HIGH LEVEL CONFIGURATION TOOL FOR DSP AND FPGA

PROJECT WORK DONE AT

KUMARAGURU COLLEGE OF TECHNOLOGY

P-1167

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT
FOR THE AWARD OF
BACHELOR OF ENGINEERING DEGREE IN INFORMATION TECHNOLOGY
OF BHARATHIAR UNIVERSITY, COIMBATORE

SUBMITTED BY

K.ARULMURUGAN (0027S0064)

D.PRABU (0027S0092)

GUIDED BY

Ms. N. SUGANTHI M.E., LECTURER

Department of Information Technology

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE - 641 006

MARCH 2004

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE - 641 006

CERTIFICATE

This is to certify that the project entitled

"HIGH LEVEL CONFIGURATION TOOL FOR DSP AND FPGA"

is the bonafide work of

K.ARULMURUGAN	(0027S0064)
D.PRABU	(0027S0092)

and submitted in partial fulfillment of the requirement for the award of Bachelor of Engineering Degree in Information Technology of Bharathiar university, Coimbatore

[illegible]

Head of the department
(Dr. S. THANGASAMY)

[illegible]

Guide
(Mrs. N.SUGANTHI)

Submitted for University Examination held on 27-3-2004

[illegible]

Internal Examiner

[illegible] 27/3/04

External Examiner

Declaration

We,

K.Arulmurugan **0027S0064**

D.Prabu **0027S0092**

declare that the project entitled "High Level Configuration Tool for DSP and FPGA", is done by us and to the best of our knowledge, a similar work has not been submitted earlier to the Bharathiar University or any other institution, for fulfillment of the requirement of the course study.

This project report is submitted on the partial fulfillment of the requirement for all awards of the degree of Bachelor of Engineering in Information Technology from Bharathiar University.

[illegible]

Place: Coimbatore. **[K.Arulmurugan]**

[illegible]

Date : 22-3-2004 . **[D.Prabu]**

Project Guided by

[illegible]

Mrs. N. Suganthi M. E.

Acknowledgement

We are highly obliged and wish to express our deep sense of gratitude to the **"Department of Information Technology" Kumaraguru College of Technology and Soliton Automation India Pvt.Ltd,** for the encouragement and help in carrying out this project work and also making it as a success.

We wish to thank our respected principal **Dr.K.K. Padmanaban . B.Sc (Engg).,** **M.Tech.,Ph.D.,** and Head of the Department **Dr.S.Thangasamy Ph.D.,** for their kind encouragement and for having provided us with all facilties.

We express our sincere thanks and indebtedness to our be-loved guide in-incharge **Mrs.N.Suganthi, M.E.,** Lecturer, Department of Information Technology and **Mr.S.K.Ramanan, Sr.Project Engineer** ,Soliton Automation for their valuable instructions and timely help throughout the work.

We are highly indebted to our course coordinator **Mrs.S.Devaki, B.E.,M.S.,** for her valuable guidance in bringing out the project well. We are extremely grateful to our class advisor **Ms.P.Sudha .B.E.,** for her support throughout the duration of the project.

We also express our thanks to all other faculty members and technical supporting staff of the Department of Computer Science and Engineering and the Department of Information Technology for their support and timely help.

We feel proud to pay our respectful thanks to our parents and friends for their enthusiasm and encouragement. Above all, we owe our gratitude to the Almighty, for showering abundant blessings on us.

Synopsis

This project "**A high level configuration tool for DSP and FPGA**" is aimed at creating a high level configuration tool for Image processing applications using DSP and FPGA chips .This project is done for **SOLITON AUTOMATION INDIA Pvt. Ltd.**

In the area of Machine Vision, the system makes use of special cameras called smart cameras which not only capture the images but does lot of real time image processing functions. The complex algorithms are done by specially designed DSP (Digital Signal Processor) or FPGA (Field programmable Gate Array) chips.

Based on the specific application we have to embed the image processing functions in these chips. The existing configuration tools are general purpose and relatively low configuration tools like CCS for DSP and ISE5 for FPGA. The programmer uses C or other HDL languages like Verilog to configure the chips.

This product acts as an **abstraction** over the existing tools. It can be used to interactively program image processing applications and produce the source files for the existing configuration tools. Finally we can invoke the existing tools to configure the chips.

Proposed tool is helpful for the application developers to visualize and create Image Processing applications interactively. Also it helps the low end customers to make subtle modifications and reconfigure as per their requirements.

TABLE OF CONTENTS

Introduction

1.1 Existing System.

The existing compilers have the following limitations

- The application developer has to provide low level C – source code for each and every operation. So there is a possibility for algorithmic errors.
- The application developer has to write implementation code for the entire image processing functions that is required for his applications.
- The application developer cannot visualize and program the application, so it becomes difficult to analyze and modify the parameters for the specified image processing functions. Hence testing the application becomes very difficult.
- The low end customer has to approach the applications developer even for subtle modifications.
- The user has to specify all the default options that is necessary for the compiler.
- Debugging cannot be easily done.

1.2 Proposed System

The proposed system provides the basic functionalities of the existing system and also has added advantages with some of the enhanced features such as

- Easy user interface

Since this application is entirely for developing image processing applications visualization of the various image processing functions becomes essential.

- No need for low level code.

The user is free from the burden of writing C- codes and processor specific instruction codes for configuring the processors. By using this application one can specify the sequence of operations and appropriate parameters for those functions by using common window controls.

- Lesser possibility for errors.

Since standard pre-defined image processing algorithms are used the possibility for occurrence of any algorithmic errors is greatly reduced or eliminated.

- Extendable and modifiable.

The sequence of the image processing functions and their required parameters can be easily modified and extended. Further refinement is made easy for the low end customers based on the varying industrial environment.

Software Requirement Specification

Software Requirement Specification

2.1 INTRODUCTION

2.1.1 Purpose

The primary purpose of this Software Requirement Specification (SRS) is to specify the functionality, interfaces and other requirements needed for the development of “**High Level Configuration Tool for DSP and FPGA**”. This specification is the primary document upon which all the subsequent design, source code and test plan will be based.

2.1.2 Scope

Proposed tool is helpful for the application developers to visualize and create Image Processing applications interactively. It acts as an abstraction over the existing general configuration tools. Also it helps the low end customers to make subtle modifications and reconfigure as per their requirements.

2.1.3 Definitions, Acronyms, Abbreviations

DSP : Digital Signal Processor. This includes a range of processors specially designed for performing complex arithmetic operations for manipulating signals.

FPGA: Field Programmable Gate Array .These chips contain millions of gates and they can be dynamically programmed to create the required circuit.

CCS : Code composer studio (IDE provided by Texas instruments for configuring DSP chips). Similar to cross compiler.

