パーソナルコンピュータを用いた音響信号の取り込み*

山崎芳男・大越　幹・伊藤　毅**
（早稲田大学理工学部）

1. まえがき

音響信号処理におけるパーソナルコンピュータを導入する場合，パーソナルコンビユータは単に音響信号の入出力に使い，処理は大型コンピュータに頼るという使い方もある。しかし，パーソナルコンビュータの処理能力は急速に進歩しており一昔前の大型機を凌ぐものさえある。今後も益々その傾向は強まるものと考えられる。従って本稿ではあくまでもパーソナルコンピュータ単独での処理を前提としての信号の取り込みについて考えることにする。

パーソナルコンピュータの記憶容量は大きくなったとはいえ，大量の音響信号を主記憶装置上で扱うのは無理であり，一般にテープレコーダなどの録音装置を補助的手段として使用する。ところで数年前から家庭用VTRを使用するPCMプロセッサや業務用の多チャネルのPCMテープレコーダが数社から市販されており，一部では音響計測や音響信号処理に導入されている。パーソナルコンピュータで音響信号を扱う場合，当然のことながら信号はディジタル化されていなければならないので，音響信号をディジタル化して記録するPCMプロセッサやPCMテープレコーダとパーソナルコンピュータを結び付ければ，有効な使い道が数多くあるものと考えられる。著者らもPCMテープレコーダと大型計算機やパーソナルコンピュータとを組み合わせて使い重宝してきた。

本稿ではPCM-9800と市販のPCMプロセッサに若干改造を加えることにより構成した広帯域音響信号の入出力システムのハードウェア及びソフトウェアをできる限り具体的に解説する。

2. システム構成

2.1 ハードウェア構成

図－1に著者らが音響信号の処理に使用しているシステムのハードウェア構成を示す。PCMテープレコーダインタフェース，大容量RAM，実時間演算装置はパーソナルコンピュータ本体の拡張スロットに実装している。詳しくは本誌41巻1号の解説を参照されたい　。

2.2 入出力信号形式

図－2に入出力に使用しているシリアル信号形式を示す。基本的には32bitの等間隔のシリアル信号とデューティ比50%の語同期用クロックからなる簡単なものである。この32bitは1語として扱ってもよいし，16bit2語として使ってもよい。また，ビット転送用のビットクロックも定義している。このビットクロックはワードクロックからPLLで作り出すことも可能であるが，あらかじめ用意しておくとシフトレジスタだけで並列データに変換できるので簡単な実験等には便利である。

インタフェースのハードウェア仕様を図－5に示す。論理レベルはTTLコンパチブルでケーブルにはツイステッドペア線，コネクタにはDINの8ピンを図のように使っている。ツイステッドペア線で10m程度の伝送は全く問題ない。

この信号形式は著者らはPCMプロセッサやCDプレーヤとコンピュータとのインタフェースあるいは各種の処理装置間の信号のやりとりにすでに10年以上使ってきた。この信号形式の特長は形式の単純さにある。すなわちシフトレジスタさえ用意すれば並列データに変換が可能であり，簡単な実験をするときなど負担が軽いので多くの人々におっくうがらずに使われて，定着した。

最近ディジタル出力端子つきのCDプレーヤが数社から登場している。その出力信号形式は1ラインで同期も信号も同時に伝送している。個別部品で構成すると同期分離等で回路構成が複雑になるが，専用LSIが市販されるようになれば，1ライン伝送インタフェースが実用化されよう。

2.3 PCMプロセッサ

音響信号を処理するには大量のデータを扱わざるをえない。従来はテープレコーダやFM方式のデータレコーダが使われていたが，時間軸精度の点で伝送路の隘路となっていた。そこでこれらの欠点を補う意味でPCMテープレコーダは注目されてきた。現時点では装置の価格や運用コストの点からもPCMテープレコーダが有

---

* Acoustic signal acquisition by personal computer.
** Yoshio Yamasaki, Miki Okoshi and Takeshi itow
(Waseda University, Tokyo, 160)

