Digital Lab 4:

Experiment 4:

Analog Interface Control

Date: 2024/04/09

Class: 電機三全英班

Group: Group 11

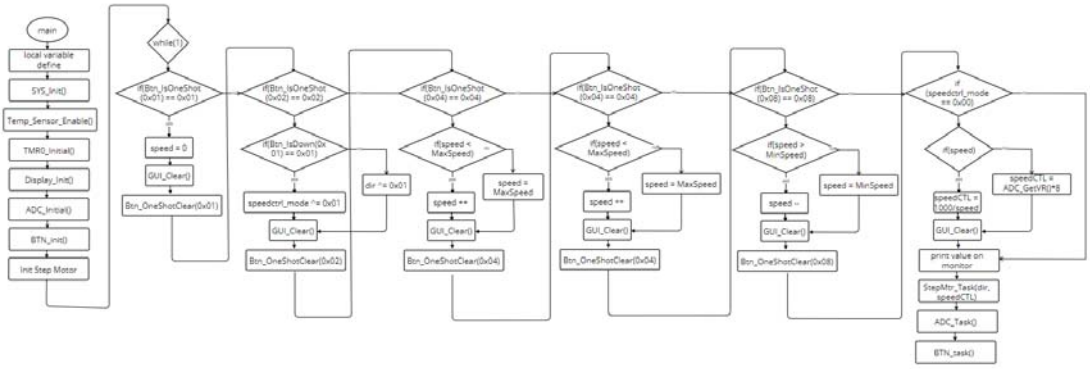
Name: B103105006 胡庭翊

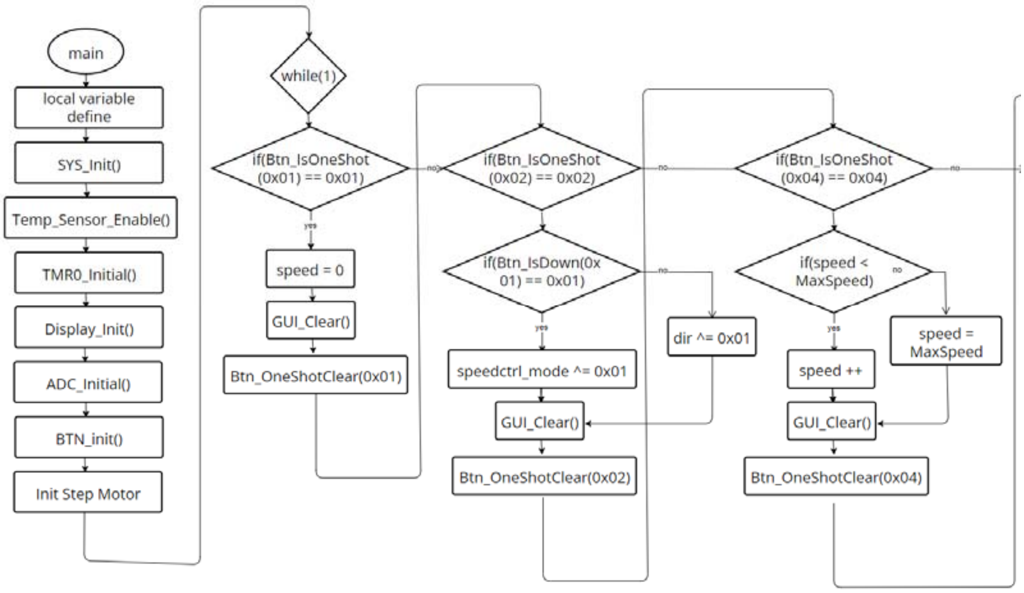
I. Annotated Code

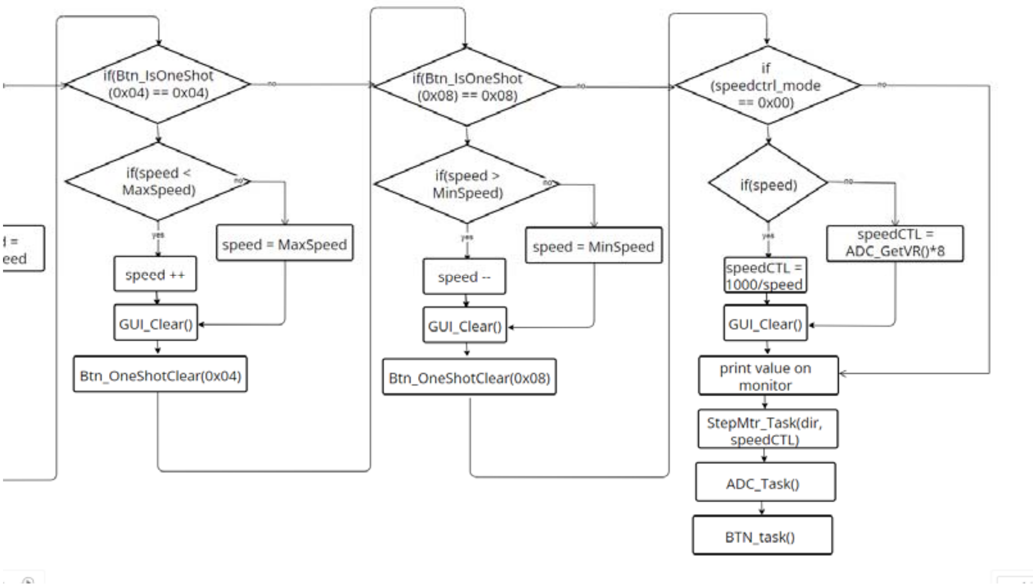
```c
#include "NuMicro.h"
#include "ADCAgent.h"
#include "TempSensor.h"
#include "system_init.h"
#include "display.h"
#include "tmr.h"
#include "GUI.h"
#include "sys.h"
#include "BNCTL.h"
#include "StepMotorAgent.h"

/* define max and mini speed */
#define MaxSpeed    17
#define MinSpeed    1

/* global variable define */
uint32_t timecount = 0;
uint32_t speed;
uint8_t  dir;
uint8_t speedctrl_mode;

int main(void)
{
    /* local variable define */
    char ADC_value_buf[20];
    char M487sensor_temp_value_buf[20];
    char thermistor_temp_value_buf[20];
    char speed_buf[20];
    char mode_buf[20];
    uint32_t  speedCTL;

    /* Init System, peripheral clock */
    SYS_Init();

    /* Init tempputer sensor */
    Temp_Sensor_Enable();

    /* Init TMR0 for timecount */
    TMR0_Initial();

    /* Opem GUI display */
    Display_Init();

    /* Init ADC */
    ADC_Initial();

    /* Init Button */
    BTN_init();

    /*Init Step Motor */
    StepMtr_Initial();
    dir = 1;
    speed = 10;
    speedctrl_mode = 0x00;

    while(1)
    {
        if(Btn_IsOneShot(0x01) == 0x01){
            //speed reset
            speed = 0;
            //clear the GUI display
            GUI_Clear();
            //clear one-shot flag
            Btn_OneShotClear(0x01);
        }
```

```
        if(Btn_IsOneShot(0x02) == 0x02){
            //direction change
            if (Btn_IsDown(0x01) == 0x01) {
                speedctrl_mode ^= 0x01; //reassign speed control mode
            }
            else {
            dir ^= 0x01;//direction change
            }
            //clear the GUI display
            GUI_Clear();
            //clear one-shot flag
            Btn_OneShotClear(0x02);
        }
        if(Btn_IsOneShot(0x04) == 0x04){
            //actual speed down
            if(speed < MaxSpeed)//speed increased
                speed ++;
            else
                speed = MaxSpeed;//set to maximum speed
            //clear the GUI display
            GUI_Clear();
            //clear one-shot flag
            Btn_OneShotClear(0x04);
        }
        if(Btn_IsOneShot(0x08) == 0x08){
            //actual speed up
            if(speed > MinSpeed)
                speed --;//speed decreased
            else
                speed = MinSpeed;//set to minimum speed
            //clear the GUI display
            GUI_Clear();
            //clear one-shot flag
            Btn_OneShotClear(0x08);
        }
        if (speedctrl_mode == 0x00) {
            /* Step motor output */
            if(speed)
