รายการอ้างอิง

1. Kainji Tasaka, Masayoshi Tamaki, Kohei Ohkubo, Somyot Srisatit, Masahiro oda, "Development of Neutron Computed Tomography Program on Microcomputer, (JAPAN Department of Nuclear Engineering , Nagoya University Furo-cho, Chikusa-ku, Nagoya 464 - 01)
2. Thomas S. Curry III et. al., Christensen's Introduction to the Physics of Diagnostic Radiology, 3rd. ed. (Lea & Febiger,Philadelphia.,), pp. 320 - 349.
3. David C.Kay and John R.Levine, Graphics File Formats, 5th. ed. (United States of America : Mcgraw-Hill Book Company., 1992), pp. 25-31.
4. Steve Rimmer, Supercharged Bitmapped Graphics, 2nd. ed. (United States of America : Mcgraw-Hill Book Company., 1992), pp. 239-275.
5. Chips and Technologies,Inc. AVer 2000 Video Window Card Technical Manual. (Singapore:Chips and Technology,Inc.,1991), (Mimeographed)

ภาคผนวก ก

โครงสร้างของไฟล์ PCX

โครงสร้างของไฟล์ PCX เริ่มต้นด้วยข้อมูล 128 ไบต์ โดยจะเก็บข้อมูลต่าง ๆ เพื่อนำไปใช้ในการ Restore ภาพ ดังตารางที่ 1

ตารางที่ 1 แสดง PCX file header

Start Byte	Size in Bytes	Contents	Interpretation
0	1	Zsoft flag	10 (0A hex) = Zsoft PCX file
1	1	Version no.	0 = PC Paintbrush 2.5 2 = " 2.8 with palette 3 = " 2.8 without palette 4 = " for Windows 5 = " 3.0 and up; also
2	1	Encodeing	1 = PCX run-length encoding
3	1	Bits per pixel	Number of bits/pixel,each pland
4	8	Image dimension	Image limits as: Xmin,Ymin,Xmax, Ymax in pixel units
12	2	Horiz.resolution	Dots/inch in X , when printed
14	2	Vert.resolution	Dots/inch in Y , when printed
16	48	Header palette	See palette discussion
64	1	Reserved	Reserved for Zsoft use; always 0
65	1	Planes	Number of color/greyscale planes
66	2	Bytes per line	Memory needed for one color plane of each horiz. line
68	2	Header palette interp.	1 = color or B&W 2 = greyscale
70	2	Video screen size, X	Number of pixels horiz. of video output -1
72	2	Video screen size, Y	Number of pixel vert. of video output -1

ข้อมูลใน Color Map ใช้ในการแทนค่า Palette Register 16 ตัว โดยที่ Palette Register 1 ตัว ของระบบแสดงผลอีจีเอจะบรรจุข้อมูล 6 บิต โดยแต่ละ 2 บิต จะแทนสีหลังคือ แดง , น้ำเงิน , เขียว ตารางที่ 2 ตัวอักษรตัวใหญ่จะแทนความเข้มของสี 75% , ตัวอักษรตัวเล็กแทน 25% , แต่ละสีหลักจะมี 4 ระดับของการแสดงสีคือ 0% , 25%, 75%, 100% ดังรูปที่ 1

ตารางที่ 2 แสดงตำแหน่งในการเก็บสี จำนวน 3 ไบต์ คือ แดง , เขียว , น้ำเงิน

ไบต์ที่	Palette	ไบต์ที่	Palette
16 , 17 , 18	0	40 , 41 , 42	8
19 , 20 , 21	1	43 , 44 , 45	9
22 , 23 , 24	2	46 , 47 , 48	10
25 , 26 , 27	3	49 , 50 , 51	11
28 , 29 , 30	4	52 , 53 , 54	12
31 , 32 , 33	5	55 , 56 , 57	13
34 , 35 , 36	6	58 , 59 , 60	14
37 , 38 , 39	7	61 , 62 , 63	15

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	R	G	B	r	g	b
x	x	75%	75%	75%	25%	25%	25%

รูปที่ 1 การแสดงระดับสีของสีหลัก

<u>การเข้ารหัสไฟล์ PCX</u>

ไฟล์ .PCX มีวิธีการเข้ารหัสที่เรียกว่า Run - Length Encoding (RLE) ซึ่งเป็นวิธีหนึ่งในการลดขนาดไฟล์ ซึ่งมีรายละเอียดดังนี้

ถ้าไบต์ใด ๆ มีค่าไม่เหมือนกับไบต์อื่น ๆ และถ้าค่า 2 บิตบนไม่เท่ากับ '11' ไบต์นั้นจะถูกเก็บลงไฟล์เลย นอกนั้นจะมีตัวนับ (Counter) อยู่คอยนับว่าเหมือนกันกี่ไบต์ แต่ต้องไม่เกิน 63 ถ้าเกินให้นำค่าตัวนับที่ได้ OR กับ 'C0h' แล้วเก็บค่าตัวนับนั้นลงไฟล์ แล้วตามด้วยค่าของไบต์นั้น ๆ

จากนั้นจึงเริ่มนับใหม่เป็น 1 ต่อไป ถ้าในกรณีของไบต์เดียวที่มีค่าบิตบนเป็น '11' ให้เขียนตัวนับ = 1 (OR with C0h = 'C1h') ลงไปในไฟล์และตามด้วยค่าไบต์นั้น ดังตารางที่ 3

เงื่อนไข	การกระทำ
ค่าที่เหมือนไบต์อื่น	นับ Counter + 1
ค่า < 'C0h' ไม่เหมือนไบต์อื่น	เก็บลงไฟล์เลย
ค่า > 'C0h' ไม่เหมือนไบต์อื่น Counter > 63	เก็บ 'C1h' ลงไฟล์ตามด้วยค่าไบต์นั้น Count = 1 เก็บค่า 'FFh' และค่าไบต์ลงไฟล์

สำหรับการถอดรหัสเพื่อนำข้อมูลไฟล์ PCX มาแสดงทางหน้าจอภาพ จะทำตรงกันข้ามกับการเข้ารหัสไฟล์ PCX

โครงสร้างของไฟล์ TGA

ไฟล์ TGA เป็นไฟล์บิตแมพกราฟิกที่พัฒนาขึ้นโดยบริษัท Truevision Company ไฟล์ข้อมูลจะเริ่มด้วยส่วน Header ของไฟล์ดังตารางที่ 1

ตารางที่ 1 แสดงโครงสร้างของไฟล์ TGA

Offset	Length	Description
0	1	ID field length
1	1	Color map type
2	1	Image type
		Color map
3	2	First color map entry
5	2	Color map length
7	1	Color map entry size
		Image data
8	2	Image X origin
10	2	Image Y origin
12	2	Image width
14	2	Image height
16	1	Bits per pixel
17	1	Image descriptor bits

- ID field length เป็นตัวบอกความยาวของข้อความอธิบายภาพ มีค่าไม่เกิน 255 โดยข้อความนี้จะอยู่ในตำแหน่งถัดจาก header หรืออยู่ในตำแหน่งไบต์ที่ 19 จากส่วนต้นของไฟล์

- Color map type บอก colormap ที่ใช้ในภาพ โดยมีค่าเป็น 0 เมื่อเป็นภาพ monochrome หรือภาพ RGB ซึ่งไม่ต้องใช้ค่า colormap แต่ถ้ามีค่าเป็น 1 หมายความว่าภาพนี้ต้องใช้ colormap มาผสมเป็น palette ด้วย

- Image type ใช้บอกชนิดของข้อมูล ดังตารางที่ 2

ตารางที่ 2 Image type codes

CODE	Description
0	No image present
1	Color - mapped, uncompressed
2	True color,uncompressed
3	Blank and white,uncompressed
9	Color - mapped,RLE compressed
10	True color,RLE compressed
11	Blank and white,RLE compressed

- Image x origin , Image y origin บอกตำแหน่งของภาพเริ่มต้นบนจอ

- Image width , Image height บอกขนาดกว้าง ยาวของภาพ

- Bit per pixel บอกจำนวนบิตของจุดสี

- Image descriptor bits บอกลักษณะการวางตัวของภาพ โดยใช้ 2 บิต คือบิต 4 และ บิต 5 ถ้าบิต 5 เป็นหนึ่ง ภาพนั้นจะสแกนจากบนลงล่าง ถ้าเป็น 0 ภาพนั้นจะสแกนจากล่างขึ้นบน ส่วนบิต 4 ถ้าเป็นหนึ่ง ภาพนั้นจะสแกนจากซ้ายไปขวา ถ้าเป็น 0 ภาพนั้นจะสแกนจากขวาไปซ้าย

ไฟล์สกุล TGA ที่ใช้งานส่วนใหญ่จะเป็นแบบ uncompressed หมายความว่า ข้อมูลภาพจะเป็นแบบบิตแมพล้วน ๆ ถ้ามีการ compressed ไว้จะต้องขยายออกก่อน การหาจำนวนไบต์ที่ใช้ในแต่ละ line สามารถหาได้จากค่า Image width ของภาพและจำนวนบิตของสี

<u>การเข้ารหัสไฟล์ TGA</u>

ไฟล์สกุล TGA สามารถเข้ารหัสเพื่อลดขนาดของไฟล์ได้ แต่ในงานวิจัยนี้ จะไม่มีการเข้ารหัส ซึ่งข้อมูลภาพจะเป็นแบบบิตแมพล้วน ๆ

ภาคผนวก ข

โปรแกรมแปลงสัญญาณภาพเป็นข้อมูลภาพ สำหรับระบบที่ใช้แผงวงจร AVER2000

โปรแกรมแปลงสัญญาณภาพเป็นข้อมูลภาพเมื่อใช้แผงวงจร AVER2000 ได้พัฒนาบน Borland c++ V 3.1 โดยเรียกใช้ AV2K.LIB ที่มาพร้อมกับแผงวงจร ขณะเดียวกันต้องมีไฟล์ AV2K.H และ AV2K.INI ประกอบอยู่ด้วย โปรแกรมแปลงสัญญาณภาพเป็นข้อมูลภาพจะประกอบด้วยตัวแปร โปรแกรมย่อยที่สำคัญดังต่อไปนี้

ตัวแปร

num - เป็นตัวแปรสำหรับจำนวนภาพทั้งหมดที่จะเก็บลงสู่หน่วยความจำสำรอง

digi - เป็นตัวแปรที่ใช้รับข้อมูลจากพอร์ตเครื่องพิมพ์ โดยใช้เป็นตัวแปรในฟังค์ชั่น main ถ้ามีค่าเป็น 127 แสดงว่ามีสัญญาณเสียงเข้ามา

โปรแกรมย่อย

befor_run() - ทำหน้าที่ให้ข้อความเตือนว่าให้เปิดเครื่องวิดีทัศน์และกดปุ่ม Enter เพื่อเตรียมเก็บข้อมูล

VideoInit() - ทำหน้าที่เปิดหน้าจอเป็นโหมดกราฟิกส์

VideoSize() - ทำหน้าที่กำหนดขนาดหน้าจอที่จะแสดงผล

VideoSaveLoad() - ทำหน้าที่ตั้งชื่อไฟล์ข้อมูลภาพที่จะเก็บให้เป็นลำดับตามตัวเลข โดยใช้ตัวแปร a เป็นตัวนับจำนวนภาพ

VideoSave() - ทำหน้าที่เก็บภาพเป็นข้อมูลภาพไว้ในหน่วยความจำสำรองโดยเก็บในรูปแบบไฟล์ TGA 24บิต