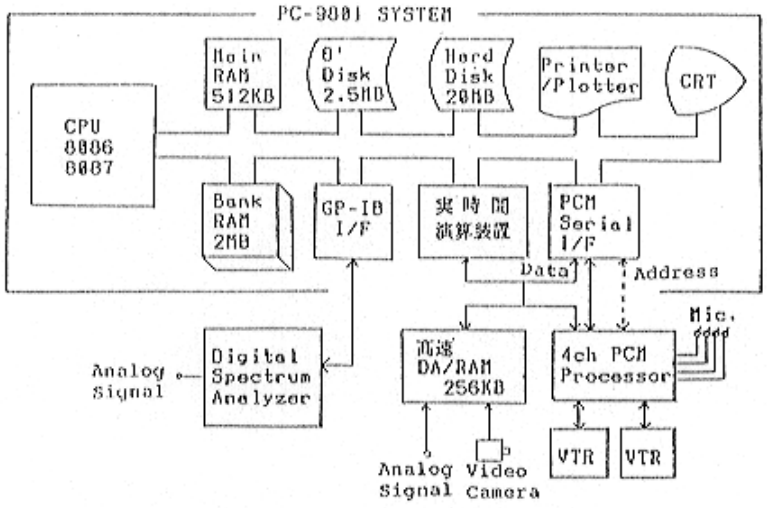


図-1　PC-9800を用いた音響信号処理システムの構成

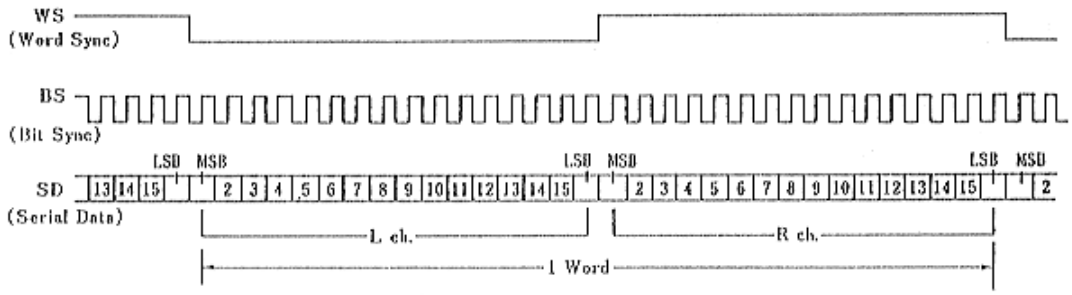


図-2　シリアルデータフォーマット

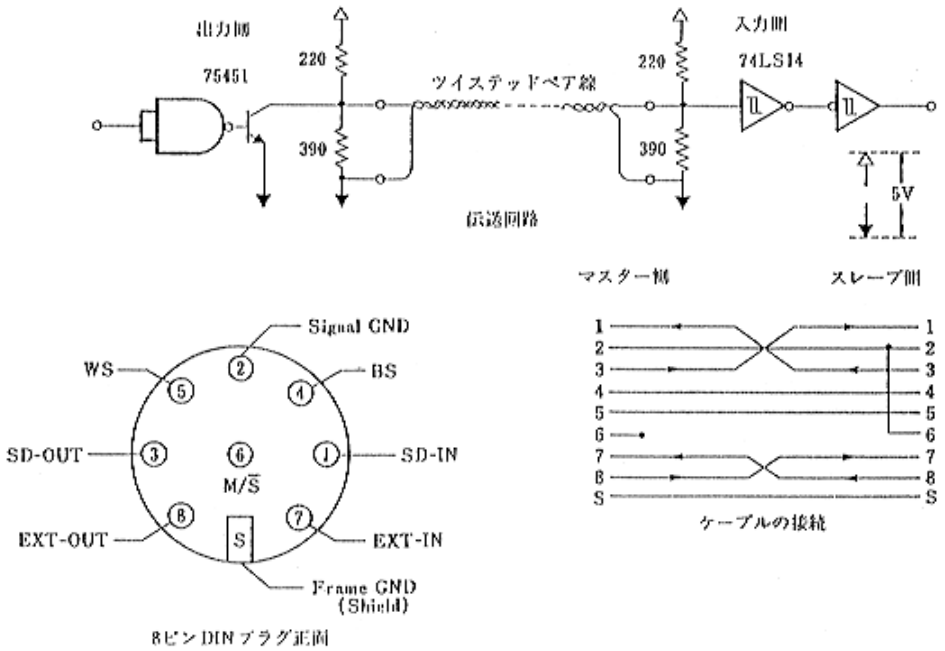
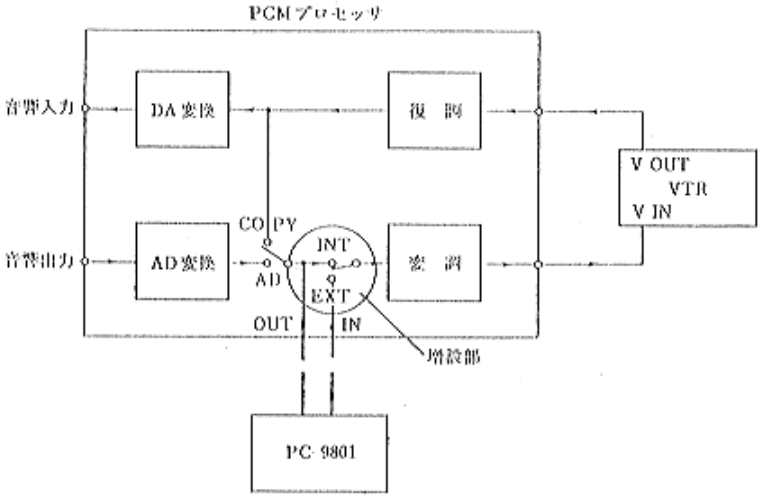


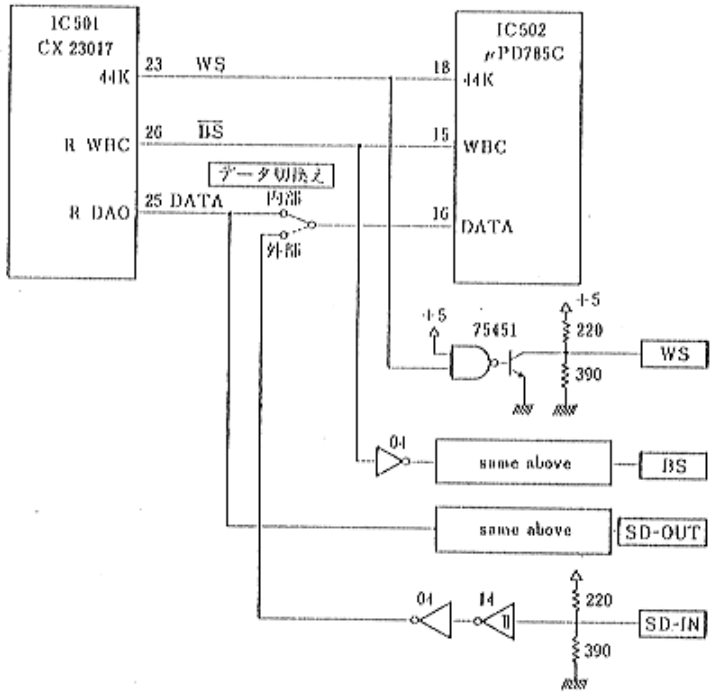
図-3　インターフェースの仕様

利になってきた。

1979年に日本電子機械工業会で家庭用VTRを利用したPCMプロセッサに関する技術ファイルがまとめられた。この信号形式に基づくPCMプロセッサが数社から発売され，最近ではLSI化と大量生産により10万円を割る価格の16bit AD，DA変換器をそなえたハー



