マルチメディア通信と分散処理 28－9
（１９８６．２．２７）

パーソナルコンピュータでの画像処理

溝口正典　天満勉　木村嘉則　石黒尚夫*

日本電気株式会社　C&Cシステム研究所　*日本電気技術情報システム開発株式会社

（1）まえがき

最近、OA分野等で、簡易なスキャナーやカメラを接続したパーソナルコンピュータ利用の画像処理や、画像処理を視覚的なマンマシンインターフェイスとして応用することが期待されてきている。また、グラフィックスの分野でもラスターグラフィック等で画像処理が身近なものとなってきた。

画像処理の特徴として同一の処理が大量のデータに対して施されることがあげられる。このような場合の高速処理手法の一つとして、パイプライン処理手法があり、プロセッサ並列に比べてコストパフォーマンスの良いシステムとなることが知られている。しかしながら、パイプライン処理は、決まった処理に対しては適しているが、処理内容の変化に対してのフレキシビリティに欠ける面がある。　筆者らはこの問題に対してパイプラインをソフトウェア的に制御し、メモリユニットをも含めた複数のプロセッサ間に任意のデータパスを作れるようにして、解決しようとした[1]。即ち複数のプロセッサをリング状に接続しデータ駆動制御を取り入れて同期の制限を取り除き、効率よく並列動作するようにした。画像処理プロセッサＩｍＰＰ（μＰＤ７２８１）はＬＳＩ化技術により、いくつかの演算処理機能を１チップ内に内臓させ、より柔軟性の高い汎用プロセッサモジュールとし、画像処理を手軽に且つ高速に出来るようにしたものであり、マルチプロセッサも容易に構成できる。

筆者らはこれまでＩｍＰＰを８個、周辺サポート用ＬＳＩのＭＡＧＩＣ（μＰＤ９３０５）、リアルタイム処理実行制御用１６ビットマイクロプロセッサ、１ＭＷメモリボード、ユーザーコマンド解析およびディスクファイルサポート用のホストとしてパーソナルコンピュータＰＣ９８００を用いたＴＩＰ－３システムを開発してきた[2]。

今回、よりパーソナルなシステムでの画像処理を可能とするため、ＰＣ９８００Ｅ（８Ｍｚ）の拡張スロットに挿入して使用でき、ＩｍＰＰを２個，ＭＡＧＩＣ、２５６ＫＷのイメージメモリを搭載したＴＩＰ－ＰＣボードを開発した。また同時にソフトウェア開発と実行のサポートとして、ユーザープログラムから容易にコールできるＣ言語によるコントロール用サブルーチン関数群を開発した。以下では，ＴＩＰ－ＰＣボードのハードウェア構成とシステムサブルーチン関数について述べ、簡単な画像処理を行った結果について報告する。

（2）ＴＩＰ－ＰＣのハードウェア構成

（2－1）ＴＩＰ－ＰＣの全体構成

ＴＩＰ－ＰＣシステムの全体構成のブロック図を図１に、また外観写真を図２に示す。ホストコンピュータにはＰＣ－９８００Ｅを使用する。

画像処理部（ＴＩＰ－ＰＣボード）はＰＣ－９８００用拡張ボード１枚（およそ１５０㎜Ｘ２４０㎜で、通常の拡張ボードより長め）に、ＩｍＰＰ（μＰＤ７２８１）２個，ＭＡＧＩＣ（μＰＤ９３０５）１個、２５６ＫＷイメージメモリ（μＰＤ４１２５６Ｖ）等を搭載している（写真１）。

イメージメモリはＩｍＰＰ用にデータ１８ビット／語構成であり、最高５００ｎｓサイクルでアクセスできる。ＤＭＡ転送はＰＣ－９８００ＥのＤＭＡチャンネル（３ＣＨ）を用いるが、ＰＣ－９８００Ｅのバス転送単位は１バイトであり、イメージメモリの下位１６ビットを転送できる。ＭＡＧＩＣからはＣＰＵＲＱ（ＣＰＵリクエスト信号）により割り込みを発生するが、割り込みレベルにＩＮＴ５を使用した。

Ｉ／Ｏポートのアドレス割り当てを表１に示す。なお、ＰＣ－９８００のグラフィックメモリ（ＶＲＡＭ）には直接ＤＭＡ転送はせず、メインメモリ内に設けたバッファ領域を介してＶＲＡＭに画像データを転送表示するようにしている。　この他、現在実験中の試作機には、イメージスキャナーやマウスがついている。

