【非公開】

No.

TR－A－0042

マルチＤＳＰで構成する多チャンネル蝸牛フィルタバンクの試作

平原　達也　　駒木根　隆士

Tatsuya Hirahara　Takashi Komakine

006

1988. 12. 2

ＡＴＲ視聴覚機構研究所

マルチＤＳＰで構成する多チャンネル蝸牛フィルタバンクの試作

昭和６３年１１月３０日

平原　達也　　駒木根　隆士

ＡＴＲ視聴覚機構研究所

目次

1. はじめに

聴覚系の音響信号に対するスペクトル分析機能をモデル化することによって、ヒトの聴覚特性を反映させた音声分析処理系を構築するための第一ステップとして、蝸牛における周波数分析モデルの一つであるカスケードパラレル型の線形蝸牛フィルタの設計とパラメータの最適化について検討を進めてきた。[1]-[7]

これまでの検討は総て汎用ワークステイション（Masscomp MC5600）上で行ってきたが、たかだか60チャンネルのフィルタリングに対して実時間の約百倍の処理時間が必要であった。プログラムをミニスーパーコンピュータ等に移植して走らせることによって処理速度は向上するが、チャンネル数を1桁ないし2桁上げようとした場合には、同じ問題が生じる。従って、数百から数千チャンネルの蝸牛フィルタバンクを構築するためには専用のハードウエアが不可欠である。

それが実現されることによって生じるメリットとしては次のようなものがある。まず、チャンネル数増加のメリットの一つは周波数分解能が上がることである。また、FFTでは線形な周波数軸上で等間隔なスペクトルしか得られないのに対して、フィルタバンクでは、対数周波数軸やBark周波数軸上で等間隔なスペクトルも得ることができる利点もある。その結果、側抑制などのチャンネル間に渡る相互作用の導入をより連続的に実現できるという利点もある。

従来、この種のフィルタバンクは主に音声の自動認識装置用の特徴パラメータ抽出のために用いられてきているが、東工大の原田らによる96チャンネルのシステム[9]以外ではそのチャンネル数はたかだか数十チャンネルどまりである。音声認識用の特徴パラメータを考えれば、チャンネル数の増加はデータ処理量の増加をもたらし好ましい方向ではないが、側抑制処理などを導入して特徴ベクトルの情報圧縮などを行えば[8]大きな問題とはならない。

多チャンネル聴覚フィルタのその他の用途としては、①生理モデルの検証、②心理物理モデルの検証、③音声知覚モデルへの応用などがあるが、後に述べるように生理データからみれば約2万チャンネル、心理物理的な周波数分解能からみれば約1500チャンネルが必要となり、本格的な応用はいまだに成されていない。

本稿では、多チャンネルの実時間蝸牛フィルタリング処理を実現するために試作した第一次システムのハードウエア、制御ソフトウエア、ホストとのインターフェイスソフトウエアについてその詳細を述べるとともに、今回の試作で明らかになった問題点等について言及する。具体的な蝸牛フィルタの設計法については別稿で詳細に述べているのでそれらを参考にされたい。

2. 試作システムの概要

2-1 実時間フィルタリングの検討

実時間でのフィルタリング処理を実現するために考慮すべき点は、基底膜上の進行波のスイープ時間及びフィルタのチャンネル数の二点である。

まず、基底膜上に生じる進行波が前庭窓から蝸牛孔近傍にまで達するまでのスイープ時間を考慮して、ある入力音声サンプルに対する分析出力の最大遅延時間τ_{dmax}は5msec.とした。[9]

一方、チャンネル数nについては、生理学的なデータに基づく内有毛細胞数に対応させるとn=10,000～30,000、心理物理的に測定した純音周波数のDL(Difference Limen)に対応させるとn=1500となる。

現時点においては、何チャンネルのフィルタバンクを実現するかは確定していないが、それぞれの目的に応じて音声認識装置用のフロントエンドとしては数十チャンネル、心理物理モデル用としては数百チャンネル、生理モデル用としては数千チャンネルのフィルタリングが必要である。そこで、実時間処理が実現可能な最大チャンネル数を見積ることにする。

Fig.1 に示す様なカスケードパラレル型のフィルタバンクを考えた場合、 nチャンネルの分析を系全体の遅延時間τ_d 以内で行うために要求される各ノッチフィルタにおける遅延時間τ_{Notch} は次式の条件を満たす必要がある。

$$\tau_{Notch} \leqq \tau_d / n \qquad (1)$$

すなわち、チャンネル数の増加にともなってτ_dが増加するので、τ_{Notch}がシステム内部のサンプリング周期fsの逆数と一致させる場合、$\tau_d \leqq \tau_{dmax}$ とするためにはfsを高くとる必要がある。

次に、Fig.1 に示すカスケードパラレル型のフィルタバンクをFig.2 に示すようなハード.ウエアシステムとして実現する場合を考える。 1つのＤＳＰユニットが内部サンプリング周期($1/f_s$)以内に分析可能なチャンネル数をN_Lとすると、遅延時間τ_d以内でnチャンネルを分析するのに必要なＤＳＰユニットの数N_uは

$$N_u = (n/N_L) \qquad (2)$$

となる。ここで、ＤＳＰのサイクルタイムをt_c、フィルタリングに必要なサイクル数をN_cとすると、N_Lは

$$N_L = (\tau_d/n)/(N_c \cdot t_c) \qquad (3)$$

と表せる。ここで、τ_d、N_c、t_cは定数であるから、定められたN_uに対して最大のチャンネル数を確保するためには$N_L=1$ とすればよい。このとき、nは、

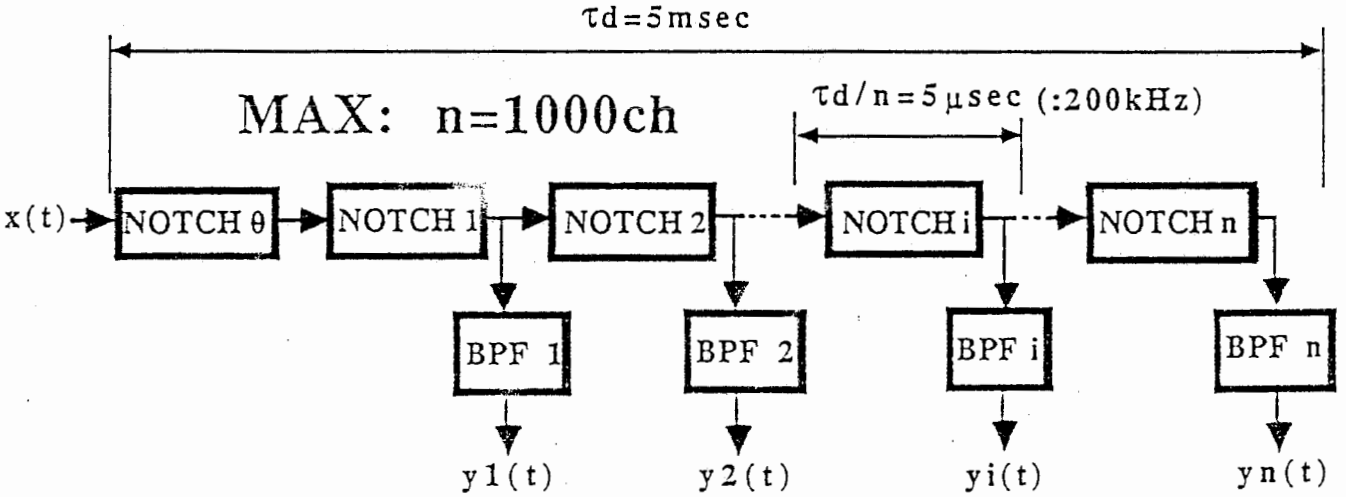
$$n = \tau_d/(N_c \cdot t_c) \qquad (4)$$

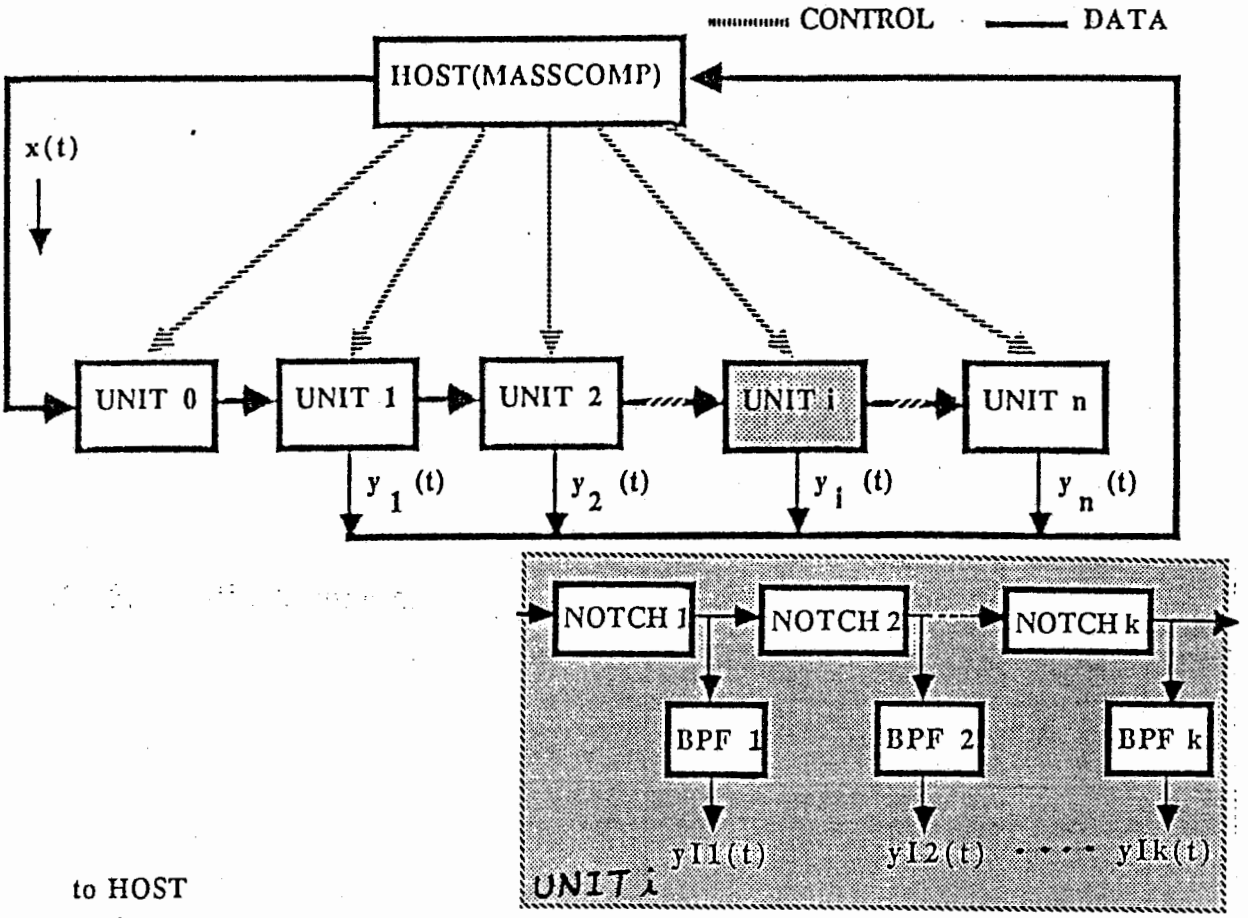
であり、サンプリング周波数としては

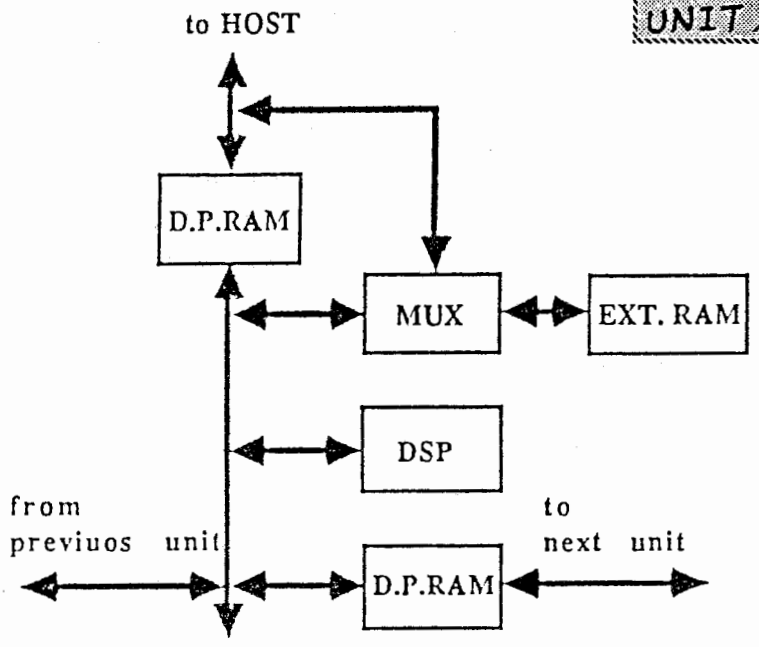
$$fs = n/\tau_d = 1/(N_L \cdot N_c \cdot t_c) = 1/(N_c \cdot t_c) \qquad (5)$$

以上のものが要求される。

具体的な値として、$N_c=30$、$t_c=150$nsec、$\tau_d=5$mseとすると、(4)式より、n=1,111となり、最大約1,100 チャンネルの実時間分析が可能であることがわかる。ただし、この場合、(5)式よりfs=220kHz が必要である。逆に、fs=20kHzとした場合には、(5)式よりn=fs・τ_d=100となる。そして、(3)式よりN_L=11.1が得られ、必要なユニット数N_uは9 となる。







DSP: μ PD77230
(32-bit floating point type)

Instruction cycle: 150nsec.

Instruction memory: 8kW

Host-DSP communication:
dual-port RAM

DSP-DSP communication:
dual-port RAM

Power supply: 5V 0.2A /unit

FIG. 2 DSP unit block diagram and specifications.

２－２　試作システムの概要

前節に示したように、マルチＤＳＰシステムで最大1,000チャンネル程度のフィルタバンクの実現が可能であることが明らかになった。そこで、試作システムの第１バージョンとして、カスケードパラレル型の多チャンネル線形蝸牛フィルタバンクを実現することにした。

Fig.2に示されるように、各ユニットは、ＤＳＰ、ホストマシンとのインターフェイスに用いる高速インストラクションメモリ、ＤＳＰ間のインターフェイス等によって構成される。本システムの物理構成は、論理構成の変更に対しても柔軟に対応できるように、各ユニットが相互に従属接続するとともに、ホストに対しては星状に接続させている。この様な物理構成は、今回必要とする直列的なデータ処理だけではなく、並列的な処理も可能としている。たとえば、入力信号を各ユニットに同時に与えることによって、並列型のフィルタバンクも容易に実現可能である。また、直列側の接続は双方向のデータの授受が可能であり、隣接チャンネル間の相互作用等の機能を付加させることにも対応できる。

ホストマシンとしては、聴覚研究室で用いられているワークステイションの一つであるMasscomp社のMC5600を用いることとし、インターフェイスはMasscompの高速パラレルインターフェイスPI16Fを介して行うこととした。

プロセッサは、汎用のディジタル・シグナル・プロセッサ（ＤＳＰ）を用いることとした。これは、32bitの浮動小数点データが扱え、処理速度も150nsec/Instructionと高速であること、プログラミングの支援ツールが整っていること、周辺デバイスが整備されているためにハードウエアの実現性も容易であること等の理由による。プロセッサの候補としてデータフロー型プロセッサの使用も検討したが、現段階では開発環境が整備されていないために、今回は採用を見送った。表１に両者のプロセッサを比較検討した結果を参考のために示す。

項目	ＤＳＰ （μPD77230）	データフロー型プロセッサ （μPD7281）
データ形式	浮動小数点　32bit	整数　16bit
処理速度	150ns/Instruction	200ns/instruction
biquad-filter	15 Instruction	20 Instruction
プログラミング	アセンブラ	フローグラフ、アセンブラ
〃　の容易さ	普通	慣れれば容易
アプリケーション	多い	少ない
開発システム	ＶＡＸ、ＰＣ９８	ＰＣ９８
ハードウエア	普通	小規模ならば容易
〃　の拡張性	普通	やや難しい
新規性	中（Multi-Cascade）	大（Data-Flow,Multi-CPU）
Hostとの整合性	容易	やや難しい
主な応用分野	音声、低周波信号	画像、並列信号処理

表１　ＤＳＰとデータフロー型プロセッサの比較

3．ハードウエアの仕様

本章では試作したシステムのハードウエア仕様について述べる。

3－1　ホストマシン

ホストマシンはマスコンプ社のMC5600シリーズとし、DACPに接続される汎用高速パラレルポートのPI16Fを使用する。

3－2　ＤＳＰ

NECのμPD77230をマスタモードで使用する。マスクされた内部ルーチンとホストからダウンロードされる制御ルーチンによりフィルタリング処理を実行させる。

μPD77230ではフィルタリングに必要なbi-quad演算を15インストラクションサイクルで実行する。すなわち、サイクルタイム150nsecの場合では、150*15=2250nsecの演算時間を必要とする。したがって、20KHzサンプリングのデータを実時間で処理する場合には

$$1/(20*10^{3})/(2250*10^{-9})=22.2$$

より、22回のbi-quad演算が実行できる。

3－3　マルチDSP接続方式

各DSPの間は縦続接続とし、ホストとDSPシステムの間は星状接続とする。

3－4　最大DSP接続数

DSPアドレスは15ビットで与えられるために、32768(=2^{15})までアドレッシングできる。チャンネル数の変更を容易にするために、各DSPのアドレスの設定は、ハードウエアスイッチによる他、初期化プログラムによる自動設定を可能とする。

3－5　DSP制御方式および外部プログラムメモリ

制御プログラムはホストCPUによりダウンロードし、アップロードは行なわない。すなわち、プログラム領域へのアクセスは、ホストからDSPへの単方向とする。また、DSP間のデータ転送に関する同期は、割り込みにより行なう。

プログラムメモリは4Kワードx32ビットで、書き込みは16ビットバス(16ビットx2→32)を用いて、DSPのHALT状態時にDMAにより行なう。また、書き込みモードは、対象のDSPユニットが全ユニット同時か、または特定の一つのユニットかの、いずれかのモードを指定できるようにする。

3－6　ホスト対DSPシステム通信

FIFO機能を有するデュアルポートメモリにより、ホストとDSPの間の双方向のデータ授受を行なう。メモリ容量は16ビットx512ワードで、DSP動作中でもバス衝突無しにアクセス可能である。

3－7　DSP間のインターフェイス

16ビットレジスタで行う。データの方向はDSPアドレスの小さい方から大きい方へ向かう一方向とする。DSP同志は同期して動作するのでFIFO機能はもたせない。

３－８　割り込み

フィルタ動作の同期は割り込みにより行なう。すなわち、DSP間i/fに設定された前段からの入力データは、ユニットに対してかけられる割り込み信号によりフィルタリングを行なわせる。結果は、次段のユニットへの計算結果がDSP間i/fに出力させ、また、そのユニットのフィルタリング結果がホスト通信i/fのメモリに設定させる。その後、DSPは内部ROMの待ちルーチンへジャンプして、次の割り込みまでループさせる。このとき、DSPはバスを解放している。

ホストは、先頭からユニットごとに、ホスト通信i/fの処理終了ステータスを監視してチャンネルデータを読み出し、その後に次のサンプル処理の開始割り込みをかける。なお、先頭のユニットの場合のみ信号データを設定してから、割り込みをかける。

３－９　ホストプログラムとＤＳＰプログラムの基本タイミング

ホスト上のプログラムとＤＳＰシステム上のプログラムの基本的なタイミングをFig.3 に示す。

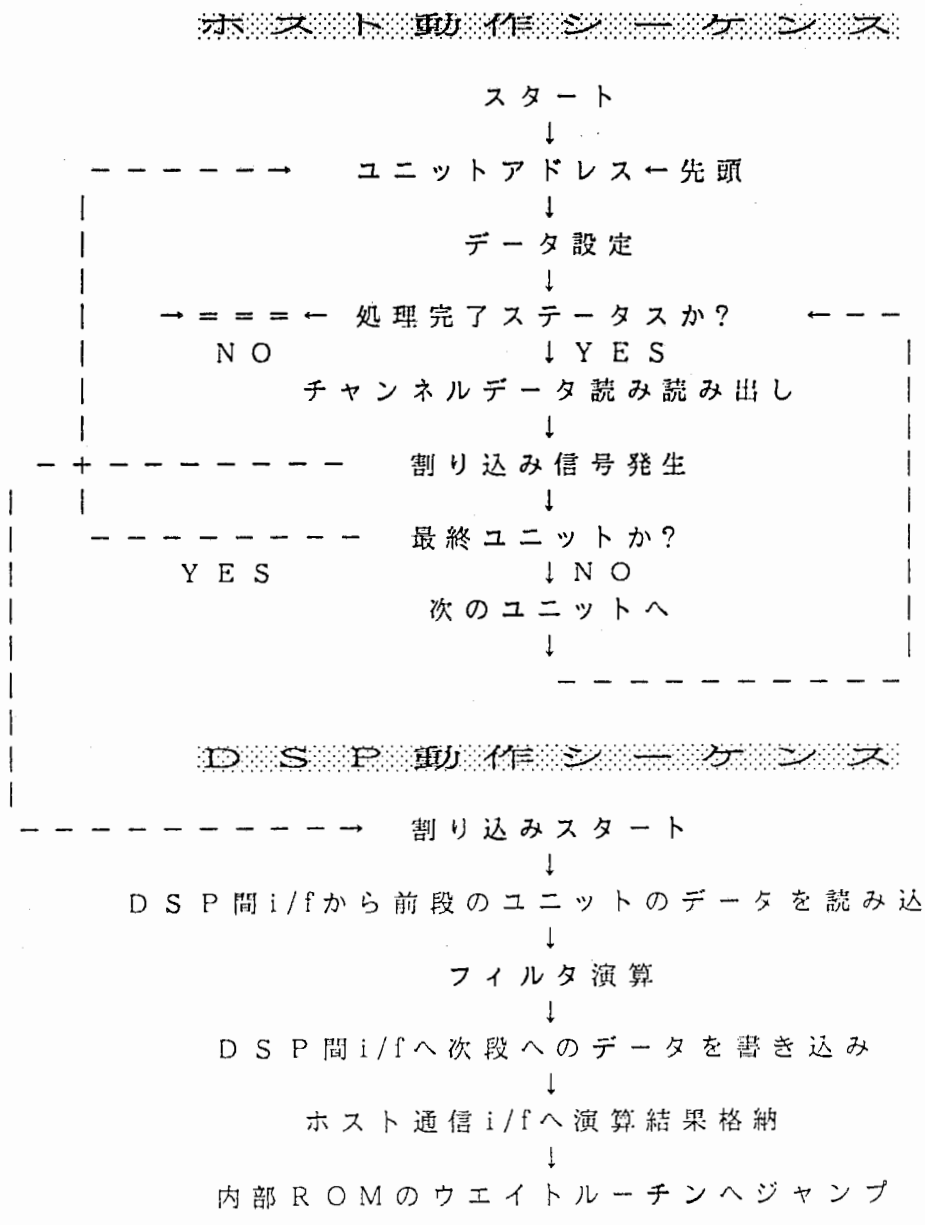


Fig.3 ホストプログラムとＤＳＰプログラムの基本動作シーケンス

4．試作システムのハードウエア構成

4－1　システムの構成

試作したＤＳＰシステムのブロックダイアグラムをFig.4 に、その全体的な回路図をFig.5-1に、各部の回路図をFig.5-2からFig.5-10として章末に示す。

