EE445L - Lab 01 Report

Kevin Gilbert
Gilberto Rodriguez

Professor Bard
Lab: Monday/Wednesday 5-6:15

Janury 31 2014

Objective

- To introduce the lab equipment
- To familiarize ourselves with Keil uVision4 and the ARM Cortex M processor
- To develop a set of useful fixed-point output routines for use in future labs.

Lab Analysis and Discussion

Problem 1. In what way is it good design of fixed.c that there is no arrow directly from the fixed.c module to the rit128x96x4.c module in the call graph for your system?

The extra level of abstraction makes fixed.c more modular. By removing the rit128x96x4.c file directly, any errors can be limited to a particular module. It limits what any individual module is doing.

Problem 2. Why is it important for the decimal point to be in the exact same physical position independent of the number being displayed?

We were asked to implement a certain resolution for the numbers to be readable by humans.

Problem 3. When should you use fixed-point over floating point? When should you use floating-point over fixed-point?

You should use fixed-point over floating point when the processor is optimized to do integer operations. You should use floating-point over fixed-point in all other cases.

Problem 4. When should you use binary fixed-point over decimal fixed-point? When should you use decimal fixed-point over binary fixed-point?

Binary fixed-point is supported in hardware and easier for computations. Decimal fixed-point is easier for humans to use In addition, base two uses shifts rather than multiplication/division as in base ten, making it faster to compute.

Problem 5. Give an example application (not mentioned in this lab assignment) for fixed-point. Describe the problem, and choose an appropriate fixed-point format. (no software implementation required).

Say we were to develop a stopwatch application on an embedded system that did not support floating point. We can use a fixed-point data format to represent milliseconds to a human user.

Problem 6. Can we use floating point on the Arm Cortex M3? If so, what is the cost?

There is no floating point on the Arm Cortex M3, but it can be simulated in software at the cost of multiple cycles.

Problem Extra Credit. Is fixed-point or floating-point arithmetic faster on the Pentium w/MMX?

Fixed-point arithmetic is faster on the Pentium with the MMX because the MMX is an extension to the Intel Pentium chip which allows for parallel integer operation.

fixed.c

```
/*************************************************************************
  File Name: fixed.c
  Author(s): Kevin Gilbert and Gilberto Rodriguez
  Initial Creation Date: 1/22/14
  Description: A set of funtions that can take in unsigned integer, signed integer,
                  and binary values and output them to the LED screen on the 1968 borad
  Lab Number: 1
  TA: Mahesh Srinivasan and Zichong Li
  Date of last revision: 1/27/14
  Hardware Configuration: n/a
*************************************************************************/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "fixed.h"

void Fixed_uDecOut2s(unsigned long n,  char *string){
char holderString[10];
if (n > 99999){
strcpy(holderString, "***.**");
}else{
sprintf(holderString, "%3lu.%02lu", n / 100, n % 100);
}
sprintf(string, holderString);
}

void Fixed_uDecOut2(unsigned long n) {
 char output[10];
 Fixed_uDecOut2s(n, output);
 printf("%s\r",output);
}

void Fixed_sDecOut3s(long n, char *string){
char holderString[10];
if ((n > 9999) || (n < -9999)){
strcpy(holderString, " *.***");
}else{
sprintf(holderString, "%s%lu.%03lu", (n >= 0) ? (" ") : ("-"), abs(n) / 1000, abs(n) % 1000);
}
sprintf(string, holderString);
}

void Fixed_sDecOut3(long n){
 char output[10];
 Fixed_sDecOut3s(n, output);
 printf("%s\r",output);
}

void Fixed_uBinOut8s(unsigned long n,  char *string){
unsigned long fixedNumber = (n * 100) / 256;
  char holderString[10];
if (n >= 256000){
 strcpy(holderString, "***.**");}
  else{
sprintf(holderString, "%3lu.%02lu", fixedNumber / 100, fixedNumber % 100);
}
  sprintf(string, holderString);
}

void Fixed_uBinOut8(unsigned long n){
 char output[6];
 Fixed_uBinOut8s(n, output);
 printf("%s\r",output);
}
```

