Internal Use Only
非公開

TR－A－0057　　24

聴覚実験用信号入出力システム

－MASSCOMP 周辺機器の構成と制御ルーチン－

平原達也　天白成一

1989. 7. 25

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷　☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

目　次

第１章　はじめに

聴覚研究室では、種々の音刺激を用いた聴覚心理物理実験を行なったり、受聴試験の刺激系列を作成するシステムとして、かねてより、マスコンプ社（現在はコンカレント社）のワークステイションＭＣ５６００を中心としたシステムを構築し、実用に供してきた。平成元年４月の本研究所への移転を機に、従来のシステムの不備な点を変更し、性能および操作性のより優れたシステムを構築した。

主な変更点は、以下の４点である。

①ＤＡ変換器をＭＡＳＳＣＯＭＰ社製のＤＡ０４ＨからＰＡＶＥＣ社製のＭＤ８３００シリーズに変更した。

②ＡＤ／ＤＡコンバータとＭＡＳＳＣＯＭＰとの接続を光ケーブルを用いたものにした。

③ＭＡＳＳＣＯＭＰ自身を計算機室内に設置した。

④ＡＤ／ＤＡコンバータと各種オーディオ機器との接続を統一した。

①と②は，計算機とＡＤ／ＤＡコンバータを光でアイソレートすることにより、機器間に生じていた直流オフセットを減じるとともにＡＤ／ＤＡ変換時のＳ／Ｎ比の向上を図ったものである。この光リンクを採用することにより、③が可能となった。その結果、計算機に対する設置環境（温度や埃）が向上したとともに、グラフィックターミナルに向かうユーザーの作業環境（騒音）も向上した。また、④は、４台あるＭＡＳＳＣＯＭＰのどれを用いても、機器間の接続ケーブルを変更することなしに、同一操作でＡＤ／ＤＡとＤＡＴの録音再生がおこなえるようになり、様々なレベルのユーザを抱える本研究室における利便の向上が図られた。

本稿では、現時点における聴覚研究室の実験システムのハードウエア構成とソフトウエア構成について述べる。第２章では、まず、聴覚研究室のＭＣ５６００のシステム構成について述べる。次に、今回導入した、ＰＡＶＥＣ社のＭＤ８３００シリーズ光リンクＡＤ／ＤＡコンバータシステムのシステム構成について述べる。そして、ＤＡ変換器出力及びＡＤ変換器入力に接続する各種オーディオ機器について述べる。第３章では、ＭＣ５６００上に作成した、ＡＤ／ＤＡ変換器制御ルーチン、パラレルＩ／Ｏを用いた被験者応答入力ルーチン、ＧＰＩＢを用いたアッテネータ制御ルーチンについて述べる。

第２章　ＭＡＳＣＯＭＰを中心とした聴覚実験システム

本研究室では、主にマスコンプ社のワークステイション（ＭＣ５６００）４台を聴覚心理物理実験や受聴試験の実験や刺激音系列の作成に用いている。本章ではこのＭＣ５６００を中心としたシステムについて述べる。

２．１　ＭＡＳＳＣＯＭＰ　ＭＣ５６００の構成

ＭＣ５６００はＲＴＵ（Real Time UNIX）をＯＳとし、大容量のメモリ（16MB）と高速なハードディスク（378MB or 568MB）、ヴェクタアクセラレータ（Ｖ.Ａ）や浮動小数点アクセラレータ（ＦＰＡ）、グラフィックシステム、それに、ＤＡＣＰと呼ばれるデータ収集・制御インターフェイスを備えており、各種の聴覚実験の準備、遂行、データ解析、等に適しているワークステイションといえる。特に，ＡＴＲのようにほとんどの計算機がネットワークで結ばれ、ＵＮＩＸをＯＳとして動作しているような環境下では、データやプログラムの融通性が高いことが大きなメリットとなっている。

しかし、実験制御などを行なう際の実時間性に関しては、ＲＴＵといえどもＵＮＩＸであるがゆえに様々なプロセスの割込み（インタラプト）が入り、真の実時間処理ができない場合もある。例えば、数ミリ秒の精度を要求する反応時間の計測をディジタルＩ／Ｏを用いて実行しようとしても単純にはできない。

図１に聴覚研究室の計算機とネットワーク環境を示す。聴覚実験用ワークステイションとしてのＭＡＳＳＣＯＭＰはhm01，hm02，hm03，hm08の４台であり、現在のところＧＦの実験室に２台（hm01とhm08）、１Ｆの実験室に２台（hm02とhm03）を設置している。図２と表１に各ＭＡＳＳＣＯＭＰの諸元と周辺機器等の設置状況を示す。

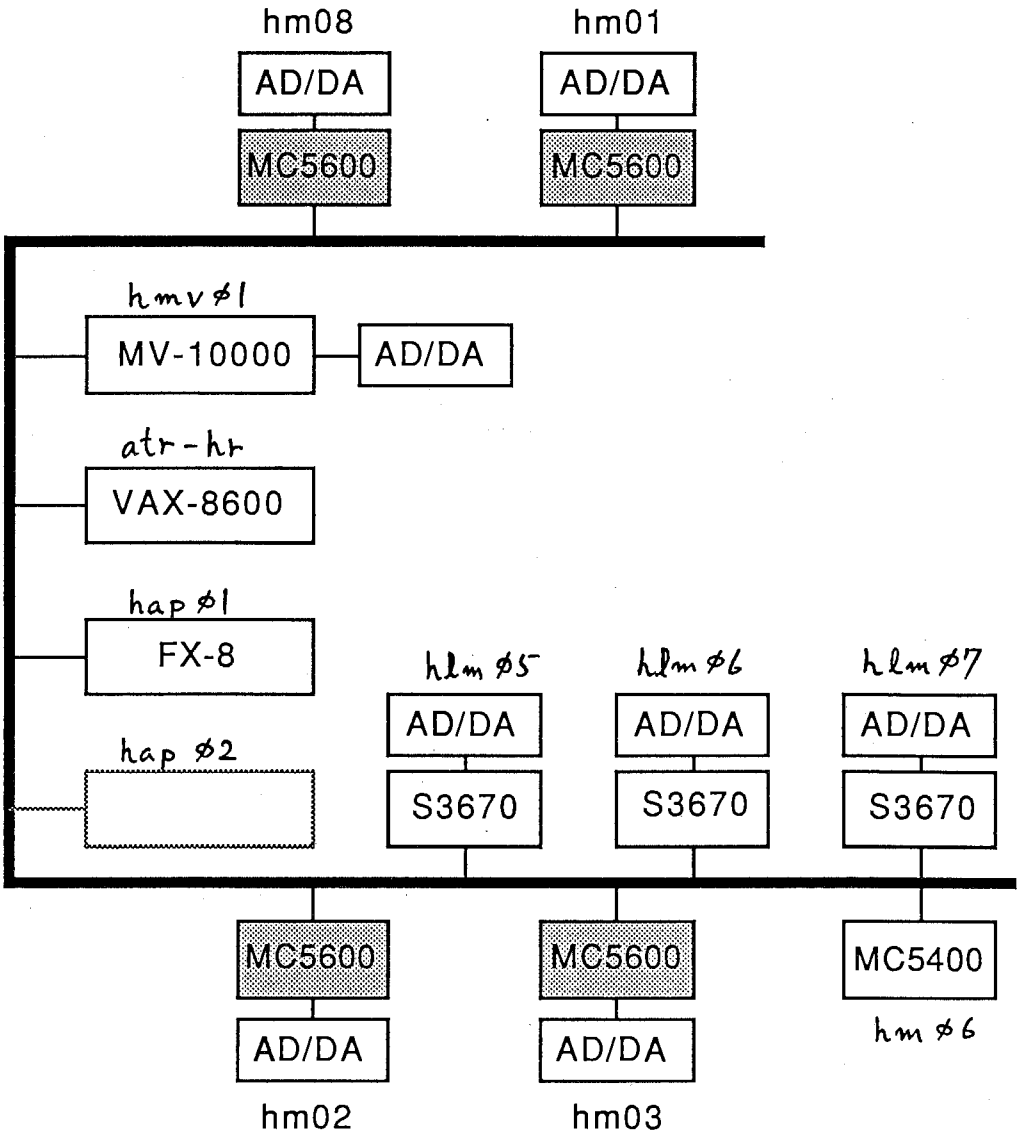


図１　聴覚研究室の計算機とネットワーク環境

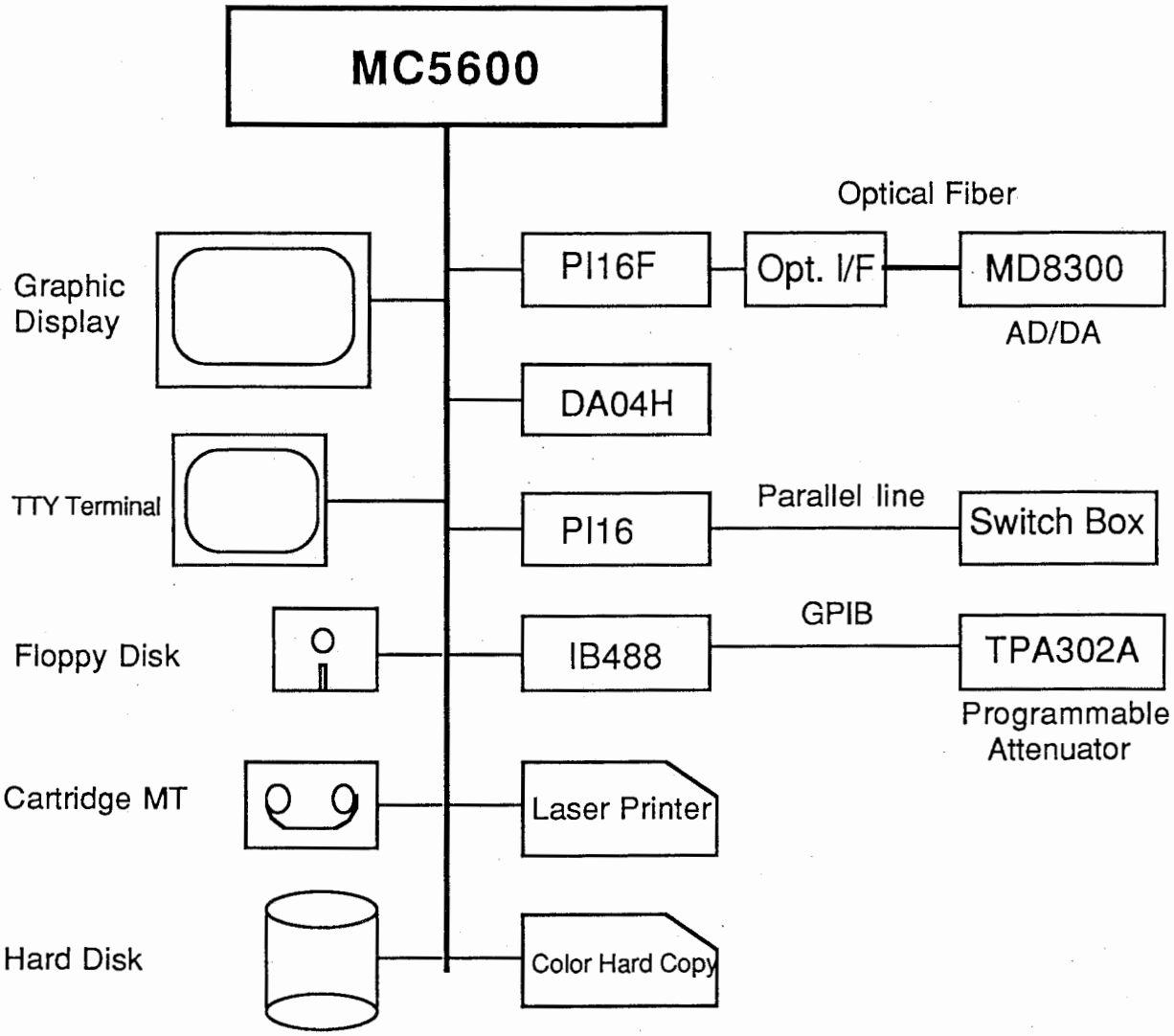


図２　ＭＡＳＳＣＯＭＰの構成

表１　各ＭＡＳＳＣＯＭＰの諸元等

項　目	hm01	hm02	hm03	hm08
ＣＰＵ	68030	68030	68030	68030
ＦＰＡ	FPA-1	FPA-1	FPA-1	FPA-1
ＶＡ	VA-1	VA-1	VA-1	VA-1
ﾒｲﾝﾒﾓﾘ	16MB	16MB	16MB	16MB(1)
ﾊｰﾄﾞﾃﾞｨｽｸ	378MB	378MB	378MB	568MB
ｸﾞﾗﾌｨｯｸﾀｰﾐﾅﾙ	12 Planes	12 Planes	12 Planes	12 Planes
ＴＴＹﾀｰﾐﾅﾙ	○	○(2)	○(2)	○
ＭＴ	○	×	×	×
ｶｰﾄﾘｯｼﾞＭＴ	×	○	○	○
DA04H	○	○	○	○
PI16	○	×	○	×
PI16F	○	○	○	○
GPIB	○	×	○	×
ﾚｰｻﾞｰﾌﾟﾘﾝﾀ	○(3)	○(4)	○(3)	○(3)
ｶﾗｰﾊｰﾄﾞｺﾋﾟｰ	○(5)	×	×	○(5)

注：

(1) Add-on 4MB memory Boardの１枚は借用中である。

(2) hm02とhm03のＴＴＹターミナルは共用しているのでケーブルの差替えが必要。

(3) hm01，hm03，hm08に接続されているレーザープリンタはＬＢＰ８Ⅱで、分解は300BPIである。なお、hm01とhm08のレーザープリンターはスイッチによる切り替えで共用している。

(4) hm02に接続されているレーザープリンタはLaser shot B406 で、分解能は240BPIである。

(5) カラーハードコピーはD-SCAN CH-5312でＯＨＰシートへのプリントも可能。

2．2　PAVEC　MD8300　AD／DA　コンバータの構成

本システムはＡＤ／ＤＡ変換部、ホストコンピュータと結ぶ200Kﾃﾞｰﾀ/secの転送レートをもつ光ファイバーインターフェイスユニットからなる。また、４つの遮断周波数を持つローパスフィルタを別筐体としてサポートしている。各部のブロックダイアグラムを図3に、各部の仕様を表２に示す。

ＰＡＶＥＣ社のローパスフィルタ部は１つしかないので、現在のところhm08に接続してある。他のシステムについては、ＮＦ社のプログラマブルフィルタ（FV665）を接続してある。FV665 の仕様を表３に示す。ＰＡＶＥＣ社のものとＮＦ社のものとでは遮断周波数の定義が異なる（ＰＡＶＥＣは０ｄＢ周波数、ＮＦは－３ｄＢ周波数）ので、それらの使用に当たっては注意が必要である。

光リンクユニットとＭＡＳＳＣＯＭＰは高速パラレルインターフェイスPI16F で接続され、各ユニットはＭＡＳＳＣＯＭＰ本体のラックに設置されている。光リンクユニットとＡＤ／ＤＡ変換部は、図４に示すように、３本の光ファイバーケーブルで接続されている。光ケーブルは最長１ｋｍまで延長できるが、現在のところ約２０ｍの長さとしている。

このＰＡＶＥＣ社のＡＤ／ＤＡコンバータを使用するに当たっていくつかの注意事項がある。

①オーバーフローランプ

ＡＤ／ＤＡ変換時に信号のオーバーフローを示すランプがＡＤ／ＤＡコンバータに装備されているが、このランプの点灯は必ずしも信号のオーバーフローを示さない。即ち、データバスのビットが０から１４まですべて１となった場合にこのオーバーフローランプが点灯するようになっている。従って、ＤＡ変換時に、+32767または-32768のデータを出力した場合にはオーバーフローランプが点灯してしまうが、データは正常にＤＡ変換されている。同様に、ＡＤ変換時には±１０Ｖの信号が入力された場合、データは+32767または-32768となり入力データは正常にＡＤ変換されているにもかかわらず、オーバーフローランプが点灯してしまう。

ＤＡ変換時のオーバーフローランプ点灯については、ユーザーがデータの振幅管理さえしておけばまったく無視してかまわない。しかし、ＡＤ変換時のオーバーフローランプの点灯については、このランプが点灯しないようにＡＤ変換器への入力信号の最大振幅を調整することを勧める。その結果、ＡＤ変換したデータのダイナミックレンジは符号付きで15bit となるが、いたしかたのないことであろう。

②ＤＣオフセットの調整

計算機とのインターフェイスを光ケーブルにすることによりＤＣオフセットは従来より小さくすることができたが、オーディオ機器を接続することによって生じる機器間のオフセットはまだ少しの残っている。ＡＤ変換時にこのＤＣオフセットが問題となるが、FV665 のオフセット調節用トリマを調整することによりこのオフセットをキャンセルすることができる。ＡＤ変換を行なう前には必ずこの調整をする

ことが必要である。

ＰＡＶＥＣのローパスフィルタにはこのような調整用トリマが前面パネルに出ていないので、前面パネルを開けて各ＰＣボード上にあるものを調整するか、別途外付けのオフセットバランス用の機器を用意しなくてはならない。

ＡＤ／ＤＡ変換器ボードにもＤＣオフセット調整用のトリマがあるが、これらは低インピーダンスの終端条件下で調整されているので、不用意にさわるべきではない。

③ＰＡＶＥＣローパスフィルタ

ＰＡＶＥＣのローパスフィルタは入力インピーダンスを高くするために、入力端子に直列抵抗が挿入されていない。そのため、入力端子に何も接続しない開放状態では、リークの電荷によって入力電位が次第に高くなり、それにともなってローパスフィルタの出力電位も高くなる。その結果、出力電位は＋１５Ｖに張りついてしまう。これを避けるためには、フィルタの入力端子には常に低インピーダンスの出力（ＤＡＴ出力、プリアンプ出力等）を接続しておけばよい。

