１９．データテーブル

制御プログラムにおけるＣＰＵやＤＳＰの演算ではＳＩＮ関数あるいは平方根などの計算を行う必要がある。このような計算はマシン語でも作成できるが非常に複雑となり，また演算時間が長くなるために利用できない。そこで「データテーブル（Data Table）」と呼ばれるデータ表をメモリ上に作成し，その領域にあらかじめ BASIC 等で作成したデータを保存し，そこから必要なデータを読み出す方法が用いられる。

簡単な例として，データテーブルを利用して「０から１０までの２乗値」を求めるが，データ量が大きくなっても考え方は同じである。まず，０から１０までの２乗値はあらかじめ計算できるので，それらの値を図のようにメモリ上に順次並べて作成する（ＣＰＵでは「ＤＳ」内，ＤＳＰではプログラムメモリ領域）。いま，その先頭アドレスを「SDAT」とすると，そこからのオフセットのアドレスに対する２乗値がそれぞれ保存されたテーブルとなっている。したがって，オフセットの値に対するアドレスのデータを読み込めば解が求められることになる。ＣＰＵでは次のようにプログラミングする（AX レジスタ内容の２乗を求める）。

ｱﾄﾞﾚｽ	ﾃﾞｰﾀ
+10	100
+ 9	81
+ 8	64
+ 7	47
+ 6	36
+ 5	25
+ 4	16
+ 3	9
+ 2	4
+ 1	1
SDAT+ 0	0

```
MOV  BX,OFFSET DGROUP:SDAT
ADD  BX,AX
MOV  AX,[BX]
```

あるいは

```
MOV  BX,AX
MOV  SI,OFFSET DGROUP:SDAT
MOV  AX,[SI+BX]
```

このテーブルを利用すれば，乗算命令を実行せずに２乗値が求められる。また，ＤＳＰではプログラムメモリの「2000 番地」からこのテーブルを作成すれば，次のプログラミングで求められる（ACC の２乗を求め，結果を変数「RES」に保存）。

```
ADLK  $2000
TBLR  RES
```

実数値や平方根などの場合には，１５章で説明した固定小数点表現によってデータテーブルを作成する。また，どの程度の分解能のデータが必要であるか，例えばＳＩＮの値は１度刻みか 0.5 度刻みか，あるいは平方根では 0.1 か 0.05 であるかなどを決定してテーブルを作成する。

このようにデータテーブルはある決まった計算を行う場合，１対１に対応した値が得られる時に非常に有効な手段である。演算はテーブル上の結果が保存してあるオフセットアドレスを求めるだけであり，実際の演算は行わないため，処理時間の大幅な短縮が可能となる。ただし，テーブルのためのメモリが必要であり，分解能が高いほどデータテーブル

メモリ領域は大きくなる。データテーブル作成のポイントは，演算するデータの値とテーブルのオフセットアドレスをどのように対応させるかである。

通常，データテーブルはかなりのメモリサイズとなるので，テーブル内のデータはBASICによって作成し，ディスクに保存してプログラム実行時にローディングする。ＣＰＵに対するデータテーブルとＤＳＰに対するデータテーブルの形式は同じであるが，メモリへのローディング方法が異なるためにBASICでのデータ作成方法が異なる。ＣＰＵのデータテーブルは通常の「バイナリイメージデータ」であるが，ＤＳＰではテーブルデータのローディングを「LD25.EXE」のアプリケーションプログラムで行うため，これに対応した「テキストイメージデータ」としてディスクに保存する。

以下，それぞれについて，サンプルとして「0.0～100.0 までの平方根値（0.1 単位）」を求めるプログラムを示す。

まず，テーブルのアドレス範囲は入力データが「0.0～100.0」であるので，それに対応するアドレスを「0 番地～1000 番地」とする。データを１６ビット（１ワード）で作成すると，求める値は「0.0～10.0」であるから，最大分解能で整数部４ビット，小数部１２ビットの固定小数点表現で表せる。すなわち，データは

