Development of an Atomic Force Microscope Suitable for Observation of Biological Samples

生物材料の観察に適した
原子間力顕微鏡の開発

(課題番号　03455011)

平成３－５年度科学研究費補助金（一般研究Ｂ）

研究成果報告書

別冊付録　プログラムソースファイル

平成７年３月

研究代表者　**安藤敏夫**
（金沢大学理学部助教授）

生物材料の観察に適した
原子間力顕微鏡の開発

(課題番号　03455011)

平成３－５年度科学研究費補助金（一般研究Ｂ）

研究成果報告書

別冊付録　プログラムソースファイル

平成７年３月

研究代表者　**安藤敏夫**
（金沢大学理学部助教授）

付録B

（プログラムソースファイル）

付録B　プログラムソースファイル

B.1 Menu Program Files

B.2 Header Program Files

B.3 Sensor Program Files

B.4 Motor Program Files

B.5 Scan Program Files

B.6 Image Program Files

B.7 Edit Program Files

B.8 Common Program Files

プログラムファイル

(B.1 Menu Program Files)

```
/*
+-----------------------------------------------------------------------+
|                              afm_menu.c                               |
|                            --------------                             |
|                                                                       |
| ＡＦＭのメニュウ環境を提供するＡＦＭ＿ＭＥＮＵ．ＥＸＥのメインモジュール |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>

#include "afm_def.h"
#include "video.h"
#include "txcondef.h"

#include    <lib¥crt.h>
#include    <lib¥key.h>
#include    <lib¥bslib.h>
#include    <lib¥keytbl.h>
#include    <lib¥_form.h>

/*
+-----------------------------------------------------------------------+
|                        プロトタイプ 宣 言                             |
+-----------------------------------------------------------------------+
*/
static void start_job(void);
       void end_job(void);
/*
-----------------------------------------------------------------------+
|                        外 部 関 数  使 用 宣 言                       |
+-----------------------------------------------------------------------+
*/
extern void init_key(void);
extern int  select_from_menu(void);
extern void call_func(int);
/*
+-----------------------------------------------------------------------+
|                                 コード                                |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名       static void start_job(void)

    コメント       初期化関数をコールする

+-----------------------------------------------------------------------*/
static void start_job(void)
{
    tinit();                        /* 望洋ライブラリー使用宣言 */
    Csloff();                       /* カーソルＯＦＦ */
    Cls();                          /* クリアスクリーン */
    FUNC_KEY(OFF);                  /* ファンクションキーの内容非表示 */

    init_avgdrv(PC9821);                /* 拡張グラフィクスドライバの初期化 */
    set_256palet(MONOCHROME);           /* パレットのセット */
    init_key();
```

```
}

/*-----------------------------------------------------------------------+

    関数名      static void end_job(void)

    コメント      終了処理

+-----------------------------------------------------------------------*/
void end_job(void)
{
    Cls();
    end_videodrv();                 /* 拡張グラフィクスドライバの使用終了 */
    tterm();                        /* 望洋ライブラリー使用終了 */
    Cslon();                        /* カーソルＯＮ */
    FUNC_KEY(ON);                   /* ファンクションキーの内容表示 */
}

/*-----------------------------------------------------------------------+

    関数名      int main(void)

+-----------------------------------------------------------------------*/
void main(void)
{
    int menu_no;

    start_job();                                /* 初期化関数の呼び出し */

    while(1){               /* メニュウループ　終了を選択するまで抜けない */
        menu_no = select_from_menu();
        call_func(menu_no);
        if(menu_no == 5200)             /* メニュウで終了が選択された場合 */
            break;
    }
    end_job();                                  /* 終了処理 */
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              afm_util.c                               |
|                             --------------                            |
|                                                                       |
|                       汎用的な関数を集めたファイル                      |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <process.h>    /* _spawnl() */
#include <string.h>     /* strcat(),strcpy() */
#include <dos.h>        /* _dos_getftime() , _dos_setftime() */
#include <lib¥dir.h>
#include <lib¥_form.h>

#include "afm_def.h"
#include "txcondef.h"
#include "popup.h"

unsigned fdate,ftime;
/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
void init_key(void);
int  getftime(char *,unsigned *,unsigned *);
int  setftime(char *,unsigned,unsigned);
int  get_file_name(char *,char *,char *);
FILE *open_file(FILE **,char *);
FILE *open_temp_file(FILE *,char *,int);
void del_temp(char *,int);
/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int dir_select(DIR *,char *,char *,char *,int,int,int,int,int,int);
extern void errer_msg(char *);
/*
+-----------------------------------------------------------------------+
|                               コード                                  |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      void init_key(void)

    コメント      キー配列の初期化。望洋ライブラリーを使う場合の
                カーソルキーの設定

+-----------------------------------------------------------------------*/
void init_key(void)
{
    int KeyChange = 0;
```

```
    if (!___DiamondKey) {
      . KeyChange = 1;
        ___pushskey();
    }
}

int getftime(char *fname,unsigned *date,unsigned *time)
{
    FILE *fp;
    int  hd;

    if((fp = fopen(fname,"r"))==NULL)
        return(1);
    hd = fileno(fp);
    _dos_getftime(hd,date,time);
    fclose(fp);
    return(0);
}

int setftime(char *fname,unsigned date,unsigned time)
{
    FILE *fp;
    int  hd;

    if((fp = fopen(fname,"a"))==NULL)
        return(1);
    hd = fileno(fp);
    _dos_setftime(hd,date,time);
    fclose(fp);
    return(0);
}

/*-----------------------------------------------------------------------+

    関数名      int get_file_name(char *file_path,char *name,char *slct_name)

    引数        char *file_path:検索するファイルのパス
                char *name:     検索するファイルの名前（ワイルドカード）
                char *slct_name:選択されたファイルのフルパス名

    返値        選択したファイルの数　何も選択しなかった場合 0

    コメント      引数で指定されたパスから、ファイルを選択する。

+-----------------------------------------------------------------------*/
int get_file_name(char *file_path,char *name,char *slct_name)
{
    DIR     buf[1];
    int     flag;

    strcat(file_path, "*.*");
    flag = dir_select(buf, file_path, name, slct_name, 1, 0,
                              POPUP1_X1, 4, POPUP1_Y1, POPUP1_Y2);
    strcat(slct_name, buf[0].name);

    return(flag);
}
```

```
/*-----------------------------------------------------------------------+

    関数名      FILE *open_file(FILE **f_ptr,char *search_path,char *wild_name)

    引数        FILE **f_ptr:       オープンするファイルを示すポインタ
                char *search_path:  検索するファイルのパス
                char *wild_name:    検索するファイルの名前（ワイルドカード）

    返値        オープンしたファイルのポインタ　何も選択しなかった場合 NULL

    コメント      引数で指定されたパスから、引数で指定された名前のファイル
                （ワイルドカードを含む）をオープンする。

+-----------------------------------------------------------------------*/
FILE *open_file(FILE **f_ptr,char *fname)
{
    if(getftime(fname,&fdate,&ftime) == 1){
                                            /* ファイルをオープンできない */
        char msg[200];

        strcpy(msg,fname);
        strcat(msg,"¥nがオープンできません¥n");
        errer_msg(msg);
        putchar('¥a');
        return(NULL);
    }
    *f_ptr = fopen(fname,"r+b");
    return(*f_ptr);
}

/*-----------------------------------------------------------------------+

    関数名      FILE *open_temp_file(FILE *fp,char *prmfile_name,int part)

    引数        FILE *fp:           ファイルポインター
                char *prmfile_name: ＡＦＭのパラメータファイル名
                int part:           パラメータファイル内のパート

    返値        選択したファイルを指すファイルポインター
                NULL: 何も選択しなかった、エラー

    コメント      スキャンのデータファイルをパラメータファイルで指定された、
                パスから選択すし、指定された名前でラムディスクにコピーし、
                そのファイルをオープンし、ファイルポインタを返す。

+-----------------------------------------------------------------------*/
FILE *open_temp_file(FILE *fp,char *prmfile_name,int part)
{
    char parameter[30][40],*file_name;
    int number_of_prm,result;

    number_of_prm = read_prm_file(prmfile_name,parameter,part);

    get_file_name(parameter[0 + number_of_prm],"*.d*",file_name);
                    /* スキャンデータファイルのデフォルトの拡張子 *.dat */

    result = _spawnl(_P_WAIT,"a:¥¥afm¥¥afm_menu¥¥image¥¥movefile.bat",
                     "a:¥¥afm¥¥afm_menu¥¥image¥¥movefile.bat",
```

```
                    file_name,parameter[9 + number_of_prm],
                    parameter[10 + number_of_prm],NULL);
            /* ラムディスクにファイルをコピーするバッチファイルを立ち上げる */

    strcat(parameter[9 + number_of_prm],parameter[10 + number_of_prm]);

    if((fp = fopen(parameter[9 + number_of_prm],"rb")) == NULL){
                                /* テンポラリーファイルをオープンできない */
        char msg[200];

        strcpy(msg,parameter[9 + number_of_prm]);
        strcat(msg,"¥nがオープンできません¥n");
        errer_msg(msg);
        putchar('¥a');
        return(NULL);
    }
    return(fp);
}

/*-----------------------------------------------------------------------+

    関数名      void del_temp(char *prmfile_name,int part)

    引数        char *prmfile_name: ＡＦＭのパラメータファイル名
                int part:           パラメータファイル内のパート

    コメント      パラメータファイルで指定された名前のラムディスクを
                消去するバッチファイルを起動する。

+-----------------------------------------------------------------------*/
void del_temp(char *prmfile_name,int part)
{
    char parameter[30][40];
    int number_of_prm,result;

    number_of_prm = read_prm_file(prmfile_name,parameter,part);

    strcat(parameter[9 + number_of_prm],parameter[10 + number_of_prm]);

    result = _spawnl(_P_WAIT,"a:¥¥afm¥¥afm_menu¥¥image¥¥del_temp.bat",
                    "a:¥¥afm¥¥afm_menu¥¥image¥¥del_temp.bat",
                    parameter[9 + number_of_prm],NULL);
            /* テンポラリファイルを削除するバッチファイルを立ち上げる */
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              mendef25.c                               |
|                            --------------                             |
|                                                                       |
|   ＡＦＭのメニュウ環境を提供するＡＦＭ＿ＭＥＮＵ．ＥＸＥのサブモジュール |
|                    ＡＦＭのメニュウの定義とその選択                    |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include    <stdio.h>
#include    <lib¥crt.h>
#include    <lib¥key.h>
#include    <lib¥bslib.h>
#include    <lib¥keytbl.h>

#include "popup.h"

extern int horizonal_select(int *,int,int,int,int,int,unsigned,unsigned);

/*
+-----------------------------------------------------------------------+
|                                メニュウ                               |
+-----------------------------------------------------------------------+
*/
MenuDef(Help, "ヘルプ")        /*-- ヘルプ --*/
    { 8100, "1:バージョン", NULL },
    { 8200, "2:  項目別  ", NULL },
MenuEnd;

MenuDef(Kakutyou, "拡張機能")        /*-- 拡張機能 --*/
    { 7100, "1:摩擦力", NULL },
    { 7200, "2:力測定", NULL },
MenuEnd;

MenuDef(Hensuu, "変数設定")        /*-- 変数設定 --*/
    { 6100, "1:      参照        ", NULL },
    { 6200, "2:      変更        ", NULL },
    { 6300, "3:パラメータファイル", NULL },
MenuEnd;

MenuDef(Syuuryou, "終了")        /*-- 終了 --*/
    { 2200, "1:ウイズドロー", NULL },
    { 5200, "2:    終了    ", NULL },
MenuEnd;

MenuDef(Zahyou, "座標")        /*-- 色 --*/
    { 4141, "1:広さ表示", NULL },
    { 4142, "2:範囲表示", NULL },
MenuEnd;

MenuDef(Houhou, "方法")        /*-- 方法 --*/
    { 4110, "1:ワイヤーフレーム", NULL },
    { 4120, "2:クイックペイント", NULL },
    { 4130, "3: シェーディング ", NULL },
    { 4140, "4:  トップビュー  ", Zahyou },
MenuEnd;
```

```
MenuDef(Iro, "色")          /*-- 色 --*/
    { 4410, "1:モノクロ", NULL },
    { 4420, "2:  赤色  ", NULL },
    { 4430, "3:  緑色  ", NULL },
    { 4440, "4:  黄色  ", NULL },
    { 4450, "5:  青色  ", NULL },
    { 4460, "6:  紫色  ", NULL },
    { 4470, "7:  水色  ", NULL },
    { 4480, "8:ＢＲＧＷ", NULL },
MenuEnd;

MenuDef(Gazou, "画像")          /*-- 画像 --*/
    { 4100, "1:  描画  ", Houhou },
    { 4200, "2: セーブ ", NULL },
    { 4300, "3: ロード ", NULL },
    { 4400, "4:パレット", Iro }, /* 子メニューは色(Iro) */
    { 4500, "5:グリッド", NULL },
    { 4600, "6: クリア ", NULL },
MenuEnd;

MenuDef(F_dot, "ドット数")          /*-- F_ドット数 --*/
    { 3110, "1:400", NULL },
    { 3120, "2:200", NULL },
    { 3130, "3:100", NULL },
MenuEnd;

MenuDef(H_dot, "ドット数")          /*-- H_ドット数 --*/
    { 3210, "1:400", NULL },
    { 3220, "2:200", NULL },
    { 3230, "3:100", NULL },
MenuEnd;

MenuDef(E_dot, "ドット数")          /*-- E_ドット数 --*/  /* ver.2.3 */
    { 3310, "1:400", NULL },
    { 3320, "2:200", NULL },
    { 3330, "3:100", NULL },
MenuEnd;

MenuDef(N_dot, "ドット数")          /*-- N_ドット数 --*/  /* ver.2.4 */
    { 3410, "1:400", NULL },
    { 3420, "2:200", NULL },
    { 3430, "3:100", NULL },
MenuEnd;

MenuDef(Scan, "スキャン")          /*-- スキャン --*/
    { 3100, "1:コンスタントフォース", F_dot },
    { 3200, "2: コンスタントハイト ", H_dot },
    { 3300, "3:エラーチェック (CF) ", E_dot },  /* ver.2.3 */
    { 3400, "4:ニアー・ファー (CF) ", N_dot },  /* ver.2.4 */
MenuEnd;

MenuDef(Jog, "ジョグ")          /*-- ジョグ --*/
    { 2410, "1:アップ", NULL },
    { 2420, "2:ダウン", NULL },
MenuEnd;

MenuDef(Motor, "モーター")          /*-- モーター --*/
    { 2100, "1:  アプローチ  ", NULL },
```

```
    { 2200, "2: ウイズドロー ", NULL },
    { 2300, "3:フォースカーブ", NULL },
    { 2400, "4:   ジョグ     ", Jog},     /* 子メニューはジョグ(Jog) */
    { 2500, "5:  一時退避    ", NULL },
MenuEnd;

MenuDef(Senser, "センサー")                /*-- センサー --*/
    { 1100, "1:センサー", NULL },
MenuEnd;

/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      int parent_menu_sct(int *selected, int menu_max)

    引数        *selected:  現在選択されているメニュウの番号
                menu_max:   選択できるメニュウの個数

    返り値      ＣＲ，ＥＳＣ，ＤＯＷＮ

    コメント      メニュウバーを表示し、選択する関数を呼ぶ。

+-----------------------------------------------------------------------*/
int parent_menu_sct(int *selected, int menu_max)
{
    int imput_key_no;

    disp(MENU_X1 + MENU_DX * 0, MENU_Y, MENU_COLOR, "センサー");
    disp(MENU_X1 + MENU_DX * 1, MENU_Y, MENU_COLOR, "モーター");
    disp(MENU_X1 + MENU_DX * 2, MENU_Y, MENU_COLOR, "スキャン");
    disp(MENU_X1 + MENU_DX * 3, MENU_Y, MENU_COLOR, " 画  像 ");
    disp(MENU_X1 + MENU_DX * 4, MENU_Y, MENU_COLOR, " 終  了 ");
    disp(MENU_X1 + MENU_DX * 5, MENU_Y, MENU_COLOR, "変数設定");
    disp(MENU_X1 + MENU_DX * 6, MENU_Y, MENU_COLOR, "拡張機能");
    disp(MENU_X1 + MENU_DX * 7, MENU_Y, MENU_COLOR, " ヘルプ ");
    imput_key_no = horizonal_select(selected, menu_max, MENU_X1, MENU_X2,
            MENU_Y, MENU_DX, MENU_COLOR, MENU_SELECT_COLOR);
    return(imput_key_no);
}

/*-----------------------------------------------------------------------+

    関数名      int select_from_menu()

    引数        *selected:  現在選択されているメニュウの番号
                menu_max:   選択できるメニュウの個数

    返り値      ＣＲ，ＥＳＣ，ＤＯＷＮ

    コメント      メニュウバーを表示し、選択する関数を呼ぶ。

+-----------------------------------------------------------------------*/
int select_from_menu()
{
```

```
static int parent_menu = 1;
int imput,menu_number;

do{
    imput = parent_menu_sct(&parent_menu,MENU_MAX);
    if(imput == ESC)
        return(0);

    while(1){   /* リターンか、エスケープを押すまで抜けない */
        int x;

        switch(parent_menu){
        case 1:
            imput = MenuSelect(Senser, &menu_number, 1, 2);
            break;
        case 2:
            imput = MenuSelect(Motor, &menu_number, 5,2);
            break;
        case 3:
            imput = MenuSelect(Scan, &menu_number, 12,2);
            break;
        case 4:
            imput = MenuSelect(Gazou, &menu_number, 28, 2);
            break;
        case 5:
            imput = MenuSelect(Syuuryou, &menu_number, 36, 2);
            break;
        case 6:
            imput = MenuSelect(Hensuu, &menu_number, 43,2);
            break;
        case 7:
            imput = MenuSelect(Kakutyou, &menu_number, 59,2);
            break;
        case 8:
            imput = MenuSelect(Help, &menu_number, 65,2);
            break;
        }

    /* メニュウをポップアップした状態で左右のカーソルキーが
    押された場合、親メニュウのＮｏを変えて、選択を示す反転表示を
    隣に移す。 */

        if(imput == LEFT || imput == RIGHT){
            x = (parent_menu - 1) * MENU_DX;
            color_(MENU_X1+x,MENU_Y,MENU_X2+x,MENU_Y,MENU_COLOR);
            if(imput == LEFT)
                parent_menu--;
            else
                parent_menu++;
            if(parent_menu < 1 || parent_menu > MENU_MAX)
                parent_menu = MENU_MAX;
            x = (parent_menu - 1) * MENU_DX;
            color_(MENU_X1+x,MENU_Y,MENU_X2+x,MENU_Y,MENU_SELECT_COLOR);
        }else{
            break;
        }
    }
}while(imput != CR);
```

```
    return(menu_number);
}
```

```
/*
+-----------------------------------------------------------------------+
|                              menuslct.c                               |
|                            --------------                             |
|                                                                       |
|   ＡＦＭのメニュウ環境を提供するＡＦＭ＿ＭＥＮＵ．ＥＸＥのサブモジュール |
|   メニュウ選択を行う３つの関数からなる。                                |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <string.h>
#include <lib¥crt.h>
#include <lib¥key.h>
#include <lib¥bslib.h>
#include <lib¥keytbl.h>
#include <lib¥window.h>

#include "popup.h"

/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
int vertical_select(int *,int,int,int,int,int,unsigned,unsigned);
int horizonal_select(int *,int,int,int,int,int,unsigned,unsigned);
int MenuSelect(MENU *,int *,int,int);
/*
+-----------------------------------------------------------------------+
|                                 コード                                |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      int vertical_select(int *no,int max,int x1,int x2,int y1,
                                    int dy,unsigned atr1,unsigned atr2)

    引数        *no:    現在選択されているメニュウの番号
                max:    選択できるメニュウの個数
                x1:     初めのメニュウの左ｘ文字座標
                x2:     初めのメニュウの右ｘ文字座標
                y1:     メニュウのｙ文字座標
                dy:     １メニュウの文字数
                atr1:   非選択色
                atr2:   選択色

    返り値      ＣＲ，ＥＳＣ，ＬＥＦＴ，ＲＩＧＨＴ

    コメント      縦方向に並んだメニュウをカーソルキーで選択する

+-----------------------------------------------------------------------*/
int vertical_select(int *no , int max , int x1 , int x2 , int y1 , int dy ,
            unsigned atr1 , unsigned atr2)
{
    register int    ch , y;
    int             Cursor = cursormode();      /* カーソル属性 */
    unsigned        Color  = currentcolor();    /* ｶﾚﾝﾄｱﾄﾘﾋﾞｭｰﾄ */
```

```
    if (*no < 1 || *no > max) *no = 1;
    cursor(CURSORoff);
    while (1) {
        y = (*no-1)*dy + y1;
        color_(x1, y, x2, y, atr2);
        ch = inkey();
        color_(x1, y, x2, y, atr1);
        switch (ch) {
            case UP   : *no = (*no >   1) ? *no-1 : max;  break;
            case DOWN : *no = (*no < max) ? *no+1 :   1;  break;
            case LEFT : break;
            case RIGHT: break;
            default   : if (ch >='1' && ch<=max+'0') *no=ch-'0'; break;
        }
    if (ch == CR || ch == ESC || ch == LEFT || ch == RIGHT) break;
    }
    color_(x1, y, x2, y, atr2);
    color(Color);
    cursor(Cursor);
    return(ch);
}

/*-----------------------------------------------------------------------+

    関数名      int horizonal_select(int *no , int max , int x1 , int x2 ,
                                     int y , int dx , unsigned atr1 , unsigned atr2)

    引数        *no:    現在選択されているメニュウの番号
                max:    選択できるメニュウの個数
                x1:     初めのメニュウの左ｘ文字座標
                x2:     初めのメニュウの右ｘ文字座標
                y:      メニュウのｙ文字座標
                dx:     １メニュウの文字数
                atr1:   非選択色
                atr2:   選択色

    返り値      ＣＲ，ＥＳＣ，ＤＯＷＮ

    コメント      横方向に並んだメニュウをカーソルキーで選択する

+-----------------------------------------------------------------------*/
int horizonal_select(int *no , int max , int x1 , int x2 , int y , int dx ,
            unsigned atr1 , unsigned atr2)
{
    /*register*/ int    ch , x;
    int             Cursor = cursormode();      /* カーソル属性 */
    unsigned        Color  = currentcolor();    /* ｶﾚﾝﾄｱﾄﾘﾋﾞｭｰﾄ */

    if (*no < 1 || *no > max) *no = max;
    cursor(CURSORoff);
    while (1) {
        x = (*no-1)*dx;
    color_(x1 + x, y, x2 + x, y, atr2);
        ch = inkey();
        color_(x1 + x, y, x2 + x, y, atr1);
        switch (ch) {
            case LEFT  : *no = (*no >   1) ? *no-1 : max;  break;
            case RIGHT : *no = (*no < max) ? *no+1 :   1;  break;
```

```
            default    : if (ch >='1' && ch<=max+'0') *no=ch-'0';
        }
    if (ch == CR || ch == ESC || ch == DOWN) break;
    }
    color_(x1 + x, y, x2 + x, y, atr2);
    color(Color);
    cursor(Cursor);
    return(ch);
}

/*-----------------------------------------------------------------------+

    関数名      int MenuSelect(MENU *m, int *no, int x, int y)

    引数        *m:         メニュー構造体
                *no:    現在選択されているメニュウの番号
                x:      ポップアップメニュウの左上ｘ文字座標
                y:      ポップアップメニュウの左上ｙ文字座標

    返り値      CR,ESC,LEFT,RIGHT

    コメント      ポップアップメニューを表示し、選択する。

+-----------------------------------------------------------------------*/
int MenuSelect(MENU *m, int *no, int x, int y)
{
    register int    i, max , flag1, flag2, len = 0;
    int             slct = 1;
    MENU            *ptr = m;
    WINDOW          buf;
    int             Cursor = cursormode();      /* カーソル属性 */
    unsigned        Color  = currentcolor();    /* ｶﾚﾝﾄｱﾄﾘﾋﾞｭｰﾄ */

    for ( ; ptr->item ; ptr++)
        if (len < (int)strlen(ptr->item)) len = strlen(ptr->item);
    max = ptr - m - 1;
    if (max < 1) return(-1);

    for (i=1; i<=max; i++) if (m[i].code == *no) slct = i;
    WINopen(&buf,x,y,x+len+3,y+max+1,0,MENU_BACK_COLOR,MENU_FLAME_COLOR);
    WINtitle(&buf, 0,0,MENU_TITLE_COLOR, m[0].item);
    color(MENU_BACK_COLOR);
    for (i = 1; i <= max; i++) {
        dputf(x+2, y+i, "%s", m[i].item);
    }
    do {
        flag2 = CR;
        flag1 = vertical_select(&slct, max, x+1, x+len+2, y+1, 1,
                MENU_BACK_COLOR, MENU_SELECT_COLOR);
    if(flag1 == CR && m[slct].sub)
            flag2 = MenuSelect(m[slct].sub, no, x+3, y+3);
        else
            *no = m[slct].code;
    } while (flag2 == ESC);
    WINclose(&buf);
    color(Color);
    cursor(Cursor);
    return(flag1);
```

```
}
/*
+-----------------------------------------------------------------------------+
|                                 終わり                                      |
+-----------------------------------------------------------------------------+
*/
```

プログラムファイル

(B.2 Header Program Files)

```
/*
+-----------------------------------------------------------------------+
|                              afm_func.h                               |
|                             --------------                            |
|                                                                       |
|                       汎用的な関数を集めたファイル                      |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#ifndef __AFM_FUNC_HEADER__
#define __AFM_FUNC_HEADER__

/*-----------------------------------------------------------------------+

    関数名      float adaverage(ch,sampling)

    引数        ch:         入力チャンネル
                sampling:   ＡＤ変換の平均回数

    返値        ＡＤ変換の平均値

    コメント      ｄｓｐの入力chチャンネルから、引数の sampling 回数
                サンプリングして、その平均を返す。

+-----------------------------------------------------------------------*/
float adaverage(ch,sampling)
short ch;
long sampling;
{
    register long count,adsum = 0;

    switch(ch){
    case 1:
        adc_int_clear();    /* 割り込みを解除することによりＡＤ変換スタート  */
        for(count = sampling;count > 0;count--){    /* 平均回数だけ繰り返す */
            intl_wait();                    /* ＡＤ終了を示す割り込みを待つ  */
            adsum += (long)getlch();            /* ＡＤ値を積算              */
        }
        break;
    case 2:
        adc_int_clear();    /* 割り込みを解除することによりＡＤ変換スタート  */
        for(count = sampling;count > 0;count--){    /* 平均回数だけ繰り返す */
            intl_wait();                    /* ＡＤ終了を示す割り込みを待つ  */
            adsum += (long)get2ch();            /* ＡＤ値を積算              */
        }
        break;
    case 3:
        adc_int_clear();    /* 割り込みを解除することによりＡＤ変換スタート  */
        for(count = sampling;count > 0;count--){    /* 平均回数だけ繰り返す */
            intl_wait();                    /* ＡＤ終了を示す割り込みを待つ  */
            adsum += (long)get3ch();            /* ＡＤ値を積算              */
        }
        break;
    default:
        adc_int_clear();    /* 割り込みを解除することによりＡＤ変換スタート  */
        for(count = sampling;count > 0;count--){    /* 平均回数だけ繰り返す */
            intl_wait();                    /* ＡＤ終了を示す割り込みを待つ  */
```

```
            adsum += (long)get4ch();            /* ＡＤ値を積算              */
        }
        break;
    }

    return( (float)adsum / (float)sampling * (float)AD_CONV );
                /* 積算した値を平均し、   */
                /*  そのビットならびを電圧を示すようスケーリングした値を返す */
}

/*-----------------------------------------------------------------------+

    関数名      void piezo_drive(ch,new_v,last_v,div)

    引数        ch:     出力チャンネル
                new_v:  最終出力電圧値
                last_v: 出力開始電圧値

    返値        無し

    コメント      ｄｓｐの ch チャンネルに、引数の last_v[ch]
                から引数の new_v に、引数の div 回数に分割して
                リニアにピエゾをドライブする

+-----------------------------------------------------------------------*/
void piezo_drive(ch,new_v,last_v,div)
short ch;
float new_v,last_v;
short div;
{
    register float _a,output_v;
    register short times;

    _a = (new_v - last_v) / (float)div;
    output_v = last_v;

    switch(ch){
    case X:
        for(times = div ; times > 0 ; times--){
            output_v += _a;
            put1ch( (short)(output_v * (float)DA_CONV) );
        }
        break;
    case Y:
        for(times = div ; times > 0 ; times--){
            output_v += _a;
            put2ch( (short)(output_v * (float)DA_CONV) );
        }
        break;
    case Z:
        for(times = div ; times > 0 ; times--){
            output_v += _a;
            put3ch( (short)(output_v * (float)DA_CONV) );
        }
        break;
    default:
        break;
    }
```

```
}

/*-----------------------------------------------------------------------+

    関数名  void piezo_2drive(ch,new_v1,last_v1,new_v2,last_v2,div)

    引き数  ch:              出力２チャンネル (XY,YZ,XZ)
            new_v1 ,new_v2 :最終出力電圧値
            last_v1,last_v2:出力開始電圧値

    返り値  無し

    コメント    ｄｓｐの１、２、３チャンネルのうちの２チャンネル分に対して、
                １チャンネル分は last_v1 から new_v1 へ、もう１チャンネル分は
                last_v2 から new_v2 へ、div 回数に分割して、リニアに電圧を出力
                する。（つまり、ピエゾをドライブする。）
                どの２チャンネルかは、引き数 ch により指定する。
                                                    （ｂｙ　清水光太朗）

+-----------------------------------------------------------------------*/
void piezo_2drive(ch,new_v1,last_v1,new_v2,last_v2,div)
short ch,div;
float new_v1,last_v1,new_v2,last_v2;
{
    float _a1,_a2;
    short times;

    _a1 = (new_v1-last_v1) / (float)div;
    _a2 = (new_v2-last_v2) / (float)div;

    switch(ch){
    case XY:
        for(times = 1 ; times <=div ; times++){
            put1ch((short)((last_v1 + _a1 * (float)times) * (float)DA_CONV) );
            put2ch((short)((last_v2 + _a2 * (float)times) * (float)DA_CONV) );
        }
        break;
    case YZ:
        for(times = 1 ; times <=div ; times++){
            put2ch((short)((last_v1 + _a1 * (float)times) * (float)DA_CONV) );
            put3ch((short)((last_v2 + _a2 * (float)times) * (float)DA_CONV) );
        }
        break;
    case XZ:
        for(times = 1 ; times <=div ; times++){
            put1ch((short)((last_v1 + _a1 * (float)times) * (float)DA_CONV) );
            put3ch((short)((last_v2 + _a2 * (float)times) * (float)DA_CONV) );
        }
        break;
    default:
        break;
    }
}

/*-----------------------------------------------------------------------+

    関数名      void wait(times)
```

引数　　times:　ダミーサンプリングの回数

返値　　無し

コメント　　引数の times 回数サンプリングすることにより特定の時間
　　ｄｓｐの動きを止める

```
+-----------------------------------------------------------------------*/
void wait(times)
long times;
{
    register long count;

    adc_int_clear();

    for(count = times;count > 0;count--){
        intl_wait();
        adc_int_clear();
    }
}

#endif

/*
+-----------------------------------------------------------------------+
|                                 終わり                                |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              afm_adda.h                               |
|                            --------------                             |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#ifndef __AFM_ADDA_HEADER__
#define __AFM_ADDA_HEADER__

/*
*    ＡＤ結果を１から４の任意の１チャンネルから読みとる関数と
*  任意の１チャンネルにＤＡを出力するインラインアセンブラ関数
*
*                    Ver 1.0   Date 92/10/8   By 中野
*
*   short   get1ch();
*   short   get2ch();
*   short   get3ch();
*   short   get4ch();
*   void    put1ch( short volt );
*   void    put2ch( short volt );
*   void    put3ch( short volt );
*   void    put4ch( short volt );
*   void    put5ch( short volt );
*   void    put6ch( short volt );
*   void    put7ch( short volt );
*   void    put8ch( short volt );
*/

/*-----------------------------------------------------------------------+

    関数名      asm short  get1...4ch()

    引数          無し

    返値          ＡＤ変換した値

    コメント      r1...4レジスターに、ＡＤの各チャンネルのアドレスを代入する。
                r1...4e の eは、レジスターにアドレスを代入する事を明示する。
                そのアドレスから値をレジスターに読み込む。読み込まれた
                r1レジスターの値は返り値となる。

+-----------------------------------------------------------------------*/
asm short  get1ch()
{
    @W
    r1e = DSP4010 + AD_CH1
    r1  = *r1
    @W
}

asm short  get2ch()
{
    @W
    r1e = DSP4010 + AD_CH2
    r1  = *r1
    @W
```

```
}

asm short  get3ch()
{
    @W
    r1e = DSP4010 + AD_CH3
    r1  = *r1
    @W
}

asm short  get4ch()
{
    @W
    r1e = DSP4010 + AD_CH4
    r1  = *r1
    @W
}

/*-----------------------------------------------------------------------+

    関数名      asm short put1...8ch(volt)

    引数          出力電圧

    返値          出力した値、０の場合エラー

    コメント      r1...8レジスターに、ＤＡの各チャンネルのアドレスを代入して
                そのアドレスに値を書き込むことがＤＡ変換を意味する。
                %はＣ言語でいうifにあたり、関数の引数であるvoltが、
                レジスター変数である場合、一般の変数である場合、ポインタ変数で
                ある場合の場合分けをする。
                  r1レジスターの値は返り値となるので、返値が０の場合は
                エラーである。

+-----------------------------------------------------------------------*/
asm short  put1ch(volt)
{
%   reg volt;                   /* 引数voltがレジスター変数の場合 */

    r1e = DSP4020 + DA_CH1      /* ＤＡ１チャンネルのアドレスをr1にセット */
    *r1 = volt                  /* そのアドレスに引数のＤＡ値を書き込む */

%   con volt;                   /* 引数voltが一般の変数の場合 */

    @W                          /* 必要な数のnopを自動的に生成 */
    r2e = DSP4020 + DA_CH1      /* ＤＡ１チャンネルのアドレスをr2にセット */
    r1  = volt                  /* ＤＡ値をr1にセット */
    *r2 = r1                    /* ＤＡ１ｃｈのアドレスにＤＡ値を書き込む */
    @W

%   mem volt;                   /* 引数voltがポインター変数の場合 */

    @W
    r1e = DSP4020 + DA_CH1      /* ＤＡ１チャンネルのアドレスをr1にセット */
    r2e = volt                  /* ポインタ変数の指し示すアドレスをr2にセット*/
    r2  = *r2                   /* ポインタ変数の指し示す値をr2にセット */
    @W
    *r1 = r2                    /* ＤＡ１チャンネルのアドレスにポインタ変数の
```

```
                                                    指し示す値を書き込む */
    @W

%   error                           /* 引数voltが上記３つ以外の場合 */
    r1  = 0                         /* エラーを示す０をr1にセット */
}

asm short  put2ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH2
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH2
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH2
    r2e = volt
    r2  = *r2
    @W
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put3ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH3
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH3
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH3
    r2e = volt
    r2  = *r2
    @W
```

```
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put4ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH4
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH4
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH4
    r2e = volt
    r2  = *r2
    @W
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put5ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH5
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH5
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH5
    r2e = volt
    r2  = *r2
    @W
```

```
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put6ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH6
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH6
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH6
    r2e = volt
    r2  = *r2
    @W
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put7ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH7
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH7
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH7
    r2e = volt
    r2  = *r2
    @W
```

```
    *r1 = r2
    @W

%   error
    r1  = 0
}

asm short  put8ch(volt)
{
%   reg volt;

    r1e = DSP4020 + DA_CH8
    *r1 = volt

%   con volt;

    @W
    r2e = DSP4020 + DA_CH8
    r1  = volt
    *r2 = r1
    @W

%   mem volt;

    @W
    r1e = DSP4020 + DA_CH8
    r2e = volt
    r2  = *r2
    @W
    *r1 = r2
    @W

%   error
    r1  = 0
}

#endif

/*
+-----------------------------------------------------------------------+
|                                 終わり                                |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              afm_def.h                                |
|                            --------------                             |
|                                                                       |
|        ＡＦＭの各種ファイルで使われるマクロ定義を集めたファイル        |
|                                                                       |
+-----------------------------------------------------------------------+
*/

/*-----------------------------------------------------------------------+

                              フラッグ関係

+-----------------------------------------------------------------------*/
#define DSP_WORKING     0
#define DSP_COMPLETE    1
#define DSP_RESTART     2
#define DSP_END         10

/*-----------------------------------------------------------------------+

                              ピエゾ関係

+-----------------------------------------------------------------------*/
/*#define V_LINEAR*/        /* 板ピエゾを電圧をリニアにドライブするとき
                                                          デファインする */

#ifndef V_LINEAR

#define A_F             3.0482537
#define B_F             -170.42199
#define C_F             1399.3945
#define A_G             -2.236458
#define B_G             -170.42101
#define C_G             1927.8559
#define MAX_SCAN_SIZE   3408.4398

#else

#define A_F             0.0
#define B_F             -150.0
#define C_F             1500.0
#define A_G             0.0
#define B_G             -150.0
#define C_G             1500.0
#define MAX_SCAN_SIZE   3000

#endif

#define MAX_PIEZO_VOLT  10.0
#define SAFE_DRIVE_X    1000
#define SAFE_DRIVE_Y    1000
#define SAFE_DRIVE_Z    1000
#define Z_PIEZO_INDEX   31.21   /* ｚピエゾの伸び [nm]/[DSPのvolt] */
#define EX              0
#define SH              1
```

```
/*-----------------------------------------------------------------------+

                          スキャン関係

+-----------------------------------------------------------------------*/
#define X                   0
#define Y                   1
#define Z                   2
#define XY                  3
#define YZ                  4
#define XZ                  5

#define SAMPLING_MAX        400
#define DAMY_MAX            50
#define CONSTANT_FORCE      0
#define CONSTANT_HEIGHT     1
#define ERROR_CHECK         2
#define NEAR_FAR            3

#define TOP_VIEW_WINDOW_X   10
#define TOP_VIEW_WINDOW_Y   36

#define SCAN_DAT_FILE       1
#define PRM_BYTE            40

#define AD_CONV (1.0 / 3276.8)
#define DA_CONV 3276.8

/*-----------------------------------------------------------------------+

                          モーター関係

+-----------------------------------------------------------------------*/
#define MOTOR_WINDOW_X      10
#define MOTOR_WINDOW_Y      80
#define MOTOR_UP            0
#define MOTOR_DOWN          1
#define MOTOR_STEP          1.09        /* 1ステップ当たりのチップの変位 */
#define APPROACH            1
#define WITHDRAW            2
#define FORCE_CURVE         3
#define JOG_UP              4
#define JOG_DOWN            5
#define TEMP_ESC            6
/*-----------------------------------------------------------------------+

                      パラメータファイル関係

+-----------------------------------------------------------------------*/
#define SENSER_PART         1
#define MOTOR_PART          2
#define SCAN_PART           3
#define IMAGE_PART          4

/*-----------------------------------------------------------------------+

                          画像処理関係
```

```
+-----------------------------------------------------------------------*/
#define PC9821 0
#define PC9801 1

#define WIRE_FRAME  1
#define QUICK_PAINT 2
#define SHADING 3

#define MONOCHROME  0
#define RED         1
#define GREEN       2
#define BLUE        4
#define YELLOW      3
#define CYAN        6
#define MAGENTA     5
#define BRGW        7

/*-----------------------------------------------------------------------+
```