図－4　PCMプロセッサの構成



図－5　PCMプロセッサPCM－501 ESの改造例

ドウェアさえ登場している。

測定，分析用としては変換精度や周辺回路に若干問題もあるかも知れないが，これらを入手して改造を加えて使うことができれば，たとえAD，DA変換器としてだけみても意味がありそうである。そのうえPCMプロセッサには強力な誤り訂正機能が完備されているので，音響信号ばかりでなくデータやプログラムすらVTRのテープに記録保存が可能である。すなわちVTRをパーソナルコンピュータで制御することにより大容量の外部記憶付きのシステムを構築することが可能である。

PCMプロセッサの一般的な構成を**図－4**に示す。**図－5**にソニーのPCMプロセッサPCM－501にインタフェース回路を付加する改造例を示す。他の多くのPCMプロセッサやCDプレーヤも標本化周波数に同期したクロックとディジタル信号を取り出せば同様の改造は容易である。

2.4　インタフエース [2),3)]

PCMプロセッサとPC－9801とのインタフェースのハードウェア構成を図－6に示す。また，表－1に入出力ポートの番地を示す。入出力とも16bitのデータラッチとシフトレジスタにより直列並列変換を行うだけの岬純な構成であり，PC－9801のユニバーサルボード1枚に十分おさまる。両者のグラウンドの縁を切るためにフォトカブラを用いたり，同期を含めた1本の線路によるやりとりも導入しているが，ここでは基本的な回路の紹介に止める。事前にPCMプロセッサを使ってVTRに

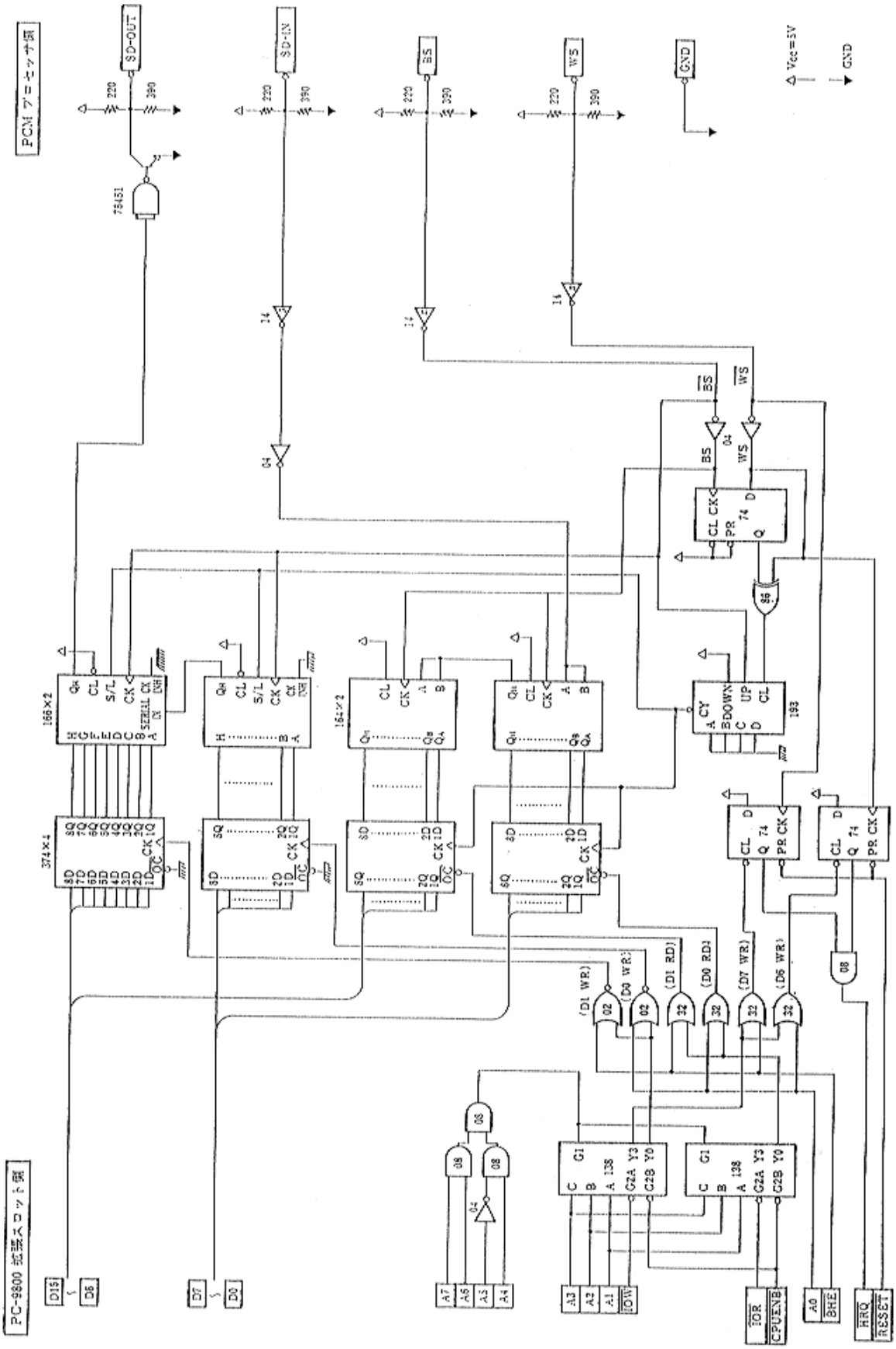


図6　PC-9800用シリアルインタフェース回路

表-1　入出力ポート番地

I/Oアドレス	機　　能
D0H (read/write)	データ下位　8ビット
D1H (read/write)	データ上位　8ビット
D6H (wirte)	L　ch.　同期要求
D7H (write)	R　ch.　同期要求

録音した信号をオフラインで取り込む場合にはディジタル信号のやりとりだけであり，グラウンドラインを介してのアナログ系へのまわりこみは問題にならない。また，特に高精度を要求したり，雑音の多い他の周辺機器と共存でもしない限りオンラインの使用に対しても実用上問題となることは少ない。

