PictureBrowser

CSEE 4840 Embedded System Design

Jean Kongpinda

Stephane Nyombayire

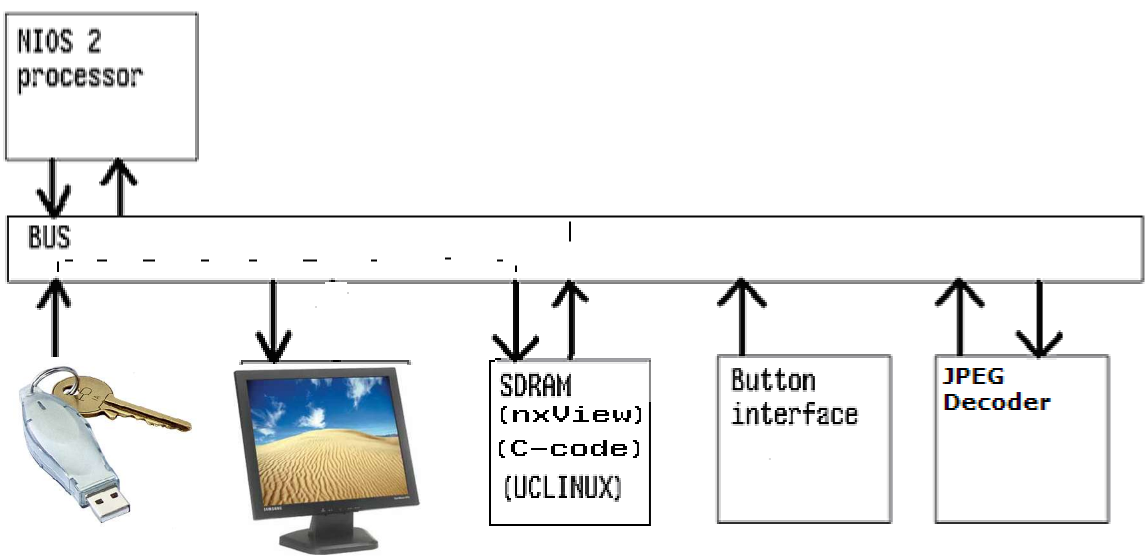
Joseanibal Colon-Ramos

Ian Roth

Table of Contents

Chapter 1: Overview

The purpose of our project is to design a picture browser using the DE2 Altera Board. The Altera Board comes equipped with a USB host, VGA output connection and four pushbuttons among other features. The picture browser finds the JPEG files located on a USB mass storage and display them on a VGA monitor attached to the board. The pushbuttons are then used to browse through the pictures. Two of the buttons are used to move forward and backward on the picture, one button is used to stretch the image while the four one ends the picture browser application. Figure 1 illustrates our system architecture:

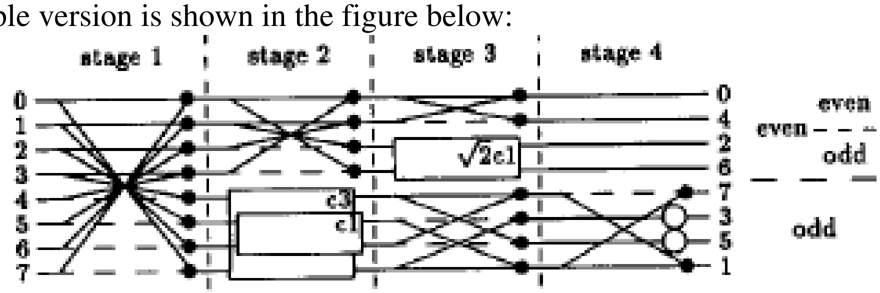


In order to achieve this, we installed the μClinux operating system on the SDRAM of the DE2 board. This operating system was configured to detect and mount the USB mass storage. We then wrote software to locate the JPEG files on the USB mass storage and call an image viewer program used for display. μClinux comes with a pre-installed JPEG image viewer called NXView. However, the JPEG decoding used by the NXView is fairly slow. We decided to have a portion of the JPEG decoding performed on the hardware of the DE2 Altera Board in order to speed up the decoding process. The decoded image was then displayed on the monitor through the VGA interface.

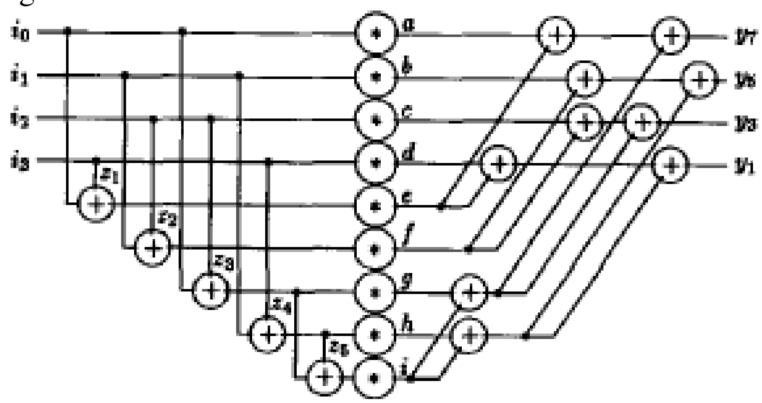
Chapter 2: JPEG Decoder

The inverse discrete cosine transform is completed in hardware due to the many arithmetic operators used in its implementation. The JPEG decoder is used by libjpeg, the software library used by nano-X, nxview, and thus the PictureBrowser software. The seminal paper "Practical Fast 1-D DCT Algorithms with 11 Multiplications" by Christoph Loeffler, Adriaan Ligtenberg and George S. Moschytz was consulted for further information about the transform and graphics from the paper have been used here.

The IDCT requires dequantization of the input values, thus 16 values are sent for each use of the hardware JPEG decoder for a total of 8 full multipliers. The rest of the multipliers are reduced since the input values are multiplied by constant "rotators". The most simple version is shown in the figure below:

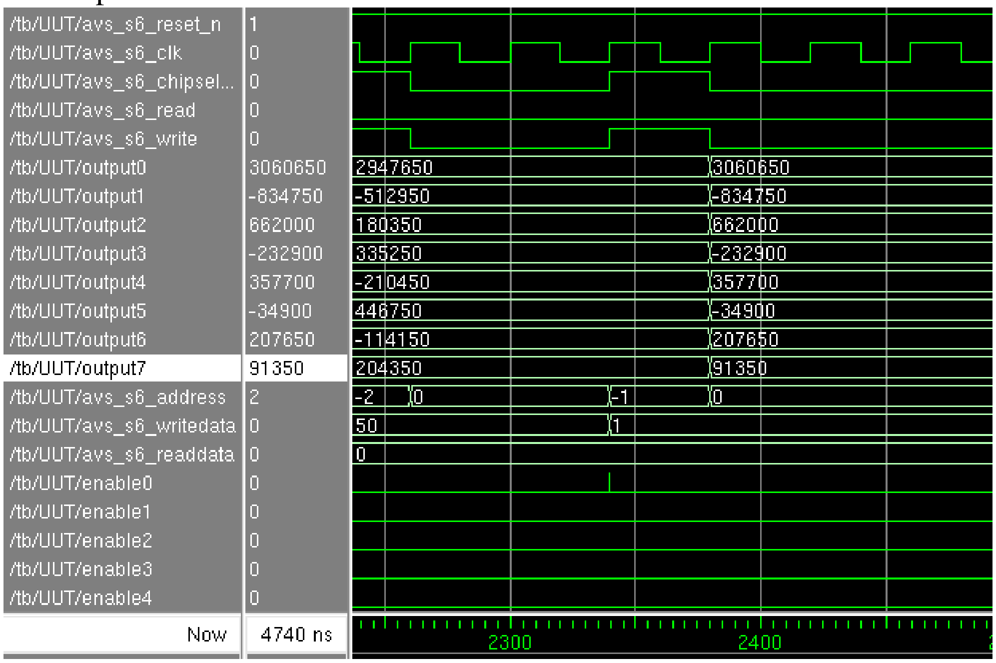


The solid circles represent addition if the lines are solid, and subtraction if the lines are dashed. The squares represent multiplication by the rotators and a lined circle is a multiplication by the square root of 2. The most basic algorithm has the fewest multiplications and additions, but there are multiple multiplications per input; an undesirable feature which adds delay. To correct this issue, 12 parallel multiplications can be used at the cost of more additions per line. The figure for the reconstructed odd portion of the algorithm is shown below:



Register Transfer Logic Design, although a familiar topic to the team, was rejected as a design methodology. The limiting factor was the speed of sequentially reading and writing input and output values; not the number of gates on the FPGA. The design team did not feel it necessary to reuse the multipliers since the component fit onto the FPGA without any problem. The highly parallel design is both simple and reusable in applications such as DSP where multiple values could arrive at the same time. One would only have to remove the input registers and output multiplexer. The stages are also set up to be about the same delay, thus pipelining would be simple to implement.

The hardware synthesizer of Quartus was able to reduce the hardware design to a point where it could be added to the original design and still fit on the available FPGA. Although it was suggested that multiple JPEG decoders were possible, it was found that the hardware design was only limited to the speed at which the 16 values could be loaded into the device and not the combinatorial logic. The waveform showing the speed at which the output values are asserted follows:



Another design consideration was the writing of input values to registers. In order to be able to write in a value at each clock cycle, addressing had to be used. First instincts led to the use of a simple FSM that sequentially enabled each input register, however the use of a counter proved to be impossible in one clock cycle, especially since the design tools would not allow the use of both the rising and falling clock edge. The addressing has the input values inputed in even addresses and the dequantization values inputed in the odd addresses. The reading is done on the lowest 8 addresses with the addresses corresponding to the output names.

The JPEG decoder hardware is implemented in the VHSIC Hardware Description

Language. Original plans called for the shifting of the output values before they were read, however this was not possible with the CAD tools used. The operator need for this shifting, sra, was not available in the IEEE libraries. Quartus included the arithmetic libraries that allowed for the sra operator, but ModelSim did not include the arithmetic libraries. The last shifting is completed in software.

The JPEG decoder was written to interface with the Altera Avalon Bus Fabric. An identifier of "s6" was used for the JPEG decoder. The bus readdata and writedata vectors were of the signed integer type, as were most other signals used to carry the input and dequantization values. Readdata and writedata were 32 bits in length with 31 as the sign bit and 0 as the LSB.

Chapter 3: Hardware System

The JPEG decoder is part of a much larger hardware system taken from Altera's software CD. The original purpose of the system was to demonstrate the VGA and PS/2 capabilities of the DE2 board. Our application required the slowing of the clock to 50MHz, but all other hardware remains a part of the system, even if it is not required for the PictureBrowser. Hardware drivers are used to interact with the VGA framebuffer, pushbuttons, and USB host device. The JPEG decoder is addressed directly in memory since it does not have an interrupt request need.