VideoMode() - ทำหน้าที่กำหนดโหมดการแสดงผลให้กับจอภาพ

wait() - เป็นโปรแกรมย่อยที่ใช้หน่วงเวลา

```
/*****************************************************************
* Program:          Video --> TGA
* Module:           EXAMPLE.C
* Description:      PC Video DOS programming example.
* Version: 1.0
* Copyright (C) Chips and Technologies, Inc. 1991
*                Teerawat Prakobphon 1994
*****************************************************************/
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <av2k.h>                       /* PC Video library include file */
#include "example.h"

int fQuit = 0;                                  /* Quit flag */
int Delay = 1;                                  /* Delay ( 1 Sec) */
int Interval = 15;                      /* Section interval (5 Sec) */
int wRevision;                                  /* Revision */
int wFormat;                                    /* Board type */
int iVideoWidth, iVideoHeight;          /* Full video width & height */
int iColor = 1;                         /* Color index */
int xExtSize,yExtSize,xStart,yStart;
int num,i,digi,xs,FP;

void befor_run()
{

    Goto_XY(20,1);
    printf("                                \n");
    Goto_XY(20,2);
    printf("                                \n");
    Goto_XY(20,3);
    printf("                                \n");
    Goto_XY(25,1);
    printf("PLAY VIDEO CASSETTE RECORDER.\n");
    Goto_XY(23,2);
    printf("AND SEARCH FOR STARTING POSITION\n");
```

```
    Goto_XY(22,3);
    printf("THEN PRESS ENTER TO START SAVING.\n");
    Goto_XY(20,25);
  /*  printf("                                    \n");
    Goto_XY(27,25);
    printf("START RUN KEY **** Enter ****\n"); */
    getch();
}


void main()
{
        ShowMessage();                          /* Control key info */

        if ( VideoInit() != 1 ) {               /* Initialize video under dos environment */
          printf("PC Video cannot be initialied. \n");
          goto End;
          }
        Wait(Interval);                         /* Wait for 5 Sec */

            /* Section 1: Scaling */
              VideoSize();                      /* Show scaling */
              if (fQuit) goto Exit;
              Wait(Interval);                   /* Wait for 5 Sec */

            befor_run();
            for(i=1;i<=num;i++)
            {
            /* Section 2: Image Saving & Loading  */
              digi = inportb(0x379);
              while(digi != 127)
              {
                digi = inportb(0x379);

              }
              Wait(Interval*9);
              VideoSaveLoad();                  /* Show saving & loading image. */
              if (fQuit) goto Exit;
```

```
                Wait(Interval);                 /* Wait for 5 Sec. */

            /* Section 6: Testing */
//              VideoTesting();                 /* Testing */

                iColor = (iColor+1)%15;         /* Change background color */
                Background(iColor);
                PCV_SetColorKey(iColor);        /* Reset color key */
            }
Exit:
        PCV_DisableVideo();                     /* Diseble video window */
        PCV_Exit();
End:
        VideoMode(0x3);                         /* Restore to normal text mode. */

} /* main */



/*************************************************************************
* VideoInit() - Change video mode, enable PV Video.                      *
*************************************************************************/
int VideoInit()
{
        VideoMode(0x12);                        /* 640x480 16-colors graphics mode */
        Background(iColor);
        if (PCV_Initialize() != 1)
            return (0);                         /* Initialize PC Video */
        iVideoWidth = PCV_GetSystemMetrics(SM_VIDEOWIDTH);
        iVideoHeight = PCV_GetSystemMetrics(SM_VIDEOHEIGHT);
        wRevision = PCV_GetSystemMetrics(SM_VERSION);
        wFormat = PCV_GetSystemMetrics(SM_BOARDTYPE);
        PCV_ClearVideoRect(0,0,CAPTURE_BUFFER_WIDTH,CAPTURE_BUFFER_HEIGHT);
        PCV_SetColorKey(iColor);                /* Set the Color Key for the Window */
        PCV_PenWindow(0, 0);
        PCV_EnableVideo();
        xExtSize = PCV_GetSystemMetrics(SM_VIDEOWIDTH)/2; /* Default video size */
        yExtSize = PCV_GetSystemMetrics(SM_VIDEOHEIGHT)/2;
```

```
        xStart = max(0,(SCREENWIDTH-xExtSize)/2);
        yStart = max(0,(SCREENHEIGHT-yExtSize)/2);
        PCV_CreateWindow(xStart,yStart,xExtSize,yExtSize,FITWINDOW);
        return( 1 );                                    /* Success */
}

/*************************************************************************
* VideoMove() - Show moving video.                                       *
*************************************************************************/
void VideoMove()
{
int i, x, y;
int xDist, yDist;

        xDist = SCREENWIDTH-xExtSize;       /* Moving distance */
        yDist = SCREENHEIGHT-yExtSize;
        x = y = 0;
        for (i=0; i<MOVENO; i++) {
            ControlKey();
            if (fQuit) return;
            PCV_SetWindowPosition(x, y);    /* Diagnoal move  */
            x = x + xDist/MOVENO;
            y = y + yDist/MOVENO;
            Wait(Delay);
        }
        PCV_SetWindowPosition(xStart,yStart);
}

/*************************************************************************
* VideoSize() - Show sized video.                                        *
*************************************************************************/
void VideoSize()
{
int xSize, ySize;
int xMax, yMax;

        xMax = min(iVideoWidth, SCREENWIDTH);
        yMax = min(iVideoHeight, SCREENHEIGHT);
```

```
        PCV_SetWindowPosition((SCREENWIDTH-xMax)/2, (SCREENHEIGHT-yMax)/2);
        PCV_SetWindowSize(xMax, yMax, FITWINDOW);
        xSize = xMax;
        ySize = yMax;

        while ( (xSize >= 32) && (ySize >= 32 ) ){
                ControlKey();                          /* Scan keyboard input */
                if(fQuit) return;
                PCV_CreateWindow((SCREENWIDTH-xSize)/2,(SCREENHEIGHT-
ySize)/2,xSize,ySize,FITWINDOW);
                xSize = xSize - SIZE_X_INC;
                ySize = ySize - SIZE_Y_INC;
                Wait(Delay/2);
                }

        PCV_SetWindowPosition(xStart, yStart);   /* Move to (xStart, yStart) */
        PCV_SetWindowSize(xExtSize, yExtSize, FITWINDOW);
}


/*************************************************************************
* VideoSaveLoad() - Saving & loading video image.                        *
*************************************************************************/
void VideoSaveLoad()
{

        char c[4];
        char a[10];
        char b[10];
        char *name;
        strcpy(a,"pic");
        itoa(i,c,10);
        strcat(a,c);
        strcpy(name,a);
        strcat(name,".tga");
        VideoSave(BM_TRG24,name);
        if(fQuit) return;
```

```
}

/*************************************************************************
* VideoSave() - Save video image.                                        *
*************************************************************************/
void VideoSave( int fType, char * filename)
{
WORD wReturnCode;

        PCV_FreezeVideo();
        Goto_XY(27,24);
        printf(" Save View Number %d ",i);
        switch (fType) {

                case BM_TRG24:
                  if ((wReturnCode = PCV_SaveImageRect( filename, 0, 0, xExtSize, yExtSize, fType, 0)) < 1)
                        PCV_TurnBorder(wReturnCode+1);
                  sound(500);
                  Wait(2);
                  sound(1000);
                  Wait(2);
                  nosound();
                  break;

                case BM_TRG16:
                  printf( " Save to 16-bit Targa file.          " );
                  if ((wReturnCode = PCV_SaveImageRect( filename, 0, 0, xExtSize, yExtSize, BM_TRG16, 0)) <
1)
                        PCV_TurnBorder(wReturnCode+1);
                  break;

                default:
                  printf(" Can not save to this file format.    ");
        }

    /*  ControlKey();*/
        PCV_UnFreezeVideo();
        Wait(Interval);
```

```
/*************************************************************************
* VideoLoad() - Load video image.                                        *
*************************************************************************/
void VideoLoad( int fType, char * filename)
{
WORD wReturnCode;

        PCV_FreezeVideo();

        Goto_XY( 20, 5);
        switch (fType) {

                case BM_TRG24:
                  printf( " Load 24-bit Targa file.          " );
                  if ((wReturnCode = PCV_LoadImageRect(filename, 0, 0)) < 1)
                        PCV_TurnBorder(wReturnCode+1);
                  break;

                case BM_TRG16:
                  printf( " Load 16-bit Targa file.          " );
                  if ((wReturnCode = PCV_LoadImageRect(filename, 0, 0)) < 1)
                        PCV_TurnBorder(wReturnCode+1);
                  break;

                default:
                  printf(" Can not load this file format.    ");
        }

        ControlKey();
        Wait(Interval);
        PCV_UnFreezeVideo();
        Wait(Interval);
}


/*************************************************************************
* Wait() - Delay for viewing the PC Video.                               *
*************************************************************************/
```

```
void Wait(int delay)   /* delay is in 1/10 seconds */
{
register int i, k;

        k = delay;
        if (delay > 0) {
        for (i = 0; i < k; i++) {
            while (!(inp(0x3da) & 0x08));
            while ((inp(0x3da) & 0x08));
            }
        }
} /* Wait */

/*************************************************************************
* ShowMessage()                                                          *
*************************************************************************/

void ShowMessage()
{
int iScanCode;
        textbackground(1);
        clrscr();
        textcolor(13);
        printf(" \n");
        printf("    DEVELOPMENT OF DATA ACQUISITION SYSTEM FOR COMPUTED TOMOGRAPHY \n");
        printf("                    USING TELEVISION TECHNIQUE \n");
        printf("                              BY  \n");
        printf("                      TEERAWAT PRAKOBPHON  \n");
        printf(" \n\n");
        textcolor(10);
        printf(" \n\n\n\n\n");
        printf(" \n\n");
        cprintf(" Number of Projection = "); scanf("%d",&num);
        cprintf(" Number of Frame/Projection = "); scanf("%d",&FP);
        cprintf("%c   press any key to start \n",7);
        iScanCode = ScanKeybd();
        if ((iScanCode == ESC_KEY) || (iScanCode == Q_KEY)) fQuit = 1;
}
```

```
/*************************************************************************
* SeanKeybd() - read the key board without echo, return extended key code *
*************************************************************************/
int SeanKeybd()
{
int ext_key_code;
int a,b;

    b = 0;
    a = getch();
    if (a == 0) {
            b = 0x100;
            a = getch();
            }
    ext_key_code = b+(a & 0xff);
    return(ext_key_code);
}

/*************************************************************************
* ControlKey() - Checking if ever press the control key.                 *
*************************************************************************/
void ControlKey()
{
        if(kbhit()){
          switch( SeanKeybd() ) {
            case ESC_KEY:                   /* Quit the program. */
            case Q_KEY:
                 fQuit = 1;
                 break;

            case P_KEY:                     /* Pause the program. */
                 while ( SeanKeybd() != C_KEY )
                 break;

            case F_KEY:                     /* Fast motion. */
                 if (Delay > 0 )
                    Delay --;
                 break;
```

```
                case S_KEY:                    /* Slow motion. */
                        if (Delay < 65534 )
                        Delay = Delay++;
                        break;

                default: return;
                }
              }
            return;
}

/*************************************************************************
* VideoMode() - Set Video Mode (BIOS function 10h, subfunction 0).      *
*************************************************************************/
int VideoMode(int mode_no)
{
union REGS reg86;
int res86;

   reg86.h.ah = 0;
   reg86.h.al = (BYTE)mode_no;
   res86 = int86(0x10,&reg86,&reg86);
   return(res86);
}

/*************************************************************************
* Goto_XY() - Positon the cursor to display, using text coordinates.    *
*************************************************************************/
void Goto_XY(int x, int y)
{
union REGS r;
        r.h.dl = (BYTE)x;
        r.h.dh = (BYTE)y;
        r.h.bh = 0;
        r.h.ah = 2;
        int86(0x10, &r, &r);
}
```

```
/*************************************************************************
* Background() - Set the Background color to blue.                       *
*************************************************************************/
void Background(int color)
{
char far *screenPtr;
uint iOffset;
uint i, j, x, y;

        outpw(0x3ce,color << 8);            /* Set color in set/reset register */
        outp(0x3ce,5);                      /* Graphics mode register */
        outp(0x3cf,0x03);                   /* Write mode 3 */

        for( i=0; i< SCREENHEIGHT; i++){
            y = i;
            for( j=0; j< SCREENWIDTH; j++){
                    x = j;
                    iOffset = SCREENWIDTH* y+ x;      /* Video memory ptr*/
                    screenPtr = (char far *) 0xa0000000L+iOffset;
                    *screenPtr = 0xff;          /* Bit mask */
                  }
            }
        outp(0x3cf,0);                      /* Write mode 0 */
}
```