なお、ＴＩＰ－ＰＣの基本デバイスである画像処理プロセッサＬＳＩ、ＩｍＰＰ（μＰＤ７２８１）、およびその周辺サポート用に開発されたＭＡＧＩＣ（μＰＤ９３０５）の詳細については既に発表されている[3][4]。

（3）ＴＩＰ－ＰＣシステムソフトウェア

ＰＣ－９８００のＤＯＳにはＭＳ－ＤＯＳを使用することにした。ユーザープログラムおよびシステムソフトウェア（ユーザープログラムからコールするサブルーチン関数とデバッガー）の記述言語としてはＣを用いた。ＰＣ－９８００のグラフィック機能は積極的に使っている。表２に主なシステムサブルーチン関数一覧、表３にデバッガーの主なコマンド一覧を示す。以下に基本的な関数を幾つか紹介する。

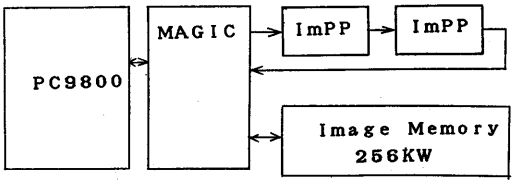

図1　ＴＩＰ－ＰＣブロック構成図

イメージメモリに画像データの格納領域を確保するアロケーション関数(alloc,ralloc)、ディスクファイルとＶＲＡＭとイメージメモリ間のロード／セーブ関数(fload,fsave,iload,isave)、ＩｍＰＰテンプレートプログラム（オブジェクトプログラム）をイメージメモリに転送する(pload)、ＩｍＰＰにセルフオブジェクトロードする関数(load)、起動トークンのデータ値にメインプログラムの変数値を代入して出力し処理を開始させる関数(start,action)、処理終了結果トークンを入力しメインプログラムの変数に値を返す関数(start,wait)等がある。このほか、ハードウェアの環境設定関数(reset,mnset,initial)、低レベルのI/O関数(imrd,imwt,tkin,tkout,status),１０ｍｓ単位のタイマーサポート関数等がある。

また、デバッガーはイメージメモリのダンプ、ＩｍＰＰからのテンプレートプログラムテーブルのダンプ、トークン入出力、ステータスチェック、ＴＩＰ－ＰＣのイニシャライズおよびリセット、等のコマンドが備えられている。

（４）画像処理実験結果

ＴＩＰ－ＰＣの使い勝手の確認と速度性能測定のために簡単なアプリケーションプログラムを実行した結果について述べる。

アフィン変換

イメージメモリ上に２５６Ｘ２５６ドットのソースイメージを置き、アフィン変換により回転処理を行った結果イメージをＰＣ－９８００ＥのＶＲＡＭへ転送表示するプログラムである。Ｃのメインプログラムを図３に示す。アフィン変換処理には約２１０ｍｓ、ディスプレイの為のＤＭＡ転送も含めて、約２５０ｍｓ毎に処理結果イメージを表示することができた。

３次元ワイヤーグラフィックス表示

マウスから入力した視点座標を使い、視点透視座標系への変換及びクリッピングを行った後、端点座標をＰＣ－９８００Ｅに送り描画する。元の物体データはイメージメモリ内に保持して置き、視点座標が与えられる毎に描画用データだけを転送し表示するループにした。ＩｍＰＰプログラムは計４種類作成し、逆数や三角関数等の演算はイメージメモリ上の変換テーブルで行った。処理結果の画面を図４に示す、なお、この例は１４７本の線分データを与えており、一回のループ時間は約０．５秒であった。

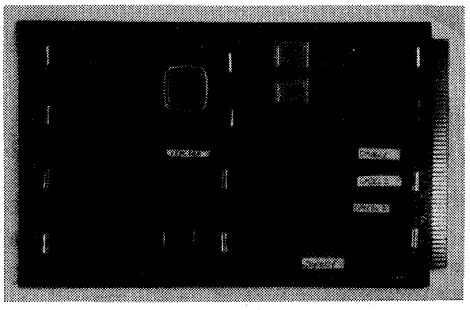

図2　ＴＩＰ－ＰＣの外観（写真１）

	Read	Write
D0	入力データレジスタ	出力データレジスタ
D2	ステータスレジスタ	モードレジスタ
D4	－	モジュールナンバーセット
D6	－	リフレシュタイマーセット
D8	－	リセット