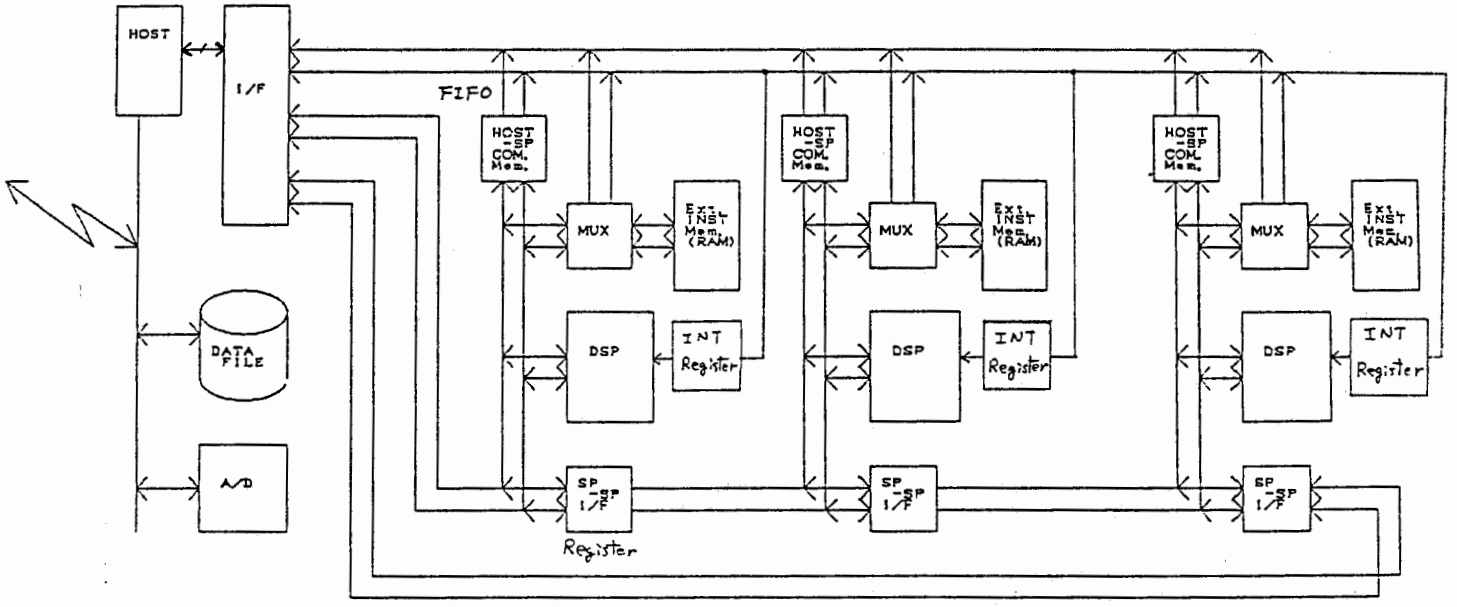


Fig.4　試作システムのブロックダイアグラム

４－２　ＤＳＰユニット

４－２－１　ＤＳＰユニット番号の設定方法（ＤＳＰ側）

本システムでは複数のユニットを使用するため、ユニットの番号を格納するレジスタが各ユニットにある。レジスタアドレスはＤＳＰ側1800Hで、データは15ビット幅で書き込み専用である。このレジスタを使用する場合には、次の手順により番号管理を行なう。

（１）Reset 直後のデフォルト状態は、全DSPユニットはBroadcast mode。
（２）HOSTからイニシャルプログラムを全DSPユニットの外部メモリへダウンロードする。
（３）イニシャルプログラムでDSPユニット番号を登録。
（４）DSPユニット間i/F を通じて、ユニットの結合順にDSPユニット番号を付与する。
（５）各DSPユニットは、登録後waitし、フィルタリングプログラムのロードと起動を待つ。
（６）最終DSPユニットの出力はHOSTに戻され、設定終了とDSPユニット総数が検出される。
（７）フィルタリングプログラムのロード時は DSPユニット選択アドレスが付加され、イニシャルプログラムで設定された DSP番号と一致した DSPユニットがこれを受ける。

以上の手順により、各 DSPユニットは、絶対的なアドレスを必要とせず、また全体の DSPユニット数の増減も容易となる。なお、各ユニットにはハード的に絶対アドレスを設定する16ビットのスイッチも備えているので、適宜使い分けることができる。

４－２－２　ＤＳＰユニットの選択方法（ホスト側）

HOSTは、ある DSPユニット内部のメモリおよびレジスタをアクセスするに先立ち、そのユニット番号をユニット選択レジスタへ書き込むことによって、そのユニットを選択状態にする必要がある。ユニット選択レジスタへの書き込みはＦＮビットを制御してバスへデータを書くことで行なわれる。

すなわち、FN3=1,FN2=1,FN0=1 or 0としてデータを書き込む。このレジスタは書き込み専用である。

DSPユニットの番号は、全節で示したようにイニシャル時に付与されている。この番号と、上記のレジスタの内容が一致した DSPユニットが選択状態になる。

指定できるユニットの範囲は0000Hから7FFFHまでで、8000HからFFFFHを指定した場合には、全 DSPユニットが選択状態になる。これを利用して一斉書き込みを行なうことができる。

４－２－３　割り込み制御方法

各 DSPユニットに対するRESET, INT, NMIは、DSPユニット選択アドレスと組合せて、各ユニットごとに発生できる。一斉モードでは、すべての DSPユニットに対してこれらを与えることができる。Fig.6 に割り込みレジスタの内容を示す。

割り込みレジスタは８ビットラッチの 74HC574で構成されており、回路の簡略化のため、立ち上がりエッジトリガであるCK入力にはユニット内アドレス4000H～7FFFH に対応するデコード信号CS#1（負論理）を用いている。このため、本レジスタのデータラッチは、選択状態から非選択状態への遷移時に行なわれ、この時ラッチされるデータは、直前の選択時のバスの内容、すなわち選択時アドレスの下位バイトである。そこで、本レジスタのアドレスは４Ｋワードにわたってイメージを持っていることを利用し、上位バイトで選択・非選択を制御し、下位バイトにラッチデータを置くことにより、レジスタへの書き込みを行なう。

例えば、RESETをかける場合には、40FBH,80FBH,40FFH,80FFHのように、アドレスを設定すれば良い。なお、本システムではハードウエアリセットは行なわないので、システム立ち上げ時には本レジスタによりリセットをかける必要がある。

割り込みレジスタのタイミングチャートをFig.7 に示す。

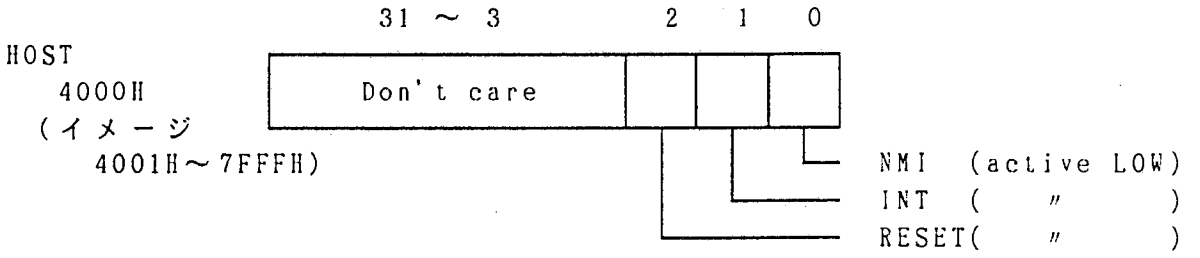


Fig.6　割り込みレジスタ

４－２－４　ＨＯＳＴ側アドレスマップ

ホスト側からみたアドレスマップをFig.8 に示す。ホスト側から見た各 DSPユニットのアドレス・データｉ／Ｆは16bit 幅である。アドレッシングは、16ビットのアドレスレジスタを用いて、 64kWord（1Word=32bit）の空間に対して行う。アドレスレジスタへのライトはFNビットの組合せで選択を行なう。すなわちFN3=1,FN2=0,FN1=1 or 0でバスへのライトを行なう。

16bitアドレスのうち、上位2bit、下位14bitをそれぞれ、デバイス番号、ワードアドレスとも呼ぶ。

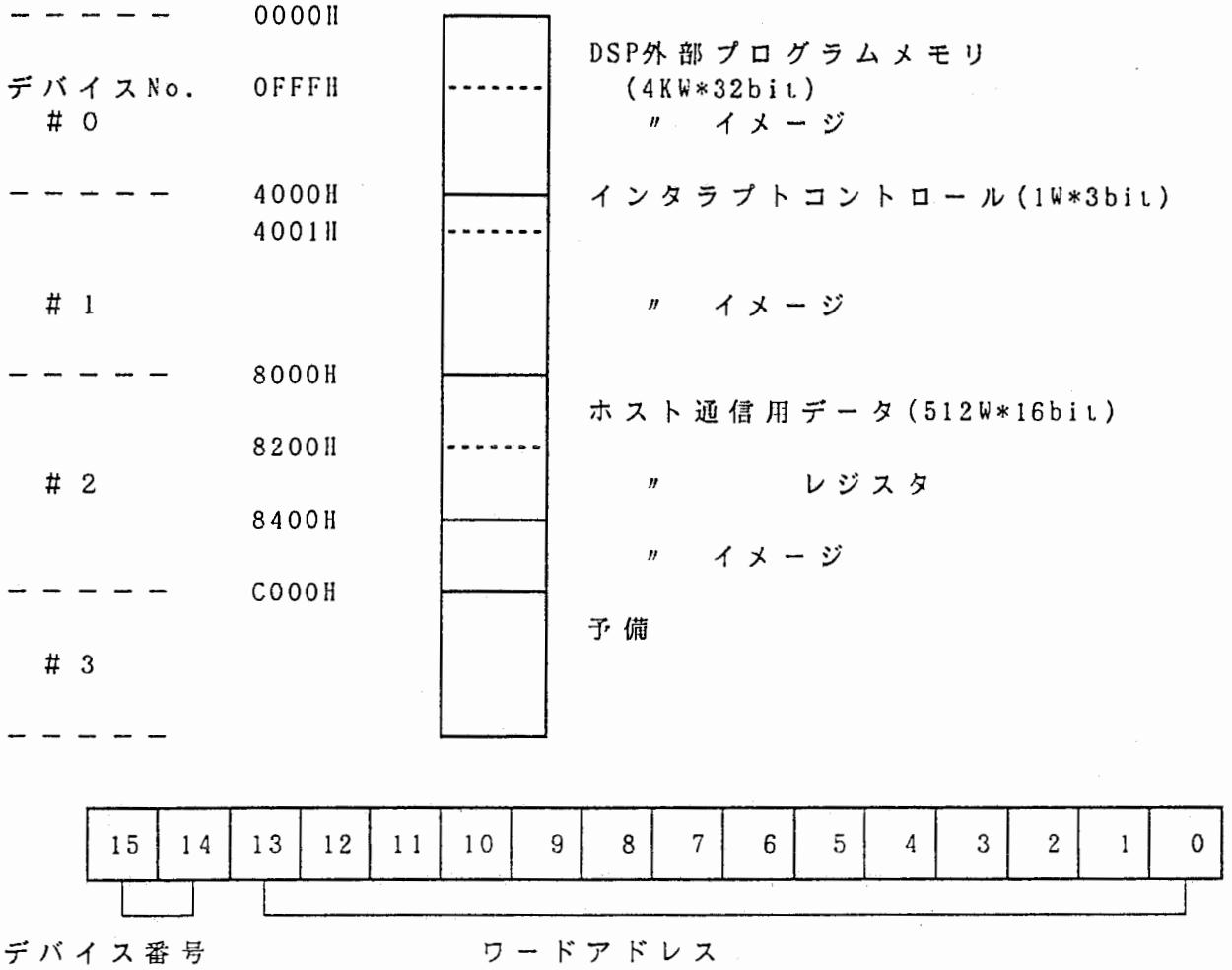
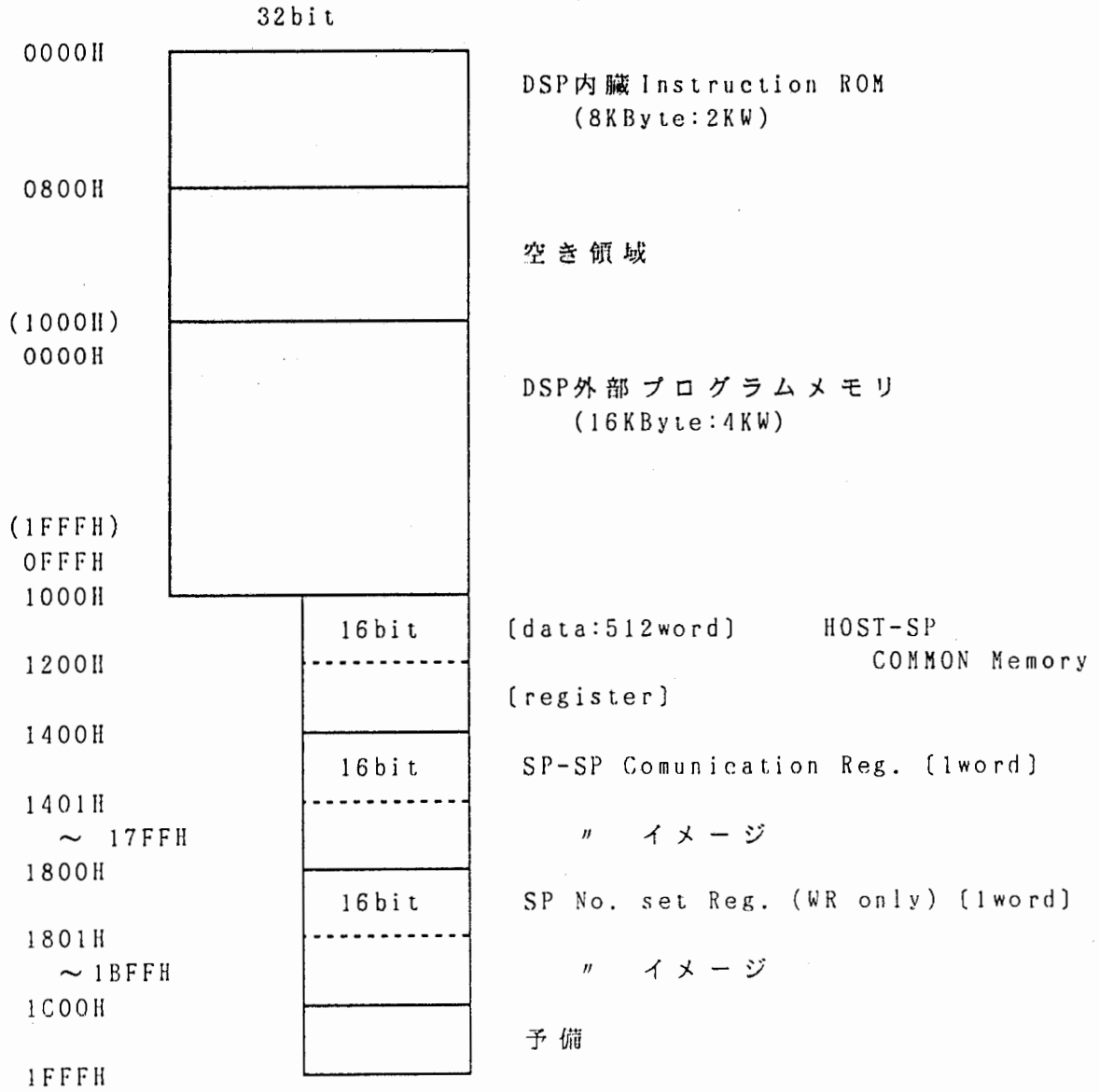


Fig.8 ホスト側からみたアドレスマップ

４－２－５　ＤＳＰ側アドレスマップ

DSP側のアドレスマップをFig.9 に示す。 DSPのデータバス幅は32bitで、 アドレスバス幅はAX,A11～A0 の13bitである。 ただし、 AX=1の後半4kWord は低速アクセス領域で、 データメモリ専用である。



○ ( )外アドレスはバスの出力アドレスを、( )内アドレスはPCの値を示す。

Fig.9　DSP側のアドレスマップ

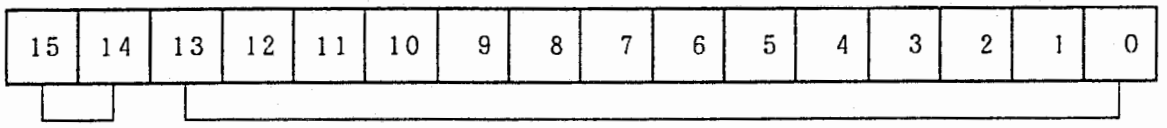
４－２－６　プログラムメモリ部へのアクセス方法

本節では、ホストから DSPユニットの外部プログラムメモリへのアクセスする方法について述べる。

ホストから DSPユニットの外部プログラムメモリへは書き込みのみ可能である。

ホストから DSPユニットの外部プログラムメモリへのアクセスは、ユニット選択レジスタおよびアドレスレジスタによりワードアドレスを指定し、さらに FN0 ビットで上位下位の16bitを指定して書き込みを行なう。すなわち、FN3=0,FN2=1 でデータライトモードが指定され、FN0=1では上位16ビットに、FN0=0では下位16ビットにデータを書き込むことができる。Fig.13にアドレスレジスタとFN1～FN3の出力信号を示す。

以上のアクセスには次の点に注意が必要である。外部プログラムメモリのデバイス番号は＃０であり、アクセス時には*CS#0 がアクティブになっている。この時、ホスト側のバスバッファが開いており、本システムでは DSP側にバスバッファを入れていないので、同時アクセス時にバス衝突を生ずる。これを避けるために、ホストからの外部プログラムメモリアクセス時には、あらかじめ DSPの割り込みプログラムなどで、このデバイス以外のアドレス上で DSPを待ち状態にしておく必要がある。また、逆に、DSPが外部プログラムメモリ上のプログラム実行中は、ホスト側からのアクセスを行なってはならない。



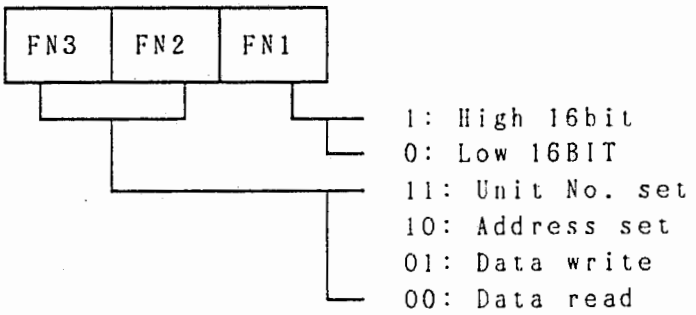


Fig.9 アドレスレジスタとFN1～FN3の出力信号

４－３　インターフェース部

４－３－１　ＤＳＰユニット上のホストインターフェイス

ＤＳＰバンクの処理データの送受など、ＤＳＰとホスト間のデータ通信は、デュアルポートメモリHD63310により行なっている。試作システムでは２つのHD63310により、16ビットデータ幅で通信できる。ワードの上位16ビットは無意味で下位16ビットのみ有効である。

ホスト側のアドレスは、データが8000Hから81FFHまでの512ワード、レジスタが8200Hからになっている。また、１Ｋワードごとに BFFFHまでイメージが発生している。

ＤＳＰ側のアドレスは、データが1000Hから11FFHまでの512ワードで、レジスタが1200Hからになっている。

デバイス内部のレジスタアドレスおよびコマンドなどの詳細は、HD63310Rの資料を参照されたい。

４－３－２　ＤＳＰユニット間のインターフェース

DSPユニット間のインターフェースは、DSPアドレス1400H番地にある32ビットのレジスタにより実現され、DSPユニット間におけるデータの送受を行う。

上記のレジスタの番地を読むことによって、前段の DSPユニットのデータが得られる。逆に、レジスタにデータを書き込むことによって、次段の DSPユニットへデータが送られる。

ハードウエアとしては、各ユニットには書き込み用の 74HC574が４個ずつ置かれ、その出力ポートと出力制御信号線が次段のユニットのＤＳＰバスに接続されている。この74HC574への書き込みアドレスと前段ユニットの出力制御アドレスが同一になっている。なお、1401Hから17FFHまではイメージ領域である。（Fig.10）

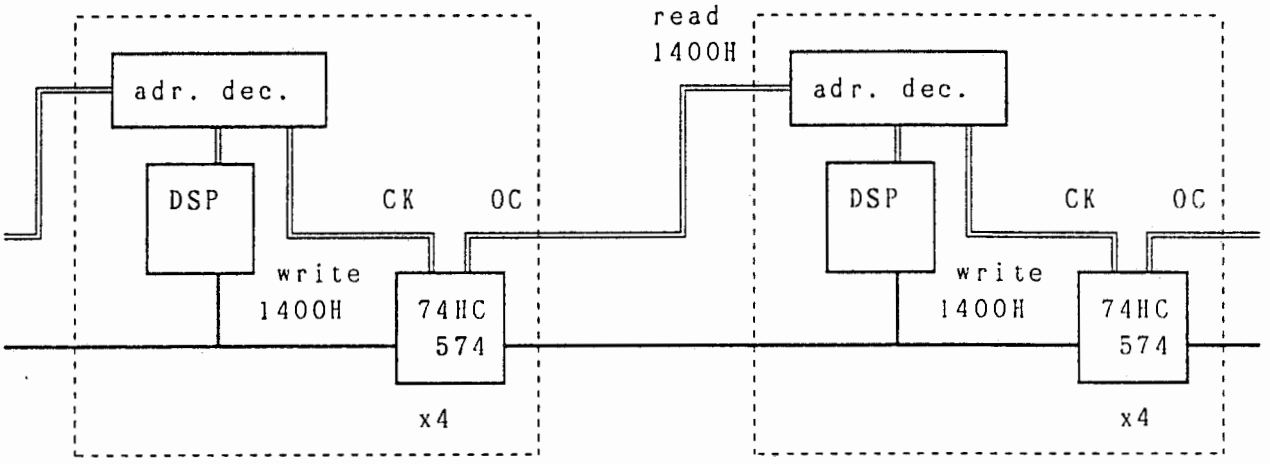


Fig.10　DSP間のインターフェイス

４－３－３　ホスト側のインターフェース（MASSCOMP PI16F）

試作システムとホストマシンであるマスコンプとの結合はマスコンプの汎用パラレルインターフェースのPI16Fを介して行なわれる。PI16Fでのデータ送受はハンドシェーク手順により行なわれる。本システムではハードおよびソフトの負担を軽減するために、入出力を常に同時に行っており、PI16F 自身のハンドシェーク信号を折返すことによりデータ送受を実現している。