ISE5: Integrated Software Environment for configuring FPGA.

Configuration : Configuration here refers programming these chips.

2.2 GENERAL DESCRIPTION

2.2.1 Product Perspective :

The existing configuration tools are general purpose and relatively low configuration tools like CCS for DSP and ISE5 for FPGA. The programmer uses C or other HDL languages like Verilog to configure the chips.

2.2.2 Product Functions

This product acts as an abstraction over the existing tools. It can be used to interactively program image processing applications using DSP and FPGA. Finally we can invoke the existing tools to configure the chips.

2.2.3 User Characteristics

The user should be familiar with the processor configurations and basic image processing functions and their purpose.He should be able to identify the sequence of image processing techniques to develop the required application.

2.2.4 General Constraints

For using this product we should have the licensed version of CCS and ISE5 to be installed in the system.

2.3 SPECIFIC REQUIREMENTS

2.3.1 Functional Requirements

2.3.1.1 Introduction

The basic function of this product is to provide an interactive interface for developing applications. This product also eliminates any possibility of algorithmic errors as this product uses standard predefined image processing algorithms. Finally it should be able to invoke the existing tools with appropriate parameters for configuration.

2.3.1.2 List of Inputs:

- The processor specifications.
- Parameters for various image processing functions.
- Sequence of image processing functions.

2.3.1.3 Information processing requirement

The proposed tool gets the sequence of image processing operations from the user along with various parameters and stores it in a script file .Using this file the appropriate intermediate files for the existing configuration tools are generated .Then it invokes the cross compiler to configure the processor.

2.3.2 Performance Requirements

2.3.2.1 Memory

The tool should generate a code which is as small in size as possible as the code is to be burnt into the embedded system. The tool should add only a little extra code into the image processing algorithms to accomplish this.

2.3.2.2 Speed

The code generated by the tool should run in the processor at the fastest speed. The tool should add no additional complexity to the code.

2.3.3 Design Constraints

2.3.3.1 External Interface Requirement

We use Xilinx Parallel Cable IV for configuring the Xilinx FPGAs. One side of the Parallel Cable connects to the PC's parallel port and the other end to the JTAG pins of the FPGAs. Similarly the TI DSPs can be configured by XDS510PP Plus, an Emulator supplied by TI. The Emulator is connected to the PC through parallel port and connected to the DSP through JTAG pins. The DSP or the FPGA is inside the camera so the Video signals are obtained directly into the processor through its data lines.

2.3.3.2 User Interfaces, Screen Formats

The user interface is to be created such that the application developer can visualize the outcome of each Image processing functions during the development. There should be provisions for editing, altering the sequence and the parameters of the various processes.

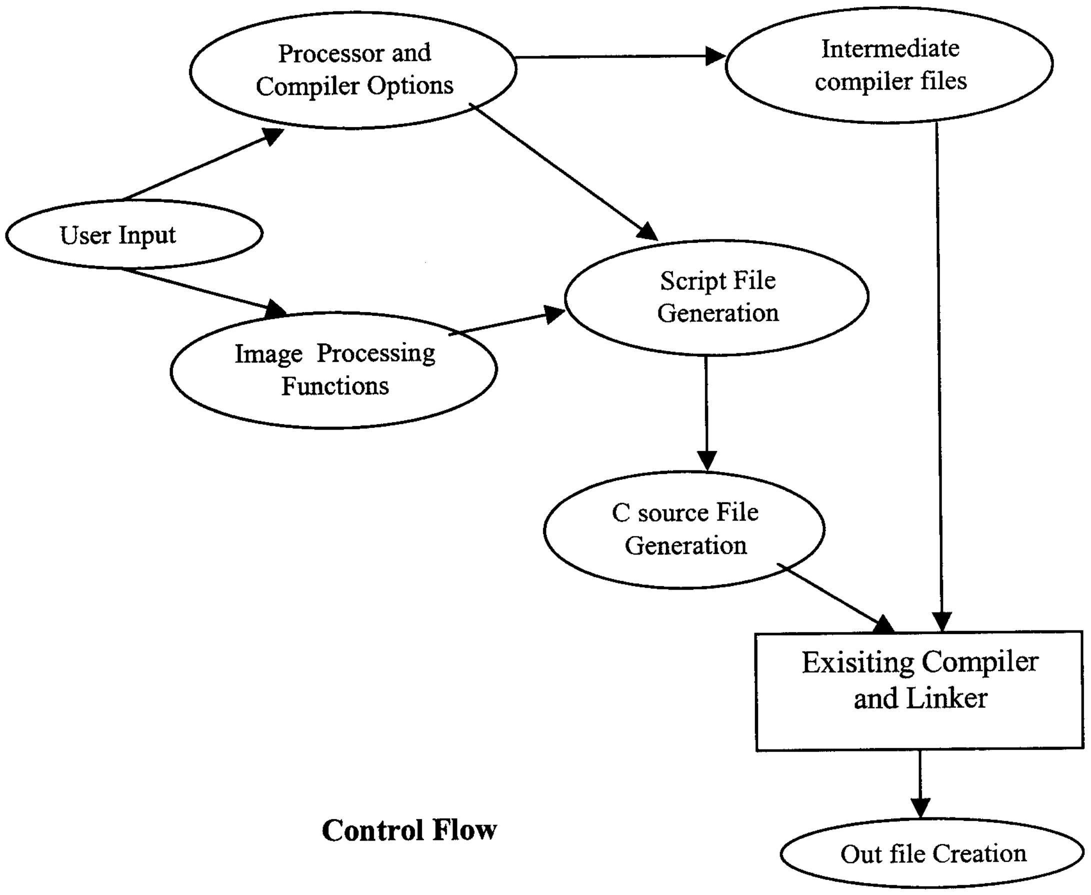
2.3.3.3 Hardware Interfaces and Software Interfaces with other systems

The hardware interface will be between the PC and the processor board which is inside the camera. The software interface will contain a script document that contains sequence of operations with parameters and processor configuration information which will be used to generate the intermediate files for the existing tools.

Design Details

Design Details

The software design of our system is done as follows:


Processor and Compiler Options
Intermediate compiler files
User Input
Script File Generation
Image Processing Functions
C source File Generation
Exisiting Compiler and Linker
Out file Creation

Control Flow

The control flow of our application is explained below.

First the application developer specifies the processor types and various compiler and linker specific options and other details for the work space creation. Also a sample bitmap image is specified and the sequences of operations are interactively specified by the application developer. Based on the information the necessary predefined header files are included and source files are generated. The files necessary for the compiling and linking the generated source files are created. Then compiling and linking is done by the existing configuration tool and finally the created output file is downloaded to the board.

The various modules in this project are:

- Frame work for interactive programming.
- Processor configuration specification
- Compiler options specification
- Script file generation
- C file generation
- Compiling and Linking

Frame work for interactive programming.

