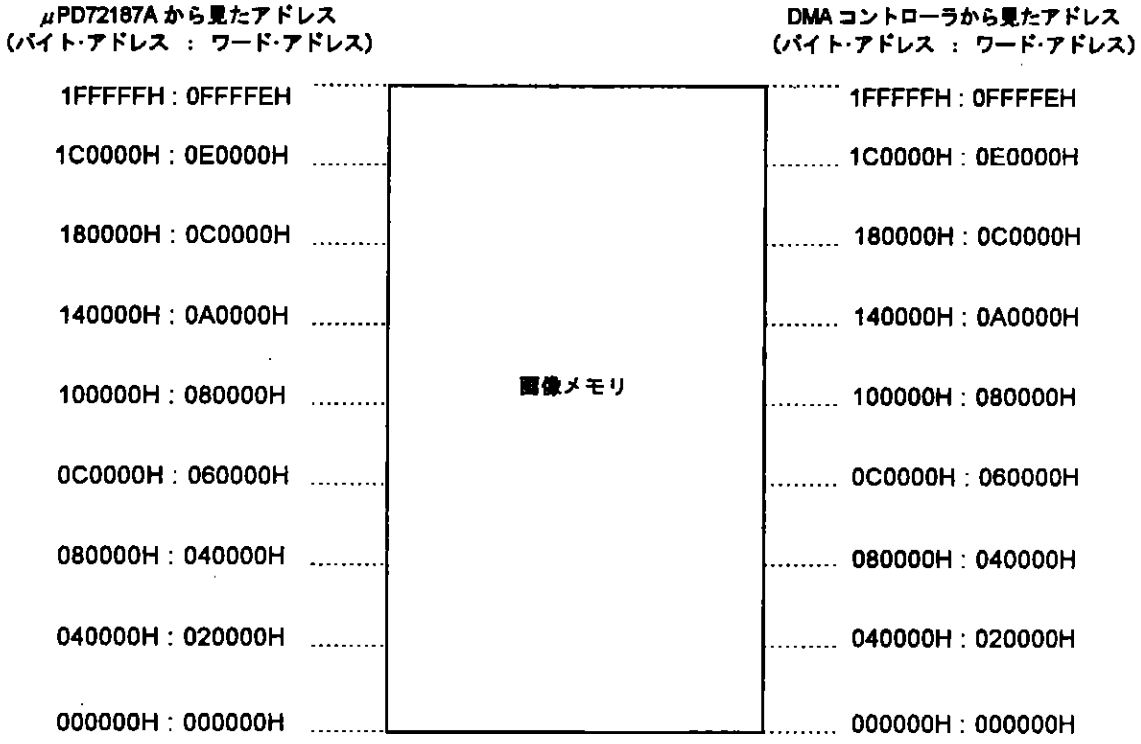
符号メモリはホストCPU（実際にはDMAコントローラ）の，画像メモリはμPD72187Aの制御下にあるメモリで，図1－2，1－3のような構成となっています。

図1－2　符号メモリのメモリ・マップ



備考　バイト･モード時は512 Kバイトの領域に，また，ワード・モード時は1Mバイトの領域にアクセス可能です。

図1－3　画像メモリのメモリ・マップ



1.5　DMAコントローラ

1.5.1　接　　続

この回路上の DMA コントローラ（μPD71071）は，ホスト CPU の DMA コントローラとカスケード接続します。

カスケード接続するホスト側の DMA チャネルは，チャネル#3 です。

1.5.2　DMA コントローラの使用するチャネル

・DMA チャネル#0，#1：ホスト，画像メモリ間の画像データ転送
　　ホスト，符号メモリ間の符号データ転送

・DMA チャネル#2　　：符号メモリ，μPD72187A 間のデータ転送
　　ホスト・メモリ，μPD72187A 間のデータ転送

・DMA チャネル#3　　：使用禁止

1.5.3　DMA 転送方向，使用チャネルと転送方向レジスタの関係

（　）で囲まれた転送は，μPD72187A 内蔵の DMA コントローラを使用した転送です。

表１－８　DMA 転送方向，使用チャネルと転送方向レジスタの関係

DMA 転送方向	DMA コントローラ使用チャネル（μPD72187A）	DMA 転送方向レジスタ
ホスト・メモリ → 符号メモリ	チャネル#0，#1	01H
符号メモリ → ホスト・メモリ	チャネル#0，#1	02H
ホスト・メモリ → 画像メモリ	チャネル#0，#1	04H
画像メモリ → ホスト・メモリ	チャネル#0，#1	08H
符号メモリ → μPD72187A (μPD72187A → 画像メモリ)	チャネル#2	10H
μPD72187A → 符号メモリ (画像メモリ → μPD72187A)	チャネル#2	20H
ホスト・メモリ → μPD72187A (μPD72187A → 画像メモリ)	チャネル#2	40H
μPD72187A → ホスト・メモリ (画像メモリ → μPD72187A)	チャネル#2	80H

1.5.4　DMA転送方向の切り替え方法

システム・レジスタ・ポート1のビット0-7によりアドレス・バス，データ・バスのバス・バッファの転送方向と，アクティブ，インアクティブを制御します。

1.5.5　DMA転送時の8ビット／16ビット転送単位の切り替え方法

（1）符号メモリのビット転送単位の切り替え

・システム・レジスタ・ポート0のビット2を0/1に切り替えることにより，符号メモリのデータ・バス幅を8/16ビットに切り替えます。

・アドレスの割り付けは8ビット単位で固定です（16ビット単位の転送時は偶数アドレスのみでアクセスする）。

（2）画像メモリのビット転送単位の切り替え

・システム・レジスタ・ポート0のビット3を0/1に切り替えることにより，画像メモリのデータ・バス幅を8/16ビットに切り替えます。

・アドレスの割り付けは，DMAコントローラ（μPD71071）で転送を行うとき，8ビット単位で固定です（16ビット単位の転送時は偶数アドレスのみでアクセスする）。

・μPD72187Aで転送を行うとき，8ビット転送時は8ビット1アドレス，16ビット転送時は16ビット1アドレスとなります。

・μPD72187Aで転送を行うとき，同一メモリ・セルに対する8ビット転送時のアドレスと16ビット転送時のアドレスの対応は次のとおりです。

8ビット転送時のアドレス値	16ビット転送時のアドレス値
000000H - 000001H	000000H
000002H - 000003H	000001H
：	：

1.6　スイッチ，ジャンパ

1.6.1　スイッチ

（１）SW1

ボードの供給電源をホスト・バスと外部電源とに切り替えるスライド・スイッチです。

・HOST　　：ホスト・バス
・EXT　　：外部電源

（２）SW2, SW3

このボードがホストに I/O として割り当てられる際の I/O アドレスを設定するスイッチです。

・SW2　　：I/O アドレス下位８ビット中の上位側４ビット
・SW3　　：I/O アドレス上位８ビット中の上位側４ビット

もし，SW2=D　SW3=5 とすると，μPD72187A のコントロール・レジスタは，50D0H 番地に割り当てられることになります。

1.6.2　ジャンパ

（１）JP1，JP3

ホスト内蔵の DMA コントローラの使用チャネルを選択するジャンパです。

・1-2 ショート　：チャネル 0
・3-4 ショート　：チャネル 3

（２）JP2

μPD72187A の INT 出力とホストの IR03-IR13 との接続を選択するジャンパです。

・1-2 ショート　：IR03
・3-4 ショート　：IR05
・5-6 ショート　：IR06
・7-8 ショート　：IR09
・9-10 ショート　：IR10
・11-12 ショート　：IR12
・13-14 ショート　：IR13

（３）JP4

画像メモリとホスト間でのデータ転送において，データ・ビットの並びを８ビット単位で反転／非反転を設定するジャンパです。

・1-2 ショート　：非反転
・3-4 ショート　：反転

（４）JP5

μPD72187A の I/O マッピング方式を設定するジャンパです。

・1-2 ショート　：直接マッピング
・3-4 ショート　：アドレス・レジスタを使用した間接マッピング

（５）JP6

μPD72187A の MACK 信号入力を内部信号にするか，外部信号にするかを設定するジャンパです。

・1-2 ショート　：内部信号
・3-4 ショート　：外部信号

（６）JP7

μPD72187A の READY 信号を内部信号にするか，外部信号にするかを設定するジャンパです。

・1-2 ショート　：内部信号
・3-4 ショート　：外部信号

［メ　　モ］

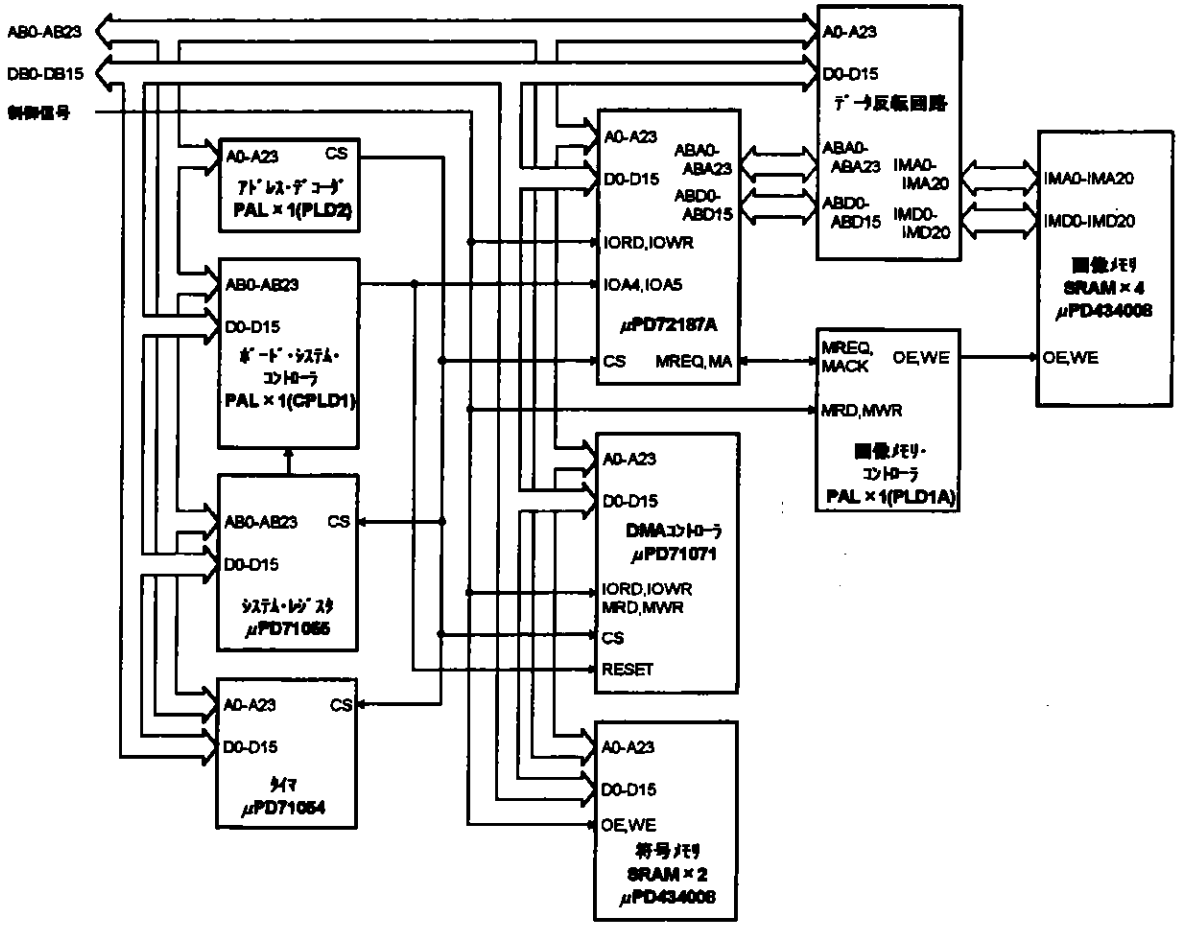
第２章　回路の説明

ここではこのアプリケーション・ノートで設計した回路について説明します。
全体回路図については，付録Ａ　全体回路図を参照してください。

2.1　ブロック図

図２－１に，この回路のブロック図を示します。

図２－１　回路のブロック図



2.2　ホスト・インタフェース

ホストCPUからは，μPD72187A，DMAコントローラ（μPD71071），タイマ（μPD71054），システム・レジスタ（μPD71055）の各レジスタがI/Oとしてマッピングされています。

回路上の符号メモリ，画像メモリおよびμPD72187A間のデータ転送に回路上のDMAコントローラを使用して，DMA転送を行っています。このとき，DMAコントローラが拡張スロット・バスを制御することになります。したがって，ホスト側の動作を停止させて拡張スロット・バスの制御権を回路上のDMAコントローラに渡さなければなりません。このDMAコントローラはホスト内部のDMAコントローラとカスケード接続されており，内部DMAコントローラは外部DMAコントローラとCPU間のHRQ/HLDAを仲介する動作しか行いません。

拡張スロット・バスの制御権をDMAコントローラが持っている間は拡張スロット・バスのMWE0信号を供給する必要があります。このため，DMAコントローラのMWR信号からMWE信号を生成し，DMAコントローラが制御権を持っている間供給します。

★ 第３章　制御シーケンス・プログラム

３.１　機能概要

制御シーケンス・プログラムは，JBIG 方式対応 ２値画像データ圧縮／伸長用プロセッサ μPD72187A を使用するためのプログラムです。

制御シーケンス・プログラムは，以下のような機能を備えています。

・μPD72187A による２値画像データの圧縮／伸長処理シーケンス制御
・設計した回路上の DMA コントローラを介した画像／符号メモリの書き換え

３.２　動作環境

PC-9800 シリーズの拡張スロットに，設計した回路を実装します。
OS は，MS-DOS Ver.3.10 以上とします。

回路は，ジャンパにより各種設定を行うことができますが，JP5 の設定は次に示すように行ってください。

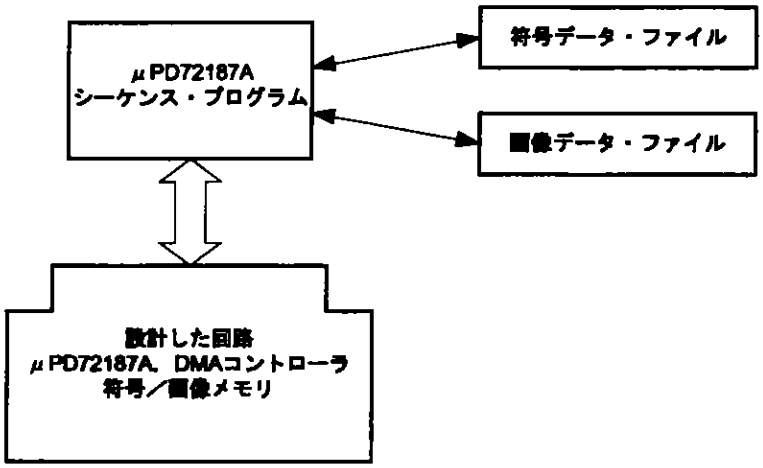
・JP5（I/O マッピング方式の設定）
　→　1-2 ショート：直接マッピング方式

また，ポート・アドレスを変更する場合は，次の２カ所の設定を変更してください。

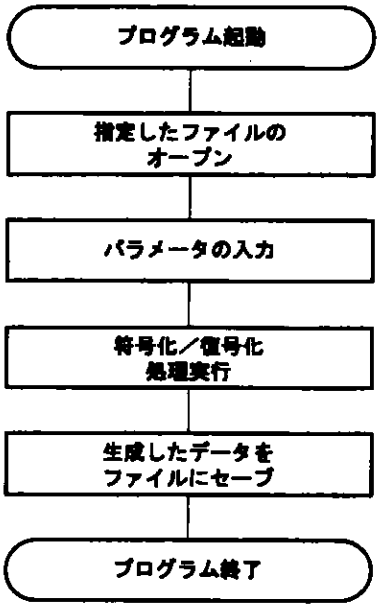
○回路の設定
　・SW2,SW3（I/O マッピング・アドレスの設定）

○制御シーケンス・プログラムの修正
　・ファイル APP187A.H 中の #define IOADDR の値

３.３　制御シーケンス・プログラムの基本構成



３.４　制御シーケンス・プログラムの流れ



３.５　プログラムの起動方法

3.5.1　実行形式

プログラムに，符号化／復号化処理を行う画像または符号データ・ファイル名と，処理後の結果をセーブするためのファイル名を与え実行します。

DEMO187A　［処理するファイル］　［処理後のファイル］

3.5.2　処理するファイル

符号化処理を行う場合には画像データ・ファイルを，復号化処理を行う場合には符号データ・ファイルを指定してください。

3.5.3　処理後のファイル

符号化処理の場合には処理後に生成される符号データのためのファイル名を，復号化処理の場合には処理後に生成される画像データのためのファイル名を，それぞれ指定してください。

シーケンス・プログラムは実行後に，これらのファイル名を使ってバイナリのファイルをオープンしようとします。もしこのとき，同じ名前のファイルがある場合は，メッセージを返さずにそのまま上書きします。

3.5.4　プログラムの終了

起動後に指定した処理が終了すると，プログラムは終了します。

３.６　プログラムで扱うデータ

このプログラムで扱うデータは，すべてバイナリ・データです。

符号化／復号化処理を行うデータには，ヘッダ情報などを付加しないでください。

このプログラムはファイル先頭からデータを処理するため，ヘッダ情報などが付加されている場合は，そのヘッダ情報も画像または符号データとして処理します。

３.７　プログラムの操作方法

3.7.1　符号化を行う場合

（１）　プログラムの起動

符号化を行う場合には，次のパラメータを設定してプログラムを起動します。

なお，パラメータに設定されているファイル名は例です。実際に処理する場合は，適切なファイル名を指定してください。

DEMO187A	TEST.IMG	TEST.COD
↑	↑	↑
プログラム	画像データのファイル名	符号データのファイル名

パラメータが不足している場合，画像データ・ファイルが見つからない場合および符号データ・ファイルを開くことができない場合には，エラーを返して終了します。

このような場合は，パラメータを確認してから，再起動してください。

○パラメータが足りない場合

Syntax error..

書式　：　demo187a　［元のファイル名］［新しいファイル名］

○画像データ・ファイルが見つからない場合

［指定したファイル名］がありません。

○符号データ・ファイルを開くことができない場合

［指定したファイル名］を開くことができません。

（２）　処理形態の入力

指定したパラメータに間違いがなく，ファイルのオープンが成功すると，画像データのファイル・サイズが表示され，処理の入力を促すメッセージが表示され，番号入力待ちとなります。

ファイル・サイズは　513,216 byte　です。

処理を入力してください。　（1. 符号化　or　2. 復号化）

符号化処理を行いますので，ここでは，“1”を入力します。

次に処理形態の入力を促すメッセージが表示され，番号入力待ちとなりますので希望する処理番号を入力します。

処理形態を入力してください
（1. 通常処理　2. TP 処理あり　3. AT 処理あり　4. TP・AT 処理あり）

（３）　画像サイズの入力

処理形態の入力が終了すると，次に処理を行う画像サイズの入力を行います。

・主走査画素数　　（横の画素数）
・副走査ライン数　　（縦のライン数）
・ストライプ・ライン数　（１ストライプ当たりのライン数）

この順番でそれぞれのパラメータを聞いてきますので，それぞれの値を入力してください。

ここで，「3. AT 処理あり」「4. TP・AT 処理あり」を選択した場合，さらに AT 画素位置を聞いてきますので，適切な値を入力してください。

ただし，このプログラムでは使用するテンプレートが３ライン・テンプレート固定になっていますので，AT 画素位置の指定できる範囲は，0, 3～127 です。（“0”を指定した場合は，AT 画素は元の位置から移動しません。）

（４）　処理開始

処理，処理形態，画像サイズなどのパラメータをすべて入力し終わると，指定した処理が開始されます。

処理中は，以下のようなメッセージが表示されます。

・現在行っている処理（ソフトウエア・リセット，パラメータ設定など）
・INT 発生
・マーカ・コード（MK）付加
・符号化終了
・符号データ・セーブ完了

（５）　処理終了

指定したすべての処理が完了すると，生成した符号データをファイルにセーブしてプログラムを終了します。

3.7.2　復号化を行う場合

（1）　プログラムの起動

復号化を行う場合には，次のパラメータを設定してプログラムを起動します。

なお，パラメータに設定されているファイル名は例です。実際に処理する場合は，適切なファイル名を指定してください。

DEMO187A	TEST.COD	TEST.IMG
↑	↑	↑
プログラム	符号データのファイル名	画像データのファイル名

パラメータが不足している場合，符号データ・ファイルが見つからない場合，または画像データ・ファイルを開くことができない場合には，エラーを返して終了します。

このような場合は，パラメータを確認してから，再起動してください。

○パラメータが足りない場合

Syntax error..