その他

```
+-----------------------------------------------------------------------*/
#define YES     0
#define NO      1
#define SUCCESS 0
#define ERROR   1

#define CHANGE  1

#define PI      3.14159265358979
/*+----------------------------------------------------------------------+
  |                                 終わり                                |
  +----------------------------------------------------------------------+*/
```

```
/*
+-----------------------------------------------------------------------+
|                              vmath.h                                  |
|                            -------------                              |
|                                                                       |
|                   Specification part of vmath package                 |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <math.h>

#define DIM 3

#define X 0
#define Y 1
#define Z 2

typedef double Element;
extern Element *stkpnt;

extern void sinit(void);
extern void prints(void);

/*
+-----------------------------------------------------------------------+
|                     v??? functons operate vectors                     |
+-----------------------------------------------------------------------+
*/

typedef Element Vector[DIM];
extern Vector vzero;

#define vdrop(n) (stkpnt = (Element *)((Vector *)stkpnt + n))
#define vnorm() (vdup(), vmul((double)1.0 / (double)sqrt(vabs2())))

extern void printv(void);

extern void vpsh(Vector);
extern void vpshi(double,double,double);
extern void vdup(void);

extern void vpop(Vector);

extern void vadd(void);
extern void vsub(void);
extern void vneg(void);
extern void vmul(double);

extern double vspro(void);
extern double vabs2(void);
extern double vcos(void);

extern void vvpro(void);

/*
+-----------------------------------------------------------------------+
|                    c?? function operate colors                        |
```

```
+-----------------------------------------------------------------------+
*/

typedef Element Color[DIM];

#define R 0
#define G 1
#define B 2

#define cpsh vpsh
#define cpshi vpshi
#define cpop vpop
#define cdup vdup
#define cadd vadd
#define cmul vmul

extern void cflt(void);

/*
+-----------------------------------------------------------------------+
|                        m??? function operate matrixes                 |
+-----------------------------------------------------------------------+
*/

typedef Element Matrix[DIM][DIM];

#define change(mat, c1, c2)¥
        {¥
            Element tmp;¥
¥
            tmp = mat[c1][c2];¥
            mat[c1][c2] = mat[c2][c1];¥
            mat[c2][c1] = tmp;¥
        }

extern Matrix munit;

extern void mpsh(Matrix);
extern void mpop(Element *);

extern void mvmul(void);
extern void mmmul(void);

/*
+-----------------------------------------------------------------------+
|                              end of vmath.c                           |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                video.h                                |
|                             ---------------                           |
|                                                                       |
|     ＰＣ９８２１の拡張グラフィクスドライバを使うためのマクロ定義と      |
|   プロトタイプ宣言を集めたもの。拡張グラフィクスドライバを使うモジュールに|
|   インクルードする。                                                   |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#define PC9821 0
#define PC9801 1

#define DEVGRDRV    "AVGDRV$$"
#define INTGRDRV    0xd8

/* 画面設定系ファンクション */

#define CMDSETSYNC  0x00
#define CMDASDSYNC  0x01
#define CMDALLOCMODE    0x02
#define CMDASKMODE  0x03
#define CMDINITFRAME    0x04
#define CMDSETFRAME 0x05
#define CMDASKFRAME 0x06
#define CMDCOMPOSE  0x07
#define CMDASKCOMP  0x08
#define CMDINIPAL   0x09
#define CMDSETPAL   0x0a
#define CMDASKPAL   0x0b
#define CMDSTART    0x19
#define CMDEND      0x1a
#define CMDGETINFO  0x1b

/* ジオメトリックファンクション */

#define CMDINITGLO  0x70
#define CMDASKGLO   0x71
#define CMDSELFRAME 0x72
#define CMDCLS      0x73
#define CMDVIEW     0x74
#define CMDPSET     0x75
#define CMDSTYLE    0x76
#define CMDLINE     0x77
#define CMDCONNECT  0x78
#define CMDBOX      0x79
#define CMDBOXFILL  0x7a
#define CMDCIRCLE   0x7b
#define CMDPAINT1   0x7c
#define CMDPAINT2   0x7d
#define CMDPOINT    0x7e
#define CMDPUTPAT   0x80
#define CMDGETPAT   0x81
#define CMDMOVPAT   0x82

/*------------------Ｅｘｔｅｒｎ　ｆｕｎｃｔｉｏｎｓ---------------------*/
```

```
extern void end_videodrv(void);
extern int  call_videodrv(int cmd,union _REGS *regs);
extern int  start_videodrv(void);
extern void init_avgdrv(int mode);
extern void set_palet(  unsigned,unsigned,unsigned,unsigned);
extern void set_256palet(int palet_mode);
extern void set_line_style(unsigned int stile);
extern void set_pxel(int x,int y,unsigned int color);
extern void set_line(int x1,int y1,int x2,int y2,unsigned int color,int attr);
extern void line_to(int x,int y,unsigned int color);
extern void set_box(int x1,int y1,int x2,int y2,unsigned intcolor);
extern void set_box_fill(int x1,int y1,int x2,int y2,unsigned int color);
extern void set_pattern(int dx,int dy,  unsigned int s_off,unsigned int s_seg,
                        int x,int y,int attr);
extern void get_pattern(int dx,int dy,int x,int y,unsigned int d_off,
                     unsigned int d_seg,int attr);
extern void move_pattern(int size_x,int size_y,int sx,int sy,
                     int dx,int dy,int attr);
extern void draw_color_bar(int x1,int y1,int width);
extern void set_grid(int x1,int y1,int length,float interval,unsigned int stile
                     ,int col,int attr);

/*------------------------------ 外部変数 ---------------------------------*/

extern union _REGS regs;
extern struct _SREGS sregs;
extern int param[20];
```

```
/*
+-----------------------------------------------------------------------------+
|                                TXCONDEF.H                                   |
|                               --------------                                |
|                                                                             |
|                      text画面制御用変数・関数定義                           |
|                                                                             |
+-----------------------------------------------------------------------------+
*/

#define  BEEP     0x07
#define  CR       0x0d
#define  ESC      0x1b
#define  ON       1
#define  OFF      0

/* ディファイン関数 */
#define  Locate(x,y)     printf("¥x1b[%d;%dH",y,x);
#define  Cls()           printf("¥x1b[2J");
#define  Cslon()         printf("¥x1b[>5l");
#define  Csloff()        printf("¥x1b[>5h");
#define  BriRev()        printf("¥x1b[5;7m");
#define  DefaAtl()       printf("¥x1b[0m");
#define  CURSOR(sw)      fprintf(stderr,"¥033[>5%c",(sw)? 'l': 'h')
#define  LINE25(sw)      fprintf(stderr,"¥033[>1%c",(sw)? 'l': 'h')
#define  Cll()           fprintf(stderr,"¥033[2K")
#define  COLOR(c)        fprintf(stderr,"¥33[%dm",((c)&0x7)+30+(((c)>>3)&1)*10)
#define  INKEY()         (kbhit()? getch(): 0)
#define  Clf()           fprintf(stderr,"¥033[>1h")
#define  FUNC_KEY(sw)    fprintf(stderr,"¥033[>1%c",(sw)? 'l': 'h')
```

```
/*
+-----------------------------------------------------------------------+
|                                popup.h                                |
|                              --------------                           |
|                                                                       |
|          ポップアップウインドウと、ポップアップメニュウのデファイン          |
|                                                                       |
+-----------------------------------------------------------------------+
*/
/******************* ポップアップウィンドウ関係 *************/

#define POPUP_SWITCH        0

#define POPUP_BACK_COLOR    White|Reverse
#define POPUP_FRAME_COLOR   Green|Reverse
#define POPUP_FRAME_COLOR2  Yellow|Reverse
#define POPUP_TITLE_COLOR   White|Reverse

#define POPUP1_X1   11
#define POPUP1_Y1   3
#define POPUP1_X2   70
#define POPUP1_Y2   14

#define POPUP2_X1   55
#define POPUP2_Y1   3
#define POPUP2_X2   80
#define POPUP2_Y2   9

#define POPUP3_X1   55
#define POPUP3_Y1   11
#define POPUP3_X2   80
#define POPUP3_Y2   15

#define POPUP4_X1   55
#define POPUP4_Y1   3
#define POPUP4_X2   80
#define POPUP4_Y2   10      /* m_motrl0(ver.1.0),m_sensl0(ver.1.0) */

#define EDIT_X1     1
#define EDIT_X2     80
#define EDIT_Y1     3
#define EDIT_Y2     7

/****************** エラーウインドウ関係 *********************/

#define ERRER_FRAME_COLOR   Red|Reverse
#define ERRER_TITLE_COLOR   Red|Reverse

#define ERRER_X1    20
#define ERRER_X2    60
#define ERRER_Y1    5
#define ERRER_Y2    20

/******************* ファイル検索関係 ************************/

#define FILESLCT_X1   11
#define FILESLCT_Y1   3
#define COLOM   70
```

```
#define FILESLCT_Y2 14

/****************** メニュウ関係 ****************************/

#define MENU_MAX 8
#define MENU_X1 1
#define MENU_X2 8
#define MENU_Y  1
#define MENU_DX 10

#define MENU_COLOR       Cyan|Reverse
#define MENU_BACK_COLOR      White|Reverse
#define MENU_FLAME_COLOR     Green|Reverse
#define MENU_TITLE_COLOR     White|Reverse
#define MENU_SELECT_COLOR    Yellow|Reverse
```

```
/* BGRAPH11.Cモジュールを利用するモジュールのためのヘッダーファイル */

/* 外部関数使用宣言 */
extern void  Init(void);
extern void  Set_color(int);
extern void  Axis(void);
extern int   Draw(int,int,float [],float []);
extern void  Autoscale(int,int,float [],float []);
extern void  Autoscalepoint(int,int,float [],float []);
extern double Max(int,int,float []);
extern double Min(int,int,float []);
extern void  Avmaxmin(int,int,float[]);
extern void  End(void);

/* 構造体・共用体  extern宣言 */
extern struct  g_area{
      int  minx;
      int  miny;
      int  maxx;
      int  maxy;
      int  zerox;
      int  zeroy;
      };
extern struct  g_scale{
      double  x;
      double  y;
      int     autox;
      int     autoy;
      double  exrtx;
      double  exrty;
      };
extern struct  g_scalepoint{
      double  x;
      double  y;
      int     autox;
      int     autoy;
      double  divx;
      double  divy;
      };
extern struct  g_unit{
      char  *x;
      char  *y;
      };
extern struct  g_valinfo{
      double  max;
      double  min;
      };
extern struct  graph{
      struct g_area       area;
      struct g_scale      scale;
      struct g_scalepoint dsp;
      struct g_unit       unit;
      struct g_valinfo    value;
      int     axcolor;
      int     spcolor;
      int     pocolor;
      } gparm;
```

プログラムファイル

(B.3 Sensor Program Files)

```
/*
+-----------------------------------------------------------------------------+
|                              m_sens10.c                                     |
|                            --------------                                   |
|                                                                             |
|     センサーからのポジション情報、入射光量の値を取り込み表示する            |
|                                                   Ver.1.02 (KOTARO)         |
|                                                                             |
+-----------------------------------------------------------------------------+
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <lib¥crt.h>
#include <lib¥key.h>
#include <lib¥bslib.h>
#include <lib¥keytbl.h>
#include <lib¥window.h>

#include "mttdsp.h"
#include "txcondef.h"
#include "popup.h"
#include "afm_def.h"
#include "bgrallex.h"                    /* ver.1.0 */

#define BADRS   0xD2                /* ベースアドレスの設定 */
#define TARGET  "t_sens10.out"      /* ver.1.02 */

/*
+-----------------------------------------------------------------------------+
|                              外部関数                                       |
+-----------------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                          short start,short stop);
/*
+-----------------------------------------------------------------------------+
|                           プロトタイプ宣言                                  |
+-----------------------------------------------------------------------------+
*/
static void init_dsp(void);
static int arg(int,char *[]);
static void set_senser_param(void);                 /* +ver.1.0 */
       void set_graph_param(void);                  /* ver.1.0 */
       void disp_scale(void);                       /* ver.1.0 */
static void set_popup_window(void);                 /* +ver.1.0 */
static void ad(void);
static void output(int *,float [],float []);        /* +ver.1.01 */
       void AutoGraph(int,int,float [],float []);   /* ver.1.01 */
static int start_job(int,char *[]);                 /* +ver.1.0 */
       void end_job(short flag);
/*
+-----------------------------------------------------------------------------+
|                           グローバル変数                                    |
+-----------------------------------------------------------------------------+
*/
```

```
static char     prm_file_name[80];
static ulong    flag_adr,times_adr,average_adr,spot_adr,intensity_adr;
                /* 共通変数のＤＳＰ４１００ボード上のアドレスを示す変数 */
static short    m_flag = DSP_WORKING;
                /* ターゲットプログラムとのコミュニケーション用フラッグ */
static long     m_times,m_average /*,samp ver.1.0*/;
static float    wait_sec,m_spot,m_intensity,samp/*ver.1.01*/;

static WINDOW buf;      /* 望洋ライブラリーのポップアップウインドウ用変数 */
/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/

/*-----------------------------------------------------------------------+

    関数名      static void init_dsp(void)

    コメント      ｄｓｐの初期化と、ターゲットプログラムのダウンロード、
                共通変数のＤＳＰ４１００ボード上のアドレスの取得

+-----------------------------------------------------------------------*/
static void init_dsp(void)
{
                                    /* ＤＳＰインターフェースライブラリー */
    dsp_init(BADRS,TARGET);         /* ＤＳＰの初期化とターゲットプログラム
                                                          のダウンロード */
    dsp_sym(flag_adr,"t_flag");     /* 共通変数のＤＳＰ上のアドレスを取得 */
    dsp_sym(times_adr,"t_times");
    dsp_sym(average_adr,"t_average");
    dsp_sym(spot_adr,"t_spot");
    dsp_sym(intensity_adr,"t_intensity");
}

/*-----------------------------------------------------------------------+

    関数名      static void arg(int argc,char *argv[])

    コメント      コマンドライン引数の処理。

+-----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[20][40];
    int counter;

                                    /* デフォルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */

    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    strcpy(prm_file_name,arguments[1]);
    return( atoi(arguments[0]) );
```

```
}

/*-----------------------------------------------------------------------+

    関数名        static void set_senser_param()

    コメント        パラメータファイルからセンサーのパラメータを読み込み、
                    ｄｓｐにパラメータを送る。

+-----------------------------------------------------------------------*/
static void set_senser_param(void)
{
    static char parameter[30][40];
    int number_of_prm;

    number_of_prm = read_prm_file(prm_file_name,parameter,SENSER_PART);

    samp = (float)atof(parameter[0 + number_of_prm]);        /* ver.1.01 */
    m_times = (long)(samp * 100000);                  /* ver.1.02 100KHz */
    m_average = atol(parameter[1 + number_of_prm]);
    dsp_put(times_adr,&m_times,4);  /* ＤＳＰインターフェースライブラリー */
    dsp_put(average_adr,&m_average,4);  /* ＤＳＰのメモリーに値を書き込む */
}

/*-----------------------------------------------------------------------+

    関数名        void set_graph_param(void)

    コメント        グラフ描画のためのパラメータの設定

+-----------------------------------------------------------------------*/
void set_graph_param(void)                          /* +ver.1.01 */
{
    gparm.unit.x  = "[sec]";            /* ｘ軸単位  秒 */
    gparm.scale.x = (double)(5.0/samp); /* ｘ軸縮尺  5/samp dots/sec */
    gparm.dsp.x   = (double)(10.0*samp);/* ｘ軸目盛  10*samp sec/目盛 */
    gparm.scale.y = (double)75.0;       /* ｙ軸縮尺  75 dots/volt */
    gparm.dsp.y   = (double)0.2;        /* ｙ軸目盛  0.2 volt/目盛 */
}

/*-----------------------------------------------------------------------+

    関数名        void disp_scale(void)

    コメント        座標軸の１目盛の値と単位を表示

+-----------------------------------------------------------------------*/
void disp_scale(void)        /* ver.1.0 */
{
    Locate(1,1);
    printf("縦軸１目盛  %4.2lf%s    横軸１目盛  %5.2lf%s",
                    gparm.dsp.y,gparm.unit.y,gparm.dsp.x,gparm.unit.x);
}

/*-----------------------------------------------------------------------+

    関数名        static void set_popup_window(void)
```

コメント　　センサーの値を表示するウインドウを開く。

```
+-----------------------------------------------------------------------*/
static void set_popup_window(void)
{                    /* ver.1.0 */
    WINopen(&buf,POPUP4_X1,POPUP4_Y1,POPUP4_X2,POPUP4_Y2,POPUP_SWITCH,
                POPUP_BACK_COLOR,POPUP_FRAME_COLOR);     /* 望洋ライブラリー */
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"センサー出力");/* 望洋ライブラリー */
    WINlocate(&buf,1,1);                                 /* 望洋ライブラリー */
}

/*-----------------------------------------------------------------------+
```

関数名　　static void ad(void)

コメント　　センサーの値を取得する。

```
+-----------------------------------------------------------------------*/
static void ad(void)
{
    do{                         /* ターゲットのサンプリング終了を待つ */
        dsp_get(&m_flag,flag_adr,(ushort)2);
    }while(m_flag == DSP_WORKING);

                                    /* ＤＳＰインターフェースライブラリー */
    dsp_get(&m_spot,spot_adr,(ushort)4);
    dsp_get(&m_intensity,intensity_adr,(ushort)4);
                                    /* ＤＳＰのメモリーから値を読み込む */

    m_flag = DSP_WORKING;           /* サンプリング再会の許可 */
    dsp_put(flag_adr,&m_flag,2);
}

/*-----------------------------------------------------------------------+
```

関数名　　static void output(void)

コメント　　センサーの値を表示する。

```
+-----------------------------------------------------------------------*/
static void output(int *i,float x[],float y[])
{                    /* ver.1.01 */
    WINlocate(&buf,1,2);                            /* 望洋ライブラリー */
    WINdprintf(&buf," Position  : % 7.4f[v]¥n",m_spot);
    WINdprintf(&buf," Intensity : % 7.4f[v]¥n¥n",m_intensity);
    WINputstr(&buf, "  ＥＳＣ : 終了");

    x[*i] = (float)(*i * samp);
    y[*i] = m_spot;
    if(Draw(0,0,&x[*i],&y[*i]) == 1){
        Axis();
        *i = -1;
    }
    (*i)++;
}

void AutoGraph(int sta,int end,float x[],float y[]) /* ver.1.01 */
{
```

```
    Autoscale(sta,end,x,y);
    Autoscalepoint(sta,end,x,y);
    Axis();
    Draw(sta,end,x,y);
}

/*----------------------------------------------------------------------+

    関数名      static int start_job(int argc,char *argv[])

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+----------------------------------------------------------------------*/
static int start_job(int argc,char *argv[])
{
    short flag;

    tinit();                        /* 望洋ライブラリーの初期化 */
    init_dsp();
    flag = arg(argc,argv);
    if(flag){                       /* コマンドラインからたち上げた場合 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    set_senser_param();
    set_graph_param();              /* ver.1.0 */
    Init();                         /* グラフィック画面の初期化 ver.1.0 */
    disp_scale();                   /* ver.1.0 */
    set_popup_window();
    dsp_run();                      /* ＤＳＰインターフェースライブラリー */
    return(flag);
}

/*----------------------------------------------------------------------+

    関数名      int end_job(void)

    コメント      ウインドウを閉じる。

+----------------------------------------------------------------------*/
void end_job(short flag)
{
    WINclose(&buf);                 /* 望洋ライブラリー */
    End();                          /* グラフィックの終了 ver.1.0 */
    if(flag){                       /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

/*----------------------------------------------------------------------+

    関数名      int main(int argc,char *argv[])

+----------------------------------------------------------------------*/
void main(int argc,char *argv[])
```

```
{
    short cmd_line; /* 1: コマンドラインからの立ち上げ */
    static int i,key_no;            /* ver.1.01 */
    static float x[512],y[512];     /* ver.1.01 */

    cmd_line = start_job(argc,argv);        /* 初期化関数の呼び出し */

    do{
        ad();                               /* ＡＤ変換値の取得 */
        output(&i,x,y);                     /* ＡＤ変換値の表示ver.1.01*/
        if(kbhit() != 0)                    /* キー入力のチェック */
            key_no = getch();
    }while(key_no != ESC && key_no != 'e');     /* ver.1.01 */
    m_flag = DSP_END;               /* ターゲットにプログラム終了を伝える */
    dsp_put(flag_adr,&m_flag,2);
    AutoGraph(0,i,x,y);                     /* グラフ再描画 ver.1.01 */
    disp_scale();
    getch();

    end_job(cmd_line);                      /* 終了処理 */
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              t_sens10.c                               |
|                            --------------                             |
|                                                                       |
|     入力の１（ポジション情報）、２（入射光量）チャンネルの値を読む    |
|   プログラム                                          Ver.1.02        |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <libap.h>
#include "lorypio.h"
#include "afm_adda.h"
#include "afm_def.h"

/*
+-----------------------------------------------------------------------+
|                            グローバル変数                             |
+-----------------------------------------------------------------------+
*/
short   t_flag = DSP_WORKING;
long    t_times = 0, t_average = 1000;
float   t_spot, t_intensity;

/*
+-----------------------------------------------------------------------+
|                                 コード                                |
+-----------------------------------------------------------------------+
*/
#include "afm_func.h"

/*-----------------------------------------------------------------------+

    関数名      void main()

    コメント      特定の時間間隔でｄｓｐの１、２チャンネルから
                サンプリングする

+-----------------------------------------------------------------------*/
void main()
{
    long  wait_times;

    if((wait_times = t_times - t_average * 2) < 0){        /* ver.1.02 */
        t_average = (long)(t_average / 5);
        wait_times = t_times - t_average * 2;
    }
    osc_start(osc_100KHZ);      /* クロックボードを最大値に設定       */
    while(1){                           /* サンプリングのための無限ループ    */
        if(t_flag == DSP_WORKING){      /* ホストがサンプリングを許可した場合*/
            wait(wait_times);           /* 上記のクロック数でt_times回待つ  */
            t_spot = adaverage(1,t_average);        /* １ＣＨのＡＤの平均   */
            t_intensity = adaverage(2,t_average);   /* ２ＣＨのＡＤの平均   */
            ieee32((int)1,&t_spot);         /* ホストコンピュータの         */
            ieee32((int)1,&t_intensity);    /*      変数フォーマットに変換  */

            t_flag = DSP_COMPLETE;      /* サンプリング終了を宣言            */
```

```
        }
        if(t_flag == DSP_END)        /* ホストが終了を命じたときループを抜ける*/
            break;
    }
}
```

```
/*
+-----------------------------------------------------------------------+
|                              t_sensor.c                               |
|                            --------------                             |
|                                                                       |
|     入力の１（ポジション情報）、２（入射光量）チャンネルの値を読む    |
|   プログラム                                                          |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <libap.h>
#include "lorypio.h"
#include "afm_adda.h"
#include "afm_def.h"

/*
+-----------------------------------------------------------------------+
|                            グローバル変数                             |
+-----------------------------------------------------------------------+
*/
short   t_flag = DSP_WORKING;
long    t_times = 0, t_average = 1000;
float   t_spot, t_intensity;

/*
+-----------------------------------------------------------------------+
|                                コード                                 |
+-----------------------------------------------------------------------+
*/
#include "afm_func.h"

/*-----------------------------------------------------------------------+

    関数名      void main()

    コメント      特定の時間間隔でｄｓｐの１、２チャンネルから
                サンプリングする

+-----------------------------------------------------------------------*/
void main()
{
    while(1){                           /* サンプリングのための無限ループ    */
        if(t_flag == DSP_WORKING){      /* ホストがサンプリングを許可した場合*/
            osc_start(osc_1KHZ);        /* クロックボードを１ＫＨｚに設定    */
            wait(t_times);              /* 上記のクロック数でt_times回待つ   */
            osc_start(osc_200KHZ);      /* クロックボードを最大値に設定      */
            t_spot = adaverage(1,t_average);/* １ＣＨのＡＤ変換の平均を得る  */
            t_intensity = adaverage(2,t_average);   /* ２ＣＨのＡＤの平均    */
            ieee32((int)1,&t_spot);     /* ホストコンピュータの変数フォーマット
                                                                  に変換    */
            ieee32((int)1,&t_intensity);
            t_flag = DSP_COMPLETE;      /* サンプリング終了を宣言            */
        }
        if(t_flag == DSP_END)       /* ホストが終了を命じたときループを抜ける*/
            break;
    }
}
```

プログラムファイル

(B.4 Motor Program Files)

```
/*
+-----------------------------------------------------------------------+
|                              m_motr10.c                               |
|                            --------------                             |
|                                                                       |
|        ステッパーモーター制御プログラム、M_MOTOR. EXEの              |
|      メインモジュール              Ver.1.0 (modified by KOTARO)        |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <stdlib.h>          /* atoi() */
#include <string.h>          /* strcpy(),strcat() */
#include <conio.h>           /* kbhit() */
#include <lib¥crt.h>
#include <lib¥key.h>
#include <lib¥bslib.h>
#include <lib¥keytbl.h>
#include <lib¥window.h>

#include "mttdsp.h"
#include "txcondef.h"
#include "popup.h"
#include "afm_def.h"
#include "bgrallex.h"

#define BADRS   0xD2
#define TARGET  "a:¥¥afm¥¥afm_menu¥¥sensor¥¥t_sensor.out"

/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
       void init_dsp(void);
static int  arg(int,char *[]);
       void error(short,char *);                    /* ver.1.0 */
static void set_motor_param(short);
       void init_motor(void);
static void set_popup_window(void);
static void set_motor_screen(void);
       void disp_scale(void);
static void ad(void);
static void output(int,int,int);
       void force_curve(void);
       int  ask_cntinue(void);
       void motor_drv(void);
static int  start_job(int,char *[]);
       void end_job(short);
/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                          short start,short stop);
extern void errer_msg(char *);
```

```
extern void c_ppd2a(void);
extern void step(int,int);
extern long c_apd2a(void);
extern void c_spd2a(void);
extern void dataset0(void);
/*
+-----------------------------------------------------------------------------+
|                              グローバル変数                                 |
+-----------------------------------------------------------------------------+
*/
static char     prm_file_name[80];
static ulong flag_adr,times_adr,average_adr,spot_adr,intensity_adr;
       short m_flag = 1,base_speed,sampling_step,direction,auto_stop,escape,
       mode;
static long  m_times,m_average;
static float wait_sec,m_spot,m_intensity,return_force,x[500],y[500],
               ret_x[500],ret_y[500],lever_index,lever_eras,engage_vol;

static WINDOW buf,buf2;

FILE *fp,*fp_eng;

/*
+-----------------------------------------------------------------------------+
|                                  コード                                     |
+-----------------------------------------------------------------------------+
*/

/*-----------------------------------------------------------------------------+

    関数名      static void init_dsp(void)

    コメント      ｄｓｐの初期化と、ターゲットプログラムのダウンロード、
                共通変数のアドレスの取得

+-----------------------------------------------------------------------------*/
void init_dsp(void)
{
    dsp_init(BADRS,TARGET);
    dsp_sym(flag_adr,"t_flag");
    dsp_sym(spot_adr,"t_spot");
    dsp_sym(intensity_adr,"t_intensity");       /* ver.1.0 */
}

/*-----------------------------------------------------------------------------+

    関数名      static int arg(int argc,char *argv[])

    返値        １：コマンドラインから　０：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+-----------------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[20][40],imput_from_key[20];
    int counter;
```

```
                                /* デフォルトの値の代入 */
strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                /* パラメータファイル名 */
strcpy(arguments[2],"4");       /* 4  : ジョグで針を上げる */
strcpy(arguments[3],"-1");      /* -1 : 停止ステップ数未指定 */

for(counter = 1;counter < argc;counter++){
    strcpy(arguments[counter - 1],argv[counter]);
}

if(argc <= 3){
    WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                    POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
    WINputstr(&buf,"デフォルト : リターンキー¥n");
    Cslon();
    while(1){
        switch(argc){
        case 1:
        case 2:
        case 3:
            WINlocate(&buf,1,1);
            WINputstr(&buf,
            "アプローチ:1   ウイズドロー:2   フォースカーブ:3¥n"
            "ジョグで針を上げる:4   針を下げる:5   一時退避:6  ");
            WINdprintf(&buf,"%s  ",arguments[2]);
            gets(imput_from_key);
            WINputstr(&buf,"¥n");
            if(*imput_from_key != 0){
                strcpy(arguments[2],imput_from_key);
            }
        }

        /* ウイズドロー : 2  の場合 */
        if(arguments[2][0] == '2'){
            WINputstr(&buf,"ウイズドローステップ数[ｎｍ]  ");
            WINdprintf(&buf,"%s  ",arguments[3]);
            gets(imput_from_key);
            WINputstr(&buf,"¥n");
            if(*imput_from_key != 0){
                strcpy(arguments[3],imput_from_key);
            }
        }

        WINputstr(&buf,"¥n以上でよろしいですね    ");
        gets(imput_from_key);
        if(imput_from_key[0] == 0){
            WINclose(&buf);
            break;
        }
    }
    Csloff();
}
strcpy(prm_file_name,arguments[1]);
mode        = atoi(arguments[2]);
escape      = atoi(arguments[3]);
```

```
    if(escape != -1)
        escape = abs(escape);           /* 絶対値だけ引き離す */
    return( atoi(arguments[0]) );
}

void error(short cmdlflag,char *fname)          /* ver.1.0 */
{
    char msg[200];

    strcpy(msg,fname);
    strcat(msg,"¥nがオープンできません");
    errer_msg(msg);
    putchar('¥a');
    end_job(cmdlflag);
}

/*-----------------------------------------------------------------------+

    関数名      static void set_motor_param()

    コメント      パラメータファイルからセンサーのパラメータを読み込む。
                モードがアプローチの場合とジョグの場合、ウインドウを開き、
                デフォルトの値の変更をたずねる。

+-----------------------------------------------------------------------*/
static void set_motor_param(short cmdlflag)
{
    char parameter[40][40],engage_name[80],buffer[40];
    int number_of_prm;

    number_of_prm = read_prm_file(prm_file_name,parameter,SCAN_PART);
    if(number_of_prm == -1) end_job(cmdlflag);

    lever_index = (float)atof(parameter[14 + number_of_prm]);
    lever_eras  = (float)atof(parameter[15 + number_of_prm]);

    number_of_prm = read_prm_file(prm_file_name,parameter,MOTOR_PART);
    if(number_of_prm == -1) end_job(cmdlflag);

    switch(mode){                                       /* ver.1.0 */
    case APPROACH:      /* アプローチ */
        /* ----ファイルオープン---- */
        ask_prm(&buf,"データファイル名",parameter,number_of_prm,1,1);
        strcpy(engage_name,parameter[0 + number_of_prm]);
        strcat(engage_name,parameter[2 + number_of_prm]);
        if((fp_eng = fopen(engage_name,"wt")) == NULL)
            error(cmdlflag,engage_name);
        strcat(parameter[0 + number_of_prm],parameter[1 + number_of_prm]);
        if((fp = fopen(parameter[0 + number_of_prm],"wt")) == NULL)
            error(cmdlflag,parameter[0 + number_of_prm]);
        /* ----モーター駆動パラメータ---- */
        base_speed      = atoi(parameter[5 + number_of_prm]);
        sampling_step   = atoi(parameter[6 + number_of_prm]);
        direction       = MOTOR_DOWN;
        auto_stop       = YES;
        /* ----グラフパラメータ---- */
        gparm.scale.x = 0.08;                   /* ｘ軸５０００ｎｍ */
        gparm.dsp.x = 500;                      /* ｘ軸５００nmごとに目盛 */
```

```
    break;
case WITHDRAW:      /* ウイズドロー */
    /* ----ファイルオープン---- */
    ask_prm(&buf,"データファイル名",parameter,number_of_prm,3,3);
    strcat(parameter[0 + number_of_prm],parameter[3 + number_of_prm]);
    if((fp = fopen(parameter[0 + number_of_prm],"wt")) == NULL)
        error(cmdlflag,parameter[0 + number_of_prm]);
    /* ----モーター駆動パラメータ---- */
    base_speed      = atoi(parameter[7 + number_of_prm]);
    sampling_step   = atoi(parameter[8 + number_of_prm]);
    direction       = MOTOR_UP;
    auto_stop       = NO;
    /* ----グラフパラメータ---- */
    gparm.scale.x = 0.08;                   /* ｘ軸５０００ｎｍ */
    gparm.dsp.x = 500;                      /* ｘ軸５００nmごとに目盛 */
    break;
case FORCE_CURVE:   /* フォースカーブ */
    /* ----ファイルオープン---- */
    ask_prm(&buf,"データファイル名",parameter,number_of_prm,4,4);
    strcpy(engage_name,parameter[0 + number_of_prm]);
    strcat(engage_name,parameter[2 + number_of_prm]);
    if((fp_eng = fopen(engage_name,"rt")) == NULL)
        error(cmdlflag,engage_name);
    fscanf(fp_eng,"%s",buffer);
    engage_vol = (float)atof(buffer);
    strcat(parameter[0 + number_of_prm],parameter[4 + number_of_prm]);
    if((fp = fopen(parameter[0 + number_of_prm],"wt")) == NULL)
        error(cmdlflag,parameter[0 + number_of_prm]);
    /* ----モーター駆動パラメータ---- */
    base_speed      = atoi(parameter[9 + number_of_prm]);
    sampling_step   = atoi(parameter[10 + number_of_prm]);
    return_force    = (float)atof(parameter[11 + number_of_prm]);
    auto_stop       = NO;
    /* ----グラフパラメータ---- */
    gparm.scale.x = 0.8 / sampling_step;    /* ｘ軸 */
    gparm.scale.y = 15.0;                   /* ｙ軸±１０ｎＮ */
    gparm.dsp.x = 100.0;                    /* ｘ軸１００[nm]ごとに目盛 */
    gparm.dsp.y = 1.0;                      /* ｙ軸１[nN]毎に目盛 */
    gparm.unit.y = "[nN]";                  /* ｙ軸の単位[nN] */
    break;
case JOG_UP:        /* ジョグで針を上げる */
    /* ----モーター駆動パラメータ---- */
    ask_prm(&buf,"ジョグ条件",parameter,number_of_prm,12,13);
    base_speed      = atoi(parameter[12 + number_of_prm]);
    sampling_step   = atoi(parameter[13 + number_of_prm]);
    direction       = MOTOR_UP;
    auto_stop       = NO;
    break;
case JOG_DOWN:      /* 針を下げる */
    /* ----モーター駆動パラメータ---- */
    ask_prm(&buf,"ジョグ条件",parameter,number_of_prm,12,13);
    base_speed      = atoi(parameter[12 + number_of_prm]);
    sampling_step   = atoi(parameter[13 + number_of_prm]);
    direction       = MOTOR_DOWN;
    auto_stop       = NO;
    break;
case TEMP_ESC:      /* 一時退避 */
    /* ----モーター駆動パラメータ---- */
```

```
        base_speed      = atoi(parameter[14 + number_of_prm]);
        sampling_step   = atoi(parameter[15 + number_of_prm]);
        direction       = MOTOR_UP;
        auto_stop       = NO;
        escape          = atoi(parameter[16 + number_of_prm]);
        /* ----グラフパラメータ---- */
        gparm.scale.x = 0.08;                   /* ｘ軸５０００ｎｍ */
        gparm.dsp.x = 500;                      /* ｘ軸５００nmごとに目盛 */
        break;
    }
}

/*-----------------------------------------------------------------------+

    関数名      void init_motor(void)

    コメント      モーターの現在位置のクリアとパラメータのセット、
                ＨＰ９８－ＰＰＤ２ボードの初期化

+-----------------------------------------------------------------------*/
void init_motor(void)
{
    dataset0();
    c_ppd2a();
}

/*-----------------------------------------------------------------------+

    関数名      static void set_popup_window(void)

    コメント      センサーの値を表示するウインドウを開く。

+-----------------------------------------------------------------------*/
static void set_popup_window(void)
{
    WINopen(&buf,POPUP4_X1,POPUP4_Y1,POPUP4_X2,POPUP4_Y2,POPUP_SWITCH,
                    POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"***測定中***");
    WINlocate(&buf,1,1);
}

/*-----------------------------------------------------------------------+

    関数名      static void set_motor_screen(void)

    コメント      モータードライブの基本画面を表示する。

+-----------------------------------------------------------------------*/
static void set_motor_screen(void)
{
    set_popup_window();
    Axis();
    Set_color(2);
}
/*-----------------------------------------------------------------------+

    関数名      void disp_scale(void)
```