The original hardware is implemented in Verilog. The JPEG decoder was written in VHDL and the test bench was written in Verilog. No problems were encountered mixing the two languages in the SOPC system builder and the ModelSim simulator.

Chapter 4: PictureBrowser Software

4.1 Hardware/Software Interface:

JPEG decoding in the hardware gets sixteen 32 bit signed integers as input from the software and outputs another set of sixteen 32 bit signed integers to the jidcint.c (lib/libjpeg). Our software component accesses the decoder as if it was a 16 word, 32 bit memory device. We read and write to the Avalon bus through IO_RD_32DIRECT and IOWR_32DIRECT. These two functions which are defined in io.h (lib/libjpeg/) are macros defined after native input/output functions specific to the NIOs2 Board.

Functions signatures for these two functions are as follows:

IOWR_32DIRECT(BASE, OFFSET, DATA);

IORD_32DIRECT(BASE, OFFSET);

The base describes the address that SOPC assigned to the JPEG component (decoder) in the hardware. The offset is 4 bytes since we are writing/reading 32 bits signed integers to/from the hardware. Finally, as mentioned above the data is a 32 bit signed integer.

4.2 Software:

We perform the inverse discrete cosine transform along with dequantization of the JPEG decoder in the hardware, in order to accomplish the task we had to disable the existing features in the libjpeg directory.

4.2.1 IDCT and Dequantization:

4.2.1.1 jddctmanager.c:

This file manages which idct is to be used by the jpeg decoder, we modified this file in order to only allow the slow inverse discrete cosine transform (idct_slow). The IDCT_SLOW is defined in jidctint.c (lib/libjpeg), it only deals with 8 x 8 DCTs.

4.2.1.2 jidcint.c:

Each of the row of the matrix are 32 bits signed integers, they are iteratively (8 times) written to the hardware along with their dequantizer counterparts. The result of the decoding from the hardware is in turn descaled and read into the output matrix.

4.2.2 Button Reader and JPEG finder:

We enabled the driver for the Altera DE2 Board buttons by adding a new module via menuconfig of the uClinux OS. We followed the steps described in the nios2 wiki to install the module. Additionally, we wrote a c program (pictureviewer.c) that recursively iterates given a root directory through directories in order to find JPEG path filenames. This program is case insensitive and does not distinguish between jpg and jpeg file extension. The path filenames are stored into an array of strings (global variable); with the four buttons a user can either view the next picture, previous one, stretch the current picture or just exit the program. The executable of pictureviewer.c is copied into the romfs directory in order to execute it on the board before we zip up the zImage. At boot up, we automatically mount the USB driver through the modified boot up script. At execution our program goes recursively through folders of the mounted USB mass storage and notify the user of the JPEG files found. The user can now navigate through her JPEG pictures and even stretch them for an enjoyable feeling. However, it is also important to note that nano-X (graphical windowing) has to be killed and restarted everytime we run nxview (interface that calls our modified libjpeg library). This causes a delay in the display of pictures but this delay is significantly less than the same process running using the original libjpeg.

Chapter 5: Support Software

Our project heavily relied on uClinux, below are the existing applications that we took advantage of:

- Nano-X: a graphical windowing application
- Nxview: application that allows display of various file formats
- Busybox: customizable system calls

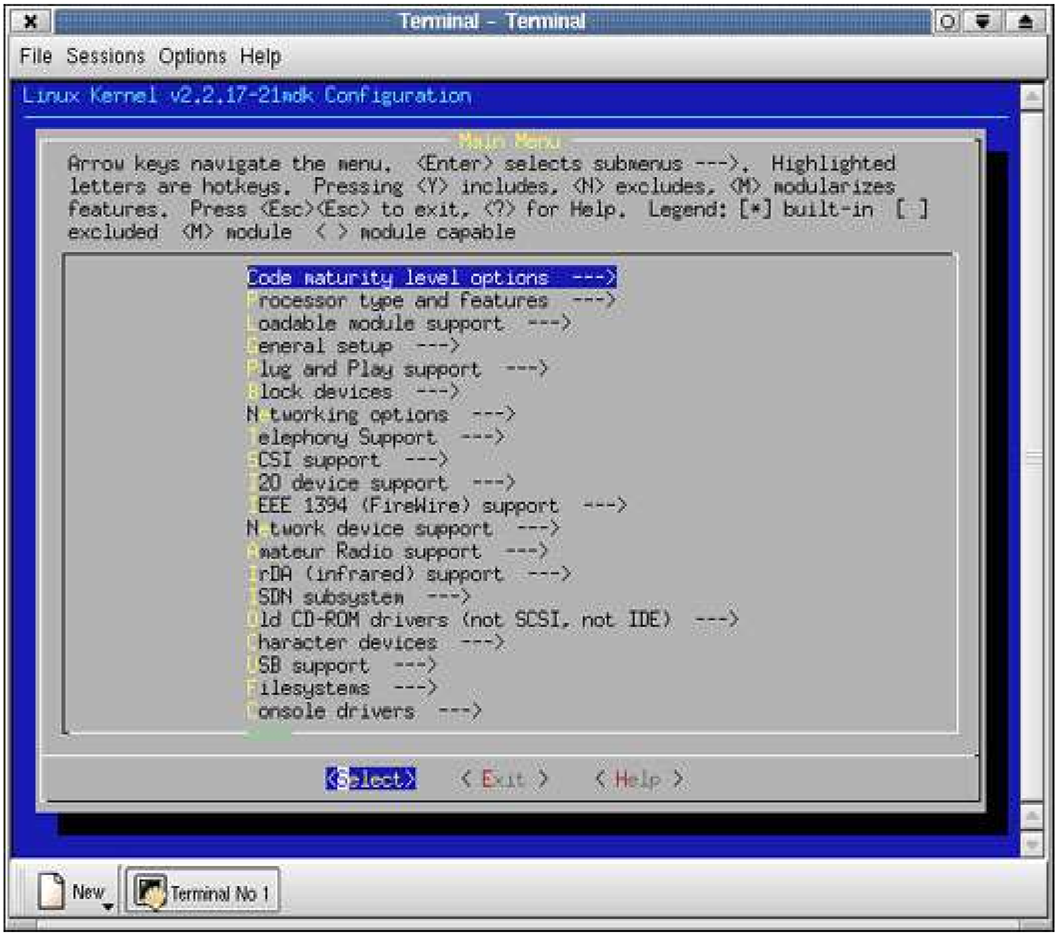
More documentation about these applications can be found at these websites:

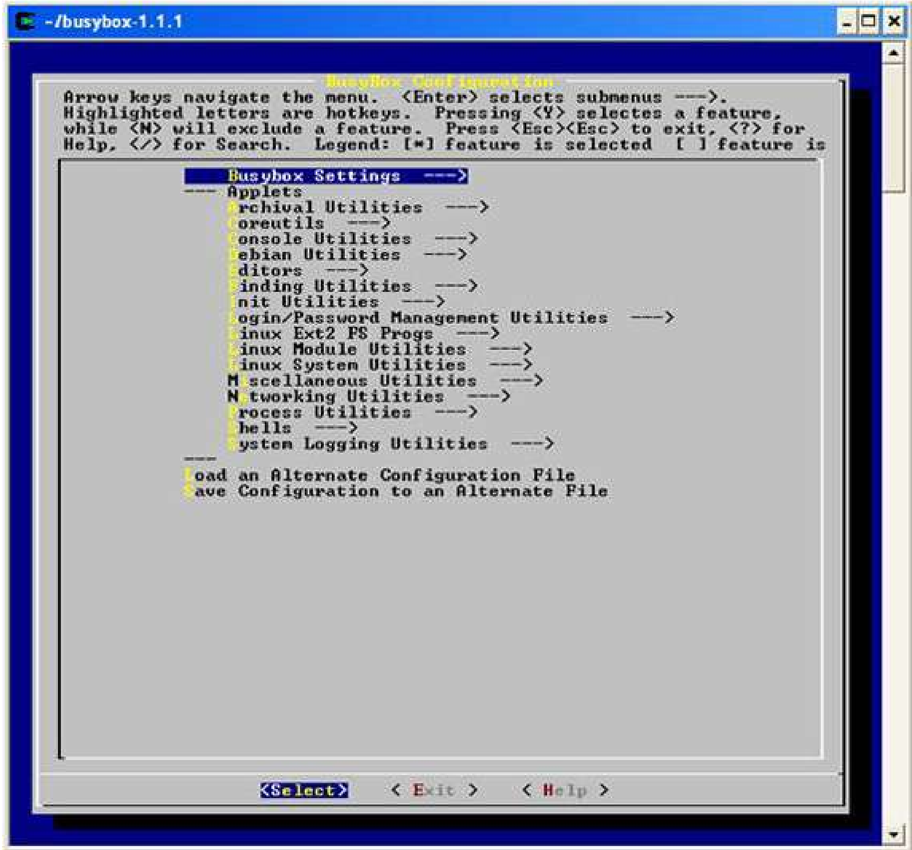
http://nioswiki.jot.com/WikiHome/OperatingSystems/UClinuxDist

http://nioswiki.jot.com/WikiHome/OperatingSystems/%C2%B5Clinux/UsbHost

http://nioswiki.jot.com/WikiHome/OperatingSystems/%C2%B5Clinux/FrameBuffer

Here are some useful screen shots for uClinux menu config:


Terminal - Terminal
File Sessions Options Help
Linux Kernel v2.2.17-21mdk Configuration
Main Menu
Arrow keys navigate the menu. <Enter> selects submenus --->. Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help. Legend: [*] built-in [ ] excluded <M> module < > module capable
Code maturity level options --->
Processor type and features --->
Loadable module support --->
General setup --->
Plug and Play support --->
Block devices --->
Networking options --->
Telephony Support --->
SCSI support --->
I2O device support --->
IEEE 1394 (FireWire) support --->
Network device support --->
Amateur Radio support --->
IrDA (infrared) support --->
ISDN subsystem --->
Old CD-ROM drivers (not SCSI, not IDE) --->
Character devices --->
USB support --->
Filesystems --->
Console drivers --->
<Select> < Exit > < Help >
New
Terminal No 1


~/busybox-1.1.1
Arrow keys navigate the menu. <Enter> selects submenus --->.
Highlighted letters are hotkeys. Pressing <Y> selectes a feature,
while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for
Help, </> for Search. Legend: [*] feature is selected [ ] feature is
Busybox Settings --->
--- Applets
rchival Utilities --->
oreutils --->
onsole Utilities --->
ebian Utilities --->
ditors --->
inding Utilities --->
nit Utilities --->
ogin/Password Management Utilities --->
inux Ext2 FS Progs --->
inux Module Utilities --->
inux System Utilities --->
M scellaneous Utilities --->
N tworking Utilities --->
rocess Utilities --->
hells --->
ystem Logging Utilities --->
---
oad an Alternate Configuration File
ave Configuration to an Alternate File
<Select> < Exit > < Help >

Chapter 6: Summary

The system is able to display JPEG pictures as planned. Implementing the Inverse Discrete Cosine Algorithm on the hardware proved to be faster than its software implementation. For that reason, the pictures are displayed faster. As a matter of fact, our program took about 100 microseconds less than before to open a JPEG picture of about 60 kB. This difference becomes very significant as you increase the picture size and the picture resolution. We were able to flip through the pictures using the pushbuttons.