書式　：　demo187a　［元のファイル名］［新しいファイル名］

○符号データ・ファイルが見つからない場合

［指定したファイル名］がありません。

○画像データ・ファイルを開くことができない場合

［指定したファイル名］を開くことができません。

（2）　処理形態の入力

指定したパラメータに間違いがなく，ファイル・オープンが成功すると，符号データのファイル・サイズが表示され，処理の入力を促すメッセージが表示され，番号入力待ちとなります。

ファイル・サイズは　513,216 byte です。
処理を入力してください。　（1. 符号化　or　2. 復号化）

復号化処理を行いますので，ここでは，“2”を入力します。

次に処理形態の入力を促すメッセージが表示され，番号入力待ちとなりますので，希望する処理番号を入力します。

処理形態を入力してください
（1. 通常処理　2. TP 処理あり　3. AT 処理あり　4. TP・AT 処理あり）

注意　復号化処理を行なうときは，符号化したときと同じ処理形態を入力してください。
符号化時と異なる処理形態を指定した場合は，正しく復号化処理を行うことができません。

（３）　画像サイズの入力

処理形態の入力が終了すると，次に処理を行う画像サイズの入力を行います。

- 主走査画素数　　（横の画素数）
- 副走査ライン数　　（縦のライン数）
- ストライプ・ライン数　（１ストライプ当たりのライン数）

この順番でそれぞれのパラメータを聞いてきますので，それぞれの値を入力してください。

（２）で，「3. AT 処理あり」「4. TP・AT 処理あり」を選択した場合，さらに AT 画素位置を聞いてきますので，適切な値を入力してください。

ただし，このプログラムでは使用するテンプレートが３ライン・テンプレート固定になっていますので，AT 画素位置の指定できる範囲は，0, 3～127 です。（“0”を指定した場合は，AT 画素は元の位置から移動しません。）

注意　復号化処理を行なうときは，符号化したときと同じパラメータを入力してください。
符号化時と異なるパラメータを指定した場合は，正しく復号化処理を行うことができません。

（４）　処理開始

処理，処理形態，画像サイズなどのパラメータをすべて入力し終わると，指定した処理が開始されます。

処理中は，以下のようなメッセージが表示されます。

- 現在行っている処理（ソフトウエア・リセット，パラメータ設定など）
- INT 発生
- マーカ・コード（MK）発見
- 復号化終了
- 画像データ・セーブ完了

（５）　処理終了

指定したすべての処理が完了すると，生成した画像データをファイルにセーブしてプログラムを終了します。

3.8　ファイルについて

3.8.1　ファイルの説明

各ファイルの説明を示します。

○EXT187A.H

EXTERN 宣言が記されているヘッダ・ファイルです。関数とグローバル変数が宣言されています。

○APP187A.H

Include する標準ヘッダ・ファイルと Define 文が記されているファイルです。評価ボードの I/O マッピング・アドレスをここで定義します。

○COM187A.C

グローバル変数を定義しているファイルです。各デバイスのレジスタ・アドレスをグローバル変数として定義します。

○DMAC187A.C

設計した回路上の DMA コントローラを制御するための関数を定義しているファイルです。

○MAIN187A.C

メイン・プログラム。187A 制御用シーケンス・プログラムと，データ入出力についての関数を定義しているファイルです。

３．９　関数の種類

各ファイルで定義されている関数と，その機能の概要を示します。

3.9.1　ファイル“MAIN187A.C”で定義されている関数

関数名	機　能
enc_norm	符号化シーケンス（通常処理）
enc_tp	符号化シーケンス（TP 処理あり）
enc_at	符号化シーケンス（AT 処理あり）
enc_tpat	符号化シーケンス（TP 処理，AT 処理あり）
dec_norm	復号化シーケンス（通常処理）
dec_tp	復号化シーケンス（TP 処理あり）
dec_at	復号化シーケンス（AT 処理あり）
dec_tpat	復号化シーケンス（TP 処理，AT 処理あり）

関数名	機　能
sysinit	回路の初期化
load_dat	回路上のメモリへ符号／画像データをファイルからロードします。
save_dat	回路上のメモリから符号／画像データをファイルへセーブします。

3.9.2　ファイル“DMAC187A.C”で定義されている関数

関数名	機　能
dma_htoc	外部 DMA コントローラを使用して，ホスト側（ファイル）と回路上の符号メモリとの間でデータを転送します。
dma_htol	外部 DMA コントローラを使用して，ホスト側（ファイル）と回路上の画像メモリとの間でデータを転送します。
dma_ltoc	外部 DMA コントローラを使用して，μPD72187A と回路上の符号メモリとの間でデータを転送します。
dma_tc	外部 DMA コントローラのターミナル・カウントを検出します。
dma_set	外部 DMA コントローラの設定を行います。

３.10　各関数の説明

制御シーケンス・プログラムで使用する関数を，次の形式で説明します。

［関数名］

制御シーケンス・プログラムに使用する関数名です。

［機能］

各関数の機能です。

［パラメータ］

各関数で使用されるパラメータです。

［返値］

各関数の実行を終了して，処理が呼び出し元に戻ってきたときに戻される値です。

3.10.1　enc_norm(forg, fnew, filesize)

［関数名］

void enc_norm(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

符号化シーケンス（通常処理：TP 処理も AT 処理も行わない）を実行します。

［パラメータ］

FILE *forg	：画像データ・ファイルのファイル・ポインタ
FILE *fnew	：符号データ・ファイルのファイル・ポインタ
unsigned long filesize	：画像ファイルのファイル・サイズ

［返値］

なし

3.10.2　enc_tp(forg, fnew, filesize)

［関数名］

void enc_tp(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

符号化シーケンス（TP 処理あり：通常処理に TP 処理を加える）を実行します。

［パラメータ］

FILE *forg	：画像データ・ファイルのファイル・ポインタ
FILE *fnew	：符号データ・ファイルのファイル・ポインタ
unsigned long filesize	：画像ファイルのファイル・サイズ

［返値］

なし

3.10.3　enc_at(forg, fnew, filesize)

［関数名］

void enc_at(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

符号化シーケンス（AT 処理あり：通常処理に AT 処理を加える）を実行します。

［パラメータ］

FILE *forg	：画像データ・ファイルのファイル・ポインタ
FILE *fnew	：符号データ・ファイルのファイル・ポインタ
unsigned long filesize	：画像ファイルのファイル・サイズ

［返値］

なし

3.10.4　enc_tpat(forg, fnew, filesize)

［関数名］

void enc_tpat(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

符号化シーケンス（TP 処理，AT 処理あり：通常処理に TP 処理と AT 処理を加える）を実行します。

［パラメータ］

FILE *forg	：画像データ・ファイルのファイル・ポインタ
FILE *fnew	：符号データ・ファイルのファイル・ポインタ
unsigned long filesize	：画像ファイルのファイル・サイズ

［返値］

なし

3.10.5　dec_norm(forg, fnew, filesize)

［関数名］

void dec_norm(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

復号化シーケンス（通常処理：TP 処理も AT 処理も行わない）を実行します。

［パラメータ］

FILE *forg	：符号データ・ファイルのファイル・ポインタ
FILE *fnew	：画像データ・ファイルのファイル・ポインタ
unsigned long filesize	：符号ファイルのファイル・サイズ

［返値］

なし

3.10.6　dec_tp(forg, fnew, filesize)

［関数名］

void dec_tp(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

復号化シーケンス（TP 処理あり：通常処理に TP 処理を加える）を実行します。

［パラメータ］

FILE *forg	：符号データ・ファイルのファイル・ポインタ
FILE *fnew	：画像データ・ファイルのファイル・ポインタ
unsigned long filesize	：符号ファイルのファイル・サイズ

［返値］

なし

3.10.7　dec_at(forg, fnew, filesize)

［関数名］

void dec_at(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

復号化シーケンス（AT 処理あり：通常処理に AT 処理を加える）を実行します。

［パラメータ］

FILE *forg	：符号データ・ファイルのファイル・ポインタ
FILE *fnew	：画像データ・ファイルのファイル・ポインタ
unsigned long filesize	：符号ファイルのファイル・サイズ

［返値］

なし

3.10.8　dec_tpat(forg, fnew, filesize)

［関数名］

void dec_tpat(FILE *forg, FILE *fnew, unsigned long filesize)

［機能］

復号化シーケンス（TP 処理，AT 処理あり：通常処理に TP 処理と AT 処理を加える）を実行します。

［パラメータ］

FILE *forg	：符号データ・ファイルのファイル・ポインタ
FILE *fnew	：画像データ・ファイルのファイル・ポインタ
unsigned long filesize	：符号ファイルのファイル・サイズ

［返値］

なし

3.10.9　sysinit( )

［関数名］

void sysinit( )

［機能］

回路の初期化を実行します。

・回路のリセット　　・符号／画像バス幅

・転送モード設定　　・DMA コントローラの初期化

・タイマ・カウンタの設定（バイナリ・モード，モード4，インクリメント R/W）

［パラメータ］

なし

［返値］

なし

3.10.10　load_dat(forg, filesize, maddr, mem)

［関数名］

void load_dat(FILE *forg, unsigned long filesize, unsigned long maddr, char mem)

［機能］

回路上にある符号メモリ／画像メモリへ，ファイル上のデータを転送します。

［パラメータ］

FILE *forg	：転送するデータ・ファイルのファイル・ポインタ
unsigned long filesize	：転送するデータ・ファイルのファイル・サイズ
unsigned long maddr	：転送先（符号／画像メモリ）のメモリ・アドレス
char mem	：転送先メモリの選択（CM：符号メモリ，IM：画像メモリ）

［返値］

なし

3.10.11　save_dat(fnew, msaddr, meaddr, mem, stripe)

［関数名］

void save_dat(FILE *fnew, unsigned long msaddr, unsigned long meaddr, char mem, int stripe)

［機能］

回路にある符号メモリ／画像メモリから，ファイルにデータを転送します。

［パラメータ］

FILE *fnew	：転送先のデータ・ファイルのファイル・ポインタ
unsigned long msaddr	：転送するデータ・ファイルのファイル・サイズ
unsigned long meaddr	：転送元（符号／画像メモリ）のメモリ・アドレス
char mem	：転送元メモリの選択（CM：符号メモリ，IM：画像メモリ）
int stripe	：符号データのストライプ数（符号化時のみ使用）

［返値］

なし

3.10.12　dma_htoc(mod, ptrh, adrc, dtl)

［関数名］

void dma_htoc(unsigned char mod, unsigned char far *ptrh,unsigned long adrc, unsigned long dtl)

［機能］

ホスト側のデータ・ファイルを，回路上の符号メモリへ転送，または回路上の符号メモリにあるデータをホスト側のデータ・ファイルに転送します。

［パラメータ］

unsigned char mod	：DMAの転送方向の設定
	（“0”：ホスト→符号メモリ，“1”：符号メモリ→ホスト）
unsigned char far *ptrh	：ホスト側DMA開始アドレス・ポインタ
unsigned long adrc	：符号メモリ側DMA開始アドレス
unsigned long dtl	：データ転送サイズ（バイト）

［返値］

なし

3.10.13　dma_htoi(mod, ptrh, adri, dtl)

［関数名］

void dma_htoi(unsigned char mod, unsigned char far *ptrh,unsigned long adri, unsigned long dtl)

［機能］

ホスト側のデータ・ファイルを，回路上の画像メモリへ転送，または回路上の画像メモリにあるデータをホスト側のデータ・ファイルに転送します。

［パラメータ］

unsigned char mod	：DMAの転送方向の設定
	（“0”：ホスト→画像メモリ，“1”：画像メモリ→ホスト）
unsigned char far *ptrh	：ホスト側DMA開始アドレス・ポインタ
unsigned long adri	：画像メモリ側DMA開始アドレス
unsigned long dtl	：データ転送サイズ（バイト）

［返値］

なし

3.10.14　dma_jtoc(mod, adrc, dtl)

［関数名］

void dma_jtoc(unsigned char mod, unsigned long adrc, unsigned long dtl)

［機能］

符号化処理時には，μPD72187Aの生成する符号データを回路上のDMAコントローラを使用して符号メモリへ書き込みます。復号化処理時には，μPD72187Aに符号データを転送します。

［パラメータ］

unsigned char mod	：DMAの転送方向の設定
	（“0”：μPD72187A→符号メモリ，“1”：符号メモリ→μPD72187A）
unsigned long adrc	：符号メモリ側DMA開始アドレス
unsigned long dtl	：データ転送サイズ（バイト）

［返値］

なし

3.10.15　dma_tc( )

[関数名]

void dma_tc( )

[機能]

回路上の DMA コントローラのターミナル・カウントを検出します。

[パラメータ]

なし

[返値]

なし

3.10.16　dma_set(channel, count, addr, mod)

[関数名]

void dma_set(unsigned char channel, unsigned long count, unsigned long addr, undsigned char mod)

[機能]

回路上の DMA コントローラの設定を行います。

[パラメータ]

unsigned char channel　：使用する DMA チャネル（“0” or “1” or “2”）
unsigned long count　：DMA 転送カウント
unsigned long addr　：DMA 転送開始アドレス
undsigned char mod　：DMA モード

[返値]

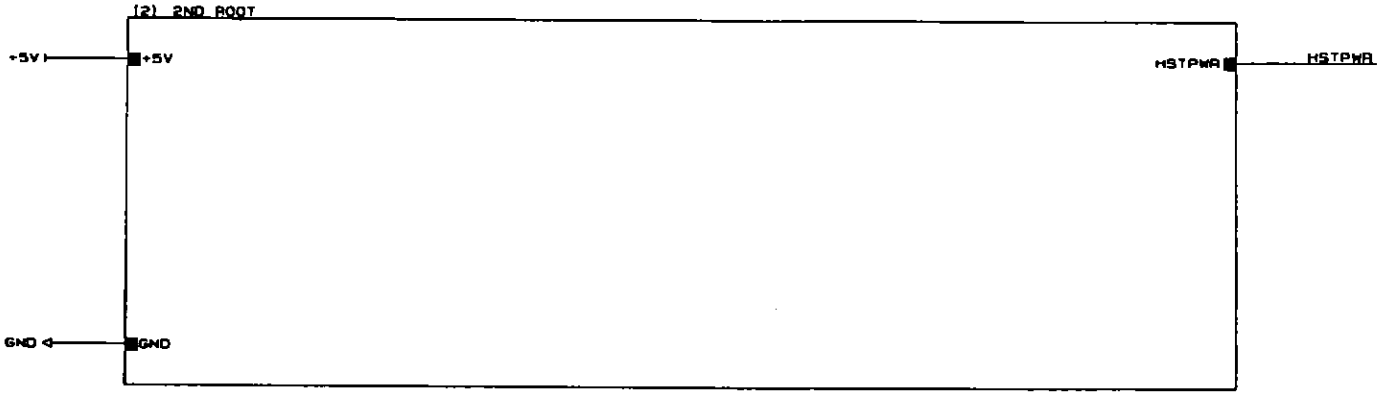
なし

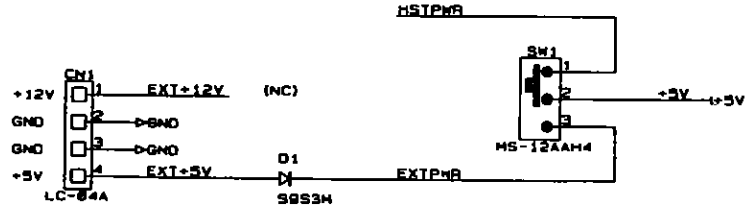
付録A　全体回路図

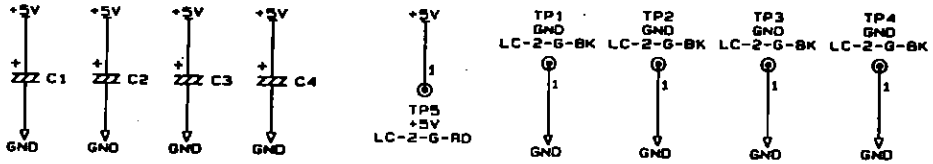
次に，全体回路図を示します。

［メ　　モ］

■A－1　ルート

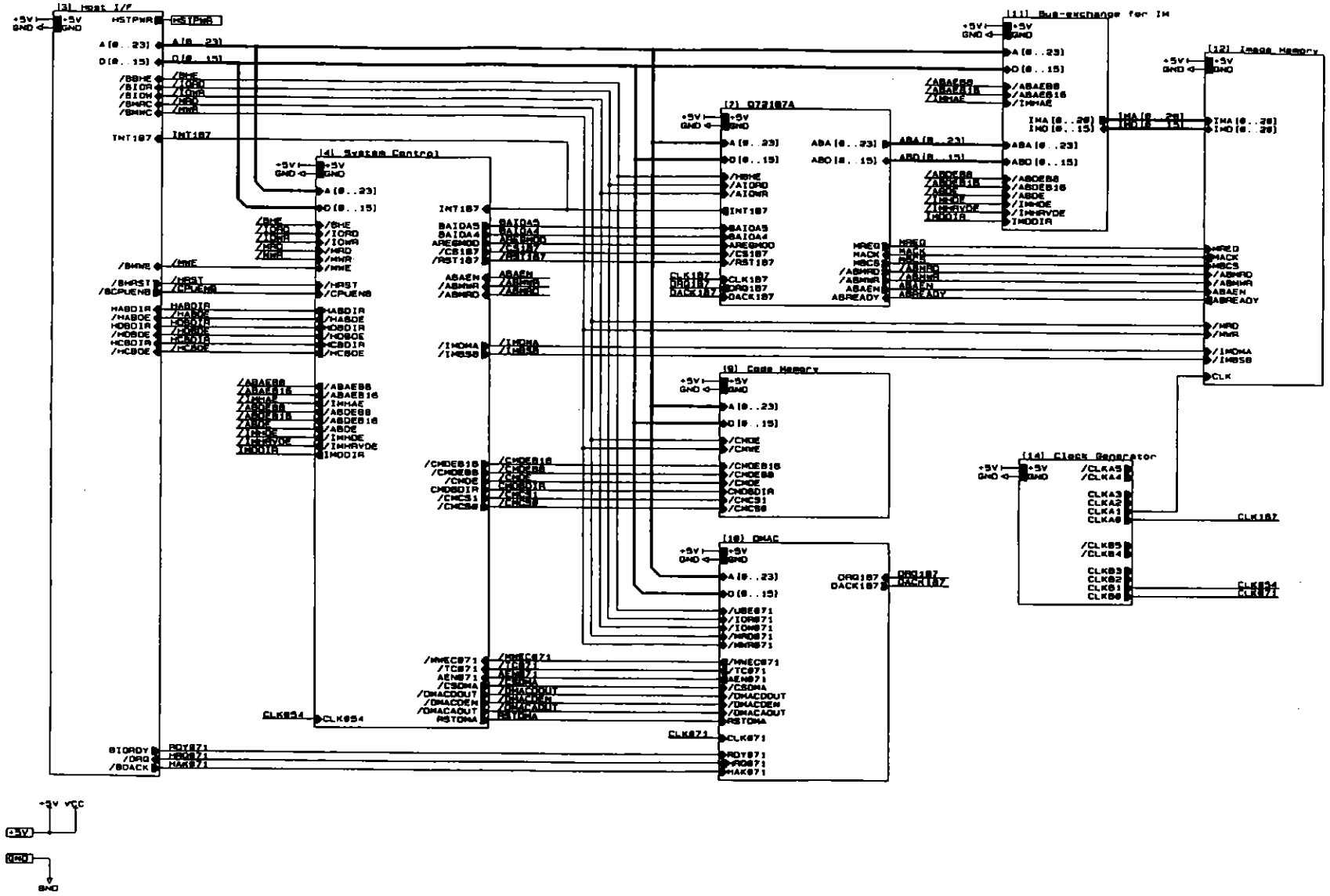

(2) 2ND ROOT
+5V
+5V
GND
GND
HSTPWR
HSTPWR


HSTPWR
CN1
+12V
GND
GND
+5V
EXT+12V
(NC)
EXT+5V
LC-04A
D1
SBS3H
EXTPWR
SW1
+5V
+5V
MS-12AAH4


+5V
C1
GND
+5V
C2
GND
+5V
C3
GND
+5V
C4
GND
+5V
TP5
+5V
LC-2-G-RD
TP1
GND
LC-2-G-BK
GND
TP2
GND
LC-2-G-BK
GND
TP3
GND
LC-2-G-BK
GND
TP4
GND
LC-2-G-BK
GND

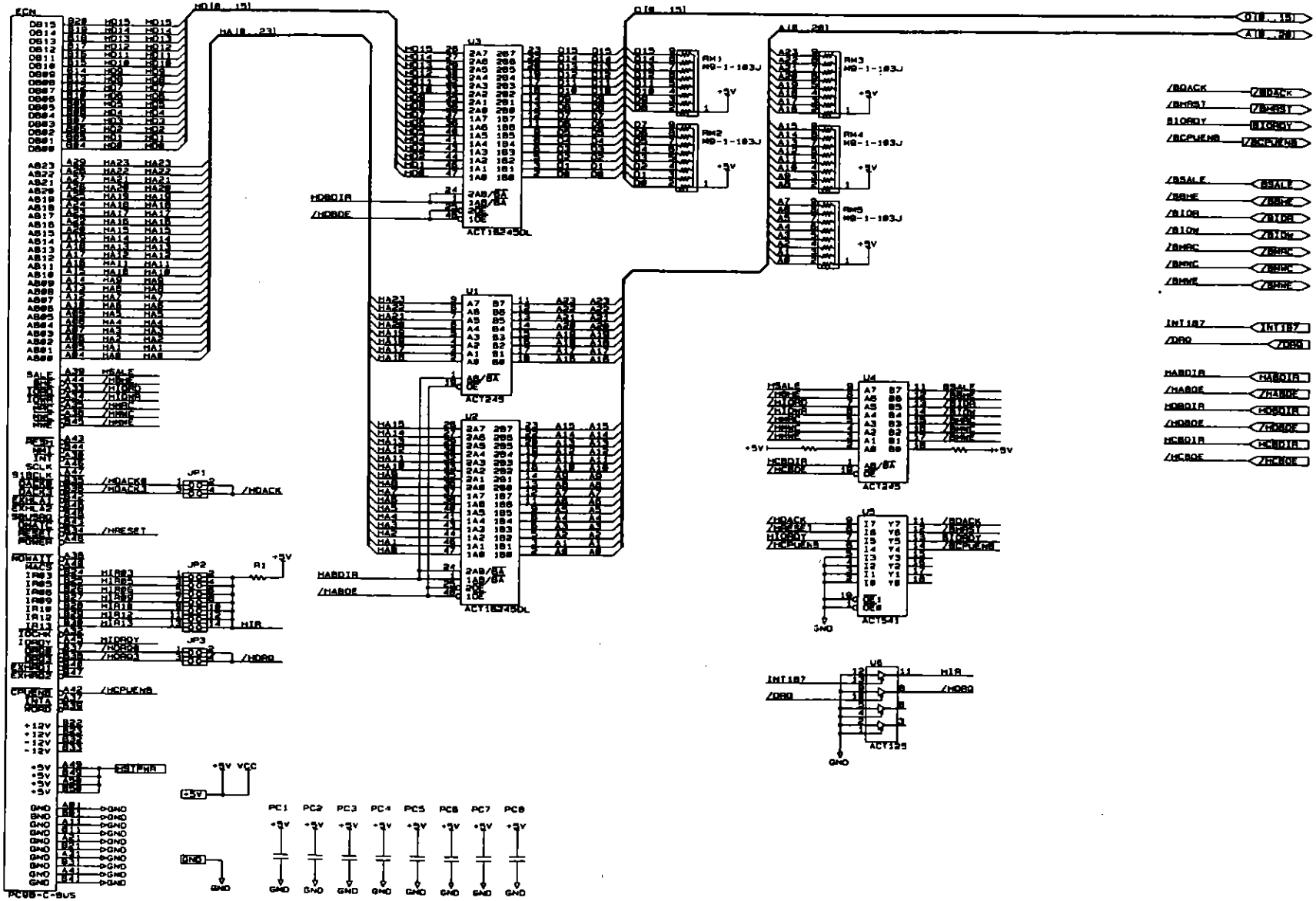
空白ページ

図A－2　セカンド・ルート


[3] Host I/F
[4] System Control
[7] Q721B7A
[8] Code Memory
[9] DMAC
[11] Bus-exchange for IM
[12] Image Memory
[14] Clock Generator
+5V
GND
INT187
CLK187
CLK871
CLK854
+5V VCC
GND