ユニット番号	ユニット名称	数量
①ＡＤ／ＤＡ変換部（筐体１）ＭＤ－８３００		
ＭＤ－８４２５Ｔ	１６ｂｉｔデジタルマルチプレクシング用特注筐体	1
ＭＤ－８１５３Ｔ	１６ビット　ＡＤＣ　（１００ｋＨｚ）	2
ＭＤ－８１０７	１６ビット　ＤＡＣ　（１００ｋＨｚﾃﾞｨｸﾞﾘｯﾁｬ付き）	2
ＭＤ－８２０７	ＴＧ（TIMING GENERATOR）	1
ＭＤ－８２０８	高精度プログラマブルペーサ（100n～409.5SEC）	1
ＭＤ－８４０３Ｔ	データ表示付きターミナルボード（特注）	1
②フィルター部（筐体２）ＦＩＬＴＥＲ　ＢＯＸ		
ＭＤ－８４２５ＺＴ５増設用筐体（特注パネル付：高さEIA 5UNIT）		1
ＭＤ－８１７１Ｔ	２ＣＨｹﾞｲﾝｱﾝﾌﾟﾌｨﾙﾀｰ	8
③オプティカルファイバーリンク部（筐体３）ＭＤ－８３０５Ｃ		
ＭＤ－８３０５Ｃ	ロングラインファイバーリンク	1
ＭＤ－８３０２Ｍ	１６ビット　パラレルインターフェース	1
特注Ｉ／Ｆ	マスコンプ用インターフェース	1

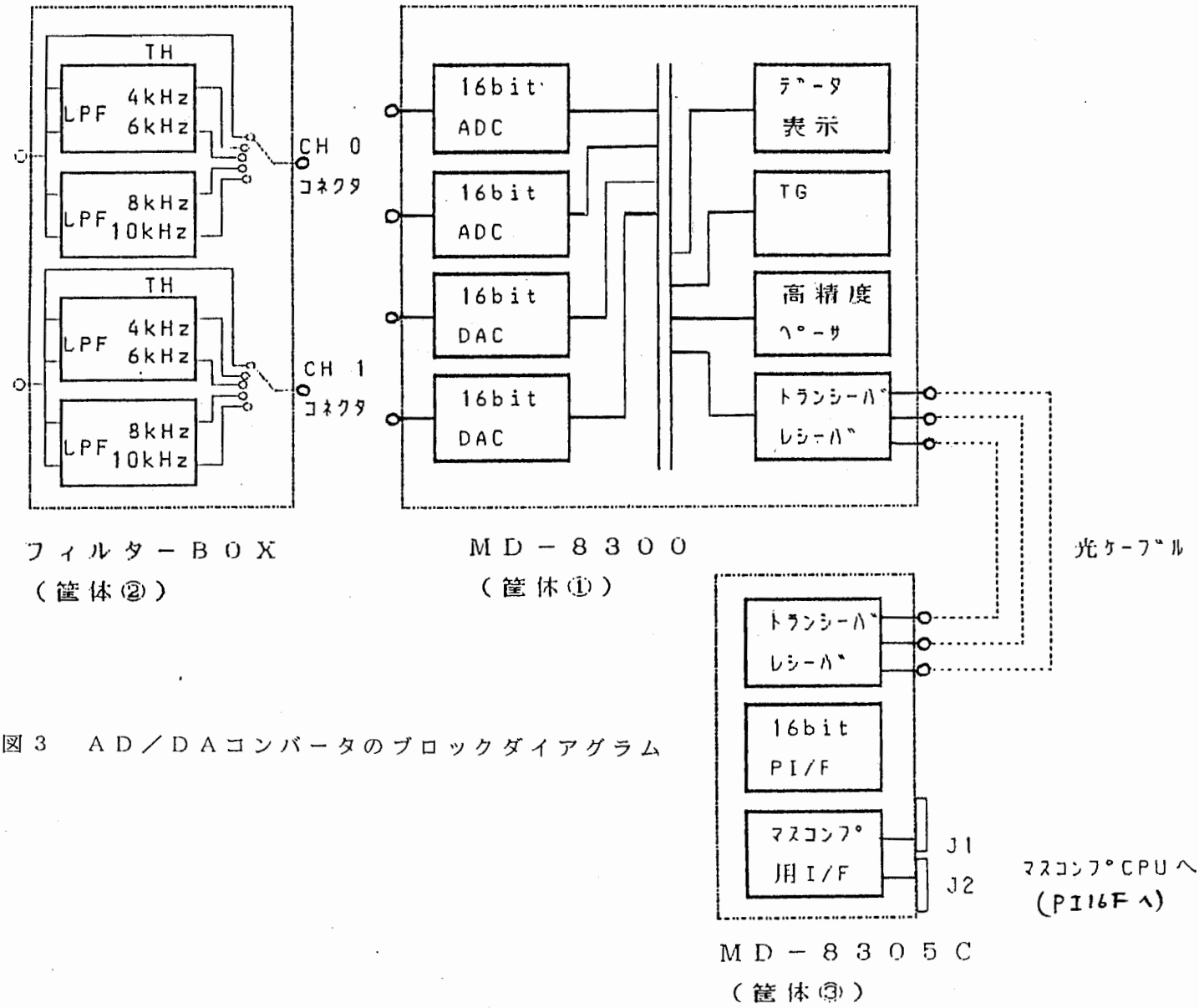


図３　ＡＤ／ＤＡコンバータのブロックダイアグラム

表２　ＡＤ／ＤＡコンバータの仕様

１）ＡＤ／ＤＡ変換部　仕様

アナログ入力	
チャンネル数	２ＣＨ　ＭＡＸ
入力形式	シングルエンド
入力レベル	±１０ VOLT
入力インピーダンス	１０ＭΩ以上
非破壊入力電圧レベル	±２５VOLT
アナログ入力コネクタ	ＢＮＣ端子×２
	アナログ入力コネクタは前面パネルに設定
Ａ／Ｄ変換性能	
分解能	１６ビット
サンプルホールド	付
同上アパーチャータイム	１００ｎＳＥＣ以下
同上不確定時間	１００ｎＳＥＣ以下
変換性能	
リニアリティ	±０．００１５％ＦＳ±１／２ＬＳＢ　ＭＡＸ
オフセット	±１ミリＶ以下
ゲインエラー	±０．０３％ＦＳ以下
スタビリティ	
リニアリティ	±　２PPM／℃
オフセット	±１０PPM／℃
ゲインエラー	±１５PPM／℃
雑音性能	
静雑音	１１０μＶ　ｒｍｓ（Typ）
出力コード	２´Ｓコンプルメント
サンプルレート	１００ｋＨｚ　ＭＡＸ
変換スループット（１チャンネル時）	１００Ｋデータ／ＳＥＣ　ＭＡＸ
（２チャンネル時）	２００Ｋデータ／ＳＥＣ　ＭＡＸ
サンプルタイミング	同時サンプル
アナログ出力	
チャンネル数	２ＣＨ　ＭＡＸ
出力形式	シングルエンド
出力レベル	±１０ VOLT
出力インピーダンス	１００Ω以下
アナログ出力コネクタ	ＢＮＣ端子×２
	アナログ出力コネクタは前面パネルに設定
Ｄ／Ａ変換性能	
分解能	１６ビット
ディグリッチャ	付

変換性能		
リニアリティ		±０．００１５％ＦＳ±１／２ＬＳＢ　ＭＡＸ 但し、１／４ＦＳ以内のレベルでは、±１／３ＬＳＢ以下
オフセット		±１ミリＶ以下
ゲインエラー		±０．０３％ＦＳ以下
スタビリティ		
リニアリティ		±３PPM／℃
オフセット		±１０PPM／℃
ゲインエラー		±４０PPM／℃
入力コード		２´Ｓコンプルメント
変換速度	（１チャンネル時）	１００Ｋ変換／ＳＥＣ　ＭＡＸ
	（２チャンネル時）	２００Ｋ変換／ＳＥＣ　ＭＡＸ （コンピュータのスループットによる）
内部ペーサ設定		１０μ～４０９．５ＳＥＣ（プログラマブル）
周期設定		基本クロック×倍数で設定する。
基本クロック		0.025, 0.1, 1.0, 10.0, 100μSEC および 1.0, 10.0, 100mSEC
倍数		１　から　４０９５倍
周期安定度		3×10^{-5}
外部ペーサ入力		ＴＴＬレベル
外部スタート制御入力		ＴＴＬレベル
データ表示		マルチプレクサアドレス、ペーサ、データバスの状態、モード、コンピュータアクセスの状態、オーバーフローを表示
外形寸法		２３７.５Ｈ×４３８Ｗ×４４０Ｄ
重量		約２０ｋｇ（特注ＭＤ－８４２５）
電源		各９０～１２８VOLT　ＡＣ，１６５ＶＡ　MAX
使用ヒューズ		３Ａ
環境条件		
保存温度		－１０～５０℃
動作温度		０～３５℃
性能保証推奨動作温度		２２～２８℃
湿度		２０～８５％結露なきこと

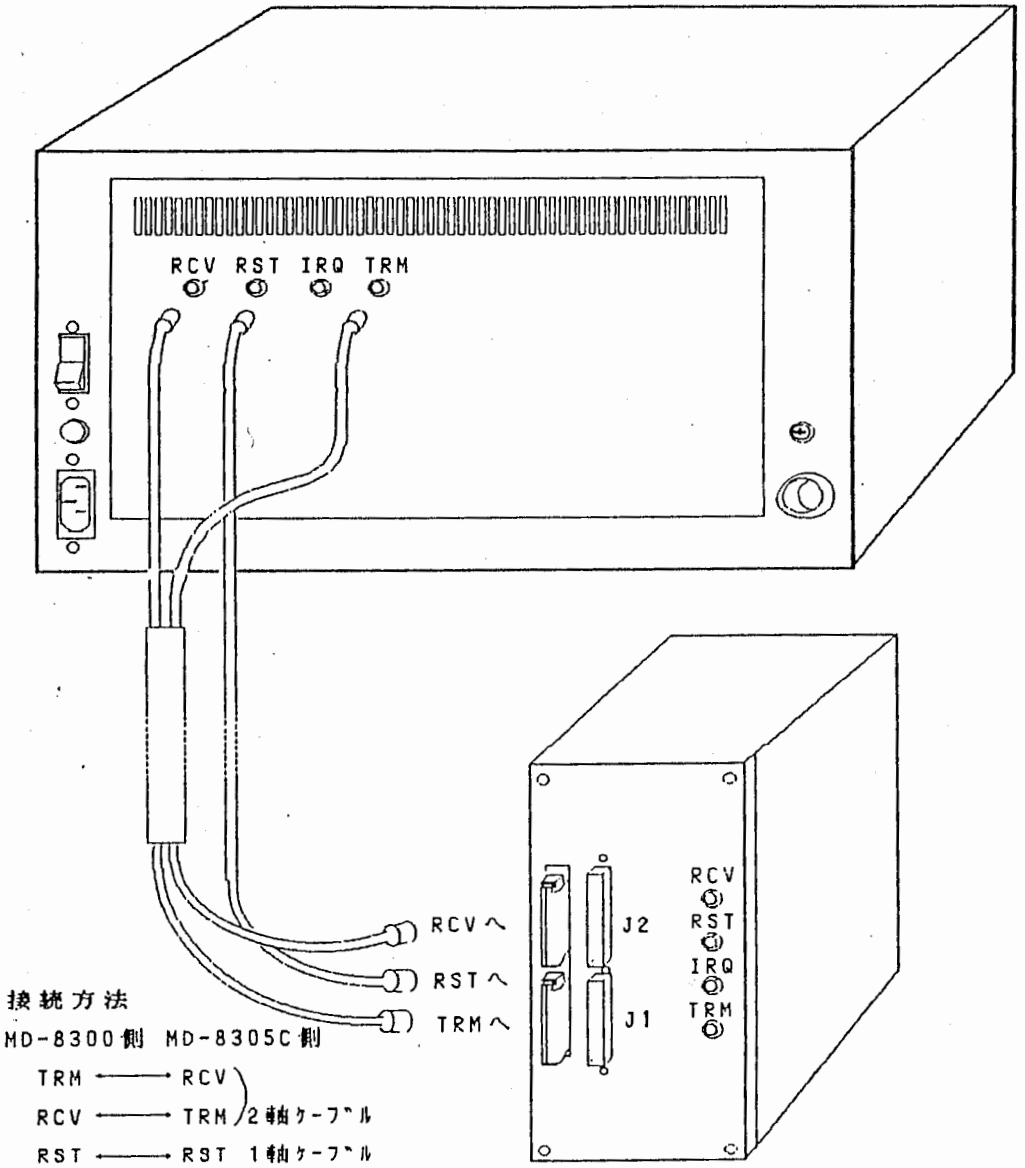
２）フィルター部　仕様　ＦＩＬＴＥＲ　ＢＯＸ

チャンネル数	４ｃｈ　ｍａｘ Ａポート及びＢポート各々２ＣＨづつ独立
フィルター特性	八次連立チェビシェフＬＰＦ
阻止域特性	
減衰傾度	１００ｄＢ／ｏｃｔ相当
減衰特性	９０ｄＢ（ｔｙｐ）２.０ｆｃ
最小減衰量	１０２ｄＢ（ｔｙｐ）
高域減衰量	８２ｄＢ以上
遮断周波数	４，６，８，１０ｋＨｚ，スルー（５点をスイッチにより切り替え） Ａポート及びＢポート各々２ＣＨづつスイッチをサポート
中心周波数確度	±５％
通過帯域特性	
利得	１（±０．１ｄＢ以内　低域にて）
リップル	±０．４ｄＢ以内
最大入力信号レベル	±１０Ｖ／通過域利得（ｆ≦１０ｋＨｚ） ±　５Ｖ／通過域利得（ｆ＞１０ｋＨｚ）
負荷インピーダンス	１０ｋΩ以下
ノイズレベル	１５０μＶｒｍｓ以下
オフセット電圧	±３ｍＶ以下
オフセットスタビリティ	０．１５ｍＶ／℃（Ｔｙｐ）
高調波歪率	０．１５％以下

外形寸法	２３７.５Ｈ×４３８Ｗ×４４０Ｄ
重量	約１５ｋｇ（特注筐体）
電源	各９０～１２８VOLT　ＡＣ，１６５ＶＡ　MAX
環境条件	
保存温度	－１０～５０℃
動作温度	０～３５℃
性能保証推奨動作温度	２２～２８℃
湿度	２０～８５％結露なきこと

３）オプティカルファイバーリンク部　仕様

最高転送速度	５μＳ／データ（２バイト）
必要ファイバーケーブル数	２　＋１（＋１はリセット信号用）
コネクタタイプ	ＳＭＡタイプ
推奨ファイバーケーブル	石英ﾀｲﾌﾟｹｰﾌﾞﾙ（YHP社製YFBR-3010/3110/3210S OP-002相当品）
ファイバーケーブル長	２０ｍ（１ｋｍまで可能）
転送方式	調歩式シリアル転送
コンピュータインターフェース	マスコンプ用 Ｊ１（ＩＮ） Ｊ２（ＯＵＴ）
電源（ｺﾝﾋﾟｭｰﾀ側ｱﾀﾞﾌﾟﾀ）	９０～１２８VOLT　ＡＣ
消費電力	４５ＶＡ
外形寸法	２５６Ｈ　×　１００Ｗ　×　１８３Ｄ
重量	約４．５ｋｇ



２．３　オーディオ系の構成

システムのオーディオ系は、民生用のプリメインアンプ、ＤＡＴ、マイクロフォンアンプ、モニタ用スピーカ等からなる。この系の構成方針は、「ＡＤ／ＤＡ系とＤＡＴ系というふたつの録音再生系を切り替えて用いるとともに、両者の間で信号のダビングを可能とする」とし、だれでも容易に操作できることを目標とした。