2.5　$\overline{\text{HRQ}}$ 端子を利用した同期

標本化周波数をPC－9800側が決定する場合には同期の問題は生じない。しかしPCMプロセッサやCDプレーヤあるいはその他の外部機器の標本化周波数でPC－9800に信号を入力できなければ取り込み装置としての魅力はない。

この場合，両者の同期が必要となる。標本化周波数44.1kHzで2チャネルの16bit信号を入出力する場合のデータの転送速度は1.41Mbit/sと高密度のフロッピーディスク並の速さであり，かつフロッピーディスクのようなデータ転送の休止区間がない。一般に外部クロックでデータを入出力するには割り込み処理が使われる。しかるに8086には割り込みや条件ジャンプといった分枝処理が遅いという欠点があり，通常の割り込み処理は使うことができない。そこでPC－9800拡張スロットの $\overline{\text{HRQ}}$ （Hold Request）端子を利用して切り抜けることとした。

$\overline{\text{HRQ}}$ 端子は元来DMA時にCPUを強制的にHOLD状態にする端子で，$\overline{\text{HRQ}}$ 端子をLowにするとCPUは停止し，Highにすると実行を再開する。従ってこの端子をCPUのOUT命令と外部クロックにより制御する回路を設けてやれば，外部クロックに同期した高速の入出力が可能となる。図－6のIOアドレスD6HとD7HのLS74がその回路である。このアドレスに対してOUT，命令を実行すると次の外部クロックの変化があるまでCPUはHOLD状態となるので，その直後に入出力命令をおけばよい。64kHz2チャネルのデータを入出力する典型的な命令シーケンスをリスト－1に示す。

```
;-----------------------
; sample of PCM INPUT
;-----------------------
;
;
CLI
CLD
MOV     AX,2000H
MOV     ES,AX           ;segment of data buffer
MOV     DI,0            ;offset  of data buffer
MOV     CX,4000H        ;number of data to get (x4 bytes)
ILOOP:  OUT     0A0H,AL ;stop CPU until positive edge of H3
IN      AX,0A0H         ;input L ch. data
STOSW                   ;store it to buffer
OUT     0A7H,AL         ;stop CPU until negative edge of H3
IN      AX,0A0H         ;input H ch. data
STOSW                   ;store it
LOOP    ILOOP           ;repeat
;
;
;-----------------------
; sample of PCM OUTPUT
;-----------------------
;
;
CLI
CLD
MOV     AX,2000H
MOV     DS,AX           ;segment of data buffer
MOV     SI,0            ;offset  of data buffer
MOV     CX,4000H        ;number of data to put
OLOOP:  LODSW           ;load data from buffer
OUT     0A6H,AL         ;stop CPU until positive edge of H3
OUT     0A0H,AX         ;output L ch. data
LODSW                   ;load data
OUT     0A7H,AL         ;stop CPU until negative edge of H3
OUT     0A0H,AX         ;output R ch. data
LOOP    OLOOP           ;repeat
;
;
END
```

リスト-1　基本的な信号入出力プログラム例

3.　ソフトウェア

3.1　入出力プログラム

PC－9800を8MHzクロックで動作させるとプログラム制御下で，測定用音源としてインパルスやM系列雑音を出力し，同時に4チャネルの取り込みが可能である。著者らは近接4点法による建築音響測定に利用している。ここでは5MHzクロックでも動作する2チャネルの入出力についてプログラムのサブセットを紹介する。

このプログラムはPC－9800の主記憶装置の20000H～9FFFFH番地の仕意の場所にディジタル化された音響信号を入出力するものである。この領域をそのまま使用すると512KByteであるが，メモリの一部をバンク切り換えで使用することにより容量は必要に応じて拡張することができる。512KByteのとき標本化周波数44.1kHzの音響信号1チャネルを5.8s，2チャネルを2.9s蓄えることができる。

プログラムはMS－DOS（Ver 2.11）上で動作し，使用言語はComputer Innovation社のOptimizing C86とMS－DOSの標準アセンブラMASMとである。リスト-2，3にそのソースリストを，**リスト-4**にコンパイル及びリンクを行うバッチファイルを示す。Opitimizing C86は64KByte以上のデータを取り扱いやすいようにbigモデルでコンパイルしている。

アセンブラ部分は1標本（4Byte）ごとの番地指定や5MHzクロックでの動作を確保するために工夫がこらされている。特に8086の64KByte単位のセグメント境界では処理速度の点で細心の注意が要求される。そのために自己書き換えの機能を使用している。また，ソースファイルも簡潔にするためにマクロ機能を多用している。

3.2　メモリ管理 [4)]

取り込まれた音響信号はC言語の変数や配列としてバイト単位で扱うのではなく，音響信号の標本単位すなわち1チャネル当たり16bit単位で絶対番地をふって扱った方が便利である。

