LONG RANGE TARGETING FOR SPACE BASED RENDEZVOUS

A FINAL REPORT
FOR GRANT
NAG-9-566

Prepared for: Leo Monford, NASA-JSC

by
L. J. Everett
and
R. C. Redfield
Texas A&M University

March 20, 1995

ABSTRACT

The work performed under this grant supported the Dexterous Flight Experiment on STS-62. The project required developing hardware and software for automating a TRAC sensor on orbit. The hardware developed for the flight has been documented through standard NASA channels since it had to pass safety, environmental, and other issues. The software has not been documented previously therefore this report provides a software manual for the TRAC code developed for the grant.

AutoTrac Analysis Tools V1.06
SOFTWARE MANUAL

by

Louis J. Everett
Mechanical Engineering
Texas A&M University
College Station, Texas 77843-3123
409-845-9591
LJE0106@Sigma.Tamu.Edu
or
LOU@Robotics.Tamu.Edu

March 13, 1995

Contents

Chapter 1

INTRODUCTION

If you notice any problems with this documentation, if you would like something explained better, or if something else needs to be indexed, contact the author.

This documentation uses the following conventions.

1. Routines with names like PutItInBlob use capital letters to indicate the pronunciation. In the index you will find the routine by the string "Put it in blob".

2. Not all routines were intended to be called by a user. These routines usually perform some esoteric type of calculation that is not generalizable. Routines of this type are called internal and are kept in a file along with the functions that call them. Their prototypes are normally found at the top of the .c file containing the code. If you want to know who calls an internal function just search the file containing the internal function, no one else can call it.

3. This document is organized by .c file. All routines in one particular .c file are grouped together in a chapter. The user callable routines are described first followed by the ones used internal to the library. Internal routines cannot be used outside of the .c file they appear in. As a user you need not worry about internal routines.

4. To determine what functions are available to the user, look in the .h files. They contain the names and prototypes of all functions that were intended for someone to call. These .h files also provide some information about what the routines do. The .h file listings appear immediately after each chapter introduction.

5. The level of indentation in a listing generally indicates what loop, or if statement the line belongs to.

6. The user data structure definitions[1] are contained in the file datatype.h. This header is included by nearly all routines. Data structures not defined in datatype.h are internal[2] and can be found in the .c file that they are used in.

7. For the most part writing a backslash is difficult to format, so all directory names are given as: a/b/c although DOS requires the division symbols to be backslashes.

1.1 DATATYPE.H

Documentation Date: 3/12/95

Data Definitions and some prototypes of functions that must be written by the user. The following listing shows the user datatypes. Note that there are 5 routines that are called by the library. They are prototyped in this file. This file controls most of the dimensions of data.

New Data Types:

See the listing.

Definitions: See the listing.

Prototypes:

```
void Status_Message(char *message);
   void Fatal_Error_Message(char *message);
   void clear_status(void);
   void printandwait(char *message);
   void gprintandwait(char *message);
```

Description:

The library routines need to interact with the user occasionally. To do this the library calls five routines. YOU must provide these routines yourself. They are:

1. Status_Message - prints a message on the screen.

2. Fatal_Error_Message - This should print a message then exit. It is called when the library discovers a major problem.

3. clear_status - This should clear the status message.

4. printandwait - This should print a message then wait for the user to hit a key.

[1] User data refers to the fact that the user has access to the data at some point in the code not that the variables have global scope.

[2] Internal data means the user cannot access it.

5. gprintandwait - This is similar to printandwait except the message occurs when a graph is active so you may want the location of the text to differ.

1.1.1 Header File Listing:

```
#ifndef DATATYPE_H
  #define DATATYPE_H
  /*Your attention please....
The targa version of the trac code consists of several self contained
library routines. This file discusses all of these routines. The library
is self contained with 5 exceptions. You must supply five subroutines for
handling messages to the operator. If you do not want to see the messages,
you may make the routines return immediately, otherwise have them print the
messages. The five routines you must write are prototyped here.*/

  /* this prints a status message on the screen*/
   void Status_Message( char *message);
  /* this prints a status message on the screen when in graphics mode*/
   void gStatus_Message( char message[]);
  /* This should print a message then exit. It is called when the library
  discovers a major problem.*/
   void Fatal_Error_Message( char *message);
  /* This clears the status message*/
   void clear_status( void);
  /*graphics mode version*/
   void gclear_status( void);
  /* This will print a message then wait for the user to hit a key*/
   void printandwait( char *message);
  /*graphics mode version*/
   void gpromptandread( char message[], char format[], void *value);
  /* This is similar except the message occurs when a graph is active
so you may want the location of the text to differ*/
   void gprintandwait( char *message);
   void gprint( char *message);

  /* In the following, the data types are defined */
   /*First some definitions for the type of camera etc.*/
     #define MAX_LEDS 4
     #define IMAGEWIDTH 512
     #define IMAGEHEIGHT 512
     #define TOTALNUMBEROFPIXELS 250000
     #define PIXEL_MAX 255
     #define VISION_BOARD "targa8.h"

    /*Define the whole image*/
     #define LEFTMOSTCOLUMN 0
     #define RIGHTMOSTCOLUMN (IMAGEWIDTH-1)
```

```
#define TOPMOSTROW (IMAGEHEIGHT-2)
#define BOTTOMMOSTROW 26
#define WHOLE_IMAGE(roi) roi.xs = LEFTMOSTCOLUMN; roi.xe = RIGHTMOSTCOLUM
roi.ys = BOTTOMMOSTROW; roi.ye = TOPMOSTROW; roi.resolution = 1
/*It seems the lower 27 lines are ????*/
/*It seems there is a problem with the top line*/
/*
I think the problem is that the board is set to vertical 486 + the top
and bottom are only half lines do to the interlace
*/

#ifdef UPPER_LEFT_ORIGIN
  #undef UPPER_LEFT_ORIGIN
  #endif
/* the origin for the targa card is lower left*/
#define LOWER_LEFT_ORIGIN
#define BBLACK 0
#define WHITE 255
#define RED 0
#define GREEN 1
#define BLUE 2
#define SUCCESS 0
#define FAIL 1
#define FALSE 0
#define TRUE 1
/*If CHECKLENGTH is defined the code will check the length of strings*/
#define CHECKLENGTH 1

typedef unsigned char Pixel;
typedef int DPixel;
typedef Pixel ImageLine[IMAGEWIDTH];
typedef DPixel DImageLine[IMAGEWIDTH];
#define NUMBEROFPIXELBINS (PIXEL_MAX + 1)
#define FULLNUMBEROFPIXELBINS (2 * PIXEL_MAX + 1)
typedef long Histogram[NUMBEROFPIXELBINS];
typedef long FullHistogram[FULLNUMBEROFPIXELBINS];

typedef struct {
 double size;
 double radius;
 double variance;
 double centx;
 double centy;
 double gray;
}OneBlob;

typedef struct {
```

```
    int xs, ys, xc, yc, resolution;
   }ROI;

/* #define MAX_BLOBS_PER_PICTURE 10 changed for leo */
   #define MAX_BLOBS_PER_PICTURE 20
   typedef struct {
     int num_of_blobs_found;
    OneBlob blobs[MAX_BLOBS_PER_PICTURE];
    OneBlob background;
     double SignalToNoiseMargin;
    DPixel posthreshold, negthreshold;
    }Image;

   #define NUM_RETROS_PER_TARGET 5
   typedef struct {
    OneBlob led;
       int num_of_led;
    OneBlob OddRetro;
       int NumOfOdd;
    OneBlob leftretro, rightretro, topretro, bottomretro;
       int left,right,top,bottom;
   }Target;

/* typedef struct {
    int tx[6], ty[6], tz[6], tyaw[6], tpitch[6], troll[6];
    double r1, r2, lx, ly, p, y;
    } Transform; */

   typedef struct {
    double FocalLength, CameraAspect, TargetWidth, TargetHeight, RangeOffset;
   }Transform;

   #define BAD_PITCH_YAW 180
   typedef struct {
    double x, y, z, yaw, pitch, roll;
   }Pose;

   typedef struct {
    double mx, my, d, ledx, ledy, roll;
   }RawPose;

   typedef struct{
     int left, right, top, bottom, corner, NormalNumOfLeds, led[MAX_LEDS];
    }BlobIds;

   typedef struct {
    double element[2];
```

```
 }Vector;

typedef struct{
 BlobIds id;                                                    140
 }AllStats;

typedef struct {
 int PrinterPortAddress;
 int NumberOfLEDSToSeek;
 int LED_State;
 int Threshold;
 int LEDControlMode;
 int NumberInSlope;
 int ThresholdPercent;                                          150
 int MinimumBlobSize;
 long MaximumBlobSize;
 double Tolerance;
 int OverLayOnLive;
 int VideoCenterRow, VideoCenterCol;
 double FocusDistance;
 }Parameters;

#endif
                                                                160
```

1.2 MAIN Programs

A sample main program is provided in the following listing. The routine contains some user datatypes defined in "datatype.h". Compile the main program with a statement like:

CL /c /AL main.c

The main program is linked to the library with options: /CO (code view) /noi (no ignore case) /ST:37000 (stack size, sometimes it is 27000), /E (???). You must also include the commercial targa libraries targraf, ltplib, the microsoft graphics library graphics and targa.lib (the latter is described in this document).

link /CO /noi /ST:37000 /E main,,,,targraf,ltplib,graphics,lib/targa.lib

1.2.1 Program Listing:

```
#include <stdlib.h>
#include <stdio.h>
#include "targa.h"
#include "user.h"

void main( void);

void main ( void) {
Posc posc;
float rob;                                                    10
RawPosc rawposc;
AllStats allstats;
Target target;
int overlaypage,overlaymode;
ROI roi;
DPixel thresh;
int FirstBright,Dim,SecondBright,LcosSecondBright;
int firstfalling,resolution,numinslope,DrawPlot,thresholdpercent,storepage;
Image image;
long minsizeblob;                                             20
double tolerance;
int calibrate;
Transform CameraTransform;
int i,j;
FILE *file;

    calibrate = 0;


    Initialize_vision(0);                                     30
    i=1;
```

```
 file = fopen("data.dat","wt");
while(i){
 overlaypage=0;
 WHOLE_IMAGE(roi);
  if(inittext()!=SUCCESS)exit(1);
 ClearPage(overlaypage);
 DrawOverStrings(WHITE,overlaypage);
 Live_Video();
 GrabEightFrames();                                                    40
 /*liveoverlay(overlaypage);*/
 resolution = 2;
 numinslope = 3;
 thresh = 0;
 DrawPlot = 0;
 thresholdpercent = 50;
 tolerance=1.;
 numinslope=4;
 minsizeblob=50;
 LeosSecondBright=-1;                                                  50
 firstfalling = FindFirstFallingAndThresh(resolution,numinslope,&thresh,
                          DrawPlot,thresholdpercent);
 FindFramesForDoubleSubtract(firstfalling, &FirstBright,&Dim,&SecondBright);
 storepage=FindStorageImage(firstfalling);
 Show_Process_Image(storepage);
 DefineImage(FirstBright,Dim,LeosSecondBright,storepage,&image,
        TOTALNUMBEROFPIXELS,minsizeblob,thresh);
  if(!TargetCosys(&image,&allstats,tolerance)) {
      LoadTargetFromStats(&image,&target,&allstats);
      Live_Video();                                                    60
      Mark_Target (&target,storepage,(IMAGEWIDTH)/2,(IMAGEHEIGHT)/2);
      Show_Process_Image(storepage);
 printf("IS IT OK 1 is yes 0 no");
 scanf("%d",&j);
  if(j){
   printf("%f %f %f %f %f %f %f %f\n",target.leftretro.centx,target.leftretro.centy,
   target.rightretro.centx,target.rightretro.centy,
   target.topretro.centx,target.topretro.centy,
   target.bottomretro.centx,target.bottomretro.centy);
   fprintf(file,"%f %f %f %f %f %f %f %f\n",target.leftretro.centx,target.leftretro.centy,
   target.rightretro.centx,target.rightretro.centy,
   target.topretro.centx,target.topretro.centy,
   target.bottomretro.centx,target.bottomretro.centy);
   }
 CameraTransform.tx[0] = 1;
 CameraTransform.ty[1] = 1;
 CameraTransform.tz[2] = 1;
 CameraTransform.tyaw[3] = 1;
```

```
        CameraTransform.tpitch[4] = 1;
        CameraTransform.troll[5] = 1;
        CameraTransform.r1 = 1270097.;
        CameraTransform.r2 = 1.;
        CameraTransform.lx = 2.361E-4;
        CameraTransform.ly = -1.858E-4;
        CameraTransform.p = 1.927E-4;
        CameraTransform.y = 2.423E-4;
            GetPose(&target,&CameraTransform,&allstats.best.id,&rawpose,&pose);
            if(calibrate){
                rob=GetAFloat();
                printf("%f, %f, %f, %f, %f, %f, %f\n",rob,rawpose.mx,
                    rawpose.my,rawpose.d,rawpose.roll,rawpose.ledx,rawpose.ledy);
                    }
            PrintRawCamera(&rawpose);
            WritePose(&pose);
            DrawOverlay(&pose,overlaypage);
            Mark_Target(&target,overlaypage,(IMAGEWIDTH)/2,(IMAGEHEIGHT)/2);
            }
             else {
                 if(inittext()!=SUCCESS)exit(1);
                ClearPage(overlaypage);
                DrawOverStrings(WHITE,overlaypage);
                printandwait("No target found - Press any key");
                }
            Live_Video();
            ClearPage(overlaypage);
            printf("Continue 1 yes, 0 no");
            scanf("%d",&i);
            }
            fclose(file);
            finishtext();
            End_vision();
            }

 void Status_Message(message)
 char message[];
{ clear_status();
  _settextposition(24,12);
  _outtext(message);
}

 void Fatal_Error_Message(message)
 char message[];
{ Status_Message(message);
  exit(1);
  }
```

```
 void clear_status()
{
   int i;
   char str[80];                                                    130
   for (i=0;i<65;i++) str[i]=' ';
  str[65] = 0;
  _settextposition(24,12);
  _outtext(str);
}

 void gprintandwait(message)
 char message[];
{ _moveto(20,1);
  _outgtext(message);                                              140
  getch();
}

 void printandwait(message)
 char message[];
{ Status_Message(message);
  getch();
  clear_status();
}                                                                  150
```

1.3 Function: Auto TRAC Analysis Tool

Documentation Date: 3/9/95

Prototypes:

See program listing.
Source File: *atat1_06.c*
Type of Function: User Callable
Header Files Used in atat1_06.c: *<float.h> <stdio.h> <graph.h> <string.h> <conio.h> <time.h> <math.h> <dos.h> <ctype.h> "targa.h" "com3_2.h"*

Description:

This routine is the actual main routine.

1.3.1 Program Listing:

```
#include < float.h>
#include <stdio.h>
#include <graph.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include <dos.h>
#include <ctype.h>
#include "targa.h"
#include "com3_2.h"

#define FREE_RUN 0
#define COMPUTER_CONTROLLED 1
#define PRINTER_PORT 2

void main(Parameters Default);
void LogTargetResults(Target *target, FILE *file);
void Main_Menu(Parameters def);
void Case_Number( int *number);
int SetParameters(Parameters *paramters);
void DisplayParameters(Parameters *parameters);
int SaveParameterSet(Parameters *param);
int ReadParameterSet(Parameters *param);
int DoesFileExist( char *name);
int AnsweredYes( char *string);
int GetAFileName( char *filename, int len, char *ext);
void DoDirectory( char *extension, char *mess);
```

```
void OutColorText( char *string);
void DisplayFrameUses( int firstbright, int dim);
void ReadInt( int *n);
void ReadLong( long *n);
void ReadsixFloat( float *x, float *y, float *z, float *yaw, float *pitch, float
*roll);
void ReadDouble( double *n);
void ReadHex( int *n);
void DesignateCenter(Parameters *param, int page);
int DoComInitialize(ComPort *port);

void main( void)

{
  Parameters DefaultParameters;
    DefaultParameters.PrinterPortAddress = 0x378;
    DefaultParameters.NumberOfLEDSToSeek = 4;
    DefaultParameters.LED_State = 0xAA; //This goes with NumberOfLEDSToSeek don't
 change one without the other
    DefaultParameters.Threshold = 135;
    DefaultParameters.LEDControlMode = FREE_RUN; //There is a problem knowing how
 many leds to seek and the led_state and this
    DefaultParameters.NumberInSlope = 3;
    DefaultParameters.ThresholdPercent = 50;
    DefaultParameters.MinimumBlobSize = 25;
    DefaultParameters.MaximumBlobSize = TOTALNUMBEROFPIXELS;
    DefaultParameters.Tolerance = 3.;  //# of radii tolerance in template match
    DefaultParameters.OverLayOnLive = 0;
    DefaultParameters.VideoCenterRow = 245;
    DefaultParameters.VideoCenterCol = 255;
    DefaultParameters.FocusDistance = 626.6;
  _clearscreen(0);
  Initialize_vision(0);
  setprinterportaddress(DefaultParameters.PrinterPortAddress);
  if(inittext("d"))Fatal_Error_Message("Could not find targa font");
  Live_Video();
  GrabEightFrames();
  Main_Menu(DefaultParameters);
  }

void Case_Number(int *number)
{
 int ch;
   ch = getch();
   switch (ch){
   case '0':              *number=0;
          break;
```

```
    case '1':               *number=1;
            break;
    case '2':               *number=2;
            break;                                                    80
    case '3':               *number=3;
            break;
    case '4':               *number=4;
            break;
    case '5':               *number=5;
            break;
    case '6':               *number=6;
            break;
    case '7':               *number=7;
            break;                                                    90
    case '8':               *number=8;
            break;
    default:    //printandwait("Invalid entry:  default to 0 - Hit any key");
            *number=8;
    }//End switch
}//End Case_Number

void Main_Menu(Parameters DefaultParameters)
 {
 #define FILE_NAME_LENGTH 13                                          100
//float x,y,z,yaw,p,r;
 ComPort port;
 int status;
 int ch, page;
 int CalibrateMode;
 Transform trans;
   ROI roi;
   DPixel thresh, autothresh;
   double sum[4];
   double intensitymax, intensitymin;                                 110
   double xs, ys;
   Image image;
   AllStats allstats;
   RawPose raw;
   Pose pose;
   Target target;
   int showgraphics;
   int jj,i;
   char text[5];
   int frame1, frame2;                                                120
   int waitt, LED_state;
   int index;
   int firstbright, dim, subtracted, accumulated, workingforhist, storeforhist,
```

```
storefordefine, overlaypage, pageformarking;
   FILE *file;
   char filename[50];
   char afile[FILE_NAME_LENGTH];
   char mess[80];
   long numof255s;

   trans.CameraAspect = 1.266;
   trans.TargetWidth = 4.2;
   trans.TargetHeight = 3.;
   trans.RangeOffset = -.5756;

   thresh = DefaultParameters.Threshold;
   firstbright = 1;
   dim = 0;
   subtracted = 4;
   accumulated = 5;
   workingforhist = 6;
   storeforhist = 7;
   storefordefine = 6;
   overlaypage = 7;
   showgraphics = 0;
   CalibrateMode =0;
   LED_state=DefaultParameters.LED_State;

  sprintf(mess,"Choose from menu->");
do{
 _clearscreen(0);
  DisplayFrameUses(firstbright, dim);
_settextposition(2,1);
OutColorText("***** Auto\vGTRAC\vW Analysis Tools - Version 1.06 *****\n");
OutColorText("* Copyright December 27, 1994 Everett & Redfield *\n");
OutColorText("* \vGTRAC\vW is a Patented Sensor by NASA's \vGLeo Monford\vW*\n");
OutColorText("*************************************************\n");
OutColorText("* (\vGA\vW)ccumulate subtracted images:          *\n");
OutColorText("* (\vGB\vW)link LEDs:                            *\n");
if(CalibrateMode)
OutColorText("* (\vGC\vW)alibrate Disable:                     *\n");
else
OutColorText("* (\vGC\vW)alibrate Enable:                      *\n");
OutColorText("* (\vGD\vW)isplay any image:                     *\n");
OutColorText("* (\vGF\vW)ind objects in image:                 *\n");
OutColorText("* (\vGG\vW)rab and subtract:                     *\n");
OutColorText("* (\vGH\vW)istogram:                             *\n");
OutColorText("* (\vGI\vW)nput/Output of Images:                *\n");
OutColorText("* (\vGL\vW)ED control mode:                      *\n");
OutColorText("* (\vGN\vW)umber of active LEDs set:             *\n");
```

```
 OutColorText("* (\vGO\vW)utput Posc To Com Port:                    *\n");
 OutColorText("* (\vGP\vW)aramcter sct:                              *\n");
 OutColorText("* (\vGQ\vW)uit and return to DOS:                     *\n");
 OutColorText("* (\vGR\vW)OI statistics:                             *\n");
 OutColorText("* (\vGS\vW)ubtract two images:                        *\n");
 OutColorText("* (\vGV\vW)ideo livc:                                 *\n");
/* OutColorText("*************************************************\n");*/
 if(DefaultParameters.LEDControlMode==COMPUTER_CONTROLLED)
 OutColorText("*    \vRComputer Control LEDs Via Digital Port\vW     *\n");
 else if(DefaultParameters.LEDControlMode==PRINTER_PORT)
 OutColorText("*    \vRComputer Control LEDs Via Printer Port\vW     *\n");
 else
 OutColorText("*         \vRFree running LEDs Active\vW              *\n");
 OutColorText("*************************************************\n");
 DisplayParameters(&DefaultParameters);

    Status_Message(mess);
    ch = getch();
    switch (ch){
    WHOLE_IMAGE(roi);

   case 'a':              //Accumulate subtracted image
   case 'A':       index = 1;
           ClearPage(accumulated);    //accumulated page
           Ones(workingforhist);     //history test page to 1111
           do{
             sprintf(mess,"Accumulating Image: Loop %d - Hit any key to stop->",index)
             Status_Message(mess);
             index++;
             roi.resolution = 1;

           if(DefaultParameters.LEDControlMode==FREE_RUN){
              Charlotte_Grab(0,1,2,3);    //grab four images
              intensitymax=0;             //initial max and min
              intensitymin=1.e20;
              WHOLE_IMAGE(roi);
              roi.resolution=3;
              for (jj=0; jj<4; jj++){    //sort intensity
                Integrate_Image(jj,&sum[jj],roi);
                if (sum[jj]>intensitymax) {
                  intensitymax=sum[jj];
                  firstbright=jj;
                  }
               if (sum[jj]<intensitymin) {
                  intensitymin=sum[jj];
                  dim=jj;
               }
```

```
            }
            roi.resolution=1;
            }
         else if(DefaultParameters.LEDControlMode==COMPUTER_CONTROLLED){
            firstbright = 0;
            dim = 1;
            Grab_Frame (firstbright, LED_state);
            Grab_Frame (dim, LEDnone);
            WHOLE_IMAGE(roi);
            }
         else if(DefaultParameters.LEDControlMode==PRINTER_PORT){
            firstbright = 0;
            dim = 1;
            Grab_Frame_Printer_Port (firstbright, LED_state);
            Grab_Frame_Printer_Port (dim, LEDnone);
            WHOLE_IMAGE(roi);
            }
         else {
            Fatal_Error_Message("Invalid LED control mode");
            }

          Show_Process_Image(accumulated);
          numof255s = NewDoAccumulation(firstbright,dim,workingforhist,accumulated)
          }while(!kbhit());
          getch();
        sprintf(mess,"Accumulated image in page %d - Choose from menu->", accumulated
        break;
case 'b':               //Blink LEDs
case 'B':       Live_Video();
        if(DefaultParameters.LEDControlMode==FREE_RUN){
           sprintf(mess,"In Free Run Mode!! - Choose from menu->");
           Status_Message(mess);
           }
        else if(DefaultParameters.LEDControlMode==COMPUTER_CONTROLLED){
           sprintf(mess,"Blinking LEDs from digital port - Choose from menu->");
           Status_Message(mess);
           do{
             SetLEDState(LED_state);
             for(jj=0;jj<30000;jj++)waitt = (int)((double)jj * (double)jj);
             SetLEDState(LEDnone);
             for(jj=0;jj<30000;jj++)waitt= (int)((double)jj * (double)jj);
             }while(!kbhit());
           }
        else if(DefaultParameters.LEDControlMode==PRINTER_PORT){
           sprintf(mess,"Blinking LEDs from printer port - Choose from menu->");
           Status_Message(mess);
           do{
```

```
                SetLEDStateUsingPrinter(LED_state);
                for(jj=0;jj<30000;jj++)waitt = (int)((double)jj * (double)jj);
                SetLEDStateUsingPrinter(LEDnone);
                for(jj=0;jj<30000;jj++)waitt= (int)((double)jj * (double)jj);
                }while(!kbhit());
              }                                                          270
            else {
              Fatal_Error_Message("Invalid LED control mode");
              }
            break;
    case 'c':               //Toggle calibrate mode
    case 'C':       if(CalibrateMode){
              fclose(file);
              CalibrateMode = 0;
              sprintf(mess,"Calibration Off - Choose from menu->");
              }                                                          280
            else{
              CalibrateMode = 1;
              Status_Message("Enter file name for calibration data->");
              scanf("%s",&filename);
              file = fopen(filename,"w");
              sprintf(mess,"Calibration On - Choose from menu->");
              }
            break;
    case 'd':               //Display any image to monitor
    case 'D':       page=0;                                            290
            do{
              Show_Process_Image(page);
              sprintf(mess,"Displaying image %d - Enter (0-7) or any key->", page);
              Status_Message(mess);
              Case_Number(&page);
              }while (page < 8);
            sprintf(mess,"Choose from menu->");
            break;
    case 'f':               //Image analysis to find blobs and identify target
    case 'F':       if(DefaultParameters.OverLayOnLive)pageformarking = overlaypage;
            else pageformarking = storefordefine;
            Status_Message("Enter image to analyze->");
            Case_Number(&page);
            Status_Message("Wait for processing...");
            Show_Process_Image(page);
            roi.resolution = 4;
            FindThreshold(page, -1, roi.resolution, DefaultParameters.NumberInSlope,
&thresh, showgraphics, DefaultParameters.ThresholdPercent);
            roi.resolution = 1;
            autothresh = thresh;                                         310
            if(DefaultParameters.Threshold>-1)thresh = DefaultParameters.Threshold;
```

```
            //sprintf(mess,"thresh permthresh and autothresh %d %d %d",thresh,DefaultParar
            //printandwait(mess);
            DefineImage(page, -1,-1,storefordefine,&image,
                    DefaultParameters.MaximumBlobSize,DefaultParameters.MinimumBlobSize
            if(DefaultParameters.OverLayOnLive){
               EraseOverlay(overlaypage);
               liveoverlay(overlaypage);
               MarkAllBlobs(image,overlaypage);
               }                                                          320
            else{
               MarkAllBlobs(image,storefordefine);
               Show_Process_Image(storefordefine);
               }
            _clearscreen(0);
            _settextposition(1,1);
            if(image.num_of_blobs_found)OutColorText("
         Average\n");
            if(image.num_of_blobs_found)OutColorText("     Blob #   x       y
    Size   Brightness\n");                                                330
            if(image.num_of_blobs_found)OutColorText("     --------- ---     ---
    ------ ------------\n");
            for(i=0;i<image.num_of_blobs_found;i++){
              xs = image.blobs[i].centx;
              ys = image.blobs[i].centy;
              sprintf(text,"%d",i);
              writetext(pageformarking, (int) xs, (int) ys, text, WHITE);
              if((i<MAX_BLOBS_PER_PICTURE) || (i<20)){
                 sprintf(mess,"%10d%10.1f%10.1f%10.0f%10.1f\n", i,image.blobs[i].centx,in
                 OutColorText(mess);                                      340
                 }
              }
              printf("\n");
            sprintf(mess,"Image %d, Thresholds P=%d, T=%d - Hit any key for template
match->",page, DefaultParameters.Threshold, autothresh);
            printandwait(mess);
            if(!TargetCosys(&image,&allstats,DefaultParameters.Tolerance,DefaultParamete
{
              LoadTargetFromStats(&image,&target,&allstats);
              if(DefaultParameters.OverLayOnLive){                        350
                 liveoverlay(overlaypage);
                 Mark_Target (&target,overlaypage,DefaultParameters.VideoCenterCol,Defau
                 }
              else {
                 Mark_Target (&target,storefordefine,DefaultParameters.VideoCenterCol,De
                 Show_Process_Image(storefordefine);
                 }
              if(CalibrateMode) LogTargetResults(&target, file);
```

```
            sprintf(mess,"Template match in image %d - Choose from menu->",page);
            }
          else sprintf(mess,"No target identified - Choose from menu->");
          break;
case 'g':               //Grab 2 images and subtract
case 'G':       Status_Message("Wait for processing...");
          if(DefaultParameters.LEDControlMode==FREE_RUN){
             Charlotte_Grab(0,1,2,3);     //grab four images
             intensitymax=0;              //initial max and min
             intensitymin=1.e20;
             WHOLE_IMAGE(roi);
             roi.resolution=3;
             for (jj=0; jj<4; jj++){    //sort intensity
               Integrate_Image(jj,&sum[jj],roi);
               if (sum[jj]>intensitymax) {
                 intensitymax=sum[jj];
                 firstbright=jj;
                 }
               if (sum[jj]<intensitymin) {
                 intensitymin=sum[jj];
                 dim=jj;
               }
             }
             roi.resolution=1;
             }
          else if(DefaultParameters.LEDControlMode==COMPUTER_CONTROLLED){
             firstbright = 0;
             dim = 1;
             Grab_Frame (firstbright, LED_state);
             Grab_Frame (dim, LEDnone);
             WHOLE_IMAGE(roi);
             }
          else if(DefaultParameters.LEDControlMode==PRINTER_PORT){
             firstbright = 0;
             dim = 1;
             Grab_Frame_Printer_Port (firstbright, LED_state);
             Grab_Frame_Printer_Port (dim, LEDnone);
             WHOLE_IMAGE(roi);
             }
          else {
             sprintf(mess,"Incorrect Led Control Mode - Choose from menu->");
             break;
             }
          //Subtract_Images(firstbright,dim,-1,subtracted,2,128,roi);
          Subtract_Images(firstbright,dim,-1,subtracted,1,0,roi);
          Show_Process_Image(subtracted);
          sprintf(mess,"Displaying Subtracted Image - Choose from menu->");
```

```
        break;
case 'h':              //Calculate histogram for any image
case 'H':       Status_Message("Enter image for histogram->");
        Case_Number(&page);
        Status_Message("Wait for processing...");                    410
        WHOLE_IMAGE(roi);
        ShowPixelsInHistogram(roi, page, -1, -1, storeforhist, workingforhist);
        sprintf(mess,"Choose from menu->");
        break;
case 'i':
case 'I':
        DoDirectory("img","\vRImage Files on the Disk");
        Status_Message("Enter the image filename (NO EXTENSION) -> ");
        if( GetAFileName(afile,FILE_NAME_LENGTH,"img") != SUCCESS){
           sprintf(mess,"Image File Name Was Not Read, Choose From Menu -> ");
           break;
           }
        Status_Message("Enter page # -> ");
        Case_Number(&page);
        if( (page < 0) || (page > 7) ){
           sprintf(mess,"Invalid page number, I/O Aborted, Choose From Menu -> ");
           break;
           }
        Status_Message("R or W (or Quit)?-> ");
        do{                                                          430
           ch = getch();
           ch = toupper(ch);
           }while( (ch != 'R') && (ch != 'W') && (ch != 'Q'));
        if(ch == 'Q') {
           ch = 'r';
           break;
           }
        if(ch == 'R') {
           if(!DoesFileExist(afile)){
              sprintf(mess,"File does not exist, I/O aborted, Choose from menu ->");
              break;
              }
           if(ReadImage(afile, page) == SUCCESS){
              sprintf(mess,"Ok, Choose from menu->");
              break;
              }
           sprintf(mess,"Error Reading Image, Choose from menu->");
           break;
           }
        if(ch == 'W'){                                               450
           if(DoesFileExist(afile)){
              if(!AnsweredYes("That file exists, overwrite (y,n)? -> ")){
```

```
                sprintf(mess,"I/O Aborted, Choose from menu->");
                break;
                }
                 }
               if(StoreImage(afile, page) == SUCCESS){
                  sprintf(mess,"Ok, Choose from menu->");
                  break;
                  }
               sprintf(mess,"Error Writing Image, Choose from menu->");
               break;
               }
            break;
   case 'l':              //Set mode for type of LED control
   case 'L':       sprintf(mess,"Choose (F)ree, (D)igital, or (P)arallel mode->"
);
           Status_Message(mess);
           ch = getch();
           switch (ch){
             case 'f' :
             case 'F' :
               DefaultParameters.LEDControlMode = FREE_RUN;
               sprintf(mess,"Free Run Mode Selected - Choose from menu->");
               break;
             case 'd' :
             case 'D' :
               DefaultParameters.LEDControlMode = COMPUTER_CONTROLLED;
               sprintf(mess,"Digital Port Controlled Mode Selected - Choose from menu->")
               break;
             case 'p' :
             case 'P' :
               DefaultParameters.LEDControlMode = PRINTER_PORT;
               sprintf(mess,"Printer Port Controlled Mode Selected - Choose from menu->")
               break;
             default:  sprintf(mess,"Invalid entry - Choose from menu->");
             }
           break;
  case 'n':              //Set number of active computer controlled LEDs
  case 'N':       Status_Message("Enter desired number of active LEDs(0-7)");
          Case_Number(&page);
          if(page == 0)LED_state=LEDnone;
          if(page == 1)LED_state=LEDone;
          if(page == 2)LED_state=0xEE;
          if(page == 3)LED_state=0x5B;
          if(page == 4)LED_state=0xAA;
          if(page == 5)LED_state=0x15;
          if(page == 6)LED_state=0x11;
          if(page == 7)LED_state=0x01;
```

```
            if(page == 8)LED_state=0x00;
            sprintf(mess,"%d active LEDs set - Choose from menu->", page);
            break;
    case 'o':
    case 'O':
            trans.FocalLength = DefaultParameters.FocusDistance;
            GetPose(&target, &trans, &(allstats.id), &raw, &pose, DefaultParameters);
            /*
            Status_Message("Enter x,y,z,yaw,p,r ->");
            ReadsixFloat(&x, &y, &z, &yaw, &p, &r);
            sprintf(mess,"Output %f %f %f %f %f %f?(y,n) ->",x,y,z,yaw,p,r);
            */
            sprintf(mess,"Output %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f?(y,n) ->",pose.x,pose.y
            if(!AnsweredYes(mess)){
               sprintf(mess,"Nothing Output - Choose from menu -> ");
               }
            else {
               status = DoComInitialize(&port);
               if(status != COM_SUCCESS)sprintf(mess,"Could not initialize com port");
               else {
                  sprintf(mess,"%9.3f %9.3f %9.3f %9.3f %9.3f %9.3f\n",pose.x,pose.y,pose.z,
                  if(com_send(port,mess) != COM_SUCCESS)sprintf(mess,"Could not transmit");
                  else sprintf(mess,"Pose Output - Choose from menu ->");
                  }
               com_close(port,1);
               }
            break;
    case 'p':
    case 'P':
            SetParameters( &DefaultParameters);
            sprintf(mess,"Parameters set - Choose from menu->");
            break;
    case 'q':               //Terminate program
    case 'Q':       finishtext();
            End_vision();
            _clearscreen(0);
    // You can save parameters manually in Parameter set menu
            /*
            If(AnsweredYes("Do you want to save the Parameters (y,n) -> ")){
               DoDirectory("pms","\vRParameter Files on Disk");
               while(SaveParameterSet(&DefaultParameters) != SUCCESS)printandwait("Failed
to Write File, Press a Key to Try Again (Cntl-C Exits).");
               }
               */
            Status_Message("Program Terminated Normally");
            break;
    case 'r':               //Calculate brightness statistics ROI
```

```
    case 'R':       Status_Message("Enter image number for statistics->");
            Case_Number(&page);
            DefineROIandCompute(page);
            sprintf(mess,"Choose from menu->");
            break;
    case 's':               //Subtract any two images
    case 'S':       Status_Message("Subtract second image from first - Enter first
image->");
            Case_Number(&frame1);
            Status_Message("Enter second image->");
            Case_Number(&frame2);
            Status_Message("Wait for processing...");
            WHOLE_IMAGE(roi);
            //Subtract_Images(frame1,frame2,-1,subtracted, 2, 128, roi);
            Subtract_Images(frame1,frame2,-1,subtracted, 1, 0, roi);
            Show_Process_Image(subtracted);
            sprintf(mess,"Subtracted image in page %d - Choose from menu->", subtracte
            break;
    case 'v':               //Display live video
    case 'V':       Live_Video();
            sprintf(mess,"Displaying Live Video - Choose from menu->");
            break;
    default:    sprintf(mess,"Invalid entry - Choose from menu->");
    }
    }while((ch!='Q') && (ch!='q'));
 }


void DisplayRectangleStats(ROI *roi, int increment)
 {
    char message[127];
    _settextposition(16,1);
    sprintf(message,"  Bottom-left: (%d,%d), upper-right: (%d,%d), Increment:
%d ",roi->xs,roi->ys,roi->xe,roi->ye,increment);
    OutColorText(message);
    _settextposition(24,34);
    }

void LogTargetResults(Target *target, FILE *file)
 {
 double mirrorx, mirrory, mirrorroll, ledx, ledy, mirrorlength;
 double dx,dy;
 int thenumber;
 _clearscreen(0);
 _settextposition(7,1);
 if((target->left > -1) && (target->right > -1)){
```

```
      mirrorx = (target->leftretro.centx + target->rightretro.centx)/2.;
      mirrory = (target->leftretro.centy + target->rightretro.centy)/2.;
      dy =    -(target->leftretro.centy - target->rightretro.centy);
      dx =    -(target->leftretro.centx - target->rightretro.centx);
      mirrorroll = atan2(dy,dx ) * 180 / M_PI;
      mirrorlength = sqrt(dy*dy + dx*dx);
      }
   else{
      mirrorx = (target->topretro.centx + target->bottomretro.centx)/2.;
      mirrory = (target->topretro.centy + target->bottomretro.centy)/2.;
      dy =    (target->topretro.centy - target->bottomretro.centy);
      dx =   -(target->topretro.centx - target->bottomretro.centx);
      mirrorroll = atan2(dx,dy ) * 180 / M_PI;
      mirrorlength = sqrt(dy*dy + dx*dx);
      }

   ledx = target->led.centx;
   ledy = target->led.centy;
   printf("\n     Mirror    x       y      length     \n");
   printf  ("              ---     ---    --------   \n");
   printf  ("         %10.1f%10.1f%10.1f\n",mirrorx,mirrory,mirrorlength);
   printf("\n      Led      x       y     roll(deg)  \n");
   printf  ("              ---     ---    ----------- \n");
   printf  ("         %10.1f%10.1f%10.1f\n",ledx,ledy,mirrorroll);
   if(file != NULL){
     Status_Message("Enter a calibration memo number->");
     ReadInt(&thenumber);
     fprintf(file,"%d,",thenumber);
     fprintf(file,"%10.1f,%10.1f,%10.1f,",mirrorx,mirrory,mirrorlength);
     fprintf(file,"%10.1f,%10.1f,%10.1f\n",ledx,ledy,mirrorroll);
     }
}

int SetParameters(Parameters *param)
 {
   char mess[80];
   int ch,ach;

   sprintf(mess,"Choose from menu->");
 do{
  _clearscreen(0);
 _settextposition(5,1);
OutColorText("*************************************************\n");
OutColorText("* (\vGA\vW)ddress of printer port set:           *\n");
OutColorText("* (\vGD\vW)esignate Target Center:              *\n");
OutColorText("* (\vGF\vW)ocus Distance of Camera:             *\n");
OutColorText("* (\vGG\vW)reatest Blob Size in Pixels:         *\n");
```

```
OutColorText("* (\vGI\vW)nput/Output Parameter set:                    *\n");
OutColorText("* (\vGL\vW)EDS - Number to seek:                         *\n");
OutColorText("* (\vGM\vW)inimum Blob Size in Pixels:                   *\n");
OutColorText("* (\vGN\vW)umber of Points in Slope Calculation:         *\n");
if(param->OverLayOnLive)
OutColorText("* (\vGO\vW)verlay on Processed image:                    *\n");
else
OutColorText("* (\vGO\vW)verlay on Live:                               *\n");
OutColorText("* (\vGP\vW)ercent Distance Between Histogram Peaks:      *\n");
OutColorText("* (\vGR\vW)eturn to MAIN Menu:                           *\n");
OutColorText("* (\vGS\vW)et Threshold:                                 *\n");
OutColorText("* (\vGT\vW)olerance for Template Match:                  *\n");
OutColorText("*****************************************************\n");
DisplayParameters(param);

   Status_Message(mess);
   ch = getch();
   ch = toupper(ch);
   switch (ch){

   case 'a':                                                          660
   case 'A':       sprintf(mess,"Enter Printer Port address (IN HEX) A&M=378,
IBM750=3BC ->");
           Status_Message(mess);
           ReadHex(&(param->PrinterPortAddress));
           setprinterportaddress(param->PrinterPortAddress);
           sprintf(mess,"Address is %x. - Choose from menu->",param->PrinterPortAddre
           Status_Message(mess);
           break;

                                                                      670
   case 'd':
   case 'D':       DesignateCenter(param,7);
           sprintf(mess,"Choose from menu -> ");
           break;

   case 'f':
   case 'F':       sprintf(mess,"Focus Distance of Camera is (%.1lf). Enter new
value->",param->FocusDistance);
           Status_Message(mess);
           ReadDouble(&(param->FocusDistance));                       680
           sprintf(mess,"Value is %.1lf. - Choose from menu->",param->FocusDistance);
           break;

   case 'g':
   case 'G':       sprintf(mess,"Greatest Blob Size is (%ld). Enter new value->",param->Ma
           Status_Message(mess);
           ReadLong(&(param->MaximumBlobSize));
```

```
            sprintf(mess,"Value is %ld. - Choose from menu->",param->MaximumBlobSize);
            break;

    case 'i':
    case 'I':
            DoDirectory("pms","\vRParameter Files on Disk");
            do{
               Status_Message("Do you want to Read (R) or Write (W) (Q=Quit)?-> ");
               ach = getch();
               ach = toupper(ach);
               }while( (ach != 'R') && (ach != 'W') && (ach != 'Q'));
            if(ach == 'R'){
               if(ReadParameterSet(param) != SUCCESS)sprintf(mess,"Failed to Read File");
               else sprintf(mess,"Choose from menu->");
               }
            else if(ach == 'W'){
               if(SaveParameterSet(param) != SUCCESS)sprintf(mess,"Failed to Write File");
               else sprintf(mess,"Choose from menu->");
               }
            break;

    case 'l':
    case 'L':       sprintf(mess,"Number of LEDS to seek is (%d). Enter new value->",param->
            Status_Message(mess);
            ReadInt(&(param->NumberOfLEDSToSeek));
            sprintf(mess,"Value is \vR%d\vW. - Choose from menu->",param->NumberOfLEDSTo
            break;

    case 'm':
    case 'M':       sprintf(mess,"Minimum Blob Size is (%d). Enter new value->",param->Mini
            Status_Message(mess);
            ReadInt(&(param->MinimumBlobSize));
            sprintf(mess,"Value is %d. - Choose from menu->",param->MinimumBlobSize);
            break;

    case 'n':
    case 'N':       sprintf(mess,"Number of Points in slope is (%d). Enter new
value ->",param->NumberInSlope);
            Status_Message(mess);
            ReadInt(&(param->NumberInSlope));
            sprintf(mess,"Value is %d. - Choose from menu->",param->NumberInSlope);
            break;

    case 'o':
    case 'O':
            if(param->OverLayOnLive){
               param->OverLayOnLive = 0;
```

```
                sprintf(mess,"Will Now Overlay on \vRProcessed\vW image. Choose from menu
->");
                }
             else {
                param->OverLayOnLive = 1;
                sprintf(mess,"Will Now Overlay on \vRLive\vW image. Choose from menu ->"
                }
             break;

    case 'p':
    case 'P':       sprintf(mess,"Threshold Percent is (%d). Enter new value ->",param->Th
            Status_Message(mess);
            ReadInt(&(param->ThresholdPercent));
            sprintf(mess,"Value is %d. - Choose from menu->",param->ThresholdPercent);
            break;

    case 'r':
    case 'R':       break;

    case 's':
    case 'S':       //Set threshold calculation manually or automatically
            sprintf(mess,"Threshold is (%d). Enter new threshold (-1 for automatic)->",para
            Status_Message(mess);
            ReadInt(&(param->Threshold));
            sprintf(mess,"Threshold is %d. - Choose from menu->",param->Threshold);
            Status_Message(mess);
            break;

    case 't':
    case 'T':       sprintf(mess,"Template Match Tolerance is (%.1f). Enter new
value ->",param->Tolerance);
            Status_Message(mess);
            ReadDouble(&(param->Tolerance));
            sprintf(mess,"Value is %.1f. - Choose from menu->",param->Tolerance);
            break;

   default:    sprintf(mess,"Invalid entry - Choose from menu->");
   }
   }while((ch!='R') && (ch!='r'));
   return 0;
}

void DoDirectory(char *extension, char *mess)
{
#define WINDOW_START_ROW 2
#define WINDOW_END_ROW 20
```

```
   #define WINDOW_WIDTH 20
   struct find_t fileinfo;
   int status;
   int row;
   int scol,ecol;
   char filename[13];
   _clearscreen(0);
   _settextposition(1,1);
   OutColorText(mess);
   scol = 1;
   ecol = WINDOW_WIDTH-2;
   _settextwindow(WINDOW_START_ROW,scol,WINDOW_END_ROW,ecol);
   row = WINDOW_START_ROW;
   sprintf(filename,"*.%s",extension);
   status = _dos_findfirst(filename,_A_NORMAL,&fileinfo);
   if(status != 0){
      OutColorText("\vRNo Files To \n");
      OutColorText("\vR  Choose From");
      }
  while(status == 0){
      OutColorText(fileinfo.name);
      OutColorText("\n");
      row++;
      if(row == WINDOW_END_ROW){
         row = WINDOW_START_ROW;
         scol = scol + WINDOW_WIDTH;
         ecol = ecol + WINDOW_WIDTH;
         _settextwindow(WINDOW_START_ROW,scol,WINDOW_END_ROW,ecol);
         }
      status = _dos_findnext(&fileinfo);
      }
  _settextwindow(1,1,25,80);
  }

int GetAFileName(char *filename, int namelength, char *ext)
 {
 char tempfilename[13];
 char *period;
 scanf("%8s",tempfilename);
 period = strchr(tempfilename,'.');
 if(period != NULL)*period = 0;
 return FormFullFileName(filename,namelength,tempfilename,ext);
 }

int DoesFileExist(char *name)
 {
 struct find_t fileinfo;
```

```
 if(_dos_findfirst(name,_A_NORMAL,&fileinfo) != 0)return 0;
 return 1;                                                          830
 }

int ReadParameterSet(Parameters *param)
 {
 #define FILE_NAME_LENGTH 13
 FILE *input;
 char filename[FILE_NAME_LENGTH];
 Status_Message("Enter the parameter filename (NO EXTENSION) -> ");
 if( (GetAFileName(filename,FILE_NAME_LENGTH,"pms") != SUCCESS) || (!DoesFileExist
){                                                                  840
    return FAIL;
    }
 input = fopen(filename,"r");
 if(fscanf(input,"%x",&(param->PrinterPortAddress))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->NumberOfLEDSToSeek))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->LED_State))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->Threshold))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->LEDControlMode))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->NumberInSlope))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->ThresholdPercent))!=1)return FAIL;   850
 if(fscanf(input,"%d",&(param->MinimumBlobSize))!=1)return FAIL;
 if(fscanf(input,"%ld",&(param->MaximumBlobSize))!=1)return FAIL;
 if(fscanf(input,"%lf",&(param->Tolerance))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->OverLayOnLive))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->VideoCenterRow))!=1)return FAIL;
 if(fscanf(input,"%d",&(param->VideoCenterCol))!=1)return FAIL;
 fclose(input);
 return SUCCESS;
 }
                                                                    860
int SaveParameterSet(Parameters *param)
 {
 #define FILE_NAME_LENGTH 13
 char filename[FILE_NAME_LENGTH];
 FILE *output;
 Status_Message("Enter the filename (NO EXTENSION) -> ");
 if( GetAFileName(filename,FILE_NAME_LENGTH,"pms") != SUCCESS) return FAIL;
 else if( DoesFileExist(filename) && (!AnsweredYes("That File Exists, Overwrite
(y,n)? -> "))) return FAIL;
 else{                                                              870
    output = fopen(filename,"w");
    if(output == NULL) return FAIL;
    fprintf(output,"%x\n",param->PrinterPortAddress);
    fprintf(output,"%d\n",param->NumberOfLEDSToSeek);
    fprintf(output,"%d\n",param->LED_State);
```

```
    fprintf(output,"%d\n",param->Threshold);
    fprintf(output,"%d\n",param->LEDControlMode);
    fprintf(output,"%d\n",param->NumberInSlope);
    fprintf(output,"%d\n",param->ThresholdPercent);
    fprintf(output,"%d\n",param->MinimumBlobSize);                    880
    fprintf(output,"%ld\n",param->MaximumBlobSize);
    fprintf(output,"%f\n",param->Tolerance);
    fprintf(output,"%d\n",param->OverLayOnLive);
    fprintf(output,"%d\n",param->VideoCenterRow);
    fprintf(output,"%d\n",param->VideoCenterCol);
    fclose(output);
    }
 return SUCCESS;
 }
                                                                      890
void DisplayFrameUses(int firstbright, int dim)
 {
 char mess[80];
 _settextwindow(13,54,25,80);
 OutColorText("\vC************************\n");
 OutColorText("* \vCNormal Image Frames\n");
 OutColorText("* \vC0-3 Used For Charlotte\n");

 OutColorText("* \vC    Grab.\n");
 //OutColorText("* \vCLEDS on 1\n");                                  900
 //OutColorText("* \vCLEDS off 0\n");
 sprintf(mess,"* \vCLEDS on %d\n",firstbright); OutColorText(mess);
 sprintf(mess,"* \vCLEDS off %d\n",dim); OutColorText(mess);
 OutColorText("* \vCSubtracted 4\n");
 OutColorText("* \vCAccumulated 5\n");
 OutColorText("* \vCTemporary Store 6\n");
 OutColorText("* \vCOverlay & Histogram 7\n");
 OutColorText("\vB************************");
 _settextwindow(1,1,25,80);
 }                                                                    910

void DisplayParameters(Parameters *parameters)
 {
 char mess[80];
 _settextwindow(2,54,25,80);
 OutColorText("\vB************************\n");
 sprintf(mess,"* Print.Port Add. \vR%x\n",parameters->PrinterPortAddress); OutColorText(m
 sprintf(mess,"*  # of LEDS to Seek \vR%d\n",parameters->NumberOfLEDSToSeek);
OutColorText(mess);
 if(parameters->Threshold == -1){                                     920
    sprintf(mess,"* Threshold \vRIs Automatic\n"); OutColorText(mess);
    }
```

```
 else {
    sprintf(mess,"* Threshold \vR%d\n",parameters->Threshold); OutColorText(mess);
    }
 sprintf(mess,"* # in Slope \vR%d\n",parameters->NumberInSlope); OutColorText(mess);
 sprintf(mess,"* Thresh. %% \vR%d\n",parameters->ThresholdPercent); OutColorText(me
 sprintf(mess,"* Min.Blob Size \vR%d\n",parameters->MinimumBlobSize); OutColorText(me
 sprintf(mess,"* Max.Blob Size \vR%ld\n",parameters->MaximumBlobSize); OutColorText(m
 sprintf(mess,"* Tolerance \vR%.1f\n",parameters->Tolerance); OutColorText(mess);
 sprintf(mess,"* Center at "); OutColorText(mess);
 sprintf(mess,"(\vR%d\vW,\vR%d\vW)\n",parameters->VideoCenterCol,parameters->Vide
OutColorText(mess);
 if(parameters->OverLayOnLive){
    sprintf(mess,"* \vRMarking on Live.\n"); OutColorText(mess);
    }
 else {
    sprintf(mess,"* \vRMarking on Processed.\n"); OutColorText(mess);
    }
 sprintf(mess,"* Focus Distance \vR%.1lf  \n",parameters->FocusDistance); OutColorText(
 //OutColorText("\vB************************");
 _settextwindow(1,1,25,80);
 }

void OutColorText(char *string)
 {
 #define black 0
 #define blue 1
 #define green 2
 #define cyan 3                                                    950
 #define red 4
 #define magenta 5
 #define brown 6
 #define white 7
 #define darkgray 8
 #define lightblue 9
 #define lightgreen 10
 #define lightcyan 11
 #define lightred 12
 #define lightmagenta 13                                           960
 #define yellow 14
 #define brightwhite 15

 int i;
 char ch[2];
 i = 0;
 ch[1] = 0;
 _settextcolor(white);
 while((ch[0] = string[i]) !=0){
```

```
    i++;                                                              970
    if(ch[0] == '\v'){
       ch[0] = string[i];
       i++;
       switch (ch[0]){
      case 'K':  _settextcolor(black ); break;
      case 'B':  _settextcolor(blue ); break;
      case 'G':  _settextcolor(green ); break;
      case 'C':  _settextcolor(cyan ); break;
      case 'R':  _settextcolor(red ); break;
      case 'M':  _settextcolor(magenta ); break;                      980
      case 'N':  _settextcolor(brown ); break;
      case 'W':  _settextcolor(white ); break;
      case 'D':  _settextcolor(darkgray ); break;
      case 'b':    _settextcolor(lightblue ); break;
      case 'g':    _settextcolor(lightgreen ); break;
      case 'c':    _settextcolor(lightcyan ); break;
      case 'r':    _settextcolor(lightred ); break;
      case 'm':    _settextcolor(lightmagenta ); break;
      case 'Y':  _settextcolor(yellow ); break;
      case 'w':    _settextcolor(brightwhite ); break;                990
      default:  _outtext(ch);
      }
       }
    else _outtext(ch);
    }
 _settextcolor(white);
 }

int AnsweredYes(char *string)
 {                                                                   1000
 int ans;
 Status_Message(string);
 do {
    ans = getch();
    ans = toupper(ans);
    if(ans == 'Y')return TRUE;
    if(ans == 'N')return FALSE;
    }
    while(1);
 }                                                                   1010

char str[100];

void ReadInt(int *n)
 {
 fgets(str,100,stdin);
```

```
 /*scanf("%99s",str);*/
 sscanf(str,"%d",n);
 }

void ReadLong(long *n)
 {
 fgets(str,100,stdin);
 /*scanf("%99s",str);*/
 sscanf(str,"%ld",n);
 }

void ReadDouble(double *n)
 {
 fgets(str,100,stdin);
 /*scanf("%99s",str);*/
 sscanf(str,"%lf",n);
 }

void ReadsixFloat(float *x, float *y, float *z, float *yaw, float *pitch, float
*roll)
 {
 fgets(str,100,stdin);
 /*scanf("%99s",str);*/
 sscanf(str,"%f %f %f %f %f %f",x,y,z,yaw,pitch,roll);
 }

void ReadHex(int *n)
 {
 fgets(str,100,stdin);
 /*scanf("%99s",str);*/
 sscanf(str,"%x",n);
 }

void EraseCross(int x, int y, int page);
void DrawCross(int x, int y, int page);
void DesignateCenter(Parameters *param, int page)
 {
    int x, y;
    int ch,ach;
    x = param->VideoCenterCol;
    y = param->VideoCenterRow;
    ClearPage(page);
    Show_Process_Image(page);
    Status_Message("Use Arrow Keys to Move Cross, Hit Q When Centered");
do{
    DrawCross(x,y,page);
    ch = getch();
```

```
    ch = toupper(ch);
    EraseCross(x,y,page);
    if(ch == 0){
       ach = getch();
       switch (ach){
       case 72:  /*up*/
           y++;                                                    1070
           break;

       case 80:  /*down*/
           y--;
           break;

       case 77:  /*right*/
           x++;
           break;
                                                                   1080
       case 75:  /*left*/
           x--;
           break;
       }
    }
    }while(ch != 'Q');
    param->VideoCenterRow = y;
    param->VideoCenterCol = x;
 }
                                                                   1090
#define LINEHALFLENGTH 10
#define WHITE 255
#define BLACK 0
void DrawCross(int x, int y, int page)
 {
 line(page,x-LINEHALFLENGTH,y,x+LINEHALFLENGTH,y,WHITE);
 line(page,x,y-LINEHALFLENGTH,x,y+LINEHALFLENGTH,WHITE);
 }

void EraseCross(int x, int y, int page)                            1100
 {
 line(page,x-LINEHALFLENGTH,y,x+LINEHALFLENGTH,y,BLACK);
 line(page,x,y-LINEHALFLENGTH,x,y+LINEHALFLENGTH,BLACK);
 }


int DoComInitialize(ComPort *port)
 {
 struct find_t fileinfo;
 int status;                                                       1110
```

```
 int baud;
 char parity;
 int numbits;
 int stopbits;
 unsigned bufsize;
 FILE *inp;
 status = _dos_findfirst("com.dat",_A_NORMAL,&fileinfo);
 if(status == 0){
    inp = fopen("com.dat","r");
    fscanf(inp,"%d\n",port);
    fscanf(inp,"%d\n",&baud);
    fscanf(inp,"%d\n",&numbits);
    fscanf(inp,"%d\n",&stopbits);
    fscanf(inp,"%d\n",&bufsize);
    fscanf(inp,"%c",&parity);
    fclose(inp);
    }
 else {
    *port = 2;
    baud = 9600;
    parity = 'N';
    numbits = 8;
    stopbits = 2;
    bufsize = (unsigned) 2048;
    }
//printf("opening port %d, baud %d, parity %c, digits %d, stop %d, buff %d\n",
//    *port,baud,parity,numbits,stopbits,bufsize );
status = com_open(*port,bufsize,baud,parity,numbits,stopbits);
if(status != COM_SUCCESS) return status;
com_raisedtr(*port);
return COM_SUCCESS;
}
```