コメント　　座標軸の１目盛の値と単位を表示

```
+-----------------------------------------------------------------------*/
void disp_scale(void)
{
    Locate(1,1);
    printf("縦軸１目盛　%4.2lf%s　　横軸１目盛　%4.0lf%s",
                    gparm.dsp.y,gparm.unit.y,gparm.dsp.x,gparm.unit.x);
}

/*-----------------------------------------------------------------------+

    関数名　　static void ad(void)

    コメント　　センサーの値を取得する。

+-----------------------------------------------------------------------*/
static void ad(void)
{
    do{                         /* ターゲットのサンプリング終了を待つ */
        dsp_get(&m_flag,flag_adr,(ushort)2);
    }while(m_flag == DSP_WORKING);

                                /* ＤＳＰインターフェースライブラリー */
    dsp_get(&m_spot,spot_adr,(ushort)4);
    dsp_get(&m_intensity,intensity_adr,(ushort)4);  /* ver.1.0 */
                                /* ＤＳＰのメモリーから値を読み込む */

    m_flag = DSP_WORKING;       /* サンプリング再会の許可 */
    dsp_put(flag_adr,&m_flag,2);
}

/*-----------------------------------------------------------------------+

    関数名　　static void output(void)

    引数　　no: 何番目のサンプリングか
            go: １：行きのサンプリング
                ０：フォースカーブ測定時の帰りのサンプリング

    コメント　　センサーの値を、フォースカーブ測定時は変更し、表示する。

+-----------------------------------------------------------------------*/
static void output(int no,int stano,int go)
{
    long abs;
    int  exflag;

    abs = c_apd2a();

    WINcls(&buf);
    WINlocate(&buf,1,1);
    WINdprintf(&buf,"   Steps   : % 5ld¥n",abs);
    WINdprintf(&buf," Position  : % 6.4f[v]¥n",m_spot);          /* ver.1.0 */
    WINdprintf(&buf," Intensity : % 6.4f[v]¥n¥n",m_intensity);   /* ver.1.0 */
    WINputstr(&buf, " ＥＳＣ: 終了¥n");

    if(mode == FORCE_CURVE){                /* フォースカーブ */
```

```
        if(go){     /* 行きのサンプリング */
            x[no] = (float)(no * sampling_step);
            y[no] = (m_spot - engage_vol) * lever_index * lever_eras;
                /* 電位を力に変換 */
            if((exflag = Draw(no,no,x,y)) != 0){
                switch(exflag){
                case 1 :
                    gparm.scale.x *= 0.75;  break;
                case 2 :
                    gparm.scale.y *= 0.75;  break;
                case 3 :
                    gparm.scale.x *= 0.75;  gparm.scale.y *= 0.75;  break;
                default:    break;
                }
                Axis(); Draw(no,stano,x,y);
            }
            fprintf(fp,"%f,%f¥n",x[no],y[no]);          /* ver.1.0 */
        }
        else{       /* 帰りのサンプリング */
            ret_x[no] = (float)(no * sampling_step);
            ret_y[no] = (m_spot - engage_vol) * lever_index * lever_eras;
                /* 電位を力に変換 */
            if((exflag = Draw(no,no,ret_x,ret_y)) != 0){
                switch(exflag){
                case 1 :
                    gparm.scale.x *= 0.75;  break;
                case 2 :
                    gparm.scale.y *= 0.75;  break;
                case 3 :
                    gparm.scale.x *= 0.75;  gparm.scale.y *= 0.75;  break;
                default:    break;
                }
                Axis(); Set_color(2);   Draw(stano,499,x,y);
                        Set_color(3);   Draw(stano,no,ret_x,ret_y);
            }
            fprintf(fp,"%f,%f¥n",ret_x[no],ret_y[no]);      /* ver.1.0 */
        }
    }
    else{
        x[no] = (float)(no * sampling_step);
        y[no] = m_spot;
        switch(Draw(no,no,x,y)){
        case 1 :
            gparm.scale.x *= 0.75;  Axis(); Draw(stano,no,x,y); break;
        case 2 :
            gparm.scale.y *= 0.75;  Axis(); Draw(stano,no,x,y); break;
        case 3 :
            gparm.scale.x *= 0.75;  gparm.scale.y *= 0.75;
            Axis(); Draw(stano,no,x,y); break;
        default:    break;
        }
        if(mode == APPROACH || mode == WITHDRAW)/* エンゲージ、ウイズドロー */
            fprintf(fp,"%ld,%f¥n",abs,m_spot); /* データをファイルに書き込む*/
    }
}

/*-----------------------------------------------------------------------+
```

関数名　　void force_curve(void)

コメント　　フォースカーブ測定

```
+-----------------------------------------------------------------------*/
void force_curve(void)
{
    int i,j,key_no;

    for(i = 499;i >= 0;i--){        /* 行きのサンプリング */
        if(kbhit() != 0){                       /* キー入力処理 */
            key_no = getch();
            if(key_no == ESC || key_no == 'e')  /* 終了 */
                break;
        }
        step(MOTOR_DOWN,sampling_step);         /* モーターのドライブ */
        ad();
        Set_color(2);
        output(i,499,1);
        if(y[i] > return_force)
            break;
    }
    if( i < 0 ) i = 0;
    for(j = i ;j <= 499;j++){   /* 帰りのサンプリング、ｉの初期化なし */
        if(kbhit() != 0){                       /* キー入力処理 */
            key_no = getch();
            if(key_no == ESC || key_no == 'e')  /* 終了 */
                break;
        }
        step(MOTOR_UP,sampling_step);           /* モーターのドライブ */
        ad();
        Set_color(3);                   /* 帰りは色を水色に */
        output(j,i,0);
    }
    if(j >= 500)    j = 499;
    Autoscale(i,j,ret_x,ret_y);
    Autoscalepoint(i,j,ret_x,ret_y);
    Axis();
    Set_color(2);
    Draw(i,499,x,y);
    Set_color(3);
    Draw(i,j,ret_x,ret_y);
}

/*-----------------------------------------------------------------------+
```

関数名　　int ask_cntinue(void)

返値　　０：ｙｅｓ　１：ｎｏ

コメント　　コンティニュウするか尋ねる

```
+-----------------------------------------------------------------------*/
int ask_cntinue(void)
{
    int key_no;

    Cslon();
```

```
    WINopen(&buf2,POPUP3_X1,POPUP3_Y1,POPUP3_X2,POPUP3_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,POPUP_FRAME_COLOR2);
    WINtitle(&buf2,0,0,POPUP_TITLE_COLOR,"コンティニュー？");
    WINputstr(&buf2,     "動作を継続しますか？¥n"
                    "     <y, n>       ¥n");
    key_no = getch();
    WINclose(&buf2);
    Csloff();
    if(key_no == 'Y' || key_no == 'y')  /* 継続 */
        return(0);
    else
        return(1);
}

/*-----------------------------------------------------------------------+

    関数名      int motor_drv(void)

    コメント      モータードライブ

+-----------------------------------------------------------------------*/
void motor_drv(void)
{
    int i,end = 0;

    while(1){
        for(i=0;i<=499;i++){ /* 最大変位を500*sumpling_step[nm]以内に設定 */
            if(kbhit() != 0){                   /* キー入力処理 */
                int key_no;

                key_no = getch();
                if(key_no == ESC || key_no == 'e'){
                    goto END;                   /* コンティニュウなしで終了 */
                }
            }
            step(direction,sampling_step);      /* モーターのドライブ */
            if(escape != -1){                   /* 退避ステップが指定してある*/
                static int all_steps = 0;
                all_steps += sampling_step;
                if(all_steps > escape){         /* 指定ステップ数を越えた */
                    goto END;                   /* コンティニュウなしで終了 */
                }
            }
            ad();
            output(i,0,1);
            if(auto_stop != 1 && direction == MOTOR_DOWN){
                if(i > 5){                      /* 最低５ステップは動かす */
                    if( (y[i] - y[i-1]) * lever_index / sampling_step
                                    > (float)MOTOR_STEP*0.8 ){
                        WINlocate(&buf,1,5);            /* ver.1.0 */
                        WINputstr(&buf, "エンゲージ      ¥n");
                        fprintf(fp_eng,"%f¥n",y[i - 5]);
                        /* ジャンプインより前の値をエンゲージ電圧とする */
                        goto END;               /* コンティニュウなしで終了 */
                    }
                }
            }
        }
```

```
        if( ask_cntinue() )     /* コンティニュウするか ? */
            break;              /* コンティニュウしない */
    }
    if(i>=500)  i = 499;
    END:
    Autoscale(0,i,x,y);
    Autoscalepoint(0,i,x,y);
    Axis();
    Draw(0,i,x,y);
}

/*-----------------------------------------------------------------------+

    関数名      static void start_job(int argc,char *argv[])

    コメント        初期化の関数をコールする

+-----------------------------------------------------------------------*/
static int start_job(int argc,char *argv[])
{
    short flag;

    tinit();                        /* 望洋ライブラリー使用宣言 */
    Init();                         /* グラフ描画の初期化 */
    init_dsp();

    flag = arg(argc,argv);
    if(flag){
        Csloff();
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    set_motor_param(flag);
    init_motor();
    dsp_run();
    set_motor_screen();
    disp_scale();
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      void end_job(void)

    コメント        ウインドウを閉じ、オープンされたファイルがあれば閉じる

+-----------------------------------------------------------------------*/
void end_job(short flag)
{
    long abs;

    disp_scale();
    abs = c_apd2a();

    getch();

    WINclose(&buf);
    Cls();
    End();                          /* グラフ描画の終了 */
```

```
    if(mode == APPROACH || mode == FORCE_CURVE){/* アプローチの場合 ver.1.0 */
        fclose(fp);
        fclose(fp_eng);
    }else if(mode == WITHDRAW){     /* ウイズドローの場合 */
        fclose(fp);
    }
    if(flag){                       /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit((int)abs);                 /* 総ステップ数を親プロセスに返す */
}


void main(int argc,char *argv[])
{
    short cmd_line;

    cmd_line = start_job(argc,argv);

    if(mode == FORCE_CURVE)         /* フォースカーブ */
        force_curve();
    else                            /* その他 */
        motor_drv();

    end_job(cmd_line);
}
```

```
/*
+-----------------------------------------------------------------------+
|                              motorfnc.c                               |
|                            --------------                             |
|                                                                       |
|        ステッパーモーター制御プログラム、M_MOTOR. EXEの              |
|      サブモジュール。（株）ハイバーテックのユーザーズマニュアル参照   |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <stdlib.h>
#include "txcondef.h"

#define portadr 0x0dd0      /** PORT ADDRESS **/

/*
+-----------------------------------------------------------------------+
|                            ライブラリー関数                           |
+-----------------------------------------------------------------------+
*/
extern  void    ppd2a();
extern  int     cpd2a();
extern  int     spd2a();
extern  void    apd2a();
/*
+-----------------------------------------------------------------------+
|                              外部変数                                 |
+-----------------------------------------------------------------------+
*/
extern  short   base_speed;
/*
+-----------------------------------------------------------------------+
|                           グローバル変数                              |
+-----------------------------------------------------------------------+
*/
char    scb[100];
char    scl[10];

unsigned int    intrpt;

struct {
    char    ctrl;       /** CTRL  ***/
    char    xmp;        /** X軸倍率**/
    unsigned int    xtime;  /** X軸時定数**/
    unsigned int    xbase;  /** X軸ベース速度**/
    unsigned int    xcreap; /** X軸クリープ速度**/
    char    xbaklsh;    /** X軸バックラッシュ**/
} ipram;

struct {
    long    actincx;
    long    actabsx;
} actualx;

char    mode[20];
char    scll[10];
```

```
/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名       void clear_pos(void)

    コメント       現在位置のクリアー

+-----------------------------------------------------------------------*/
void clear_pos(void)
{
    actualx.actincx=0l;
    actualx.actabsx=0l;
}

/*-----------------------------------------------------------------------+

    関数名       void dataset0(void)

    コメント       X軸パラメータの設定

+-----------------------------------------------------------------------*/
void dataset0(void)
{
    clear_pos();

    intrpt=0xffff;       /*X軸、Y軸割り込み未使用 */
    ipram.ctrl=0x4c;     /*X軸、原点復帰マイナス方向、割り込み無し*/
    ipram.xmp=0x00;      /*倍率 (0:1 ,1:2 ,2:4 ,3:5 ,4:10 ,5:20 ,6:25,7:30)**/
    ipram.xtime=6500;    /*時定数    ***/
    ipram.xbase=10;      /*R 1    *******/
    ipram.xcreap=10;     /*クリープ   ************/
    ipram.xbaklsh=0;     /*バックラッシュ*************/

    ipram.xbase=base_speed;      /** X軸ベース速度  **/
}

/*-----------------------------------------------------------------------+

    関数名       void c_ppd2a(void)

    コメント       初期化パラメータのセット

+-----------------------------------------------------------------------*/
void c_ppd2a(void)  /** ppd2a(intrpt,boardadr,&ipram[0],&scb[0]) **/
{
    int i,j;

    i=intrpt;j=portadr;
    ppd2a(i,j,&ipram,&scb[0]);
}

/*-----------------------------------------------------------------------+

    関数名       void c_spd2a(void)
```

コメント　　ハンドラーステータスの取り込み

```
+-----------------------------------------------------------------------*/
void c_spd2a(void)
{
    unsigned  char  a;
    int stat,i;
    Locate(1,1);
    Cll();
    printf("  ステータスの結果を取り出す            spd2a     ¥n");
    printf(" ――――――――――――――――――――――――――――――― ¥n");
    Locate(4,2);
    i=portadr;stat=spd2a(i,&scb[0]);

    printf("   処理結果： %x¥n",stat);
    a=scb[88];
    printf("   動作終了割り込みステータス： %x¥n",a);
    a=scb[20];
    printf("   サーボアラーム，偏差カウンタステータス： %x¥n",a);
    gets(mode);
}

/*-----------------------------------------------------------------------+
```

関数名　　void step(int direc , int step_n)

コメント　　ジョグコマンドの実行

```
+-----------------------------------------------------------------------*/
void step(int direc , int step_n)
{
    int ip,stat;

    ip=portadr;

    scl[0] = 0x34 + direc;
    scl[1] = (char)step_n;

    if( cpd2a(ip,&scl[0],&scb[0]) ){        /***コマンド実行 ****/
        printf("cpd2a 0x34 error¥n");
        c_spd2a();
        exit(1);
    }

    for(;;){
        if( (stat = spd2a(ip,&scb[0]) ) == 0x10){   /**コマンド終了を待つ**/
            return;
        }
    }
}

/*-----------------------------------------------------------------------+
```

関数名　　long c_apd2a(void)

コメント　　現在位置の取り込み

```
+-----------------------------------------------------------------------*/
long c_apd2a(void)
{
    int i;
    long    absx,increx;    /*** アブソリュート，インクリメンタル **/
    i=portadr;apd2a(i,&scb[0],&actualx);
    increx=actualx.actincx;
    absx=actualx.actabsx;
    return(absx);
}
```

プログラムファイル

(B.5 Scan Program Files)

```
/*
+-----------------------------------------------------------------------+
|                              scan_m26.c                               |
|                            --------------                             |
|                                                                       |
|              M_SCAN. EXEのメインモジュール。                            |
|                                    ver. 2. 6(debug by KOTARO)         |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <conio.h>          /* kbhit() */
#include <stdlib.h>         /* atoi(), atol(), exit() */
#include <math.h>           /* atof() */
#include <string.h>         /* strcpy() */
#include <lib¥crt.h>        /* tinit(), tterm() */
#include <lib¥window.h>     /* winopen() */

#include "b:¥dsp32sl¥dspif¥mttdsp.h"
#include "afm_def.h"
#include "popup.h"
#include "txcondef.h"
#include "video.h"

#define BADRS       0xD2
#define TARGET      "t_scan46.out"  /* ver.2.6 */
/*
+-----------------------------------------------------------------------+
|                            グローバル変数                               |
+-----------------------------------------------------------------------+
*/
short   m_div_times[3]/*,m_div_times_z*/,m_warm_up,m_both,  /* by KOTARO */
        m_start[2],m_scan_size,m_number_of_samples,m_flag = 0,m_mode,
        /* m_start[3] -> m_start[2] by KOTARO */
        m_damy[2] = {0,0},m_either,m_interval/* ver.2.6 */;
long    m_time,m_average;
float   m_lever_index,m_original_spot,m_zpiezo_index,m_near/*ver.2.4*/,
        m_height[SAMPLING_MAX],t_rev[SAMPLING_MAX],
        m_subdat[SAMPLING_MAX]/* ver.2.5 */;

ulong   div_times_adr/*,div_times_z_adr*/,warm_up_adr,both_adr,/*by KOTARO*/
        start_adr,scan_size_adr,number_of_samples_adr,flag_adr,mode_adr,
        damy_adr,
        time_adr,average_adr,
        /*senser_range_adr,*/ zpiezo_index_adr,near_adr,/* by KOTARO */
        lever_index_adr,original_spot_adr,
        height_adr[2],rev_adr[2],subdat_adr[2],subrev_adr[2],/* ver.2.6 */
        either_adr,interval_adr/* ver.2.6 */;

char    prm_file_name[80];
short   file_type,height_range;
float   scan_force,lever_eras,engage_vol;

struct dat_file_format{
    int     what_file;
    int     number_of_samples;
    int     scan_size;
    float   tilt_x;
```

```
    float   tilt_y;
    float   max_dat;
    float   min_dat;
    int     start[2];
} dat_file;

WINDOW  buf;

FILE *fp,*fp_eng,    /* スキャンデータと、エンゲージ電圧のファイルポインタ */
     *subfp /* ver.2.5 */;
/*
+-----------------------------------------------------------------------+
|                    プロトタイプ宣言                                   |
+-----------------------------------------------------------------------+
*/
static  void init_dsp(void);
static  int  arg(int,char *[]);
        void open_error(short,short,char *);    /* ver.2.5 */
static  void set_scan_param(short);
        void get_rev_dat(void);
        void get_height_dat(void);
static  void init_file(FILE *);                 /* ver.2.5 */
        void paint(short);
static  void filewrite(FILE *,float []);        /* ver.2.5 */
        int  start_job(int,char *[]);
        void end_job(short);
        int  keyin(void);                       /* ver.2.6 */
/*
+-----------------------------------------------------------------------+
|                                外部関数                               |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                          short start,short stop);
extern void cls_256(void);
extern void write_prm_to_file(FILE *,struct dat_file_format *);
/*extern int    read_prm_file(char *,char [][40],int);*/
/*extern int    ask_prm(WINDOW *,char *,char [][40],short,short,short);*/
extern void errer_msg(char *);
extern void set_scan_screen(int);
extern void scan_no(int);
extern void ber_scale(void);
extern void dot_set(int,int);
extern void repaint(void);
/*
+-----------------------------------------------------------------------+
|                                コード                                 |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      static void init_dsp(void)

    コメント      ｄｓｐの初期化と、ターゲットプログラムのダウンロード、
                共通変数のアドレスの取得

+-----------------------------------------------------------------------*/
```

```
static void init_dsp(void)
{
    dsp_init(BADRS,TARGET);
    dsp_sym(div_times_adr,"t_div_times");           /*          */
    /*dsp_sym(div_times_z_adr,"t_div_times_z");*/   /* by KOTARO */
    dsp_sym(warm_up_adr,"t_warm_up");               /*          */
    dsp_sym(start_adr,"t_start");
    dsp_sym(scan_size_adr,"t_scan_size");
    dsp_sym(number_of_samples_adr,"t_number_of_samples");
    dsp_sym(time_adr,"t_time");
    dsp_sym(flag_adr,"t_flag");
    dsp_sym(mode_adr,"t_mode");
    dsp_sym(original_spot_adr,"t_original_spot");
    dsp_sym(zpiezo_index_adr,"t_zpiezo_index");     /* by KOTARO */
    dsp_sym(lever_index_adr,"t_lever_index");
    dsp_sym(damy_adr,"t_damy");
    dsp_sym(average_adr,"t_average");
    dsp_sym(both_adr,"t_both");
    dsp_sym(near_adr,"t_near");                     /* ver.2.4 */
    dsp_sym(height_adr[0],"t_height");              /* ver.2.6 */
    dsp_sym(rev_adr[0],"t_rev");                    /* ver.2.6 */
    dsp_sym(subdat_adr[0],"t_subdat");              /* ver.2.6 */
    dsp_sym(subrev_adr[0],"t_subrev");              /* ver.2.6 */
    dsp_sym(either_adr,"t_either");                 /* ver.2.6 */
    dsp_sym(interval_adr,"t_interval");             /* ver.2.6 */
    height_adr[1] = height_adr[0] + SAMPLING_MAX * 4;   /* ver.2.6 */
    rev_adr[1] = rev_adr[0] + SAMPLING_MAX * 4;         /* ver.2.6 */
    subdat_adr[1] = subdat_adr[0] + SAMPLING_MAX * 4;   /* ver.2.6 */
    subrev_adr[1] = subrev_adr[0] + SAMPLING_MAX * 4;   /* ver.2.6 */
}

/*-----------------------------------------------------------------------+

    関数名      static void arg(int argc,char *argv[])

    返値        0：コマンドラインから　1：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+-----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[20][40],imput_from_key[20];
    int counter;

                                    /* デフォルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */
    strcpy(arguments[2],"0");       /* 0 : コンスタントフォース */
    strcpy(arguments[3],"200");     /* 200 : サンプリング回数 */

    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    if(argc <= 4){                  /* コマンドライン引数が足りないとき */
```

```
        WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
        WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
        WINputstr(&buf,"デフォルト : リターンキー¥n");
        Cslon();
        while(1){
            switch(argc){
            case 1:
            case 2:
            case 3:
                WINlocate(&buf,1,2);
                WINputstr(&buf,
                "0 : コンスタントフォース   1 : コンスタントハイト¥n");
                WINlocate(&buf,1,3);
                WINputstr(&buf,
                "2 : エラーチェック (CF) 3 : ニアー・ファー (CF) ¥n");
                WINdprintf(&buf,"%s  ",arguments[2]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[2],imput_from_key);
                }
            case 4:
                WINlocate(&buf,1,5);
                WINputstr(&buf,"サンプリングドット¥n");
                WINdprintf(&buf,"%s  ",arguments[3]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[3],imput_from_key);
                }
            default:
                break;
            }

            WINputstr(&buf,"¥n以上でよろしいですね    ");
            gets(imput_from_key);
            if(imput_from_key[0] == 0){
                WINclose(&buf);
                break;
            }
        }
        Csloff();
    }

    strcpy(prm_file_name,arguments[1]);
    m_mode = atoi(arguments[2]);
    m_number_of_samples = atoi(arguments[3]);
    return( atoi(arguments[0]) );
}

void open_error(short b_t,short cmdlflag,char *file_name)   /* ver.2.5 */
{
    char msg[200];
    static char *btn[3] = { "バイナリモードで", "テキストモードで", "" };

    strcpy(msg,file_name);
    strcat(msg,"¥nが");
    strcat(msg,btn[b_t]);
    strcat(msg,"オープンできません");
```

```
    errer_msg(msg);
    putchar('¥a');
    end_job(cmdlflag);
}

/*-----------------------------------------------------------------------+

    関数名      static int set_scan_param(void)

    コメント      パラメータファイルからスキャンのパラメータを読み込み、
                ウインドウを開き、デフォルトの値の変更をたずねる。その後
                ｄｓｐにパラメータを送る。

+-----------------------------------------------------------------------*/
static void set_scan_param(short cmdlflag)  /* int -> void by KOTARO  */
{                                           /* void -> short cmdlflag */
    static char parameter[40][40],buffer[40],name[40],subname[40],
                *expand[4]={ ".daf" , ".dah" , ".dae" , ".dan" }; /*ver.2.5*/
    int number_of_prm,no;

    if(m_mode == CONSTANT_FORCE || m_mode == ERROR_CHECK
                                || m_mode == NEAR_FAR ){    /* ver.2.4 */
        number_of_prm = read_prm_file(prm_file_name,parameter,MOTOR_PART);
        if(number_of_prm == -1) /* by        */
            end_job(cmdlflag);  /*    KOTARO */
        strcat(parameter[0 + number_of_prm],parameter[2 + number_of_prm]);
        if((fp_eng = fopen(parameter[0 + number_of_prm],"rt")) == NULL)
            open_error(2,cmdlflag,parameter[0 + number_of_prm]); /*ver.2.5*/
        fscanf(fp_eng,"%s",buffer);
        engage_vol = (float)atof(buffer);       /* エンゲージ次の電圧を得る */

        if( m_mode == NEAR_FAR )    no = 6; /* ver.2.4 */
        else                        no = 5;
    }
    else
        no = 4;

    number_of_prm = read_prm_file(prm_file_name,parameter,SCAN_PART);
    if(number_of_prm == -1) /* by        */
        end_job(cmdlflag);  /*    KOTARO */
    ask_prm(&buf,"スキャン条件",parameter,number_of_prm,1,no);

    strcat(parameter[0 + number_of_prm],parameter[1 + number_of_prm]);
    strcpy(name,parameter[0 + number_of_prm]);      /* ver.2.5 */
    strcpy(subname,parameter[0 + number_of_prm]);   /* ver.2.5 */
    strcat(name,expand[m_mode]);                    /* ver.2.5 */
                                /* デフォルトの拡張子 .dat */

    if(file_type == 0){
        if((fp = fopen(name,"w+b")) == NULL)
            open_error(0,cmdlflag,name);            /*ver.2.5*/
    }
    else{
        if((fp = fopen(name,"w+t")) == NULL)
            open_error(1,cmdlflag,name);        /*ver.2.5*/
    }
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR){    /* ver.2.5 */
        if(m_mode == ERROR_CHECK)   strcat(subname,".def");
```

```
    else                        strcat(subname,".dnf");
    if(file_type == 0){
        if((subfp = fopen(subname,"w+b")) == NULL)
            open_error(0,cmdlflag,subname);
    }
    else{
        if((subfp = fopen(subname,"w+t")) == NULL)
            open_error(1,cmdlflag,subname);
    }
}

m_start[X]      = atoi(parameter[2 + number_of_prm]);
m_start[Y]      = atoi(parameter[3 + number_of_prm]);
m_scan_size     = atoi(parameter[4 + number_of_prm]);
if(m_start[X] > m_start[Y]){
    if(m_scan_size > MAX_SCAN_SIZE - m_start[X])
        m_scan_size = (short)MAX_SCAN_SIZE - m_start[X];
}else{
    if(m_scan_size > MAX_SCAN_SIZE - m_start[Y])
        m_scan_size = (short)MAX_SCAN_SIZE - m_start[Y];
}
scan_force  = (float)atof(parameter[5 + number_of_prm]);
m_near          = (float)atof(parameter[6 + number_of_prm]); /* ver.2.4 */
m_average       = atol(parameter[7 + number_of_prm]);
/*m_damy[X]     = atoi(parameter[8 + number_of_prm]);*/ /* ver.2.6 */
m_interval      = atoi(parameter[8 + number_of_prm]);   /* ver.2.6 */
m_time          = (long)(atof(parameter[9 + number_of_prm]) * 0.357143);
                        /* クロックが３５７ＫＨｚに設定されているため */
m_div_times[X]  = atoi(parameter[10 + number_of_prm]);
m_div_times[Y]  = atoi(parameter[11 + number_of_prm]);
m_div_times[Z]  = atoi(parameter[12 + number_of_prm]);
m_zpiezo_index  = (float)atof(parameter[13 + number_of_prm]);
m_lever_index   = (float)atof(parameter[14 + number_of_prm]);
lever_eras      = (float)atof(parameter[15 + number_of_prm]);
m_warm_up       = atoi(parameter[16 + number_of_prm]);
file_type       = atoi(parameter[17 + number_of_prm]);
m_both          = atoi(parameter[18 + number_of_prm]);

m_original_spot = scan_force / (m_lever_index * lever_eras) + engage_vol;
                        /* フィードバック目標電圧設定 */

dsp32((int)1,&m_zpiezo_index);
dsp32((int)1,&m_lever_index);
dsp32((int)1,&m_original_spot);
dsp32((int)1,&m_near);

dsp_put(div_times_adr,m_div_times,6);
/*dsp_put(div_times_z_adr,&m_div_times_z,2);*/
dsp_put(warm_up_adr,&m_warm_up,2);
dsp_put(start_adr,m_start,4);
dsp_put(scan_size_adr,&m_scan_size,2);
dsp_put(number_of_samples_adr,&m_number_of_samples,2);
dsp_put(mode_adr,&m_mode,2);
dsp_put(time_adr,&m_time,4);
dsp_put(average_adr,&m_average,4);
dsp_put(zpiezo_index_adr,&m_zpiezo_index,4);
dsp_put(lever_index_adr,&m_lever_index,4);
dsp_put(original_spot_adr,&m_original_spot,4);
```

```
    /*dsp_put(damy_adr, m_damy, 2);*/                    /* ver.2.6 */
    dsp_put(interval_adr, &m_interval, 2);               /* ver.2.6 */
    dsp_put(both_adr, &m_both, 2);
    dsp_put(near_adr, &m_near, 4);     /* ver.2.4 */

    Locate(1, 1);
}

/*-----------------------------------------------------------------------+

    関数名        void get_rev_data(void)

    コメント        帰りのデータをホスト側に取り込む

+-----------------------------------------------------------------------*/
void get_rev_data(void)
{   /* ver.2.6 */
    dsp_get(&m_either, either_adr, (ushort)(sizeof(short)));
    dsp_get(m_height, rev_adr[m_either],
                        (ushort)(sizeof(float) * m_number_of_samples));
    ieee32(m_number_of_samples, m_height);
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR){
        dsp_get(m_subdat, subrev_adr[m_either],
                        (ushort)(sizeof(float) * m_number_of_samples));
        ieee32(m_number_of_samples, m_subdat);
    }
}

/*-----------------------------------------------------------------------+

    関数名        void get_height_data(void)

    コメント        伸び時の測定されたデータをホスト側に取り込みむ

+-----------------------------------------------------------------------*/
void get_height_data(void)
{   /* ver.2.6 */
    dsp_get(&m_either, either_adr, (ushort)(sizeof(short)));
    dsp_get(m_height, height_adr[m_either],
                        (ushort)(sizeof(float) * m_number_of_samples));
    ieee32(m_number_of_samples, m_height);
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR){
        dsp_get(m_subdat, subdat_adr[m_either],
                        (ushort)(sizeof(float) * m_number_of_samples));
        ieee32(m_number_of_samples, m_subdat);
    }
}

/*-----------------------------------------------------------------------+

    関数名        static void init_file(void)

    コメント        オープンされているデータファイルの先頭から４０バイト
                ダミーのファイル情報を書き込む。

+-----------------------------------------------------------------------*/
static void init_file(FILE *lfp)
{
```

```
    int result;
    float buffer[10];

    result = fseek(lfp,(long)0,SEEK_SET);
    result = fwrite(buffer,1,PRM_BYTE,lfp);
}

void paint(short data_y)
{
    short data_x;

    for(data_x = 0;data_x < m_number_of_samples;data_x++){
        if(data_y == 0){
            dat_file.max_dat = m_height[data_x];
            dat_file.min_dat = dat_file.max_dat - (float)0.001;
        }else if(m_height[data_x] > dat_file.max_dat
                            && m_height[data_x] > dat_file.min_dat){
            dat_file.max_dat = m_height[data_x];
        }else if(m_height[data_x] < dat_file.min_dat
                            && m_height[data_x] < dat_file.max_dat){
            dat_file.min_dat = m_height[data_x];
        }
        dot_set(data_x,data_y);
    }
}

/*-----------------------------------------------------------------------+

    関数名      static void filewrite(void)

    コメント      オープンされているデータファイルに、スキャンで得られた
                1ラインのデータを書き込む。

+-----------------------------------------------------------------------*/
static void filewrite(FILE *lfp,float data[])
{
    short counter;
    size_t result;

    if(file_type == 0){
        result = fwrite(data,4,m_number_of_samples,lfp);
    }else{
        for(counter = 0;counter < m_number_of_samples;counter++){
            result = fprintf(lfp,"%d,%f¥n",counter,data[counter]);
        }
    }
}

/*-----------------------------------------------------------------------+

    関数名      void start_job(int argc,char *argv[])

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
int start_job(int argc,char *argv[])
{
    short flag;
```

```
    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */
    cls_256();                      /* グラフィクス画面のクリア */

    init_dsp();
    flag = arg(argc,argv);
    if(flag){                       /* コマンドラインからたち上げた場合 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
        set_256palet(MONOCHROME);   /* ２５６色パレットの設定 */
    }
    set_scan_param(flag);
    dsp_run();
    Cls();                          /* クリアスクリーン */

                                    /* データ構造体に値を代入 */
    dat_file.what_file          = SCAN_DAT_FILE;
    dat_file.number_of_samples  = m_number_of_samples;
    dat_file.scan_size          = m_scan_size;
    dat_file.tilt_x             = (float)0.0;
    dat_file.tilt_y             = (float)0.0;
    dat_file.start[X]           = m_start[X];
    dat_file.start[Y]           = m_start[Y];

    set_scan_screen((int)1);        /* スキャン範囲表示 */
    init_file(fp);
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR)
        init_file(subfp);
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      int end_job(void)

    コメント      データファイルの先頭にファイル情報を書き込み、
                リペイント後、ポーズ。そのご、データファイルをクローズし、
                ウインドウを閉じる。コマンドラインからたちあげた場合のみ
                ライブラリーとドライバの使用を終了する。

+-----------------------------------------------------------------------*/
void end_job(short flag) /* int -> void by KOTARO */
{
    write_prm_to_file(fp,&dat_file);    /* fp added by KOTARO */
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR)     /* ver.2.5 */
        write_prm_to_file(subfp,&dat_file);
    repaint();
    fclose(fp);
    if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR)     /* ver.2.5 */
        fclose(subfp);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"**** 測定終了 ***");

    getch();                        /* ポーズ */

    WINclose(&buf);
    if(flag){                       /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
```

```
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}


int keyin(void)
{
    int ret = 0,result;

    if(kbhit() != 0){
        switch(getch()){
        case 'r':                           /* リスタート */
            result = fseek(fp,(long)PRM_BYTE,SEEK_SET);
            if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR)
                result = fseek(subfp,(long)PRM_BYTE,SEEK_SET);
            m_flag = DSP_RESTART;   dsp_put(flag_adr,&m_flag,2);
            ret = 1;    break;
        case 'e':                           /* 強制終了 */
            m_flag = DSP_END;   dsp_put(flag_adr,&m_flag,2);
            ret = 2;    break;
        default:                            /* ポーズ */
            m_flag = DSP_COMPLETE;  dsp_put(flag_adr,&m_flag,2);
            getch();
            m_flag = DSP_WORKING;   dsp_put(flag_adr,&m_flag,2);
            break;
        }
    }
    return(ret);
}

void main(int argc,char *argv[])
{
    static int j,cmd_line,key;

    cmd_line = start_job(argc,argv);            /* 初期化関数の呼び出し */
    for( j = 0 ; j <= m_number_of_samples - 1 ; j++ ){
        scan_no(j + 1);                 /* 何番目のスキャンか表示 */
        ber_scale();                    /* カラーバーの再スケーリング*/
        key = keyin();                  /* キー入力処理 */
        if(key == 1){                       /* リスタート */
            j = -1;     continue;
        }
        else if(key == 2)   break;          /* 強制終了 */

        do{                     /* ターゲットのサンプリング終了を待つ */
            dsp_get(&m_flag,flag_adr,(ushort)2);
        }while(m_flag != DSP_COMPLETE);
        m_flag = DSP_WORKING;   dsp_put(flag_adr,&m_flag,2);

        get_height_data();              /* データ取得 */
        paint(j);                       /* 描画 */
        filewrite(fp,m_height);         /* データをファイルに書き込む*/
        if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR) /* ver.2.5 */
            filewrite(subfp,m_subdat);
        if(m_both == 1){
```

```
            get_rev_data();             /* 帰りのデータ取得 */
            /*paint(j+1);*/
            filewrite(fp,m_height);     /* データをファイルに書き込む*/
            if(m_mode == ERROR_CHECK || m_mode == NEAR_FAR) /* ver.2.5 */
                filewrite(subfp,m_subdat);
        }
    }
    end_job(cmd_line);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              t_scan46.c                               |
|                            --------------                             |
|                                                                       |
|     ＡＦＭのスキャンを行うＴ＿ＳＣＡＮ４６．ＯＵＴのソースファイル。      |
|     さらに新しい２次曲線フィットアルゴリズム                            |
|     ｘ，ｙピエゾ立ち上げ分割回数別々に設定可能。                        |
|     エラーチェックモード、ニアー・ファーモード（ＣＦのデータを同時取得）  |
|     +間引きスキャン（ただし、ダミースキャン廃止、データ転送方式変更）    |
|                                          (改 by KOTARO)               |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <libap.h>
#include <math.h>       /* sqrt() */
#include "lorypio.h"
#include "afm_adda.h"
#include "afm_def.h"

/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
void  init();
float f();
float g();
float inv_f();
float inv_g();
void  calcu_rect();
float detv_ex();
float detv_sh();
float feedback();
void  near_far();       /* ver.4.5 */
void  sampling();       /* ver.4.5 */
void  init_loop();
void  piezo_set();
void  last_job();
void  f_drive();        /* ver.4.6 */
void  b_drive();        /* ver.4.6 */
/*
+-----------------------------------------------------------------------+
|                          グローバル変数                               |
+-----------------------------------------------------------------------+
*/
short   t_div_times[3] = { 60 , 60 , 225 }, t_warm_up = 3, t_both = 0,
        t_start[2], t_scan_size, t_number_of_samples, t_flag, t_mode,
        t_damy[2] = {0,0}, t_either = 1, t_interval, t_repeat = 1/* ver.4.6 */;
long    t_time, t_average = 1;
float   t_lever_index, t_original_spot, t_zpiezo_index, t_near,
        t_height[2][SAMPLING_MAX], t_rev[2][SAMPLING_MAX],    /* ver.4.6 */
        t_subdat[2][SAMPLING_MAX], t_subrev[2][SAMPLING_MAX];/* ver.4.6 */

float   drive_v[3], last_drive_v[3], d_displace, rectify_f[2], rectify_g[2],
        start_v[2], reverse_v[2];
/*
+-----------------------------------------------------------------------+
```

```
+-----------------------------------------------------------------------+
*/
#include "afm_func.h"

/*-----------------------------------------------------------------------+

    関数名      void init()

    引数        無し

    返値        無し

    コメント      ピエゾ電圧を示す変数の初期化、１ドット当たりの長さ[nm]の
                計算、クロックボードのクロック発生

+-----------------------------------------------------------------------*/
void init()
{
    drive_v[X] = 0.0;
    drive_v[Y] = 0.0;
    drive_v[Z] = 0.0;

    last_drive_v[X] = (float)0.0;
    last_drive_v[Y] = (float)0.0;
    last_drive_v[Z] = (float)0.0;

    d_displace = (float)t_scan_size / (float)(t_number_of_samples);
    /*if(d_displace * t_damy[X] > t_start[X]){
        t_damy[X] = t_start[X] / d_displace;
        t_start[X] -= d_displace * t_damy[X];
    }*/                                         /* ver.4.6 */

    osc_start(osc_357KHZ);
}

/*-----------------------------------------------------------------------+

    関数名      float f(voltage)

    引数        voltage:    ピエゾにかける電圧

    返値        ピエゾの伸び


    コメント      板ピエゾの＋１０ｖから－１０ｖをかけたときの
                伸びのヒステリシスに２字曲線をフィットしたもの。
                引数の voltage[v] のピエゾ電圧をかけたときの伸びを返す。

+-----------------------------------------------------------------------*/
float f(voltage)
float voltage;
{
    register float  _a = A_F,
                    _b = B_F;
            float   _c = C_F;

    return(_c + voltage * (_b + voltage * _a));
```

```
}

/*-----------------------------------------------------------------------+

    関数名        float g(voltage)

    引数          voltage:    ピエゾにかける電圧

    返値          ピエゾの伸び

    コメント        板ピエゾの－１０ｖから＋１０ｖをかけたときの
                  縮みのヒステリシスに２字曲線をフィットしたもの。
                  引数の voltage[v] のピエゾ電圧をかけたときの伸びを返す。

+-----------------------------------------------------------------------*/
float g(voltage)
float voltage;
{
    register float  _a = A_G,
                    _b = B_G;
             float  _c = C_G;

    return(_c + voltage * (_b + voltage * _a));
}

/*-----------------------------------------------------------------------+

    関数名        float inv_f(length)

    引数          length: ピエゾの伸び

    返値          電圧

    コメント        板ピエゾの＋１０ｖから－１０ｖをかけたときの
                  伸びのヒステリシスに２字曲線をフィットしたもの。
                  解の公式により引数の length[nm] だけ伸びるときの
                  ピエゾ電圧を返す。

+-----------------------------------------------------------------------*/
float inv_f(length)
float length;
{
    register float  _a = A_F,
                    _b = B_F;
             float  _c = C_F;

#ifndef V_LINEAR        /* ドライブがリニアでないとき */

    return((-_b - sqrt(_b*_b - 4.0*_a*(_c-length)))/(2.0*_a));

#else

    return( (length - _c) / _b );

#endif
}

/*-----------------------------------------------------------------------+
```