マスコンプからのリード／ライト信号は同時に発生するため、FN3およびFN2の信号によってリードおよびライトのモードを区別している。すなわち、FN3=0,FN2=1,FN1=1 or 0 で書き込み、FN3=0,FN2=0,FN1=1 or 0 で読み込みを行なう。なお、本システムではリード／ライトとも可能なデバイスは共有メモリ HD63310だけであり、他はライトのみ可能になっている。

ホストインターフェイスの接続図をFig.11に、出力信号FN3～FN1の内容をFig.12に、PI16Fとのハンドシェイクタイミングチャートを Fig.13と Fig.14 に示す。また、表2、表3 にPI16Fと接続する入出力ケーブルのコネクタ接続配置図を示す。表4 にはカードエッジのピン配置を示す。

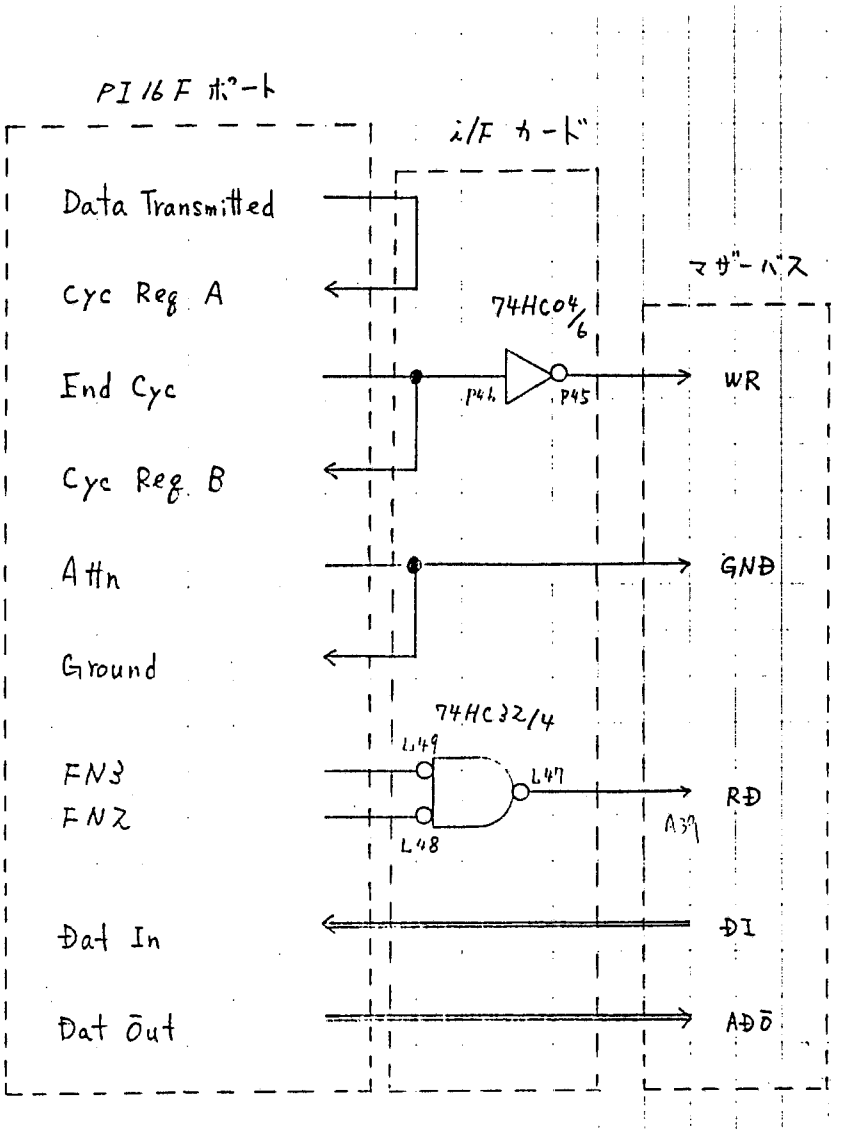


Fig. 11 ホストインターフェイス接続図

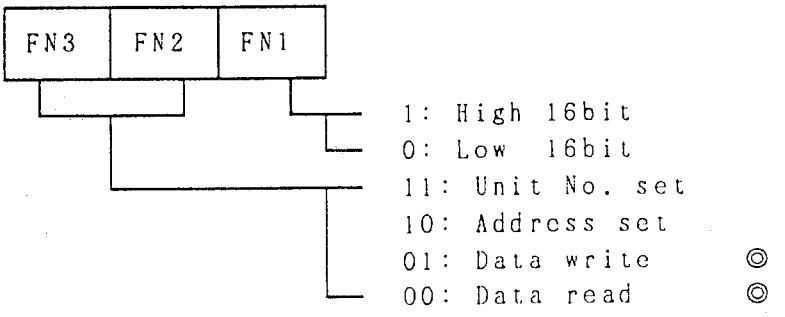
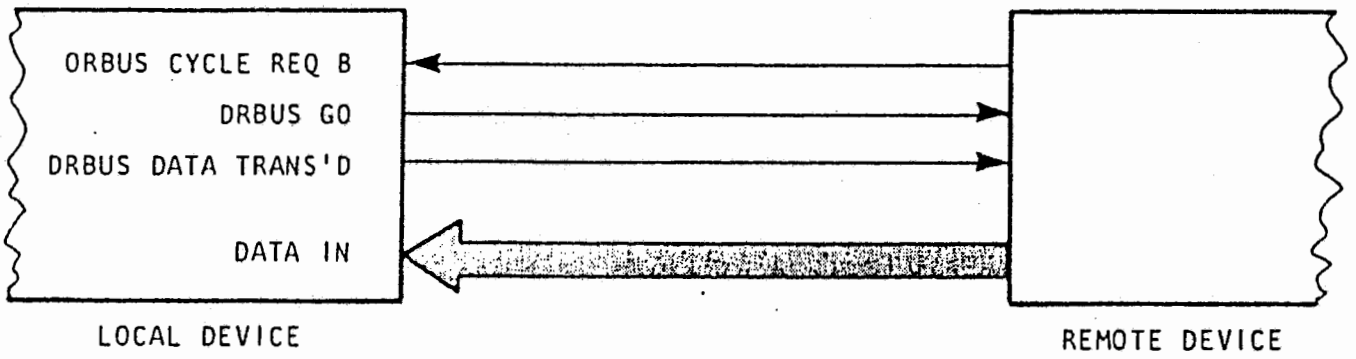
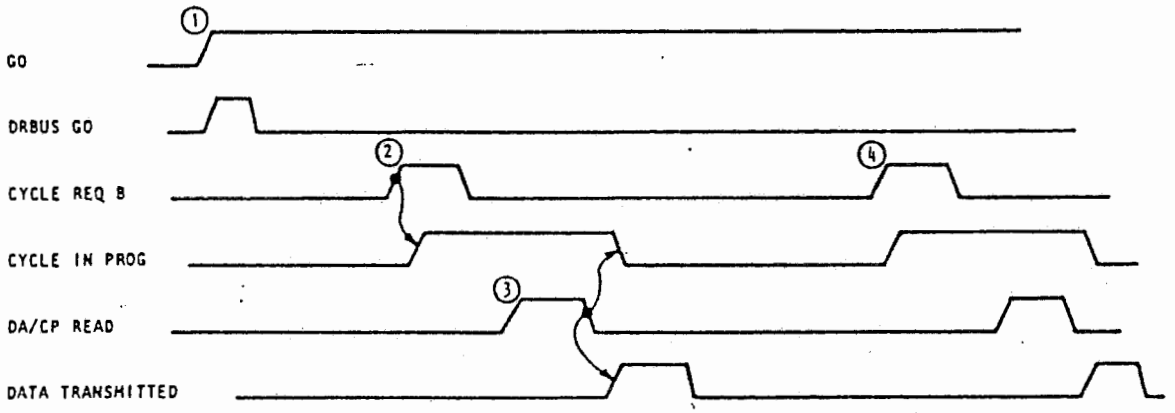


Fig.12 ホストインターフェイスの出力信号

Fig-13 PI 16 F ハンドシェーク タイミング図 (入力)

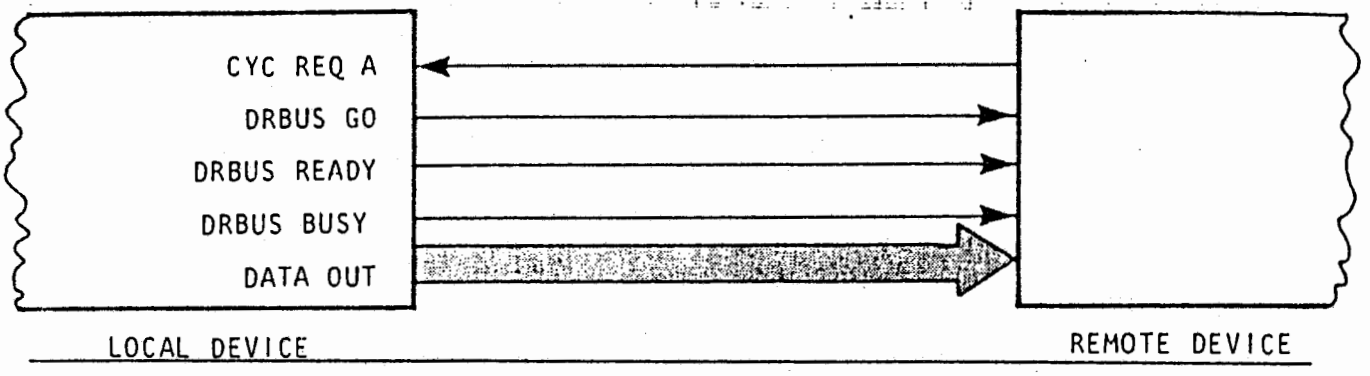


Principal Signals for an Inbound Transfer.

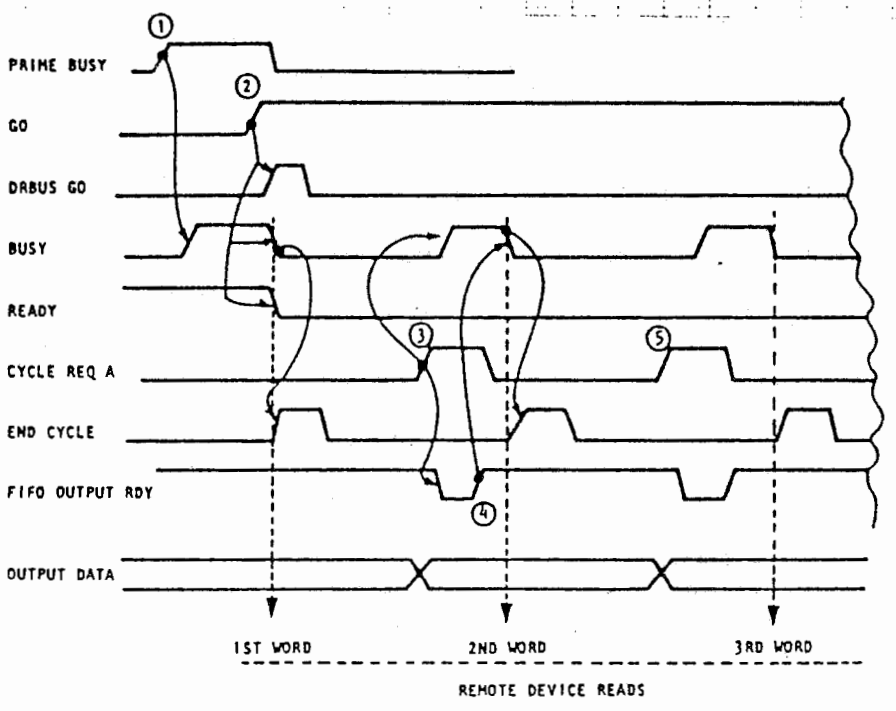


Inbound transfer with handshake.

Fig. 14 PI16F ハンドシェーク タイミング図 (出力)



Principal Signals during an Outbound Transfer.

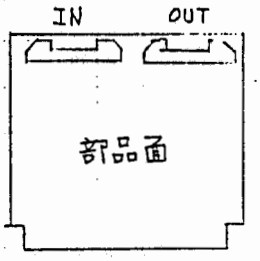


Outbound transfer - FIFO not empty.

表 2 PI16F 対 i/F ボード コネクタ表※ (40pin フラットケーブル x 2)

Input Connector Pinout

A (A) Name	Type	Pin	Name	Type	B (B)
GND		1	DRBUS BUSY	OUT	
GND		2	DRBUS ATTN	IN	
GND		3	DRBUS AUXSTAT3	IN	
GND		4	DRBUS AUXSTAT1	IN	
DRBUS FN3	OUT	5	DRBUS FN3	OUT	
GND		6	DRBUS AUXSTAT4	IN	
GND		7	DRBUS FN2	OUT	
GND		8	DRBUS AUXSTAT5	IN	
GND		9	DRBUS FN1	OUT	
GND		10	DRBUS GO	OUT	
GND		11	GND		
GND		12	GND		
DRBUS DATIN 7	IN	13	DRBUS DATIN 8	IN	
DRBUS DATIN 6	IN	14	DRBUS DATIN 9	IN	
DRBUS DATIN 5	IN	15	DRBUS DATIN 10	IN	
DRBUS DATIN 4	IN	16	DRBUS DATIN 11	IN	
DRBUS DATIN 3	IN	17	DRBUS DATIN 12	IN	
DRBUS DATIN 2	IN	18	DRBUS DATIN 13	IN	
DRBUS DATIN 1	IN	19	DRBUS DATIN 14	IN	
DRBUS DATIN 0	IN	20	DRBUS DATIN 15	IN	



Output Connector Pinout

A (a) Name	Type	Pin	Name	Type	B (a)
GND		1	DRBUS CYC REQ A	IN	
GND		2	DRBUS FN2	OUT	
GND		3	DRBUS READY	OUT	
GND		4	DRBUS DATA TRANS'D	OUT	
DRBUS AUXSTAT2	IN	5	DRBUS DSTAT A	IN	
GND		6	DRBUS INIT	OUT	
GND		7	DRBUS DSTAT B	IN	
GND		8	DRBUS DSTAT C	IN	
GND		9	DRBUS DSTAT C	IN	
GND		10	DRBUS END CYCLE	OUT	
GND		11	DRBUS CYC REQ B	IN	
GND		12	GND		
DRBUS DATOUT 7	OUT	13	DRBUS DATOUT 8	OUT	
DRBUS DATOUT 6	OUT	14	DRBUS DATOUT 9	OUT	
DRBUS DATOUT 5	OUT	15	DRBUS DATOUT 10	OUT	
DRBUS DATOUT 4	OUT	16	DRBUS DATOUT 11	OUT	
DRBUS DATOUT 3	OUT	17	DRBUS DATOUT 12	OUT	
DRBUS DATOUT 2	OUT	18	DRBUS DATOUT 13	OUT	
DRBUS DATOUT 1	OUT	19	DRBUS DATOUT 14	OUT	
DRBUS DATOUT 0	OUT	20	DRBUS DATOUT 15	OUT	

※ Pin No., A/B side 名は、i/F ボード側に対応。PI16F 側は 別紙参照。

表3

PIB下側
コネクタ表

Input Connector Pinout

Pin	Name	Type	Pin	Name	Type
1	GND		2	DRBUS BUSY	OUT
3	GND		4	DRBUS ATTN	IN
5	GND		6	DRBUS AUXSTAT3	IN
7	GND		8	DRBUS AUXSTAT1	IN
9	DRBUS FN3	OUT	10	DRBUS FN3	OUT
11	GND		12	DRBUS AUXSTAT4	IN
13	GND		14	DRBUS FN2	OUT
15	GND		16	DRBUS AUXSTAT5	IN
17	GND		18	DRBUS FN1	OUT
19	GND		20	DRBUS GO	OUT
21	GND		22	GND	
23	GND		24	GND	
25	DRBUS DATIN 7	IN	26	DRBUS DATIN 8	IN
27	DRBUS DATIN 6	IN	28	DRBUS DATIN 9	IN
29	DRBUS DATIN 5	IN	30	DRBUS DATIN 10	IN
31	DRBUS DATIN 4	IN	32	DRBUS DATIN 11	IN
33	DRBUS DATIN 3	IN	34	DRBUS DATIN 12	IN
35	DRBUS DATIN 2	IN	36	DRBUS DATIN 13	IN
37	DRBUS DATIN 1	IN	38	DRBUS DATIN 14	IN
39	DRBUS DATIN 0	IN	40	DRBUS DATIN 15	IN

Output Connector Pinout

Pin	Name	Type	Pin	Name	Type
1	GND		2	DRBUS CYC REQ A	IN
3	GND		4	DRBUS FN2	OUT
5	GND		6	DRBUS READY	OUT
7	GND		8	DRBUS DATA TRANS'D	OUT
9	DRBUS AUXSTAT2	IN	10	DRBUS DSTAT A	IN
11	GND		12	DRBUS INIT	OUT
13	GND		14	DRBUS DSTAT B	IN
15	GND		16	DRBUS DSTAT C	IN
17	GND		18	DRBUS DSTAT C	IN
19	GND		20	DRBUS END CYCLE	OUT
21	GND		22	DRBUS CYC REQ B	IN
23	GND		24	GND	
25	DRBUS DATOUT 7	OUT	26	DRBUS DATOUT 8	OUT
27	DRBUS DATOUT 6	OUT	28	DRBUS DATOUT 9	OUT
29	DRBUS DATOUT 5	OUT	30	DRBUS DATOUT 10	OUT
31	DRBUS DATOUT 4	OUT	32	DRBUS DATOUT 11	OUT
33	DRBUS DATOUT 3	OUT	34	DRBUS DATOUT 12	OUT
35	DRBUS DATOUT 2	OUT	36	DRBUS DATOUT 13	OUT
37	DRBUS DATOUT 1	OUT	38	DRBUS DATOUT 14	OUT
39	DRBUS DATOUT 0	OUT	40	DRBUS DATOUT 15	OUT

表4

カードエッジコネクタ表

部品面			半田面		
A1	GND		B1	GND	1番は、部品面をエッジ側から見て右。
A2	GND		B2	GND	
A3	GND		B3	GND	
A4	AD01	PIGF OUTPUT と対応	B4	NC	
A5	AD02				
A6	AD03				
A7	AD04				
A8	AD05				
A9	AD06				
A10	AD07				
A11	AD08		B5		
A12	AD09		–	NC	
A13	AD010		B46		
A14	AD011				
A15	AD012				
A16	AD013				
A17	AD014				
A18	AD015				
A19	AD016				
A20					
A21	DI1	PIGF INPUT と対応			
A22	DI2				
A23	DI3				
A24	DI4				
A25	DI5				
A26	DI6				
A27	DI7				
A28	DI8				
A29	DI9				
A30	DI10				
A31	DI11				
A32	DI12				
A33	DI13				
A34	DI14				
A35	DI15				
A36	DI16				
A37					
A38	*WR				
A39	*RD				
A40	FN1				
A41	FN2				
A42	FN3				
A43					
A44					
A45					
A46					
A47			B47	NC	
A48	VCC		B48	VCC	
A49	VCC		B49	VCC	
A50	VCC		B50	VCC	