One of the biggest issues we had was dealing the μClinux operating software. In order to debug and test out design, we had to recompile the kernel every time. That recompilation took about an average of 30-40 minutes. It was also difficult to debug problems on the μClinux software itself since it has a fairly large code base. Also, the DE2 development itself has limited resources. It turned out that the 8MB SDRAM available on it is not the ideal size for a μClinux kernel. However, we were able to carefully program it to run both our software and the operating system. On the hardware side of things, we wish we started working on it early enough because it turned out to be quite a challenge. The main problem was interfacing the rest of the software JPEG decoding with the result from the IDCT performed on hardware. ModelSim was very helpful as a simulation program to test the hardware. Using ModelSim, we were able to check the validity of the hardware IDCT by comparing it wave outputs to the outputs we would expect in a C IDCT. However, Quartus and ModelSim have different VHDL libraries so we had to try to make our VHDL code compatible to both for testing purposes.

The most important lesson we learned how to handle debugging processes. The nice thing about having a project on picture decoding is that from analyzing the resulting pictures, you can make intelligent guesses on which part of the decoding process is faulty. Also, simulation on hardware is very helpful and we hoped we had started using it from the beginning. It saved us a lot of Quartus compilation time. In addition, the interfacing between hardware and software must be done delicately. We should have first tried a small sample code to test the timing and communication between the two. μClinux is a great operation system, but there are still some improvement to be made.

Chapter 7: Work Breakdown

Jean Kongpinda: Hardware simulation, μClinux configuration, nxview modification.

Stephane Nyombayire: libjpeg modification and testing, software/hardware interface, button configuration.

Joseanibal Colon-Ramos: PictureBrowser software, software integration.

Ian Roth: Hardware IDCT, hardware simulation, nxview modification.

Chapter 8: Code

8.1 Hardware

JPEG_decode.vhd

```
-------------------------------------------------------------
-----------------------
--
-- JPEG_decode
--
-- Modified 5/05/2007
-- Ian Roth
-------------------------------------------------------------
-----------------------

library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity JPEG_decode is

  port (
     avs_s6_clk,
     avs_s6_chipselect,
     avs_s6_reset_n,
     avs_s6_read,
     avs_s6_write : in std_logic;
     avs_s6_address: in std_logic_vector(3 downto 0);
     avs_s6_writedata : in signed(31 downto 0);
     avs_s6_readdata : out signed(31 downto 0)
  );

end JPEG_decode;

architecture rtl of JPEG_decode is
```

```
component reggate is

  port (
     clk,reset_n : in std_logic;
     input : in signed(31 downto 0);
     enable : in std_logic;
     output : out signed(31 downto 0)
  );

end component reggate;

component mux is

  port (
     input0, input1, input2, input3,
     input4, input5, input6, input7 : in signed(31
downto 0);
     sel : in std_logic_vector(2 downto 0);
     output : out signed(31 downto 0)
  );
end component mux;

component quantmux is

  port (
     input : in signed(31 downto 0);
     dequant : in signed(31 downto 0);
     output : out signed(31 downto 0)
  );

end component quantmux;

  constant cos0 : signed(31 downto 0) :=
"00000000000000000000100110001110";-- 2446 FIX
(0.298631336)
  constant cos1 : signed(31 downto 0) :=
"00000000000000000000110001111100";-- 3196 FIX
(0.390180644)
  constant cos2 : signed(31 downto 0) :=
"00000000000000000001000101010001";-- 4433 FIX
```

```
(0.541196100)
  constant cos3 : signed(31 downto 0) :=
"00000000000000000001100001111110";-- 6270 FIX
(0.765366865)
  constant cos4 : signed(31 downto 0) :=
"00000000000000000001110011001101";-- 7373 FIX
(0.899976223)
  constant cos5 : signed(31 downto 0) :=
"00000000000000000010010110100001";-- 9633 FIX
(1.175875602)
  constant cos6 : signed(31 downto 0) :=
"00000000000000000011000000001011";-- 12299 FIX
(1.501321110)
  constant cos7 : signed(31 downto 0) :=
"00000000000000000011101100100001";-- 15137 FIX
(1.847759065)
  constant cos8 : signed(31 downto 0) :=
"00000000000000000011111011000101";-- 16069 FIX
(1.961570560)
  constant cos9 : signed(31 downto 0) :=
"00000000000000000100000110110011";-- 16819 FIX
(2.053119869)
  constant cos10 : signed(31 downto 0):=
"00000000000000000101001000000011";-- 20995 FIX
(2.562915447)
  constant cos11 : signed(31 downto 0):=
"00000000000000000110001001010100";-- 25172 FIX
(3.072711026)
  constant ncos1 : signed(31 downto 0) :=
"11111111111111111111001110000100";-- 3196 FIX
(0.390180644)
  constant ncos4 : signed(31 downto 0) :=
"11111111111111111110001100110011";-- 7373 FIX
(0.899976223)
  constant ncos7 : signed(31 downto 0) :=
"11111111111111111100010011011111";-- 15137 FIX
(1.847759065)
  constant ncos8 : signed(31 downto 0) :=
"11111111111111111100000100111011";-- 16069 FIX
(1.961570560)
```

```
  constant ncos10 : signed(31 downto 0):=
"11111111111111111010110111111101";-- 20995 FIX
(2.562915447)

  signal
enable0,enable1,enable2,enable3,enable4,enable5,enable6,
enable7,

enable8,enable9,enable10,enable11,enable12,enable13,enab
le14,enable15 : std_logic;
  signal tobus : signed(31 downto 0);
  signal enableout : std_logic;
  signal enable : std_logic_vector(15 downto 0);
  signal dequant0,
          dequant1,
          dequant2,
          dequant3,
          dequant4,
          dequant5,
          dequant6,
          dequant7 : signed(31 downto 0);
  signal input0,
          input1,
          input2,
          input3,
          input4,
          input5,
          input6,
          input7 : signed(31 downto 0);
  signal stage10,
          stage11,
          stage12,
          stage13,
          stage14,
          stage15,
          stage16,
          stage17 : signed(31 downto 0);
  signal stage22, stage23 : signed(63 downto 0);
  signal stage20,
          stage21,
```

```
          stage24,
          stage25,
          stage26,
          stage27,
          stage28,
          stage29,
          stage210,
          stage211,
          stage212 : signed(31 downto 0);
  signal stage30,
          stage31,
          stage32,
          stage33,
          stage34,
          stage35,
          stage36,
          stage37,
          stage38,
          stage39,
          stage310,
          stage311,
          stage312 : signed(63 downto 0);
  signal stage40,
          stage41,
          stage42,
          stage43,
          stage44,
          stage45,
          stage46,
          stage47 : signed(63 downto 0);
  signal temp0, temp1, temp2, temp3, temp4, temp5, temp8
: signed(63 downto 0);
  signal temp6, temp7 : signed(31 downto 0);
  signal output0,
          output1,
          output2,
          output3,
          output4,
          output5,
          output6,
```

```
          output7 : signed(31 downto 0);

begin

  REGMUX : mux port map (
    input0 => output0,
    input1 => output1,
    input2 => output2,
    input3 => output3,
    input4 => output4,
    input5 => output5,
    input6 => output6,
    input7 => output7,
    sel => avs_s6_address(2 downto 0),
     output => tobus
  );

  R0 : reggate port map (
     clk => avs_s6_clk,
     reset_n => avs_s6_reset_n,
     input => avs_s6_writedata,
     enable => enable0,
     output => input0
  );

  R1 : reggate port map (
     clk => avs_s6_clk,
     reset_n => avs_s6_reset_n,
     input => avs_s6_writedata,
     enable => enable2,
     output => input1
  );

  R2 : reggate port map (
     clk => avs_s6_clk,
     reset_n => avs_s6_reset_n,
     input => avs_s6_writedata,
     enable => enable4,
     output => input2
  );
```

```
R3 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable6,
   output => input3
);

R4 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable8,
   output => input4
);

R5 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable10,
   output => input5
);

R6 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable12,
   output => input6
);

R7 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable14,
   output => input7
);
```

```
Q0 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable1,
   output => dequant0
);

Q1 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable3,
   output => dequant1
);

Q2 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable5,
   output => dequant2
);

Q3 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable7,
   output => dequant3
);

Q4 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable9,
   output => dequant4
);
```

```
Q5 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable11,
   output => dequant5
);

Q6 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable13,
   output => dequant6
);

Q7 : reggate port map (
   clk => avs_s6_clk,
   reset_n => avs_s6_reset_n,
   input => avs_s6_writedata,
   enable => enable15,
   output => dequant7
);

DE0 : quantmux port map (
   input => input0,
   dequant => dequant0,
   output => stage10
);

DE1 : quantmux port map (
   input => input1,
   dequant => dequant1,
   output => stage11
);

DE2 : quantmux port map (
   input => input2,
   dequant => dequant2,
```

```
    output => stage12
  );

  DE3 : quantmux port map (
    input => input3,
    dequant => dequant3,
    output => stage13
  );

  DE4 : quantmux port map (
    input => input4,
    dequant => dequant4,
    output => stage14
  );

  DE5 : quantmux port map (
    input => input5,
    dequant => dequant5,
    output => stage15
  );

  DE6 : quantmux port map (
    input => input6,
    dequant => dequant6,
    output => stage16
  );

  DE7 : quantmux port map (
    input => input7,
    dequant => dequant7,
    output => stage17
  );

  temp6 <= stage10 + stage14;
  temp7 <= stage10 - stage14;
  stage20 <= temp6(31) & temp6(17 downto 0) &
"0000000000000"; -- shift by CONST_BITS
  stage21 <= temp7(31) & temp7(17 downto 0) &
"0000000000000"; -- shift by CONST_BITS
  temp8 <= (stage12 + stage16) * cos2;
```

```
stage22 <= (stage16 * ncos7) + temp8;
stage23 <= (stage12 * cos3) + temp8;
stage24 <= stage17;
stage25 <= stage15;
stage26 <= stage13;
stage27 <= stage11;
stage28 <= stage24 + stage27;
stage29 <= stage25 + stage26;
stage210 <= stage26 + stage24;
stage211 <= stage25 + stage27;
stage212 <= stage210 + stage211;
stage30 <= stage20 + stage23;
stage31 <= stage21 + stage22;
stage32 <= stage21 - stage22;
stage33 <= stage20 - stage23;
stage34 <= stage24 * cos0;
stage35 <= stage25 * cos9;
stage36 <= stage26 * cos11;
stage37 <= stage27 * cos6;
stage38 <= stage28 * ncos4;
stage39 <= stage29 * ncos10;
stage310 <= stage210 * ncos8;
stage311 <= stage211 * ncos1;
stage312 <= stage212 * cos5;
temp0 <= stage310 + stage312;
temp1 <= stage311 + stage312;
temp2 <= stage34 + stage38 + temp0;
temp3 <= stage35 + stage39 + temp1;
temp4 <= stage36 + stage39 + temp0;
temp5 <= stage37 + stage38 + temp1;
stage40 <= stage30 + temp5;
stage41 <= stage31 + temp4;
stage42 <= stage32 + temp3;
stage43 <= stage33 + temp2;
stage44 <= stage33 - temp2;
stage45 <= stage32 - temp3;
stage46 <= stage31 - temp4;
stage47 <= stage30 - temp5;
output0 <= stage40(63) & stage40(30 downto 0); --
Reduce bit count for multiplied lines and keep the sign
```

```
bit
  output1 <= stage41(63) & stage41(30 downto 0);
  output2 <= stage42(63) & stage42(30 downto 0);
  output3 <= stage43(63) & stage43(30 downto 0);
  output4 <= stage44(63) & stage44(30 downto 0);
  output5 <= stage45(63) & stage45(30 downto 0);
  output6 <= stage46(63) & stage46(30 downto 0);
  output7 <= stage47(63) & stage47(30 downto 0);

  enable0 <= avs_s6_write and enable(0) and
avs_s6_chipselect;
  enable1 <= avs_s6_write and enable(1) and
avs_s6_chipselect;
  enable2 <= avs_s6_write and enable(2) and
avs_s6_chipselect;
  enable3 <= avs_s6_write and enable(3) and
avs_s6_chipselect;
  enable4 <= avs_s6_write and enable(4) and
avs_s6_chipselect;
  enable5 <= avs_s6_write and enable(5) and
avs_s6_chipselect;
  enable6 <= avs_s6_write and enable(6) and
avs_s6_chipselect;
  enable7 <= avs_s6_write and enable(7) and
avs_s6_chipselect;
  enable8 <= avs_s6_write and enable(8) and
avs_s6_chipselect;
  enable9 <= avs_s6_write and enable(9) and
avs_s6_chipselect;
  enable10 <= avs_s6_write and enable(10) and
avs_s6_chipselect;
  enable11 <= avs_s6_write and enable(11) and
avs_s6_chipselect;
  enable12 <= avs_s6_write and enable(12) and
avs_s6_chipselect;
  enable13 <= avs_s6_write and enable(13) and
avs_s6_chipselect;
  enable14 <= avs_s6_write and enable(14) and
avs_s6_chipselect;
  enable15 <= avs_s6_write and enable(15) and
```

```
avs_s6_chipselect;

  with avs_s6_address select
    enable <= "0000000000000001" when "0000",
                 "0000000000000010" when "0001",
                 "0000000000000100" when "0010",
                 "0000000000001000" when "0011",
                 "0000000000010000" when "0100",
                 "0000000000100000" when "0101",
                 "0000000001000000" when "0110",
                 "0000000010000000" when "0111",
                 "0000000100000000" when "1000",
                 "0000001000000000" when "1001",
                 "0000010000000000" when "1010",
                 "0000100000000000" when "1011",
                 "0001000000000000" when "1100",
                 "0010000000000000" when "1101",
                 "0100000000000000" when "1110",
                 "1000000000000000" when "1111",
                 "0000000000000000" when others;

  enableout <= avs_s6_read and avs_s6_chipselect;

  with enableout select
    avs_s6_readdata <= tobus when '1',

"00000000000000000000000000000000" when others;


end architecture rtl;
```