関数名　　float inv_g(length)

引数　　length: ピエゾの伸び

返値　　電圧

コメント　　板ピエゾの－１０ｖから＋１０ｖをかけたときの
縮みのヒステリシスに２字曲線をフィットしたもの。
解の公式により、引数の length[nm] の伸びとなるときの
ピエゾ電圧を返す。

```
+-----------------------------------------------------------------------*/
float inv_g(length)
float length;
{
    register float  _a = A_G,
                    _b = B_G;
             float  _c = C_G;

#ifndef V_LINEAR        /* ドライブがリニアでないとき */

    return((-_b - sqrt(_b*_b - 4.0*_a*(_c-length)))/(2.0*_a));

#else

    return( (length - _c) / _b );

#endif
}

/*-----------------------------------------------------------------------+
```

関数名　　void calcu_rect(ch,begin_v,begin_l,point_v,point_l)

引数　　ch:　　０：ｘ、１：ｙ軸
begin_v:　　平行移動した曲線の始点のｖ座標
begin_l:　　平行移動した曲線の始点のｌ座標
point_v:　　通る点のｖ座標
point_l:　　通る点のｌ座標

返値　　無し

コメント　　beginの位置に平行移動された曲線ｆまたはｇが、
pointを通るように、rectify_fまたはrectify_gを計算する。

```
+-----------------------------------------------------------------------*/
void calcu_rect(ch,begin_v,begin_l,point_v,point_l)
short ch;
float begin_v,begin_l,point_v,point_l;
{
    if(begin_l < point_l){                      /* 伸びの場合 */
        rectify_f[ch] = (point_l - begin_l) /
                        f(point_v + MAX_PIEZO_VOLT - begin_v);
    }else{                                      /* 縮みの場合 */
        rectify_g[ch] = (point_l - begin_l) /
        ( g(point_v - MAX_PIEZO_VOLT - begin_v) - MAX_SCAN_SIZE);
    }
```

```
}

/*-----------------------------------------------------------------------+

    関数名      float detv_ex(ch,begin_v,begin_l,length)

    引数        ch:         0：x、1：y軸
                begin_v:    平行移動した曲線の始点のv座標
                begin_l:    平行移動した曲線の始点のl座標
                length:     ピエゾの伸び

    返値        電圧

    コメント      ピエゾを伸びる方向にドライブする場合、引数の length[nm]
                だけ伸びるときの電圧を返す。

+-----------------------------------------------------------------------*/
float detv_ex(ch,begin_v,begin_l,length)
short ch;
float begin_v,begin_l,length;
{
    return(inv_f((length-begin_l) / rectify_f[ch]) - MAX_PIEZO_VOLT + begin_v);
}

/*-----------------------------------------------------------------------+

    関数名      float detv_sh(ch,length)

    引数        ch:         0：x、1：y軸
                begin_v:    平行移動した曲線の始点のv座標
                begin_l:    平行移動した曲線の始点のl座標
                length:     ピエゾの伸び

    返値        電圧

    コメント      ピエゾを縮む方向にドライブする場合、引数の length[nm]
                だけの伸となるときの電圧を返す。

+-----------------------------------------------------------------------*/
float detv_sh(ch,begin_v,begin_l,length)
short ch;
float begin_v,begin_l,length;
{
    return( inv_g((length - begin_l) / rectify_g[ch] +(float)MAX_SCAN_SIZE)
                                        + (float)MAX_PIEZO_VOLT + begin_v);
}

/*-----------------------------------------------------------------------+

    関数名      float feedback()

    引数        無し

    返値        高さの差

    コメント      コンスタントハイトの場合のZピエゾにかけるフィードバックの
                量を求め、フィードバックをかける。フィードバックの量が
                オーバーレンジの場合は、フィードバックをかけない。
```

返値は１つ前の点との高さの差[nm]。

```
+-----------------------------------------------------------------------*/
float feedback()
{
    static      short first = 1 , dz/*ver4.6*/;
    static      float height = 0.0;
                float dif_height,d_drive_v_z,sensor_v,cor_height;
    register    float *zp,*lzp;

    zp  = &drive_v[Z];
    lzp = &last_drive_v[Z];
    dz  = t_div_times[Z];   /*ver4.6*/

    sensor_v = adaverage(1,t_average);
    dif_height = (sensor_v - t_original_spot) * t_lever_index;
    d_drive_v_z = dif_height / t_zpiezo_index;
    *zp += d_drive_v_z;

    if(fabs(*zp) <= MAX_PIEZO_VOLT){
        if(first){
            piezo_drive((short)Z,*zp,*lzp,(short)SAFE_DRIVE_Z);
            *lzp = *zp;
            first--;
            /*wait(t_time);
            cor_height=(adaverage(1,t_average)-t_original_spot)*t_lever_index;
            if(fabs(cor_height) > 6.0){
                *zp += cor_height / t_zpiezo_index;
                piezo_drive((short)Z,*zp,*lzp,dz);
                *lzp = *zp;
            }*/
            return(0.0);        /*フィードバック最初の高さを０とする*/
        }
        else{
            piezo_drive((short)Z,*zp,*lzp,dz);
            *lzp = *zp;
            /*wait(t_time);
            cor_height=(adaverage(1,t_average)-t_original_spot)*t_lever_index;
            if(fabs(cor_height) > 6.0){
                *zp += cor_height / t_zpiezo_index;
                piezo_drive((short)Z,*zp,*lzp,dz);
                *lzp = *zp;
            }*/
        }
    }
    else{
        *zp -= d_drive_v_z;
        dif_height = 0.0;
    }

    height += dif_height;
    return(height);
}

/*-----------------------------------------------------------------------+

    関数名      float near_far()
```

引数　　無し

返値　　形状（？）からのずれ（ｎｍ）

コメント　　コンスタントフォースと同様にフィードバックをかけた後、
カンチレバーを t_near[nm](+:near,-:far) だけ動かし、
センサーの応答をみる。そして、カンチレバーを元に戻す。

```
+-----------------------------------------------------------------------*/
void near_far(dat,subdat)          /* ver.4.5 */
float *dat,*subdat;
{
    *subdat = feedback();

    drive_v[Z] -= t_near / t_zpiezo_index;
    piezo_drive((short)Z,drive_v[Z],last_drive_v[Z],t_div_times[Z]);

    wait(t_time);
    *dat = (adaverage(1,t_average) - t_original_spot) * t_lever_index;

    piezo_drive((short)Z,last_drive_v[Z],drive_v[Z],t_div_times[Z]);
    drive_v[Z] = last_drive_v[Z];
}

/*-----------------------------------------------------------------------+
```

関数名　　float sampling()

引数　　無し

返値　　無し

コメント　　コンスタントハイトの場合は単にＡＤ変換の関数を呼び
コンスタントフォースの場合はフィードバックの関数を呼ぶ。

```
+-----------------------------------------------------------------------*/
void sampling(dat,subdat)          /* ver.4.5 */
float *dat,*subdat;
{
    wait(t_time);
    if(t_mode == CONSTANT_HEIGHT)
        *dat = ( adaverage(1,t_average) - t_original_spot ) * t_lever_index;
    else if(t_mode == CONSTANT_FORCE )
        *dat = feedback();
    else if(t_mode == ERROR_CHECK){
        *subdat = feedback();
        wait(t_time);
        *dat = (adaverage(1,t_average) - t_original_spot) * t_lever_index;
    }
    else
        near_far(dat,subdat);
}

/*-----------------------------------------------------------------------+
```

関数名　　void init_loop(ch)

引数　　ch:　　０：ｘ、１：ｙ軸

返値　　無し

コメント　　ピエゾを定常的なヒステリシスループに入れる。

```
+-----------------------------------------------------------------------*/
void init_loop(ch)
short ch;
{
    short i,j;
    float displace = (float)MAX_SCAN_SIZE / 2 , dam_dat,dam_subdat;

    if(t_start[ch] <= MAX_SCAN_SIZE / 2){
                                    /* t_start[ch]が中心より下にある場合 */
        /* 初期履歴曲線を求める */
        calcu_rect(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
                                        (float) 10.0,(float)0.0);
        /* スキャン開始点の電圧を求める */
        start_v[ch] = detv_sh(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
                                        (float)t_start[ch]);
        /* 初期履歴曲線上をドライブ */
        while(1){
            drive_v[ch] = detv_sh(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
                                                            displace);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);         /* ver.4.5 */
            displace -= d_displace;
            if(displace <= t_start[ch])
                break;
        }

        rectify_f[ch] = 1.0;
        /* 伸びのヒステリシスは補正なしで、平行移動だけ */
        for(i = 0;i < t_number_of_samples + t_damy[ch];i++){
            drive_v[ch] = detv_ex(ch,start_v[ch],(float)t_start[ch],
                                    d_displace * i + t_start[ch]);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);         /* ver.4.5 */
        }

        reverse_v[ch] = last_drive_v[ch];
        /* 縮みのヒステリシスの補正値を求める */
        calcu_rect(ch,reverse_v[ch],
            (float)t_start[ch] + t_scan_size + d_displace * t_damy[ch],
            start_v[ch],(float)t_start[ch]);
        for(i = t_number_of_samples + t_damy[ch] - 1;i >= 0;i--){
            drive_v[ch] = detv_sh(ch,reverse_v[ch],
            (float)t_start[ch] + t_scan_size + d_displace * t_damy[ch],
            d_displace * i + t_start[ch]);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);         /* ver.4.5 */
        }
    }else{                          /* t_start[ch]が中心より上にある場合 */
        /* 初期履歴曲線を求める */
        calcu_rect(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
```

```
                                        (float)-10.0,(float)MAX_SCAN_SIZE);
        /* スキャン終了点の電圧を求める */
        reverse_v[ch] = detv_ex(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
                                    (float)t_start[ch] + t_scan_size);
        /* 初期履歴曲線上をドライブ */
        while(1){
            drive_v[ch] = detv_ex(ch,(float)0.0,(float)MAX_SCAN_SIZE / 2,
                                                                displace);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);          /* ver.4.5 */
            displace += d_displace;
            if(displace >= t_start[ch] + t_scan_size)
                break;
        }

        rectify_g[ch] = 1.0;
        /* 縮みのヒステリシスは補正なしで、平行移動だけ */
        for(i = t_number_of_samples + t_damy[ch] - 1;i >= 0;i--){
            drive_v[ch] = detv_sh(ch,reverse_v[ch],
            (float)t_start[ch] + t_scan_size + d_displace * t_damy[ch],
            d_displace * i + t_start[ch]);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);          /* ver.4.5 */
        }

        start_v[ch] = last_drive_v[ch];
        /* 伸びのヒステリシスの補正値を求める */
        calcu_rect(ch,start_v[ch],(float)t_start[ch],reverse_v[ch],
        (float)t_start[ch] + t_scan_size + d_displace * t_damy[ch]);
    }

    for(j = 0;j < t_warm_up;j++){
        for(i = 0;i < t_number_of_samples + t_damy[ch];i++){
            drive_v[ch] = detv_ex(ch,start_v[ch],(float)t_start[ch],
                                    d_displace * i + t_start[ch]);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);          /* ver.4.5 */
        }

        for(i = t_number_of_samples + t_damy[ch] - 1;i >= 0;i--){
            drive_v[ch] = detv_sh((short)ch,reverse_v[ch],
            (float)t_start[ch] + t_scan_size + d_displace * t_damy[ch],
            d_displace * i + t_start[ch]);
            piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
            last_drive_v[ch] = drive_v[ch];
            sampling(&dam_dat,&dam_subdat);          /* ver.4.5 */
        }
    }
}

/*-----------------------------------------------------------------------+

    関数名      void piezo_set(ch,v1,l1,v2,l2)

    引数        ch:     0：x、1：y軸
```

v1:　　現在のピエゾ電圧
l1:　　現在のピエゾの伸び
v2:　　目標ピエゾ電圧
l2:　　目標ピエゾ長さ

返値　　無し

コメント　　引数 ch で指定されたピエゾを電圧 v1 伸び l1 から
電圧 v2 伸び l2 までサンプリングしながら（つまり、
コンスタントフォースの場合にフィードバックをかけながら）
リニアにドライブする。

```
+-----------------------------------------------------------------------*/
void piezo_set(ch,v1,l1,v2,l2)
short ch;
float v1,l1,v2,l2;
{
    float _a,_b,extend = l1,dam_dat,dam_subdat/*ver.4.5*/;

    _a = (l2 - l1) / (v2 - v1);
    _b = l1 - _a * v1;

    while(1){
        drive_v[ch] = (extend - _b) / _a;
        piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
        last_drive_v[ch] = drive_v[ch];
        sampling(&dam_dat,&dam_subdat);          /* ver.4.5 */
        if(l1 < l2){
            extend += d_displace;
            if(extend >= l2){
                drive_v[ch] = v2;
                piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
                last_drive_v[ch] = drive_v[ch];
                break;
            }
        }else{
            extend -= d_displace;
            if(extend <= l2){
                drive_v[ch] = v2;
                piezo_drive(ch,drive_v[ch],last_drive_v[ch],t_div_times[ch]);
                last_drive_v[ch] = drive_v[ch];
                break;
            }
        }
    }
}

/*-----------------------------------------------------------------------+
```

関数名　　void last_job()

引数　　無し

返値　　無し

コメント　　スキャンが終了した後，X，Yピエゾを伸びが0になるまで
縮める。

```
+-----------------------------------------------------------------------*/
void last_job(no)
short no;
{
    piezo_set(Y, last_drive_v[Y], (float)(t_start[Y] + d_displace * no),
                            (float)0.0, (float)MAX_SCAN_SIZE / 2);
    piezo_set(X, last_drive_v[X], (float)t_start[X],
                            (float)0.0, (float)MAX_SCAN_SIZE / 2);

    piezo_drive(Z, (float)0.0, last_drive_v[Z], (short)SAFE_DRIVE_Z);
    last_drive_v[Z] = drive_v[Z];
}


void f_drive(ch, i)                         /* ver.4.6 */
short ch, i;
{
    drive_v[ch] = detv_ex(ch, start_v[ch], (float)t_start[ch],
                                    d_displace * i + (float)t_start[ch]);
    piezo_drive(ch, drive_v[ch], last_drive_v[ch], t_div_times[ch]);
    last_drive_v[ch] = drive_v[ch];
}
void b_drive(ch, i)                         /* ver.4.6 */
short ch, i;
{
    drive_v[ch] = detv_sh(ch, reverse_v[ch], (float)t_start[ch] + t_scan_size,
                                    d_displace * i + (float)t_start[ch]);
    piezo_drive(ch, drive_v[ch], last_drive_v[ch], t_div_times[ch]);
    last_drive_v[ch] = drive_v[ch];
}

/*-----------------------------------------------------------------------+

    関数名      void main()

+-----------------------------------------------------------------------*/
void main()                                 /* ver.4.6 */
{
    short y, x1, x2, i = 0, j = 0, repeat;
    register float *dp, *sp, *rp, *srp, *dq, *sq, *rq, *srq;

    init();
    init_loop(Y);
    init_loop(X);

    for( y = 0 ; y <= t_number_of_samples - 1 ; y++ ){
        f_drive((short)Y, y);    /* ｙ軸スキャン */
        dq = &t_height[j][0];   sq = &t_subdat[j][0];
        rq = &t_rev[j][t_number_of_samples - 1];
        srq = &t_subrev[j][t_number_of_samples - 1];
        repeat = t_repeat;
        for( x1 = 0 ; x1 <= t_interval - 1 ; x1++ ){
            /* 行き */  dp = dq;    sp = sq;
            for( x2 = 0 ; x2 <= t_number_of_samples - 1 ; x2++ ){
                if( x2 == t_interval * i + x1 ){
                    f_drive((short)X, x2);
                    sampling(dp, sp);
                    i++;
```

```
                }
                dp++;   sp++;
            }
            f_drive((short)X, t_number_of_samples - 1);
            i--;
            /* 帰り */  rp = rq;    srp = srq;
            for( x2 = t_number_of_samples - 1 ; x2 >= 0 ; x2-- ){
                if( x2 == t_interval * i + x1 ){
                    b_drive((short)X, x2);
                    sampling(rp, srp);
                    i--;
                }
                rp--;   srp--;
            }
            b_drive((short)X, 0);
            i = 0;
            if( repeat != 0 && t_interval != 1 ){
                repeat--;
                x1 = -1;
            }
        }
        /* キー入力判定 */
        if(t_flag == DSP_WORKING){          /* キー入力なし */
            t_either ^= 0x0001;
            t_flag = DSP_COMPLETE;  /* x軸1ライン往復スキャン終了 */
            j ^= 0x0001;
        }
        else if(t_flag == DSP_COMPLETE){    /* ポーズ */
            while(t_flag == DSP_COMPLETE)
                ;
            t_either ^= 0x0001;
            t_flag = DSP_COMPLETE;  /* x軸1ライン往復スキャン終了 */
            j ^= 0x0001;
        }
        else if(t_flag == DSP_RESTART){     /* リスタート */
            t_flag = DSP_WORKING;
            piezo_set(Y, last_drive_v[Y], (float)t_start[Y] + d_displace * y,
                                            start_v[Y], (float)t_start[Y]);
            y = -1; j = 0;  t_either = 1;
        }
        else if(t_flag == DSP_END)  break;  /* 強制終了 */
    }
    last_job(y);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

プログラムファイル

(B.6 Image Program Files)

```
/*
+-----------------------------------------------------------------------+
|                                shade.h                                | |
|                               ---------                               |
     |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include "vmath.h"

typedef struct _Surface {
    Vector icol;
    Vector ocol;
    int n;
} Surface;

typedef struct _Illum {
    struct _Illum *next;
    Vector dist;
    Color col;
} Illum;

/*
+-----------------------------------------------------------------------+
|                            end of shade.h                             |
 |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                wf_scan.c                              |
|                             ---------------                           |
|                                                                       |
|                                                                       |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include "vmath.h"
#include "video.h"

/*------------------Extern functions---------------------*/

extern double getdata(int average,long dx,long dy);
extern void fview(double,double,double);
extern void getpxel(void);

/*--------------------------プロトタイプ宣言----------------------------*/

void drawline(double px,double py,double h,int first);
void wf_scan(int step);

/*----------------------------- 外部変数 -------------------------------*/

extern short number_of_samples;
extern double data_min,height_rate,px,py;
extern Vector eye, cent;

/*-------------------------------コード----------------------------------*/

/*-----------------------------------------------------------------------+

    関数名      void drawline(double px,double py,double h,int first)

    引数        double px:  スクリーンのX座標
                double py:  スクリーンのY座標
                double h:   データの高さ
                int first:  線の引き始め：1

    コメント      引数で指定される座標に、前回線を引き終えた座標から
                データの高さにより色を変えて線を引く

+-----------------------------------------------------------------------*/
void drawline(double px,double py,double h,int first)
{
    int palet_no;
    static int last_pixel[2];

    palet_no = (int)(h / ((double)number_of_samples * height_rate) * 255);

    if(palet_no > 255)
        palet_no = 255;
    else if(palet_no < 0)
        palet_no = 0;
```

```
    if(first)
        set_line((int)px,(int)py,(int)px,(int)py,palet_no,0);
    else
        set_line(last_pixel[X],last_pixel[Y],(int)px,(int)py,palet_no,0);

    last_pixel[X] = (int)px;
    last_pixel[Y] = (int)py;
}

/*-----------------------------------------------------------------------+

    関数名      void wf_scan(int step)

    引数        int step:  ワイヤーフレームの線の間隔

    コメント      引数で指定される間隔で、視線の奥の方からスキャンデータ
                を読み出し、スクリーン上の座標を求め、線を引く

+-----------------------------------------------------------------------*/
void wf_scan(int step)
{
    long dx, dy;
    float position[3];
    double height;
    static short first = 0;

    position[X] = (float)(eye[X] - cent[X]);  /* 視線の向きを計算 */
    position[Y] = (float)(eye[Y] - cent[Y]);

    if(fabs(position[Y]/position[X]) <= 1){
        if(position[X] >= 0){              /* データ平面(0,0)が1番奥の隅 */
            for (dx = 0;dx < number_of_samples - 1;dx += step){
                first = 1;
                for(dy = 0;dy < number_of_samples - 1;dy += step){
                    height = getdata(1,(long)dx,(long)dy);
                    fview((long)dx,(long)dy,height);
                    getpxel();
                    drawline(px,py,height,first);
                    if(first)
                        first--;
                }
                if(_kbhit())
                    break;
            }
        }else{          /* データ平面(number_of_samples,0)が1番奥の隅 */
            for (dx = number_of_samples - 1;dx >= 0;dx -= step) {
                first = 1;
                for(dy=0;dy < number_of_samples-1;dy+=step){
                    height = getdata(1,dx,dy);
                    fview(dx,dy,height);
                    getpxel();
                    drawline(px,py,height,first);
                    if(first)
                        first--;
                }
                if(_kbhit())
                    break;
            }
```

```
        }
    }else{
        if(position[Y] >= 0){           /* データ平面(0,0)が１番奥の隅 */
            for (dy = 0;dy < number_of_samples - 1;dy += step){
                first = 1;
                for(dx = 0;dx < number_of_samples - 1;dx += step){
                    height = getdata(1,dx,dy);
                    fview(dx,dy,height);
                    getpxel();
                    drawline(px,py,height,first);
                    if(first)
                        first--;
                }
                if(_kbhit())
                    break;
            }
        }else{          /* (number_of_samples,number_of_samples)が１番奥の隅 */
            for (dy = number_of_samples- 1 ;dy >= 0;dy -= step){
                first = 1;
                for(dx=0;dx < number_of_samples-1;dx+=step){
                    height = getdata(1,dx,dy);
                    fview(dx,dy,height);
                    getpxel();
                    drawline(px,py,height,first);
                    if(first)
                        first--;
                }
            }
        }
    }
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              vmath.c                                  |
|                            ------------                               |
|                                                                       |
|          ベクトル演算パッケージ・Cプログラムブック２参照              |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include <math.h>
#include <process.h>
#include "vmath.h"
Vector vzero = {
    0.0, 0.0, 0.0
};

Matrix munit = {
    1.0, 0.0, 0.0,
    0.0, 1.0, 0.0,
    0.0, 0.0, 1.0
};


#define STKMAX 300

static Element stk[STKMAX];
Element *stkpnt;


void sinit(void)
{
    stkpnt = stk + STKMAX;
}


void prints(void)
{
    Element *p;
    int i;

    i = 0;
    printf("¥n-- stack top --¥n");
    for (p = stkpnt; p < stk + STKMAX; ++p) {
        printf("%-10g", *p);
        if (++i >= 3) {
            i = 0 ;
            putchar('¥n');
        } else
            putchar(',');
    }
    printf("¥n-- stack bottom --¥n¥n");
}


void vpsh(Vector vct)
{
```

```
    if (stkpnt > stk + STKMAX) {
        printf("VECTOR OPERATION PACKAGE: Stack underflow V¥n");
        exit(2);
    }
    stkpnt -= DIM;
    if (stkpnt < stk) {
        printf("VECTOR OPERATION PACKAGE: Stack Overflow V¥n");
        exit(2);
    }
    stkpnt[X] = vct[X];
    stkpnt[Y] = vct[Y];
    stkpnt[Z] = vct[Z];
}


void vpshi(double x,double y,double z)
{
    Vector tmp;

    tmp[X] = x;
    tmp[Y] = y;
    tmp[Z] = z;
    vpsh(tmp);
}


void vpop(Vector vct)
{
    vct[X] = stkpnt[X];
    vct[Y] = stkpnt[Y];
    vct[Z] = stkpnt[Z];
    stkpnt += DIM;
}


void printv(void)
{
    printf("(%-10g, %-10g, %-10g)¥n", stkpnt[X], stkpnt[Y], stkpnt[Z]);
    stkpnt += DIM;
}


void vdup(void)
{
    vpsh(stkpnt);
}


void vadd(void)
{
    Element *a;

    a = stkpnt;
    stkpnt += DIM;
    stkpnt[X] += a[X];
    stkpnt[Y] += a[Y];
    stkpnt[Z] += a[Z];
}
```

```
void vsub(void)
{
    Element *a;

    a = stkpnt;
    stkpnt += DIM;
    stkpnt[X] -= a[X];
    stkpnt[Y] -= a[Y];
    stkpnt[Z] -= a[Z];
}


void vneg(void)
{
    stkpnt[X] = -stkpnt[X];
    stkpnt[Y] = -stkpnt[Y];
    stkpnt[Z] = -stkpnt[Z];
}


void vmul(double val)
{
    stkpnt[X] *= val;
    stkpnt[Y] *= val;
    stkpnt[Z] *= val;
}


void cflt(void)
{
    Element *a;

    a = stkpnt;
    stkpnt += DIM;
    stkpnt[X] *= a[X];
    stkpnt[Y] *= a[Y];
    stkpnt[Z] *= a[Z];
}


double vspro(void)
{
    Element *a;
    Element *b;
    double product/*, p*/;

    a = stkpnt;
    b = stkpnt + DIM;
    stkpnt += 2 * DIM;
    product = b[X] * a[X] + b[Y] * a[Y] + b[Z] * a[Z];
    return (product);
}


double vabs2(void)
{
```

```
    double product;
    Element *a;

    a = stkpnt;
    product = a[X] * a[X] + a[Y] * a[Y] + a[Z] * a[Z];
    stkpnt += DIM;
    return (product);
}


double vcos(void)
{
    double a, b;
    Vector x;

    vdup();
    b = sqrt(vabs2());
    vpop(x);
    vdup();
    a = sqrt(vabs2());
    vpsh(x);
    return (vspro() / a / b);
}


void vvpro(void)
{
    struct SFVVPRO {
        Vector j;
        Vector i;
        Element nvvpro;
    } *f;

    Vector temp;

    f = (struct SFVVPRO *)stkpnt;
    temp[X] = f->i[Y] * f->j[Z] - f->i[Z] * f->j[Y];
    temp[Y] = f->i[Z] * f->j[X] - f->i[X] * f->j[Z];
    temp[Z] = f->i[X] * f->j[Y] - f->i[Y] * f->j[X];
    stkpnt = &(f->nvvpro);
    vpsh(temp);
}


void mpsh(Matrix mat)
{
    Element *m, *s;

    if (stkpnt > stk + STKMAX) {
        printf("VECTOR OPERATION PACKAGE: Stack underflow M¥n");
        exit(2);
    }
    s = stkpnt -= DIM * DIM;
    if (stkpnt < stk) {
        printf("VECTOR OPERATION PACKAGE: Stack Overflow M¥n");
        exit(2);
    }
    for (m = (Element *)mat; m < mat[DIM]; ++m)
```

```
        *(s++) = *m;
}


void mpop(Element *mat)
{
    struct SFMPOP {
        Matrix matrix;
        Element nmpop;
    } *f;

    Element *m;

    f = (struct SFMPOP * )stkpnt;
    for (m = f->matrix[0]; m < f->matrix[DIM]; ++m)
        *(mat++) = *m;
    stkpnt = &(f->nmpop);
}


void mvmul(void)
{
    struct SFMVMUL {
        Vector v;
        Matrix m;
        Element nmvmul;
    } *f;

    Vector tmp;

    f = (struct SFMVMUL *)stkpnt;
    tmp[X] = (f->m)[X][X] * (f->v)[X]
                 + (f->m)[X][Y] * (f->v)[Y]
                       + (f->m)[X][Z] * (f->v)[Z];
    tmp[Y] = (f->m)[Y][X] * (f->v)[X]
                 + (f->m)[Y][Y] * (f->v)[Y]
                       + (f->m)[Y][Z] * (f->v)[Z];
    tmp[Z] = (f->m)[Z][X] * (f->v)[X]
                 + (f->m)[Z][Y] * (f->v)[Y]
                       + (f->m)[Z][Z] * (f->v)[Z];
    stkpnt = &(f->nmvmul);
    vpsh(tmp);
}


void mmmul(void)
{
    struct SFMMMUL {
        Matrix b;
        Matrix a;
        Element nmmmul;
    } *f;

    Element (*ap)[3];
    Matrix temp;
    Element (*tp)[3];
    Element *p, *q, *r;
```

```
    f = (struct SFMMMUL *)stkpnt;
    ap = f->a;
    for (tp = temp; tp < temp + DIM; ++tp) {
        r = *f->b;
        for (p = *tp; p < *tp + DIM; ++p) {
            *p = (double)0;
            for (q = *ap; q < *ap +DIM; ++q) {
                *p += *q * *r;
                r += 3;
            }
            r -= 8;
        }
        ap++;
    }
    stkpnt = &(f->nmmmul);
    mpsh(temp);
}

/*
+-------------------------------------------------------------------------+
|                              end of vmath.c                             |
+-------------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                                                       |
|                              shade.c                                  |
|                                                                       |
|                   シェーディングをするモジュール                      |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "shade.h"
#include "vmath.h"

/*------------------Ｅｘｔｅｒｎ　ｆｕｎｃｔｉｏｎｓ---------------------*/
extern double getdata(int,long,long);
extern void viewedge(void);
extern void pxlset(void);

/*------------------Ｅｘｔｅｒｎ　Ｖａｒｉａｂｌｅｓ---------------------*/
extern Vector eye, view;
extern Matrix lens, inv;
extern int px, py;
extern double window, sx, sy, hd, hv;

/*--------------------------プロトタイプ宣言-----------------------------*/
void shade(long,long,double);
static double ipow(double,int);
void scanpy(void);
void fnorm(long,long,double);
void viewv(void);

/*----------------------------Ｐｕｂｌｉｃ-------------------------------*/

Surface surf = {          /* サンプル表面の性質 */
    {0.0, 0.5, 0.0},      /* 拡散反射係数　Ｒ，Ｇ，Ｂ */
    {0.0, 1.5, 0.0},      /* 鏡面反射係数　Ｒ，Ｇ，Ｂ */
    500,                  /* 小面分布関数のパラメータ */
};

Illum lamp0 = {
    NULL,
    {0.0, -1.0, 1.0},     /* 光源の向き */
    {10.0, 10.0, 10.0},   /* 光源の色　Ｒ，Ｇ，Ｂ */
};

Vector norm;
Color ccolor;             /* ペイントする色のＲ，Ｇ，Ｂ成分 */

Surface *s = &surf;
Illum *l = &lamp0;

/*------------------------------コード---------------------------------*/
/*-----------------------------------------------------------------------+

    関数名        static double ipow(double a,int n)
```

```
    コメント        引数ａのｎ（整数）乗を計算する
                    Ｃプログラムブック２参照

+-----------------------------------------------------------------------*/
static double ipow(double a,int n)
{
    double ans;

    ans = 1.0;
    a = (n > 0) ? a : (1.0 / a);
    for (n = abs(n); n; n >>=1) {
        if (n & 0x01)
            ans *=a;
        a *= a;
    }
    return(ans);
}

/*-----------------------------------------------------------------------+

    関数名      void shade(long x,long y,double hd)

    コメント        引数で指定されたデータ座標の点をシェーディングする
                    Ｃプログラムブック２参照

+-----------------------------------------------------------------------*/
void shade(long x,long y,double hd)
{
    double a, b;

    fnorm(x,y,hd);

    vpsh(l->dist);
    vnorm();
    vpsh(norm);
    if ((a = vcos()) >0.0) {
        vpsh(l->dist);
        vpsh(view);
        vsub();
        vpsh(norm);
        b = vcos();

        cpsh(s->icol);
        cpsh(s->ocol);
        cmul(ipow(b, s->n));
        cadd();
        cmul(a);
        vpop(ccolor);
        pxlset();
    } else {
        cpsh(vzero);
        cpop(ccolor);
    }
}

/*-----------------------------------------------------------------------+

    関数名      void fnorm(long lx,long ly,double hdn)
```

コメント　　引数で指定されたベクトルを大きさ１に規格化する

```
+-----------------------------------------------------------------------*/
void fnorm(long lx,long ly,double hdn)
{
    static double hdx, hdy;

    hdx = (double)getdata(9,(long)(lx+1),ly);
    hdy = (double)getdata(9,lx,(long)(ly+1));

    vpshi((Element)1.0,(Element)0.0,(Element)hdx-hd);
    vpshi((Element)0.0,(Element)1.0,(Element)hdy-hd);
    vvpro();
    vnorm();
    vpop(norm);
}

/*-----------------------------------------------------------------------+

    関数名      void scanpy(void)

    コメント      ｙピクセルを１つ進め、そのときのスクリーン座標のｙ成分を
                計算する

+-----------------------------------------------------------------------*/
void scanpy(void)
    {
    py = py - 1;
    sy = (double)(py - 200.0)/200.0;
}

/*-----------------------------------------------------------------------+

    関数名      void scanpy(void)

    コメント      視点からスクリーン座標のｘ，ｙ成分の位置を通して
                データを眺めるときの視線の向きのベクトル view を計算する

+-----------------------------------------------------------------------*/
void viewv(void)
{
    mpsh(lens);
    vpshi((double)(window*sx),(double)(window*sy),(double)1.0);
    mvmul();
    vnorm();
    vpop(view);
}
/*
+-----------------------------------------------------------------------+
|                              end of shade.c                           |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                pset.c                                 |
|                             --------------                            |
|                                                                       |
|          拡張グラフィクスドライバを呼び出して点を描画するモジュール          |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include <graph.h>
#include <math.h>

#include "vmath.h"
#include "video.h"

#define C_STEP 255            /* kido controll */
/*
+-----------------------------------------------------------------------+
|                              外部変数                                 |
+-----------------------------------------------------------------------+
*/
extern Vector eye, view, lens, inv;
extern Color ccolor;
extern int px, py, number_of_samples;
extern double window, sx, sy, hd, hv, height_rate;
/*
+-----------------------------------------------------------------------+
|                           プロトタイプ宣言                            |
+-----------------------------------------------------------------------+
*/
void q_paint(void);
void pxlset(void);
int rnum(void);
/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名       void q_paint(void)

    コメント       クイックペイントで、データの高さ（グローバル hd）により
                色を変えて、ピクセル px,py（グローバル変数)に点を描画する

+-----------------------------------------------------------------------*/
void q_paint(void)
{
    int palet_no;

    palet_no = (int)(hd / ((double)number_of_samples * height_rate) * 255);

    if(palet_no > 255){
        palet_no = 255;
    }else if(palet_no < 0){
        palet_no = 0;
```

```
    }

    set_box_fill(px,py,px,py,palet_no);
}

/*-----------------------------------------------------------------------+

    関数名      void pxlset(void)

    コメント      シェーディングの場合、グローバル変数 b より色を決め
                ピクセル px,py (グローバル変数)に点を描画する。その際
                ディザリング (乱数により色の境界をぼんやりさせる) をしている。
                Cプログラムブック2参照

+-----------------------------------------------------------------------*/
void pxlset(void)
{
    int b,rndam;
    double bd, r;
    int dot;

    vpsh(ccolor);
    bd = sqrt(vabs2()) * C_STEP;
    b = (int)bd;

    r = (double)((rndam = rnum()) & 0xff) / 256.0;
    if ((bd - b)>r)
        ++b;

    if(b > 255)
        b = 255;

    set_box_fill(px,py,px,py,b);
}

/*-----------------------------------------------------------------------+

    関数名      int rnum(void)

    コメント    randum number generation bu M-keiretsu
                上記のディザリングのための乱数を発生させる関数
                Cプログラムブック2参照

+-----------------------------------------------------------------------*/
#define MASK 0x40004000L

int rnum(void)
{
    static long rnd = 12345678L;

    rnd <<= 1;
    if ((rnd & MASK) == 0 || (rnd & MASK) == MASK)
        rnd++;
    return ((int) rnd);
}
/*
+-----------------------------------------------------------------------+
|                              end of paint.c                           |
```

```
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              save_img.c                               |
|                            --------------                             |
|                                                                       |
|            ２５６色画像を６４０＊４８０ドットでセーブする。            |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <process.h>        /* exit() */
#include <dos.h>            /* _PF_OFF, _FP_SEG */
#include <string.h>         /* strcpy() */
#include <malloc.h>         /* malloc() */
#include <lib¥crt.h>        /* tinit(), tterm() */
#include <lib¥window.h>     /* winopen() */

#include "afm_def.h"
#include "txcondef.h"
#include "video.h"
#include "popup.h"

/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name, char parameter[][40], int dummy_no);
/*----------------------------Ｐｕｂｌｉｃ-------------------------------*/

char    prm_file_name[80];  /* パラメータファイル名 */
short   cmd_line;           /* 1: コマンドラインからの立ち上げ */

FILE *fp;

WINDOW buf;