โปรแกรมแปลงข้อมูลภาพเป็นข้อมูลเชิงตัวเลขสำหรับระบบที่ใช้แผงวงจร Video Blaster

โปรแกรมนี้จะแปลงข้อมูลภาพที่ถูกเก็บในรูปแบบ PCX เป็นข้อมูลเชิงตัวเลข โดยนำภาพขึ้นแสดงทางหน้าจอก่อน จากนั้นอ่านข้อมูลภาพเป็นข้อมูลเชิงตัวเลขเก็บลงหน่วยความจำสำรอง โปรแกรมได้พัฒนาขึ้นบน TURBO C V.2.0 ซึ่งมีตัวแปร โปรแกรมย่อยและฟังค์ชั่นที่สำคัญดังต่อไปนี้

ตัวแปร

Struct Hearder - เป็นตัวแปรที่ใช้เก็บโครงสร้างของไฟล์ PCX

Datacode[700] - เป็นตัวแปรที่ใช้เก็บข้อมูลเชิงตัวเลขของภาพ

MAXLINE,MAXCOLUMN - เป็นตัวแปรกำหนดขนาดความละเอียดของจอภาพ

ADDSCRN,Offset - เป็นตัวแปรบอกตำแหน่งของหน่วยความจำแสดงผล

filepcx - เป็นตัวแปรพอยน์เตอร์ของไฟล์PCX

Value - เป็นตัวแปรเก็บค่าที่อ่านได้จากไฟล์ .PCX

column,movebyte,ShitinLine,drcpcount,drope - เป็นตัวแปรที่ใช้นับและควบคุมการแสดงผลให้อยู่ในขอบเขตของจอ

plane - เป็นตัวแปรที่ใช้เลือก plane สีเพื่อการแสดงภาพ

โปรแกรมย่อย

menu() - โปรแกรมย่อยแสดงและรับข้อมูลเริ่มต้นของระบบเพื่อเก็บตัวแปรต่าง ๆ

frame1(),frame2() - โปรแกรมย่อยใช้แสดงกรอบของข้อมูล

show_end() - โปรแกรมย่อยใช้แสดงข้อมูลตอนจบโปรแกรม

char256(),outtextxy() - โปรแกรมย่อยใช้แสดงตัวอักษรในโหมดกราฟิกส์

quit_rtn() - เมื่อมี Error,Quit จะพิมพ์ข้อความแสดงให้ทราบ

putpixel() - ใช้เขียนค่าสีลงในจุดที่กำหนดทางจอภาพ

getpixel() - อ่านค่าสีจากจุดจอภาพเป็นข้อมูลเชิงตัวเลข

line() - โปรแกรมย่อยลากเส้นตรงตามจุดที่กำหนด

box() - ใช้วาดกล่องสี่เหลี่ยม

show_color() - แสดงตัวอย่างสีของภาพ 64 ระดับสี

setdac() - ใช้กำหนดค่าสีในตารางสีของภาพ

screenmono() - ใช้กำหนดค่าสีในตารางสีของภาพให้เป็นแบบสีเทา (gray scale)

Read_key() - ใช้รับค่าที่กดจากแป้นพิมพ์

Data_s() - ลากเส้นตรงสีดำ พร้อมกับเก็บค่าสีเดิมของภาพไว้ในตัวแปร

Line_Dataup() - เลื่อนเส้นตรงขึ้นข้างบน โดยเก็บค่าสีเดิมไว้ก่อนที่จะลากเส้นใหม่

Data_x(),Data_xx(),

Line_x(),Line_xr() - ใช้ลากและเลื่อนเส้นตรงตำแหน่งซ้ายขวาของภาพที่จะแปลงข้อมูลภาพ

get_data_code() - อ่านข้อมูลภาพเป็นข้อมูลเชิงตัวเลขโดยหาค่าเฉลี่ย 4 จุดด้านบนและล่างของตำแหน่งที่จะแปลงข้อมูล แล้วหาค่าเฉลี่ยจากนั้นเก็บไว้ในตัวแปร Datacode[i]

Save_data() - เก็บข้อมูลภาพที่แปลงเป็นข้อมูลเชิงตัวเลขไว้ในหน่วยความจำสำรอง

put_number() - แสดงตัวเลขในโหมดกราฟิกส์

Select_data() - เลือกตำแหน่งที่จะแปลงข้อมูลภาพโดยใช้แป้นลูกศร

ฟังค์ชั่น

get_dismode() - ฟังค์ชั่นอ่านโหมดจอภาพของแผงวงจรแสดงผล

get_BIOS_display_mode() - ฟังค์ชั่นอ่านโหมดปัจจุบันก่อนที่จะมีการรันโปรแกรม

openfile() - ฟังค์ชั่นใช้ในการเปิดไฟล์ .PCX เท่านั้น

set_dismode() - ฟังค์ชั่นเปลี่ยนโหมดการแสดงผลเป็นโหมดกราฟิกส์

Enable_Plane() - ฟังค์ชั่นเลือก plane เพื่อเขียนข้อมูล

Zoom2In() - ฟังค์ชั่นแปลงข้อมูลจาก 2 ไบต์เป็น 1 ไบต์

PutItZoorm() - ฟังค์ชั่นในการเขียนข้อมูล 1 ไบต์ที่ plane0 plane เดียว โดยผ่านการ zoom

PutIt1() - ฟังค์ชั่นในการเขียนข้อมูล 1 ไบต์ที่ plane0 plane เดียวโดยไม่ผ่านการ zoom

Show_1_Plane() - ฟังค์ชั่นใช้ควบคุมการเขียนข้อมูล 1 ไบต์ที่ plane ที่ 0

show_8inPlane() - ฟังค์ชั่นในการเขียนข้อมูล 1 ไบต์ สำหรับ 1 จุดภาพ พร้อมกันทั้ง 4 plane ในภาพ 256 สี