表1　入出力ポートアドレス割り当て

一般にＩｍＰＰプログラムの処理時間は次ぎの様な式で表すことができる。

Ｔａ＝Ｔｌ＋Ｔｓ＋Ｔｅ＋Ｔｒ　　（１）

Ｔａ：全処理時間

Ｔｌ：ＩｍＰＰへのセルフオブジェクトロード時間

Ｔｓ：起動トークン出力処理時間

Ｔｅ：ＩｍＰＰ処理実行時間

Ｔｒ：処理終了結果トークンの入力処理時間

更に、Ｔｌ、Ｔｓ、Ｔｒはシステムのオーバーヘッド部と１トークン当たりの処理時間に分類でき、

Ｔｌ＝Ｔｌｏ＋Ｔｌｔ＊Ｎ　　（２）

Ｔｓ＝Ｔｓｏ＋Ｔｓｔ＊Ｍ　　（３）

Ｔｒ＝Ｔｒｏ＋Ｔｒｔ＊Ｌ　　（４）

Ｔｌｏ、Ｔｓｏ、Ｔｒｏ：各システム関数オーバーヘッド分

Ｔｌｔ：１トークン当たりのセルフオブジェクトロード時間

Ｔｓｔ：１起動トークン当たりの出力処理時間

```
#include        <stdio.h>
#include        <tbl.h>
#include        <math.h>
#include        <limits.h>
int     cost[24],sint[24],m[24];        /* SIN COS TABLE */
/* ============== SIN COS DATA TABLE SET ================= */
int     sncs_set()
{
int     i;
double  pai=3.1415926535;
double  p;
        p = 2 * pai /24;
for(i=0;i<24;i++) {
        cost[i]= (int)(cos(p * (double)i) * 8192);
        sint[i]= (int)(sin(p * (double)i) * 8192);
        }
}
/* ============== ImPP PROGRAM LISTING =================== */
struct module   imppio[7]={     "dma1.ipp",5,0,
                                "dma2.ipp",3,0,
                                "dma1r.ipp",5,0,
                                "dma2r.ipp",3,0,
                                "afn48.ipp",9,1,
                                "afn49.ipp",9,1,
                                "*"};
/* ================ MAIN PROGRAM ========================== */
main(){
int     imno1,imno2;
int     stat,end48[2],end49[2];
int     i,j;
int     a,b,loopend;
unsigned rhad,rlad,wlad;
int     sx,sy,sa,dspwid,imgwid,centr;
long    rladdr,wladdr;

 sx=256;
 sy=256;
 sa= (sx / 8) * sy;
 dspwid = 640 / 8;
 imgwid = sx / 8;
 centr = (sx / 16)*(sy / 2) + (sx / 16) / 2;
 loopend = 36;
 initial();
 screen(3,0,0,1);
 sncs_set();
 /*****          image memory allocation          *****/
 imno1= ialloc("b",sx,sy);
 /*****          source image load                *****/
 stat=fload("inkan.img",'i',imno1);
 /*****          program load                     *****/
 stat=pload("afn48.ipp");
 stat=pload("afn49.ipp");
 stat=imad(imno1,&rhad,&rlad);
 rladdr=((long)rhad << 16) + ((long)rlad & 0x0000ffff1);
 wlad   = 0xa000;               /* IM WRITE AREA POINTER */
 wladdr = 0x0000a0001;          /* POINTER FOR DMA2 CALL */
 /*****          source image display             *****/
 stat=dma2(bank,offset,rladdr,sa);
 for(i=0;i<sy;i++)
      poke(0xa000,0x8000+dspwid*i,buff+imgwid*i,imgwid);
 printf("== START ==\n");
 a=0;
 while(a++ < loopend) {
   b=a&24;
   /*****        SELF OBJECT LOAD                 *****/
   load("afn48.ipp");
   load("afn49.ipp");
   /*****        START MN 8                       *****/
   action("afn48.ipp",
        cost[b],sint[b],0,-sint[b],cost[b],0,rlad+centr,wlad,-sa/2);
   /*****        START MN 9                       *****/
   action("afn49.ipp",
        cost[b],sint[b],0,-sint[b],cost[b],0,rlad+centr,wlad+sa/4,0);
   /*****        VRAM WRITE                       *****/
   for(i=0;i<sy;i++)
      poke(0xa000,0x8000+imgwid+dspwid*i,buff+imgwid*i,imgwid);
   /*****        WAIT CPU REQ. MN 8               *****/
   wait("afn48.ipp",end48);
   /*****        WAIT CPU REQ. MN 9               *****/
   wait("afn49.ipp",end49);
   /*****        DMA     IM ---> BUFF             *****/
   stat=dma2(bank,offset,wladdr,sa);
   }
 printf("== AFFINE_END ==\n");
 getchar();
 cls(2);                /* CLEAR VRAM */
}
```