/*--------------------------------コード----------------------------------*/
/*-----------------------------------------------------------------------+

    関数名      static int arg(int argc, char *argv[])

    返値        １：コマンドラインから　０：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+-----------------------------------------------------------------------*/
static int arg(int argc, char *argv[])
{
    char arguments[30][40];
    int counter;

                                    /* デフルトの値の代入 */
    strcpy(arguments[0], "1");      /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1], "a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */
```

```
    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    strcpy(prm_file_name,arguments[1]);
    return( atoi(arguments[0]) );
}

/*-----------------------------------------------------------------------+

    関数名      static int set_saveimg_param(void)

    コメント      パラメータファイルからパラメータを読み込み、
                ウインドウを開き、ファイル名をたずねる。

+-----------------------------------------------------------------------*/
static int set_save_img_prm(void)
{
    char parameter[40][40],imput_from_key[40],file_name[20] = {""};
    int number_of_prm;

    number_of_prm = read_prm_file(prm_file_name,parameter,IMAGE_PART);

    WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                    POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"ファイル名");

    Cslon();
    while(1){
        WINlocate(&buf,1,1);
        WINputstr(&buf,"ファイル名：拡張子付き *.i*¥n");
        WINdprintf(&buf,"%s",parameter[0 + number_of_prm]);
        WINdprintf(&buf,"%s",file_name);
        gets(imput_from_key);
        if(*imput_from_key != 0){
            strcpy(file_name,imput_from_key);
        }
        WINputstr(&buf,"¥n以上でよろしいですね    ");
        gets(imput_from_key);
        if(imput_from_key[0] == 0 || imput_from_key[0] == 'y' ||
                                        imput_from_key[0] == 'Y'){
            strcat(parameter[0 + number_of_prm],file_name);
            break;
        }
    }
    WINclose(&buf);
    Csloff();

    if((fp = fopen(parameter[0 + number_of_prm],"w+b")) == NULL){
        char msg[200];

        strcpy(msg,parameter[0 + number_of_prm]);
        strcat(msg,"¥nがオープンできません");
        errer_msg(msg);
        putchar('¥a');
        end_job();
    }
}
```

```
/*-----------------------------------------------------------------------+

    関数名        void save_image(void)

    コメント        グラフィック画面を切り取りセーブする

+-----------------------------------------------------------------------*/
void save_image(void)
{
    int result,file_size[2] = {640,480},*data_buf,i;
    unsigned int seg_buf,off_buf;

    result = fwrite(file_size,sizeof(int),(size_t)2,fp);
        /* 画像ファイルのフォーマット・最初の４バイトは画面のサイズを示す */

    data_buf = (int *)malloc((size_t)26000);
                                        /* 画像を読み込むメモリーを確保 */
    for (i = 0 ; i < 12 ; i++) {        /* 画面を縦方向に１２分割してセーブ */
        seg_buf = _FP_SEG(data_buf);
        off_buf = _FP_OFF(data_buf);
        get_pattern(640,40,0,40 * i,off_buf,seg_buf,0);
                                        /* グラフィクスドライバー AVGDRV.C */
        result = fwrite(data_buf,(int)1,(size_t)25600,fp);
    }
    free(data_buf);                     /* メモリー解放 */
    fclose(fp);
}

/*-----------------------------------------------------------------------+

    関数名        void start_job(int argc,char *argv[])

    コメント        グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
void start_job(int argc,char *argv[])
{
    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */
    init_key();

    cmd_line = arg(argc,argv);
    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    set_save_img_prm();
}

/*-----------------------------------------------------------------------+

    関数名        int end_job(void)

    コメント        コマンドラインからたちあげた場合のみ
                望洋ライブラリーとドライバの使用を終了する。

+-----------------------------------------------------------------------*/
```

```
int end_job(void)
{
    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

void main(int argc,char *argv[])
{
    start_job(argc,argv);           /* 初期化関数の呼び出し */
    save_image();
    end_job();
}
/*
+-------------------------------------------------------------------------+
|                                 終わり                                  |
+-------------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              load_img.c                               |
|                            --------------                             |
|                                                                       |
|            ２５６色画像を６４０＊４８０ドットでロードする。            |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <process.h>
#include <dos.h>            /* _PF_OFF,_FP_SEG */
#include <string.h>         /* strcpy() */
#include <conio.h>          /* kbhit() */
#include <malloc.h>         /* malloc() */

#include "afm_def.h"
#include "txcondef.h"
#include "video.h"

/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern FILE *open_file(FILE **f_ptr,char *search_path,char *wild_name);

/*----------------------------Ｐｕｂｌｉｃ------------------------------*/

char    prm_file_name[80];  /* パラメータファイル名 */
short   cmd_line;           /* 1: コマンドラインからの立ち上げ */

FILE *fp;

/*-------------------------------コード----------------------------------*/

/*-----------------------------------------------------------------------+

    関数名      static int arg(int argc,char *argv[])

    返値        １：コマンドラインから　０：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+-----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[40][40];
    int counter;

                                    /* デフルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */

    for(counter = 1;counter < argc;counter++){
```

```
        strcpy(arguments[counter - 1],argv[counter]);
    }

    strcpy(prm_file_name,arguments[1]);
    return( atoi(arguments[0]) );
}

/*-----------------------------------------------------------------------+

    関数名      void open_file(void)

    コメント      パラメータファイルに指定されたファール検索パスから、
                ファイル名をたずね、ファイルをオープンする関数を呼ぶ。

+-----------------------------------------------------------------------*/
FILE *select_file(void)
{
    char parameter[30][40],*file_name,name_buf[80];
    int number_of_prm,result;

    number_of_prm = read_prm_file(prm_file_name,parameter,IMAGE_PART);
    return ( open_file(&fp,parameter[0 + number_of_prm],"*.i*") );
                            /* 画像ファイルのデフォルトの拡張子 *.i* */
}

/*-----------------------------------------------------------------------+

    関数名      void load_img(void)

    コメント      グラフィック画面をロードする

+-----------------------------------------------------------------------*/
void load_img(void)
{
    int result,size[2],*data_buf,i;
    unsigned int seg_buf,off_buf;

    result = fread(size,sizeof(int),(size_t)2,fp);
                                    /* 最初の４バイトは画面サイズ */
    if(size[0] != 640){
        char msg[200];

        strcat(msg,"このファイルは画像ファイルではありません¥n");
        errer_msg(msg);
        putchar('¥a');
        end_job();
    }

    data_buf = (int *)malloc((size_t)26000);
                                    /* 画像を読み出すメモリーを確保 */

    for (i = 0 ; i < 12 ; i++) {        /* 画面を縦方向に１２分割してロード */
        result = fread(data_buf,(int)1,(size_t)25600,fp);
        seg_buf = _FP_SEG(data_buf);
        off_buf = _FP_OFF(data_buf);
        set_pattern(640,40,off_buf,seg_buf,0,40 * i,0);
                                    /* グラフィクスドライバー AVGDRV.C */
    }
```

```
    free(data_buf);                     /* メモリー解放 */
    fclose(fp);
}

/*-----------------------------------------------------------------------+

    関数名      static void start_job(int argc,char *argv[])

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
static void start_job(int argc,char *argv[])
{
    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */
    init_key();
    Cls();

    cmd_line = arg(argc,argv);
    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        set_256palet(MONOCHROME);   /* ２５６色パレットの設定 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    if( select_file() == NULL)
        end_job();
}

/*-----------------------------------------------------------------------+

    関数名      int end_job(void)

    コメント      ポーズ後、コマンドラインからたちあげた場合のみ
                望洋ライブラリーとドライバの使用を終了する。

+-----------------------------------------------------------------------*/
int end_job(void)
{
    getch();

    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

void main(int argc,char *argv[])
{
    start_job(argc,argv);                   /* 初期化関数の呼び出し */
    load_img();
    end_job();
}
```

```
/*
+-----------------------------------------------------------------------+
|                                                                       |
|                              line.c                                   |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include <stdlib.h>             /*abs()*/

#include "afm_def.h"
#include "vmath.h"

/*------------------Ｅｘｔｅｒｎ　ｆｕｎｃｔｉｏｎｓ---------------------*/

extern void edge(long,long);
extern void shade(long,long,double);
extern void pxlset(void);
extern void scanpy(void);
extern void viewv(void);
extern double getdata(int,long,long);

/*--------------------------プロトタイプ宣言----------------------------*/

void line(long,long,long,long);           /*proto*/
void point(long,long);
int comp(long,long);

/*---------------------------Ｐｕｂｌｉｃ-------------------------------*/

extern Vector eye, view;
extern Matrix lens, inv;
extern int px, py, mode;
extern short number_of_samples;
extern double window, sx, sy, data_min,hight_scale;
extern FILE *fp;

double hd, hv;

/*-------------------------------コード----------------------------------*/

/*-----------------------------------------------------------------------+

    関数名      void line(long x1,long y1,long x2,long y2)

    引数        long x1:視点から眺めたときの手前のデータの端のｘ座標
                long y1:視点から眺めたときの手前のデータの端のｙ座標
                long x2:視点から眺めたときの奥のデータの端のｘ座標
                long y2:視点から眺めたときの奥のデータの端のｙ座標

    コメント      離散データであるスキャンデータを、引数で指定される２点間を
                結ぶ直線に沿って順に進めてゆく
                Ｃプログラムブック１参照

+-----------------------------------------------------------------------*/
void line(long x1,long y1,long x2,long y2)
{
```

```
long x, y, dx, dy;
long e, c1, c2, inc;

x = x1; y = y1;

edge(x,y);                /* 視界の最下点の処理 */

dx = labs(x2 - x1);
dy = labs(y2 - y1);
if ( dx > dy){
    c2 = 2 * dy;
    e = c2 - dx;
    c1 = e - dx;
    if  (y1 < y2)
        inc = 1;
    else
        inc = -1;
    y = y1;

    if (x2 - x1 + y2 - y1 < 0
            || (x2 - x1 + y2 - y1 == 0 && x2 - x1 > 0)){
        for (x = x1 - 1; x >= x2; x--){
            if (e > 0){
                y += inc;
                e += c1;
            }else
                e += c2;
            point(x, y);    /* 描画処理 */
        }
    } else{
        for (x = x1 + 1; x <= x2; x++){
            if (e > 0){
                y += inc;
                e += c1;
            }else
                e += c2;
            point(x, y);    /* 描画処理 */
        }
    }
}else{
    c2 = 2 * dx;
    e = c2 - dy;
    c1 = e - dy;
    if (x1 < x2)
        inc = 1;
    else
        inc = -1;
    x = x1;

    if (x2 - x1 + y2 - y1 < 0
            || (x2 - x1 + y2 - y1 == 0 && x2 - x1 > 0)){
        for (y = y1 - 1; y >= y2; y--){
            if (e > 0){
                x += inc;
                e += c1;
            }else
                e += c2;
            point(x, y);    /* 描画処理 */
```

```
            }
        } else{
            for (y = y1 + 1; y <= y2; y++){
                if (e > 0){
                    x += inc;
                    e += c1;
                }else
                    e += c2;
                point(x, y);    /* 描画処理 */
            }
        }
    }
}

/*----------------------------------------------------------------------+

    関数名      int comp(long x,long y)

    引数        long x:データのｘ座標
                long y:データのｙ座標

    返値        ０：データの方が大きい
                １：視線の高さの方が大きい
                ２：両者が等しい

    コメント      現在の視線の引数で指定される座標(x,y)の時の高さと
                その座標のデータの高さとを比較する

+----------------------------------------------------------------------*/
int comp(long x,long y)
{
    int flag;

    hd = getdata(9,x,y);

    if(fabs(view[X]) > fabs(view[Y])){
        hv = eye[Z]+(double)((x-eye[X])*view[Z]/view[X]);
    }else{
        hv = eye[Z]+(double)((y-eye[Y])*view[Z]/view[Y]);
    }

    if(hd > hv)
        flag = 0;
    else if(hd < hv)
        flag = 1;
    else
        flag = 2;

    return(flag);
}

/*----------------------------------------------------------------------+

    関数名      void point(long x,long y)

    引数        long x:データのｘ座標
                long y:データのｙ座標
```

コメント　　指定された座標のデータの描画処理

```
+-----------------------------------------------------------------------*/
void point(long x,long y)
{
    int flag;

    flag = comp(x,y);
    if(flag == 1){              /* 視線の高さの方が大きい */
        return;                 /* 視線は低いデータを飛び越える */
    }
    if(flag == 2){              /* 両者が等しい */
        if(mode == QUICK_PAINT)
            q_paint();          /* クイックペイント */
        else if(mode == SHADING)
            shade(x,y,hd);      /* シェーディング */

        scanpy();               /* ｙピクセルを１つ進める */
        viewv();                /* そのときの視線を求める */
        return;
    }else{                      /* データの方が大きい */
        if(mode == QUICK_PAINT)
            q_paint();
        else if(mode == SHADING)
            shade(x,y,hd);

        scanpy();               /* ｙピクセルを１つ進める */
        viewv();                /* そのときの視線を求める */

        while((flag = comp(x,y)) == 0){     /* データの方がまだ大きい */
            if(mode == QUICK_PAINT)
                q_paint();
            else if(mode == SHADING)
                pxlset();

            scanpy();           /* ｙピクセルを１つ進める */
            viewv();            /* そのときの視線を求める */
        }
    }
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              img_util.c                               |
|                            ---------------                            |
|                                                                       |
|                 ３Ｄ画像を描く IMG.EXE のサブモジュール                |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include "vmath.h"

#define PRM_BYTE 40

#define change(mat, c1, c2)¥
        {¥
            Element tmp;¥
¥
            tmp = mat[c1][c2];¥
            mat[c1][c2] = mat[c2][c1];¥
            mat[c2][c1] = tmp;¥
        }

/*--------------------------プロトタイプ宣言------------------------------*/

double readdata(long dx,long dy);
double getdata(int average,long dx,long dy);
void fview(double,double,double);
void getpxel(void);
void calcu_lens(double);

/*----------------------------- 外部変数 --------------------------------*/

extern int      number_of_samples;
extern double   px,py,data_min,grad_x,grad_y,height_scale;
extern Vector   eye, cent;

extern FILE     *fp;

/*----------------------------Ｐｕｂｌｉｃ-------------------------------*/

double window;
Vector view;
Matrix lens, inv;

/*-------------------------------コード----------------------------------*/

/*-----------------------------------------------------------------------+

    関数名      double readdata(long dx,long dy)

    引数        long dx:    Ｘ座標
                long dy:    Ｙ座標

    返値        データ

    コメント      スキャンデータファイルから引数で指定された座標のデータを
```

読み出す

```
+-----------------------------------------------------------------------*/
double readdata(long dx,long dy)
{
    double height;
    float fh;
    int result;

    result = fseek(fp,(long)((dy*number_of_samples + dx)
                        * 4 + PRM_BYTE),0);
    result = fread((void *)&fh,4,1,fp);

    height = ((double)fh - data_min) * height_scale;
    return(height);
}

/*-----------------------------------------------------------------------+

    関数名      double getdata(long dx,long dy)

    引数        int:        平均の数
                long dx:    X座標
                long dy:    Y座標

    返値        データの平均値

    コメント      スキャンデータファイルから引数で指定された座標のデータを
                その付近１，４，９（引数）ポイントの平均を返す

+-----------------------------------------------------------------------*/
double getdata(int average,long dx,long dy)
{
    static double buff;

    buff = 0.0;

    buff += readdata(dx,dy);

    if(average > 1){
        average = 4;
        buff += readdata(dx+1,dy);
        buff += readdata(dx,dy+1);
        buff += readdata(dx+1,dy+1);
    }
    if(average > 4){
        average = 9;
        buff += readdata(dx+2,dy);
        buff += readdata(dx+2,dy+1);
        buff += readdata(dx,dy+2);
        buff += readdata(dx+1,dy+2);
        buff += readdata(dx+2,dy+2);
    }
    buff = buff / average - (dx * grad_x + dy * grad_y) * height_scale;
    return(buff);
}

/*-----------------------------------------------------------------------+
```

関数名　　void fview(double x,double y,double h)

引数　　double x:　データ空間上のX座標
　　double y:　データ空間上のY座標
　　double h:　データ空間上のZ座標

コメント　　引数で指定されたデータ空間上の座標を視点から眺めたときの視線の向きを示すベクトルをポップする

```
+-----------------------------------------------------------------------*/
void fview(double x,double y,double h)
{
    vpshi(x,y,h);
    vpsh(eye);
    vsub();
    vnorm();
    vpop(view);
}

/*-----------------------------------------------------------------------+
```

関数名　　void fview(double x,double y,double h)

コメント　　視線の向きを示すベクトル view とスクリーン平面との交点を計算する

```
+-----------------------------------------------------------------------*/
void getpxel(void)
{
    Vector upxel;
    double sx, sy;

    mpsh(inv);
    vpsh(view);
    mvmul();
    vpop(upxel);

    sx =  upxel[X]/upxel[Z]/window;
    sy =  upxel[Y]/upxel[Z]/window;

    px = 200.0 * sx + 320.0;
    py = 200.0 * sy + 200.0;
}

/*-----------------------------------------------------------------------+
```

関数名　　void calcu_lens(double zoom)

引数　　double zoom:　倍率

コメント　　スクリーン座標からデータ座標への一次変換 lenz とデータ座標からスクリーン座標への一次変換 inv を計算する。

```
+-----------------------------------------------------------------------*/
void calcu_lens(double zoom)
{
    window = 12.0 / zoom;
```

```
    sinit();

    vpsh(eye);              /* eye */
    vpsh(cent);             /* cent */
    vpsh(eye);
    vsub();                 /* direct */
    vpshi(0.0,0.0,1.0);     /* up */

    vpop(lens[Y]);
    vnorm();
    vpop(lens[Z]);

    vpsh(lens[Z]);
    vpsh(lens[Y]);
    vvpro();
    vnorm();
    vpop(lens[X]);

    vpsh(lens[Z]);
    vpsh(lens[X]);
    vvpro();
    vnorm();
    vpop(lens[Y]);

    mpsh(lens);             /* get inverse */
    mpop((Element *)inv);
    change(lens, X, Y);
    change(lens, Y, Z);
    change(lens, Z, X);

    vpop(eye);
}
/*
+-------------------------------------------------------------------------+
|                                 終わり                                  |
+-------------------------------------------------------------------------+
*/
```

```
/*
+-------------------------------------------------------------------------+
|                              img_main.c                                 |
|                             ------------                                |
|                                                                         |
|       拡張グラフィクスドライバーを使って、スキャンデータの              |
|     ワイヤーフレーム像、クイックペイント像、シェーディング像を描画する  |
|     メインモジュール                                                    |
|                                                                         |
+-------------------------------------------------------------------------+
*/
#include <stdio.h>
#include <process.h>
#include <stdlib.h>          /* atoi() */
#include <math.h>            /* atof() */
#include <conio.h>           /* getch() */
#include <string.h>          /* strcpy() */
#include <lib¥crt.h>         /* tinit(), tterm() */
#include <lib¥window.h>      /* winopen() */

#include "afm_def.h"
#include "popup.h"
#include "vmath.h"
#include "video.h"
#include "txcondef.h"

/*------------------Extern functions---------------------*/

extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                        short start,short stop);
extern FILE *open_temp_file(FILE *fp,char *prmfile_name,int part);
extern void calcu_lens(double);
extern void Axismain(void);             /* <==  modified by Kotaro */
extern void del_temp(char *prmfile_name,int part);

/*----------------------------Public------------------------------*/

char    prm_file_name[80];
short   number_of_samples,step,mode;
double  length_of_samples,data_max,data_min,px,py;/*<== modified by Kotaro*/
double  height_rate,height_scale,zoom,grad_x,grad_y;

struct dat_file_format{ /* スキャンデータファイルパラメータ構造体 */
    int     what_file;
    int     number_of_samples;
    int     scan_size;
    float   tilt_x;
    float   tilt_y;
    float   max_dat;
    float   min_dat;
} dat_file;

FILE *fp;

WINDOW buf;

Vector eye, cent;   /* データ空間上の視線の位置と、データの中心座標 */
```

```
/*-------------------------------コード---------------------------------*/
/*----------------------------------------------------------------------+

    関数名      static int arg(int argc,char *argv[])

    返値        1：コマンドラインから　0：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[30][40],imput_from_key[20];
    int counter;

                                        /* デフルトの値の代入 */
    strcpy(arguments[0],"1");           /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                        /* パラメータファイル名 */
    strcpy(arguments[2],"1");           /* 1 : ワイヤーフレーム */

    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    if(argc <= 2){                      /* コマンドライン引数が足りないとき */
        WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                            POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
        WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
        WINputstr(&buf,"デフォルト : リターンキー¥n");
        Cslon();
        while(1){
            switch(argc){
            case 1:
            case 2:
            case 3:
                WINlocate(&buf,1,2);
                WINputstr(&buf,
                "1: ワイヤーフレーム  2: クイックペイント¥n"
                "3: シェーディング  ");
                WINdprintf(&buf,"%s  ",arguments[2]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[2],imput_from_key);
                }
            default:
                break;
            }

            WINputstr(&buf,"¥n以上でよろしいですね    ");
            gets(imput_from_key);
            if(imput_from_key[0] == 0){
                WINclose(&buf);
                break;
            }
        }
```

```
        Csloff();
    }
    strcpy(prm_file_name,arguments[1]);
    mode = atoi(arguments[2]);
    return( atoi(arguments[0]) );
}

/*-----------------------------------------------------------------------+

    関数名      void set_img_param(void)

    コメント      パラメータファイルからスキャンのパラメータを読み込み、
                ウインドウを開き、デフォルトの値の変更をたずねる。

+-----------------------------------------------------------------------*/
void set_img_param(void)
{
    char parameter[40][40],imput_from_key[40];
    int number_of_prm,no;

    number_of_prm = read_prm_file(prm_file_name,parameter,IMAGE_PART);

    if(mode == WIRE_FRAME)
        no = 6;
    else
        no = 5;

    ask_prm(&buf,"描画の条件",parameter,number_of_prm,1,no);

    eye[X] = atof(parameter[1 + number_of_prm]);
    eye[Y] = atof(parameter[2 + number_of_prm]);
    eye[Z] = atof(parameter[3 + number_of_prm]);
    zoom = atof(parameter[4 + number_of_prm]);
    height_rate = atof(parameter[5 + number_of_prm]);   /* 横に対する縦の割合*/
    step = atoi(parameter[6 + number_of_prm]);  /* ワイヤーフレームの線間隔 */
}

/*-----------------------------------------------------------------------+

    関数名      void read_prm(void)

    コメント      スキャンデータファイルからのパラメータを読み込む。

+-----------------------------------------------------------------------*/
void read_prm(void)
{
    int file_type;

    read_prm_from_file(&dat_file);

    file_type           = dat_file.what_file;
    number_of_samples   = dat_file.number_of_samples;
    length_of_samples   = dat_file.scan_size;
    grad_x              = dat_file.tilt_x;
    grad_y              = dat_file.tilt_y;
    data_max            = dat_file.max_dat;
    data_min            = dat_file.min_dat;
```

```
    if(file_type != SCAN_DAT_FILE){
        char msg[200];

        strcpy(msg,"このファイルはスキャンデータの¥nファイルではありません");
        errer_msg(msg);
        puts("¥a");
        putchar('¥a');
        end_job();
    }

    height_scale = height_rate * number_of_samples
                                    / (data_max - data_min);

    cent[X] = (double)(dat_file.number_of_samples / 2);
    cent[Y] = (double)(dat_file.number_of_samples / 2);
    cent[Z] = ((data_max + data_min) / 2 - data_min) * height_scale;
}

/*-----------------------------------------------------------------------+

    関数名      static int start_job(int argc,char *argv[])

    返値        １：コマンドラインから　０：メニューから　立ちあげ

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
static int start_job(int argc,char *argv[])
{
    short flag;

    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */
    cls_256();                      /* グラフィクス画面のクリア */
    Cls();                          /* クリアスクリーン */
    set_line_style(0xffff);         /* ラインを実線にセット */
    init_key();                     /* キーテーブルの初期化 */

    flag = arg(argc,argv);          /* 引数の処理 */
    if(flag){                       /* コマンドラインから立ち上げた場合 */
        set_256palet(MONOCHROME);   /* ２５６色パレットの設定 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    if( (fp = open_temp_file(fp,prm_file_name,(int)IMAGE_PART)) == NULL )
        end_job(flag);              /*   描画するデータファイルを
                                       ラムディスクにコピー */
    Cls();
    set_img_param();
    read_prm();
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      int end_job(short flag)

    コメント      データファイルの先頭にファイル情報を書き込み、
```

リペイント後、ポーズ。そのご、データファイルをクローズし、
ウインドウを閉じる。

```
+-----------------------------------------------------------------------*/
int end_job(short flag)
{
    fclose(fp);
    del_temp(prm_file_name,(int)IMAGE_PART);    /* テンポラリファイルの消去 */
    Cls();
    getch();                                    /* ポーズ */

    if(flag){                       /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
        Cls();                      /* クリアスクリーン */
    }
    exit(0);
}

void main(int argc,char *argv[])
{
    short cmd_line;                         /* 1: コマンドラインから立ち上げ */

    cmd_line = start_job(argc,argv);        /* 初期化関数の呼び出し */

    calcu_lens(zoom);                       /* 一次変換の行列を計算する。*/
    if(mode == WIRE_FRAME){                 /* ワイヤーフレーム */
        wf_scan(step);
    }else if(mode == QUICK_PAINT || mode == SHADING){
        scan_pxel(cmd_line);
    }
    Axismain();                             /* 座標軸を描画 */
    end_job(cmd_line);                      /* 終了処理 */
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                grid.c                                 |
|                             --------------                            |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <conio.h>          /* getch() */
#include <stdlib.h>         /* atoi(), atol(), exit() */
#include <math.h>           /* atof() */
#include <string.h>         /* strcpy(); */
#include <lib¥crt.h>        /* tinit(), tterm() */
#include <lib¥window.h>     /* winopen() */

#include "afm_def.h"
#include "popup.h"
#include "video.h"
#include "txcondef.h"

/*---------------------------Ｐｕｂｌｉｃ-------------------------------*/

char    prm_file_name[80];
float       div_no;

WINDOW buf;     /* 望洋ライブラリーのポップアップウインドウ用変数 */

/*-------------------------------コード----------------------------------*/
/*-----------------------------------------------------------------------+

    関数名      static int arg(int argc,char *argv[])

    返値        １：コマンドラインから　０：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。

+-----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[20][40],imput_from_key[80];
    int counter;

                                    /* デフォルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */
    strcpy(arguments[2],"10");      /* 領域の分割数 */

    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    if(argc <= 3){                  /* コマンドライン引数が足りないとき */
        WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
        WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
        WINputstr(&buf,"デフォルト : リターンキー¥n");
        Cslon();
```

```
        while(1){
            switch(argc){
            case 1:
            case 2:
            case 3:
                WINlocate(&buf,1,2);
                WINputstr(&buf,
                "領域の分割数     ");
                WINdprintf(&buf,"%s  ",arguments[2]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[2],imput_from_key);
                }
            default:
                break;
            }

            WINputstr(&buf,"¥n以上でよろしいですね     ");
            gets(imput_from_key);
            if(imput_from_key[0] == 0){
                WINclose(&buf);
                break;
            }
        }
        Csloff();
    }
    strcpy(prm_file_name,arguments[1]);
    div_no = atof(arguments[2]);
    return( atoi(arguments[0]) );
}

/*-----------------------------------------------------------------------+

    関数名      static int start_job(int argc,char *argv[])

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
static char start_job(int argc,char *argv[])
{
    char flag = 0x00;

    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */

    if( arg(argc,argv) ){
        flag = flag | 0x01;         /* コマンドラインからの立ちあげ */
        Cls();
        set_256palet(MONOCHROME);           /* ２５６色パレットの設定 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      int end_job(void)
```

コメント　　データファイルの先頭にファイル情報を書き込み、
リペイント後、ポーズ。そのご、データファイルをクローズし、
ウインドウを閉じる。

```
+-----------------------------------------------------------------------*/
int end_job(char flag)
{
    if(flag & 0x01){                /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

void main(int argc,char *argv[])
{
    char result;
    float interval;

    result = start_job(argc,argv);              /* 初期化関数の呼び出し */

    interval = SAMPLING_MAX / div_no;

    set_grid(TOP_VIEW_WINDOW_X+1,TOP_VIEW_WINDOW_Y+1,SAMPLING_MAX,
        interval,0xf0f0,100,4);
    getch();
    set_grid(TOP_VIEW_WINDOW_X+1,TOP_VIEW_WINDOW_Y+1,SAMPLING_MAX,
        interval,0xf0f0,100,4);

    end_job(result);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                                                                       |
|                              edge.c                                   |
|                                                                       |
|                 ３Ｄ画像を描く IMG.EXE のサブモジュール               |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <conio.h>
#include <stdio.h>
#include <process.h>

#include "afm_def.h"
#include "vmath.h"

#define swap(a, b) {long temp; temp = a; a = b; b = temp;}
#define EDGE 5      /* 2 */

/*------------------Ｅｘｔｅｒｎ　ｆｕｎｃｔｉｏｎｓ---------------------*/

extern void line(long,long,long,long);
extern int  comp(long,long);
extern void q_paint(void);
extern void shade(long,long,double);
extern void scanpy(void);
extern void viewv(void);
extern int  end_job(int);

/*--------------------------プロトタイプ宣言----------------------------*/

void scan_pxel(int flag);
void viewedge(void);
void edge(long,long);
void getypxel(long,long,double);
void clip(long,long,long,long);
int looker(long,long);
void frameview(long,long);

/*----------------------------Ｐｕｂｌｉｃ-------------------------------*/

extern Vector eye;
extern Matrix lens, inv;
extern int px, py, mode;
extern short number_of_samples;
extern double window, hd, hv;

double sx, sy;

Vector view;

/*------------------------------コード----------------------------------*/

/*-----------------------------------------------------------------------+

    関数名      void scan_pxel(int flag)

    引数        int flag:   １：コマンドラインから　０：メニューから　立ちあげ
```

コメント　　スクリーンのX座標 px を進めていく

```
+-----------------------------------------------------------------------*/
void scan_pxel(int flag)
{
    for(px = 320;px >= 0;px--){          /* スクリーンの中央から左を描く */
        if(kbhit() != 0)                 /* キー入力があった場合、終了 */
            end_job(flag);
        viewedge();
        sinit();                         /* ベクトル演算のスタックをクリア */
    }
    for(px = 321;px < 640;px++){         /* スクリーンの中央から右を描く */
        if(kbhit() != 0)                 /* キー入力があった場合、終了 */
            end_job(flag);
        viewedge();
        sinit();                         /* ベクトル演算のスタックをクリア */
    }
}

/*-----------------------------------------------------------------------+
```

関数名　　void viewedge(void)

コメント　　視点からスクリーンのフレームを通してデータ空間を眺めるとき、視点と、スクリーンの縦１ライン(extern int px)の上と下の隅を通る直線と、データ平面が交わる２点を求める

```
+-----------------------------------------------------------------------*/
void viewedge(void)
{
    long x1,y1, x2,y2;
    /*double dx1,dy1, dx2,dy2;*/
    Vector farview;

    /* sx, sy はスクリーン座標系のｘ、ｙ座標で
        sx は 0, 640 ドットが -1.6, 1.6 に
        sy は 0, 480 ドットが -1.0, 1.4 に対応する */
    sx = (double)((px - 320) / 200.0);
    sy = (double)1.4;                    /* 480 - 200ピクセル / 200ピクセル */

    mpsh(lens);
    vpshi(window * sx, window * sy, 1.0);
    mvmul();
    vnorm();
    vpop(view);          /* スクリーンフレームの下の隅のベクトル */

    sy = (double)-1.0;                   /* 0 - 200ピクセル / 200ピクセル */
    mpsh(lens);
    vpshi(window * sx, window * sy, 1.0);
    mvmul();
    vnorm();
    vpop(farview);       /* スクリーンフレームの上の隅のベクトル */

    /* データ平面との交点を求める。0.5を加えてキャストする事で四捨五入*/
    x1 = (long)(eye[X]-eye[Z]*view[X]/view[Z] + 0.5);
    y1 = (long)(eye[Y]-eye[Z]*view[Y]/view[Z] + 0.5);
    x2 = (long)(eye[X]-eye[Z]*farview[X]/farview[Z] + 0.5);
```

```
    y2 = (long)(eye[Y]-eye[Z]*farview[Y]/farview[Z] + 0.5);

    clip(x1,y1,x2,y2);        /* ２点間をデータが存在する領域にクリップ */
}

/*-----------------------------------------------------------------------+

    関数名        void clip(long x1,long y1,long x2,long y2)

    コメント        データ平面上の２点間をデータが存在する領域にクリップする
                    その際視点から見て手前の点と奥の点が入れ替わらないように
                    している
                    Ｃプログラムブック１参照

+-----------------------------------------------------------------------*/
void clip(long x1,long y1,long x2,long y2)
{
    long c1, c2;
    int /*and,*/r = 0;

    c1 = looker(x1, y1);
    c2 = looker(x2, y2);

    for ( ; ; ){
        if (c1 == 0 && c2 == 0){
            if(r % 2 == 1){
                swap(x1,x2);
                swap(y1,y2);
            }
            line(x1, y1, x2, y2);
            return;
        }

        if (c1 & c2)
            break;

        if (c2 == 0) {
            swap(x1, x2);
            swap(y1, y2);
            swap(c1, c2);
            ++r;
        }
        if ( c2 & 1) {
            y2 = y2 + (y1 - y2) * (0 - x2) / (x1 - x2);
            x2 =  0;
        }else if (c2 & 2) {
            y2 = y2 + (y1 - y2)¥
            * ((long)number_of_samples - EDGE - x2) / (x1 - x2);
            x2 = number_of_samples - EDGE;
        } else if (c2 & 4) {
            x2 =  x2 + (x1 - x2) * (0 - y2) / (y1 - y2);
            y2 = 0;
        }else {
            x2 = x2 + (x1 - x2)¥
            * ((long)number_of_samples - EDGE - y2) / (y1 - y2);
            y2 = number_of_samples - EDGE;
        }
        c2 = looker(x2, y2);
```

```
    }
}

/*-----------------------------------------------------------------------+

    関数名        int looker(long x,long y)

    コメント        データ平面上の２点がどの領域にあるかを判断する
                Ｃプログラムブック１参照

+-----------------------------------------------------------------------*/
int looker(long x,long y)
{
    int c;

    c = 0;
    if (x < 0)
        c |= 1;
    if ((long)number_of_samples < x)
        c |= 2;
    if (y < 0)
        c |= 4;
    if ((long)number_of_samples < y)
        c |= 8;
    return (c);
}

/*-----------------------------------------------------------------------+

    関数名        void edge(long x,long y)

    コメント        視点からスクリーンフレームを通してデータを見たときの
                視界の下隅の処理をする

+-----------------------------------------------------------------------*/
void edge(long x,long y)
{
    int flag = comp(x,y);

    if(flag == 0){              /* 視線よりデータが上にあるとき */
        frameview(x,y);         /* データの端を見る向きのデータ空間上の
                                    ベクトルを求める */
        getypxel(x,y,hd);       /* そのベクトルがスクリーンと交わる点の
                                    ピクセルを得る */
        if(mode == QUICK_PAINT)
            q_paint();
        else if(mode == SHADING)
            shade(x,y,hd);
        scanpy();               /* ｙピクセルを１つ進める */
        viewv();                /* そのときの視線を求める */
        return;
    }else if(flag == 2){        /* 視線とデータが同じ高さにあるとき */
        if(mode == QUICK_PAINT)
            q_paint();
        else if(mode == SHADING)
            shade(x,y,hd);
        scanpy();               /* ｙピクセルを１つ進める */
        viewv();                /* そのときの視線を求める */
```

```
        return;
    }else                          /* 視線よりデータが下にあるとき */
        return;
}

/*-----------------------------------------------------------------------+

    関数名      void getypxel(long x,long y,double hd)

    コメント      視線のベクトルがスクリーンと交わる点のピクセルを求める

+-----------------------------------------------------------------------*/
void getypxel(long x,long y,double hd)
{
    Vector upxel;

    mpsh(inv);
    vpsh(view);
    mvmul();        /* 視線をデータ空間からスクリーン空間へ一次変換 */
    vpop(upxel);

    sy = upxel[Y]/upxel[Z]/window;  /* 変換したベクトルのｙ軸成分を規格化して
                                       取り出す */
    py = (int)(200 * sy + 200 + 0.5);   /* その成分をピクセル値に直す
                                           0.5をたして四捨五入 */
}

/*-----------------------------------------------------------------------+

    関数名      void frameview(long x,long y)

    引数        long x: データ空間のｘ座標
                long y: データ空間のｙ座標

    コメント      引数と外部変数 hd で指定されるデータ空間上の点を
                視点から眺めるときのベクトルを求める

+-----------------------------------------------------------------------*/
void frameview(long x,long y)
{
    vpshi((Element)x,(Element)y,hd);
    vpsh(eye);
    vsub();
    vnorm();
    vpop(view);
}

/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              axisp02.c                                |
|                            --------------                             |
|                                                                       |
|          ３Ｄ画像に座標軸を書き込むファイル　Ｖｅｒ．０．２            |
|                                          ｂｙ　清水　光太朗           |
|                                                                       |
|       (注) 中野さんの作った３Ｄ画像描画プログラムは、焦点を持つ       |
|            投影技法を用いている。この axisp02.c のプログラムの        |
|            目盛りを打つアルゴリズムは、これに対応していない。          |
|            また、いくつかの関数は、改善の余地がある。                  |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <math.h>
#include "video.h"

#define  AXISMAX  10
#define  DVXAXS   (double)4.0          /* ｘ軸方向の分割数 */
#define  DVYAXS   (double)4.0          /* ｙ               */
#define  DVZAXS   (double)1.0          /* ｚ               */
#define  LNCOL    (unsigned long)255   /* 軸線の色   */
#define  LNWID    (unsigned int)0      /* 軸線の太さ */
#define  X        0
#define  Y        1

/*
+-----------------------------------------------------------------------+
|                            構造体定義                                 |
+-----------------------------------------------------------------------+
*/
struct point {
    double  spcx;   /* ３次元空間上の点のｘ座標 */
    double  spcy;   /*                  ｙ     */
    double  spcz;   /*                  ｚ     */
    double  plnx;   /* ２次元平面上の点のｘ座標 */
    double  plny;   /*                  ｙ     */
} pnt[AXISMAX];

struct onescale {
    double  x;      /* ｘ軸の最初の目盛りの３次元空間上でのｘ座標 */
    double  y;      /* ｙ                                 ｙ     */
    double  z;      /* ｚ                                 ｚ     */
} onescl;