一方，膨大な信号の入出力や編集あるいは処理を一つ

```
        /*****************************************
                pc.c
            pcm processor interface
                demonstration
            1985 Accoustic Lab. Waseda Univ.
        *****************************************/
#include "stdio.h"

typedef int void ;
        /* unsigned data type */
typedef unsigned char byte ;
typedef unsigned int word ;
typedef unsigned long dword ;
        /* get high/low part of int/long */
#define hiword(l) ((word *)&l)[1]
#define loword(l) (word)(l)
        /* direct console i/o */
#define getcon() (char)bdos(1)
#define getkey() (char)bdos(8)

        /***** PCM I/O routines *****/
extern byte pcmstartpage,pcmendpage ;
extern word pcmstartofst,pcmendofst ;
void pcmoutplr(),pcminplr() ;

        /***** global variables *****/
char chr[128] ;
dword pcmsadrs,pcmeadrs ;

#define NULKEY -1L

int yesno(msg)
  char *msg ;
{
  char c ;
  printf(msg) ;
  c=getcon() ;
  if (c=='Y') return 1 ;
  else if (c=='N') return 0 ;
  else return -1 ;
}

dword keydata(msg,fmt)
  char *msg,*fmt ;
{
  static dword val=0 ;
  printf(msg) ;
  fgets(chr,40,stdin) ;
  if (sscanf(chr,fmt,&val)==0) val=NULKEY ;
  return val ;
}

pcmadrsset()
{
  byte bk ; char c ; dword ad ;
  pa0:  printf("Current I/O mode is....¥n") ;
  printf("       from %5lxH¥n",pcmsadrs) ;
  printf("         to  %5lxH¥n",pcmeadrs) ;
  if (yesno("OK? ([Y]/N) ")!=0) return ;
  pa1:
  printf("¥nStart address (20000-9FFFF)? [%5lxH] ",pcmsadrs) ;
    ad=keydata("","%5lx") ;
    if (ad!=NULKEY) pcmsadrs=ad ;
  printf("End address   (20000-9FFFF)? [%5lxH] ",pcmeadrs) ;
    ad=keydata("","%5lx") ;
    if (ad!=NULKEY) pcmeadrs=ad ;
  if (pcmsadrs<0x20000L || pcmsadrs>0xa0000L) goto pa1 ;
  if (pcmeadrs<0x20000L || pcmeadrs>0xa0000L) goto pa1 ;
  if (pcmsadrs>=pcmeadrs) goto pa1 ;
  pcmstartofst=loword(pcmsadrs) ;
  pcmendofst=loword(pcmeadrs) ;
  pcmstartpage=hiword(pcmsadrs)-2 ;
  pcmendpage=hiword(pcmeadrs)-2 ;
  goto pa0 ;
}

char keywait()
{
  char c ; word n ;
  c=getkey() ;
  while ((inportb(0x41) & 0x08)==0)
    for (n=0;n<500;n++) ;
  putchar('¥n') ;
  return c ;
}

main()
{
    byte cmd,c,n,flgint ;
    word m ;
  pcmsadrs=0x20000L ;
  pcmeadrs=0x9ffffL ;
  pcmstartpage=0 ; pcmstartofst=0 ;
  pcmendpage=7   ; pcmendofst=0xffff ;
  printf
  ("     ----- pcm 2ch. interface demonstration -----¥n") ;
  while(1)
  {
    printf
    ("I: Input  O: Output  Q: End¥n") ;
    printf("Command? ") ;
    cmd=getcon() ;
    if (cmd=='Q')
      return ;
    if (cmd=='I')
    {
      printf("¥n< PCM INPUT >¥n") ;
      pcmadrsset() ;
      printf("¥n         push space to input") ;
      c=keywait() ;
      if (c==' ') pcminplr() ;
    }
    if (cmd=='O')
    {
      printf("¥n< PCM OUTPUT >¥n") ;
      pcmadrsset() ;
      printf("¥n         push space to output") ;
      printf("¥n         return to repeat") ;
      c=keywait() ;
      if (c==' ' || c=='¥r') pcmoutplr() ;
    }
  }
}
```

リスト -2　プログラムリストC言語部（PC. C）

```
.sall
        ;*************************************
        ;  PCM processor interface for C86
        ;  simple 2ch. subset
        ;*************************************
PUBLIC  PCMOUTLR,PCMINPLR
PUBLIC  PCMSTARTPAGE,PCMSTARTOFST,PCMENDPAGE,PCMENDOFST

        ;***** Bank port *****
PAGENUM EQU     8       ;number of segment page
        ;***** I/O address *****
APORT   EQU     0D0H
LSPORT  EQU     0D8H
RSPORT  EQU     0D1H
NOPCODE EQU     90H
RETCODE EQU     0C3H

LSYNC   MACRO
         OUT    LSPORT,AL
        ENDM
RSYNC   MACRO
         OUT    RSPORT,AL
        ENDM
MOVDS   MACRO   SGM
         MOV    AX,SGM
         MOV    DS,AX
        ENDM
MOVES   MACRO   SGM
         MOV    AX,SGM
         MOV    ES,AX
        ENDM

JMPOW   MACRO   L,N
         DW     OFFSET L&&N
        ENDM
JMPTBL  MACRO   LBL
        N = 0
        REPT    PAGENUM
         JMPOW  LBL,%N
         N = N+1
        ENDM
        ENDM
DDW     MACRO   D,W
        DD      D
        DW      W
        ENDM

DGROUP  GROUP   @DATAC
        ;***** common variables with C86 *****
@DATAC  SEGMENT PUBLIC  'DATAC'
PCMSTARTPAGE    DB      ?
PCMENDPAGE      DB      ?
PCMSTARTOFST    DW      ?
PCMENDOFST      DW      ?
@DATAC  ENDS

PCMSUB  SEGMENT 'CODE'
        ASSUME  CS:PCMSUB,DS:PCMSUB

FIRSTSEGLNG     DW      ?
LASTSEGLNG      DW      ?
RETOFST         DW      ?
LOOPOFST        DW      ?
RETSAVE         DB      ?
FIRSTSEG        DW      ?


        ;----- L/R-ch output -----
LROUTSTBL       LABEL WORD
        JMPTBL  OUTLRS
LROUTETBL       LABEL WORD
        JMPTBL  OUTLRE
LROUTCTBL       LABEL WORD
        JMPTBL  OUTLRC

SEGOUTLR        MACRO   X,BK,SG
                LOCAL   LLL,XXX
OUTLRC&BK:MOV   CX,0FFFFH
        MOVDS   SG
         LSYNC
        LODSW
        OUT     DX,AX
        LODSW
         RSYNC
        OUT     DX,AX
        JCXZ    XXX
OUTLRS&BK:
LLL:     LSYNC
        LODSW
        OUT     DX,AX
        LODSW
         RSYNC
        OUT     DX,AX
        LOOP    LLL
OUTLRE&BK:
XXX:
        ENDM

PCMOUTLR        PROC    FAR

        PUSH    BP
        MOV     BP,SP
        PUSH    DS
        PUSH    ES
        MOVES   DS
        MOVDS   CS

        MOV     AX,ES:PCMSTARTOFST
        SHR     AX,1
        SHR     AX,1    ;AX=AX/4
        MOV     CX,4000H
        SUB     CX,AX
        MOV     FIRSTSEGLNG,CX

        MOV     DL,ES:PCMENDPAGE
        XOR     DH,DH
        SHL     DL,1                    ;DX=end table index
        MOV     AX,ES:PCMENDOFST
```