initColorPalette() - ฟังค์ชั่นใช้ในการเลือก color palette register ของไฟล์ที่เป็น .PCX ที่อ่านมา

```
/*****************************************************************/
/*          DATA ACQUISITION FROM PCX FILES Version 1.0
/*                          by Teerewat Prakobphon
/*****************************************************************/

#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>
#include <stdlib.h>
#include <fcn.h>

#define UP_ARROW    0x4800
#define DOWN_ARROW  0x5000
#define LEFT_ARROW  0x4B00
#define RIGHT_ARROW 0x4D00
#define END         0x4F00
#define DOWN        0x5100
#define RETURN      0x1c0d
#define ESCAPE      0x011b

void char256(int,int,int,char);
void putpixel(int,int,int);
void show_chdata();
void put_number(int,int,int,int);
void show_pic();
void save_datacon(int,int);
void run_con(int stn,int x,int xx,int y);
void show_line(int,int,int);
void se();

menu();
struct {
        unsigned char Password;
        unsigned char Version;
        unsigned char Encoding;
        unsigned char BPP;
        int         WinDemX1;
```

```
        int     WinDemY1;
        int     WinDemX2;
        int     WinDemY2;
        int     HoriResolut;
        int     VertResolut;
        unsigned char ColorMap[16][3];
        unsigned char reserve;
        unsigned char NumofPlane;
        int     BytePLine;
        int     PaletteInfo;
    } Hearder;
FILE *filepcx;
int MAXLINE;
int MAXCOLUMN;
int MaxLoop,linecount;
char D[100][300];
int Data[700];
int Data0[700];
int Datacode[700];
int Data1[600];
int Do1[600],Do2[600],Do0[600];
int Do3[600],Do4[600];
int Xd[50];
int Xa[50];
int Xd1[50];
int Xa1[50];
int ch;
int datan;
char name[10];
char ade[10];


unsigned char index,arg2[3];
unsigned ADDSCRN,Offset;
unsigned char zoom,zoomvalue[2],Value,counter,plane;
unsigned int column,movebyte,ShiftInLine,dropcount,drope;
char    CharByte[20];
```

```
menu()
{

 int x1,y1,x2,y2;
    clrscr();
    x1 = 1; y1 = 1; x2 = 79; y2 = 25;

    textcolor( LIGHTMAGENTA );
    frame(x1,y1,x2,y2);
    textcolor( LIGHTCYAN );
    gotoxy(8,5);
    cprintf("การพัฒนาระบบเก็บข้อมูลสำหรับสร้างภาพโทโมกราฟีแบบคำนวณด้วยเทคนิคโทรทัศน์\n");
    gotoxy(20,3);
    cprintf("ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ \n");
    gotoxy(30,2);
    cprintf("จุฬาลงกรณ์มหาวิทยาลัย\n");
    textcolor( LIGHTRED );
    gotoxy(5,18);
    cprintf("โปรดใส่จำนวนโพรไฟล์ที่จะเก็บ  ");
    scanf("%d",&totn);
    gotoxy(5,20);
    cprintf("โปรดใส่ชื่อโพรไฟล์แรก  ");
    scanf("%s",name);
    gotoxy(5,22);
    cprintf("จะเก็บข้อมูลไว้ในไฟล์ชื่อ");
    scanf("%s",ede);

}

frame(int x1,int y1,int x2,int y2)
{
 int i;
  gotoxy(x1,y1);
  cprintf("%c",218);
  for(i=1;i<=x2-x1-1;++i)
  cprintf("%c",196);
  cprintf("%c",191);
  for(i=1;i<=y2-y1;++i)
```

```
  {
   gotoxy(x1,y1+i);
   cprintf("%c",179);
   gotoxy(x2,y1+i);
   cprintf("%c",179);
  }
   gotoxy(x1,y2);
   cprintf("%c",192);
   for(i=1;i<=x2-x1-1;++i)
   cprintf("%c",196);
   cprintf("%c",217);


}

show_end(int otn,int detan)
{


 int x1,y1,x2,y2;


   clrscr();
   x1 = 15; y1 = 12; x2 = 65; y2 = 16;
   textcolor(BLUE);
   frame2(x1,y1,x2,y2);
   textcolor(GREEN);
   gotoxy(19,13);
   cprintf("คอมพิวเตอร์ได้เก็บข้อมูลไปแล้วจำนวน");
   gotoxy(55,13);
   cprintf("%d",otn);
   gotoxy(58,13);
   cprintf("โพรไฟล์");
   gotoxy(24,15);
   cprintf("แต่ละโพรไฟล์มีจำนวนข้อมูล = ");
   gotoxy(52,15);
   cprintf("%d",detan);
   textcolor(RED);
```

```
        gotoxy(19,20);
        oprintf("ข้อมูลจะเก็บไว้ในไดรฟ์ B ไฟล์ชื่อ %s",ada );
        getch();
    }

    frame2(int x1,int y1,int x2,int y2)
    {
     int i;
      gotoxy(x1,y1);
      oprintf("%c",201);
      for(i=1;i<=x2-x1-1;++i)
      oprintf("%c",205);
      oprintf("%c",187);
      for(i=1;i<=y2-y1;++i)
      {
        gotoxy(x1,y1+i);
        oprintf("%c",186);
        gotoxy(x2,y1+i);
        oprintf("%c",186);
      }
        gotoxy(x1,y2);
        oprintf("%c",200);
        for(i=1;i<=x2-x1-1;++i)
        oprintf("%c",205);
        oprintf("%c",188);
        gotoxy(5,5);

}

void char256(int x,int y,int color,char character)
{
  int e,ef;
  int  ea;
  ef = character;

  for(e=1;e<=20;e++)
     {
```

```
        ee = Ascii[af].CharByte[e];
        if(((ee>>7)&0x01)!=0)
        putpixel(x,y+e,color);
        if(((ee>>6)&0x01)!=0)
        putpixel(x+1,y+e,color);
        if(((ee>>5)&0x01)!=0)
        putpixel(x+2,y+e,color);
        if(((ee>>4)&0x01)!=0)
        putpixel(x+3,y+e,color);
        if(((ee>>3)&0x01)!=0)
        putpixel(x+4,y+e,color);
        if(((ee>>2)&0x01)!=0)
        putpixel(x+5,y+e,color);
        if(((ee>>1)&0x01)!=0)
        putpixel(x+6,y+e,color);
        if((ee&1)!=0)
        putpixel(x+7,y+e,color);

        }
}

void outtextxy(int x,int y,int color ,char *stringin)
{
    char *strin ;
    int  e=x;
    int  af;
    strin = stringin;
    while((*(strin)!=0) && (e<640))
            {
            af = *(strin);
            if((af>=212)&&(af<=219))
                    {
                      e = e-8;
                    }
              if((af>=231)&&(af<=238))
                    {
                     e = e-8;
                    }
```

```
                if((af>=209)&&(af<210))
                    {
                     e = e-8;
                    }
                char256(e,y,color,*(strin));
                e+=8;
                strin++;
              }
}



FILE *openfile(name)
char *name;
{
   FILE *fileptr;
   strcat(name,".pcx");
   fileptr = fopen(name,"rb");
   if(fileptr==NULL)
    {
      printf("\nError in Opening File -> %s",name);
      exit(0);
    }
   return fileptr;
}

unsigned char get_display_mode()
{
   _AH = 0x1a;
   _AL = 0;
   geninterrupt(0x10);
   return(_BL);
}

unsigned char get_BIOS_display_mode()
{
   _AH = 0x0f;
   geninterrupt(0x10);
```

```
    return(_AL);
}

void set_dismode(mode)
int mode;
{
    _AH = 0;
    _AL = mode;
    geninterrupt(0x10);
}

quit_rtn()
{
    fclose(filepcx);
    set_dismode(index);
}
void Enable_Plane(Plane_Enable)
unsigned char Plane_Enable;
{
    outportb(0x3c4,2);
    outportb(0x3c5,Plane_Enable);
}

unsigned char Zoom2In(v)
unsigned char v[2];
{
    unsigned char mask,a,b=0;
    int p,r;
    for (p=0;p<2;p++)
    {
     mask = 0x80;
     a = 0x00;
     for (r=0;r<4;r++)
     {
       a = a | ((v[p] & mask) << r);
       mask = mask >> 2;
     }
     b = b | (a >> 4*p);
```

```
    }
    return(b);
}
PutItZoom(zoomvalue)
unsigned char zoomvalue[2];
{
  if ((linecount%2)==0)
  {
   zoomvalue[zoom]=Value;
   zoom++;
   if(zoom==2)
   {
     zoom = 0;
     Value = Zoom2In(zoomvalue);
     pokeb(ADDSCRN,Offset,Value);
     Offset++;
     column++;
     if( movebyte ==column)
     {
          Offset += ShiftInLine;
          linecount++;
          column = 0;
     }
   }
  }
  else
  {
    dropcount++;
    if(dropcount==Heerder.BytePLine)
     {
     linecount++;
     dropcount = 0;
     }
  }
}
PutIt1()
{
  pokeb(ADDSCRN,Offset,Value);
```

```
    Offset++;
    column++;
    if(movebyte==column)
     {
      Offset += ShiftInLine;
      linecount++;
      column = 0;
     }
}
Show_1_Plane()
{
   int i;
   Enable_Plane(8);
   Offset = 0;
   linecount = 0;
   dropcount = 0;
   if (strcmp(arg2,"-z")==0)
   {
    if((Heerder.BytePLine/2) < MAXCOLUMN)
     {
          movebyte = Heerder.BytePLine/2;
          ShiftInLine = MAXCOLUMN - Heerder.BytePLine/2;
     }
     else
     {
          movebyte = MAXCOLUMN;
          ShiftInLine = 0;
     }
     if( ((Heerder.WinDemY2+1)/2) > MAXLINE)
          MaxLoop = MAXLINE;
     else
          MaxLoop = ((Heerder.WinDemY2 + 1)/2;

   else;
    column = 0;
    zoom = 0;
    fread(&Value,1,1,filepcx);
    while ((!feof(filepcx)) || ((MaxLoop*2)==linecount))
```

```
    {
      if (Value > 0xa0)
            {
              counter = (Value^0xa0);
              fread(&Value,1,1,filepax);
              for(i=counter;i>0;i--)
                if(stromp(arg2,"-z")==0)
                      PutItZoom(zoomvalue);
                    else PutIt1();
            }
           else
            if(stromp(arg2,"-z")==0)
              PutItZoom(zoomvalue);
            else PutIt1();
           fread(&Value,1,1,filepax);
    }
   Enable_Plane(0x0f);
}

/*put pixel in specific color on screen */
void putpixel(int XXXX,int YYYY,int CCCC)
{
    long i;
    i=((long)YYYY*MAXCOLUMN+XXXX);
    outp(0x3cd,0x40|(i>>16));
    pokeb(0xa000,i&0xffff,CCCC);
}

int getpixel(int x,int y)
{
    long i;
    register unsigned int isf;
            i=((long)y*MAXCOLUMN+(long)x);
            isf = (unsigned int)(((i>>16)&0x000f)<<4);
            outp(0x3cd,isf);
            return(unsigned int)(peekb(0xa000,(unsigned int)i&0xffffU));
}
```

```
void line(int x1,int y1,int x2,int y2,int color)
{
 register int e;
 register int ydif = y2-y1;
 register int xdif = x2-x1;
 long i;

 if(abs(ydif) > abs(xdif))
 {
  if (ydif == 0) return;
  if (y2 > y1)
 for(e = y1;e<=y2;e++)
  {
   i=((long)e*MAXCOLUMN+x1+(int)((((long)xdif*(long)(e-y1)))/ydif));
   outp(0x3cd,(int)(0x40|(i>>16)));
   pokeb(0xa000,i&0xffff,color);
   }
   else
 for(e = y1;e>=y2;e--)
  {
    i=((long)e*MAXCOLUMN+x1+(int)((((long)xdif*(long)(e-y1)))/ydif));
    outp(0x3cd,0x40|(i>>16));
    pokeb(0xa000,i&0xffff,color);
   }
 }
 else
 {
  if (xdif == 0) return;
  if (x2 > x1)
 for(e = x1;e<=x2;e++)
  {
  i=((long)(y1+(int)((((long)ydif*(long)(e-x1)))/xdif))*MAXCOLUMN+e);
  outp(0x3cd,0x40|(i>>16));
  pokeb(0xa000,i&0xffff,color);
  }
  else
 for(e = x1;e>=x2;e--)
  {
```

```
        i=((long)(y1+(int)((((long)ydif*(long)(a-x1)))/xdif))*MAXCOLUMN+a);
        outp(0x3cd,0x40|(i>>16));
        pokeb(0xa000,i&0xffff,color);
        }
    }
}

void box(int x1,int y1,int x2,int y2,int color)
{
    register int a,b;
    register long i;

    char chold=0,chnew;
    outp(0x3cd,0x40);
    for(b=y1;b<=y2;b++)
        for(a=x1;a<=x2;a++)
          {
          i=((long)b*MAXCOLUMN+a);
          if(chold !=(chnew=(0x40|(i>>16))))
            {
              chold = chnew;
              outp(0x3cd,chold);
            }
          pokeb(0xa000,i&0xffff,color);
          }
}

void show_color()
{
   int x,y,c;
   c = 0;
        for(y=1;y<=2;y++)
          for(x=0;x<=64;x++)
          {
            box((x*10)),(y*10+440),((x*10+10)),(((y+1)*10)+440),c);
            c = c+2;
          }
}
```

```
void setdac(int color,int red,int green,int blue)
{
    outportb(0x03c8,(unsigned char)color);
    outportb(0x03c9,(unsigned char)red);
    outportb(0x03c9,(unsigned char)green);
    outportb(0x03c9,(unsigned char)blue);
}

void screenmono(void)
{
    int a;
    for(a=0;a<256;a++)
        setdac(a,(int)((float)a*59.0/256.0),(int)((float)a*59.0/256.0),
                (int)((float)a*59.0/256.0));
}

void Show_BinPlane()
{
  int i,j,x,y;
  unsigned off;
  x=0;y=0;off=0;
  Enable_Plane(0x0f);
  fread(&Value,1,1,filepcx);
  while( !feof(filepcx))
   {
    if (Value > 0xc0)
        {
          counter = (Value^0xc0);
          fread(&Value,1,1,filepcx);
          for(i=counter;i>0;i--)
           {
           if(x<640)
           {
             putpixel(x,y,Value);
           }
           x++;
           if(x==Hearder.BytePLine)
```

```
                    {
                     x=0;
                     y++;
                     off+=ShiftInLine;
                     if(y==(Heerder.WinDemY2+1))
                       return;
                    }
                }
            }
        else
            {
             if(x<840)
              {
                putpixel(x,y,Value);
              }
             x++;
             if(x==Heerder.BytePLine)
              {
                   x=0;
                   y++;
                   off+=ShiftInLine;
                   if(y==(Heerder.WinDemY2+1))
                     return;
              }
            }
          fread(&Value,1,1,filepcx);
    }
    Enable_Plane(0x0f);
}
/*initial color reg in 256 color */
initColorPalette()
{
    unsigned char
    VGAColorMap[256][3],VGAPltt;
    int i,j,tem;
    fseek(filepcx,-769,SEEK_END);
    if(fread(&VGAPltt,1,1,filepcx)!=1)
      quit_rtn("Error in reading VGAColor ");
```

```
    if (VGAPltt==0x0c)
      if (fread(VGAColorMap,768,1,filepcx)!=1)
            quit_rtn("Error in reading VGAColor ");
      else;
    else
      quit_rtn("Error in Format read -- 256 color palette");
    for(i=0;i<=255;i++)
      for (j=0;j<=2;j++)
            VGAColorMap[i][j] = VGAColorMap[i][j] >> 2;
    for(i=0;i<=255;i++)
     {
      outportb(0x03c8,i);
      for (j=0;j<=2;j++)
      outportb(0x03c9,VGAColorMap[i][j]);
      }
}
void print_error()
      {
            printf("\n Loading .PCX file to EGA/VGA and VGA display");
            printf("\n       BY : MR.TEERAWAT PRAKOBPHON  ");
            printf("\n\n Usage : VGAPCX [d:Path]pcx-file(.pcx) [-option]");
            printf("\n [-option]  : '-z' => Zoom in 2 twice (only b/w)");
            printf("\n             : '-p' => Use our ColorMap as Palette Reg");
            printf("\n");
            exit(0);
      }
void se_mode()
{
    switch(get_display_mode())
    {
     case 8 : if(Heerder.BPP==8)
                  {
                    MAXLINE = 480;
                    MAXCOLUMN=640;
                    ADDSCRN = 0xA000;
                    set_dismode(0x2e);
                    initColorPalette();
                  }
```

```
                    break;
        default     : quit_rtn("Not show"); break;
        }
}

void show_pax()
{
if(Heerder.BytePLine < MAXCOLUMN)
 {
   movebyte = Heerder.BytePLine;
   ShiftInLine = MAXCOLUMN - Heerder.BytePLine;
   drope = 0;
 }
else
 {
  movebyte = MAXCOLUMN;
  ShiftInLine = 0;
  drope = Heerder.BytePLine - MAXCOLUMN;
 }
  if( (Heerder.WinDemY2+1) > MAXLINE)
      MaxLoop = MAXLINE;
  else
      MaxLoop = Heerder.WinDemY2 + 1;
  fseek(filepax,128,SEEK_SET);
  if((Heerder.NumofPlane==1)&&(Heerder.BPP==1))
     Show_1_Plane();
  else
     if(Heerder.BPP==8)
     {
       Show_8inPlane();
       show_color();
     }
}
void show_pax1()
{
if(Heerder.BytePLine < MAXCOLUMN)
 {
   movebyte = Heerder.BytePLine;
```

```
    ShiftInLine = MAXCOLUMN - Heerder.BytePLine;
    drope = 0;
  }
 else
  {
   movebyte = MAXCOLUMN;
   ShiftInLine = 0;
   drope = Heerder.BytePLine - MAXCOLUMN;
  }
  if( (Heerder.WinDemY2+1) > MAXLINE)
      MaxLoop = MAXLINE;
  else
      MaxLoop = Heerder.WinDemY2 + 1;
  fseek(filepax,128,SEEK_SET);
  if((Heerder.NumofPlane==1)&&(Heerder.BPP==1))
    Show_1_Plane();
  else
    if(Heerder.BPP==8)
    {
      Show_8inPlane();
    }
}

int Read_Key(int key)
{
   return bioskey(key);
}

void Liney()
{
    line(601,0,601,450,254);
}

Data_s(int y)
{
    int x;
    for(x=0;x<=599;x++)
          Data0[x] = getpixel(x,y);
```

```
    for(x=0;x<=599;x++)
        putpixel(x,y,254);
        getch();
    return(y);
}

Line_Dataup(int y)
{
    int x;
    for(x=0;x<=599;x++)
        Data1[x] = (int)getpixel(x,y+1);
    for(x=0;x<=599;x++)
        putpixel(x,y,Data0[x]);
    for(x=0;x<=599;x++)
        Data0[x] = Data1[x];
    return(y);
}

Line_Down(int y)
{
    int x;
    for(x=0;x<=599;x++)
        Data1[x] = getpixel(x,y-1);
    for(x=0;x<=599;x++)
        putpixel(x,y,Data0[x]);
    for(x=0;x<=599;x++)
        Data0[x] = Data1[x];
    return(y);
}

Data_x(int x,int y)
{
 int i;
 for(i=0;i<=50;i++)
  Xd[i] = getpixel(x,y-30+i);
 line(x,(y-30),x,(y+20),254);
 getch();
}
```

```
Data_xx(int xx,int y)
{
 int i;
 for(i=0;i<=50;i++)
  Xxr[i] = getpixel(xx,(y-30+i));
 line(xx,(y-30),xx,(y+20),254);
 getch();
}

Line_xd(int x,int y)
{
 int i;
 for(i=0;i<=50;i++)
  Xxd1[i] = getpixel(x-1,(y-30+i));
 for(i=0;i<=50;i++)
  putpixel(x,(y-30+i),Xxd[i]);
 for(i=0;i<=50;i++)
  Xxd[i] = Xxd1[i];
}

Line_xr(int xx,int y)
{
 int i;
 for(i=0;i<=50;i++)
  Xxr1[i] = getpixel((xx+1),(y-30+i));
 for(i=0;i<=50;i++)
  putpixel(xx,(y-30+i),Xxr[i]);
 for(i=0;i<=50;i++)
  Xxr[i] = Xxr1[i];
}

void show_line(int x,int xx,int y)
{
     Line_xd(xx,y);
     Line_xr(x,y);
     line(x,y,xx,y,0);
}
```

```
void Clear_disply()
{
    int y;
    for(y=0;y<=260;y++)
        line(0,y,600,y,125);
}

void get_data_code(int x,int xx,int y)
{
    int i;
    for(i=x;i<=xx;i++)
        {
          Do1[i] = Data1[i];
          Do0[i] = getpixel(i,y-1);
          Do2[i] = getpixel(i,y-2);
          Do3[i] = getpixel(i,y+1);
          Do4[i] = getpixel(i,y+2);
          Datacode[i] = ((Do1[i]+Do0[i]+Do2[i]+Do3[i]+Do4[i])/5);
          D[1][i] = Datacode[i];
        }
}

void get_datacon(int ctn,int x,int xx,int y)
{
    int i;
    for(i=x;i<=xx;i++)
        {
          Do1[i] = getpixel(i,y);
          Do0[i] = getpixel(i,y-1);
          Do2[i] = getpixel(i,y-2);
          Do3[i] = getpixel(i,y+1);
          Do4[i] = getpixel(i,y+2);
          Datacode[i] = ((Do1[i]+Do0[i]+Do2[i]+Do3[i]+Do4[i])/5);
          D[ctn][i]   = Datacode[i];
        }
}

void show_get_data(int x,int xx)
```

```
{
    int i;
    for(i=x;i<=xx;i++)
    putpixel(i,(Datacode[i])*2,0);
}

void save_data(int x,int xx)
{
    int i;
    FILE *fpo;
    if((fpo = fopen(ada,"wb"))==NULL)
    {
        puts("cannot open file outfile\n");
        exit(1);
    }
    for(i=x;i<=xx;i++)
    {
        fprintf(fpo,"%d\n\r",(Datacode[i])*2);
    }
    fclose(fpo);
}

void save_datacon(int x,int xx)
{
  int i;
  FILE *fpo;
  if((fpo=fopen(ada,"ab"))==NULL)
  {
    puts("cannot open file outfile\n");
    exit(1);
  }
  for(i=x;i<=xx;i++)
  {
    fprintf(fpo,"%d\n\r",(Datacode[i])*2);
  }
  fclose(fpo);
}
void put_number(int x,int y,int color,int num)
```

```
{
    int A,B,C;
    int a,b,c;
    A=(num/100);
    C=((num%100)%10);
    B=(((num-C)%100)/10);
    a=A+48;
    b=B+48;
    c=C+48;
    char256(x,y,color,a);
    x=x+8;
    char256(x,y,color,b);
    x=x+8;
    char256(x,y,color,c);
}

void run_con(int stn,int x,int xx,int y)
{
    int i;
    char s[4];
    char a[10];
    for(i=2;i<=stn;i++) {
    strcpy(a,"pic");
    itoa(i,c,10);
    strcat(a,c);
    filepcx = openfile(a);
    show_pic();
    get_datacon(stn,x,xx,y);
    save_datacon(x,xx);
    fclose(filepcx);        /* close filepcx */
    }
    quit_rtn();
    show_end(stn,datan);
    getch();
    exit(0);
}
quit_grtn()
{
```

```
    set_dismode(index);
}
Select_data(int otn)
{
    int x;
    int xx;
    int y;
    int i;

    x  = 200;
    xx = 400;
    y  = 300;
    Liney();
    for(;;)
    {
        i = Read_Key(0);
        switch(i) {
                case UP_ARROW  :
                                Data_s(y);
                                Line_Dataup(y);
                                box(610,400,635,420,125);
                                put_number(610,400,0,y);
                                y = y+1;
                                break;

                case DOWN_ARROW :
                                Data_s(y);
                                Line_Down(y);
                                y = y-1;
                                box(610,400,635,420,125);
                                put_number(610,400,0,y);
                                break;

                case LEFT_ARROW :  Data_x(x,y);
                                Line_xl(x,y);
                                x = x-1;
                                box(40,400,65,420,125);
                                put_number(40,400,0,x);
```

```
                    break;

            case RIGHT_ARROW :  Data_xx(x,y);
                    Line_xr(x,y);
                    x = x+1;
                    box(40,400,65,420,125);
                    put_number(40,400,0,x);
                    break;

            case END        :  Data_x(xx,y);
                    Line_xl(xx,y);
                    xx = xx-1;
                    box(550,400,575,420,125);
                    put_number(550,400,0,xx);
                    break;

            case DOWN       :  Data_xx(xx,y);
                    Line_xr(xx,y);
                    xx = xx+1;
                    box(550,400,575,420,125);
                    put_number(550,400,0,xx);
                    break;

            case RETURN     :  get_data_code(x,xx,y);
                    Clear_disply();
                    show_get_data(x,xx);
                    datan = xx-x;
                    show_line(x,xx,y);
                    break;

            case ESCAPE     :
                    save_data(x,xx);
                    run_con(ctn,x,xx,y);
            }
        }
    }
```

```
void show_pic()
{
    fread(&Heerder,70,1,filepcx);
    if(Heerder.Password != 0x0a)
      quit_rtn("File not .PCX format.");
    se_mode();
    show_pcx();
}

void show_pic1()
{
    fread(&Heerder,70,1,filepcx);
    if(Heerder.Password != 0x0a)
      quit_rtn("File not .PCX format.");
    se_mode();
    show_pcx1();
}

void clear_Show()
{
    int x;
    for(x=10;x<=240;x++)
        line(51,x,500,x,0);
}

void Show_pro(int x,int xx)
{

    int y;
    line(50,10,50,250,90);
    line(50,250,500,250,90);
    for(y=x;y<=xx;y++)
        putpixel(y,(D[1][y])+10,100);
}

void pre_show()
{
    char *name = "show";
```

```
    index = get_BIOS_display_mode();
    filepox = openfile(name);
    show_pic1();
    getch();
    quit_rtn();
}

void main()
{
    int i;
    int y;

    char c[4];
    char a[10];
    Readfont();
    pre_show();
    clrscr();
    menu();
      index = get_BIOS_display_mode();
    filepox = openfile(name);
    show_pic();
    getch();
    Select_data(ctn);
}
```