mux.vhd

```
-- Modified 3/17/2007
-- Ian Roth

library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;

entity mux is

  port (
     input0, input1, input2, input3,
     input4, input5, input6, input7 : in signed(31
downto 0);
     sel : in std_logic_vector(2 downto 0);
     output : out signed(31 downto 0)
  );
end mux;

architecture behavior of mux is
begin
  with sel select
     output(31 downto 0) <=  input0(31 downto 0) when
"000",
                       input1(31 downto 0) when "001",
                       input2(31 downto 0) when "010",
                       input3(31 downto 0) when "011",
                       input4(31 downto 0) when "100",
                       input5(31 downto 0) when "101",
                       input6(31 downto 0) when "110",
                       input7(31 downto 0) when others;
```

end behavior;

quantmux.vhd

```
----------------------------------------------------------
-- quantmux.vhd
--
-- Modified 4/27/2007
-- Ian Roth
----------------------------------------------------------

library ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;


entity quantmux is

  port (
     input : in signed(31 downto 0);
     dequant : in signed(31 downto 0);
     output : out signed(31 downto 0)
  );

end quantmux;

architecture behavior of quantmux is

  signal temp : signed(31 downto 0);
  signal multi : signed (63 downto 0);

begin

  multi <= (input * dequant);
  temp <= multi(31 downto 0);

  with dequant select
     output <= input when
"00000000000000000000000000000001",
               temp when others;

end behavior;
```

reggate.vhd

```
-- Modified 4/19/2007
-- Ian Roth

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity reggate is

  port (
     clk, reset_n : in std_logic;
     input : in signed(31 downto 0);
     enable : in std_logic;
     output : out signed(31 downto 0)
  );

end reggate;

architecture behavior of reggate is
begin
  process (clk, reset_n, enable)
  begin
     if reset_n = '0' then
       output <= "00000000000000000000000000000000";
     elsif clk'event and clk = '1' then
       if enable = '1' then
         output <= input;
       end if;
     end if;
  end process;

end behavior;
```

tb.v

```
module tb;

reg avs_s6_clk;
reg  avs_s6_chipselect;
reg  avs_s6_reset_n;
reg  avs_s6_read;
reg  avs_s6_write;
reg [3:0] avs_s6_address;
reg  [31:0] avs_s6_writedata;

wire [31:0] avs_s6_readdata;

JPEG_decode UUT (
     avs_s6_clk,
     avs_s6_chipselect,
     avs_s6_reset_n,
     avs_s6_read,
     avs_s6_write,
     avs_s6_address,
     avs_s6_writedata,
     avs_s6_readdata );


initial begin

     avs_s6_clk <= 0;
     avs_s6_chipselect <= 0;
     avs_s6_reset_n <= 0;
     avs_s6_read <= 0;
     avs_s6_write <= 0;
     avs_s6_writedata <= 0;
    avs_s6_address <= 0;

    #200;
    avs_s6_reset_n <= 1'b1;
    #310;
```

```
     nios_write(4'd0, 32'd50);
     nios_write(4'd1, 32'd1);
    nios_write(4'd2, 32'd50);
    nios_write(4'd3, 32'd1);
    nios_write(4'd4, 32'd50);
    nios_write(4'd5, 32'd1);
    nios_write(4'd6, 32'd50);
    nios_write(4'd7, 32'd1);
    nios_write(4'd8, 32'd50);
    nios_write(4'd9, 32'd1);
    nios_write(4'd10, 32'd50);
    nios_write(4'd11, 32'd1);
    nios_write(4'd12, 32'd50);
    nios_write(4'd13, 32'd1);
    nios_write(4'd14, 32'd50);
    nios_write(4'd15, 32'd1);


    #1000;
    nios_read(4'd0);
     nios_read(4'd1);
     nios_read(4'd2);

    // nios_write(32'h80);
    #1000;
   $finish;

end

always #20 avs_s6_clk <= ~avs_s6_clk;

task nios_read;
input [31:0] addr;
begin

   @(posedge avs_s6_clk ) ;
   avs_s6_address <= addr;
   avs_s6_writedata <= 32'h0;
   avs_s6_read <= 1'b1;
   avs_s6_write <= 1'b0;
```

```
    avs_s6_chipselect <= 1'b1;

    @(posedge avs_s6_clk ) ;
    // address <= 0;
    avs_s6_writedata <= 32'h0;
    avs_s6_read <= 1'b0;
    avs_s6_chipselect <= 1'b0;


    @(posedge avs_s6_clk ) ;

end
endtask

task nios_write;
input [3:0] addr;
input [31:0] data;
begin

    @(posedge avs_s6_clk ) ;
    avs_s6_address <= addr;
    avs_s6_writedata <= data;
    avs_s6_read <= 1'b0;
    avs_s6_write <= 1'b1;
    avs_s6_chipselect <= 1'b1;

    @(posedge avs_s6_clk ) ;
    avs_s6_address <= 0;
    avs_s6_writedata <= data;
    avs_s6_read <= 1'b0;
    avs_s6_write <= 1'b0;
    avs_s6_chipselect <= 1'b0;
    @(posedge avs_s6_clk ) ;


end
endtask

endmodule
```