リスト -3－1

```
121:          MOV     BL,ES:PCMSTARTPAGE
122:          CMP     BL,ES:PCMENDPAGE
123:            JNZ   POUTLR1
124:            SUB   AX,ES:PCMSTARTOFST      ;if start=end page
125:            SHR   AX,1
126:            SHR   AX,1
127:            INC   AX
128:            MOV   FIRSTSEGLNG,AX
129:            JMP   SHORT POUTLR2
130: POUTLR1:SHR     AX,1
131:          SHR     AX,1
132: POUTLR2:MOV     DI,<OFFSET LROUTCTBL>
133:          MOV     CS:[DI],AX          ;"MOV CX,3FFFH" code
134:          MOV     LOOPOFST,DI
135:
136:          MOV     BX,<OFFSET LROUTETBL>
137:          MOV     AL,<DX>
138:          MOV     RETSAVE,AL      ;move instruction
139:          MOV     BYTE PTR <BX>,RETCODE
140:          MOV     RETOFST,BX
141:
142:          MOV     BL,ES:PCMSTARTPAGE
143:          XOR     DH,DH
144:          SHL     BL,1                ;BX=start table index
145:          PUSH    BX
146:            ADD   BL,ES:PCMSTARTPAGE      ;BX=start bank/seg table index
147:            ADC   DH,0
148:            MOV   AX,<DX+OFFSET PAGETBL+1>
149:            MOV   FIRSTSEG,AX
150:          POP     BX
151:          MOV     DX,<DX+OFFSET LROUTSTBL>
152:
153:          MOV     BP,0333H
154:          MOV     DI,9999H
155:          MOV     SI,ES:PCMSTARTOFST
156:          MOV     CX,FIRSTSEGLNG
157:          MOV     DX,AFORT
158:          MOV     DS,FIRSTSEG
159:          MOV     AX,OFFSET OUTLR99
160:          PUSH    AX
161:          CLD
162:          CLI
163:          JMP     BX
164:
165:          SEGOUTLR        0,0,2888H
166:          SEGOUTLR        1,0,3888H
167:          SEGOUTLR        2,0,4888H
168:          SEGOUTLR        3,0,5888H
169:          SEGOUTLR        4,0,6888H
170:          SEGOUTLR        5,0,7888H
171:          SEGOUTLR        6,0,8888H
172:          SEGOUTLR        7,0,9888H
173:          POP     AX
174:
175: OUTLR00:
176:          IN      AL,41H
177:          CMP     AL,0CH
178:          JNZ     OUTLR993
179:            SUB   BP,2    ;push OUTLR99 address
180:            MOV   DS,CS:FIRSTSEG
```

```
181:            MOV   SI,ES:PCMSTARTOFST
182:            MOV   CX,CS:FIRSTSEGLNG
183:            JMP   DX      ;repeat output
184:
185: OUTLR993:XOR     AX,AX
186:          OUT     DX,AX
187:          STI
188:          MOV     AX,C9
189:          MOV     DS,AX
190:          MOV     BX,RETOFST
191:          MOV     AL,RETSAVE
192:          MOV     <BX>,AL
193:          MOV     BX,LOOPOFST
194:          MOV     <BX+1],3FFFH
195:          POP     ES
196:          POP     DS
197:          POP     DI
198:          RET
199: PCMOUTLR         ENDP
200:
201:          ;----- L/R-ch input -----
202: LR1MPSTBL        LABEL WORD
203:          JMPTBL  1MPLR3
204:
205: LR1MPETBL        LABEL WORD
206:          JMPTBL  1MPLR0
207:
208: LR1MPCTBL        LABEL WORD
209:          JMPTBL  1MPLRC
210:
211: SEG1MPLR         MACRO   M,DK,SG
212:            LOCAL LLL,KKK
213: 1MPLRC0N:MOV     CX,3FFFH
214:          MOVES   SG
215:          IN      AX,DK
216:            HSYNC
217:          STOSD
218:          IN      AX,DK
219:          STOSD
220:            LSYNC
221:          JCXZ    KKK
222: 1MPLR34:
223: LLL:     IN      AX,DX
224:            HSYNC
225:          STOSW
226:          IN      AX,DX
227:          STOSW
228:            LSYNC
229:          LOOP    LLL
230: ENPLRECH:
231: KKK:
232:          ENDM
233:
234: PCM1MPLR         PROC    FAR
235:
236:          PUSH    BP
237:          MOV     BP,SP
238:          PUSH    DS
239:          PUSH    ES
240:          MOVES   DS
```