/*
+-----------------------------------------------------------------------+
|                    外 部 変 数  使 用 宣 言                           |
+-----------------------------------------------------------------------+
*/
extern short  number_of_samples;
extern double length_of_samples,data_max,data_min,height_scale,px,py;
extern double eye[],cent[];

/*
+-----------------------------------------------------------------------+
|                    外 部 関 数  使 用 宣 言                           |
```

```
+-----------------------------------------------------------------------------+
*/
extern double getdata(int, long, long);
extern void   fview(double, double, double);
extern void   getpxel(void);

/*
+-----------------------------------------------------------------------------+
|                          プロトタイプ宣言                                   |
+-----------------------------------------------------------------------------+
*/
double DSign(double);
long DLRound(double);
void InputSpcp(double, double, double);
void ConvAxis(void);
void RotAxis(double [], double []);
void RotPoint(double, double, double, double *, double *);
double Asclp(double, double, double);
void Subasp(double, double *, double *);
void DrawAxis(void);
void DrawSclp(void);
void Axismain(void);

/*
+-----------------------------------------------------------------------------+
|                            ソースコード                                     |
+-----------------------------------------------------------------------------+
*/
/*
+-----------------------------------------------------------------------------+
|                                                                             |
|   関数名    double Dsign(val)                                               |
|                                                                             |
|   引き数    double val : double型の任意の値                                 |
|                                                                             |
|   返り値     1.0 : 引き数 val が、正の数                                    |
|             -1.0 : 引き数 val が、負の数                                    |
|              0.0 : 引き数 val が、0                                         |
|                                                                             |
|   コメント  引き数 val の符号を返す。                                        |
|                                                                             |
+-----------------------------------------------------------------------------+
*/
double DSign(double val)
{
    double sgn;

    if (val>0.0)       sgn = 1.0;
    else if (val<0.0)  sgn = -1.0;
    else if (val==0.0) sgn = 0.0;
    return(sgn);
}
/*
+-----------------------------------------------------------------------------+
|                                                                             |
|   関数名    long DLRound(val)                                               |
|                                                                             |
|   引き数    double val : double型の任意の値                                 |
```

```
|                                                                          |
|   返り値     引き数 val を四捨五入した値 (long型)                         |
|                                                                          |
|   コメント   関数 double DSign() を使用。                                 |
|                                                                          |
+--------------------------------------------------------------------------+
*/
long DLRound(double val)
{
    return((long)(val+DSign(val)*0.5));
}

/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名     void InputSpcp(xds,yds,zds)                                 |
|                                                                          |
|   引き数     double xds : ｘ軸の最初の目盛りの３次元空間上でのｘ座標      |
|                     yds : ｙ軸の最初の目盛りの３次元空間上でのｙ座標      |
|                     zds : ｚ軸の最初の目盛りの３次元空間上でのｚ座標      |
|                                                                          |
|   返り値     なし                                                        |
|                                                                          |
|   コメント   座標軸を書くために必要な３次元空間上の点の座標を、           |
|              構造体 pnt[].spcx,spcy,spcz に格納する。                    |
|              関数 long DLRound() , 外部関数 getdata() を使用。           |
+--------------------------------------------------------------------------+
*/
void InputSpcp(double xds,double yds,double zds)
{
    double x_scale,y_scale;

    x_scale = number_of_samples/length_of_samples;
    y_scale = x_scale;

    pnt[0].spcx = (double)0.0;      /* 原点 */
    pnt[0].spcy = (double)0.0;
    pnt[0].spcz = (double)0.0;

    pnt[1].spcx = pnt[0].spcx;      /* ｚ軸の上限 */
    pnt[1].spcy = pnt[0].spcy;
    pnt[1].spcz = height_scale*ceil((double)((data_max-data_min)/zds))*zds;

    pnt[2].spcx = pnt[0].spcx;      /* ｚ軸の下限 & ｘ軸の下限 */
    pnt[2].spcy = pnt[0].spcy;      /* この場合、原点に等しい   */
    pnt[2].spcz = pnt[0].spcz;

    pnt[3].spcx = (double)(number_of_samples-1);/* ｘ軸の上限 & ｙ軸の下限 */
    pnt[3].spcy = pnt[0].spcy;
    pnt[3].spcz = pnt[2].spcz;

    pnt[4].spcx = (double)(number_of_samples-1);    /* ｙ軸の上限 */
    pnt[4].spcy = (double)(number_of_samples-1);
    pnt[4].spcz = pnt[2].spcz;

    pnt[5].spcx = pnt[4].spcx;      /* ｙ軸の上限から始まる縦の線の上限 */
    pnt[5].spcy = pnt[4].spcy;
    pnt[5].spcz = getdata(1,DLRound(pnt[5].spcx),DLRound(pnt[5].spcy));
```

```
    pnt[6].spcx = pnt[3].spcx;      /* ｘ軸の上限から始まる縦の線の上限 */
    pnt[6].spcy = pnt[3].spcy;
    pnt[6].spcz = getdata(1,DLRound(pnt[6].spcx),DLRound(pnt[6].spcy));

    pnt[7].spcx = x_scale*xds;            /* length -> number */
    pnt[7].spcy = pnt[2].spcy;      /* ｘ軸の最初の目盛り */
    pnt[7].spcz = pnt[2].spcz;

    pnt[8].spcx = pnt[3].spcx;      /* ｙ軸の最初の目盛り */
    pnt[8].spcy = y_scale*yds;            /* length -> number */
    pnt[8].spcz = pnt[3].spcz;

    pnt[9].spcx = pnt[2].spcx;      /* ｚ軸の最初の目盛り */
    pnt[9].spcy = pnt[2].spcy;
    pnt[9].spcz = height_scale*zds;        /* length -> number */
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名    void ConvAxis(void)                                       |
|                                                                       |
|   引き数    なし                                                      |
|                                                                       |
|   返り値    なし                                                      |
|                                                                       |
|   コメント  構造体 pnt[].spcx,spcy,spcz に格納された３次元空間上の点の |
|             座標を２次元平面上の点の座標に変換し、構造体 pnt[].plnx,plny |
|             に格納する。     外部関数 fview(),getpxel() 使用。        |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void ConvAxis(void)
{
    int  i;

    for ( i=0 ; i<=AXISMAX-1 ; i++ ) {
        fview(pnt[i].spcx,pnt[i].spcy,pnt[i].spcz);
        getpxel();
        pnt[i].plnx = px;
        pnt[i].plny = py;
    }
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名    void RotAxis(eye,cent)                                    |
|                                                                       |
|   引き数    double eye[]  ： ３次元データ空間上での視線の位置         |
|                    cent[] ： データの中心座標                         |
|                                                                       |
|   返り値    なし                                                      |
|                                                                       |
|   コメント  見る方向が変わっても軸が常に手前に来るように、座標軸を構成する |
|             ために必要な点を一次変換する。                             |
|             関数 void RotPoint(),long DLRound(),外部関数 getdata() を使用。 |
```

```
+-----------------------------------------------------------------------+
*/
void RotAxis(double eye[],double cent[])
{
    int    i;
    double angle,cx,cy;

    if ((eye[X]>cent[X])&&(eye[Y]>cent[Y])) {
        angle = 90.0;
        cx=(double)(number_of_samples-1);  cy=(double)0.0;
    }
    else if ((eye[X]<cent[X])&&(eye[Y]>cent[Y])) {
        angle = 180.0;
        cx=(double)(number_of_samples-1); cy=(double)(number_of_samples-1);
    }
    else if ((eye[X]<cent[X])&&(eye[Y]<cent[Y])) {
        angle = -90.0;
        cx=(double)0.0;  cy=(double)(number_of_samples-1);
    }
    else if ((eye[X]>cent[X])&&(eye[Y]<cent[Y])) {
        angle = 0.0;
        cx=(double)0.0;  cy=(double)0.0;
    }
    for ( i=0 ; i<=AXISMAX-1 ; i++ )
        RotPoint(angle,cx,cy,&pnt[i].spcx,&pnt[i].spcy);
    pnt[5].spcz = getdata(1,DLRound(pnt[5].spcx),DLRound(pnt[5].spcy));
    pnt[6].spcz = getdata(1,DLRound(pnt[6].spcx),DLRound(pnt[6].spcy));
}
/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名    void RotPoint(angle,cx,cy,*x,*y)                          |
|                                                                       |
|   引き数    double angle : 回転角度 (degree)                          |
|                   cx,cy : 平行移動のｘ，ｙ成分                        |
|                   *x,*y : 一次変換したい点(x,y)のポインター            |
|                                                                       |
|   返り値    なし                                                      |
|                                                                       |
|   コメント  平面上の点(x,y)を、角度 angle だけ原点のまわりで回転し、   |
|             (cx,cy) だけ平行移動する。つまり、一次変換する。          |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void RotPoint(double angle,double cx,double cy,double *x,double *y)
{
    double  rad = 3.141592654/180.0,newx,newy;

    newx = cos(rad*angle)**x-sin(rad*angle)**y+cx;
    newy = sin(rad*angle)**x+cos(rad*angle)**y+cy;
    *x = newx;
    *y = newy;
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名    double Asclpnt(dvaxis,dmin,dmax)                          |
```

```
|                                                                          |
|   引き数    double dvaxis    : データ範囲を何分割した所に目盛りを入れるか  |
|                   dmin,dmax : データの最大値，最小値                      |
|                                                                          |
|   返り値    座標軸の最初の目盛りの値                                      |
|                                                                          |
|   コメント  座標軸の最初の目盛りの値を、データの最大値・最小値から自動的に |
|             決定する。但し、データの最大値・最小値が共に正だったり負だった |
|             りした場合、最小値←→最大値という範囲ではなく、０←→最大値、 |
|             または、最小値←→０という範囲をもとにして、その範囲を何分割す |
|             るかによって決定する。     関数 void Subasp() を使用。       |
|              (注) Ver.0.2では、dmin,dmax が共に 0.0 のときは未サポート。 |
+--------------------------------------------------------------------------+
*/
double Asclpnt(double dvaxis,double dmin,double dmax)
{
    double  a,asp1,asp2,abmax,abmin,axisdsp;

    abmax=fabs(dmax);
    abmin=fabs(dmin);
    if ((abmax!=0.0)||(abmin!=0.0)) {
        if (abmax>=abmin)
            a=abmax/dvaxis;
        else if (abmax<abmin)
            a=abmin/dvaxis;
        Subasp(a,&asp1,&asp2);
        if ((a/asp1)<(asp2/a))
            axisdsp=asp1;
        else  if ((a/asp1)>=(asp2/a))
            axisdsp=asp2;
    }
    return(axisdsp);
}
/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名    void Subasp(a,*asp1,*asp2)                                   |
|                                                                          |
|   引き数    double a     : データ範囲を適当に分割した値                  |
|                   *asp1 : 区切りのいい目盛りの小さい方の値のポインター  |
|                   *asp2 : 区切りのいい目盛りの大きい方の値のポインター  |
|                                                                          |
|   返り値    なし                                                         |
|                                                                          |
|   コメント  データ範囲を適当に分割した値から、区切りのいい目盛りを二つ   |
|             見つけだす。     関数 double Asclpnt() の下請け関数。        |
|              (注) 最寄りの区切りのいい目盛りの値を一つ返すようにした方  |
|                   がよい。改善の余地あり。                                |
+--------------------------------------------------------------------------+
*/
void Subasp(double a,double *asp1,double *asp2)
{
    double a1,a2,a5,a0;

    a1=pow((double)10.0,floor(log10(a)));
    a2=a1*2.0;
    a5=a1*5.0;
    a0=a1*10.0;
```

```
    if ((a>=a1)&&(a<a2)) {
        *asp1=a1;
        *asp2=a2;
    }
    else  if ((a>=a2)&&(a<a5)) {
        *asp1=a2;
        *asp2=a5;
    }
    else  if ((a>=a5)&&(a<a0)) {
        *asp1=a5;
        *asp2=a0;
    }
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名     void DrawAxis(void)                                      |
|                                                                       |
|   引き数     なし                                                     |
|                                                                       |
|   返り値     なし                                                     |
|                                                                       |
|   コメント   ３Ｄ画像に座標軸を書き込む関数。   外部関数 set_line() を使用。 |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void DrawAxis(void)
{
    int  i;

    for ( i=1 ; i<=4 ; i++ )
        set_line((int)pnt[i].plnx,(int)pnt[i].plny,(int)pnt[i+1].plnx,
                 (int)pnt[i+1].plny,(unsigned int)LNCOL,0);
    set_line((int)pnt[3].plnx,(int)pnt[3].plny,(int)pnt[6].plnx,
                 (int)pnt[6].plny,(unsigned int)LNCOL,0);
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名     void DrawSclp(void)                                      |
|                                                                       |
|   引き数     なし                                                     |
|                                                                       |
|   返り値     なし                                                     |
|                                                                       |
|   コメント   ３Ｄ画像に目盛りを書き込む関数。   外部関数 set_line() を使用。 |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void DrawSclp(void)
{
    int  i,o[3],e[3];
    double  *small,*big;
    double xp,yp,dx,dy,dxp1,dxp2,dyp1,dyp2;
```

```
    o[0]=2;  o[1]=3;  o[2]=0;
    e[0]=3;  e[1]=4;  e[2]=1;

    for ( i=7 ; i<=9 ; i++ ) {
        xp = pnt[o[i-7]].plnx;
        yp = pnt[o[i-7]].plny;
        dx = pnt[i].plnx-xp;
        dy = pnt[i].plny-yp;
        if (i==7) {
            small = &yp; big = &pnt[e[i-7]].plny;
            dxp1 = 0.0; dxp2 = 0.0; dyp1 = -3.0; dyp2 = 3.0;
        }
        else if ((i==8)||(i==9)) {
            small = &pnt[e[i-7]].plny; big = &yp;
            dxp1 = -3.0; dxp2 = 3.0; dyp1 = 0.0; dyp2 = 0.0;
        }
        while( *big >= *small ) {
            set_line((int)(xp+dxp1),(int)(yp+dyp1),(int)(xp+dxp2),
                    (int)(yp+dyp2),(unsigned int)LNCOL,0);
            xp += dx;
            yp += dy;
        }
    }
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名    void Axismain(void)                                       |
|                                                                       |
|   引き数    なし                                                      |
|                                                                       |
|   返り値    なし                                                      |
|                                                                       |
|   コメント  このモジュールのメイン関数的な役割を果たす関数。          |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void Axismain(void)
{
    onescl.x = Asclpnt(DVXAXS,(double)0.0,length_of_samples);
    onescl.y = Asclpnt(DVYAXS,(double)0.0,length_of_samples);
    onescl.z = Asclpnt(DVZAXS,(double)0.0,(data_max-data_min));

    InputSpcp(onescl.x,onescl.y,onescl.z);
    RotAxis(eye,cent);
    ConvAxis();
    DrawAxis();
    DrawSclp();
}
```

```
/*
+-----------------------------------------------------------------------------+
|                                  topvmain.c                                 |
|                                --------------                               |
|                                                                             |
+-----------------------------------------------------------------------------+
*/
#include <stdio.h>
#include <conio.h>          /* getch() */
#include <stdlib.h>         /* atoi(), atol(), exit() */
#include <math.h>           /* atof() */
#include <string.h>         /* strcpy(); */
#include <lib¥crt.h>        /* tinit(), tterm() */
#include <lib¥window.h>     /* winopen() */

#include "afm_def.h"
#include "popup.h"
#include "video.h"
#include "txcondef.h"

/*
+-----------------------------------------------------------------------------+
|                                 外部関数                                    |
+-----------------------------------------------------------------------------+
*/
extern void write_prm_to_file(FILE *,struct dat_file_format *);
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
extern int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                          short start,short stop);
extern FILE *open_file(FILE **f_ptr,char *);
extern  void read_prm_from_file(struct dat_file_format *);
extern  void errer_msg(char *);          /* errormsg.c */
extern  void dot_set(int,int);           /* scan_utl.c */
extern  void set_scan_screen(int);
extern  void ber_scale(void);
extern  void scan_no(int);
extern  void cls_256(void);              /* avgdrv.c   */
extern  void init_key(void);             /* afm_util.c */
extern  int  get_file_name(char *,char *,char *);
extern  int  setftime(char *,unsigned,unsigned);
/*
+-----------------------------------------------------------------------------+
|                          プロトタイプ宣言                                   |
+-----------------------------------------------------------------------------+
*/
static  int arg(int,char *[]);
        FILE *select_file(short);
        void read_prm(short);
        float read_datfile(int,int);
        void check_tilt(float *,float *);
        int set_topview_param(short);
        float readdata(short,short);
        void data_scan(void);
static  short start_job(int,char *[]);
        void end_job(short);

/*----------------------------Ｐｕｂｌｉｃ-------------------------------*/
```

```
char    prm_file_name[80];
int     axis_mode,auto_tilt;
float   m_height[400];

struct dat_file_format{
    int     what_file;
    int     number_of_samples;
    int     scan_size;
    float   tilt_x;
    float   tilt_y;
    float   max_dat;
    float   min_dat;
    int     start_x;
    int     start_y;
} dat_file;

FILE *fp;

WINDOW buf;     /* 望洋ライブラリーのポップアップウインドウ用変数 */

extern unsigned fdate,ftime;
char *file_name;
/*--------------------------------コード----------------------------------*/
/*-----------------------------------------------------------------------+

    関数名      static int arg(int argc,char *argv[])

    返値        1：コマンドラインから　0：メニューから　立ちあげ

    コメント      コマンドライン引数の処理。

+-----------------------------------------------------------------------*/
static int arg(int argc,char *argv[])
{
    char arguments[20][40],imput_from_key[80];
    int counter;

                                    /* デフォルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */
    strcpy(arguments[2],"0");       /* スキャン広さ表示 */

    for(counter = 1;counter < argc;counter++){
        strcpy(arguments[counter - 1],argv[counter]);
    }

    if(argc <= 3){                  /* コマンドライン引数が足りないとき */
        WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
        WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
        WINputstr(&buf,"デフォルト : リターンキー¥n");
        Cslon();
        while(1){
            switch(argc){
            case 1:
            case 2:
            case 3:
```

```
                WINlocate(&buf,1,2);
                WINputstr(&buf,
                "0 : 広さ表示  1 : 範囲表示¥n");
                WINdprintf(&buf,"%s  ",arguments[2]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[2],imput_from_key);
                }
            default:
                break;
            }

            WINputstr(&buf,"¥n以上でよろしいですね    ");
            gets(imput_from_key);
            if(imput_from_key[0] == 0){
                WINclose(&buf);
                break;
            }
        }
        Csloff();
    }
    strcpy(prm_file_name,arguments[1]);
    axis_mode = atoi(arguments[2]);
    auto_tilt = atoi(arguments[3]);
    return( atoi(arguments[0]) );
}

/*-----------------------------------------------------------------------+

    関数名      void open_file(void)

    コメント      パラメータファイルに指定されたファール検索パスから、
                ファイル名をたずね、ファイルをオープンする関数を呼ぶ。

+-----------------------------------------------------------------------*/
FILE *select_file(short flag)
{
    char parameter[30][40] /*, *file_name , name_buf[80]*/;
    int number_of_prm , result;

    number_of_prm = read_prm_file(prm_file_name,parameter,IMAGE_PART);
    if(number_of_prm == -1)     /* エラー */
        end_job(flag);
    result=get_file_name(parameter[0 + number_of_prm],"*.d*",file_name);
    if(result == 1)
        return ( open_file(&fp,file_name) );
    else
        return(NULL);
}

void read_prm(short flag)
{
    read_prm_from_file(&dat_file);

    if(dat_file.what_file != SCAN_DAT_FILE){
        char msg[200];

        strcpy(msg,"このファイルはスキャンデータの¥nファイルではありません");
```

```
        errer_msg(msg);
        puts("¥a");
        putchar('¥a');
        end_job(flag);
    }
}

/*-----------------------------------------------------------------------+

    関数名      float float read_datfile(int dx,int dy)

    引数        int dx: X座標
                int dy: Y座標

    返値        データ

    コメント      スキャンデータファイルから引数で指定された座標のデータを
                読み出す

+-----------------------------------------------------------------------*/
float read_datfile(int dx,int dy)
{
    float height;
    int result;

    result = fseek(fp,(long)(((long)dy * (long)dat_file.number_of_samples + dx)
                        * 4 + PRM_BYTE),0);
    result = fread((void *)&height,4,1,fp);
    return(height);
}

#define AVERAGE 50

void check_tilt(float *d_x,float *d_y)
{
    int i,step;
    float h1 = 0.0f,h2 = 0.0f;

    step = dat_file.number_of_samples / AVERAGE;

    for(i = 0;i < dat_file.number_of_samples;i += step){
        h1 += read_datfile(0,i);
        h2 += read_datfile(dat_file.number_of_samples - 1,i);
    }
    *d_x = (h2 -h1) / AVERAGE;

    h1 = h2 = 0.0f;

    for(i = 0;i < dat_file.number_of_samples;i += step){
        h1 += read_datfile(i,0);
        h2 += read_datfile(i,dat_file.number_of_samples - 1);
    }
    *d_y = (h2 -h1) / AVERAGE;
}

/*-----------------------------------------------------------------------+

    関数名      int set_topview_param(void)
```

返値　　マニュアルで、傾きを０以外に設定した場合：１　それ以外：０
　　オートの場合常に１

コメント　　ウインドウを開き、デフォルトのパラメータの変更をたずねる。

```
+-----------------------------------------------------------------------*/
int set_topview_param(short flag)
{
    char parameter[30][40];
    int number_of_prm,change = 1 /*,i*/ ;
    float dif_x = (float)0.0 , dif_y = (float)0.0;

    number_of_prm = read_prm_file(prm_file_name,parameter,IMAGE_PART);
    if(number_of_prm == -1)     /* エラー */
        end_job(flag);
    auto_tilt = atoi(parameter[11 + number_of_prm]);

    if(auto_tilt){
        check_tilt(&dif_x,&dif_y);
    }else{
        change = ask_prm(&buf,"傾き補正",parameter,number_of_prm,7,8);

        if(change){
            dif_x = (float)atof(parameter[7 + number_of_prm]);
            dif_y = (float)atof(parameter[8 + number_of_prm]);
        }
    }

    dat_file.tilt_x = dif_x / dat_file.number_of_samples;
    dat_file.tilt_y = dif_y / dat_file.number_of_samples;

    return(change);
}

float readdata(short dx,short dy)
{
    float fh;
    short result;

    result = fread((void *)&fh,4,1,fp);

    return(fh - dx * dat_file.tilt_x - dy * dat_file.tilt_y);
}

void data_scan(void)
{
    short data_x, data_y,result,first = 1;

    result = fseek(fp,(long)PRM_BYTE,SEEK_SET);

    for(data_y = 0;data_y < dat_file.number_of_samples;data_y++){
        for(data_x = 0;data_x < dat_file.number_of_samples;data_x++){
            m_height[data_x] = readdata(data_x,data_y);
            if(first){
                dat_file.max_dat = m_height[data_x];
                dat_file.min_dat = dat_file.max_dat - (float)0.001;
                first--;
```

```
            }else if(m_height[data_x] > dat_file.max_dat
                                && m_height[data_x] > dat_file.min_dat){
                dat_file.max_dat = m_height[data_x];
            }else if(m_height[data_x] < dat_file.min_dat
                                && m_height[data_x] < dat_file.max_dat){
                dat_file.min_dat = m_height[data_x];
            }
            /*dot_set(data_x,data_y);*/
        }
    }

    result = fseek(fp,(long)PRM_BYTE,SEEK_SET);

    for(data_y = 0;data_y < dat_file.number_of_samples;data_y++){
        for(data_x = 0;data_x < dat_file.number_of_samples;data_x++){
            m_height[data_x] = readdata(data_x,data_y);
            dot_set(data_x,data_y);
        }
    }
}

/*-----------------------------------------------------------------------+

    関数名      static int start_job(int argc,char *argv[])

    コメント      グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
static short start_job(int argc,char *argv[])
{
    short flag = 0x0000;

    tinit();                        /* 望洋ライブラリーの初期化 */
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */
    cls_256();                      /* グラフィクス画面のクリア */
    Cls();

    init_key();
    if( arg(argc,argv) ){
        flag = flag | 0x0001;       /* コマンドラインからの立ちあげ */
        Csloff();
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
        set_256palet(MONOCHROME);           /* ２５６色パレットの設定 */
    }
    if( select_file(flag) == NULL )
        end_job(flag);
    read_prm(flag);
    if( set_topview_param(flag) )
        flag = flag | 0x0002;       /* データを傾き補正する場合 */
    set_scan_screen(axis_mode);
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      int end_job(short flag)

    コメント      データファイルの先頭にファイル情報を書き込み、
```

リペイント後、ポーズ。そのご、データファイルをクローズし、
ウインドウを閉じる。

```
+-----------------------------------------------------------------------*/
void end_job(short flag)
{
    getch();                        /* ポーズ */

    if(flag & 0x0001){
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

void main(int argc,char *argv[])
{
    short result;

    result = start_job(argc,argv);              /* 初期化関数の呼び出し */

    data_scan();
    ber_scale();
    scan_no(dat_file.number_of_samples);
    if(result & 0x02){
        write_prm_to_file(fp,&dat_file);
        fclose(fp);
        setftime(file_name,fdate,ftime);
    }
    end_job(result);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

プログラムファイル

(B.7 Edit Program Files)

```
/*
+-----------------------------------------------------------------------+
|                              editmain.c                               |
|                            --------------                             |
|                                                                       |
|     ＡＦＭのパラメータファイルＰＲＭ＿ＦＩＬＥ．ＡＦＭを変更する      |
|    ＥＤＩＴ＿ＰＲＭ．ＥＸＥのメインモジュール。                        |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <string.h>        /* strcpy() */

#include <lib¥crt.h>
#include <lib¥key.h>
#include <lib¥bslib.h>
#include <lib¥keytbl.h>
#include <lib¥window.h>

#include "afm_def.h"
#include "popup.h"
#include "txcondef.h"

/*
+-----------------------------------------------------------------------+
|                            グローバル変数                             |
+-----------------------------------------------------------------------+
*/
char    prm_file_name[80];
short   cmd_line;

WINDOW  buf;
/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      int arg(int argc,char *argv[])

    返値        １，変更　０，参照

    コメント      コマンドライン引数の処理。引数が足りない場合にウインドウを
                開き、デフォルトの引数の値の変更をたずねる。

+-----------------------------------------------------------------------*/
int arg(int argc,char *argv[])
{
    char arguments[10][40],imput_from_key[80];
    int counter,mode;
                                    /* デフルトの値の代入 */
    strcpy(arguments[0],"1");       /* 1 : コマンドラインから立ち上げ */
    strcpy(arguments[1],"a:¥¥afm¥¥afm_menu¥¥prm_file.afm");
                                    /* パラメータファイル名 */
    strcpy(arguments[2],"0");       /* 0 : 参照のみ */

    for(counter = 1;counter < argc;counter++){
```

```
        strcpy(arguments[counter - 1],argv[counter]);
    }

    if(argc <= 3){
        WINopen(&buf,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
        WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"引数設定");
        WINputstr(&buf,"デフォルト : リターンキー¥n¥n");
        while(1){
            switch(argc){
            case 1:
            case 2:
            case 3:
                WINlocate(&buf,1,2);
                WINputstr(&buf,"0 : 参照のみ  1 : 変更¥n");
                WINputstr(&buf,arguments[2]);
                gets(imput_from_key);
                if(*imput_from_key != 0){
                    strcpy(arguments[2],imput_from_key);
                }
            default:
                break;
            }

            WINputstr(&buf,"¥n¥n以上でよろしいですね    ");
            gets(imput_from_key);
            if(imput_from_key[0] == 0){
                WINclose(&buf);
                break;
            }
        }
    }

    cmd_line = atoi(arguments[0]);
    strcpy(prm_file_name,arguments[1]);
    mode = atoi(arguments[2]);
    return(mode);
}

/*-----------------------------------------------------------------------+

    関数名      void edit(int edit_mode,int file_part)

    引数        edit_mode:1, 変更  0, 参照
                file_part:1, 開始  2, ステッパーモーター  3, スキャン
                          4, 画像処理

    コメント      ポップアップウインドウを開き、パラメータファイルの
                file_partのデフォルトの値を表示する。edit_modeが1のときは
                その値の変更をたずねる。

+-----------------------------------------------------------------------*/
void edit(int edit_mode,int file_part)
{
    char parameter[50][40],imput_from_key[40];
    int number_of_prm,i,change = 0;

    number_of_prm = read_prm_file(prm_file_name,parameter,file_part);
```

```
    WINopen(&buf,EDIT_X1,EDIT_Y1,EDIT_X2,EDIT_Y1 + number_of_prm+4,
            POPUP_SWITCH,POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,Green|Reverse,"変数設定");

    WINlocate(&buf,1,1);
    WINputstr(&buf,
        "デフォルトでよい場合はリターンキーを押してください");

    for(i = 0;i < number_of_prm;i++){
        WINlocate(&buf,2,i + 3);
        WINdprintf(&buf,"%s¥n",parameter[i]);
    }

    while(1){                               /* 書き換えのループ */
        for(i = 0;i < number_of_prm;i++){   /* デフォルトの値の表示 */
            WINlocate(&buf,36,i + 3);
            WINdprintf(&buf,"%22s¥n",parameter[i+number_of_prm]);
        }

        if(edit_mode){                      /* 変更の場合 */
            for(i = 0;i < number_of_prm;i++){
                WINlocate(&buf,58,i + 3);   /* デフォルトの値の隣に
                                                        カーソル表示*/
                WINputstr(&buf,"                        ");
                WINlocate(&buf,58,i + 3);
                gets(imput_from_key);
                WINlocate(&buf,58,i + 4);
                if(*imput_from_key != 0){   /* キー入力があった場合 */
                    strcpy(parameter[i + number_of_prm],
                            imput_from_key);    /* デフォルトの値の書き換え */
                    change = 1;                 /* 書き換えがあったことの印 */
                }
            }
        }
        WINlocate(&buf,1,1);
        WINputstr(&buf,
        "以上でよろしいですね                                ");
        WINlocate(&buf,22,1);
        gets(imput_from_key);
        if(imput_from_key[0] == 0 || imput_from_key[0] == 'y' ||
                        imput_from_key[0] == 'Y'){
            break;                              /* ループを抜ける */
        }
    }
    if(change){
        WINlocate(&buf,1,1);
        WINputstr(&buf,
        "ファイルを上書きします                              ");
        WINlocate(&buf,24,1);
        gets(imput_from_key);
        if(imput_from_key[0] == 0 || imput_from_key[0] == 'y'
                    || imput_from_key[0] == 'Y'){
            save_prm_file(prm_file_name,parameter,file_part,number_of_prm * 2);
        }                               /* 変更した値で、ファイルの上書き */
    }
    WINclose(&buf);
}
```

```
/*-----------------------------------------------------------------------+

    関数名        static int start_job(int argc,char *argv[])

    コメント        グラフィクス画面を設定し、初期化の関数を呼び出す。

+-----------------------------------------------------------------------*/
static int start_job(int argc,char *argv[])
{
    short flag;

    tinit();                        /* 望洋ライブラリーの初期化 */
    init_key();
    init_avgdrv(PC9821);            /* グラフィクス画面の設定 */

    flag = arg(argc,argv);
    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        Csloff();                   /* カーソルＯＦＦ */
        FUNC_KEY(OFF);              /* ファンクションキーの内容非表示 */
    }
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名        void end_job(void)

    コメント        終了の処理

+-----------------------------------------------------------------------*/
void end_job(void)
{
    Cslon();                        /* カーソルＯＮ */
    if(cmd_line){                   /* コマンドラインからたち上げた場合 */
        tterm();                    /* 望洋ライブラリーの使用終了 */
        end_videodrv();             /* 拡張グラフィクスドライバの使用終了 */
        FUNC_KEY(ON);               /* ファンクションキーの内容表示 */
        Cslon();                    /* カーソルＯＮ */
    }
    exit(0);
}

/*-----------------------------------------------------------------------+

    関数名        void main(int argc,char *argv[])

+-----------------------------------------------------------------------*/
void main(int argc,char *argv[])
{
    int edit_mode,parent_menu_no = 1,imput_key;

    edit_mode = start_job(argc,argv);           /* 初期化関数の呼び出し */

    WINopen(&buf,     EDIT_X1,EDIT_Y1,EDIT_X2,EDIT_Y2,
            POPUP_SWITCH,POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"変数設定");
    WINlocate(&buf,1,1);
```

```
    WINputstr(&buf,"メニュウを選んで下さい¥n");

    imput_key = parent_menu_sct(&parent_menu_no,4);
                        /* 親メニュウ選択 (1 - 4) */

    WINclose(&buf);

    if(imput_key == ESC){              /* 変数設定中止 */
        return;
    }

    edit(edit_mode,parent_menu_no);

    end_job();                                  /* 終了処理 */
}
/*
+-------------------------------------------------------------------------+
|                                 終わり                                  |
+-------------------------------------------------------------------------+
*/
```