オーディオ系のブロックダイアグラムを図５に示す。

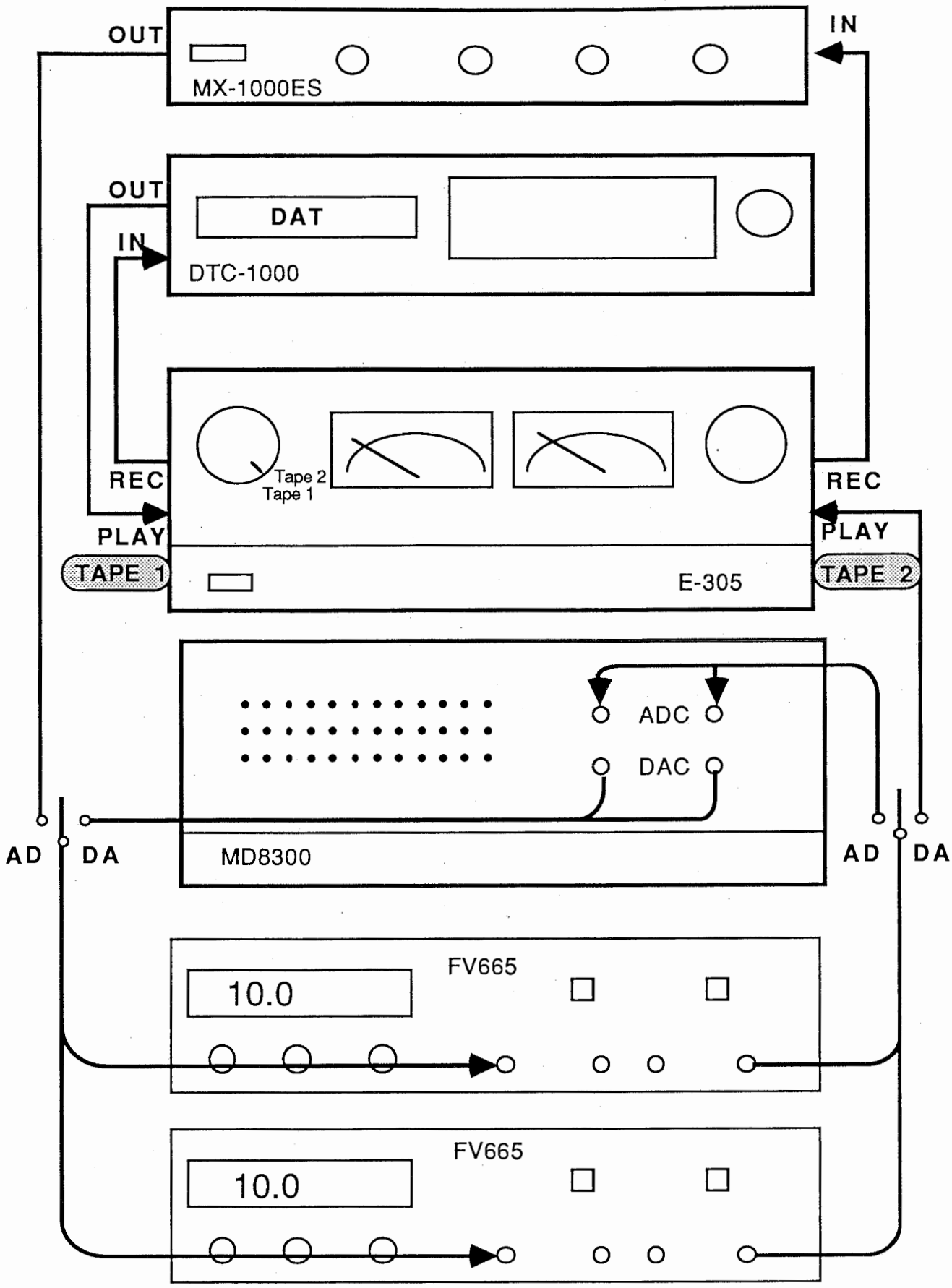
マイクロフォンアンプはソニーのMX-1000ES を用いている。このアンプはＤＡＴの出力信号をＡＤする際のレベル変換アンプとしても使用している。

ＤＡＴはソニーのDTC1000ES あるいはTCD-D10 を用いている。

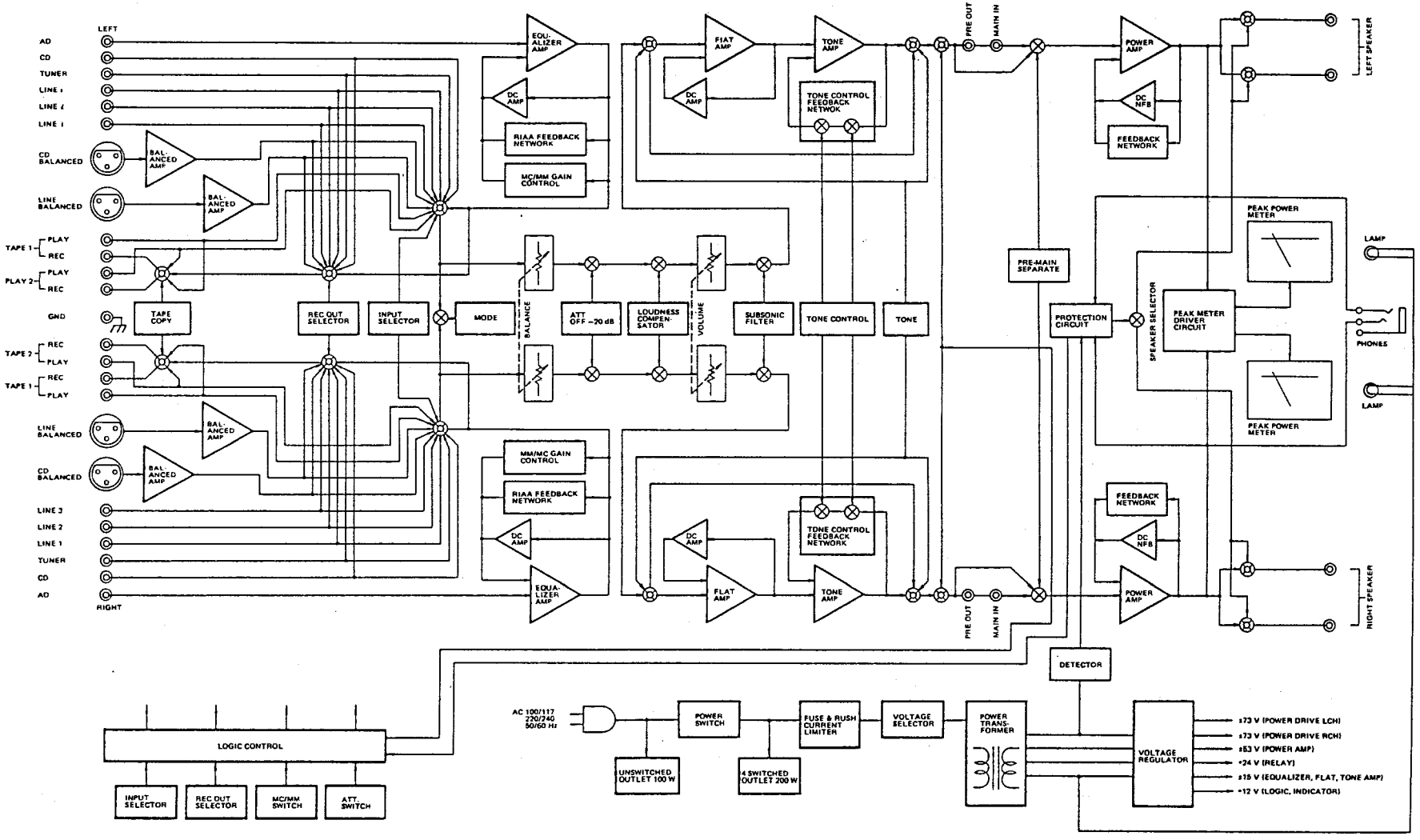
プリメインアンプはアキュフェーズのE-305 あるいはテクニクスのSU-V60を用いている。このアンプにはモニター用のスピーカ（ダイヤトーンDS-107V） が接続してあり、ＤＡＴもしくはＤＡからの出力信号をモニターできる。ＡＤ／ＤＡとＤＡＴはそれぞれTape 1、Tape 2に接続してあるので、ソースセレクトスイッチとダビングセレクトスイッチを選択することによってふたつのソースの選択とダビング方向を選択できる。

アンチエイリアジング用のローパスフィルタにはＮＦのFV665 あるいはＰＡＶＥＣのフィルタを用いている。FV665 は各チャンネルごとに１台用意されているので、フィルタをカスケード接続することにより-96dB/oct.の減衰特性をもたせることができる。なお、現在のところ、ＡＤ変換時とＤＡ変換時には入出力ケーブルをつなぎ替えなければならない。

図５　オーディオ系のブロックダイアグラム等



E305 のブロックダイアグラム



E305の諸元

連続平均出力

180W/ch　4Ω負荷
130W/ch　8Ω負荷
(両チャンネル同時動作 20～20,000Hz間
ひずみ率0.02%)

全高調波ひずみ率

0.02%　4～16Ω負荷
(両チャンネル同時動作　0.25W～連続平均出力間
20～20,000Hz間)

IMひずみ率

0.01%

周波数特性

MAIN AMP INPUT　: 20～20,000Hz　0　−0.2dB
(定格出力時)
0.5～150,000Hz　0　−3.0dB
(1W出力時)
HIGH LEVEL INPUT : 20～20,000Hz　0　−0.2dB
(定格出力時)
LOW LEVEL INPUT　: 20～20,000Hz　+0.2　−0.5dB
(定格出力時)

ダンピング・ファクター

100 (8Ω負荷　50Hz)

定格入力・入力インピーダンス

入力端子	入力感度		入力インピーダンス
	定格出力時	EIA(1W出力時)	
AD INPUT(MC)	0.128mV	0.01mV	100Ω
AD INPUT(MM)	4.3mV	0.38mV	47kΩ
HIGH LEVEL INPUT	125mV	11.8mV	20kΩ
BALANCED INPUT	125mV	11.8mV	40kΩ
MAIN AMP INPUT	1.28V	121mV	20kΩ

ディスク最大入力

MM入力　: 300mVrms、1kHz、ひずみ率　0.005%
(REC OUT)
MC入力　: 8.0mVrms、1kHz、ひずみ率　0.005%
(REC OUT)

定格出力・出力インピーダンス

PRE OUTPUT　1.28V　200Ω
TAPE REC OUTPUT　125mV　200Ω (ADより)
HEADPHONES　0.4V　適合インピーダンス
4～100Ω

ゲイン

MAIN INPUT → OUTPUT : 28dB
HIGH LEVEL INPUT → PRE OUTPUT : 20dB
AD INPUT (MM) → TAPE REC OUTPUT : 29dB
AD INPUT (MC) → TAPE REC OUTPUT : 60dB

トーン・コントロール

ターンオーバー周波数及び可変範囲
低音 : 300Hz　±10dB (50Hz)
高音 : 3kHz　±10dB (20kHz)

ラウドネス・コンペンセーター

+6dB (100Hz)
(VOLUMEコントロール　−30dBにて)

S/N・入力換算雑音

入力端子	入力ショート・A-補正		EIA S/N
	定格入力時 S/N	入力換算雑音	
MAIN AMP INPUT	123dB	−121dBV	102dB
HIGH LEVEL INPUT	108dB	−126dBV	82dB
BALANCED INPUT	90dB	−108dBV	82dB
AD INPUT (MM)	86dB	−137dBV	80dB
AD INPUT (MC)	66dB	−147dBV	75dB

サブソニック・フィルター

17Hz、−12dB/oct

アッテネーター

−20dB

パワーメーター

対数圧縮型ピークレベル表示
dB目盛及び8Ω負荷時の出力直読

負荷インピーダンス

2～16Ω

使用半導体

77 Tr　34 FET　20 IC　78 Di

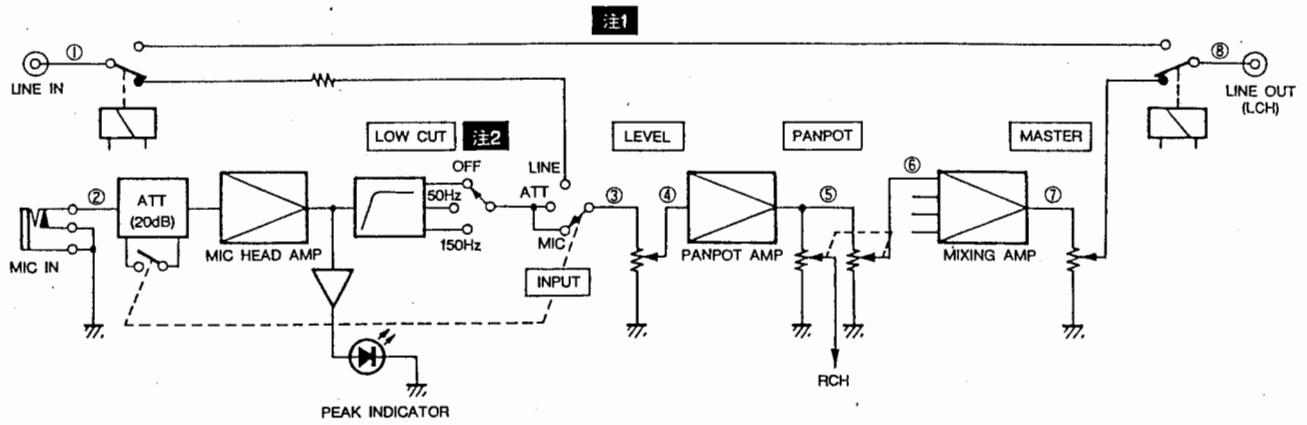
電源・消費電力

100V　117V　220V　240V　50/60Hz
無入力時　60W
電気用品取締法　310W
8Ω負荷定格出力時　490W

寸法・重量

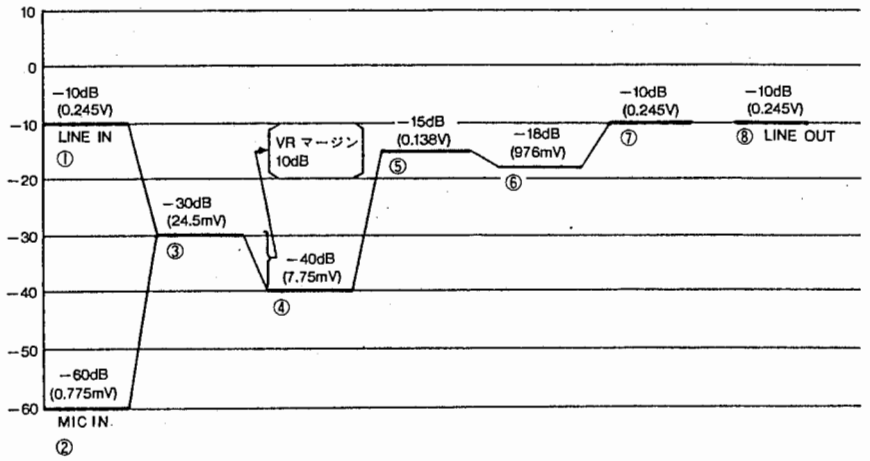
幅475 mm × 高さ170 mm(脚含む) × 奥行375 mm
20.5kg

ブロックダイヤグラム



注1　入力3または4チャンネル
注2　入力1または2チャンネルのみ

レベルダイヤグラム



入力端子

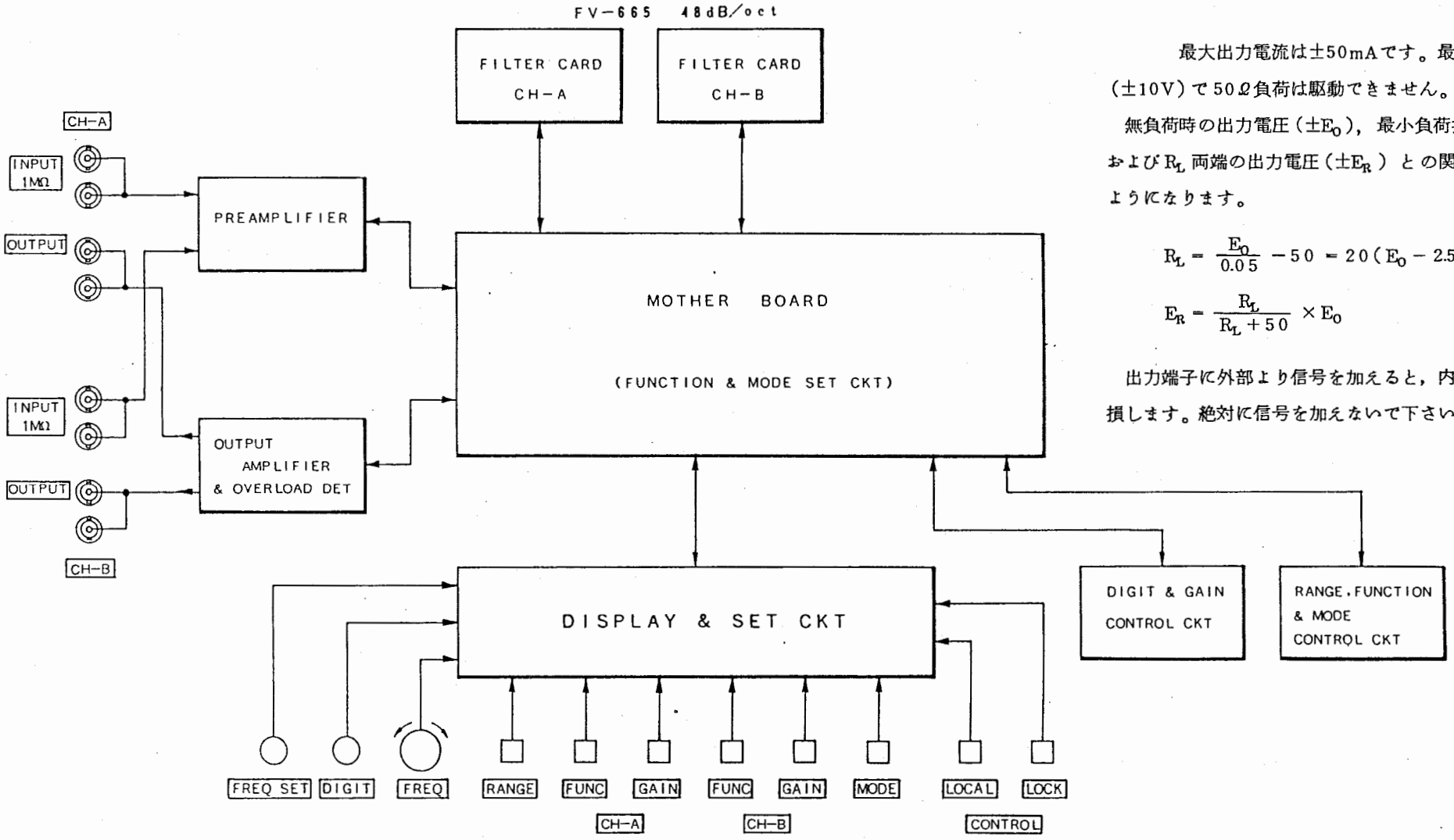
端子	端子形状	数	最小入力レベル	入力インピーダンス
MIC	標準ジャック	4	−70dB 0.245mV	6kΩ
LINE IN	ピンジャック	4	−20dB 77.5mV	50kΩ以上