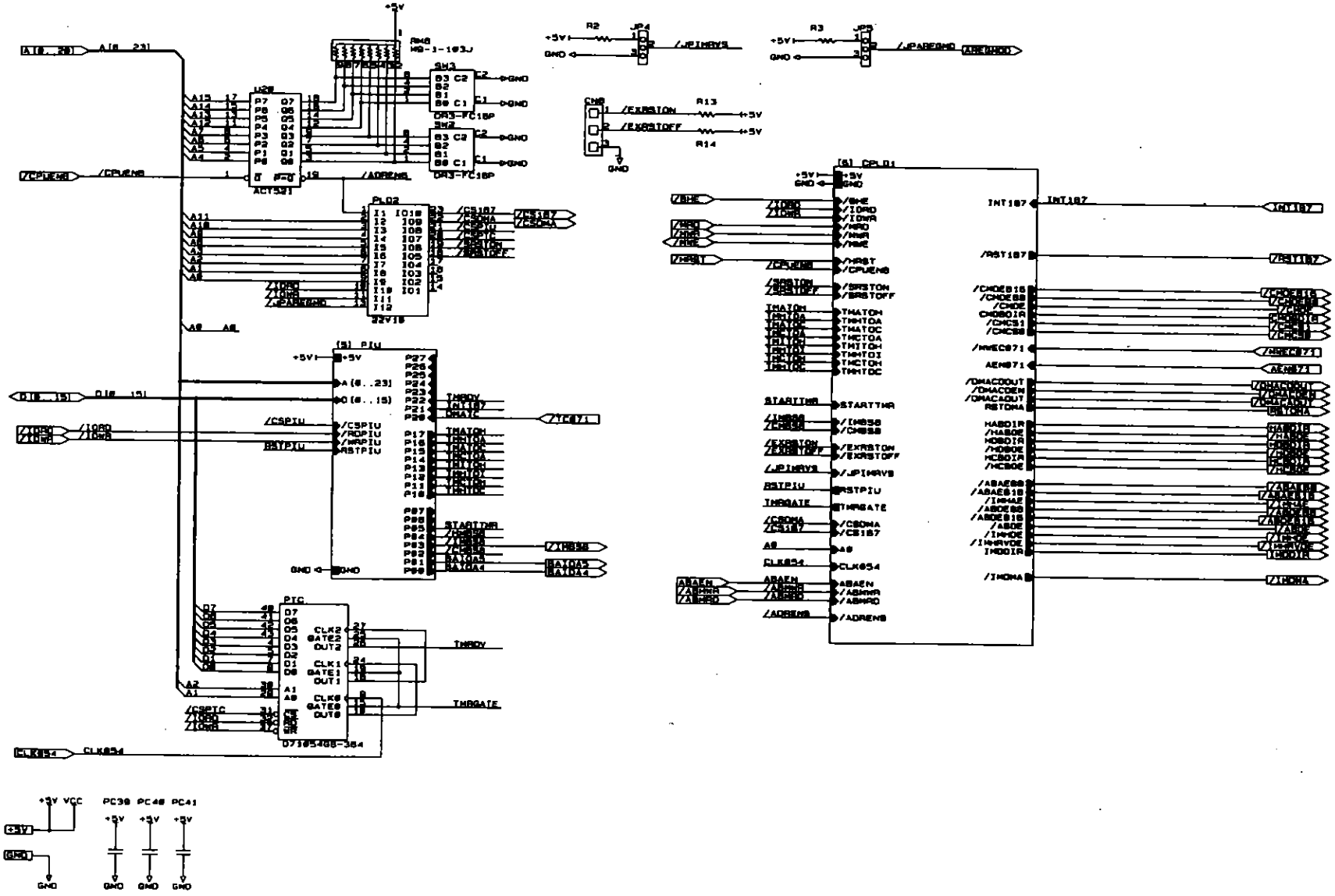
空白ページ

図A－3　ホスト・インタフェース



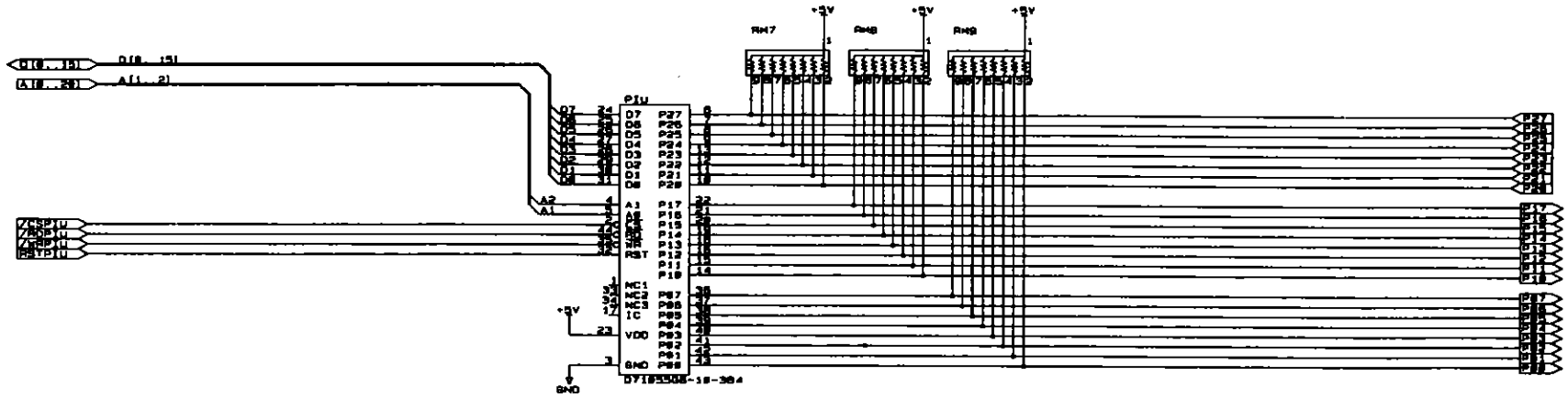
空白ページ

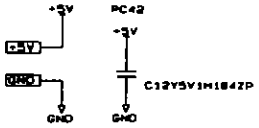
図A－4　システム・コントロール



空白ページ

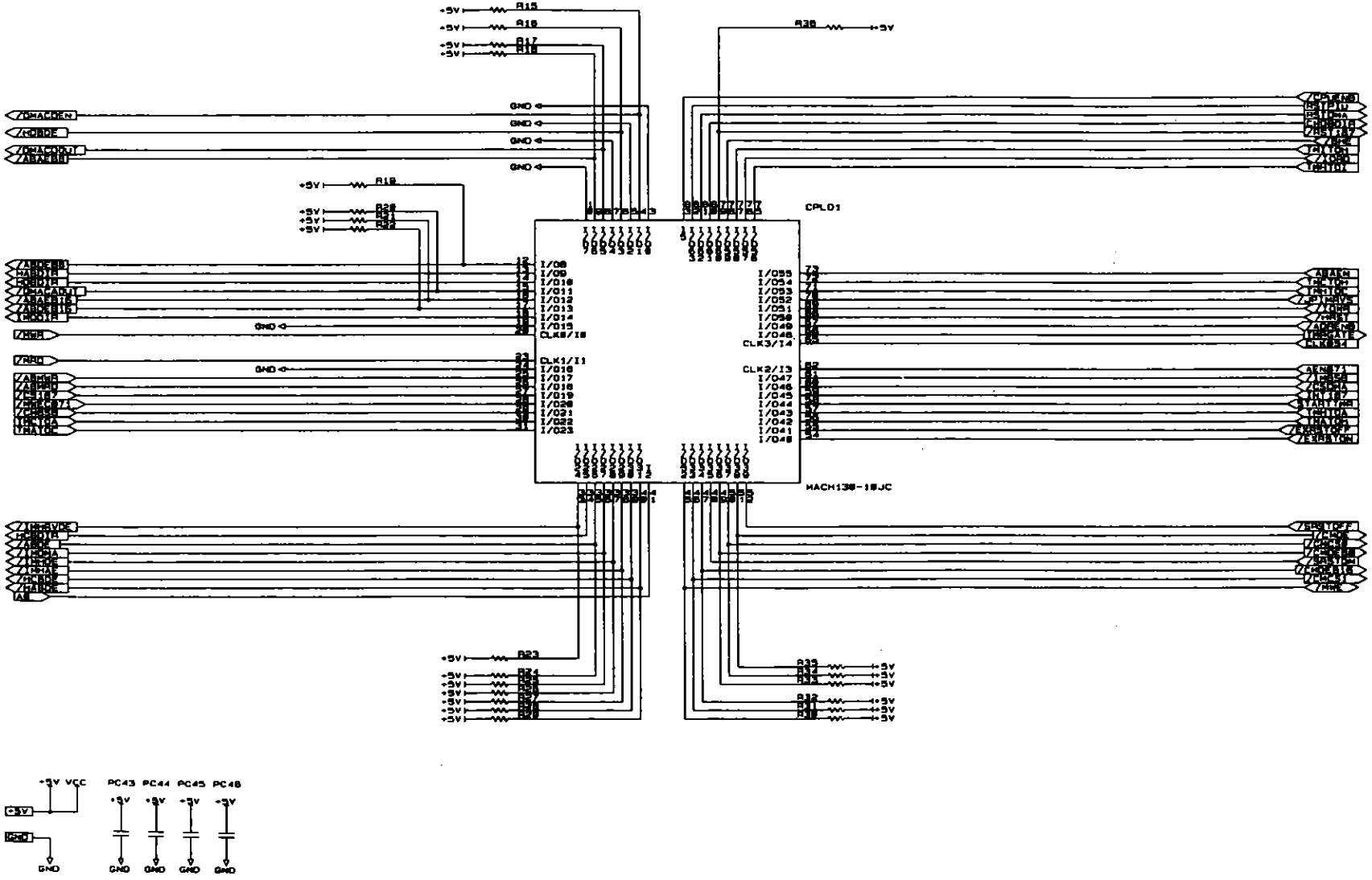
図A－5　PIU


PIU
RM7
RM8
RM9
+5V
GND
A2
A1
RST
VDD
NC1
NC2
NC3
IC


+5V
PC42
+5V
C12Y5V1H104ZP
GND
GND

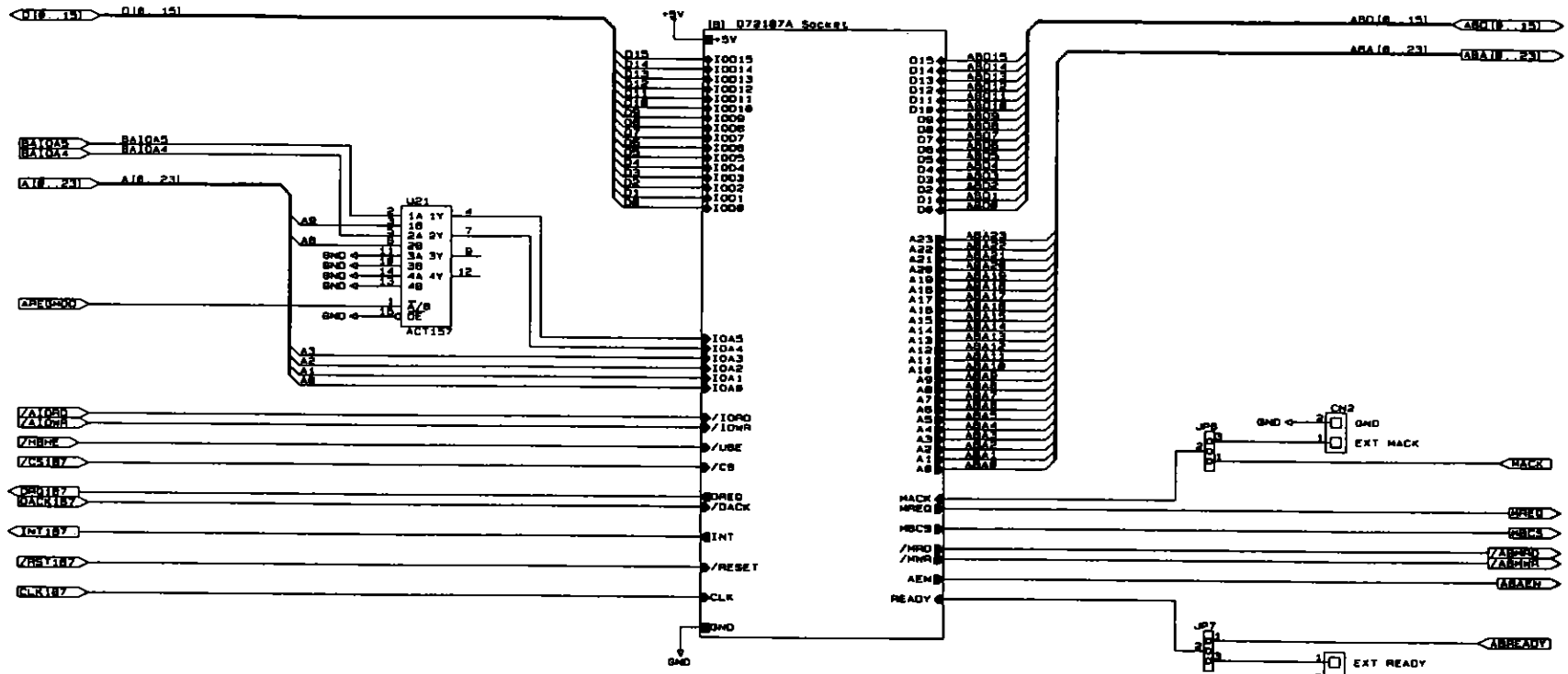
空白ページ

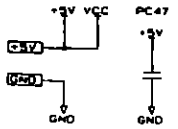
図A－6　CPLD1


CPLD1
MACH130-10JC
+5V
GND
I/O8
I/O9
I/O10
I/O11
I/O12
I/O13
I/O14
I/O15
CLK0/I0
CLK1/I1
I/O16
I/O17
I/O18
I/O19
I/O20
I/O21
I/O22
I/O23
CLK2/I2
CLK3/I3
I/O47
I/O46
I/O45
I/O44
I/O43
I/O42
I/O41
I/O40
I/O55
I/O54
I/O53
I/O52
I/O51
I/O50
I/O49
I/O48
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
R37
R38
/DMACDEN
/HOBDE
/DMACOUT
/ABAEBH
/ABOEBH
/DMACADUT
/ABAEBTH
/ABOEBTH
/HOOTH
/RWR
/RRD
/ABHWR
/ABHRD
/HWTH
/HWRCNTH
/CHRDH
/HRTOTH
/HATOH
/HBSYDE
/HDDTH
/HBDE
/HDDA
/HDDE
/HDAT
/HCBDE
/HABDE
/HAE
/SPSTOPH
/CHDE
/CHRDH
/CHWRH
/HWRCNTH
/HDDTH
/HRTOTH
/HATOH
/HBSYDE
/HRDH
+5V VCC
GND
PC43
PC44
PC45
PC48