This module involves in creating an interactive user interface. The image processing operations can be added in sequence. The view was created to display the changes that take place in the image after applying various functions. The effect of altering the parameters for each of the function will be displayed side by side. The framework also provides options for adding, deleting and modifying the sequence of image processing functions to be applied in developing the application.

Processor configuration specification

Here we select the type of the processor series which has to be configured based on the developers choice and the appropriate header and library files that are specific for the processor family is included in the project.

Compiler and Linker options specification

This module involves providing various compiling options through simple selection controls for the existing compiler .These include options for basic, advanced, files, feedback, parser, diagnostics, assembly and preprocessor options for the compiler. These must be provided as parameters to the compiler.

The linker options include heap size, stack size, library include paths, code entry points, debug symbol merging, conditional linking and global static symbol options.

Script file generation

The script file generation includes saving the various selected compiler, linker and project specifications .It also holds the sequence of operations that were selected along with their parameters. The generated script file can be edited and modified.

C file generation

From the generated script file we create the C source files by including all the specified functions with including appropriate header files available separately for each of those functions. This file is saved to the workspace.

Compiling and Linking

This module involves developing an intermediate file that holds the information about the various compiler, linker and processor options. Then all the source files, header files and the intermediate files are provided to the existing compiler and the compilation is performed. After linking a final output file is generated.

Implementation

Implementation

4.1 Environment profile

To perform image processing functions, the DSP processor circuit should be configured and should be updated in the BIOS. Texas Instruments (TI) DSPs can be configured by XDS510PP Plus, an Emulator supplied by TI. The Emulator is connected to the PC through parallel port and connected to the DSP through JTAG pins. The application program has to be written using the editor provided by the IDE, compiled and embedded in the DSP board. The image to be processed is obtained from a high speed camera, and fed to the DSP circuit board directly. Then the real time image processing is done by the DSP processor.

The tool which is developed, acts as an abstraction over the existing compiler for DSP. This system is developed in Windows environment using Visual C++, as Windows provides easy to use interactive controls, which facilitates the development of image processing applications. Further Windows provides advanced APIS which can be effectively used to visualize the development stages.

4.2 Program Description

The various modules present in the project and the important implementation classes involved in each of the modules with their respective functions are explained below.

Frame work for interactive programming.

The classes used for implementing this functionality are **CMainFrame, CImageView, CImageDoc** and **CImageApp.**

CMainFrame includes user interfaces like status bar, menu bar, frame, title bar and all other document view controls. It also includes appropriate handlers for these controls that map to the corresponding functions to be performed. This is created by the MFC Appwizard.

In **CImageView**, we override the **OnDraw()** function to display the picture. This view is updated frequently to reflect the changes and functions that the application developer makes.

CImageDoc is an important document class that contains the information like image path that is used in other modules also. It also contains the sequence of image processing functions with their appropriate parameters, which will be saved and used for further operations.

CImageApp is the windows application object.

Processor and workspace configuration specification

The important class in this module is **ProjOption**.

This obtains information regarding the compiler location, project location, and project type, location of the library files, target processor type and the target library file location.

Compiler options specification

The important class in this module is the **compiler** .The other classes are

Advance:

This contains controls to get information like RTS modifications, AutoInlining Threshold, memory models, RTS calls, Interrupt Threshold, Speculate Threshold, etc.

Feedback:

This contains controls to get information regarding whether to generate optimizer comments, single iteration view of SP loops, etc.

Filesdlg:

This contains controls for specifying asm file extension, obj file extension, asm directory, obj directory, temporary file directory, absolute listing directory.

Assembly:

This contains controls to get information regarding whether to generate assembly listing, to keep generated assembly files, pre defined names, copy file, include file etc.

Diagnos:

This contains controls to set diagnostic information.

Parser:

This contains controls to get parsing options like embedding C++, output raw listing, supporting c++ runtime etc.

Preprocess:

This contains controls to get preprocessing options like whether to define or undefined symbols, specification of search paths etc.

Script file generation

Here, we override the **Serialize()** function of **CImageDoc** to serialize all the collected information and store it in a document format. This can be de-serialized when the document is opened for editing next time. The same information is used to generate the intermediate script file.

C file generation

The C file generation is done using **CMainFrame::OnScript()** message handling function. The other functions included in this module are

write_header():

This is used to include the appropriate header files for the selected image processing functions in the generated C file.

write_functions():

This is used to include the appropriate functions along with their arguments. The image pointer is passed as an argument to these functions.

Compiling and Linking

The compiling and linking is initiated by the handler CMainFrame:: **OnOptionCompilelink()** .we use the system command to invoke the existing compiler by supplying the **Debug.lkf** file to create the object file and then linking is done with the generated object files, to create the required **volume.out** file which will be downloaded to the processor board.

Testing

Testing

Software testing is the process of uncovering the defects in function, in logic, in implementation with the objective of assuring that defined inputs will produce actual results that agree with required results.

Testing is an important phase in the software development life cycle and no software is complete without putting it to severe testing.

The software developed was put to test at different stages .the individual modules developed were tested for consistency and correctness by supplying different inputs. The integrated system was also subjected to immense testing.

The unit testing for our system was done for the following conditions.

- The first module of creating the frame work was tested with varied inputs. The possible occurrences of errors were noted and some defects were traced back to the implementation phase. For example the user can't go for the next processing function without saving the parameters in the previous function.
- The processor configuration module and compiler options module were tested in detail by varying the options each time and noting that the correct argument for the selected option is generated in the intermediate file generation phase.
- The script file generation is developed based on the serialization concept and hence it was elaborately tested so that the de-serialization occurs correctly and the previously selected options are available when the document is opened the next time.
- The generation of C source file is checked by specifying varying options and by providing different sequences of operation. Also the appropriate

header files inclusion is checked against the selected image processing operations.

The integration testing involved the following:

The main modules to be integrated to the frame work were compiler and processor options specification and the source file generation. The application was tested as a whole by integrating the modules and the script file generated was used to produce the intermediate compiler files and the source code files .These files were then fed to the compiler to produce the required .out file. This proved to be successful.

Future Enhancements

Future Enhancements

- The application can be used to process only one image at a time. This can be extended to include more than one image at a time and perform some operations like subtraction, comparison etc.
- Now as such the application includes only two image processing operations but later on the remaining image processing operations can be added to the framework with appropriate controls.
- The speed of the program is a little less. This can be improved by storing the intermediate images in the buffer and then using the same for displaying
- Database capabilities can be added to the system so that the intermediate results can be added to the local database. This can be an enhancement to the development process.

Conclusion

Conclusion

The Ease of use and user friendliness of this tool for developing the application programs involving image processing is the foremost feature of this utility.It helps both the application developer and the end customer.