プログラムファイル

(B.8 Common Program Files)

```
/*  bgraph.c(float)Ver.1.1用ヘッダーファイル   BGRA11.H  */

#define  GREEN   2   /*       AREASX                                  */
#define  CYAN    3   /* ┌──────┬──────────────── 640 ──────────┐ */
#define  YELLOW  6   /* │      ⋮                                  │ */
#define  WHITE   7   /* │      ⋮                                  │ */
#define  D2GRAY  15  /* │      ⋮AREAOX│←   AREADX   →│          │ */
                     /* AREASY···┌──────┼──────────────┐ ─        │ */
#define  AREASX  10  /* │      │      ┃              │ ↑        │ */
#define  AREASY  80  /* │      │      ┃              │AREADY    │ */
#define  AREADX  400 /* │      │      ┃              │          │ */
#define  AREADY  150 /* │      │      ┃              │ ↓        │ */
#define  AREAOX  0   /* │      ├──座標軸──────────┤ ─        │ */
#define  AREAOY  150 /* 400    │      ┃              │AREAOY    │ */
                     /* │      └──────┻ viewport ──┘ ─        │ */
/* プロトタイプ宣言*//* │                                        │ */
void  Init(void);    /* └────────────────────────────────────┘ */
void  Axis(void);
int   Draw(int,int,float [],float []);
void  Sclpoint(void);
void  Dscalepointx(int);
void  Dscalepointy(int);
void  Autoscale(int,int,float [],float []);
void  Autoscalepoint(int,int,float [],float []);
double Subasp(double);
double Max(int,int,float []);
double Min(int,int,float []);
void  Avmaxmin(int,int,float[]);
void  End(void);
void  Set_color(int);

/* 構造体・共用体  定義 */
struct  g_area{     /*    グラフ描画範囲(in viewport area)      */
      int  minx;            /* 左上隅 viewport x座標 */
      int  miny;            /* 左上隅 viewport y座標 */
      int  maxx;            /* 右下隅 viewport x座標 */
      int  maxy;            /* 右下隅 viewport y座標 */
      int  zerox;           /* 座標軸原点の viewport x座標 */
      int  zeroy;           /* 座標軸原点の viewport y座標 */
      };
struct  g_scale{    /*     グラフの縮尺（1単位何ドットか)      */
      double  x;
      double  y;
      int     autox;        /* 1 : オートスケーリング */
      int     autoy;        /* 1 : オートスケーリング */
      double  exrtx; /*オートスケーリングの時、1/(1+gparm.scale.exrtx)倍 */
      double  exrty; /*(for x),1/(1+gparm.scale.exrty)倍(for y)にして表示*/
      };
struct  g_scalepoint{   /* グラフの目盛り（1目盛り何単位か)    */
      double  x;            /* -1.0 : 目盛りをふらない */
      double  y;            /* -1.0 : 目盛りをふらない */
      int     autox;        /* 1 : オート */
      int     autoy;        /* 1 : オート */
      double  divx;         /* オートの時、何分割した所に目盛りを入れるか*/
      double  divy;         /* オートの時、何分割した所に目盛りを入れるか*/
      };
struct  g_unit{     /*    グラフの横軸・縦軸の単位              */
      char  *x;
```

```
        char  *y;
        };
struct  g_valinfo{        /*    グラフデータの最大値・最小値        */
        double  max;
        double  min;
        };
struct  graph{  /* 下の値はデフォルト値 */
        struct g_area       area;   /*No care (グラフ描画範囲 in viewport)*/
        struct g_scale      scale;  /* 何dot/1単位 , 1:auto , 割増率 */
        struct g_scalepoint dsp; /*何単位/1目盛(-1.0:No 目盛),1:auto,分割数*/
        struct g_unit       unit;     /* 単位(for x,y) */
        struct g_valinfo    value;    /* No care (グラフデータの max,min) */
        int     axcolor;              /* 軸線の色      */
        int     spcolor;              /* グリッドの色 */
        int     pocolor;              /* ドットの色    */
        } gparm={{ 0 , 0 , AREAOX+AREADX , AREAOY+AREADY , AREAOX , AREADY},¥
          {  0.8 , 30.0 , 1 , 1 ,  0.0 ,  0.0 },  /*1次元:30.0 2次元:200.0*/
          { 50.0 ,  1.0 , 1 , 1 , 10.0 ,  5.0 },   /*1次元:1.0  2次元:0.1  */
          { "[nm]" , "[volt]" },      ¥
          {  0.0 ,  0.0  },           ¥
          WHITE,D2GRAY,GREEN          ¥
        };
```

```
/*
+-----------------------------------------------------------------------+
|                              scnut121.c                               |
|                            --------------                             |
|                                                                       |
|     ＡＦＭのスキャンを行うＭ＿ＳＣＡＮ．ＥＸＥのサブ・モジュール。      |
|                                   ｖｅｒ．２．１  (debug by KOTARO)   |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <lib¥window.h>     /* winopen() */

#include "txcondef.h"
#include "video.h"
#include "popup.h"
#include "afm_def.h"

/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
void scan_no(int);
void ber_scale(void);
void set_scan_screen(int);
void dot_set(int,int);
void repaint(void);

extern void set_color_bar(int,int,int);

/*
+-----------------------------------------------------------------------+
|                              外部変数                                 |
+-----------------------------------------------------------------------+
*/
extern float    m_height[SAMPLING_MAX];

extern struct   dat_file_format{
    int     what_file;
    int     number_of_samples;
    int     scan_size;
    float   dif_x;
    float   dif_y;
    float   max_dat;
    float   min_dat;
    int     start[2];
} dat_file;

extern WINDOW   buf;

extern FILE     *fp;    /*  ファイルハンドル  */

/*
+-----------------------------------------------------------------------+
|                                コード                                 |
+-----------------------------------------------------------------------+
*/
```

```
/*-----------------------------------------------------------------------+

    関数名        void scan_no(int scan_line)

    コメント        スキャン中に開かれるウインドウに現在のスキャン数と
                全スキャン数を表示する。

+-----------------------------------------------------------------------*/
void scan_no(int scan_line)
{
    WINdputf(&buf,5,5,"scan %3d/%3d",scan_line,dat_file.number_of_samples);
}

/*-----------------------------------------------------------------------+

    関数名        void ber_scale(void)

    コメント        カラーバーに対するスケーリングを表示する。

+-----------------------------------------------------------------------*/
void ber_scale(void)
{
    Locate(61,10);
    printf("%4.1f[nm]    ",dat_file.max_dat - dat_file.min_dat);
}

/*-----------------------------------------------------------------------+

    関数名        set_scan_screen(void)

    コメント        スキャンの基本画面を表示する。

+-----------------------------------------------------------------------*/
void set_scan_screen(int mode)
{
    set_box(TOP_VIEW_WINDOW_X,TOP_VIEW_WINDOW_Y,
            TOP_VIEW_WINDOW_X + SAMPLING_MAX + 1,
            TOP_VIEW_WINDOW_Y + SAMPLING_MAX + 1,255);

    set_color_bar(440,TOP_VIEW_WINDOW_Y + 400 - 256,30);

    if(mode == 0){                          /* 広さ表示 */
        Locate(2,25);                       /* xスキャン範囲 */
        printf("0[nm]");
        Locate(50,25);
        printf("%d[nm]",dat_file.scan_size);
        Locate(1,24);                       /* yスキャン範囲 */
        printf("0[nm]");
        Locate(1,2);
        printf("%d[nm]",dat_file.scan_size);
    }else{                                  /* 範囲表示 */
        Locate(2,25);                       /* xスキャン範囲 */
        printf("%d[nm]",dat_file.start[X]);
        Locate(50,25);
        printf("%d[nm]",dat_file.start[X] + dat_file.scan_size);
        Locate(1,24);                       /* yスキャン範囲 */
        printf("%d[nm]",dat_file.start[Y]);
```

```
        Locate(1,2);
        printf("%d[nm]",dat_file.start[Y] + dat_file.scan_size);
    }

    Locate(63,23);                  /* カラーバー */
    printf("0[nm]");
    ber_scale();

    WINopen(&buf,POPUP2_X1,POPUP2_Y1,POPUP2_X2,POPUP2_Y2,POPUP_SWITCH,
                    POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(&buf,0,0,POPUP_TITLE_COLOR,"***測定中***");
    WINlocate(&buf,1,1);
    WINputstr(&buf,"     e:終了  r:再実行    ¥n");
    WINputstr(&buf,"       その他:ポーズ      ¥n");
    scan_no(0);
}

/*-----------------------------------------------------------------------+

    関数名      void dot_set(int i,int j)

    コメント      新しく測定されたデータを２５６階調に変換し、表示する。

+-----------------------------------------------------------------------*/
void dot_set(int i,int j)
{
    int color;

    color = (int)( (m_height[i] - dat_file.min_dat)
            / (dat_file.max_dat - dat_file.min_dat) * 255 );
    if(color > 255){
        color = 255;
    }
    else if(color < 0){
        color = 0;
    }

    if(dat_file.number_of_samples == 400){
/*      set_pxel(TOP_VIEW_WINDOW_X + 1 + i,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j,
             (unsigned int)color);*******************************/
        set_box_fill(TOP_VIEW_WINDOW_X + 1 + i,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j,
             TOP_VIEW_WINDOW_X + 1 + i,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j,
             (unsigned int)color);
    }
    else if(dat_file.number_of_samples == 200){
        set_box_fill(TOP_VIEW_WINDOW_X + 1 + i*2,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j*2 - 1,
             TOP_VIEW_WINDOW_X + 1 + i*2 + 1,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j*2,
             (unsigned int)color);
    }
    else{
        set_box_fill(TOP_VIEW_WINDOW_X + 1 + i*4,
             TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j*4 - 3,
             TOP_VIEW_WINDOW_X + 1 + i*4 + 3,
```

```
            TOP_VIEW_WINDOW_Y + SAMPLING_MAX - j*4,
            (unsigned int)color);
    }
}

/*-----------------------------------------------------------------------+

    関数名        void repaint(void)

    コメント        最終的に得られた最大値最小値のスケーリングで、
                再び画像を書き直す。

+-----------------------------------------------------------------------*/
void repaint(void)
{
    int i,j,result;

    result = fseek(fp,(long)PRM_BYTE,SEEK_SET);

    for(j = 0;j < dat_file.number_of_samples;j++){
        result = fread(m_height,sizeof(float),dat_file.number_of_samples,fp);
        for(i = 0;i < dat_file.number_of_samples;i++){
            dot_set(i,j);
        }
    }
}

/*
+-----------------------------------------------------------------------+
|                                 終わり                                |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              file_utl.c                               |
|                             --------------                            |
|                                                                       |
|          スキャンデータのファイルを操作する関数を集めたモジュール          |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <string.h>          /* strcpy(); */
#include <lib¥window.h>      /* winopen() */
#include <lib¥dir.h>         /* Dirselect() */

#include "afm_def.h"
#include "popup.h"

/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern  struct  dat_file_format{
    int     what_file;
    int     number_of_samples;
    int     scan_size;
    float   tilt_x;
    float   tilt_y;
    float   max_dat;
    float   min_dat;
    int     start_x;
    int     start_y;
} dat_file;

extern  FILE    *fp;


void write_prm_to_file(FILE *,struct dat_file_format *); /* by KOTARO */
void read_prm_from_file(struct dat_file_format *);


/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      void write_prm_to_file(struct dat_file_format *parameter)

    引数        struct dat_file_format *parameter:
                            スキャンデータファイルのパラメータを示す構造体

    コメント      スキャンデータのファイルの先頭に引数の構造体の値を書き込む

+-----------------------------------------------------------------------*/
void write_prm_to_file(FILE *wfp,struct dat_file_format *parameter)
{                   /* by KOTARO */
```

```
    int rc;
    rc = fseek(wfp,(long)0,SEEK_SET);
    rc = fwrite(parameter,sizeof(struct dat_file_format),1,wfp);
}

/*-----------------------------------------------------------------------+

    関数名      void read_prm_from_file(struct dat_file_format *parameter)

    引数        struct dat_file_format *parameter:
                              スキャンデータファイルのパラメータを示す構造体

    コメント      スキャンデータのファイルの先頭から引数の構造体に
                パラメータの値を読み出す。

+-----------------------------------------------------------------------*/
void read_prm_from_file(struct dat_file_format *parameter)
{
    int rc;
    rc = fseek(fp,(long)0,SEEK_SET);
    rc = fread(parameter,sizeof(struct dat_file_format),1,fp);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              effermsg.c                               |
|                             --------------                            |
|                                                                       |
|              エラーウインドウを開きメッセージを表示する。              |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <conio.h>

#include <lib¥crt.h>
#include <lib¥key.h>
#include <lib¥bslib.h>
#include <lib¥keytbl.h>
#include <lib¥window.h>

#include "popup.h"

/*
+-----------------------------------------------------------------------+
|                            グローバル変数                             |
+-----------------------------------------------------------------------+
*/
WINDOW errer_buf;
/*
+-----------------------------------------------------------------------+
|                                 コード                                |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      void errer_msg(char *msg[])

    引数        *msg[]: 表示するエラーメッセージ

    返値        無し

    コメント      エラーウインドウを開きメッセージを表示する。

+-----------------------------------------------------------------------*/
void errer_msg(char *msg)
{


    WINopen(&errer_buf,ERRER_X1,ERRER_Y1,EDIT_X2,ERRER_Y2,POPUP_SWITCH,
                                        POPUP_BACK_COLOR,ERRER_FRAME_COLOR);
    WINtitle(&errer_buf,0,0,ERRER_TITLE_COLOR,"エラー");
    WINputstr(&errer_buf,msg);
    getch();
    WINclose(&errer_buf);
}
/*
+-----------------------------------------------------------------------+
|                                 終わり                                |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------+
|                              edit_prm.c                               |
|                            --------------                             |
|                                                                       |
|     ＡＦＭのパラメータファイルＰＲＭ＿ＦＩＬＥ．ＡＦＭを参照、変更する     |
|   関数からなる。                                                       |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <string.h>        /* strcpy() */
#include <lib¥_form.h>
#include <lib¥window.h>     /* winopen() */

#include "afm_def.h"
#include "popup.h"
#include "txcondef.h"

extern void errer_msg(char *);


int read_prm_file(char *,char [][40],int);
int save_prm_file(char *,char [][40],int,int);
int ask_prm(WINDOW *,char *,char [][40],short,short,short);

/*
+-----------------------------------------------------------------------+
|                                 コード                                |
+-----------------------------------------------------------------------+
*/
/*---------------------------------------------------------------------+

    関数名      int read_prm_file(char *name,char parameter[][40],int dummy_no)

    引数        *name:              パラメータファイル名
                parameter[][40]:    パラメータを書き出す文字配列
                dummy_no:           空読みするパートの数

    返値        パラメータの数
                －１：エラー

    コメント      ＡＦＭのパラメータファイル*nameをdummy_noのパート空読みし、
                次のパートのパラメータを文字列に書き出す。

+---------------------------------------------------------------------*/
int read_prm_file(char *name,char parameter[][40],int dummy_no)
{
    FILE *prm_file_ptr;
    const char seps[] = " ";    /* 以下で使う strtok() のデリミタ */
    char buffer[100];
    char *token;
    int lines,result;

    if((prm_file_ptr = fopen(name,"r")) == NULL)    /* ファイルオープン */
    {
        char msg[200];
```

```
        strcpy(msg,"ＡＦＭのパラメータファイル¥n");
        strcat(msg,name);
        strcat(msg,"¥nが見つかりません。¥n");
        errer_msg(msg);
        putchar('¥a');
        return(-1);                          /* －１：エラー */
    }

    for(lines = 0;lines < dummy_no;lines++){    /* ファイルの空読み */
        do{
        result = fscanf(prm_file_ptr,"%s",buffer);
        }while(buffer[0] != '*');
    }

    for(lines = 0;lines < 100;lines++){
        result = fscanf(prm_file_ptr,"%s",buffer);      /* １行読み出し */
        token = strtok(buffer,seps);            /* 後半のスペース部分のカット*/
        if(*token == '*')                       /* ファイルパートの終わり */
            break;
        strcpy(parameter[lines],token);         /* パラメータの書き出し */
    }

    fclose(prm_file_ptr);
    return( lines / 2 );                    /* パラメータの数を返す */
}

/*-----------------------------------------------------------------------+

    関数名      int save_prm_file(char *name,char parameter[][40],int dummy_no,
                                                        int save_lines)

    引数        *name:              パラメータファイル名
                parameter[][40]:    新しいパラメータの配列
                dummy_no:           空読みするパートの数
                sava_lines:         書き込む行数

    返値        ０：成功

    コメント      ＡＦＭのパラメータファイル*nameをdummy_noのパート空読みし、
                次のパートにparameter[][]からsave_linesだけファイルに
                上書きする。

+-----------------------------------------------------------------------*/
int save_prm_file(char *name,char parameter[][40],int dummy_no,int save_lines)
{
    FILE *prm_file_ptr;
    int i,result;
    char buffer[100];

    if((prm_file_ptr = fopen(name,"r+")) == NULL)  /*ファイルのオープン*/
    {
        char msg[200];

        strcpy(msg,"ＡＦＭのパラメータファイル¥n");
        strcat(msg,name);
        strcat(msg,"¥nが見つかりません。¥n");
        errer_msg(msg);
        puts("¥a");
```

```
        return(-1);                                /* －１：エラー */
    }

    for(i = 0;i < dummy_no;i++){
        do{                                        /* ファイルの空読み */
            result = fscanf(prm_file_ptr,"%s",buffer);
        }while(buffer[0] != '*');
    }

    result = fseek(prm_file_ptr,0,SEEK_CUR);       /* 読み出しから書き込みへ*/

    result = fprintf(prm_file_ptr,"¥n");
    for(i = 0;i < save_lines;i++){                 /*パラメータの上書き */
        result = fprintf(prm_file_ptr,"%-70s¥n",parameter[i]);
    }

    fclose(prm_file_ptr);
    return(0);
}

int ask_prm(WINDOW *buff,char *title,char prm_str[][40],short prm_no,
                                        short start,short stop)
{
    char  imput_from_key[40];
    short i,flag = 0;

    WINopen(buff,POPUP1_X1,POPUP1_Y1,POPUP1_X2,POPUP1_Y2,POPUP_SWITCH,
                    POPUP_BACK_COLOR,POPUP_FRAME_COLOR);
    WINtitle(buff,0,0,POPUP_TITLE_COLOR,title);

    Cslon();
    while(1){
        WINlocate(buff,1,1);
        WINputstr(buff,
        "デフォルトでよい場合はリターンキーを押してください¥n");

        for(i = start;i <= stop;i++){
            WINdprintf(buff,"¥n%s  ",prm_str[i]);
            WINdprintf(buff,"%s  ",prm_str[i + prm_no]);
            gets(imput_from_key);
            if(*imput_from_key != 0){
                strcpy(prm_str[i + prm_no],imput_from_key);
                flag = CHANGE;
            }
        }

        WINputstr(buff,"¥n以上でよろしいですね    ");
        gets(imput_from_key);
        if(imput_from_key[0] == 0 || imput_from_key[0] == 'y' ||
                                        imput_from_key[0] == 'Y'){
            break;
        }
    }

    WINclose(buff);
    Csloff();
    return(flag);
}
```

```
/*
+-----------------------------------------------------------------------+
|                                 終わり                                |
+-----------------------------------------------------------------------+
*/
```

```
/*************************************************************************
*   ■ 望洋ライブラリ ■                    "BSdirwin.C"    【C】 PC-9800  *
*-----------------------------------------------------------------------*
*   ディレクトリ検索モジュール                                by 柴田 望洋    *
*--------------------------- Public Interface ---------------------------*
* int DIRselect(DIR *sel, char *path1, char *path2, int max, int func,     *
*               int x1, int col, int y1, int y2);                         *
*************************************************************************/

/****************************************

    ワイルドカードが使えるように

            ２３７，５２４，５８３行  書き換え  ｂｙ  中野

                        ************************************************/


#include    <ctype.h>
#include    <stdio.h>
#include    <stdlib.h>
#include    <string.h>
#include    <lib¥crt.h>
#include    <lib¥dir.h>
#include    <lib¥dos.h>
#include    <lib¥key.h>
#include    <lib¥bslib.h>
#include    <lib¥memory.h>
#include    <lib¥keytbl.h>
#include    <lib¥window.h>
#include    <lib¥_form.h>
#if !defined(LATTICE)
#include    <jctype.h>
#endif

#if defined(__ZTC__)
    #define stricmp(s1,s2)  strcmpi((s1),(s2))
#endif

#define MAX         192         /* 最大ディレクトリ数 */
#define WID         13          /* １ファイル名表示幅 */
#define erase1(y)   disps(X1+1,y,White|Reverse,X2-X1-1)
#define Color1(n)   (Select[n] ? Cyan|Reverse : White|Reverse)
#define Color2(n)   (Select[n] ? Blue|Reverse : Cyan|Reverse)

#define disp_path()     (disp(X1+WID,Y0, Magenta|Reverse , pathname), ¥
                            color(Green|Reverse), _putchrn(' ',X2-X1-WID-strlen
(pathname)))

static char     path[64],
                pathname[64];
static int      window = 0,
                column = 6,
                X1 =  1,
                X2 = 80,
                Y0 = 10,
                Y1 = 10,
                Y2 = 19,
```

```
                tab = 4,
                DispMax = 48,   /* 選択ウィンドウに表示する最大数 */
                ArcNo = -1,     /* ファイルの数 */
                DirNo = -1;     /* ディレクトリの数 */

static DIR      *DirBuf;        /* ディレクトリ格納バッファ */
static char     *Select;
static int      Count;          /* 選択された個数 */

static WINDOW   wbuf;

/*----- 選択フラグ配列をクリア -----*/
static void Clear(char *Select)
{
    Count = 0;
    memset(Select, 0, MAX * sizeof(char));
}

/*----- 選択・非選択トグルスイッチ -----*/
static void Selection(int n)
{
    if (Select[n]) {
        Count--;    Select[n] = 0;
    } else {
        Count++;    Select[n] = 1;
    }
}

/*----- pathnameからパス名pathを作成 -----*/
static void MakePath(void)
{
    int i    = strlen(pathname);
    int wild = 0;

    path[0] = '¥0';
    for ( ; i >= 0; i--) {
        if ((pathname[i]=='*')||(pathname[i]=='?'))
            wild = 1;
        else if ((wild) && ((pathname[i]=='¥¥')||pathname[i]==':'))
            break;
    }
    if (i < 0)
        path[0] = '¥0';
    else {
        strncpy(path, pathname, i+1);
        path[i+1] = '¥0';
    }
}

/*----- タブ展開 -----*/
static void detab(char *s)
{
    char    t[256] , *s1 = s;
    int     pos = 0;

    t[0] = '¥0';
    while (*s) {
        if (*s == '¥t') {
```

```
            int nb = tab - pos % tab;
            while (nb) {
                t[pos++] = ' ';
                nb--;
            }
        } else
            t[pos++] = *s;
        s++;
    }
    t[pos] = '¥0';
    strcpy(s1 , t);
}

/*----- タイプ画面 -----*/
static int Type(int no)
{
    WINDOW  buf;
    FILE    *fp;
    char    id[256] , s[256];
    int     i , ch , c;
    long    pos[17] , temp;

    WINopen(&buf, 5, 7, 76, 24, 0, White, Blue|Reverse);
    strcpy(id , path);
    strcat(id , DirBuf[no].name);
    WINtitle(&buf , 0 , 0 , Green|Reverse , id);
    fp = fopen(id,"r");
    color(White);
    for (i=0; i<16; i++) {
        pos[i] = ftell(fp);
        if (fgets(s , 256 , fp) == NULL) break;
        s[70] = '¥0';
        detab(s);
        s[70] = '¥0';
        dput(6 , i+8 , s);
    }
    pos[16] = ftell(fp);
    while (1) {
        if ( (ch = inkey()) == CR) break;
        switch (ch) {
         case HT  : tab = (tab == 4) ? 8 : 4;
         case DOWN: fseek(fp , pos[16] , SEEK_SET);
                    if (fgets(s , 256 , fp) == NULL) continue;
                    roll_text(6,8,75,23,-1);
                    for (i=0; i<16; i++) pos[i] = pos[i+1];
                    pos[16] = ftell(fp);
                    s[70] = '¥0';
                    detab(s);
                    s[70] = '¥0';
                    dput(6 , 23 , s); break;
         case UP  : if (pos[0] == 0L) break;
                    if (fseek(fp , (pos[0]<3L) ? 0L : pos[0]-3L , SEEK_SET))
                        continue;
                    if (pos[0] > 2L) {
                        while ((c = getc(fp)) != EOF && c != '¥n') {
                            if (fseek(fp , -2L , SEEK_CUR)) break;
                        }
                    }
```

```
                    if ( (temp = ftell(fp)) == -1L) {
                        fseek(fp , temp=0L , SEEK_SET);
                    }
                    if (fgets(s , 256 , fp) == NULL) continue;
                    roll_text(6,8,75,23,1);
                    for (i=16; i>0; --i) pos[i] = pos[i-1];
                    pos[0] = temp;
                    s[70] = '¥0';
                    detab(s);
                    s[70] = '¥0';
                    dput(6 , 8 , s); break;
        default :   break;
        }
    }
    fclose(fp);
    WINclose(&buf);
    return(0);
}

/*----- ダンプ画面 -----*/
static int Dump(int no)
{
    WINDOW  buf;
    FILE    *fp;
    unsigned char   id[128] , s[257];
    int     i , j , ch;
    int     k = 0 , flag=0;

    WINopen(&buf,5,7,76,24,0, White, Red|Reverse);
    strcpy(id , path);
    strcat(id , DirBuf[no].name);
    WINtitle(&buf , 0 , 0 , Green|Reverse , id);
    fp = fopen(id, "rb");
    fread(s , 1 , 257 , fp);
    color(White);
    for (i=0; i<16; i++) {
        _locate(8 , i+8);
        for (j=0; j<16; j++)
            _dprintf("%02X ",(unsigned int)s[(i<<4)+j]);
        _putstr("  ");
        if (flag) s[i<<4] = ' ';
        for (j=0; j<16; j++) {
            k = (i << 4) + j;
            ch = s[k];
            if (flag) {
                _putchr(ch);
                flag = 0;
            } else if (isprint(ch)||iskanji(ch)) {
                _putchr(ch);
                if (iskanji(ch)) flag = 1;
            } else
                _putchr('.');
        }
        if (flag)
            _putchr(s[i<<5]);
    }
    fclose(fp);
    ch = inkey();
```

```
    WINclose(&buf);
    return(0);
}

/*----- ディレクトリ読み込み -----*/
static int GetDir(char *name)           /* (void) -> (char *name) by Nakano */
{
    char    path2[64];
    DIR     buf;

    MakePath();
    strcpy(path2, path);
    strcat(path2, name);
    DirNo = -1;
    if (!DIRfirst(path2, &buf, FAarch))
        do {
            DirBuf[++DirNo] = buf;
            if (DirNo >= MAX-1) break;
        } while (!DIRnext(&buf));
    ArcNo = DirNo;
    if (DirNo >= MAX-1) return(DirNo);
    if (!DIRfirst(path2, &buf, FAdirec))
        do {
            if (buf.attrib != FAdirec || !strcmp(buf.name, "."))
                continue;
            DirBuf[++DirNo] = buf;
            strcat(DirBuf[DirNo].name, "¥¥");
            if (DirNo >= MAX-1) return(DirNo);
        } while (!DIRnext(&buf));
    return(DirNo);
}

/*----- ガイド[↑][↓]の表示 -----*/
static void dispn(register int start)
{
    disp(X2-5, Y2,  Green|Reverse, "    ");
    if (start > 0)              disp(X2-5, Y2, Magenta|Reverse,"↑");
    if (start+DispMax < DirNo)  disp(X2-3, Y2, Blue|Reverse,   "↓");
}

static void disp_dir(register int ptr)
{
    register int    i , j;

    color(White|Reverse);
    bcls(X1+1 , Y1+1 , X2-1 ,Y2-1);
    j = (ptr+DispMax < DirNo) ? DispMax : DirNo-ptr;
    for (i = 0; i <= j; i++)
        dputf(i%column*WID+X1+2, i/column+Y1+1, "%-12.12s", DirBuf[ptr+i].name);
    if (DirNo > DispMax) dispn(ptr);
}

static void disp_dir1(int y, int ptr)
{
    register int    i , j;

    erasel(y);
    j = (ptr+column-1 < DirNo) ? column-1 : DirNo-ptr;
```

```
    for (i = 0; i <= j; i++)
        dispf(i*WID+X1+2,y, Color1(ptr+i), "%-12.12s", DirBuf[ptr+i].name);
}

/*----- 一行ロールアップ -----*/
static void RollUp(int *start)
{
    if (*start+DispMax >= DirNo) return;
    (*start) += column;
    roll_text(X1+1, Y1+1, X2-1, Y2-1, -1);
    disp_dir1(Y2-1, *start+DispMax-column+1);
    dispn(*start);
}

/*----- 一行ロールダウン -----*/
static void RollDown(int *start)
{
    if (*start == 0) return;
    (*start) -= column;
    roll_text(X1+1, Y1+1, X2-1, Y2-1, 1);
    disp_dir1(Y1+1, *start);
    dispn(*start);
}

/*----- 次ページへ -----*/
static void NextPage(int *start)
{
    if (*start+DispMax > DirNo) return;
    *start += DispMax+1;
    disp_dir(*start);
    dispn(*start);
}

/*----- 前ページへ -----*/
static void PrevPage(int *start)
{
    if (*start == 0) return;
    *start -= DispMax+1;
    if (*start < 0) *start = 0;
    disp_dir(*start);
    dispn(*start);
}

/*----- [↑]処理 -----*/
static int up(int *start,int pos)
{
    if (pos >= column) return(pos-column);
    if (*start == 0) return(pos);
    RollDown(start);
    return(pos);
}

/*----- [↓]処理 -----*/
static int down(int *start,int pos)
{
    int end;

    end = (*start+DispMax < DirNo) ? DispMax: DirNo-*start;
```

```
    if (pos <= end-column) return(pos+column);
    if (*start+DispMax > DirNo) return(pos);
    RollUp(start);
    return(pos);
}

/*----- 情報表示（タイムスタンプなど） -----*/
static void DispInfo(int pos)
{
    union {
        struct {
        #if defined(LATTICE)
            unsigned    y : 7;
            unsigned    m : 4;
            unsigned    d : 5;
        #else
            unsigned    d : 5;
            unsigned    m : 4;
            unsigned    y : 7;
        #endif
        } d;
        unsigned x;
    } md;
    union mt {
        struct {
        #if defined(LATTICE)
            unsigned    h : 5;
            unsigned    m : 6;
            unsigned    s : 5;
        #else
            unsigned    s : 5;
            unsigned    m : 6;
            unsigned    h : 5;
        #endif
        } t;
        unsigned x;
    } mt;

    md.x = DirBuf[pos].date;        /* タイムスタンプ（日付） */
    mt.x = DirBuf[pos].time;        /* タイムスタンプ（時刻） */
    color(Yellow|Reverse);
    dputf(X2-28, Y1, "%04d/%02d/%02d", md.d.y+1980, md.d.m, md.d.d);
    dputf(X2-17, Y1, "%02d:%02d:%02d", mt.t.h, mt.t.m, mt.t.s);
    dputf(X2- 8, Y1, "%8ld", DirBuf[pos].size);
}

static int      start = 0;

static int SelectDir(int max)
{
    int             ch;
    static int      end,
                    pos = 0 , x , y;

    cursor(CURSORoff);
    do {
        end = (start+DispMax < DirNo) ? DispMax : DirNo-start;
        if (pos > end)
```

```
            do pos -= column; while (pos>end);
        if (pos < 0) pos = 0;
        x = pos%column*WID+X1+2;
        y = pos/column+Y1+1;
        color_(x,y,x+11,y,Color2(start+pos));
        DispInfo(start+pos);
        ch = inkey();
        if (ch == ' ') {
            if (start+pos > ArcNo) return(start+pos);
            Selection(start+pos);
        } else if (ch == 0x09) {
            Selection(start+pos);
            color_(x,y, x+11,y, Blue|Reverse);
            return(-1);
        }
        color_(x,y,x+11,y,Color1(start+pos));
        switch (ch) {
         case 'O' :
         case 'T' :
         case 't' : ch = Type(start+pos); break;
         case '1' :
         case 'D' :
         case 'd' : ch = Dump(start+pos); break;
         case UP  : pos = up(&start,pos);   break;
         case DOWN  : pos = down(&start,pos);break;
         case RIGHT : if (pos < end) ++pos;  break;
         case LEFT  : if (pos > 0)   --pos;  break;
         case 0x03 : NextPage(&start);      break;
         case 0x12 : PrevPage(&start);      break;
         case 0x1A : RollUp(&start);        break;
         case 0x17 : RollDown(&start);      break;
        }
        if (Count == max) break;
    } while ((ch != CR) && (ch != ESC));
    cursor(CURSORon);
    return((ch == ESC) ? -2 : -1 );
}

/*----- パス名入力 -----*/
static int InputPath(void)
{
    int     flag ,pos;

    pos = 1;
    flag = _readstr(pathname, 60,X1+WID,Y0,&pos, Magenta|Reverse,
                    Green|Reverse,1);
    if (flag != ESC) MakePath();
    return(flag);
}

/*----- ディレクトリ選択ウィンドウをオープン -----*/
static int OpenWindow(char *path1, int x1, int col, int y1, int y2)
{
    if (window) return(0);
    column = col;
    X1 = x1;
    X2 = x1 + column * WID + 1;
    if (X2 > 80) {
```

```
        column = (79-X1) / WID;
        X2 = x1 + column * WID + 1;
    }
    Y0 = (column < 4) ? y1-1 : y1;
    Y1 = y1;
    Y2 = y2;
    DispMax = column * (y2-y1-1) - 1;
    strcpy(pathname, path1);
    if (WINopen(&wbuf, X1,Y0, X2,Y2, WinNoFrame, White|Reverse, Green|Reverse))
        return(1);
    if ((DirBuf = (DIR  *)calloc(MAX, sizeof(DIR ))) == NULL) return(2);
    if ((Select = (char *)calloc(MAX, sizeof(char))) == NULL) return(3);
    if (column < 4)
        disps(X1+1,Y1, Green|Reverse, X2-X1-1);
    disp(X1+1,Y0, Green|Reverse , "<PATH NAME>");
    disp_path();
    window = 1;
    Count = 0;
    return(0);
}

/*----- ディレクトリ選択ウィンドウをクローズ -----*/
static void CloseWindow(int flag)
{
    if (flag < 3) free(Select);
    if (flag < 2) free(DirBuf);
    if (flag < 1) WINclose(&wbuf);
    window = 0;
}

#define Ext if (KeyChange) { KeyChange = 0; ___popskey(); }

/*-----------------------------------------------------------------------
    int DIRselect(DIR *sel, char *path1, char *path2, int max, int func,
                  int x1, int col, int y1, int y2)
-----------------------------------------------------------------------*/
int dir_select(DIR *sel, char *path1, char *name, char *path2, int max,
                int func, int x1, int col, int y1, int y2)

{
    char            old_path[64];
    register int    i, j, flag;
    int             KeyChange = 0;

    if (func & DIR_CLOSE) {
        CloseWindow(0);
        Ext;
        return(0);
    }
    /*if (!___DiamondKey) {
        KeyChange = 1;
        ___pushskey();
    }*/
    if (!window) {
        if ((flag = OpenWindow(path1, x1, col, y1, y2)) != 0) {
            CloseWindow(flag);
            Ext;
            return(0);
```

```
    }
}
GetDir(name);                           /* () -> ("*.*") by Nakano */
disp_dir(0);
while (1) {
    Clear(Select);
    if (DirNo != -1) {
        while (1) {
            if ((flag = SelectDir(max)) > 0) break;
            if (func & DIR_CONF) {
                if (confirm1(10,10,"") == 1) break;
            } else
                break;
        }
        if (flag >= 0) {
            char    *p,*p2;
            if ((p = strrchr(pathname , '¥¥')) == NULL)
                p = strchr(pathname, ':');
            if (DirBuf[flag].name[0] == '.') {
                if (p == NULL)
                    strcpy(pathname, "¥¥*.*");
                else if (*p == ':')
                    strcpy(p+1, "¥¥*.*");
                else {
                    for (p2 = p; (--p) - pathname >= 0; ) {
                        if (*p == '¥¥') break;
                        if (*p == ':') {p++; break; }
                    }
                    while (*p2) *p++ = *p2++;
                    *p = '¥0';
                }
            } else {
                if (p == NULL) p = pathname; else p++;
                strinsstr(p, DirBuf[flag].name);
            }
            disp_path();
            Clear(Select);
            start = 0;
            goto A;
        }
        if (flag != -2) {
            for (i = j = 0; i < MAX; i++) {
                if (Select[i]) {
                    sel[j] = DirBuf[i];
                    j++;
                }
            }
            if ((func & DIR_REMAIN) == 0) CloseWindow(0);
            strcpy(path2, path);
            Ext;
            return(Count);
        }
    }
    strcpy(old_path , (DirNo == -1)? "" : pathname);
    do {
        if (InputPath() == ESC) {
            if ((func & DIR_REMAIN) == 0) CloseWindow(0);
            Ext;
```

```
                return(-1);
            }
            A:  ;
        } while (GetDir("*.*") == -1);          /* () -> ("*.*") by Nakano */
        if (stricmp(old_path , pathname)) {
            disp_dir(0);
        }
    }
}

#if 0

int main(void)
{
    char    path[2][64],name[15];
    DIR     buf[2];
    int     flag;

    tinit();
    _tcls();
    strcpy(path[0], "a:¥¥afm¥¥data¥¥*.*");
    strcpy(name, "*.*");
    flag = dir_select(buf, path[0], name, path[1], 2, 0, 6, 4, 5, 10);
        if (flag>=1) printf("%s%s¥n", path[1], buf[0].name);
    if (flag>=2) printf("%s%s¥n", path[1], buf[1].name);
    tterm();
    return(0);
}
#endif
```

```
/*
+-----------------------------------------------------------------------+
|                              dirsct10.c                               |
|                            --------------                             |
|                                                                       |
|        望洋ライブラリのファイル（ディレクトリ）選択ウィンドウを        |
|        利用して、ファイル選択を行うためのモジュール　(Ver.1.0)        |
|                                                                       |
+-----------------------------------------------------------------------+
*/
#include <stdio.h>
#include <process.h>    /* _spawnl() */
#include <string.h>     /* strcat(),strcpy() */
#include <dos.h>        /* _dos_getftime() , _dos_setftime() */
#include <lib¥dir.h>
#include <lib¥_form.h>

#include "afm_def.h"
#include "txcondef.h"
#include "popup.h"

unsigned fdate,ftime;
/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
void init_key(void);
int  getftime(char *,unsigned *,unsigned *);
int  setftime(char *,unsigned,unsigned);
int  get_file_name(char *,char *,char *);
FILE *open_file(FILE **,char *);
FILE *open_temp_file(FILE *,char *,int);
void del_temp(char *,int);
/*
+-----------------------------------------------------------------------+
|                              外部関数                                 |
+-----------------------------------------------------------------------+
*/
extern int read_prm_file(char *name,char parameter[][40],int dummy_no);
                                                        /* edit_prm.c */
extern int dir_select(DIR *,char *,char *,char *,int,int,int,int,int,int);
                                                        /* dirwin.c */
extern void errer_msg(char *);                          /* errormsg.c */
/*
+-----------------------------------------------------------------------+
|                                コード                                 |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      void init_key(void)

    コメント      キー配列の初期化。望洋ライブラリーを使う場合の
                カーソルキーの設定

+-----------------------------------------------------------------------*/
void init_key(void)
```

```
{
    int KeyChange = 0;

    if (!___DiamondKey) {
        KeyChange = 1;
        ___pushskey();
    }
}

int getftime(char *fname,unsigned *date,unsigned *time)
{
    FILE *fp;
    int  hd;

    if((fp = fopen(fname,"r"))==NULL)
        return(1);
    hd = fileno(fp);
    _dos_getftime(hd,date,time);
    fclose(fp);
    return(0);
}

int setftime(char *fname,unsigned date,unsigned time)
{
    FILE *fp;
    int  hd;

    if((fp = fopen(fname,"a"))==NULL)
        return(1);
    hd = fileno(fp);
    _dos_setftime(hd,date,time);
    fclose(fp);
    return(0);
}

/*-----------------------------------------------------------------------+

    関数名      int get_file_name(char *file_path,char *name,char *slct_name)

    引数        char *file_path:検索するファイルのパス
                char *name:     検索するファイルの名前（ワイルドカード）
                char *slct_name:選択されたファイルのフルパス名

    返値        選択したファイルの数　何も選択しなかった場合 0

    コメント      引数で指定されたパスから、ファイルを選択する。

+-----------------------------------------------------------------------*/
int get_file_name(char *file_path,char *name,char *slct_name)
{
    DIR     buf[1];
    int     flag;

    strcat(file_path, "*.*");
    flag = dir_select(buf, file_path, name, slct_name, 1, 0,
                              POPUP1_X1, 4, POPUP1_Y1, POPUP1_Y2);
    strcat(slct_name, buf[0].name);
```

```
    return(flag);
}

/*-----------------------------------------------------------------------+

    関数名      FILE *open_file(FILE **f_ptr,char *search_path,char *wild_name)

    引数        FILE **f_ptr:       オープンするファイルを示すポインタ
                char *search_path:  検索するファイルのパス
                char *wild_name:    検索するファイルの名前（ワイルドカード）

    返値        オープンしたファイルのポインタ　何も選択しなかった場合 NULL

    コメント      引数で指定されたパスから、引数で指定された名前のファイル
                （ワイルドカードを含む）をオープンする。

+-----------------------------------------------------------------------*/
FILE *open_file(FILE **f_ptr,char *fname)
{
    if(getftime(fname,&fdate,&ftime) == 1){
                                        /* ファイルをオープンできない */
        char msg[200];

        strcpy(msg,fname);
        strcat(msg,"¥nがオープンできません¥n");
        errer_msg(msg);
        putchar('¥a');
        return(NULL);
    }
    *f_ptr = fopen(fname,"r+b");
    return(*f_ptr);
}

/*-----------------------------------------------------------------------+

    関数名      FILE *open_temp_file(FILE *fp,char *prmfile_name,int part)

    引数        FILE *fp:           ファイルポインター
                char *prmfile_name: ＡＦＭのパラメータファイル名
                int part:           パラメータファイル内のパート

    返値        選択したファイルを指すファイルポインター
                NULL: 何も選択しなかった、エラー

    コメント      スキャンのデータファイルをパラメータファイルで指定された、
                パスから選択すし、指定された名前でラムディスクにコピーし、
                そのファイルをオープンし、ファイルポインタを返す。

+-----------------------------------------------------------------------*/
FILE *open_temp_file(FILE *fp,char *prmfile_name,int part)
{
    char parameter[30][40],*file_name;
    int number_of_prm,result;

    number_of_prm = read_prm_file(prmfile_name,parameter,part);

    get_file_name(parameter[0 + number_of_prm],"*.d*",file_name);
                    /* スキャンデータファイルのデフォルトの拡張子 *.dat */
```

```
    result = _spawnl(_P_WAIT,"a:¥¥afm¥¥afm_menu¥¥image¥¥movefile.bat",
                     "a:¥¥afm¥¥afm_menu¥¥image¥¥movefile.bat",
                     file_name,parameter[9 + number_of_prm],
                     parameter[10 + number_of_prm],NULL);
            /* ラムディスクにファイルをコピーするバッチファイルを立ち上げる */

    strcat(parameter[9 + number_of_prm],parameter[10 + number_of_prm]);

    if((fp = fopen(parameter[9 + number_of_prm],"rb")) == NULL){
                                /* テンポラリーファイルをオープンできない */
        char msg[200];

        strcpy(msg,parameter[9 + number_of_prm]);
        strcat(msg,"¥nがオープンできません¥n");
        errer_msg(msg);
        putchar('¥a');
        return(NULL);
    }
    return(fp);
}

/*-----------------------------------------------------------------------+

    関数名      void del_temp(char *prmfile_name,int part)

    引数        char *prmfile_name: ＡＦＭのパラメータファイル名
                int part:           パラメータファイル内のパート

    コメント      パラメータファイルで指定された名前のラムディスクを
                消去するバッチファイルを起動する。

+-----------------------------------------------------------------------*/
void del_temp(char *prmfile_name,int part)
{
    char parameter[30][40];
    int number_of_prm,result;

    number_of_prm = read_prm_file(prmfile_name,parameter,part);

    strcat(parameter[9 + number_of_prm],parameter[10 + number_of_prm]);

    result = _spawnl(_P_WAIT,"a:¥¥afm¥¥afm_menu¥¥image¥¥del_temp.bat",
                     "a:¥¥afm¥¥afm_menu¥¥image¥¥del_temp.bat",
                     parameter[9 + number_of_prm],NULL);
            /* テンポラリファイルを削除するバッチファイルを立ち上げる */
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```