出力端子

端子	端子形状	数	基準出力レベル	負荷インピーダンス
LINE OUT	ピンジャック	2	−10dB 0.245V	10kΩ以上

ローカットフィルター(1、2CHのみ)
50Hz/150Hz (6dB/oct)
マイクロホン入力ピークインジケーター
−20dB以上の入力時点灯
周波数特性　10Hz〜150kHz $^{+0}_{-3}$dB
ひずみ率　0.005%
入力換算ノイズレベル　−130dBV(MIC入力、JIS A)
S/N　MIC：68dB (JIS A)
LINE：73dB (JIS A)

マイクロフォンアンプ MX1000ESの
ブロックダイアグラム　(上段)
レベルダイアグラム　(中段)
諸元　(下段)



最大出力電流は±50mAです。最大出力電圧（±10V）で50Ω負荷は駆動できません。

無負荷時の出力電圧（$\pm E_O$），最小負荷抵抗（R_L）およびR_L両端の出力電圧（$\pm E_R$）との関係は下式のようになります。

$$R_L = \frac{E_O}{0.05} - 50 = 20(E_O - 2.5)$$

$$E_R = \frac{R_L}{R_L + 50} \times E_O$$

出力端子に外部より信号を加えると，内部回路が破損します。絶対に信号を加えないで下さい。

MODEL FV−664/665 BLOCK DIAGRAM

FV665の諸元

遮断周波数（fc）

可変範囲　0.01 Hz ～ 159.9 kHz

レンジおよび分解能

レンジ*	分解能
0.1 ～ 159.9 kHz	100 Hz
0.01 ～ 15.99 kHz	10 Hz
1 ～ 1599 Hz	1 Hz
0.1 ～ 159.9 Hz	0.1 Hz
0.01 ～ 15.99 Hz	0.01 Hz

fc における利得（ただし，平坦部を0dBとする。）

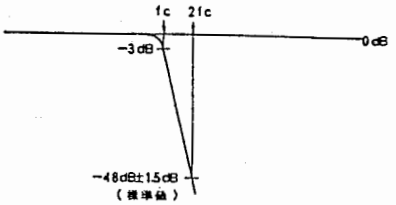
FV−665　$-3^{+1}_{-1.2}$)dB　LP−MF，HP

$-15.3^{+1}_{-1.2}$)dB LP−PL

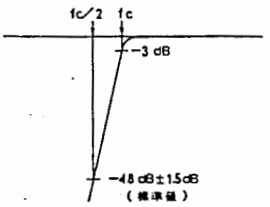
ただし，0.1～159.9 kHz レンジの誤差はそれぞれ $^{+2}_{-2.5}$) dB

通過帯域利得誤差

FV-665		
0.1～159.9kレンジを除いたレンジ		
±0.3dB	DC～fc／2 2fc～100kHz	LP−MF HP
$^{+0.2}_{-0.4}$) dB	DC～fc／10	LP−PL
0.1～159.9kレンジ		
±0.6dB	DC～fc／2	LP−MF
$^{+0.5}_{-0.7}$)dB	DC～fc／10	LP−PL
$^{+1}_{-3}$)dB	2fc～1MHz	HP



ローパスフィルタ（MF）



ハイパスフィルタ

減衰特性　FV−665　48 dB／oct　±4 dB

最大減衰量　90dB以上　f_{IN} ≦100kHz)

80dB以上　f_{IN} ≦1MHz

チャネル間位相整合　FV−664　2°　FV−665　4°　DC～2fc（LP）または fc／2～100 kHz（HP），fc ≦ 100 kHz ，TYP

スルー時の周波数特性　DC～1 MHz　+0.5，−3 dB

遅延特性　LP−PL時，図6－5，6による

入力特性

インピーダンス　1 MΩ／100 pF以下，不平衡，BNC−R（正・背面並列接続）

最大電圧　±10V以上（利得 0 dB）　±1 V以上（利得 20 dB）　DC～300 kHz　1 MHz では左記の×0.4

許容最大電圧　±150 V

利　得*　0，20 dB

出力特性

インピーダンス　50Ω，不平衡，BNC−R（正・背面並列接続）

最大電圧　±10 V以上，DC～300 kHz，1 MHz では±4 V

最大電流　±50 mA

高調波ひずみ率　0.1 %以下，f_{IN} ≦ 10 kHz　0.3 %以下，f_{IN} ≦ 100 kHz　ただし，f_{IN} はフィルタ平坦部またはスルー時の入力周波数

最大出力，150Ω負荷

直流オフセット　パネル面調整器により0に調整可能，可変範囲　約±30 mV

ドリフト

~~FV-664　100μV/℃　TYP~~，FV-665　200μV／℃　TYP

ただし周波数表示器が10以上設定のとき

ノイズ　100μVrms（DC～100kHz帯域）

800μVrms TYP（DC～ 10MHz帯域 ，HP）

チャネル間クロストーク　−75 dB以下（DC～100 kHz）

−65 dB以下（DC～1 MHz）

２．４　応答スイッチ

マスキング実験や様々な弁別閾を求める心理物理実験において、刺激音の制御を被験者の応答に応じて行ないたい場合が多くある。そのような場合のために、ＭＡＳＳＣＯＭＰのパラレルインターフェイスPI16を用いた応答スイッチを用意してある。この応答スイッチの構成図を図６に、接続図を図７に示す。

この応答スイッチのボタンには、最も押下音の少ないもの（日本デンヨーEP720）を用いており、マスキング実験などにおいても安心して使用できるように設計してある。また、このスイッチにはＬＥＤが組み込まれているので、PI16の出力を利用して、被験者に教示を与えることも可能である。ただ、この応答スイッチは一人の被験者用に設計してあるために、多数の被験者からの反応を求めるような実験には使用できない。そのようなシステムも各種の心理物理実験には必要であり、今後整備していく必要がある。

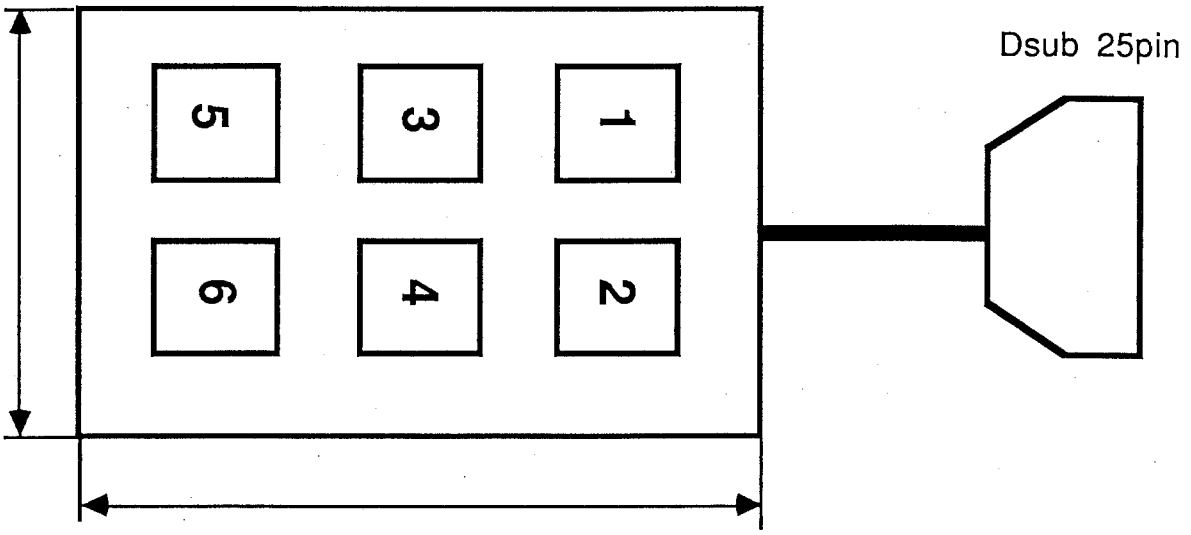
マスコンプのPI16とは最も単純な方法で接続してある。図７に示されるようにストローブ信号をCK10より与え、データビットの1～6までをスイッチでグランドに落とすことによってデータを確定している。例えば、４番のスイッチを押下した場合４ビット目が０で他のビットには１が立つのでデータは(F7)Hとなる。ただし、PI16は２バイト毎にデータを読みこむので、上位バイトに押下したスイッチのデータが、下位バイトには(00)Hが入るために、実際にとりこまれるデータは(F7FF)Hとなる。

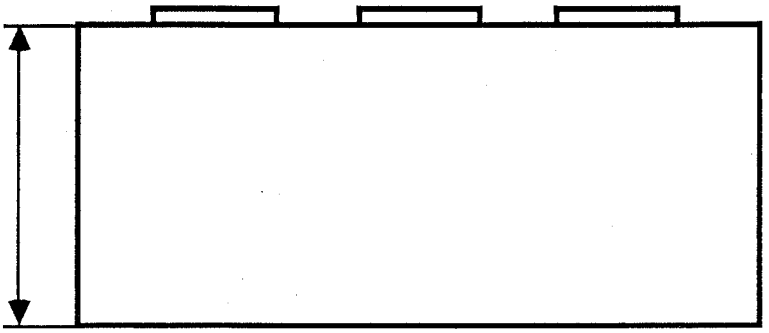
２．５　プログラマブルアッテネータ

マスキング実験や音圧に関係する弁別閾を求める心理物理実験において、刺激音の音圧を計算機の指示によって自動的に制御したい場合が多くある。そのような場合のために、ＭＡＳＳＣＯＭＰのＧＰＩＢインターフェイスIB488 に接続して用いるプログラマブルアッテネータを用意してある。

このプログラマブルアッテネータは㈱多摩川電子社製のTPA-308A型で、その仕様を表３に示す。使用可能周波数はDCから300MHzまでと可聴周波数帯域（DC～20kHz）を扱うにはオーバースペックである。このアッテネータの特性インピーダンスは50Ωと低いので、入力端子に接続する機器の出力インピーダンスが高い場合や、低インピーダンスのものでもドライブ能力が低いものの場合（例えばFV665）には注意が必要である。即ち、FV665 などを直接アッテネータに接続すると波形歪が生じる場合がある。この現象を避けるためには、入力端子にと直列に200Ω～1KΩの抵抗を接続して、アッテネータの入力インピーダンスを上げてやればよい。ただし、この場合、減衰量の少ない部分でアッテネータの表示値と実際の減衰量の誤差が大きくなるので補正が必要である。将来的には10kΩ 程度の特性インピーダンスを持つアッテネータに変更する必要がある。

図６　応答スイッチの構成





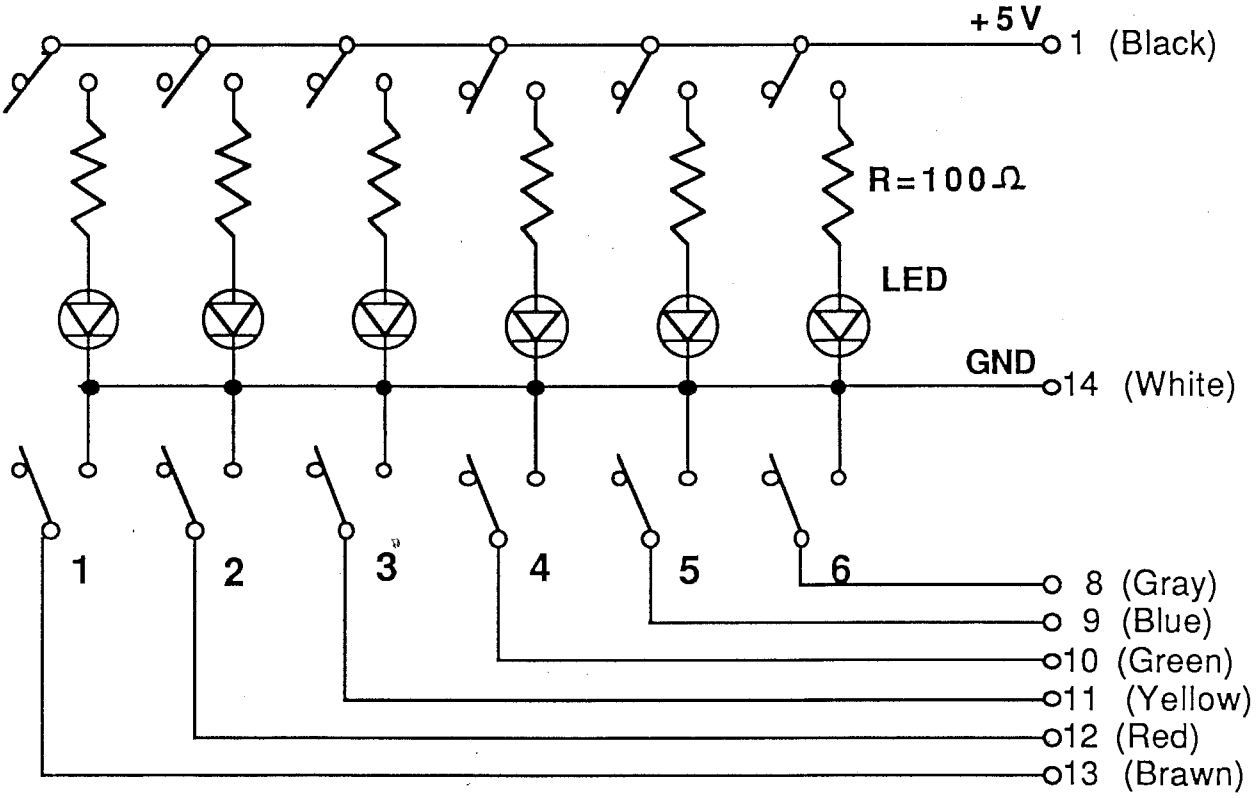
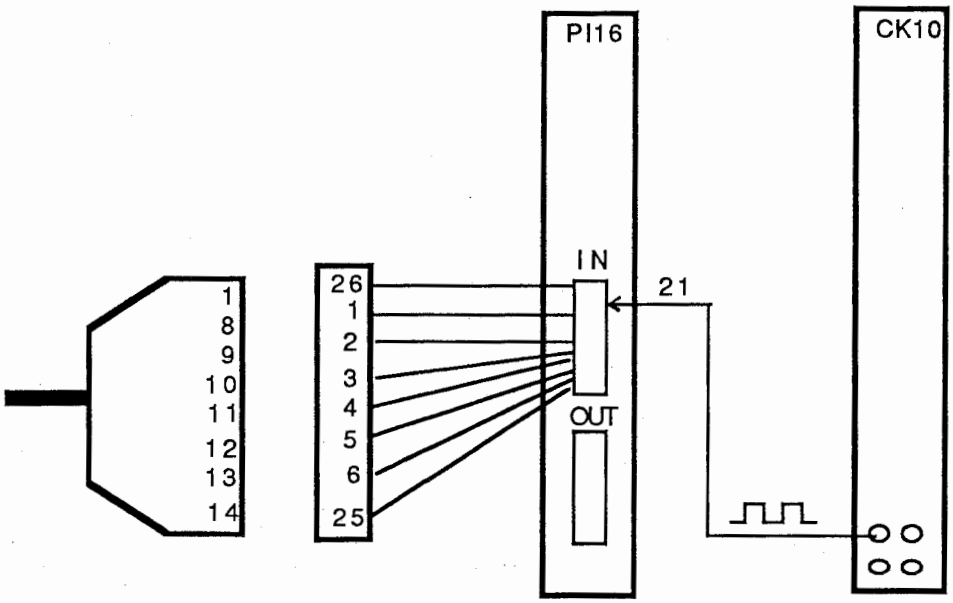
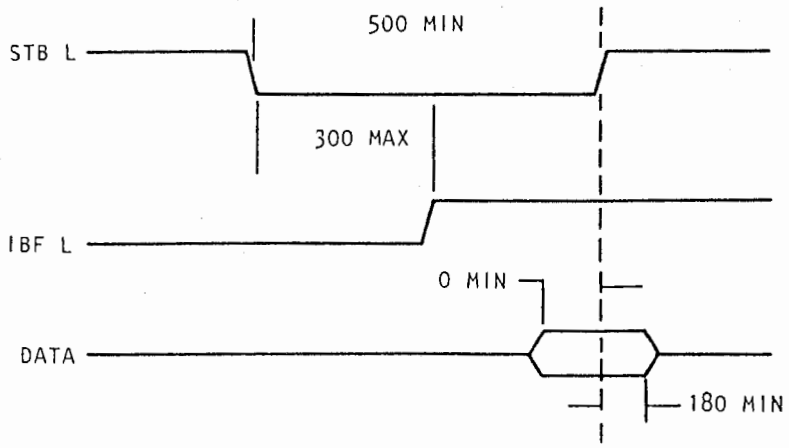


図７　応答スイッチの接続図



PI16 Connector INPUT			
+5V	26	25	GND
	24	23	
	22	21	STB (L) Strobe Input
	20	19	IBF (H) Input Buffer Full
	18	17	
IN15 (H)	16	15	IN7 (H)
IN14 (H)	14	13	IN6 (H)
IN13 (H)	12	11	IN5 (H)
IN12 (H)	10	9	IN4 (H)
IN11 (H)	8	7	IN3 (H)
IN10 (H)	6	5	IN2 (H)
IN9 (H)	4	3	IN1 (H)
IN8 (H)	2	1	IN0 (H)