空白ページ

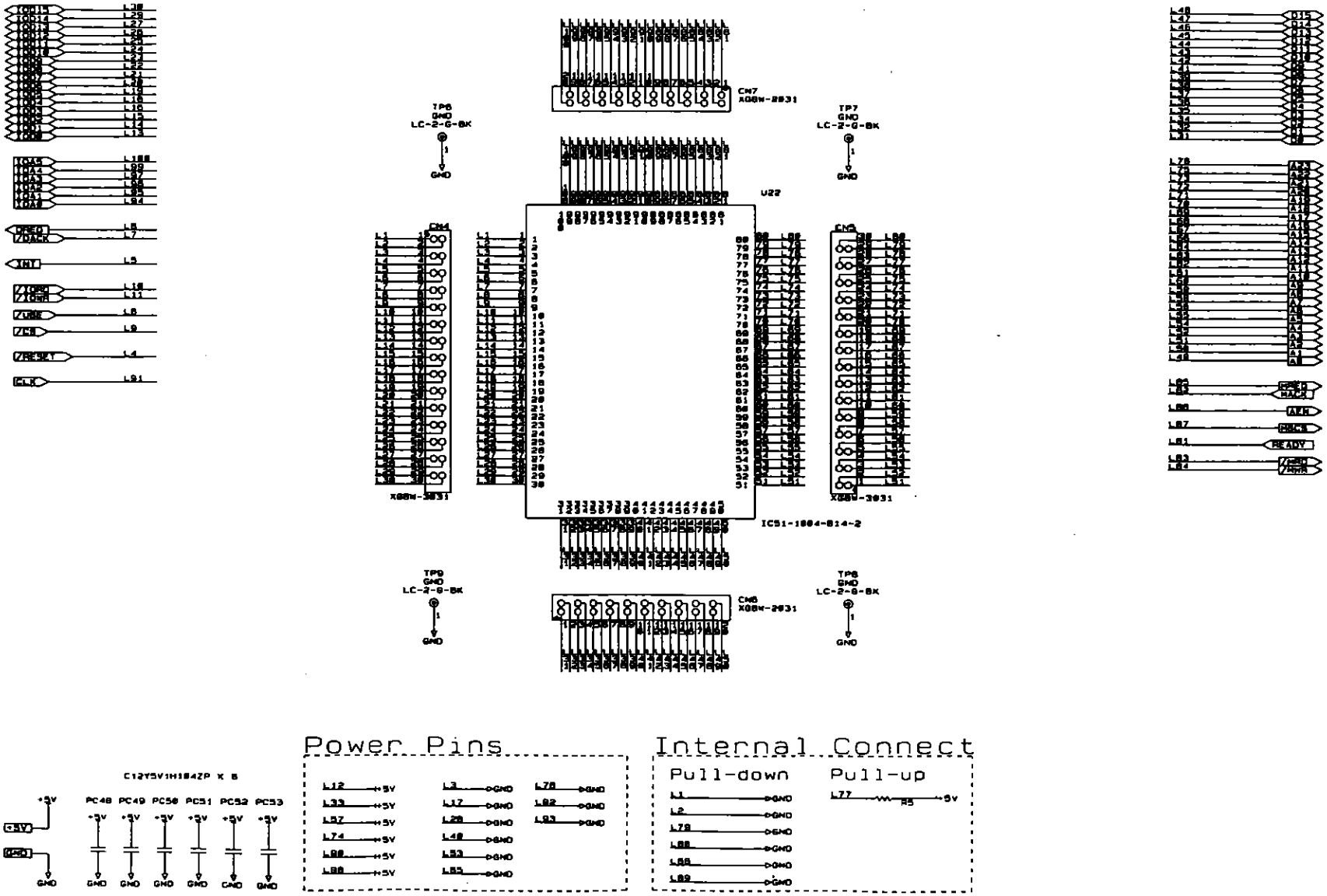
図A－7　μPD72187A





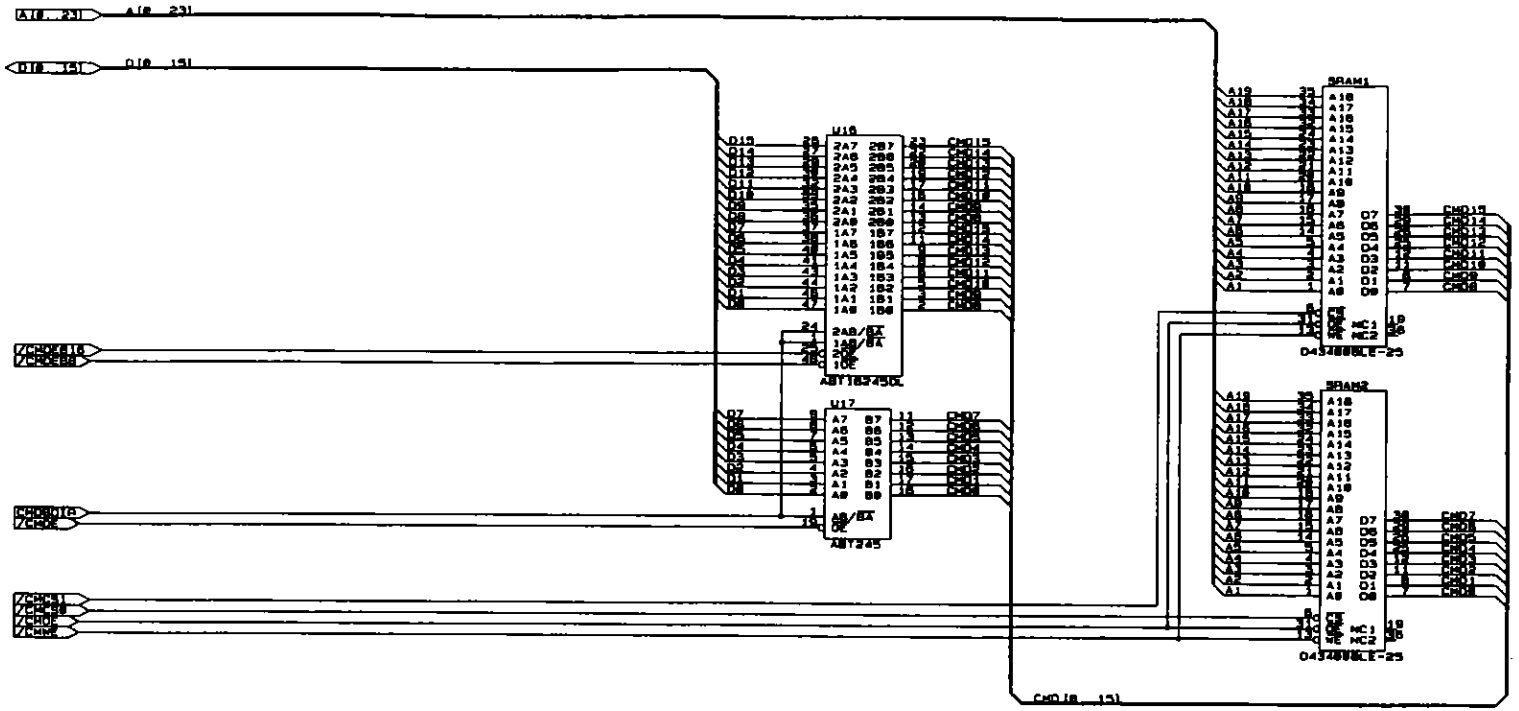
空白ページ

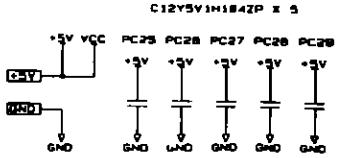
図A－8　μPD72187Aソケット



空白ページ

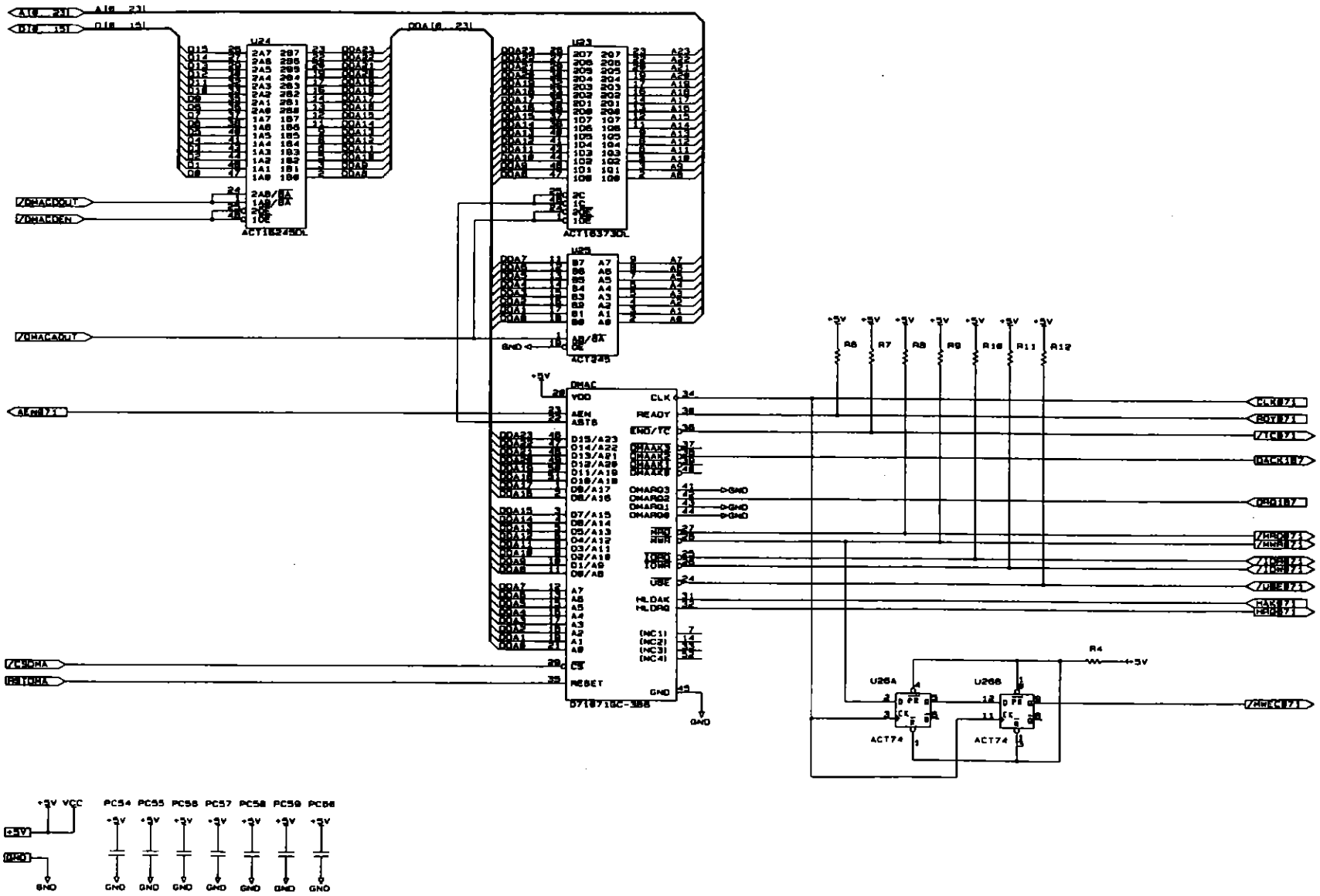
図A－9　コード・メモリ


A[8..23]
D[0..15]
U16
U17
SRAM1
SRAM2
CMD[0..15]


C12Y5V1H184ZP x 5
+5V
GND
VCC
PC25
PC26
PC27
PC28
PC29

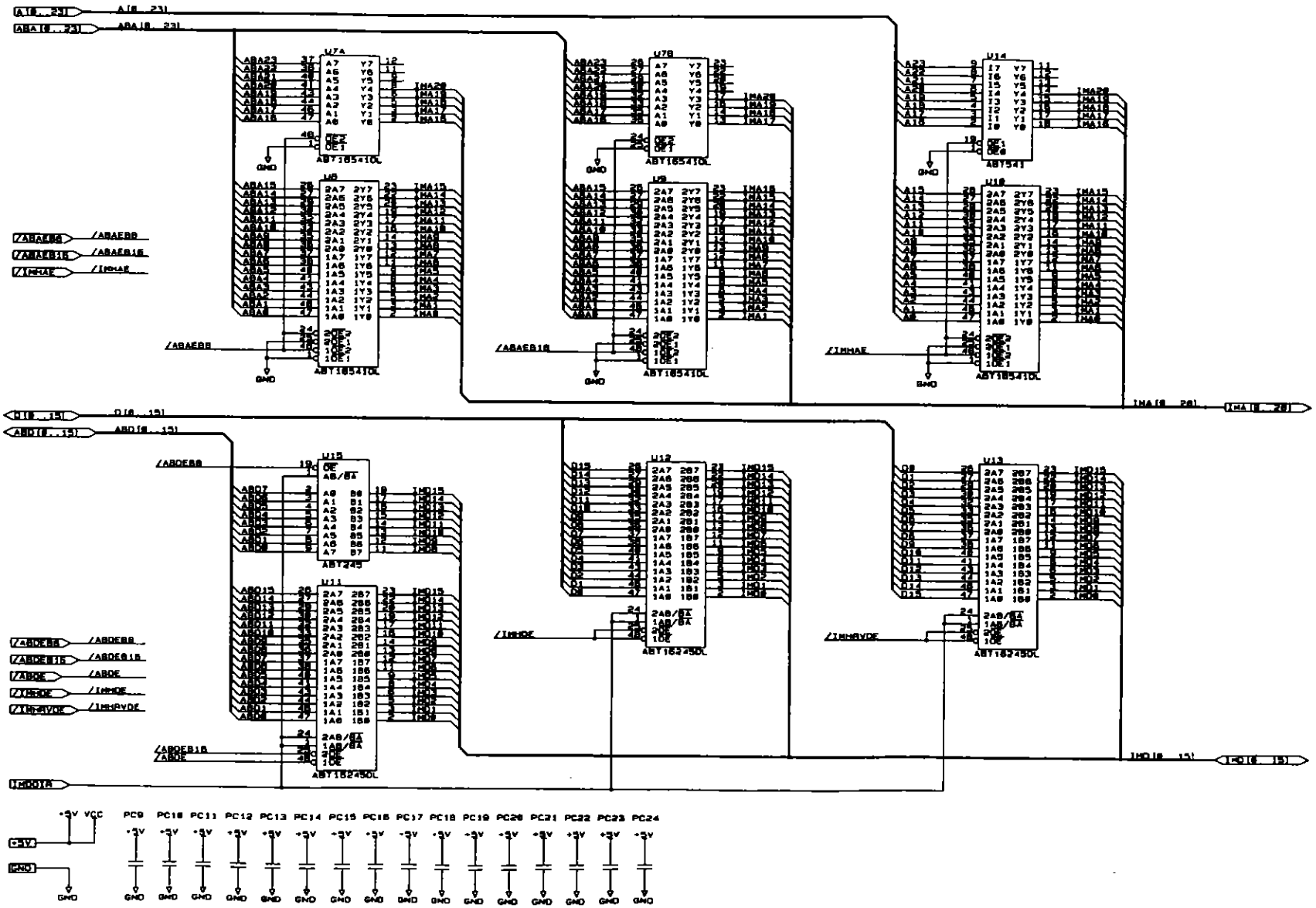
空白ページ

図A－10　DMAC


U24
U23
U25
ACT16245DL
ACT163730L
ACT245
/DMACDOUT
/DMACDEN
/DMACADOUT
AEN[7:1]
/CSDMA
/RSTDMA
DMAC
VDD
CLK
READY
END/TC
AEN
ASTB
RESET
GND
D71871C-386
R8
R7
R9
R10
R11
R12
R4
+5V
U26A
U26B
ACT74
/HWEC[7:1]
PC54 PC55 PC56 PC57 PC58 PC59 PC60
VCC

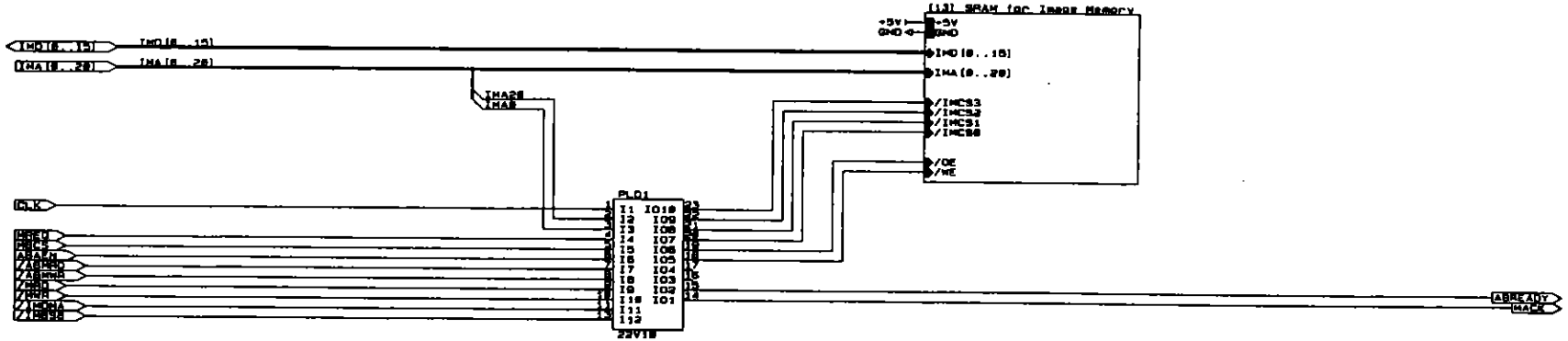
空白ページ

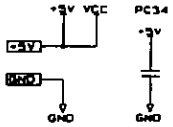
図A－11　イメージ・メモリ用バス・インタフェース



空白ページ

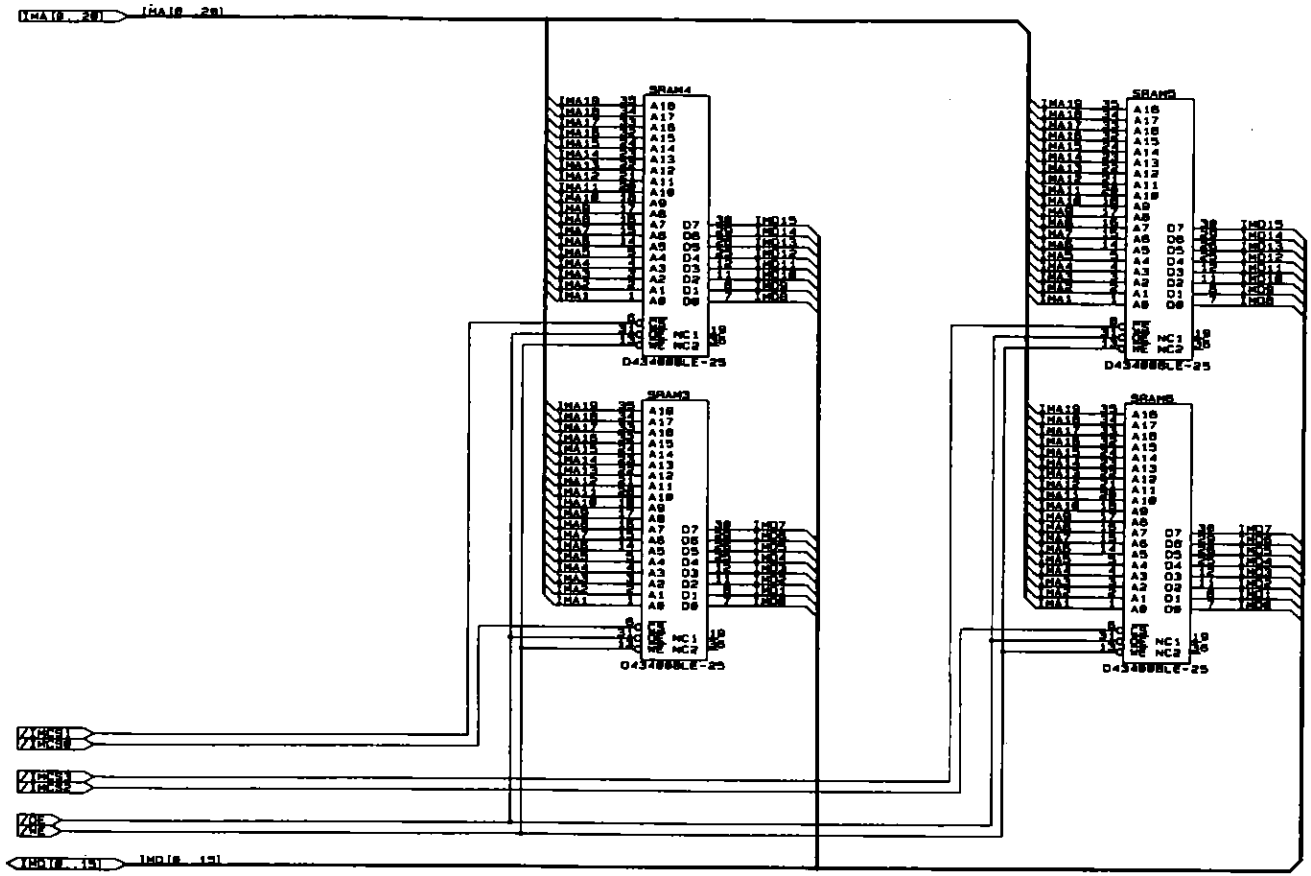
図A－12　イメージ・メモリ・ブロック

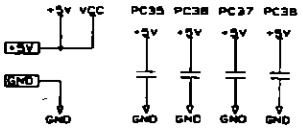




空白ページ

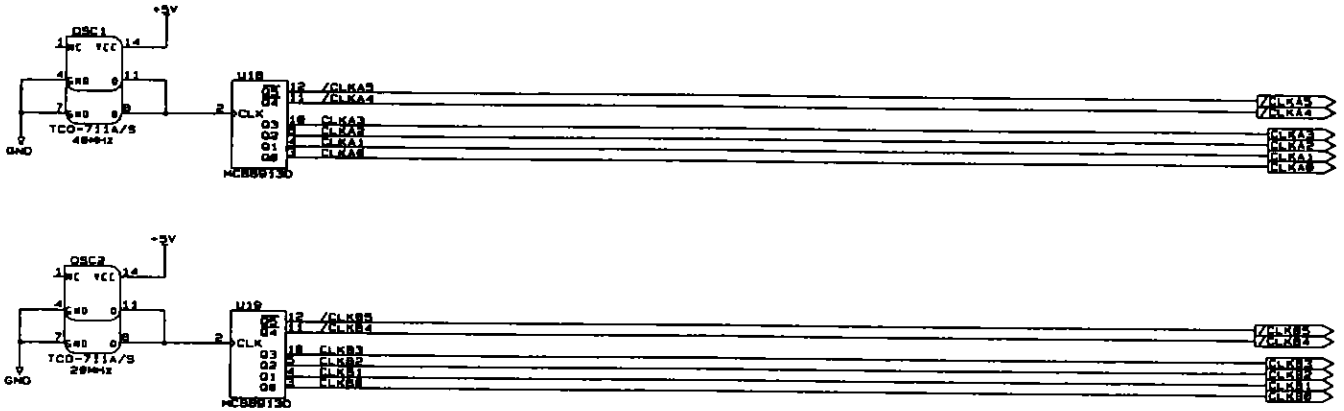
図A－13　イメージ・メモリ用SRAM

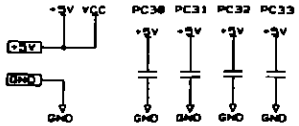




空白ページ

図A－14　クロック・ジェネレータ





空白ページ

付録B　ＰＡＬ論理仕様

ここでは使用したPLDの論理仕様について説明します。論理式はPALASM（PAL Assemble）の表記法を使用しており，次のような意味を持っています。

/	論理否定
*	論理積
+	論理和
=	フリップ・フロップを含まない入出力関係
:=	フリップ・フロップを含む入出力関係

説明は以下の順で行います。

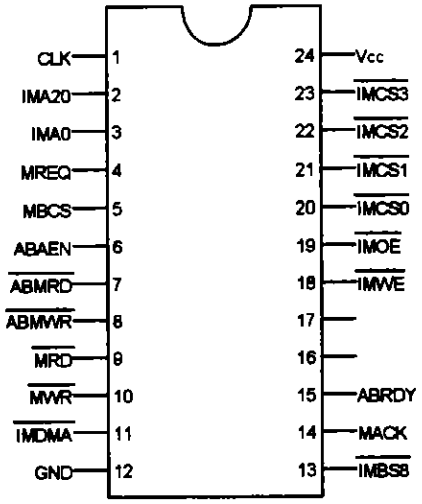
回路図上の番号
- ［品　　名］
- ［機　　能］
- ［ピン配列］
- ［論 理 式］

