ECE3552: Final Project
6 May 2009

Florida Institute of Technology
Department of Electrical and Computer Engineering
150 West University Boulevard
Melbourne, Florida 32901
www.fit.edu • (321) 674-8060

Thomas Misilo

Submitted to:
Veton Këpuska
Department of Electrical and Computer Engineering
150 West University Boulevard
Melbourne, Florida 32901
www.fit.edu • (321) 674-8060
# Table of Contents

List of Figures \hspace{3.5cm} 2

1 Problem Statement \hspace{3.5cm} 3

2 Solutions to Problems \hspace{3.5cm} 3

\hspace{1cm} 2.1 Matlab \hspace{2.5cm} 3

\hspace{1cm} 2.2 Blackfin \hspace{2.5cm} 3

3 Project Results \hspace{3.5cm} 5

4 Concluding Remarks \hspace{3.5cm} 7

5 References \hspace{3.5cm} 7

A Blackfin C Code \hspace{3.5cm} 8

\hspace{1cm} A.1 main.c Source Code \hspace{2.5cm} 8

\hspace{1cm} A.2 BF533 Flags.h Source Code \hspace{2.5cm} 10

\hspace{1cm} A.3 Initialization.c Source Code \hspace{2.5cm} 11

\hspace{1cm} A.4 ISRs.c Source Code \hspace{2.5cm} 14

B Matlab Code \hspace{3.5cm} 15

\hspace{1cm} B.1 imageoverlay.m Source Code \hspace{2.5cm} 15
# List of Figures

1. Image of Coyote for Matlab ........................................... 4
2. Image of overlay for Matlab ............................................ 4
3. Image of Coyote for the Blackfin .................................... 4
4. Image of overlay for the Blackfin .................................... 4
5. Output from Blackfin ..................................................... 5
6. Output from Matlab ....................................................... 6
1 Problem Statement

The Microcomputer Systems II project was a culmination of the semesters labs and classes. The project was to incorporate an idea I had for a project of creating an image overlay. The idea came from television news reports, where they add the reporters name on top of the live feed.

2 Solutions to Problems

The solutions that first came to mind, was doing it the project with a live video stream. However, due to my knowledge of how to make the boards do the live video streaming, I went to a still image, that I would overlay. The project was first done in Matlab, and then coded over to Blackfin processor.

2.1 Matlab

For the Matlab code I realized after a lot of thinking and reading, that this project could be done with a couple of for loops. I start off by reading in the two images. I then start by subtract 255 from all the pixel values from the overlay, so that the image would not have any pure white pixels. I then use for loops, to copy the original image to the output image, and then copy the overlay into that image, by subtracting the values from one another. The Matlab code is available in Appendix B.1. The original images used in the Matlab can be seen in Figure 1 and Figure 2.

2.2 Blackfin

After creating the Matlab code I went to coding it in VisualDSP++. The code for this is located in main.c which can be seen in Appendix A.1. I start by first inverting the overlay image, which is done by subtract 255 from every value in each pixel. I then initialize the imgOutput variable to zero, so that no problems should occur when copying the imgInput to imgOutput. Figure 3 is a copy of the original image that I used for the Blackfin, and Figure 4 is the overlay image used for the Blackfin. The reason the images are smaller than those in Figure’s 1 and 2, is because the time of transferring the images back and forth over the USB connection, was taking forever for the large bitmap images.
Figure 1: Image of Coyote for Matlab (495px x 353px)

Wile E. Coyote

Figure 2: Image of overlay for Matlab (495px x 75px)

Wile E. Coyote

Figure 3: Image of Coyote for the Blackfin (200px x 143px)

Wile E. Coyote

Figure 4: Image of overlay for the Blackfin (200px x 30px)
3 Project Results

The overall result is that it works for the single image, as seen in Figure 5 which is the output from the Blackfin. However, it is not as detailed as Figure 6. I could not figure this out, as I did try with the larger images on the Blackfin but it seemed to only make the end result worse, so that is why I stayed with the smaller pictures for the report and project. I would have liked to make the image look more like the Matlab output, however everything I tried could not get it to work, as the Matlab and C code are both similar and close to identical in code and structure. I would also have liked to have implemented this onto a TV screen and do live video streaming. However, I ran out of time to implement this currently. I would however like to see this done, with ability to press different buttons and have different overlays appear on the screen, as well as being able to clear it, and have it return to the original image.

Figure 5: Output from Blackfin (200px x 143px)
Figure 6: Output from Matlab (495px x 353px)
4 Concluding Remarks

I believe this project was a success, as it demonstrated the ability to take a still image, and overlay another image so that it appears like the image should belong in the image. It lays the groundwork for implementing this with a streaming video source. There is also the ability for someone to take this project and output the result to a television screen.