NOTE: ALL TIMING IS IN NANOSECONDS

INPUT HANDSHAKE

適用周波数	ＤＣ～３００ＭＨｚ
特性インピーダンス	５０Ω
減衰量	０～９９．９ｄＢ （０．１ｄＢステップ有り）
減衰量誤差	±（２％＋０．２）ｄＢ以下
挿入損失	４ｄＢ以下
ＶＳＷＲ	１．３ｄＢ以下
減衰量表示	１０進３桁
信号切換速度	４ｍｓ以下
入出力コネクタ	ＢＮＣ－Ｊ
制御方式	ＧＰ－ＩＢ制御、ＢＣＤ制御 （外部制御）および手動制御
電源	ＡＣ１００Ｖ ±１０％　約２５ＶＡ
周囲温度	５℃～３５℃

プログラマブルアッテネータ　の諸元

第３章　ＭＡＳＳＣＯＭＰ上の周辺機器制御ソフトウェアライブラリー

３．１　概説

３．１．１　使用方法

ＭＡＳＳＣＯＭＰの周辺機器を制御するライブラリーとして、以下の２種類のものがある。

①ｌｉｂｐａｖｅｃ．ａ
　ＰＡＶＥＣ　Ａ／Ｄ・Ｄ／Ａコンバータを制御する。

②ｌｉｂｄａｃｐ．ａ
　応答スイッチ、ならびに、プログラマブルアッテネータを制御する。

これらのライブラリーは、共に／ｕｓｒ／ｌｏｃａｌ／ｌｉｂにインストールされているので、目的のプログラムを作成するときは、コンパイル時に－ｌオプションでリンクすれば良い。

例えば、作成しているプログラムのファイルがｍｙｐｒｏｇ．ｃならば、以下の様にすれば良い。

```
%cc myprog.c -o myprog -lpavec -ldacp -lmr -lm
```

-lmr -lm　は、ＭＡＳＳＣＯＭＰの「Data Acquisition Application Subroutines Library」を使用する際に必要なもので、ｌｉｂｐａｖｅｃ．ａ、ｌｉｂｄａｃｐ．ａは内部でこれらを使っているので必要である。

３．１．２　ＭＡＳＳＣＯＭＰのＡ／Ｄ・Ｄ／Ａについて

ＭＡＳＳＣＯＭＰのＡ／Ｄ・Ｄ／Ａについて、簡単に説明する。詳しくは、マニュアル「 Data Acquisition Application Programing Manual 」を参照されたい。

ＭＡＳＳＣＯＭＰのＡ／Ｄ・Ｄ／Ａの方法には、以下の３種類の方法がある。

①アレイ　トランスファ（Array Transfer）
②キュード　トランスファ（Queued Transfer）
③ダイレクト　ディスク　トランスファ（Direct Disk Transfer）

①アレイ　トランスファ（Array Transfer）

メモリーからＡ／Ｄ・Ｄ／Ａする方法で、データ長が短いときに有効である。使い勝手は、比較的簡単である。データ長が決っているときは、メモリーをmallocを使って割り当てなくて良いので、特に簡単である。

②キュード　トランスファ（Queued Transfer）

リング　バッファ(Ring Buffer)と呼ばれる環状のバッファを使って、Ａ／Ｄ・Ｄ／Ａを行なう。データ長が予め分ってないときなどには、便利である。長いデータをＡ／Ｄ・Ｄ／Ａするときは、この方法を使うことを勧める。

リング　バッファを管理しないといけない分だけ、使い勝手は複雑になる。

③ダイレクト　ディスク　トランスファ（Direct Disk Transfer）

大容量のデータを、しかも、高速にＡ／Ｄ・Ｄ／Ａしたいときに有効な方法である。

しかし、この方法は先の２種類の方法とは異なり、以下の制約条件がある。

・Memory-locked processes
・Disk-locked processes
・Contiguous disk files

このため、通常の使用には不向きであるので、本ライブラリーではこの方法は使用していない。

３．２　ＰＡＶＥＣによるＡ／Ｄ・Ｄ／Ａ(libpavec.a)

３．２．１　使用手順

ここでは、libpavec.aに含まれる関数の大まかな使い方を説明する。
ＰＡＶＥＣを使用するには、一般には以下の手順が必要である。
①ＰＡＶＥＣが接続されているデバイス(PI16F)をオープンする
②ＰＡＶＥＣを初期化する
③メモリー等を割り当てる
④ファイル等を開く
⑤Ａ／Ｄ・Ｄ／Ａを行なう
⑥デバイス(PI16F)をクローズする
⑦その他後処理をする
ただし、nice値に関しては、libpavec.aに含まれる関数の内部で、特に操作していので必要があれば、利用者の方で設定／解除を行なわなければならない。
以下の説明で用いる変数は、予め次の様に宣言されているものとする。

```
double  freq;           /* 標本化周波数〔kHz〕 */
double  duration;       /* 時間長〔msec〕 */
char    ch0, ch1;       /* チャネルの使用／不使用のフラグ */
long    length;         /* データの個数 */
FILE    *fp;            /* ファイル　ポインタ */
union
   {    short   *buf;
        int     dummy;
   } data;              /* データ */
```

アレイ　トランスファ (Array Transfer)の場合は、概略以下の様になる。
①②に対して
Ｄ／Ａの場合：pavec_init(freq, duration, ch0, ch1, "w", &length)
Ａ／Ｄの場合：pavec_init(freq, duration, ch0, ch1, "r", &length)
⑤に対して
Ｄ／Ａの場合：pavec_xout(data.buf, length)
Ａ／Ｄの場合：pavec_xin(data.buf, length)
⑥⑦に対して
pavec_stop()
なお、pavec_file_out(fp, freq, duration, ch0, ch1)は、内部でこれら一連の処理を行なっている。
キュード　トランスファ (Queued Transfer)の場合は、概略以下の様になる。
①②に対して
Ｄ／Ａの場合：pavec_init(freq, duration, ch0, ch1, "w", &length)

Ａ／Ｄの場合：pavec_init(freq, duration, ch0, ch1, "r", &length)
③④に対して
pavec_setQuedBuf(fp)
⑤に対して
Ｄ／Ａの場合：pavec_qout()
Ａ／Ｄの場合：pavec_qin()
⑥⑦に対して
pavec_stop()

もう少し細かく制御したいときは、以下の様な使い方もできる。

①に対して
pavec_open()
②に対して
pavec_set_paket(freq, duration, ch0, ch1, &length)
Ｄ／Ａの場合：pavec_mode("w")
Ａ／Ｄの場合：pavec_mode("r")
③に対して
pavec_mallocQBuf()
pavec_releaseQBuf(fp)
⑤に対して
Ｄ／Ａの場合：pavec_qout()
Ａ／Ｄの場合：pavec_qin()
⑥⑦に対して
pavec_stop()

３．２．２　使用上の注意

ＰＡＶＥＣを使用する前には、必ず電源が入っていることを確認すること。いったんＰＡＶＥＣの初期化等に失敗すると、ＤＡＣＰが全て使えなくなる可能性がある。もし、そのような状態になれば、スーパーユーザになって /etc/loaddacp を実行しないといけない。各機器の調整を行ない /etc/loaddacp を実行しても、Ａ／Ｄ・Ｄ／Ａが行なわれない場合は、機器が故障している可能性がある。

キュード　トランスファを用いてＡ／Ｄ・Ｄ／Ａを行なうと、少しのディレイ(delay)［約１秒程度］が生じることがある。これは、キュードバッファを解放するのに少し時間がかかるためである。

３．２．３　使用例

［サンプルプログラム（ｄａｄａ）］

説明：

このプログラムは、ファイル名をたずねてきて、そのファイルを順次Ｄ／Ａし続ける。
pavec_file_out(fp, freq, time, ch0, ch1)を使用している。

使用例：

％ｄａｄａ　２０．０　１　１　３

リスト：

```
# include       <stdio.h>
# include       <sys/types.h>
# include       <sys/stat.h>
# define        MAX_NAME        80
# define        TRUE            1
# define        FALSE           0

/*
-----------------------------
        main
-----------------------------
 */
main( argc, argv )
   int  argc;
   char *argv[];
   {
        /* set parameter */
extern  double  atof();
        double  freq     = atof( argv[1] );
        char    ch0      = atoi( argv[2] );
        char    ch1      = atoi( argv[3] );
        int     interval= atoi( argv[4] );
        /* D/A loop */
        da_loop( freq, ch0, ch1, interval );
        printf( "¥nHona'Sai-Nara !!¥n" );
   }

/*
---------------------------------
        da_loop
---------------------------------
 */
da_loop( freq, ch0, ch1, interval )
double  freq;           /* sampling frequency [kHz] */
char    ch0, ch1;       /* channel flag [0 or 1]*/
int     interval;       /* waitting interval [sec]*/
   {
        char    file[MAX_NAME];
        int     len;
```

```
        long    length;
        double  time;
        /* D/A loop */
        while ( 1 )
           {    printf( "Please input file name == > " );
                if ( !fgets( file, MAX_NAME, stdin ) )
                        break;
                else
                if ( (len = strlen( file )) <= 1 )
                        break;
                file[len-1] = 0;
                if ( get_file_length( file, &length ) )
                   {    /* set D/A time */
                        time = (double)length/freq;
                        /* Do D/A */
                        da_DA( file, freq, time, ch0, ch1 );
                        /* sleep interval */
                        sleep( interval );
                   }
                else
                        fprintf( stderr, "No such file %s exist !!¥n", file );
           }/* end-of-loop */
   }

/*
---------------------------------
        da_DA
---------------------------------
 */
da_DA( name, freq, time, ch0, ch1 )
char    *name;          /* file name */
double  freq;           /* sampling frequency [kHz] */
double  time;           /* sample total time [msec] */
char    ch0, ch1;       /* channel flag [0 or 1]*/
   {
        FILE    *fp, *fopen();
nice( -20 );
        if ( (fp = fopen( name, "r" )) != NULL )
           {    pavec_file_out( fp, freq, time, ch0, ch1 );
                fclose( fp );
           }
        else
                fprintf( stderr, "Can't open file %s !!¥n", name );
nice(  20 );
  }

/*
---------------------------------
        get_file_length
---------------------------------
 */
get_file_length( file, length )
char    *file;
```

```
int     *length;
   {
        struct  stat buf;

        /* set length */
        if ( stat( file, &buf ) == 0 )
           {    *length = buf.st_size/2;
                return( TRUE );
           }
        else
                return( FALSE );
   }
```

関数の説明：

```
main( argc, argv )
   int  argc;
   char *argv[];
```

コマンドを実行したときの引数リストから、必要な変数の値を得る。

argv[1] -> freq　　標本化周波数［kHz］

argv[2] -> ch0　　チャネル０の使用／不使用のフラグ[0または1]

argv[3] -> ch1　　チャネル１の使用／不使用のフラグ[0または1]

argv[4] -> interval　　Ｄ／Ａの間隔[sec]

```
da_loop( freq, ch0, ch1, interval )
double  freq;           /* sampling frequency [kHz] */
char    ch0, ch1;       /* channel flag·[0 or 1]*/
int     interval;       /* waitting interval [sec]*/
```

ファイル名をたずねてきて、それをda_DA()に渡す。

Ｄ／Ａ毎に、interval[sec]だけｓｌｅｅｐする。

```
da_DA( name, freq, time, ch0, ch1 )
char    *name;          /* file name */
double  freq;           /* sampling frequency [kHz] */
double  time;           /* sample total time [msec] */
char    ch0, ch1;       /* channel flag [0 or 1]*/
```

指定されたファイルを開いて、pavec_file_out()にファイルポインタを渡す。

```
get_file_length( file, length )
char    *file;
int     *length;
```

stat()を使ってファイルの属性を獲得して、ファイルのバイト長を２で割った値をlengthに返す。

この関数は、もしstat()が失敗すると０を、stat()が成功すると１を、返す。

[サンプルプログラム（ｆｄａ）]

説明：

ファイルを指定された時間だけＤ／Ａする。

pavec_file_out(fp, freq, time, ch0, ch1)を使用している。

使用例：

％ｆｄａ　２０．０　３０００．０　１　０　／ＣＯＮＴＩＧ／Ａ．２０．ＡＤ

リスト：

```
/*
 *
D/A sample program
with uscing pavec_file_out( fp, freq, time, ch0, ch1 )
 */
# include       <stdio.h>

main( argc, argv )
   int  argc;
   char *argv[];
   {
        /* set parameter */
extern  double  atof();
        double  freq    = atof( argv[1] );
        double  time    = atof( argv[2] );
        char    ch0     = atoi( argv[3] );
        char    ch1     = atoi( argv[4] );
        char    *name   = argv[5];
        FILE    *fp, *fopen();
nice( -20 );
        if ( (fp = fopen( name, "r" )) != NULL )
           {    pavec_file_out( fp, freq, time, ch0, ch1 );
                fclose( fp );
           }
        else
                fprintf( stderr, "Can't open file %s !!¥n", name );
nice(  20 );
        exit( 0 );
   }
```

[サンプルプログラム（ｑａｄ）]

説明：
　指定された時間だけＡ／Ｄし、標準出力にデータを書きだす。
　pavec_qin()を使用している。
使用例：
％ｑａｄ　２０．０　３０００．０　１　０　＞　ＤＡＴＡ．ＡＤ
リスト：

```
/*
 *
A/D sample program
with useing pavec_qin()
 */
# include       <stdio.h>

main( argc, argv )
   int  argc;
   char *argv[];
   {
        /* set parameter */
extern  double  atof();
        double  freq    = atof( argv[1] );
        double  time    = atof( argv[2] );
        char    ch0     = atoi( argv[3] );
        char    ch1     = atoi( argv[4] );
        long    length;
nice( -20 );
        /* initialize PAVEC */
        if ( !pavec_init( freq, time, ch0, ch1, "r", &length ) )
                exit( -1 );
        /* read  for output  queued buffer */
        pavec_setQuedBuf( stdout );
        pavec_qin();
        pavec_stop();
nice(  20 );
        exit( 0 );
   }
```

[サンプルプログラム（ｑｄａ）]

説明：
　標準入力からデータを読みこんで、指定された時間だけＤ／Ａする。
　pavec_qout()を使用している。
使用例：
％ｑｄａ　２０．０　３０００．０　１　０　＜　／ＣＯＮＴＩＧ／Ａ．２０．ＡＤ
リスト：

```
/*
 *
D/A sample program
with useing pavec_qout()
 */
# include       <stdio.h>

main( argc, argv )
   int  argc;
   char *argv[];
   {
        /* set parameter */
extern  double  atof();
        double  freq    = atof( argv[1] );
        double  time    = atof( argv[2] );
        char    ch0     = atoi( argv[3] );
        char    ch1     = atoi( argv[4] );
        long    length;
nice( -20 );
        /* initialize PAVEC */
        if ( !pavec_init( freq, time, ch0, ch1, "w", &length ) )
                exit( -1 );
        /* qued D/A transfer */
        pavec_setQuedBuf( stdin );
        pavec_qout();
        pavec_stop();
nice(  20 );
        exit( 0 );
   }
```

［サンプルプログラム（ｘｄａ）］

説明：
　ファイルを指定された時間だけＤ／Ａする。
　pavec_xout()を使用している。
使用例：
％ｘｄａ　２０．０　３０００．０　１　０　／ＣＯＮＴＩＧ／Ａ．２０．ＡＤ
リスト：

```
/*
 *
D/A sample program
with useing pavec_xout( array, length )
 */
# include       <stdio.h>
# include       <fcntl.h>

main( argc, argv )
   int  argc;
   char *argv[];
   {
        /* set parameter */
extern  double  atof();
        double  freq    = atof( argv[1] );
        double  time    = atof( argv[2] );
        char    ch0     = atoi( argv[3] );
        char    ch1     = atoi( argv[4] );
        char    *name   = argv[5];
union
   {    short   *buf;
        int     dummy;
   }    data;
        char    *p, *malloc();
        long    length, arraysize;
        int     fd;
nice( -20 );
        /* initialize PAVEC */
        if ( !pavec_init( freq, time, ch0, ch1, "w", &length ) )
                exit( -1 );
        /* memory allocation */
        arraysize = length*sizeof(short);
        if ( (p = malloc( arraysize )) == NULL )
                exit( -1 );
        data.buf = (short *)p;
        /* read  for output  queued buffer */
        if ((fd = open( name, O_RDONLY )) < 0 )
           {    fprintf( stderr, "Can't open file %s !!¥n", name );
                exit( -1 );
           }
        else
        if ( read( fd, data.buf, arraysize ) <= 0)
```

```
            {     fprintf( stderr, "read  error!!¥n" );
                  close( fd );
                  exit( -1 );
            }
        /* D/A convert */
        pavec_xout( data.buf, length );
        pavec_stop();
nice(  20 );
        exit( 0 );
    }
```

3．3　応答スイッチの制御(libdacp.a)

3．3．1　使用例

```
# include       <stdio.h>
main()
  {
        int     response, pi16_get_key();
        pi16_init( 800.0 );     /* Initialize PI16, set CLK7 */
        while( (response = pi16_get_key( 1000 )) != 6 )
                printf( "response = %d¥n", response );
        pi16_close();
  }
```

応答スイッチは、PI16に接続されており、応答スイッチの制御は、PI16に対する入出力になる。

応答スイッチを使用するには、以下の手順が必要である。

①デバイス(PI16)をオープンし、クロック(CLK7)を設定する

②応答スイッチからの入力を得る

③デバイス(PI16,CLK7)をクローズする

①は、pi16_init(freq)［freq:標本化周波数（単位　Hz　）］によって行なう。応答スイッチを使うときは、クロックとして CLK7 を使用している。pi16_init(freq)は、初期化に成功すると１を、失敗すると０を返す。