โปรแกรมแปลงข้อมูลภาพเป็นข้อมูลเชิงตัวเลขสำหรับระบบที่ใช้แผงวงจร AVER2000

โปรแกรมนี้จะแปลงข้อมูลที่ถูกเก็บในรูปแบบ TGA เป็นข้อมูลเชิงตัวเลข โดยนำภาพขึ้นแสดงทางหน้าจอก่อนจากนั้นอ่านข้อมูลภาพเป็นข้อมูลเชิงตัวเลข เก็บลงหน่วยความจำสำรอง โปรแกรมได้พัฒนาขึ้นบน TURBO C V.2.0 ซึ่งมีตัวแปร โปรแกรมย่อยและฟังค์ชั่นที่สำคัญดังต่อไปนี้

ตัวแปร

width,depth,bytes,bits - เป็นตัวแปรที่ใช้กำหนดขนาดของภาพ ที่จะแสดงทางหน้าจอ

n - เป็นตัวแปรที่ใช้เก็บจำนวนภาพทั้งหมดที่จะแปลงข้อมูล

Data[400],Data0[400],
Data1[400],Datap[400],
Datau[400],Datas[400] - เป็นตัวแปรที่ใช้เก็บข้อมูลเชิงตัวเลขของภาพที่ระดับบนและล่าง ของตำแหน่งที่จะแปลงข้อมุล เพื่อคำนวณหาค่าเฉลี่ย

Xxl[21],Xxr[21],
Xxl1[21],Xxr1[21] - ใช้ลากและเลื่อนเส้นตรงตำแหน่งซ้ายขวาของภาพที่จะแปลงข้อมูลภาพ