リスト－3－2

```
241:          MOVES   ES
242:
243:          MOV     AX,ES:PCMSTARTOFST
244:          SUB     AX,1
245:          SHR     AX,1    ;AX=AX/4
246:          MOV     CX,4888H
247:          SUB     CX,AX   ;CX=1st segment length
248:          MOV     FIRSTSEGLNG,CX
249:
250:          MOV     BL,ES:PCMENDPAGE
251:          XOR     BH,BH
252:          SHL     BL,1                ;BX=end table index
253:          MOV     AX,ES:PCMENDOFST
254:          MOV     DL,ES:PCMSTARTPAGE
255:          CMP     DL,ES:PCMENDPAGE
256:            JNZ   P1MPLR1
257:            SUB   AX,ES:PCMSTARTOFST      ;if start=end page
258:            SHR   AX,1
259:            SHR   AX,1
260:            INC   AX
261:            MOV   FIRSTSEGLNG,AX
262:            JMP   SHORT P1MPLR2
263: P1MPLR1:SHR     AX,1
264:          SHR     AX,1
265:          MOV     DI,<BX+OFFSET LR1MPCTBL>      ;"MOV   CX,3FFFH" code
266:          MOV     <DI+1],AX           ;AX=last segment length
267:          MOV     LOOPOFST,DI
268:
269: P1MPLR2:MOV     BX,<BX+OFFSET LR1MPSTBL>
270:          MOV     AL,<BX>
271:          MOV     RETSAVE,AL      ;save instruction
272:          MOV     BYTE PTR <DX>,RETCODE
273:          MOV     RETOFST,DX
274:
275:          MOV     BL,ES:PCMSTARTPAGE
276:          XOR     DH,DH
277:          SHL     BL,1                ;BX=start table index
278:          PUSH    BX
279:            ADD   BL,ES:PCMSTARTPAGE      ;BX=start bank/seg table index
280:            ADC   DH,0
281:            MOV   AX,<DX+OFFSET PAGETBL+1>
282:            MOV   FIRSTSEG,AX
283:          POP     BX
284:          MOV     DX,<DX+OFFSET LR1MPSTBL>
285:
286:          MOV     BP,0333H
287:          MOV     DI,9999H
288:          MOV     DL,ES:PCMSTARTOFST
289:          MOV     CX,FIRSTSEGLNG
290:          MOV     DX,AFORT
291:          MOV     ES,FIRSTSEG
292:          MOV     AX,OFFSET 1MPLR99
293:          PUSH    AX
294:          CLD
295:          CLI
296:          LSYNC
297:          JMP     BX
298:
299:          SEG1MPLR        0,0,2888H
300:          SEG1MPLR        1,0,3888H
```

```
301:          SEG1MPLR        2,0,4888H
302:          SEG1MPLR        3,0,5888H
303:          SEG1MPLR        4,0,6888H
304:          SEG1MPLR        5,0,7888H
305:          SEG1MPLR        6,0,8888H
306:          SEG1MPLR        7,0,9888H
307:          POP     AX
308:
309: 1NTLR00:
310:          STI
311:          MOV     BX,RETOFST
312:          MOV     AL,RETSAVE
313:          MOV     <BX],AL
314:          MOV     BX,LOOPOFST
315:          MOV     <BX+1],3FFFH
316:          POP     ES
317:          POP     DS
318:          POP     DI
319:          IRET
320: PCM1MPLR         ENDP
321:
322: PAGETBL LABEL   BYTE
323:          DBW     0,2999H
324:          DBW     0,3999H
325:          DBW     0,4999H
326:          DBW     0,5999H
327:          DBW     0,6999H
328:          DBW     0,7999H
329:          DBW     0,8999H
330:          DBW     0,9999H
331:
332: PCM30B   END3
333:          END
334:
```

リスト－3－3

リスト－3　プログラムリストアセンブラ部（PCSUB, ASM）

```
A>type b:@pc.bat
****** Batch command to generate PC.EXE ******
MASM B:PCSUB,B:PCSUB;
CC1 B:PC -B1
CC2 B:PC
CC3 B:PC
CC4 B:PC
LINK B:PC+B:PCSUB,B:PC,CON,C86D2N;
```

リスト－4　PC, EXEを作成するパッチコマンド

のプログラムにまとめるとプログラムが巨大化し能率が悪い。そこで幾つかのプログラムを組み合わせて使うことになる。この場合データをどのプログラムからもアクセスする必要があり，この点からもデータに絶対番地が要求される。

ところでPC－9800のMS－DOSのメモリマップは**図－7**のような構成となっている。従ってCOMAND.COMや他のプログラムの実行で音響信号を格納する20000H～9FFFFH番地の内容が破壊されてしまう。そこで取り込みプログラムの実行に先立ちTPA（Transient Program Area）の先頭にあるメモリコントロールブロックを書き換えてMS－DOSで使用するメモリを20000H以下に制限してやる必要がある。**リスト－5**がこのプログラムLIMIT．COMのリストである。**図－7**が“LIMIT2000”を実行した後のメモリマップである。セグメントアドレス2000H（実アドレス20000H）以降の領域はMS－DOS本体からは無視され，通常のMS－DOSコマンドを使用している限り20000H～9FFFFH番地はデータ領域として確保される。

リスト－6にPCMプロセッサからの入力を実行した例を示す。

```
A>DEBUG LIMIT.COM
-D180 2AF
1424:0180  0C 1E 30 02 04 38 CD 21-3C 02 75 0B 88 00 A0 0E
1424:0190  C8 26 0A 26 EA 3F 00 88-00 E4 07 01 05 D3 E0 05
1424:01A0  00 20 A3 34 02 0C D0 40-0E C8 00 3E 00 00 05 72
1424:01B0  20 0A 02 00 E0 95 00 72-20 30 00 34 02 77 3E 0B
1424:01C0  1E 34 02 01 C3 00 0D 30-C3 76 32 A3 32 02 A1 32
1424:01D0  02 2B 06 30 02 26 A3 03-00 8A 0B 02 B4 09 CD 21
1424:01E0  26 A1 03 00 03 06 30 02-E0 AA 00 0A 79 02 04 09
1424:01F0  CD 21 CD 20 0A 36 02 04-09 CD 21 CD 20 0A 50 02
1424:0200  04 09 CD 21 CD 20 53 51-57 33 DB 8F 97 02 B9 10
1424:0210  00 2E 3A 81 74 03 43 E2-F0 0A C3 5F 59 58 C3 24
1424:0220  0F 53 00 97 02 2E D7 50-C3 53 0D 0A 0A 07 E0 D5
1424:0230  FF 75 16 0A E8 43 0A 07-E0 C0 FF 75 0C D0 E4 D0
1424:0240  E4 D0 E4 D0 E4 0A C4 5B-C3 F9 50 C3 53 52 E0 D0
1424:0250  FF 72 09 0A D0 42 42 E0-CF FF 0A E3 5A 58 C3 53
1424:0260  50 D0 E0 D0 E0 D0 EB D0-E0 0B DA EB B1 FF B0 07
1424:02F0  43 58 E0 AA FF 00 07 50-C3 52 06 C4 E0 E0 FF 42
1424:0280  42 0A C4 EB D9 FF 5A C3-52 0A A9 82 E0 D0 FF B4
1424:0210  09 CD 21 5A C3 52 0A A7-02 E0 DD FF B4 09 CD 21
1424:02A0  5A C3 00 00 00 00 00 00-00 00 00 00 00 00 00 00
1424:0230  00 00 00 00 00 00 20 4D-53 20 44 4F 53 20 56 65
1424:02C0  72 32 2E 30 20 00 EA 97-70 02 CC 03 52 03 70 03
1424:02D0  93 03 60 02 C5 02 B7 24-03 01 B3 02 03 0A 01 5B
1424:02E0  02 AA 95 73 91 A0 02 C5-02 B7 24 20 03 5A 03 4F
1424:02F0  03 01 03 93 03 67 01 40-24 40 01 48 96 A2 96 9E
1424:0300  02 F0 20 4D 53 20 44 4F-53 20 02 C5 06 67 07 70
1424:0290  02 05 02 DC 02 B7 24 30-31 32 33 34 35 36 37 30
1424:02A0  39 41 42 43 44 45 46 30-30 30 30 24 00 00 00 00
-Q

A>
```