8.2 Software

io.h

```
#ifndef __IO_H__
#define __IO_H__

/*****************************************************
**********************
*
*
* License Agreement
*
*
*
* Copyright (c) 2003 Altera Corporation, San Jose,
California, USA.           *
* All rights reserved.
*
*
*
* Permission is hereby granted, free of charge, to any
person obtaining a     *
* copy of this software and associated documentation
files (the "Software"),  *
* to deal in the Software without restriction, including
without limitation   *
* the rights to use, copy, modify, merge, publish,
distribute, sublicense,    *
* and/or sell copies of the Software, and to permit
persons to whom the       *
* Software is furnished to do so, subject to the
following conditions:        *
*
*
* The above copyright notice and this permission notice
shall be included in  *
* all copies or substantial portions of the Software.
*
*
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
```

```
ANY KIND, EXPRESS OR  *
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY,    *
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE *
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER      *
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING     *
* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
USE OR OTHER          *
* DEALINGS IN THE SOFTWARE.
*
*
*
* This agreement shall be governed in all respects by
the laws of the State   *
* of California and by the laws of the United States of
America.              *
*
*
* Altera does not recommend, suggest or require that
this reference design    *
* file be used in conjunction or combination with any
other product.          *
********************************************************
**********************/

/* IO Header file for Nios II Toolchain */

#include "alt_types.h"
#ifdef __cplusplus
extern "C"
{
#endif /* __cplusplus */

//#ifndef SYSTEM_BUS_WIDTH
//#error SYSTEM_BUS_WIDTH undefined
//#endif
```

```
/* Dynamic bus access functions */

#define __IO_CALC_ADDRESS_DYNAMIC(BASE, OFFSET) \
  ((void *)(((alt_u8*)BASE) + (OFFSET)))

#define IORD_32DIRECT(BASE, OFFSET) \
  __builtin_ldwio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)))
#define IORD_16DIRECT(BASE, OFFSET) \
  __builtin_ldhuio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)))
#define IORD_8DIRECT(BASE, OFFSET) \
  __builtin_ldbuio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)))

#define IOWR_32DIRECT(BASE, OFFSET, DATA) \
  __builtin_stwio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)), (DATA))
#define IOWR_16DIRECT(BASE, OFFSET, DATA) \
  __builtin_sthio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)), (DATA))
#define IOWR_8DIRECT(BASE, OFFSET, DATA) \
  __builtin_stbio (__IO_CALC_ADDRESS_DYNAMIC ((BASE),
(OFFSET)), (DATA))

/* Native bus access functions */

#define __IO_CALC_ADDRESS_NATIVE(BASE, REGNUM) \
  ((void *)(((alt_u8*)BASE) + ((REGNUM) *
(SYSTEM_BUS_WIDTH/8))))

#define IORD(BASE, REGNUM) \
  __builtin_ldwio (__IO_CALC_ADDRESS_NATIVE ((BASE),
(REGNUM)))
#define IOWR(BASE, REGNUM, DATA) \
  __builtin_stwio (__IO_CALC_ADDRESS_NATIVE ((BASE),
(REGNUM)), (DATA))

#ifdef __cplusplus
}
```

```
#endif
```

#endif /* __IO_H__ */

jddctmgr.c

```
/*
 * jddctmgr.c
 *
 * Copyright (C) 1994-1996, Thomas G. Lane.
 * This file is part of the Independent JPEG Group's
software.
 * For conditions of distribution and use, see the
accompanying
README file.
 *
 * This file contains the inverse-DCT management logic.
 * This code selects a particular IDCT implementation to
be used,
 * and it performs related housekeeping chores.  No code
in this
file
 * is executed per IDCT step, only during output pass
setup.
 *
 * Note that the IDCT routines are responsible for
performing
coefficient
 * dequantization as well as the IDCT proper.  This
module sets up
the
 * dequantization multiplier table needed by the IDCT
routine.
 */

#define JPEG_INTERNALS
#include "jinclude.h"
#include "jpeglib.h"
#include "jdct.h"                /* Private declarations
for DCT subsystem */
```

```
/*
 * The decompressor input side (jdinput.c) saves away
the
appropriate
 * quantization table for each component at the start of
the first
scan
 * involving that component.  (This is necessary in
order to
correctly
 * decode files that reuse Q-table slots.)
 * When we are ready to make an output pass, the saved
Q-table is
converted
 * to a multiplier table that will actually be used by
the IDCT
routine.
 * The multiplier table contents are IDCT-method-
dependent.  To
support
 * application changes in IDCT method between scans, we
can remake
the
 * multiplier tables if necessary.
 * In buffered-image mode, the first output pass may
occur before
any data
 * has been seen for some components, and thus before
their Q-tables
have
 * been saved away.  To handle this case, multiplier
tables are
preset
 * to zeroes; the result of the IDCT will be a neutral
gray level.
 */


/* Private subobject for this module */
```

```
typedef struct {
  struct jpeg_inverse_dct pub;        /* public fields
*/

  /* This array contains the IDCT method code that each
multiplier
table
   * is currently set up for, or -1 if it's not yet set
up.
   * The actual multiplier tables are pointed to by
dct_table in the
   * per-component comp_info structures.
   */
  int cur_method[MAX_COMPONENTS];
} my_idct_controller;

typedef my_idct_controller * my_idct_ptr;


/* Allocated multiplier tables: big enough for any
supported variant
*/

typedef union {
  ISLOW_MULT_TYPE islow_array[DCTSIZE2];
#ifdef DCT_IFAST_SUPPORTED
  IFAST_MULT_TYPE ifast_array[DCTSIZE2];
#endif
#ifdef DCT_FLOAT_SUPPORTED
  FLOAT_MULT_TYPE float_array[DCTSIZE2];
#endif
} multiplier_table;


/* The current scaled-IDCT routines require ISLOW-style
multiplier
tables,
 * so be sure to compile that code if either ISLOW or
SCALING is
```

```
requested.
 */
#ifdef DCT_ISLOW_SUPPORTED
#define PROVIDE_ISLOW_TABLES
#else
#ifdef IDCT_SCALING_SUPPORTED
#define PROVIDE_ISLOW_TABLES
#endif
#endif


/*
 * Prepare for an output pass.
 * Here we select the proper IDCT routine for each
component and
build
 * a matching multiplier table.
 */

METHODDEF(void)
start_pass (j_decompress_ptr cinfo)
{
  my_idct_ptr idct = (my_idct_ptr) cinfo->idct;
  int ci, i;
  jpeg_component_info *compptr;
  int method = 0;
  inverse_DCT_method_ptr method_ptr = NULL;
  JQUANT_TBL * qtbl;

  for (ci = 0, compptr = cinfo->comp_info; ci <
cinfo->num_components;
       ci++, compptr++) {
    /* Select the proper IDCT routine for this
component's scaling
*/
    switch (compptr->DCT_scaled_size) {
#ifdef IDCT_SCALING_SUPPORTED
    case 1:
      method_ptr = jpeg_idct_1x1;
      method = JDCT_ISLOW;        /* jidctred uses
```

```
islow-style table */
      break;
    case 2:
      method_ptr = jpeg_idct_2x2;
      method = JDCT_ISLOW;        /* jidctred uses
islow-style table */
      break;
    case 4:
      method_ptr = jpeg_idct_4x4;
      method = JDCT_ISLOW;        /* jidctred uses
islow-style table */
      break;
#endif
    case DCTSIZE:
      method_ptr = jpeg_idct_islow;
      method = JDCT_ISLOW;
      break;

      switch (cinfo->dct_method) {
#ifdef DCT_ISLOW_SUPPORTED
      case JDCT_ISLOW:
        method_ptr = jpeg_idct_islow;
        method = JDCT_ISLOW;
        break;
#endif
#ifdef DCT_IFAST_SUPPORTED
      case JDCT_IFAST:
        method_ptr = jpeg_idct_islow;
        method = JDCT_ISLOW;
        break;
#endif
#ifdef DCT_FLOAT_SUPPORTED
      case JDCT_FLOAT:
        method_ptr = jpeg_idct_islow;
        method = JDCT_ISLOW;
        break;
#endif
      default:
        ERREXIT(cinfo, JERR_NOT_COMPILED);
        break;
```

```
      }
      break;
    default:
      ERREXIT1(cinfo, JERR_BAD_DCTSIZE, compptr-
>DCT_scaled_size);
      break;
    }
    idct->pub.inverse_DCT[ci] = method_ptr;
    /* Create multiplier table from quant table.
     * However, we can skip this if the component is
uninteresting
     * or if we already built the table.  Also, if no
quant table
     * has yet been saved for the component, we leave
the
     * multiplier table all-zero; we'll be reading
zeroes from the
     * coefficient controller's buffer anyway.
     */
    if (! compptr->component_needed || idct->cur_method
[ci] ==
method)
      continue;
    qtbl = compptr->quant_table;
    if (qtbl == NULL)                /* happens if no
data yet for component */
      continue;
    idct->cur_method[ci] = method;
    switch (method) {
#ifdef PROVIDE_ISLOW_TABLES
    case JDCT_ISLOW:
      {
        /* For LL&M IDCT method, multipliers are equal
to raw quantization
         * coefficients, but are stored as ints to
ensure access
efficiency.
         */
        ISLOW_MULT_TYPE * ismtbl = (ISLOW_MULT_TYPE *)
compptr->dct_table;
```

```
        for (i = 0; i < DCTSIZE2; i++) {
    ismtbl[i] = (ISLOW_MULT_TYPE) qtbl->quantval[i];
    }
       }
       break;
#endif
#ifdef DCT_IFAST_SUPPORTED
     case JDCT_IFAST:
      {
        /* For LL&M IDCT method, multipliers are equal
to raw quantization
         * coefficients, but are stored as ints to
ensure access
efficiency.
         */
        ISLOW_MULT_TYPE * ismtbl = (ISLOW_MULT_TYPE *)
compptr->dct_table;
        for (i = 0; i < DCTSIZE2; i++) {
    ismtbl[i] = (ISLOW_MULT_TYPE) qtbl->quantval[i];
    }
       }
       break;
#endif
#ifdef DCT_FLOAT_SUPPORTED
    case JDCT_FLOAT:
      {
        /* For LL&M IDCT method, multipliers are equal
to raw quantization
         * coefficients, but are stored as ints to
ensure access
efficiency.
         */
        ISLOW_MULT_TYPE * ismtbl = (ISLOW_MULT_TYPE *)
compptr->dct_table;
        for (i = 0; i < DCTSIZE2; i++) {
    ismtbl[i] = (ISLOW_MULT_TYPE) qtbl->quantval[i];
    }
       }
       break;
#endif
```

```
    default:
      ERREXIT(cinfo, JERR_NOT_COMPILED);
      break;
      }
  }
}



/*
 * Initialize IDCT manager.
 */

GLOBAL(void)
jinit_inverse_dct (j_decompress_ptr cinfo)
{
  my_idct_ptr idct;
  int ci;
  jpeg_component_info *compptr;

  idct = (my_idct_ptr)
    (*cinfo->mem->alloc_small) ((j_common_ptr) cinfo,
JPOOL_IMAGE,
                                SIZEOF
(my_idct_controller));
  cinfo->idct = (struct jpeg_inverse_dct *) idct;
  idct->pub.start_pass = start_pass;

  for (ci = 0, compptr = cinfo->comp_info; ci <
cinfo->num_components;
       ci++, compptr++) {
    /* Allocate and pre-zero a multiplier table for each
component
*/
    compptr->dct_table =
      (*cinfo->mem->alloc_small) ((j_common_ptr) cinfo,
JPOOL_IMAGE,
                                  SIZEOF
(multiplier_table));
    MEMZERO(compptr->dct_table, SIZEOF
```

```
(multiplier_table));
    /* Mark multiplier table not yet set up for any
method */
    idct->cur_method[ci] = -1;
  }
}
```