図3　アフィン変換メインプログラム

ハードウェア	
reset	リセット
creset	MAGICのリセット
mnset	モジュールナンバーセット
initial	システム初期化
メモリ管理	
ialloc	イーメジメモリの領域確保
ralloc	領域解放
imad	領域の先頭アドレス獲得
データ転送	
dma1	PC9800からIMへのDMA転送
dma2	IMからPC9800へのDMA転送
dmacset	DMAコントローラの設定
pload	ImPPプログラムを
fload	データファイルをIMへ転送
fsave	IMからデータファイルに転送
iload	VRAMからIMへ転送
isave	IMからVRAMへ転送
プログラム実行	
load	セルフオブジェクトロード
start	起動トークン出力と終了トークン入力
exec	load&start
action	起動トークン出力
wait	終了トークン入力
低レベルi/o	
imrd	イメージメモリ読みだし
imwt	イメージメモリ書き込み
tkin	トークン入力
tkout	トークン出力
status	MAGICステータスリード
timset	タイマー初期設定
timread	タイマーカウントリード

表2　システムサブルーチン関数一覧

dump	イメージメモリのダンプ
rlt	ImPPのLTのダンプ
rft	ImPPのFTのダンプ
rdm	ImPPのDMのダンプ
status	MAGICステータスリード
reset	システムリセット
mnset	モジュールナンバーセット
tin	トークン入力
tout	トークン出力
clr	ImPPプログラムのクリア

表3　デバッガーコマンド一覧

Ｔｒｔ：１処理終了結果トークン当たりの
　　入力時間
Ｎ　：オブジェクトトークン数
Ｍ　：起動トークン数
Ｌ　：処理終了結果トークン数

の様に表現できる。従って，個々のＩｍＰＰのオブジェクトプログラムに対してＴｌ，Ｔｓ，Ｔｒを見積る事ができる。今回の試作システムから求めた値は次のようであった。ただし、これらの値は処理実行時間に比べて小さい。

Ｔｌｏ	１．５０	ｍｓ
Ｔｓｏ＋Ｔｒｏ	１．５５	ｍｓ
Ｔｌｔ	０．６９	ｍｓ
Ｔｓｔ	２４３	μｓ
Ｔｒｔ	３００	μｓ

Ｔｅについては実行時の稼働率が推定し難く、処理時間を正確に見積ることは一般には難しい。しかし、動的なＩｍＰＰのＬＴ稼働率（ρ）、メモリアクセス稼働率（τ）と、処理実行総トークン数（ＬＴを通過する総トークン数（Ｎｌ）、もしくは総メモリアクセストークン数（Ｎｍ）およびＩｍＰＰとメモリアクセスのサイクル時間との間には次の（５）式のような関係がある。ただし、処理ボトルネックとなる項は処理プログラムに依存しており、ＰＵで消滅するトークン数をもＮｌに含めて計算する場合もある。

$$Te = max(t*Nl/\alpha,\ s*Nm/\tau) \quad (5)$$

ただし、ｔ：ＩｍＰＰパイプラインサイクル時間
　　ｓ：ＭＡＧＩＣパイプラインサイクル時間

次ぎに、ディスプレイへの表示を行ったりする場合は、ＤＭＡ転送を考慮する必要がある。ＰＣ９８００からＩＭへのＤＭＡ転送の処理時間Ｔｄ１と、ＩＭからＰＣ９８００へのＤＭＡ転送の処理時間Ｔｄ２、及びメインメモリのバッファからＰＣ９８００のＶＲＡＭへの転送時間Ｔｖｗのそれぞれについても、同様にオーバーヘッド分と、転送バイトに比例する分に分けることができる。なお、ＴｖｗはＩｍＰＰの処理実行とオーバーラップさせて同時処理すれば、見かけ上０となる。