０が 0000.0000 0000 0000

１０が 1010.0000 0000 0000

となる。１０は実際の「２ ｀倍＝４０９６倍」に対応する。したがって，得られる値は

$$\frac{1}{2^{12}} = 0.000244$$

の精度となる。データテーブルの大きさは「1ﾜｰﾄﾞ×1000=1000ﾜｰﾄﾞ=2000ﾊﾞｲﾄ」となる。

ＣＰＵに対応するテーブルのデータを作成するBASICプログラムが「EX191.BAS」である。１６ビットデータを計算し，上位および下位８ビットのデータを求め，「POKE」文によりメモリに書き込み，「BSAVE」によってドライブＢのサブディレクトリ「BINARY」にファイル名「EX191.TBL」として保存する（保存データのサイズは2003バイトとなる）。

```
100 'EX191.BAS
110  DEF SEG=&H8000
120   AD=0
130  FOR I=0 TO 1000
140   DD=INT(SQR(I/10)*2^12)
150   DH=INT(DD/256)
160   DL=DD-DH*256
170    POKE AD,DL: POKE AD+1,DH
180    AD=AD+2
190  NEXT I
200   BSAVE "B:\BINARY\EX191.TBL",0,AD
210  END
```

Ｃプログラム「EX191.C」ではデータテーブルの先頭アドレス「tbl1」をグローバル変数として「char」定義する。「open」および「read」文によりディスクから順次データを読み込み，ＡＳＭプログラムのデータ領域にロードする。入力データは小数点第１位まで考慮するために，１０倍してＡＳＭプログラムへデータを引き渡す。ＡＳＭプログラムでテーブルから得られた値は実際値の「4096 倍」となっているので，Ｃプログラムではその値を「4096」で除算し，実数形式で値を表示する。た

だし，テーブル内のデータは「32767」を越える値があり，これは「int」定義では負の数を表すために，「unsigned int」によって正の数として実数値に変換する。

ＡＳＭプログラム「EX191A.ASM」では，データセグメント内で，各変数を「DW」でワード定義する。データテーブルはＣプログラムで「char」宣言したので，「DB」のバイト宣言し，「2010 バイト」を確保し「dup(0)」で内容を「０」に初期化する。メインプログラムでは「SI」レジスタにデータテーブルの先頭アドレスを格納し，「BX」レジスタをそのオフセットとしてデータ転送を行い，所望のデータを読み出す。「BX」レジスタはオフセットであるが，データテーブルの各データを１６ビットで作成しているため，物理アドレスではその２倍のオフセットとなるので，「SHL BX,1」を実行して２倍を得ている。

```
/* EX191.C */
#include <io.h>
#include <fcntl.h>
extern char tbl1;
extern int a,b;

main()
{
int fd;
char *ptbl1;
float d;

/* - DATA TABLE LOADING - */
 ptbl1=&tbl1;
  fd=open("B:\\BINARY\\EX191.TBL",O_RDONLY¦O_BINARY);
  read(fd,ptbl1,2010);
  close(fd);

printf("input=");scanf("%f",&d);
a=(int)(d*10.0);
rint();
d=(unsigned int)(b)/4096.0;
printf("answer=%f\n\n",d);
}
```

```
;EX191A.ASM
DGROUP   group  mdata
mdata    segment word public 'data'
         assume  ds:DGROUP
         public  _tbl1
         public  _a,_b
_a       DW  ?
_b       DW  ?
;* DATA TABLE *
_tbl1    DB  2010 dup(0)
mdata    ends

_TEXT    segment byte public 'CODE'
         assume  cs:_TEXT,ds:DGROUP
         public  _rint
_rint    proc near
         MOV  BX,_a
         SHL  BX,1
         MOV  SI,OFFSET DGROUP:_tbl1
         MOV  AX,[BX+SI]
         MOV  _b,AX
         RET
_rint    endp
_TEXT    ends
         end
```