jidctint.c

```
/*
 * jidctint.c
 *
 * Copyright (C) 1991-1998, Thomas G. Lane.
 * This file is part of the Independent JPEG Group's
software.
 * For conditions of distribution and use, see the
accompanying README file.
 *
 * This file contains a slow-but-accurate integer
implementation of the
 * inverse DCT (Discrete Cosine Transform).  In the IJG
code, this routine
 * must also perform dequantization of the input
coefficients.
 *
 * A 2-D IDCT can be done by 1-D IDCT on each column
followed by 1-D IDCT
 * on each row (or vice versa, but it's more convenient
to emit a row at
 * a time).  Direct algorithms are also available, but
they are much more
 * complex and seem not to be any faster when reduced to
code.
 *
 * This implementation is based on an algorithm
described in
 *   C. Loeffler, A. Ligtenberg and G. Moschytz,
"Practical Fast 1-D DCT
 *   Algorithms with 11 Multiplications", Proc. Int'l.
Conf. on Acoustics,
 *   Speech, and Signal Processing 1989 (ICASSP '89),
```

```
pp. 988-991.
 * The primary algorithm described there uses 11
multiplies and 29 adds.
 * We use their alternate method with 12 multiplies and
32 adds.
 * The advantage of this method is that no data path
contains more than one
 * multiplication; this allows a very simple and
accurate implementation in
 * scaled fixed-point arithmetic, with a minimal number
of shifts.
 */

#define JPEG_INTERNALS
#include "jinclude.h"
#include "jpeglib.h"
#include "jdct.h"      /* Private declarations for DCT
subsystem */

#include <io.h>

#include <stdio.h>

#ifdef DCT_ISLOW_SUPPORTED


/*
 * This module is specialized to the case DCTSIZE = 8.
 */

#if DCTSIZE != 8
  Sorry, this code only copes with 8x8 DCTs. /*
deliberate syntax err */
#endif

#define IOWR_JPEG_DATA(base, offset, data) \
  IOWR_32DIRECT(base, (offset) * 4, data)
#define JPEG_BASE 0x00400000

#define IORD_JPEG_DATA(base, offset) \
```

```
  IORD_32DIRECT(base, (offset) * 4)

/*
 * The poop on this scaling stuff is as follows:
 *
 * Each 1-D IDCT step produces outputs which are a
factor of sqrt(N)
 * larger than the true IDCT outputs.  The final outputs
are therefore
 * a factor of N larger than desired; since N=8 this can
be cured by
 * a simple right shift at the end of the algorithm.
The advantage of
 * this arrangement is that we save two multiplications
per 1-D IDCT,
 * because the y0 and y4 inputs need not be divided by
sqrt(N).
 *
 * We have to do addition and subtraction of the integer
inputs, which
 * is no problem, and multiplication by fractional
constants, which is
 * a problem to do in integer arithmetic.  We multiply
all the constants
 * by CONST_SCALE and convert them to integer constants
(thus retaining
 * CONST_BITS bits of precision in the constants).
After doing a
 * multiplication we have to divide the product by
CONST_SCALE, with proper
 * rounding, to produce the correct output.  This
division can be done
 * cheaply as a right shift of CONST_BITS bits.  We
postpone shifting
 * as long as possible so that partial sums can be added
together with
 * full fractional precision.
 *
 * The outputs of the first pass are scaled up by
PASS1_BITS bits so that
```

```
 * they are represented to better-than-integral
precision.  These outputs
 * require BITS_IN_JSAMPLE + PASS1_BITS + 3 bits; this
fits in a 16-bit word
 * with the recommended scaling.  (To scale up 12-bit
sample data further, an
 * intermediate INT32 array would be needed.)
 *
 * To avoid overflow of the 32-bit intermediate results
in pass 2, we must
 * have BITS_IN_JSAMPLE + CONST_BITS + PASS1_BITS <= 26.
Error analysis
 * shows that the values given below are the most
effective.
 */

#if BITS_IN_JSAMPLE == 8
#define CONST_BITS  13
#define PASS1_BITS  2
#else
#define CONST_BITS  13
#define PASS1_BITS  1       /* lose a little precision
to avoid overflow */
#endif

/* Multiply an INT32 variable by an INT32 constant to
yield an INT32 result.
 * For 8-bit samples with the recommended scaling, all
the variable
 * and constant values involved are no more than 16 bits
wide, so a
 * 16x16->32 bit multiply can be used instead of a full
32x32 multiply.
 * For 12-bit samples, a full 32-bit multiplication will
be needed.
 */

#if BITS_IN_JSAMPLE == 8
#define MULTIPLY(var,const)  MULTIPLY16C16(var,const)
#else
```

```
#define MULTIPLY(var,const)  ((var) * (const))
#endif


/* Dequantize a coefficient by multiplying it by the
multiplier-table
 * entry; produce an int result.  In this module, both
inputs and result
 * are 16 bits or less, so either int or short multiply
will work.
 */

#define DEQUANTIZE(coef,quantval)  (((ISLOW_MULT_TYPE)
(coef)) * (quantval))


/*
 * Perform dequantization and inverse DCT on one block
of coefficients.
 */

GLOBAL(void)
jpeg_idct_islow (j_decompress_ptr cinfo,
jpeg_component_info * compptr,
          JCOEFPTR coef_block,
          JSAMPARRAY output_buf, JDIMENSION output_col)
{
  int counterint;
    int cycleint;
  JCOEFPTR inptr;
  ISLOW_MULT_TYPE * quantptr;
  int * wsptr;
  JSAMPROW outptr;
  JSAMPLE *range_limit = IDCT_range_limit(cinfo);
  int ctr;
  int workspace[DCTSIZE2];  /* buffers data between
passes */
  SHIFT_TEMPS

  /* Pass 1: process columns from input, store into work
```

```
array. */
  /* Note results are scaled up by sqrt(8) compared to a
true IDCT; */
  /* furthermore, we scale the results by 2**PASS1_BITS.
*/

  inptr = coef_block;
  quantptr = (ISLOW_MULT_TYPE *) compptr->dct_table;
  wsptr = workspace;
  for (ctr = DCTSIZE; ctr > 0; ctr--) {
    /* Due to quantization, we will usually find that
many of the input
     * coefficients are zero, especially the AC terms.
We can exploit this
     * by short-circuiting the IDCT calculation for any
column in which all
     * the AC terms are zero.  In that case each output
is equal to the
     * DC coefficient (with scale factor as needed).
     * With typical images and quantization tables, half
or more of the
     * column DCT calculations can be simplified this
way.
     */

    if (inptr[DCTSIZE*1] == 0 && inptr[DCTSIZE*2] == 0
&&
     inptr[DCTSIZE*3] == 0 && inptr[DCTSIZE*4] == 0 &&
     inptr[DCTSIZE*5] == 0 && inptr[DCTSIZE*6] == 0 &&
     inptr[DCTSIZE*7] == 0) {
      /* AC terms all zero */
      int dcval = DEQUANTIZE(inptr[DCTSIZE*0], quantptr
[DCTSIZE*0]) << PASS1_BITS;

      wsptr[DCTSIZE*0] = dcval;
      wsptr[DCTSIZE*1] = dcval;
      wsptr[DCTSIZE*2] = dcval;
      wsptr[DCTSIZE*3] = dcval;
      wsptr[DCTSIZE*4] = dcval;
      wsptr[DCTSIZE*5] = dcval;
```

```
      wsptr[DCTSIZE*6] = dcval;
      wsptr[DCTSIZE*7] = dcval;

      inptr++;              /* advance pointers to next
column */
      quantptr++;
      wsptr++;
      continue;
    }

    IOWR_JPEG_DATA(JPEG_BASE,0,inptr[DCTSIZE*0]);
    IOWR_JPEG_DATA(JPEG_BASE,1,quantptr[DCTSIZE*0]);
    IOWR_JPEG_DATA(JPEG_BASE,2,inptr[DCTSIZE*1]);
    IOWR_JPEG_DATA(JPEG_BASE,3,quantptr[DCTSIZE*1]);
    IOWR_JPEG_DATA(JPEG_BASE,4,inptr[DCTSIZE*2]);
    IOWR_JPEG_DATA(JPEG_BASE,5,quantptr[DCTSIZE*2]);
    IOWR_JPEG_DATA(JPEG_BASE,6,inptr[DCTSIZE*3]);
    IOWR_JPEG_DATA(JPEG_BASE,7,quantptr[DCTSIZE*3]);
    IOWR_JPEG_DATA(JPEG_BASE,8,inptr[DCTSIZE*4]);
    IOWR_JPEG_DATA(JPEG_BASE,9,quantptr[DCTSIZE*4]);
    IOWR_JPEG_DATA(JPEG_BASE,10,inptr[DCTSIZE*5]);
    IOWR_JPEG_DATA(JPEG_BASE,11,quantptr[DCTSIZE*5]);
    IOWR_JPEG_DATA(JPEG_BASE,12,inptr[DCTSIZE*6]);
    IOWR_JPEG_DATA(JPEG_BASE,13,quantptr[DCTSIZE*6]);
    IOWR_JPEG_DATA(JPEG_BASE,14,inptr[DCTSIZE*7]);
    IOWR_JPEG_DATA(JPEG_BASE,15,quantptr[DCTSIZE*7]);

    /*
    for(counterint = 0; counterint < 10; counterint++) {
      cycleint = cycleint++;
    }
    */

    wsptr[DCTSIZE*0]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,0), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*1]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,1), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*2]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,2), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*3]=(int) DESCALE(IORD_JPEG_DATA
```

```
(JPEG_BASE,3), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*4]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,4), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*5]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,5), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*6]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,6), CONST_BITS-PASS1_BITS);
    wsptr[DCTSIZE*7]=(int) DESCALE(IORD_JPEG_DATA
(JPEG_BASE,7), CONST_BITS-PASS1_BITS);

    inptr++;           /* advance pointers to next
column */
    quantptr++;
    wsptr++;
  }

  /* Pass 2: process rows from work array, store into
output array. */
  /* Note that we must descale the results by a factor
of 8 == 2**3, */
  /* and also undo the PASS1_BITS scaling. */

  IOWR_JPEG_DATA(JPEG_BASE,1,1);
  IOWR_JPEG_DATA(JPEG_BASE,3,1);
  IOWR_JPEG_DATA(JPEG_BASE,5,1);
  IOWR_JPEG_DATA(JPEG_BASE,7,1);
  IOWR_JPEG_DATA(JPEG_BASE,9,1);
  IOWR_JPEG_DATA(JPEG_BASE,11,1);
  IOWR_JPEG_DATA(JPEG_BASE,13,1);
  IOWR_JPEG_DATA(JPEG_BASE,15,1);

  wsptr = workspace;
  for (ctr = 0; ctr < DCTSIZE; ctr++) {
    outptr = output_buf[ctr] + output_col;
    /* Rows of zeroes can be exploited in the same way
as we did with columns.
     * However, the column calculation has created many
nonzero AC terms, so
     * the simplification applies less often (typically
5% to 10% of the time).
```

```
     * On machines with very fast multiplication, it's
possible that the
     * test takes more time than it's worth.  In that
case this section
     * may be commented out.
     */

#ifndef NO_ZERO_ROW_TEST
    if (wsptr[1] == 0 && wsptr[2] == 0 && wsptr[3] == 0
&& wsptr[4] == 0 &&
     wsptr[5] == 0 && wsptr[6] == 0 && wsptr[7] == 0) {
      /* AC terms all zero */
      JSAMPLE dcval = range_limit[(int) DESCALE((INT32)
wsptr[0], PASS1_BITS+3)
                     & RANGE_MASK];

      outptr[0] = dcval;
      outptr[1] = dcval;
      outptr[2] = dcval;
      outptr[3] = dcval;
      outptr[4] = dcval;
      outptr[5] = dcval;
      outptr[6] = dcval;
      outptr[7] = dcval;

      wsptr += DCTSIZE;          /* advance pointer to
next row */
      continue;
    }
#endif

    /* Even part: reverse the even part of the forward
DCT. */
    /* The rotator is sqrt(2)*c(-6). */
    IOWR_JPEG_DATA(JPEG_BASE,0,wsptr[0]);
    IOWR_JPEG_DATA(JPEG_BASE,2,wsptr[1]);
    IOWR_JPEG_DATA(JPEG_BASE,4,wsptr[2]);
    IOWR_JPEG_DATA(JPEG_BASE,6,wsptr[3]);
    IOWR_JPEG_DATA(JPEG_BASE,8,wsptr[4]);
    IOWR_JPEG_DATA(JPEG_BASE,10,wsptr[5]);
```

```
    IOWR_JPEG_DATA(JPEG_BASE,12,wsptr[6]);
    IOWR_JPEG_DATA(JPEG_BASE,14,wsptr[7]);

    /*
    for(counterint = 0; counterint < 10; counterint++) {
      cycleint = cycleint++;
    }
    */

    outptr[0]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,0),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[1]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,1),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[2]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,2),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[3]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,3),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[4]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,4),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[5]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,5),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[6]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,6),
                         CONST_BITS+PASS1_BITS+3)&
RANGE_MASK];
    outptr[7]= range_limit[(int)DESCALE(IORD_JPEG_DATA
(JPEG_BASE,7),
                         CONST_BITS+PASS1_BITS+3)&
```

```
RANGE_MASK];

    wsptr += DCTSIZE;       /* advance pointer to next
row */
  }
}
```