*name - เป็นตัวแปรใช้เก็บชื่อไฟล์ .TGA ที่จะแปลงข้อมูล

da[10] - ใช้เก็บชื่อไฟล์ที่เก็บข้อมูลเชิงตัวเลข

struct TGAHEAD - เป็นตัวแปรที่ใช้เก็บโครงสร้างของไฟล์ TGA

โปรแกรมย่อย

menu() - โปรแกรมย่อยแสดงและรับข้อมูลเริ่มต้นของระบบเพื่อเก็บตัวแปรต่างๆ

write_char_gen() - ใช้เขียนตัวอักษรภาษาไทยใน Text mode

box() - ใช้วาดกรอบสี่เหลี่ยมบริเวณตำแหน่งที่กำหนด

putline() - ใช้วาดเส้นตรงที่สามารถกำหนดลักษณะของเส้นได้

readtgaline() - อ่านภาพไฟล์ .TGA เก็บไว้ในหน่วยความจำ

Getkey() - เป็นฟังค์ชั่นรับค่าจากแป้นพิมพ์

information() - เป็นโปรแกรมย่อยแสดงขนาดพร้อมทั้งชื่อของภาพทางจอภาพ

unpacktga() - เป็นโปรแกรมย่อยแสดงภาพ .TGA ทางจอภาพ

Data_s() - ลากเส้นตรงสีดำ พร้อมกับเก็บค่าสีเดิมของภาพไว้ในตัวแปร

Line_up() - เลื่อนเส้นตรงขึ้นด้านบน โดยเก็บค่าสีเดิมไว้ก่อนที่จะลากเส้นใหม่

Line_Down() - เลื่อนเส้นตรงลงด้านล่าง โดยเก็บค่าสีเดิมไว้ก่อนที่จะลากเส้นใหม่

select_x(),select_xd(),

select_xdxx(),show_locate1(),

show_locate2(),show_locate3() - แสดงบริเวณที่จะแปลงข้อมูลภาพเป็นข้อมูลเชิงตัวเลข ขณะที่ใช้แป้นลูกศร

save_data() - เก็บข้อมูลภาพแรกที่แปลงเป็นข้อมูลเชิงตัวเลข ไว้ในหน่วยความจำสำรอง

save_datacon() - เก็บข้อมูลภาพต่อ ๆ ไปที่แปลงเป็นข้อมูลเชิงตัวเลขแล้ว ไว้ในหน่วยความจำสำรอง

w_config(),

save_config() - ใช้เก็บค่าตำแหน่งที่จะแปลงข้อมูลภาพลงหน่วยความจำสำรองเป็นไฟล์ชื่อ config.sys