②は、pi16_get_key(COUNT)［COUNT:待ち時間（単位　なし）］によって行なう。ここで、COUNTには適当な数値が設定されていれば良い。pi16_get_key(COUNT)は、スイッチが押されるとその数値（１から６）を返す。

③は、pi16_close()によって行なう。

3．4　プログラマブルアッテネータの制御(libdacp.a)

3．4．1　使用例

```
# include        <stdio.h>
main()
   {
         int      response, pi16_get_key();
         gpib_init( 1000 );        /* Initialize GPIB */
         gpib_att_controle( 1, 40.0 );
         gpib_close();
   }
```

プログラムブルアッテネータは、GPIBに接続されており、制御は、GPIBに対する入出力になる。

プログラムブルアッテネータを使用するには、以下の手順が必要である。

①デバイス(GPIB)をオープンする

②プログラムブルアッテネータに出力を送る

③デバイス(GPIB)をクローズする

①は、gpib_init(TIMEOUT)［TIMEOUT:タイムアウト（単位　msec　）］によって行なう。gpib_init(TIMEOUT)は、初期化に成功すると1を、失敗すると0を返す。

②は、gpib_att_controle(which, value)［which:0または1　value:アッテネータの値］によって行なう。

③は、gpib_close()によって行なう。

第４章　ライブラリ関数の説明

ｌｉｂｄａｃｐ．ａ、ｌｉｂｐａｖｅｃ．ａに含まれる関数を次の形式にしたがって説明する。

ＮＡＭＥ	
関数名	（ライブラリ名）
ＳＹＮＴＡＸ	
呼び出し形式	
ＡＲＧＵＭＥＮＴＳ	
引数の説明	
ＲＥＴＵＲＮＥＤ　ＶＡＬＵＥ	
戻り値	
ＤＥＳＣＲＩＰＴＩＯＮ	
関数の機能説明	
ＮＯＴＥＳ	
注意事項	
ＳＥＥ　ＡＬＳＯ	
参照すべき関数	

N A M E

g p i b _ a t t _ c o n t r o l e (libdacp.a)

S Y N T A X

```
int     address;
float   att_v;
gpib_att_controle( address, att_v );
```

A R G U M E N T S

address　　GPIBのアドレス［現在０、１に設定］

att_v　　アッテネータの設定値［単位 dB (0.0から99.9)］

R E T U R N E D　V A L U E

1:TRUE　　成功したとき

0:FALSE　　失敗したとき

D E S C R I P T I O N

アッテネータを制御する。

NAME

gpib_close (libdacp.a)

SYNTAX

```
gpib_close();
```

DESCRIPTION

GPIBをクローズする。

SEE ALSO

gpib_init

N A M E

g p i b _ i n i t (libdacp.a)

S Y N T A X

```
int     timeout;
gpib_init( timeout );
```

A R G U M E N T S

timeout タイムアウト

R E T U R N E D V A L U E

1:TRUE 成功したとき

0:FALSE 失敗したとき

D E S C R I P T I O N

GPIBの初期化を行う。

S E E A L S O

gpib_close

N A M E

p i l 6 _ c l o s e (libdacp.a)

S Y N T A X

```
pil6_close();
```

D E S C R I P T I O N

PI16をクローズする。

S E E A L S O

pil6_init

N A M E

p i l 6 _ g e t _ k e y (libdacp.a)

S Y N T A X

```
int     timeout;
pil6_get_key( timeout );
```

A R G U M E N T S

timeout タイムアウト

R E T U R N E D V A L U E

1から6までの数値で、押下された応答スイッチ上のボタン番号に対応している。

D E S C R I P T I O N

応答スイッチから入力を受けつける。

N A M E

p i l 6 _ i n i t (libdacp.a)

S Y N T A X

```
float   freq;
pil6_init( freq );
```

A R G U M E N T S

freq タイムアウト

R E T U R N E D V A L U E

1:TRUE 成功したとき

0:FALSE 失敗したとき

D E S C R I P T I O N

PI16の初期化を行う。

S E E A L S O

pil6_close

N A M E

p i 1 6 _ x i n (libdacp.a)

S Y N T A X

```
union
   {     short    *data;
         int      dummy;
   }     buffer;
int      nitem;
int      timeout;
pi16_xin( buffer.data, nitem, timeout );
```

A R G U M E N T S

buffer　　PI16から入力されたデータ
nitem　　取込む個数（必ず偶数でなければならない）
timeout　　タイムアウト

R E T U R N E D　V A L U E

1:TRUE　　成功したとき
0:FALSE　　失敗したとき

D E S C R I P T I O N

PI16からデータを取込む。

NAME

pavec_close (libpavec.a)

SYNTAX

```
pavec_close();
```

DESCRIPTION

PI16Fのクローズを行う。

SEE ALSO

pavec_stop,pavec_open

N A M E

p a v e c _ f r e e Q B u f (libpavec.a)

S Y N T A X

```
pavec_freeQBuf();
```

R E T U R N E D V A L U E

1:TRUE 常に

D E S C R I P T I O N

キュード バッファ用に割り当てられていたメモリーを解放する。

S E E A L S O

pavec_mallocQBuf,pavec_releaseQBuf

N A M E

p a v e c _ f i l e _ o u t (libpavec.a)

S Y N T A X

```
# include      <stdio.h>
FILE    *fp;
double  freq, duration;
char    ch0, ch1;
pavec_file_out( fp, freq, duration, ch0, ch1 );
```

A R G U M E N T S

fp	ファイル　ポインタ
freq	標本化周波数［単位 kHz］
duration	時間長［単位 msec］
ch0, ch1	チャネルの設定フラグ０または１

R E T U R N E D　V A L U E

1:TRUE	成功したとき
0:FALSE	失敗したとき

D E S C R I P T I O N

ファイルポインタで指定されたストーリームのデータをＤ／Ａ出力（アレイ・トランスファ）する。関数内部でパベックの初期化やメモリーのアロケーションを行なっている。最も簡便な、Ｄ／Ａ出力関数である。

また、２チャネルで出力するときは、データをチャネル分だけ複製する。

NAME

pavec_init (libpavec.a)

SYNTAX

```
double  freq, duration;
char    ch0, ch1;
char    *mode;
long    length;
pavec_init( freq, duration, ch0, ch1, mode, &length );
```

ARGUMENTS

freq	標本化周波数［単位 kHz］
duration	時間長［単位 msec］
ch0, ch1	チャネルの設定フラグ０または１
mode	Ａ／Ｄをおこなうときは"r"を、Ｄ／Ａのときは"w"を指定する
length	設定されたデータ長を返す

RETURNED VALUE

1:TRUE	成功したとき
0:FALSE	失敗したとき

DESCRIPTION

PI16Fのオープンとパベックの初期化等を行なう。pavec_file_out(fp, freq, time, ch0, ch1)以外の関数を使用する前に、必ず呼出さないといない。

SEE ALSO

pavec_set_paket,pavec_open,pavec_mode

N A M E

p a v e c _ m a l l o c Q B u f (libpavec.a)

S Y N T A X

```
pavec_mallocQBuf();
```

R E T U R N E D V A L U E

1:TRUE 成功したとき

0:FALSE 失敗したとき

D E S C R I P T I O N

キュード バッファ用にメモリーを割り当てる。

S E E A L S O

pavec_freeQBuf,pavec_releaseQBuf

NAME

pavec_mode (libpavec.a)

SYNTAX

```
char    *mode;
pavec_mode( mode );
```

ARGUMENTS

mode　　Ａ／Ｄをおこなうときは"r"を、Ｄ／Ａのときは"w"を指定する

RETURNED VALUE

1:TRUE　　成功したとき

0:FALSE　　失敗したとき

DESCRIPTION

ＰＡＶＥＣのモードを設定する。ＰＡＶＥＣのモードとは、Ａ／ＤかＤ／Ａというだけではなく、チャネル数や、標本化周波数も含まれる。

具体的には、この関数は、設定されたこれらのモードをＰＡＶＥＣにPI16Fを通して命令している。この関数を実行することによって、はじめて、ＰＡＶＥＣは、初期化されているのである。

pavec_init(freq, duration, ch0, ch1, mode, &length)は内部で、この関数を呼び出している。単独で使用する際は、事前にpavec_set_paket(freq, duration, ch0, ch1, mode, &length)を使用してライブラリー内部の変数を初期化しなければならない。

SEE ALSO

pavec_init,pavec_set_paket

N A M E

p a v e c _ _ o p e n (libpavec.a)

S Y N T A X

```
pavec_open();
```

R E T U R N E D V A L U E

1:TRUE 成功したとき

0:FALSE 失敗したとき

D E S C R I P T I O N

PI16Fのオープンを行う。

S E E A L S O

pavec_set_paket,pavec_close

NAME

pavec_qin (libpavec.a)

SYNTAX

```
pavec_qin();
```

RETURNED VALUE

1:TRUE　　成功したとき
0:FALSE　　失敗したとき

DESCRIPTION

キュード・A／Dトランスファを実行する。この関数を使用する前には、キュード　バッファがA／D用に設定されてないといけない。キュード　バッファを設定するにいは、pavec_setQuedBuf(fp)を使うか、または、pavec_mallocQBuf(),pavec_releaseQBuf(fp)を使えば良い。

SEE ALSO

pavec_setQuedBuf,pavec_mallocQBuf,pavec_releaseQBuf

NAME

pavec_qin_loop (libpavec.a)

SYNTAX

```
pavec_qin_loop();
```

RETURNED VALUE

1:TRUE　　成功したとき
0:FALSE　　失敗したとき

DESCRIPTION

キュード・A／Dトランスファを繰り返し実行する。この関数を使用する前には、キュード　バッファがA／D用に設定されてないといけない。キュード　バッファを設定するにいは、pavec_setQuedBuf(fp)を使うか、または、pavec_mallocQBuf(),pavec_releaseQBuf(fp)を使えば良い。

NOTES

この関数を停止するには、なんらかの割り込みでしか行なえない。子プロセスでこの関数を使用していて、かつ、その子プロセスが端末から制御が放れているならば、kill するしかない。

これらの場合、PAVECのCPUランプが点灯した状態のままになってしまうので、CPUランプを消したいならば、なんらかの方法で初期化しないといけない。

SEE ALSO

pavec_qin,pavec_setQuedBuf,pavec_mallocQBuf,pavec_releaseQBuf

NAME

pavec_qout (libpavec.a)

SYNTAX

```
pavec_qout();
```

RETURNED VALUE

1:TRUE 成功したとき
0:FALSE 失敗したとき

DESCRIPTION

キュード・D／Aトランスファを実行する。この関数を使用する前には、キュード バッファがD／A用に設定されてないといけない。キュード バッファを設定するにいは、pavec_setQuedBuf(fp)を使うか、または、pavec_mallocQBuf(),pavec_releaseQBuf(fp)を使えば良い。

SEE ALSO

pavec_setQuedBuf,pavec_mallocQBuf,pavec_releaseQBuf

NAME

pavec_relaseQBuf (libpavec.a)

SYNTAX

```
# include      <stdio.h>
FILE    *fp;
pavec_releaseQBuf( fp );
```

RETURNED VALUE

1:TRUE　　成功したとき
0:FALSE　　失敗したとき

DESCRIPTION

キュード・トランスファを行なう際のファイルポインタを指定する。
この関数が失敗する原因としては、D／Aのときであればファイルの読み込みエラーが考えられる。

SEE ALSO

pavec_freeQBuf,pavec_mallocQBuf

NAME

pavec_set_paket (libpavec.a)

SYNTAX

```
double  freq, duration;
char    ch0, ch1;
char    *mode;
long    length;
pavec_set_paket( freq, duration, ch0, ch1, mode, &length );
```

ARGUMENTS

freq	標本化周波数［単位 kHz］
duration	時間長［単位 msec］
ch0, ch1	チャネルの設定フラグ０または１
mode	Ａ／Ｄをおこなうときは"r"を、Ｄ／Ａのときは"w"を指定する
length	設定されたデータ長を返す

RETURNED VALUE

1:TRUE	成功したとき
0:FALSE	失敗したとき

DESCRIPTION

ライブラリー内部で使用している帯域変数等を初期化している。

pavec_init(freq, duration, ch0, ch1, mode, &length)は内部で、この関数を呼び出している。

SEE ALSO

pavec_init,pavec_mode

NAME

pavec_setQuedBuf　　(libpavec.a)

SYNTAX

```
# include      <stdio.h>
FILE   *fp;
pavec_setQuedBuf( fp );
```

RETURNED VALUE

1:TRUE　　成功したとき
0:FALSE　　失敗したとき

DESCRIPTION

キュード バッファを設定し、キュード・トランスファを行なう際のファイルポインタを指定する。

この関数が失敗する原因としては、メモリーが割り当てられなかったか、または、D／Aのときであればファイルの読み込みエラーが考えられる。

SEE ALSO

pavec_freeQBuf,pavec_mallocQBuf,pavec_releaseQBuf

NAME

pavec_stop (libpavec.a)

SYNTAX

```
pavec_stop();
```

RETURNED VALUE

1:TRUE 常に

DESCRIPTION

開いていたデバイス(PI16F)をクローズし、使用していたメモリーを解放する。pavec_file_out(fp, freq, time, ch0, ch1)以外の関数の使用後に、必ず呼出さないといけない。

SEE ALSO

pavec_close()

NAME

pavec_xin (libpavec.a)

SYNTAX

```
union
   {    short   *data;
        int     dummy;
   }    buffer;
long    length;
pavec_xin( buffer.data, length );
```

ARGUMENTS

buffer　　データ
length　　データ長

RETURNED VALUE

1:TRUE　　常に

DESCRIPTION

アレイ・A／Dトランスファを実行する。

NAME

pavec_xout (libpavec.a)

SYNTAX

```
union
   {    short   *data;
        int     dummy;
   }    buffer;
long    length;
pavec_xout( buffer.data, length );
```