#endif /* DCT_ISLOW_SUPPORTED */

alt_types.h

```
#ifndef __ALT_TYPES_H__
#define __ALT_TYPES_H__

/*******************************************************
**********************
*
*
* License Agreement
*
*
*
* Copyright (c) 2003 Altera Corporation, San Jose,
California, USA.            *
* All rights reserved.
*
*
*
* Permission is hereby granted, free of charge, to any
person obtaining a     *
* copy of this software and associated documentation
files (the "Software"),  *
* to deal in the Software without restriction, including
without limitation   *
* the rights to use, copy, modify, merge, publish,
distribute, sublicense,    *
* and/or sell copies of the Software, and to permit
persons to whom the       *
* Software is furnished to do so, subject to the
following conditions:        *
*
```

```
*
* The above copyright notice and this permission notice
shall be included in  *
* all copies or substantial portions of the Software.
*
*
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR  *
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY,    *
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE *
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,
DAMAGES OR OTHER      *
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
OTHERWISE, ARISING     *
* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE
USE OR OTHER         *
* DEALINGS IN THE SOFTWARE.
*
*
*
* This agreement shall be governed in all respects by
the laws of the State   *
* of California and by the laws of the United States of
America.              *
*
*
* Altera does not recommend, suggest or require that
this reference design    *
* file be used in conjunction or combination with any
other product.          *
********************************************************
*********************/

/*
 * Don't declare these typedefs if this file is included
by assembly source.
 */
```

```
#ifndef ALT_ASM_SRC
typedef signed char  alt_8;
typedef unsigned char  alt_u8;
typedef signed short alt_16;
typedef unsigned short alt_u16;
typedef signed long alt_32;
typedef unsigned long alt_u32;
typedef long long alt_64;
typedef unsigned long long alt_u64;
#endif

#define ALT_INLINE        __inline__
#define ALT_ALWAYS_INLINE __attribute__
((always_inline))
#define ALT_WEAK          __attribute__((weak))
```

#endif /* __ALT_TYPES_H__ */

picturebrowser.c

```
#include <stdio.h>
#include <errno.h> //needed for errnum
#include <string.h>//needed for strerror
#include <shimix_pio_button.h>

#include <fcntl.h>
#include <dirent.h>
#include <sys/stat.h>
#include <sys/types.h>

#define BUTTON_DEVICE "/dev/shimix"


static char* pics [100];
//pics = (char *) malloc(100);
static int count = 0;

void get_file_info(char *name, char *path){
  struct stat file_stats;
  if(stat(name, &file_stats) == -1){
    printf("\nFailed to access %s\n", name);
```

```
    return;
  }
  printf("\nFile found!\n");
  printf(  "        File name: %s\n", name);
  printf(  "    Relative path: %s\n", path);
  printf(  "  Protection mode: %d\n",
file_stats.st_mode);
  printf(  " User id of owner: %d\n",
file_stats.st_uid);
  printf(  "Group id of owner: %d\n",
file_stats.st_gid);
  printf(  "    Size in bytes: %d\n",
file_stats.st_size);
  if(count < 100){
    if(file_stats.st_size <= 20000){
      pics[count]= strdup(name);
      printf("Added pic[%d] = %s\n",count, name);
      count ++;
    }
    else{
      printf("File is too big. Skipping file...\n");
    }

  }
  if((file_stats.st_mode & S_IFMT) == S_IFDIR){
    printf("The file is a directory.\n\n");
  }
}

void str_tolower(char *s)
 {
     while(*s)
     {
         *s=tolower(*s);
         s++;
     }
 }


int StartSearch(char *name, char *path, int
```

```
found_signal){

  DIR *dfd;
  struct dirent *dp;
  struct stat file_stat;

  if((dfd = opendir(path)) == NULL){
    printf("Directory [%s], failed to open.", path);
    return;
  }
  while (( dp = readdir(dfd)) != NULL){
    if(strcmp(dp->d_name, ".") == 0 || strcmp(dp-
>d_name, "..") == 0){
      continue; /*skipping itself and parent
directories*/
    }
    char slash[2] = "";

    char *path_cpy = (char *) malloc(sizeof(char)*strlen
(path)+strlen(dp->d_name)+2);
    strcpy(path_cpy, path);

    char *d_nameCopy = (char *) malloc(sizeof(char)
*strlen(dp->d_name)+2);
    strcpy(d_nameCopy, dp->d_name);

    char *combined = strcat(path_cpy, slash);
    char *full_name = strcat(combined, dp->d_name);

    if(stat(full_name, &file_stat) == -1){
      printf("\nFailed to access: %s\nSkipping...\n ",
full_name);
      continue; /*skip bad file or directory*/
    }
    if((file_stat.st_mode & S_IFMT) == S_IFDIR){
      StartSearch(name, full_name, found_signal); /
*recursive search*/
    }

    str_tolower(d_nameCopy);
```

```
    if(strstr(d_nameCopy, name)){
      get_file_info(full_name, path);
      found_signal = 1;
      //FILE FOUND!!!
    }

    free(path_cpy);
    free(d_nameCopy);
  }
  closedir(dfd);
  return found_signal;
}


int main(int argc, char *argv[])
{
  FILE * fButton;          //handle to button device
  char buf[1]; //read buffer to store value of button
pushed when read from fButton
  int numBytesRead = 0;
  char keyInput = 0;
  unsigned int ioctlCmd;
  printf("Launching button reader v1.3\n");

/* Establishing the root directory */
  char* initial_path = "home/";
  int found_signal = 0;
  char * jpg = ".jpeg";
  int find = StartSearch(jpg, initial_path,
found_signal);
  jpg= ".jpg";
  find = StartSearch(jpg, initial_path, found_signal);

  char modprobe[28] =   "modprobe\tshimix_pio_button";
  char makeDevice[29] = "mknod\t/dev/shimix\tc\t63\t0";
  //printf("mknod is ran\n\n");
  int x = system(modprobe);
  x = system(makeDevice);
```

```
  fButton = fopen(BUTTON_DEVICE, "r"); //open button
device for read-only
  if (fButton==NULL) {
    printf("Error opening up %s\n", BUTTON_DEVICE);
    return -1;
  }

  ioctl(fileno(fButton), SBLD_IOCT_LEDBUTTONNUM, 1); //
enable displaying button pushed
  ioctl(fileno(fButton), SBLD_IOCT_LEDCOUNTER, 1);   //
enable button counters

  printf("Entering main loop\n\n");

  char command1[8] = "nano-X&";
  char killall[20] = "killall\t-9\tnano-X";
  // position of our cursor through the pics
  int pos = 0;
  //int j;
  char andPersand[4] = "\t&";
  char resize[6] = "\t12&";
  char nxview[9] = "nxview\t";
  char *command2;
  short first_time = 0;


  while (keyInput!='q' && keyInput!='Q') {
    printf("awaiting button: ");
    if ((numBytesRead=read(fileno(fButton), buf, sizeof
(char), 0))<= 0) { //if no bytes read or error
      printf("Error reading %s\n", BUTTON_DEVICE);
    }
    else {
      x = system(killall);
      //j = sleep(1);
      if(first_time == 0){
     printf("executing command nano-X....\n");
     x = system(command1);
     //j = sleep(1);
     command2 = (char *) malloc(sizeof(char)*strlen
```

```
(nxview));
     strcpy(command2, nxview);
     printf("%d %u\n", buf[0], buf[0]);
     strcat(command2, pics[pos]);
     strcat(command2, andPersand);
     printf("executing %s ....\n",command2);
     x = system(command2);
     //j = sleep(1);
     free(command2);
     first_time = 1;
      }
      else{
     if (buf[0] == 2) {
       printf("executing command nano-X....\n");
       x = system(command1);
       //j = sleep(1);
       command2 = (char *) malloc(sizeof(char)*strlen
(nxview));
       strcpy(command2, nxview);
       printf("%d %u\n", buf[0], buf[0]);
       if (pos == count-1){
         pos = 0;
       }
       else{
         pos = pos + 1;
       }
       strcat(command2, pics[pos]);
       strcat(command2, andPersand);
       printf("executing %s ....\n",command2);
       x = system(command2);
       //j = sleep(1);
       free(command2);
     }
     else{
       if (buf[0] == 4) {
         printf("executing command nano-X....\n");
         x = system(command1);
         //j = sleep(1);
         command2 = (char *) malloc(sizeof(char)*strlen
(nxview));
```

```
          strcpy(command2, nxview);
          printf("%d %u\n", buf[0], buf[0]);
          if (pos <= 0){
            pos = count-1;
          }
          else{
            pos  = pos - 1;
          }
          strcat(command2, pics[pos]);
          strcat(command2, andPersand);
          printf("executing %s ....\n",command2);
          x = system(command2);
          //j = sleep(1);
          free(command2);
        }
        else{
          if (buf[0] == 8){
            printf("executing command nano-X....\n");
            x = system(command1);
            //j = sleep(1);
            command2 = (char *) malloc(sizeof(char)
*strlen(nxview));
            strcpy(command2, nxview);
            printf("%d %u\n", buf[0], buf[0]);
            strcat(command2, pics[pos]);
            strcat(command2, resize);
            printf("executing %s ....\n",command2);
            x = system(command2);
            //j = sleep(1);
            free(command2);
          }
        }
      }
       }
     }
       //printf("Press q to quit or any other key to
continue reading buttons\n");
       //keyInput = getchar();
   }
   int i;
```

```
  for(i = 0; i < count; i++){
    free(pics[i]);
  }

  printf("Button Reader terminating\n");
}
```