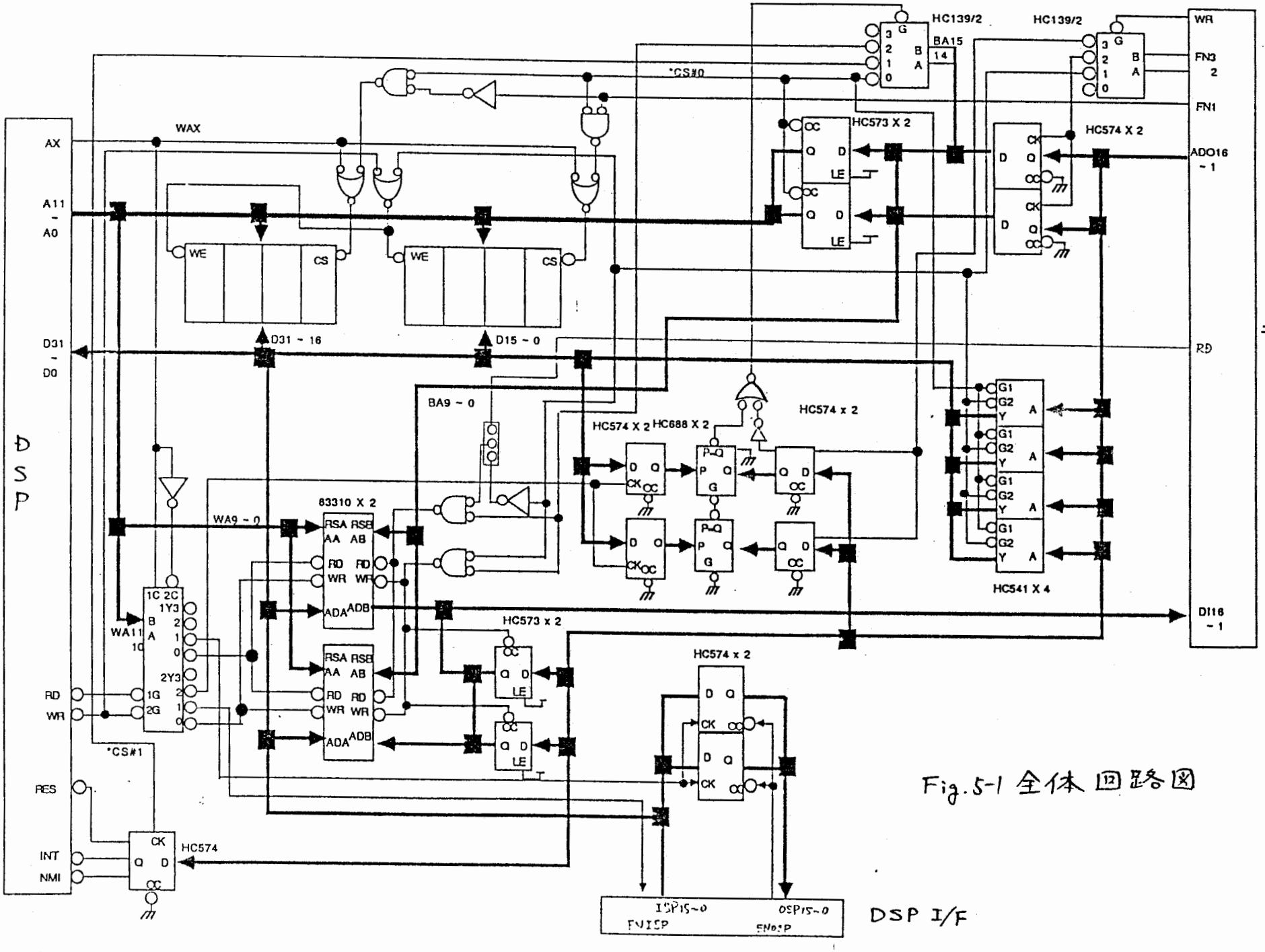
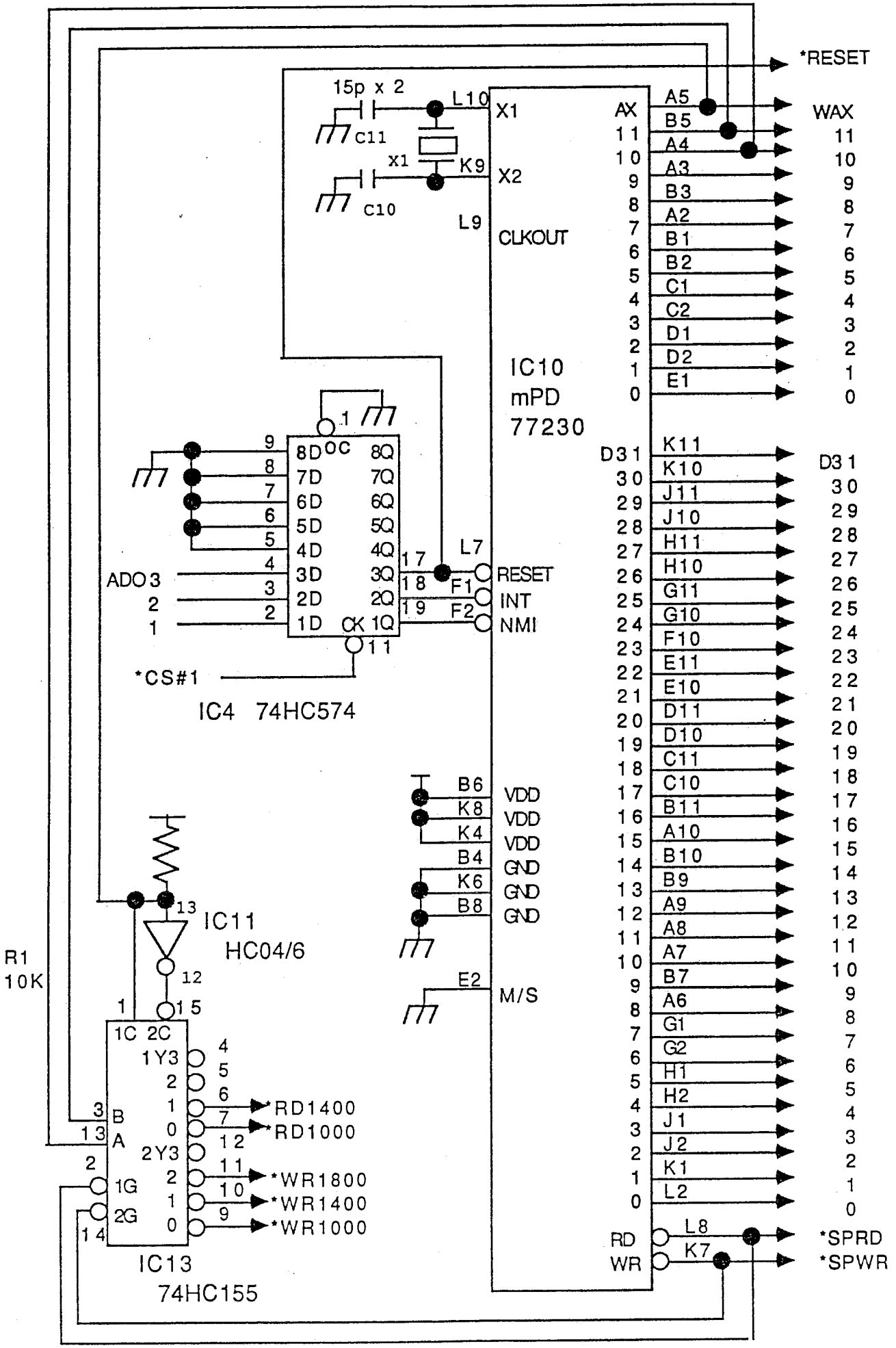
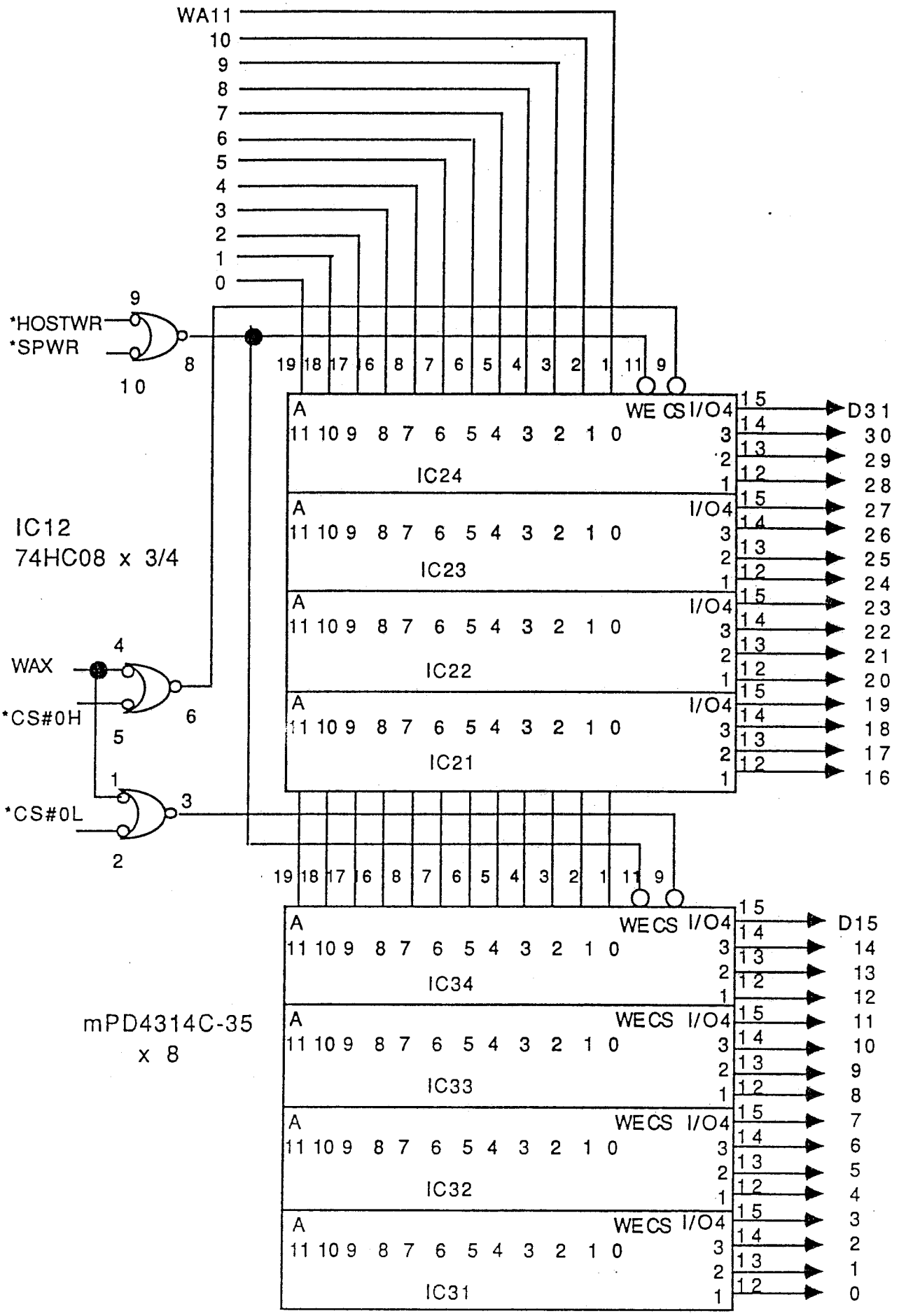
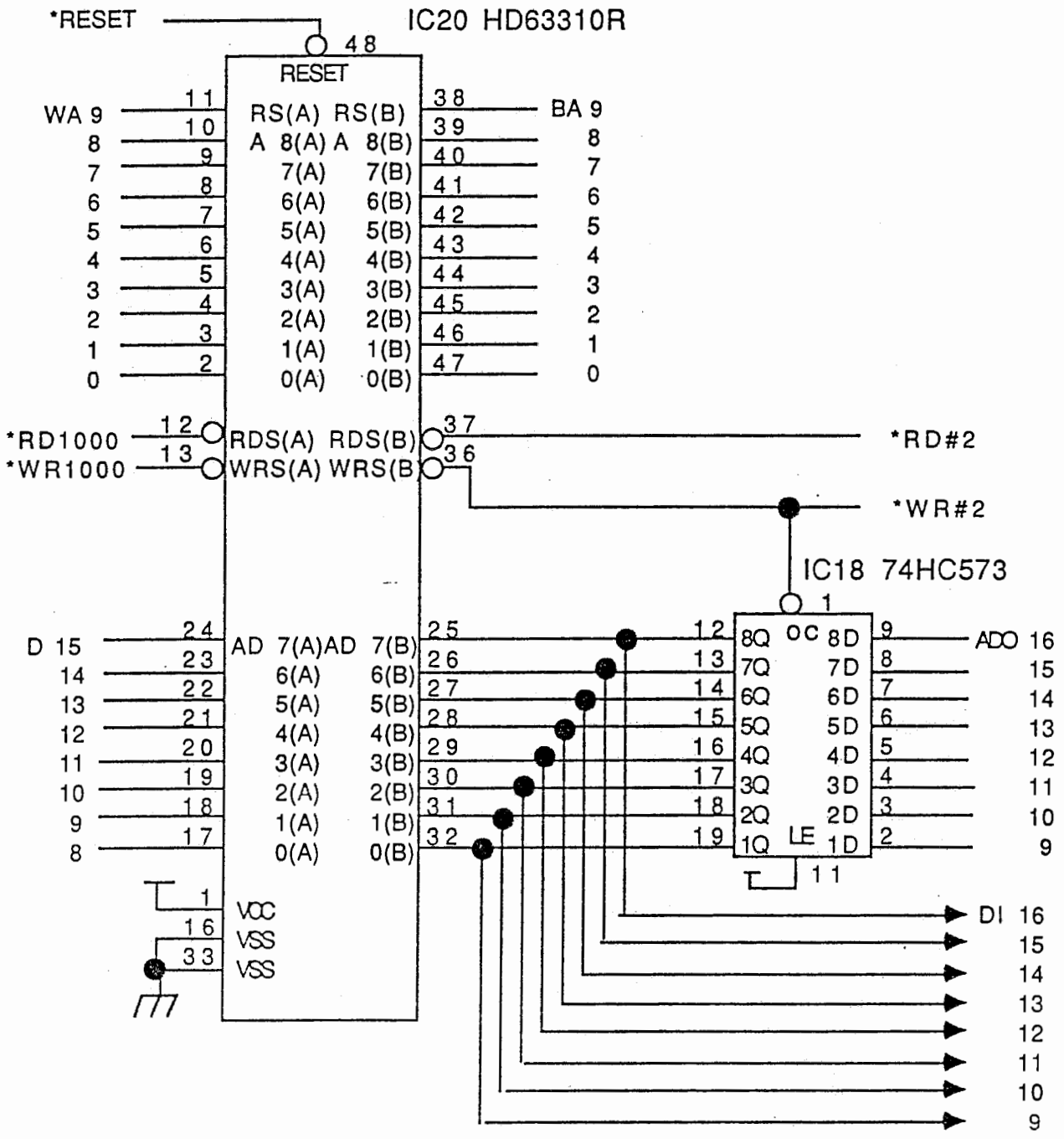
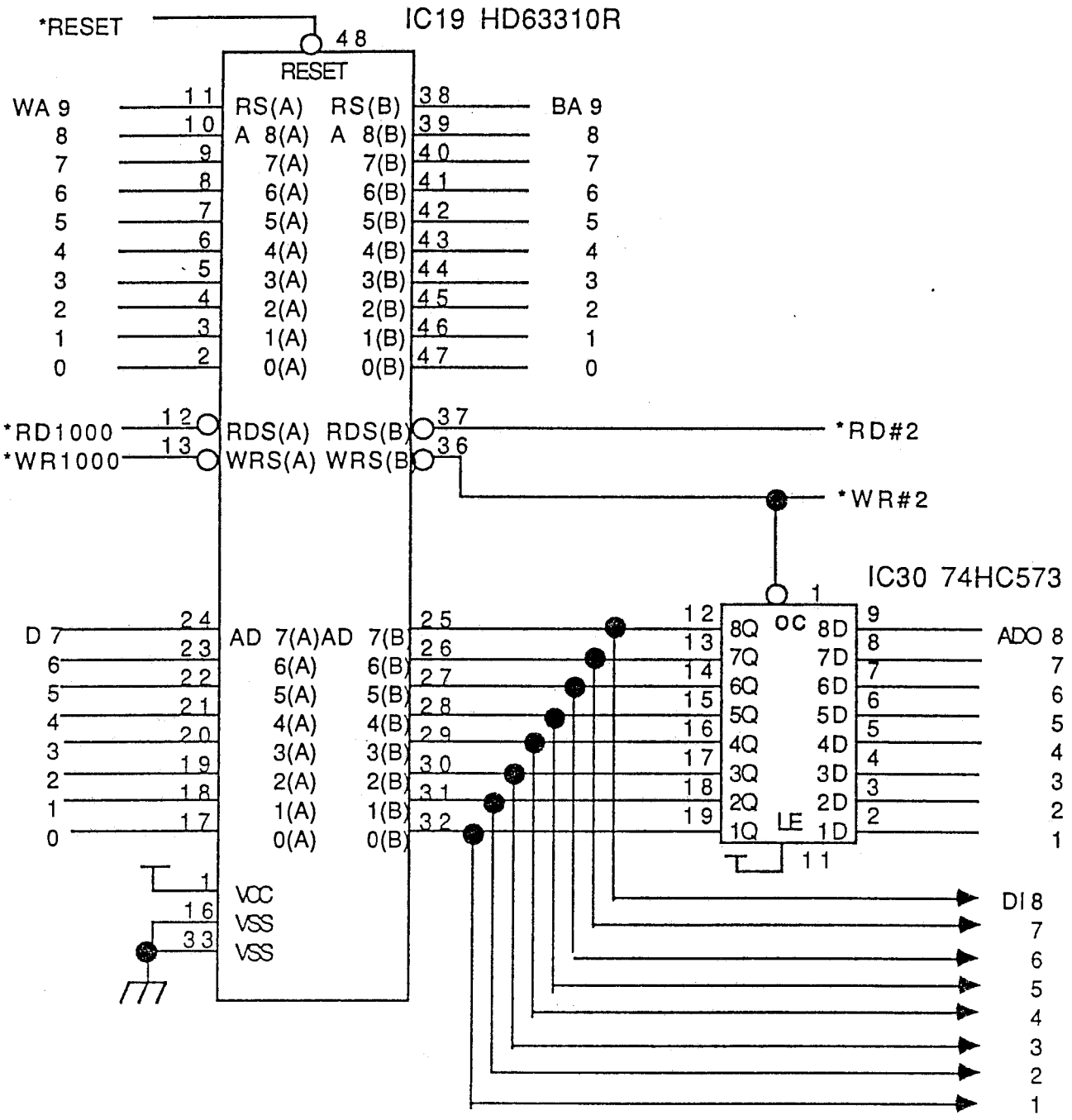

DSP
AX
A11 ~ A0
D31 ~ D0
RD
WR
RES
INT
NMI
*CS#1
WAX
WA9 ~ 0
WA11 10
BA9 ~ 0
D31 ~ 16
D15 ~ 0
WE
CS
*CS#0
HC139/2
BA15
14
HC573 X 2
HC574 X 2
HC574 x 2
HC688 X 2
HC541 X 4
HC573 x 2
HC574
63310 X 2
RSA RSB
AA AB
RD RD
WR WR
ADA ADB
1C 2C
1Y3
2Y3
1G 2G
WR
FN3 2
FN1
ADO16 ~ 1
RD
DI16 ~ 1
⇒ to PI16F
ISP15~0
FNISP
OSP15~0
ENOSP
DSP I/F

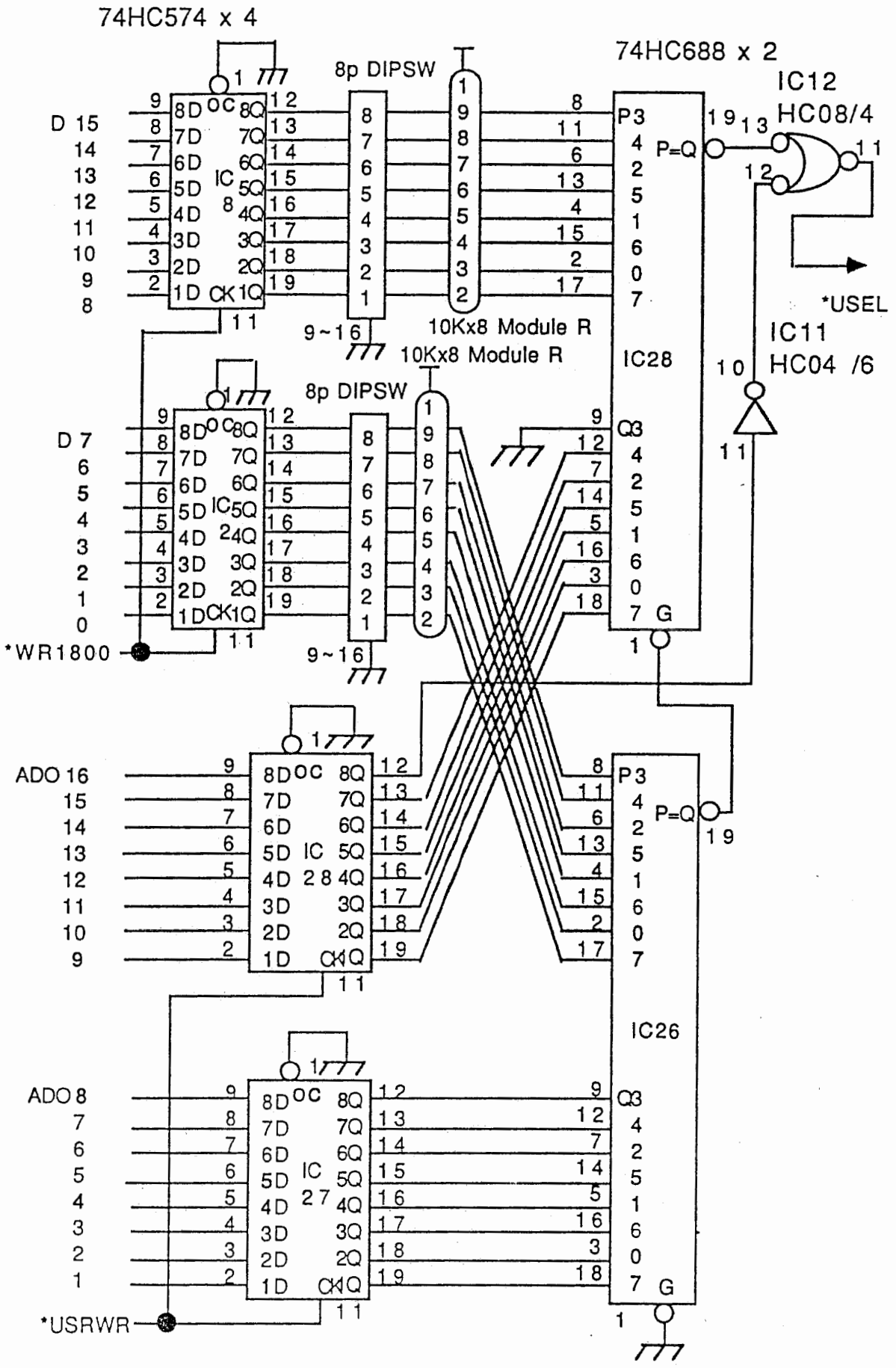
Fig.5-1 全体回路図

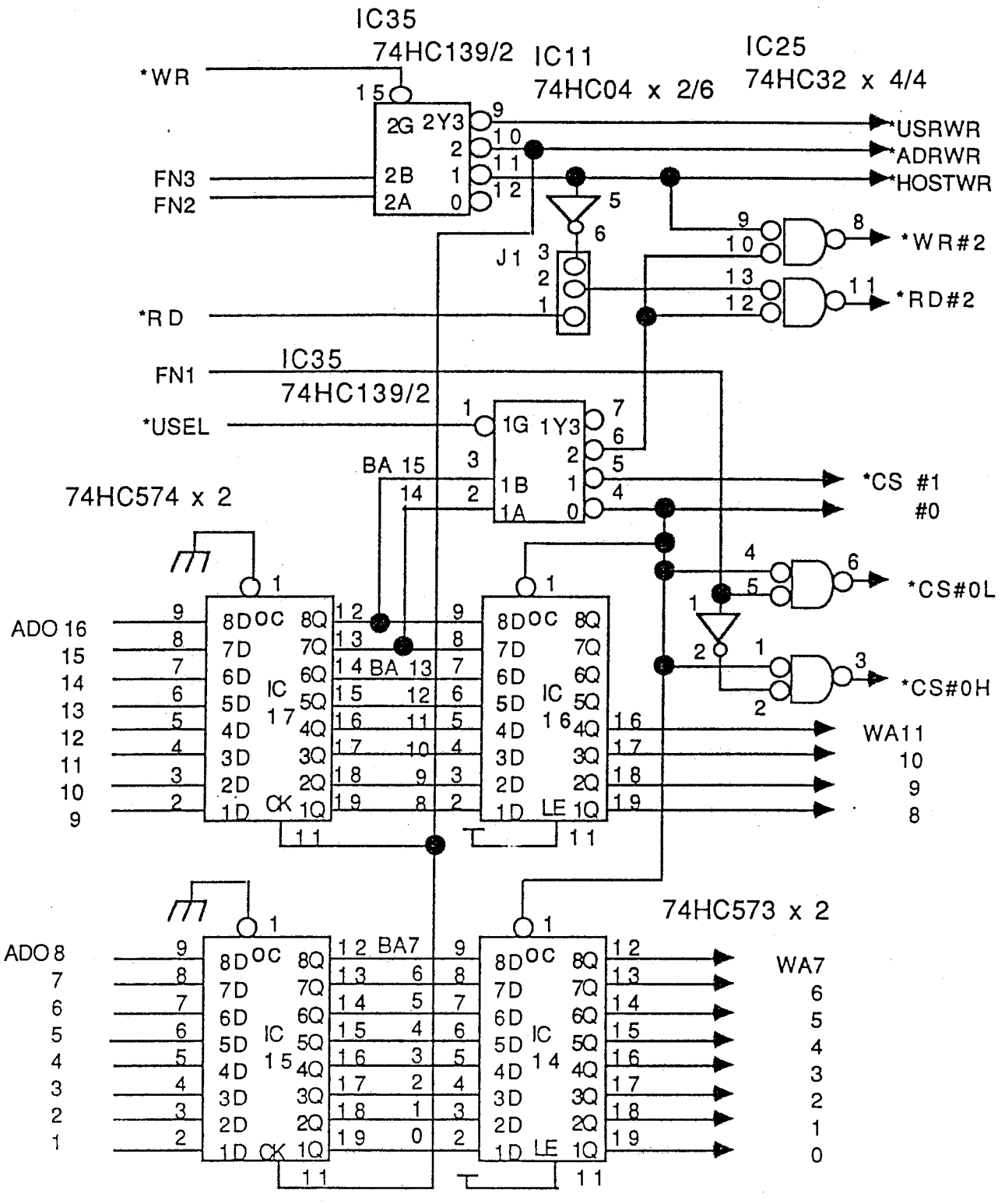

*RESET
15p x 2
C11
C10
X1
L10
K9
L9
X1
X2
CLKOUT
IC10
mPD
77230
AX
11
10
9
8
7
6
5
4
3
2
1
0
A5
B5
A4
A3
B3
A2
B1
B2
C1
C2
D1
D2
E1
WAX
11
10
9
8
7
6
5
4
3
2
1
0
D31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
K11
K10
J11
J10
H11
H10
G11
G10
F10
E11
E10
D11
D10
C11
C10
B11
A10
B10
B9
A9
A8
A7
B7
A6
G1
G2
H1
H2
J1
J2
K1
L2
D31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
RD
WR
L8
K7
*SPRD
*SPWR
L7
17
18
19
F1
F2
RESET
INT
NMI
ADO3
2
1
9
8
7
6
5
4
3
2
1
8D
7D
6D
5D
4D
3D
2D
1D
OC
CK
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
*CS#1
IC4 74HC574
B6
K8
K4
B4
K6
B8
E2
VDD
VDD
VDD
GND
GND
GND
M/S
R1
10K
IC11
HC04/6
13
12
1
15
1C 2C
1Y3
2Y3
1G
2G
B
A
3
13
2
14
4
5
6
7
12
11
10
9
*RD1400
*RD1000
*WR1800
*WR1400
*WR1000
IC13
74HC155