$$Td1 = Td1o + Td1t*N \quad (6)$$

$$Td2 = Td2o + Td2t*M \quad (7)$$

$$Tvw = Tvwt*M \quad (8)$$

Ｎ，Ｍ　：　転送バイト数

実測値は次のような値であった。

Ｔｄ１ｏ	２．０５	ｍｓ
Ｔｄ２ｏ	２．０７	ｍｓ
Ｔｄ１ｔ	４．０５	μｓ
Ｔｄ２ｔ	３．５８	μｓ
Ｔｖｗ	５．６５	μｓ

（５）　検討

アフィン変換については実際のＩｍＰＰのアフィン処理実行時間は２１０ｍｓであるが、角度パラメータを変化させてループにしたときには、オーバーヘッドやＤＭＡ転送も含めて２５０ｍｓであった。Ｃで同一のアルゴリズムを記述、実行してみたところ７秒であり、ＴＩＰ－ＰＣにより処理が容易に高速化される事をある程度確認出来た。３次元ワイヤーグラフィックスでは、ＩｍＰＰで行っているマトリックス演算は１回のループ当たり約７０ｍｓ程度に高速化できたが、残念ながらＶＲＡＭへの描画時間がかなりの部分を占めており、描画本数が多いと画面のちらつきが見られ、 図４ に示した例では約０．５秒であった。このように、高々ＩｍＰＰ２個による処理ではあるが、比較的小さな画像で色々なアルゴリズムを試みたり、データを変化させてみたりするパーソナルコンピュータでの画像処理には強力な手助けとなる。

また、プログラム開発の観点からは、シミュレータでは扱えない大きなメモリ空間が使え，ＴＩＰ－３システムやＩｍＰＰを用いたより大きなシステムをターゲットとするＩｍＰＰプログラムの開発ツールや、教育ツールとしても有用であると感じられた。

システムのオーバーヘッドについては前述した様にそれほどのものではない。システム関数をアセンブラ記述すれば幾らか高速化もされるであろうが、ＩｍＰＰの処理実行時間に比べれば効果はあまり期待出来ない。むしろＤＭＡ転送や描画の部分で、改善の余地がかなり残されており、特にＩｍＰＰ処理とグラフィックルーチンとのインターフェイスには改善の余地がある。

ユーザーインターフェイスの点では、まだまだ使い勝手の良いものに改善して行くべきところは多い。例えば、（１）現在はコーディングされた順にイメージメモリに領域を割り当てているが、ＭＡＧＩＣでのアドレス空間が上位側８ビットと下位側１６ビットに分割しているため、異なる上位側アドレスに跨ったメモリ空間をアロケ

ートしないようにする。（2）ＩｍＰＰ内部の標準である１８ビットデータや、Ｃ言語の浮動小数点数とのインターフェイスをとれば、更にプログラムが容易になる。（3）関数テーブル作成もしくは関数データファイルロード等のユーティリティを整備する。等が課題と言える。

パーソナルコンピュータレベルでこのような画像処理が手軽に出来るようになれば、画像をユーザーインターフェースに用いた多様な利用法が出てくるのではないかとみている。さらに、音声等も組み合わせた視聴覚インターフェイスの発展が期待されよう。

（6）あとがき

画像処理プロセッチップＩｍＰＰとそれを用いたパソコンレベルの画像処理システムＴＩＰ－ＰＣシステムの構成、システムサブルーチン関数とそれを利用したプログラム記述、および実際に簡単なアプリケーションプログラム実行した結果と速度性能について述べた。ＴＩＰ－ＰＣボードを備えることにより、従来大変時間の掛かっていた画像処理が、パーソナルコンピュータ上で手軽に実現できるようになった。

今後の課題としては、システム関数やＤＭＡの高速化による改善、ユーザーインターフェイスの向上がある。なお、このシステムの上に汎用のコマンドインタプリタ型のＩｍＰＰプログラム実行環境も現在開発中であり、それについては別の機会に報告したい。

謝辞　最後に、日頃励ましと有益な御助言を頂く、パタン認識研究部浅井部長に感謝するとともに、本稿をまとめるにあたり貴重な御意見、御協力を頂いた、岩下主任、緑川氏に深謝いたします。