ＤＳＰによってデータテーブルを利用する場合には次のようにプログラミングする。BASIC プログラム「EX192.BAS」のデータ作成は４章の「EX47.BAS」と同様であり，ＤＳＰプログラムメモリ（共有メモリ）へのローディングは「LD25.EXE」を用いて行う。データテーブルはＤＳＰオフセットアドレスの「2000H 番地」から配置するため，その先頭オフセットアドレスをＣＰＵのオフセットアドレスとして「4000H」をデータファイルの最初に記述し，そのデータに続いて平方根

```
100 'EX192.BAS
110  OPEN "B:\DFILE\EX192T.0J" FOR OUTPUT AS #1
120   STA$="4000"
130    PRINT #1,STA$;
140  FOR I=0 TO 1000
150   DA=INT(SQR(I/10)*2^12)
160   DD$=RIGHT$("000"+HEX$(DA),4)
170    PRINT #1,DD$;
180  NEXT I
190   CLOSE
200  END
```

データを作成する。テーブルファイル「EX192T.OJ」の大きさは「4009バイト」であり，これをドライブBのサブディレクトリ「DFILE」に作成する。

Cソースプログラム「EX192.C」では「LD25.EXE」（６章参照）を子プロセスとして実行し，共有メモリへＤＳＰプログラムのリロケータブルファイルとテーブルデータをローディングし，データ入力語にＡＳMサブルーチンをコールする。

```
/* EX192.C */
#include <process.h>
extern int a,b;

main()
{
float d;
/* - DSP Prog. LOADING - */
outp(0x02d0,0);
outp(0x02d2,(inp(0x02d2) & 0xc0));
outp(0x02d2,(inp(0x02d2) | 0x40));
 spawnlp(P_WAIT,"d:ld25","d:ld25",
        "b:\\dfile\\ex192.oj","-m","c000",NULL);
/* - DATA TABLE LOADING - */
spawnlp(P_WAIT,"d:ld25","d:ld25",
        "b:\\dfile\\ex192t.oj","-m","c000",NULL);

printf("input=");scanf("%f",&d);
a=(int)(d*10.0);
rdint();
d=(unsigned int)(b)/4096.0;
printf(" answer=%f\n\n",d);
}
```

```
;EX192A.ASM
DGROUP  group  mdata
mdata   segment word public 'data'
        assume  ds:DGROUP
        public  _a,_b
_a      DW  ?
_b      DW  ?
mdata   ends

_TEXT   segment byte public 'CODE'
        assume  cs:_TEXT,ds:DGROUP
        public  _rdint
_rdint  proc near
; DSP Start
        MOV  DX,02D2H
        IN   AL,DX
        AND  AL,0BFH
        OR   AL,01H
        OUT  DX,AL
; DPR Output
        MOV  02D4H
        MOV  AX,_a
        OUT  DX,AX
; DPR Check and Data Input
        MOV  DX,02D2H
DWAIT:  IN   AL,DX
        AND  AL,10H
        JZ   DWAIT
        MOV  DX,02D4H
        IN   AX,DX
        MOV  _b,AX
; DSP Stop
        MOV  DX,02D2H
        IN   AL,DX
        OR   AL,40H
        AND  AL,0FEH
        OUT  DX,AL
        RET
_rdint  endp
_TEXT   ends
        end
```

ＡＳMソースプログラム「EX192A.ASM」は基本的に先の「EX181A.ASM」と同じであるが，ここではＤＳＰとの間で引き渡すデータが１ワード１つであるので，ＤＰＲによって行う。したがって，ＡＳMでは共有メモリへのデータ転送は必要ない。プログラムではまず，ＤＳＰをスタートし，ＤＰＲへデータを出力する。その後，ＤＳＰでの処理を待ち，ＤＳＰからのデータをチェックして，ＤＰＲから結果のデータを取り込み，終了する。

ＤＳＰプログラム「EX192.A32」では，ＤＰＲのデータをチェックし，ＣＰＵから送られたデータをＤＰＲから取り込む。その後，必要なデータをテーブルから参照し，これをＤＰＲへ出力する。