5 References

- Power Point Slides located at http://my.fit.edu/~vkepuska/ece3552
- Veton Këpuska
- Ronald Ramdhan
A Blackfin C Code

A.1 main.c Source Code

Listing 1: main.c source code

```c
/* ************************************************************************* *
** Name: FIO pins, Interrupts, and Timer in C **
** *************************************************************************/
(C) Copyright 2006 – Analog Devices, Inc. All rights reserved.
Project Name: BF533 Flags C
Date Modified: 3/31/03 Ver 1.01
Software: VisualDSP++ 4.5
Hardware: ADSP-BF533 EZ-KIT Lite
Connections: Switch SW9.1 has to be turned on in order to connect SW7 to PF8
Purpose: To demonstrate the configuration of FIO pins, Timer, and Interrupts in C

***************************************************************************/
#include "BF533,Flags.h"
#include "ccblkfn.h"
#include "sysreg.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

// Variables
short sLight_Move_Direction = 0;

// Variables used to store and hold images
unsigned char imgInput[143][200]; // 143x200 (height x width)
unsigned char overlayInput[30][200]; // 30x200 (height x width)
unsigned char imgOutput[143][200]; // 143x200 (height x width)

void performOverlay(int orgwidth, int orgheight, int overwidth, int overheight)
{
    // Initialize i,j, heightDiff to 0
    int i = 0;
    int j = 0;
```
int heightDiff = 0;
// Determine height difference between orgheight and overlay height
heightDiff = orgheight - overheight;

// Subtract 255 from the overlayInput image so it is inverted and will overlay nicer.
// Inverts overlay image
for (i = 0; i < overheight; i++)
{
  for (j = 0; j < overwidth; j++)
  {
    overlayInput[i][j] = 255 - overlayInput[i][j];
  }
}

// set i, j to 0
i = 0;
j = 0;

// initialize imgOutput to 0
for (i = 0; i < orgheight; i++)
{
  for (j = 0; j < orgwidth; j++)
  {
    imgOutput[i][j] = 0;
  }
}
i = 0;
j = 0;

// Copy imgInput to imgOutput
for (i = 0; i < orgheight; i++)
{
  for (j = 0; j < orgwidth; j++)
  {
    imgOutput[i][j] = imgInput[i][j];
  }
}
i = 0;
j = 0;

// copy overlayInput onto imgOutput
for (i = 0; i < overheight + 1; i++)
{
  for (j = 0; j < orgwidth + 1; j++)
  {
    imgOutput[heightDiff+i][j] = imgOutput[heightDiff+i][j] - overlayInput[i][j];
  }
}
Function : main //

void main(void) {

    sysreg_write(reg_SYSCFG, 0x32); // Initialize System Configuration Register
    Init_Flags();
    Init_Timers();
    Init_EBIU();
    Init_Flash();
    Init_Interupts();
    Init_SDRAM();

    // performs overlay of function
    performOverlay(200, 143, 200, 30);
    while(1);
}

A.2 BF533 Flags.h Source Code

Listing 2: BF533 Flags.h source code

#ifndef _BF533_FLAGS_DEFINED
#define _BF533_FLAGS_DEFINED

// Header files

#include <sys\exception.h>
#include <cdefBF533.h>
#include <sys/05000311.h>

// Symbolic constants

#define pFlashA_PortB_Dir (volatile unsigned char *)0x20270007
#define pFlashA_PortB_Data (volatile unsigned char *)0x20270005

// addresses for Port B in Flash A
#define PortB_Dir (volatile unsigned char *)0x20270007

// Global variables

extern short sLight_Move_Direction ;
// Would need following lines if going to use push buttons
extern unsigned char imgInput[143][200]; // 495x353 (width x height)
A.3 Initialization.c Source Code

Listing 3: Initialization.c source code

```c
#include "BF533_Flags.h"

void Init_Flags(void) {
    // Changed to F to enable 4 buttons
    FIO_ANOM_0311_INEN_W(0x0F00); // *FIO_INEN = 0x0100, workaround for anomaly 05000311.
    FIO_ANOM_0311_DIR_W(0x0000); // For more information please refer to the comment header in file sys/05000311.h
    FIO_ANOM_0311_EDGE_W(0x0F00);
    FIO_ANOM_0311_MASKA_W(0x0F00, pFIO_MASKA_D);
}
```
// Parameters: None //
// //
// Return: None //
//
// Description: This function initialises Timer0 for PWM mode. //
// It is used as reference for the 'shift-clock'. //
//-------------------------------------------------------------------//

void Init_Timers(void)
{
    // IRQ enabled, PWMOUT Mode, count to end of period
    *pTIMER0_CONFIG = 0x0019;
    *pTIMER0_PERIOD = 0x00800000;
    *pTIMER0_WIDTH = 0x00400000;
    *pTIMER_ENABLE = 0x0001;
}

// Function: Init_EBIU //
// //
// Parameters: None //
// //
// Return: None //
//
// Description: This function initialises and enables the asynchronous //
// memory banks for the External Bus Interface Unit (EBIU), so //
// that access to Flash A is possible. //
//-------------------------------------------------------------------//

void Init_EBIU(void)
{
    *pEBIU_AMBCTL0 = 0x7bb07bb0;
    *pEBIU_AMBCTL1 = 0x7bb07bb0;
    *pEBIU_AMGCTL = 0x000f;
}