参考文献

[1] 天満他「可変パイプライン方式の画像処理プロセッサＴＩＰ－１」
信学会論文誌
Vol.J68-D No.4 pp853-860 (1985)

[2] 森下他「画像プロセッサＴＩＰ－３ハードウェア構成」
情処　コンピュータビジョン研究会
Vol. CV-32-1 (1984)

[3] μＰＤ７２８１　ユーザーズマニュアル
日本電気（株）　IEM-877 (1984)

[4] μＰＤ９３０５　ユーザーズマニュアル
日本電気（株）　IEM-895 (1984)

[5] μＰＤ７２８１　アプリケーションライブラリ
日本電気（株）　IEM-900 (1985)

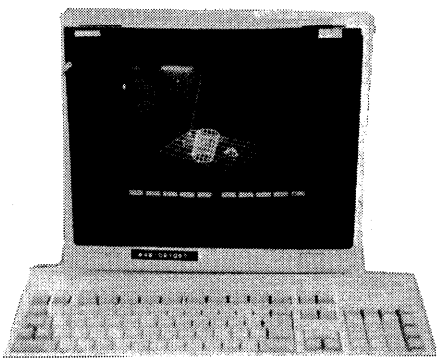

図4　３Ｄワイヤーグラフィックス表示例（写真２）

```
;*************************************************
;*      AFFINE TRANSFORMATION                    *
;*************************************************
MODULE   AFFINE = 8 ;      MODULE NUMBER = 8
EQUATE   L  =  16;
EQUATE   H  =  16;
EQUATE   V  = 128;
EQUATE   HOST  =  0;
EQUATE   READ  =  1;
EQUATE   WRITE =  5;
EQUATE   RHAD  =  30H;
EQUATE   MNUMB =  0H;   IF MODULE = 9 THEN MNUMB = 10H
EQUATE   WHAD  =  10H;
;*************************************************
;*       INPUT  OUTPUT                           *
;*************************************************
INPUT    RDATA AT 0;
INPUT    SETA,SETB,SETC,SETD,SETE,SETF;
INPUT    LSAO,LDAO,STATTK;
OUTPUT   RDAT,WDAT,WADR,PEND;
;*************************************************
;*       LINK                                    *
;*************************************************
LINK       SETA2     = FCV  (SETA); CONVERT 2'COMP TO ABS
LINK       SETB2     = FCV  (SETB); CONVERT 2'COMP TO ABS
LINK       SETC2     = FCV  (SETC); CONVERT 2'COMP TO ABS
LINK       SETD2     = FCV  (SETD); CONVERT 2'COMP TO ABS
LINK       SETE2     = FCV  (SETE); CONVERT 2'COMP TO ABS
LINK       SETF2     = FCV  (SETF); CONVERT 2'COMP TO ABS
LINK       STATK2    = FCV  (STATTK); CONVERT 2'COMP TO ABS
LINK                 = FSETA(SETA2); A SET
LINK                 = FSETB(SETB2); B SET
LINK                 = FSETC(SETC2); C SET
LINK                 = FSETD(SETD2); D SET
LINK                 = FSETE(SETE2); E SET
LINK                 = FSETF(SETF2); F SET
LINK                 = FSETRB (LSAO  ); READ ADDRESS BASE SET
LINK       X1,X2,X3  = FGENE  ( X46    , STATK2)    ;
LINK       X4,X5,X6  = FCOPY3( X1                )    ;
LINK       X7        = FBR    ( X4                )    ;
LINK       X8        = FCR    ( X7                )    ;
LINK                 = FW1    ( X8                )    ;
LINK       X9        = FBRD   ( X5                )    ;
LINK       X10       = FCRD   ( X9                )    ;
LINK                 = FW2    ( X10               )    ;
LINK       X11       = FRD2   ( X6                )    ;
LINK       X12,X13,X14     = FGEN2 ( X35 , X11 )    ;
LINK         X15,X16 = FCOPY2( X12               )    ;
LINK         X17     = FTT    ( X15              )    ;
LINK         X18     = FNYD   ( X17              )    ;
LINK         X19     = PSH    ( X18              )    ;
LINK         X20     = FTTD   ( X16              )    ;
LINK         X21     = PNY    ( X20              )    ;
LINK         X22     = PSHD   ( X21              )    ;
LINK         X23     = PIMX   ( X19              )    ;
LINK         X24     = FADDX  ( X23  ,X28        )    ;
LINK         X25     = FADDZ  ( X24              )    ;
LINK         RDAT    = FREAD  ( X25              )    ; IM READ
LINK         X26,X27 = FCOPY2( X22               )    ;
LINK         X28     = FSA    ( X263             )    ;
LINK         X265    = FSA4   ( X26              )    ;
LINK         X264    = FSA5   ( X265             )    ;
LINK         X261,X262=FSA1   ( X264             )    ;
LINK         X263    =FSA2    ( X261             )    ;
LINK         X263    =FSA3    ( X262             )    ;
LINK         X29     = PMY    ( X27              )    ;
LINK         X30     = FSUB   ( X29              )    ;
LINK         X31     = PBG1   ( RDATA  , X30     )    ;
LINK         X32     = PSHL3  ( X31              )    ;
LINK         X33,X34 = PARI   ( X32              )    ;
LINK         X35     = PQ1    ( X13    , X34     )    ;
LINK         PACKD   = PACC   ( X33              )    ;
LINK         X36,X37 = FCOPY2( PACKD             )    ;
LINK         WDAT  ,WADR   = FWRITE( X36 , X39 )    ;
LINK         X38     = PQ2    ( X40    , X37     )    ;
LINK         X39,X40,X41 = PGEN5 ( X38    , X43 )    ;
LINK         X42     = PQ6    ( X44    , X41     )    ;
LINK         X43,X44,X45 = PGEN7 ( X42  , LDAO  )    ;
LINK         PEND    =  ENDN ( X45    , X3      )    ;
LINK         X46     =  PQ7  ( X2     , X14     )    ;
```