Lab1.c

```
#include <stdio.h>
#include <stdlib.h>
#include "fixed.h"
#include "output.h"
// const will place these structures in ROM
const struct outTestCase{       // used to test routines
  unsigned long InNumber;       // test input number
  char OutBuffer[10];           // Output String
```

```
};
const struct s_outTestCase {
  signed long InNumber;
  char OutBUffer[10];
};

typedef const struct outTestCase outTestCaseType;
typedef const struct s_outTestCase s_outTestCaseType;
outTestCaseType outTests1[10]={
{     0,  "  0.00" }, //      0/100 = 0.00
{     1,  "  0.01" }, //      1/100 = 0.01
{    99,  "  0.99" }, //     99/100 = 0.99
{   100,  "  1.00" }, //    100/100 = 1.00
{   999,  "  9.99" }, //    999/100 = 0.99
{  1000,  " 10.00" }, //   1000/100 = 10.00
{  9999,  " 99.99" }, //   9999/100 = 99.99
{ 10000,  "100.00" }, //  10000/100 = 100.00
{ 99999,  "999.99" }, //  99999/100 = 999.99
{100000,  "***.**" }, //  error condition
};

s_outTestCaseType outTests2[10]={
{-100000,  " *.***"  }, //      error condition
{ -10000,  " *.***"  }, //      error condition
{  -9999,  "-9.999" }, //   -9999/100 = -9.999
{   -999,  "-0.999" }, //    -999/100 = -0.999
{     -1,  "-0.001" }, //      -1/100 = -0.001
{      0,  " 0.000" }, //       0/100 =  0.000
{     123, " 0.123" }, //     123/100 =  0.123
{    1234, " 1.234" }, //    1234/100 =  1.234
{    9999, " 9.999" }, //    9999/100 =  9.999
{   10000, " *.**"  }, //  error condition
};

outTestCaseType outTests3[16]={
{     0,  "  0.00" }, //      0/256 = 0.00
{     4,  "  0.01" }, //      4/256 = 0.01
{    10,  "  0.03" }, //     10/256 = 0.03
{   200,  "  0.78" }, //    200/256 = 0.78
{   254,  "  0.99" }, //    254/256 = 0.99
{   505,  "  1.97" }, //    505/256 = 1.97
{  1070,  "  4.17" }, //   1070/256 = 4.17
{  5120,  " 20.00" }, //   5120/256 = 20.00
{ 12184,  " 47.59" }, //  12184/256 = 47.59
{ 26000,  "101.56" }, //  26000/256 = 101.56
{ 32767,  "127.99" }, //  32767/256 = 127.99
{ 32768,  "128.00" }, //  32768/256 = 128
{ 34567,  "135.02" }, //  34567/256 = 135.02
```

```
{123456,  "482.25" }, // 123456/256 = 482.25
{255998,  "999.99" }, // 255998/256 = 999.99
{256000,  "***.**" }  // error
};
unsigned int Errors,AnError;
char Buffer[10];
int main(void){ // possible main program that tests your functions
  unsigned int i;
  Output_Init();
  Output_Color(15);

  Errors = 0;
  for(i=0; i<16; i++){
    printf("uBinOut8: ");
    Fixed_uBinOut8(outTests3[i].InNumber);
  }
  printf("________________\r");
  Delay(8000000);

  for(i=0; i<10; i++) {
printf("sDecOut3: ");
Fixed_sDecOut3(outTests2[i].InNumber);
  }
  printf("________________\r");
  Delay(8000000);
  for(i=0;i<10;i++) {
printf("uDecOut2: ");
Fixed_uDecOut2(outTests1[i].InNumber);
  }
  Delay(8000000);
    if(strcmp(Buffer, outTests3[i].OutBuffer)){
      Errors++;
      AnError = i;
    }
  for(;;) {} /* wait forever */
}
```