Ruide Xie

Lab Report

Assembly file:

```
_write_0:
   BSET LATA, #0x0
   repeat #3
   nop
   BCLR LATA, #0x0
   repeat #6
   nop
   return

_write_1:
   BSET LATA, #0x0
   repeat #10
   nop
   BCLR LATA, #0x0
   nop
   return

_xie_wait_100us:
   repeat #1593
   nop
   return
_xie_wait_1ms:
   repeat #15993
   nop
   return
```

The delay function cycles were calculated using desired time delay divided by 62.5ns. 100us/62.5ns-1-2-3-1=1593. 1ms/62.5ns-1-2-3-1=15993. The total cycles inside both write 1 and write 0 functions are 20. In write_0, high pulse takes 1+1+4=6cycles, and low pulse takes 1+1+7+3+2=14 cycles. In write_1, high pulse takes 1+1+11=13 cycles, and low pulse takes 1+1+3+2=7 cycles. The write_1 function was first set to 11 high cycles and 9 low cycles, however it did not work on the PIC24 so it was guessed that the high pulse was not sensitive enough to send signal 1. Thus after increasing the high pulse and decreasing the low pulse 2 cycles each, the function works perfect with a total of 20 cycles unchanged. Color of red(write_1*8, write_0*16), green(write_0*8, write_1*8, write_0*8), blue(write_0*16, write_1*8), and purple(write_1*1, write_0*15, write_1*1, write_0*7) were tested, the iLED showed all correct colors and demo was presented.

Main file code:

```
#include "xc.h"
#include "xie_lab2b_asmLib_v001.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 5
void setup(void){
   CLKDIVbits.RCDIV = 0;  //Set RCDIV=1:1 (default 2:1) 32MHz or FCY/2=16M
   AD1PCFG = 0x9fff;
   TRISA = 0b1111111111111110;
   LATA = 0x0000;
```

```
}

void writeColor(int r, int g, int b){
    xie_wait_100us();
    int i;
    for(i=7; i>=0; i--){
        if((r&(1<<i))==0){
            write_0();
        }else{
            write_1();
        }
    }
    int j;
    for(j=7; j>=0; j--){
        if((g&(1<<j))==0){
            write_0();
        }else{
            write_1();
        }
    }
    int k;
    for(k=7; k>=0; k--){
        if((b&(1<<k))==0){
            write_0();
        }else{
            write_1();
        }
    }
}

void delay(int delay_in_ms){
    int i;
    for(i=0;i<delay_in_ms;i++){
        xie_wait_1ms();
    }
}

unsigned long int packColor(unsigned char Red, unsigned char Grn, unsigned char
Blu){
```

```
    unsigned long int RGBval = 0;
    RGBval = ((long) Red << 16) | ((long) Grn << 8) | ((long) Blu);
    return RGBval;
}

unsigned char getR(uint32_t RGBval){
    unsigned char Red = 0;
    Red = (unsigned char) (RGBval >> 16);
    return Red;
}

unsigned char getG(uint32_t RGBval){
    unsigned char Green = 0;
    Green = (unsigned char) (RGBval >> 8 );
    return Green;
}
unsigned char getB(uint32_t RGBval){
    unsigned char Blue = 0;
    Blue  = (unsigned char) (RGBval >> 0 );
    return Blue;
}

void writePacCol(uint32_t PackedColor){
    unsigned char red=getR(PackedColor);
    unsigned char green=getG(PackedColor);
    unsigned char blue=getB(PackedColor);
    writeColor(red,green,blue);
}

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;
```

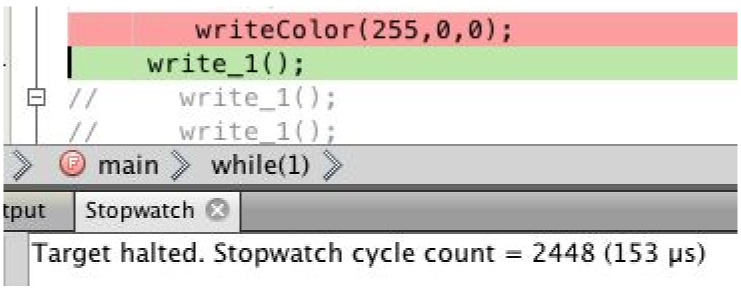
```
    return packColor(WheelPos * 3, 255 - WheelPos * 3, 0);
}

int main(void) {
    setup();
    xie_wait_100us();
    while(1){
//        writeColor(255,0,0);
     write_1();
     write_1();
     write_1();
     write_1();
     write_1();
     write_1();
     write_1();
     write_1();

     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
     write_0();
    delay(10);
        int FrameNumber;
   }
        for(FrameNumber = 0;  FrameNumber<= 255; FrameNumber++){
            uint32_t wheelVar=Wheel(FrameNumber);
            writePacCol(wheelVar);
```

```
            delay(10);
        }
    }
    return 0;
}
```

The hard coded program that calls write_1 and write_0 functions 24 times in total takes 480 cycles in the demo of lighting red color. The new writeColor(red,green,blue) function rather takes 2448 cycles. It is shown that the new function takes more time than the hardcode. However it shows huge convenience of usage and the saving of space to demo color.



The next part of the lab is to achieve the colorful animation using iLED. The method in this implementation is the wheel call. The wheel function is called inside the cyclical function which takes the FrameNumber each at a time inside a loop. It reaches the goal of gradual transition from a color to another, for instance from red transitioning to blue, the algorithm calcualtes (255,0,0) to (254,0,1)...all the way to (0,0,255) which completes the transition. With wheel function inside, the color is first packed into 32 bit with bit shifting and masking in the packColor function, and then a wheelvalue can be generated using the wheel algorithm, and lastly the color is unpacked using bit shifting and casting. It is no worry about the bits infront of the 7th bit because unsigned char type only takes the last 8 bits of number. The unpacked color will be written inside the writePacCol function which generates a color on iLED. A delay function can be necessary for human vision to catch the stay of that single color. This whole process will repeat the desired amount of time set by the user to achieve color animation.