（付録）

アフィン変換ＩｍＰＰプログラム

```
;*************************************************
;*       FUNCTION                                *
;*************************************************
FUNCTION     FCV     = CVT2AB                    ;
FUNCTION     FGENE   = COPYBK ( 1 , 32),CNTGE ( V ) ;
FUNCTION     FCOPY3  = COPYM ( 3 , 0)            ;
FUNCTION     FBR     = MUL       ,RDCYCS( CNTB ,1 ) ;
FUNCTION     FCR     = ADD       ,RDCYCS( CNTC , 1) ;
FUNCTION     FW2     =           WRCYCS( W2BUF, 1) ;
FUNCTION     FBRD    = MUL       ,RDCYCS( CNTE ,1 ) ;
FUNCTION     FRD2    =           ,RDCYCS( CNT0 ,1 ) ;
FUNCTION     FCRD    = ADD       ,RDCYCS( CNTF , 1) ;
FUNCTION     FW1     =           WRCYCS( W1BUF, 1) ;
FUNCTION     FCOPY2  = COPYM( 2, 0 )             ;
FUNCTION     FGEN2   = COPYBK (16 , 32),CNTGE(H   ) ;
FUNCTION     FTT     = MUL       ,RDCYCS( CNTD  ,1 ) ;
FUNCTION     PNY     = ADD       ,RDCYCS( W1BUF,1 ) ;
FUNCTION     PSH     = SHR       ,RDCYCS( SF    ,1 ) ;
FUNCTION     FTTD    = MUL       ,RDCYCS(CNTA  ,1 ) ;
FUNCTION     PNYD    = ADD       ,RDCYCS( W2BUF,1 ) ;
FUNCTION     PSHD    = SHR       ,RDCYCS( SF    ,1 ) ;
FUNCTION     PIMX    = MUL(Y)    ,RDCYCS(CIMX  ,1 ) ;
FUNCTION     FADDX   = ADD       ,QUEUE ( QUE1 ,16) ;
FUNCTION     FADDZ   = ADD       ,RDCYCS(BASE1 ,1 ) ;
FUNCTION     FREAD   = OUT1 ( READ, MNUMB + 0    )  ;
FUNCTION     FSA     = SHR       ,RDCYCS( SCNT ,1 ) ;
FUNCTION     FSA4    = CMPXCH(LE),RDCYCS( C127 ,1 ) ;
FUNCTION     FSA5    = CMPXCH(GE),RDCYCS( C128 ,1 ) ;
FUNCTION     FSA1    = CMP(BRC,GE),RDCYCS( C0000,1 );
FUNCTION     FSA2    = SUB,RDCYCS(C000F,1)       ;
FUNCTION     FSA3    = SUB,RDCYCS(C0000,1)       ;
FUNCTION     PMY     = AND       ,RDCYCS( C000F,1 ) ;
FUNCTION     FSUB    = SUB (XCH) ,RDCYCS(C15   ,1 ) ;
FUNCTION     PBG1    = GET1      ,QUEUE ( BSF  ,16) ;
FUNCTION     PSHL3   = SHL       ,RDCYCS( CSH  ,16) ;
FUNCTION     PARI    = COUNT (16)                ;
FUNCTION     PQ1     = QUEUE (QUE3 , 1 )         ;
FUNCTION     PACC    = ACC, COUNT(16)            ;
FUNCTION     FWRITE  = OUT2 ( WRITE ,20H,0),QUEUE ( QUE4,16 );
FUNCTION     PQ2     = QUEUE( QUE5, 1 )          ;
FUNCTION     PGEN5   = COPYBK(1,1),CNTGE(H   )   ;
FUNCTION     PQ6     = QUEUE( QUE8 , 1 )         ;
FUNCTION     PGEN7   = COPYBK(1,L) ,CNTGE ( V )  ;