rc

```
hostname uClinux
mount -t proc proc /proc
mount -t sysfs sysfs /sys
mount -t usbfs none /proc/bus/usb
mount -t vfat dev/sda1 mnt  # add this line
mkdir /var/tmp
mkdir /var/log
mkdir /var/run
mkdir /var/lock
mkdir /var/empty
ifconfig lo 127.0.0.1
route add -net 127.0.0.0 netmask 255.0.0.0 lo
# dhcpcd -p -a eth0 &
cat /etc/motd
```

nxview.c

```
/*
 * Copyright (c) 2000, 2001 Greg Haerr
<greg@censoft.com>
 *
 * nxview - Nano-X image viewer
 *
 * Autorecognizes and displays BMP, GIF, JPEG, PNG and
XPM files
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MWINCLUDECOLORS
#include "nano-X.h"
#include <errno.h> //needed for errnum
#include <shimix_pio_button.h>
```

```
#include <fcntl.h>
#include <dirent.h>
#include <stat.h>
#include <types.h>

//#define BUTTON_DEVICE "/dev/shimix"
//static int count = 0;
/*

static char* pics [100];
static int count = 0;

void get_file_info(char *name, char *path){
  struct stat file_stats;
  if(stat(name, &file_stats) == -1){
    printf("\nFailed to access %s\n", name);
    return;
  }
  printf("\nFile found!\n");
  printf(  "        File name: %s\n", name);
  printf(  "    Relative path: %s\n", path);
  printf(  "  Protection mode: %d\n",
file_stats.st_mode);
  printf(  " User id of owner: %d\n",
file_stats.st_uid);
  printf(  "Group id of owner: %d\n",
file_stats.st_gid);
  printf(  "    Size in bytes: %d\n",
file_stats.st_size);
  if(count < 100){
    if(file_stats.st_size <= 20000){
      pics[count]= strdup(name);
      printf("Added pic[%d] = %s\n",count, name);
      count ++;
    }
    else{
      printf("File is too big. Skipping file...\n");
    }

  }
```

```
  if((file_stats.st_mode & S_IFMT) == S_IFDIR){
    printf("The file is a directory.\n\n");
  }
}
*/
/*
int StartSearch(char *name, char *path, int
found_signal){

  DIR *dfd;
  struct dirent *dp;
  struct stat file_stat;

  if((dfd = opendir(path)) == NULL){
    printf("Directory [%s], failed to open.", path);
    return;
  }
  while (( dp = readdir(dfd)) != NULL){
    if(strcmp(dp->d_name, ".") == 0 || strcmp(dp-
>d_name, "..") == 0){
      continue; //skipping itself and parent
directories//
    }
    char slash[2] = "";

    char *path_cpy = (char *) malloc(sizeof(char)*strlen
(path)+strlen(dp->d_name)+2);
    strcpy(path_cpy, path);

    char *d_nameCopy = (char *) malloc(sizeof(char)
*strlen(dp->d_name)+2);
    strcpy(d_nameCopy, dp->d_name);

    char *combined = strcat(path_cpy, slash);
    char *full_name = strcat(combined, dp->d_name);

    if(stat(full_name, &file_stat) == -1){
      printf("\nFailed to access: %s\nSkipping...\n ",
full_name);
      continue; //skip bad file or directory//
```

```
    }
    if((file_stat.st_mode & S_IFMT) == S_IFDIR){
      StartSearch(name, full_name, found_signal); //
recursive search//
    }

    str_tolower(d_nameCopy);

    if(strstr(d_nameCopy, name)){
      get_file_info(full_name, path);
      found_signal = 1;
      //FILE FOUND!!!
    }

    free(path_cpy);
    free(d_nameCopy);
  }
  closedir(dfd);
  return found_signal;
}
*/

int main(int argc,char **argv)
{
  GR_IMAGE_ID image_id;
  GR_WINDOW_ID    window_id;
  GR_GC_ID    gc_id;
  GR_SIZE         w = -1;
  GR_SIZE         h = -1;
  GR_EVENT    event;
  GR_SCREEN_INFO  sinfo;
  GR_IMAGE_INFO   info;
  char        title[256];
  int pos = 0;
  int count = 10;
  /*
  FILE * fButton;          //handle to button device
  char buf[1]; //read buffer to store value of button
pushed when read from fButton
  int numBytesRead = 0;
```

```
  char keyInput = 0;
  unsigned int ioctlCmd;
  int pos = 0;
  int count = 10;
  printf("Launching button reader v1.3\n");

  fButton = fopen(BUTTON_DEVICE, "r"); //open button
device for read-only
  if (fButton==NULL) {
    printf("Error opening up %s\n", BUTTON_DEVICE);
    return -1;
  }

  ioctl(fileno(fButton), SBLD_IOCT_LEDBUTTONNUM, 1); //
enable displaying button pushed
  ioctl(fileno(fButton), SBLD_IOCT_LEDCOUNTER, 1);   //
enable button counters
  */


  /* Establishing the root directory */
  /*  char* initial_path = "/home/";
  int found_signal = 0;
  char * jpg = ".jpeg";
  int find = StartSearch(jpg, initial_path,
found_signal);
  jpg= ".jpg";
  find = StartSearch(jpg, initial_path, found_signal);
  int pos = 0;
  int next_image = 0;
  int first_time = 0;*/
  //while(1){


  if (argc < 2) {
    printf("Usage: nxview <image file> [stretch]\n");
    exit(1);
  }
```

```
  if (GrOpen() < 0) {
    fprintf(stderr, "cannot open graphics\n");
    exit(1);
  }



  if (!(image_id = GrLoadImageFromFile(argv[1], 0))) {
    //if (!(image_id = GrLoadImageFromFile(pics[pos],
0))) {
    fprintf(stderr, "Can't load image file: %s\n", argv
[1]);
    exit(1);
  }
    /*
    if(first_time == 0){
      GrGetImageInfo(image_id, &info);
      w = info.width;
      h = info.height;
    }
    first_time = 1;
    */

  if(argc > 2) {
    // stretch to half screen size
    GrGetScreenInfo(&sinfo);
    w = sinfo.cols/2;
    h = sinfo.rows/2;
  }
  else {
    GrGetImageInfo(image_id, &info);
    w = info.width;
    h = info.height;
  }


    sprintf(title, "nxview %s", argv[1]);
    window_id = GrNewWindowEx(GR_WM_PROPS_APPWINDOW,
title,
```

```
                    GR_ROOT_WINDOW_ID, (640 - w)/2, (480
- h)/2, w, h, BLACK);

    GrSelectEvents(window_id,
            GR_EVENT_MASK_CLOSE_REQ|
GR_EVENT_MASK_EXPOSURE);

    GrMapWindow(window_id);

    gc_id = GrNewGC();

    while (1) {


      GrGetNextEvent(&event);
      // if(next_image == 1){
      //event.type = GR_EVENT_TYPE_CLOSE_REQ;
      //}
      switch(event.type) {
      case GR_EVENT_TYPE_CLOSE_REQ:
     GrDestroyWindow(window_id);
     GrDestroyGC(gc_id);
     GrFreeImage(image_id);
     GrClose();
     //break;
     exit(0);
     /* no return*/
      case GR_EVENT_TYPE_EXPOSURE:
     GrDrawImageToFit(window_id, gc_id, 0,0, w,h,
image_id);
     break;
      }

      /*
      printf("awaiting button: ");
      if ((numBytesRead=read(fileno(fButton), buf,
sizeof(char), 0))<=0) { //if no bytes read or error
     printf("Error reading %s\n", BUTTON_DEVICE);
      }
      else{
```

```
 next_image = 1;
 if (buf[0] == 2) {
   printf("%d %u\n", buf[0], buf[0]);
   //GrGetImageInfo(image_id, &info);
   //w = info.width;
   //h = info.height;
   if (pos == count-1){
     pos = 0;
   }
   else{
     pos = pos + 1;
   }
 }
 else{
   if (buf[0] == 4) {
     printf("%d %u\n", buf[0], buf[0]);
     //GrGetImageInfo(image_id, &info);
     //w = info.width;
     //h = info.height;
     if (pos <= 0){
       pos = count-1;
     }
     else{
       pos  = pos - 1;
     }
   }
   else{
     if (buf[0] == 8){
       printf("%d %u\n", buf[0], buf[0]);
       //stretch to half screen size//
       //GrGetScreenInfo(&sinfo);
       //w = sinfo.cols/2;
       //h = sinfo.rows/2;
       }
       }
       }
       }*/
}
//next_image = 0;
return 0;
```

```
}

//  int i;
//  for(i = 0; i < count; i++){
//    free(pics[i]);
//  }

//return 0;
//}
```

Chapter 9: License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder

saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein.

You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this

License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS