Digital Lab 4:

Experiment 2:

Interrupt Service Routine and Clock

Date: 2024/03/19

Class: 電機三全英班

Group: Group 11

Name: B103105006 胡庭翊

I. Annotated Code

```c
#include "stdio.h"
#include "NuMicro.h"
#include "tmr.h"
#include "system_init.h"
#include "GUI.h"
#include "display.h"

/* define */
#define MaxSpeed    50000       //led toogle speed  51000 - 50000 = 1000 ==>0.1s
#define MinSpeed    1000        //led toogle speed  51000 - 1000  = 50000==>5s
#define SW_UP       PC9         //UP                JoyStick
#define SW_DOWN     PG4         //DOWN              JoyStick
#define SW_CTR      PG3         //CENTER            JoyStick
#define LED1 PH6                //LED R1
#define LED2 PH7                //LED G1

uint32_t SpeedCtl;
uint32_t timecount;
uint32_t sec = 0;
uint32_t hour = 0;
uint32_t min = 0;

void Clock_Task(void);
void clock_init(void);
void clock_tick(void);
void LED_showing(uint32_t SpeedCtl);
void GPIO_init(void);
uint32_t JoyStick(unsigned char BTN_state);

typedef union{
    struct{
        //1 BIT SIZE VAR
        unsigned UP     :1;
        unsigned DOWN   :1;
        unsigned CTR    :1;
    };
unsigned char JState;
}Joystick_union;
// UP, DOWN, CTR, JState, are included in the union
Joystick_union JSUnion;
```

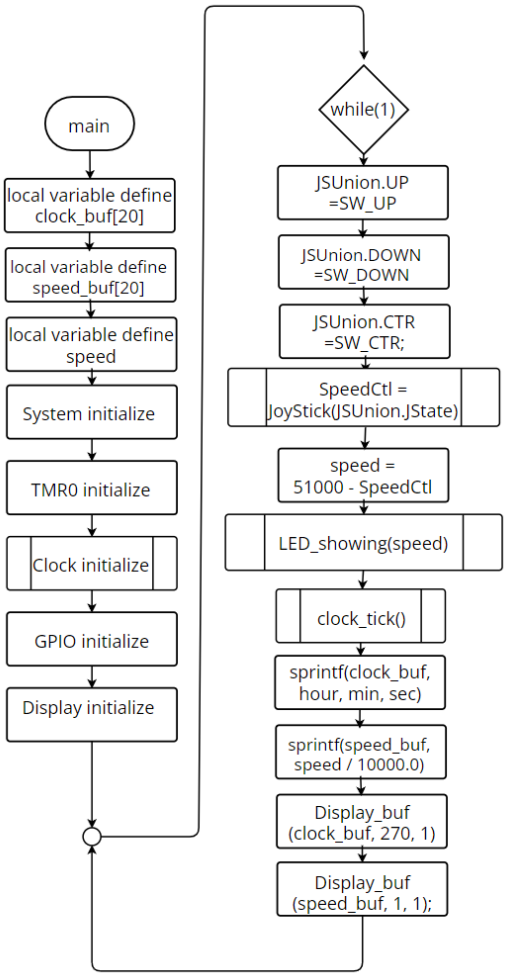
```c
int main(void)
{
        char clock_buf[20];
        char speed_buf[20];
        uint32_t speed;


        SYS_Init();

        TMR0_Initial();

        clock_init();

        GPIO_init();

        Display_Init();

    while(1)
        {

                JSUnion.UP      =SW_UP;
                JSUnion.DOWN    =SW_DOWN;
                JSUnion.CTR     =SW_CTR;
                //bottons' input signals are assigned into variables in union

                SpeedCtl = JoyStick(JSUnion.JState);
                speed = 51000 - SpeedCtl; //speedCrl will be the amout of speed decreased

                LED_showing(speed); //variable speed will effect how the led shown
                clock_tick();

                sprintf(clock_buf, "%02d:%02d:%02d", hour, min, sec);
                sprintf(speed_buf, "speed = %.1f (s)",speed / 10000.0); //the convertion of the time unit

                Display_buf(clock_buf, 270, 1);
                Display_buf(speed_buf, 1, 1);
        }
}
```

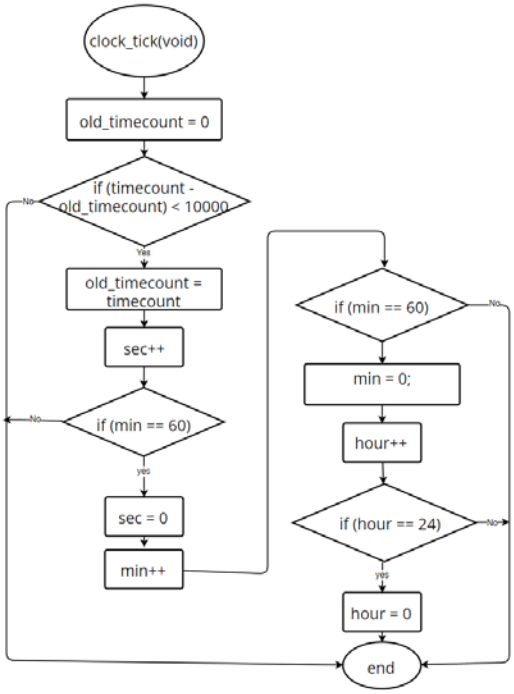
```c
/* GPIO initialize */
void GPIO_init(void)
{
        GPIO_SetMode(PA, BIT0, GPIO_MODE_INPUT) ;     // SW1
        GPIO_SetMode(PH, BIT6, GPIO_MODE_OUTPUT) ; // LEDR1
        GPIO_SetMode(PH, BIT7, GPIO_MODE_OUTPUT) ; // LEDG1
        GPIO_SetMode(PC, BIT9, GPIO_MODE_INPUT) ;     // Joystyick_UP
        GPIO_SetMode(PG, BIT4, GPIO_MODE_INPUT) ;     // Joystyick_DOWN
        GPIO_SetMode(PG, BIT3, GPIO_MODE_INPUT) ;     // Joystyick_CENTER
}

//time initialize
void clock_init(void)
{
        sec = 0;
        min = 0;
        hour = 0;
}

// define clock that can count automatically with hour, minute, sec convertion
void clock_tick(void)
{
        static uint32_t old_timecount = 0;

        if((uint32_t)(timecount - old_timecount) < 10000)
                return;

        old_timecount = timecount;
        sec++;
        if (sec == 60)
    {
        sec = 0;
        min++;
        if (min == 60)
        {
            min = 0;
            hour++;
            if (hour == 24)
                hour = 0;
        }
    }
}
```

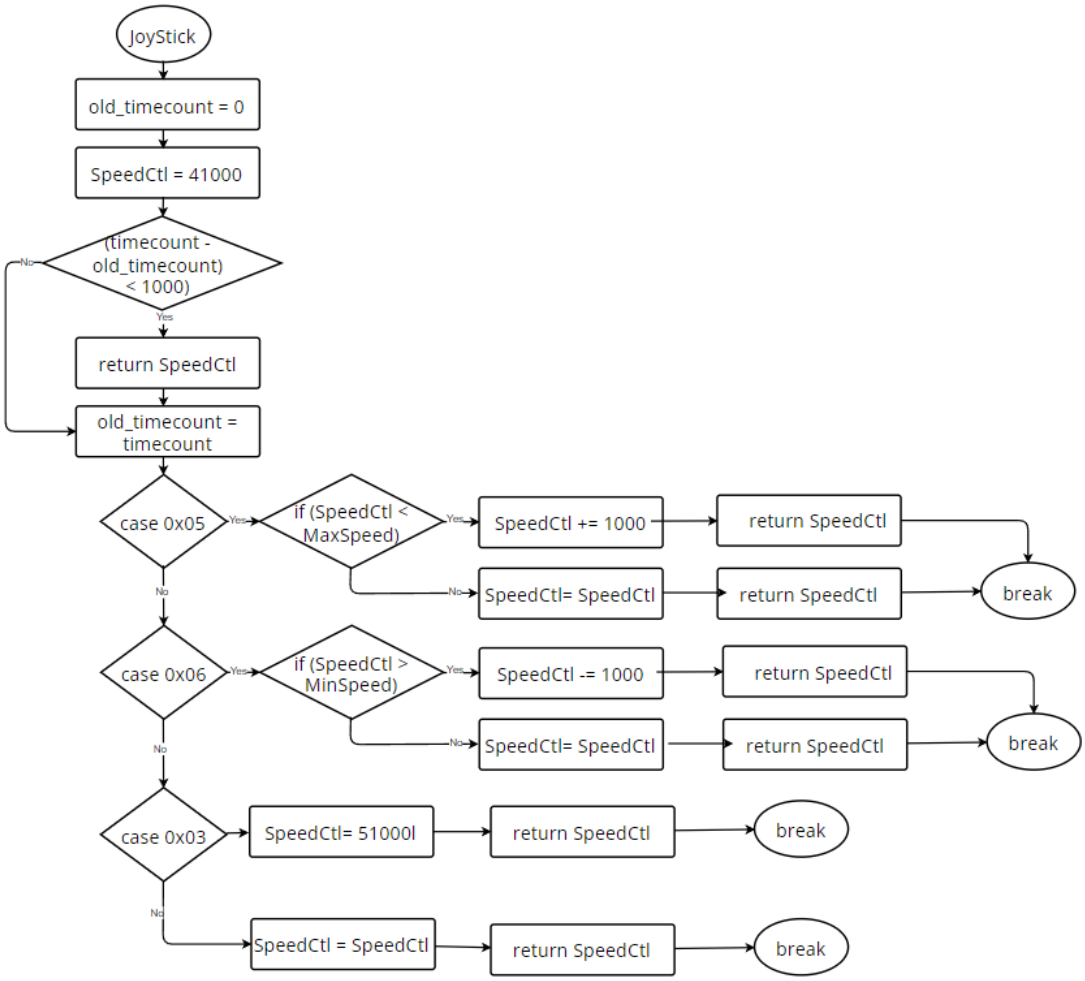
```c
uint32_t JoyStick(unsigned char BTN_state)
{
        static uint32_t old_timecount = 0;
        static uint32_t SpeedCtl = 41000;

        if((uint32_t)(timecount - old_timecount) < 1000){
            return SpeedCtl;}


        old_timecount = timecount;

        switch(BTN_state)
        {
            case 0x05: //CDU=101, down is pressed
            //speed down until SpeedCtrl meets MaxSpeed (SpeedCtl is the opposite of speed)
                if (SpeedCtl < MaxSpeed){
                    SpeedCtl += 1000;}
                else
                    SpeedCtl= SpeedCtl;
                break;
            case 0x06: //CDU=110, up is pressed
            //speed up until SpeedCtrl meets MinSpeed (SpeedCtl is the opposite of speed)
                if (SpeedCtl > MinSpeed){
                    SpeedCtl -= 1000;}
                else
                    SpeedCtl= SpeedCtl;
                break;
            case 0x03://CDU=011, center is pressed
            //speed == 0
                    SpeedCtl = 51000;
                break;
            default:
                    SpeedCtl = SpeedCtl;
                break;
        }//switch
    return SpeedCtl;
}
```