FUNCTION     ENDN    = OUT1 ( HOST , 0 )   ,QUEUE ( QUE9,1  );
FUNCTION     PQ7     = QUEUE ( QUE11 , 1 )       ;
FUNCTION     FSETA   = WRCYCS( CNTA   , 1 )      ;
FUNCTION     FSETB   = WRCYCS( CNTB   , 1 )      ;
FUNCTION     FSETC   = WRCYCS( CNTC   , 1 )      ;
FUNCTION     FSETD   = WRCYCS( CNTD   , 1 )      ;
FUNCTION     FSETE   = WRCYCS( CNTE   , 1 )      ;
FUNCTION     FSETF   = WRCYCS( CNTF   , 1 )      ;
FUNCTION     FSETRB  = WRCYCS( BASE1  , 1 )      ;
;*************************************************
;*       MEMORY                                  *
;*************************************************
MEMORY       CNTC    = 0         ;
MEMORY       CNTB    = 2000H     ;
MEMORY       W1BUF   = AREA(1)   ;
MEMORY       CNTE    = 0         ;
MEMORY       CNTF    = 0         ;
MEMORY       W2BUF   = AREA(1)   ;
MEMORY       CNT0    = -4096     ;
MEMORY       CNTD    = 2000H     ;
MEMORY       SF      =  2        ;
MEMORY       CNTA    = 0         ;
MEMORY       CIMX    = L         ;
MEMORY       QUE1    = AREA(16)  ;
MEMORY       BASE1   = 0         ;
MEMORY       SCNT    = 4         ;
MEMORY       C000F   = 000FH     ;
MEMORY       C0000   = 0         ;
MEMORY       C15     = 15        ;
MEMORY       BSF     = AREA(16)  ;
MEMORY       CSH     = 15,14,13,12,11,10,9,8,7,6,5,4,3,2,1,0;
MEMORY       QUE3    = 1         ;
MEMORY       QUE4    = AREA(16)  ;
MEMORY       QUE5    = 1         ;
MEMORY       QUE8    = AREA(1)   ;
MEMORY       QUE9    = AREA(1)   ;
MEMORY       QUE11   = AREA(1)   ;
MEMORY       C127    = 127       ;
MEMORY       C128    =-128       ;
;*************************************************
;*       START                                   *
;*************************************************
 START;
DATA         EXEC ( WRITE   , RHAD, 0000H );
DATA         EXEC ( WRITE   , WHAD, 0000H );
DATA         EXEC ( AFFINE  , SETA, 2000H );
DATA         EXEC ( AFFINE  , SETB, 0000H );
DATA         EXEC ( AFFINE  , SETC, 0000H );
DATA         EXEC ( AFFINE  , SETD, 0000H );
DATA         EXEC ( AFFINE  , SETE, 2000H );
DATA         EXEC ( AFFINE  , SETF, 0000H );
DATA         EXEC ( AFFINE  , LSAO, 0000H );  SRC ADDRESS
DATA         EXEC ( AFFINE  , LDAO,   10H );  DST ADDRESS
DATA         EXEC ( AFFINE  , STATTK,  0 );
 END;
```