The application developers job of writing complex codes as well as following elaborate error checking procedures is greatly reduced and also the efficiency of the code is improved.

This tool can be supplied along with the finished product to the end customer who can make some small enhancements or modifications to suite his environment.It can serve as a complete tool for application development by adding all the possible image processing functions.

Bibliography

Bibliography

- **Books**
  - Jeff Prosise ,"**Programming windows with MFC**", WP Publishers &Distributors (P) LTD, 1999, Second Edition.
  - David J.Kruglinski, Scot Wingo and George Shepherd "**Programming Visual C++**", Microsoft Press, 1998, Fifth Edition.
  - Dennis M Ritchie, "**The C Programming Language**", Prentice Hall of India Private Limited, June 1999, Second Edition.
- **Reference Manuals**
  - TMS320C6000 Code Composer Studio Manuals.
  - TMS320C6000 Optimizing C Compiler User's Guide.
  - TMS320C64x DSP Library Programmer's Reference.
- **Websites**
  - www.ti.com/sc/docs/dsps/hotline/support.htm
  - www.ti.com/dsps

Appendix

Appendix

9.1 Sample Source code

```
#include "stdafx.h"
#include "image5.h"
#include "MainFrm.h"
#include "image5Doc.h"
#include "buildoption.h"
#include "myprog.h"
#include "dirtree.h"
#include <string.h>
#include <direct.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
//
int find_size(char *src );
int write_header(CImage5Doc *pDoc,CFile *mFile);
int write_functions( CImage5Doc *pDoc,CFile *output1);
char* extract1(char st[],char de[],int count)
{
  char *token=(char*)malloc(100);
  token=strtok(st,de);
  while((token!=NULL)&&(count!=1))
   { count--;
    token=strtok(NULL,de);
   }
   if(count==1)
     return token;
   return NULL;
}
/////////////////////////////////////////////////////////////////////////////
// CMainFrame

IMPLEMENT_DYNCREATE(CMainFrame, CFrameWnd)

BEGIN_MESSAGE_MAP(CMainFrame, CFrameWnd)
        //{{AFX_MSG_MAP(CMainFrame)
        ON_WM_CREATE()
        ON_COMMAND(ID_FUNCTIONS_GRAYSCALE, OnFunctionsGrayscale)
        ON_COMMAND(ID_FUNCTIONS_THRESHOLD, OnFunctionsThreshold)
        ON_BN_CLICKED(IDDONE, OnDone)
        ON_LBN_DBLCLK(IDC_LIST1, OnDblclkList1)
        ON_NOTIFY(NM_RELEASEDCAPTURE, IDC_SLIDER1,
        OnReleasedcaptureSlider1)
        ON_WM_MOVE()
        ON_COMMAND(ID_SCRIPT, OnScript)
        ON_COMMAND(ID_OPTION_BUILDOPTION, OnOptionBuildoption)
        ON_COMMAND(ID_PROGRESS, OnProgress)
        ON_COMMAND(ID_VIEW_REFRESH, OnViewRefresh)
        ON_COMMAND(ID_OPTION_COMPILELINK, OnOptionCompilelink)
        ON_COMMAND(ID_OPTION_PROJECTOPTIONS, OnOptionProjectoptions)
        ON_COMMAND(ID_MYDIALOG, OnMydialog)
        //}}AFX_MSG_MAP
   ON_MESSAGE(WM_UPDATE_LISTV, OnApply)
END_MESSAGE_MAP()

static UINT indicators[] =
{
        ID_SEPARATOR,           // status line indicator
        ID_INDICATOR_CAPS,
        ID_INDICATOR_NUM,
        ID_INDICATOR_SCRL,
};

/////////////////////////////////////////////////////////////////////////////
// CMainFrame construction/destruction

CMainFrame::CMainFrame()
{
        prog=NULL;
        // TODO: add member initialization code here

}

CMainFrame::~CMainFrame()
{
}

int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
        if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
                return -1;

   if (!m_windlist.Create(this, IDD_LISTING,
                CBRS_LEFT|CBRS_TOOLTIPS|CBRS_FLYBY, IDD_LISTING))
        {
                TRACE0("Failed to create DlgBar\n");
                return -1;      // fail to create
        }
  if (!m_windlist2.Create(this, IDD_DEF,
                CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
                IDD_DEF))
        {
                TRACE0("Failed to create DlgBar\n");
                return -1;      // fail to create
        }
        if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT, WS_CHILD |
        WS_VISIBLE | CBRS_TOP
                | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
        CBRS_SIZE_DYNAMIC) ||
                !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
        {
                TRACE0("Failed to create toolbar\n");
                return -1;      // fail to create
        }

        m_windlist.EnableDocking(CBRS_ALIGN_ANY);
        m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
        EnableDocking(CBRS_ALIGN_ANY);
        DockControlBar(&m_wndToolBar);
        return 0;
}

BOOL CMainFrame::PreCreateWindow(CREATESTRUCT& cs)
{
        if( !CFrameWnd::PreCreateWindow(cs) )
                return FALSE;

        cs.style = WS_OVERLAPPED | WS_CAPTION | FWS_ADDTOTITLE
                | WS_THICKFRAME | WS_SYSMENU | WS_MINIMIZEBOX |
                WS_MAXIMIZEBOX | WS_MAXIMIZE;
        return TRUE;
}

/////////////////////////////////////////////////////////////////////////////
// CMainFrame diagnostics

#ifdef _DEBUG
void CMainFrame::AssertValid() const
{
        CFrameWnd::AssertValid();
}
```

```
void CMainFrame::Dump(CDumpContext& dc) const
{
        CFrameWnd::Dump(dc);}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CMainFrame message handlers


void CMainFrame::OnFunctionsGrayscale()
{
        CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
        temp->AddString("grayscale");

                UINT nIndicator = ID_SEP;
                CStatusBar m_wndStatusBar;
m_wndStatusBar.Create (this);
m_wndStatusBar.SetIndicators (&nIndicator, 3);
 count=((CImage5Doc *)(this->GetActiveDocument()))->addcount();
        temp->SetCurSel(count-1);
        CMainFrame::ShowControlBar(&m_windlist2,0,0);
   m_windlist2.DestroyWindow();
    if (!m_windlist2.Create(this, IDD_GRAYSCALE,
                CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
                IDD_GRAYSCALE))
        {
                TRACE0("Failed to create DlgBar\n");
                return ;      // fail to create
        }
   currentop=2;
   CMainFrame::ShowControlBar(&m_windlist2,1,0);
        viewcount=count;
}

 void CMainFrame::OnFunctionsThreshold()
 {

        CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
        int nIndex=temp->AddString("threshold");
        UINT nIndicator = ID_SEPARATOR;
        CStatusBar m_wndStatusBar;
        m_wndStatusBar.Create (this);
        count=((CImage5Doc *)(this->GetActiveDocument()))->addcount();
        temp->SetCurSel(count-1);
        CMainFrame::ShowControlBar(&m_windlist2,0,0);
        m_windlist2.DestroyWindow();

        if (!m_windlist2.Create(this, IDD_THRESHOLD,
                CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
                IDD_THRESHOLD))
        {
                TRACE0("Failed to create DlgBar\n");
                return ;     // fail to create
        }

         CSliderCtrl * mslider=(CSliderCtrl*)m_windlist2.GetDlgItem(IDC_SLIDER1);
         CEdit* medit=(CEdit*)m_windlist2.GetDlgItem(IDC_EDIT2);
         CMainFrame::ShowControlBar(&m_windlist2,1,0);
         mslider->SetRange(0,256);
         mslider->SetPos(0);
         medit->SetWindowText("0");
         currentop=1;
         viewcount=count;

}

void CMainFrame::OnDone()
{

  int index;
  CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
  index=temp->GetCurSel();
  CString param(BuildString());
  ((CImage5Doc *)(this->GetActiveDocument()))->addstring(index,param);
  CMainFrame::ShowControlBar(&m_windlist2,0,0);
  m_windlist2.DestroyWindow();
    if (!m_windlist2.Create(this, IDD_DEF,
                CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
         IDD_DEF))
         {
                TRACE0("Failed to create DlgBar\n");
                return ;
         }

   CMainFrame::ShowControlBar(&m_windlist2,1,0);
   m_windlist.EnableWindow(true);
 }

 CString CMainFrame::BuildString()
 {
         CString temp1;
```

```
switch(currentop)
      {
      case 2:
              temp1="grayscale,";
              break;
      case 1:
              temp1="threshold,";