ARGUMENTS

buffer　　データ

length　　データ長

RETURNED VALUE

1:TRUE　　常に

DESCRIPTION

アレイ・D／Aトランスファを実行する。

[付録：ソースリスト]

```
#
#-----------------------------------------
#       PAVEC Library instalation Makefile
#-----------------------------------------
#               S.Tenpaku(JAPAN)
#               7.June.1989
#-----------------------------------------
#
SoundLAB = /usr2/tis/work/SoundLAB
INC      = $(SoundLAB)/include
LIB      = ../lib
RTU4     = 4

#
SOURCES = da_XOUT.c damain.c pavec.c util.c pavec.h
OBJECTS = da_XOUT.o damain.o pavec.o util.o

#
#       set complie option
#
.c.o:
        cc -O -c -DRTU4=$(RTU4) -DSoundLAB='"$(SoundLAB)"' -I$(INC) $<

library: $(OBJECTS) \
        $(LIB)/libpavec1.a \
        $(LIB)/libpavec.a

install: library
        chown root *
        chmod 0644 *
        cd $(LIB);chown bin *.a;chmod 0644 *.a
        cd $(INC);chown bin *.h;chmod 0644 *.h
        make clean
#
$(LIB)/libpavec1.a: $(OBJECTS)
        rm -f  libpavec1.a
        ar q   libpavec1.a `lorder $(OBJECTS) | tsort 2> /dev/null`
        mv -f  libpavec1.a $(LIB)
        ranlib $(LIB)/libpavec1.a

$(LIB)/libpavec.a: $(OBJECTS)
        rm -f  libpavec.a
        ar q   libpavec.a util.o pavec.o
        mv -f  libpavec.a $(LIB)
        ranlib $(LIB)/libpavec.a

debug: main.o
        make library
        cc main.o ../lib/libpavec.a -o pavec -lmr -lm

clean:
        rm -f *.o *.bak *~ core

copy:
        cp $(SOURCES) $(DST)/PAVEC
        cp Makefile   $(DST)/PAVEC

# demo
demo: library
        rm -f *
#
# end-of-file ( PAVEC/Makefile )
#
```

```
/*
 *
 *              D/A ARRAY TRANSFER
-----------------------------------------------------------
 :- library libpavec1.a
 :- public  da_XOUT( fchan, nchans, freq, buffer, length )
 :_ useing  mouseShape(), errorMsg()
-----------------------------------------------------------
 *                      S.Tenpaku(JAPAN)
 *                      7.June.1989 re-write( for PAVEC )
-----------------------------------------------------------
 */
# include       <stdio.h>
# include       <mouse.h>

# define        MAX_CHANNELS    2       /* PAVEC max channels */
# define        TRUE    1
# define        FALSE   0

/*
 *
---------------------------------------------
        da_XOUT
---------------------------------------------
 */
da_XOUT( fchan, nchans, freq, data, length )
   int    fchan;         /* First channel to be sampled */
   int    nchans;        /* Number of channels to be sampled */
   double freq;          /* Clock frequency [kHz] */
   short  *data;         /* data */
   long   length;        /* Number of frames to be transferd */
/*
 * NOTE:
 *      If useing 2 channels, the buffer length must be 2*length.
 *
 *      channel 0 --> buffer{ 0, 2, 4, .... }
 *      channel 1 --> buffer{ 1, 3, 5, .... }
 */
   {
        double  time = (double)length/freq;
        char    ch0, ch1;

        /* caluculate frequency */
        if ( freq <= 0 )
           {    errorMsg( "ILLEGAL FREQUENCY FOR THE D/A TRANSFER" );
                return( FALSE );
           }
        if ( length <= 0 )
           {    errorMsg( "ILLEGAL LENGTH FOR THE D/A TRANSFER" );
                return( FALSE );
           }

        /*
         *      when uesing over 2 channels,
         *      set incr is 1
         */
        if ( nchans == 1 )
           {    if ( fchan == 0 )
                   {    ch0 = 1;
                        ch1 = 0;
                   }
                else
                   {    ch0 = 0;
                        ch1 = 1;
                   }
           }
        else
        if ( nchans == 2 )
           {    ch0 = ch1 = 1;
           }
        else
           {    errorMsg( "NON SUPORTED CHANNEL NUMBER ( FCHANS <= %d )", MAX_CHANNELS )
;
                return( FALSE );
           }

        /* change mouse shape */
nice( -20 );
        mouseShape( _SPEAKER );

        /* initialize PAVEC */
        if ( !pavec_init( freq, time, ch0, ch1, "w", &length ) )
           {    errorMsg( "PAVEC INITIALIZE FAILED !!" );
                return( FALSE );
```

```
            }

        /* D/A convert */
        pavec_xout( data, length );

        /* D/A stop */
        pavec_stop();

        /* all TRUE */
nice( 20 );
        return( TRUE );
    }


/* mrevwt */
mrcompleted( pathno )
    int  pathno;
    {
        int     endstatus = 1;
        int     i = 0;

        while( endstatus == 1 )
            {   mrevck( pathno, &endstatus );
                i++;
            }

        if ( endstatus != 2 )
            {   errorMsg( " Not Completed Transfer !" );
                return( FALSE );
            }
        else
                return( TRUE );
    }

/* end-of-file ( da_XOUT.c ) */
```

```c
/*
 *
 ----------------------------------------
        File D/A output program
 ----------------------------------------
 *
        Progrmed by S.Tenpaku(JAPAN)
                        22.June.1988
 */

# include       <stdio.h>

# define        TRUE    1
# define        FALSE   0

union
   {    short   *buf;   /* data array */
        int     dummy;  /* data baundary */
   }    data;

/*
 *
 ----------------------------------------
        daSmain
 ----------------------------------------
 */
daSmain( buf, freq, channel, length )
   short  *buf;         /* buffer of data */
   double freq;         /* sampling frequency [kHz] */
   int    channel;      /* D/A channel */
   int    length;       /* file length */
   {
        int     fchan   = 0;
        int     nchans  = 1;

        long    i, j;
        char    *p, *malloc();

        /* set channel */
        switch( channel )
           {
        case 0:
                fchan  = 0;
                nchans = 1;
                break;
        case 1:
                fchan  = 1;
                nchans = 1;
                break;
        case 2:
                fchan  = 0;
                nchans = 2;
                break;
        default:
                return( FALSE );
                break;
           }

        /* malloc */
        if ( ( p = malloc( length*2*nchans ) ) == NULL )
                return( FALSE );
        data.buf = (short *)p;

        /* copy buffer */
        if ( nchans == 1 )
           {    for ( i = j = 0 ; j < length ; i++)
                   {    data.buf[j++] = buf[i];
                   }
           }
        else
           {    for ( i = j = 0 ; i < length ; i++)
                   {    data.buf[j++] = buf[i];
                        data.buf[j++] = buf[i];
                   }
           }

        /* D/A */
        da_XOUT( fchan, nchans, freq, data.buf, length );
        free( p );

        /* done */
        return( TRUE );
   }
```

```
/*
-----------------------------------------------------------
        PAVEC A/D D/A converter library
-----------------------------------------------------------
                programed by S.Tenpaku (JAPAN)
                        27.April.1989
                         7.June .1989(Completed)
-----------------------------------------------------------
pavec_setQuedBuf( fp )
FILE *fp;
----------------------------------------
pavec_qout()
----------------------------------------
pavec_qin()
----------------------------------------
pavec_qin_loop()
----------------------------------------
pavec_xin( data, length )
short   *data;
long    length;
----------------------------------------
pavec_xout( data, length )
short   *data;
long    length;
----------------------------------------
pavec_init( freq, duration, ch0, ch1, mode, length )
double  freq;           sampling freqency [kHz]
double  duration;       data duration [msec]
char    ch0;            channel 0 flag [0:Not used or 1:Used]
char    ch1;            channel 1 flag [0:Not used or 1:Used]
char    *mode;          pavec used mode ["r":A/D or "w":D/A]
long    *length;        return PAVEC.length
----------------------------------------
pavec_stop()
----------------------------------------
 */
# include       "pavec.h"

/*
 *
-------------------------------
        ad_service
-------------------------------
A/D Service Routine of AST
 */
static
ad_service()
   {
        short   *retptr;
        int     size;

        /* Buffer Get from Ringbuffer */
        mrbufget( inpath, 0, &retptr );
        if ( PAVEC.flag )
           {    PAVEC.count += PAVEC.quesize;
                if ( PAVEC.count < PAVEC.length )
                        size = PAVEC.quesize;
                else
                if ( PAVEC.length < PAVEC.quesize )
                   {    size = PAVEC.length;
                        PAVEC.flag = 0;
                   }
                else
                   {    size = PAVEC.quesize - PAVEC.count + PAVEC.length;
                        PAVEC.flag = 0;
                   }
                /* write data */
                fwrite( (char *)retptr, sizeof(*retptr), size, PAVEC.fp );
           }
        /* Buffer Release */
        mrbufrel( inpath, retptr );
        return;
   }

/*
 *
-------------------------------
        da_service
-------------------------------
D/A Service Routine of AST
 */
static  int
da_service()
   {
```

```
/*
 *
 ---------------------------------------------
        daFmain
 ---------------------------------------------
 */
daFmain( buf, freq, channel, length )
   float  *buf;          /* buffer of data */
   double freq;          /* sampling frequency [kHz] */
   int    channel;       /* D/A channel */
   int    length;        /* file length */
   {
        int     fchan   = 0;
        int     nchans  = 1;

        long    i, j;
        char    *p, *malloc();

        /* set channel */
        switch( channel )
           {
        case 0:
                fchan  = 0;
                nchans = 1;
                break;
        case 1:
                fchan  = 1;
                nchans = 1;
                break;
        case 2:
                fchan  = 0;
                nchans = 2;
                break;
        default:
                return( FALSE );
                break;
           }

        /* malloc */
        if ( ( p = malloc( length*2*nchans ) ) == NULL )
                return( FALSE );
        data.buf = (short *)p;

        /* copy buffer */
        if ( nchans == 1 )
           {    for ( i = j = 0 ; j < length ; i++)
                   {    data.buf[j++] = buf[i];
                   }
           }
        else
           {    for ( i = j = 0 ; i < length ; i++)
                   {    data.buf[j++] = buf[i];
                        data.buf[j++] = buf[i];
                   }
           }

        /* D/A */
        da_XOUT( fchan, nchans, freq, data.buf, length );
        free( p );

        /* done */
        return( TRUE );
   }

/* end-of-file (damain.c) */
```

```
        short   *retptr;
        int     size;

        if ( PAVEC.flag )
           {    /* Buffer Get from Ringbuffer */
                mrbufget( outpath, 0, &retptr );

                PAVEC.count += PAVEC.quesize;
                if ( PAVEC.count < PAVEC.length )
                        size = PAVEC.quesize;
                else
                if ( PAVEC.length < PAVEC.quesize )
                   {    size = PAVEC.length;
                        PAVEC.flag = 0;
                   }
                else
                   {    size = PAVEC.quesize - PAVEC.count + PAVEC.length;
                        PAVEC.flag = 0;
                   }
                /* read data */
                if ( feof( PAVEC.fp ) )
                        PAVEC.flag = 0;
                else
                if ( fread( (char *)retptr, sizeof(*retptr), size, PAVEC.fp ) <= 0 )
                        PAVEC.flag = 0;
                else
                        mrbufrel( outpath, retptr );    /* Buffer Release */
           }
        return;
    }

/*
 *
-----------------------------
        pavec_qout
-----------------------------
 */
pavec_qout()
    {
        if ( PAVEC.mode )
           {    /* Do D/A service */
                mrpfomod( outpath,
                          ON,
                          ON,
                          ON,
                          OFF,
                          ON,
                          WRITE_FUNC,
                          END_FUNC      );

                /* scanning service */
                mrxoutq ( outpath,
                          PAVEC.quesize,
                          PAVEC.length,
                          da_service );

                /* 10 Sec Wait */
                mrevwt  ( outpath,
                          0,
                          RW_TIMEOUT );
                return( TRUE );
           }
        else
                return( FALSE );
    }

/*
 *
-----------------------------
        pavec_qin
-----------------------------
 */
pavec_qin()
    {
        if ( PAVEC.mode )
                return( FALSE );
        else
           {    /* Do A/D service */
                mrpfimod( inpath,
                          ON,
                          ON,
                          OFF,
                          READ_FUNC,
                          END_FUNC );
```

```
                /* service */
                mrxinq  ( inpath,
                          PAVEC.quesize,
                          PAVEC.length,
                          ad_service );

                /* 10 Sec Wait */
                mrevwt  ( inpath,
                          0,
                          RW_TIMEOUT );
                return( TRUE );
            }
    }

/*
 *
-----------------------------
        pavec_qin_loop
-----------------------------
 */
pavec_qin_loop()
   {
        if ( PAVEC.mode )
                return( FALSE );
        else
           {    /* Do A/D service */
                mrpfimod( inpath,
                          ON,
                          ON,
                          OFF,
                          READ_FUNC,
                          END_FUNC );

                /* loop forever */
                while( 1 )
                   {    /* service */
                        mrxinq  ( inpath,
                                  PAVEC.quesize,
                                  PAVEC.length,
                                  ad_service );
                        /* 10 Sec Wait */
                        mrevwt  ( inpath,
                                  0,
                                  RW_TIMEOUT );
                        /* init flag and count */
                        PAVEC.flag  = TRUE;
                        PAVEC.count = 0;
                   }
                return( TRUE );
            }
    }

/*
 *
--------------------------
        pavec_setQuedBuf
--------------------------
Set up QUED BUFFER.
 */
pavec_setQuedBuf( fp )
FILE *fp;
   {
        int     quesize ;
        char    *p, *malloc();

        /* initialize PAVEC paket */
        PAVEC.fp    = fp;
        PAVEC.flag  = TRUE;
        PAVEC.count = 0;
        if ( PAVEC.bufptr != (short *)-1 )
           {    free( (char *)PAVEC.bufptr );
                PAVEC.bufptr = (short *)-1;
           }

        /*   Set Size of File and Buffer   */
        quesize  = PAVEC.quesize*NBUFFS*sizeof(short);

        /*
         *      Queued buffer allocation
         */
        if ( (p = malloc( quesize )) == NULL )
                return( FALSE );
        PAVEC.bufptr = (short *)p;
```

```
        /*
         *      Buffer Release
         */
        if ( PAVEC.mode )
           {    /* for D/A */
                if ( fread( (char *)PAVEC.bufptr, sizeof(*PAVEC.bufptr), quesize, fp ) <
 -> = 0 )
260                {    fclose( fp );
                        return( FALSE );
                   }
                mrbufall( outpath,
                          PAVEC.bufptr,
                          NBUFFS,
                          PAVEC.quesize*2 );
           }
        else
           {    /* for A/D */
270             mrbufall( inpath,
                          PAVEC.bufptr,
                          NBUFFS,
                          PAVEC.quesize*2 );
           }
        /* done */
        return( TRUE );
    }

/*
 *
280 -------------------------
        pavec_freeQBuf
 -------------------------
 */
pavec_freeQBuf()
    {
        /* initialize PAVEC paket */
        if ( PAVEC.bufptr != (short *)-1 )
           {    free( (char *)PAVEC.bufptr );
290             PAVEC.bufptr = (short *)-1;
           }
        /* done */
        return( TRUE );
    }

/*
 *
 -------------------------
        pavec_mallocQBuf
300 -------------------------
 */
pavec_mallocQBuf()
    {
        int     quesize ;
        char    *p, *malloc();

        /* initialize PAVEC paket */
        if ( PAVEC.bufptr != (short *)-1 )
                return( TRUE );
310     /*   Set Size of File and Buffer   */
        quesize  = PAVEC.quesize*NBUFFS*sizeof(short);
        if ( (p = malloc( quesize )) == NULL )
                return( FALSE );
        /* done */
        PAVEC.bufptr = (short *)p;
        return( TRUE );
    }

/*
320 *
 -------------------------
        pavec_releaseQBuf
 -------------------------
 */
pavec_releaseQBuf( fp )
FILE *fp;
    {
        int     quesize  = PAVEC.quesize*NBUFFS*sizeof(short);

330     /* initialize PAVEC paket */
        PAVEC.fp    = fp;
        PAVEC.flag  = TRUE;
        PAVEC.count = 0;

        /*
```

```c
         *      Buffer Release
         */
        if ( PAVEC.mode )
           {    /* for D/A */
                if ( fread( (char *)PAVEC.bufptr, sizeof(*PAVEC.bufptr), quesize, fp ) <
-> = 0 )
                   {    fclose( fp );
                        return( FALSE );
                   }
                mrbufall( outpath,
                          PAVEC.bufptr,
                          NBUFFS,
                          PAVEC.quesize*2 );
           }
        else
           {    /* for A/D */
                mrbufall( inpath,
                          PAVEC.bufptr,
                          NBUFFS,
                          PAVEC.quesize*2 );
           }
        /* done */
        return( TRUE );
   }

/*
 *
-------------------------
        pavec_xin
-------------------------
array in transfer
 */
pavec_xin( data, length )
short   *data;
long    length;
   {
        /* Do A/D service */
        mrpfimod( inpath,
                  ON,
                  ON,
                  OFF,
                  READ_FUNC,
                  END_FUNC );

        /* ARRAY Transfer */
        mrpdxin(  inpath,
                  length,
                  data );

        /* 10 Sec Wait */
        mrevwt   ( inpath,
                  0,
                  RW_TIMEOUT );
        /* done */
        return( TRUE );
   }

/*
 *
-------------------------
        pavec_xout
-------------------------
array out transfer
 */
pavec_xout( data, length )
short   *data;
long    length;
   {
        /* Do D/A service */
        mrpfomod( outpath,
                  ON,
                  ON,
                  ON,
                  OFF,
                  ON,
                  WRITE_FUNC,
                  END_FUNC      );

        /* ARRAY Transfer */
        mrpdxout( outpath,
                  length,
                  data          );

        /* 10 Sec Wait */
```

```
        mrevwt  ( outpath,
                  0,
                  RW_TIMEOUT );
        /* done */
        return( TRUE );
    }

/*
 *
----------------------------------
        pavec_init
----------------------------------
 */
int
pavec_init( freq, duration, ch0, ch1, mode, length )
double  freq;           /* sampling freqency [kHz]      */
double  duration;       /* data duration [msec]         */
char    ch0;            /* channel 0 flag [0 or 1]      */
char    ch1;            /* channel 1 flag [0 or 1]      */
char    *mode;          /* pavec used mode              */
long    *length;        /* return PAVEC.length          */
    {
        if ( pavec_set_paket( freq, duration, ch0, ch1, length ) && /* set up paket */
             pavec_open() &&                                        /* open device */
             pavec_mode( mode )         )                           /* init device */
                return( TRUE );
        else
           {    pavec_stop();
                return( FALSE );
           }
   }

/*
 *
-------------------------
        pavec_stop
-------------------------
 */
int
pavec_stop()
    {
        int     sts;

        /* stop a transfer */