1.4 Function: init_vid

Documentation Date: 3/9/95

Prototypes:

```
void main(void);
```

Source File: *init_vid.c*
Type of Function: User Callable
Header Files Used in init_vid.c: *<float.h> <stdio.h> <graph.h> <string.h> <conio.h> <time.h> <math.h> <stdlib.h> "targa.h"*

Description:

This routine initializes the video. The system call used in the Initialize_Video function does not work in the actual main program due to memory problems. This simple small program calls the initialize video function to set the targa parameters into the targa.par file. To get any benefit from it, you should delete the targa.par before running the program. Otherwise, it will simply read the targa.par file and exit.

1.4.1 Program Listing:

```
#include < float.h>
#include <stdio.h>
#include <graph.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include <stdlib.h>
#include "targa.h"

                                                                    10
void main( void);

void main() {
int overlaypage;
ROI roi;

    Initialize_vision(0);
    overlaypage=7;
    WHOLE_IMAGE(roi);
     if(inittext("d"))exit(1);                                      20
    ClearPage(overlaypage);
    DrawOverStrings(WHITE,overlaypage);
    Live_Video();       }
```

1.5 messages

Documentation Date: 3/9/95
New Data Types:
None.
Definitions: Local Definitions:

- STATPOS = 24,1 - Row and column for the Status Message.
- GSTATPOS = 1,1 - Row and column for the Graphics Status Message.

Prototypes:

`See Datatypes.h for Global prototypes.`
Source File: *messages.c*
Type of Function: User Callable
Header Files Used in messages.c: *<stdio.h> <graph.h> <conio.h> <stdlib.h> "targa.h"*

Description:

This file contains the message writing routines.

1. Status_Message - Prints a status message.
2. clear_status - Clears the status message.
3. Fatal_Error_Message - Prints a status message then quits the program.
4. printandwait - Prints a status message and waits for the user to hit any key.
5. gStatus_Message - Prints a status message on a graphics screen.
6. gclear_status - Clears a graphics mode status message.
7. gprintandwait - Prints a graphics status message and waits for the user to hit a key.
8. gpromptandread - Prints a graphics status and reads a value.

1.5.1 Program Listing:

```
#include <graph.h>
#include <stdlib.h>
#include <conio.h>
#include <stdio.h>
#include "targa.h"
```

```
 #define STATPOS 24,1
/*#define GSTATPOS 20,1*/
 #define GSTATPOS 1,1

 int gscanf( char *format,  void *value);
/*Put OutColorText in an appropriate file*/
 void OutColorText( char *string);

 void Status_Message(message)
 char message[];
{ clear_status();
  _settextposition(STATPOS);
  OutColorText(message);
}

 void clear_status()
{
  _settextposition(STATPOS);
  _outtext("
       ");
}

 void Fatal_Error_Message(message)
 char message[];
{ Status_Message(message);
  exit(1);
  }

 void printandwait(message)
 char message[];
{
  Status_Message(message);
  getch();
  clear_status();
}

 void gStatus_Message(message)
 char message[];
{
  gclear_status();
  _settextposition(GSTATPOS);
  OutColorText(message);
/*
  gclear_status();
  _moveto(GSTATPOS);
  _outgtext(message);
```

```
  */
}

 void gclear_status()
{
  _settextposition(GSTATPOS);
  _outtext("                                                    60
  ");
  /*
  _settextposition(GSTATPOS);
  _outgtext("
  ");
  */
}

 void gprintandwait(message)
 char message[];                                                70
{
  gStatus_Message(message);
  getch();
  gclear_status();
  /*
  gStatus_Message(message);
  getch();
  gclear_status();
  */
}                                                              80

 void gpromptandread( char message[], char format[], void *value)
{
_displaycursor(_GCURSORON);
gStatus_Message(message);
scanf(format,value);
gclear_status();
_displaycursor(_GCURSOROFF);
}

                                                               90
 #define LEN 80
 int gscanf( char *format, void *value)
{
 char c[2];
 char string[LEN+1];
 int i;
i=0;
c[1] = 0;
 do{
  c[0] = ( char) getch();                                     100
```

```
  _outgtcxt(c);
   if(i<LEN){
    string[i]=c[0];
    i++;
    }
  } while(c[0]!='\r');
   return(sscanf(string,format,valuc));
}
```

1.6 Com3_2.c

Documentation Date: 3/9/95

Description:

This file contains the routines used to drive the serial communications port. See the listing for further information.

1.6.1 Header File Listing:

```
/*-----------------------------------------------------------------
/*                                                    */
/* FILE   : com.h                                          */
/*                                                    */
/* Header file for interrupt driven serial I/O.                    */
/*                                                    */
/* AUTHOR  : Russell Neeper                                */
/*                                                    */
/* MODIFICATIONS :                                         */
/*                                                    */                    10
/*  date       programmer     version     comments              */
/*  ----    ----------  -------   ----------     */
/*   Spring '89      RN        1.0     initial version
 */
/*   Fall '90       RN        2.0     Fixed initialization bugs,
 */
/*                      No longer uses BIOS, added        */.
/*                      flush command.          */
/*   1/6/95         LJE     3.2     RENAMED FLUSH        */
/*                      CHANGED INT TO COMPORT DATATYPE */    20
/*                      SWITCH SUCCESS AND FAIL     */
/*                                                    */
/*-----------------------------------------------------------------

typedef char  ComPort;
#define COM1      0
#define COM2      1
/*
#define COM_SUCCESS 1
#define COM_FAILURE 0                                           30
*/
#define COM_SUCCESS 0
#define COM_NONEAVAIL 2
#define COM_BADPORT 3
#define COM_TIMEOUT 4
#define COM_BADBAUD 5
```

```
#define COM_BADPARITY 6
#define COM_BADBITS 7
#define COM_BADSTOP 8
#define COM_NOMEMORY 9
#define COM_FALSE 0
#define COM_TRUE 1

/* function prototypes */

/* void com_break(ComPort port,unsigned int l); */

/* Return the number of available input chars on a COM port */
    int com_avail(ComPort port);
/* Flush all input from the COM port */
    void com_flush_input(ComPort port);
/* Reads a character from port.  If there isn't a char available it returns
COM_NONEAVAIL Otherwise it puts the character in *c and returns COM_SUCCESS
*/
    int  com_read(ComPort port, char *c);
/* Writes character out port. Actually puts it in the queue. If it CANNOT
Output the character within OUTTRIES, it returns COM_TIMEOUT. If you pass
a bad port address it returns COM_BADPORT. Success returns COM_SUCCESS. */
    int  com_write(ComPort port, char c);
/* This sends an entire string out. If it succeeds it returns COM_SUCCESS.
If it cannot output a character within OUTTRIES, it returns COM_TIMEOUT. If
you pass a bad port address it returns COM_BADPORT. */
    int  com_send(ComPort port, char *msg);
/*This routine opens and initializes the com port, it allocates its own memory
for the buffer too.
Note that baud is the actual number except 38 -> 38400 115 -> 115200
This will return: COM_SUCCESS COM_BADBAUD COM_BADPARITY COM_BADBITS COM_BA
    COM_NOMEMORY COM_BADPORT
    int com_open([1, 2]Com, Buffer Size, Baud Rate (eg. 1200),
            ['N', 'n' , 'O', 'o', 'E', 'e']Parity, [7, 8]Bits,
            [1, 2]StopBits)*/
    int  com_open(ComPort port, unsigned size, int baud,
            char parity, int digits, int stop);
/* This closes the port and frees the buffer. If leavedtr == 0 it clears dtr
otherwise it sets it. */
    void com_close(ComPort port, char leavedtr);
/* Returns True (non zero) if carrier is present, False (zero) otherwise */
    int  com_carrier(ComPort port);
/* Returns True (non zero) if DSR is present, False (zero) otherwise */
    int  com_dsr(ComPort port);
/* Returns True (non zero) if PORT is READY for a new output character, False
(zero) otherwise */
    int  com_ready(ComPort port);
```

```
/* Drop the DTR signal on a COM port */
    void com_dropdtr(ComPort port);
/* Raise the DTR signal on a COM port */
    void com_raisedtr(ComPort port);
/* Reads a character from port. If there isn't a char available it waits for
one. Otherwise it puts the character in *c and returns COM_SUCCESS */
    char com_read_wait(ComPort port);                                    90
```

1.6.2 Program Listing:

```
#define This_Version "com3_2.h"
/* Defines for various port parameters */
#define BITS_7    0x00        /* Data bits */
#define BITS_8    0x10
#define STOP_1    0x00        /* Stop bits */
#define STOP_2    0x20
#define PARITY_NONE  0x00     /* Parity modes */
#define PARITY_ODD   0x40
#define PARITY_EVEN  0x80

    /* The basic problem with this code is that it doesn't manipulate
(on its own) the dtr flags and it doesn't have an output buffer.  On
output it will wait for a slot to open before returning.  On input, if
the buffer isn't read frequently enough, data will overrun.

    To use it, I use 2048 for the buffsize.  Then, put a com_open(1,
BUFFSIZE, 9600, 'N', 8, 1) or whatever you need the parameters to be.
To write something out, use com_write or com_send,
and to read a character, call com_read.  If com_read returns
COM_NONEAVAIL, then no character is available (i.e., the code is interrupt
driven, and doesn't halt waiting for new characters.)

    Look at the function declarations to get the parameters.  This code
has been used with Turbo C andwith Microsoft C 6.0.

Command          File      Header File        Description
---------------- --------- ------------------ ----------------------------
com_open         com.c     com.h        Opens serial communication port
com_close                      Closes serial communication port
com_break                      Sends a break
com_avail                      Returns # of chars waiting to read
com_flush_input                     Flushes all received characters
com_read                       Reads a character
com_write                      Writes a character
com_send                       Sends a string
com_carrier                    Tests if a carrier is present
com_dsr                        Tests if Data Set Ready is present
com_ready                      Tests if comm port can transmit
com_dropdtr                    Drops DTR signal
com_raisedtr                        Raises DTR signal
com_read_wait                       Reads a character, if not avail. it waits

*/
/*-------------------------------------------------------------------------
/*                                                                       */
/*  FILE   : com.c                                                       */
```

```
/*                                                    */
/*  These routines provide interrupt driven serial I/O.                    */
/*                                                    */
/*  AUTHOR  : Russell Neeper                                    */
/*                                                    */
/*  MODIFICATIONS :                                    */
/*                                                    */
/*   date        programmer      version      comments                */
/*   ----        ----------      -------      ----------              */
/*    Spring '89      RN          1.0      initial version
 */
/*    Fall '90        RN          2.0      Fixed initialization bugs,
 */
/*                        No longer uses BIOS, added          */
/*                        flush command.              */
/*    Jan 17, 92      LJE       2.1      Added a few characters to      */
/*                        eliminate warnings in TurboC      */
/*    Jan 6, 95       LJE       3.2      Cleaned up some comments       */
/*                        Fixed com_send              */
/*                        Changed name of flush           */
/*                    Changed return values to be descriptive */
/*                                                    */
/*----------------------------------------------------------------------*/

#include <dos.h>
#include <bios.h>
#include <conio.h>
#include <stdlib.h>
#include This_Version

/* Various PIC/COM masks and values */

#define PIC_MASK   0x21
#define PIC_EOI    0x20
#define ERR_MSK    0x9E

/* Definitions for interrupt handling */

/* COM port offsets */

#define COM_DATA   0 /* Data received on this I/O address */
#define COM_IER    1 /* This register enables interrupts */
#define COM_LCR    3 /* Line control register */
#define COM_MCR    4 /* Control Register (signals) */
#define COM_STAT   5 /* Status Register */
#define COM_MSR    6 /* Modem Status Register */
```

```
/* Data for installing COM port interrupts */

 #define COM_INT_1   0x0C   /* 0x0C handles IRQ4 or COM1 by standard */
 #define INT_MASK_1  0x10   /* Mask for PIC (programmable interrupt controller)
8
259A */
 #define COM_INT_2   0x0B   /* 0x0B handles IRQ3 or COM2 by standard */
 #define INT_MASK_2  0x08   /* Mask for PIC (programmable interrupt controller)
8
259A */

 #ifdef _TURBOC_
 void interrupt (far *old_com1_int)();/*LJE Added far */
 void interrupt (far *old_com2_int)();
/*void interrupt (*old_com1_int)();
void interrupt (*old_com2_int)();*/
 #else
 void (interrupt far *old_com1_int)();
 void (interrupt far *old_com2_int)();
 #define inportb inp
 #define outportb outp
 #define outport outpw
 #define disable _disable
 #define enable _enable
 #define setvect _dos_setvect
 #define getvect _dos_getvect
 #endif

 static void doenable( int); /*LJE*/

volatile char *msgBuff1;
 unsigned int size1,avail1;
volatile unsigned int lastPos1 = 0; /*LJE 1/6/95 added unsigned*/
volatile unsigned int curPos1 = 0; /*LJE 1/6/95 added unsigned*/

volatile char *msgBuff2;
 unsigned int size2,avail2;
volatile unsigned int lastPos2 = 0; /*LJE 1/6/95 added unsigned*/
volatile unsigned int curPos2 = 0; /*LJE 1/6/95 added unsigned*/

 unsigned COMADDR1;
 unsigned COMADDR2;

/*-----------------------------------------------------------------

 void far interrupt com1_int() /*LJE added far*/
{
```

```
    if ((inportb(COMADDR1+COM_STAT) & ERR_MSK) == 0) {
      /* If no error occured, then store the character */
      msgBuff1[curPos1] = ( char) inportb(COMADDR1);/*LJE added char 1/5/95*/
       if (++curPos1 >= size1) curPos1 = 0;
      avail1++;
    } else {
      /* Else, remove the erroneous character */
      inportb(COMADDR1);
    }
    outportb(0x20,0x20);                                                    150
}

 void far interrupt com2_int() /*LJE added far*/
{
    if ((inportb(COMADDR2+COM_STAT) & ERR_MSK) == 0) {
      /* If no error occured, then store the character */
      msgBuff2[curPos2] = ( char) inportb(COMADDR2);
       if (++curPos2 >= size2) curPos2 = 0;
      avail2++;
    } else {                                                                160
      /* Else, remove the erroneous character */
      inportb(COMADDR2);
    }
    outportb(0x20,0x20);
}

/*-------------------------------------------------------------------
/*This routine sends a break out port, then delays for l. Notice it is commented
out because delay doesn't work */
/*                                                                      170
void com_break(ComPort port, unsigned int l)
{
   unsigned int addr;

   addr = ((port == 2) ? COMADDR2 : COMADDR1) + COM_LCR;
   outportb(addr,inportb(addr) | 0x40); delay(l);
   outportb(addr,inportb(addr) ^ 0x40);
   outportb(addr+COM_IER,0x0B);
}
*/                                                                      180
/*-------------------------------------------------------------------
/* Return the number of available input chars on a COM port          */
/*-------------------------------------------------------------------

 int com_avail(ComPort port)
{
   switch(port) {
```

```
      case 1:  return (avail1);
      case 2:  return (avail2);
      default:  return 0;
    }
  }

/*-------------------------------------------------------------
/* Flush all input from the COM port                    */
/*-------------------------------------------------------------

 void com_flush_input(ComPort port)
{
   switch (port) {
     case 1:
     disable(); /* Disallow ints while flushing */
     avail1 = curPos1 = lastPos1 = 0;
     enable();
      break;
     case 2:
     disable(); /* Disallow ints while flushing */
     avail2 = curPos2 = lastPos2 = 0;
     enable();
      break;
     default: break;
   }
}

/*-------------------------------------------------------------
/* Reads a character from port.  If there isn't a char available it returns
COM_NONEAVAIL Otherwise it puts the character in *c and returns COM_SUCCESS
*/
 int com_read(ComPort port,  char *c)
{
   switch (port) {
     case 1:
      if (curPos1 != lastPos1) {
        *c = msgBuff1[lastPos1];
         if (++lastPos1 >= size1) lastPos1 = 0;
        disable();
        avail1--;
        enable();
         return COM_SUCCESS;
      }
      break;
     case 2:
      if (curPos2 != lastPos2) {
        *c = msgBuff2[lastPos2];
```

```
        if (++lastPos2 >= size2) lastPos2 = 0;
        disable();
        avail2--;
        enable();
        return COM_SUCCESS;
      }
      break;
    }
    return COM_NONEAVAIL;
}

/*-------------------------------------------------------------
/* Reads a character from port. If there isn't a char available it waits for
one. Otherwise it puts the character in *c and returns COM_SUCCESS */
 char com_read_wait(ComPort port)
{
     char c;
     while(com_read(port,&c) == COM_NONEAVAIL){}
     return(c);
}

/*-------------------------------------------------------------
/* Writes character out port. Actually puts it in the queue. If it CANNOT
Output the character within OUTTRIES, it returns COM_TIMEOUT. If you pass
a bad port address it returns COM_BADPORT. Success returns COM_SUCCESS. */
 int com_write(ComPort port, char c)
{
   #define OUTTRIES 10000
   unsigned int addr, t=0;
  t = 0;/*LJE 1/6/95 I never trusted the initialization in the declaration*/
   if (port == 1) addr = COMADDR1;
   else if (port == 2) addr = COMADDR2;
   else return COM_BADPORT;

   while (!(inportb(addr+COM_STAT) & 0x20) && (++t < OUTTRIES));
   if (t >= OUTTRIES) return COM_TIMEOUT;
  outportb(addr, c);
   return COM_SUCCESS;
}

/*-------------------------------------------------------------
/* This sends an entire string out. If it succeeds it returns COM_SUCCESS.
If it cannot output a character within OUTTRIES, it returns COM_TIMEOUT. If
you pass a bad port address it returns COM_BADPORT. */
 int com_send(ComPort port, char *msg)
{
   int c; /*LJE 1/6/95 used to be char and set to 0*/
```

```
    c = COM_SUCCESS; /*LJE 1/6/95 Not sure how this ever worked without this*/
    while (*msg && (c == COM_SUCCESS)) c = com_write(port,*msg++);
    return c;
}

/*-------------------------------------------------------------------
/* Enable a COM port given its I/O port address and other data      */
/*-------------------------------------------------------------------

static void doenable( int addr)                                    290
{
 /* Set DLAB bit to zero to ensure that we */
 /* access the correct COM port registers  */
 outportb(addr + COM_LCR,inportb(addr + COM_LCR) & 0x7F);

 /* Turn off the chip's interrupts to start with */
 outportb(addr + COM_IER,0);
 outportb(addr + COM_MCR,8); /* Just DTR up for now */

 /* Read status and data ports to clear any outstanding errors */  300
 inportb(addr + COM_STAT);
 inportb(addr);

 /* Set ints for data available */
 outportb(addr + COM_IER,1);
}

/*-------------------------------------------------------------------
/*This routine opens and initializes the com port, it allocates its own memory  310
for the buffer too.
com_open([1, 2]Com, Buffer Size, Baud Rate (eg. 1200),
  ['N', 'n' , 'O', 'o', 'E', 'e']Parity, [7, 8]Bits,
  [1, 2]StopBits)
Note that baud is the actual number except
    b=38 -> 38400
    b=115 -> 115200
This will return:
    COM_SUCCESS
    COM_BADBAUD
    COM_BADPARITY                                                  320
    COM_BADBITS
    COM_BADSTOP
    COM_NOMEMORY
    COM_BADPORT
    */
int com_open(ComPort port, unsigned int size, int baud, char parity, int digits,
int stop)
```

```
{
   unsigned int addr;                                                    330
   static int divisors[11] =          /* COM port baud rate divisors */
     {0x0417,0x0300,0x0180,0x00E0,0x0060,
     0x0030,0x0018,0x000C,0x0006,0x0003,
     0x0001};
   unsigned valuc,parms;
   unsigned char ch;
   int b,p,d,s;

   switch (baud) {
     case 110:  b=0;  break;                                             340
     case 150:  b=1;  break;
     case 300:  b=2;  break;
     case 600:  b=3;  break;
     case 1200: b=4;  break;
     case 2400: b=5;  break;
     case 4800: b=6;  break;
     case 9600: b=7;  break;
     case 19200:  b=8;  break;
     case 38:     b=9;  break;
     case 115: b=10;  break;                                             350
     default:  return(COM_BADBAUD);
 }
  switch (parity) {
     case 'N':
     case 'n':          p=PARITY_NONE;  break;
     case 'E':
     case 'e':          p=PARITY_EVEN;  break;
     case 'O':
     case 'o':          p=PARITY_ODD;  break;
     default:  return(COM_BADPARITY);                                    360
}
  switch (digits) {
     case 7: d=BITS_7;  break;
     case 8: d=BITS_8;  break;
     default:  return(COM_BADBITS);
}
  switch (stop) {
     case 1: s=STOP_1;  break;
     case 2: s=STOP_2;  break;
     default:  return(COM_BADSTOP);                                      370
}

  switch (port) {
     case 1:
     addr = COMADDR1 = *(( int far *)0x400L);
```

```
    avail1 = curPos1 = lastPos1 = 0;
    size1 = size;
    msgBuff1 = ( char *) malloc(size1);
     if (!msgBuff1)  return(COM_NOMEMORY);

    docnable(COMADDR1);  /* Setup the regs */

    /* Setup the ISR details */
    old_com1_int = getvect(COM_INT_1);
    setvect(COM_INT_1,com1_int);        /* attach com interrupt */
    /* Now turn on DTR, RTS and OUT2: all ready */
    outportb(COMADDR1 + COM_MCR,0x0B);

    /* Program the PIC to handle COM1 interrupts */
    ch = ( char) inportb(PIC_MASK); /* Read current mask */
    ch &= (0xFF^INT_MASK_1);/* Reset mask for COM1 */
    outportb(PIC_MASK,ch);  /* Send it to the 8259A */

     break;
    case 2:
    addr = COMADDR2 = *(( int far *)0x402L);

    avail2 = curPos2 = lastPos2 = 0;
    size2 = size;
    msgBuff2 = ( char *) malloc(size2);
     if (!msgBuff2)  return(COM_NOMEMORY);

    docnable(COMADDR2);  /* Setup the regs */

    /* Setup the ISR details */
    old_com2_int = getvect(COM_INT_2);
    setvect(COM_INT_2,com2_int);        /* attach com interrupt */

    /* Now turn on DTR, RTS and OUT2: all ready */
    outportb(COMADDR2 + COM_MCR,0x0B);

    /* Program the PIC to handle COM2 interrupts */
    ch = ( char) inportb(PIC_MASK); /* Read current mask */
    ch &= (0xFF^INT_MASK_2);/* Reset mask for COM2 */
    outportb(PIC_MASK,ch);  /* Send it to the 8259A */

     break;

   default:  return(COM_BADPORT);
}
```

```
    parms = b | p | d | s;

    valuc = (parms & BITS_8) ? 0x03 : 0x02;
     if (parms & STOP_2) valuc |= 0x04;
     if (parms & PARITY_ODD) valuc |= 0x08;
     if (parms & PARITY_EVEN) valuc |= 0x18;

    outportb (addr + COM_LCR,valuc | 0x80);  /* Set parms and DLAB */
    outport  (addr,divisors[parms & 0x0F]);  /* Set the baud rate */
    outportb (addr + COM_LCR,valuc);   /* Clear DLAB bit */

     return(COM_SUCCESS);
}

/*-----------------------------------------------------------------
/* This closes the port and frees the buffer. If leavedtr == 0 it clears dtr
otherwise it sets it. */
 void com_closc(ComPort port,  char leavedtr)
{
    char ch;

    switch(port) {
      case 1:
      curPos1 = 0; lastPos1 = 0;
      ch = ( char) inportb(PIC_MASK); /* Get 8259A (PIC) Mask */
      ch |= INT_MASK_1; /* Set Interrupt Mask COM1 */
      outportb(PIC_MASK,ch);  /* Write int. mask to 8259A*/
      /* Clear the interrupt enable register */
      outportb(COMADDR1 + COM_IER,0);
      /* Clear OUT2, and set DTR as required */
       if (leavedtr)
        outportb(COMADDR1 + COM_MCR,1);
       else
        outportb(COMADDR1 + COM_MCR,0);
      setvect(COM_INT_1,old_com1_int);      /* Restore COM1 int */
      free(( void*)msgBuff1); /*LJE added void* to avoid warning*/
       break;
      case 2:
      curPos2 = 0; lastPos2 = 0;
      ch = ( char) inportb(PIC_MASK); /* Get 8259A (PIC) Mask */
      ch |= INT_MASK_2; /* Set Interrupt Mask COM1 */
      outportb(PIC_MASK,ch);  /* Write int. mask to 8259A*/
      /* Clear the interrupt enable register */
      outportb(COMADDR2 + COM_IER,0);
      /* Clear OUT2, and set DTR as required */
       if (leavedtr)
        outportb(COMADDR2 + COM_MCR,1);
```

```
      else
        outportb(COMADDR2 + COM_MCR,0);
      setvect(COM_INT_2,old_com2_int);      /* Restore COM2 int */
      free(( void*)msgBuff2); /*LJE added void* to avoid warning*/
       break;
      default:  break;
    }
}

/*-------------------------------------------------------------
/* Test to see if a carrier is present                        */
/* Returns True (non zero) if carrier is present, False (zero) otherwise
*/
/*-------------------------------------------------------------

int  com_carrier(ComPort port)
{
  switch (port)
   {
    case 1:  return ((inportb (COMADDR1 + COM_MSR) & 0x80) != 0);
    case 2:  return ((inportb (COMADDR2 + COM_MSR) & 0x80) != 0);
    default:  return COM_FALSE;
  }
}

/*-------------------------------------------------------------
/* Test to see if the DSR (Data Set Ready) signal is present  */
/* Returns True (non zero) if DSR is present, False (zero) otherwise  */
/*-------------------------------------------------------------

int  com_dsr(ComPort port)
{
  switch (port)
   {
    case 1:  return ((inportb (COMADDR1 + COM_MSR) & 0x20) != 0);
    case 2:  return ((inportb (COMADDR2 + COM_MSR) & 0x20) != 0);
    default:  return COM_FALSE;
  }
}

/*-------------------------------------------------------------
/* Test to see if the COM port is ready for a new             */
/* character to transmit.                                     */
/* Returns True (non zero) if PORT is READY, False (zero) otherwise  */
/*-------------------------------------------------------------

int  com_ready(ComPort port)
```

```
{
  switch (port)
   {
    case 1:  return ((inportb (COMADDR1 + COM_STAT) & 0x20) != 0);
    case 2:  return ((inportb (COMADDR2 + COM_STAT) & 0x20) != 0);
     default:  return COM_FALSE;
   }
}

/*-------------------------------------------------------------------
/* Drop the DTR signal on a COM port                              */
/*-------------------------------------------------------------------

 void com_dropdtr(ComPort port)
{
  switch (port)
   {
    case 1: outportb (COMADDR1 + COM_MCR,0x0A ); break;
    case 2: outportb (COMADDR2 + COM_MCR,0x0A ); break;
  }
}

/*-------------------------------------------------------------------
/* Raise the DTR signal on a COM port                             */
/*-------------------------------------------------------------------

 void com_raisedtr(ComPort port)
{
  switch (port)
   {
    case 1: outportb (COMADDR1 + COM_MCR,0x0B ); break;
    case 2: outportb (COMADDR2 + COM_MCR,0x0B ); break;
  }
}

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

1.7 MAKEFILES, Directories and Other Files

The "main" programs (those listed in this chapter) are kept in a directory together. They are maintained using the following makefile.

1.7.1 Make File Listing:

```
a : testhist.exe

testhist.exe : testhist.obj messages.obj modlib\targa.lib makefile com3_2.obj
 link /st:27000 /CO /E /NOI /NOLOGO testhist+messages+irqsetup+com3_2,,,targraf+ltplib+grapl

init_vid.exe : init_vid.obj modlib\targa.lib makefile
 link /st:27000 /CO /E /NOI /NOLOGO init_vid+messages+irqsetup,,,targraf+ltplib+graphics+moc

init_vid.obj : init_vid.c targa.h
 cl /nologo /W4 /Fs /Zi /c /AL init_vid.c

testhist.obj : testhist.c targa.h com3_2.h
 cl /nologo /W3 /Fs /Zi /c /AL testhist.c

messages.obj: messages.c targa.h
 cl /AL /nologo /c /W3 /Fs /Zi messages.c

user.obj : user.c user.h targa.h
 cl /AL /nologo /c /W3 /Fs /Zi user.c

com3_2.obj : com3_2.c com3_2.h
 cl /nologo /W3 /Fs /Zi /c /AL com3_2.c
```

The library is located in a subdirectory of the main files and are maintained via makefile, makelib and makedoc.bat files. The library's makefile does the following:

1. Compiles the .c files using options: /W4 (warnings), /Fs (listing), /Zi (code view debugger), /c (compile only), /AL (large memory model, compatible with Targa).

2. Puts the .obj files in a library. Uses makelib for a listing of the .obj to put in the library.

3. Calls makedoc.bat to build the targa.h file.

4. Puts targa.h in the PARENT directory of the library (this would be the directory containing the main routines). The idea is your main program etc. goes in some directory (call it c:/X) and the TRAC library goes in c:/X/lib. Targa.h (which the makefile deposits in c:/X) contains all of the .h files so your main program need only include a single .h file.

The makefile is processed with the command: nmake(Enter). The library's makefile, makelib and makedoc.bat files follow. Watch out for the following. In the line that compiles targa8.c there is an option DTAMU this says the file is being compiled at TAMU (version 6 of MSC). There are some version incompatibilities we noticed. If DTAMU is used, the code compiles assuming version 6. Otherwise it assumes version 5.

1.7.2 Make File Listing:

```
#The following are in the library
#removed geometry to use newgeo never compiled since multiple leds
#targa.lib: makelib PLOT.OBJ MEMORY.OBJ geometry.OBJ MISC.OBJ POSE.OBJ TARGA8.OBJ
targa.lib: makelib PLOT.OBJ MEMORY.OBJ MISC.OBJ POSE.OBJ TARGA8.OBJ BLOB.OBJ DIP.
 del targa.lib
 lib @makelib
 makedoc.bat

blob.obj : blob.c datatype.h blob.h targa8.h memory.h misc.h
 cl /nologo /W4 /Fs /Zi /c /AL blob.c

dip.obj : dip.c datatype.h plot.h dip.h target.h
 cl /nologo /W4 /Fs /Zi /c /AL dip.c

geometry.obj : geometry.c datatype.h geometry.h memory.h
 cl /nologo /W4 /Fs /Zi /c /AL geometry.c

memory.obj : memory.c memory.h
 cl /nologo /W4 /Fs /Zi /c /AL memory.c
```

```
MISC.obj : MISC.c misc.h datatype.h
 cl /nologo /W4 /Fs /Zi /c /AL MISC.c

PLOT.obj : PLOT.c plot.h datatype.h
 cl /nologo /W4 /Fs /Zi /c /AL PLOT.c

posc.obj : posc.h datatype.h posc.c misc.h
 cl /nologo /W4 /Fs /Zi /c /AL posc.c

TARGA8.obj : TARGA8.c targa8.h datatype.h
 cl /DTAMU /nologo /W4 /Fs /Zi /c /AL TARGA8.c

TARGET.obj : TARGET.c datatype.h targa8.h target.h misc.h
 cl /nologo /W4 /Fs /Zi /c /AL TARGET.c

targutil.obj : targutil.c targutil.h datatype.h targa8.h misc.h
 cl /nologo /W4 /Fs /Zi /c /AL targutil.c

accum.obj : accum.c datatype.h accum.h targa8.h
 cl /nologo /W4 /Fs /Zi /c /AL Accum.c

roi.obj : roi.c datatype.h roi.h TARGA8.h TARGET.h PLOT.h MISC.h DIP.h
 cl /nologo /W4 /Fs /Zi /c /AL roi.c
```

1.7.3 Make File Listing:

```
targa.lib
Y
+PLOT.OBJ     &
+MEMORY.OBJ   &
+MISC.OBJ     &
+POSE.OBJ     &
+TARGA8.OBJ   &
+BLOB.OBJ     &
+DIP.OBJ      &
+TARGUTIL.OBJ &
+GEOMETRY.OBJ &
+ACCUM.OBJ &
+ROI.OBJ &
+TARGET.OBJ   ;
```

1.7.4 Batch File Listing:

```
rename datatype.h datatype.hh
copy *.h lib.hh
rename datatype.hh datatype.h
copy datatype.h lib.h
type lib.hh >> lib.h
copy lib.h ..\targa.h
del lib.hh
del lib.h
```

Chapter 2

accum.c, accum.h

Documentation Date: 3/9/95

2.1 Accumulating Images To Build Signal

The routines in this file accumulate the signal from a series of images to build the signal. They are used when there is low light levels returning such as in long distance applications.
New Data Types:
None.
Definitions: None.

2.1.1 Header File Listing:

```
#ifndef ACCUM_H
    #define ACCUM_H
    long NewDoAccumulation( int on, int off, int accum, int accumshow);
    void DoAccumulation( int on, int off, int accum);
    #endif
```

2.2 Function: NewDoAccumulation

Documentation Date: 3/9/95

Prototypes:

```
long NewDoAccumulation(int on, int off, int accum, int accumshow);
```

Source File: *accum.c*
Type of Function: User Callable
Header Files Used in accum.c: *"datatype.h" "targa8.h" "accum.h"*

Description:

This routine takes four image numbers. Image numbers on and off are obvious. Frame accum is a temporary image that holds the history of bright pixels. Image accumshow is the accumulated image. Don't display accum, it will look weird. Accumshow can be displayed. Basically the on and off are subtracted, any subtracted pixel that has intensity greater than 1 gets its subtracted intensity added to the corresponding pixel in accumshow. Now, the history is tested for the pixel. If the pixel had a subtracted intensity greater than 1 in any three of the previous four processing steps. Then the accumulated pixel in accumshow is left alone. If the pixel was not "bright enough" in three of the four previous steps, the pixel in accumshow is set to zero (black). The return value is the number of saturated pixels in accumshow. Accumulations greater than 255 (white) are chopped back to 255.

2.3 Function: DoAccumulation

Documentation Date: 3/9/95

Prototypes:

```
void DoAccumulation(int on, int off, int accum);
```

Source File: *accum.c*
Type of Function: User Callable
Header Files Used in accum.c: *"datatype.h" "targa8.h" "accum.h"*

Description:

This routine takes three image numbers. Image numbers on and off are obvious. Frame accum is a temporary image that holds the accumulated image. Accum can be displayed. Basically the on and off are subtracted, any subtracted pixel that has intensity greater than 0 gets its subtracted intensity added to the corresponding pixel in accum. Accumulations greater than 255 (white) are chopped back to 255.

2.3.1 Program Listing:

```
#include "datatype.h"
#include "targa8.h"
#include "accum.h"

ImageLine onLine, offLine;
ImageLine accumLine;
void DoAccumulation( int on, int off, int accum)
{
 int i,j;
 long temp;
 for(i=0;i<IMAGEHEIGHT;i++){
  GetLine(i, on, onLine);
  GetLine(i, off, offLine);
  GetLine(i, accum, accumLine);
   for(j=0;j<IMAGEWIDTH;j++){
    temp = (( long) onLine[j] - ( long) offLine[j]);
     if (temp > 0)temp = ( long) accumLine[j] + temp;
     else temp = 0;
     if(temp > 255) temp = 255;
    accumLine[j] = (Pixel) temp;
    }
  PutLine(i, accum, accumLine);
  }
}

ImageLine testaccumLine;
long NewDoAccumulation( int on, int off, int testaccum, int accum)
{
 int i,j;
 long temp,temp2;
 long numof255s;
numof255s = 0;
 for(i=0;i<IMAGEHEIGHT;i++){
  GetLine(i, on, onLine);        //Single line from "on" frame
  GetLine(i, off, offLine);      //Single line from "off" frame
  GetLine(i, accum, accumLine);  //Single line from accumulation frame
  GetLine(i, testaccum, testaccumLine);  //Single line from "history" frame
   for(j=0;j<IMAGEWIDTH;j++){
    temp = (( long) onLine[j] - ( long) offLine[j]);  //Difference of on and
off (i,j) pixel
     if (temp > 1){      //If difference is 2 or more gray levels
      temp2 = ( long) accumLine[j] + temp;   //Add difference of pixels to accumulation
      temp = (( long) testaccumLine[j])/2;   //Shift bits in history line
      temp = 128 + temp;    //Turn on first bit in pixel history
      }
     else {      //Difference is less than 1 gray level
```

```
        temp = (( long) testaccumline[j])/2; //No difference, shift bits in history
line
        temp2= ( long) accumline[j]; //Does not change, added 1/13/95
        }                                                                    50
       if(temp > 255) temp = 255;        //truncate history frame (necessary?)
       if(temp2 > 255){ //Truncate at brightest level
        temp2 = 255;
        numof255s++; //Count number of brightest pixels
        }
      accumline[j] = (Pixel) temp2;              //Set permanent accumulation line
      testaccumline[j] = (Pixel) temp; //Set permanent history line
      temp = temp / 16;          //Temp is 8 bits, look at first 4 bits
//pixel on in 3 of last 4 subtractions
//     if( !(((temp>12) && (temp<16)) || (temp==11) || (temp==7)) ) accumline[j]
= (Pixel) 0;
//0110,0111,1011,1100,1101,1110,1111 original test for good pixel
      if( !(((temp>10) && (temp<16)) || (temp==6) || (temp==7)) ) accumline[j]
= (Pixel) 0;
     }
   PutLine(i, accum, accumline);          //Put accumulation line back in frame
   PutLine(i, testaccum, testaccumline);          //Put history line back in frame
   }
 return numof255s;
}                                                                            70
```

Chapter 3

blob.c, blob.h

Documentation Date: 6/27/94

3.1 Perform Blob Analysis.

The one user callable routine in this file subtracts images and performs a blob analysis.
New Data Types:

- typedef long Bsize;
- typedef DPixel Tag; /*A name for the blob*/
- typedef long DoWa;
- struct BlobStruct { Tag tag; /*Its name*/ char active; /*Active blobs are still growing*/ Bsize size; /*Number of Pixels*/ Bsize rsum; /*Row number sum*/ Bsize csum; /*Column number sum*/ Bsize totalbright; /*Sum of all pixels*/ DoWa totalbrightsq; /*Sum of all pixels sq.*/ long numwraps; struct BlobStruct *previous; struct BlobStruct *next; };
- typedef struct BlobStruct BLOBSTRUC;

Definitions:

- BIGINT = 2000000000
- IsBright(pixel,NegThreshold,PosThreshold) = ((pixel<NegThreshold) || (pixel>PosThreshold))
- LeftNeighbor(row,column,start,resol) = ((column>start)?(row[column-resol]):start)
- UpperNeighbor(row,column) = (row[column])

- NoSubtract = 1
- SingleSubtract = 2
- DoubleSubtract = 3
- SameSign(a,b) = (((a > 0) && (b > 0)) || ((a < 0) && (b < 0)))

3.1.1 Header File Listing:

```
#ifndef BLOB_H
    #define BLOB_H
/*This is intended for the Targa board */
/*blob analysis for page1 - page2*/
/*Description:
 posblobs have intensity above PosThreshold
 negblobs have intensity below NegThreshold
 on return Neg(pos)numbertofind = actual number found
 If StoragePage is nonnegative StoragePage contains the absolute value of subtracted
pages
 The blobs are stored in posblobs and negblobs
 If SecondOn >=0 then double subtract
    Threshold of FirstOn - Dim and Threshold of SecondOn - Dim
 If Off < 0 then no subtract
 Else Threshold of FirstOn - Off
  */
int ExtractBlobs( int FirstOn, int Off, int SecondOn, int StoragePage, int *NegNumberToFind,
DPixel NegThreshold, OneBlob *negblobs, int *PosNumberToFind, DPixel PosThreshold,
OneBlob *posblobs, OneBlob *background, long LargestToFind, long SmallestToFind,
 ROI roi);
/*Errors: 0 is success */
    #define ERROR_CHECKING
    #define BAD_XS 1
    #define BAD_XE 2
    #define BAD_YS 3
    #define BAD_YE 4
    #define BAD_RESOLUTION 5
    #endif
```

3.2 Function: ExtractBlobs

Documentation Date: 6/24/94

Prototypes:

```
int ExtractBlobs(int FirstOn, int Off, int SecondOn, int StoragePage,
int *NegNumberToFind, DPixel NegThreshold, OneBlob *negblobs, int
*PosNumberToFind, DPixel PosThreshold, OneBlob *posblobs, OneBlob
*background, long LargestToFind, long SmallestToFind, ROI roi)
```

Source File: *blob.c*
Type of Function: User Callable
Header Files Used in blob.c: *<stdlib.h> <stdio.h> <math.h> "datatype.h" "targa8.h" "blob.h" "memory.h" "misc.h"*

Description:

This routine performs a single or double subtract on images and determines the blobs present in the resulting image. It can find both positive and negative blobs simultaneously. A positive blob is one with intensity > 0 after subtraction. A negative blob has intensity < 0 after subtraction. The roi is a Region of Interest and determines the subset of the image to search.

FirstOn, Off, SecondOn and StoragePage are image numbers. To perform blob analysis without subtraction set Off and SecondOn to -1. To perform single subtraction set SecondOn = -1. If you do not want to see the image as it is being processed set StoragePage = -1. The only TARGA image memory that is modified is the image located at StoragePage. It is possible to perform the blob analysis without destroying any image data, just set StoragePage = -1. When "storing" (displaying) the results as they are processed (when StoragePage > -1), the routine shows:

1. the FirstOn frame (when Off = -1) line by line as the blobs are extracted; or

2. FirstOn - Off line by line.

PosNumberToFind (NegNumberToFind) is the number of positive (negative) blobs to find. Either one can be zero. The routine first finds all of the blobs, then it picks out the PosNumberToFind (NegNumberToFind) largest positive (negative) blobs and ignores the rest. You do not save much time by setting these to small values since the routine first finds all blobs then selects them.

PosThreshold (NegThreshold) is a positive (negative) number that indicates whether a pixel belongs in a blob. The test applied is: "A pixel is Bright (should be in a blob) IF its intensity < NegThreshold OR its intensity > PosThreshold". This test is applied differently for each subtraction algorithm as follows:

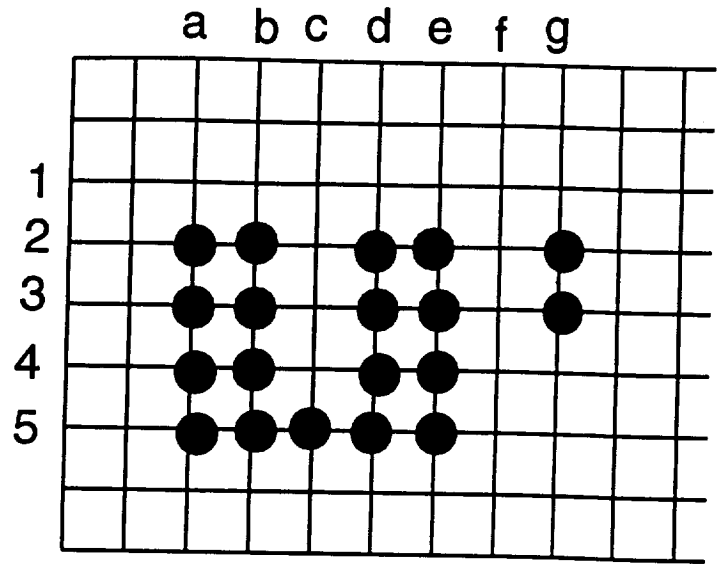

Figure 3.1: A U Shaped Blob and a Small Piece of "Noise".

1. For No Subtraction: Is FirstOn pixel intensity bright?
2. For Single Subtraction: Is FirstOn - Off pixel intensity bright?
3. For Double Subtraction: Is FirstOn - Off pixel intensity bright AND is SecondOn - Off pixel intensity bright AND are they both positive or both negative? If so, their average intensity is used in the blob.

If you do not want to find negative blobs and you want the algorithm to execute quickly, you should set NegThreshold -255 so nothing negative will pass the brightness test.

The background is a blob of everything that does not pass the threshold test. LargestToFind and SmallestToFind are size limits (in pixels) for the blobs. It is applied after each line is processed. All blobs that have stopped growing are tested to see if they pass the size restrictions. If they fail, they are put in the background and deleted.

The roi is a region of interest. Its data structure contains the starting and ending rows and columns of a rectangle to search and the resolution to use in the search. If resolution is 1, the routine searches every row and column in the roi. At resolution 2, every second row and column are searched.

3.2.1 Theory

Figure 3.1 shows a u shaped blob in an image and a one pixel piece of noise. The horizontal lines indicate rows in the image. Numbers on the left indicate row numbers. The dots indicate pixels in the image. The a, b, c ... on the top indicate columns in the image. This figure will be used to describe the method used to find blobs.

To find blobs, the algorithm keeps only two lines of image in memory. One line is the one currently being processed (call it the current line), the second line is the one just previous to the current (call it the previous line). The previous data line does not actually contain image data. As the explanation proceeds it will become clear what the data is. When the algorithm starts, previous is set to zero. Since the algorithm implements a resolution (in the roi) the previous and current lines may be separated. For example, if resolution is 2 and previous is line 3 then current is line 5. Also note that image subtraction (if performed) is done "on the fly" so in actuality, the routine may use one, two or three lines of data for current depending on whether no, single or double subtraction is being performed. To understand blobbing however just imagine that a subtraction has already been performed and we have only two lines of data to look at.

When processing line 1 (see figure 3.1), no pixels pass the brightness test because none of them contain part of the u shape. When line 1 finishes, we have no blobs and previous is still all zero. As we process line 2 as current, pixel 2,a is found to be bright. The algorithm looks to the left of pixel 2,a and above 2,a to see if those pixels were also bright (how this is done will become clear when 2,b is discussed). Since they were not bright, pixel 2,a is a member of a new blob. A blob is created, given a tag or name (1 for example) and pixel 2,a is added to the new blob (see routine PutItInBlob). After adding 2,a to the blob named 1, the number 1 is stored in 1,a (previous line, column a).[1] When 2,a is added to blob 1, the blob is set active (growing). Next pixel 2,b is tested and found to be bright. The routine looks left (actually in previous,a) and finds it non-zero (it is 1) which says pixel 2,b is a member of blob 1. Pixel 2,b is added to blob 1 and 1 is stored in previous,b. Pixel 2,c is found dim so it is ignored hence previous,c is left zero. Pixel 2,d is tested and found bright, previous,c is zero so 2,d is considered a member of a new blob (call it blob 2). A blob is created and 2,d is added to it. Blob 2 is made active (growing) and 2 (the blob pixel 2,d belongs to) is stored in previous,d. Pixel 2,e is added to blob 2. Pixel 2,g becomes a member of blob 3.

At the end of the line, the routine looks for any inactive blobs. All three blobs are active (growing) so we set them all inactive and proceed to process line 3. When 3,a is found bright, we look left (nothing) and up (look at previous,a). Since previous,a is 1, pixel 3,a is added to blob 1 and previous,a is again set to 1 (no change actually). Since blob 1 had something added, it is set active (growing). Eventually we get to 3,d which is added to blob 2 and blob 2 is set active. After processing 3,e no other pixels are bright so we are at the end of the line. At the end we look for inactive blobs and find blob 3. Since blob 3 has stopped growing it is tested against the size limits, if it passes it is kept. In fact, if blob 3 manages to pass the size tests, it will be retested after every line. This is a waste of time but makes the logic easier.

[1] Actually previous,a is tested before 1 is stored in it. It will soon become clear why the test is performed first.

The only other interesting consideration is when pixel 5,c and 5,d are encountered. When 5,c is found bright and previous,b is found equal to 1, pixel 5,c is added to blob 1 and previous,c is set to 1 (standard procedure). Now when 5,d is found bright and previous,c is 1, it is added to blob 1. Now before previous,d is changed to 1 it is tested (standard procedure). When previous,d is discovered to equal 2, the algorithm realizes that blobs 1 and 2 are actually the same. The routine adds blob 2 to blob 1 (see JoinBlobs) then replaces all values of 2 in previous to 1.

3.3 Function: FindBlobPointer

Documentation Date: 6/24/94

Prototypes:

`BLOBSTRU *FindBlobPointer(Tag tag)`
Source File: *blob.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in blob.c: *<stdlib.h> <stdio.h> <math.h> "datatype.h" "targa8.h" "blob.h" "memory.h" "misc.h"*

Description:

The BLOBSTRU is only used in blob.c, it is a linked list of blobs. The list is created by tacking new blobs onto the end of the list. The tag is basically a Long Int number that numbers the blob. This routine searches the linked list looking the the blob with a tag equal to what is passed to it. If it cannot find one, it returns NULL. If it finds the correct blob, it returns its address.

3.3.1 Theory

This routine can certainly be improved. For example, most times we search for the blob most recently added to the list (at the end), yet we begin the search at the beginning. In addition, this could be eliminated with maybe a hashing table (I think that is what it is called).

3.4 Function: MakeAllBlobsInActive

Documentation Date: 6/24/94

Prototypes:

`void MakeAllBlobsInActive()`
Source File: *blob.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in blob.c: *<stdlib.h> <stdio.h> <math.h> "datatype.h" "targa8.h" "blob.h" "memory.h" "misc.h"*

Description:

This routine looks through the BLOBSTRUCT linked list and sets all blobs as inactive. This indicates that their size is no longer changing.

3.4.1 Theory

Whenever a pixel is added to a blob, the blob is set active which means it is growing. After a line has been processed, all inactive blobs are tested to see if they pass the minimum and maximum size limits then they are kept. If they fail the tests they are deleted. This is done to reduce the amount of memory used since there may be a large number of very small blobs. After testing for active and inactive blobs, all of them are set inactive. In this way when processing on the next line is finished, the only active blobs will be the ones that have grown during that line.

3.5 Function: PutItInBlob

Documentation Date: 6/24/94

Prototypes:

```
void PutItInBlob(DPixel *pixel, int row, int column, DPixel gray,
Tag tag)
```

Source File: *blob.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in blob.c: *<stdlib.h> <stdio.h> <math.h> "datatype.h" "targa8.h" "blob.h" "memory.h" "misc.h"*

Description:

Every pixel that passes the threshold test is added to a blob. The routine receives a pointer to the pixel, the pixel's row and column coordinates on the screen, its intensity and the name (tag) of the blob that the pixel belongs to. The routine increments the blob named tag by the values sent, sets it active, stores tag where the pixel used to be then returns.

3.5.1 Theory

This routine helps to compute the following items for every blob:

1. Area,
2. Row and column centroid,
3. Average brightness,
4. Standard deviation of brightness (actually it is the variance).

Area is computed as the number of pixels in the blob.

$$A = \sum_{i=1}^{N_b}(1)$$

Here A is area, N_b is the number of pixels in blob b. Row centroid is computed as:

$$R = \frac{\sum_{i=1}^{N_b} r_i}{\sum_{i=1}^{N_b}(1)}$$

The term r_i is the row position of the ith pixel in blob b. Column centroid is computed similarly. Average brightness is:

$$B = \frac{\sum_{i=1}^{N_b} I_i}{\sum_{i=1}^{N_b}(1)}$$

The brightness deviation is:

$$D = \frac{\sum_{i=1}^{N_b}(I_i - B)}{\sum_{i=1}^{N_b}(1)}$$

$$D = \frac{\sum_{i=1}^{N_b} I_i^2}{A} - B^2$$

In this routine, only the sums are computed. The other calculations are performed in CopyToBlobArray.

When a pixel in added to a blob the blob is made active. To understand why this is done, see routine MakeAllBlobsInactive. To assist in processing the next image line, the pixel intensity is replaced by the blob tag, see routine BlobsInOneRow.

3.6 Function: Miscellaneous Support Routines in Blob.c

Documentation Date: 6/27/94

Prototypes:

```
void KillAllBlobs()
   void DumpAllBlobsIntoBackKill(void)
   void CopyToBlobArray(BLOBSTRU * blob, OneBlob array[], int whichone,
int resolution)
   BLOBSTRU * GetLargestPosBlob()
   BLOBSTRU * GetLargestNegBlob()
   Tag NewBlob()
   void BlobsInOneRow(int SubtractType, int rownum, int xs, int xe,
int resolution, ImageLine one, ImageLine two, ImageLine three, DImageLine
neighbors, DPixel NegThreshold, DPixel PosThreshold)
   void JoinBlobs(DImageLine row, Tag parentnum, Tag childnum)
   void DumpBlobIntoBackgroundThenKill(BLOBSTRU *blob)
   void DumpSmallInactBackKill(Bsize smallsize)
   void DumpSmallBigInactBackKill(Bsize smallsize, Bsize largesize)
   void KillBlob(BLOBSTRU *blob)
```

Source File: *blob.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in blob.c: *<stdlib.h> <stdio.h> <math.h> "datatype.h" "targa8.h" "blob.h" "memory.h" "misc.h"*

Description:

- KillAllBlobs - Killing a blob means freeing the memory it uses and piecing the linked list back together. This kills all blobs.
- DumpAllBlobsIntoBackKill - This routine joins a blob to the background blob then kills the blob.
- CopyToBlobArray - When we are finished finding blobs, we copy the linked list into the return structure. It also computes centroids etc. using the sums collected during blobbing.
- GetLargestPosBlob - Returns the address of the largest positive blob in the linked list.
- GetLargestNegBlob - Returns the address of largest negative blob in the linked list.
- NewBlob - Allocate memory for a new blob structure. Fit it into the linked list. Returns its tag.

- BlobsInOneRow - For a row rownum, check the bright pixels to see which blob they belong to.
- JoinBlobs - JoinBlobs join two blobs together then kill the child.
- DumpBlobIntoBackgroundThenKill - Join a blob to the background blob, then kill the blob.
- DumpSmallInactBackKill - This walks the blob linkage and kills all inactive blobs that are too small.
- DumpSmallBigInactBackKill - Join blobs that are too small or too big to the background blob, then kill the blob.
- KillBlob - To Kill a blob means to remove it from the linked list and free its memory.

3.6.1 Program Listing:

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include "datatype.h"
#include VISION_BOARD
#include "blob.h"
#include "memory.h"
#include "misc.h"

/************************************
Local Prototypes and Definitions                                    10
************************************/
/*A Bright pixel is < NegThreshold or > PosThreshold */
#define IsBright(pixel,NegThreshold,PosThreshold) ((pixel<NegThreshold) || (pixel>PosThreshol
#define LeftNeighbor(row,column,start,resol) ((column>start)?(row[column-resol]):start)
#define UpperNeighbor(row,column) (row[column])
/*This is a linked list of blobs */
typedef long Bsize;
typedef DPixel Tag;          /*A name for the blob*/
typedef long DoWa;                                                  20
#define BIGINT 2000000000 /*if DoWa is a long*/
struct BlobStruct {
    Tag tag;              /*Its name*/
    char active;              /*Active blobs are still growing*/
    Bsize size;           /*Number of Pixels*/
    Bsize rsum;           /*Row number sum*/
    Bsize csum;           /*Column number sum*/
    Bsize totalbright;        /*Sum of all pixels*/
    DoWa totalbrightsq;       /*Sum of all pixels^2*/
    long numwraps;                                                  30
    struct BlobStruct *previous;
    struct BlobStruct *next;
   };
typedef struct BlobStruct BLOBSTRU;

void DumpBlobIntoBackgroundThenKill(BLOBSTRU *blob);
void DumpAllBlobsIntoBackKill( void);
/*The tag is a number designating the blob */
Tag NewBlob( void);
/*JoinBlobs join two blobs together then kill the child*/             40
void JoinBlobs(DImageLine row, Tag parent, Tag child);
/*This puts information into blob tagged tag, then stores tag in the pixel at
row, col*/
void PutItInBlob(DPixel *tagptr, int row, int col, DPixel pixel, Tag tag);
/*MakeAllBlobsInActive This walks the blob linkage and makes all of them inactive.
this means they have finished growing*/
```

```
 void MakeAllBlobsInActive( void);
 void KillAllBlobs( void);
 void CopyToBlobArray(BLOBSTRU * blob, OneBlob array[], int whichone, int resolution);
BLOBSTRU * GetLargestPosBlob( void);
BLOBSTRU * GetLargestNegBlob( void);
Tag NewBlob();
/*DumpSmallInactiveIntoBackgroundThenKill This walks the blob linkage and kills
all inactive blobs that are too small */
 void DumpSmallInactBackKill(Bsize small);
 void DumpSmallBigInactBackKill(Bsize smallsize, Bsize largesize);
 void KillSmallandBigInactive(Bsize SmallestToFind, Bsize LargestToFind);
 void MakeAllBlobsInActive();
/*To Kill a blob means to remove it from the linked list and free its memory*/
 void KillBlob(BLOBSTRU *blob);
 void BlobsInOneRow( int SubtractType, int rownum, int xs, int xe, int resolution,
ImageLine one, ImageLine two, ImageLine three, DImageLine neighbors, DPixel
NegThreshold, DPixel PosThreshold);
/*FindBlobPointer looks for a particular blob tag and returns a pointer to the
tagged blob. NULL is a blob with tag is not found*/
BLOBSTRU *FindBlobPointer(Tag tag);

/**********************************
Local Data
**********************************/
BLOBSTRU *FirstBlob;
Tag BlobTag;
BLOBSTRU BackGround;

 #define NoSubtract 1
 #define SingleSubtract 2
 #define DoubleSubtract 3

/*This routine performs a single or double subtract on images and
determines the blobs present in the resulting image. It can find
positive blobs (those with intensity >0 after subtraction) and
negative blobs (intensity < 0 after sub.). The roi determines the region
of interest to search in.
    FirstOn, Off, SecondOn and StoragePage are image numbers.
    NegNumberToFind, NegThreshold and negblobs control finding the
negative blobs.
    PosNum... control finding the positive blobs.
    background is a blob of everything that does not pass the thresholds.
    LargestToFind, Small... and roi are sizes and region of interest.
*/
 int ExtractBlobs( int FirstOn, int Off, int SecondOn, int StoragePage,
    int *NegNumberToFind, DPixel NegThreshold, OneBlob *negblobs,
    int *PosNumberToFind, DPixel PosThreshold, OneBlob *posblobs,
```

```
    OneBlob *background, long LargestToFind, long SmallestToFind,
    ROI roi)
  {
  ImageLine *imline;
  ImageLine *one, *two, *three;
   int i,r,c;
   int SubtractType;                                                    100
  BLOBSTRU *TempBlob;
   static DImageLine *neighbors;
   #ifdef ERROR_CHECKING
      if( (roi.xs < 0) || (roi.xs > IMAGEWIDTH)) return(BAD_XS);
      if( (roi.xe <= roi.xs) || (roi.xe > IMAGEWIDTH)) return(BAD_XE);
      if( (roi.ys < 0) || (roi.ys > IMAGEHEIGHT)) return(BAD_YS);
      if( (roi.ye <= roi.ys) || (roi.ye > IMAGEHEIGHT)) return(BAD_YE);
      if( (roi.resolution < 1) || (roi.resolution > IMAGEWIDTH) ) return(BAD_RESOLUTION);
      if( (roi.resolution > (roi.xe - roi.xs)) || (roi.resolution > (roi.ye-roi.ys))) return(BAD_RESO
      #endif                                                            110

  neighbors = (DImageLine *) MALLOC( sizeof(DImageLine));
  one = (ImageLine *) MALLOC( sizeof(ImageLine));
   if( (neighbors == NULL) || (one == NULL) )
    Fatal_Error_Message("Out of Memory in Beginning of Blob.");

  /*Determine method of subtraction. Possibilities are:
    1. Don't subtract, use FirstOn image (Selected if Off < 0)
    2. Single subtract, use FirstOn - Off (Select if SecondOn < 0)
    3. Double subtract, use FirstOn - Off anded with SecondOn - Off     120
        (both subtractions must pass the threshold test) */
  SubtractType = NoSubtract;
   if(Off >= 0){
    SubtractType = SingleSubtract;
    two = (ImageLine *) MALLOC( sizeof(ImageLine));
     if( (two == NULL) ) Fatal_Error_Message("Out of Memory in Beginning of Blob.");
     if(SecondOn >= 0){
        SubtractType = DoubleSubtract;
        three = (ImageLine *) MALLOC( sizeof(ImageLine));
         if( (three == NULL) ) Fatal_Error_Message("Out of Memory in Beginning of Blob.");
        }
    }

  /*If a non-negative storage page is given then show the image while
processing*/
   if(StoragePage > -1) {
    imline = (ImageLine *) MALLOC( sizeof(ImageLine));
     if( (imline == NULL) ) Fatal_Error_Message("Out of Memory in Beginning
of Blob.");
    }                                                                   140
```

```
  LargestToFind = LargestToFind / (roi.resolution * roi.resolution);
  SmallestToFind = SmallestToFind / (roi.resolution * roi.resolution);

  /*The blobs are put in a linked list for this routine only.
Initialize list*/
  FirstBlob = NULL;
  BlobTag = 0;
  BackGround.size = (Bsize) 0;
  BackGround.rsum = (Bsize) 0;
  BackGround.csum = (Bsize) 0;
  BackGround.totalbright = (Bsize) 0;
  /*The following allows extra long ints*/
  BackGround.totalbrightsq = (DoWa) 0;
  BackGround.numwraps = 0;

 /*Prepare to process first line*/
  if(StoragePage == -1)Status_Message("Processing");
  for(i=0;i<IMAGEWIDTH;i++)(*neighbors)[i]=0;

 /*Process each image line*/
  for(r=roi.ys;r<roi.ye;r=r+roi.resolution){
   /*First get each line*/
   /*GetLine and PutLine are device (TARGA) dependent*/
    switch (SubtractType){
       case DoubleSubtract:
        /*This says copy line r from SecondOn image
            into array three*/
        GetLine(r,SecondOn,*three);
       case SingleSubtract:
        GetLine(r,Off,*two);
       case NoSubtract:
        GetLine(r,FirstOn,*one);
     }
  /*Display the line if necessary*/
   if(StoragePage > -1){
     if(Off > -1){
       for(c=roi.xs;c<roi.xe;c=c+roi.resolution){
          (*imline)[c] = (Pixel) abs( (DPixel) (*one)[c] - (DPixel) (*two)[c] );
          }
       PutLine(r,StoragePage,*imline);
      }
     else PutLine(r,StoragePage,*one);
    }

 /*Find the pixels that belong to blobs in this row.
  All blobs that have had pixels added become active during this step*/
```

```
    BlobsInOneRow(SubtractType, r, roi.xs, roi.xe, roi.resolution, *one, *two,
*three, *neighbors, NegThreshold, PosThreshold);
    /*Any blob that is inactive at this point has finished growing.
    If it is through growing, check its size and kill it is the
    wrong size. Kill means delete after dumping the information into
    the background. */
    DumpSmallBigInactBackKill((Bsize) SmallestToFind, (Bsize) LargestToFind);
    /*by making all blobs inactive, only the ones activated by the next row
will reactivate*/
    MakeAllBlobsInActive();
    }

  /*We have all the blobs were gonna get
    Walk the list returning the largest blobs
    Copy the largest blobs into the proper storage locations*/
   for(r=0;r<*PosNumberToFind;r++){
    TempBlob = GetLargestPosBlob();
     if(TempBlob != NULL){
         CopyToBlobArray(TempBlob,posblobs,r,roi.resolution);
         KillBlob(TempBlob);
         }
     else *PosNumberToFind = r;
    }
  for(r=0;r<*NegNumberToFind;r++){
    TempBlob = GetLargestNegBlob();
     if(TempBlob != NULL){
         CopyToBlobArray(TempBlob,negblobs,r,roi.resolution);
         KillBlob(TempBlob);
         }
     else *NegNumberToFind = r;
    }
 /*All remaining blobs are small ones*/
 DumpAllBlobsIntoBackKill();
 /*Copy the background to its proper storage area*/
 CopyToBlobArray(&BackGround,background,0,roi.resolution);

 /*Free memory and return*/
 FREE(neighbors);
 FREE(one);
  if(Off >= 0){
   FREE(two);
    if(SecondOn >= 0) FREE(three);
   }
  if(StoragePage > -1)FREE(imline);
  if(StoragePage == -1)clear_status();
  return(SUCCESS);
 }
```

```
/**********************************
Local Functions
Killing a blob means freeing the memory it uses and piecing the linked
list back together. This kills all blobs.
**********************************/
void KillAllBlobs()
{
BLOBSTRU *blob, *prev;
blob = FirstBlob;
 while(blob != NULL){
   prev = blob;
   blob = blob->next;
   KillBlob(prev);
   }
 }

/*This routine joins a blob to the background blob then kills the
blob.*/
 void DumpAllBlobsIntoBackKill( void)
 {
 BLOBSTRU *blob, *prev;
 blob = FirstBlob;
  while(blob != NULL){
    prev = blob;
    blob = blob->next;
    DumpBlobIntoBackgroundThenKill(prev);
    }
  }

/*When we are finished finding blobs, we copy the linked list into the
return structure. It also computes centroids etc. using the sums
collected during blobbing.*/
 void CopyToBlobArray(BLOBSTRU * blob, OneBlob array[], int whichone, int resolution)
{
    double temp;
   temp = ( double) blob->numwraps;
   temp = temp * ( double) BIGINT;
   temp = temp + blob->totalbrightsq;
   array[whichone].centx = ( ( double) blob->csum )/ ( ( double) blob->size );
   array[whichone].centy = ( ( double) blob->rsum )/ ( ( double) blob->size );
   array[whichone].gray = ( ( double) blob->totalbright )/ ( ( double) blob->size
);
   array[whichone].variance = temp / ( ( double) blob->size ) - array[whichone].gray*array[whichone
   array[whichone].size = blob->size * resolution * resolution;
```

```
    array[whichone].radius = sqrt(array[whichone].size / M_PI);
   }

 /*Returns the largest positive blob in the linked list.*/
 BLOBSTRU * GetLargestPosBlob()
 {
   BLOBSTRU *blob, *retbl;
   Bsize largest;
   blob = FirstBlob;                                          290
   retbl = NULL;
   largest = 0;
    while(blob!=NULL){
      if(blob->totalbright < 0) blob=blob->next;
      else{
       if(blob->size > largest){
         largest = blob->size;
         retbl = blob;
         }
       blob=blob->next;                                       300
       }
      }
   return retbl;
 }

 /*Returns the largest negative blob in the linked list.*/
 BLOBSTRU * GetLargestNegBlob()
 {
   BLOBSTRU *blob, *retbl;
   Bsize largest;                                             310
   blob = FirstBlob;
   retbl = NULL;
   largest = 0;
    while(blob!=NULL){
      if(blob->totalbright > 0) blob = blob->next;
      else{
       if(blob->size > largest){
         largest = blob->size;
         retbl = blob;
         }
       blob=blob->next;                                       320
       }
      }
   return retbl;
 }

/*Allocate memory for a new blob structure. Fit it into the linked
list. Returns its tag.*/
```

```
Tag NewBlob()
{                                                                    330
   BLOBSTRU *newblob;
   newblob = (BLOBSTRU *) MALLOC( sizeof(BLOBSTRU));
    if(newblob==NULL){
     Fatal_Error_Message("Out of Memory.");
     }
   BlobTag++;
    if(BlobTag<0){
     Fatal_Error_Message("This is a critical error.  You have wrapped around on the
blob tags.");
     exit(1);                                                        340
     }
   newblob->tag = BlobTag;
   newblob->size = newblob->csum = newblob->rsum = newblob->totalbright = 0;
   newblob->totalbrightsq = (DoWa) 0;
   newblob->numwraps = 0;
   newblob->previous = NULL;
   newblob->next = FirstBlob;
   newblob->active = -1;
    if(FirstBlob!=NULL)FirstBlob->previous = newblob;
   FirstBlob = newblob;                                             350
    return(BlobTag);
   }

 #define SameSign(a,b) (((a > 0) && (b > 0)) || ((a < 0) && (b < 0)))
/*Set NegThreshold = - PIXEL_MAX  to get only positive pixels */
/* For row rownum, This routine checks the bright pixels to see which blob they
belong to*/
 void BlobsInOneRow( int SubtractType, int rownum, int xs, int xe, int resolution,
ImageLine one, ImageLine two, ImageLine three, DImageLine neighbors, DPixel
NegThreshold, DPixel PosThreshold)                                   360
 {
  int c;
 DPixel upper, left;
 DPixel temp;
  int YesItIsBright;
 DPixel pixmag;
  for(c=xs;c<xe;c=c+resolution) {
/*If pixel is bright consider it or else set it 0.*/
     YesItIsBright = 0;
     pixmag = 0;                                                     370
      switch (SubtractType){
       case NoSubtract:
          pixmag = one[c];
           if(IsBright(pixmag, NegThreshold, PosThreshold))YesItIsBright = 1;
           break;
```

```
        case SingleSubtract:
           pixmag = (DPixel) ( (DPixel)one[c] - (DPixel)two[c] );
            if(IsBright(pixmag, NegThreshold, PosThreshold))YesItIsBright = 1;
            break;
        case DoubleSubtract:                                                    380
           pixmag = (DPixel) ( (DPixel)one[c] - (DPixel)two[c] );
           temp = (DPixel) ( (DPixel)three[c] - (DPixel)two[c] );
            if( (IsBright(pixmag, NegThreshold, PosThreshold)) && (IsBright(temp, NegThreshold,
PosThreshold)) && (SameSign(pixmag,temp)) )YesItIsBright = 1;
           pixmag = (pixmag + temp)/2;
      }
      if(YesItIsBright){
        upper=UpperNeighbor(neighbors,c);
        left=LeftNeighbor(neighbors,c,xs,resolution);
         if(!upper && !left){                                                   390
          PutItInBlob(&neighbors[c],rownum,c,pixmag,NewBlob());
          }
         else {
          PutItInBlob(&neighbors[c],rownum,c,pixmag,max(left,upper));
           if( (left!=upper) && (min(left,upper)) )JoinBlobs(neighbors,upper,left);
          }
        }
      else {
        BackGround.size++;
        BackGround.totalbrightsq = BackGround.totalbrightsq + (DoWa) (pixmag*pixmag);
         if(BackGround.totalbrightsq > BIGINT){
          BackGround.numwraps++;
          BackGround.totalbrightsq = BackGround.totalbrightsq - BIGINT;
          }
        BackGround.totalbright = BackGround.totalbright + pixmag;
        neighbors[c]=0;
     }
    }
}

                                                                            410
   /*FindBlobPointer looks for a particular blob tag and returns a
pointer to the tagged blob.  NULL is a blob with tag is not found. The
BLOBSTRU is a linked list of blobs and tag is basically a long int that
identifies the blob */
BLOBSTRU *FindBlobPointer(Tag tag)
{
   BLOBSTRU *blob;
   /*FirstBlob is global to this file*/
   blob = FirstBlob;
   /*Until you reach the last blob ask are you it?*/                          420
    while(blob!=NULL)
    {
```

```
    if(blob->tag==tag) return(blob);
    blob = blob->next;
    }
   /* If did not find it, return NULL*/
    return((BLOBSTRU *)NULL);
   }

/*JoinBlobs join two blobs together then kill the child*/                    430
 void JoinBlobs(DImageLine row, Tag parentnum, Tag childnum)
{
    int i;
   BLOBSTRU *parent, *child;
   parent = FindBlobPointer(parentnum);
   child = FindBlobPointer(childnum);
    if((parent==NULL) || (child==NULL))Fatal_Error_Message("Either parent or
child not found.");
   parent->size += child->size;
   parent->rsum += child->rsum;                                             440
   parent->csum += child->csum;
   parent->totalbright += child->totalbright;
   parent->totalbrightsq += child->totalbrightsq;
    if(parent->totalbrightsq > BIGINT){
      parent->numwraps++;
      parent->totalbrightsq = parent->totalbrightsq - BIGINT;
      }
   parent->numwraps += child->numwraps;
   parent->active = -1;
   KillBlob(child);                                                         450
    for(i=0;i<IMAGEWIDTH;i++) if(row[i]==childnum)row[i]=parentnum;
   }

/*Join a blob to the background blob, then kill the blob.*/
 void DumpBlobIntoBackgroundThenKill(BLOBSTRU *blob)
{
   BackGround.size += blob->size;
   BackGround.totalbright += blob->totalbright;
   BackGround.totalbrightsq += blob->totalbrightsq;
    if(BackGround.totalbrightsq > BIGINT){                                  460
      BackGround.numwraps++;
      BackGround.totalbrightsq = BackGround.totalbrightsq - BIGINT;
      }
   BackGround.numwraps += blob->numwraps;
   KillBlob(blob);
   }

   /*Every pixel that passes the threshold test is added to a blob.
The routine receives a pointer to the pixel (it will be changed), the
```

```
/*pixel's row, column (y, x) coordinates on the screen, its intensity
(gray) and the name (rather the number) of the blob, the pixel belongs
to.*/
 void PutItInBlob(DPixel *pixel, int row, int column, DPixel gray, Tag tag)
{
  BLOBSTRU *blob;
  /*Find the pointer to the blob named tag*/
   if((blob = FindBlobPointer(tag))==NULL){
     Fatal_Error_Message("Invalid blob tag.");
     }
  /*Increase the blob's area, column sum, row sum, total
brightness and total brightness squared. Of these, really only the
area, column and row sum are critical. The other two provide an
average brightness and its standard deviation. These are not critical*/
   blob->size++;
   blob->csum += column;
   blob->rsum += row;
   blob->totalbright += gray;
   /* This is weird. On large blobs, the total brightness squared
often exceeds what an int or long can handle. Since the
code was developed on a computer without a floating point CPU it was too
slow to use floats (with exponents, we do not need more than 10 digits
accuracy) so we use this strange code to gain extra long ints.*/
    blob->totalbrightsq += (DoWa) ((DoWa)gray* (DoWa) gray);
    if(blob->totalbrightsq > BIGINT){
      blob->numwraps++;
      blob->totalbrightsq = blob->totalbrightsq - BIGINT;
      }

     /*Blobs are active when they are growing.  Since we just added a
pixel to this blob it is active*/
    blob->active = -1;

     /*We label this pixel as belonging to the blob named tag.  Suppose
the line being processed is line 5. Once we are finished processing line
5, it will have a zero wherever there was a pixel failing the threshold
test.  For every pixel that did pass, there will be a tag indicating
which blob that pixel belonged to.  As we process line 6, when a blob
passes the threshold test we compare it to the number in line 5. If the
pixel in line 5 is tag, then the pixel in 6 is a member of blob tag.*/
    *pixel = tag;
    }

/*DumpSmallInactiveIntoBackgroundThenKill This walks the blob linkage and kills
all inactive blobs that are too small */
 void DumpSmallInactBackKill(Bsize smallsize)
 {
```

```
   BLOBSTRU *blob, *temp;
   blob = FirstBlob;
    while(blob!=NULL){
      if(blob->active)blob=blob->next;                                    520
      else{
       if(blob->size>=smallsize)blob=blob->next;
       else {
         temp=blob;
         blob=blob->next;
         DumpBlobIntoBackgroundThenKill(temp);
         }
       }
     }
}                                                                         530

/*Join blobs that are too small or too big to the background blob, then
kill the blob.*/
 void DumpSmallBigInactBackKill(Bsize smallsize, Bsize largesize)
{
   BLOBSTRU *blob, *temp;
   blob = FirstBlob;
    while(blob!=NULL){
      if(blob->active)blob=blob->next;
      else{                                                               540
       if((blob->size <= largesize) && (blob->size>=smallsize) ) blob=blob->next;
       else {
         temp=blob;
         blob=blob->next;
         DumpBlobIntoBackgroundThenKill(temp);
         }
       }
     }
}
                                                                          550
   /*MakeAllBlobsInActive This walks the blob linkage and makes all of
them inactive. this means they have finished growing*/
 void MakeAllBlobsInActive()
{
   BLOBSTRU *blob;
   blob = FirstBlob;
    while(blob!=NULL){
      blob->active=0;
      blob=blob->next;
      }                                                                   560
}

/*To Kill a blob means to remove it from the linked list and free its memory*/
```

```
void KillBlob(BLOBSTRU *blob)
{
   if(blob->previous==NULL){
    FirstBlob = blob->next;
     if(FirstBlob!=NULL)FirstBlob->previous=NULL;
    }
    else {                                                                    570
     blob->previous->next = blob->next;
      if(blob->next!=NULL)blob->next->previous = blob->previous;
     }
   FREE(blob);
   }
```