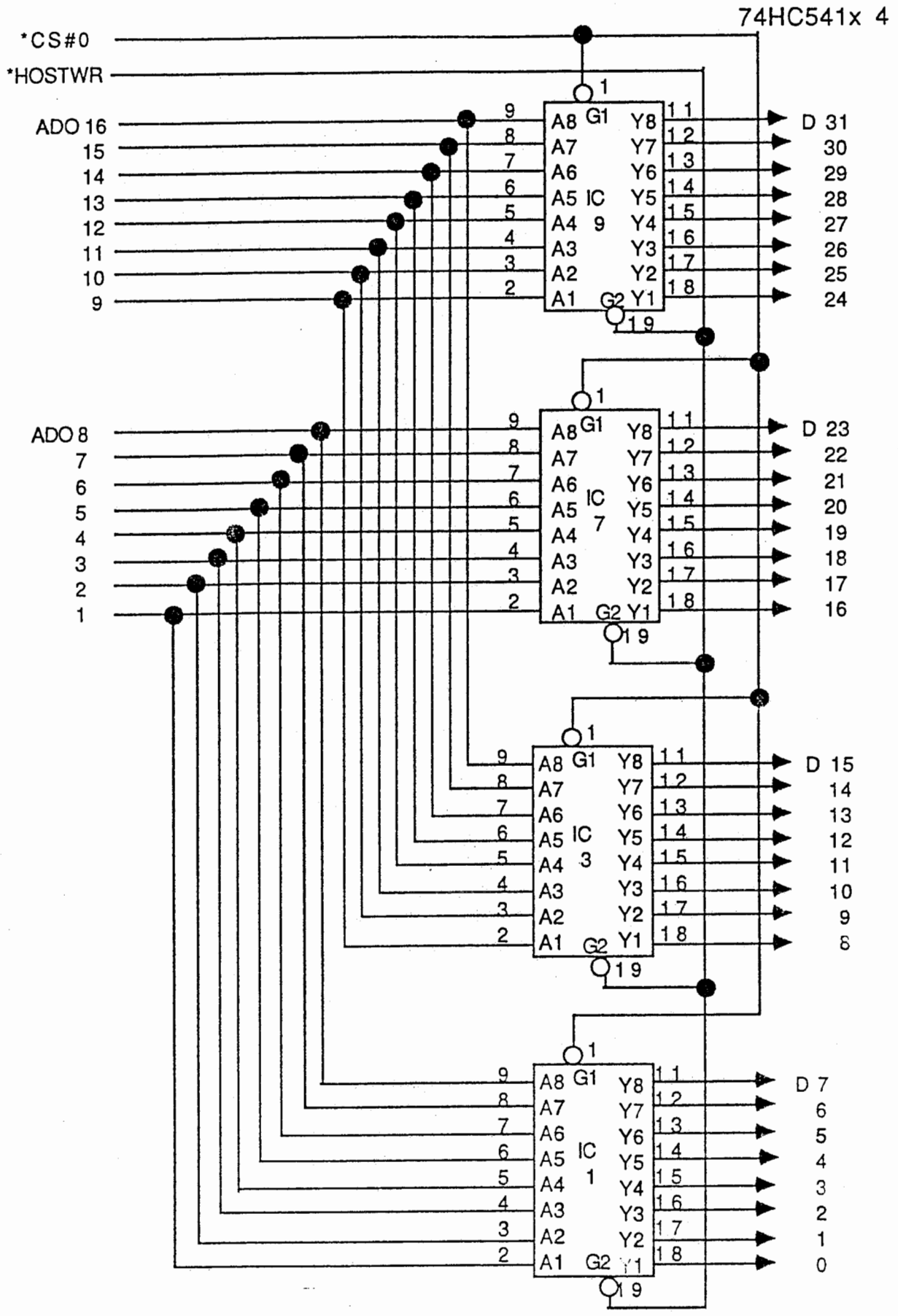

WA11
10
9
8
7
6
5
4
3
2
1
0
*HOSTWR
*SPWR
IC12
74HC08 x 3/4
WAX
*CS#0H
*CS#0L
IC24
IC23
IC22
IC21
IC34
IC33
IC32
IC31
mPD4314C-35
x 8
D31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
D15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

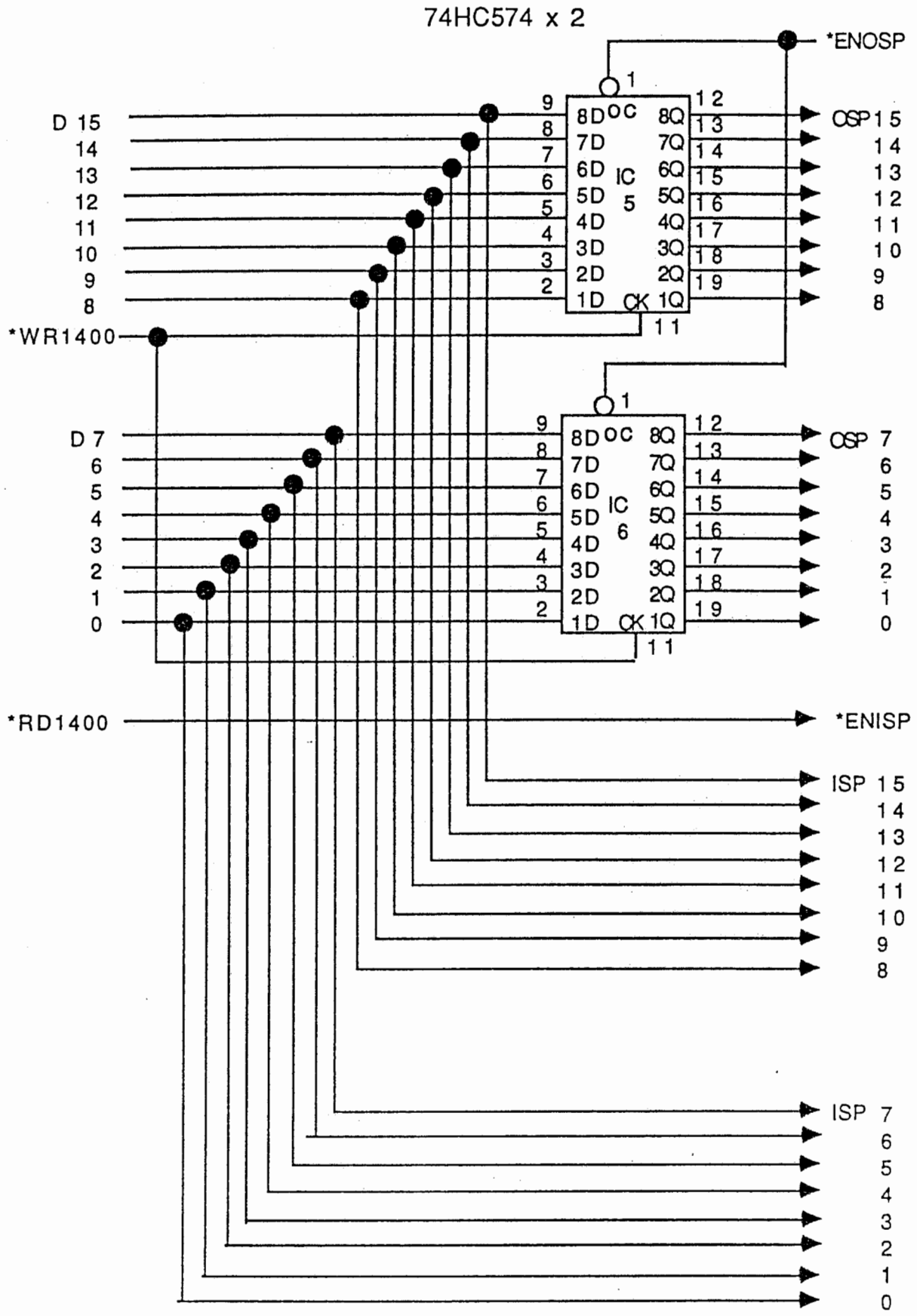

*RESET
IC20 HD63310R
48
RESET
WA 9
8
7
6
5
4
3
2
1
0
11
10
9
8
7
6
5
4
3
2
RS(A) RS(B)
A 8(A) A 8(B)
7(A) 7(B)
6(A) 6(B)
5(A) 5(B)
4(A) 4(B)
3(A) 3(B)
2(A) 2(B)
1(A) 1(B)
0(A) 0(B)
38
39
40
41
42
43
44
45
46
47
BA 9
8
7
6
5
4
3
2
1
0
*RD1000
*WR1000
12
13
RDS(A) RDS(B)
WRS(A) WRS(B)
37
36
*RD#2
*WR#2
IC18 74HC573
1
D 15
14
13
12
11
10
9
8
24
23
22
21
20
19
18
17
AD 7(A) AD 7(B)
6(A) 6(B)
5(A) 5(B)
4(A) 4(B)
3(A) 3(B)
2(A) 2(B)
1(A) 1(B)
0(A) 0(B)
25
26
27
28
29
30
31
32
12
13
14
15
16
17
18
19
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
OC
LE
8D
7D
6D
5D
4D
3D
2D
1D
9
8
7
6
5
4
3
2
ADO 16
15
14
13
12
11
10
9
11
1
16
33
VCC
VSS
VSS
DI 16
15
14
13
12
11
10
9

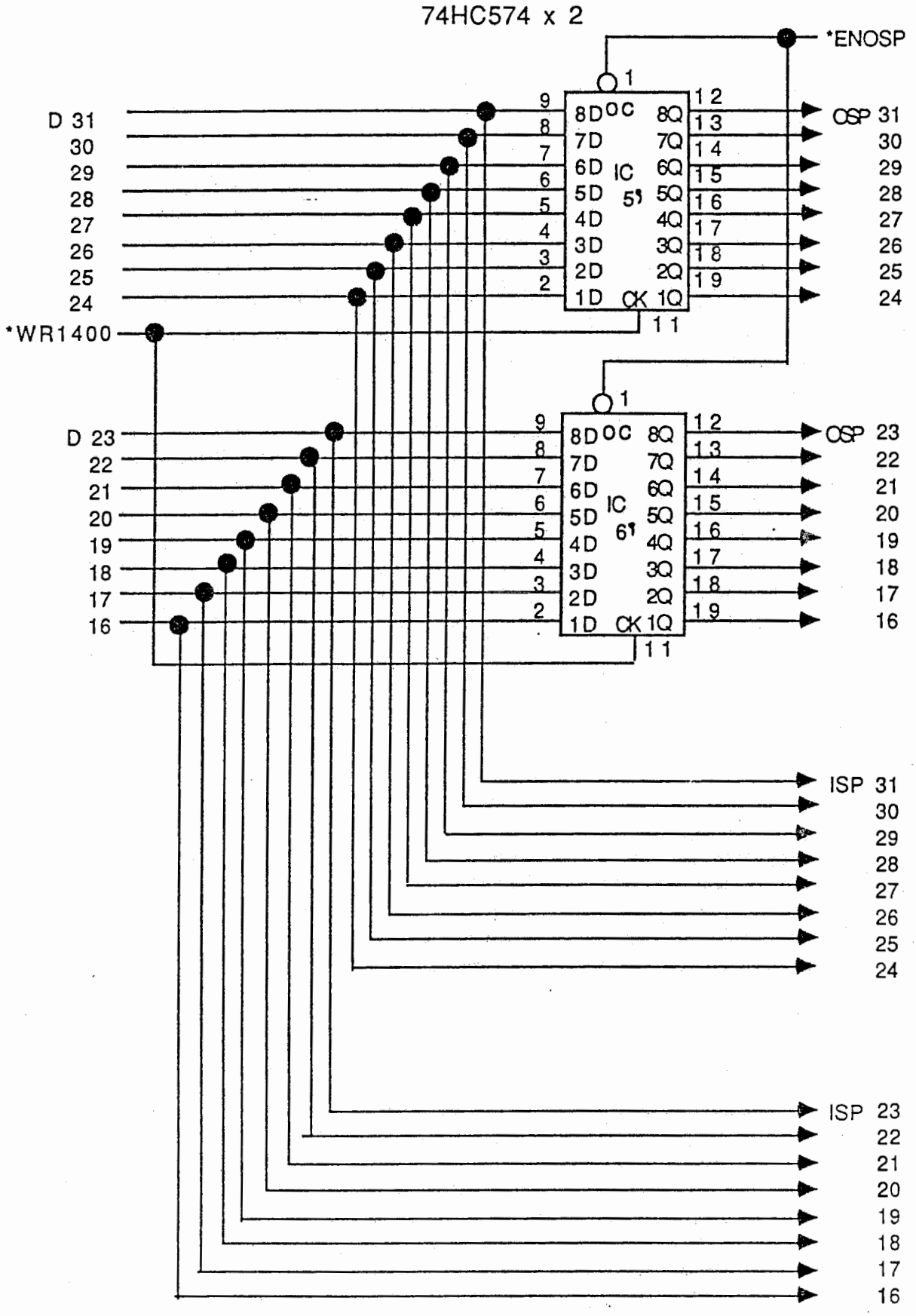

IC19 HD63310R
*RESET
48
RESET
WA 9
8
7
6
5
4
3
2
1
0
11
10
9
8
7
6
5
4
3
2
RS(A)
A 8(A)
7(A)
6(A)
5(A)
4(A)
3(A)
2(A)
1(A)
0(A)
RS(B)
A 8(B)
7(B)
6(B)
5(B)
4(B)
3(B)
2(B)
1(B)
0(B)
38
39
40
41
42
43
44
45
46
47
BA 9
8
7
6
5
4
3
2
1
0
*RD1000
*WR1000
12
13
RDS(A)
WRS(A)
RDS(B)
WRS(B)
37
36
*RD#2
*WR#2
IC30 74HC573
1
OC
LE
11
12
13
14
15
16
17
18
19
8Q
7Q
6Q
5Q
4Q
3Q
2Q
1Q
8D
7D
6D
5D
4D
3D
2D
1D
9
8
7
6
5
4
3
2
ADO 8
7
6
5
4
3
2
1
D 7
6
5
4
3
2
1
0
24
23
22
21
20
19
18
17
AD 7(A)
6(A)
5(A)
4(A)
3(A)
2(A)
1(A)
0(A)
AD 7(B)
6(B)
5(B)
4(B)
3(B)
2(B)
1(B)
0(B)
25
26
27
28
29
30
31
32
DI 8
7
6
5
4
3
2
1
1
VCC
16
VSS
33
VSS


74HC574 x 4
74HC688 x 2
IC12
HC08/4
8p DIPSW
8p DIPSW
10Kx8 Module R
10Kx8 Module R
9~16
9~16
D 15
D 7
ADO 16
ADO 8
*WR1800
*USRWR
*USEL
IC11
HC04 /6
IC28
IC26
P=Q
P=Q
G
G


IC35
74HC139/2
IC11
74HC04 x 2/6
IC25
74HC32 x 4/4
*WR
FN3
FN2
2G 2Y3
2B
2A
*USRWR
*ADRWR
*HOSTWR
J1
*WR#2
*RD#2
*RD
FN1
IC35
74HC139/2
*USEL
1G 1Y3
1B
1A
BA 15
BA 13
*CS #1
#0
74HC574 x 2
*CS#0L
*CS#0H
ADO 16
15
14
13
12
11
10
9
8D OC 8Q
7D 7Q
6D 6Q
5D 5Q
4D 4Q
3D 3Q
2D 2Q
1D CK 1Q
IC 17
IC 16
LE
WA11
10
9
8
74HC573 x 2
ADO 8
7
6
5
4
3
2
1
BA7
IC 15
IC 14
WA7
6
5
4
3
2
1
0


74HC541x 4
*CS#0
*HOSTWR
ADO 16
15
14
13
12
11
10
9
ADO 8
7
6
5
4
3
2
1
IC 9
IC 7
IC 3
IC 1
G1
G2
A8
A7
A6
A5
A4
A3
A2
A1
Y8
Y7
Y6
Y5
Y4
Y3
Y2
Y1
D 31
30
29
28
27
26
25
24
D 23
22
21
20
19
18
17
16
D 15
14
13
12
11
10
9
8
D 7
6
5
4
3
2
1
0


74HC574 x 2
*ENOSP
D 15
14
13
12
11
10
9
8
*WR1400
IC 5
OSP 15
14
13
12
11
10
9
8
D 7
6
5
4
3
2
1
0
IC 6
OSP 7
6
5
4
3
2
1
0
*RD1400
*ENISP
ISP 15
14
13
12
11
10
9
8
ISP 7
6
5
4
3
2
1
0



※ SP-SP i/F 32bit 化対応回路

5．試作システムのソフトウエア構成

本章では、試作した蝸牛フィルタバンクのソフトウエア構成について述べる。
試作システムのソフトウエアは、マスコンプ上の制御モニタプログラムとＤＳＰ上のフィルタリングプログラムの２つから成る。現在、２つのバージョンがあるが、データ値の妥当性の確認がとれているのは１つである。すなわち、モニタプログラム”mon1 ”と、フィルタリングプログラム”main00 ”の組合せである。
以下に、これらのプログラムについて説明する。

５－１　ホスト上のモニタプログラム

本プログラムはマスコンプから、ＰＩ１６Ｆに接続されたＤＳＰシステムを制御するモニタプログラムである。本プログラムはマスコンプのＲＴＵの上で、Ｃ言語により書かれている。

５－１－１　モニタコマンド

以下に、各コマンドについて説明する。

①　ｕコマンド（Unit Select)

ＤＳＰユニットの選択を行なう。各ユニットがあらかじめ持っているユニット番号と、本コマンドで指定された値が一致したユニットが選択状態となる。
入力値域は、１６進で０００００ＨからＦＦＦＦＨまででであるが、８０００ＨからＦＦＦＦＨまでは全ユニット同時選択モードとなる。

②　ｒコマンド（Reset Unit)

選択状態にあるユニットにハードウエアリセット信号を送る。

③　ｉコマンド（Interupt Unit)

選択状態にあるユニットにマスカブル割り込み信号を送る。

④　ｌコマンド（Load Unit)

選択状態にあるユニットの外部インストラクションメモリ（１０００Ｈ－１ＦＦＦＨ）にマスコンプからオブジェクトプログラムをロードする。データ形式は７７２３０用オブジェクトコンバータ出力の、ＨＥＸ形式とする。ファイル名は属性まで含めてすべて入力する。

⑤　ｍコマンド（Main Memory Opcrarion)

選択状態にあるユニットの外部インストラクションメモリ（１０００Ｈ－１ＦＦＨ）のデータの操作を行なう。次の３つのサブコマンドがある。

（１）ｃサブコマンド（Change Memory by Word)
ワード単位でデータを書き込む。書き込むアドレスを指定しデータを入力する。”．”でスキップ、”／”で終了する。
（２）ｆサブコマンド（Fill Memory)
指定した２つのアドレス間を、指定したデータで満たす。
（３）ｑサブコマンド（Quit to the command mode)
コマンド待ち状態へ戻る。

⑥　ｃコマンド（Communicate between sharing memory）

選択状態にあるユニットの共有メモリのデータの操作を行なう。次の４つのサブコマンドがある。

（１）ｄサブコマンド（Dump Memory）
　指定した２つのアドレス間のデータを表示する。
（２）ｃサブコマンド（Change Memory by Word）
　ワード単位でデータを書き込む。書き込むアドレスを指定しデータを入力する。"．"でスキップ、"／"で終了する。
（３）ｆサブコマンド（Fill Memory）
　指定した２つのアドレス間を、指定したデータで満たす。
（４）ｑサブコマンド（Quit to the command mode）
　コマンド待ち状態へ戻る。