PLD1A

［品　　名］ PAL22V10

［機　　能］ 画像メモリ・コントローラ

［ピン配置］



［論 理 式］

MACK := MREQ

ABRDY = Vcc

$\overline{IMWE}$ = $\overline{ABMWR}$ * ABAEN

\+ $\overline{MWR}$ * $\overline{IMDMA}$

$\overline{IMOE}$ = $\overline{ABMRD}$ * ABAEN

\+ $\overline{MRD}$ * $\overline{IMDMA}$

$\overline{IMCS0}$ = /IMA20 * /IMA0 * $\overline{IMBS8}$ * ABAEN

\+ /IMA20 * / $\overline{IMBS8}$ * ABAEN

\+ /IMA20 * $\overline{IMDMA}$

$\overline{IMCS1}$ = /IMA20 * IMA0 * $\overline{IMBS8}$ * ABAEN

\+ /IMA20 * / $\overline{IMBS8}$ * ABAEN

\+ /IMA20 * $\overline{IMDMA}$

$\overline{IMCS2}$ = IMA20 * /IMA0 * $\overline{IMBS8}$ * ABAEN

\+ IMA20 * / $\overline{IMBS8}$ * ABAEN

\+ IMA20 * $\overline{IMDMA}$

$\overline{IMCS3}$ = IMA20 * IMA0 * $\overline{IMBS8}$ * ABAEN

\+ IMA20 * / $\overline{IMBS8}$ * ABAEN

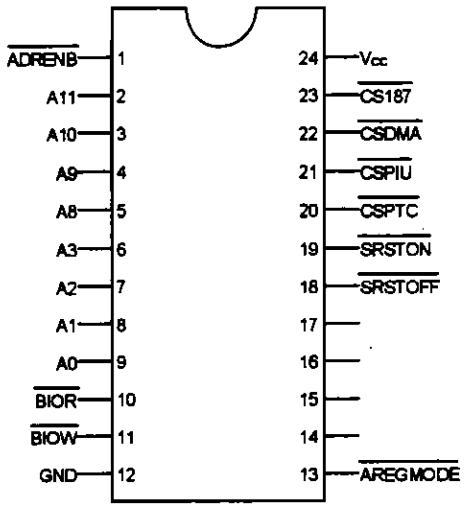
\+ IMA20 * $\overline{IMDMA}$

PLD2

［品　　名］ PAL22V10

［機　　能］ アドレス・デコーダ

［ピン配置］



［論 理 式］

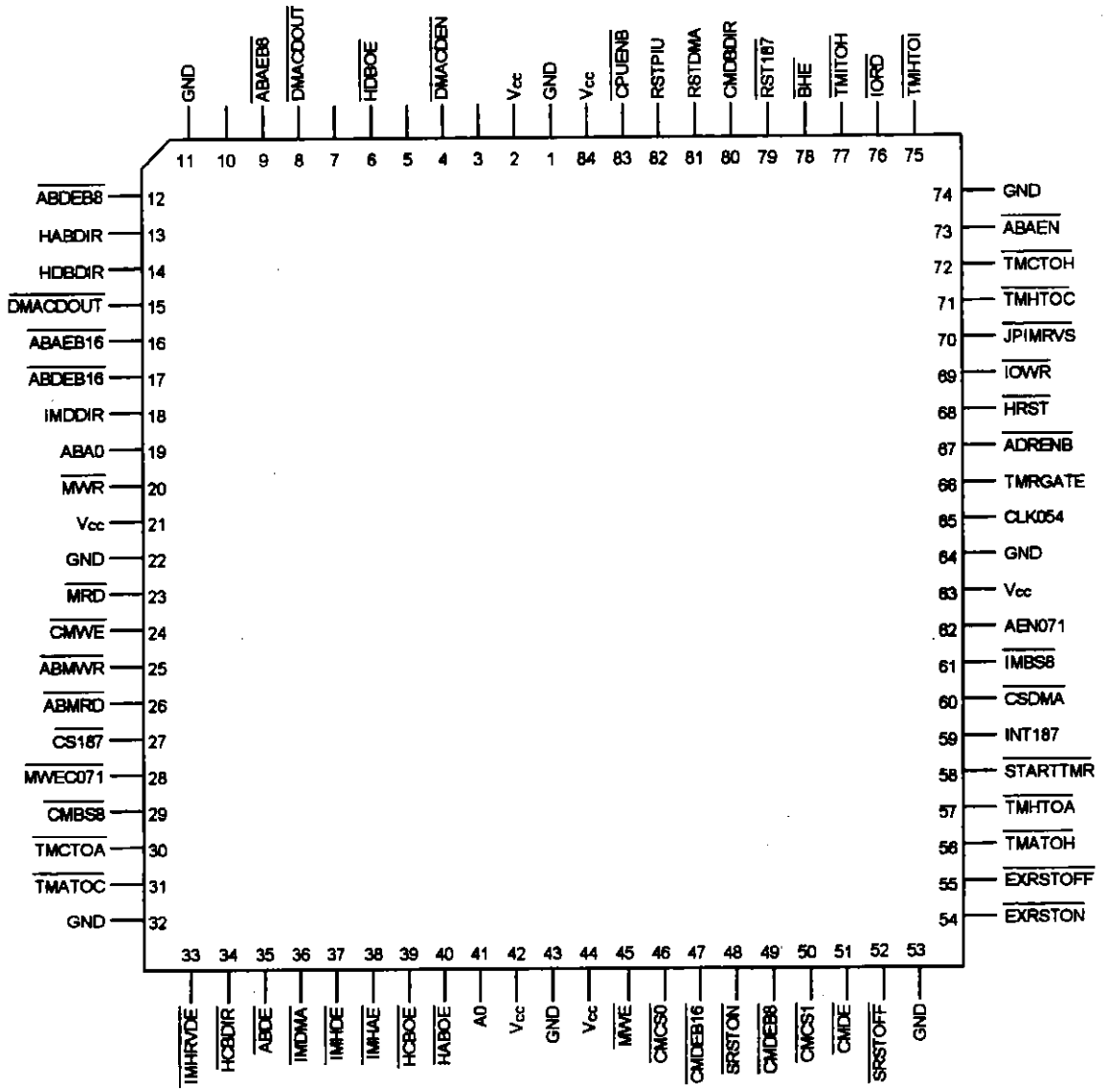
```
STRING    X0Yxh     '( /A11 * /A10 * /A9 * /A8 )'
STRING    X1Yxh     '( /A11 * /A10 * /A9 * A8 )'
STRING    X2Yxh     '( /A11 * /A10 * A9 * /A8 )'
STRING    X3YLh     '( /A11 * /A10 * A9 * A8 * /A3 )'
STRING    X4Yxh     '( /A11 * A10 * /A9 * /A8 )'
STRING    X5YLh     '( /A11 * A10 * /A9 * A8 * /A3 )'
STRING    X6Y0h     '( /A11 * A10 * A9 * /A8 * /A3 * /A2 * /A1 * /A0 )'
STRING    X6Y2h     '( /A11 * A10 * A9 * /A8 * /A3 * /A2 * A1 * /A0 )'
STRING    IOACS     '( /BIOR + /BIOW )'

/CS187   = /ADRENB * ( X0Yxh + X1Yxh + X2Yxh ) * IOACS * /AREGMODE
         + /ADRENB * ( X0Yxh )                 * IOACS * /AREGMODE
/CSDMA   = /ADRENB * X4Yxh * IOACS
/CSPIU   = /ADRENB * X3YLh * IOACS
/CSPTC   = /ADRENB * X5YLh * IOACS
/SRSTON  = /ADRENB * X6Y0h * /BIOW
/SRSTOFF = /ADRENB * X6Y2h * /BIOW
```

CPLD1

［品　　名］ MACH130

［機　　能］ 評価ボード・システム・コントローラ

［ピン配置］



[論 理 式]

```
STRING   IMCS    '(TMITOH  +  TMHTOI)'
STRING   MMCS    '(TMITOH  +  TMHTOI  +  TMCTOH +  TMHTOC )'
STRING   HMRD    '(TMHTOA  +  TMHTOI  +  TMHTOC)'
STRING   HMWR    '(TMATOH  +  TMITOH  +  TMCTOH)'

TMHA     = (TMATOH + TMHTOA)
TMHTOIC  = (TMHTOI + TMHTOC)
TMICTOH  = (TMITOH + TMCTOH)
CMCS     = (TMCTOA + TMATOC + TMCTOH + TMHTOC)
```

[内部信号]

```
NODE 42 SRST
NODE 55 XRST
NODE 50 MRST
NODE 41 TMHA
NODE 56 TMHTOIC
NODE 17 TMICTOH
NODE 54 CMCS
```

[ホスト・バッファ・コントロール]

```
/HABDIR  = /CPUENB * AEN071
HABOE    = /(AEN071 * (TMATOC + TMCTOA) )
/HDBDIR  = IORD + ( MWR * AEN071 * HMWR)
HDBOE    = CPUENB * ADRENB * ( IORD + IOWR )
         + HMRD * MRD * AEN071
         + HMWR * MWR * AEN071
/HCBDIR  = AEN071 * TMHA
         + AEN071 * TMHTOIC * MRD
         + AEN071 * TMICTOH * MWR
HCBOE    = /AEN071 + AEN071 * TMHA
         + AEN071 * TMHTOIC * MRD
         + AEN071 * TMICTOH * MWR
```

[符号メモリ・コントロール]

/CMCS0 = CMCS * /A0 * /(TMHTOC * /MRD) * /(TMCTOH * /MWR)

/CMCS1 = CMCS * (A0 * /BHE) * /(TMHTOC * /MRD) * /(TMCTOH * /MWR)

/CMDEB16 = CMCS * / CMBS8 * /BHE * AEN071

/CMDEB8 = CMCS * /CMBS8 * A0 * AEN071

/CMDE = CMCS * /A0 * AEN071

/CMDBDIR = (TMCTOA + TMCTOH) * /MRD * AEN071

/CMWE = (TMHTOC + TMATOC) * /MWR * AEN071

[画像メモリ・データ・バス・コントロール]

/ABAEB8 = /IMBS8 * ABAEN * /(AEN071 * (TMHTOI + TMITOH))

/ABAEB16 = / /IMBS8 * ABAEN * /(AEN071 * (TMHTOI + TMITOH))

/IMHAE = IMCS * /ABAEN * AEN071

/ABDEB16 = / /IMBS8 * ABAEN * (/ABMRD + /ABMWR) * /(AEN071 * (TMHTOI + TMITOH))

/ABDEB8 = /IMBS8 * ABA0 * ABAEN * (/ABMRD + /ABMWR) * /(AEN071 * (TMHTOI + TMITOH))

/ABDE = /IMBS8 * ABAEN * /ABA0 * (/ABMRD + /ABMWR) * /(AEN071 * (TMHTOI + TMTOH))

+ / /IMBS8 * ABAEN * (/ABMRD + /ABMWR) * /(AEN071 * (TMHTOI + TMITOH))

/IMHDE = (/MRD + /MWR) * IMCS * / JPIMRVS * /ABAEN * AEN071

IMHRVDE = (/MRD + /MWR) * IMCS * /JPIMRVS * /ABAEN * AEN071

/IMDDIR = TMITOH * /MRD * AEN071 + /ABMRD * / /ABMWR * ABAEN

/IMDMA = TMHTOI * /MWR * AEN071 * /ABAEN + TMITOH * /MRD * AEN071 * /ABAEN

[DMAコントローラ　データ／アドレス・コントロール]

/DMACDOUT = /CSDMA * /IORD + MMCS * /MWR * AEN071

/DMACDEN = /CSDMA * (/IORD + /IOWR) + (/MRD + /MWR) * MMCS AEN071

/DMACAOUT = AEN071

/MWE = /MWR * /MWEC071

/MWE.TRST = AEN071

[タイマ・コントロール]

TMRGATE = (/CS187 * IOWR * STARTTMR) + (TMRGATE * /INT187)

TMRGATE.CLKF= CLK054

TMRGATE.RSTF= MRST

［リセット・コントロール］

RSTPIU = $\overline{\text{HRST}}$ + SRST + XRST

$\overline{\text{RST187}}$ = $\overline{\text{HRST}}$ + SRST + XRST

RSTDMA = $\overline{\text{HRST}}$ + SRST + XRST

SRST = $\overline{\text{SRSTON}}$ + (SRST * / $\overline{\text{SRSTOFF}}$ * / $\overline{\text{HRST}}$)

XRST = $\overline{\text{EXRSTON}}$ + (XRST * / $\overline{\text{EXRSTOFF}}$ * / $\overline{\text{HRST}}$)