r_config(),

load_config() - ใช้อ่านค่าตำแหน่งที่จะแปลงข้อมูลภาพจากหน่วยความจำสำรอง

select() - เลือกตำแหน่งที่จะแปลงข้อมูลภาพโดยใช้แป้นลูกศร

show_end() - แสดงข้อมูลตอนจบโปรแกรม

```
/*************************************************************************/
/*        DATA ACQUISITION FROM  TARGA FILES Version 1.0                 */
/*      Original code from Supercharged Bitmapped Graphics               */
/*            Enhanced by Teerawat Prakobphon                            */
/*                  Jan,29,1994                                          */
/*          Copyright 1992 by  Steve Rimmer                              */
/*          Copyright 1994 by  Teerawat Prakobphon                       */
/*      From VGA mode 13, Tseng Labs Super VGA, Trident Super VGA        */
/*************************************************************************/

#include "stdio.h"
#include "conio.h"
#include "dos.h"
#include "alloc.h"
#include "mem.h"
#include "string.h"
#include "stdarg.h"
#include "VGAFON.I"

#define STEP  10

#define HOME         0x4700
#define CURSOR_UP    0X4800
#define PAGE_UP      0X4900
#define CURSOR_LEFT  0X4b00
#define Shift_LEFT   0X7300
#define CURSOR_RIGHT 0X4d00
#define Shift_RIGHT  0X7400
#define END          0X4F00
#define CURSOR_DOWN  0X5000
#define PAGE_DOWN    0X5100
#define RETURN       0x1c0d

#define RGB_RED   0
#define RGB_GREEN 1
#define RGB_BLUE  2
#define RGB_SIZE  3
```

```
#define pixels2bytes(n)  ((n+7)/8)
#define fgetw(fp) (fgetc(fp)+(fgetc(fp)<<8))

#define greyvalue(r,g,b)(((r*30)/100) + ((g*59)/100) + ((b*11)/100))

unsigned char far *kbd_head = (unsigned char far *) 0x41AL;
unsigned char far *kbd_tail = (unsigned char far *) 0x41CL;

#define clearkb()               \
        {                       \
          *kbd_head  =  *kbd_tail; \
        }
int width,depth,bytes,bits;
int n;
int background = 0;
unsigned char Data[400];
unsigned char Data0[400];
unsigned char Data1[400];
unsigned char Datap[400];
unsigned char Datau[400];
unsigned char Datas[400];
int Xd[21];
int Xa[21];
int Xd1[21];
int Xa1[21];
int x1_data;
int x2_data;
int count;
int ax1,ax2,ay;
int x,xx,y;
char *name;
char da[10];

unsigned char palette[768];

#pragma option -a-
typedef struct {
                unsigned char identsize;
```

```
                unsigned char colourmaptype;
                unsigned char imagetype;
                unsigned int colourmapstart;
                unsigned int colourmaplength;
                unsigned char colourmapbits;
                unsigned int xstart,ystart;
                unsigned int width,depth;
                unsigned char bits;
                unsigned char descriptor;
                } TGAHEAD;

TGAHEAD tga;

/*************** function prototype ***************/
char *farPtr(char *p,long l);
char *_fastcall getline(int n,int width);
char *_fastcall mono2vga(char *p,int width);
char *_fastcall rgb2vga(char *p,int width);
void _fastcall putline(char *p,unsigned int n);
int _fastcall GetKey(void);
void error(char *s);
void information(void);
void freebuffer(void);
void  write_char_gen( int ch, int n, char far *map );
void  restore_char_gen( void );
void  box( int x1, int y1, int x2, int  y2 );

char masktable[8] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};
char *buffer=NULL;
char path[81];

#include "c:svga1.c"

/*=========================================================================
  menu
=========================================================================*/
void  menu()
```

```
{
 int  i;

 clrscr();
 write_char_gen(128,128,(char far *)font);
 textbackground( BLUE );
 textcolor( LIGHTGRAY );
 box( 1, 1, 80, 25 );
 window( 2, 2, 79, 24 );
 clrscr();

 textcolor( YELLOW );
 cputs("                                    \n\r");
 cputs("                          ภาควิชาวิศวกรรมเคมี \n\r" );
 cputs("                                    \n\r");
 cputs("                          จุฬาลงกรณ์มหาวิทยาลัย\n\r");
 cputs("                                    \n\r");
 cputs("              ระบบเก็บข้อมูลสำหรับสร้างภาพโทโมกราฟด้วยเทคนิคโทรทัศน์ \n\r" );
 cputs("                                    \n\r");

 textcolor( LIGHTRED );
 cputs( "                          Version 1.0 \n\r");
 cputs( "                          30 มกราคม 2537  \n\r" );
 cputs("\n\r\n\r");
 textcolor( LIGHTCYAN );

 cputs( "                                    \n\r" );
 cputs( " จำนวนโพรไฟล์เทากับ " ); scanf("%d",&n);cputs("\r\n");
 cputs( "                                    \n\r" );
 cputs( " โพรไฟล์แรกคือ " ); scanf("%s",name);cputs("\r\n");
 cputs( "                                    \n\r" );
 cputs( " ระบบข้อมูลไว้ในแฟ้มงาน " );scanf("%s",de);
 cputs( "                                    \n\r" );


 textcolor(LIGHTMAGENTA|BLINK);
 cputs("\n\r  Press Key Continue \n\r");
 while ( kbhit() ) getch();          /* gets a key stroke     */
```

```
    while ( kgetch() );
    textattr( LIGHTGRAY );
    clrscr();

    /* restores old character generator from ROM */
    restore_char_gen();
}

/*=========================================================================
  write_char_gen()
    changes character generator for 'n' characters starting
    from the character 'ch' to 'map'
=========================================================================*/
void  write_char_gen( int ch, int n, char far *map )
{
  _CX = n;                    /* 'n' characters              */
  _DX = (unsigned char)ch;     /* start with character 'ch'   */
  _BX = 0x1000;                /* 16 bytes per character
                              using map number 0            */
  _ES = FP_SEG( map );         /* segment of new char gen     */
  _BP = FP_OFF( map );         /* offset of new char gen      */
  _AX = 0x1100;                /* func. 11h - load char gen
                                        sub func. 0 - user defined  */
  geninterrupt( 0x10 );        /* BIOS call                   */
}

/*=========================================================================
  restore_char_gen()
    restores old character generator from ROM
=========================================================================*/
void  restore_char_gen()
{
  _BL = 0;                    /* using map number 0          */
  _AX = 0x0002;                /* func. 0 - set mode
                                        AL = text mode 2            */
  geninterrupt( 0x10 );        /* BIOS call                   */
}
```

```
/*=============================================================
  box()
    draws a box
=============================================================*/
void  box( int x1, int y1, int x2, int  y2 )
{
#define  S_VERT       149
#define  S_TOP_LEFT   220
#define  S_TOP_RIGHT  150
#define  S_BOT_LEFT   221
#define  S_BOT_RIGHT  151
#define  S_HORIZ      222

  int  x, y;
  struct text_info  t;

  gotoxy( x1, y1 );                  /* top          */
  putch( S_TOP_LEFT );
  for ( x=x1+1; x<x2; x++ )
    putch( S_HORIZ );
  putch( S_TOP_RIGHT );

  for ( y=y1+1; y<y2; y++ ) {        /* left & right */
    gotoxy( x1, y );
    putch( S_VERT );
    gotoxy( x2, y );
    putch( S_VERT );
  }

  gotoxy( x1, y2 );                  /* bottom       */
  putch( S_BOT_LEFT );
  for ( x=x1+1; x<x2; x++ )
    putch( S_HORIZ );

  /* outputs directly to Video RAM to prevent scrolling */
  gettextinfo( &t );
  poke( ( ( peekb( 0x0040, 0x0049 ) != 7 ) ?
            0xB800 : 0xB000 ), (y2-1)*160+((x2-1)<<1),
```

```
            S_BOT_RIGHT | ( tattribute << 8 ) );
}


void error(char *s)
{
    closegraph256();
    freebuffer();
    puts(s);
    exit(1);
}

char *farPtr(char *p,long l)
{
    unsigned int seg,off;

    seg = FP_SEG(p);

    off = FP_OFF(p);
    seg += (off / 16);
    off &= 0x000f;
    off += (unsigned int)(l & 0x000fL);
    seg += (unsigned)(l / 16L);
    p = MK_FP(seg,off);
    return(p);
}

void _fastcall putline(char *p,unsigned int n)
{
    if (depth)
    memcpy(farPtr(buffer,(long)n*(long)width),p,width);
}

char *_fastcall getline(int n,int width)
{
    return(farPtr(buffer,(long)n*(long)width));
}
```

```
#pragma warn -par
int getbuffer(unsigned long n,int bytes, int lines)
{
    if((buffer = farmalloc(n)) == NULL) return(0);
    else return(1);
}
void freebuffer(void)
{
    if(buffer != NULL) farfree(buffer);
    buffer = NULL;
}

void _fastcall reverse(char *p,TGAHEAD *tga)
{
    char *pr;
    int i;

    if (!(tga->descriptor & 0x10)) return;
    if ((pr=malloc(tga->width)) != NULL)
    {
        for (i=0;i<tga->width;++i) pr[i] = p[tga->width-1-i];
        memcpy(p,pr,tga->width);
        free(pr);
    }
}

/* read one line in the appropriate mode */

int readtgaline(char *p,FILE *fp,TGAHEAD *tga)
{
    int c,i,n=0,size;
    int r,g,b,linesize;

    if (tga->bits == 1) linesize = pixels2bytes(tga->width);
    else linesize = tga -> width;

    if (tga -> imagetype == 0x01 ||
        tga -> imagetype == 0x02 ||
```

```
        tga -> imagetype == 0x03)
    {
     switch(tga->bits)
     {
      case 1  : fread(p,1,pixels2bytes(tga->width),fp);break;
      case 8  : fread(p,1,tga->width,fp);break;
      case 16 : for(i=0; i<tga->width;i++)
                 {
                  o = fgetw(fp);
                  r = ((o>>10) & 0x1f) << 3;
                  g = ((o>> 5) & 0x1f) << 3;
                  b = (o & 0x1f) << 3;
                  *p++ = r;
                  *p++ = g;
                  *p++ = b;
                 }break;
      case 24 : for (i=0;i<tga->width;++i)
                 {
                  b = fgeto(fp);
                  g = fgeto(fp);
                  r = fgeto(fp);
                  *p++ = r;
                  *p++ = g;
                  *p++ = b;
                 }break;
      case 32 : for(i=0;i<tga->width;++i)
                 {
                  b = fgeto(fp);
                  g = fgeto(fp);
                  r = fgeto(fp);
                  fgeto(fp);
                  *p++ = r;
                  *p++ = g;
                  *p++ = b;
                 }break;
     }
    }
    else
```

```
{
 do {
        o = fgeto(fp);
        size = (o & 0x7F)+1;
        n+= size;
        if (o & 0x80)
        {
         switch(tga->bits)
         {
          case 1 :
          case 8 : o = fgeto(fp);
                   for (i=0;i<size;i++) *p++ = o;
                   break;
          case 16: o = fgetw(fp);
                   r = ((o>>10) & 0x1F) << 3;
                   g = ((o >>5) & 0x1F) << 3;
                   b = (o & 0x1f) << 3;
                   for(i = 0 ;i<size;i++)
                   {
                    *p++ = r;
                    *p++ = g;
                    *p++ = b;
                   }break;
        case 24: b = fgeto(fp);
                   g = fgeto(fp);
                   r = fgeto(fp);
                   for (i=0;i<size;i++)
                   {
                    *p++ = r;
                    *p++ = g;
                    *p++ = b;
                   }break;
        case 32: b = fgeto(fp);
                   g = fgeto(fp);
                   r = fgeto(fp);
                   fgeto(fp);
                   for(i=0;i<size;i++)
                   {
```

```
                *p++ = r;
                *p++ = g;
                *p++ = b;
               }break;
 }
}
else
{
 switch(tga->bits)
 {
  case 1 :
  case 8 : for (i=0;i<size;i++) *p++ = fgetc(fp);
               break;
  case 16: for (i=0;i<size;i++)
               {
                c = fgetw(fp);
                r = ((c>>10) & 0x1F)<<3;
                g = ((c>>5)  & 0x1F)<<3;
                b = (c & 0x1F) << 3;
                *p++ = r;
                *p++ = g;
                *p++ = b;
               }break;
  case 24: for (i=0;i<size;i++)
               {
                b = fgetc(fp);
                g = fgetc(fp);
                r = fgetc(fp);
                *p++ = r;
                *p++ = g;
                *p++ = b;
               }break;
  case 32: for (i=0;i<size;++i)
               {
                b = fgetc(fp);
                g = fgetc(fp);
                r = fgetc(fp);
                *p++ = r;
```

```
                    *p++ = g;
                    *p++ = b;
                   }break;
            }
           }
          } while (n<linesize);
    }
    return(ferror(fp));
}

/* convert a monochrome line in to en eight bit line */

char *_fastcall mono2vga(char *p,int width)
{
    char *pr;
    register int i;

    if ((pr=malloc(width)) != NULL)
    {
     for(i=0;i<width;++i)
     {
         if (p[i>>3] & masktable[i & 0x0007])
           pr[i] = 1;
         else
           pr[i] = 0;
     }
     return(pr);
    }else return(NULL);
}

/* convert en RGB line into en eight bit line */

char * _fastcall rgb2vga(char *p,int width)
{
 char *pr;
 register int i;

 if ((pr=malloc(width)) != NULL)
```

```
  {
   for (i=0;i<width;++i)
   {
    pr[i] = greyvalue(p[RGB_RED],p[RGB_GREEN],p[RGB_BLUE]);
    p += RGB_SIZE;
   }
   return(pr);
  }else return(NULL);
}

int _fastcall GetKey()
{
    register int c;
    c = getch();
    if (!(c & 0x00ff)) c = getch() << 8;
    return(c);
}

void strmfe(char *newc,char *oldc,char *ext)
{
    while (*oldc != 0 && *oldc != '.') *newc++ = *oldc++;
    *newc++='.';
    while(*ext) *newc++ = *ext++;
    *newc = 0;
}

void information(void)
{
    locate(63,12);
    gprintf(250,"X Start: %u",tga.xstart);
    locate(63,13);
    gprintf(250,"Y Start: %u",tga.ystart);
    locate(63,14);
    gprintf(250,"Image Width: %u",tga.width);
    locate(63,15);
    gprintf(250,"Image Depth: %u",tga.depth);
    locate(23,26);
    gprintf(230,"ReadTGA 1.0");
```

```
    locate(30-(7+strlen(path)/2),27);
    gprintf(250,"File : %s",path);
}

void unpacktga(FILE *fp)
{
    char *p,*pr;
    register int i;
    int n,startline,endline,inoline;

    memcpy(palette,"\000\000\000\377\377\377",6);
    /* get the header */

    if (fread((char *)&tga,1,sizeof(TGAHEAD),fp) != sizeof(TGAHEAD))
        error("Error reading TGA header");

    if (tga.imagetype==0x01 || tga.imagetype==0x02 ||
        tga.imagetype==0x03 || tga.imagetype==0x09 ||
        tga.imagetype==0x0a || tga.imagetype==0x0b)
        {
        /* set the details */
        width = tga.width;
        depth = tga.depth;
        bits  = tga.bits;
        /* work out the line width */
        if (bits==1) bytes = pixels2bytes(width);
        else if (bits == 8) bytes = width;
            else bytes = width*3;

        /* set a default monochrome palette */
        memcpy(palette,"\000\000\000\377\377\377",6);
        fseek(fp,(long)tga.identsize,SEEK_CUR);

        /* get the palette */
        if (tga.colourmaptype)
        {
        switch(tga.colourmapbits)
        {
```

```
  case 24 : for(i=tga.colourmapstart; i<tga.colourmaplength;++i)
            {
             if(i>= 768) break;
             palette[i*RGB_SIZE+RGB_BLUE]  = fgetc(fp);
             palette[i*RGB_SIZE+RGB_GREEN] = fgetc(fp);
             palette[i*RGB_SIZE+RGB_RED]   = fgetc(fp);
            }
            break;
  case 16 : for (i=tga.colourmapstart;i<tga.colourmaplength;++i)
            {
             if(i>=768) break;
             n = fgetw(fp);
             palette[i*RGB_SIZE+RGB_BLUE]  = ((n>>10) & 0x1f)<<3;
             palette[i*RGB_SIZE+RGB_GREEN] = ((n>>5) & 0x1f) << 3;