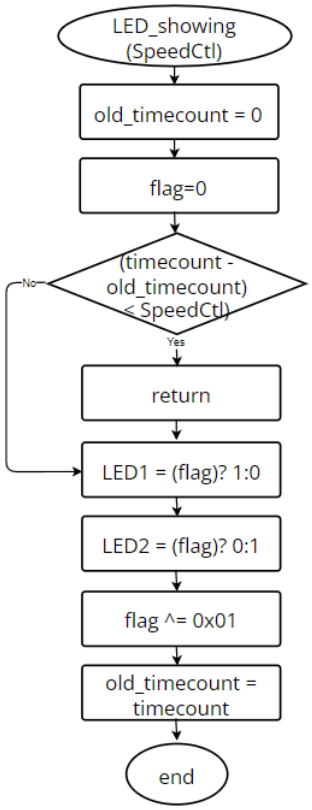
```c
void LED_showing(uint32_t SpeedCtl)
{
        static uint32_t old_timecount = 0;
        static unsigned char flag=0;
        if((uint32_t)(timecount - old_timecount) < SpeedCtl) //delay count
            return;

        LED1 = (flag)? 1:0;
        LED2 = (flag)? 0:1;
        flag ^= 0x01; //
        old_timecount = timecount;

}
```

II. Program Flow






JoyStick
old_timecount = 0
SpeedCtl = 41000
(timecount - old_timecount) < 1000)
No
Yes
return SpeedCtl
old_timecount = timecount
case 0x05
Yes
if (SpeedCtl < MaxSpeed)
Yes
SpeedCtl += 1000
return SpeedCtl
No
SpeedCtl= SpeedCtl
return SpeedCtl
break
No
case 0x06
Yes
if (SpeedCtl > MinSpeed)
Yes
SpeedCtl -= 1000
return SpeedCtl
No
SpeedCtl= SpeedCtl
return SpeedCtl
break
No
case 0x03
SpeedCtl= 51000l
return SpeedCtl
break
No
SpeedCtl = SpeedCtl
return SpeedCtl
break


LED_showing (SpeedCtl)
old_timecount = 0
flag=0
(timecount - old_timecount) < SpeedCtl)
No
Yes
return
LED1 = (flag)? 1:0
LED2 = (flag)? 0:1
flag ^= 0x01
old_timecount = timecount
end

III. Thoughts

This electrical engineering experiment provided me with the opportunity to delve into the applications of Interrupt Service Routine (ISR) and Clock in embedded systems, implementing related code in C language. While initially the code seemed lengthy and the content progressively more complex, as the experiment progressed, I gained a deeper understanding of the importance of ISR and Clock and how they collaborate to ensure the proper functioning of the system.

A noteworthy achievement was reached towards the end of the experiment when we successfully utilized joystick input to adjust the blinking speed of LEDs, which was quite exhilarating for me. However, what intrigued me was the discovery that on the joystick, the button for increasing speed actually slowed down the blinking of LEDs, whereas the button for decreasing speed accelerated the blinking. This phenomenon puzzled me until the assistant explained the underlying reason.

As it turned out, the use of case in the sample code to handle button events resulted in concurrent execution. This meant that when button events occurred, the system simultaneously processed multiple events, resulting in seemingly contradictory functionalities of the buttons. This finding deepened my understanding of concurrent execution in code and taught me how to appropriately handle such situations in practical applications.

Overall, this experiment proved to be quite enriching for me. It not only deepened my understanding of ISR and Clock but also taught me how to control system functionalities using joystick input. Although there were challenges encountered during the experiment, these challenges helped me gain a deeper insight into the workings of embedded systems, laying a solid foundation for my future learning and research endeavors.