MRST = $\overline{\text{HRST}}$ + SRST + XRST

［メ　　モ］

★ 付録C　制御シーケンス・プログラムのプログラム・リスト

制御シーケンス・プログラム・リストを示します。

```
/*****************************************************************************
*       ファイル名  :  EXT187A.H
*       内容        :  EXTERN 定義ファイル
*****************************************************************************/

/*  DMAC187A.C  */
extern void dma_htoc() ;
extern void dma_htoi() ;
extern void dma_jtoc() ;
extern void dma_restart() ;
extern void dma_tc() ;
extern void dma_set() ;

/*  MAIN187A.C  */
extern void main() ;
extern void enc_norm() ;
extern void enc_tp() ;
extern void enc_at() ;
extern void enc_tpat() ;
extern void dec_norm() ;
extern void dec_tp() ;
extern void dec_at() ;
extern void dec_tpat() ;
extern void sysinit() ;
extern void load_dat() ;
extern void save_dat() ;

/*  COM187A.C  */

/* uPD72187A */
extern unsigned int d187a_ctr ;
extern unsigned int d187a_strl ;
extern unsigned int d187a_strh ;
extern unsigned int d187a_mdrl ;
extern unsigned int d187a_mdrh ;
extern unsigned int d187a_pell ;
extern unsigned int d187a_pelh ;
extern unsigned int d187a_linl ;
extern unsigned int d187a_linh ;
extern unsigned int d187a_libl ;
extern unsigned int d187a_libh ;
extern unsigned int d187a_stpl ;
extern unsigned int d187a_stph ;
extern unsigned int d187a_atrl ;
extern unsigned int d187a_atrh ;

extern unsigned int d187a_atll ;
extern unsigned int d187a_atlh ;
extern unsigned int d187a_dmsl ;
extern unsigned int d187a_dmsm ;
extern unsigned int d187a_dmsh ;
extern unsigned int d187a_dmel ;
extern unsigned int d187a_dmem ;
extern unsigned int d187a_dmeh ;
extern unsigned int d187a_dbfl ;
extern unsigned int d187a_dbfh ;
extern unsigned int d187a_hsb ;
```

```
extern unsigned int d187a_mkb ;

extern unsigned int d187a_mkwl ;
extern unsigned int d187a_mkwh ;
extern unsigned int d187a_cmd1 ;
extern unsigned int d187a_cmd2 ;
extern unsigned int d187a_ist ;
extern unsigned int d187a_imr ;
extern unsigned int d187a_msk ;
extern unsigned int d187a_tpr ;

/* uPD71071 */
extern unsigned int b_dma_init ;
extern unsigned int b_dma_channel ;
extern unsigned int b_dma_countl ;
extern unsigned int b_dma_counth ;
extern unsigned int b_dma_adrl ;
extern unsigned int b_dma_adrm ;
extern unsigned int b_dma_adrh ;
extern unsigned int b_dma_devicel ;
extern unsigned int b_dma_deviceh ;
extern unsigned int b_dma_mod ;
extern unsigned int b_dma_status ;
extern unsigned int b_dma_tmpl ;
extern unsigned int b_dma_tmph ;
extern unsigned int b_dma_request ;
extern unsigned int b_dma_mask ;

/* uPD71055 */
extern unsigned int b_sysreg_pt0 ;
extern unsigned int b_sysreg_pt1 ;
extern unsigned int b_sysreg_pt2 ;
extern unsigned int b_sysreg_cmd ;

/* uPD71054 */
extern unsigned int b_tmcnt_ct0 ;
extern unsigned int b_tmcnt_ct1 ;
extern unsigned int b_tmcnt_ct2 ;
extern unsigned int b_tmcnt_ctl ;

/* reset registar */
extern unsigned int b_reset_on ;
extern unsigned int b_reset_off ;

/* Host bus */
extern unsigned char h_dma_cmd ;
extern unsigned char h_dma_mod ;
extern unsigned char h_dma_request ;
extern unsigned char h_dma_singmsk ;
extern unsigned char h_dma_allmsk ;
extern unsigned char h_dma_status ;
extern unsigned char h_dma_count ;
extern unsigned char h_dma_addr ;
extern unsigned char h_dma_bank ;
extern unsigned char h_dma_clrmsk ;
extern unsigned char h_dma_bnkaddai ;
```

```
/*****************************************************************************
*       ファイル名  :  APP187A.H
*       内容        :  Include & define
*****************************************************************************/

/* Include ファイル */

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>


/* Switch 3 & 4 (ボードの I/O マッピング・アドレス) */
#define IOADDR 0x000050D0


#define FTMS_MAX 8          /*  ファイルからボード上のメモリへの転送量  */

#define CM 0                /*  符号メモリ選択フラグ  */
#define IM 1                /*  画像メモリ選択フラグ  */


/*************************************************************
*     μPD72187Aのレジスタ・アドレス
*************************************************************/

#define A_CTR               0x00000000
#define A_STRL              0x00000002
#define A_STRH              0x00000003
#define A_MDRL              0x00000004
#define A_MDRH              0x00000005
#define A_PELL              0x00000006
#define A_PELH              0x00000007
#define A_LINL              0x00000008
#define A_LINH              0x00000009
#define A_LIBL              0x0000000A
#define A_LIBH              0x0000000B
#define A_STPL              0x0000000C
#define A_STPH              0x0000000D
#define A_ATRL              0x0000000E
#define A_ATRH              0x0000000F

#define A_ATLL              0x00000100
#define A_ATLH              0x00000101
#define A_DMSL              0x00000102
#define A_DMSM              0x00000103
#define A_DMSH              0x00000104
#define A_DMEL              0x00000106
#define A_DMEM              0x00000107
#define A_DMEH              0x00000108
#define A_DBFL              0x0000010A
#define A_DBFH              0x0000010B
#define A_HSB               0x0000010C
#define A_MKB               0x0000010E

#define A_MKWL              0x00000200
```

```
#define A_MKWH              0x00000201
#define A_CMD1              0x00000202
#define A_CMD2              0x00000204
#define A_IST               0x00000206
#define A_LMR               0x00000208
#define A_MSK               0x0000020A
#define A_TPR               0x0000020C


/*************************************************************
*      μPD71071 のレジスタ・アドレス
*************************************************************/

#define D_INIT              0x00000400
#define D_CHNL              0x00000401
#define D_CNTL              0x00000402
#define D_CNTH              0x00000403
#define D_ADRL              0x00000404
#define D_ADRM              0x00000405
#define D_ADRH              0x00000406
#define D_DEVL              0x00000408
#define D_DEVH              0x00000409
#define D_MOD               0x0000040A
#define D_STAT              0x0000040B
#define D_TMPL              0x0000040C
#define D_TMPH              0x0000040D
#define D_REQ               0x0000040E
#define D_MASK              0x0000040F


/*************************************************************
*      μPD71055 のレジスタ・アドレス
*************************************************************/

#define PIU_PT0             0x00000300      /*  システム・レジスタ  ポート  0     */
#define PIU_PT1             0x00000302      /*  システム・レジスタ  ポート  1     */
#define PIU_PT2             0x00000304      /*  システム・レジスタ  ポート  2     */
#define PIU_CMD             0x00000306      /*  システム・レジスタ  コマンド      */


/*************************************************************
*      μPD71054 のレジスタ・アドレス
*************************************************************/

#define PTC_CT0             0x00000500      /*  タイマ／カウンタ  0               */
#define PTC_CT1             0x00000502      /*  タイマ／カウンタ  1               */
#define PTC_CT2             0x00000504      /*  タイマ／カウンタ  2               */
#define PTC_CTL             0x00000506      /*  タイマ・コントロール・ワード      */


/*************************************************************
*      μPD71054 のレジスタ・アドレス
*************************************************************/

#define RESET_ON            0x00000600
#define RESET_OFF           0x00000602
```

```
/*************************************************************
*      ホスト(PC-9800)のDMAレジスタ・アドレス
*************************************************************/

#define H_DMA_CMD  0x0011          /*  ライト時                */
#define H_DMA_MOD  0x0017
#define H_DMA_REQ  0x0013
#define H_DMA_SMSK 0x0015
#define H_DMA_AMSK 0x001F
#define H_DMA_STAT 0x0011          /*  リード時                */
#define H_DMA_CNT  0x000F          /*  DMAチャネル  3          */
#define H_DMA_ADDR 0x000D          /*  DMAチャネル  3          */
#define H_DMA_BANK 0x0025          /*  DMAチャネル  3          */
#define H_DMA_CMSK 0x001D
#define H_DMA_BAMD 0x0029
```

```
/*****************************************************************************
*       ファイル名　：　COM187A.C
*       内容　　　：　静的変数定義ファイル
*****************************************************************************/

#include "app187a.h"
#include "ext187a.h"


/*********************************************************
*       μPD72187Aのレジスタの実アドレス
*********************************************************/

unsigned int      d187a_ctr         = A_CTR  + IOADDR ;
unsigned int      d187a_strl        = A_STRL + IOADDR ;
unsigned int      d187a_strh        = A_STRH + IOADDR ;
unsigned int      d187a_mdrl        = A_MDRL + IOADDR ;
unsigned int      d187a_mdrh        = A_MDRH + IOADDR ;
unsigned int      d187a_pell        = A_PELL + IOADDR ;
unsigned int      d187a_pelh        = A_PELH + IOADDR ;
unsigned int      d187a_linl        = A_LINL + IOADDR ;
unsigned int      d187a_linh        = A_LINH + IOADDR ;
unsigned int      d187a_libl        = A_LIBL + IOADDR ;
unsigned int      d187a_libh        = A_LIBH + IOADDR ;
unsigned int      d187a_stpl        = A_STPL + IOADDR ;
unsigned int      d187a_stph        = A_STPH + IOADDR ;
unsigned int      d187a_atrl        = A_ATRL + IOADDR ;
unsigned int      d187a_atrh        = A_ATRH + IOADDR ;

unsigned int      d187a_atll        = A_ATLL + IOADDR ;
unsigned int      d187a_atlh        = A_ATLH + IOADDR ;
unsigned int      d187a_dmsl        = A_DMSL + IOADDR ;
unsigned int      d187a_dmsm        = A_DMSM + IOADDR ;
unsigned int      d187a_dmsh        = A_DMSH + IOADDR ;
unsigned int      d187a_dmel        = A_DMEL + IOADDR ;
unsigned int      d187a_dmem        = A_DMEM + IOADDR ;
unsigned int      d187a_dmeh        = A_DMEH + IOADDR ;
unsigned int      d187a_dbfl        = A_DBFL + IOADDR ;
unsigned int      d187a_dbfh        = A_DBFH + IOADDR ;
unsigned int      d187a_hsb         = A_HSB  + IOADDR ;
unsigned int      d187a_mkb         = A_MKB  + IOADDR ;

unsigned int      d187a_mkwl        = A_MKWL + IOADDR ;
unsigned int      d187a_mkwh        = A_MKWH + IOADDR ;
unsigned int      d187a_cmd1        = A_CMD1 + IOADDR ;
unsigned int      d187a_cmd2        = A_CMD2 + IOADDR ;
unsigned int      d187a_ist         = A_IST  + IOADDR ;
unsigned int      d187a_lmr         = A_LMR  + IOADDR ;
unsigned int      d187a_msk         = A_MSK  + IOADDR ;
unsigned int      d187a_tpr         = A_TPR  + IOADDR ;


/*********************************************************
*       μPD71071のレジスタの実アドレス
*********************************************************/

unsigned int      b_dma_init        = D_INIT + IOADDR ;
```

```
unsigned int      b_dma_channel     = D_CHNL + IOADDR ;
unsigned int      b_dma_countl      = D_CNTL + IOADDR ;
unsigned int      b_dma_counth      = D_CNTH + IOADDR ;
unsigned int      b_dma_adrl        = D_ADRL + IOADDR ;
unsigned int      b_dma_adrm        = D_ADRM + IOADDR ;
unsigned int      b_dma_adrh        = D_ADRH + IOADDR ;
unsigned int      b_dma_devicel     = D_DEVL + IOADDR ;
unsigned int      b_dma_deviceh     = D_DEVH + IOADDR ;
unsigned int      b_dma_mod         = D_MOD  + IOADDR ;
unsigned int      b_dma_status      = D_STAT + IOADDR ;
unsigned int      b_dma_tmpl        = D_TMPL + IOADDR ;
unsigned int      b_dma_tmph        = D_TMPH + IOADDR ;
unsigned int      b_dma_request     = D_REQ  + IOADDR ;
unsigned int      b_dma_mask        = D_MASK + IOADDR ;

/*************************************************************
*       μPD71055のレジスタの実アドレス
*************************************************************/

unsigned int      b_sysreg_pt0      = PIU_PT0 + IOADDR ;
unsigned int      b_sysreg_pt1      = PIU_PT1 + IOADDR ;
unsigned int      b_sysreg_pt2      = PIU_PT2 + IOADDR ;
unsigned int      b_sysreg_cmd      = PIU_CMD + IOADDR ;

/*************************************************************
*       μPD71054のレジスタの実アドレス
*************************************************************/

unsigned int      b_tmcnt_ct0       = PTC_CT0 + IOADDR ;
unsigned int      b_tmcnt_ct1       = PTC_CT1 + IOADDR ;
unsigned int      b_tmcnt_ct2       = PTC_CT2 + IOADDR ;
unsigned int      b_tmcnt_ctl       = PTC_CTL + IOADDR ;

/*************************************************************
*       リセット・レジスタの実アドレス
*************************************************************/

unsigned int      b_reset_on        = RESET_ON  + IOADDR ;
unsigned int      b_reset_off       = RESET_OFF + IOADDR ;

/*************************************************************
*       ホスト側DMAコントローラの実アドレス
*************************************************************/

unsigned char     h_dma_cmd         = H_DMA_CMD ;
unsigned char     h_dma_mod         = H_DMA_MOD ;
unsigned char     h_dma_request     = H_DMA_REQ ;
unsigned char     h_dma_singmsk     = H_DMA_SMSK ;
unsigned char     h_dma_allmsk      = H_DMA_AMSK ;
unsigned char     h_dma_status      = H_DMA_STAT ;
unsigned char     h_dma_count       = H_DMA_CNT ;
unsigned char     h_dma_addr        = H_DMA_ADDR ;
unsigned char     h_dma_bank        = H_DMA_BANK ;
```

```
unsigned char     h_dma_clrmsk      = H_DMA_CMSK ;
unsigned char     h_dma_bnkaddai    = H_DMA_BAMD ;
```

```
/*****************************************************************************
*       ファイル名　:  DMAC187A.C
*       内容　　　:  DMAコントローラ制御プログラム
*****************************************************************************
*       DMA channel #0,#1 : ホストマシン <--> 符号メモリ／画像メモリ
*                     #2 : μPD72187A <--> 符号メモリ／ホスト・マシン
*****************************************************************************/

/* Include file */
#include "APP187A.H"
#include "EXT187A.H"


/*****************************************************************************
*       ホスト・マシン符号メモリ間データ転送
*
*       void dma_htoc ( mod, ptrh, adrc, dtl )
*       unsigned char      mod  : DMA転送方向  0 : ホスト→符号メモリ
*                                              1 : 符号メモリ→ホスト
*       unsigned char far *ptrh : ホスト側DMA開始アドレス・ポインタ
*       unsigned long      adrc : 符号メモリ側DMA開始アドレス
*       unsigned long      dtl  : データ転送サイズ  (byte)
*****************************************************************************/
void dma_htoc( mod, ptrh, adrc, dtl )
unsigned char       mod  ;
unsigned char far  *ptrh ;
unsigned long       adrc ;
unsigned long       dtl  ;
{
        unsigned long     adrh, sizew, adr1, adr2 ;
        unsigned char     stat ;

        /* データ転送時は一時的に16ビット幅に設定 */
        stat = inp( b_sysreg_pt0 );
        outp( b_sysreg_pt0, ( stat | 0x0C ));

        /* DMA全チャネル・マスク */
        outp( b_dma_mask, 0x0F );

        /* DMAモード設定 */
        if ( mod == 0 ) {

                /* ホスト→符号メモリ転送 */
                outp( b_sysreg_pt1, 0xFE );

                /* ポインタ型のアドレスをlong型に変換 */
                adr1 = (unsigned long)FP_SEG( ptrh );
                adr2 = (unsigned long)FP_OFF( ptrh );
                adrh = ( adr1 * 16 ) + adr2;

                sizew = dtl;

                /* DMAコントローラ設定 */
                outp( H_DMA_MOD, 0xC3 );
                outp( H_DMA_SMSK,0x03 );
                dma_set( (unsigned char)0, sizew, adrh, (unsigned char)0x81 );
                dma_set( (unsigned char)1, sizew, adrc, (unsigned char)0x81 );
```

```
                outp( b_dma_devicel, 0x01 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_request, 0x01 );

                dma_tc();

                outp( b_dma_mask, 0x0F );

                outp( b_sysreg_pt1, 0xFE );
                outp( b_sysreg_pt0, stat );
        } else {

                /* 符号メモリ→ホスト転送 */
                outp( b_sysreg_pt1, 0xFD );

                /* ポインタ型のアドレスを long 型に変換 */
                adr1 = (unsigned long) FP_SEG( ptrh );
                adr2 = (unsigned long) FP_OFF( ptrh );
                adrh = ( adr1 * 16 ) + adr2;

                sizew = dtl;

                /* DMA コントローラ設定 */
                dma_set( (unsigned char)0, sizew, adrc, (unsigned char)0x81 );
                outp( H_DMA_MOD, 0xC3 );
                outp( H_DMA_SMSK,0x03 );
                dma_set( (unsigned char)1, sizew, adrh, (unsigned char)0x81 );

                outp( b_dma_devicel, 0x21 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_request, 0x01 );

                dma_tc();

                outp( b_dma_mask, 0x0F );

                outp( b_sysreg_pt1, 0xFD );
                outp( b_sysreg_pt0, stat );
        }
}


/*****************************************************************************
*       ホスト・マシン画像メモリ間データ転送
*
*       void dma_htoi ( mod, ptrh, adri, dtl )
*       unsigned char      mod  : DMA転送方向  0 : ホスト→画像メモリ
*                                              1 : 画像メモリ→ホスト
*       unsigned char far *ptrh : ホスト側 DMA 開始アドレス・ポインタ
*       unsigned long      adri : 画像メモリ側 DMA 開始アドレス
*       unsigned long      dtl  : データ転送サイズ  (byte)
*****************************************************************************/
void dma_htoi( mod, ptrh, adri, dtl )
unsigned char         mod  ;
```

```
unsigned char far  *ptrh ;
unsigned long         adri ;
unsigned long         dtl  ;
{
        unsigned long     adrh, sizew, adr1, adr2 ;
        unsigned char     stat ;

        /* データ転送時は一時的に 16 ビット幅に設定 */
        stat = inp( b_sysreg_pt0 );
        outp( b_sysreg_pt0, ( stat | 0x0C ));

        /* DMA 全チャネルマスク */
        outp( b_dma_mask, 0x0F );

        /* DMA モード設定 */
        if ( mod == 0 ) {

                /* ホスト→画像メモリ転送 */
                outp( b_sysreg_pt1, 0xFB );

                /* ポインタ型のアドレスを long 型に変換 */
                adr1 = (unsigned long)FP_SEG( ptrh );
                adr2 = (unsigned long)FP_OFF( ptrh );
                adrh = ( adr1 * 16 ) + adr2;

                sizew = dtl;

                /* DMA コントローラ設定 */
                outp( H_DMA_MOD, 0xC3 );
                outp( H_DMA_SMSK, 0x03 );
                dma_set( (unsigned char)0, sizew, adrh, (unsigned char)0x81 );
                dma_set( (unsigned char)1, sizew, adri, (unsigned char)0x81 );

                outp( b_dma_devicel, 0x01 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_request, 0x01 );

                dma_tc();

                outp( b_dma_mask, 0x0F );

                outp( b_sysreg_pt1, 0xFB );
                outp( b_sysreg_pt0, stat );
        } else {

                /* 画像メモリ→ホスト転送 */
                outp( b_sysreg_pt1, 0xF7 );

                /* ポインタ型のアドレスを long 型に変換 */
                adr1 = (unsigned long)FP_SEG( ptrh );
                adr2 = (unsigned long)FP_OFF( ptrh );
                adrh = ( adr1 * 16 ) + adr2 ;

                sizew = dtl ;

                dma_set( (unsigned char)0, sizew, adri, (unsigned char)0x81 );
```

```
                outp( H_DMA_MOD, 0xC3 );
                outp( H_DMA_SMSK,0x03 );
                dma_set( (unsigned char)1, sizew, adrh, (unsigned char)0x81 );

                outp( b_dma_devicel, 0x21 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_request, 0x01 );

                dma_tc();

                outp( b_dma_mask, 0x0F );

                outp( b_sysreg_pt1, 0xF7 );
                outp( b_sysreg_pt0, stat );
        }
}


/*****************************************************************************
*       μPD72187A符号メモリ間転送
*
*       void dma_jtoc ( mod, adrc, dtl )
*       unsigned char mod  : DMA転送方向 0 : μPD72187A→符号メモリ
*                                        1 : 符号メモリ→μPD72187A
*       unsigned char far *adrc     : 符号メモリ側DMA転送開始アドレス
*       unsigned int dtl   : データ転送サイズ  (byte)
*****************************************************************************/
void dma_jtoc( mod, adrc, dtl )
unsigned char      mod ;
unsigned long      adrc ;
unsigned long      dtl ;
{
        unsigned long sizew ;

        outp( b_dma_mask, 0x0F ) ;

        /* DMAモード設定 */
        if ( mod == 0 ) {

                /* ホスト→符号メモリ転送 */
                outp( b_sysreg_pt1, 0xDF);

                sizew = dtl ;

                dma_set( (unsigned char)2, sizew, adrc, (unsigned char)0x04 );

                outp( b_dma_devicel, 0x00 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_mask, 0x0B );
        }
        else {

                /* 符号メモリ→ホスト転送 */
                outp( b_sysreg_pt1, 0xEF);
```

```
                sizew = dtl ;

                dma_set( (unsigned char)2, sizew, adrc, (unsigned char)0x08 );

                outp( b_dma_devicel, 0x00 );
                outp( b_dma_deviceh, 0x01 );

                outp( b_dma_mask, 0x0B );
        }
}


/*****************************************************************************
*       DMAコントローラ　TC（ターミナル・カウント）検出
*
*       void dma_tc()
*****************************************************************************/
void dma_tc( void )
{
        unsigned char stat ;

        stat = 0 ;

        while ( (stat & 0x03) != 0x03 ) {
                stat = inp( b_dma_status ) ;
        }
}


/*****************************************************************************
*       DMAコントローラ設定
*
*       void dma_set( channel, count, addr, mod )
*       unsigned char channel  : DMAチャネル ( 0 or 1 or 2 )
*       unsigned long count    : DMA転送カウント
*       unsigned long addr     : DMA転送アドレス
*       unsigned char mod      : DMAモード
*****************************************************************************/
void dma_set( channel, count, addr, mod )
unsigned char     channel ;
unsigned long     count ;
unsigned long     addr ;
unsigned char     mod ;
{
        outp( b_dma_channel, channel ) ;

        count-- ;
        outp( b_dma_countl, (unsigned char) (count & 0x00FF) ) ;
        outp( b_dma_counth, (unsigned char) ( (count >> 8) & 0x00FF ) ) ;

        outp( b_dma_adrl, ( unsigned char )( (addr) & 0x000000FF ) ) ;
        outp( b_dma_adrm, ( unsigned char )( (addr >>  8) & 0x000000FF ) ) ;
        outp( b_dma_adrh, ( unsigned char )( (addr >> 16) & 0x000000FF ) ) ;

        outp( b_dma_mod, mod ) ;
}
```

```
/*************************************************************************
*       ファイル名　:  MAIN187A.C
*       内容　　:  μPD72187A制御シーケンス・プログラム
*************************************************************************/

/* Include File */
#include "APP187A.H"
#include "EXT187A.H"

/*************************************************************************
*       メイン
*************************************************************************/
void main (argc, argv)
int     argc ;
char    *argv[] ;
{

        FILE    *forg, *fnew ;
        unsigned long     filesize ;
        int     i, j ;


        /* 起動時エラー処理 & ファイル・オープン */
        if ( argc != 3 ) {
                printf("Syntax error ..¥n");
                printf("書式 : demo187a [元のファイル名] [新しいファイル名] ¥n");
                exit(1);

        }

        if ((forg = fopen (argv[1], "rb")) == NULL) {
                printf("[%s] がありません。¥n", argv[1] );
                exit(1);
        }

        if ((fnew = fopen (argv[2], "ab")) == NULL) {
                printf("[%s] を開くことができません。 ¥n", argv[2] );
                exit(1);
        }


        /* ファイル・サイズ検索 */
        fseek (forg, 0L, 2);
        filesize = (unsigned long)ftell( forg );
        printf("ファイル・サイズは %lu byte です。¥n¥n", filesize );
        rewind ( forg );


        /* Input method & type */
        printf("処理を入力してください。 ( 1.符号化 or 2.復号化 )  ");
        scanf("%d", &i);

        printf("処理形態を入力してください¥n");
        printf("( 1. 通常処理  2. TP処理あり  3. AT処理あり  4. TP・AT処理あり )  ");
        scanf("%d", &j);

        j = (i * 4 + j) - 4;
```

```
        switch ( j ) {
                case 1 : printf("符号化（通常）処理開始¥n");
                         enc_norm ( forg, fnew, filesize );
                         break;

                case 2 : printf("符号化（TPあり）処理開始¥n");
                         enc_tp ( forg, fnew, filesize );
                         break;

                case 3 : printf("符号化（ATあり）処理開始¥n");
                         enc_at ( forg, fnew, filesize );
                         break;

                case 4 : printf("符号化（TP&ATあり）処理開始¥n");
                         enc_tpat ( forg, fnew, filesize );
                         break;

                case 5 : printf("復号化（通常）処理開始¥n");
                         dec_norm ( forg, fnew, filesize );
                         break;

                case 6 : printf("復号化（TPあり）処理開始¥n");
                         dec_tp ( forg, fnew, filesize );
                         break;

                case 7 : printf("復号化（ATあり）処理開始¥n");
                         dec_at ( forg, fnew, filesize );
                         break;

                case 8 : printf("復号化（TP&ATあり）処理開始¥n");
                         dec_tpat ( forg, fnew, filesize );
                         break;

                default: break;
        }

        fclose ( fnew );
        fclose ( forg );

        printf("処理終了¥n¥n");
}


/*****************************************************************************
*       符号化（通常）処理シーケンス
*
*       void enc_norm ( forg, fnew, filesize )
*               FILE *forg              :  画像データ・ファイルのポインタ
*               FILE *fnew              :  符号データ・ファイルのポインタ
*               unsigned long filesize  :  画像データ・ファイルのサイズ（byte）
*****************************************************************************/
void enc_norm( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
```

```
unsigned char     stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l ;
unsigned int      hsize, vline, strip ;
int               stripe_line, stripe_amari ; /* ストライプ数 */
int               flag1 = 0 ;                 /* 第１ストライプ処理を示すフラグ */
int               i ;                         /* 汎用変数 */

/* 主走査画素数、副走査ライン数の入力 */
printf("主走査画素数は？　：");
scanf("%ud", &hsize );
if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
} else if ( hsize <= 0 ) {
        hsize = 1 ;
}

printf("副走査ライン数は？：");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？：");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}


/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始 ¥n");
load_dat ( forg, filesize, 0L, 1M );

/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0 ;
```

```
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}

/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0 ;
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INT のクリア */
outp( d187a_ist, 0x00 );

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x01 );
outp( d187a_mdrh, 0x08 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );

/* ストライプ処理のための for ループ */

for ( i = stripe_line ; i > 0 ; i — ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        outp( d187a_atrl, 0x00 );
        outp( d187a_atrh, 0x00 );

        outp( d187a_hsb, 0x01 );

/* 画像データ DMA 転送アドレス設定 */

        if ( flag1 == 0 ) {
        /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
                outp( d187a_dmsm, 0x00 );
                outp( d187a_dmsh, 0x00 );
                /* 終了アドレス */
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );

                /* ホスト・バス側外部 DMA コントローラ起動 */
                printf("符号バス(ホスト・バス)側 DMA 転送開始¥n¥n") ;
                dma_jtoc( (unsigned char)0, (unsigned long)0, (unsigned long)50000L );

                flag1 = 1 ;
        } else {
                /* 開始アドレス */
```

```
        stat = inp( d187a_dmel );
        outp( d187a_dmsl, stat );
        stat = inp( d187a_dmem );
        outp( d187a_dmsm, stat );
        stat = inp( d187a_dmeh );
        outp( d187a_dmsh, stat );

        /* 終了アドレス */
        outp( d187a_dmel, 0xBF );
        outp( d187a_dmem, 0xD4 );
        outp( d187a_dmeh, 0x07 );
}

/* μPD72187A符号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT 発生¥n");

stat = 0 ;
while ( (stat & 0x01) != 0x01 ) {
        stat = inp( d187a_ist );
}
printf("STFEND 確認¥n¥n") ;

/* INTクリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0xDE;
outp( d187a_ist, stat );


/* マーカ・コード付加 */

/* ホスト・バスをマーカ・コードに設定 */
outp( d187a_hsb, 0x00 );

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0xFF );
printf(" MK (FFH) 付加¥n") ;

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0x02 );
printf(" MK (02H) 付加¥n") ;
```

```
                /* ダミー・データ付加 */
                stat = 0 ;
                while ( 1 ) {
                        stat = inp( d187a_strl );
                        printf("strl = 0x%2X    ¥n", stat );

                        /* ダミー・データ付加する必要あり？ */
                        if ( (( stat & 0x02 ) == 0x02 ) && (( stat & 0x04 ) == 0x04 ) ) {
                                if ((stat & 0x10) == 0x10) {
                                        outp( d187a_mkb, 0x00 );
                                        printf("ダミー・データ付加¥n");
                                        continue ;
                                }
                        } else if ( (( stat & 0x02 ) != 0x02) && (( stat & 0x04 ) != 0x04 ) ) {
                                break;
                        }
                }

                /* QM リセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMA コントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("符号化終了¥n¥n") ;

        /* 符号データをファイルにセーブ */
        save_dat( fnew, 0L, 0x00070000L, CM, stripe_line );

        printf("符号データ・セーブ完了¥n");

        return;
}


/*****************************************************************************
*       符号化（TP あり）処理シーケンス
*
*       void enc_tp ( forg, fnew, filesize )
*               FILE *forg              :  画像データ・ファイルのポインタ
*               FILE *fnew              :  符号データ・ファイルのポインタ
*               unsigned long filesize  :  画像データ・ファイルのサイズ (byte)
*****************************************************************************/
void enc_tp( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char    stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l ;
        unsigned int     hsize, vline, strip ;
        int              stripe_line, stripe_amari ; /* ストライプ数 */
        int              flag1 = 0 ;                 /* 第１ストライプ処理を示すフラグ */
        int              i ;                         /* 汎用変数 */

        /* 主走査画素数、副走査ライン数の入力 */
        printf("主走査画素数は？   :");
```

```
scanf("%ud", &hsize );
if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
} else if ( hsize <= 0 ) {
        hsize = 1 ;
}

printf("副走査ライン数は？：");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？：");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始 ¥n");
load_dat ( forg, filesize, 0L, 1M );

/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0 ;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}

/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0 ;
while ( (stat & 0x40) != 0x40 ) {
```

```
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INTのクリア */
outp( d187a_ist, 0x00 );

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x07 );
outp( d187a_mdrh, 0x08 );

printf("TPパラメータ設定¥n");
outp( d187a_tpr, 0x06 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );

for ( i = stripe_line ; i > 0 ; i — ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        outp( d187a_atrl, 0x00 );
        outp( d187a_atrh, 0x00 );

        outp( d187a_hsb, 0x01 );

        /* 画像データDMA転送アドレス設定 */

        if ( flag1 == 0 ) {
        /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
                outp( d187a_dmsm, 0x00 );
                outp( d187a_dmsh, 0x00 );
                /* 終了アドレス */
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );

                /* ホスト・バス側外部DMAコントローラ起動 */
                printf("符号バス(ホスト・バス)側 DMA転送開始¥n¥n") ;
                dma_jtoc( (unsigned char)0, (unsigned long)0, (unsigned long)50000L );

                flag1 = 1 ;
        } else {
                        /* 開始アドレス */
                        stat = inp( d187a_dmel );
                        outp( d187a_dmsl, stat );
                        stat = inp( d187a_dmem );
                        outp( d187a_dmsm, stat );
                        stat = inp( d187a_dmeh );
                        outp( d187a_dmsh, stat );

                        /* 終了アドレス */
```

```
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );
}

/* μPD72187A 符号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT 発生¥n");

stat = 0 ;
while ( (stat & 0x01) != 0x01 ) {
        stat = inp( d187a_ist );
}
printf("STFEND 確認¥n¥n") ;

/* INT クリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0xDE;
outp( d187a_ist, stat );


/* マーカ・コード付加 */

/* ホスト・バスをマーカ・コードに設定 */
outp( d187a_hsb, 0x00 );

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0xFF );
printf(" MK (FFH) 付加¥n") ;

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0x02 );
printf(" MK (02H) 付加¥n") ;

/* ダミー・データ付加 */
stat = 0 ;
while ( 1 ) {
        stat = inp( d187a_strl );
        printf("strl = 0x%2X    ¥n", stat );

        /* ダミー・データ付加する必要あり？ */
        if ( (( stat & 0x02 ) == 0x02 ) && (( stat & 0x04 ) == 0x04 ) ) {
```

```
                        if ((stat & 0x10) == 0x10) {
                                outp( d187a_mkb, 0x00 );
                                printf("ダミー・データ付加¥n");
                                continue ;
                        }
                } else if ( (( stat & 0x02 ) != 0x02) && (( stat & 0x04 ) != 0x04 ) ) {
                        break;
                }
        }

        /* QM リセット実行 */
        outp( d187a_ctr, 0x07 );
    }

    /* DMA コントローラ・リセット */
    outp( b_dma_init, 0x03 ) ;

    printf("符号化終了¥n¥n") ;

    /* 符号データをファイルにセーブ */
    save_dat( fnew, 0L, 0x00070000L, CM, stripe_line );

    printf("符号データ・セーブ完了¥n");

    return;
}


/*************************************************************************
*       符号化 (AT あり) 処理シーケンス
*
*       void enc_at ( forg, fnew, filesize )
*               FILE *forg              :  画像データ・ファイルのポインタ
*               FILE *fnew              :  符号データ・ファイルのポインタ
*               unsigned long filesize  :  画像データ・ファイルのサイズ (byte)
*************************************************************************/
void enc_at( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
    unsigned char    stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l, at_position ;
    unsigned int     hsize, vline, strip, atpixel ;
    int              stripe_line, stripe_amari ; /* ストライプ数 */
    int              flag1 = 0 ;                 /* 第1ストライプ処理を示すフラグ */
    int              i ;                         /* 汎用変数 */

    /* 主走査画素数、副走査ライン数の入力 */
    printf("主走査画素数は？  :");
    scanf("%ud", &hsize );
   if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
    } else if ( hsize <= 0 ) {
        hsize = 1 ;
    }

    printf("副走査ライン数は？:");
```

```
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？：");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

printf("AT 画素位置は？　：");
scanf("%ud", &atpixel );
if ( atpixel > 127 || ( atpixel < 3 && atpixel != 0 ) ) {
        atpixel = 0 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

at_position = (unsigned char)( atpixel & 0x000000EF );
at_position |= 0x80 ;

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始 ¥n");
load_dat ( forg, filesize, 0L, IM );

/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0 ;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}

/* STRAM クリア */
printf("STRAM　0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0 ;
```

```
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}

/* STRAM  0クリア完了 INT のクリア */
outp( d187a_ist, 0x00 );

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x01 );
outp( d187a_mdrh, 0x08 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );

/* ストライプ処理のための for ループ */

for ( i = stripe_line ; i > 0 ; i -- ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        printf("AT パラメータ設定¥n");
        outp( d187a_atrl, 0x80 );
        outp( d187a_atll, 0xFF );
        outp( d187a_atlh, 0xFF );
        outp( d187a_atrh, at_position );

        outp( d187a_hsb, 0x01 );

        /* 画像データ DMA 転送アドレス設定 */

        if ( flag1 == 0 ) {
                /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
                outp( d187a_dmsm, 0x00 );
                outp( d187a_dmsh, 0x00 );
                /* 終了アドレス */
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );

                /* ホスト・バス側外部 DMA コントローラ起動 */
                printf("符号バス(ホスト・バス)側 DMA 転送開始¥n¥n") ;
                dma_jtoc( (unsigned char)0, (unsigned long)0, (unsigned long)50000L );

                flag1 = 1 ;
        } else {
                /* 開始アドレス */
                stat = inp( d187a_dmel );
                outp( d187a_dmsl, stat );
                stat = inp( d187a_dmem );
                outp( d187a_dmsm, stat );
```

```
        stat = inp( d187a_dmeh );
        outp( d187a_dmsh, stat );

        /* 終了アドレス */
        outp( d187a_dmel, 0xBF );
        outp( d187a_dmem, 0xD4 );
        outp( d187a_dmeh, 0x07 );
}

/* μPD72187A符号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT発生¥n");

stat = 0 ;
while ( (stat & 0x01) != 0x01 ) {
        stat = inp( d187a_ist );
}
printf("STFEND 確認¥n¥n") ;

/* INTクリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0xDE;
outp( d187a_ist, stat );


/* マーカ・コード付加 */

/* ホスト・バスをマーカ・コードに設定 */
outp( d187a_hsb, 0x00 );

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0xFF );
printf(" MK (FFH) 付加¥n") ;

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0x02 );
printf(" MK (02H) 付加¥n") ;

/* ダミー・データ付加 */
stat = 0 ;
while ( 1 ) {
        stat = inp( d187a_strl );
```

```
                        printf("strl = 0x%2X    ", stat );

                        /* ダミー・データ付加する必要あり？ */
                        if ( (( stat & 0x02 ) == 0x02 ) && (( stat & 0x04 ) == 0x04 ) ) {
                                if ((stat & 0x10) == 0x10) {
                                        outp( d187a_mkb, 0x00 );
                                        printf("ダミー・データ付加¥n");
                                        continue ;
                                }
                        } else if ( (( stat & 0x02 ) != 0x02) && (( stat & 0x04 ) != 0x04 ) ) {
                                break;
                        }
                }

                /* CM リセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMA コントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("符号化終了¥n¥n") ;

        /* 符号データをファイルにセーブ */
        save_dat( fnew, 0L, 0x00070000L, CM, stripe_line );

        printf("符号データ・セーブ完了¥n");

        return;
}


/*****************************************************************************
*       符号化（TP&AT あり）処理シーケンス
*
*       void enc_tpat ( forg, fnew, filesize )
*               FILE *forg              :  画像データ・ファイルのポインタ
*               FILE *fnew              :  符号データ・ファイルのポインタ
*               unsigned long filesize  :  画像データ・ファイルのサイズ (byte)
*****************************************************************************/
void enc_tpat( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char    stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l, at_position ;
        unsigned int     hsize, vline, strip, atpixel ;
        int              stripe_line, stripe_amari ; /* ストライプ数 */
        int              flag1 = 0 ;                 /* 第１ストライプ処理を示すフラグ */
        int              i ;                         /* 汎用変数 */

        /* 主走査画素数、副走査ライン数の入力 */
        printf("主走査画素数は？   :");
        scanf("%ud", &hsize );
       if ( hsize > 0xFFFF ) {
                hsize = 0xFFFF ;
        } else if ( hsize <= 0 ) {
```

```
        hsize = 1 ;
}

printf("副走査ライン数は？：");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？：");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

printf("AT 画素位置は？　：");
scanf("%ud", &atpixel );
if ( atpixel > 127 || ( atpixel < 3 && atpixel != 0 ) ) {
        atpixel = 0 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

at_position = (unsigned char)( atpixel & 0x000000EF );
at_position |= 0x80 ;

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始 ¥n");
load_dat ( forg, filesize, 0L, IM );

/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0 ;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}
```

```
/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0 ;
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INTのクリア */
outp( d187a_ist, 0x00 );

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x07 );
outp( d187a_mdrh, 0x08 );

printf("TPパラメータ設定¥n");
outp( d187a_tpr, 0x06 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );


/* ストライプ処理のための for ループ */

for ( i = stripe_line ; i > 0 ; i — ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        printf("ATパラメータ設定¥n");
        outp( d187a_atrl, 0x80 );
        outp( d187a_atll, 0xFF );
        outp( d187a_atlh, 0xFF );
        outp( d187a_atrh, at_position );

        outp( d187a_hsb, 0x01 );

        /* 画像データ DMA転送アドレス設定 */

        if ( flag1 == 0 ) {
        /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
                outp( d187a_dmsm, 0x00 );
                outp( d187a_dmsh, 0x00 );
                /* 終了アドレス */
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );

                /* ホスト・バス側外部 DMAコントローラ起動 */
                printf("符号バス(ホスト・バス)側 DMA転送開始¥n¥n") ;
                dma_jtoc( (unsigned char)0, (unsigned long)0, (unsigned long)50000L );
```

```
        flag1 = 1 ;
} else {
        /* 開始アドレス */
        stat = inp( d187a_dmel );
        outp( d187a_dmel, stat );
        stat = inp( d187a_dmem );
        outp( d187a_dmsm, stat );
        stat = inp( d187a_dmeh );
        outp( d187a_dmsh, stat );

        /* 終了アドレス */
        outp( d187a_dmel, 0xBF );
        outp( d187a_dmem, 0xD4 );
        outp( d187a_dmeh, 0x07 );
}

/* μPD72187A符号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT発生¥n");

stat = 0 ;
while ( (stat & 0x01) != 0x01 ) {
        stat = inp( d187a_ist );
}
printf("STFEND 確認¥n¥n") ;

/* INTクリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0xDE;
outp( d187a_ist, stat );


/* マーカ・コード付加 */

/* ホスト・バスをマーカ・コードに設定 */
outp( d187a_hsb, 0x00 );

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
outp( d187a_mkb, 0xFF );
printf(" MK (FFH) 付加¥n") ;

/* マーカ・コード書き込み確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
```

```
                outp( d187a_mkb, 0x02 );
                printf("MK (02H) 付加¥n") ;

                /* ダミー・データ付加 */
                stat = 0 ;
                while ( 1 ) {
                        stat = inp( d187a_strl );
                        printf("strl = 0x%2X    ", stat );

                        /* ダミー・データ付加する必要あり？ */
                        if ( (( stat & 0x02 ) == 0x02 ) && (( stat & 0x04 ) == 0x04 ) ) {
                                if ((stat & 0x10) == 0x10) {
                                        outp( d187a_mkb, 0x00 );
                                        printf("ダミー・データ付加¥n");
                                        continue ;
                                }
                        } else if ( (( stat & 0x02 ) != 0x02) && (( stat & 0x04 ) != 0x04 ) ) {
                                break;
                        }
                }

                /* QM リセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMA コントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("符号化終了¥n¥n") ;

        /* 符号データをファイルにセーブ */
        save_dat( fnew, 0L, 0x00070000L, CM, stripe_line );

        printf("符号データ・セーブ完了¥n");

        return;
}


/*****************************************************************************
*       復号化（通常）処理シーケンス
*
*       void dec_norm ( forg, fnew, filesize )
*               FILE *forg              :  符号データ・ファイルのポインタ
*               FILE *fnew              :  画像データ・ファイルのポインタ
*               unsigned long filesize  :  符号データ・ファイルのサイズ (byte)
*****************************************************************************/
void dec_norm( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char     stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l ;
        unsigned int      hsize, vline, strip ;
        int               stripe_line, stripe_amari ; /* ストライプ数 */
        int               flag1 = 0 ;                 /* 第１ストライプ処理を示すフラグ */
        int               i ;                         /* 汎用変数 */
```

```
/* 主走査画素数、副走査ライン数の入力 */
printf("主走査画素数は？　:");
scanf("%ud", &hsize );
if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
} else if ( hsize <= 0 ) {
        hsize = 1 ;
}

printf("副走査ライン数は？:");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？:");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}


/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始¥n");
load_dat ( forg, filesize, 0L, CM );


/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}
```

```
/* STRAM クリア */
 printf("STRAM  0クリア¥n");
 outp( d187a_cmd1, 0x08 );
 stat = 0;
 while ( (stat & 0x40) != 0x40 ) {
          stat = inp( d187a_ist );
 }
 /* STRAM  0クリア完了 INTのクリア */
 outp( d187a_ist, 0x00 );

 stat = inp( d187a_ist );
 printf("STAT = 0x%2X¥n¥n");

 /* モード設定 */
 printf("モード&パラメータ設定¥n");
 outp( d187a_mdrl, 0x00 );
 outp( d187a_mdrh, 0x08 );

 /* パラメータ設定 */
 outp( d187a_pell, hsize_l );
 outp( d187a_pelh, hsize_h );

 outp( d187a_stpl, strip_l );
 outp( d187a_stph, strip_h );

 /* ストライプ処理のための for ループ */

 for ( i = stripe_line ; i > 0 ; i — ) {

          outp( d187a_linl, vline_l );
          outp( d187a_linh, vline_h );

          outp( d187a_atrl, 0x00 );
          outp( d187a_atrh, 0x00 );

          outp( d187a_hsb, 0x01 );

          /* 画像データ DMA転送アドレス設定 */

          if ( flag1 == 0 ) {
          /* 開始アドレス */
                   outp( d187a_dmsl, 0x00 );
                   outp( d187a_dmsm, 0x00 );
                   outp( d187a_dmsh, 0x00 );
                   /* 終了アドレス */
                   outp( d187a_dmel, 0xBF );
                   outp( d187a_dmem, 0xD4 );
                   outp( d187a_dmeh, 0x07 );

                   /* ホスト・バス側外部 DMA コントローラ起動 */
                   printf("符号バス(ホスト・バス)側 DMA転送開始¥n¥n");
                   dma_jtoc( (unsigned char)1, (unsigned long)0, (unsigned long)50000L );

                   flag1 = 1 ;
          } else {
                   /* 開始アドレス */
                   stat = inp( d187a_dmel );
```

```
            outp( d187a_dmsl, stat );
            stat = inp( d187a_dmem );
            outp( d187a_dmsm, stat );
            stat = inp( d187a_dmeh );
            outp( d187a_dmsh, stat );

            /* 終了アドレス */
            outp( d187a_dmel, 0xBF );
            outp( d187a_dmem, 0xD4 );
            outp( d187a_dmeh, 0x07 );
}

/* μPD72187A復号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
            stat = inp( d187a_strh );
}
printf("INT発生    ") ;

stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
            stat = inp( d187a_ist );
}
printf("MKDTC 検出¥n¥n");

/* INTクリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0x7E;
outp( d187a_ist, stat );


/* マーカ・コード読み出し */

/* マーカ・コード読み出し確認 */
printf("マーカ・コード読み出し¥n");
                              stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
            stat = inp( d187a_strl );
}
stat = 0 ;
stat = inp( d187a_mkb );
printf("MKB = 0x%2X¥n", stat );

/* マーカ・コード読み出し確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
            stat = inp( d187a_strl );
}
stat = 0 ;
stat = inp( d187a_mkb );
printf("MKB = 0x%2X¥n¥n", stat );

/* 最終バイト処理 */
```

```
                printf("最終バイト処理開始¥n¥n") ;

                outp( d187a_cmd1, 0xC2 );
                stat = 0 ;
                while ( (stat & 0x80) != 0x80 ) {
                        stat = inp( d187a_strh );
                }

                printf("INT 発生    ") ;

                stat = 0 ;
                while ( (stat & 0x02) != 0x02 ) {
                        stat = inp( d187a_ist );
                }
                printf("PIXEND 検出¥n¥n") ;

                /* INT クリア */
                stat =  inp( d187a_strh ) & 0x7F;
                outp( d187a_strh, stat );
                stat =  inp( d187a_ist ) & 0xFD;
                outp( d187a_ist, stat );

                /* QM リセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMA コントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("復号化終了¥n¥n") ;

        /* 画像データをファイルに転送 */
        save_dat( fnew, 0L, 0x0007D4C0L, IM, 0 );

        printf("画像データ・セーブ完了¥n") ;

        return ;
}


/*****************************************************************************
*       復号化 (TP あり) 処理シーケンス
*
*       void dec_tp ( forg, fnew, filesize )
*               FILE *forg              :  符号データ・ファイルのポインタ
*               FILE *fnew              :  画像データ・ファイルのポインタ
*               unsigned long filesize  :  符号データ・ファイルのサイズ (byte)
*****************************************************************************/
void dec_tp( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char    stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l ;
        unsigned int     hsize, vline, strip ;
        int              stripe_line, stripe_amari ; /* ストライプ数 */
        int              flag1 = 0 ;                 /* 第1ストライプ処理を示すフラグ */
```

```
int             i ;                     /* 汎用変数 */

/* 主走査画素数、副走査ライン数の入力 */
printf("主走査画素数は？   :");
scanf("%ud", &hsize );
if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
} else if ( hsize <= 0 ) {
        hsize = 1 ;
}

printf("副走査ライン数は？ :");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？ :");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始¥n");
load_dat ( forg, filesize, 0L, CM );


/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}
```

```
/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0;
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INTのクリア */
outp( d187a_ist, 0x00 );

stat = inp( d187a_ist );
printf("STAT = 0x%2X¥n¥n");

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x06 );
outp( d187a_mdrh, 0x08 );

printf("TPパラメータ設定¥n");
outp( d187a_tpr, 0x06 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );

for ( i = stripe_line ; i > 0 ; i — ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        outp( d187a_atrl, 0x00 );
        outp( d187a_atrh, 0x00 );

        outp( d187a_hsb, 0x01 );

        /* 画像データ DMA転送アドレス設定 */

        if ( flag1 == 0 ) {
        /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
                outp( d187a_dmsm, 0x00 );
                outp( d187a_dmsh, 0x00 );
                /* 終了アドレス */
                outp( d187a_dmel, 0xBF );
                outp( d187a_dmem, 0xD4 );
                outp( d187a_dmeh, 0x07 );

                /* ホスト・バス側外部 DMAコントローラ起動 */
                printf("符号バス(ホスト・バス)側 DMA転送開始¥n¥n");
                dma_jtoc( (unsigned char)1, (unsigned long)0, (unsigned long)50000L );

                flag1 = 1 ;
        } else {
                        /* 開始アドレス */
```

```
                    stat = inp( d187a_dmel );
                    outp( d187a_dmel, stat );
                    stat = inp( d187a_dmem );
                    outp( d187a_dmem, stat );
                    stat = inp( d187a_dmeh );
                    outp( d187a_dmeh, stat );

                    /* 終了アドレス */
                    outp( d187a_dmel, 0xBF );
                    outp( d187a_dmem, 0xD4 );
                    outp( d187a_dmeh, 0x07 );
}

/* μPD72187A復号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT発生    ") ;

stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_ist );
}
printf("MKDTC 検出¥n¥n");

/* INTクリア */
stat =  inp( d187a_strh ) & 0x7F;
outp( d187a_strh, stat );
stat =  inp( d187a_ist ) & 0x7E;
outp( d187a_ist, stat );


/* マーカ・コード読み出し */

/* マーカ・コード読み出し確認 */
printf("マーカ・コード読み出し¥n");
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
                stat = inp( d187a_strl );
}
stat = 0 ;
stat = inp( d187a_mkb );
printf("MKB = 0x%2X¥n", stat );

/* マーカ・コード読み出し確認 */
stat = 0 ;
while ( (stat & 0x10) != 0x10 ) {
        stat = inp( d187a_strl );
}
stat = 0 ;
stat = inp( d187a_mkb );
printf("MKB = 0x%2X¥n¥n", stat );
```

```
                /* 最終バイト処理 */
                printf("最終バイト処理開始¥n¥n") ;

                outp( d187a_cmd1, 0xC2 );
                stat = 0 ;
                while ( (stat & 0x80) != 0x80 ) {
                        stat = inp( d187a_strh );
                }

                printf("INT発生    ") ;

                stat = 0 ;
                while ( (stat & 0x02) != 0x02 ) {
                        stat = inp( d187a_ist );
                }
                printf("PIXEND 検出¥n¥n") ;

                /* INTクリア */
                stat =  inp( d187a_strh ) & 0x7F;
                outp( d187a_strh, stat );
                stat =  inp( d187a_ist ) & 0xFD;
                outp( d187a_ist, stat );

                /* QMリセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMAコントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("復号化終了¥n¥n") ;

        /* 画像データをファイルに転送 */
        save_dat( fnew, 0L, 0x0007D4C0L, IM, 0 );

        printf("画像データ・セーブ完了¥n") ;

        return ;
}


/*****************************************************************************
*       復号化（ATあり）処理シーケンス
*
*       void dec_at ( forg, fnew, filesize )
*               FILE *forg              :  符号データ・ファイルのポインタ
*               FILE *fnew              :  画像データ・ファイルのポインタ
*               unsigned long filesize  :  符号データ・ファイルのサイズ (byte)
*****************************************************************************/
void dec_at( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char     stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l, at_position ;
        unsigned int      hsize, vline, strip, atpixel ;
```

```
int             stripe_line, stripe_amari ; /* ストライプ数 */
int             flag1 = 0 ;               /* 第1ストライプ処理を示すフラグ */
int             i ;                       /* 汎用変数 */

/* 主走査画素数、副走査ライン数の入力 */
printf("主走査画素数は？　:");
scanf("%ud", &hsize );
if ( hsize > 0xFFFF ) {
        hsize = 0xFFFF ;
} else if ( hsize <= 0 ) {
        hsize = 1 ;
}

printf("副走査ライン数は？:");
scanf("%ud", &vline );
if ( vline > 0x1FFF ) {
        vline = 0x1FFF ;
} else if ( vline <= 0 ) {
        vline = 1 ;
}

printf("ストライプ・ライン数は？:");
scanf("%ud", &strip );
if ( strip > vline ) {
        strip = vline ;
} else if ( strip <= 0 ) {
        strip = 128 ;
}

printf("AT画素位置は？　:");
scanf("%ud", &atpixel );
if ( atpixel > 127 || ( atpixel < 3 && atpixel != 0 ) ) {
        atpixel = 0 ;
}

hsize_l = (unsigned char)( hsize & 0x000000FF );
hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
vline_l = (unsigned char)( vline & 0x000000FF );
vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
strip_l = (unsigned char)( strip & 0x000000FF );
strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

at_position = (unsigned char)( atpixel & 0x000000EF );
at_position |= 0x80 ;

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始¥n");
```

```
load_dat ( forg, filesize, 0L, CM );


/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}

/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0;
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INT のクリア */
outp( d187a_ist, 0x00 );

stat = inp( d187a_ist );
printf("STAT = 0x%2X¥n¥n");

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x00 );
outp( d187a_mdrh, 0x08 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );

/* ストライプ処理のための for ループ */

for ( i = stripe_line ; i > 0 ; i — ) {

        outp( d187a_linl, vline_l );
        outp( d187a_linh, vline_h );

        printf("AT パラメータ設定¥n");
        outp( d187a_atrl, 0x80 );
        outp( d187a_atll, 0xFF );
        outp( d187a_atlh, 0xFF );
        outp( d187a_atrh, at_position );

        outp( d187a_hsb, 0x01 );


        /* 画像データ DMA 転送アドレス設定 */

        if ( flag1 = 0 ) {
        /* 開始アドレス */
                outp( d187a_dmsl, 0x00 );
```

```
            outp( d187a_dmsm, 0x00 );
            outp( d187a_dmsh, 0x00 );
            /* 終了アドレス */
            outp( d187a_dmel, 0xBF );
            outp( d187a_dmem, 0xD4 );
            outp( d187a_dmeh, 0x07 );

            /* ホスト・バス側外部 DMA コントローラ起動 */
            printf("符号バス(ホスト・バス)側 DMA 転送開始¥n¥n");
            dma_jtoc( (unsigned char)1, (unsigned long)0, (unsigned long)50000L );

            flag1 = 1 ;
    } else {
            /* 開始アドレス */
            stat = inp( d187a_dmel );
            outp( d187a_dmsl, stat );
            stat = inp( d187a_dmem );
            outp( d187a_dmsm, stat );
            stat = inp( d187a_dmeh );
            outp( d187a_dmsh, stat );

            /* 終了アドレス */
            outp( d187a_dmel, 0xBF );
            outp( d187a_dmem, 0xD4 );
            outp( d187a_dmeh, 0x07 );
    }

    /* μPD72187A 復号化開始コマンド発行 */
    outp( d187a_cmd1, 0xC1 );

    /* レスポンス待ち */
    stat = 0 ;
    while ( (stat & 0x80) != 0x80 ) {
            stat = inp( d187a_strh );
    }
    printf("INT 発生    ") ;

    stat = 0 ;
    while ( (stat & 0x80) != 0x80 ) {
            stat = inp( d187a_ist );
    }
    printf("MKDTC 検出¥n¥n");

    /* INT クリア */
    stat =  inp( d187a_strh ) & 0x7F;
    outp( d187a_strh, stat );
    stat =  inp( d187a_ist ) & 0x7E;
    outp( d187a_ist, stat );


    /* マーカ・コード読み出し */

    /* マーカ・コード読み出し確認 */
    printf("マーカ・コード読み出し¥n");
    stat = 0 ;
    while ( (stat & 0x10) != 0x10 ) {
            stat = inp( d187a_strl );
```

```
                }
                stat = 0 ;
                stat = inp( d187a_mkb );
                printf("MKB = 0x%2X¥n", stat );

                /* マーカ・コード読み出し確認 */
                stat = 0 ;
                while ( (stat & 0x10) != 0x10 ) {
                        stat = inp( d187a_strl );
                }
                stat = 0 ;
                stat = inp( d187a_mkb );
                printf("MKB = 0x%2X¥n¥n", stat );

                /* 最終バイト処理 */
                printf("最終バイト処理開始¥n¥n") ;

                outp( d187a_cmd1, 0xC2 );
                stat = 0 ;
                while ( (stat & 0x80) != 0x80 ) {
                        stat = inp( d187a_strh );
                }

                printf("INT 発生    ") ;

                stat = 0 ;
                while ( (stat & 0x02) != 0x02 ) {
                        stat = inp( d187a_ist );
                }
                printf("PIXEND 検出¥n¥n") ;

                /* INT クリア */
                stat =  inp( d187a_strh ) & 0x7F;
                outp( d187a_strh, stat );
                stat =  inp( d187a_ist ) & 0xFD;
                outp( d187a_ist, stat );

                /* QM リセット実行 */
                outp( d187a_ctr, 0x07 );
        }

        /* DMA コントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("復号化終了¥n¥n") ;

        /* 画像データをファイルに転送 */
        save_dat( fnew, 0L, 0x0007D4C0L, IM, 0 );

        printf("画像データ・セーブ完了¥n") ;

        return ;
}

/*****************************************************************************
```

```
*       復号化（TP&AT あり）処理シーケンス
*
*       void dec_tpat ( forg, fnew, filesize )
*               FILE *forg              :  符号データ・ファイルのポインタ
*               FILE *fnew              :  画像データ・ファイルのポインタ
*               unsigned long filesize  :  符号データ・ファイルのサイズ (byte)
*************************************************************************/
void dec_tpat( forg, fnew, filesize )
FILE    *forg ;
FILE    *fnew ;
unsigned long    filesize ;
{
        unsigned char    stat, hsize_h, hsize_l, vline_h, vline_l, strip_h, strip_l, at_position ;
        unsigned int     hsize, vline, strip, atpixel ;
        int              stripe_line, stripe_amari ; /* ストライプ数 */
        int              flag1 = 0 ;                 /* 第１ストライプ処理を示すフラグ */
        int              i ;                         /* 汎用変数 */

        /* 主走査画素数、副走査ライン数の入力 */
        printf("主走査画素数は？  :");
        scanf("%ud", &hsize );
        if ( hsize > 0xFFFF ) {
                hsize = 0xFFFF ;
        } else if ( hsize <= 0 ) {
                hsize = 1 ;
        }

        printf("副走査ライン数は？:");
        scanf("%ud", &vline );
        if ( vline > 0x1FFF ) {
                vline = 0x1FFF ;
        } else if ( vline <= 0 ) {
                vline = 1 ;
        }

        printf("ストライプ・ライン数は？:");
        scanf("%ud", &strip );
        if ( strip > vline ) {
                strip = vline ;
        } else if ( strip <= 0 ) {
                strip = 128 ;
        }

        printf("AT 画素位置は？  :");
        scanf("%ud", &atpixel );
        if ( atpixel > 127 || ( atpixel < 3 && atpixel != 0 ) ) {
                atpixel = 0 ;
        }

        hsize_l = (unsigned char)( hsize & 0x000000FF );
        hsize_h = (unsigned char)( (hsize >> 8) & 0x000000FF );
        vline_l = (unsigned char)( vline & 0x000000FF );
        vline_h = (unsigned char)( (vline >> 8) & 0x0000001F );
        strip_l = (unsigned char)( strip & 0x000000FF );
        strip_h = (unsigned char)( (strip >> 8) & 0x0000001F );

        at_position = (unsigned char)( atpixel & 0x000000EF );
```

```
at_position |= 0x80 ;

stripe_line = vline / strip ;
stripe_amari = vline % strip ;

if ( stripe_amari > 0 ) {
        stripe_line ++ ;
}

/* ボードの初期化 */
sysinit();

/* ボード上にファイルを転送 */
printf("ロード開始¥n");
load_dat ( forg, filesize, 0L, CM );


/* ソフトウェア・リセット */
printf("ソフトウェア・リセット¥n");
outp( d187a_ctr, 0x0E );
stat = 0;
while ( (stat & 0x07) != 0x07 ) {
        stat = inp( d187a_ctr );
}

/* STRAM クリア */
printf("STRAM  0クリア¥n");
outp( d187a_cmd1, 0x08 );
stat = 0;
while ( (stat & 0x40) != 0x40 ) {
        stat = inp( d187a_ist );
}
/* STRAM  0クリア完了 INT のクリア */
outp( d187a_ist, 0x00 );

stat = inp( d187a_ist );
printf("STAT = 0x%2X¥n¥n");

/* モード設定 */
printf("モード&パラメータ設定¥n");
outp( d187a_mdrl, 0x06 );
outp( d187a_mdrh, 0x08 );

printf("TP パラメータ設定¥n");
outp( d187a_tpr, 0x06 );

/* パラメータ設定 */
outp( d187a_pell, hsize_l );
outp( d187a_pelh, hsize_h );

outp( d187a_stpl, strip_l );
outp( d187a_stph, strip_h );


/* ストライプ処理のための for ループ */

for ( i = stripe_line ; i > 0 ; i — ) {
```

```
outp( d187a_linl, vline_l );
outp( d187a_linh, vline_h );

printf("ATパラメータ設定¥n");
outp( d187a_atrl, 0x80 );
outp( d187a_atll, 0xFF );
outp( d187a_atlh, 0xFF );
outp( d187a_atrh, at_position );

outp( d187a_hsb, 0x01 );

/* 画像データDMA転送アドレス設定 */

if ( flag1 == 0 ) {
        /* 開始アドレス */
        outp( d187a_dmsl, 0x00 );
        outp( d187a_dmsm, 0x00 );
        outp( d187a_dmsh, 0x00 );
        /* 終了アドレス */
        outp( d187a_dmel, 0xBF );
        outp( d187a_dmem, 0xD4 );
        outp( d187a_dmeh, 0x07 );

        /* ホスト・バス側外部DMAコントローラ起動 */
        printf("符号バス(ホスト・バス)側 DMA転送開始¥n¥n");
        dma_jtoc( (unsigned char)1, (unsigned long)0, (unsigned long)50000L );

        flag1 = 1 ;
} else {
        /* 開始アドレス */
        stat = inp( d187a_dmel );
        outp( d187a_dmsl, stat );
        stat = inp( d187a_dmem );
        outp( d187a_dmsm, stat );
        stat = inp( d187a_dmeh );
        outp( d187a_dmsh, stat );

        /* 終了アドレス */
        outp( d187a_dmel, 0xBF );
        outp( d187a_dmem, 0xD4 );
        outp( d187a_dmeh, 0x07 );
}

/* μPD72187A復号化開始コマンド発行 */
outp( d187a_cmd1, 0xC1 );

/* レスポンス待ち */
stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_strh );
}
printf("INT発生    ") ;

stat = 0 ;
while ( (stat & 0x80) != 0x80 ) {
        stat = inp( d187a_ist );
```