⑦　ｑコマンド（Quit to the shell）

モニタプログラムを終了し、シェルへもどる。

５－１－２　モニタプログラムの操作方法

モニタプログラムの操作は以下のように行なう。

①　"mon1"起動。モニタモードに入る。
②　"u"コマンドで、"ffff"ユニットを選択。全ユニットを選択状態にする。選択状態は、各ＤＳＰボード上のLEDの点灯で確認できる。
③　"s"コマンドで、選択されている全ボードのＤＳＰをRESET状態のままにする。RESET状態は、第一ユニット上のLEDの点灯で確認できる。このとき、各ＤＳＰはデータバス、アドレスバスを解放している。
④　ＤＳＰプログラムをロードするのに、"l"コマンドでロードモードに入る。サブコマンド"c"を入力後、フィルタリングプログラム名を要求してくるので、"newcfil.hex"を指定するとヘキサダンプが表示されロードされる。
⑤　フィルタパラメータをロードするのに、"l"コマンドでロードモードに入る。サブコマンド"f"を入力することにより、自動的にフィルターパラメータ"../notchNx"と"../bpfNx"がマスコンプ上のバッファに取り込まれ、次にユニット数とユニット当りのフィルタチャンネル数を要求してくるので、それぞれ例えば"３"と"３"を指定すると３チャンネルづつ３ユニットの９チャンネルのシステムとなる。その後、スタートのチャンネル番号を要求してくるので"62"を入力すると、62チャンネルから54チャンネルまでのパラメータが表示されながらロードされる。
⑥　"u"コマンドで、"ffff"ユニットを選択。全ユニットを選択状態にする。選択状態は、各ＤＳＰボード上のLEDの点灯で確認できる。
⑦　"r"コマンドで、選択されている全ボードのＤＳＰをRESET状態から解放し、プログラム実行させる。RESET 状態からの解放は、第一ユニット上のLEDの滅灯で確認できる。この後、各ＤＳＰは各部の初期化を行いROM内でループ状態に入っている。
⑧　フィルタリングの実行とデータ授受を行なうため、"g"コマンドを入力すると、まず、入力データファイル名を要求してくるので、ファイル名を属性を含めて入力する。次にユニット数とデータ数を要求してくるので、それぞれ例えば、"3"と"20"を入力すると、先頭の20のデータのフィルタリング結果が表示される。

本プログラムを運用するに当たっては、以下の点に注意する必要がある。すなわち、外部インストラクションメモリはライトのみ可能であるために、選択状態のユニットでは、外部インストラクションメモリへのアクセス時に、マスコンプとＤＳＰがバス衝突を生じる可能性がある。したがって、ｍおよびｌコマンド使用時にはｉまたはｒコマンドなどにより、ＤＳＰのアクセスを止める必要がある。また、ＤＳＰがプログラム実行中は、ユニットを選択してはいけない。

５－２　ＤＳＰ上のフィルタリングプログラム

５－２－１　概　要

試作したＤＳＰシステムでは、μPD77230はﾏｽﾀ･ﾓｰﾄﾞで使用され、外部インストラクションのメインプログラムと、内部マスクROM上のアプリケーションプログラムで動作する。外部インストラクション・アクセス・モードでは、アドレスポートA0～A11は、プログラムカウンタPC0～PC11のデータ出力に専有される。このため、アドレスポートを使用する命令、すなわち外部メモリのリード、ライト命令は、外部インストラクションに置くことはできない。したがって、マスクROM上の汎用の外部メモリのリード、ライトルーチンをＣＡＬＬしなければならない。

５－２－２　フィルタリング処理

カスケードパラレル型のフィルタリングを実現１つのブロック構成をFig.15 に示す。NOTCHおよびBPFのフィルタリング機能は２次のbiquad型で実現される。これらの基本ルーチンはμPD77230のマスクROM上にあり、入力データおよびフィルタ係数を与えることでフィルタリングが実行される。

それぞれのフィルタリングの実行には、以下の３つの処理が必要である。

①　フィルタ係数の設定（RAM0のアドレス指定）
②　ディレイの設定　　（RAM1のアドレス指定）
③　Biquad Filter演算プログラムの呼び出し

作成したプログラムでは、入出力データはRAM1を通じて伝達される。また、①②の各パラメータの指定に先立ち、インストラクションメモリ上にHOSTからダウンロードされた各パラメータのデータをRAM0、RAM1上に転送しなければならない。

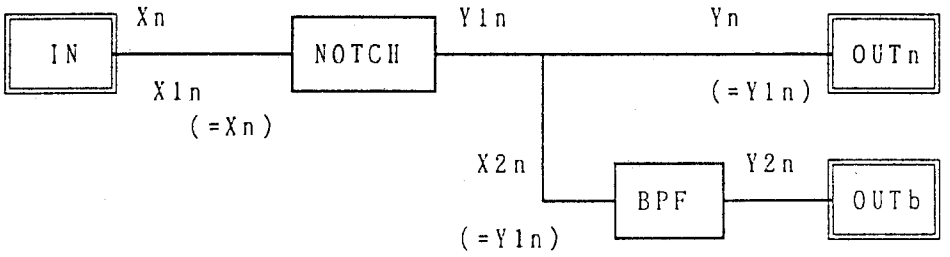


Fig.15　１段のカスケードパラレル型のフィルタリング処理

５－２－３　ＤＳＰプログラム

ＤＳＰ上のプログラム"main00.asm"は、MASSCOMPとの間の共有メモリを介してMASSCOMP経由で隣接ユニット間のデータ授受を行なう３チャンネルのフィルタリングプログラムである。各フィルタリングステージでは、マスクROM上のユーティリティを用いて２次のbi-quad演算を行い、入出力値の受渡しには、基本的に、ワーキングレジスタ "WR0"を用いている。

フィルタのパラメータは、事前に外部インストラクションメモリの0E10Hから一つのフィルタ当り５ワードづつノッチ、バンドパスの順序で格納しておく。ただし、MASSCOMP上でのパラメータの表現形式とＤＳＰ上での表現形式には差異があるためにあらかじめMASSCOMP上でフォーマットの変換をする必要がある。これについては次節で詳しく述べる。

作成したＤＳＰプログラムの動作を以下に示す。

ハードウエアをリセットした後、プログラムは０番地から実行が開始されるが、マスタモードではマスクROM内のプログラムは無条件に1000番地へジャンプする。

1000番地からの"main00.asm"の初期化部で、前述の外部メモリ上のパラメータを内部の汎用メモリファイルRAM0の1EFHからアドレス低位へ向かって転送する。ここで、アドレスの方向が変化している点に注意する必要がある。この操作により、ＤＳＰのアプリケーション・ノート252頁のメモリロケーションを実現している。なお、フィルタのディレイバッファは同様に内部メモリファイルRAM1をクリアした後、1EFHから各フィルタ当り２ワードづつ割り当てている。

以上の処理後、プログラムはマスクROM内の無限ループルーチンへジャンプし、待機状態に入る。

ここで、MASSCOMPから割り込み信号 "INT"をＤＳＰへ与えるとプログラムはマスクROM内の100番地へ実行を移し、マスタモードでは無条件に1100番地へジャンプする。1100番地からはフィルタリングの本体プログラムが入っており、その処理終了後に再び ROM内の無限ループルーチンへジャンプする。この後 "INT"信号を与えるごとにフィルタリングが実行される。

メインルーチンでは、まずフィルタ中のBUSYフラグとして共有メモリの先頭ワード（ＤＳＰ側1000H番地、MASSCOMP側 8000H番地：以下ＤＳＰ側のみで記述する）をクリアする。MASSCOMPはこのワードを監視し、値が0000Hの間はフィルタリングデータを取り込まない。

各ユニットへの入力データ（32bit）はMASSCOMPから、共有メモリの1002H番地（上位16bit）と1003H番地（下位16bit）に与えられているので、まずこれを32bitの形式に変換する。μPD77230の命令には直接 32bitワード上で16bitをシフトする命令がないためサブルーチン"sftl16"を用いてこれを実現している。

次にノッチ、バンドパスの順にフィルタリングを行なう。各チャンネルのノッチフィルタの出力はRAM1上の1F0H番地から格納される。すなわち、そのユニット上の第一チャンネルのノッチフィルタは入力"WR0"、出力"RAM1:1F0H"であり、第ｎチャンネルのノッチフィルタでは入力"RAM1:1F0H+n-2"、出力"RAM1:1F0H+n-1"である。また、そのユニットの最終チャンネルのノッチフィルタの出力は、次段のユニットへ渡すため、共有メモリの1004H 番地（上位16bit）と1005H番地（下位16bit）へ格納される。

各チャンネルのバンドパスフィルタの出力は1006H番地（上位16bit）と 1007H番地（下位 16bit）から順に格納される。すなわち、第ｎチャンネルのバンドパスフィルタの出力は1006H+n-1番地（上位16bit）と1007H+n-1番地（下位16bit）に格納される。なお、各出力（32bit）の共有メモリへの格納は前述したように16bitシフトサブルーチン"sftr16"を用いて行なっている。

以上の処理終了後、BUSYフラグである1000H番地に00FFHを書いて、マスクROM上の無限ループへジャンプし、待機状態に入る。

5－3　フィルタ係数のフォーマット

マスコンプ上における各フィルタの係数は以下に示す形式でアスキーファイルとして与えられている。

```
chno  a0 a1 a2 b0 b1 b2
```

ここで、chnoはフィルタのチャンネル番号で整数で与えられる。a0,a1,a2,b0,b1,b2は次式及びFig.16に示されるようなフィルタの係数で、floatで与えられる。

$$H(z) = \frac{b0 + b1z^{-1} + b2z^{-2}}{a0 + a1Z^{-1} + a2z^{-2}} \qquad (6)$$

where a0 = 1

一方、ＤＳＰシステムのマスクＲＯＭ内のBiquad Filter Module のフィルタ係数をA0,A1,A2,B1,B2 とすると、それらは次式及びFig.16(b)で与えられる。

$$H(z) = A0 \cdot \frac{1 + A1z^{-1} + A2z^{-2}}{1 - B1z^{-1} - B2z^{-2}} \qquad (7)$$

ここで、両式の右辺同士を等しいと置くと次の関係が得られる。

$$\begin{aligned} A0 &= b0 \\ A1 &= b1/b0 \\ A2 &= b2/b0 \\ B1 &= -a1 \\ B2 &= -a2 \end{aligned} \qquad (8)$$

したがって、ＤＳＰシステムへは、１０進浮動小数点形式で与えられるこの５つのパラメータ（A0,A1,A2,B1,B2 ）を３２ビット２進浮動小数点形式に変換したものを転送すれば良い。

フィルタパラメータの一例とデータフォーマット変換プログラムを付録に示す。

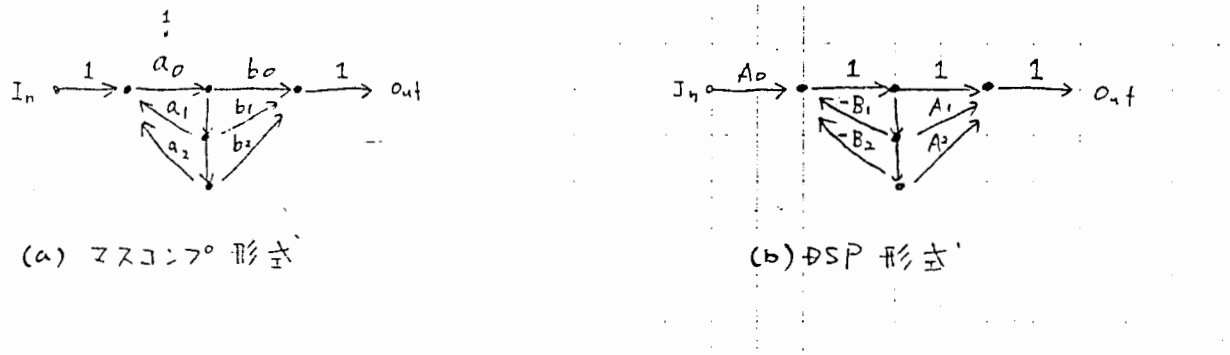


Fig.16　フィルタ係数の形式

6. 試作システムの問題点

以上、試作した第1次システムのハード・ソフト両面について述べてきた。本システムの試作によって、マルチDSPによる基本的なフィルタリング処理の確認ができたが、ホストとのインターフェイスにおいていくつかのシステム設計上の問題点があることも明らかになった。本章ではこの問題点について述べる。

マスコンプの汎用パラレルインターフェース PI16Fは、大量のデータブロックを一括転送するときに有効に機能する。本システムでは、数十から数百のユニットを切り替えながら、数ワードから数十ワードの小ブロックのデータを収集することを要求したため、データ転送に必要な準備のオーバヘッドの割合が増大し、その結果、DSPシステムとホスト間のデータ通信処理速度が上がらなかった。

また、PI16Fは16ビット幅であるため、試作システムのDSPであるμPD77230のデータ幅32ビットとは整合性が良くない。すなわち、試作システム上では各データは32ビットの浮動小数点で表されているために、それらを転送するためには、上位下位16ビットに分割合成する必要があり、その結果、さらにオーバヘッドが大きくなっている。

さらに、PI16Fのデータ入出力は2ワード（x16ビット）が基本単位になっているため、デュアルポートメモリをアドレス自動インクリメントモードで使用する場合に不都合である。

ハードウエアの問題点としては以下の点が上げられる。

試作システムでは、メモリのリード信号として、PI16Fのハンドシェーク用のリードストローブ信号を用いたが、この方法では、以下に述べるような問題があることが解った。すなわち、ハンドシェーク時にはポートデータが確定した後に信号を発生するのに対し、メモリではリード信号入力時からアクセスタイムが経過した後にデータが確定する。このため、PI16F を通じて試作システムのメモリからデータ読む際に、データの読み誤りが頻発した。この問題を解決するために2つの方法を試みた。

まず、第一の方法として、PI16F では2つのリードストローブが組になって発生するのを利用する方法がある。すなわち、初めのストローブでメモリリード信号をオン、後のストローブでそれをオフする。そして、後のストローブ信号が発生する時刻においてデュアルポートメモリの出力は確定されているので、対応するマスコンプ上のバッファにメモリー上のデータが得られるはずである。しかし、この方法では、DSPシステムのバス上ではデータが確定しているにもかかわらずマスコンプ上ののバッファにはデータが正しく取り込まれなかった。（全てFFFFHとなった）原因は、不明である。

第二の方法としては、メモリーのライト時以外では常にリード状態にしておく方法がある。ただし、デュアルポートメモリはパルス状のリード信号が必要なので、この方法を用いる場合にはリードごとにデバイスセレクトをオフするようにソフト的にアドレス制御する必要がある。この方法では、リードエラーは起きなかったが、基本的な問題が生じた。すなわち、1ワードごとにセレクト信号がオフ・オンするようにアドレスを変えなければリードパルスが与えられないために、デュアルポートメモリの機能の一つである自動アドレス・インクリメントモードでの間接リードを用いてブロックリードをすることができない。このため、リード時のオーバヘッドが大きくなってしまった。

以上の問題は、マスコンプの汎用パラレル入出力の設計が、相手側としてインテリジェント化されたシステムを、想定していることに起因している。したがって、本システムもDSPユニット群の出力データをマスコンプへのブロック転送に的した様式に整え、また高速なハンドシェークを行なうインテリジェントインターフェースを設計し直す必要がある。

7．まとめ

本稿では、多チャンネルの蝸牛フィルタの実時間処理を実現することを目的として試作したＤＳＰシステムについて述べてきた。今回の第一次試作システムでは、当初予定していた実時間処理を実現することはできなかったが、基本的なシステム設計、多数のＤＳＰを用いたシステムによる高速フィルタリング処理の基本動作の確認、及び、ホストインターフェイスの問題点を明らかにすることができた。

今後は、今回の試作で明らかになった問題について更に検討を加え、第二次の試作を行うとともに、実用システムの開発をすすめる予定である。

参考文献

〔1〕 駒木根、平原 (1987)
「蝸牛の周波数分析機能を模擬するフィルターバンクの一構成法」
電子情報通信学会技術報告 SP87-45, pp.65-72

〔2〕 駒木根、平原 (1987)
「蝸牛の周波数分析機能を模擬するフィルタバンク － Cascade/Parallel 型の構成 －」
日本音響学会講演論文集 昭和62年10月 pp.279-280

〔3〕 駒木根、平原 (1988)
「ディジタル蝸牛フィルタのハードウエア構成」
日本音響学会講演論文集 昭和63年 3月 pp.321-322

〔4〕 T.Komakine and T.Hirahara (1988)
"An Auditory Filterbank Design and its Hardware Implementation with DSP", J.Acoust.Soc.Am. Suppl.1, 83, p.S18

〔5〕 平原、駒木根 (1988)
「適応Q機能を持つ非線形蝸牛フィルタ」
昭和63信学会全国大会予稿 pp. A-1-109 - 110

〔6〕 平原、駒木根 (1988)
「適応Q型非線形蝸牛フィルタの特性」
日本音響学会講演論文集 昭和63年10月 pp.363-364

〔7〕 平原、駒木根 (1988)
「適応Q機能を持つ蝸牛フィルタモデル」
日本音響学会 聴覚研究会資料 H-88-33

〔8〕 浜田、平原、今村 (1988)
「臨界帯域幅フィルタ分析による子音認識－ＬＰＣ分析との比較－」
昭和63信学会全国大会予稿 pp. A-1-115 - 116

〔9〕 原田、河原田 (1986)
「音声の高分解能周波数分析－/mi/,/ni/の特徴抽出－
電子情報通信学会論文誌 Vol. J69-A, No.5, pp.620-627

〔10〕 「フィルター設計とＤＳＰ入門」
インターフェース 1987年11月号

〔11〕 「はじめてのディジタル信号処理」
インターフェース 1986年4月号

〔12〕 「TMS320活用とＤＳＰシステム活用」
インターフェース 1988年3月号

〔13〕 「μPD77230 ユーザーズ・マニュアル」
「μPD77230 シミュレータ ユーザーズ・マニュアル」
「μPD77230 アプリケーション・ノートⅠⅡⅢ」
日本電気株式会社 1987

〔14〕 「HD63310R 資料」（株）日立製作所 1987.9

〔15〕 「EVAKIT-77230 取扱説明所」日本電気株式会社 1986.10.1

〔16〕 「PI16F Hardware Mannual」
「Data Acquisision Application Programming Mannual」 Masscomp