Chapter 4

dip.c, dip.h

Documentation Date: 6/27/94

4.1 Sequencing and Thresholds

The routines in this file perform image sequencing and determine thresholds.
New Data Types:
None.
Definitions:

- YAXISMIN - the minimum value for the y axis of histograms.
- YAXISMAX - the maximum value for the y axis of histograms.
- NOVALLEY - (-1) return value if the valley finding algorithm fails.
- NOPEAK - (-2) return value if the peak finding algorithm fails.

4.1.1 Header File Listing:

```
#ifndef DIP_H
  #define DIP_H
  /*We subtract and do a brightness histogram of images i+2 − i for
i from 0 to 3.
    We process every resolution rows and columns of the image.
    If ShowHist<>0 then we plot the histogram with a yaxis from
YAXISMIN to YAXISMAX.
    Whichever set has the largest positive histogram is the all on
minus all off frame.
    To determine the threshold we find the left and right valleys of
the positive histogram then set the threshold to percentvalley between
them. EG. if percentvalley = 0 threshold = left edge, if 100 then its
the right edge.
    Numinslope is the number of data points used in the best fit slope
```

```
calculation.
    Thresh is the threshold we compute.
    It returns the image # of the first falling edge EG. suppose the
images are off on on off off on on. Then it returns 2.
    */
   int FindFirstFallingAndThresh( int resolution, int numinslope,
DPixel *thresh, int ShowHist, int percentvalley);
   int FindThreshold( int OnFrame, int DimFrame, int resolution, int numinslope,
DPixel *thresh, int ShowHist, int percentvalley);
   #define YAXISMIN 0
   #define YAXISMAX 100
   #define NOVALLEY -1
   #define NOPEAK -2
   #endif
```

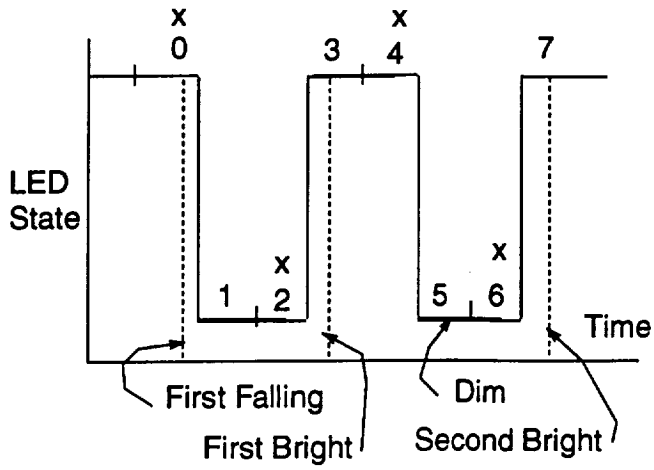

Figure 4.1: Eight Frames Grabbed by the System.

4.2 Function: FindFirstFallingAndThresh

Documentation Date: 6/24/94

Prototypes:

```
int FindFirstFallingAndThresh(int resolution, int numinslope, DPixel
*thresh, int ShowHist, int percentvalley);
```

Source File: *dip.c*
Type of Function: User Callable
Header Files Used in dip.c: *<float.h> <stdio.h> <string.h> "datatype.h" "plot.h" "dip.h" "target.h"*

Description:

This routine finds the first image with a falling edge of the LED, For example, suppose image-LEDstate is: 0-LEDon, 1-LEDon, 2-LEDoff, 3-LEDoff, ... Then this would return 1. It also determines a threshold for the blob analysis. The threshold is returned through the parameters.

4.2.1 Theory

The program grabs eight frames. Two consecutive frames have the LED on. The first of the two has the LED on brighter than the second (a hardware problem). After two on frames, there are two consecutive off frames. The second off however has the LED on dimly (hardware problems). Suppose frames have been grabbed as figure 4.1 shows. The figure shows frames 3 and 4 as a consecutive set of On frames. Frame 0, is the first falling frame.

The algorithm computes a histogram for four images: Image 2 - Image 0, Image 3 - Image 1, Image 4 - Image 2, and Image 5 - Image 3. Two of

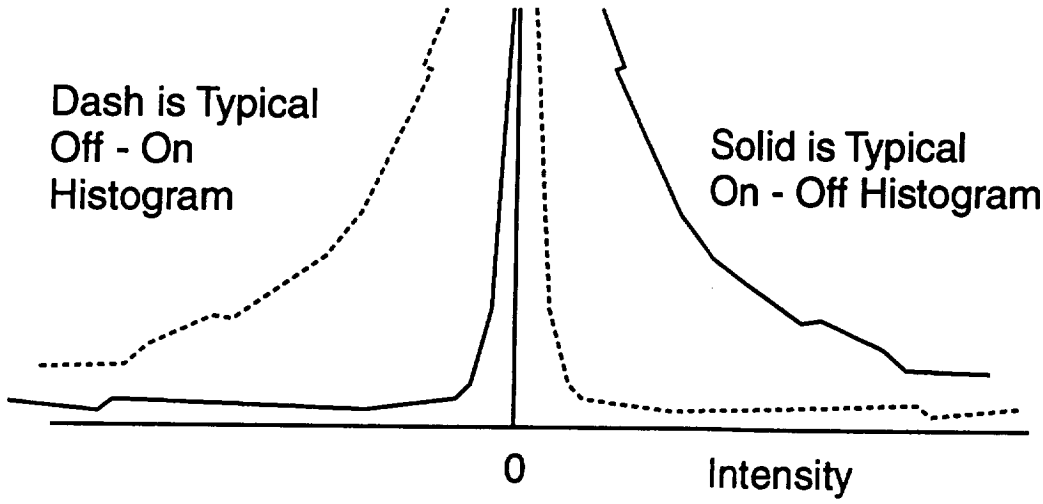

Figure 4.2: Typical On-Off and Off-On Histograms.

these are off minus on frames and two are on minus off. Figure 4.2 shows a typical histogram for an on-off and off-on image. When we compute these histograms, we use a subset of the data. We process only every "resolution" line and column (user selectable).

After the histograms are computed, we determine the right edge of the center hump of the histogram. See figure 4.3. The parameter numinslope is used in the algorithm for finding the edge. After finding the edge, we compute the area under the histogram from the edge to intensity of 255. The two images that correspond to LEDon minus LED off have a significantly larger area. We select the second in sequence of these two[1] as the first falling. The only detail is that the frames could be 0, 3 (see figure 4.1) where 0 is the second in the sequence. The first in the sequence of two on minus off frames is used to determine the threshold.

To determine the threshold, we compute the left edge of the target hump (if it exists). See figure 4.4. If there is no left edge, or if it is lower than the right edge, then threshold is equal to the right edge. Otherwise, threshold is a percentage of the distance between the edges. Zero percent is the right edge of the background, one hundred percent is the left edge of the target.

[1]The second in sequence does not necessarily have the smaller of the two areas because area is very sensitive to the edge finder. The on minus off frames however have such a larger positive area, they have always been correctly identified (as far as we know).

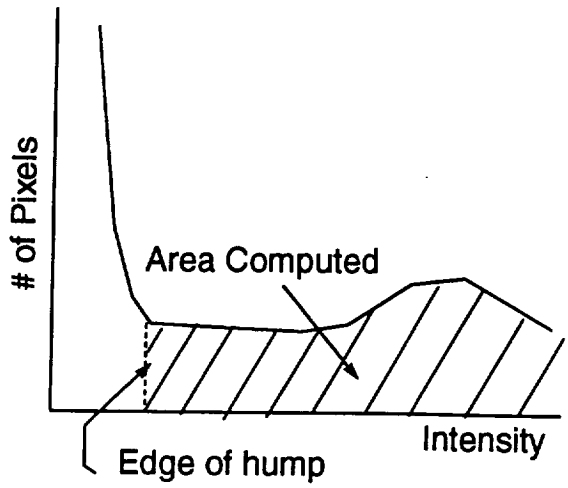

Figure 4.3: The Right Edge of the Center Hump, and the Area Under the Histogram.

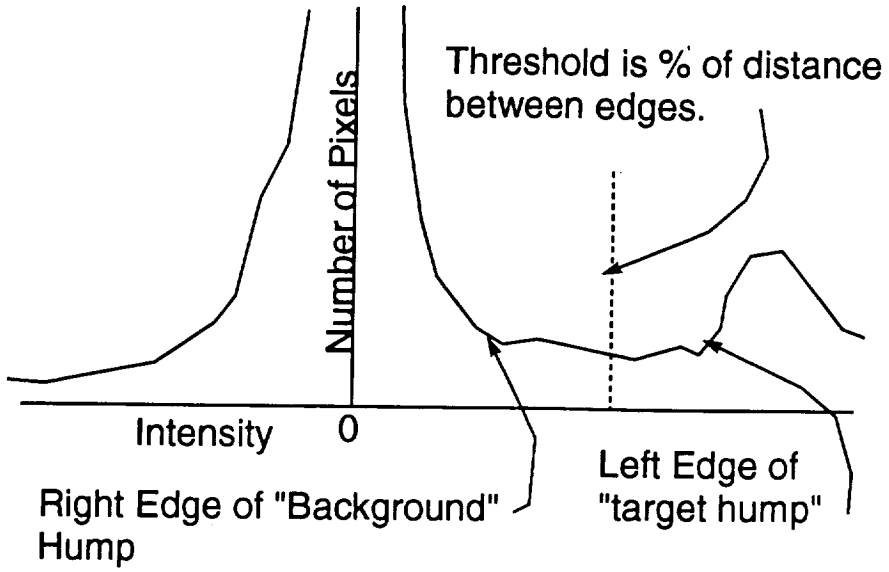

Figure 4.4: The Right and Left Edges of the Background and Target.

4.3 Function: SlopeOfN

Documentation Date: 6/27/94

Prototypes:

```
float SlopeOfN (int numinslope, Histogram hist, long xs, long x, long
xy, long y)
```

Source File: *dip.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in dip.c: *<float.h> <stdio.h> <string.h> "datatype.h" "plot.h" "dip.h" "target.h"*

Description:

This routine computes the best fit slope to N data.

4.3.1 Theory

Think of a histogram as an x,y plot where x is the intensity and y is the number of pixels at that intensity. This routine fits a line to N consecutive data points and returns the slope of the line.

The error between the best fit line and the data is:

$$E = \sum_{i=j} N + j \left(mx_i + b - y_i\right)^2$$

To minimize E we determine the critical points of the error:

$$\sum_{i=j} N + j2 \left(mx_i + b - y_i\right) = 0$$

$$\sum_{i=j} N + j2 \left(mx_i + b - y_i\right) x_i = 0$$

These equations have the following solution for slope m:

$$m = \frac{N \sum_{i=j}^{N+j} \left(x_i y_i\right) - \sum_{i=j}^{N+j} \left(x_i\right) \sum_{i=j} N + j y_i}{N \sum_{i=j} N + j \left(x_i^2\right) - \sum_{i=j} N + j x_i \sum_{i=j} N + j x_i} \tag{4.1}$$

This routine implements equation 4.1

4.4 Function: XOfValley

Documentation Date: 6/24/94

Prototypes:

```
int XOfValley(Histogram hist, int numinslope, int start, int end);
```

Source File: *dip.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in dip.c: *<float.h> <stdio.h> <string.h> "datatype.h" "plot.h" "dip.h" "target.h"*

Description:

This routine computes the array offset (the x or intensity value if you think of a plot) of the right edge of a hump. It searches the histogram between start and end for the valley. It returns the valley offset value when there is no error. It returns NOPEAK and NOVALLEY (defined in dip.h) if an error does occur.

4.4.1 Theory

A valley is defined as the first location where the slope of a best fit line to part of the data changes from negative to non-negative. The region for finding the best fit slope is defined with "numinslope". For example, if numinslope is 3 then "best fit" slopes are computed using points 0,1,2 then 1,2,3 then 2,3,4 etc. until the slope is negative and becomes non-negative. Actually the routine first seeks the peak value between start and end, then seeks a negative slope, then seeks a non-negative slope. If the non-negative slope is found with data 2,3,4 a 2 is returned.

4.5 Function: Miscellaneous Support Routines in Dip.c

Documentation Date: 3/10/95

Prototypes:

```
void showhistogram(Histogram bigy[4], int which, int maxy, char *message,
GraphData *graphdata, long xthresh)
   void MakeXNHist(Histogram from, Histogram to)
   long SumOfHist (Histogram hist, int starting)
   void RunningSum (long oldest, long newest, long *sum)
   void RunningSumsOverN (int numinslope, int startingx, Histogram
hist, long *xs, long *x, long *xy, long *y)
   void FirstSumsOverN (int numinslope, int startingx, Histogram hist,
long *xs, long *x, long *xy, long *y)
   int IndexOfLargest (Histogram hist, int start, int end);
   int FindThreshold(int OnFrame, int DimFrame, int resolution, int
numinslope, DPixel *thresh, int ShowHist, int percentvalley);
```

Source File: *dip.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in dip.c: *<float.h> <stdio.h> <string.h> "datatype.h" "plot.h" "dip.h" "target.h"*

Description:

- showhistogram: displays a histogram on the computer monitor.
- MakeXNHist: reverses the order of the histogram. This is done to simplify the process of finding the left edge of the target hump (see FindFirstFallingAndThresh). After reversing the histogram we can find the right edge of the target hump instead of the left edge.
- SumOfHist: computes the area under the histogram from starting to 255.
- RunningSum: computes a running sum. It is used to determine the best fit slope to the data. A running sum adds a new point and subtracts an old point.
- RunningSumsOverN: computes the four sums needed to find best fit slopes. x, x squared, xy, and y where x is the intensity, and y is the # pixels with that intensity.
- FirstSumsOverN: this routine gets the four sums over N started.

- IndexOfLargest: returns the index (histograms are stored as an array) of the maximum y value. It begins the search from the end back toward the start.

- FindThreshold: This routine is nearly identical to FindFirstFallingAndThresh except it does not sequence the images.

4.5.1 Program Listing:

```
#include < float.h>
#include <stdio.h>
#ifndef NeXT
    #include <graph.h>
    #endif
#include <string.h>
#include "datatype.h"
#include "plot.h"
#include "dip.h"
#include "target.h"

/******
Local functions
*******/
    long SumOfHist (Histogram hist, int starting);
    void RunningSum ( long oldest, long newest, long *sum);
    void RunningSumsOverN ( int numinslope, int startingx, Histogram hist, long
*xs, long *x, long *xy, long *y);
    void FirstSumsOverN ( int numinslope, int startingx, Histogram hist, long
*xs, long *x, long *xy, long *y);
    float SlopeOfN ( int numinslope, Histogram hist, long xs, long x, long xy,
long y);
    int IndexOfLargest (Histogram hist, int start, int end);
    int IndexOfFirstOneBelow100(Histogram hist, int start, int end);
    int XOfValley(Histogram hist, int numinslope, int starting, int end);
    void MakeXNHist(Histogram from, Histogram to);
    void showhistogram(Histogram bigy[4], int which, int maxy, char *message,
GraphData *graphdata, long xthresh);

Histogram hist[4];
Histogram histtemp;
long xaxis[NUMBEROFPIXELBINS+5], yaxis[NUMBEROFPIXELBINS+5];

/*This routine will display a histogram on the computer monitor*/
 void showhistogram(Histogram bigy[4], int which, int maxy, char
*message, GraphData *graphdata, long xthresh)
 {
  int i,k;
  long temp;
 k = 0;
 /*Histograms are positive and negative display only the positive part*/
  for(i=NUMBEROFPIXELBINS/2;i<NUMBEROFPIXELBINS;i++){
   temp = bigy[which][i];
   /*If the data is too large set it equal to the max*/
    if(temp < maxy)yaxis[k] = temp;
    else yaxis[k] = maxy;
```

```
#ifdef CHECKLENGTH
if(k>=(NUMBEROFPIXELBINS+5))Fatal_Error_Message("Blew it in showhistogram");
#endif
    k++;
    }
#ifndef NeXT
 SetupGraph(graphdata);
 Plot(xaxis,yaxis,NUMBEROFPIXELBINS/2,LINES,PLOT_YELLOW);
  if(xthresh > 0)PlotVerticalLine(xthresh, PLOT_DARK_RED);
 gprintandwait(message);
 QuitPlot();
    #endif
 }

/* This routine finds the first image with a falling edge of the LED,
For example, suppose image-LEDstate is:
 0-LEDon, 1-LEDon, 2-LEDoff, 3-LEDoff, ...
 Then this would return 1 */
 int FindFirstFallingAndThresh( int resolution, int numinslope,
DPixel *thresh, int ShowHist, int percentvalley)
  {
   char message[40];
  GraphData graphdata;
   int imageframe;
   int first, second, i;
   long total[4], max, nextmax;
   int valley[4];
   int maxindex, nextindex;
   int returnvalue, valleyvalue;
   int secondstarting, secondending, secondvalley;
   int temp,tempp;
  ROI roi;

/* A macro to say the entire image is of interest*/
  WHOLE_IMAGE(roi);

/* We will process every resolution line and column, This reduces the
amount of processing. We typically use resolution of 3 or 4. As the
target gets further away, this value will have to decrease.*/
  roi.resolution = resolution;
  max = -1;

/* If we want to display graphs, get them ready */
   if(ShowHist){
     for(i=0;i<128;i++)xaxis[i]=2*i;
     #ifdef CHECKLENGTH
     if(i>(NUMBEROFPIXELBINS+5))Fatal_Error_Message("Blew it in findfirstfallingandt
```

```
#endif
graphdata.minx = 0;
graphdata.maxx = 255;
graphdata.miny = YAXISMIN;
graphdata.maxy = ( long) YAXISMAX;
graphdata.majxtic = 8;
graphdata.majytic = 5;
graphdata.minxtic = 4;                                                    100
graphdata.minytic = 5;
graphdata.DefaultFontAndColor = TRUE;
}

/* First we sequence the images. That is find out which are on frames an
which are off. We do this by subtracting 2-0, 3-1, 4-2, 5-3. Two of
these will be mostly negative results and two will be mostly positive.
*/
  for(imageframe = 0;imageframe < 4;imageframe++){                        110
   SubAndHistogram(imageframe, imageframe+2, -1. hist[imageframe], roi);
   if(ShowHist){
   sprintf(message,"Histogram of Image %d - %d HitKey",imageframe+2,imageframe);
#ifdef CHECKLENGTH
if(( int)strlen(message)>=40)Fatal_Error_Message("Blew it in findfirstfallingandthresh");
#endif
    showhistogram(hist,imageframe,100,message,&graphdata,0);
    }
   /*Find the approximate intensity of the right edge of the center hump*/
   valley[imageframe] = XOfValley(hist[imageframe], numinslope, 0, NUMBEROFPIXELBINS
-1 );
   /*Find the area under the histogram from the right edge to 255*/
   if(valley[imageframe] < 0)total[imageframe] = 0;
   else total[imageframe] = SumOfHist (hist[imageframe], valley[imageframe]);
  }
/*Two of the areas will be significantly larger than the other two,
  they represent properly sequenced images. Find the largest one.*/
max = 0;
maxindex = 0;
 for(imageframe = 0;imageframe < 4;imageframe++){                         130
  if(total[imageframe] > max){
  max = total[imageframe];
  maxindex = imageframe;
  }
 }
/* Now find the second largest */
nextmax = 0;
nextindex = 0;
 for(imageframe=0;imageframe < 4;imageframe++){
  if( (total[imageframe] > nextmax) && (imageframe != maxindex) ){        140
```

```
        nextmax = total[imageframe];
        nextindex = imageframe;
        }
      }

    /* Due to a hardware LED problem, we want to base threshold on the
   first of two consecutive on frames, but we want to return the second on
   frame as the first falling.  From above we have a very robust method of
   finding the two LED on frames, now we must determine which one is the
   good LED on frame and which one has "punched through".  A human could
   look at the two histograms and recognize which one corresponds to the
   good LED on frame, but our area algorithm is sensitive to the valley we
   found.  Therefore, what we do is say we want the image that occurs first
   in the sequence.  It turns out that the two choices maxindex and
   nextindex are always sequential or else they are 0 and 3.  So first we
   put them in numerical order.  */
     if(nextindex < maxindex){
      first = nextindex;
      second = maxindex;
      }
     else {
      first = maxindex;
      second = nextindex;
      }

    /* Now if the first happens to be 0 then the other is 3 so 3 is
   actually the good LED on frame and 0 is the first falling*/
     if( (first == 0) && (second == 3) ){
      returnvalue = first;
      valleyvalue = second;
      }
     else {
      returnvalue = second;
      valleyvalue = first;
      }

    /* Now we look for the left edge of the histogram hump caused by
   the target, if it exists. to do this we simply change the +- sign of
   the intensity of the thresholding histogram (the valleyvalue
   histogram) then use the same algorithm as finding the right edge of the
   histogram's center hump*/
     MakeXNHist(hist[valleyvalue],histtemp);

    /*Set the search area from 0 (beginning of the array) to the edge
   of the hump already found, then find an edge called secondvalley*/
     secondstarting = 0;
     secondending = NUMBEROFPIXELBINS - valley[valleyvalue];
```

```
    secondvalley = XOfValley(histtemp, numinslope, secondstarting, secondending);

    /*If a second valley is not found, base threshold on the edge of                190
the hump.  If one is found, and it is to the right of the first one, set
threshold at a percentage between the two edges*/
    if(secondvalley < 0) *thresh = (valley[valleyvalue] - 128) * 2;
    else {
      secondvalley = NUMBEROFPIXELBINS - secondvalley;
       if(secondvalley < valley[valleyvalue] ) *thresh = (valley[valleyvalue]
- 128) * 2;
       else *thresh = (((100-percentvalley)*valley[valleyvalue]+percentvalley*secondvalley)/100
- 128) * 2;
     }                                                                             200
/*
  *thresh = (valley[valleyvalue] - 128) * 2;
*/
  temp = (valley[valleyvalue] -128)*2;
  tempp = (secondvalley-128)*2;
  /*
  printf("first second valley are %d and %d\n",temp,tempp);
  printandwait("hit a key");
  */
  if(ShowHist){                                                                    210
    sprintf(message,"The line is the threshold, HitKey");
    showhistogram(hist,valleyvalue,100,message,&graphdata,( long) *thresh);
    }
  return returnvalue;
 }

int FindThreshold( int OnFrame, int DimFrame, int resolution, int numinslope,
DPixel *thresh, int ShowHist, int percentvalley)
 {
  char message[40];                                                                220
 GraphData graphdata;
  int i;
  long total[4], max;
  int valley[4];
  int returnvalue;
  int secondstarting, secondending, secondvalley;
  int temp,tempp;
 ROI roi;

 WHOLE_IMAGE(roi);                                                                 230
 roi.resolution = resolution;
 max = -1;
  if(ShowHist){
    for(i=0;i<128;i++)xaxis[i]=2*i;
```

```
#ifdef CHECKLENGTH
if(i>(NUMBEROFPIXELBINS+5))Fatal_Error_Message("Blew it in findfirstfallingandthre
#endif
    graphdata.minx = 0;
    graphdata.maxx = 255;
    graphdata.miny = YAXISMIN;
    graphdata.maxy = ( long) YAXISMAX;
    graphdata.majxtic = 8;
    graphdata.majytic = 5;
    graphdata.minxtic = 4;
    graphdata.minytic = 5;
    graphdata.DefaultFontAndColor = TRUE;
    }

  SubAndHistogram(OnFrame, DimFrame, -1, hist[0], roi);
//  valley[0] = XOfValley(hist[0], numinslope, 0, NUMBEROFPIXELBINS -1 );
//  Search starts at gray level 2 since accumulation zeros gray level 1
   valley[0] = XOfValley(hist[0], numinslope, 2, NUMBEROFPIXELBINS -1 );
    if(valley[0] < 0)total[0] = 0;
    else total[0] = SumOfHist (hist[0], valley[0]);
   MakeXNHist(hist[0],histtemp);
   secondstarting = 0;
   secondending = NUMBEROFPIXELBINS - valley[0];
   secondvalley = XOfValley(histtemp, numinslope, secondstarting, secondending);
    if(secondvalley < 0){
      if(DimFrame > -1) *thresh = (valley[0] - 128) * 2;
      else *thresh = valley[0];
     }
   else {
     secondvalley = NUMBEROFPIXELBINS - secondvalley;
      if(secondvalley < valley[0] ){
       if(DimFrame > -1) *thresh = (valley[0] - 128) * 2;
       else *thresh = valley[0];
      }
      else{
       if(DimFrame > -1) *thresh = (((100-percentvalley)*valley[0]+percentvalley*secondvalley)/1
- 128) * 2;
       else *thresh = ((100-percentvalley)*valley[0]+percentvalley*secondvalley)/100;
      }
     }
  if(ShowHist){
    temp = valley[0];
    if(DimFrame > -1) temp = (temp-128)*2;
   tempp = secondvalley;
    if(DimFrame > -1) tempp = (tempp-128)*2;
   sprintf(message,"The line is the threshold, HitKey");
   showhistogram(hist,0,100,message,&graphdata,( long) *thresh);
```

```
    }
   return returnvalue;
  }

/*This routine reverses the order of the histogram. This is done to
simplify the process of finding the left edge of the target hump. After
reversing the histogram we can find the right edge of the target hump
instead of the left edge.*/
 void MakeXNHist(Histogram from, Histogram to)                          290
  {
   int i;
   for(i=0;i<NUMBEROFPIXELBINS;i++){
     to[i] = from[NUMBEROFPIXELBINS-1-i];
    }
  }

/*This routine computes the area under the histogram from starting to
255*/
 long SumOfHist (Histogram hist, int starting)                          300
  {
   long sum;
   int i;
  sum = 0;
   for(i=starting;i<NUMBEROFPIXELBINS;i++){
     sum = sum + hist[i];
    }
   return sum;
  }
                                                                        310
/*This routine computes a running sum. It is used to determine the best
fit slope to the data.*/
 void RunningSum ( long oldest, long newest, long *sum)
  {
  *sum = *sum - oldest;
  *sum = *sum + newest;
  }

/*To compute the best fit slope we need four sums. x, x squared, xy, and
y where x is the intensity, and y is the # pixels with that intensity*/  320
 void RunningSumsOverN ( int numinslope, int startingx, Histogram hist, long *xs,
 long *x, long *xy, long *y)
  {
  /*Sum of hist[startingx] + hist[startingx + 1] + ... + hist[startingx + numinslope
-1] */
  /* startingx > 0 and startingx < NUMBEROFPIXELBINS - numinslope */
   long oxs, ox, oxy, oy;
   long nxs, nx, nxy, ny;
```

```
    ox = startingx - 1;
    oy = hist[ox];                                                    330
    nx = ox + numinslope;
    ny = hist[nx];
    oxs = ox * ox;
    nxs = nx * nx;
    oxy = ox * oy;
    nxy = nx * ny;
    RunningSum(ox,nx,x);
    RunningSum(oxs,nxs,xs);
    RunningSum(oxy,nxy,xy);
    RunningSum(oy,ny,y);                                              340
    }

/*Once we are rolling we use RunningSumsOverN. It subract one number
and adds one. To get started however we have to use this routine*/
 void FirstSumsOverN ( int numinslope, int startingx, Histogram hist, long *xs,
 long *x, long *xy, long *y)
  {
   int i;
  *x = *xs = *y = *xy = 0;
   for(i=startingx;i<startingx + numinslope;i++){                      350
     *x = (*x) + i;
     *xs = (*xs) + i * i;
     *y = (*y) + hist[i];
     *xy = (*xy) + hist[i] * i;
     }
  }

/*This routine computes the best fit slope to N data*/
 float SlopeOfN ( int numinslope, Histogram hist, long xs, long x, long xy, long
y)                                                                    360
  {
   long slope, denom;
  denom = xs * numinslope - x * x;
  slope = xy * numinslope - x * y;
   if(denom == 0){
     if(slope < 0) return (- FLT_MAX);
     return FLT_MAX;
    }
   return ( ( ( float) slope) / ( ( float) denom ) );
  }                                                                   370

/*This routine returns the index (histograms are stored as an array) of
the maximum y value. It begins the search from the end back toward the
start*/
 int IndexOfLargest (Histogram hist, int start, int end)
```

```
  {
   int i, bigindex;
   long max;
  bigindex = NUMBEROFPIXELBINS-1;
  max = hist[NUMBEROFPIXELBINS-1];                                    380
  bigindex = end;
  max = hist[bigindex];

   for(i=(end-1);i>(start-1);i--){
     if(hist[i] > max){
     bigindex = i;
     max = hist[i];
     }
    }
   return bigindex;                                                   390
  }

 int IndexOfFirstOneBelow100(Histogram hist, int start, int end)
  {
   int i, bigindex;
  bigindex = start;
   for(i=start;i<end;i++){
     if(hist[i] > 100) bigindex = i;
    }
   return bigindex;                                                   400
  }

/*This routine computes the intensity (the x value if you think of
a plot) where a hump ends.  It searches the histogram between start and
end fof the valley location.  It returns the x value when there is no
error.  It returns NOPEAK (-2) if it can not find a hump to start with.
It returns NOVALLEY (-1) if it cannot find a valley after finding the
hump. Note that start and end are array values not the actual
intensity.*/
 int XOfValley(Histogram hist, int numinslope, int start, int end)   410
  {
   int peak, valley;
   long xs, x, xy, y;
   float slope;
  /*Find the largest value between start and end*/
  peak = IndexOfLargest(hist, start, end);

  /*We fit numinslope data points to a line.  Once we get rolling it
works different than the initial calculation.*/
  FirstSumsOverN (numinslope, peak, hist, &xs, &x, &xy, &y);          420
  slope = SlopeOfN (numinslope, hist, xs, x, xy, y);
```

```
  /*Now we're cookin! Move right until we hit the end of the histogram.
Each time drop the left most data from our calculation and include a new
one on the right. Compute the best fit slope over numinslope data
points. Look for a negative slope to indicate finding the downward
surface of the peak. I am not sure why this is needed actually. */
    while( (slope >= 0) && (peak < end+1-numinslope) ){
     peak++;
     RunningSumsOverN (numinslope, peak, hist, &xs, &x, &xy, &y);          430
     slope = SlopeOfN (numinslope, hist, xs, x, xy, y);
     }
  if(slope >= 0) return NOPEAK;

 /*Ok so we have the true peak so continue finding slopes looking for
it to become positive.*/
  for(valley= peak+1;valley<end+1-numinslope;valley++ ){
    RunningSumsOverN (numinslope, valley, hist, &xs, &x, &xy, &y);
    slope = SlopeOfN (numinslope, hist, xs, x, xy, y);
     if(slope >= 0) return valley;                                        440
    }
  return NOVALLEY;
 }
```

Chapter 5

geometry.c, geometry.h

Documentation Date: 3/12/95

5.1 Sort out the blob information.

The one user callable routine in this file computes which blob belongs to the top, bottom, etc.
New Data Types:

Definitions:

- NOTENOUGHBLOBS = -1 (global) Error return.
- NOTARGETFOUND = -2 (global) Error return.