リスト-5　LIMITコマンド（DEBURGによるダンプリスト）

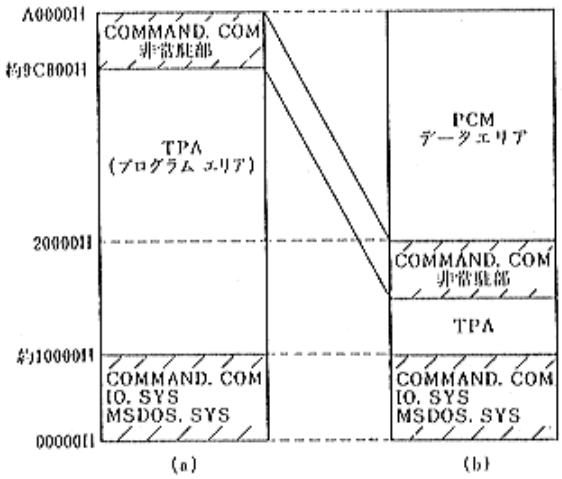


図-7　PC-9800MS-DOSメモリマップ

```
A>LIMIT                                  LIMIT.COM
 セグメント   A000H  非満を MS-DOS で使用します   <-- 引数を与えないと MS-DOS
                                                     の管理するメモリ上限を
                                                     表示

A>LIMIT 2000                             <-- 音響信号入出力の前に必
 セグメント   2000H  非満を MS-DOS で使用します       ず MS-DOS 使用メモリを
                                                     制限する

A>PC                                     <-- PC.EXEの起動
----- pcm 2ch, interface demonstration -----
I: Input  O: Output  Q: End
Command? I                               <-- 入力 I コマンド
< PCM INPUT >
Current i/o mode is....                  <-- 入出力データアドレスを
        from 20000H                          表示
          to  9FFFFH
OK? ([Y]/N) Y                            <-- OK なら Y を押す
        push space to input              <-- SPACE KEY で入力開始

I: Input  O: Output  Q: End
Command? O                               <-- 出力 O コマンド
< PCM OUTPUT >
Current i/o mode is....
        from 20000H
          to  9FFFFH
OK? ([Y]/N) N                            <-- 入出力アドレスを変更
Start address (20000-9FFFF)? [20000H] 30000      するとき N を押し
End address   (30000-9FFFF)? [9FFFFH] 5FFFF  <-- アドレスを入力
Current i/o mode is....                      START = 4n
        from 30000H                          END   = 4m-1
          to  5FFFFH
OK? ([Y]/N) Y                            <-- 確認して Y
        push space to output             <-- SPACE KEY で出力開始
        return to repeat                 <-- RETURN KEY で繰返し
                                             (他のKEYを押すまで)
I: Input  O: Output  Q: End
Command? Q                               <-- Q コマンドで終了

A>LIMIT A000                             <-- 終了後全メモリをMS-DOS
 セグメント   A000H  非満を MS-DOS で使用します       管理下に戻す

A>
```

リスト-6　PC．EXEの実行例

4. むすび

マイクロプロセッサの出現により，コンピュータのハードウェアや機械語が使用者に開放された。この意味は大きく，使用者が目的に応じたコンピュータを自ら構成することが可能になった。回路構成やOSがすべて開放されているわけではないパーソナルコンピュータの場合でも，若干のハードウェアの知識があれば各種の機器との組み合わせにより文字通りの“パーソナル”コンピュータシステムを構築することができる。著者らは大量生産によって大幅に価格の低減したPCMプロセッサをいわば少し大きなハイブリッド集積回路としてとらえ，入出力回路を付け加えてパーソナルコンピュータの入出力装置として利用している。

一昔前には大規模な装置と莫大な費用を要したシステムが手軽に身近に実現できるようになった。一般には価格の低減だけに目がいきがちであるが，実は身近に手軽に使えるということはより重要な意味をもっている。考えてみればコンピュータが貴重品扱いされている間はまだまだ本物ではなく，コンピュータの存在を意識せずにいつでも好きな時に好きなように使えるようになって初めてその真価を発揮するといえる。パーソナルコンピュータの進歩はすでにその環境を十分与えてくれているといえよう。我々は正にいまその対応能力を問われているといえるのではないだろうか。

文　献

1) 山崎芳男,“パーソナルコンピュータでどんな音響処理ができるか,” 音響学会誌41，51－55（1985）.
2) iAPX86　ファミリ・ユーザーズマニュアル（インテルジャパン，茨城，1981）.
3) 壁谷正洋，森野雅彦，PC－9801全回路図（アスキー出版東京，1983）.
4) 標準MS－DOSハンドブック（アスキー出版，東京，1984）.