                speedCTL = 1000/speed; //actual speed is (1000/speed)
            else
                speedCTL = 0;
        }
        else{
            speedCTL = ADC_GetVR()*8;
        }


        /* Print ADC value */
        sprintf(ADC_value_buf, "ADC value : %03d", ADC_GetVR());
        Display_buf(ADC_value_buf, 1, 1);
        /* Print Sensor temperature */
        sprintf(M487sensor_temp_value_buf, "M487sensor_temp : %2.1f", ADC_GetM487Temperature());
        Display_buf(M487sensor_temp_value_buf, 1, 40);
        /* Print Thermistor temperature */
        sprintf(thermistor_temp_value_buf, "ThermistorTemp : %d", ADC_ConvThermistorTempToReal());
        Display_buf(thermistor_temp_value_buf, 1, 79);
        /* write motor state buffer */
        sprintf(speed_buf,"Speed : %02d rpm" , speed*6);//6~102
        Display_buf(speed_buf, 1, 118);
        /* write motor speed mode buffer */
        sprintf(mode_buf,"Mode : %d" , speedctrl_mode);
        Display_buf(mode_buf, 1, 160);

        /* Drivers */
        /* Motor Task */
        StepMtr_Task(dir, speedCTL);
        /* Get ADC value */
        ADC_Task();
        /* Scan button*/
        BTN_task();

    }
}
```

II. Program Flow







III. Thoughts

The experiment this time involved utilizing C language in conjunction with a stepper motor to achieve Analog Interface Control. Having already experimented with controlling stepper motors using C language in the previous session, this experiment provided an opportunity to further explore this field. Utilizing the same circuit board we soldered in the previous experiment, we aimed to become familiar with the embedded program structure for analog interface control and design a structured program for multiple mode control in embedded programming.

The familiarity with controlling stepper motors using C language from the previous experiment served as a solid foundation for this task. However, delving into analog interface control presented its own set of challenges. We needed to understand how to integrate analog signals into our program effectively, ensuring smooth and precise control over the stepper motor.

One of the highlights of this experiment was designing a structured program for multiple mode control in embedded programming. This required careful planning and organization of code to accommodate different control modes while maintaining efficiency and readability.

Overall, this experiment provided a valuable learning experience, allowing us to deepen our understanding of embedded programming concepts and further hone our skills in C language. By successfully completing Analog Interface Control and designing a structured program for multiple mode control, we gained valuable insights into the complexities of embedded systems and the importance of structured programming in such contexts.