5.1.1 Header File Listing:

```
#ifndef GEOMETRY_H
  #define GEOMETRY_H
 /* This routine takes image blobs and fits them to a pattern.
 To reduce needless searching if a blob is not within tolerance*blob_radius
 of where it needs to be then we skip it.
 It returns Allstats loaded, a zero return is success, other errors are
 defined */
 int TargetCosys(Image *image, AllStats *allstats, double tolerance, int NumOfLedsToFind);
 #define NOTENOUGHBLOBS -1
 #define NOTARGETFOUND -2
 #endif
```

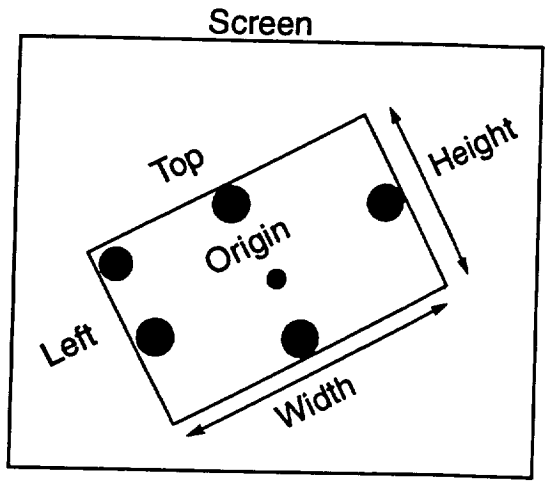

Figure 5.1: A Sample of the Target Showing the Definition of Height, Width, Top and Left.

5.2 Function: TargetCosys

Documentation Date: 3/12/95

Prototypes:

```
int TargetCosys(Image *image, AllStats *allstats, double tolerance,
int NumOfLedsToFind);
```

Source File: *geometry.c*

Type of Function: Routine TargetCosys is user callable all others are internal.

Header Files Used in geometry.c: *<math.h> <stddef.h> <stdlib.h> "datatype.h" "geometry.h" "memory.h"*

Description:

There are several routines in this file. The only user callable one is TargetCosys.

TargetCosys sorts the blobs in the image and determines which are top, bottom, etc. It is basically a tree search, tolerance is used to prune limbs from the tree. It returns either SUCCESS, NOTARGETFOUND, or NOTENOUGHBLOBS. The algorithm allows one retro to be missing provided the expected location of the blob on the mirror is also missing.

Figure 5.1 shows what a target is suppose to look like. Figure 5.2 shows the definition of the mirror and screen coordinate systems. Figure 5.3 shows the definition of error and the expected locations.

It tries all combinations of retros until it finds the one that produces the least error in what the target should look like. The target is expected to have five retros on the left right, top and bottom center edges. The fifth retro is in

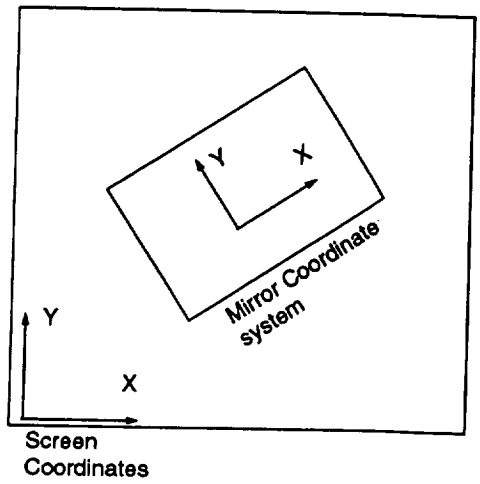

Figure 5.2: Coordinate Systems.

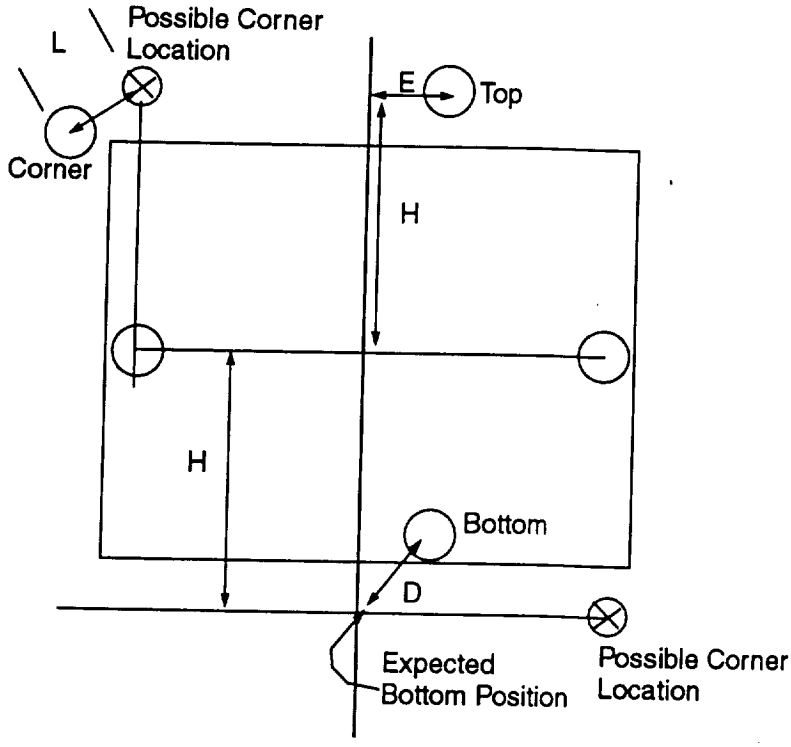

Figure 5.3: Definition of Expected Locations and Errors.

the upper left hand corner. Any image with less than 4 blobs is not a target because there are five retros and only one can be missing.

The routine requires the blobs to be sorted from largest to smallest. If they are not, it fails.

The routine picks two blobs at random and calls them the left and right. It then converts blob coordinates into mirror coordinates. Mirror center is the average of the left and right. +x points from left to right.

Now it searches for a top or bottom blob. Top must have a small x mirror coordinate. After finding top and bottom blobs it swaps left/right top/bottom if necessary to make the left/right larger than top/bottom.

The LED blobs are those that lie on the mirror. Basically the routine searches through the blobs until it finds one that has not been assigned to something else (like the corner or top retro blob). For each unassigned blob, it checks to see if it lies on the mirror.

5.2.1 Program Listing:

```
#include <math.h>
#include <stddef.h>
#include <stdlib.h>
#include "datatype.h"
#include "geometry.h"
#include "memory.h"


/* This routine takes image blobs and fits them to a pattern.
To reduce needless searching if a blob is not within tolerance*blob_radius
of where it needs to be then we skip it.
It returns Allstats loaded, a zero return is success, other errors are
defined */

void ConvertCoordinates( double bx, double by, double ox, double oy, double vx,
double vy, double px, double py, double *nx, double *ny);
double mag( double x, double y);
int PickTwoAtRandom( int *left, int *right, int max);
void ComputeError( int left, int right, int top, int bottom, int corner, int
led[MAX_LEDS] );
int CouldThisBeMissing( double x, double y, double xx, double xy, double yx,
double yy, double ox, double oy);
int FindLed( int left, int right, int top, int bottom, int corner, int numofblobs);
int Blob2IsBiggerThan1(Image *image, int one, int two);
void FindSomeLeds( int left, int right, int top, int bottom, int corner, int
numofblobs, int numtofind, int ledsfound[MAX_LEDS]);
int NextBlobOnMirror( int start, int left, int right, int top, int bottom, int
corner, int numofblobs);
int BlobIsOnMirror( int blobnumber);
```

```
 static int lastleft, lastright;
 double besterror;
 int bestleft, bestright, besttop, bestbottom, bestcorner, bestled[MAX_LEDS],
bestfound;
 double *mirrorx, *mirrory;/*Blob x and y coordinates relative to the mirror*/
 double *centx, *centy;/*Temporary Blob x and y coordinates*/
 double *radiusXtolerance;/*Blob radii times the tolerance*/
 double *sizes, *radius;
 double leftmostx, rightmostx, topmosty, bottommosty;

 int TargetCosys(Image *image, AllStats *allstats, double tolerance, int NumOfLedsToFind)
 {
  double leftx, rightx, lefty, righty;
  double originx, originy, dist;
  double xx, xy;/*The x axis of the mirror, from left to right*/
  double yx, yy;/*The y axis of the mirror, 90 degrees CCW from x axis*/
  double nx, ny;
  double toperror, bottomerror, possiblecornerx, possiblecornery;

  int left, right, top, bottom, led[MAX_LEDS], edge, nextedge, corner, i, j;
  int numofblobs;
  int needtoswitch;

   allstats->id.NormalNumOfLeds = NumOfLedsToFind;
 lastleft = 0; /*Used in the permutations.*/
 lastright = 0;
 numofblobs = image->num_of_blobs_found;

/*Make sure the blobs are sorted from largest to smallest. This is needed for
the find leds.*/
  for(i=0;i<numofblobs;i++)
    for(j=i+1;j<numofblobs;j++) if(Blob2IsBiggerThan1(image,i,j))Fatal_Error_Message("Sort
blobs large to small before geometry.");

 besterror = 10000.; bestleft = bestright = besttop = bestbottom = bestcorner
= bestfound = 0;

 centx = ( double *) MALLOC(numofblobs * sizeof( double));
 centy = ( double *) MALLOC(numofblobs * sizeof( double));
 mirrorx = ( double *) MALLOC(numofblobs * sizeof( double));
 mirrory = ( double *) MALLOC(numofblobs * sizeof( double));
 sizes = ( double *) MALLOC(numofblobs * sizeof( double));
 radius = ( double *) MALLOC(numofblobs * sizeof( double));
 radiusXtolerance = ( double *) MALLOC(numofblobs * sizeof( double));
  if(
    (centx == NULL) ||
```

```
    (centy == NULL) ||
    (radiusXtolerance == NULL) ||
    (sizes == NULL) ||
    (radius == NULL) ||                                              80
    (mirrory == NULL) ||
    (mirrorx == NULL) )
    Fatal_Error_Message("Not enough memory in geometry.c");

  for(i=0;i<numofblobs;i++){
   radiusXtolerance[i] = tolerance * image->blobs[i].radius;
   sizes[i] = image->blobs[i].size;
   radius[i] = image->blobs[i].radius;
   centy[i] = image->blobs[i].centy / 1.266;
   centx[i] = image->blobs[i].centx;                                 90
   }

  while(PickTwoAtRandom(&left, &right, numofblobs-1)){
   leftx = centx[left];
   lefty = centy[left];
   rightx = centx[right];
   righty = centy[right];
   xx = rightx - leftx;
   xy = righty - lefty;
   dist = mag(xx,xy);
   originx = (leftx + rightx)/2.;                                   100
   originy = (lefty + righty)/2.;

   /*Convert all blob coordinates into mirror coordinates.*/
   if(dist>0){
    xx = xx / dist;                  xy = xy / dist;
    yx = -xy;                    yy = xx;
    /*The left and right are special.*/
    mirrorx[left] = -dist/2.;      mirrory[left] = 0.;
    mirrorx[right] = dist/2.;      mirrory[right] = 0.;              110
    /*For all others.*/
     for(i=0;i<numofblobs;i++){
      if ( (i != right) && (i != left) ){
       ConvertCoordinates(centx[i], centy[i], originx, originy, xx, xy, yx, yy,
&nx, &ny);
        mirrorx[i] = nx;             mirrory[i] = ny;
       }
     }/*All blobs converted*/
    leftmostx = mirrorx[left] - image->blobs[left].radius;
    rightmostx = mirrorx[right] + image->blobs[right].radius;        120

  /*For all blobs that are not the left and right.*/
   for(edge=0;edge<numofblobs;edge++){ if( (edge != left) && (edge != right)
```

```
){
      /*If this blob can be a top or a bottom.*/
      toperror = fabs(mirrorx[edge]);
       if( toperror < radiusXtolerance[edge] ){
       /*Found an edge*/
        if( mirrorx[right] < fabs(mirrory[edge]) )needtoswitch = 1;
        else needtoswitch = 0;                                          130
       /*From next blob to the end.*/
        for(nextedge=edge+1;nextedge<numofblobs;nextedge++){ if( (nextedge != left)
&& (nextedge != right) ){
         /*If this blob is an opposite edge.*/
         bottomerror = mag( mirrorx[nextedge], mirrory[nextedge] + mirrory[edge]
);
          if( bottomerror < radiusXtolerance[nextedge] ){
          /*Make sure top has > 0 yvalue*/
           if(mirrory[edge] > 0.){
            top = edge;                                                 140
            bottom = nextedge;
            }
           else{
            top = nextedge;
            bottom = edge;
            }
          topmosty = mirrory[top] + image->blobs[top].radius;
          bottommosty = mirrory[bottom] - image->blobs[bottom].radius;
          /* Found a second edge so corner could be missing. sort out right left top
bottom compute error.*/                                                 150
           if(needtoswitch){
            /*What I think is left and right is actually top and bottom*/
            /*corner can be left/down or right/up*/
            /*try the later*/
            possiblecornerx = mirrorx[right];
            possiblecornery = mirrory[top];
             if(CouldThisBeMissing(possiblecornerx,possiblecornery,xx,xy,yx,yy,originx,originy)){
              FindSomeLeds(left, right, top, bottom, -1, numofblobs, NumOfLedsToFind,
led);
              ComputeError(top,bottom,right,left,-1,led);               160
              }
             else { /*If right/up is not missing, is left/down?*/
              possiblecornerx = mirrorx[left];
              possiblecornery = mirrory[bottom];
               if(CouldThisBeMissing(possiblecornerx,possiblecornery,xx,xy,yx,yy,originx,originy)){
               FindSomeLeds(left, right, top, bottom, -1, numofblobs, NumOfLedsToFind,
led);
               ComputeError(bottom,top,left,right,-1,led);
               }
             }                                                          170
```

```
        }/*End need to switch*/
       else {/*Don't need to switch*/
        /*What I think is left and right is*/
        /*corner can be left/up or right/down*/
        /*try the former*/
        possiblecornerx = mirrorx[left];
        possiblecornery = mirrory[top];
         if(CouldThisBeMissing(possiblecornerx,possiblecornery,xx,xy,yx,yy,originx,originy)){
          FindSomeLeds(left, right, top, bottom, -1, numofblobs, NumOfLedsToFind,
led);
                                                                          180
          ComputeError(left,right,top,bottom,-1,led);
          }
         else { /*since left/up not missing is right/down*/
          possiblecornerx = mirrorx[right];
          possiblecornery = mirrory[bottom];
           if(CouldThisBeMissing(possiblecornerx,possiblecornery,xx,xy,yx,yy,originx,originy)){
           FindSomeLeds(left, right, top, bottom, -1, numofblobs, NumOfLedsToFind,
led);
          ComputeError(right,left,bottom,top,-1,led);
          }
                                                                          190
         }
       }/*End no need to switch*/
      /*we have a left right and two edges we have tested the possibity
        that corner is missing now see if it is there*/
      /*For all blobs not left, right top or bottom*/
       for(corner=0;corner<numofblobs;corner++){ if( (corner != left) && (corner !=
right) && (corner != edge) && (corner != nextedge) ){
        if( fabs(mirrorx[corner] - mirrorx[left]) < radiusXtolerance[corner] ){
         /*Corner is on left*/
          if( fabs(mirrory[corner] - mirrory[top]) < radiusXtolerance[corner]
){
          /*corner is on left and top Should it be? only if not switched*/
           if(!needtoswitch){
            FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
            ComputeError(left,right,top,bottom,corner,led);
            }
          }/*end of corner on left top*/
          else  if( fabs(mirrory[corner] - mirrory[bottom]) < radiusXtolerance[corner]
){
                                                                          210
          /*corner is on left bottom should it be? only if switched*/
           if(needtoswitch){
            FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
            ComputeError(bottom,top,left,right,corner,led);
            }
          }/*end of corner on left bottom*/
```

```
            }/*end of corner on left*/
           else if( fabs(mirrorx[corner] - mirrorx[right]) < radiusXtolerance[corner]
){                                                                          220
            /*Corner is on right*/
            if( fabs(mirrory[corner] - mirrory[top]) < radiusXtolerance[corner]
){
            /*corner is on right and top, should it be? only if switched*/
             if(needtoswitch){
              FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
              ComputeError(top,bottom,right,left,corner,ld');          ^B^A   I');      ^B^A   I
 int numofblobs, int numtofind, int ledsfound[MAX_LEDS]);
 int NextBlobOnMirror( int start, int left, int right, int top, int bottom, int
corner, int numofblobs);
 int BlobIsOnMirror( int blobnumber);

 static int lastleft, lastright;
 double besterror;
 int bestleft, bestright, besttop, bestbottom, bestcorner, bestled[MAX_LEDS],
bestfound;
 double *mirrorx, *mirrory;/*Blob x and y coordinates relative to the mirror*/
 double *centx, *centy;/*Temporary Blob x and y coordinates*/
 double *radiusXtolerance;/*Blob radii times the tolerance*/                240
 double *sizes, *radius;
 double leftmostx, rightmostx, topmosty, bottommosty;

 int TargetCosys(Image *image, AllStats *allstats, double tolerance, int NumOfLedsToFind)
 {
  double leftx, rightx, lefty, righty;
  double originx, originy, dist;
  double xx, xy;/*The x axis of the mirror, from left to right*/
  double yx, yy;/*The y axis of the mirror, 90 degrees CCW from x axis*/
  double nx, ny;                                                            250
  double toperror, bottomerror, possiblecornertop].radius;
         bottommosty = -topmosty;
         }
         else {
         top = -1;
         bottom = edge;
         bottommosty = mirrory[bottom] - image->blobs[bottom].radius;
         topmosty = - bottommosty;
         }
         for(corner=0;corner<numofblobs;corner++){ if( (corner != left) && (corner
!= right) && (corner != edge) ){
         /*Does it have the correct y coordinate*/
          if( fabs( fabs(mirrory[corner]) - fabs(mirrory[edge]) ) < radiusXtolerance[corner]
){
```

```
            /*yes it has correct y value does it have a correct x?*/
             if( fabs(mirrorx[corner] - mirrorx[left]) < radiusXtolerance[corner] ){
              /*corner is over/under left*/
               if( (mirrory[corner] > 0.) && !needtoswitch){
               /*corner is left up with no switch needed*/
               FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
               ComputeError(left,right,top,bottom,corner,led);
               }
               else if( (mirrory[corner] < 0.) && needtoswitch){
               /*corner is left down with switch needed*/
               FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
               ComputeError(bottom,top,left,right,corner,led);
               }
             }/*end of corner at left*/
            else if( fabs(mirrorx[corner] - mirrorx[right]) < radiusXtolerance[corner]
){
             /*corner is over/under right*/
              if( (mirrory[corner] > 0.) && needtoswitch){
              /*corner is right and up with switch needed*/
              FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
              ComputeError(top,bottom,right,left,corner,led);
              }
              else if( (mirrory[corner] < 0.) && !needtoswitch){
              /*corner is right down with no switch needed*/
              FindSomeLeds(left, right, top, bottom,corner, numofblobs, NumOfLedsToFind,
led);
              ComputeError(right,left,bottom,top,corner,led);
              }
            }/*end of corner at right*/
          }/*end of correct y*/
        }}/*finished searching over all possible corners*/
       }/*End of opposite edge could be missing*/
     }/*End of processing one edge find*/
    }}/*End search of all blobs that are not left and right*/
  }/*end if distance between left and right is non zero*/
}/*end of picking two at random*/
FREE(mirrorx);
FREE(mirrory);
FREE(sizes);
FREE(radius);
FREE(radiusXtolerance);
 if(bestfound){
  allstats->id.left = bestleft;
  allstats->id.right= bestright;
```

```
    allstats->id.top= bcsttop;
    allstats->id.bottom= bcstbottom;
    allstats->id.corncr= bcstcornc;
     for(i=0;i<MAX_LEDS;i++){
      allstats->id.lcd[i]= bcstlcd[i];
      }
     return 0;
    }
   else{
    allstats->id.lcft = -1;
    allstats->id.right= -1;
    allstats->id.top= -1;
    allstats->id.bottom= -1;
    allstats->id.corncr= -1;
     for(i=0;i<MAX_LEDS;i++){
      allstats->id.lcd[i]= -1;
      }
     return 1;
    }
 }/*end of program*/

 int CouldThisBcMissing( double x, double y, double xx, double xy, double yx,
 double yy, double ox, double oy)
 {
  int sx, sy;
 sx = ( int) (x*xx + y*yx + ox);
 sy = ( int) (x*xy + y*yy + oy);
  if( (sx < 0) || (sx > IMAGEWIDTH) ) return 1;
  if( (sy < 0) || (sy > IMAGEHEIGHT) ) return 1;
  return 0;
 }

 void ComputcError( int lcft, int right, int top, int bottom, int cornc, int
lcd[MAX_LEDS] )
 {
  int i;
  double thiscrror;
 thiscrror =mag(mirrorx[cornc] - mirrorx[lcft], mirrory[cornc]-mirrory[top])
+
        fabs(mirrorx[top]) +
        mag(mirrorx[bottom], mirrory[bottom] + mirrory[top]) +
        fabs(mirrory[lcft]) +
        mag(mirrorx[right] + mirrorx[lcft], mirrory[right]);
 if(thiscrror < bcstcrror){
   bcstcrror = thiscrror;
   bcstlcft = lcft;
   bcstright = right;
```

```
    besttop = top;
    bestbottom = bottom;
    bestcorner = corner;
    bestfound = 1;
     for(i=0;i<MAX_LEDS;i++) bestled[i] = led[i];
    }
 }

 void FindSomeLeds( int left, int right, int top, int bottom, int corner, int
numofblobs, int numtofind, int ledsfound[MAX_LEDS])
 {
  int i;
  int lastonefound;
 lastonefound = -1;
  for(i=0;i<MAX_LEDS;i++)ledsfound[i]=-1;
  for(i=0;i<numtofind;i++){
   ledsfound[i] = NextBlobOnMirror(lastonefound+1, left, right, top, bottom,
corner, numofblobs);
    if(ledsfound[i]==-1) return;
    else lastonefound = ledsfound[i];
   }
 }

 int NextBlobOnMirror( int start, int left, int right, int top, int bottom, int
corner, int numofblobs)
 {
  int i;
  for(i=start;i<numofblobs;i++) if( (i != left) && (i != right) && (i != top)
&& (i != bottom) && (i != corner) ){
    if(BlobIsOnMirror(i)) return i;
   }
  return -1;
 }

 int BlobIsOnMirror( int i)
 {
  if( (mirrorx[i] > leftmostx) && (mirrorx[i] < rightmostx) && (mirrory[i] <
topmosty) && (mirrory[i] > bottommosty) ) return 1;
  return 0;
 }

/*bx,by are coordinates relative to b, ox,oy are origin relative to b,
vx,vy is unit vector along new x relative to b, px,py is unit vector along
new y relative to b, nx,ny are new coordinates relative to new x and new y*/
 void ConvertCoordinates( double bx, double by, double ox, double oy, double vx,
 double vy, double px, double py, double *nx, double *ny)
 {
```

```
  double o2bx, o2by;
  o2bx = bx - ox;
  o2by = by - oy;
  *nx = o2bx * vx + o2by * vy;
  *ny = o2bx * px + o2by * py;
  }

 /*Return 0 if there are no more permutations. 1 otherwise*/
  int PickTwoAtRandom( int *left, int *right, int max)
  {
   if(lastright < max){
    lastright++;
    *right = lastright;
    *left = lastleft;
     return 1;
    }
   else if(lastleft < max){
    lastleft++;
    lastright = lastleft;
    lastright++;
    *right = lastright;
    *left = lastleft;
    if(lastright > max) return 0;
    else return 1;
   }
  else return 0;
 }

 double mag( double x, double y)
 {
  return sqrt(x*x + y*y);
 }

 int Blob2IsBiggerThan1(Image *image, int one, int two)
 {
  if (image->blobs[two].size > image->blobs[one].size ) return -1;
  return 0;
 }
```

Chapter 6

target.c, target.h

Documentation Date: 3/10/95

6.1 Perform Target Calculations.

The routines in this file compute things related to a target.
New Data Types:
None.
Definitions:

1. LINEHALFLENGTH = 10, Lines drawn on the screen are double this value in pixels long.

6.1.1 Header File Listing:

```
/*****************************************
*  target.H                    *
*****************************************/
#ifndef TARGET_H
  #define TARGET_H
  /*defines the target data structure*/
  void LoadTargetFromStats(Image *image, Target *target, AllStats *allstats);
  /*Marks the target on page */
  /* In addition to marking the blobs (blob marks are set by radius)
       it marks center of screen. Screen middle is given by
       middlex and middley. Center Mark lengths are 2*LINEHALFLENGTH */
  #define LINEHALFLENGTH 10
  void Mark_Target (Target *target, int page, int middlex, int middley);
  void MarkAllBlobs(Image image, int page);
  /*returns last three numbers given first*/
  void FindFramesForDoubleSubtract( int FirstFallingEdge, int *FirstBright,
int *Dim, int *SecondBright);
  /*finds an open frame one that isn't first bright, dim or second bright*/
  int FindStorageImage( int FirstFallingEdge);
```

```
/*Does the blob analysis and loads the image data structure*/
int DefineImage( int FirstOn, int Off, int SecondOn, int storepage, Image
*theimage, long LargestToFind, long SmallestToFind, int thresh);
/*Subtract FirstOn minus Dim Plus SecondOn
(if SecondOn<0 use single subtract)
(if Dim<0 don't subtract)
using our double subtract algorithm,
return the histogram in hist, only consider the ROI area of the image*/
void SubAndHistogram( int FirstOn, int Dim, int SecondOn, Histogram hist,
ROI roi);
/*Subtract FirstOn minus Dim Plus SecondOn
(if SecondOn<0 use single subtract)
(if Dim<0 don't subtract)
using our double subtract algorithm,
return the Fullhistogram in hist, only consider the ROI area of the image.
The difference is
this and regular histogram is the bins range from -PIXEL_MAX to +PIXEL_MAX
*/
void FullSubAndHistogram( int FirstOn, int Dim, int SecondOn, FullHistogram
hist, ROI roi);
/*The following computes statistics of a selected area*/
void ComputeStatistics( int FirstOn, int Dim, int SecondOn, ROI roi,
DPixel *Brightest, DPixel *Dimmest, DPixel *Average, DPixel *StdDev,
long *NumberOfPixels);
/*The following routines are not currently used*/
/*The following is useful if no subtraction is needed*/
void DoHistogram( int Frame, Histogram hist, ROI roi);
/*
This will subtract buffer1 - buffer2 put the result in Output
Output = (Buffer1 - Buffer2)/DivideBy + OffsetToAdd
*/
void Subtract_Images ( int FirstOn, int Off, int SecondOn, int Store, int
DivideBy, int OffsetToAdd, ROI roi);
double PositionDifference(OneBlob *one, OneBlob *two);

/*Puts an O around the blob the size of O depends on the radius*/
void MarkOneRetro(OneBlob *blob, int page);
#endif
```

6.2 Function: Global Target Routines

Documentation Date: 3/10/95

Prototypes:

```
void LoadTargetFromStats(Image *image, Target *target, AllStats *allstats);
   void Mark_Target (Target *target, int page, int middlex, int middley);
   void FindFramesForDoubleSubtract(int FirstFallingEdge, int *FirstBright,
int *Dim, int *SecondBright);
   int FindStorageImage(int FirstFallingEdge);
   int DefineImage(int FirstOn, int Off, int SecondOn, int storepage,
Image *theimage, long LargestToFind, long SmallestToFind, int thresh);
   void SubAndHistogram(int FirstOn, int Dim, int SecondOn, Histogram
hist, ROI roi);
   void FullSubAndHistogram(int FirstOn, int Dim, int SecondOn, FullHistogram
hist, ROI roi);
   void MarkOneRetro(OneBlob *blob, int page);
   void ComputeStatistics(int FirstOn, int Dim, int SecondOn, ROI
roi, DPixel *Brightest, DPixel *Dimmest, DPixel *Average, DPixel *StdDev,
long *NumberOfPixels);
   void MarkAllBlobs(Image image, int page);
   double PositionDifference(OneBlob *one, OneBlob *two);
   void DoHistogram(int Frame, Histogram hist, ROI roi);
   void Subtract_Images (int FirstOn, int Off, int SecondOn, int Store,
int DivideBy, int OffsetToAdd, ROI roi);
```

Source File: *target.c*
Type of Function: User Callable
Header Files Used in target.c: *<math.h> <stdio.h> "datatype.h" "target.h" "targa8.h" "misc.h" "blob.h"*

Description:

There are several routines in this file. The user callable ones are:

1. FindStorageImage - The storage image is the targa page that can be used to hold an image so the user can see the processing as it occurs. Due to hardware problems, the leds blink On, On, Off, Off etc. (see Figure 6.1). The second On is used in Leo's double subtract but the second Off is never used so it is the Storage Image. Since firstfallingedge is the image number of the second On, the second off is two more than that.

2. FindFramesForDoubleSubtract - The standard double subtract uses the first on, first off, and the next first on.

3. DefineImage - Calls the blob analysis routine and loads the image data structure.

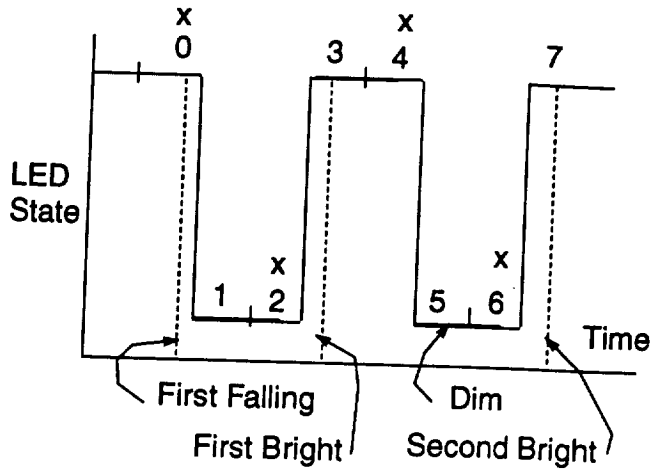

Figure 6.1: Blink Pattern of LEDs, Second Off is Never Used.

4. SubAndHistogram - This routine subtracts images and performs a histogram. It is capable of double subtract (if SecondOn > -1) but not showing the results. ROI defines the region of interest, only rows and columns in the ROI are processed. The histogram is stored as an array. The double subtract histogram probably does not work, it has never been used. For double subtract the histogram computes (Firston + Secondon - 2 Dim) which can be -510 to +510. We want this to be an array index so it should be converted to a range between 0 and NUMBEROFPIXELBINS-1 (presently 256-1 = 255). If a value outside the range is stored a pointer conflict can result.

   For Single subtract compute (Firston - Dim) which can be from -255 to +255. We store this as an array index between 0 and 255 by dividing by 2 (makes the range -127 to +127) and adding 128 (range is 1 to 255). This dividing and adding business should be generalized so NUMBEROFPIXELBINS can be changed at will.

5. FullSubAndHistogram - This routine subtracts images and performs a histogram. It differs from SubAndHistogram in the following way. When an image is subtracted, the result can be from -255 to +255 (-PIXEL_MAX, +PIXEL_MAX) but SubAndHistogram converts the subtracted image into a range from 0 to 255, which means compacts it by dividing by 2 (makes the range -127 to +127) then adding 128. This routine (FullSubAndHistogram) does not divide by 2 nor add anything.

   It is capable of double subtract (if SecondOn > -1) but has not been tested in this mode. ROI defines the region of interest, only rows and columns in the ROI are processed. The histogram is stored as an array. The double subtract histogram probably does not work, it has never been used. For double subtract the histogram computes (Firston + Secondon - 2 Dim)/2 which can be -255 to +255 (-PIXEL_MAX, +PIXEL_MAX).

For Single subtract compute (Firston - Dim) which can be from -255 to +255 (-PIXEL_MAX, +PIXEL_MAX).

For no subtract the histogram ranges from 0 to 255 (0, +PIXEL_MAX).

We store the histogram as an array index between 0 and FULLNUMBEROFPIXELBINS which is currently PIXEL_MAX *2 +1 or 0 to 511. Index (array value) 0 corresponds to the number of pixels with intensity of -255 (-PIXEL_MAX) and index 511 is the number at intensity +255 (+PIXEL_MAX).

The code is written so that PIXEL_MAX can be changed at will and the routine should continue to run. We have never changed PIXEL_MAX however to test this.

6. LoadTargetFromStats - The image data structure contains the blob information, allstats contains information about which blob is top, etc. Target is loaded with the blob information. There should probably be a better way to do this instead of copying. If the id in allstats is = -1 it means the blob is missing.

   The AllStats data structure contains information about which blobs are retros and which ones are likely to be LEDs. The target data structure wants a single centroid location for the LED so this routine converts the number of LED's into a single LED blob. The algorithm is not fancy and if the number of LED's found differs from what is expected the routine says none were found.

7. Mark_Target - This routine puts an LED mark (an X with one white line and one black line so it shows up) on the LED blob. It draws an LED mark on the corner retro also. It also draws Retro marks (circles with radius equal to the blob radius, one white and one just inside it with black). It also puts a + at the 0,0 location.

8. MarkOneRetro - Draws a white and black circle over a blob. The black circle has radius equal to the blob, the white's radius is 2 pixels larger.

9. ComputeStatistics - This routine will compute the magnitude of the Brightest and Dimmest intensity, the average intensity, the standard deviation of intensity and the number of pixels in the region of interest. It works with no subtraction, single subtraction and double subtraction.

10. MarkAllBlobs - Draws a circle around all the blobs. It uses MarkOneRetro to do the marking.

11. PositionDifference - Computes the distance in pixels between two blobs.

12. DoHistogram - Computes the histogram of an image.

13. Subtract_Images - Subtracts two images and stores it. Only works on pixels inside the ROI.

6.3 Function: Local Target Routines

Documentation Date: 3/10/95

Prototypes:

```
void MarkOneLED(OneBlob *blob, int page);
   void CopyOneBlob(OneBlob *from, OneBlob *to);
   double TotalSizeOfAllBlobs(OneBlob blobs[], int numblobs);
   void Integrate_Image (int Frame, double *sum, ROI roi);
```

Source File: *target.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in target.c: *<stdio.h> <math.h> "datatype.h" "target.h" "targa8.h" "misc.h" "blob.h"*

Description:

There are several internal routines in this file this section describes all but one. The internal routines described are:

1. TotalSizeOfAllBlobs - adds up the size of all blobs.
2. CopyOneBlob - copies one blob into another.
3. MarkOneRetro - draws a white and black circle over a blob.
4. MarkOneLED - draws a white and black x over a blob.
5. Integrate_Image - Sums the intensity of all pixels inside the ROI to determine the total brightness.

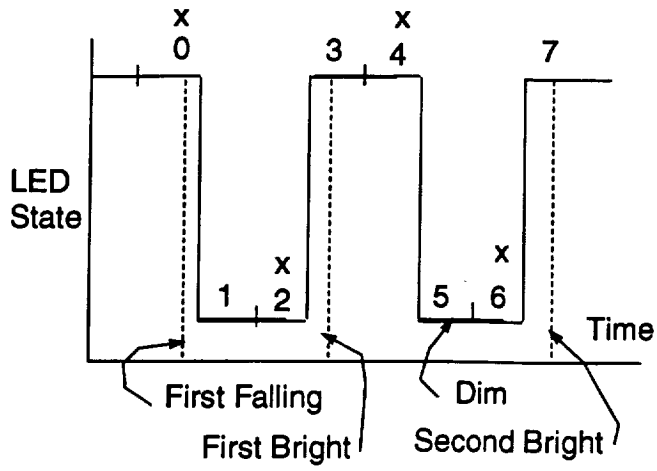

Figure 6.2: Eight Frames Grabbed by the System.

6.4 Function: FindFramesForDoubleSubtract

Documentation Date: 6/24/94

Prototypes:

```
void FindFramesForDoubleSubtract(int FirstFallingEdge, int *FirstBright,
int *Dim, int *SecondBright);
```

Source File: *target.c*
Type of Function: User Callable
Header Files Used in target.c: *<math.h> "datatype.h" "target.h" "targa8.h" "misc.h" "blob.h"*

Description:

This routine determines which image frames are On and which are Dim. It locates two On frames and the Off frame between them. FirstFallingEdge (passed to the routine) is used for the calculation.

6.4.1 Theory

The program grabs eight frames. Two consecutive frames have the LED on. The first of the two has the LED on brighter than the second (a hardware problem). After two on frames, there are two consecutive off frames. The second off however has the LED on dimly (hardware problems). Suppose frames have been grabbed as figure 6.2 shows. The figure shows frames 3 and 4 as a consecutive set of On frames. Frame 0, is what is called the first falling frame (defined elsewhere). What this routine does is set FirstBright = 3, Dim = 5, and SecondBright = 7. If the FirstFallingFrame is > 0 then FirstBright = FirstFalling -1.

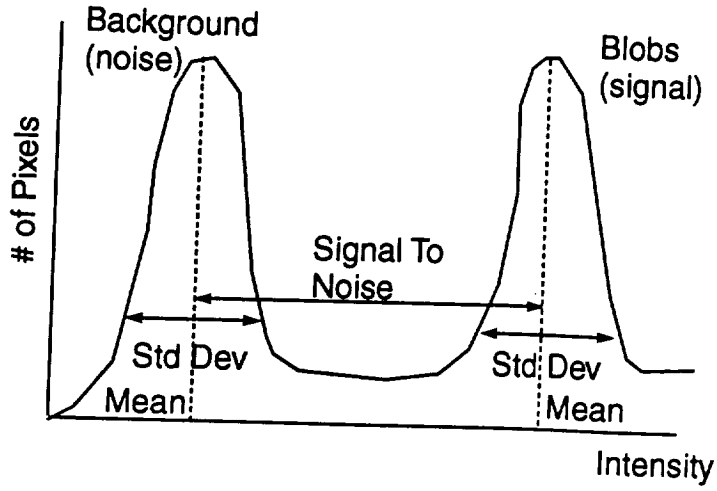

Figure 6.3: A Hypothetical Histogram.

6.5 Function: ComputeSignalToNoise

Documentation Date: 6/24/94

Prototypes:

```
void ComputeSignalToNoise(Image *image);
```

Source File: *target.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in target.c: *<math.h> "datatype.h" "target.h" "targa8.h" "misc.h" "blob.h"*

Description:

This routine computes a number which it stores in the image data structure. The number is supposed to indicate a degree of confidence in the blobs that were found. The greater the number the more confidence. It is not a probability therefore the number can be > 1.

6.5.1 Theory

Once the blobs have been determined in an image, every pixel belongs to either a blob or the background. Suppose you draw a histogram of the image. Ideally it would look something like figure 6.3 We know which pixels belong the blobs and which are background. We compute the mean and standard deviation of both distributions. The Blob hump is considered the signal, the background hump is considered the noise. The Signal to Noise value is defined as the Intensity distance between (Blob Mean - 1 Blob Standard Deviation) and (Background Mean + 1 Background Standard Deviation). If the Signal to Noise value is large then the pixels considered to be blobs are "separated"

from the background. If it is small, the blobs are less distinguishable from background.

6.5.2 Program Listing:

```
/*@@@*************************************************************************
                    target.C
*************************************************************************/
#include <math.h>
#include <stdio.h>
#include "datatype.h"
#include "target.h"
#include "targa8.h"
#include "misc.h"
#include "blob.h"                                                    10


  /*Puts an X over the blob the size of X depends on the blob radius*/
   void MarkOneLED(OneBlob *blob, int page);
  /*Puts an O around the blob the size of O depends on the radius*/
   void CopyOneBlob(OneBlob *from, OneBlob *to);
   double TotalSizeOfAllBlobs(OneBlob blobs[], int numblobs);
   void ComputeSignalToNoise(Image *image);

/*The storage image is the targa page that can be used to hold an      20
image so the user can see the processing as it occurs. Due to hardware
problems we blink the led On, On, Off, Off etc. The
second On is sometimes used in Leo's double subtract but the second Off
is never used so pick it as the Storage Image. Since firstfallingedge is
the image number of the second On, then the second off is two more than
that.*/
 int FindStorageImage( int FirstFallingEdge)
  {
   return(FirstFallingEdge + 2);
      /*This is the bad off*/                                          30
  }

/*The standard double subtract uses the first on, first off, and the
next first on.*/
 void FindFramesForDoubleSubtract( int FirstFallingEdge, int *FirstBright, int
*Dim, int *SecondBright)
    {
     if(FirstFallingEdge == 0)*FirstBright = 3;
     else *FirstBright = FirstFallingEdge - 1;
    *Dim = *FirstBright + 2;                                         40
    *SecondBright = *Dim + 2;
    }

ImageLine firstonline,offline,secondonline,storeline;/*This produces a stack
overflow unless static*/
```

```
/*This routine calls the blob analysis routine*/
 int DefineImage( int FirstOn, int Off, int SecondOn, int storepage, Image *theimage,
 long LargestToFind, long SmallestToFind, int thresh)
   {                                                                    50
    int error;
   DPixel neg,pos;
   OneBlob negblobs[5];/*Changed from 1 to debug*/
    int numofnegtofind;
   ROI roi;
   WHOLE_IMAGE(roi);
   neg = (DPixel) -thresh;
   pos = (DPixel) thresh;
   theimage->posthreshold = pos;
   theimage->negthreshold = neg;                                        60
   theimage->num_of_blobs_found = MAX_BLOBS_PER_PICTURE;
   numofnegtofind = 0;
   error = ExtractBlobs(FirstOn, Off, SecondOn, storepage, &numofnegtofind,
neg, negblobs, &theimage->num_of_blobs_found, pos, theimage->blobs, &theimage->background,
LargestToFind, SmallestToFind, roi);
   ComputeSignalToNoise(theimage);
    return(error);
   }

/*This routine subtracts images and performs a histogram. It is capable   70
of double subtract (if SecondOn > -1) but not showing the results. ROI
defines the region of interest.*/
 void SubAndHistogram( int FirstOn, int Dim, int SecondOn, Histogram hist, ROI
roi)
   {
    int row,col,i;
   DPixel temp;
   /*The histogram must be an array. Set it zero at first*/
    for(i=0;i<NUMBEROFPIXELBINS;i++)hist[i]=0;
   /*For all rows in the ROI*/                                            80
    for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
     GetLine(row,FirstOn,firstonline);
      if(Dim > -1)GetLine(row,Dim,offline);
      if(SecondOn > -1)GetLine(row,SecondOn,secondonline);
      for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
      /*For double sub. compute (Firston + Secondon - 2 Dim), this
can be -510 to +510. We want this to be an array index so it should be
converted to a range between 0 and NUMBEROFPIXELBINS. This routine
should be corrected. Double subtract has never been used in histogram so
perhaps this is why this bug was never found.*/                          90
       if(SecondOn > -1)temp = max (( ( (DPixel) firstonline[col] - 2*(DPixel) offline[col]
+ (DPixel) secondonline[col] )/2+ 128),0);
      /*Single subtract Firston - Dim can be from -255 to +255.  We store
```

```
this as an array index between 0 and 255 so divide by 2 (makes the range
-127 to +127) and add 128 (range is 1 to 255).      This is not too good
because the index should range from 0 to NUMBEROFPIXELBINS-1. It does
in this case since NUMBEROFPIXELBINS=256 but it should be more
general.*/
        else if (Dim > -1) temp = max (( ( (DPixel) firstonline[col] - (DPixel) offline[col]
)/2+ 128),0);
        else temp = firstonline[col];                                          100
 #ifdef CHECKLENGTH
 if(( int)temp>=NUMBEROFPIXELBINS)Fatal_Error_Message("Blew it in Subandhistogram");
 #endif
       hist[ ( int) temp ]++;
       }
      }
   }

 void FullSubAndHistogram( int FirstOn, int Dim, int SecondOn, FullHistogram hist,
ROI roi)
   {
    int row,col,i;
   DPixel temp;
    for(i=0;i<FULLNUMBEROFPIXELBINS;i++)hist[i]=0;
   /*
   for(i=0;i<FULLNUMBEROFPIXELBINS;i++)printf("_%d|%d",i,hist[i]);
   */
    for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
     GetLine(row,FirstOn,firstonline);
      if(Dim>-1)GetLine(row,Dim,offline);                                      120
      if(SecondOn > -1)GetLine(row,SecondOn,secondonline);
      for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
       if(SecondOn > -1)temp = ( (DPixel) firstonline[col] - 2*(DPixel) offline[col]
+ (DPixel) secondonline[col] )/2;
       else if(Dim>-1)temp = (DPixel) firstonline[col] - (DPixel) offline[col] ;
       else temp = (DPixel) firstonline[col];
 #ifdef CHECKLENGTH
 if((( int)temp+PIXEL_MAX)>=FULLNUMBEROFPIXELBINS)Fatal_Error_Message("Blew it
in FullSubandhistogram");
 #endif                                                                        130
      hist[ ( int) temp + PIXEL_MAX]++;
      }
      /*
  for(i=0;i<FULLNUMBEROFPIXELBINS;i++)printf("_%d|%d",i,hist[i]);
  printandwait("ready");
  */
    }
  }
                                                                              140
```

```
 void ComputeStatistics( int FirstOn, int Dim, int SecondOn, ROI roi,
DPixel *Brightest, DPixel *Dimmest, DPixel *Average, DPixel *StdDev,
 long *NumberOfPixels)
 {
  double sum, sumsq, dave;
  int numrows, numcols;
 DPixel maximum, minimum;
 DPixel temp;
  int row,col;
 maximum = -PIXEL_MAX;
 minimum = PIXEL_MAX;
 numrows = numcols = 0;
 sum = sumsq = 0.;
  for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
      numrows++;
      GetLine(row,FirstOn,firstonline);
       if(Dim > -1) GetLine(row,Dim,offline);
       if(SecondOn > -1)GetLine(row,SecondOn,secondonline);
      numcols=0;
       for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
       numcols++;
        if( (Dim > -1) && (SecondOn > -1) )temp = ( (DPixel) firstonline[col] - 2*(DPixel)
offline[col] + (DPixel) secondonline[col] )/2;
        else  if(Dim > -1) temp = (DPixel) firstonline[col] - (DPixel) offline[col]
;
        else temp = (DPixel) firstonline[col];
       maximum = max(maximum,temp);
       minimum = min(minimum,temp);
       sum = sum + ( double) temp;
       sumsq = sumsq + ( double) temp * ( double) temp;
       }
      }
*NumberOfPixels = numcols * numrows;
dave = (sum/ ( double)*NumberOfPixels);
*Average = (DPixel) dave;
*StdDev = (DPixel) sqrt(sumsq/ ( double)*NumberOfPixels - dave * dave);
*Brightest = maximum;
*Dimmest = minimum;
}

/*The image data structure contains the blob information, allstats
contains information about which blob is top, etc. Target is loaded with
the blob information. There should probably be a better way to do this
instead of copying. If the id in allstats is = -1 it means the blob is
missing.*/
 void LoadTargetFromStats(Image *image, Target *target, AllStats *allstats)
  {
```

```
    int counter,numofledsfound;
    double tempsize;
   target->num_of_led = 0;                                                    190
   target->led.size = 0.;
   target->led.gray = 0.;
   target->led.centx = 0.;
   target->led.centy = 0.;
   numofledsfound = 0;
    for(counter=0;(allstats->id.led[counter]>-1)&&(counter<allstats->id.NormalNumOfLeds);counte
    numofledsfound++;
    tempsize = image->blobs[allstats->id.led[counter]].size;
    target->led.size = target->led.size + tempsize;
    target->led.gray = target->led.gray + tempsize*image->blobs[allstats->id.led[counter]].gray;
    target->led.centx = target->led.centx + image->blobs[allstats->id.led[counter]].centx;
    target->led.centy = target->led.centy + image->blobs[allstats->id.led[counter]].centy;
    }
    printf("counter is %d",numofledsfound);
   if(numofledsfound == allstats->id.NormalNumOfLeds){
    target->num_of_led = 1;
    target->led.gray = target->led.gray/target->led.size;
    target->led.centx = target->led.centx/numofledsfound;
    target->led.centy = target->led.centy/numofledsfound;
    }                                                                         210
 target->led.radius = 1.;
 target->led.variance = 0.;

 target->NumOfOdd = 0;
  if(allstats->id.corner > -1){
   CopyOneBlob(&image->blobs[allstats->id.corner],&target->OddRetro);
   target->NumOfOdd = 1;
   }
 target->left = 0;
  if(allstats->id.left > -1){                                              220
   CopyOneBlob(&image->blobs[allstats->id.left],&target->leftretro);
   target->left = 1;
   }
target->right = 0;
 if(allstats->id.right > -1){
  CopyOneBlob(&image->blobs[allstats->id.right],&target->rightretro);
  target->right = 1;
  }
target->top = 0;
 if(allstats->id.top > -1){                                                230
  CopyOneBlob(&image->blobs[allstats->id.top],&target->topretro);
  target->top = 1;
  }
target->bottom = 0;
```

```
    if(allstats->id.bottom > -1){
     CopyOneBlob(&image->blobs[allstats->id.bottom],&target->bottomretro);
     target->bottom = 1;
     }
   }

/*This routine puts an LED mark (an X with one white line and one
black line so it shows up) on the LED blob. It draws an LED mark on the
corner retro also. It also draws Retro marks (circles with radius equal
to the blob radius, one white and one just inside it with black). It
also puts a + at the 0,0 location.*/
 void Mark_Target (Target *target, int page, int middlex, int middley)
   {
   line(page,middlex-LINEHALFLENGTH,middley,middlex+LINEHALFLENGTH,middley,WHITE
   line(page,middlex,middley-LINEHALFLENGTH,middlex,middley+LINEHALFLENGTH,WHITE
    if(target->left) MarkOneRetro(&target->leftretro,page);
    if(target->top) MarkOneRetro(&target->topretro,page);
    if(target->right) MarkOneRetro(&target->rightretro,page);
    if(target->bottom) MarkOneRetro(&target->bottomretro,page);
    if(target->num_of_led) MarkOneLED(&target->led,page);
    if(target->NumOfOdd){
     MarkOneLED(&target->OddRetro,page);
     MarkOneRetro(&target->OddRetro,page);
     }
   }

 void MarkAllBlobs(Image image, int page)
 {
  int i;
  for(i=0;i<image.num_of_blobs_found;i++){
    MarkOneRetro(&image.blobs[i], page);
    }
 }

/************
 Local Routines
 ************/
/*This routine adds up the size of all blobs.*/
 double TotalSizeOfAllBlobs(OneBlob blobs[], int numblobs)
   {
    int i;
    double returnvalue;
   returnvalue = 0;
    for(i=0;i<numblobs;i++){
     returnvalue = returnvalue + blobs[i].size;
     }
    return returnvalue;
```

```
  }

void ComputeSignalToNoise(Image *image)
 {
 /*This needs a better algorithm to compute the background's variance, the
shortcut formula does not work*/
  int i;
  double totalsize;
  double totalvariance, totalaverage;                                    290
 image->SignalToNoiseMargin = 0;
  return;
  if(image->num_of_blobs_found == 0){
   image->SignalToNoiseMargin = -image->background.gray - sqrt(image->background.variance);
    return;
   }
 totalvariance = 0.;
 totalaverage = 0.;
 /* Add all blob sizes together */
 totalsize = TotalSizeOfAllBlobs(image->blobs,image->num_of_blobs_found);300
 /* For each blob add its contribution to the distribution's mean and
   variance */
  for(i=0;i<image->num_of_blobs_found;i++){
   totalvariance = totalvariance + image->blobs[i].size*image->blobs[i].variance;
   totalaverage = totalaverage + image->blobs[i].size*image->blobs[i].gray;
   }
 /* Compute the Blob mean and Standard Deviation */
 totalvariance = totalvariance / totalsize;
 totalaverage = totalaverage / totalsize;
 /* Value is Intensity difference between:                              310
   (Blob Mean - Blob Std. Dev.) and
   (Back Mean + Back Std. Dev.) */
 image->SignalToNoiseMargin = ((totalaverage - sqrt(totalvariance)) - (image->background.gray
+ sqrt(image->background.variance)));
 }

/* This routine copies one blob into another */
void CopyOneBlob(OneBlob *from, OneBlob *to)
  {
  to->size = from->size;                                                 320
  to->centx = from->centx;
  to->centy = from->centy;
  to->gray = from->gray;
  to->radius = from->radius;
  to->variance = from->variance;
  }

/*This routine draws a white and black circle over a blob*/
```

```
 void MarkOneRetro(OneBlob *blob,  int page)
   {/*fix to be general*/                                          330
   circle(page, ( int) blob->centx, ( int) blob->centy, ( int) blob->radius, BBLACK);
   circle(page, ( int) blob->centx, ( int) blob->centy, ( int) (blob->radius +
2), WHITE);
   }

/*this routine draws a white and black x over a blob*/
 #define HALFSIZE 2
 #define FULLSIZE 4
 void MarkOneLED(OneBlob *blob,  int page)
   {                                                               340
    int startlightx,startlighty,startdarkx,startdarky;
    int endlightx,endlighty,enddarkx,enddarky;

   startlightx = (( int) blob->centx) - ( int) blob->radius * HALFSIZE;
   endlightx = startlightx + ( int) (blob->radius * FULLSIZE);

   startlighty = (( int) blob->centy) - ( int) blob->radius * HALFSIZE;
   endlighty = startlighty + ( int) (blob->radius * FULLSIZE);

   startdarkx = ( int) blob->centx + ( int) blob->radius * HALFSIZE;   350
   enddarkx = startdarkx - ( int) (blob->radius * FULLSIZE);

   startdarky = ( int) blob->centy - ( int) blob->radius * HALFSIZE;
   enddarky = startdarky + ( int) (blob->radius * FULLSIZE);
   line(page, startlightx, startlighty, endlightx, endlighty, WHITE);
   line(page, startdarkx, startdarky, enddarkx, enddarky, BBLACK);
   }

/*The following are not currently used*/
/*****************************************                          360
*  Histogram of IMAGE              *
*****************************************/
 void DoHistogram( int Frame, Histogram hist, ROI roi)
   {
    int row,col;
   ImageLine line;
    int i;
    for(i=0;i<NUMBEROFPIXELBINS;i++)hist[i]=0;
    for(row=roi.ys;row<roi.yc;row=row+roi.resolution){
     GetLine(row,Frame,line);                                      370
      for(col=roi.xs;col<roi.xc;col=col+roi.resolution){
#ifdef CHECKLENGTH
if(( int)line[col]>=NUMBEROFPIXELBINS)Fatal_Error_Message("Blew it in Dohistogram");
#endif
      hist[ ( int) line[col] ]++;
```

```
    }
   }
  }

/*****************************************
* SUBTRACT_IMAGES        *
*****************************************
void Subtract_Images (int FirstOn, int Off, int SecondOn, int Store, int DivideBy,
int OffsetToAdd, ROI roi)
   {
   int row,col;
   DPixel temp;

  //Older subtract routine that should be slower with "if" statements
  //buried in the loop
  //This will subtract buffer1 - buffer2 put the result in Output
  // Output = (Buffer1 - Buffer2)/DivideBy + OffsetToAdd

   for(col=0;col<IMAGEWIDTH;col++)storeline[col]=0;
   for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
        GetLine(row,FirstOn,firstonline);
        if(Off>-1)GetLine(row,Off,offline);
        if(SecondOn > -1)GetLine(row,SecondOn,secondonline);
        for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
            if(SecondOn > -1)temp = max (( ( (DPixel) firstonline[col] - 2*(DPixel) offline[col]
+ (DPixel) secondonline[col] )/DivideBy + OffsetToAdd),0);
            else if(Off>-1)temp = max (( ( (DPixel) firstonline[col] - (DPixel) offline[col]
)/DivideBy + OffsetToAdd),0);
            else temp = (DPixel) firstonline[col];
            storeline[col] = (Pixel) temp;
            }
        PutLine(row,Store,storeline);
        }
  }
   */
/*****************************************
* SUBTRACT_IMAGES        *
*****************************************/
void Subtract_Images ( int FirstOn, int Off, int SecondOn, int Store, int DivideBy,
int OffsetToAdd, ROI roi)
  {
   int row,col;
  DPixel temp;
  /*
  This will subtract buffer1 - buffer2 put the result in Output
  Output = (Buffer1 - Buffer2)/DivideBy + OffsetToAdd
  */
```

```
    for(col=0;col<IMAGEWIDTH;col++)storeline[col]=0;      //Do we need this?
    if( (Off>-1) && (SecondOn<0) ){  //Single subtract
     for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
     GetLine(row,FirstOn,firstonline);
     GetLine(row,Off,offline);
      for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
       temp = max (( ( (DPixel) firstonline[col] - (DPixel) offline[col] )/DivideBy
+ OffsetToAdd),0);                                                    430
       storeline[col] = (Pixel) temp;
     }
     PutLine(row,Store,storeline);
     }
   }

    if( (Off>-1) && (SecondOn>-1) ){      //Double subtract
     for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
     GetLine(row,FirstOn,firstonline);
     GetLine(row,Off,offline);                                        440
     GetLine(row,SecondOn,secondonline);
      for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
       temp = max (( ( (DPixel) firstonline[col] - 2*(DPixel) offline[col] + (DPixel)
secondonline[col] )/DivideBy + OffsetToAdd),0);
       storeline[col] = (Pixel) temp;
     }
     PutLine(row,Store,storeline);
     }
   }
    if( (Off<0) && (SecondOn<0) ){     //No subtraction at all, Why??   450
     for(row=roi.ys;row<roi.ye;row=row+roi.resolution){
     GetLine(row,FirstOn,firstonline);
      for(col=roi.xs;col<roi.xe;col=col+roi.resolution){
       temp = (DPixel) firstonline[col];
       storeline[col] = (Pixel) temp;
     }
     PutLine(row,Store,storeline);
     }
   }
} //End of subtraction routine                                        460
/*****************************************
*  Integrate_Image*
*****************************************/
/*This routine integrates the gray scale values over an entire image
  to indicate the total brightness of the image*/

void Integrate_Image ( int Frame,  double *sum, ROI roi)
  {
   int row,col;
```

```
   *sum=0;                                                          470
    for(row=roi.ys;row<roi.yc;row=row+roi.resolution){
     GetLine(row,Frame,firstonline);
      for(col=roi.xs;col<roi.xc;col=col+roi.resolution){
      *sum = *sum + ( ( double) firstonline[col] );
     }
   }
   }


/* This routine returns the distance between two blobs */           480
double PositionDifference(OneBlob *one, OneBlob *two)
  {
   return( sqrt( square(one->centx-two->centx) + square(one->centy-two->centy)
) );
  }
```

Chapter 7

memory.c, memory.h

Documentation Date: 6/27/94

7.1 Memory Management Functions

The routines in this file perform memory management.
New Data Types:
None.
Definitions:

1. MALLOC - define as MyMalloc if you want memory debugging functions. Define it as malloc if you do not.
2. FREE - define as MyFree if you want memory debugging. Define it as free if you do not.

7.1.1 Header File Listing:

```
#ifndef MEMORY_H
  #define MEMORY_H
 /*
 #define MALLOC MyMalloc
 #define FREE MyFree
 */
 #define MALLOC malloc
 #define FREE free

 /*To make sure memory allocations are freed MyMalloc and MyFree keep
 count of the number of allocations*/
 void MyFree( void *a);
 void *MyMalloc(size_t size);
 #endif
```

7.2 Function: Various Memory Functions

Documentation Date: 6/27/94

Prototypes:

```
void increasemem()
   void decreasemem()
   void MyFree(a)
   void *MyMalloc(size_t size)
```

Source File: *memory.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in memory.c: *<malloc.h> <stdio.h> "memory.h"*

Description:

These routines count the number of times a malloc and free is performed. They are used to make sure all allocated memory is freed. The file memory.h contains two definitions that cause the commands MALLOC and FREE to either point to the c library functions or these. The routines increasemem and decreasemem increment (decrement) a counter and print messages.

7.2.1 Program Listing:

```
#ifndef NeXT
    #include <malloc.h>
    #else
    #include <stdlib.h>
    #endif
#include <stdio.h>
#include "memory.h"

static int numberofaddresses=0;
   void increasemem( void);
   void decreasemem( void);

void increasemem()
  {
  numberofaddresses++;
  printf("There are %d memory allocations.\r",numberofaddresses);
  }

void decreasemem()
  {
  numberofaddresses--;
  printf("There are %d memory allocations.\r",numberofaddresses);
  }
```

```
void MyFree(a)
 void *a;
{
    free(a);
    decreasemem();
    }

void *MyMalloc(size_t size)
{
 void *a;
    a = malloc( size );
     if(a != NULL) increasemem();
     return(a);
    }
```

Chapter 8

misc.c, misc.h

Documentation Date: 3/10/95

8.1 Miscellaneous Functions

These are miscellaneous routines that do not fit elsewhere.
New Data Types:
None.
Definitions:

- min - a macro defined to return the minimum of two numbers.
- max - a macro defined to return the maximum of two numbers.
- M_PI - Pi
- FILENAME_STORAGE_TOO_SMALL = 10 - Error Return

8.1.1 Header File Listing:

```
/*****************************************
* MISC.H                    *
*                           *
* Miscellaneous procedures.             *
*****************************************/
#ifndef MISC_H
  #define MISC_H
  #ifndef min
   #define min(a,b) (( (a) < (b) ) ? (a) : (b))
   #endif
  #ifndef max
   #define max(a,b) (( (a) > (b) ) ? (a) : (b))
   #endif
  #ifndef M_PI
```

```
  #define M_PI          3.14159265358979323846
  #endif
/*returns +1 if x>0 -1 if x<0 returns 0 if x=0*/
 double sgn( double x);
/* returns x*x */
 double square ( double x);                                    20
/* rounds x up or down*/
 double round( double x);
/* returns a 4 quardrant inverse tangent in degrees*/
 double atan2d( double numer, double denom);
/*Takes a filename and an extension and constructs the fullname
fullname is dimensioned len
Returns SUCCESS or FILENAME_STORAGE_TOO_SMALL */
 #define FILENAME_STORAGE_TOO_SMALL 10
 int FormFullFileName( char *fullname, int len, char *name, char *ext);
 #endif                                                        30
```

8.2 Function: Miscellaneous Functions

Documentation Date: 6/27/94

Prototypes:

```
double sgn(double x);
   double square (double x);
   double round(double x);
   double atan2d(double numer, double denom);
   int FormFullFileName(char *fullname, int len, char *name, char
*ext);
```

Source File: *misc.c*
Type of Function: User Callable
Header Files Used in misc.c: *<math.h> "misc.h"*

Description:

1. sgn(double x) - returns the +- sign of x.
2. square - returns the square of x.
3. round(double x) - returns x rounded off to nearest integer.
4. atan2d(double numer, double denom) - returns the inverse tangent of numer/denom in degrees. It returns the proper quadrant.
5. FormFullFileName - Forms a full file name given the name string and extension string. Len is the length of the Fullname string storage space.

8.2.1 Program Listing:

```
/*@@@*************************************************************************
                    MISC.C
              Miscellaneous procedures.
*************************************************************************/
#include <math.h>
#include <string.h>
#include <stdio.h>
#include "datatype.h"
#include "misc.h"

                                                                    10
/*****************************************
*  round                    *
*****************************************/
double round( double number)
 {
```

```
  int i;
 i = ( int) ( number + sgn(number) * .5);
  return(( double) i);
 }

/*****************************************
* SGN                    *
*****************************************/
double sgn( double x)
  {
   if(x < 0.) return(-1.);
   if(x > 0.) return(1.);
   return(0.);
  }

/*****************************************
* SQUARE                 *
*****************************************/
double square ( double x)
  {
   return (x * x);
  }

double atan2d( double numer, double denom)
  {
   if( (numer == 0.) && (denom == 0.) ) return(0.);
   return(atan2(numer,denom)*180./M_PI);
  }

int FormFullFileName( char *fullname, int len, char *name, char *ext)
{
 int i,j;
 int lenofname, lenofext;
i = j = 0;
lenofname = strlen(name);
lenofext = strlen(ext);
 if((lenofname+lenofext+2)>len) return FILENAME_STORAGE_TOO_SMALL;
sprintf(fullname,"%s.%s",name,ext);
 return SUCCESS;
}
```

Chapter 9

plot.c, plot.h

Documentation Date: 3/12/95

9.1 Generate 2D plot

The routines in this file generate a 2D plot.
New Data Types:
typedef struct long minx, maxx, miny, maxy, majxtic, majytic, minxtic, minytic; int DefaultFontAndColor;/*Set true to take defaults*/ char FontFile[100], FontName[100]; int BackColor,BorderColor,TextColor,LabelColor; GraphData;
Definitions:

1. FONT_FILE - "tmsrb.fon" the name of the font for axis labels. I think the way this works is the font file gives the character shapes then the font name t'tms rmn'h12w6b gives the size. Check the MSC documentation, I believe there is an environment variable C searches for the font files. If you cannot find the proper environment variable to set then copy the font file you need into the main program's directory. If the program cannot find the font file you specify it will print the message "Couldn't register the font." If it cannot find the font inside the font file it prints "Couldn't set the font."

2. POINTS - 0 indicates that points are to be drawn.

3. LINES - 1 indicates that lines are to be drawn.

4. Colors for the plot:

   (a) PLOT_BLACK - 0

   (b) PLOT_DARK_BLUE - 1

   (c) PLOT_DARK_GREEN - 2

   (d) PLOT_TURQ - 3

(e) PLOT_DARK_RED - 4

(f) PLOT_DARK_PURPLE - 5

(g) PLOT_BROWN - 6

(h) PLOT_LIGHT_GRAY - 7

(i) PLOT_DARK_GRAY - 8

(j) PLOT_LT_BLUE - 9

(k) PLOT_LIGHT_GREEN - 10

(l) PLOT_CYAN - 11

(m) PLOT_LIGHT_RED - 12

(n) PLOT_LIGHT_PURPLE - 13

(o) PLOT_YELLOW - 14

(p) PLOT_WHITE - 15

5. Default values:

(a) DEFAULT_FONT "t'tms rmn'h12w6b"

(b) DEFAULT_BACK - PLOT_BLACK

(c) DEFAULT_BORDER - PLOT_LIGHT_GRAY

(d) DEFAULT_TEXT - PLOT_CYAN

(e) DEFAULT_LABEL - PLOT_YELLOW

9.1.1 Header File Listing:

```
#ifndef PLOTDATA_H
  #define PLOTDATA_H
 /*Colors*/
#define PLOT_BLACK 0
#define PLOT_DARK_BLUE 1
#define PLOT_DARK_GREEN 2
#define PLOT_TURQ 3
#define PLOT_DARK_RED 4
#define PLOT_DARK_PURPLE 5
#define PLOT_BROWN 6
#define PLOT_LIGHT_GRAY 7
#define PLOT_DARK_GRAY 8
#define PLOT_LT_BLUE 9
#define PLOT_LIGHT_GREEN 10
#define PLOT_CYAN 11
#define PLOT_LIGHT_RED 12
#define PLOT_LIGHT_PURPLE 13
#define PLOT_YELLOW 14
```

```
#define PLOT_WHITE 15

   typedef struct {
    long minx, maxx, miny, maxy, majxtic, majytic, minxtic, minytic;
    int DefaultFontAndColor;/*Set true to take defaults*/
    char FontFile[100], FontName[100];
    int BackColor,BorderColor,TextColor,LabelColor;
   }GraphData;

/*Default values*/
    #define DEFAULT_FONT "t'tms rmn'h12w6b"
   /*#define FONT_FILE "c:\\language\\quickc\\bin\\tmsrb.fon"*/
    #define FONT_FILE "tmsrb.fon"
    #define DEFAULT_BACK  PLOT_BLACK
    #define DEFAULT_BORDER  PLOT_LIGHT_GRAY
    #define DEFAULT_TEXT  PLOT_CYAN
    #define DEFAULT_LABEL  PLOT_YELLOW
 /*There are two types of plots POINTS and LINES*/
  #define POINTS 0
  #define LINES  1

  void SetupGraph( GraphData *graphdata );
  void QuitPlot ( void );
  void Plot( long xdata[], long ydata[], int num_pts, int plot_type, int color
);
  void PlotVerticalLine( long x, int color);
  void EraseVerticalLine( long x, long y, int ycolor);

  #endif
```

9.2 Function: Plot modified from a program Written by Roy Chancellor.

Documentation Date: 3/12/95

Prototypes:

```
void SetupGraph( GraphData *graphdata );
   void QuitPlot ( void );
   void Plot( long xdata[], long ydata[], int num_pts, int plot_type,
int color );
   void PlotVerticalLine(long x, int color);
   void EraseVerticalLine(long x, long y, int ycolor);
```

Source File: *plot.c*
Type of Function: User Callable
Header Files Used in plot.c: *<stdio.h> <graph.h> <conio.h> <process.h> "datatype.h" "misc.h" "plot.h"*

Description:

There are several user callable routines in this file. This has only been used on SVGA computers running DOS 6.0 and above. The callable routines follow:

1. EraseVerticalLine - This routine erases a vertical line. It is very slow and could be improved. The line is located at a given x value. The y value is an attempt to indicate where a curve's data point is located. The routine plots a single point at x,y in the specified color. This is an attempt to redraw the figure. It doesn't work well.

2. PlotVerticalLine - Plots a vertical line at x using specified color. The line extends from the plot minimum to plot maximum.

3. Plot - Plots num_pts points in specified color as either a line graph or point graph. The data is located in xdata and ydata.

4. QuitPlot - Resets the video mode into text mode.

5. SetupGraph - Uses the specified GraphData to establish the font, colors, and axes limits. The graphics video mode is set to _VRES16COLOR. See MSC documentation for an explanation of this definition.

9.2.1 Program Listing:

```
/*****************************************************************************
*   MODULE:        PLOT.C                    *
*                                            *
*   CREATED BY:    ROY CHANCELLOR            *
```

```
 *                                                                    *
 *   DATE CREATED:    10-27-93                                        *
 *                                                                    *
 *   DATE LAST MODIFIED:  10-27-93                                    *
 *            6-30-94 L. Everett - Added support for                  *
 *                multiple graphs                                     *
 *                                                                    *
 *   VERSION:         1.00                                            *
 *                                                                    *
 *   This program creates a graph on the screen and plots (x,y) points on *
 *   the screen.  The data points are passed to the plotting subroutine  *
 *   as integer arrays.  Tthe user must also pass the number of data     *
 *   points in the arrays and whether to draw lines between consecutive  *
 *   points or just plot the points individually.  A sample function call *
 *   would look like the following                                    *
 *                                                                    *
 *   plot_data( xdata, ydata, 1000, 1 );  (1 = lines, 0 = points)     *
 *                                                                    *
 *   This program must be linked with histogrm.obj and the main program  *
 *   which calls it.  For the two calls to setup_graph in this routine,  *
 *   the file names must be changed to whatever you are using.        *
 *                                                                    *
 **********************************************************************/

/* Header Files...                                                   */
#include <stdio.h>
#include <graph.h>
#include <conio.h>
#include <process.h>

#define TRUE 1

#include "plot.h"
#include "datatype.h"
#include "misc.h"

/*The following are local definitions and prototypes*/
struct axis_data
{
    double  lx;  /* low value of x axis data  */
    double  hx;  /* high value of x axis data */
    double  ly;  /* low value of y axis data  */
    double  hy;  /* high value of y axis data */
    int  ntx;    /* major x ticks */
    int  nty;    /* major y ticks */
    int  nmtx;   /* minor x ticks */
    int  nmty;   /* minor y ticks */
```

```
    int  back_color; /* background color */
    int  border_color;    /* border color */
    int  text_color;      /* text color */
    int  label_color;     /* axis label color */
};


  /* Function Prototypes...                                          */
   void get_graph_data( struct axis_data *a, GraphData *graphdata );
   void font_stuff( char *Font, char *FontFile);
   void set_mode_and_rectangle( struct axis_data );
   void draw_graph_axes( struct axis_data );
   void PlotErrorExit( char *message, int exitvalue);

 static struct axis_data xy;

/*****************************************************************************/

 void EraseVerticalLine( long x, long y, int ycolor)
 {
  long xx[1],yy[1];
 /*Erase the line*/
 PlotVerticalLine(x,xy.back_color);
 /*Draw tics*/
 draw_graph_axes( xy );
 xx[0]=x;
  if(x == ( long)xy.lx){
   PlotVerticalLine(x,xy.border_color);
   }
  else  if(x == ( long)xy.hx){
   PlotVerticalLine(x,xy.border_color);
   }
  else{
   yy[0]=0;
   /*Draw point on xaxis*/
   Plot(xx,yy,1,POINTS,xy.border_color);
   yy[0]=( long) xy.hy;
   /*Draw point on upper border*/
   Plot(xx,yy,1,POINTS,xy.border_color);
   }
 /*Draw plot point*/
 yy[0]=y;
 Plot(xx,yy,1,POINTS,ycolor);
 }

 void PlotVerticalLine( long xx, int color)
 {
```

```
 long x[2],y[2];
x[0] = x[1] = ( long) xx;
y[0] = ( long) xy.ly;                                                    100
y[1] = ( long) xy.hy;
Plot(x,y,2,LINES,color);
}

void Plot( long xdata[], long ydata[], int num_pts, int plot_type, int color
)
{
     int i;
     double tx, ty;
                                                                         110
    _setcolor( color );
     switch( plot_type )
    {
         case POINTS:
             for( i = 0; i < num_pts; ++i ){
                 tx = min( max(( double)xdata[i],xy.lx) ,xy.hx);
                 ty = min( max(( double)ydata[i],xy.ly) ,xy.hy);
                 _setpixel_w( tx, ty );
                 }
              break;                                                     120
        case LINES:
            tx = min( max(( double)xdata[0],xy.lx) ,xy.hx);
            ty = min( max(( double)ydata[0],xy.ly) ,xy.hy);
            _moveto_w( tx, ty );
             for( i = 1; i < num_pts; ++i )
            {
                tx = min( max(( double)xdata[i],xy.lx) ,xy.hx);
                ty = min( max(( double)ydata[i],xy.ly) ,xy.hy);
                _lineto_w( tx, ty );
            }                                                            130
             break;
    } /* end switch */
} /* end plot_data */

/*****************************************************************************/

 void QuitPlot ( void )
{
  _setvideomode(_DEFAULTMODE);
}                                                                        140

/*****************************************************************************/
 void font_stuff( char *Font, char *FontFile)
{
```

```
    if( _registerfonts( FontFile ) < 0 )PlotErrorExit("\n\n Couldn't Register The
Font.      ",10);
    if( _setfont( Font ) < 0 ) PlotErrorExit("\n\n Couldn't Set The Font.      ",11);
} /* end font_stuff */

                                                                          150
void PlotErrorExit( char *message, int exitvalue)
{
_setvideomode( _DEFAULTMODE );
gprintandwait(message);
exit( exitvalue );
}

/*************************************************************************
*   MODULE:          HISTOGRM.C                    *
*                                                  *
*   CREATED BY:      ROY CHANCELLOR                *
*                                                  *
*   DATE CREATED:    10-27-93                      *
*                                                  *
*   DATE LAST MODIFIED:  10-27-93                  *
*                                                  *
*   VERSION:         1.00                          *
*                                                  *
*   This program sets up the graphs for all programs.                *
*                                                  *
*   The axis limits, number of major tick marks, and number of minor       *
*   tick marks are stored in an ASCII file.  The filename including        *
*   drive, directory, and filename must be passed to this program.         *
*   For example, this subroutine may be called as                          *
*    setup_graph( "c:\\directory\\filename.ext" );                         *
*              ^this must be a \\ and not a \                              *
*   To use this routine in another program it must first be compiled and   *
*   then linked into the main program.  It will work with all memory       *
*   models.  Also, the FONT_FILE definition must be changed to the         *
*   actual location of the font file.                                      *
*                                                  *
*************************************************************************/

void SetupGraph( GraphData *graphdata )
{
    /* set up the fonted text outputs...                              */
        if(graphdata->DefaultFontAndColor){
            font_stuff(DEFAULT_FONT,FONT_FILE);
            xy.back_color     = DEFAULT_BACK;
            xy.border_color = DEFAULT_BORDER;
            xy.text_color     = DEFAULT_TEXT;
            xy.label_color    = DEFAULT_LABEL;
```

```
            }
        else {
            font_stuff(graphdata->FontName,graphdata->FontFile);
            xy.back_color   = graphdata->BackColor;
            xy.border_color = graphdata->BorderColor;
            xy.text_color   = graphdata->TextColor;
            xy.label_color  = graphdata->LabelColor;
            }
    /* get the axis limits from file...                                    */
        get_graph_data( &xy, graphdata );
    /* Setup the colors for background, labels, border, and ticks...       */
    /* set the videomode and draw the border...                            */
        set_mode_and_rectangle( xy );
    /* draw LINEAR-LINEAR graph axes...                                    */
        draw_graph_axes( xy );
    /* Set the color to the background color for other programs to read... */
        _setcolor( xy.back_color );
} /* end setup_graph */

/*****************************************************************************/

void get_graph_data( struct axis_data *xy, GraphData *graphdata )
{
  xy->lx = graphdata->minx;
  xy->hx = graphdata->maxx;
  xy->ly = graphdata->miny;
  xy->hy = graphdata->maxy;
  xy->ntx= ( int) graphdata->majxtic;
  xy->nty= ( int) graphdata->majytic;
  xy->nmtx= ( int) graphdata->minxtic;
  xy->nmty= ( int) graphdata->minytic;
} /* end get_graph_data */

/*****************************************************************************/

void draw_graph_axes( struct axis_data xy )
{
    char    num_lab[10];
    int  i;
    double  deltax, deltay, xrange, yrange, xpos, ypos, xmpos, ympos;
    struct  xycoord view;
/*  double    xlo, xhi, ylo, yhi, low_tick_y, low_tick_x;
    int   i, nmtx, nmty, num_times_x, num_times_y;
    struct    xycoord view, phys;*/

    deltax = ( xy.hx - xy.lx ) / xy.ntx;
    xrange = xy.hx - xy.lx;
```

```
    deltay = ( xy.hy - xy.ly ) / xy.nty;
    yrange = xy.hy - xy.ly;

/************** Make The X AXIS Ticks, Minor ticks, and Number Labels ******/

    /* MAJOR TICKS HERE...                                    */
     for( i = 0; i <= xy.ntx; ++i )
     {
          _setcolor( xy.border_color );
          xpos = xy.lx + i * deltax;
          _moveto_w( xpos, xy.ly );
          _lineto_w( xpos, xy.ly + .025 * yrange );
          _moveto_w( xpos, xy.hy );
          _lineto_w( xpos, xy.hy - .025 * yrange );

          _setcolor( xy.label_color );
          sprintf( num_lab, "%.01f", xpos );
          view = _getviewcoord_w( xpos, xy.ly );
          _moveto( view.xcoord - 6/*16*/, view.ycoord + 9 );
          _outgtext( num_lab );
     } /* end for */

    _setcolor( xy.border_color );
    /* MINOR TICKS HERE...                                    */
     for( i = 0; i <= xy.ntx * xy.nmtx; ++i )
     {
          xmpos = xy.lx + i * deltax / xy.nmtx;
          _moveto_w( xmpos, xy.ly );
          _lineto_w( xmpos, xy.ly + .01 * yrange );
          _moveto_w( xmpos, xy.hy );
          _lineto_w( xmpos, xy.hy - .01 * yrange );
     } /* end for */

/************** Make The Y AXIS Ticks, Minor ticks, and Number Labels******/

    /* MAJOR TICKS HERE...                                    */
     for( i = 0; i <= xy.nty; ++i )
     {
          ypos = xy.ly + i * deltay;
          _setcolor( xy.border_color );
          _moveto_w( xy.lx, ypos );
          _lineto_w( xy.lx + .02 * xrange, ypos );
          _moveto_w( xy.hx, ypos );
          _lineto_w( xy.hx - .02 * xrange, ypos );
          sprintf( num_lab, "%.01f", ypos );
          _setcolor( xy.label_color );
          view = _getviewcoord_w( xy.lx, ypos );
```

```
        _moveto( view.xcoord - 60/*49*/, view.ycoord - 5 );
        _outgtext( num_lab );
    } /* end for */
                                                                    290
    _setcolor( xy.border_color );
    /*MINOR TICKS HERE...                          */
     for( i = 0; i <= xy.nty * xy.nmty; ++i )
    {
        ympos = xy.ly + i * deltay / xy.nmty;
        _moveto_w( xy.lx, ympos );
        _lineto_w( xy.lx + .01 * xrange, ympos );
        _moveto_w( xy.hx, ympos );
        _lineto_w( xy.hx - .01 * xrange, ympos );
    } /* end for */                                                 300
} /* end draw_graph_axes */

/*****************************************************************************/

 void set_mode_and_rectangle(  struct axis_data xy )
{
     double xrange, yrange, xlo, xhi, ylo, yhi;

    _setvideomode( _VRES16COLOR );
                                                                    310
    xrange = xy.hx - xy.lx;
    yrange = xy.hy - xy.ly;
    xlo    = xy.lx - 0.2 * xrange;
    ylo    = xy.ly - 0.3 * yrange;
    xhi    = xy.hx + 0.2 * xrange;
    yhi    = xy.hy + 0.2 * yrange;

    /* create a window for plotting real data...                 */
    _setwindow( TRUE, xlo, ylo, xhi, yhi );
    _setcolor( xy.border_color );                                   320
    /* draw a rectangle for the graph border...              */
    _rectangle_w( _GBORDER, xy.lx, xy.ly, xy.hx, xy.hy );
} /* end set_mode_and_rectangle */
```

Chapter 10

pose.c, pose.h

Documentation Date: 3/10/95

10.1 Determining Pose

The routines in this file convert blob data into a pose. They also convert from one coordinate system into another.
New Data Types:
None.
Definitions:

- Missing - an internal operator to test if a retro is missing.
- CAMERAASPECT = 1.266

10.1.1 Header File Listing:

```
#ifndef POSE_H
  #define POSE_H
  /* The following computes a pose for the target */
  /* The pose and rawpose are defined via Redfield's convention */
  void GetPose(Target *target, Transform *trans, BlobIds *ids, RawPose *raw,
Pose *pose);
  /* The following may not work when yaw and pitch are out of bound */
  void TransformPose(Pose *from, Pose *to, Transform *transform);
  #endif
```

10.2 Function: GetPose

Documentation Date: 3/12/95

Prototypes:

```
void GetPose(Target *target, Transform *trans, BlobIds *ids, RawPose
*raw, Pose *pose, Parameters param)
```

Source File: *pose.c*
Type of Function: User Callable
Header Files Used in pose.c: *<math.h> <string.h> "datatype.h" "misc.h" "pose.h"*

Description:

This routine computes the Pose and Rawpose. Pose is the x, y, range, pitch, yaw and roll of the target relative to the camera measured in inches and degrees. I think the x is range positive away from the camera. The +y is to the horizontal right on the monitor, the +z is vertical down. The rawpose is pose measured in pixel values.[1] The raw x, y (column, row), roll, range coordinates are based on the right and left retro location. If either right or left is missing, the top and bottom are used. Since only one retro can be missing (if more are gone this routine should not be called) either the right/left pair or the top/bottom pair will be present. The 0,0 is at the row and column specified in the parameters data structure (VideoCenterRow, VideoCenterCol). Of course parameter's row and column are measured from the lower left monitor corner. If the LED is missing, the routine returns pitch and yaw of 180 degrees and the raw pitch and yaw is set to BAD_PITCH_YAW.

10.3 Function: TransformPose

Documentation Date: 6/27/94

Prototypes:

```
void TransformPose(Pose *from, Pose *to, Transform *transform)
   double OneRow(Pose *pose, int t[6]);
```

Source File: *pose.c*
Type of Function: TransformPose is user callable, OneRow is internal.
Header Files Used in pose.c: *<math.h> <string.h> "datatype.h" "misc.h" "pose.h"*

[1] Actually range is in an odd unit, and roll is in degrees.

Description:

TransformPose performs a coordinate transformation. Actually this is a dumb method for doing this. This should be converted to use 4 by 4 homogenious matrices. The transformation "matrix" is a 6 by 6 matrix which premultiplies pose to get a new pose. Since it has never really been used, I don't trust it.

OneRow is simply multiplies a 1 by 6 vector by a 6 by 1 pose.

10.3.1 Program Listing:

```
/*Make sure camera parameters are correct
     r1 = 1270097.;
     r2 = 1.;
     h = 2.361E-4;
     v = -1.858E-4;
     p = 1.927E-4;
     y = 2.423E-4;
     */
#include <math.h>
#include <string.h>
#include "datatype.h"
#include "misc.h"
#include "pose.h"

#define Missing <0
//STS-62 target width/height (11.75"/74.75"), Charlotte (4.2"/3")
#define CAMERAASPECT 1.266

   /*This routine computes the Pose and Rawpose (basically the
Pose in pixels) given the target (blob locations).*/
void GetPose(Target *target, Transform *trans, BlobIds *ids, RawPose *raw, Pose
*pose, Parameters param)
  {
   double dx,dy,yaw,pitch,range,horiz,vert;
   double TargetAspect;
  TargetAspect=trans->TargetWidth/trans->TargetHeight;
  /*The raw x, y (column, row) coordinates are based on the right and
left retro location, if they are missing use the top and bottom. Since
only one retro can be missing, right & left or top and bottom will be
there. mx is the x or column position, my is the y or row position. 0,0
is center of the monitor. dx, dy are used in range.*/
   if( (ids->left Missing) || (ids->right Missing) ){
     raw->mx = (target->topretro.centx+target->bottomretro.centx)/2 - ( double)param.Vid
     raw->my = (target->topretro.centy+target->bottomretro.centy)/2 - ( double)param.Vid
     dx = target->topretro.centx - target->bottomretro.centx;
     dy = (target->bottomretro.centy - target->topretro.centy)/trans->CameraAspect;
     raw->d = TargetAspect* sqrt(dx*dx + dy*dy);
```

```
      raw->roll = atan2d(dy,dx);
      }
    else{
      raw->mx = (target->leftretro.centx+target->rightretro.centx)/2 - ( double)param.VideoCe
      raw->my = (target->leftretro.centy+target->rightretro.centy)/2 - ( double)param.VideoCe
      dx = target->leftretro.centx - target->rightretro.centx;
      dy = (target->leftretro.centy - target->rightretro.centy)/trans->CameraAspect;
      raw->d = sqrt(dx*dx + dy*dy);
      raw->roll = atan2d(dx,dy);
      }
  /*Now convert the raw information into real units using the camera
lens parameters*/
  range = trans->TargetWidth*trans->FocalLength/raw->d;
  //FocalLength in pixels
  horiz = raw->mx/trans->FocalLength*range;
  vert = raw->my/trans->CameraAspect/trans->FocalLength*range;
   if(target->num_of_led != 1){
     pitch = 180.;
     yaw = 180.;
     raw->ledx = BAD_PITCH_YAW;
     raw->ledy = BAD_PITCH_YAW;
     }
   else {
    raw->ledx = target->led.centx - ( double) param.VideoCenterCol;
    raw->ledy = target->led.centy - ( double) param.VideoCenterRow;
    pitch = atan2d(raw->ledy/trans->CameraAspect/trans->FocalLength, 1.0);
    yaw = atan2d(raw->ledx/trans->FocalLength, 1.0);
    }
 range = trans->TargetWidth*trans->FocalLength/raw->d+trans->RangeOffset;
 raw->roll = raw->roll + 90.;
 pose->y = horiz;
 pose->z = -vert;
 pose->x = range;
 pose->yaw = yaw;
 pose->pitch = pitch;
 pose->roll = raw->roll;
 }

/*The following routine performs a coordinate transformation. Actually
this is a dumb method for doing this. This should be converted to use 4
by 4 homogenious matrices. The transformation "matrix" is a 6 by 6
matrix which is premultiplied by pose to get a new pose. Since it has
never really been used, I don't trust it.*/
/*
double OneRow(Pose *pose, int t[6]);
void TransformPose(Pose *from, Pose *to, Transform *transform)
  {
```

```
  to->x = OneRow(from,transform->tx);
  to->y = OneRow(from,transform->ty);
  to->z = OneRow(from,transform->tz);
  if(from->yaw == BAD_PITCH_YAW)to->yaw = BAD_PITCH_YAW;
    else to->yaw = OneRow(from,transform->tyaw);
  if(from->pitch == BAD_PITCH_YAW)to->pitch = BAD_PITCH_YAW;
    else to->pitch = OneRow(from,transform->tpitch);
  to->roll = OneRow(from,transform->troll);
  }

double OneRow(Pose *pose, int t[6])
  {
  return (double) t[0]*pose->x + (double) t[1]*pose->y + (double) t[2]*pose->z
+ (double) t[3]*pose->yaw + (double) t[4]*pose->pitch + (double) t[5]*pose->roll;
  }
*/
```

Chapter 11

qtarga8.c, qtarga8.h

Documentation Date: 3/9/95

11.1 Hardware Control of TARGA - Interrupt Drive

The routines in this file are low level TARGA specific functions. These routines were written by Mr. Qin (QIN@mentor.cc.purdue.edu). The actual qtarga8.c file contains several pieces of code that are the same as found in targa8.c. This documentation includes only the pieces of code that are DIFFERENT from that in targa8.c.
New Data Types:
See Targa8 Documentation.
Definitions: See Targa8 Documentation.

11.2 Function: Global Targa Interrupt Routines

Documentation Date: 3/9/95

Prototypes:

```
void GrabEightFrames_ForSync (void);
   void ResyncFieldCount(int FirstBrightField);
   void GrabOnOffFrame();
   int Live_Video();
   void liveoverlay(int page);
```

Source File: *qtarga8.c*
Type of Function: User Callable
Header Files Used in qtarga8.c: *<conio.h> "stdinc.h" "ifc.h"*

Description:

There are several user callable routines in this file. The targa board is capable of generating an interrupt on vertical retrace. These routines catch the interrupt and keep count of which image is currently coming in. When the leds are blinked via the camera's sync pulse, this allows us to tell whether the leds are on or off.

1. GrabEightFrames_ForSync - This routine must be called to acquire the initial sync. The program grabs 8 images and sets the frame counter properly. Once you determine which image is the on you reset the counter again.

2. ResyncFieldCount - Once you know which of the 8 frames from GrabEightFrames_ForSync is the first bright field, you call this routine to set the frame counter properly.

3. GrabOnOffFrame - This routine will grab three frames (LED on, off then on again) and store them in buffers 1, 2, and 3 respectively.

4. Live_Video - This is a modification of the Live_Video in targa8.c. There are some problems where routines clear the interrupt enable bit. This routine works properly with interrupts whereas the one in targa8.c doesn't.

5. liveoverlay - This is a repair of the routine in targa8.c that supports interrupts.

11.2.1 Program Listing:

```
/*This was developed by Q We do not use it. For reference purposes only.*/
/*Only the differences in targa8.c are shown*/
#include <conio.h>
#include "stdinc.h"
#include "ifc.h"

/*****************************************
*  Grab_Eight_Frame                      *
*****************************************/
void GrabEightFrames_ForSync ( void)
{                                                                    10
  /* Turn off interrupt */
 _inp (0x222);
 _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
 _outp (0x220+0x401, 0x90);
 _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
 _outp (0x220+0xc02, inp(0x220+0xc02) & 0x3f);
 _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
 _inp(0x222);
```

```
    set_page (0, DISPLAY_LIVE);
    FieldGrab(&currRegs,0);
    GrabNextField();

    set_page (1, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (2, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (3, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (4, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (5, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (6, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (7, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    /* Turn on interrupt */
   _inp (0x222);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _outp (0x220+0x401, 0x90);
   _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
   _outp (0x220+0xc02, inp(0x220+0xc02) | 0xc0);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _inp(0x222);

   reset_field_count();
}

void ResyncFieldCount( int FirstBrightField)
{
```

```
    while (field_count_is() != FirstBrightField);

  reset_field_count();
}

void GrabOnOffFrame()
{
   unsigned char rList[] = {DISPMODE, 0};

   while (field_count_is() != 15);

  currRegs.dispMode = 1;        /* Put Targa board in Live mode          */
  WriteSet(&currRegs, rList);       /* It need to be in Live mode to capture */

  /* Turn off interrupt */
  _inp (0x222);
  _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
  _outp (0x220+0x401, 0x90);
  _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
  _outp (0x220+0xc02, inp(0x220+0xc02) & 0x3f);
  _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
  _inp(0x222);

  set_page (1, DISPLAY_LIVE);  /* First LED On frame is in memory 1 */
  FieldGrab(&currRegs,0);
  GrabNextField();

  set_page (2, DISPLAY_LIVE);  /* LED Off frame is in memory 2 */
  GrabNextField();
  GrabNextField();

  GrabNextField();
  GrabNextField();

  set_page (3, DISPLAY_LIVE);  /* Second LED On frame is in memory 3 */
  GrabNextField();
  GrabNextField();

 GrabNextField();
 GrabNextField();

 set_page (4, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 GrabNextField();
 GrabNextField();
```

```
    GrabNextField();
    GrabNextField();

    /* Turn on interrupt */
   _inp (0x222);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _outp (0x220+0x401, 0x90);
   _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
   _outp (0x220+0xc02, inp(0x220+0xc02) | 0xc0);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _inp(0x222);

}
/*****************************************
* Live_Video                    *
* Initialize Vision system              *
*****************************************/
int Live_Video()
{
    unsigned char rList[] = {DISPMODE, 0};

   currRegs.dispMode = 1;
   WriteSet(&currRegs, rList);
    return SUCCESS;
}

/*Turn on live overlay*/
void liveoverlay( int page)
{
  unsigned char rList[] = {DISPMODE, 0};

 Show_Process_Image(page);
 currRegs.dispMode = 2;
 WriteSet(&currRegs, rList);

}
```

Chapter 12

roi.c, roi.h

Documentation Date: 3/9/95

12.1 Defining Rectangular Region Of Interest

The routines in this file draw a rectangle on the cpu monitor and compute statistics of the image contained inside the rectangle. It will also highlight the pixels with intensity between two specified limits.
New Data Types:
ALineOfPixels is defined, see the listing.
Definitions:

- DIVIDEBY = 2 - Amount to divide by if images are subtracted.
- ADD = 129 - Amount to add on after dividing.

12.1.1 Header File Listing:

```
#ifndef ROI_H
  #define ROI_H
  void ShowPixelsInHistogram(ROI roi, int Bright, int Dim, int SecondBright,
int StoragePage, int WorkingPage); void DefineROI(ROI *roi, int page, int *increment);
  void DefineROIandCompute( int page);
  #endif
```

12.2 Function: Show Pixels In Histogram

Documentation Date: 3/9/95

Prototypes:

```
void ShowPixelsInHistogram(ROI roi, int Bright, int Dim, int SecondBright,
int StoragePage, int WorkingPage); void DefineROI(ROI *roi, int page,
int *increment);
```

Source File: *roi.c*
Type of Function: User Callable
Header Files Used in roi.c: *<float.h> <stdio.h> <graph.h> <string.h> <conio.h> <time.h> <math.h> <ctype.h> "datatype.h" "roi.h" "TARGA8.h" "TARGET.h" "PLOT.h" "MISC.h" "DIP.h"*

Description:

This routine draws a histogram of an image then prompts you for a lower and upper limit. It draws colored lines on the histogram denoting these limits then on the image display it sets all pixels between the limits to 255 (white) and those outside the limit to 0 (black). It will tell you how many pixels lie between the limits. If the lower and upper limits are equal, it tells you how many pixels have the intensity level equal to the limit. It has the ability to use an ROI defined for the image but this has not been tested. If Bright, Dim and SecondBright are greater than -1, a double subtract is performed. If Bright and Dim are greater than -1, a single subtract is used. If only Bright is greater than -1, no subtraction is used. The image buffers StoragePage and WorkingPage are temporary image storage. The image in StoragePage and WorkingPage will be wiped out.

12.3 Function: Define ROI and Compute

Documentation Date: 3/9/95

Prototypes:

```
void DefineROIandCompute(int page);
```

Source File: *roi.c*
Type of Function: User Callable
Header Files Used in roi.c: *<float.h> <stdio.h> <graph.h> <string.h> <conio.h> <time.h> <math.h> <ctype.h> "datatype.h" "roi.h" "TARGA8.h" "TARGET.h" "PLOT.h" "MISC.h" "DIP.h"*

Description:

This routine allows the user to define a rectangle on the image screen then it returns some statistics on the pixels inside the rectangle.

12.3.1 Program Listing:

```
#include < float.h>
#include <stdio.h>
#include <graph.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include <ctype.h>
#include "datatype.h"
#include "roi.h"
#include "TARGA8.h"
#include "TARGET.h"
#include "PLOT.h"
#include "MISC.h"
#include "DIP.h"

#if IMAGEHEIGHT > IMAGEWIDTH
   typedef struct{
    int start, end;
   Pixel pixel[IMAGEHEIGHT];
   }ALineOfPixels;
#else
   typedef struct{
    int start, end;
   Pixel pixel[IMAGEWIDTH];
    }ALineOfPixels;
#endif
ALineOfPixels theline;

void ContrastRectangle( int x, int y, int width, int height, int page);
void DisplayRectangleStats(ROI *roi, int increment);
void MoveRectangle( int fx, int fy, int fwidth, int fheight, int tx, int ty,
int twidth, int theight, int page);
void RestoreRectangle( int x, int y, int width, int height, int page);
void RestoreHorizonalLine(ALineOfPixels line, int y, int page);
void ContrastHorizontalLine(ALineOfPixels *line, int x, int y, int width, int
page);
void MoveHorizontalLineFromTo( int fx, int fy, int tx, int ty, int width, int
page, ALineOfPixels line);
void ContrastVerticalLine(ALineOfPixels *line, int x, int y, int height, int
```

```
page);
 void RestoreVerticalLine(ALineOfPixels line, int x, int page);
 void MoveVertialLineFromTo( int fx, int fy, int tx, int ty, int height, int page,
ALineOfPixels line);
 void CopyRoi(ROI *from, ROI *to);


 #define DIVIDEBY 2
 #define ADD 129

                                                                          50
 void ShowPixelsInHistogram(ROI roi, int Bright, int Dim, int SecondBright, int
StoragePage, int WorkingPage)
  {
  GraphData graphdata;
  FullHistogram x, y;
  Pixel low, high, oldlow, oldhigh;
   int i,j;
   long number;
   char message[128];
  ImageLine line;                                                         60
   char ans;

   for(i=-PIXEL_MAX;i<=PIXEL_MAX;i++)x[i+PIXEL_MAX]=i;
 graphdata.minx = 0;
 graphdata.maxx = PIXEL_MAX;
 graphdata.miny = YAXISMIN;
 graphdata.maxy = ( long) YAXISMAX;
 graphdata.majxtic = 8;
 graphdata.majytic = 5;
 graphdata.minxtic = 4;                                                  70
 graphdata.minytic = 5;
 graphdata.DefaultFontAndColor = TRUE;

 roi.resolution = 1;
 FullSubAndHistogram(Bright, Dim, SecondBright, y, roi);
 Subtract_Images(Bright,Dim,SecondBright,StoragePage, DIVIDEBY, ADD, roi);
 Show_Process_Image(WorkingPage);
 CopyFrame(StoragePage,WorkingPage);

 SetupGraph(&graphdata);                                                 80
 Plot(&x[PIXEL_MAX], &y[PIXEL_MAX], FULLNUMBEROFPIXELBINS/2+1,LINES,PLOT_DARI
 oldlow=0;
 oldhigh=0;
  do{
   number = 0;
   gpromptandread("Enter low range-> ","%d",&low);
   EraseVerticalLine(( long) oldlow,y[oldlow+PIXEL_MAX],PLOT_DARK_RED);
```

```
        PlotVerticalLine(( long)low,PLOT_YELLOW);
        oldlow=low;
        low = max(low, (Pixel) 0);
        low = min(low, (Pixel) 255);
        gpromptandread("Enter high range-> ","%d",&high);
        EraseVerticalLine(( long) oldhigh,y[oldhigh+PIXEL_MAX],PLOT_DARK_RED);
        PlotVerticalLine(( long) high, PLOT_DARK_BLUE);
        oldhigh=high;
        high = max(high, (Pixel) 0);
        high = min(high, (Pixel) 255);
        for(i=0;i<IMAGEHEIGHT;i++){
         GetLine(i,StoragePage,line);
          for(j=0;j<IMAGEWIDTH;j++)
          if((line[j]>=low)&&(line[j]<=high)){
           line[j]=PIXEL_MAX;
           number++;
           }
         else line[j] = 0;
        PutLine(i,WorkingPage,line);
        }
      sprintf(message,"Number of pixels-> %ld, (S)top, (D)isplay original or (R)edraw?",nun
      gStatus_Message(message);
      ans = ( char) getch();
       if((ans=='r')||(ans=='R')){
        QuitPlot();
        SetupGraph(&graphdata);
        Plot(&x[PIXEL_MAX], &y[PIXEL_MAX], FULLNUMBEROFPIXELBINS/2+1,LINES,PLOT_I
        PlotVerticalLine(( long)oldlow,PLOT_YELLOW);
        PlotVerticalLine(( long) oldhigh, PLOT_DARK_BLUE);
        }
      else if((ans=='d')||(ans=='D')) CopyFrame(StoragePage,WorkingPage);
     } while((ans!='s') && (ans!='S'));
     QuitPlot();
}

static ROI roisave = {IMAGEWIDTH/2,IMAGEHEIGHT/2,IMAGEWIDTH/2+10,IMAGEHEIGI
    int xs, ys, xe, ye, resolution;

void DefineROIandCompute( int page)
{
 int ans;
ROI roi;
DPixel Brightest, Dimmest, Average, StdDev;
 int increment;
 long NumberOfPixels;
roi.xs = roisave.xs;
roi.xe = roisave.xe;
```

```
    roi.ys = roisave.ys;
    roi.ye = roisave.ye;
    roi.resolution = roisave.resolution;
    Show_Process_Image(page);
    increment = 10;
     do{
       DefineROI(&roi,page,&increment);                                    140
       RestoreRectangle(roi.xs,roi.ys,roi.xe-roi.xs,roi.ye-roi.ys,page);
       ComputeStatistics(page, -1, -1, roi, &Brightest, &Dimmest, &Average, &StdDev,
   &NumberOfPixels);
       ContrastRectangle(roi.xs,roi.ys,roi.xe-roi.xs,roi.ye-roi.ys,page);
     _clearscreen(0);
      _settextposition(10,1);
      printf("      Brightest   Dimmest   Average   StdDev  # of Pixels\n");
      printf("     ----------- --------- --------- -------- ------------\n");
      printf("%10d %10d %10d %10d %10ld\n",( int) Brightest,Dimmest,Average,StdDev,NumberOfPixel
      Status_Message("(S)top ROI statistics or any key for more->");
      ans = getch();
      RestoreRectangle(roi.xs,roi.ys,roi.xe-roi.xs,roi.ye-roi.ys,page);
      } while(ans != 's');
   roisave.xs = roi.xs;
   roisave.xe = roi.xe;
   roisave.ys = roi.ys;
   roisave.ye = roi.ye;
   roisave.resolution = roi.resolution;
   }
                                                                           160
   void CopyRoi(ROI *from, ROI *to){
   to->xs = from->xs;
   to->ys = from->ys;
   to->xe = from->xe;
   to->ye = from->ye;
   to->resolution = from->resolution;
   }

   void DefineROI(ROI *roi,  int page,  int *oldincrement)
   {                                                                       170
    int ach,ch;
    int increment;
   ROI oldroi;
   increment = *oldincrement;
   _clearscreen(0);
   _settextposition(6,1);
   _outtext("   Your options are:  \n\n");
   _outtext("   Move ROI:          Use the Arrow Keys\n");
   _outtext("   Size ROI:          (W)ider, (N)arrower, (T)aller, (S)horter\n");
   _outtext("                or    Home,    End,        PageUp,   PageDown\n");
```

```
_outtext("   Modify Increment:  Increase(+) or Decrease(-)\n");
_outtext("   Examine data:          (G)et statistics of ROI\n\n\n");
_outtext("   Rectangle position (x,y) = (Horz.,Vert.)  from lower left:");
Status_Message("Enter desired option->");
CopyRoi(roi,&oldroi);
DisplayRectangleStats(roi, increment);
 do{
  ContrastRectangle(roi->xs, roi->ys, roi->xe - roi->xs, roi->ye - roi->ys,
page);
  ch = getch();
  ch = toupper(ch);
  if((ch != 'G')){
    if(ch == 0)ach = getch();
    else ach = ch;
    switch (ach){
     case '+':      increment++;
                break;
     case '-':      increment--;
                break;
     case 71:
     case 'W':
     case 'w':      roi->xs=roi->xs-increment;
                roi->xe=roi->xe+increment;
                break;
     case 79:
     case 'N':
     case 'n':      roi->xs=roi->xs+increment;
                roi->xe=roi->xe-increment;
                break;
     case 73:
     case 'T':
     case 't':      roi->ys=roi->ys-increment;
                roi->ye=roi->ye+increment;
                break;
     case 81:
     case 'S':
     case 's':      roi->ys=roi->ys+increment;
                roi->ye=roi->ye-increment;
                break;
    case 72:
                roi->ys=roi->ys+increment;
                roi->ye=roi->ye+increment;
                break;
    case 80:
                roi->ys=roi->ys-increment;
                roi->ye=roi->ye-increment;
                break;
```

```
        case 77:
                roi->xs=roi->xs+increment;                                    230
                roi->xc=roi->xc+increment;
                 break;
        case 75:
                roi->xs=roi->xs-increment;
                roi->xc=roi->xc-increment;
                 break;
      }
    }
   increment = max(increment,1);
   roi->xs = max(0,roi->xs);                                                  240
   roi->xc = min(IMAGEWIDTH,roi->xc);
   roi->xc = max(roi->xc,roi->xs+1);
   roi->ys = max(0,roi->ys);
   roi->yc = min(IMAGEHEIGHT,roi->yc);
   roi->yc = max(roi->yc,roi->ys+1);
   DisplayRectangleStats(roi, increment);
   RestoreRectangle(oldroi.xs, oldroi.ys, oldroi.xc - oldroi.xs, oldroi.yc
- oldroi.ys, page);
   CopyRoi(roi,&oldroi);
   } while(ch != 'G');                                                        250
ContrastRectangle(roi->xs, roi->ys, roi->xc - roi->xs, roi->yc - roi->ys, page);
*oldincrement = increment;
}

/*Local routines below here*/

static ALineOfPixels left, right, top, bottom;

void ContrastRectangle( int x, int y, int width, int height, int page)
{                                                                             260
ContrastHorizontalLine(&bottom, x, y, width, page);
ContrastHorizontalLine(&top, x, y+height, width, page);
ContrastVerticalLine(&left, x, y+1, height-2, page);
ContrastVerticalLine(&right, x+width, y+1, height-2, page);
}

void MoveRectangle( int fx, int fy, int fwidth, int fheight, int tx, int ty,
int twidth, int theight, int page)
{
RestoreRectangle(fx,fy,fwidth,fheight,page);                                  270
ContrastRectangle(tx,ty,twidth,theight,page);
}

void RestoreRectangle( int x, int y, int width, int height, int page)
{
```

```
 RestoreVerticalLine(left, x, page);
 RestoreVerticalLine(right, x+width, page);
 RestoreHorizonalLine(top, y+height, page);
 RestoreHorizonalLine(bottom, y, page);
 }

 void RestoreHorizonalLine(ALineOfPixels line, int y, int page)
 {
  int i;
  if( (y>-1) && (y<IMAGEHEIGHT) ) for(i=line.start;i<line.end;i++)SetAPixel(i,y,page,line.pi
 }

 void ContrastHorizontalLine(ALineOfPixels *line, int x, int y, int width, int
page)
 {
  int i;
 line->start = max(0,x);
 line->end = min(IMAGEWIDTH,x+width);
  if( (y>-1) && (y<IMAGEHEIGHT) ) for(i=line->start;i<line->end;i++){
     line->pixel[i] = ContrastPixel(i,y,page);
     }
 }

 void MoveHorizontalLineFromTo( int fx, int fy, int tx, int ty, int width, int
page, ALineOfPixels line)
 {
 RestoreHorizonalLine(line, fy, page);
 ContrastHorizontalLine(&line, tx, ty, width, page);
 }

 void ContrastVerticalLine(ALineOfPixels *line, int x, int y, int height, int
page)
 {
  int i;
 line->start = max(0,y);
 line->end = min(IMAGEHEIGHT,y+height);
  if( (x>-1) && (x<IMAGEWIDTH) ) for(i=line->start;i<line->end;i++) line->pixel[i]
= ContrastPixel(x,i,page);
 }

 void RestoreVerticalLine(ALineOfPixels line, int x, int page)
 {
  int i;
  if( (x>-1) && (x<IMAGEWIDTH) ) for(i=line.start;i<line.end;i++)SetAPixel(x,i,page,line.pix
 }

 void MoveVertialLineFromTo( int fx, int fy, int tx, int ty, int height, int page,
```

```
ALineOfPixels line)
{
RestoreVerticalLine(line,fx,page);
ContrastVerticalLine(&line, tx, ty, height, page);
}
```

Chapter 13

targa8.c, targa8.h

Documentation Date: 3/12/95

13.1 Hardware Control of TARGA

The routines in this file are low level TARGA specific functions. The idea is to place everything that accesses a TARGA register or TARGA written code in this file.

New Data Types:

None.

Definitions: Global Definitions (all error returns):

- TARGAFONT = systemb.tsf , Font used in the live overlay. The DOS environment variable TFONTS sets the directory to search for the font file. If the TFONTS environment is not set, it searches the directory of the main program. To set the TFONTS use set TFONTS="blah/blah" in the autoexec.bat file.
- WRONG_MODE = -2
- BAD_SETUPBLOCK = -3
- BAD_READBUFF = -4
- BAD_WRITEBUFF = -5

Local Definitions:

- DISPLAY_MEMORY = 0 Indicates you want the targa to show the contents of memory rather than live or overlay.
- DISPLAY_LIVE = 1 Indicates you want live video.
- vPan1 = ((short) 0) Vertical pan value.

- vPan0 = ((short) 256 + vPan1) There are two pages per byte address. Each one is given by a different vpan.
- PixelDepth = 1 Eight bit pixels.
- HorRes = IMAGEWIDTH The number of pixels in a row.
- NumPages = 8 Maximum number of images in storage.
- AbsRowNum(ImagRow,Page) = ((ImagRow) + ((Page)*IMAGEHEIGHT)) Helps compute the byte offset of a pixel we want.
- RowBytes = ((PixelDepth)*(HorRes)) The number of bytes in an image row.
- RowPerBank = ((32768)/RowBytes) The number of image lines in a bank.
- LineSize = (PixelDepth*HorRes) Same as RowBytes. I don't know why there are two of them.
- DELTAY = (-30) The space between lines of overlay.
- XSTART = (370) The starting horizontal position of overlay pose text.
- YSTART = (450) The starting vertical position of overlay pose text.
- S_IWRITE = 0200 This is defined in MSC somewhere, it indicates the type of binary file to open. I could not find the .h file where it was defined so I defined it myself.
- Led control. The following set the number of Led which turns on when using plug and play and printer port control. If you want LED one and two on then you want LEDone && LEDtwo. To be honest, we don't do it this way but we ought to.
  1. LEDnone = 0xFF /*1111,1111*/, no LEDs on.
  2. LEDone = 0xFE /*1111,1110*/, LED one on.
  3. LEDtwo = 0xFD /*1111,1101*/, LED two on.
  4. LEDthree = 0xFB /*1111,1011*/
  5. LEDfour = 0xF7 /*1111,0111*/
  6. LEDfive = 0xEF /*1110,1111*/
  7. LEDsix = 0xDF /*1101,1111*/
  8. LEDseven = 0xBF /*1011,1111*/
  9. LEDeight = 0x7F /*0111,1111*/
  10. LEDall = 0x00 /*0000,0000*/
- AND = &, Makes logical statements readable.

13.1.1 Header File Listing:

```
#ifndef TARGA8_H
  #define TARGA8_H
  #ifndef DATATYPE_H
    #include "datatype.h"
    #endif
  /***********
  Local Definitions
  ***********/
  #define WRONG_MODE -2
  #define BAD_SETUPBLOCK -3                                          10
  #define BAD_READBUFF -4
  #define BAD_WRITEBUFF -5

  #define LEDnone 0xFF     /*1111,1111*/
  #define LEDone 0xFE      /*1111,1110*/
  #define LEDtwo 0xFD      /*1111,1101*/
  #define LEDthree 0xFB    /*1111,1011*/
  #define LEDfour 0xF7     /*1111,0111*/
  #define LEDfive 0xEF     /*1110,1111*/
  #define LEDsix 0xDF      /*1101,1111*/                             20
  #define LEDseven 0xBF    /*1011,1111*/
  #define LEDeight 0x7F    /*0111,1111*/
  #define LEDall 0x00      /*0000,0000*/
  #define AND &
  int SetLEDState( int OnOff);
  int SetLEDStateUsingPrinter( int OnOff);

  int Live_Video( void);
  int GetLine( int imagerow, int page, ImageLine line);
  int PutLine( int imagerow, int page, ImageLine line);              30
  int SetAPixel( int xcol, int yrow, int page, Pixel value);
 Pixel ContrastPixel( int xcol, int yrow, int page);
  int Initialize_vision( int color_to_capture);
  int End_vision( void);
  int Show_Process_Image( int Buffer);
  void CopyFrame( int From, int To);
  int Grab_Frame ( int Buffer, int LED_state);
  int Grab_Frame_Printer_Port ( int Buffer, int LED_state);
  int ClearAllPages( void);
  int ClearPage( int page);                                          40
  void Grab_Two_Frame ( int BufferOne, int BufferTwo);
  void Grab_Four_Frame ( int BufferOne, int BufferTwo, int BufferThree, int
BufferFour);
  void Charlotte_Grab( int BufferOne, int BufferTwo, int BufferThree, int
BufferFour);
  void Integrate_Image ( int Frame, double *sum, ROI roi);
```

```
    void GrabEightFrames ( void);
    void GrabEightSlowly ( int numfieldstowait);/*field is 1/60 of a second*/
    void GrabSixFrames ( void);
    void circle( int page, int xcen, int ycen, int radius, Pixel color);
    void line( int page, int xs, int ys, int xe, int ye, Pixel color);
/*Displaymode can be 1 for Live display else Memory Display*/
    int set_page( int page_number, int displaymode);


    #define TARGAFONT "systemb.tsf"
/*Initializes the text for the overlay (loads a font) returns 0 is
success, returns 1 if cannot find font*/
    int inittext( char *font);
/*Frees memory used by font*/
    void finishtext( void);
/*Writes the pose on the overlay and updates the overlay string memory*/
    void DrawOverlay(Pose *screenpose, int overlaypage);
/*Writes the overlay string memory on the overlaypage in the color
White is 255 Black is 0 */
    void DrawOverStrings(Pixel color, int overlaypage);
/*actually erases the entire page*/
    void EraseOverlay( int overlaypage);
/*Turns live overlay on and selects page for display*/
    void liveoverlay( int page);
/*Prints text at xs, ys on page with color*/
    void writetext( int page, int xs, int ys, char *text, Pixel color);
    void setprinterportaddress( int address);
    #endif
```

13.2 Function: Global Targa Routines

Documentation Date: 3/12/95

Prototypes:

```
int Live_Video(void);
   int GetLine(int imagerow, int page, ImageLine line);
   int PutLine(int imagerow, int page, ImageLine line);
   int Initialize_vision(int color_to_capture);
   int End_vision(void);
   int Show_Process_Image(int Buffer);
   void CopyFrame(int From, int To);
   int Grab_Frame (int Buffer, int LED_state);
   int ClearAllPages(void);
   int ClearPage(int page);
   void Grab_Two_Frame (int BufferOne, int BufferTwo);
   void Grab_Four_Frame (int BufferOne, int BufferTwo, int BufferThree,
int BufferFour);
   void GrabEightFrames (void);
   void GrabEightSlowly (int numfieldstowait);/*field is 1/60 of a
second*/
   void GrabSixFrames (void);
   void circle(int page, int xcen, int ycen, int radius, Pixel color);
   void line(int page, int xs, int ys, int xe, int ye, Pixel color);
   int set_page(int page_number, int displaymode);
   int inittext(void);
   void finishtext(void);
   void DrawOverlay(Pose *screenpose, int overlaypage);
   void DrawOverStrings(Pixel color, int overlaypage);
   void EraseOverlay(int overlaypage);
   void liveoverlay(int page);
   int SetLEDState(int OnOff);
   int SetLEDStateUsingPrinter(int OnOff);
   int SetAPixel(int xcol, int yrow, int page, Pixel value);
   Pixel ContrastPixel(int xcol, int yrow, int page);
   int Grab_Frame_Printer_Port (int Buffer, int LED_state);
   void Charlotte_Grab(int BufferOne, int BufferTwo, int BufferThree,
int BufferFour);
   void Integrate_Image (int Frame, double *sum, ROI roi);
   void writetext(int page, int xs, int ys, char *text, Pixel color);
   void setprinterportaddress(int address);
```

Source File: *targa8.c*
Type of Function: User Callable

Header Files Used in targa8.c: *<targraf.h> <io.h> <stdlib.h> <string.h> <fcntl.h> <conio.h> <time.h> "targa8.h" "datatype.h"*

Description:

There are several user callable routines in this file. There may be some differences between this code and the flight code. I do not know why this happened nor whether the differences are necessary but the differences were present at one point in time. Where I knew of differences, I documented them. Search for the word flight in the code to find them. Some of the documentation for these programs can be found in the hardware and software manuals produced by Targa. Page numbers in this documentation refer to these manuals. In version 5 of MSC (used on some machines at NASA) there was a problem with _inp and _outp. To get around this we use an #ifdef TAMU statement to change to inp and outp. You may need to modify this to make it work for you. If the code compiles, don't worry. If you have a problem with inp and outp it will not compile. The tap setting doesn't quite work as we expect. The settings used prevent the captured images from shifting side to side from the live video. Granted it looks weird. Basically what we do is set the tap to different values when the image is captured than when it is displayed.

1. Initialize_vision - This routine must be called before any other targa routine. In addition to initialization the routine sets the pen size for overlays. Part of initialization is to run the program LIVE.BAT which is generated via another Targa utility.

   We have noticed problems with this routine.

   (a) It is used with End_vision to trap all targa stuff between the two calls. If you put an initialize_vision ... end_vision in a loop, the routine will crash big time after several iterations. I think there is a memory leak in some of the targa code which we have no source for. The solution is to call initialize_vision once at the beginning of your code and call end_vision once at the end.

   (b) When the code is used with a "real" program there isn't enough memory to do a system call to initialize some of the hardware. Our solution was to write a do nothing but initialize program which generates a TARGA.PAR file. Then this routine looks to see if TARGA.PAR is in the current directory, if it is, we read it rather than doing a system call.

   We experimentally determined the setting for the pan register. If the value is 0 the image jumps to the side when you capture an image. I do not understand why the number we use works.

   Originally we used the simple method shown on 1-31 of the Targa hardware manual to capture green only. Now we use recapture mode (1-31 hardware) to capture any color we want.

2. End_vision - Terminate Vision system.
3. Grab_Frame - This routine sets the LED state and grabs a frame.
4. Grab_Two_Frame - Obvious?
5. Grab_Four_Frame - Ditto.
6. Grab_Eight_Slowly This routine grabs a frame (2 fields) then waits numtopause vertical blanking periods 1/60 of a second I think. Grabs another etc.
7. GrabEightFrames - Grabs 8 quickly.
8. GrabSixFrames - Grabs 6 quickly.
9. Live_Video - Display a live image, turns off overlay.
10. Show_Process_Image - Display a stored image, turns off overlay.
11. ClearPage - Erases a video page.
12. ClearAllPages - Erases 8 pages.
13. CopyFrame - Copies video from page From into page To.
14. GetLine - Gets row imagerow from page. It actually copies data from one memory address to another. SLOW.
15. PutLine - Copies data into row imagerow of page. SLOW.
16. circle - Generates a circle. It draws with a specified color in Targa memory. You may not see it unless you display the page it is drawn on.
17. line - Draws a line.
18. inittext - Prepares for live overlay but does not show anything. It must be called after Initialize_vision but before any overlay stuff. It searches for the TFONTS environment variable. If TFONTS is set, it loads the font file from the specified directory, otherwise it loads the font from the current directory. Fonts take a lot of memory so call finishtext when you are finished with the fonts. If the program cannot load the font for whatever reason, bad name, bad directory or insufficient memory, the function returns FAIL.
19. liveoverlay - Overlay page on top of live. To turn it off call Live_Video or reset the page.
20. DrawOverlay - Draw screenpose on the overlay page. Load the internal strings overx, overy ... then do it.

21. DrawOverStrings - This routine writes the text in internal storage overx, overy ... onto Targa page overlaypage using color. It probably should not be a user callable routine since it is only called by DrawOverlay.

22. EraseOverlay - Erases the overlay. The simplest logic to erase the overlay is to set it zero. SLOW!

23. finishtext - The fonts used in the overlay (set up by inittext) takes up memory. Dump them when finished.

24. Grab_Frame_Printer_Port - Sets the LEDS using the printer port then grabs an image.

25. Charlotte_Grab - In the Charlotte system, the LEDs are blinking a a slower rate and are not synchronized to anything. This routine grabs a sufficient number of frames that we have a high probability of catching an on and an off LED state.

26. reset_field_count - See the chapter on QTARGA8.

27. GrabEightFrames_ForSync - See the chapter on QTARGA8.

28. ResyncFieldCount - See the chapter on QTARGA8.

29. GrabOnOffFrame - See the chapter on QTARGA8.

30. ContrastPixel - Sets the contrast of a pixel. If the original pixel is less than 128 it sets it to 255, if it is greater than 128 it sets it zero.

31. SetAPixel - Sets one pixel intensity.

32. SetLEDState - For plug and play, a DIDO board with sufficient current drive capability is used. The board has a base address of 0X300 and LEDs are connected to pins of the board. A byte is sent to the routine to designate how many LEDs to turn on. The definitions in the header file should be used to specify how many LEDs are to be turned on. A TTL low turns the LED on.

33. setprinterportaddress - Sets the address of the printer port. We find this number on our computer by noting the value displayed by the BIOS on boot up.

34. SetLEDStateUsingPrinter - Similar to SetLEDState except it uses the printer port. I think the drive capability of the printer port is insufficient so you will have to use a transistor drive circuit. TTL high turns the LEDs on. The software inverts the number sent to it so the software is compatible between Plug and Play and printer port.

13.3 Function: Local Targa Routines

Documentation Date: 6/30/94

Prototypes:

```
void dowrite(void);
   int rightmode(void);
   int capture(void);
   int DumpState(char *file, void *reg);
   int ReadState(char *file, void *reg);
   void writetext(int page, int xs, int ys, char *text, Pixel color);
```

Source File: *targa8.c*
Type of Function: Internal to the Library, Not User Callable
Header Files Used in targa8.c: *<targraf.h> <io.h> <stdlib.h> <string.h> <fcntl.h> "targa8.h" "datatype.h"*

Description:

Some of the documentation for these programs can be found in the hardware and software manuals produced by Targa. Page numbers in this documentation refer to these manuals.

1. dowrite - TARGA is controlled via several registers, this routine copies the internal data into the hardware registers.

2. rightmode - Returns true is the TARGA is in the correct mode of operation. 512 rows, 512 columns and 8 bits.

3. capture - Grabs a single frame (2 fields).

4. DumpState - Does a binary write to a file (targa.par is currently used) storing the contents of the Targa registers. This is used in conjuction with Initialize_Vision to avoid the system call.

5. ReadState - Does a binary read from a file (targa.par is currently used) determining the previous contents of the Targa registers. This is used in conjuction with Initialize_Vision to avoid the system call.

6. writetext - Writes overlay text into Targa memory.

13.3.1 Program Listing:

*/*NOTE NOTE NOTE NOTE – There may be some differences between this code and the flight code. I do not know why this happened and if it is necessary but it was true at one point. Where I knew of differences, I*

```
documented them. Search for the word flight to find them.*/

/*In this file, comments to the right of a statement refer to page
numbers is hardware or software. These are books written by Targa.
They came with the board and software.*/
/*During Charlotte Grab the tap works properly. I did not change other grab
routines because I do not have time to test them.*/
#include <targraf.h>
#include <io.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <conio.h>
#include <time.h>
#include "datatype.h"
#include "targa8.h"

/************************************
    Local Static Variables
************************************/
TplusRegs    currRegs;              /* TARGA+ structure    */
TplusRegPtr  cRegPtr;    /* current structure pointer */
grafPort     thePort; /*used by targa routines to store pen size etc.*/
int BankBase;

/************************************
    Local Definitions
************************************/
/*Version 5 does not allow _inp and _outp but they use it at NASA change
these defines depending on the version of c you use*/
#ifdef TAMU
    #define _inp inp
    #define _outp outp
    #endif

#define DISPLAY_MEMORY 0
#define DISPLAY_LIVE 1
#define LowBankAddress 0xc00000
#define HighBankAddress 0xc00000
/*
#define LowBankAddress ( (long) BankBase << 16 )
#define HighBankAddress ( ((long) BankBase << 16 ) + 32768)
*/
/* check out these bank addresses especially Highbank 1-13 hardware*/
/* The following defs are from 1-28 of the hardware manual*/
/*
#define vPan0 ((short) 13)
```

```
 #define vPan1 ((short) 256 + vPan0)
 */
  #define vPan1 (( short) 0)
  #define vPan0 (( short) 256 + vPan1)
 /* PixelDepth is defined as the number of bytes per pixel
    HorRes = 512 for image widths <= 512 or 1024 for widths > 512
    1-12 of hardware*/
       #define PixelDepth 1
       #define HorRes IMAGEWIDTH
       #define NumPages 8
       #define AbsRowNum(ImagRow,Page) ((ImagRow) + ((Page)*IMAGEHEIGHT))
       #define RowBytes ((PixelDepth)*(HorRes))
       #define RowPerBank ((32768)/RowBytes)
       #define LineSize (PixelDepth*HorRes)
 /*typedef ImageLine Bank[RowPerBank];*/

 /************************************
     Local Function and Prototypes
 ************************************/
  int rightmode( void);
  int capture( void);
  int DumpState( char *file,  void *reg);
  int ReadState( char *file,  void *reg);

 /*This returns true is the TARGA is in the correct mode of operation.
 512 rows, 512 columns and 8 bits.*/
  int rightmode( void)
 {
      if((currRegs.width  > 512) |          /* 16 bit lores        */
        (currRegs.height > 512) |
        (currRegs.depth != 1  ))
     {
         puts("Use TMODE 1, 2, 5, 6, 9, 10\n"); return(FALSE);
     }
      return(TRUE);
     }

/*This routine grabs a single frame.*/
 int capture( void)
{
     FrameGrab(&currRegs);          /* frame grab          */
      return SUCCESS;
}

/*This routine sets a page (0 to 7) once the page is set, a capture
routes the data into the page. displaymode can be either 1 for live or 0
for stored. */
```

```
unsigned char PList[5] = {MODE2,PAGE,WHICH8,BYCAP,VPAN};
int set_page( int page_number, int displaymode)
                                                                    100
{
/* We will define video page numbers as: 1 - lowest byte lowest vPan,
        0 - lowest byte highest vPan
        3 - next byte lowest vPan etc */
     if(!rightmode()) return(WRONG_MODE);
     if(displaymode == DISPLAY_LIVE)
        currRegs.dispMode = 1;
         else
        currRegs.dispMode = 0;
     switch( page_number ){
         case 0:                                                    110
        currRegs.page = 0;
            currRegs.vPan    = ( short int) vPan0;        /* vPan page 0      */
            currRegs.byCap = 1; /*5-34 hardware*/
            currRegs.which8 = 0;       /*display lowest 8 bits VRAM */
             break;
         case 1:
            currRegs.vPan    = ( short int) vPan1;
            currRegs.byCap = 1; /*5-34 hardware*/
            currRegs.which8 = 0;       /*display lowest 8 bits VRAM */
             break;                                                 120
         case 2:
            currRegs.vPan    = ( short int) vPan0;        /* vPan page 0      */
            currRegs.byCap = 2; /*5-34 hardware*/
            currRegs.which8 = 1;
             break;
         case 3:
            currRegs.vPan    = ( short int) vPan1;
            currRegs.byCap = 2; /*5-34 hardware*/
            currRegs.which8 = 1;
             break;                                                 130
         case 4:
            currRegs.vPan    = ( short int) vPan0;        /* vPan page 0      */
            currRegs.byCap = 4; /*5-34 hardware*/
            currRegs.which8 = 2;
             break;
         case 5:
            currRegs.vPan    = ( short int) vPan1;
            currRegs.byCap = 4; /*5-34 hardware*/
            currRegs.which8 = 2;
             break;                                                 140
         case 6:
            currRegs.vPan    = ( short int) vPan0;        /* vPan page 0      */
            currRegs.byCap = 8; /*5-34 hardware*/
            currRegs.which8 = 3;
```

```
            break;
        case 7:
            currRegs.vPan    = ( short  int) vPan1;
            currRegs.byCap = 8; /*5-34 hardware*/
            currRegs.which8 = 3;
            break;
        default:
            puts("Page must be 0-7\n");
            return FAIL;
        }
    WriteSet(&currRegs,PList);
    return SUCCESS;
    }

/*****************************************
    Externally callable routines
*****************************************/

/*****************************************
* Initialize Vision system              *
*****************************************/
/*This routine must be called before any other targa routine. We have
noticed problems with it. 1. It is used with End_vision to trap all
targa stuff between the two calls. If you put an initialize_vision ...
end_vision is a loop, the routine will crash big time after several
iterations. I think there is a memory leak in some of the targa code
which we have no source for. The solution is to call initialize_vision
once at the beginning of your code and call end_vision once at the end.
2. When the code is used with a "real" program there isn't enough memory
to do a system call to initialize some of the hardware. Our solution
was to write a do nothing but initialize program which generates a
TARGA.PAR file. Then this routine looks to see if TARGA.PAR is in the
current directory, if it is, we read it rather than doing a system
call.

In addition to initialization the routine sets the pen size for
overlays.

*/
int Initialize_vision( int color_to_capture)
{
    if ( InitGraphics() < 0 )/*2-65 of software*/
    {
        puts("TARGA driver not available");
        exit(-1);
    }
    /*
```

```
    setprinterportaddress(0x378);
    */
    if(!GraphInit(&currRegs))
     /* Puts board in raw mode, resets it and gets registers
        returns true is successful false otherwise (3-18) */
    {
        puts("TARGA+ driver not available\n"); return FAIL;
    }
    /*If there is a targa.par file, read it otherwise try to do the
system call.*/
    if( ReadState("targa.par", &currRegs) != SUCCESS ){
     /*This runs a file called live.bat in the current directory.*/
     system("LIVE");
     /*Create a new targa.par file for next time.*/
     DumpState("targa.par",&currRegs);
     }
    SetPenSize(3,3);
    SetTPlusMode(&currRegs,1,0);/* set pixel depth 1, interlaced = 0*/
    /* Set camera to rgb 1-26 hardware*/
        currRegs.rgb = 0; /*1 is rgb, 0 is composite*/
        currRegs.sVideo = 0; /*0 for rgb*/
    /* Set initial vertical panning */
        currRegs.top[0] = 255;
        currRegs.top[1] = 1023;
        currRegs.bot[0] = 0;
        currRegs.bot[1] = 768;
        currRegs.vPan = vPan0; /*What the heck 1-28 hardware*/
    /* Set horizontal pan 1-29 hard. If this value is 0 the image
jumps to the side if you 1. are watching live, 2. grab an image and 3.
display the image. What we did was experimentally determine the number
to use so the jump is minimum. I do not understand why 12 works. (Nor
do I really care.)*/
/* The following line changed*/
        currRegs.tap = 17;
        currRegs.tap = 5;
   /*Originally we used the simple method shown on 1-31 hardware to
capture green only. We do not use this stuff, but I left it here just
in case we needed this information. It is so hard to get otherwise. If
you set it to b&w here it will only capture the green input 1-30,31
hard. However perhaps we could capture a 32 bit image then move the
color we want to another page.      */
        /*currRegs.monoSrc = 0;*/ /*0 takes green, 1 takes chromakeyer 1-31 hard*/
        /*currRegs.cm2 = 0; *//*0 applys monosrc to all 3 bytes of input, 1 does not
take monosrc*/
        /*currRegs.cm1 = 0; *//*This means 16 bit color 5-57 hardware, when cm2 =
0 this doesn't matter*/
        /*currRegs.cm3 = 0; *//*0 means do not recapture, 1 means capture from the
```

```
blenders 5-60 hard*/
  /* Now we use recapture mode 1-31 hardware to capture any color we want*/
        currRegs.cm3 = 1; /*This is the key. Make sure we are not overlaying or else
we will capture the overlay also*/
        switch (color_to_capture){ /*See 1-31 hardware for the following*/
            case GREEN:
                currRegs.cm1 = 1;
                currRegs.cm2 = 0;
                 break;
            case BLUE:
                currRegs.cm1 = 0;
                currRegs.cm2 = 0;
                 break;
            default: /*RED*/
                currRegs.cm1 = 1;
                currRegs.cm2 = 1;
                }
    /* Set the live image to be 8 bit green only*/
        currRegs.live8 = 0;      /*1 means map green input to all 24 bits (pg5-61)
                0 means map 3 colors to 24 bits a 0 means the live video will be color*/
    /* make sure vram rather than border colors is sent to the buffer port mux*/
        currRegs.bufferPortColor = 0; /* 0 uses vram, 1 uses border color 5-56 hardware*/
    /* make sure vram display is set to 8 bits */
        currRegs.bufferPortSrc = 0; /* 0 sets 8 bit mode 5-56 hardware*/
    /* make sure board is set for bank addressing*/
        currRegs.map = 0;
    currRegs.fgp = 0; /*0 means no processing to the data (1-39,5-61)
            1 means remove excessive blue*/
    currRegs.genlock = 1; /*1 means sync is generated by targa used for
                camera only I think, not needed in memory display*/
    currRegs.livePortInv = 0; /*0 means don't invert port mux (1-39)
                1 means do invert*/
    currRegs.liveMixBypass = 1;     /*1 means bypass any mixing (1-39,5-57)
                    0 means use the mixer */
    currRegs.livePortSrc = 0; /*0 means portmux accepts live video (1-39,5-61)
                1 means accepts stored data*/
    currRegs.dispMode = 1;/*1 means live
                0 means stored */
    currRegs.maskLH = 0;/*I don't think this is needed. It protects some memory
planes */
    currRegs.genCtrl = 5;/*I don't know why but this differed between A&M and the
original flight hardware make it 0 for consistency*/
    /*Originally this was 0 on flight changed to 5 for hitachi camera*/
    WriteAll(&currRegs);
     switch (currRegs.base){/*5-36 hard*/
        case 0: BankBase = 0x8000;
                break;
```

```
        case 1: BankBase = 0x9000;
                break;
        case 2: BankBase = 0xA000;
                break;
        case 3: BankBase = 0xB000;
                break;
        case 4: BankBase = 0xC000;
                break;
        case 5: BankBase = 0xD000;
                break;
        case 6: BankBase = 0xE000;
                break;
        case 7: BankBase = 0xF000;
                break;
       }
    return SUCCESS;
}

/*****************************************
* End_vision                             *
* Terminate Vision system                *
*****************************************/
int End_vision( void)
{
    EndGraphics();
    if(GraphEnd (&currRegs)) return(SUCCESS); else return(FAIL);
          /*Writes register and closes driver (3-17)
            returns true if success, false otherwise*/
    }

/*****************************************
* Grab_Frame                             *
* This routine sets the LED state and    *
* grabs a frame. It is hardware dependent.*
*****************************************/
int Grab_Frame ( int Buffer, int LED_state)
   {
    int returnvalue;
    unsigned char rList[] = {TAP,0};
    currRegs.tap = 0;
    WriteSet(&currRegs,rList);
    SetLEDState(LED_state);
    if(set_page (Buffer, DISPLAY_LIVE) != SUCCESS) return FAIL;
    returnvalue = capture();
    SetLEDState(LEDnone);

    currRegs.tap = 5;
```

```
    WriteSet(&currRegs,rList);

    }

  int Grab_Frame_Printer_Port ( int Buffer,  int LED_state)
    {
     int returnvalue;
     unsigned  char rList[] = {TAP,0};
    currRegs.tap = 0;
    WriteSet(&currRegs,rList);
    SetLEDStateUsingPrinter(LED_state);
     if(set_page (Buffer, DISPLAY_LIVE) != SUCCESS) return FAIL;
    returnvalue = capture();
    SetLEDStateUsingPrinter(LEDnone);

    currRegs.tap = 5;
    WriteSet(&currRegs,rList);

    }

  int Grab_Frame_Nowait ( int Buffer,  int LED_state)
    {
     unsigned  char rList[] = {TAP,0};
    currRegs.tap = 0;
    WriteSet(&currRegs,rList);
     if(set_page (Buffer, DISPLAY_LIVE) != SUCCESS) return FAIL;
    SetLEDState(LED_state);
    FieldGrab(&currRegs,0);
    GrabNextField();
    SetLEDState(LEDnone);
    GrabNextField();
    GrabNextField();

    currRegs.tap = 5;
    WriteSet(&currRegs,rList);

    }

/*****************************************
*  Grab_Two_Frame                        *
*****************************************/
 void Grab_Two_Frame ( int BufferOne,  int BufferTwo)
    {
     unsigned  char rList[] = {TAP,0};
    currRegs.tap = 0;
    WriteSet(&currRegs,rList);
    /*if start by grabbing odd they are all half on
```

```
   if start by grabbing even they are all all way on or all way off
   if you pause inbetween grabbing even/odd some will be all on some all off
   by convention frame grab begins on an odd field
   */
   set_page (BufferOne, DISPLAY_LIVE);
   /*
   There may be some confusion here. The flight hardware probably needs
the following FieldGrab but our hardware was modified so we need
FieldGrag(&currRegs,1);
   */
   FieldGrab(&currRegs,0);
   GrabNextField();

   set_page (BufferTwo, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();


   currRegs.tap = 5;
   WriteSet(&currRegs,rList);

   }

/*****************************************
*  Grab_Four_Frame                      *
*****************************************/
 void Grab_Four_Frame ( int BufferOne,  int BufferTwo,  int BufferThree,  int BufferFour)
   {
    unsigned  char rList[] = {TAP,0};
   currRegs.tap = 0;
   WriteSet(&currRegs,rList);
   set_page (BufferOne, DISPLAY_LIVE);
   FieldGrab(&currRegs,0);
   /*
   There may be some confusion here. The flight hardware probably needs
the preceeding FieldGrab but our hardware was modified
   FieldGrab(&currRegs,1);
   */
   GrabNextField();

   set_page (BufferTwo, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();

   set_page (BufferThree, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();
```

```
    set_page (BufferFour, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    currRegs.tap = 5;
    WriteSet(&currRegs,rList);

    }

/*****************************************
*  Charlotte_Grab*
*****************************************/
void Charlotte_Grab( int BufferOne,  int BufferTwo,  int BufferThree,  int BufferFour)
   {

    unsigned  char rList[] = {TAP,0};
   currRegs.tap = 0;
   WriteSet(&currRegs,rList);

   set_page (BufferOne, DISPLAY_LIVE);
   FieldGrab(&currRegs,0);
   GrabNextField();

   set_page (BufferTwo, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();
   GrabNextField();
   GrabNextField();

   set_page (BufferThree, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();
   GrabNextField();
   GrabNextField();

   set_page (BufferFour, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();
   GrabNextField();
   GrabNextField();

   currRegs.tap = 5;
   WriteSet(&currRegs,rList);

   }
```

```
/*****************************************
*  Grab_Eight_Slowly                    *
* This routine grabs a frame (2 fields) then
* waits numtopause vertical blanking periods
* 1/60 of a second I think. Grabs another...
*****************************************/
void GrabEightSlowly ( int numtopause)
  {
   int i;
   unsigned  char rList[] = {TAP,0};
  currRegs.tap = 0;
  WriteSet(&currRegs,rList);
  set_page (0, DISPLAY_LIVE);
  FieldGrab(&currRegs,0);
  GrabNextField();

  set_page (1, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();
   for(i=numtopause;i>0;i--) VBLWait();

  set_page (2, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();

  set_page (3, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();
   for(i=numtopause;i>0;i--)VBLWait();

  set_page (4, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();

  set_page (5, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();
   for(i=numtopause;i>0;i--)VBLWait();

  set_page (6, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();

  set_page (7, DISPLAY_LIVE);
  GrabNextField();
  GrabNextField();
```

```
    currRegs.tap = 5;
    WriteSet(&currRegs,rList);

    }

/*****************************************
*  Grab_Eight_Frame                     *
*****************************************/
void GrabEightFrames ( void)
    {
     unsigned  char rList[] = {TAP,0};
    currRegs.tap = 0;
    WriteSet(&currRegs,rList);
    set_page (0, DISPLAY_LIVE);
    FieldGrab(&currRegs,0);
    GrabNextField();

    set_page (1, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (2, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (3, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (4, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (5, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (6, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    set_page (7, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();
```

```
   currRegs.tap = 5;
   WriteSet(&currRegs,rList);

   }

/*****************************************
*  Grab_Six_Frames              *
*****************************************/
 void GrabSixFrames ( void)
   {
    unsigned  char rList[] = {TAP,0};
   currRegs.tap = 0;
   WriteSet(&currRegs,rList);
   set_page (0, DISPLAY_LIVE);
   FieldGrab(&currRegs,0);
   GrabNextField();

   set_page (1, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();

   set_page (2, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();

   set_page (3, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();

   set_page (4, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();

   set_page (5, DISPLAY_LIVE);
   GrabNextField();
   GrabNextField();


   currRegs.tap = 5;
   WriteSet(&currRegs,rList);

   }

 void reset_field_count( void);
/*****************************************
*  Grab_Eight_Frame_ForSync          *
*****************************************/
```

```
void GrabEightFrames_ForSync ( void)
{
  /* Turn off interrupt */
 _inp (0x222);
 _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
 _outp (0x220+0x401, 0x90);
 _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
 _outp (0x220+0xc02, inp(0x220+0xc02) & 0x3f);
 _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
 _inp(0x222);

 set_page (0, DISPLAY_LIVE);
 FieldGrab(&currRegs,0);
 GrabNextField();

 set_page (1, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (2, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (3, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (4, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (5, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (6, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 set_page (7, DISPLAY_LIVE);
 GrabNextField();
 GrabNextField();

 /* Turn on interrupt */
_inp (0x222);
_outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
_outp (0x220+0x401, 0x90);
```

```
  _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
  _outp (0x220+0xc02, inp(0x220+0xc02) | 0xc0);
  _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
  _inp(0x222);

  reset_field_count();
}

int field_count_is( void);                                          670

void ResyncFieldCount( int FirstBrightField)
{
   while (field_count_is() != FirstBrightField);

  reset_field_count();
}

void GrabOnOffFrame()
{                                                                   680
   unsigned char rList[] = {DISPMODE, 0};

   while (field_count_is() != 15);

  currRegs.dispMode = 1;        /* Put Targa board in Live mode          */
  WriteSet(&currRegs, rList);       /* It need to be in Live mode to capture */

  /* Turn off interrupt */
  _inp (0x222);
  _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);                     690
  _outp (0x220+0x401, 0x90);
  _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
  _outp (0x220+0xc02, inp(0x220+0xc02) & 0x3f);
  _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
  _inp(0x222);

   set_page (1, DISPLAY_LIVE);  /* First LED On frame is in memory 1 */
   FieldGrab(&currRegs,0);
   GrabNextField();
                                                                    700
   set_page (2, DISPLAY_LIVE);  /* LED Off frame is in memory 2 */
   GrabNextField();
   GrabNextField();

   GrabNextField();
   GrabNextField();

   set_page (3, DISPLAY_LIVE);  /* Second LED On frame is in memory 3 */
```

```
    GrabNextField();
    GrabNextField();

    GrabNextField();
    GrabNextField();

    set_page (4, DISPLAY_LIVE);
    GrabNextField();
    GrabNextField();

    GrabNextField();
    GrabNextField();

    GrabNextField();
    GrabNextField();

    /* Turn on interrupt */
   _inp (0x222);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _outp (0x220+0x401, 0x90);
   _outp (0x220+0xc00, inp(0x220+0x402) | 0x40);
   _outp (0x220+0xc02, inp(0x220+0xc02) | 0xc0);
   _outp (0x220+0xc00, inp(0x220+0x402) & 0xbf);
   _inp(0x222);

}
/*****************************************
* Live_Video                             *
* Initialize Vision system               *
*****************************************/
 int Live_Video()
{
    unsigned char rList[] = {DISPMODE, 0};

   currRegs.dispMode = 1;/*1 means live
                0 means stored */
   WriteSet(&currRegs, rList);
    return SUCCESS;
}

/*****************************************
* Show_Process_Image                     *
* This was modified for the Targa to
  accept the image number 0-7
*****************************************/
int Show_Process_Image( int Buffer)
  {
```

```
    set_page(Buffer,DISPLAY_MEMORY);
     return SUCCESS;
    }

/****************************************
The following sets a page to 0
****************************************/
 int ClearPage( int page)
    {
    ImageLine line;
     int row;
     for(row=0;row<IMAGEWIDTH;row++)line[row]=0;
     for(row=0;row<IMAGEHEIGHT-1;row++) PutLine(row, page, line);
     return SUCCESS;
    }

/****************************************
The following sets a page bits to all 1s
****************************************/
 int Ones ( int page)
    {
    ImageLine line;
     int row;
     for(row=0;row<IMAGEWIDTH;row++)line[row]=255;
     for(row=0;row<IMAGEHEIGHT-1;row++) PutLine(row, page, line);
     return SUCCESS;
    }

/****************************************
The following sets all memory to 0
****************************************/
 int ClearAllPages( void)
    {
     int page;
     for(page=0;page<NumPages;page++)ClearPage(page);
     return SUCCESS;
    }

/****************************************
The following copies video from page From into page To
****************************************/
 void CopyFrame( int From,  int To)
   {
   ImageLine line;
    int i;
    for(i=0;i<IMAGEHEIGHT;i++){
     GetLine(i,From,line);
```

```
        PutLinc(i,To,linc);
        }
      }

/*****************************************
The following gets row imagerow from page
0 <= imagerow <= IMAGEHEIGHT
*****************************************/
int GetLinc( int imagerow,  int page, ImageLinc linc)
    {
     int row;
     unsigned  int count;
    count =  sizeof(Pixel) * IMAGEWIDTH;
    row = AbsRowNum(imagerow,page);
     if(!SetUpBlock(0,row,IMAGEWIDTH-1,row+1)) return(BAD_SETUPBLOCK);
     if(!ReadBuff(linc,count)) return(BAD_READBUFF);
     return SUCCESS;
    }

/*****************************************
The following puts row imagerow into page
0 <= imagerow <= IMAGEHEIGHT
*****************************************/
int PutLinc( int imagerow,  int page, ImageLinc linc)
    {
     int row;
     unsigned  int count;
    count =  sizeof(Pixel) * IMAGEWIDTH;
    row = AbsRowNum(imagerow,page);
     if(!SetUpBlock(0,row,IMAGEWIDTH-1,row+1)) return(BAD_SETUPBLOCK);
     if(!WriteBuff(linc,count)) return(BAD_WRITEBUFF);
     return SUCCESS;
    }

Pixel ContrastPixel( int col,  int imagerow,  int page)
    {
     int row;
    Pixel linc, contrast;
     unsigned  int count;
    count =  sizeof(Pixel);
    row = AbsRowNum(imagerow,page);
     if(!SetUpBlock(col,row,col,row)) return(BAD_SETUPBLOCK);
     if(!ReadBuff(&linc,count)) return(BAD_READBUFF);
     if(linc>(PIXEL_MAX/2))contrast = 0;
     else contrast = PIXEL_MAX;
     if(!WriteBuff(&contrast,count)) return(BAD_WRITEBUFF);
     return linc;
```

```
    }

int SetAPixel( int col, int imagerow, int page, Pixel value)
    {
     int row;
     unsigned int count;
    count = sizeof(Pixel);
    row = AbsRowNum(imagerow,page);
     if(!SetUpBlock(col,row,col+1,row+1)) return(BAD_SETUPBLOCK);
     if(!WriteBuff(&value,count)) return(BAD_WRITEBUFF);
     return SUCCESS;
    }

/*****************************************
This routine generates a circle. It draws it in TArga memory. You may
not see it unless you display the page it is drawn on.
*****************************************/
 void circle( int page, int xcen, int ycen, int radius, Pixel color)
   {
   Rect frame;
   Point pt;
   pt.x = xcen;
   pt.y = AbsRowNum(ycen,page);
   CenterRect(&frame, pt, 2*radius, 2*radius);
   SetForeColor( ( unsigned long) color);
   StrokeOval(frame);
   }

/*****************************************
This routine draws a line.
*****************************************/
 void line( int page, int xs, int ys, int xe, int ye, Pixel color)
   {
   SetForeColor( ( unsigned long) color);
   MoveTo(xs,AbsRowNum(ys,page));
   LineTo(xe,AbsRowNum(ye,page));
   }

/*From here down is the overlay stuff*/
/*These hold the data so it can be erased*/

 static char overx[15], overy[15], overz[15], overw[15], overp[15], overr[15];

/*This prepares for live overlay but does not show anything*/
 int inittext( char *font)
 {
  char fontname[127];
```

```
    char *cnvstring;
    if(font[0] == 'd'){
     Status_Message("Using default font");
     cnvstring = getenv("TFONTS");
      if(cnvstring!=NULL){
      strcpy(fontname,cnvstring);
      strcat(fontname,"\\");
      strcat(fontname,TARGAFONT);
      }
      if(LoadStrokeFont(fontname)) return FAIL;
     }
    else{
     Status_Message("Using specified font");
      if(LoadStrokeFont(font)) return FAIL;
     }
   /*
   currRegs.liveMixSrc = 1;
   currRegs.liveMixInv = 1;
   */
   currRegs.compareEnb = 1;
   currRegs.overlaySrc = 2;
   currRegs.overlayInv = 0;
   currRegs.liveMixColor = 0;
   currRegs.alpha8 = 1;
   currRegs.liveMixGain = 1;
   currRegs.liveMixZero = 0;
   currRegs.notOVLLevel = 0;
   WriteAll(&currRegs);
   sprintf(overx,"X ");
   sprintf(overy,"Y ");
   sprintf(overz,"Z ");
   sprintf(overw,"y ");
   sprintf(overp,"p ");
   sprintf(overr,"r ");
    return SUCCESS;
  }

/*Turn on live overlay. To turn it off call Live_Video or set the
page.*/
 void liveoverlay( int page)
{
   unsigned char rList[] = {DISPMODE, 0};

  Show_Process_Image(page);
  currRegs.dispMode = 2;
  WriteSet(&currRegs, rList);
```

```
}

/*Draw screenpose on the overlay page*/
 void DrawOverlay(Pose *screenpose, int overlaypage)
{
 EraseOverlay(overlaypage);
 sprintf(overx,"X %6.1lf",screenpose->x);
 sprintf(overy,"Y %6.1lf",screenpose->y);
 sprintf(overz,"Z %6.1lf",screenpose->z);
  if(screenpose->yaw == BAD_PITCH_YAW)sprintf(overp,"p XXXXX");
     else sprintf(overp,"p %6.2lf",screenpose->pitch);
  if(screenpose->pitch == BAD_PITCH_YAW)sprintf(overw,"y XXXXX");
     else sprintf(overw,"y %6.2lf",screenpose->yaw);
 sprintf(overr,"r %6.1lf",screenpose->roll);
  #ifdef CHECKLENGTH
    if(( int)strlen(overx)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    if(( int)strlen(overy)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    if(( int)strlen(overz)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    if(( int)strlen(overp)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    if(( int)strlen(overw)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    if(( int)strlen(overr)>=15)Fatal_Error_Message("Blew it in DrawOverlay");
    #endif
 DrawOverStrings(WHITE,overlaypage);
 }

/* The vertical distance between rows of overlay text*/
 #define DELTAY (-30)
/* The starting location of the text*/
 #define XSTART (370)
 #define YSTART (450)
/*This routine writes the text in overx, overy ... onto Targa page
overlaypage using color.*/
 void DrawOverStrings(Pixel color, int overlaypage)
{
 int y;
y = YSTART;
writetext(overlaypage,XSTART,y,overx,color);
y = y + DELTAY;
writetext(overlaypage,XSTART,y,overy,color);
y = y + DELTAY;
writetext(overlaypage,XSTART,y,overz,color);
y = y + DELTAY;
writetext(overlaypage,XSTART,y,overp,color);
y = y + DELTAY;
writetext(overlaypage,XSTART,y,overw,color);
y = y + DELTAY;
writetext(overlaypage,XSTART,y,overr,color);
```

```
 }

/*The simplest logic to erase the overlay is to set it zero. SLOW*/
 void EraseOverlay( int overlaypage)
{
 ClearPage(overlaypage);
 }

/*This writes text on the screen*/
 void writetext( int page, int xs, int ys, char *text, Pixel color)          1000
  {
  SetForeColor( ( unsigned long) color);
  MoveTo(xs,AbsRowNum(ys,page));
  DrawString(text);
  }

/*The fonts take up memory. Dump them when finished.*/
 void finishtext()
 {
 DisposeStrokeFont();                                                        1010
 }

/*Ok I lied. These last two routines do a binary read and write to a
file and store the contents of the Targa registers. They are used in
conjuction with Initialize_Vision to avoid the system call.*/
 #define S_IWRITE 0200
 int DumpState( char *file, TplusRegs *reg)
  {
   int handle, numbytes, j;
  numbytes = sizeof(TplusRegs);
  handle = open(file, O_BINARY | O_WRONLY | O_CREAT, S_IWRITE);              1020
   if(handle == -1) return FAIL;
  j = write(handle, ( char *) reg, numbytes);
   if(j!=numbytes) return FAIL;
  close(handle);
   return SUCCESS;
 }

int ReadState( char *file, TplusRegs *reg)
 {                                                                           1030
  int handle, numbytes, j;
 numbytes = sizeof(TplusRegs);
 handle = open(file,O_BINARY | O_RDONLY);
  if(handle == -1) return FAIL;
 j = read(handle, ( char *) reg, sizeof(ImageLine));
  if(j!=numbytes) return FAIL;
 close(handle);
```

```
    return SUCCESS;
   }

 int SetLEDState( int OnOff)                                          1040
{

     unsigned  base_address, control_byte, port_a, port_b, port_c;
     int       select;

     base_address = 0x300;          /*base address of digital I/O port*/
     control_byte = base_address + 3;  /*control byte address          */
     port_a = base_address;         /*port A address                 */
     port_b = base_address + 1;     /*port B address                 */   1050
     port_c = base_address + 2;     /*port C address                 */

     select = 0x00;
     outp (control_byte, select);        /*set all ports to output          */

     outp (port_b, OnOff);          /*send integer to port b           */
     return 0;

}

 int PrinterPortAddress;                                              1060
 void setprinterportaddress( int address)
{
  PrinterPortAddress = address;
  }

 int SetLEDStateUsingPrinter( int OnOff)
{
    outp(PrinterPortAddress,~OnOff);
     return 0;                                                        1070
}
```

Chapter 14

targutil.c, targutil.h

Documentation Date: 3/10/95

14.1 Reading, and storing images.

The routines in this file read and write images from/to disk. One out of place routine displays images and pauses between each one.
New Data Types:
None.
Definitions: None.

14.1.1 Header File Listing:

```
#ifndef TARGUTIL_H
   #define TARGUTIL_H
  /*Stores (reads) image page in filename*/
   int StoreImage( char *filename, int page);
   int ReadImage( char *filename, int page);
  /*Stores (reads) all six (eight) images begining at page 0
     using filenames: zero.suffix one.suffix etc.*/
   void writeall6( char *suffix);
   void readall6( char *suffix);
   void readall8( char *suffix);
   void writeall8( char *suffix);

  /* Shows status message, says hit a key, shows the video page then
  waits*/
   void showpage( int page, char *message);
  /* Says what page is showing then shows it waits for key*/
   void showall6( void);
   void showall8( void);
   #endif
```

14.2 Function: StoreImage, ReadImage, writeall6, readall6, readall8, writeall8, showpage, showall6, showall8

Documentation Date: 3/12/95

Prototypes:

```
int StoreImage(char *filename, int page);
   int ReadImage(char *filename, int page);
   void writeall6(char *suffix);
   void readall6(char *suffix);
   void readall8(char *suffix);
   void writeall8(char *suffix);
   void showpage(int page, char *message);
   void showall6(void);
   void showall8(void);
```

Source File: *targutil.c*
Type of Function: User Callable
Header Files Used in targutil.c: *<stdio.h> <string.h> "datatype.h" "targutil.h" "targa8.h" "misc.h"*

Description:

1. StoreImage stores a page (one image) to a file. If the file cannot be opened then it returns FAIL (1) otherwise it returns SUCCESS (0). The default directory for the file is the current directory.

2. ReadImage reads a page (one image) from a file. If the file cannot be opened then it returns FAIL (1) otherwise it returns SUCCESS (0). The default directory for the file is the current directory.

3. writeall6 stores the first six images in files. You specify the suffix for the files then their names are data/zero, data/one etc. Note that if the subdirectory data does not exist in the current directory, then the routine does nothing. Note that error correction or at least reporting should be implemented. It would not be hard to do since the routine calls StoreImage which does report error.

4. writeall8 stores all eight images in files. You specify the suffix for the files then their names are data/zero, data/one etc. Note that if the subdirectory data in the current directory does not exist, then the routine does nothing. Note that error correction or at least reporting should be implemented.

5. showpage shows a frame of TARGA video (a page) then calls printandwait with a message to the user.

6. showall6 shows six images with a printandwait call between each one.

7. showall8 shows eight images with a printandwait call between each one.

8. readall6 reads six images from disk. If subdirectory data does not exist in nothing happens. This should at least implement an error return.

9. readall8 reads eight images from disk. If subdirectory data does not exist nothing happens. This should implement an error return.

14.2.1 Program Listing:

```
#include <stdio.h>
#include <string.h>
#include "datatype.h"
#include "targutil.h"
#include "targa8.h"
#include "misc.h"

/*This tga_header was put in by Greg Newman, I don't know why*/
unsigned char tga_header[18] = {0,0,3,0,0,0,0,0,0,0,0,0,0,2,0,2,8,48};

int StoreImageExt( char *file, char *suffix, int page);
int ReadImageExt( char *file, char *suffix, int page);

#define FILE_NAME_LENGTHS 20
int StoreImageExt( char *file, char *suffix, int page)
{
  char filename[FILE_NAME_LENGTHS];
  if(FormFullFileName(filename, FILE_NAME_LENGTHS, file, suffix) != SUCCESS)
    Fatal_Error_Message("Blew it in StoreImageExt");
  return StoreImage(filename,page);
}

int ReadImageExt( char *file, char *suffix, int page)
{
  char filename[FILE_NAME_LENGTHS];
  if(FormFullFileName(filename, FILE_NAME_LENGTHS, file, suffix) != SUCCESS)
    Fatal_Error_Message("Blew it in ReadImageExt");
  return ReadImage(filename,page);
}

/*This routine stores a page (one image) to a file. The file name is
given by: let file = "abcd" and suffix = "xyz" then the file is
"abcd.xyz". If the file cannot be opened then it returns FAIL (1)
```

```
 otherwise it returns SUCCESS (0)*/
 int StoreImage( char *filename,  int page)
{ ImageLine line;
   char message[80];
  FILE *ofile;
   int i;
  ofile = fopen(filename, "wb");                                    40
   if (ofile == NULL)  return FAIL;
  sprintf(message,"Saving Image %d to %s",page,filename);
 #ifdef CHECKLENGTH
 if(( int)strlen(message) >= 40)Fatal_Error_Message("Blew it in StoreImage");
 #endif
  Status_Message(message);
  fwrite(tga_header, sizeof(tga_header),1,ofile);
   for(i=0;i<IMAGEHEIGHT;i++)
   { GetLine(i, page, line);
     fwrite(line, sizeof(ImageLine),1,ofile);                       50
   }
  fclose(ofile);
   return SUCCESS;
}

/*This routine reads a page (one image) from a file. The file name is
given by: let file = "abcd" and suffix = "xyz" then the file is
"abcd.xyz". If the file cannot be opened then it returns FAIL (1)
otherwise it returns SUCCESS (0)*/
 int ReadImage( char *filename,  int page)                          60
{ ImageLine line;
   int i;
  FILE *ifile;
   char temp_header[18];
  ifile = fopen(filename,"rb");
   if (ifile == NULL)  return FAIL;
  fread(temp_header, sizeof(tga_header),1,ifile);
   for(i=0;i<IMAGEHEIGHT;i++)
   { fread(&line, sizeof(ImageLine),1,ifile);
     PutLine(i, page, line);                                        70
   }
  fclose(ifile);
   return SUCCESS;
}

/*This routine stores the first six images in files. You specify the
suffix for the files then their names are data\zero, data\one etc. Note
that if the subdirectory data in the current directory does not exist,
then the routine does nothing. Note that error correction or at least
reporting should be implemented. It would not be hard to do.*/      80
```

```
void writeall6( char *suffix)
 {
 StoreImageExt("data\\zero", suffix, 0);
 StoreImageExt("data\\one", suffix, 1);
 StoreImageExt("data\\two", suffix, 2);
 StoreImageExt("data\\three", suffix, 3);
 StoreImageExt("data\\four", suffix, 4);
 StoreImageExt("data\\five", suffix, 5);
 }

/*This routine stores all eight images in files. You specify the
suffix for the files then their names are data\zero, data\one etc. Note
that if the subdirectory data in the current directory does not exist,
then the routine does nothing. Note that error correction or at least
reporting should be implemented. It would not be hard to do.*/
 void writeall8( char *suffix)
  {
  StoreImageExt("data\\zero", suffix, 0);
  StoreImageExt("data\\one", suffix, 1);
  StoreImageExt("data\\two", suffix, 2);
  StoreImageExt("data\\three", suffix, 3);
  StoreImageExt("data\\four", suffix, 4);
  StoreImageExt("data\\five", suffix, 5);
  StoreImageExt("data\\six", suffix, 6);
  StoreImageExt("data\\seven", suffix, 7);
  }

/*This routine shows a frame of TARGA video (a page), prints a message
and waits for a key press*/
 void showpage( int page,  char *message)
{
     char mess[40];
    sprintf(mess,"%s %d - Hit Key",message,page);
#ifdef CHECKLENGTH
  if(( int)strlen(mess) >= 40)Fatal_Error_Message("Blew it in showpage");
#endif
    Show_Process_Image(page);
    printandwait( mess );
    }

/*This routine shows six images with a pause between each one*/
void showall6(){
   showpage(0,"Now showing page");
   showpage(1,"Now showing page");
   showpage(2,"Now showing page");
   showpage(3,"Now showing page");
   showpage(4,"Now showing page");
```

```
    showpage(5,"Now showing page");
    }

/*This routine shows eight images with a pause between each one*/
 void showall8(){
    showpage(0,"Now showing page");
    showpage(1,"Now showing page");
    showpage(2,"Now showing page");
    showpage(3,"Now showing page");
    showpage(4,"Now showing page");
    showpage(5,"Now showing page");
    showpage(6,"Now showing page");
    showpage(7,"Now showing page");
    }

/*This routine reads six images from disk. If subdirectory data does not
exist nothing happens. This should implement at least an error return.*/
 void readall6( char *suffix)
  {
  Show_Process_Image(0);
  ReadImageExt("data\\zero", suffix, 0);
  Show_Process_Image(1);
  ReadImageExt("data\\one", suffix, 1);
  Show_Process_Image(2);
  ReadImageExt("data\\two", suffix, 2);
  Show_Process_Image(3);
  ReadImageExt("data\\three", suffix, 3);
  Show_Process_Image(4);
  ReadImageExt("data\\four", suffix, 4);
  Show_Process_Image(5);
  ReadImageExt("data\\five", suffix, 5);
  }

  /*This routine reads eight images from disk.  If subdirectory data
does not exist nothing happens.  This should implement at least an error
return.*/
void readall8( char *suffix)
  {
  Show_Process_Image(0);
  ReadImageExt("data\\zero", suffix, 0);
  Show_Process_Image(1);
  ReadImageExt("data\\one", suffix, 1);
  Show_Process_Image(2);
  ReadImageExt("data\\two", suffix, 2);
  Show_Process_Image(3);
  ReadImageExt("data\\three", suffix, 3);
  Show_Process_Image(4);
```

```
ReadImageExt("data\\four", suffix, 4);
Show_Process_Image(5);
ReadImageExt("data\\five", suffix, 5);
Show_Process_Image(6);
ReadImageExt("data\\six", suffix, 6);
Show_Process_Image(5);
ReadImageExt("data\\seven", suffix, 7);
}
```

Index