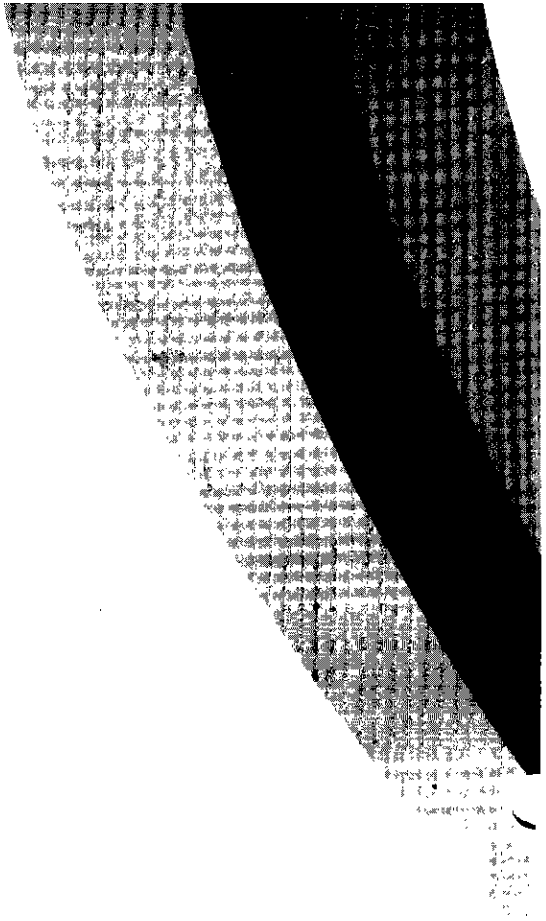
```
        }
        printf("MKDTC 検出¥n¥n");

        /* INTクリア */
        stat =  inp( d187a_strh ) & 0x7F;
        outp( d187a_strh, stat );
        stat =  inp( d187a_ist ) & 0x7E;
        outp( d187a_ist, stat );


        /* マーカ・コード読み出し */

        /* マーカ・コード読み出し確認 */
        printf("マーカ・コード読み出し¥n");
        stat = 0 ;
        while ( (stat & 0x10) != 0x10 ) {
                stat = inp( d187a_strl );
        }
        stat = 0 ;
        stat = inp( d187a_mkb );
        printf("MKB = 0x%2X¥n", stat );

        /* マーカ・コード読み出し確認 */
        stat = 0 ;
        while ( (stat & 0x10) != 0x10 ) {
                stat = inp( d187a_strl );
        }
        stat = 0 ;
        stat = inp( d187a_mkb );
        printf("MKB = 0x%2X¥n¥n", stat );

        /* 最終バイト処理 */
        printf("最終バイト処理開始¥n¥n") ;

        outp( d187a_cmd1, 0xC2 );
        stat = 0 ;
        while ( (stat & 0x80) != 0x80 ) {
                stat = inp( d187a_strh );
        }

        printf("INT発生    ") ;

        stat = 0 ;
        while ( (stat & 0x02) != 0x02 ) {
                stat = inp( d187a_ist );
        }
        printf("PIXEND 検出¥n¥n") ;

        /* INTクリア */
        stat =  inp( d187a_strh ) & 0x7F;
        outp( d187a_strh, stat );
        stat =  inp( d187a_ist ) & 0xFD;
        outp( d187a_ist, stat );

        /* QM リセット実行 */
        outp( d187a_ctr, 0x07 );
}
```

```
        /* DMAコントローラ・リセット */
        outp( b_dma_init, 0x03 ) ;

        printf("復号化終了¥n¥n") ;

        /* 画像データをファイルに転送 */
        save_dat( fnew, 0L, 0x0007D4C0L, IM, 0 );

        printf("画像データ・セーブ完了¥n") ;

        return ;
}


/*****************************************************************************
*
*       ボード上のシステム初期化
*
*****************************************************************************/
void sysinit()
{
        outp( b_reset_on, 1 );      /* ボード・リセット・ON */
        outp( b_reset_off, 1 );     /* ボード・リセット・OFF */
        outp( b_sysreg_cmd, 0x89 ); /* */
        outp( b_sysreg_pt0, 0xFB ); /* 符号／画像バス幅設定 */
        outp( b_sysreg_pt1, 0xFF ); /* 転送モード設定 */
        outp( b_dma_init, 0x03 );   /* DMAC 初期化 */

        outp( b_tmcnt_ct1, 0x34 );  /* カウンタ#0,バイナリ・カウント,モード4,インクリメントR/W*/
        outp( b_tmcnt_ct0, 0xFF );  /* カウンタ#0 クリア */
        outp( b_tmcnt_ct0, 0xFF );

        outp( b_tmcnt_ct1, 0x74 );  /* カウンタ#1,バイナリ・カウント,モード4,インクリメントR/W*/
        outp( b_tmcnt_ct1, 0xFF );  /* カウンタ#1 クリア */
        outp( b_tmcnt_ct1, 0xFF );

        outp( b_tmcnt_ct1, 0xB4 );  /* カウンタ#2,バイナリ・カウント,モード4,インクリメントR/W*/
        outp( b_tmcnt_ct2, 0xFF );  /* カウンタ#2 クリア */
        outp( b_tmcnt_ct2, 0xFF );
}


/*****************************************************************************
*       ボード上のメモリにデータ転送
*
*       void load_dat ( forg, filesize, maddr, mem )
*               FILE *forg              : 転送するファイルのポインタ
*               unsigned long  filesize: 転送するファイルのサイズ (byte)
*               unsigned long  maddr    : 転送先のメモリアドレス
*               char  mem               : 転送先メモリ選択
*                                         (CM :符号メモリ  IM :画像メモリ)
*****************************************************************************/
void load_dat( forg, filesize, maddr, mem )
FILE     *forg ;
unsigned long     filesize ;
```

```
unsigned long     maddr ;
char     mem ;
{
        unsigned char buffer[FTMS_MAX] ;
        unsigned char far *buf ;
        unsigned long ftms ;

        ftms = (unsigned long)(FTMS_MAX / 2) ;
        ftms &= 0xFFFFFFFE ;

        buf = buffer ;

        if ( mem == CM ) {

                /* 符号メモリへの転送 */
                if (( maddr < 0 ) || ( maddr > 0x7FFFF )) {
                        maddr = 0 ;
                }
                if ( filesize > 0x7FFFF ) {
                        filesize = 0x7FFFF ;
                }
                for ( ; filesize >= FTMS_MAX ; filesize -= FTMS_MAX , maddr += FTMS_MAX ) {
                        fread( buffer, FTMS_MAX, 1, forg ) ;
                        dma_htoc( (unsigned char)0, buf, maddr, ftms ) ;
                }
                if ( filesize != 0 ) {
                        fread( buffer, (int)filesize, 1, forg );
                        dma_htoc( (unsigned char)0, buf, maddr, filesize ) ;
                }
        } else if ( mem == IM ) {

                /* 画像メモリへの転送 */
                if (( maddr < 0 ) || ( maddr > 0x1FFFFF )) {
                        maddr = 0 ;
                }
                if ( filesize > 0x1FFFFF ) {
                        filesize = 0x1FFFFF ;
                }
                for ( ; filesize >= FTMS_MAX ; filesize -= FTMS_MAX , maddr += FTMS_MAX ) {
                        fread( buffer, FTMS_MAX, 1, forg ) ;
                        dma_htoi( (unsigned char)0, buf, maddr, ftms ) ;
                }
                if ( filesize != 0 ) {
                        fread( buffer, (int)filesize, 1, forg ) ;
                        dma_htoi( (unsigned char)0, buf, maddr, filesize ) ;
                }
        }
}

/*****************************************************************************
*       ボード上のメモリからデータ転送
*
*       void save_dat ( fnew, msaddr, meaddr, mem )
*               FILE  *fnew             : 転送先ファイルのポインタ
*               unsigned long  msaddr   : 転送元の転送開始アドレス
*               unsigned long  meaddr   : 転送元の転送終了アドレス
*               char  mem               : 転送先メモリ選択
```

```
*                                        (CM :符号メモリ  IM :画像メモリ)
*               int   stripe            : ストライプ数 (符号化時のみ使用)
*****************************************************************************/
void save_dat( fnew, msaddr, meaddr, mem, stripe )
FILE     *fnew ;
unsigned long     msaddr ;
unsigned long     meaddr ;
char     mem ;
int      stripe ;
{
        unsigned char     buffer[FTMS_MAX] ;
        unsigned char far *buf ;
        unsigned long     baddr, bmemadr ;
        unsigned long     ftms ;
        int      i, go_amari = 0, next_check = 0 ;

        ftms = (unsigned long)(FTMS_MAX / 2) ;
        ftms &= 0xFFFFFFFE ;

        bmemadr = 0L ;

        buf = buffer ;

        if ( mem == CM ) {

                /* 符号メモリからの転送 */
                if (( msaddr <= 0 ) || ( msaddr >= 0x7FFFF )) {
                        msaddr = 0 ;
                }
                if (( meaddr <= 0 ) || ( meaddr >= 0x7FFFF )) {
                        meaddr = 0x7FFFF ;
                }
                baddr = meaddr - msaddr ;
                if ( baddr <= 0 ) {
                        baddr = 1 ;
                }
                for ( ; baddr >= FTMS_MAX ; baddr -= FTMS_MAX , bmemadr += FTMS_MAX ) {
                        dma_htoc( (unsigned char)1, buf, bmemadr, ftms ) ;
                        for ( i = 0 ; i < FTMS_MAX ; i ++ ) {
                                if ( next_check == 1 && buffer[0] != 0x00 ) {
                                        stripe -- ;
                                        if ( stripe <= 0 ) {
                                                baddr = 1 ;
                                                go_amari = 1 ;
                                                break ;
                                        } else {
                                                next_check = 0 ;
                                                continue ;
                                        }
                                } else {
                                        next_check = 0 ;
                                }
                                if (buffer[i]==0xFF && i<(FTMS_MAX-1) && buffer[i+1]!=0x00) {
                                        stripe -- ;
                                        if ( stripe <= 0 ) {
                                        baddr = i + 2 ;
                                                go_amari = 1 ;
```

```
                                        break ;
                                } else {
                                        continue ;
                                }
                        }else if ( buffer[i] == 0xFF && i == (FTMS_MAX - 1) ) {
                                next_check = 1 ;
                                break ;
                        }
                }
                if ( go_amari == 1 ) {
                        break ;
                } else {
                        fwrite( buffer, FTMS_MAX, 1, fnew ) ;
                }
        }
        if ( baddr != 0 ) {
                dma_htoc( (unsigned char)1, buf, bmemadr, baddr ) ;
                fwrite( buffer, (int)baddr, 1, fnew ) ;
        }
} else if ( mem == IM ) {

        /* 画像メモリからの転送*/
        if (( msaddr < 0 ) || ( msaddr > 0x1FFFFF )) {
                msaddr = 0 ;
        }
        if (( meaddr < 0 ) || ( meaddr > 0x1FFFFF )) {
                meaddr = 0x1FFFFF ;
        }
        baddr = meaddr - msaddr ;
        if ( baddr < 0 ) {
                baddr = 1 ;
        }
        for (; baddr >= FTMS_MAX ; baddr -= FTMS_MAX , bmemadr += FTMS_MAX ) {
                dma_htoi( (unsigned char)1, buf, bmemadr, ftms ) ;
                fwrite( buffer, FTMS_MAX, 1, fnew ) ;
        }
        if ( baddr != 0 ) {
                dma_htoi( (unsigned char)1, buf, bmemadr, baddr ) ;
                fwrite( buffer, (int)baddr, 1, fnew ) ;
        }
}
}
```