// Function: Init_Flash //
// //
// Parameters: None //
// //
// Return: None //
//
// Description: This function sets the pin direction of Port B in Flash A //
// to output. //
// The LEDs on the ADSP-BF533 EZ-KIT are connected to Port B. //
//-------------------------------------------------------------------//

void Init_Flash(void)
{
    *pFlashA_PortB_Dir = 0x3f;
}

// Function: Init_Interrupts //
// Parameters: None //
// Return: None //
// Description: This function initialises the interrupts for Timer0 and //
// FlagA (PF8). //

/*
  * @author
  * @version
  */

void Init_Interrupts(void)
{
    // assign core IDs to interrupts
    *pSIC_IAR0 = 0xffffffff;
    *pSIC_IAR1 = 0xffffffff;
    *pSIC_IAR2 = 0xffffffff;  // Timer0 -> ID4; FlagA -> ID5

    // assign ISRs to interrupt vectors
    register_handler(ik_ivg11, Timer0_ISR);  // Timer0 ISR -> IVG 11
    register_handler(ik_ivg12, FlagA_ISR);   // FlagA ISR -> IVG 12

    // enable Timer0 and FlagA interrupt
    *pSIC_IMASK = 0x00090000;
}

// Initializes SDRAM

/*
  * @author
  * @version
  */

void Init_SDRAM(void)
{
    // EBIUS_SDSTAT is The SDRAM control Status Register, which needs
    // to be enabled as well as the SDRS bit in the control status register
    // has to be high which means, Will power up on next SDRAM
    // access if SDRAM enabled

    if (*pEBIU_SDSTAT & SDRS) {

        //SDRAM Refresh Rate Control Register
        // Optimum Settings for SDRAM are set

        // Following register is an RDIV value from the formula
        // RDIV= ((Fslck x Tref)/NRA)−(Tras+Trp)
        // Pg 748–749 of Hardware Reference Manual for more description
        *pEBIU_SDRRC = 0x00000817; //0406

        //SDRAM Memory Bank Control Register
        // Sets system clock to optimum settings
        // the 1 sets 9bits for SDRAM external bank column width
        // the 13 sets 32MB external SDRAM size as well as enabling the SDRAM
        // external bank
        // Page 745 of Hardware Reference Manual for more description

        // If board is revision <=1.5 use following line


A.4 ISRs.c Source Code

Listing 4: ISRs.c source code

```c
#include "BF533 Flags.h"

// Function: Timer0_ISR // Parameter: None // Return: None //
// Description: This ISR is executed every time Timer0 expires. // The old LED pattern is shifted by one; the direction depends on the state of sLight_Move_Direction, which is changed in FlagA_ISR. //
__interrupt void Timer0_ISR(void)
{
    static unsigned char ucActive_LED = 0x01;
    // confirm interrupt handling
    *pTIMER_STATUS = 0x0001;
    // shift old LED pattern by one
    if(sLight_Move_Direction)
    {
        if((ucActive_LED = ucActive_LED >> 1) != 0x00) ucActive_LED = 0x20;
    }
    else
    {
```

if ((ucActive_LED = ucActive_LED << 1) == 0x40) ucActive_LED = 0x01;
}

// write new LED pattern to Port B
*pFlashA PortB_Data = ucActive_LED;
}

// Function: FlagA_ISR
// Parameters: None
// Return: None
// Description: This ISR is called every time the button connected to PF8 is pressed.
// The state of flag sLight_Move_Direction is changed, so the shift-direction for the LED pattern in Timer0_ISR changes.

EX_INTERRUPT_HANDLER(FlagA_ISR)
{
    // confirm interrupt handling
    FIO_ANOM_0311_FLAG_W(0x0100, pFIO_FLAG_C); // *pFIO_FLAG_C = 0x0100, workaround for anomaly 05000311.

    // If implementing buttons use the following line
    if (*pFIO_FLAG_C == 0x0100) {
        *pFIO_FLAG_C = 0x0100;
    }

    // toggle direction of moving light
    sLight_Move_Direction = ~sLight_Move_Direction;
}

B Matlab Code

B.1 imageoverlay.m Source Code

Listing 5: imageoverlay.m source code

function imageoverlay(orgImg, orgOverlay)
%
% orgImg = Original JPG image
% orgOverlay = Original Overlay (white background)
%
orgImg = imread(orgImg);
overlay = imread(orgOverlay);
overlaySize = size(overlay);
overlayHeight = overlaySize(1);

output=orgImg;
overlay2=255−overlay;
output(354−overlayHeight:353,1:495)=output(354−overlayHeight:353,1:495)−overlay2;

imshow(output);

for i=1:353
    for j=1:495
        output2(i,j)=orgImg(i,j);
    end
end
imshow(output2);

for i=1:75
    for j=1:495
        output2(353−overlayHeight+i,j)=output(353−overlayHeight+i,j)−overlay2(i,j);
    end
end
imshow(output2);