              char estring[4];
                      CSliderCtrl * mslider=(CSliderCtrl*)m_windlist2. GetDlgItem
                      (IDC_SLIDER1);
              sprintf(estring,"%d",mslider->GetPos());
              temp1.Insert(1000,estring);
              break;

      }
      presentop.Empty();
      presentop=temp1;
      return temp1;

}

void CMainFrame::OnDblclkList1()
{
 CString selected,scriptselect;
 int index;
 CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
 CImage5Doc* imdoc=(CImage5Doc *)(this->GetActiveDocument());
 index=temp->GetCurSel();
 viewcount=index+1;
 temp->GetText(index,selected);
 scriptselect=imdoc->GetOp(index);
 CMainFrame::ShowControlBar(&m_windlist2,0,0);
 m_windlist2.DestroyWindow();

 //first
 if(selected.Compare("threshold")==0)
 {
  if (!m_windlist2.Create(this, IDD_THRESHOLD,
                  CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
        IDD_THRESHOLD))
        {
                TRACE0("Failed to create DlgBar\n");
                return ;     // fail to create
        }
  currentop=1;
  CMainFrame::ShowControlBar(&m_windlist2,1,0);
```

```
CSliderCtrl * mslider=(CSliderCtrl*)m_windlist2.GetDlgItem(IDC_SLIDER1);
char *temp4=(char *)malloc(200);
 char *temp5=(char *)malloc(500);
 strcpy(temp4,scriptselect);
 temp5 =(extract1(temp4,", ",2));
 mslider->SetRange(0,256);
 mslider->SetPos(atoi(temp5));
 char estring[4];
CEdit* medit=(CEdit*)m_windlist2.GetDlgItem(IDC_EDIT2);
sprintf(estring,"%d",mslider->GetPos());
medit->SetWindowText(estring);

 }
//second
        else
        {
                currentop=2;
    if (!m_windlist2.Create(this, IDD_GRAYSCALE,
                CBRS_BOTTOM|CBRS_TOOLTIPS|CBRS_FLYBY|WS_VISIBLE,
                IDD_GRAYSCALE))
        {
                TRACE0("Failed to create DlgBar\n");
                return ;      // fail to create
        }
     CMainFrame::ShowControlBar(&m_windlist2,1,0);
        }
  m_windlist.EnableWindow(false);
  this->GetActiveDocument()->UpdateAllViews(NULL);
}

LRESULT CMainFrame::OnApply (WPARAM wParam, LPARAM lParam)
{
        char temp10[100];
        CImage5Doc* pDoc=(CImage5Doc *)(this->GetActiveDocument());
        ASSERT_VALID(pDoc);

        if(pDoc->show_list )
        {
        CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
        char *temp8=(char *)malloc(200);
        char *temp9=(char *)malloc(500);
        temp->ResetContent();
        for(int i=0;i<pDoc->GetCount();i++)
   {

          strcpy(temp8,pDoc->GetOp(i));
        temp9 =(extract1(temp8,",",1));
        temp->AddString(temp9);

 }
      pDoc->show_list =false;
      }


 CDC *pDC=m_windlist.GetDC();
 CDC memDC;

 HBITMAP hBitmap = (HBITMAP) ::LoadImage (NULL,pDoc->bitpath ,
 IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE | LR_CREATEDIBSECTION);
      if(hBitmap==NULL)
              return 0;
 pDoc->m_bitmap.Attach (hBitmap);
 CBitmap* pBitmap = pDoc->GetBitmap ();
 DIBSECTION ds;
 pBitmap->GetObject (sizeof (DIBSECTION), &ds);
 memDC.CreateCompatibleDC (pDC);
 memDC.SelectObject (pBitmap);
 itoa(pDC->StretchBlt (10,20,175,175, &memDC, 0, 0, ds.dsBm.bmWidth,
ds.dsBm.bmHeight, SRCCOPY),temp10,10);
 pDoc->m_bitmap.DeleteObject();
 return 0;
}


void CMainFrame::OnReleasedcaptureSlider1(NMHDR* pNMHDR, LRESULT*
pResult)
{
  char estring[4];
  CSliderCtrl * mslider=(CSliderCtrl*)m_windlist2.GetDlgItem(IDC_SLIDER1);
       CEdit* medit=(CEdit*)m_windlist2.GetDlgItem(IDC_EDIT2);
       sprintf(estring,"%d",mslider->GetPos());
       medit->SetWindowText(estring);
  int index;
  CListBox * temp=(CListBox*)m_windlist.GetDlgItem(IDC_LIST1);
  index=temp->GetCurSel();
   CString param(BuildString());
  ((CImage5Doc *)(this->GetActiveDocument()))->addstring(index,param);
  this->GetActiveDocument()->UpdateAllViews(NULL);
  *pResult = 0;
 }

 BOOL CMainFrame::OnCommand(WPARAM wParam, LPARAM lParam)
```

```
{

        return CFrameWnd::OnCommand(wParam, lParam);
}

void CMainFrame::OnMove(int x, int y)
{
        CFrameWnd::OnMove(x, y);

}

void CMainFrame::OnScript()
{
        int flag=0;
        char temp33[100];
        CString temp55;
        char str_main[]="void main(){\r\n";
        CString var="int image[640][480];\r\n";
        CImage5Doc* pDoc=(CImage5Doc *)(this->GetActiveDocument());
        temp55=pDoc->projopt.m_spl;
        temp55.Replace ("\\","\\\\");
        strcpy(temp33,temp55.GetBuffer (temp55.GetLength ()));
        MessageBox(temp33);
        if(chdir(temp33)!= 0 )
                MessageBox("chdir error");
        CFile output1("outfile.c",CFile::modeCreate|CFile::modeReadWrite );
        write_header(pDoc,&output1);
        output1.Write(str_main,sizeof(str_main)-1);
        output1.Write (var,var.GetLength ());
        write_functions(pDoc,&output1);
        output1.Write ("}\r\n",3);
        output1.Close ();


}

int find_size(char *src )
{
        int i=0;
        while(src[i]!='\r')
                i++;
        return (i+2);
}
```

```
int write_header(CImage5Doc *pDoc,CFile *output1)
{
        int flag=0;
        char *temp8=(char *)malloc(200);
   char *temp9=(char *)malloc(500);

        for(int k=0;k<pDoc->GetCount();k++)
   {
                  strcpy(temp8,pDoc->GetOp(k));
                   temp8=strtok(temp8,",");
                  strcat(temp8,".h\"\r\n");
                 if(k==0)
          {  flag=1;
          }
          else
          {
          for(int j=0;j<k;j++)
                 {
          strcpy(temp9,pDoc->GetOp(j));
                  temp9=strtok(temp9,",");
                  strcat(temp9,".h\"\r\n");
                  if(strcmp(temp8,temp9)==0 )
                  {    flag=0;
                          break;
                  }
                  else flag=1;
                 }
           }

           if(flag==1)
                  {
                          output1->Write ("#include \"",sizeof("#include "));
                          output1->Write (temp8,find_size(temp8));
                         flag=0;

                  }
          }
          return true;
   }

   int write_functions( CImage5Doc *pDoc,CFile *output1)
   {
          char *temp8=(char *)malloc(200);
          char *temp9=(char *)malloc(500);
          char   *temp10 = (char *) malloc(600);
                  for(int i=0;i<pDoc->GetCount();i++)
```

```
{
        strcpy(temp8,pDoc->GetOp(i));
   temp9=strtok(temp8,",");
        strcpy(temp10,temp9);

        strcat(temp10,"(image");
        while( temp9 != NULL )
        {
                temp9 = strtok( NULL, "," );
                 if(temp9!=NULL)
                 {
                  strcat(temp10,",");
                        strcat(temp10,temp9);
                 }
        }

         strcat(temp10,");\r\n");
         output1->Write (temp10,find_size(temp10));
  }
                return 0;
}

void CMainFrame::OnOptionBuildoption()
{
        char temp10[100];
        CImage5Doc* pDoc=(CImage5Doc *)(this->GetActiveDocument());
        pDoc->bo.m_compiler.m_filesdlg.m_cod=pDoc->projopt.m_spl+"\\Debug";
        pDoc->bo.m_compiler.m_filesdlg.OnChangeCod1();
        pDoc->bo.DoModal ();
        CFile output1("Debug.lkf",CFile::modeCreate|CFile::modeReadWrite );
        output1.Write (pDoc->projopt.m_clf,pDoc->projopt.m_clf.GetLength ());
        output1.Write(pDoc->bo.m_compiler.m_editcompile, pDoc->bo.m_compiler.
        m_editcompile.GetLength());
        output1.Write(pDoc->projopt.m_cpfiles,pDoc->projopt.m_cpfiles.GetLength ());
        output1.Write ("\r\n",2);
        output1.Write ("\r\n",2);
        getcwd(temp10,sizeof(temp10));
        output1.Write (pDoc->projopt.m_clo,pDoc->projopt.m_clo.GetLength ());
        output1.Write(pDoc->bo.m_linker.m_slinker, pDoc- >
        bo.m_linker.m_slinker.GetLength());
        output1.Write (pDoc->projopt.m_tlf , pDoc->projopt.m_tlf.GetLength ());
        CString lib4;
        lib4=pDoc->projopt.m_spl;
        lib4=" \""+lib4+"\\Debug\\";
        CString lib5;
        lib5=pDoc->projopt.m_spl;
```

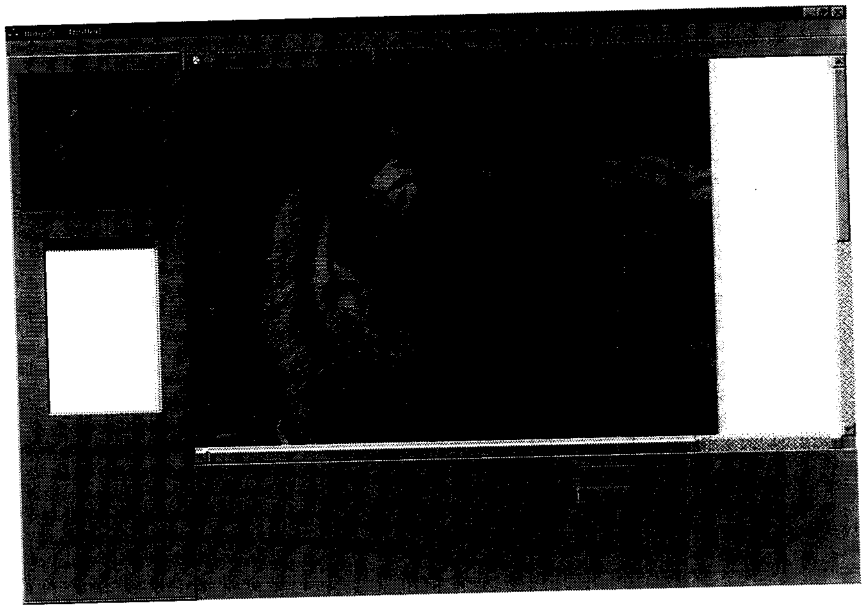
```
            CString lib3=" "+lib4+"load.obj\""+lib4+"vectors.obj\"
        \""+lib5+"\\volume.cmd\""+lib4+"outfile.obj\"";
        output1.Write (lib3,lib3.GetLength ());
        output1.Close ();

}

void CMainFrame::OnProgress()
{

        if(prog!=NULL)
        {
                prog->SetFocus();
        }
        else
        {
                prog=new myprog;
                prog->Create(IDD_PROGRESS);
                prog->step_prog(0,200,1);
                prog->progress_step();
                prog->progress_step();
                prog->ShowWindow (SW_SHOW);

         }
         prog->DestroyWindow();
         prog=NULL;
 }

 void CMainFrame::OnViewRefresh()
 {
                AfxGetMainWnd ()->SendMessage (WM_UPDATE_LISTV, 0, 0);
         }

 void CMainFrame::OnOptionCompilelink()
 {

         CImage5Doc* pDoc=(CImage5Doc *)(this->GetActiveDocument());
         CString temp11= "\""+ pDoc->projopt.m_cl + "/bin/cl6x.exe -@\"debug.lkf\"";

         CString temp,temp3;
         char temp1[100],temp33[100];

         temp3=pDoc->projopt.m_spl;
         temp3.Replace ("\\","\\\\");
         strcpy(temp33,temp3.GetBuffer (temp3.GetLength ()));
         if(chdir(temp33)!= 0 )
```

```
                MessageBox("chdir error");

        temp=pDoc->projopt.m_spl+"\\Debug";
        temp.Replace ("\\","\\\\");
        strcpy(temp1,temp.GetBuffer (temp.GetLength ()));
        rmdir(temp1);
        if(mkdir(temp1)!=0)
                MessageBox("mkdir error");

        if(system(temp11.GetBuffer (temp11.GetLength ())))
        {
                MessageBox("compiling unsuccessfull: ERROR ");
        }
        else
        {
                        MessageBox("compiling successfull: SUCCESSFULL");
        }

}

void CMainFrame::OnOptionProjectoptions()
{
        CImage5Doc* pDoc=(CImage5Doc *)(this->GetActiveDocument());
        CString temp11=" \"load.asm\" \"vectors.asm\" \"outfile.c\"";
        pDoc->projopt.m_cpfiles=temp11;
        CString lib1="-z -i\"C:/ti/c6000/bios/lib\" -i\"C:/ti/c6000/rtdx/lib\" -
i\"C:/ti/c6000/cgtools/lib\"";
        pDoc->projopt.m_clo=lib1;
        pDoc->projopt.DoModal();
}

void CMainFrame::OnMydialog()
{

        dirtree t;
        t.DoModal ();
}


// image5View.cpp : implementation of the CImage5View class
//

#include "stdafx.h"
#include "image5.h"
#include "image5Doc.h"
#include "image5View.h"
```

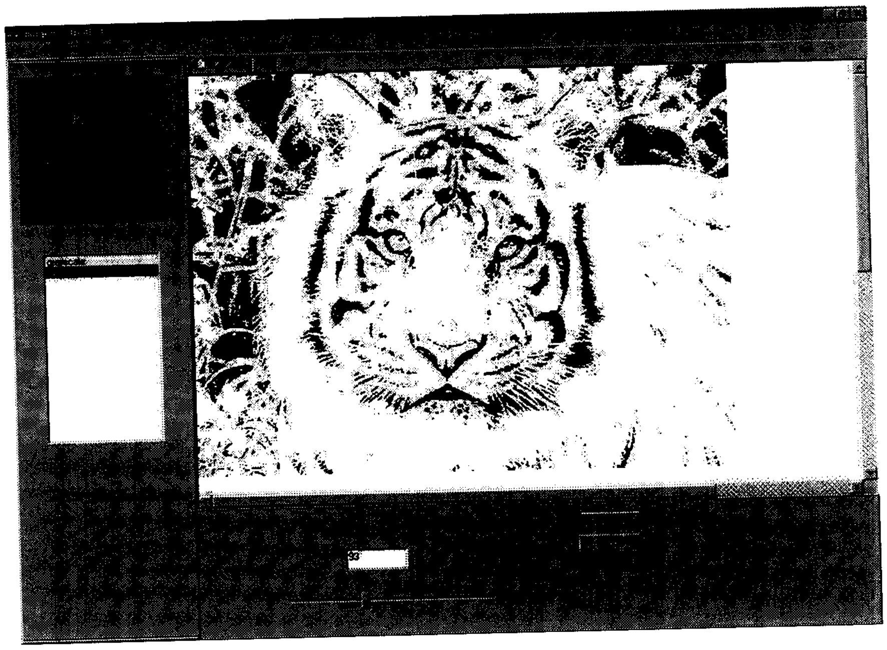
```
#include "MainFrm.h"
#include<string.h>

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

char* extract2(char st[],char de[],int count)
{
  char *token=(char*)malloc(100);
  token=strtok(st,de);
  while((token!=NULL)&&(count!=1))
  { count--;
   token=strtok(NULL,de);
  }
  if(count==1)
    return token;
  return NULL;
}

/////////////////////////////////////////////////////////////////////////////
// CImage5View

IMPLEMENT_DYNCREATE(CImage5View, CScrollView)

BEGIN_MESSAGE_MAP(CImage5View, CScrollView)
	//{{AFX_MSG_MAP(CImage5View)
		// NOTE - the ClassWizard will add and remove mapping macros here.
		//    DO NOT EDIT what you see in these blocks of generated code!
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CImage5View construction/destruction

CImage5View::CImage5View()
{
	hBitmap=NULL;
	prog=NULL;
	flag=0;
}

CImage5View::~CImage5View()
{
}

BOOL CImage5View::PreCreateWindow(CREATESTRUCT& cs)
{
        return CScrollView::PreCreateWindow(cs);
}

//////////////////////////////////////////////////////////////////////////////
// CImage5View drawing

void CImage5View::OnDraw(CDC* pDC)
{

        CWaitCursor wait;
        int vcount,i=0,j=0;
        CImage5Doc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);
        vcount=((CMainFrame *)(this->GetTopLevelFrame()))->viewcount;
        AfxGetMainWnd ()->SendMessage (WM_UPDATE_LISTV, 0, 0);

  if(vcount>0)
  {
   if(!flag)
        { hBitmap = (HBITMAP) ::LoadImage (NULL, pDoc->bitpath,
                        IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE |
                        LR_CREATEDIBSECTION);
   m_bitmap.Attach (hBitmap);
   m_bitmap.GetObject (sizeof (DIBSECTION), &ds);
   CDC memDC;
   memDC.CreateCompatibleDC (pDC);
   memDC.SelectObject (hBitmap);
   for(int i=0;i<ds.dsBm.bmWidth;i++)
        for(int j=0;j<ds.dsBm.bmHeight;j++)
                pDoc->info[i][j]=memDC.GetPixel(i,j);
   m_bitmap.DeleteObject();
   flag=1;
   }

       for(int i=0;i<ds.dsBm.bmWidth;i++)
        for(int j=0;j<ds.dsBm.bmHeight;j++)
          temp1[i][j]=pDoc->info[i][j];

   for(i=0;i<vcount;i++)
        {
                Invokefun(pDoc->GetOp(i),temp1,640,480);
        }
  for( i=0;i<ds.dsBm.bmWidth;i++)
        for( j=0;j<ds.dsBm.bmHeight;j++)
          pDC->SetPixel(i,j,temp1[i][j]);
  wait.Restore ();
 }
}

void CImage5View::OnInitialUpdate()
{
        CScrollView::OnInitialUpdate();
        SetScrollSizes(MM_TEXT, CSize(1000,1000));
}

/////////////////////////////////////////////////////////////////////////////
// CImage5View diagnostics

#ifdef _DEBUG
void CImage5View::AssertValid() const
{
        CScrollView::AssertValid();
}

void CImage5View::Dump(CDumpContext& dc) const
{
        CScrollView::Dump(dc);
}

CImage5Doc* CImage5View::GetDocument() // non-debug version is inline
{
        ASSERT(m_pDocument->IsKindOf(RUNTIME_CLASS(CImage5Doc)));
        return (CImage5Doc*)m_pDocument;
}
#endif //_DEBUG

/////////////////////////////////////////////////////////////////////////////
// CImage5View message handlers

void CImage5View::Invokefun(CString func,COLORREF mem [][480], int width, int
height)
{
        if(func.IsEmpty())
                return ;
        CString funcname;
        char *temp4=(char *)malloc(500);
        char *temp5=(char *)malloc(200);
```

```
    strcpy(temp4,func);
    temp5 =(extract2(temp4,", ",1));

    //threshold
    if(strcmp(temp5,"threshold")==0)
    {
    strcpy(temp4,func);
    temp5=(extract2(temp4,", ",2));
      int cmp= atoi(temp5);
      for(int i=0;i<width;i++)
      {
                          for(int j=0;j<height;j++)
                      {

                      COLORREF temp=mem[i][j];
                      int r=GetRValue(temp)+ GetGValue(temp) +GetBValue(temp);
                        if(r>cmp)
                              mem[i][j]=RGB(255,255,255);
                      else
                               mem[i][j]=RGB(0,0,0);
                      }
       }
      }

      //grayscale

      if(strcmp(temp5,"grayscale")==0)
      {
          for(int i=0;i<width;i++)
       {
              for(int j=0;j<height;j++)
                      {
                      COLORREF temp=mem[i][j];
                      int r=GetRValue(temp) +GetGValue(temp)+ GetBValue(temp);
                         r=r/3;
                      mem[i][j]=RGB(r,r,r);

                      }
        }
       }

return ;
}

void CImage5View::progress_start(int i, int j, int k)
{
        prog->SetFocus();
        prog->step_prog(i,j,k);
        prog->ShowWindow (SW_SHOW);
}
```

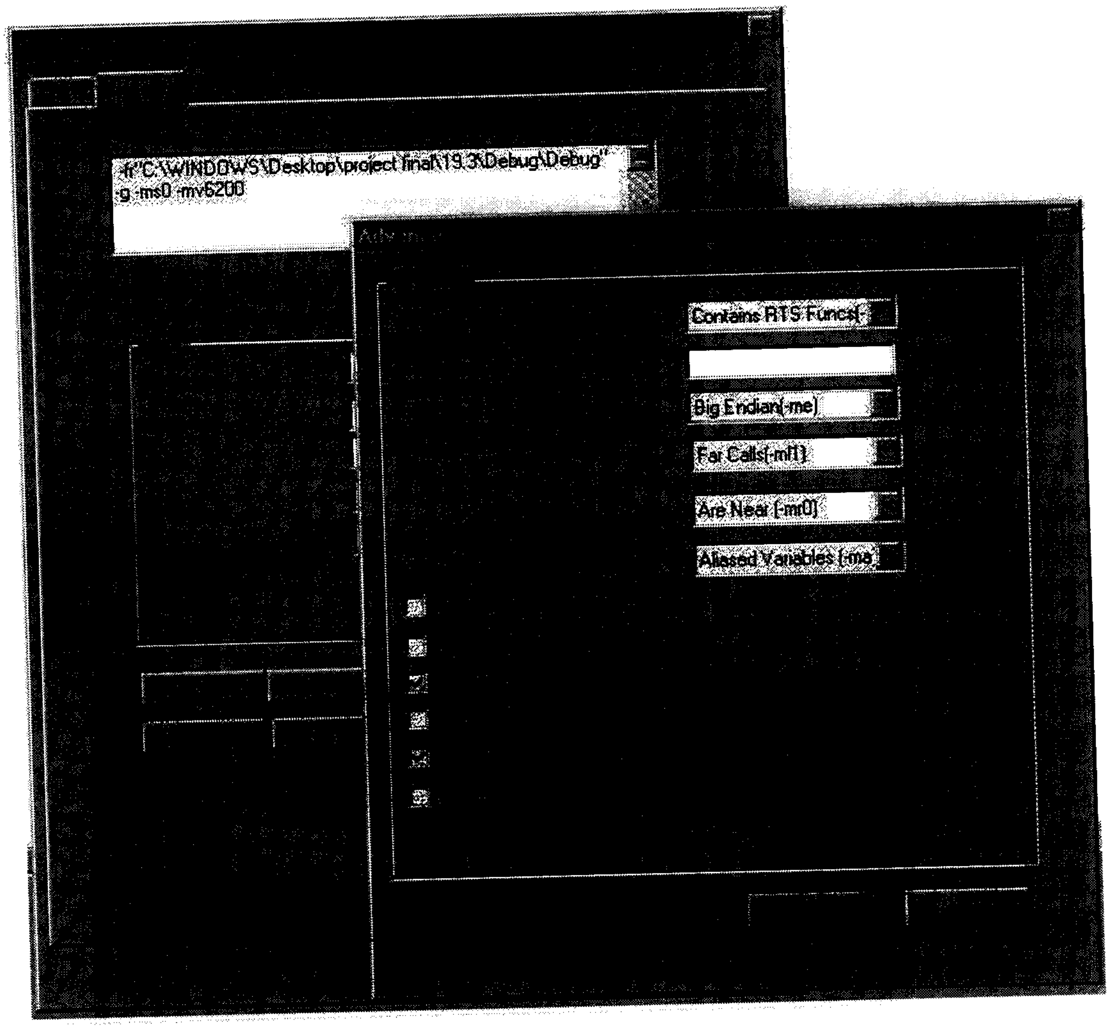
9.2 Sample Screen Shots



User Interface Framework



Compiler options specification



```
-fr"C:\WINDOWS\Desktop\project final\19.3\Debug\Debug"
-g -ms0 -mv6200
```

Contains RTS Funcs(-

Big Endian(-me)

Far Calls(-mf1)

Are Near [-mr0]

Aliased Variables [-ma