—— お問い合わせは，最寄りのNECへ ——

【営業関係お問い合わせ先】

半導体第一販売事業部 半導体第二販売事業部 半導体第三販売事業部		〒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 (大代表)
中部支社	半導体第一販売部 半導体第二販売部	〒460 名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋 名古屋	(052)222-2170 (052)222-2190
関西支社	半導体第一販売部 半導体第二販売部 半導体第三販売部	〒540 大阪市中央区城見一丁目4番24号（NEC関西ビル）	大阪 大阪 大阪	(06) 945-3178 (06) 945-3200 (06) 945-3208

北海道支社	札幌 (011)231-0161	太田支店	太田 (0276)46-4011	富山支店	富山 (0764)31-8461
東北支社	仙台 (022)267-8740	宇都宮支店	宇都宮 (028)621-2281	三重支店	津 (0592)25-7341
岩手支店	盛岡 (019)651-4344	小山支店	小山 (0285)24-5011	京都支社	京都 (075)344-7824
山形支店	山形 (0236)23-5511	長野支社	松本 (0263)35-1662	神戸支社	神戸 (078)333-3654
郡山支店	郡山 (0249)23-5511	甲府支店	甲府 (0552)24-4141	中国支社	広島 (082)242-5504
いわき支店	いわき (0246)21-5511	埼玉支社	大宮 (048)641-1411	鳥取支店	鳥取 (0857)27-5311
長岡支店	長岡 (0258)36-2155	立川支社	立川 (0425)26-5981	岡山支店	岡山 (086)225-4455
土浦支店	土浦 (0298)23-6161	千葉支社	千葉 (043)238-8116	四国支社	高松 (0878)36-1200
水戸支店	水戸 (029)226-1717	静岡支社	静岡 (054)255-2211	新居浜支店	新居浜 (0897)32-5001
神奈川支社	横浜 (045)324-5524	北陸支社	金沢 (0762)23-1621	松山支店	松山 (089)945-4149
群馬支店	高崎 (0273)26-1255	福井支店	福井 (0776)22-1866	九州支社	福岡 (092)271-7700

【本資料に関する技術お問い合わせ先】

半導体ソリューション技術本部 システムマイクロ技術部	〒210 川崎市幸区堀越三丁目484番地	川崎	(044)548-8891	半導体 インフォメーションセンター FAX(044)548-7900 (FAXにてお願い致します)
半導体販売技術本部 東日本販売技術部	〒108-01 東京都港区芝五丁目7番1号（NEC本社ビル）	東京	(03)3798-9619	
半導体販売技術本部 中部販売技術部	〒460 名古屋市中区錦一丁目17番1号（NEC中部ビル）	名古屋	(052)222-2125	
半導体販売技術本部 西日本販売技術部	〒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	

C97.1