        if ( outpath < 0 && inpath < 0 )
                return( TRUE );
        mrstop( outpath, 1, &sts );
        mrstop( inpath,  1, &sts );

        /* reset PAVEC */
        pavec_control( INIT_DATA, INIT_FUNC );

        /* close all device */
        pavec_close();

        /*
         *      Free Memory Allocation
         *
        if ( PAVEC.bufptr != (short *)-1 )
           {    free( (char *)PAVEC.bufptr );
                PAVEC.bufptr = (short *)-1;
           }

        /*
         *      Close A/D-Converter
         */
        inpath  = -1;
        outpath = -1;

        /* done */
        return( TRUE );
    }

/*
 *
-------------------------
        pavec_open
-------------------------
 */
pavec_open()
    {
static  char    flag = TRUE;
        /* ---- init ---- */
        if ( flag )
```

```
            {     PAVEC.bufptr = (short *)-1;
                  flag = FALSE;
            }

        /* init path no */
        inpath  = -1;
        outpath = -1;

        /* open inpath */
        mropen  ( &inpath, PFI0, 0 );
        if ( inpath < 0 )
                return( FALSE );

        /* open outpath */
        mropen  ( &outpath, PFO0, 0 );
        if ( outpath < 0 )
            {   pavec_close();
                return( FALSE );
            }

        /* Initialize inbound and outbound transfer mode table */
        mrpfiinit( inpath );
        mrpfoinit( outpath );

        /* done */
        return( TRUE );
    }

/*
 *
----------------------------
        pavec_mode
----------------------------
 */
pavec_mode( mode )
register char   *mode;
    {
        int     tmpfreq ;
        char    Mode;

        /* check mode */
        if ( *mode == 'r' || *mode == 'a' || *mode == 'i' )
                Mode = PAVEC_AD_MODE;
        else
        if ( *mode == 'w' || *mode == 'd' || *mode == 'o' )
                Mode = PAVEC_DA_MODE;
        else
                return( FALSE );
        /*
         *      Converter init
         */
        if ( !pavec_control( INIT_DATA,         INIT_FUNC ) )
                return( FALSE );

        /*
         *      Mpx address x'sfer
         */
        if ( !pavec_control( MPX_XNS_DATA,      CONT_FUNC ) )
                return( FALSE );
        if ( !pavec_control( MPX_ADDRESS,       DATA_FUNC ) )
                return( FALSE );

        /*
         *      Pavec Paser Format Conversion (Sets to external Area)
         */
        tmpfreq = pavec_pv_conv();
        if ( !pavec_control( PACER_XNS_DATA,    CONT_FUNC ) )
                return( FALSE );
        if ( !pavec_control( tmpfreq,           DATA_FUNC ) )
                return( FALSE );

        /*
         *      A/D D/A data x'sfer mode setting
         */
        if ( !pavec_control( MODE_XNS_DATA,     CONT_FUNC ) )
                return( FALSE );
        if ( !pavec_control( AUTO_START_DATA,   DATA_FUNC ) )
                return( FALSE );
        PAVEC.mode = Mode;
        if ( PAVEC.mode )
            {   /**  D/A conveter write start   **/
                if ( !pavec_control( WRITE_ENB_DATA,    CONT_FUNC ) )
                        return( FALSE );
            }
```

```
        else
           {    /**  A/D conveter read start    **/
                if ( !pavec_control( READ_ENB_DATA,     CONT_FUNC ) )
                        return( FALSE );
           }

        /* doen */
        return( TRUE );
   }

/*
 *
----------------------------------
        pavec_control
----------------------------------
        Set PFO mode.
 */
static
pavec_control( data, func )
int     data;
int     func;
   {
static  union
   {    short   outbuf[BUFF_SIZE];      /* for control word */
        int     outdum[1];
   } control;
register int    i;

        /* set table */
        for ( i = 0 ; i < 4 ; i++ )
                control.outbuf[i] = data;

        /* out the control word */
        mrpfomod( outpath,
                  ON,
                  ON,
                  ON,
                  OFF,
                  ON,
                  func,
                  END_FUNC );
        mrpdxout( outpath,
                  BUFF_SIZE,
                  control.outbuf );

        /* wait for event */
        if ( !mrevwait( outpath, TIMEOUT ) )
           {    pavec_close();
                return( FALSE );
           }
        else
                return( TRUE );
   }

/*
 *
----------------------------------
        pavec_set_paket
----------------------------------
 */
pavec_set_paket( freq, duration, ch0, ch1, length )
double  freq;           /* sampling frequency [kHz]     */
double  duration;       /* data duration [msec]         */
char    ch0, ch1;       /* channel flag TRUE or FALSE   */
long    *length;        /* return PAVEC.length          */
   {
        /*   set frequency   */
        if ( freq < MINFREQ || freq > MAXFREQ )
                return( FALSE );
        PAVEC.freq = freq * 1000.0;     /* [kHz] => [Hz] */

        /* set duration */
        if ( duration <= 0 )
                return( FALSE );
        if ( duration > RW_TIMEOUT )
                return( FALSE );
        PAVEC.duration = duration/1000.0;       /* [msec] => [sec] */

        /* set channel */
        if ( ch0 )
           {    if ( ch1 )
                   {    PAVEC.nchan = 2;
                        MPX_ADDRESS = MPX_ADRS_2CH;
                   }
```

```
                else
                   {    PAVEC.nchan = 1;
                        MPX_ADDRESS = MPX_ADRS_CH0;
                   }
            }
         else
            {    if ( ch1 )
                   {    PAVEC.nchan = 1;
                        MPX_ADDRESS = MPX_ADRS_CH1;
                   }
                else
                        return( FALSE );
            }
         /* set length */
         PAVEC.length = PAVEC.freq*PAVEC.duration*PAVEC.nchan;
         /* set quesize */
         PAVEC.quesize = PAVEC.freq*PAVEC.nchan;
         /* done */
         *length = PAVEC.length;
         return( TRUE );
    }

/*
 *
----------------------------------
         pavec_pv_conv
----------------------------------
         For PAVEC.
 */
static  int
pavec_pv_conv()
    {
         unsigned long   tmp0,tmp1,tmp2;

         tmp0 = 10000000 / PAVEC.freq;

         tmp1 = tmp0 & MASK00FF;
         tmp2 = tmp0 << 4;
         tmp2 = tmp2 & MASKF000;
         tmp0 = tmp1 | tmp2;

         return ( (int)tmp0 );
    }

/*
 *
---------------------------
         pavec_close
---------------------------
 */
pavec_close()
    {
         if ( outpath > 0 )
                mrclose( outpath );
         if ( inpath > 0 )
                mrclose( inpath );
         inpath = outpath = -1;
    }

/*
 *
---------------------------
         mrevwait
---------------------------
 */
static
mrevwait( pathno, wtime )
    int  pathno; /* DACP path number */
    int  wtime;  /* wait loop number */
/*
NOTE:
wtime's unit is not time.
It is maximum loop counter.
*/
    {
register int    i = 0;
         int    status = -1;

         while( status != 2 )
            {    if ( i > wtime )
                        return( FALSE );        /* failure */
                else
                        mrevck( pathno, &status );
                i++;
```

```
            }
        /* success */
        return( TRUE );
    }

/* end-of-file (pavec.c) */
```

```
/*
 *      Required include files
 */
# include       <stdio.h>
# include       <mr.h>
# include       <fcntl.h>

/*
 *      PAVEC 16-BIT A/D D/A CONVERTER
 *      Control function define table
 */
# define        MINFREQ         4               /* A/D sample minimum frequency [kHz] */
# define        MAXFREQ         100             /* A/D sample maximum frequency [kHz] */
# define        MAX_FRAME       1000000         /* Maximan Frame Length [sec] */

# define        ON              1               /* pi16f contril bit on */
# define        OFF             0               /* pi16f contril bit off */
# define        BUFF_SIZE       4               /* Output buffer size for pi16f */
# define        TIMEOUT         2000            /* Time out for 2-sec */
# define        RW_TIMEOUT      1000000         /* Time out 1000-sec for I/O */

# define        MPX_ADRS_CH0    0x0000          /* Mpx address select 0-ch */
# define        MPX_ADRS_CH1    0x0101          /* Mpx address select 1-ch */
# define        MPX_ADRS_2CH    0x0100          /* Mpx address select 0/1-ch */
# define        AUTO_START_DATA 0x0001          /* Internal pacer & auto start */

# define        DATA_FUNC       0               /* FN=0 for data word x'sfer */
# define        CONT_FUNC       1               /* FN=1 for control word x'sfer */
# define        INIT_FUNC       3               /* FN=3 for init control word x'sfer */
# define        READ_FUNC       4               /* FN=4 for data read start FN-bit */
# define        WRITE_FUNC      0               /* FN=0 for data word x'sfer */
# define        END_FUNC        0               /* Reset all FN-bit mask */

# define        INIT_DATA       0x0052          /* Initilyze control word */
# define        MODE_XNS_DATA   0x0052          /* Mode x'nsfer control word */
# define        MPX_XNS_DATA    0x0010          /* Mpx adrs x'nsfer control word */
# define        PACER_XNS_DATA  0x0011          /* Pacer x'nsfer control word */
# define        READ_ENB_DATA   0x0080          /* Data read  enable control word */
# define        WRITE_ENB_DATA  0x0000          /* Data write enable control word */

/*
 *      Frequency-control-code of PAVEC
 */
# define        MASK00FF        0x00ff          /* mask for creat freq */
# define        MASKF000        0xf000          /* mask for creat freq */
# define        MASK0100        0x0100          /* mask of freq-base 1 micro-sec */

/*
 *      Device Name
 */
# define        PFI0    "/dev/dacp0/pfi0"       /* pi16f input device name  */
# define        PFO0    "/dev/dacp0/pfo0"       /* pi16f output device name */
# define        CK10    "/dev/dacp0/clk0"       /* Clock module "ck10" */

# define        NBUFFS          6
# define        NEAREST         0
# define        SQUARE          4
# define        WIDTH           0.0
# define        BLKSIZE         4096

/*
 *      A/D or D/A control
 */
# define        PAVEC_AD_MODE   0       /* used for A/D converter       */
# define        PAVEC_DA_MODE   1       /* used for D/A converter       */

/*
 *      Define  "UNIONS" & "STRUCTS"
 */
static  struct
    {   char    mode;           /* A/D or D/A mode              */
        FILE    *fp;            /* file pointer                 */
        int     nchan;          /* sample number of channel     */
        double  freq;           /* sampling frequency           */
        double  duration;       /* data duration [msec]         */
        long    length;         /* sample length                */
        short   *bufptr;        /* ring buffer pointer          */
        int     quesize;        /* Qued Buffer Size             */
        char    flag;           /* Qued Buffer flag             */
        int     count;          /* Qued Buffer count            */
    }   PAVEC;
```

```
| static  int     inpath ;                /* PI16F Input-bound path-no.   */
| static  int     outpath;                /* PI16F Output-bound path-no.  */
| static  int     clkpath;                /* CK10  path-no.               */
|
| /*
|  *
|  */
90| static  int     MPX_ADDRESS;
|
| /* end-of-file (pavec.h) */
```

```
/*
-----------------------------------------------------------
        PAVEC A/D D/A converter Utilities
-----------------------------------------------------------
                programed by S.Tenpaku (JAPAN)
                        7.June.1989
-----------------------------------------------------------
 */
# include       <stdio.h>

# define        TRUE    1
# define        FALSE   0

/*
 *
-------------------------------------------
        pavec_file_out
-------------------------------------------
 */
pavec_file_out( fp, freq, duration, ch0, ch1 )
FILE    *fp;            /* FILE pointer                 */
double  freq;           /* sampling frequency [kHz]     */
double  duration;       /* data duration [msec]         */
char    ch0;            /* channel 0 flag [0 or 1]      */
char    ch1;            /* channel 1 flag [0 or 1]      */
   {
union
   {    short   *buf;   /* data array */
        int     dummy;  /* data baundary */
   }    data;
        short   one[1];
        long    i, length;
        int     nchans  = ch0 + ch1;
        char    *p, *malloc();

        /* malloc */
        length = freq*duration*nchans;
        if ( ( p = malloc( length*sizeof(short) ) ) == NULL )
                return( FALSE );
        data.buf = (short *)p;

        /* copy buffer */
        i = 0;
        if ( nchans == 1 )
           {    while( !feof( fp ) )
                   {    if ( fread( (char *)one, sizeof(*one), 1, fp ) <= 0 )
                                break;
                        else
                        if ( i < length )
                                data.buf[i++] = one[0];
                        else
                                break;
                   }
           }
        else
           {    while( !feof( fp ) )
                   {    if ( fread( (char *)one, sizeof(*one), 1, fp ) <= 0 )
                                break;
                        else
                        if ( i < length )
                           {    data.buf[i++] = one[0];
                                data.buf[i++] = one[0];
                           }
                        else
                                break;
                   }
           }
        /* initialize PAVEC */
        duration = (double)i/freq/nchans;
        if ( !pavec_init( freq, duration, ch0, ch1, "w", &length ) )
           {    free( p );
                return( FALSE );
           }
        /* D/A convert */
        pavec_xout( data.buf, length );

        /* D/A stop */
        pavec_stop();
        free( p );
        return( TRUE );
   }

/* end-of-file( util.c ) */
```

```
#
#-----------------------------------------
#       DACP Library instalation Makefile
#-----------------------------------------
#               S.Tenpaku(JAPAN)
#               26.June.1989
#-----------------------------------------
#
LIB      = ../lib
LIBNAME  = libdacp.a
#
SOURCES = pil6.c gpib.c timer.c
OBJECTS = pil6.o gpib.o timer.o

#
#       set complie option
#
.c.o:
        cc -O -c $<

$(LIB)/$(LIBNAME): $(OBJECTS)
        rm -f  $(LIBNAME)
        ar q   $(LIBNAME) `lorder $(OBJECTS) | tsort 2> /dev/null`
        mv -f  $(LIBNAME) $(LIB)
        ranlib $(LIB)/$(LIBNAME)

install: $(LIB)/$(LIBNAME)
        chown root *
        chmod 0644 *
        cd $(LIB);chown bin *.a;chmod 0644 *.a
        cd $(INC);chown bin *.h;chmod 0644 *.h
        make clean
#
debug: main.o
        make library
        cc main.o ../lib/libpavec.a -o pavec -lmr -lm

clean:
        rm -f *.o *.bak *~ core

copy:
        cp $(SOURCES) $(DST)/DACQ
        cp Makefile   $(DST)/DACQ

# demo
demo: library
        rm -f *
#
# end-of-file ( PAVEC/Makefile )
#
```

```
# include       <stdio.h>

# define        TRUE    1
# define        FALSE   0

# define        GPIB    "/dev/dacp0/gpib0"
static  int     gpib = -1;

/*
 *
-------------------------------
        gpib_init
-------------------------------
 */
gpib_init( wtime )
int     wtime;
   {
        mribopen( &gpib, GPIB );
        if ( gpib < 0 )
                return( FALSE );
        mribsettimout( gpib, wtime );
        mribenbremall( gpib );
        return( TRUE );
   }

/*
 *
-------------------------------
        gpib_close
-------------------------------
 */
gpib_close()
   {    if ( gpib > 0 )
           {    mribclose( gpib );
                gpib = -1;
           }
   }

/*
 *
-------------------------------
        gpib_att_controle
-------------------------------
Attenuator Controling Routine
 */
gpib_att_controle( address, att_v )
int     address;
float   att_v;
   {
        int     ndevs   = 1;
        int     lenstr  = 6;
static  int     addr[2] = {10, 11};
        char    att_value[10];
        char    msg[10];

        if ( address == 0 || address == 1 )
           {    sprintf( att_value, "%4.1f", att_v);
                msg[0] = att_value[0];
                msg[1] = att_value[1];
                msg[2] = att_value[2];
                msg[3] = att_value[3];
                msg[4] = 0x0d;
                msg[5] = 0x0a;
                msg[6] = 0;
                mribsend( gpib, msg, lenstr, ndevs, &addr[address] );
                return( TRUE );
           }
        else
                return( FALSE );
   }

/* end-of-file( gpib.c ) */
```

```
# include       <stdio.h>
/*# include     <mr.h>*/

# define        PI16            "/dev/dacp0/pdi0"
# define        CLK7            "/dev/dacp0/clk7"
# define        NEAREST         0       /*  clock parameters  */
# define        SQUARE          4
# define        WIDTH           0.0

# define        TRUE    1
# define        FALSE   0

static  int     pi16 = -1;
static  int     clk7 = -1;

/*
 *
----------------------------
        pi16_init
----------------------------
 */
pi16_init( freq )
float   freq;
   {
        double  ret_freq, ret_width;

        mropen( &pi16, PI16, 0 );               /* initialize PI16 */
        if ( pi16 < 0 )
                return( FALSE );
        mrclkopn(       &clk7, CLK7, 0 );       /*  set clk7 to PI16  */
        if ( clk7 < 0 )
                return( FALSE );
        mrclk1( clk7, NEAREST, freq, &ret_freq, SQUARE, WIDTH, &ret_width, 0 );
        mrclktrig( pi16, 1, clk7 );
        return( TRUE );
   }

/*
 *
----------------------------
        pi16_close
----------------------------
 */
pi16_close( freq )
   {
        if ( pi16 > 0 );
           {    mrclose( pi16 );
                pi16 = -1;
           }
        if ( clk7 > 0 )
           {    mrclose( clk7 );
                clk7 = -1;
           }
   }

/*
 *
---------------------------------
        pi16_xin
---------------------------------
 */
pi16_xin( array, nitem, maxcount )
short   *array;
int     nitem;
int     maxcount;
   {
register int count = 0;
        int evstat;
        int status;

        mrpdxin( pi16, nitem, array );
        do {    mrevck( pi16, &evstat );
                if ( count > maxcount )
                   {    mrstop( pi16, 1, &status );
                        return( FALSE );
                   }
                count++;
           } while( evstat == 1 );
        return( TRUE );
   }

/*
 *
----------------------------------
```

```
        pi16_get_key
-----------------------------------
Getting Response from Switch Box (with 6 button)
 */
pi16_get_key( wcount )
int     wcount;
   {
static  union
           {    short   array[2];
                int     dummy;
           }    buffer;
        int     key;
register int    i;

        /* turned on key */
        do {    pi16_xin( buffer.array, 2, wcount );
                key = -((buffer.array[0] >> 8) + 1);
           } while( key == 0 ); /*  no response  */
        for ( i = 0 ; key != 0 ; i++ )
                key >>= 1;

        /* turned off key */
        do {    pi16_xin( buffer.array, 2, wcount );
                key = -((buffer.array[0] >> 8) + 1);
           } while( key != 0 ); /*  response  */

        /* done */
        return( i );
  }

/* end-of-file( pi16.c ) */
```

```
#include <sys/time.h>
#include <sys/types.h>
#include <sys/stat.h>

static  struct timeval tp;
static  struct timezone tzp;

/*
 *
------------------------
        timer
------------------------
'timer()' makes 'waiting time'.
 */
timer(wait_sec, wait_usec)
long    wait_sec, wait_usec;
    {
        long end_sec, end_usec;
        get_time();
        end_sec         = tp.tv_sec + wait_sec;
        end_usec        = tp.tv_usec + wait_usec;
        if ( end_usec >= 1000000 )
           {    end_sec = end_sec + 1;
                end_usec = end_usec - 1000000;
           }
        do {    get_time();
           } while ((end_sec - tp.tv_sec) > 0);
        do {    get_time();
           } while ((end_sec == tp.tv_sec) && (end_usec - tp.tv_usec) > 0);
    }

static
get_time()
   {
        int t;
        t = gettimeofday( &tp, &tzp );
        if ( t != 0 )   printf( "Time read ERROR!!\n" );
   }

/* end-of-file( timer.c ) */
```