付　録

```
/* cc -o xxxx xxxx.c -lmr -lm  [ on MASSCOMP ] */
/* ---------------------------------------------------------*/
/********* New DSP monitor program   ********************
                                    1988/02/12 by T.K.
              ureg.c                1988/02/18
               mon.c                1988/03/08
              nmon.c                1988/03/08
              newm.c                1988/10/05
              load func is added    1988/10/13
**********************************************************/
/* ---------------------------------------------------------*/

#include <stdio.h>
#include <mr.h>
FILE *fd,*fopen();

/* ---------------------------------------------------------*/

main()
{
        char ans[10];
        unsigned unum;

        printf("\nWelcome to monitor island!!!!!!!\n");
   do(
        printf("\n\nENTER COMMAND [? is HELP] >>>");
        scanf("%s",ans);
        if(strcmp(ans,"?")==0)
                printf("\n[m]EMORY,[c]OMMON,[r]ESET,[u]NIT,[l]oad,[i]nt,[q]UIT\n\n");

        if(strcmp(ans,"r")==0)
        {
                pfio(0x7ffb,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0xfffb,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0x7fff,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0xffff,4); /* Interupt reg. address: 4000H~7FFFH */
                /*pfio(0xfffb,1);*/ /* RES:L,INT:H,NMI:H */
                /*pfio(0xffff,1);*/ /* RES:H,INT:H,NMI:H */
        }
        if(strcmp(ans,"s")==0)
        {
                pfio(0x4000,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0x8000,4); /* Interupt reg. address: 4000H~7FFFH */
        }
        if(strcmp(ans,"i")==0)
        {
                pfio(0x7ffd,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0xfffd,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0x7fff,4); /* Interupt reg. address: 4000H~7FFFH */
                pfio(0xffff,4); /* Interupt reg. address: 4000H~7FFFH */
        }

        if(strcmp(ans,"m")==0)
                mem();

        if(strcmp(ans,"u")==0)
        {
                printf("\n>>NEW UNIT No. [Hex]=");
                scanf("%x",&unum);
                pfio(unum,6);
        }

        if(strcmp(ans,"c")==0)
                com();

        if(strcmp(ans,"l")==0)
                load();

    )while(strcmp(ans,"q")!=0);
    mrclosall();
    printf("\nSEE YOU AGAIN!!!.....\n\n");
}

/* ---------------------------------------------------------*/
mem()
{
        char ans[10];
        unsigned addr= -1,data= -1;
        unsigned saddr,eaddr;
        unsigned long ldata;
   do(
        printf("\n\nI.M. access COMMAND<<[c]HANGE,[f]ULL>>>>=");
        scanf("%s",ans);
        if(strcmp(ans,"c")==0)
```

```
        {
                printf("\nSTART ADDRESS [Hex word:000~FFF]= ");
                scanf("%x",&addr);
                do{
                        printf("%04x  ",addr);
                        pfio(addr,4);   /*adrset(addr);*/
                        printf("xxxxxxxx   ->");
                        ldata=gethexnum();
                        printf("%d",ldata);
                        if(ldata>=0)
                        {
                                pfio(ldata/0x10000,3);
                                pfio(ldata%0x10000,2);
                        }
                        if(ldata!= -3)
                                addr++;
                }while(ldata!= -2);
        }
        if(strcmp(ans,"f")==0)
        {
                printf("\nSTART ADDRESS [Hex word:000~FFF]= ");
                scanf("%x",&saddr);
                printf("\nEND ADDRESS   [Hex word:000~FFF]= ");
                scanf("%x",&eaddr);
                printf("\nFULL DATA [Hex word:00000000~FFFFFFFF]= ");
                scanf("%lx",&ldata);
                for(addr=saddr;addr<=eaddr;addr++)
                {
                        pfio(addr,4);   /*adrset(addr);*/
                        pfio(ldata/0x10000,3);
                        pfio(ldata%0x10000,2);
                }
        }
    )while(strcmp(ans,"q")!=0);
}

/* ------------------------------------------------------*/
com()
{
        char ans[10],c_buf;
        unsigned addr= -1,data= -1;
        unsigned saddr,eaddr;
        unsigned short idata;
        int err_flg=0;
        long ldata;
   do{
        printf("\n\nC.M. access COMMAND<<[d]UMP,[c]HANGE,[f]ULL>>>>=");
        scanf("%s",ans);
        if(strcmp(ans,"d")==0)
        {
                printf("\nSTART ADDRESS [Hex word:000~1FF]= ");
                scanf("%x",&saddr);
                printf("\nEND ADDRESS   [Hex word:000~1FF]= ");
                scanf("%x",&eaddr);
                for(addr=saddr;addr<=eaddr;addr++)
                {
                        if(addr%8==0 || addr==saddr)
                                printf("\n%04x  ",addr);
                        pfio(0x8000+addr,4);    /*adrset(0x8000+addr);*/
                        idata=pfio(0,0);
                        printf("xxxx%04x ",idata);
                }
        }
        if(strcmp(ans,"c")==0)
        {
                printf("\nSTART ADDRESS [Hex word:000~1FF]= ");
                scanf("%x",&addr);
                do{
                        printf("%04x  xxxx",addr);
                        pfio(0x8000+addr,4);    /*adrset(0x8000+addr);*/
                        idata=pfio(0,0);
                        printf("%04x  ",idata);
                        printf(" -> xxxx");
                        /*scanf("%x",&ldata);*/
                        ldata=gethexnum();
                        if(ldata>=0)
                                pfio(ldata,2);
                        if(ldata!= -3)
                                addr++;
                }while(ldata!= -2);
        }
        if(strcmp(ans,"f")==0)
        {
                printf("\nSTART ADDRESS [Hex word:000~1FF]= ");
```

```
                scanf("%x",&saddr);
                printf("\nEND ADDRESS   [Hex word:000~1FF]= ");
                scanf("%x",&eaddr);
                printf("\nFULL DATA [Hex byte:00~FF]= ");
                scanf("%x",&data);
                for(addr=saddr;addr<=eaddr;addr++)
                {
                        pfio(0x8000+addr,4);    /*adrset(0x8000+addr);*/
                        pfio(data,2);
                }
        }
    }while(strcmp(ans,"q")!=0);
}
/* ----------------------------------------------------------*/

gethexnum()
{
        long nc=0,sum=0;
        int c=0;
        for(c=0;c<10000;c++)
                c=c     ;
        while(sum>=0)
        {
                c=getchar();
                if((c=='\n')&&(nc!=0))
                        break;
                /*if((c=='\n')&&(nc==0))*/
                if(c=='.')
                        sum= -1;
                if(c=='/')
                        sum= -2;
                if((c>='0')&&(c<='9'))
                        sum=sum*16+c-'0';
                if((c>='a')&&(c<='f'))
                        sum=sum*16+c-'a'+10;
                nc++;
        }
        /*printf("\nsum=%d%d",sum,nc);*/
        return(sum);
}

/* ----------------------------------------------------------*/

/* A program to test the PI16F in loop back mode
               copied from pf2.c
               chg to mrpdinout from mrxinoutq
                    1988/09/30  T.KOMAKINE */

pfio(outdata,fndata)
        unsigned short outdata;
        int fndata;
{
        unsigned short  inarray[2], outarray[2];
        int outdev = -1, indev = -1;
        int outstat[2], instat[2];

        outarray[0]=outdata;
        outarray[1]=outdata;

        mropen(&indev,"/dev/dacp0/pfi0",0);
        mropen(&outdev,"/dev/dacp0/pfo0",0);

        mrpfomod(outdev,1,1,1,0,1,fndata,0);
        mrpfimod(indev,1,1,1,0,1,fndata,0);

        mrpdxin(indev,2,inarray);
        mrpdxout(outdev,2,outarray);

        mrevwt(indev,instat,1000);
        mrevwt(outdev,outstat,1000);

        mrclosall();
        return(inarray[0]);
}

/*----------------------------------------------------------
  Loading Hex type object to MASSCOMP

                    1988/10/12  T.KOMAKINE
----------------------------------------------------------*/
/*----------------------------------------------------------
  Input record is as follows
    : nn aaaa 00 DD.....DD SS       nn is the No. of data (byte)
                .                   aaaa is byte addr
```

```
                .                       DD is object data
                .                       SS is check sum code
    : 00 0000 01            FF          These code is ASCII char. code
-------------------------------------------------------*/

load()
{
        char hexfname[50],cans[10];
        char inbuf[100];
        int i,j,numbyte,c;
        unsigned int obuf,adr;

        printf("\nDistnation ? Rom[1] or Ram[2] ");
        scanf("%d",&c);
        printf("\n--------- Available files ----------\n");
        system("ls *.hex");
        printf("-------------------------------------\n");
        printf("\nFile name = ");
        scanf("%s",hexfname);
        if((fd=fopen(hexfname,"r"))==NULL)
        {
                printf("***** File not found *****\n");
                exit(0);
        }

        fscanf(fd,"%s",inbuf);
    while(ghexnum(inbuf[7])*16+ghexnum(inbuf[8])==0)
    {
        printf("\n");
        adr=0;
        for(i=3;i<7;i++)
                adr=adr*16+ghexnum(inbuf[i]);

        if(c==1)
                printf("%04x\t%04x\t",adr/4+0x1000,adr/4);
        else    printf("%04x\t",adr/4-0x1000);
        printf("%x\t",(ghexnum(inbuf[1])*16+ghexnum(inbuf[2]))/4);

        for(i=0;i<(ghexnum(inbuf[1])*16+ghexnum(inbuf[2]))/4;i++)
        {
                obuf=0;
                for(j=0;j<8;j++)
                        obuf=obuf*16+ghexnum(inbuf[i*8+9+j]);

                printf("%08x ",obuf);
                if(c==1)
                        pfio(adr/4+i,4);
                else    pfio(adr/4-0x2000+i,4);
                pfio(obuf/0x10000,3);
                pfio(obuf%0x10000,2);
        }
        fscanf(fd,"%s",inbuf);
    }
    printf("\n");
}

/*   converting ASCII to HEX    */
ghexnum(cha)
        char cha;
{
        int num;

        if((cha>='0')&&(cha<='9'))
                num=cha-'0';
        if((cha>='A')&&(cha<='F'))
                num=cha-'A'+10;
        if((cha>='a')&&(cha<='f'))
                num=cha-'a'+10;
        return(num);
}
```

```
#include <math.h>

main()
{
        double bftrans();
        double inp,outq;
        unsigned long int inq,outp;
        int a;

        printf("\ffloat-->binary..[1]    binary-->float..[2]  ???=");
        scanf("%d",&a);
        printf("input data  = ");
        if(a==1)
        {
                scanf("%lf",&inp);
                outp=fbtrans(inp);
                printf("output data = ");
                printf("%08x\n",outp);
        }
        if(a==2)
        {
                scanf("%x",&inq);
                outq=bftrans(inq);
                printf("output data = ");
                printf("%lf\n",outq);
        }
}

/*--------------------------------------
  Trans. float number to
       floating binary 32 bit word
                  for mPD77230
                        1988/10/19 T.K.

        input... decimal floating number
        output.. 32bit binary float
--------------------------------------*/

fbtrans(inp)
        double inp;
{
        double pow(),log10(),fabs(),kasuu,m=1.0,n;
        int sisuu;
        unsigned long int outp=0;

        if(inp!=0.0)    sisuu=log10(fabs(inp))/log10(2.0)+1;
        else            sisuu= -128;
        kasuu=fabs(inp)/pow(2.0,(float)sisuu);
        if((kasuu<0.5)&&(kasuu!=0))
        {
                sisuu-=1;
                kasuu*=2.0;
        }
        if(sisuu<=0)    sisuu=256+sisuu;
        outp=sisuu*2;
        if(inp<0.0)
        {
                kasuu=1.0-kasuu;
                outp+=1;
        }
        for (n= -1.0;n>= -23.0;n-=1.0)
        {
                outp*=2;
                kasuu=kasuu-pow(2.0,n);
                if(kasuu>=0.0)  outp+=1;
                else    kasuu=kasuu+pow(2.0,n);
        }
        return(outp);
}

/*--------------------------------------
  Trans. float binary 32 bit word
       to float decimal
                  for mPD77230
                        1988/10/21 T.K.

        input.. 32bit binary float
        output... decimal floating number
--------------------------------------*/

double
bftrans(inp)
        unsigned long int inp;
{
```

32bit 2進表示
(μPD77230フォーマット)
と
10進浮動小数点
表示
の
相互変換
プログラム

```
        double pow(),m=1.0,n,outp=0.0;
        int sisuu;
        unsigned long int kasuu;

        sisuu=inp/0x1000000;
        kasuu=inp%0x1000000;
        /*printf("\n%x %x\n",sisuu,kasuu);*/

        if(sisuu>0x80)  sisuu=sisuu-0x100;
        if(kasuu>0x800000)
        {
                kasuu=0x1000000-kasuu;
                m= -1.0;
        }
        for(n= -1.0;n>= -23.0;n-=1.0)
        {
                /*printf("%f  ",kasuu/pow(2.0,23.0+n));*/
                if(kasuu/pow(2.0,23.0+n)>=1.0)
                {
                        outp+=pow(2.0,n);
                        /*printf(" <%f> ",outp);*/
                        kasuu=(int)kasuu%(int)pow(2.0,23.0+n);
                }
        }
        /*printf("\n%f\t%f\t%d",m,outp,sisuu);*/
        outp=m*outp*pow(2.0,(double)sisuu);
        /*printf("\n%f",outp);*/
        return(outp);
}
```

10進 → 2進 の時は [1]

```
hm03Koma@54>chgflo

float-->binary..[1]     binary-->float..[2]  ???=1
input data   = 32000
output data = 0f7d0000
```

2進 → 10進 の時は [2]

```
hm03Koma@55>chgflo

float-->binary..[1]     binary-->float..[2]  ???=2
input data   = 0f7d0000
output data = 32000.000000
```

```
;
;-----------------------------------------*
;   Main routin for 1st unit filtering
;     Clear RAM0 AND RAM1
;     Filter parameter set to RAM0 from EX-memory
;     Wait for INT to do filtering
;     Filtering 3 channels
;                                  1988/08/18
;   data area is changed           1988/10/21
;   16bit shift func is added      1988/10/24
;                            T.KOMAKINE
;-----------------------------------------*
;
        name    main2
;================================
; MODULE ENTRY ON MASKED ROM
;================================
raminie equ     017h            ;RAM0,1 clr prog. entry on ROM
htra2e  equ     038h            ;Trans. Ex.mem to RAM0 prog. entry on ROM
waitie  equ     0b8h            ;Wait and INT enable prog. entry on ROM
lmpoe   equ     375h            ;(WR0)->(AR) prog. entry adr. on ROM
lmpie   equ     327h            ;(WR0)<-(AR) prog. entry adr. on ROM
bfe     equ     4f2h            ;Bi-quad func. entry adr. on ROM
;
;================================
; MEMORY LOCATION
;================================
wcntra0 equ     200h            ;The number of trans. EX-mem -> RAM0
sadrexm equ     0e00h           ;EX-mem trans start addr. (E00H~FFFH)
eadrra0 equ     1ffh            ;RAM0 end address   (1FFH~000H)
HOSPCOM equ     1000h           ;HOST-DSP com. mem top adr.
SPSPCOM equ     1400h           ;DSP-DSP com. port adr.
;
IOM     equ     1f0h            ;DATA buff adr. on RAM1
;
;================================
; FILTER PARAMETER TABLE
;================================
CNF0    equ     1efh            ;Notch0 para. top adr. on RAM0
CBF0    equ     CNF0-5          ;BPF0 para. top adr.
CNF1    equ     CNF0-10         ;Notch1 para. top adr.
CBF1    equ     CNF0-15         ;BPF1 para. top adr.
CNFe    equ     CNF0-20         ;Notche para. top adr.
CBFe    equ     CNF0-25         ;BPFe para. top adr.
;
;================================
; FILTER DELAY TABLE
;================================
DNF0    equ     1efh            ;Delay beff top adr. for Notch0 on RAM1
DBF0    equ     DNF0-2          ;Delay beff top adr. for BPF0
DNF1    equ     DNF0-4          ;Delay beff top adr. for Notch1
DBF1    equ     DNF0-6          ;Delay beff top adr. for BPF1
DNFe    equ     DNF0-8          ;Delay beff top adr. for Notche
DBFe    equ     DNF0-10         ;Delay beff top adr. for BPFe
;
        ram0seg at 0h           ;
;
        ds      200h            ;
;
        ram1seg at 0h           ;
;
        ds      200h            ;
;
;-----------------------------------------------------------------
        imseg   at 1000h        ;Main routin start after RESET
;
;*************************************
;   Load parameter to RAM0,1
;         from EXT memory
;*************************************
;
        ei                      ;interupt enable
;
        call    raminie         ;RAM0,1 clear
        nop                     ;
;
;..now parameters heve been set on EXM by DMA while DSP is in RESET mode.
;    then they must be moved to RAM0
;
        ldi     lc,wcntra0-1    ;set the number of translation(1FFH)
        ldi     ix0,eadrra0     ;set the end address of RAM0(1FFH)
        ldi     ar,sadrexm      ;set the start address of EXM(E00H)
        call    htra2e          ;Parameter load RAM0 <- EXT.mem
        spcix0                  ;
```

```
;
;..now all things which must be done before filtering, was finished.
;   so, it must be waiting for the start signal from the HOST.
;
        call    waitle          ;Loop & waiting the first INT
        nop                     ;
;
;-------------------------------------------------------------------
        imseg   at 1100h        ;Program start after INT
;
;**************************************
;   Filtering
;**************************************
;
;..at the begining of filtering, function start flag must be set.
;    so that the HOST does not read the old data.
;
        ldi     wr0,000h        ;
        ldi     ar,HOSPCOM      ;
        call lmpoe              ;filtering start flg. 1000h=0
        nop                     ;
;
;++++++++++++++++++++++++++++++++++++++++++++++++
;   Sampled input data get
;       data is from EXM (0FFFH)
;++++++++++++++++++++++++++++++++++++++++++++++++
;
;..the input data is always set at the common memory.
;    it is devided to low 16bit (1003H) & high 16bit (1002H).
;
        ldi     ar,HOSPCOM+3    ;INPUT data LOW is here 1003H
        call    lmpie           ;(WR0)<-(AR)=(EXM:0FFFH)
        nop                     ;
        ldi     wr2,0ffffh      ;16bit mask (0000 0000 0000 0000
                                ;                1111 1111 1111 1111)
        and     wr2,ib
          mov   non,wr0         ;mask the high 16bit
        ldi     ar,HOSPCOM+2    ;INPUT data HIGH is here 1002H
        call    lmpie           ;(WR0)<-(AR)=(EXM:0FFFH)
        nop                     ;
        call    sft116          ;Input 16 bit shift LEFT to higher 16 bit
        nop                     ;
        or      wr0,ib
          mov   non,wr2         ;
;
;++++++++++++++++++++++++++++++++++++++++++++++++
;   Notch(0)
;     input data is in WR0
;     filtering result is WR0 & RAM1(IOM)
;    Data sending to next STAGE
;++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     ix1,DNF0        ;
        ldi     ix0,CNF0        ;
;
        call    bfe             ;Result -> (WR0)
        nop                     ;
;
        ldi     ix1,IOM         ;
        mov     ram1,wr0        ;(IOM:RAM1)<-(WR0)
;
;++++++++++++++++++++++++++++++++++++++++++++++++
;   BPF(0)
;     input data is in WR0
;     filtering result is in WR0
;    Data sending to HOST
;     port adr. for HOST is low speed EXT-mem(1000H)
;++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     ix1,DBF0        ;
        ldi     ix0,CBF0        ;
;
        call    bfe             ;Result -> (WR0)
        nop                     ;
;
        ldi     ar,HOSPCOM+7    ;OUTPUT 1st ch Low -> EXM 1007H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
        call    sftr16          ;Top 16 bit shift Right to next 16 bit
        nop                     ;
        ldi     ar,HOSPCOM+6    ;OUTPUT 1st ch High -> EXM 1006H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
;
```

```
;+++++++++++++++++++++++++++++++++++++++++++++++++
;   Notch(1)
;     input data is in WR0
;     filtering result is WR0 & RAM1(IOM+ch No.)
;    Data sending to next STAGE
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     ix1,IOM         ;
        mov     wr0,ram1        ;(WR0)<-(IOM:RAM1)
;
        ldi     ix1,DNF1        ;
        ldi     ix0,CNF1        ;
;
        call    bfe             ;Result -> (WR0)
        nop                     ;
;
        ldi     ix1,IOM+1       ;
        mov     ram1,wr0        ;(IOM:RAM1)<-(WR0)
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;   BPF(1)
;     input data is in WR0
;     filtering result is in WR0
;    Data sending to HOST
;     port adr. for HOST is low speed EXT-mem(1000H)
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     ix1,DBF1        ;
        ldi     ix0,CBF1        ;
;
        call    bfe             ;Result -> (WR0)
        nop                     ;
;
        ldi     ar,HOSPCOM+9    ;OUTPUT 2nd ch Low -> EXM 1009H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
        call    sftr16          ;Top 16 bit shift Right to next 16 bit
        nop                     ;
        ldi     ar,HOSPCOM+8    ;OUTPUT 2nd ch High -> EXM 1008H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;   Notch(e)
;     input data is in IOM+previous ch No.
;     filtering result is WR0
;    Data sending to next UNIT
;     port adr. for SP is low speed EXT-mem(1400H)
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     ix1,IOM+1       ;
        mov     wr0,ram1        ;(WR0)<-(IOM+1:RAM1)
;
        ldi     ix1,DNFe        ;
        ldi     ix0,CNFe        ;
;
        call    bfe             ;Result -> (WR0)
        nop                     ;
        mov     wr2,wr0         ;
;
;       ldi     ar,0ffeh        ;NEXT STAGE INPUT IS IN EXM 0FFEH
;       call    lmpoe           ;(WR0)->(AR):SP-SP port
;       nop                     ;
        ldi     ar,HOSPCOM+5    ;OUTPUT to next unit Low -> EXM 1005H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
        call    sftr16          ;Top 16 bit shift Right to next 16 bit
        nop                     ;
        ldi     ar,HOSPCOM+4    ;OUTPUT to next unit High -> EXM 1004H
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;   BPF(e)
;     input data is in WR0
;     filtering result is in WR0
;    Data sending to HOST
;     port adr. for HOST is low speed EXT-mem(1000H)
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
        mov     wr0,wr2         ;
        ldi     ix1,DBFe        ;
        ldi     ix0,CBFe        ;
;
```

```
        call    bfe             ;Result -> (WRO)
        nop                     ;
;
        ldi     ar,0ffbh        ;OUTPUT 3rd ch -> EXM 0FFBH
        call    lmpoe           ;(WR0)->(AR):FIFO
        nop                     ;
        ldi     ar,HOSPCOM+0bh  ;OUTPUT 3rd ch Low -> EXM 100BH
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
        call    sftr16          ;Top 16 bit shift Right to next 16 bit
        nop                     ;
        ldi     ar,HOSPCOM+0ah  ;OUTPUT 3rd ch High -> EXM 100AH
        call    lmpoe           ;(WR0)->(AR)
        nop                     ;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;   Waiting for next sample
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
        ldi     wr0,0ffh        ;
        ldi     ar,HOSPCOM      ;
        call    lmpoe           ;filtering end flg 1000h=0ffh
        nop                     ;
;
        call    waitle          ;Loop & waiting for next INT
        nop                     ;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;  Shift 16 bit right ( just 16 bit are available )
;   Input:  xxyynnmmppqqrr (hex55bit in wr0)
;   Output: xx00xxyynnmmpp (hex55bit in wr0)
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
sftr16:
        wrbel8                  ;set mode 54~47 bit to 7~0 bit trans.
        mov     wr1,wr0         ;
        shlm    wr1
          setsvl        1fh     ;shift 31bit left as higher 8bit
        shrm    wr0
          setsvr        10h     ;shift 16bit right as lower 8bit
        wrbord                  ;reset trans. mode
        or      wr0,ib
          mov   non,wr1
            setsvl      00h     ;make 16bit & reset shift mode
        ret                     ;
        nop                     ;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++
;  Shift 16 bit left ( just 16 bit are available )
;   Input:  xxyynnmmppqqrr (hex55bit in wr0)
;   Output: nnmm0000000000 (hex55bit in wr0)
;+++++++++++++++++++++++++++++++++++++++++++++++++
;
sftl16:
        mov     wr1,wr0         ;
        ldi     wr0,0ffffh      ;
        and     wr1,ib
          mov   non,wr0         ;
        mov     wr0,wr1         ;
        shlm    wr1
          setsvl        10h     ;shift 16bit left as lower 8bit
        shrm    wr0
          setsvr        1fh     ;shift 31bit right as higher 8bit
        wrbl8e                  ;set mode 7~0 bit to 54~47 bit trans.
        mov     wr0,wr0         ;
        wrbord                  ;reset trans. mode
        or      wr0,ib
          mov   non,wr1
            setsvl      00h     ;make 16bit & reset shift mode
        ret                     ;
        nop                     ;
end
:
```