```
/*
+-----------------------------------------------------------------------------+
|                          BGRAPH11.C     (include file : BGRA.H)              |
|                         --------------                                      |
|                                                                             |
|      ２次元グラフ描画モジュール（float型対応）    Ｖｅｒ．１．１            |
|                                                      ｂｙ　清水 光太朗      |
|                                                                             |
+-----------------------------------------------------------------------------+
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <graph.h>
#include "bgrall.h"     /* デファイン・プロトタイプ宣言・構造体定義 */
/*#include "txcondef.h"*/   /* Csloff() 実は、使用していない */

/*
+-----------------------------------------------------------------------------+
|                            ソースコード                                     |
+-----------------------------------------------------------------------------+
*/
/*
+-----------------------------------------------------------------------------+
|                                                                             |
|   関数名     void Init(void)                                                |
|                                                                             |
|   引き数     なし                                                           |
|                                                                             |
|   返り値     なし                                                           |
|                                                                             |
|   コメント   グラフ画面を初期化する。   関数 void Axis() を使用。           |
|                                                                             |
+-----------------------------------------------------------------------------+
*/
void Init(void)
{
    /*Csloff();*/
    _setvideomoderows(_98RESS16COLOR,25); /* ビデオモード・テキスト行数設定 */
    _setbkcolor(0x010101L);               /* 背景色設定 */
    _remappalette(15,0x020202L);          /* パレット番号１５の色設定 */
    _setactivepage(0);                    /* アクティブ・ページを０に設定 */
    _setvisualpage(0);                    /* ビジュアル・ページを０に設定 */
    _setviewport((int)AREASX+gparm.area.minx,(int)AREASY+gparm.area.miny,¥
            (int)AREASX+gparm.area.maxx,(int)AREASY+gparm.area.maxy);
    Axis(); /* 座標軸を書く */              /* ビューポートの設定 */
}

/*
+-----------------------------------------------------------------------------+
|                                                                             |
|   関数名     void Set_color(palet_no)                                       |
|                                                                             |
|   引き数     int palet_no : グラフの色（描画する点の色）                    |
|                                                                             |
|   返り値     なし                                                           |
|                                                                             |
|   コメント   palet_no を構造体gparm.pocolor に格納し、グラフの色を設定する。|
```

```
|                                                                          |
+--------------------------------------------------------------------------+
*/
void Set_color(int palet_no)
{
    gparm.pocolor = palet_no;
}

/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名     int Draw(datmin,datmax,x[],y[])                             |
|                                                                          |
|   引き数     int   datmin,datmax : データ配列の添字の最小値，最大値       |
|              float x[] , y[]     : float型の任意のデータ配列のポインター |
|                                                                          |
|   返り値     0 : 正常終了     1 : ｘ軸の描画範囲からはみ出した           |
|              2 : ｙ軸の描画範囲からはみ出した                            |
|              3 : ｘ，ｙ両方の描画範囲からはみ出した                      |
|                                                                          |
|   コメント   ( x[datmin] , y[datmin] ) から ( x[datmax] , y[datmax] ) まで |
|              のデータを、構造体 gparm.pocolor の色で、gparm.scale.x,y の |
|              スケーリングで、グラフにする（点をプロットする）。          |
|                                                                          |
+--------------------------------------------------------------------------+
*/
int Draw(int gdatmin,int gdatmax,float x[],float y[])
{
    int  i,exflag=0x0000;
    double dxp,dyp;

    _setcolor(gparm.pocolor);
    for(i=gdatmin;i<=gdatmax;i++){
        dxp =  gparm.scale.x * (double)x[i] + gparm.area.zerox;
        dyp = -gparm.scale.y * (double)y[i] + gparm.area.zeroy;
        if((dxp>=(double)gparm.area.minx)&&(dxp<=(double)gparm.area.maxx)&&¥
            (dyp>=(double)gparm.area.miny)&&(dyp<=(double)gparm.area.maxy))
                _setpixel((int)dxp,(int)dyp);
        else if((dxp<(double)gparm.area.minx)||(dxp>(double)gparm.area.maxx))
            exflag |= 0x0001;
        else if((dyp<(double)gparm.area.miny)||(dyp>(double)gparm.area.maxy))
            exflag |= 0x0002;
    }
    _setcolor(WHITE);
    return(exflag);
}

/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名     void Axis(void)                                             |
|                                                                          |
|   引き数     なし                                                        |
|                                                                          |
|   返り値     なし                                                        |
|                                                                          |
|   コメント   ビューポート内をクリアし、構造体 gparm.axcolor の色で、     |
|              gparm.area , gparm.zerop などの範囲の情報をもとに座標軸を書く。|
```

```
|            関数 void Sclpoint() を使って、目盛り及びグリッドも打つ。          |
|                                                                              |
+------------------------------------------------------------------------------+
*/
void Axis(void)
{
    _clearscreen(_GVIEWPORT);
    Sclpoint();
    _setcolor(gparm.axcolor);
    _moveto(gparm.area.minx,gparm.area.zeroy);
    _lineto(gparm.area.maxx,gparm.area.zeroy);
    _moveto(gparm.area.zerox,gparm.area.miny);
    _lineto(gparm.area.zerox,gparm.area.maxy);
}
/*
+------------------------------------------------------------------------------+
|                                                                              |
|   関数名     void Sclpoint(void)                                             |
|                                                                              |
|   引き数     なし                                                            |
|                                                                              |
|   返り値     なし                                                            |
|                                                                              |
|   コメント   目盛り及びグリッドを書く。但し、構造体 gparm.dsp.x,y が -1.0    |
|              のときは、目盛り及びグリッドは書かない。                         |
|              関数 void Axis() の下請け関数。                                 |
|              関数 void Dscalepointx(),Dscalepointy() を使用。                |
+------------------------------------------------------------------------------+
*/
void Sclpoint(void)
{
    double  ix,iy,dspx,dspy;

    if (gparm.dsp.x!=-1.0) {
        ix=0.0;
        while(1) {
            dspx=gparm.scale.x*ix+gparm.area.zerox;
            if (dspx>(double)gparm.area.maxx)
                break;
            if (dspx>=(double)gparm.area.zerox) {
                Dscalepointx((int)dspx);
                ix+=gparm.dsp.x;
            }
        }
        ix=0.0;
        while(1) {
            dspx=gparm.scale.x*ix+gparm.area.zerox;
            if (dspx<(double)gparm.area.minx)
                break;
            if (dspx<=(double)gparm.area.zerox) {
                Dscalepointx((int)dspx);
                ix-=gparm.dsp.x;
            }
        }
    }
    if (gparm.dsp.y!=-1.0) {
        iy=0.0;
        while(1) {
```

```
            dspy=-gparm.scale.y*iy+gparm.area.zeroy;
            if (dspy<(double)gparm.area.miny)
                break;
            if (dspy<=(double)gparm.area.zeroy) {
                Dscalepointy((int)dspy);
                iy+=gparm.dsp.y;
            }
        }
        iy=0.0;
        while(1) {
            dspy=-gparm.scale.y*iy+gparm.area.zeroy;
            if (dspy>(double)gparm.area.maxy)
                break;
            if (dspy>=(double)gparm.area.zeroy) {
                Dscalepointy((int)dspy);
                iy-=gparm.dsp.y;
            }
        }
    }
}
/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名     void Dscalepointx(dspx)                                  |
|              void Dscalepointy(dspy)                                  |
|                                                                       |
|   引き数     int dspx,dspy : 目盛り及びグリッドが書かれる画面上の座標  |
|                                                                       |
|   返り値     なし                                                     |
|                                                                       |
|   コメント   x軸に対しては void D--x() が 引き数 dspx で示されるx座標に、|
|              y軸に対しては void D--y() が 引き数 dspy で示されるy座標に、|
|              構造体 gparm.spcolor の色でグリッドを書き、gparm.axcolor の色|
|              で目盛りを書く。  関数 void Sclpoint() の下請け関数。      |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void Dscalepointx(int dspx)
{
    _setcolor(gparm.spcolor);
    _moveto(dspx,gparm.area.miny);
    _lineto(dspx,gparm.area.maxy);
    _setcolor(gparm.axcolor);
    _setpixel(dspx,gparm.area.zeroy-1);
    _setpixel(dspx,gparm.area.zeroy-2);
}
void Dscalepointy(int dspy)
{
    _setcolor(gparm.spcolor);
    _moveto(gparm.area.minx,dspy);
    _lineto(gparm.area.maxx,dspy);
    _setcolor(gparm.axcolor);
    _setpixel(gparm.area.zerox+1,dspy);
    _setpixel(gparm.area.zerox+2,dspy);
}

/*
+-----------------------------------------------------------------------+
```

```
|                                                                          |
|   関数名     void Avmaxmin(datmin,datmax,y[])                            |
|                                                                          |
|   引き数     int   datmin,datmax : データ配列の添字の最小値，最大値       |
|              float y[]           : float型の任意のデータ配列のポインター |
|                                                                          |
|   返り値     なし                                                        |
|                                                                          |
|   コメント   データ配列 y[datmin]←→y[datmax] の最大値 ，最小値を見つけて、|
|              構造体 gparm.value.max ， gparm.value.min に格納する。      |
|              関数 double Max() ， Min() を使用。                          |
+--------------------------------------------------------------------------+
*/
void Avmaxmin(int datmin,int datmax,float y[])
{
    gparm.value.max=Max(datmin,datmax,y);
    gparm.value.min=Min(datmin,datmax,y);
}
/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名     double Max(gdatmin,gdatmax,maxi[])                          |
|              double Min(gdatmin,gdatmax,mini[])                          |
|                                                                          |
|   引き数     int   gdatmin,gdatmax : データ配列の添字の最小値，最大値    |
|              float maxi[] ， mini[] : float型の任意のデータ配列のポインター |
|                                                                          |
|   返り値     データの最大値 ，最小値 (double型)                          |
|                                                                          |
|   コメント   データ配列 maxi[datmin]←→maxi[datmax] の最大値 ，          |
|              データ配列 mini[datmin]←→mini[datmax] の最小値 を見つける。 |
|                                                                          |
+--------------------------------------------------------------------------+
*/
double Max(int gdatmin,int gdatmax,float maxi[])
{
    int     i;
    double  max=(double)maxi[gdatmin];

    for (i=gdatmin+1;i<=gdatmax;i++)
        max=max(max,(double)maxi[i]);
    return(max);
}
double Min(int gdatmin,int gdatmax,float mini[])
{
    int     i;
    double  min=(double)mini[gdatmin];

    for (i=gdatmin+1;i<=gdatmax;i++)
        min=min(min,(double)mini[i]);
    return(min);
}

/*
+--------------------------------------------------------------------------+
|                                                                          |
|   関数名     void Autoscale(gdatmin,gdatmax,x[],y[])                     |
|                                                                          |
```

```
|   引き数     int   gdatmin,gdatmax : データ配列の添字の最小値，最大値          |
|              float x[] , y[]        : float型の任意のデータ配列のポインター   |
|                                                                                |
|   返り値     なし                                                              |
|                                                                                |
|   コメント   構造体 gparm.scale.autox,autoy の値が 1 の時、一単位何ドットか    |
|              という縮尺（構造体 gparm.scale.x,y の値）を、データ配列           |
|              x[gdatmin]←→x[gdatmax] , y[gdatmin]←→y[gdatmax] の最大値 ・      |
|              最小値から自動的に決定する。但し、データの最大値・最小値が共に    |
|              正だったり負だったりした場合、最小値←→最大値という範囲ではな     |
|              く、０←→最大値、または、最小値←→０という範囲をもとにして        |
|              決定する。また、その範囲を 構造体 gparm.scale.exrtx,y の値だけ   |
|              割増しすることができる。（つまり、グラフの大きさは               |
|              1/(1+gparm.scale.exrtx)倍(for x) 及び 1/(1+gparm.scale.exrty)倍 |
|              (for y) されることになる。）  関数 double Max(),Min() を使用。   |
|               (注) min,max が共に 0.0 のときは何もしない。                     |
+--------------------------------------------------------------------------------+
*/
void Autoscale(int gdatmin,int gdatmax,float x[],float y[])
{
    double  max,min,area;

    if (gparm.scale.autox==1) {
        max=Max(gdatmin,gdatmax,x);
        min=Min(gdatmin,gdatmax,x);
        if ((max*min)<0.0) {
            area=max-min;
            max+=area*gparm.scale.exrtx;
            min-=area*gparm.scale.exrtx;
            gparm.scale.x=min(AREADX/max,AREAOX/fabs(min));
        }
        else {
            if (max>0.0) {
                area=max;
                max+=area*gparm.scale.exrtx;
                gparm.scale.x=AREADX/max;
            }
            else if (min<0.0) {
                area=-min;
                min-=area*gparm.scale.exrtx;
                gparm.scale.x=AREAOX/fabs(min);
            }
        }
    }
    if (gparm.scale.autoy==1) {
        max=Max(gdatmin,gdatmax,y);
        min=Min(gdatmin,gdatmax,y);
        if ((max*min)<0.0) {
            area=max-min;
            max+=area*gparm.scale.exrty;
            min-=area*gparm.scale.exrty;
            gparm.scale.y=min(AREADY/max,AREAOY/fabs(min));
        }
        else {
            if (max>0.0) {
                area=max;
                max+=area*gparm.scale.exrty;
                gparm.scale.y=AREADY/max;
```

```
            }
            else if (min<0.0) {
                area=-min;
                min-=area*gparm.scale.exrty;
                gparm.scale.y=AREAOY/fabs(min);
            }
        }
    }
}

/*
+-----------------------------------------------------------------------------+
|                                                                             |
|   関数名     void Autoscalepoint(gdatmin,gdatmax,x[],y[])                   |
|                                                                             |
|   引き数     int   gdatmin,gdatmax : データ配列の添字の最小値, 最大値       |
|              float x[] , y[]       : float型の任意のデータ配列のポインター |
|                                                                             |
|   返り値     なし                                                           |
|                                                                             |
|   コメント   構造体 gparm.dsp.autox,autoy の値が 1 の時、一目盛り何単位か   |
|              (構造体 gparm.dsp.x,y の値) を、データ配列 x[gdatmin]←→       |
|              x[gdatmax] , y[gdatmin]←→y[gdatmax]の最大値,最小値から自動的 |
|              に決定する。但し、データの最大値・最小値が共に正だったり負だっ |
|              たりした場合、最小値←→最大値という範囲ではなく、0←→最大値 |
|              または、最小値←→0という範囲をもとにしてその範囲を何分割する |
|              か (構造体 gparm.dsp.divx,divyの値) によって決定する。       |
|              関数 double Subasp() , double Max(),Min() を使用。           |
|              (注) abmin,abmax が共に 0.0 のときは何もしない。             |
+-----------------------------------------------------------------------------+
*/
void Autoscalepoint(int gdatmin,int gdatmax,float x[],float y[])
{
    double  a,abmax,abmin;

    if (gparm.dsp.autox==1) {
        abmax=fabs(Max(gdatmin,gdatmax,x));
        abmin=fabs(Min(gdatmin,gdatmax,x));
        if ((abmax!=0.0)||(abmin!=0.0)) {
            if (abmax>=abmin)
                a=abmax/gparm.dsp.divx;
            else if (abmax<abmin)
                a=abmin/gparm.dsp.divx;
            gparm.dsp.x=Subasp(a);
        }
    }
    if (gparm.dsp.autoy==1) {
        abmax=fabs(Max(gdatmin,gdatmax,y));
        abmin=fabs(Min(gdatmin,gdatmax,y));
        if ((abmax!=0.0)||(abmin!=0.0)) {
            if (abmax>=abmin)
                a=abmax/gparm.dsp.divy;
            else if (abmax<abmin)
                a=abmin/gparm.dsp.divy;
            gparm.dsp.y=Subasp(a);
        }
    }
}
```

```
/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名     double Subasp(a)                                         |
|                                                                       |
|   引き数     double a      : データ範囲を適当に分割した値             |
|                                                                       |
|   返り値     区切りのいい目盛りの値                                   |
|                                                                       |
|   コメント   データ範囲を適当に分割した値から、区切りのいい目盛りの値を一つ |
|              見つけだす。     関数 void Autoscalepoint() の下請け関数。 |
|                                                                       |
+-----------------------------------------------------------------------+
*/
double Subasp(double a)
{
    double a1,a2,a5,a0,asp1,asp2,ret;

    a1=pow((double)10.0,floor(log10(a)));
    a2=a1*2.0;
    a5=a1*5.0;
    a0=a1*10.0;

    if((a>=a1)&&(a<a2)){
        asp1=a1;
        asp2=a2;
    }
    else if((a>=a2)&&(a<a5)){
        asp1=a2;
        asp2=a5;
    }
    else if((a>=a5)&&(a<a0)){
        asp1=a5;
        asp2=a0;
    }
    if((a/asp1)<(asp2/a))
        ret=asp1;
    else if((a/asp1)>=(asp2/a))
        ret=asp2;

    return(ret);
}

/*
+-----------------------------------------------------------------------+
|                                                                       |
|   関数名     void End(void)                                           |
|                                                                       |
|   引き数     なし                                                     |
|                                                                       |
|   返り値     なし                                                     |
|                                                                       |
|   コメント   グラフィック画面をデフォルトに戻す。                     |
|                                                                       |
+-----------------------------------------------------------------------+
*/
void End(void)
{
```

```
    _setvideomode(_DEFAULTMODE);
    _setbkcolor(_98BLACK);
    _clearscreen(_GCLEARSCREEN);
}
```

```
/*
+-----------------------------------------------------------------------+
|                              avgdrv.c                                 |
|                             --------------                            |
|                                                                       |
|              拡張グラフィクスドライバ・ファンクションモジュール        |
|                                                                       |
|          アスキー出版、ＰＣ－９８００シリーズテクニカルデータブック    |
|                        ＭＵＬＴＩＭＥＤＩＡ編参照                     |
|                                                                       |
|                                                                       |
+-----------------------------------------------------------------------+
*/

#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
#include <string.h>
#include <io.h>
#include "afm_def.h"

#define DEVGRDRV.    "AVGDRV$$"
#define INTGRDRV     0xd8

/* 画面設定系ファンクション */

#define CMDSETSYNC  0x00
#define CMDASDSYNC  0x01
#define CMDALLOCMODE    0x02
#define CMDASKMODE  0x03
#define CMDINITFRAME    0x04
#define CMDSETFRAME 0x05
#define CMDASKFRAME 0x06
#define CMDCOMPOSE  0x07
#define CMDASKCOMP  0x08
#define CMDINIPAL   0x09
#define CMDSETPAL   0x0a
#define CMDASKPAL   0x0b
#define CMDSTART    0x19
#define CMDEND      0x1a
#define CMDGETINFO  0x1b

/* ジオメトリックファンクション */

#define CMDINITGLO  0x70
#define CMDASKGLO   0x71
#define CMDSELFRAME 0x72
#define CMDCLS      0x73
#define CMDVIEW     0x74
#define CMDPSET     0x75
#define CMDSTYLE    0x76
#define CMDLINE     0x77
#define CMDCONNECT  0x78
#define CMDBOX      0x79
#define CMDBOXFILL  0x7a
#define CMDCIRCLE   0x7b
```

```
#define CMDPAINT1   0x7c
#define CMDPAINT2   0x7d
#define CMDPOINT    0x7e
#define CMDPUTPAT   0x80
#define CMDGETPAT   0x81
#define CMDMOVPAT   0x82

/*
+-----------------------------------------------------------------------+
|                          プロトタイプ宣言                             |
+-----------------------------------------------------------------------+
*/
void end_videodrv(void);
int  call_videodrv(int cmd,union _REGS *regs);
int  start_videodrv(void);
void init_avgdrv(int mode);
void cls_256(void);
void set_palet( unsigned char r,
        unsigned char g,
        unsigned char b,
        unsigned char color_no);
void set_256palet(int);
void set_line_style(unsigned int stile);
void set_pxel(int x,int y,unsigned int color);
void set_line(int x1,int y1,int x2,int y2,unsigned int color,int attr);
void line_to(int x,int y,unsigned int color);
void set_box(int x1,int y1,int x2,int y2,unsigned intcolor);
void set_box_fill(int x1,int y1,int x2,int y2,unsigned int color);
void set_pattern(int dx,int dy, unsigned int s_off,unsigned int s_seg,
                         int x,int y,int attr);
void get_pattern(int dx,int dy,int x,int y,unsigned int d_off,
                      unsigned int d_seg,int attr);
void move_pattern(int size_x,int size_y,int sx,int sy,int dx,int dy,int attr);
void set_color_bar(int x1,int y1,int width);
void set_grid(int x1,int y1,int length,float interval,unsigned int stile
                      ,int col,int attr);
/*
+-----------------------------------------------------------------------+
|                           グローバル変数                              |
+-----------------------------------------------------------------------+
*/
int far *param,buffer[20];

union _REGS regs;
struct _SREGS sregs;

/*
+-----------------------------------------------------------------------+
|                              コード                                   |
+-----------------------------------------------------------------------+
*/
/*-----------------------------------------------------------------------+

    関数名      void end_videodrv(void)

    コメント      拡張グラフィクスドライバの使用終了宣言

+-----------------------------------------------------------------------*/
```

```
void end_videodrv(void)
{
    call_videodrv(CMDEND,&regs);
}

/*----------------------------------------------------------------------+

    関数名      int call_videodrv(int cmd,union _REGS *regs)

    引数        cmd:    拡張グラフィクスドライバ・ファンクションナンバー
                *regs:  レジスター共用体

    返値        ０以外：エラーの起こったファンクションナンバー

    コメント      ＡＨレジスターにファンクションナンバーをセットして、ソフト
                ウエア割り込みにより拡張グラフィクスドライバーのファンクション
                を呼び出す。

+----------------------------------------------------------------------*/
int call_videodrv(int cmd,union _REGS *in_regs)
{
    union _REGS out_regs;

    in_regs->h.ah = cmd;
    _int86x(INTGRDRV,in_regs,&out_regs,&sregs);
    if(out_regs.x.cflag == 1){
        fprintf(stderr,"拡張ドライバーエラー　Ｎｏ%x¥n",cmd);
        end_videodrv();
        exit(cmd);
    }
    return(0);
}

/*----------------------------------------------------------------------+

    関数名      int start_videodrv(void)

    引数        無し

    返値        ０以外：エラー

    コメント      拡張グラフィクスドライバーが組み込んであるかの確認と、
                拡張グラフィクスドライバの使用開始宣言

+----------------------------------------------------------------------*/
int start_videodrv(void)
{
    int fh;

    if((fh = open(DEVGRDRV,0)) == EOF){
        fprintf(stderr,"拡張ドライバーが組み込まれていません¥n");
        exit(1);
    }
    close(fh);

    param = buffer;

    call_videodrv(CMDSTART,&regs);
```

```
    return(0);
}

/*-----------------------------------------------------------------------+

    関数名      void init_avgdrv(int mode)

    引数        mode:   １：６４０×４８０ドット
                        ２：６４０×４００ドット

    返値        無し

    コメント      拡張グラフィクスドライバーの各種初期化

+-----------------------------------------------------------------------*/
void init_avgdrv(int mode)
{
    start_videodrv();

    if(mode == PC9821){         /* 同期モードの設定 */
        regs.h.al = 0x0c;       /* 31KHZ */
        regs.h.bh = 0x31;       /* 480ライン, 25 行 */
        regs.h.bl = 0x01;       /* テキスト表示許可 */
        call_videodrv((int)CMDSETSYNC,&regs);
    }else{
        regs.h.al = 0x08;       /* 24KHZ */
        regs.h.bh = 0x21;       /* 400ライン, 25 行 */
        regs.h.bl = 0x01;       /* テキスト表示許可 */
        call_videodrv((int)CMDSETSYNC,&regs);
    }

    regs.h.al = 0x40;       /* 画面モードの設定 */
    regs.h.bh = 0x09;
    regs.h.bl = 0x09;
    call_videodrv(CMDALLOCMODE,&regs);

    regs.h.al = 0x00;       /* 画面フレームの初期化 */
    regs.x.bx = 0xffd8;     /*-40*/
    regs.x.cx = 0;
    regs.h.dh = 0x00;
    regs.h.dl = 0x00;
    call_videodrv(CMDSETFRAME,&regs);

    regs.h.al = 0x00;       /* 画面優先順位の設定 */
    regs.h.bh = 0x20;
    regs.h.bl = 0x00;
    regs.h.ch = 0x00;
    call_videodrv(CMDCOMPOSE,&regs);

    regs.h.al = 0x00;       /* ジオメトリック機能の初期化 */
    regs.x.bx = 0x2000;
    call_videodrv(CMDINITGLO,&regs);

    regs.h.al = 0x01;       /* 描画フレームの設定 */
    regs.h.bl = 0x10;
    call_videodrv(CMDSELFRAME,&regs);
```

```
    if(mode == PC9821){
        param[0] = 0;           /* 描画領域の設定 */
        param[1] = 0;
        param[2] = 639;
        param[3] = 479;
        _segread(&sregs);
        regs.x.bx = _FP_OFF(param);
        call_videodrv(CMDVIEW,&regs);
    }else{
        param[0] = 0;           /* 描画領域の設定 */
        param[1] = 0;
        param[2] = 639;
        param[3] = 399;
        sregs.es = _FP_SEG(param);
        regs.x.bx = _FP_OFF(param);
        call_videodrv(CMDVIEW,&regs);
    }
}

/*-----------------------------------------------------------------------+

    関数名      void cls_256(void)

    引数        無し

    返値        無し

    コメント      拡張グラフィクス画面のクリア

+-----------------------------------------------------------------------*/
void cls_256(void)
{
    call_videodrv(CMDCLS,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void set_palet( unsigned char r,unsigned char g,
                                unsigned char b,unsigned char color_no)

    引数        r:          赤の輝度（０～２５５）
                g:          青の輝度（０～２５５）
                b:          緑の輝度（０～２５５）
                color_no:   設定するパレットナンバー

    返値        無し

    コメント      指定されたパレットナンバーのパレットの赤、青、緑の輝度を
                設定する

+-----------------------------------------------------------------------*/
void set_palet( unsigned char r,
        unsigned char g,
        unsigned char b,
        unsigned char color_no)
{
    regs.h.al = 0x04;
    regs.h.bh = color_no;
```

```
    regs.h.bl = b;
    regs.h.ch = r;
    regs.h.cl = g;
    call_videodrv(CMDSETPAL,&regs);
}


/*-----------------------------------------------------------------------+

    関数名      void set_256palet(int palet_mode)

    引数        palet_mode: MONOCHROME  0
                            RED         1
                            GREEN       2
                            YELLOW      3
                            BLUE        4
                            MAGENTA     5
                            CYAN        6
                            BRGW        7

    返値        無し

    コメント      指定されたパレットモードのパレットに設定する

+-----------------------------------------------------------------------*/
void set_256palet(int palet_mode)
{
    int i,r,g,b;

    if(palet_mode == MONOCHROME){
        r = g = b = i = 0;
        set_palet((unsigned char)r,(unsigned char)g,
                    (unsigned char)b,(unsigned char)i);

        for(i = 1;i <= 30;i++){
            r = i;
            g = i;
            b = i + 30;

            set_palet((unsigned char)r,(unsigned char)g,
                    (unsigned char)b,(unsigned char)i);
        }
        for(i = 31;i <= 255;i++){
            r = i;
            g = i;
            b = (int)(60 + (float)195 / 225 * (i - 30));

            set_palet((unsigned char)r,(unsigned char)g,
                    (unsigned char)b,(unsigned char)i);
        }
    }else if(palet_mode <= 6){
        for(i = 0;i <= 225;i++){
            if(palet_mode & RED)
                r = i + 30;
            else
                r = 0;
            if(palet_mode & GREEN)
                g = i + 30;
```

```
        else
            g = 0;
        if(palet_mode & BLUE)
            b = i + 30;
        else
            b = 0;

        set_palet((unsigned char)r, (unsigned char)g,
                (unsigned char)b, (unsigned char)i);
    }
    for(i = 226;i <= 255;i++){
        if(palet_mode & RED)
            r = 255;
        else
            r = 255 / 30 * (i - 225);
        if(palet_mode & GREEN)
            g = 255;
        else
            g = 255 / 30 * (i - 225);
        if(palet_mode & BLUE)
            b = 255;
        else
            b = 255 / 30 * (i - 225);

        set_palet((unsigned char)r, (unsigned char)g,
                (unsigned char)b, (unsigned char)i);
    }
}else if(palet_mode == BRGW){
    for(i = 0;i <= 55;i++){
        r = 0;
        g = 0;
        b = (i + 30) * 3;

        set_palet((unsigned char)r, (unsigned char)g,
                (unsigned char)b, (unsigned char)i);
    }

    for(i = (unsigned int)56;i <= (unsigned int)140;i++){
        r = (i - 55) * 3;
        g = 0;
        b = 255 - (i - 55) * 3;

        set_palet((unsigned char)r, (unsigned char)g,
                (unsigned char)b, (unsigned char)i);
    }

    for(i = 141;i <= 225;i++){
        r = 255 - (i - 140) *3;
        g = (i - 140) * 3;
        b = 0;

        set_palet((unsigned char)r, (unsigned char)g,
                (unsigned char)b, (unsigned char)i);
    }

    for(i = 226;i <= 255;i++){
        r = (i - 226) * (255 / 30);
        g = 255;
```

```
            b = (i - 226) * (255 / 30);

            set_palet((unsigned char)r,(unsigned char)g,
                    (unsigned char)b,(unsigned char)i);
        }
    }
}
/*-----------------------------------------------------------------------+

    関数名      void set_pxel(int x,int y,unsigned int color)

    引数        x:      x座標
                y:      y座標
                color:  パレットナンバー

    返値        無し

    コメント      指定された座標にパレットナンバーの色の点を表示

+-----------------------------------------------------------------------*/
void set_pxel(int x,int y,unsigned int color)
{
    param[0] = x;
    param[1] = y;
    param[2] = color;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_SEG(param);
    call_videodrv(CMDPSET,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void set_line_style(unsigned int stile)

    引数        stile:  スタイルを示すドット

    返値        無し

    コメント      ラインスタイルの設定

+-----------------------------------------------------------------------*/
void set_line_style(unsigned int stile)
{
    regs.x.bx = stile;
    call_videodrv(CMDSTYLE,&regs);
}

void set_line(int x1,int y1,int x2,int y2,unsigned int color,int attr)
{
    param[0] = x1;
    param[1] = y1;
    param[2] = x2;
    param[3] = y2;
    param[4] = color;
    param[5] = attr * 256;
    regs.h.al = 0x00;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
```

```
    call_videodrv(CMDLINE,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void line_to(int x,int y,unsigned int color)

    引数        x:      x座標
                y:      y座標
                color:  パレットナンバー

    返値        無し

    コメント      前回の終点から指定された座標まで線分を描画

+-----------------------------------------------------------------------*/
void line_to(int x,int y,unsigned int color)
{
    param[0] = x;
    param[1] = y;
    param[2] = color;
    regs.h.al = 0x01;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDLINE,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void set_box(int x1,int y1,int x2,int y2,unsigned int color)

    引数        x1:     左上x座標
                y1:     左上y座標
                x2:     右下x座標
                y2:     右下y座標
                color:  パレットナンバー

    返値        無し

    コメント      箱の描画

+-----------------------------------------------------------------------*/
void set_box(int x1,int y1,int x2,int y2,unsigned int color)
{
    param[0] = x1;
    param[1] = y1;
    param[2] = x2;
    param[3] = y2;
    param[4] = color;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDBOX,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void set_box_fill(int x1,int y1,int x2,int y2,
                                                unsigned int color)
```

```
    引数        x1:     左上ｘ座標
                y1:     左上ｙ座標
                x2:     右下ｘ座標
                y2:     右下ｙ座標
                color:  パレットナンバー

    返値        無し

    コメント        箱の塗りつぶし描画

+-----------------------------------------------------------------------*/
void set_box_fill(int x1,int y1,int x2,int y2,unsigned int color)
{
    param[0] = x1;
    param[1] = y1;
    param[2] = x2;
    param[3] = y2;
    param[4] = color;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDBOXFILL,&regs);
}

/*-----------------------------------------------------------------------+

    関数名      void set_pattern(int dx,int dy, unsigned int s_off,
                                 unsigned int s_seg,int x,int y,int attr)

    引数        dx:     描画ｘサイズ
                dy:     描画ｙサイズ
                s_off:  描画パターン格納オフセットアドレス
                s_seg:  描画パターン格納セグメントアドレス
                x:      ｘ座標
                y:      ｙ座標
                attr:   アトリビュート

    返値        無し

    コメント        矩形パターンの描画

+-----------------------------------------------------------------------*/
void set_pattern(int dx,int dy, unsigned int s_off,unsigned int s_seg,
                             int x,int y,int attr)
{
    int *ptr;

    ptr = (int *)malloc((size_t)5500);
    if(ptr == NULL){
        exit(0x80);
    }
    memset(ptr,0xffff,5500);

    param[0] = dx;
    param[1] = dy;
    param[2] = _FP_OFF(ptr);
    param[3] = _FP_SEG(ptr);
    param[4] = s_off;
```

```
    param[5] = s_seg;
    param[6] = x;
    param[7] = y;
    param[8] = attr;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDPUTPAT,&regs);
    free(ptr);
}

/*-----------------------------------------------------------------------+

    関数名      void get_pattern(int dx,int dy,int x,int y,unsigned int d_off,
                        unsigned int d_seg,int attr)

    引数        dx:     転送ｘサイズ
                dy:     転送ｙサイズ
                x:      ｘ座標
                y:      ｙ座標
                d_off:  転送パターン格納オフセットアドレス
                d_seg:  転送パターン格納セグメントアドレス
                attr:   アトリビュート

    返値        無し

    コメント      矩形パターンの取得

+-----------------------------------------------------------------------*/
void get_pattern(int dx,int dy,int x,int y,unsigned int d_off,
                        unsigned int d_seg,int attr)
{
    int *ptr;

    ptr = (int *)malloc((size_t)5500);
    if(ptr == NULL){
        exit(0x81);
    }
    memset(ptr,0xFFFF,5500);

    param[0] = dx;
    param[1] = dy;
    param[2] = _FP_OFF(ptr);
    param[3] = _FP_SEG(ptr);
    param[4] = x;
    param[5] = y;
    param[6] = d_off;
    param[7] = d_seg;
    param[8] = attr;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDGETPAT,&regs);
    free(ptr);
}

/*-----------------------------------------------------------------------+

    関数名      void move_pattern(int size_x,int size_y,int sx,int sy,
                                        int dx,int dy,int attr)
```

```
    引数        size_x: 転送ｘサイズ
                size_y: 転送ｙサイズ
                sx:     転送元ｘ座標
                sy:     転送元ｙ座標
                dx:     転送先ｘ座標
                dy:     転送先ｙ座標
                attr:   アトリビュート

    返値        無し

    コメント        矩形パターンの取得

+-----------------------------------------------------------------------*/
void move_pattern(int size_x,int size_y,int sx,int sy,int dx,int dy,int attr)
{
    int far *ptr;

    ptr = (int far*)malloc(10000);
    if(ptr == NULL){
        exit(0x82);
    }
    memset(ptr,0xffff,5500);

    param[0] = size_x;
    param[1] = size_y;
    param[2] = _FP_OFF(ptr);
    param[3] = _FP_SEG(ptr);
    param[4] = sx;
    param[5] = sy;
    param[6] = dx;
    param[7] = dy;
    param[8] = attr;
    sregs.es = _FP_SEG(param);
    regs.x.bx = _FP_OFF(param);
    call_videodrv(CMDMOVPAT,&regs);
    free(ptr);
}

/*-----------------------------------------------------------------------+

    関数名      void set_color_bar(int x1,int y1,int width)

    引数        x1:     左上のｘ座標
                y1:     左上のｙ座標
                width:  バーの幅

    返値        無し

    コメント        カラーバーの描画

+-----------------------------------------------------------------------*/
void set_color_bar(int x1,int y1,int width)
{
    int color;

    set_box(x1,y1,x1 + width,y1 + 256 + 1,255);
    set_line_style(0xffff);
```

```
    for(color = 0;color < 256;color++){
        set_line(x1 + 1,y1 + 256 - color,
             x1 + width - 1,y1 + 256 - color,color,0);
    }
}
/*-----------------------------------------------------------------------+

    関数名      void set_grid(int x1,int y1,int length,int interval,
                                        unsigned int stile,int col)

    引数        x1:         左上のｘ座標
                y1:         左上のｙ座標
                length:     グリッドの長さ
                interval:   グリッドの間隔
                stile:      ラインスタイル
                col:        グリッドのカラー番号

    返値        無し

    コメント      グリッドの描画

+-----------------------------------------------------------------------*/
void set_grid(int x1,int y1,int length,float interval,unsigned int stile
                ,int col,int attr)
{
    int i;

    set_line_style(stile);
    for(i = 1;i < length/interval;i++){
        set_line( (int)(x1 + interval * i) ,y1,
                  (int)(x1 + interval * i) ,y1 + length,col,attr);
        set_line(x1, (int)(y1 + interval * i) ,
                    x1 + length, (int)(y1 + interval * i) ,col,attr);
    }
    set_line_style(0xffff);
}
/*
+-----------------------------------------------------------------------+
|                              終わり                                   |
+-----------------------------------------------------------------------+
*/
```