Andrea Smith

EE 2361

Lab 2b

**How many cycles does your hard coded program take to write 24-bits?**

By using the logic analyzer, the hard coded program takes about 450 cycles.

**How many cycles does your new function take to write 24-bits?)**

By using the logic analyzer again, the new function takes about 800 cycles.

/\*  
\* File:   smit9523\_lab2b\_main.c  
\* Author: andreasmith  
\*  
\* Created on February 11, 2020, 8:05 PM  
\*/  
  
#include "xc.h"  
#include "smit9523\_lab2b\_assembly.h"  
#include "stdint.h"  
// CW1: FLASH CONFIGURATION WORD 1 (see PIC24 Family Reference Manual 24.1)  
#pragma config ICS = PGx1          // Comm Channel Select (Emulator EMUC1/EMUD1 pins are shared with PGC1/PGD1)  
#pragma config FWDTEN = OFF        // Watchdog Timer Enable (Watchdog Timer is disabled)  
#pragma config GWRP = OFF          // General Code Segment Write Protect (Writes to program memory are allowed)  
#pragma config GCP = OFF           // General Code Segment Code Protect (Code protection is disabled)  
#pragma config JTAGEN = OFF        // JTAG Port Enable (JTAG port is disabled)  
  
  
// CW2: FLASH CONFIGURATION WORD 2 (see PIC24 Family Reference Manual 24.1)  
#pragma config I2C1SEL = PRI       // I2C1 Pin Location Select (Use default SCL1/SDA1 pins)  
#pragma config IOL1WAY = OFF       // IOLOCK Protection (IOLOCK may be changed via unlocking seq)  
#pragma config OSCIOFNC = ON       // Primary Oscillator I/O Function (CLKO/RC15 functions as I/O pin)  
#pragma config FCKSM = CSECME      // Clock Switching and Monitor (Clock switching is enabled,   
                                       // Fail-Safe Clock Monitor is enabled)  
#pragma config FNOSC = FRCPLL      // Oscillator Select (Fast RC Oscillator with PLL module (FRCPLL))  
  
#define PERIOD 7 // period of blinks in milliseconds  
  
void setup(void) {  
    CLKDIVbits.RCDIV = 0; // Sets RCDIV = 1:1 (default 2:1) 32MHz or FCY/2=16M [Changes max clk rate    ]  
    AD1PCFG = 0x9fff;            //sets all pins to digital I/O  
    AD1PCFG = 0x9fff;  
    TRISA = 0b1111111111111110;  
    TRISB = 0x0000;    
    delay\_100us();  
}  
  
void delay(int delay\_in\_ms) {  
    int i = 0;  
    for (i = 0; i < delay\_in\_ms; i++) {  
        delay\_1ms();  
    }  
}  
  
void foreverLoop(void) {  
    while(1) {  
        LATA = 0x0000;  
        gradient();  
          
//        PRELAB  
//        // Red  
//        write\_1();  
//        write\_1();  
//        write\_1();  
//        write\_1();  
//        write\_0();  
//        write\_1();  
//        write\_0();  
//        write\_1();  
//          
//        // Blue  
//        write\_1();  
//        write\_1();  
//        write\_1();  
//        write\_1();  
//        write\_0();  
//        write\_1();  
//        write\_0();  
//        write\_1();  
//          
//        // Green  
//        write\_0();  
//        write\_0();  
//        write\_0();  
//        write\_0();  
//        write\_0();  
//        write\_1();  
//        write\_0();  
//        write\_1();  
          
    }  
}  
  
void writeColor(int r, int g, int b) {  
//   LATA = 0x0000;  
    int i = 0;  
  
        // Red  
        for(i = 0; i < 8; i++) {  
            if ((0b10000000 & r) == 0b10000000) {  
                write\_1();  
            }  
            else {  
                write\_0();  
            }  
            r = r << 1;  
        }  
  
        // Green  
        for(i = 0; i < 8; i++) {  
            if ((0b10000000 & g) == 0b10000000) {  
                write\_1();  
            }  
            else {  
                write\_0();  
            }  
            g = g << 1;  
        }  
  
        // Blue  
        for (i = 0; i < 8; i++) {  
            if ((0b10000000 & b) == 0b10000000) {  
                write\_1();  
            }  
            else {  
                write\_0();  
            }  
            b = b << 1;  
        }  
}  
  
unsigned char getR(uint32\_t RGBval) {  
    return (unsigned char) (RGBval >> 16);  
}  
  
unsigned char getG(uint32\_t RGBval) {  
    return (unsigned char) (RGBval >> 8);  
}  
  
unsigned char getB(uint32\_t RGBval) {  
    return (unsigned char) (RGBval >> 0);  
}  
  
uint32\_t packColor(unsigned char Red, unsigned char Grn, unsigned char Blu) {  
    return ((long) Red << 16) | ((long) Grn << 8) | ((long) Blu);  
}  
  
void writePacCol(uint32\_t PackedColor) {  
    writeColor(getR(PackedColor), getG(PackedColor), getB(PackedColor));  
}  
  
void gradient(void) {  
    int i;  
    for (i = 255; i > 0; i--) {  
        writeColor(i, 0, 255-i);  
        delay(PERIOD);  
    }  
    for (i = 0; i < 255; i++) {  
        writeColor(i, 0, 255-i);  
        delay(PERIOD);  
    }  
}  
  
uint32\_t Wheel(unsigned char WheelPos) {  
  WheelPos = 255 - WheelPos;  
  if(WheelPos < 85) {  
    return packColor(255 - WheelPos \* 3, 0, WheelPos \* 3);  
  }  
  if(WheelPos < 170) {  
    WheelPos -= 85;  
    return packColor(0, WheelPos \* 3, 255 - WheelPos \* 3);  
  }  
  WheelPos -= 170;  
  return packColor(WheelPos \* 3, 255 - WheelPos \* 3, 0);  
}  
  
int main(void) {  
    setup();  
    int i = 0;  
    while(1) {  
        for(i = 0; i<255; i++) {  
            writePacCol(Wheel(i));  
            delay(PERIOD);  
        }  
    }  
    return 0;  
}