	64	1.000000	1.590270	0.727222	0.877387	1.624186	0.815919	0.504700
	63	1.000000	1.512377	0.786334	0.907959	1.561501	0.829251	0.641372
	62	1.000000	1.358201	0.788871	0.900713	1.419064	0.827297	0.717360
	61	1.000000	1.182254	0.782403	0.890216	1.252708	0.821733	0.765213
	60	1.000000	1.000694	0.779878	0.882660	1.079430	0.818483	0.797902
	59	1.000000	0.817744	0.782935	0.878607	0.903875	0.818197	0.821526
	58	1.000000	0.632856	0.784801	0.874557	0.725412	0.817688	0.839305
	57	1.000000	0.449547	0.786538	0.870941	0.547752	0.817393	0.853096
	56	1.000000	0.270746	0.791611	0.869372	0.374136	0.818849	0.864038
10	55	1.000000	0.097281	0.796877	0.868258	0.205314	0.820586	0.872872
	54	1.000000	-0.069226	0.801855	0.867327	0.042946	0.822355	0.880098
	53	1.000000	-0.227401	0.806809	0.866677	-0.111522	0.824252	0.886069
	52	1.000000	-0.376246	0.811737	0.866280	-0.257046	0.826258	0.891042
	51	1.000000	-0.515025	0.816368	0.865985	-0.392869	0.828228	0.895208
	50	1.000000	-0.644732	0.824601	0.867626	-0.519682	0.831926	0.898718
	49	1.000000	-0.764366	0.832425	0.869268	-0.636716	0.835507	0.901686
	48	1.000000	-0.874035	0.839842	0.870898	-0.744050	0.838959	0.904207
	47	1.000000	-0.974068	0.846858	0.872502	-0.841987	0.842274	0.906356
	46	1.000000	-1.064828	0.853259	0.873966	-0.930881	0.845346	0.908195
20	45	1.000000	-1.147035	0.859297	0.875394	-1.011412	0.848280	0.909775
	44	1.000000	-1.222322	0.866466	0.877475	-1.085083	0.851752	0.911137
	43	1.000000	-1.290434	0.873200	0.879459	-1.151734	0.855041	0.912316
	42	1.000000	-1.352027	0.879529	0.881350	-1.212004	0.858156	0.913340
	41	1.000000	-1.407733	0.885480	0.883149	-1.266510	0.861108	0.914234
	40	1.000000	-1.458145	0.891082	0.884863	-1.315831	0.863905	0.915017
	39	1.000000	-1.503813	0.896363	0.886496	-1.360503	0.866557	0.915704
	38	1.000000	-1.545235	0.901347	0.888051	-1.401015	0.869075	0.916311
	37	1.000000	-1.582862	0.906059	0.889535	-1.437807	0.871469	0.916848
	36	1.000000	-1.617097	0.910521	0.890952	-1.471275	0.873746	0.917325
30	35	1.000000	-1.648300	0.914753	0.892306	-1.501770	0.875917	0.917750
	34	1.000000	-1.676789	0.918775	0.893603	-1.529606	0.877988	0.918130
	33	1.000000	-1.702849	0.922603	0.894845	-1.555059	0.879968	0.918471
	32	1.000000	-1.726730	0.926255	0.896037	-1.578377	0.881864	0.918777
	31	1.000000	-1.748652	0.929743	0.897183	-1.599774	0.883682	0.919053
	30	1.000000	-1.768811	0.933082	0.898286	-1.619443	0.885427	0.919302
	29	1.000000	-1.787381	0.936283	0.899349	-1.637554	0.887107	0.919527
	28	1.000000	-1.804515	0.939358	0.900376	-1.654257	0.888725	0.919731
	27	1.000000	-1.820348	0.942317	0.901368	-1.669685	0.890286	0.919917
	26	1.000000	-1.835000	0.945168	0.902329	-1.683955	0.891795	0.920086
40	25	1.000000	-1.848578	0.947921	0.903260	-1.697173	0.893256	0.920240
	24	1.000000	-1.861176	0.950584	0.904165	-1.709430	0.894673	0.920380
	23	1.000000	-1.872881	0.953162	0.905044	-1.720811	0.896048	0.920508
	22	1.000000	-1.883765	0.955664	0.905901	-1.731388	0.897386	0.920625
	21	1.000000	-1.893898	0.958095	0.906737	-1.741228	0.898688	0.920732
	20	1.000000	-1.903338	0.960460	0.907552	-1.750388	0.899959	0.920830
	19	1.000000	-1.912139	0.962765	0.908350	-1.758923	0.901199	0.920919
	18	1.000000	-1.920349	0.965015	0.909132	-1.766878	0.902413	0.921001
	17	1.000000	-1.928011	0.967214	0.909898	-1.774296	0.903602	0.921075
	16	1.000000	-1.935165	0.969367	0.910650	-1.781215	0.904767	0.921143
50	15	1.000000	-1.941843	0.971477	0.911389	-1.787668	0.905912	0.921205
	14	1.000000	-1.948078	0.973548	0.912117	-1.793686	0.907038	0.921261
	13	1.000000	-1.953897	0.975583	0.912835	-1.799295	0.908146	0.921312
	12	1.000000	-1.959325	0.977586	0.913543	-1.804521	0.909239	0.921358
	11	1.000000	-1.964384	0.979560	0.914243	-1.809385	0.910318	0.921399
	10	1.000000	-1.969094	0.981507	0.914935	-1.813906	0.911384	0.921436
	9	1.000000	-1.973473	0.983431	0.915621	-1.818102	0.912439	0.921469
	8	1.000000	-1.977537	0.985333	0.916301	-1.821989	0.913484	0.921497
	7	1.000000	-1.981299	0.987217	0.916975	-1.825579	0.914521	0.921522
	6	1.000000	-1.984772	0.989084	0.917646	-1.828885	0.915550	0.921543
60	5	1.000000	-1.987967	0.990937	0.918313	-1.831916	0.916573	0.921561
	4	1.000000	-1.990892	0.992777	0.918977	-1.834683	0.917590	0.921575
	3	1.000000	-1.993554	0.994605	0.919639	-1.837191	0.918603	0.921586

ch a_0 a_1 a_2 b_0 b_1 b_2 IC

	64	01400000	0165c6fb	005d159c	00704e37	0167f2a9	00687008	00409a02
	63	01400000	0160cac8	0064a697	00743800	0163efa1	006a24e5	0052187a
	62	01400000	0156ecc3	0064f9b9	00734a90	015ad1f1	0069e4de	005bd273
	61	01400000	014baa0c	006425c8	0071f299	01502c5e	00692e8c	0061f27f
	60	01400000	01400b5e	0063d30a	0070fb00	01451561	0068c40d	006621a7
	59	01400000	0068abd5	00643736	00707631	0073b22d	0068baad	006927c3
	58	01400000	0051016c	0064745b	006ff17b	005cda4c	0068aa00	006b6e58
	57	01400000	ff731583	0064ad46	006f7afe	00461cbc	0068a055	006d323f
	56	01400000	ff454f9c	00655382	006f4794	ff5fc760	0068d00b	006e98cc
10	55	01400000	fd639da1	00660010	006f2314	fe691eea	006908f6	006fba45
	54	01400000	fdb91cd1	0066a32f	006f0492	fc57f412	006942ed	0070a70d
	53	01400000	fe8b9218	00674584	006eef45	fd8dcd2d	00698116	00716ab5
	52	01400000	ff9fae57	0067e6ff	006ee243	ffbe323b	0069c2d2	00720daa
	51	01400000	00be13a9	00687ebf	006ed898	ff9b6cef	006a0360	0072962c
	50	01400000	00ad796b	00698c86	006f0e5e	00bd7b0f	006a7c8d	00730931
	49	01400000	009e2941	006a8ce7	006f442c	00ae8017	006af1e4	00736a72
	48	01400000	00901f9f	006b7ff1	006f7995	00a0c2f8	006b6302	0073bd0e
	47	01400000	008351bd	006c65d7	006fae25	009439c5	006bcfa2	00740379
	46	01400000	01bbd9db	006d3797	006fde1e	0088d8e4	006c344c	00743fbb
20	45	01400000	01b696fa	006dfd71	00700ce9	01bf4506	006c9470	00747381
	44	01400000	01b1c579	006ee85b	00705119	01ba8e00	006d0635	0074a023
	43	01400000	01ad6987	006fc504	0070921c	01b649fd	006d71fb	0074c6c5
	42	01400000	01a97863	00709468	0070d013	01b26e86	006dd80e	0074e853
	41	01400000	01a5e7b3	00715768	00710b06	01aef180	006e38c9	0075059e
	40	01400000	01a2adc0	00720ef9	00714330	01abc96c	006e9470	00751f46
	39	01400000	019fc187	0072bc05	007178b3	01a8ed84	006eeb56	007535c9
	38	01400000	019d1ade	00735f56	0071aba7	01a655c5	006f3dd9	007549ad
	37	01400000	019ab263	0073f9bd	0071dc48	01a3faf8	006f8c4b	00755b46
	36	01400000	0198817b	00748bf3	00720ab7	01a1d6a1	006fd6e8	00756ae7
30	35	01400000	01968240	007516a0	00723715	019fe300	00701e0c	007578d4
	34	01400000	0194af7d	00759a6b	00726195	019e1aef	007061e9	00758548
	33	01400000	01930485	007617da	00728a47	019c79e9	0070a2ca	00759075
	32	01400000	01917d41	00768f86	0072b157	019afbdf	0070e0eb	00759a7c
	31	01400000	01901615	007701d1	0072d6e4	01999d4d	00711c7d	0075a387
	30	01400000	018ecbcc	00776f3b	0072fb09	01985b0b	007155ac	0075abb0
	29	01400000	018d9b8c	0077d81f	00731dde	01973250	00718cb8	0075b30f
	28	01400000	018c82d3	00783ce2	00733f85	019620a7	0071c1bd	0075b9be
	27	01400000	018b7f6b	00789dd7	00736006	019523e1	0071f4e4	0075bfd7
	26	01400000	018a8f5c	0078fb43	00737f84	01943a14	00722656	0075c560
40	25	01400000	0189b0e5	00795579	00739e06	01936184	00725636	0075ca6c
	24	01400000	0188e27e	0079acbc	0073bbad	019298b2	007284a5	0075cf03
	23	01400000	018822b7	007a0136	0073d87b	0191de3b	0072b1b3	0075d334
	22	01400000	01877064	007a5332	0073f490	019130f0	0072dd8b	0075d70a
	21	01400000	0186ca60	007aa2db	00740ff5	01908fb8	00730835	0075da8b
	20	01400000	01862fb5	007af05a	00742aa9	018ff9a4	007331db	0075ddc1
	19	01400000	01859f83	007b3be2	007444d0	018f6dce	00735a7d	0075e0ac
	18	01400000	01851900	007b859c	00745e6f	018eeb78	00738244	0075e35c
	17	01400000	01849b77	007bcdab	00747789	018e71ef	0073a93a	0075e5c9
	16	01400000	01842641	007c1437	0074902d	018e0092	0073cf67	0075e803
50	15	01400000	0183b8d8	007c595b	0074a865	018d96d8	0073f4ec	0075ea0b
	14	01400000	018352b0	007c9d38	0074c03f	018d343f	007419d2	0075ebe1
	13	01400000	0182f359	007cdfe7	0074d7c6	018cd859	00743e20	0075ed8d
	12	01400000	01829a6b	007d2189	0074eefa	018c82ba	007461f1	0075ef0f
	11	01400000	01824788	007d6238	007505ea	018c3309	0074854c	0075f067
	10	01400000	0181fa5d	007da205	00751c97	018be8f6	0074a83b	0075f19d
	9	01400000	0181b29e	007de111	00753311	018ba437	0074cacd	0075f2b2
	8	01400000	01817008	007e1f64	00754959	018b6488	0074ed0b	0075f39d
	7	01400000	01813265	007e5d20	00755f6f	018b29b6	00750f06	0075f46e
	6	01400000	0180f97e	007e9a4d	0075756c	018af38c	007530be	0075f51e
60	5	01400000	0180c526	007ed706	00758b47	018ac1e3	00755243	0075f5b5
	4	01400000	01809539	007f1351	0075a109	018a948d	00757396	0075f62b
	3	01400000	0180699c	007f4f37	0075b6bb	018a6b76	007594c8	0075f687

	63	1.000000	1.113452	0.519651	-0.240174	0.000000	0.240174	0.240174
	62	1.000000	0.990868	0.561916	-0.219042	0.000000	0.219042	0.219042
	61	1.000000	0.848065	0.592224	-0.203888	0.000000	0.203888	0.203888
	60	1.000000	0.694753	0.616770	-0.191615	0.000000	0.191615	0.191615
	59	1.000000	0.536192	0.638046	-0.180977	0.000000	0.180977	0.180977
	58	1.000000	0.375810	0.657232	-0.171384	0.000000	0.171384	0.171384
	57	1.000000	0.216117	0.674955	-0.162522	0.000000	0.162522	0.162522
	56	1.000000	0.059104	0.691567	-0.154216	0.000000	0.154216	0.154216
	55	1.000000	-0.093585	0.707273	-0.146364	0.000000	0.146364	0.146364
10	54	1.000000	-0.240574	0.722191	-0.138905	0.000000	0.138905	0.138905
	53	1.000000	-0.380736	0.736388	-0.131806	0.000000	0.131806	0.131806
	52	1.000000	-0.513199	0.749900	-0.125050	0.000000	0.125050	0.125050
	51	1.000000	-0.637349	0.762747	-0.118627	0.000000	0.118627	0.118627
	50	1.000000	-0.752834	0.774939	-0.112531	0.000000	0.112531	0.112531
	49	1.000000	-0.859548	0.786487	-0.106757	0.000000	0.106757	0.106757
	48	1.000000	-0.957605	0.797403	-0.101299	0.000000	0.101299	0.101299
	47	1.000000	-1.047293	0.807704	-0.096148	0.000000	0.096148	0.096148
	46	1.000000	-1.129038	0.817412	-0.091294	0.000000	0.091294	0.091294
	45	1.000000	-1.203352	0.826553	-0.086724	0.000000	0.086724	0.086724
20	44	1.000000	-1.270799	0.835156	-0.082422	0.000000	0.082422	0.082422
	43	1.000000	-1.331958	0.843252	-0.078374	0.000000	0.078374	0.078374
	42	1.000000	-1.387404	0.850875	-0.074563	0.000000	0.074563	0.074563
	41	1.000000	-1.437688	0.858056	-0.070972	0.000000	0.070972	0.070972
	40	1.000000	-1.483325	0.864829	-0.067586	0.000000	0.067586	0.067586
	39	1.000000	-1.524792	0.871223	-0.064388	0.000000	0.064388	0.064388
	38	1.000000	-1.562522	0.877269	-0.061365	0.000000	0.061365	0.061365
	37	1.000000	-1.596907	0.882994	-0.058503	0.000000	0.058503	0.058503
	36	1.000000	-1.628298	0.888424	-0.055788	0.000000	0.055788	0.055788
	35	1.000000	-1.657008	0.893582	-0.053209	0.000000	0.053209	0.053209
30	34	1.000000	-1.683315	0.898491	-0.050754	0.000000	0.050754	0.050754
	33	1.000000	-1.707466	0.903172	-0.048414	0.000000	0.048414	0.048414
	32	1.000000	-1.729680	0.907642	-0.046179	0.000000	0.046179	0.046179
	31	1.000000	-1.750151	0.911919	-0.044041	0.000000	0.044041	0.044041
	30	1.000000	-1.769051	0.916018	-0.041991	0.000000	0.041991	0.041991
	29	1.000000	-1.786530	0.919954	-0.040023	0.000000	0.040023	0.040023
	28	1.000000	-1.802725	0.923739	-0.038130	0.000000	0.038130	0.038130
	27	1.000000	-1.817754	0.927386	-0.036307	0.000000	0.036307	0.036307
	26	1.000000	-1.831724	0.930906	-0.034547	0.000000	0.034547	0.034547
	25	1.000000	-1.844729	0.934309	-0.032845	0.000000	0.032845	0.032845
40	24	1.000000	-1.856853	0.937604	-0.031198	0.000000	0.031198	0.031198
	23	1.000000	-1.868171	0.940800	-0.029600	0.000000	0.029600	0.029600
	22	1.000000	-1.878749	0.943905	-0.028048	0.000000	0.028048	0.028048
	21	1.000000	-1.888649	0.946925	-0.026538	0.000000	0.026538	0.026538
	20	1.000000	-1.897922	0.949868	-0.025066	0.000000	0.025066	0.025066
	19	1.000000	-1.906618	0.952740	-0.023630	0.000000	0.023630	0.023630
	18	1.000000	-1.914779	0.955547	-0.022226	0.000000	0.022226	0.022226
	17	1.000000	-1.922443	0.958295	-0.020853	0.000000	0.020853	0.020853
	16	1.000000	-1.929647	0.960988	-0.019506	0.000000	0.019506	0.019506
	15	1.000000	-1.936419	0.963631	-0.018185	0.000000	0.018185	0.018185
50	14	1.000000	-1.942790	0.966229	-0.016885	0.000000	0.016885	0.016885
	13	1.000000	-1.948783	0.968787	-0.015606	0.000000	0.015606	0.015606
	12	1.000000	-1.954422	0.971308	-0.014346	0.000000	0.014346	0.014346
	11	1.000000	-1.959726	0.973797	-0.013102	0.000000	0.013102	0.013102
	10	1.000000	-1.964715	0.976256	-0.011872	0.000000	0.011872	0.011872
	9	1.000000	-1.969403	0.978690	-0.010655	0.000000	0.010655	0.010655
	8	1.000000	-1.973806	0.981102	-0.009449	0.000000	0.009449	0.009449
	7	1.000000	-1.977937	0.983496	-0.008252	0.000000	0.008252	0.008252
	6	1.000000	-1.981806	0.985874	-0.007063	0.000000	0.007063	0.007063
	5	1.000000	-1.985424	0.988240	-0.005880	0.000000	0.005880	0.005880
60	4	1.000000	-1.988800	0.990598	-0.004701	0.000000	0.004701	0.004701
	3	1.000000	-1.991939	0.992949	-0.003525	0.000000	0.003525	0.003525

ch q0 q1 q2 b0 b1 b2 k

	63	01400000	014742cc	004283ec	fe8507e9	80000000	fe7af816	fe7af816
	62	01400000	007ed4c3	0047ecdd	fe8fd9ba	80000000	fe702645	fe702645
	61	01400000	006c8d64	004bcdfe	fe979bfd	80000000	fe686402	fe686402
	60	01400000	0058edaa	004ef251	fe9de4a3	80000000	fe621b5c	fe621b5c
	59	01400000	0044a1f0	0051ab7d	fea356fb	80000000	fe5ca904	fe5ca904
	58	01400000	ff603515	0054202d	fea8405b	80000000	fe57bfa4	fe57bfa4
	57	01400000	fe6ea6e3	005664ec	feacc9ea	80000000	fe533615	fe533615
	56	01400000	fc790b84	00588544	feb10a99	80000000	fe4ef566	fe4ef566
	55	01400000	fda02b40	005a87eb	feb50fc7	80000000	fe4af038	fe4af038
10	54	01400000	fe84d37c	005c70c1	feb8e171	80000000	fe471e8e	fe471e8e
	53	01400000	ff9e8815	005e41f6	febc83ec	80000000	fe437c13	fe437c13
	52	01400000	00be4f7e	005ffcb9	febff972	80000000	fe40068d	fe40068d
	51	01400000	00ae6b59	0061a1b1	fd8686a4	80000000	fd79795b	fd79795b
	50	01400000	009fa322	00633133	fd8cc4ac	80000000	fd733b53	fd733b53
	49	01400000	0091fa54	0064ab9b	fd92ae4b	80000000	fd6d51b4	fd6d51b4
	48	01400000	00856d33	0066114d	fd984513	80000000	fd67baec	fd67baec
	47	01400000	01bcf926	006762d8	fd9d8b60	80000000	fd62749f	fd62749f
	46	01400000	01b7bdd7	0068a0f4	fda283d3	80000000	fd5d7c2c	fd5d7c2c
	45	01400000	01b2fc47	0069cc7d	fda731d2	80000000	fd58ce2d	fd58ce2d
20	44	01400000	01aeab3a	006ae664	fdab9991	80000000	fd54666e	fd54666e
	43	01400000	01aac133	006befae	fdafbeb9	80000000	fd504146	fd504146
	42	01400000	01a734c5	006ce978	fdb3a5c1	80000000	fd4c5a3e	fd4c5a3e
	41	01400000	01a3fceb	006dd4c7	fdb7531d	80000000	fd48ace2	fd48ace2
	40	01400000	01a11134	006eb2b7	fdbacabc	80000000	fd453543	fd453543
	39	01400000	019e69ce	006f843c	fdbe1112	80000000	fd41eeed	fd41eeed
	38	01400000	019bffa3	00704a59	fc825311	80000000	fc7dacee	fc7dacee
	37	01400000	0199cc46	007105f2	fc882f94	80000000	fc77d06b	fc77d06b
	36	01400000	0197c9f7	0071b7e0	fc8dbf05	80000000	fc7240fa	fc7240fa
	35	01400000	0195f394	007260e5	fc930728	80000000	fc6cf8d7	fc6cf8d7
30	34	01400000	01944491	007301c0	fc980e49	80000000	fc67f1b6	fc67f1b6
	33	01400000	0192b8e0	00739b23	fc9cd91e	80000000	fc6326e1	fc6326e1
	32	01400000	01914cec	00742d9c	fca16ce7	80000000	fc5e9318	fc5e9318
	31	01400000	018ffd86	0074b9c3	fca5cdd5	80000000	fc5a322a	fc5a322a
	30	01400000	018ec7de	00754013	fcaa009f	80000000	fc55ff60	fc55ff60
	29	01400000	018da97e	0075c10d	fcae086b	80000000	fc51f794	fc51f794
	28	01400000	018ca027	00763d14	fcb1e8e6	80000000	fc4e1719	fc4e1719
	27	01400000	018ba9eb	0076b495	fcb5a4ac	80000000	fc4a5b53	fc4a5b53
	26	01400000	018ac508	007727ed	fcb93f6c	80000000	fc46c093	fc46c093
	25	01400000	0189eff5	0077976f	fcbcbbc2	80000000	fc43443d	fc43443d
40	24	01400000	01892952	00780368	fb803686	80000000	fb7fc979	fb7fc979
	23	01400000	01886fe2	00786c22	fb86c226	80000000	fb793dd9	fb793dd9
	22	01400000	0187c293	0078d1e1	fb8d1d8a	80000000	fb72e275	fb72e275
	21	01400000	0187205f	007934d6	fb934ce3	80000000	fb6cb31c	fb6cb31c
	20	01400000	01868872	00799546	fb995464	80000000	fb66ab9b	fb66ab9b
	19	01400000	0185f9f8	0079f362	fb9f3626	80000000	fb60c9d9	fb60c9d9
	18	01400000	01857442	007a4f5d	fba4f659	80000000	fb5b09a6	fb5b09a6
	17	01400000	0184f6b1	007aa969	fbaa960b	80000000	fb5569f4	fb5569f4
	16	01400000	018480a9	007b01a7	fbb01a79	80000000	fb4fe586	fb4fe586
	15	01400000	018411b6	007b5842	fbb583a5	80000000	fb4a7c5a	fb4a7c5a
50	14	01400000	0183a954	007bad64	fbbad6cb	80000000	fb452934	fb452934
	13	01400000	01834723	007c0136	fa8027d8	80000000	fa7fd827	fa7fd827
	12	01400000	0182eabf	007c53d2	fa8a7a41	80000000	fa7585be	fa7585be
	11	01400000	018293d9	007ca561	fa94ab1d	80000000	fa6b54e2	fa6b54e2
	10	01400000	0182421c	007cf5f4	fa9ebe9c	80000000	fa614163	fa614163
	9	01400000	0181f54d	007d45b6	faa8b6d8	80000000	fa574927	fa574927
	8	01400000	0181ad29	007d94c0	fab29802	80000000	fa4d67fd	fa4d67fd
	7	01400000	0181697a	007de332	fabc664d	80000000	fa4399b2	fa4399b2
	6	01400000	01812a17	007e311e	f98c47a1	80000000	f973b85e	f973b85e
	5	01400000	0180eed0	007e7ea5	f99fa97e	80000000	f9605681	f9605681
60	4	01400000	0180b780	007ecbea	f9b2fa93	80000000	f94d056c	f94d056c
	3	01400000	01808412	007f18f3	f88c7e28	80000000	f87381d7	f87381d7

FE70D98A

-0.217088