マイクロコンピュータを用いたアナログデータの取り込み法

坂　田　省　吾

広島大学総合科学部人間行動研究講座

(1991年10月31日受理)

A method of analog data sampling using a microcomputer

Shogo SAKATA

Abstract

I have developed a framework of analog data sampling for psychophysiological laboratories. Within this framework, an experimenter can specify stimulus events and their timing, collection of responses, and collection of physiological data. Therefore many investigators will be able to treat some analog data more easily.

序　　論

心理学的な現象をみる際のひとつの従属変数として生理学的指標を扱うことがある。生体電気現象としては，心電図，筋電図，脳波等非常に多くあるが，いずれもアナログデータである。従来，生体電気反応の解析にはチャート紙に書き出したものを定規を用いて数値化する方法から，専用の非常に高価な解析装置を用いて処理する方法まで多々存在するが，前者の方法では膨大な時間と労力がかかり，現在のように大量のデータ処理を必要とされる場合には事実上適さない。また後者の場合は理想ではあるが，現在の研究費の実状から，極一部の恵まれた研究者だけしか使用できない。従って，現在多くの研究者が必要としているものは，データ処理能力が高くて，取扱いが容易でかつ比較的安価なシステムである。

現在の心理学，生理学の研究ではマイクロコンピュータを使用する頻度が増えてきており，神経生理学，生理心理学の分野では特に必須のものである。またその他の分野でも見回してみると多くの研究室で，一台ぐらいはマイクロコンピュータが備え付けられている。その意味でも扱い慣れたマイクロコンピュータを用いてアナログデータの処理が可能となれば，高価な専用機よりも，むしろ安価な汎用機の方が現状に適していると言える。そこで現在日本の中では最も多く使用されているマイクロコンピュータの機種を用いたアナログデータの取り込み法について報告する。なお，この方法の開発の基本的な方針として，(1)汎用性が高く，ソフトウェア，ハードウェアとも大きな変更をせずに利用できること，(2)ソフトウェアは考えの流れが理解しやすいプログラムで，メインテナンスも容易であること，とした。

アナログデータをデジタルデータに変換して処理する方法の説明は，ハードウェアの面とソフトウェアの面に大きく分けられる。本報告では，理論的なものよりも，実際上使用する際に

必要とされる具体的なものについて，できるだけ詳細に記述した。なおソフトウェアについては，基本方針(2)に従って，PASCAL 言語を用いて当研究室でプログラムを作成し，現在実際に生理心理学の実験に使用している。

ハードウェア環境

動作環境は，PC-9801E（日本電気（株））以降の機種で可能であるが，メインメモリーが640 KB 以上あることが最低条件で，PC-9801VX 以降の CPU が80286以上の機種が望ましい。本報告では，PC-9801DX2 に装着した例を説明する。たとえ CPU が8086であってももちろん正常に動作はするが，データ転送速度が遅く，実用上はかなり制限される。クロック周波数の遅い機種での対処方法としては，RAM ディスクを設定して，そこに書き込ませるようにすればよい。またデータを書き込むディスクが，1 MB のフロッピィディスクの場合は，容量がわずかしかないので，ディスク交換等，注意を要する。その点，20 MB 以上の空き容量のあるハードディスクを接続している場合には心配はいらないだろう。またデータを書き込む動作速度の点でもハードディスクがある方がフロッピィディスクより格段に速く，推奨される。A/D 変換ボードとして使用したものは，カノープス電子製の ADX-98E である。このボードではシングル・エンド入力モードと差動入力モードの２つのモードが設定できるが，本報告においてはノイズの影響を受けにくく操作が容易な差動入力モードについて扱うものとした。ADX-98E には16本のアナログ入力端子があり，差動入力モードでは２本１組で１チャンネルの信号接続に使用できる。具体的にはアナログデータの取り込みは，BNC コネクターを持った 8CH 用ケーブル（HAB-1106-12, カノープス電子（株））を介して ADX-98E に接続した。

A/D 変換ボードについて

A/D 変換ボードの初期設定は，装着するマイコン本体にも依存するが，次の場合を想定した。PC-9801DX2 に装着して，入力電圧範囲を −5 V〜+5 V とし，変換結果の形式をコンプリメントバイナリ形式，割り込み機能を使用せず，拡張機能を使用する。その場合の設定は表 1. のようになる。またこの時の ADX-98E の I/O ポートのアドレスは，他のボードを使用しないのであればディップスイッチの DS1，DS2 によって 00D0h と設定する。図 1. に ADX-98E ボード上のディップスイッチのポートアドレス設定を示す。その際に ADX-98E が使用する I/O ポートの名称とポートアドレスの関係を表 2. に示す。ボード上の各設定を確認した後，PC-9801DX2 本体の裏にある４つの拡張スロットの内のどこかに差し込めばよい。

なお，このボードでは入力されたアナログ電圧をデジタル値に変換するのは，12ビットバイナリで行うので分解能は4096分の１（2^{12} 分の１）になる。さらに実際の電圧値の選択は，0 V〜+5 V, 0 V〜+10 V, −5 V〜+5 V, −10 V〜+10 V の４通りの選択が可能である。また変換したデジタルデータの表現形式において，オフセットバイナリ形式とコンプリメントバイナリ形式の２通りが可能であるので，アナログ入力信号と変換結果の対応関係は表 3. のように全部で８通りになり，この中から選択することとなる。本報告においては，コンプリメントバイナリ形式で変換範囲を− 5V〜+ 5V とした。具体的に対応する電圧値は表 4. のようになり，約 2.5 mV である。

このボードの最も大きな特徴は，A/D 変換結果をコンピュータ内部のメインメモリーに DMA (Direct Memory Access) 転送できることにある。DMA は CPU (Central Processing

Unit) の働きを介さずに入出力装置とメインメモリーの間で直接データを転送する方式で，これを用いると非常に高速でデータを転送できる。

表1.　ADX-98E の基板上のジャンパー線とディップスイッチ3の設定

基板上のジャンパースイッチの設定		左のジャンパースイッチに対応したディップスイッチ3 (DS3) の設定	
JP 1	10 V	DS 3-1	ON
JP 2	BIP	DS 3-2	ON
JP 3	8D	DS 3-4	ON
JP 4	8D		
JP 5	CMP	DS 3-3	ON
JP 6	8D		
JP 7	OTH	DS 3-5	ON
JP 8	OTH		
JP 9	DISI		
JP10	8D		
JP11	N	DS 3-6	ON
JP12	OUT	DS 3-7	OFF
JP13	IN	DS 3-8	OFF
JP14	×1		

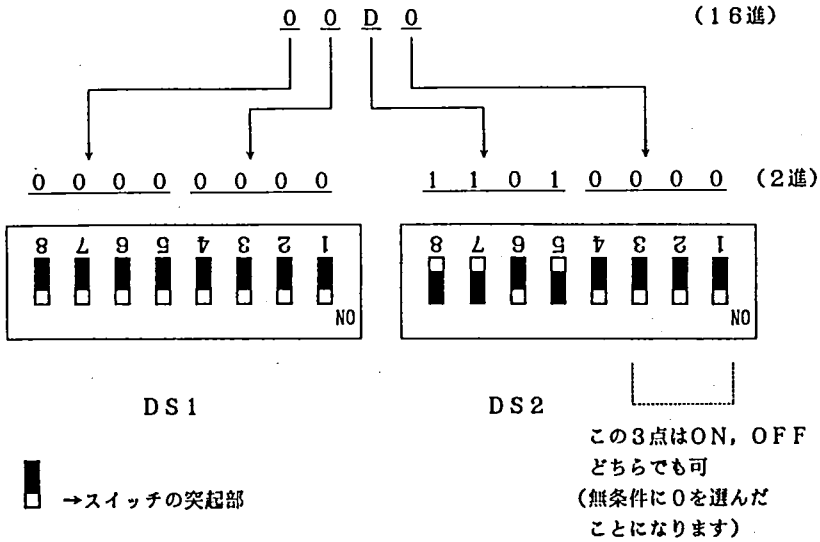

図1.　ADX-98E の基板上のポートアドレス設定のためのディップスイッチ1，2の設定

表2. I/O ポートアドレスを DS1 と DS2 で 00D0h と設定した時に，ADX-98E が使用する I/O ポートとそのアドレスの関係

ポート名称	READ/WRITE	PORT ADDRESS (16進数)
CONTL	WRITE	00D0h
BCSET	WRITE	00D1h
FREQSEL	WRITE	00D2h
MODE	WRITE	00D3h
SEQSEL	WRITE	00D4h
CHSEL	WRITE	00D6h
RDSTS	READ	00D0h
LBYTE	READ	00D2h
HBYTE	READ	00D4h
RDSW	READ	00D6h

表3. A/D 変換範囲の4通りとデータ表現形式2通りの組み合わせとそれに対応する変換結果

データ形式	変換範囲	対応する変換結果（2進表記）
OFB	−5 V～ +5 V	0000 0000 0000～1111 1111 1111
OFB	−10 V～+10 V	0000 0000 0000～1111 1111 1111
OFB	0 V～ +5 V	0000 0000 0000～1111 1111 1111
OFB	0 V～+10 V	0000 0000 0000～1111 1111 1111
CMP	−5 V～ +5 V	1000 0000 0000～0111 1111 1111
CMP	−10 V～+10 V	1000 0000 0000～0111 1111 1111
CMP	0 V～ +5 V	1000 0000 0000～0111 1111 1111
CMP	0 V～+10 V	1000 0000 0000～0111 1111 1111

ソフトウェア環境

ここで用いた PASCAL 言語は，ボーランド社製 TURBO PASCAL Version 6.0 を用いた。この開発言語は処理速度が速く，しかもビット操作，ファイル操作が容易である。PASCAL 言語は一般的に開発言語として使用されるC言語よりも文法的に厳格であり，コンパイルが正常にできない限り実行はできず，従ってプログラムの暴走の危険性も少ない。また，プログラムを作成あるいは修正する際のエディタの環境およびデバック環境も TURBO PASCAL の場合には統合開発環境設定により整備されている。これを使用すれば一度に多数のプログラムを読み込めたり，操作が判らない場合にはヘルプ機能によりオンラインで参照できるなど，プログラムのメインテナンスの面でも容易である。さらに，各研究室での実際の使用に合うようにプログラムを組むのに，一度作成したプログラムの中で手続きの共通の部分があれば，あたかも部品を組み合わせるように，様々なプログラムに応用できる。これらは現在パーソナルコンピューターの標準的な OS である MS-DOS 上で動作する。なお，本報告で例として使用する A/D 変換用プログラムを「ADCON」と名付けた。このプログラム内には A/D 変換ボードの

表4.　−5 V〜+5 V の範囲のアナログ入力信号と変換結果の対応

コードが遷移するアナログ入力電圧	変換結果 OFB	変換結果 CMP	コードの中心におけるアナログ入力電圧
	1111 1111 1111	0111 1111 1111	+4.99756 V
+4.99634 V			
	1111 1111 1110	0111 1111 1110	+4.99512 V
+4.99390 V			
	1111 1111 1101	0111 1111 1101	+4.99268 V
+4.99146 V			
	⋮	⋮	
+0.00366 V			
	1000 0000 0001	0000 0000 0001	+0.00244 V
+0.00122 V			
	1000 0000 0000	0000 0000 0000	0.00000 V
−0.00122 V			
	0111 1111 1111	1111 1111 1111	+0.00244 V
−0.00366 V			
	⋮	⋮	
−4.99390 V			
	0000 0000 0010	1000 0000 0010	−4.99512 V
−4.99634 V			
	0000 0000 0001	1000 0000 0001	−4.99756 V
−4.99878 V			
	0000 0000 0000	1000 0000 0000	−5.00000 V

命令用に「DMAMLT8」と名付けたコントロールファイルが埋め込まれている。「DMAMLT8」については2バイトの命令が16進数で書かれている。これはアナログデータの取り込み開始時点とそのタイミングについてだけ注意すればよく，その他は意識する必要はない。サンプリングクロック周波数コードの表を表5. に，また，その全ソースリストは APPENDIX として最後に示す。実際に使用する際には，作成したプログラムをコンパイルして実行型のファイルに落としておき，MS-DOS のプロンプトが表示されているコマンド待ち状態から「ADCON」と入力するだけでプログラムが作動するものとした。従って本システムの利用者はプログラムの内容を完全に理解していなくても，プログラムを意識することなく使用できるものとした。「ADCON」の基本仕様は 8ch 同時取り込みで，サンプリングクロック周波数が 1 KHz，つまり1ポイント＝1 msec で，1波形512点，512 msec 取り込みとした。それが連続して100回繰り返される。データの取り込み順序は，ケーブルについている番号 0 〜 7 の順である。プログラムを開始する前提条件として，B ドライブに，「¥SUB??」と名付けたディレクトリと，その下位に「CH0」〜「CH7」のサブディレクトリを予め作成しておく。また，取り込み波形の表示も可能であるが，ソフトウェア環境として，波形表示をさせるためにグラフィック画面を使用しているので，TURBO PASCAL に付属している PC98. BGI を A ドライブのルートディレクトリ上に配置しておかなければならない。

A/D 変換について

ここでは12ビットバイナリによりアナログ入力信号をデジタルデータに変換しているので，16ビットを一度に送るワード型転送でない限り，8ビットずつ2回データを転送することになる。PC-9801DX2 でおこなう DMA ベースアドレス設定や転送バイト数は16ビット指定であるが，そのコントローラである 8237A に書き込むのは1度に8ビットとなる。そこで実際に

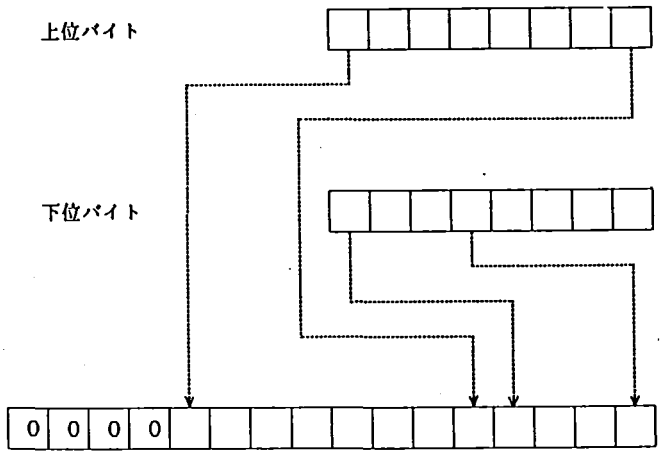

図2.　A/D 変換結果の読み込み

は下位8ビット，上位8ビットを分けて書き込むことになるが，この時，同じポートに続けて2バイト書き込むと，初めのものが下位，次が上位と判定される。16ビットの所にデータとしては12ビット転送することになるので4ビットのずれが生じる。実際には図2. のような型でデータを転送しなければならないので，全体を右に4ビットシフトするために16 (2^4) で除する必要がある。また，8237A の転送バイト数のカウンタは16ビットなので，連続して転送できるデータ数は最大 64 K ($2^{16}=65536$) バイトとなる。1回の A/D 変換によって2バイトのデータが発生するので，これは32768点の変換に相当する。さらに8チャンネル同時取り込みを行うと，1チャンネル当たりの点数は32768/8＝4096となる。もっと大量のデータを連続して DMA 転送したい場合には，ADX-98E のバンクカウント機能を利用すればよいが，その説明は割愛する。

「ADCON」プログラム操作

MS-DOS の起動している状態から，「ADCON」と入力すると画面上に，このプログラムの取り込み設定条件と，「リターンキーで処理を始めます」とメッセージが表示される。取り込みを開始したい時点でリターンキーを押すと，画面上に「Sampling start !!」と表示され，約1秒毎に100回，「Sampling (No. 1)」から順に「Sampling (No. 100)」まで表示される。この間，自動インクリメント機能により「Sampling (No. xx)」が表示されるたびに，0チャンネルから7チャンネルまでに順に512回アナログデータがデジタルデータに変換されて，コンピュータ内部のメインメモリーに直接転送されている。ちなみに，A/D 変換の項で述べたように，1ポイントにつき2バイトのデータが発生するので，512ポイント取り込みにより，1波形（1チャンネルについて1回の取り込み）では1024バイト消費する。そして，そのデータは，サンプリングの番号が増えるたびにその前のデータに連続して取り込まれていく。最終的に100回取り込みを終えたところで，Aドライブのルートディレクトリ上に “WAVE. DAT” の名前でいったん保存される。その消費バイト数は，512ポイント×8チャンネル×100回×2バイト＝819200バイトである。その後，このプログラムでは，あらかじめ作成しておいた，被験者番号のついたサブディレクトリにファイルを保存するために，被験者番号の入力を求める表示が出る。つまり，画面上に「INPUT Subject No.??＝」と表示されるので，番号を入力する。この操

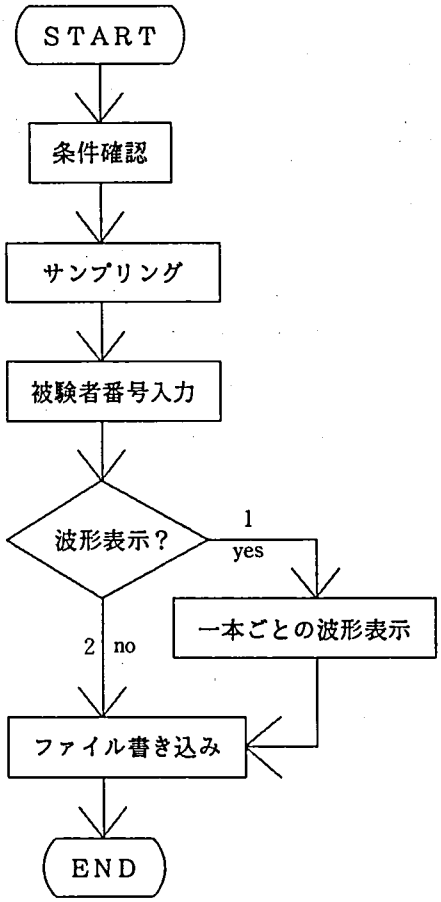

図3.　A/D 変換プログラム「ADCON」のフローチャート

作は，各チャンネル毎，各サンプリング毎に番号をつけて，別のファイルとして保存する作業のためにサブディレクトリを指示してやることである。次に，「波形を表示しますか？」と，質問が出るので，取り込みの波形表示をしたいときは1を選択し，波形表示が必要ないときは2を選択する。1を選択すると，チャンネル0の1回目のサンプリング波形から，次々に100回目までディスプレイ上に波形が表示される。波形は，電位0の緑色の直線が画面中央に水平に表示された後，左から右に描かれていく。チャンネル0は緑色で，チャンネル1は水色で，以下，赤，紫，黄，白，青，緑の順で描かれていく。波形を表示しない2を選択すればファイル名のみが画面上に表示され，ファイルへの書き込みがなされて終了する。

使用結果と考察

図3. に「ADCON」プログラムのフローチャートを示した。AドライブにMS-DOSシステムと「ADCON. EXE」，「PX98. BGI」ファイルの入ったフロッピーディスクを準備し，Bドライブに指定されたサブディレクトリを予め作っておいたフロッピーディスクを準備した。このときどちらのフロッピーディスクも空き容量が819200バイト以上あることを確認した。次に取り込み条件の確認をしてリターンキーを押した。アナログデータの取り込みが開始され，画面

上には何回目の取り込みが行われているか表示された。この間の所用時間は約1分33秒であった。次に被験者番号の入力を求められたので，予め作成しておいたディレクトリ「¥SUB01」の「01」を入力した。波形表示の有無を尋ねられたので，「表示する—＞1」を選択したところ，画面中央左半分に取り込み波形が表示され，Bドライブのディスクに保存された。8チャンネル各100回分の波形を表示して保存するのに約26分要した。また，同じことを最初からもう一度実行して，波形表示の有無の所で，「表示しない—＞2」を選択したところ，画面上には保存されるファイル名が表示されるのみで，波形は表示されず，所用時間約10分で終了した。この結果から，フロッピーディスクを使用した場合のデータのファイル書き込み操作だけで，約10分かかることになる。いちいち取り込まれる波形のチェックが必要ないのであれば，一度動作の確認だけしておいて，あとは2を選択して，波形表示をさせない方が作業時間が短縮される。ハードディスクに書き込んだ場合を試していないが，フロッピーディスクよりも速くなることは確実である。またRAMディスクを使用すれば，そのスピードは格段に向上すると考えられる。

本報告では，1 msecで512ポイント取り込みの例について説明した。もちろんプログラム上で，表5.を参照してサンプリングクロックの設定を変更すれば，任意のサンプリングクロックで512ポイント取り込み可能である。また，ポイント数においても，プログラム上のDLNの定数を現在512としているものを1024とすれば，1024ポイントの取り込みも可能である。ただ，その時注意することは，ハードウェアの面で使用しているコンピュータのメモリーの上限を越えないように設定しなければならないことである。本プログラムの次の段階として，サンプリングクロックと取り込みデータポイント数については，使用者が任意に入力できる入力画面を加えたいと考える。

表5.　サンプリングクロック周波数コード数

コード	周波数	コード	周波数	コード	周波数	コード	周波数
00	(1 M)	10	10 k	20	100	30	1
01	100 k	11	1 k	21	10	31	1/10
02	(500 k)	12	5 k	22	50	32	1/2
03	(333.3 k)	13	3.3 k	23	33.3	33	1/3
04	(250 k)	14	2.5 k	24	25	34	1/4
05	200 k	15	2 k	25	20	35	1/5
06	166.6 k	16	1.6 k	26	16.6	36	1/6
07	83.3 k	17	833.3	27	8.3	37	1/12
08	100 k	18	1 k	28	10	38	1/10
09	10 k	19	100	29	1	39	1/100
0A	50 k	1A	500	2A	5	3A	1/20
0B	33.3 k	1B	333.3	2B	3.3	3B	1/30
0C	25 k	1C	250	2C	2.5	3C	1/40
0D	20 k	1D	200	2D	2	3D	1/50
0E	16.6 k	1E	166.6	2E	1.6	3E	1/60
0F	8.3 k	1F	83.3	2F	0.83	3F	1/20

周波数の単位はHz

注1)　たとえば16.6は16.666……を表わします。

注2)　(　)内の周波数は使用できません。

謝　辞

本報告のプログラムの作成には教室の頃末敦夫君，多久和学君の協力を得た。記して感謝の意を表します。

参 考 文 献

カノープス電子（株） 1990 ADX-98E ユーザーズガイド

塩谷孝夫 1990 BASIC による高速・大容量データサンプリングプログラムの開発．日本生理学雑誌，52, 345-354.

古川原誠 1990 パーソナルコンピュータを用いたインパルスデータ収集分析システム．日本生理学雑誌，52, 385-392.

APPENDIX A 「ADCON」プログラムの全ソースリスト

```
program AD_CONVERT;
{$R+}
uses   DMAMLT8,dos,printer,graph,crt;

const  ch_num = 8   ;
       DLN    = 512 ;
       Times  = 100 ;
type
    SMPtype   = array[1..DLN,1..ch_num] of integer;
    save_type = array[1..DLN] of integer ;
var
    SMP              : SMPtype absolute $8c00:$0;
    save_data        : save_type        ;
    Fv_Smp           : file of SMPtype  ;
    FF               : file of save_type ;
    i,x,x1,x2,y1,y2,y0,tr,trial,disp_on,p: integer;
    col,ch,jj,ct,ch_set,trg_max,ctl_max  : integer;
    BANK , DBA       : word             ;
    FID,strct        : string[80]       ;
    subname          : string[2]        ;
    channel          : string[1]        ;
    rk               : char             ;

{==================  sampling  ======================}
procedure sampling;
  begin
    TextColor(Yellow)   ;
    gotoxy(2,8)         ;
    writeln('Aドライブと Bドライブの空き容量が 819200 ﾊﾞｲﾄ以上ありますか？');
    gotoxy(2,10)        ;
    writeln('Bドライブに¥SUBxx¥CH0 ～ ¥CH7のディレクトリが作ってありますか？');
    gotoxy(18,12)       ;
    writeln('確認したら、リターンキーを押して下さい');
    readln              ;
    assign( Fv_Smp , 'A:¥wave.dat' );  { このドライブがＲＡＭディスクだと高速 }
    rewrite(Fv_Smp)     ;
    clrscr              ;
    gotoxy(15,8)        ;
    writeln('８チャンネル　アナログデータ　取り込みプログラム')        ;
    gotoxy(13,10)       ;
    writeln('設定値は　１ポイント　１ msec　　512 ポイント取り込み')   ;
    gotoxy(18,12)       ;
    writeln('１００回繰り返して　１００波形取り込みます')             ;
    writeln;
    TextColor(White)    ;
    writeln('                        リターンキーで処理を始めます')      ;
```

```
    readln; writeln('Sampling start !!');
    for tr:=1 to Times do
    begin
      DMAcheck( Seg(sMP) , Ofs(sMP) , DLN , ch_num , BANK , DBA )      ;
      Norm8237a( BANK , DBA , DLN , ch_num )                           ;
      ADX98setup( $11 , $FF );  { $11 = 1000 Hz : 1 point = 1 msec.  表 5. 参照 }
      StartConversion;
      repeat until RDSTS(DMAMODE) =0 ;
      write(Fv_Smp,Smp);
      writeln('Sampling ( No.',tr:3,' )');
    end;
    close(Fv_Smp);
  end;

procedure clear_data;
  var  ch,i: integer;
  begin
    for ch:=1 to ch_num do begin
     for i:=1 to DLN do begin
       SMP[i,ch]:=0; save_data[i]:=0;
    end; end;
  end;

procedure Parameters;   { ユーザーが，パラメーターを入力します }
  var  sure : char;
 begin
   repeat;
    TextColor(Yellow);
    writeln('Bドライブに¥SUBxx¥ch0 ～ ch7のディレクトリを予め作った空き容量');
    writeln('819200 KB 以上のデータディスクを挿入して下さい。');writeln;
    writeln('保存されるファイル名はたとえば、¥SUB01¥CH0¥CH0-1.0 となります');
    writeln; TextColor(White);
    write(' INPUT Subject No.?? =  ');readln(subname);
    write(' 波形を表示しますか? 表示する-->1  表示しない-->2'); readln(disp_on);
    writeln; write('よろしいですか？ (y/n) ');readln(sure);
   until Upcase(sure)='Y';
 end;

procedure Wave_Save;      { WAVE DATAをSAVEします }
  var  ch,m,p      : integer  ;
       sample_num  : string[3];
  begin
      Str( ch_set-1, channel ); Str( Trial, sample_num );
      FID := 'B:¥SUB'+subname+'¥CH'+channel+
             '¥CH'+channel+'-'+sample_num+ '.'+ channel ;
      writeln('出力ファイル名  -> ',FID );
      assign( FF , FID );
      rewrite( FF );
      write( FF , save_data);
      close( FF );
  end; { of Wave_Save }
```

```
{================== Graphic ==========================}
procedure Graph_Open;
var   GraphDriver, GraphMode, MaxColor : integer;
    begin
      DetectGraph( GraphDriver, GraphMode )     ;
      InitGraph( GraphDriver, GraphMode,' ')    ;
    end;

function posX : longint;
  begin
    posX:=Round((640/(DLN*2))*x) ;
  end;

function posY : longint;
  begin
    case Smp[x,ch]>=0 of
         true : posY:=Round(200-(300/640)*save_data[x]);
         false: posY:=Round(200+abs((300/640)*save_data[x]));
    end;
  end;

procedure graph_screen;
  begin
      clrscr;
      Graph_Open;
      x1:=0; x2:=639 ; y1:=0 ; y2:=399;
      SetViewPort(x1,y1,x2,y2,ClipOn);
      x:=511;y0:=200; SetColor(green);
      MoveTo(0,y0);LineTo(posX,y0); Setcolor(White);
      for ch:=ch_set to ch_set do
      begin
        SetColor((ch mod 7)+1);
        for x:=1 to DLN do begin
          save_data[x]:=Smp[x,ch] div $10;
          if x=1 then MoveTO(posX,posY) else LineTO(posX,posY);
          end;
      CloseGraph;
      end;
 end;

procedure data_trans;
  begin
      for ch:=ch_set to ch_set do begin
        for x:=1 to DLN do
          save_data[x]:=Smp[x,ch] div $10;
      end;
  end;
```

```
(************* main ***************)
begin
  clrscr;
  clear_data;
  sampling;
  clrscr;
  textreverse(noreverse);
  Parameters;
  for ch_set:=1 to ch_num do
  begin
    writeln; writeln('◆ファイル SAVE'); writeln;
      for trial:=1 to Times do
        begin
        case disp_on of
          1 : Graph_Screen;
          2 : data_trans  ;
          end; { of case disp_on }
        Wave_Save;
        end;
  end;
  writeln; TextColor(Green);
  writeln('アナログデータのデジタルデータへの変換および取り込みは無事終了しました
');
end.
```

APPENDIX B 「DMAMLT8」ファイルの全ソースリスト

```
{-------------------------------------------------------------------------------
  多チャンネル多点変換  for ADX-98E DMA ( USERS GUIDE p.64 )
  'DMAMLT8.TPU' インクルードファイル                          By S. Sakata
 -------------------------------------------------------------------------------}
unit DMAMLT8;
interface
type  flag_RDSTS =( DATARDY,OVERRUN,DMAMODE,DMAORUN,WAITTRIG,CHNSEQ,AUX1,AUX2 );
      flag_LBYTE =( CD3 , CD2 , CD1 , CD0 , DGI3 , DGI2 , DGI1 , DGI0 )       ;

Function  RDSTS( flagname : flag_RDSTS ) : byte;      { DMAの状態読み込み   }
Function  LBYTE( flagname : flag_LBYTE ) : byte;      { ディジタル入力読み込み }
Function  DMAcount0 : boolean ;{ DMA ｺﾝﾄﾛｰﾗｰ 8237A の3ﾁｬﾝﾈﾙの残りﾊﾞｲﾄ数のﾁｪｯｸ }
Function  CurrentAddress  : integer      ;        { カレントアドレスの読みだし }
procedure DMAcheck( Segment, Offset, DLN, CHNUM  : word; var Bank, DBA : word );
procedure Norm8237A ( BANK,DBA,DLN,CHNUM : word );  { for DMA controller 8237A }
procedure Pre8237A  ( BANK,DBA,DLN,CHNUM : word );      { 8237A for Pretrigger }
procedure ADX98setup( FREQ,CHSEQ : word );     { ｻﾝﾌﾟﾘﾝｸﾞｸﾛｯｸ、取り込みﾎﾟｲﾝﾄ数 }
procedure StartConversion;                  { トリガを入れずに連続して取り込み }
procedure PostTrigger  ;  { DMAｺﾝﾄﾛｰﾗｰ8267Aは、ノーマル仕様でお使い下さい   }
procedure PreTrigger   ;  { DMAｺﾝﾄﾛｰﾗｰ8267Aは、プレトリガ仕様でお使い下さい }

implementation
procedure DMAcheck( Segment, Offset, DLN, CHNUM : word ; var Bank, DBA : word );
  var SegmentA, PhysicA, Errorcheck : longint   ;
  begin
      PhysicA:=0;
      SegmentA:=Segment;
      PhysicA:=SegmentA*16+Offset;
      Bank:=(PhysicA shr 16) and $000F;
      DBA:=PhysicA and $FFFF;
      Errorcheck := DBA + DLN * CHNUM * 2 - 1 ;
      if Errorcheck >= 65536 then writeln(' 64K SECTION ERROR!!! ');
  end { of DMAcheck }  ;

      { トリガのない状態あるいはポストトリガを用いる設定}
procedure Norm8237A( BANK,DBA,DLN,CHNUM : word );   { for DMA controller 8237A }
  var BCOUNT    : integer;
  begin
     port[$0025]:=Bank                              ;{ Set DMA bank }
  { Set up 8237A }
     port[$0015]:=$07                               ;{ mask DMA channel 3  }
     port[$0017]:=$47                               ;{ mode register       }
     port[$0019]:=$00                               ;{ clear first/last FF }
     port[$000D]:=Lo(DBA)                           ;{ address set  Lo     }
     port[$000D]:=Hi(DBA)                           ;{ address set  Hi     }
     BCOUNT := DLN * CHNUM * 2 - 1                  ;
     port[$000F]:=Lo(BCOUNT)                        ;{ byte count set Lo   }
     port[$000F]:=Hi(BCOUNT)                        ;{ byte count set Hi   }
  end { of DMAsetup }  ;
```

```
      {プレトリガを用いる設定}
procedure Pre8237A( BANK, DBA, DLN, CHNUM : word );     { 8237A for Pretrigger }
  var BCOUNT    : integer    ;
  begin
     port[$0025]:=Bank                              ; { Set DMA bank }
  { Set up 8237A }
     port[$0015]:=$07                               ;{ mask DMA channel 3  }
     port[$0017]:=$57                               ;{ ｵｰﾄｲﾆｼｬﾗｲｽﾞ付       }
     port[$0019]:=$00                               ;{ clear first/last FF }
     port[$000D]:=Lo(DBA)                           ;{ address set  Lo     }
     port[$000D]:=Hi(DBA)                           ;{ address set  Hi     }
     BCOUNT := DLN * CHNUM * 2 - 1                  ;
     port[$000F]:=Lo(BCOUNT)                        ;{ byte count set Lo   }
     port[$000F]:=Hi(BCOUNT)                        ;{ byte count set Hi   }
  end { of 8237ASet }  ;

procedure ADX98setup( FREQ,CHSEQ : word );     { ｻﾝﾌﾟﾘﾝｸﾞｸﾛｯｸ、取り込みﾎﾟｲﾝﾄ数 }
  begin
     port[$00D2]:=FREQ and $3F ;  { set sampling frequency }
     port[$00D4]:=CHSEQ        ;  { set channel sequence   }
  end { of ADX98setup }        ;

procedure StartConversion;                  { トリガを入れずに連続して取り込み }
  begin
     port[$0015]:=$03          ;  { reset mask }
     port[$00D0]:=$CB          ;  { set control word $CB='11001011' }
  end { of StartConversion }   ;

procedure PostTrigger;    { DMAｺﾝﾄﾛｰﾗｰ-8267Aは、ノーマル仕様でお使い下さい   }
  begin
     port[$0015]:=$03          ;  { reset mask }
     port[$00D0]:=$DB          ;  { set control word $DB='11011011' }
  end { of PostTrigger }       ;

procedure PreTrigger ;    { DMAｺﾝﾄﾛｰﾗｰ-8267Aは、プレトリガ仕様でお使い下さい }
  begin
     port[$0015]:=$03          ;  { reset mask - ch.3 }
     port[$00D0]:=$FB          ;  { set control word $FB='11111011' }
  end { of pretrigger }        ;

Function DMAcount0 : boolean ; { DMA ｺﾝﾄﾛｰﾗｰ 8237A の3ﾁｬﾝﾈﾙの残りﾊﾞｲﾄ数のﾁｪｯｸ }
  var CCH , CCL : byte ;
  begin
    CCL := port[$000F] ;
    CCH := port[$000F] ;
    if ( CCL <> $FF ) or ( CCH <> $FF )
       then DMAcount0  := false
       else DMAcount0  := true ;
  end { of DMAcount0 } ;
```

```
Function  CurrentAddress : integer ;              { カレントアドレスの読みだし }
  var CCH , CCL : byte   ;
  begin
    CCL := port[$000D]   ;
    CCH := port[$000D]   ;
    CurrentAddress:=CCH*256+CCL+1  ;
  end ;

      {   Analog-PRO DMA  I／Oポートのアクセス
          Analog-PRO DMA  の各ポートをアクセスします。}
Function  RDSTS( flagname : flag_RDSTS ) : byte;      { DMAの状態読み込み   }
   var     st , Bit : byte;
   begin
     st := port[$D0];
     case flagname of
       DATARDY  : Bit := st and $80 ;
       OVERRUN  : Bit := st and $40 ;
       DMAMODE  : Bit := st and $20 ;
       DMAORUN  : Bit := st and $10 ;
       WAITTRIG : Bit := st and $08 ;
       CHNSEQ   : Bit := st and $04 ;
       AUX1     : Bit := st and $02 ;
       AUX2     : Bit := st and $01 ;
     end;
     RDSTS := Bit     ;
   end { of RDSTS }   ;

Function  LBYTE( flagname : flag_LBYTE ) : byte;      { ディジタル入力読み込み }
   var      st, Bit   : byte        ;
   begin
     st := port[$D2]  ;
     case flagname of
       CD3      : Bit := st and $80 ;
       CD2      : Bit := st and $40 ;
       CD1      : Bit := st and $20 ;
       CD0      : Bit := st and $10 ;
       DGI3     : Bit := st and $08 ;
       DGI2     : Bit := st and $04 ;
       DGI1     : Bit := st and $02 ;
       DGI0     : Bit := st and $01 ;
   end;
      LBYTE := Bit    ;
   end { of LBYTE }   ;
end.
```