             palette[i*RGB_SIZE+RGB_RED]   = (n & 0x1f) << 3;
            }
            break;
 }
}
if (bits > 8)
{
 for (i=0;i<256;i++)
  memset(palette+(i*RGB_SIZE),i,RGB_SIZE);
}
else if (bits==1) memcpy(palette,"\000\000\000\377\377\377",6);
/* image buffer */
if (!getbuffer((long)width*(long)depth,width,depth))
 error("Can't allocate image buffer");
/* allocate a line buffer */
if ((p=malloc(bytes))==NULL)
   error("Can't allocat line buffer");
if (!(tga.descriptor & 0x20))
{
 startline = depth-1;
 endline   = -1;
 incline   = -1;
}
else {
```

```
        startline = 0;
        endline   = depth;
        inoline   = 1;
    }
/* read all the line */
for (i=startline;i != endline;i+=inoline)
{
 if (readtgaline(p,fp,&tga))
 {
  free(p);
  freebuffer();
  error("Read Error !");
 }
/* translate the line type into VGA */
switch(bits)
{
 case 1  : pr = mono2vga(p,width);
           if(pr != NULL)
           {
            reverse(pr,&tga);
                putline(pr,i);
                free(pr);
           }
           else
           {
            free(p);
            error("Memory Error !");
           }break;
 case 8  : reverse(p,&tga);
           putline(p,i);
           break;
 case 16:
 case 24:
 case 32: pr = rgb2vga(p,width);
          if(pr != NULL)
          {
           reverse(pr,&tga);
           putline(pr,i);
```

```
                    free(pr);
                   }
                   else
                   {
                    free(p);
                    error("Memory Error");
                   }break;
              }

              }
              free(p);
              showimage();
             }else error("Memory Error !");
}

int Data_s(int y)
{
    int x;
    for(x=320;x<=650;x++)
    {
      Data0[x] = (int)getpixel(x,y);
      putpixel(x,y,250);
      Datap[x] = (int)getpixel(x,y-4);
      Datau[x] = (int)getpixel(x,y+4);
      Datas[x] = ((Data0[x]+Datap[x]+Datau[x]))/3);
    }
    return(y);
}

int Line_up(int y)
{
   int x;
   for(x=320;x<=650;x++)
   {
     Data1[x] = (int)getpixel(x,y+4);
     putpixel(x,y,Data0[x]);
     Data0[x] = Data1[x];
   }
```

```
    return(y);
}

int Line_Down(int y)
{
    int x;
    for(x=320;x<=650;x++)
    {
      Data1[x] = (int)getpixel(x,y-4);
      putpixel(x,y,Data0[x]);
      Data0[x] = Data1[x];
    }
    return(y);
}



void select_x(int x1,int x2,int y)
{
     locate(x1,y);
     gprintf(250,"Ü");
     locate(x2,y);
     gprintf(250,"Ü");

}

void select_xdx(int x1,int y)
{
     locate(x1,y);
     gprintf(0," ");

}

void select_xdxx(int x2,int y)
{
     locate(x2,y);
     gprintf(0," ");

}
```

```
void open_pic()
{
    strmfe(path,name,"TGA");
    strupr(path);
}

void save_data(int x1,int x2)
{
    int i;
    FILE *fpo;
    if((fpo = fopen(da,"wb"))==NULL)
    {
        puts("cannot open file outfile\n");
        exit(1);
    }
    for(i=x1;i<=x2;i++)
    {
        fprintf(fpo,"%d\r\n",Data[i]);
    }
    fclose(fpo);
}

void save_datacon(int x1,int x2)
{
    int i;
    FILE *fpo;
    if((fpo=fopen(da,"ab"))==NULL)
    {
      puts("cannot open file outfile\n");
      exit(1);
    }
    for(i=x1;i<=x2;i++)
    {
        fprintf(fpo,"%d\r\n",Data[i]);
    }
    fclose(fpo);
}
```

```
void run_con(int stn,int j)
{
    register int k;
    int i;
    FILE *fp;
    char c[4];
    char a[10];
    for(i=2;i<=stn;i++)
    {
     strcpy(a,"pic");
     itoa(i,c,10);
     strcat(a,c);
     name = a;
     open_pic();
     if((fp = fopen(path,"rb")) != NULL)
     {
      unpacktga(fp);
     /*
      k = GetKey();
      clearkb(); */
      Data_a(j);
      fclose(fp);
      save_datacon(x1_data,x2_data);
      closegraph256();

     }
    }
    clrscr();
}

void show_locate1(int x1)
{
    locate(14,23);
    gprintf(250,"     ");
    locate(10,23);
    gprintf(250,"X1 =  %d",x1);
}
```

```
void show_locate2(int x2)
{
    locate(28,23);
    gprintf(250,"    ");
    locate(25,23);
    gprintf(250,"X2 =  %d",x2);
}

void show_locate3(int y)
{
    locate(48,23);
    gprintf(250,"       ");
    locate(45,23);
    gprintf(250,"Y =  %d",y);
}

void w_config(int x1,int x2,int y)
{
    FILE *fo;
    if((fo = fopen("otoon.sys","w")) == NULL)
    {
      locate(55,24);
      gprintf(255,"Error in open file\n");
      exit(1);
    }
    fprintf(fo,"%d %d %d\n",x1,x2,y);
    printf("\007");
    fclose(fo);
}

void r_config()
{
    FILE *fo;
    if((fo = fopen("otoon.sys","r")) == NULL)
    {
      printf("Error in open file\n");
```

```
        exit(1);
    }
    fscanf(fo,"%d %d %d\n",&ax1,&ax2,&ay);
    printf("\007");
    fclose(fo);
}

void save_config(int x1,int x2,int y)
{
    char ch;
    locate(55,23);
    gprintf(255,"SAVE SELECT DATA (Y/N)");
    ch = getche();
    if (ch == 'y')
    {
        w_config(x1,x2,y);
    }
}

void load_config()
{
    char ch;
    locate(55,21);
    gprintf(255,"LOAD SELECT DATA (Y/N)");
    ch = getche();
    if (ch == 'y')
    {
        r_config();
        y = ay;
        x = ax1;
        xx = ax2;
    }
    else
    {
        y = 250;
        x = 10;
        xx = 20;
```

```
    }
}

void select()
{
    int yx;
  /*  int x1_data;
    int x2_data; */
    register int c;
    yx = 15;
    load_config();
    Data_s(y);
    do{
      `
          x1_data = 350+(x-3)*7.7;
          x2_data = 350+(xx-3)*7.7;
          count = x2_data-x1_data+1;
          select_x(x,xx,yx);
          c = GetKey();
          clearkb();
          switch(c)
          {
            case CURSOR_UP   : Line_Down(y);
                               if ((y/4) < 1)
                               {
                                    sound(100);
                                    delay(50);
                                    sound(1000);
                                    delay(50);
                                    nosound();
                                    y = 6;
                               }
                               y = y-4;
                               show_looste3(y/4);
                               Data_s(y);
                               break;

          case CURSOR_DOWN : Line_up(y);
```

```
                    y = y+4;
                    show_locate3(y/4);
                    Data_s(y);
                    break;

case CURSOR_LEFT : select_xdx(x,yx);
                    if (x<1)
                    {
                        x = 1;
                    }
                    x = x-1;
                    show_locate1(x);
                    break;

case CURSOR_RIGHT: select_xdx(x,yx);
                    if (x>xx-2)
                    {
                     sound(100);
                     delay(50);
                     sound(1000);
                     delay(50);
                     nosound();
                     x = x-1;
                     }
                    x = x+1;
                    show_locate1(x);
                    break;

case Shift_LEFT  : select_xdxd(xx,yx);
                    xx = xx-1;
                    show_locate2(xx);
                    break;

case Shift_RIGHT : select_xdxd(xx,yx);
                    xx = xx+1;
                    show_locate2(xx);
                    break;
```

```
            case HOME       : save_config(x,xx,y);
                              closegraph256();
                              save_data(x1_data,x2_data);
                              run_con(n,y);
                              break;
          }
       }while(c != 27);

}

void show_end()
{
      clrscr();
      window(15,12,65,15);
      textbackground(1);
      clrscr();
      gotoxy(16,2);
      cprintf(" Number of Data = %d",count);
}

void cdecl main()
{

 register int c;
 FILE *fp;

  menu();
  open_pic();

  if((fp = fopen(path,"rb")) != NULL)
  {
   unpacktga(fp);
   fclose(fp);

   select();
   c = GetKey();
   clearkb();
```

```
    }else printf("Error opening %s",path);
    clrscr();
    gotoxy(15,10);
    printf("count = %d\n",count);

    show_end();
    getch();
}
```

```
/*******************************/
/*  SVGA.C                     */
/*******************************/
#include "string.h"
#include "stdio.h"
#include "dos.h"
#include "alloc.h"
#include "conio.h"
#include "dos.h"

#define VGA      0
#define TSENG    1
#define TRIDENT  2
#define NOTVGA   3

int screenmode = VGA;
int screenwidth;
int screendepth;
unsigned long screentable[480];
unsigned char banknumber = 0;

void _fastcall slvga(char *p,int y,int size);
void _fastcall sltseng(char *p,int y,int size);
void _fastcall sltrident(char *p,int y,int size);
void _fastcall (*setline)(char *p,int y,int size);

void setvgapalette(unsigned char *p,int n)
{
    register int i;
    outportb(0x3c6,0xff);
    for (i=0;i<n;++i)
    {
        outportb(0x3c8, (unsigned char)i);
        outportb(0x3c9, (*p++)>>2);
        outportb(0x3c9, (*p++)>>2);
        outportb(0x3c9, (*p++)>>2);
    }
}
```

```
void closegraph256(void)
{
    _AX = 0x0003;
    geninterrupt(0x10);
}

int getmode(void)
{
   return(peekb(0,0x449));
}

void putpixel(int x,int y,int c)
{
    long i;
    i=((long)y*480+x);
    outp(0x3cd,0x40|(i>>16));
    pokeb(0xa000,i&0xffff,c);
}

unsigned int getpixel(int x,int y)
{
    long i;
    register unsigned int isf;
            i = ((long)y*480+(long)x);
            isf = (unsigned int)(((i>>16)&0x000f)<<4);
            outp(0x3cd,isf);
            return(unsigned int)(peekb(0xa000,(unsigned int)i&0xffffU));
}

void line(int x1,int y1,int x2,int y2,int color)
{
    register int a;
    register int ydif = y2-y1;
    register int xdif = x2-x1;
    long i;

    if(abs(ydif) > abs(xdif))
    {
```

```
    if (ydif == 0) return;
    if (y2 > y1)
  for(a = y1;a<= y2;a++)
    {
        i = ((long)a*480+x1+(int)((((long)xdif*(long)(a-y1)))/ydif));
        outp(0x3cd,(int)(0x40|(i>>16)));
        pokeb(0xa000,i&0xffff,color);
    }
    else
    for(a = y1;a >= y2;a--)
        {
         i=((long)a*480+x1+(int)((((long)xdif*(long)(a-y1)))/ydif));
         outp(0x3cd,0x40|(i>>16));
         pokeb(0xa000,i&0xffff,color);
        }
  }
  else
  {
    if (xdif == 0) return;
    if (x2 > x1)
    for(a = x1;a<=x2;a++)
    {
        i = ((long)(y1+(int)((((long)ydif*(long)(a-x1)))/xdif))*480+a);
        outp(0x3cd,0x40|(i>>16));
        pokeb(0xa000,i&0xffff,color);
    }
  }
}

void opengraph256(void)
{
    unsigned int i;

    _AX = 0X002e;
    geninterrupt(0x10);
    if (getmode() == 0x2e) screenmode = TSENG;
    else
    {
```

```
        _AX = 0x005d;
        geninterrupt(0x10);
        if (getmode() == 0x5d) screenmode = TRIDENT;
        else
        {
          _AH = 0x0013;
          geninterrupt(0x10);
          if (getmode() == 0x13) screenmode = VGA;
          else screenmode = NOTVGA;

        }
}
switch (screenmode)
{
 case VGA : screenwidth = 320;
                screendepth = 200;
                for (i=0;i<screendepth;i++)
                    screentable[i] = (unsigned long)(i*screenwidth);
                setline = slvga;
                break;
case TRIDENT : screenwidth = 640;
                    screendepth = 480;
                    for (i=0;i<screendepth;++i)
                    {
                    screentable[i] = (long)screenwidth*(long)i;
                    if ((unsigned int)screentable[i] > 0xfffe-(unsigned int)screenwidth)
                         screentable[i] |= 0x80000000L;
                    }
                    setline = sltrident;
                    break;
    case TSENG  : screenwidth = 640;
                    screendepth = 480;

                    for (i=0;i<screendepth;++i)
                    {
                    screentable[i] = (long)screenwidth*(long)i;
                    if ((unsigned int)screentable[i] > 0xfffe-(unsigned int)screenwidth)
                         screentable[i] |= 0x80000000L;
```

```
                    }
                    setline = sltseng;
                    break;
            case NOTVGA : error("Not a VGA Card, This program not support !");
                    closegraph256();
                    break;
    }

}

void _fastcall tsengbank(unsigned int n)
{
    if (n==banknumber) return;
    banknumber = n;
    disable();
    outportb(0x3bf,0x03);
    outportb(0x3d8,0xa0);
    outportb(0x3cd,((n & 0x0f) << 4) | n);
    enable();
}

void _fastcall tridentbank(unsigned int n)
{
    if (n==banknumber) return;
    banknumber = n;
    disable();
    outportb(0x3ce,6);
    outportb(0x3cf,((inportb(0x3cf)|4)<<0)|6);
    outportb(0x3c4,0x0b);
    inportb(0x3c4);
    outport(0x03c4,((n^2)<<8)|0x0e);
    enable();
}

void _fastcall slvga(char *p,int y,int size)
{
    memcpy(MK_FP(0xa000,screentable[y]),p,size);
}
```

```
void _fastcall sltseng(char *p,int n,int size)
{
    register int i;
    unsigned long l;

    l = screentable[n];
    tsengbank((unsigned int)(l>>16));
    if (l & 0x80000000L)
    {
      i = (unsigned int)(0x10000L-(l & 0xffffL));
      if (size > i)
      {
            memcpy(MK_FP(0xa000,(unsigned int)l),p,i);
            tsengbank(banknumber+1);
            memcpy(MK_FP(0xa000,0),p+i,size-i);
      }
      else memcpy(MK_FP(0xa000,(unsigned int)l),p,size);
    }
    else memcpy(MK_FP(0xa000,(unsigned int)l),p,size);
}

void _fastcall sltrident(char *p,int n,int size)
{
    register int i;
    unsigned long l;
    l = screentable[n];
    tridentbank((unsigned int)(l>>16));
    if (l & 0x80000000L)
    {
        i = (unsigned int)(0x10000L-(l & 0xffffL));
        if (size > i)
        {
          memcpy(MK_FP(0xa000,(unsigned int)l),p,i);
          tridentbank(banknumber+1);
          memcpy(MK_FP(0xa000,0),p+i,size-i);
        }
        else memcpy(MK_FP(0xa000,(unsigned int)l),p,size);
    }
```

```
    else memcpy(MK_FP(0xa000,(unsigned int)ll),p,size);
}

void locate(int x,int y)
{
    _AH = 2;
    _BH = 0;
    _DH = y;
    _DL = x;
    geninterrupt(0x10);
}

int cdecl gprintf(char color,char *fmt, ...)
{
    va_list argptr;
    char str[140];
    int cnt, i;
        va_start(argptr,fmt);
        cnt = vsprintf(str,fmt,argptr);
        for (i=0;i<cnt;i++)
        {
          _AL = str[i];
          _AH = 0x0e;
          _BL = color;
          _BH = 0;
          geninterrupt(0x10);
        }
        va_end(argptr);
        return(cnt);
}

void showimage(void)
{
    register int i,x=0,y=0,c,n;
    int monitorwidth = 500,monitordepth = 400;
    opengraph256();
    setvgapalette(palette,256);
    if (screenmode == VGA)
```

```
    {
      monitorwidth = screenwidth-1;
      monitordepth = screendepth-11;
      locate(0,24);
    }
    else
    {
    information();
    locate(10,25);
    }
    gprintf(200,"Use Home,End or %c %c %c %c to scroll image",24,24,26,27);
    if (width > monitorwidth) n = monitorwidth;
    else n = width;
   /* do { */
          for (i=0;i<=monitordepth;i++)
          {
           o = y + i;
              if (o >= depth) break;
              (*setline)(getline(o,width)+x,i,n);
          }

         freebuffer();

}
```

ประวัติผู้เขียน

นายธีรวัฒน์ ประกอบผล เกิดเมื่อวันที่ 4 พฤษภาคม พ.ศ. 2511 ที่อำเภอพระสมุทรเจดีย์ จังหวัดสมุทรปราการ จบการศึกษาปริญญาตรีจาก ภาควิชาฟิสิกส์ประยุกต์ (โซลิดเสตตอิเลคทรอนิค) คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปี พ.ศ. 2533 จากนั้นเข้าศึกษาต่อที่ ภาควิชานิวเคลียร์เทคโนโลยี คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ. 2534