これらのプログラムではデータを入力する時に，毎回プログラムを実行し，ＤＳＰプログラムのローディングやＤＳＰのスタート，ストップを行わなければならない。

```
;EX192.A32
        ORG  0
        B    MPRO
        RET

        DSEG 4
         BUFF: BSS 1
         VA:   BSS 1
         VB:   BSS 1
        DEND

        BSS   47

MPRO:   DINT
        LDPK 4
;DPR DATA CHECK AND INPUT
        LACK 1
WAIT:   IN   BUFF,PA0
        AND  BUFF
        BZ   WAIT
        IN   VA,PA1
;TABLE REFERENCE
        LAC  VA
        ADLK $2000
        TBLR  VB
;DPR OUTPUT
        OUT  VB,PA1
LOP:    B    LOP
        END
```

次に示すC，ＡＳMおよびＤＳＰプログラム（「EX193.C」，「EX193A.ASM」および「EX193.A32」）は連続してデータの入力が行えるように，修正したものである。なお，プログラムは「負値」を入力して終了する。このプログラムはでは，ＡＳMは単にＤＰＲとのデータの入出力を行うだけであるので，この操作ををＣプログラムで行えば，ＡＳMプログラムは不要であるが，これについては各自，検討されたい。

```
/* EX193.C */
#include <process.h>
extern int a,b;

main()
{
float d;
/* - DSP Prog. LOADING - */
outp(0x02d0,0);
outp(0x02d2,(inp(0x02d2) & 0xc0));
outp(0x02d2,(inp(0x02d2) | 0x40));
 spawnlp(P_WAIT,"d:ld25","d:ld25",
         "b:\\dfile\\ex192.oj","-m","c000",NULL);
/* - DATA TABLE LOADING - */
spawnlp(P_WAIT,"d:ld25","d:ld25",
         "b:\\dfile\\ex192t.oj","-m","c000",NULL);
/* - DSP START - */
outp(0x02d2,(inp(0x02d2) & 0xbf));
outp(0x02d2,(inp(0x02d2) | 0x01));

while(-1)
{
printf("input=");scanf("%f",&d);
if(d<0.0) break;
a=(int)(d*10.0);
rdint();
d=(unsigned int)(b)/4096.0;
printf(" answer=%f\n\n",d);
}
/* - DSP STOP - */
outp(0x02d2,(inp(0x02d2) & 0xfe));
outp(0x02d2,(inp(0x02d2) | 0x40));
printf("Program is ended.\n");
}
```

```
;EX193A.ASM
DGROUP   group  mdata
mdata    segment word public 'data'
         assume  ds:DGROUP
         public  _a,_b
_a       DW  ?
_b       DW  ?
mdata    ends

_TEXT    segment byte public 'CODE'
         assume  cs:_TEXT,ds:DGROUP
         public  _rdint
_rdint   proc near
; DPR Output
         MOV  02D4H
         MOV  AX,_a
         OUT  DX,AX
; DPR Check and Data Input
         MOV  DX,02D2H
DWAIT:   IN   AL,DX
         AND  AL,10H
         JZ   DWAIT
         MOV  DX,02D4H
         IN   AX,DX
         MOV  _b,AX
         RET
_rdint   endp
_TEXT    ends
         end
```

```
;EX193.A32
         ORG  0
         B    MPRO
         RET

         DSEG 4
          BUFF: BSS 1
          VA:   BSS 1
          VB:   BSS 1
         DEND

         BSS   47

MPRO:    DINT
         LDPK 4
;DPR DATA CHECK AND INPUT
LOP:     LACK 1
WAIT:    IN   BUFF,PA0
         AND  BUFF
         BZ   WAIT
         IN   VA,PA1
;TABLE REFERENCE
         LAC  VA
         ADLK $2000
         TBLR  VB
;DPR OUTPUT
         OUT  VB,PA1
         B    LOP
         END
```