Documentation

Embedded Controller - STM32F411 Driver Library

Written by: Han TaeGeon

Program: C/C++

IDE/Compiler: Keil uVision 5

OS: Win11

MCU: STM32F411RE, Nucleo-64

GPIO Digital InOut

Header File

```
#include "ecGPIO.h"
```

```
#include "ecRCC.h"

// MODER
#define INPUT  0x00
#define OUTPUT 0x01
#define AF     0x02
#define ANALOG 0x03

// IDR & ODR
#define HIGH 1
#define LOW  0

// PUDR
#define EC_NONE 0
#define  EC_PU 1
#define EC_PD 2

// OTYPER
#define EC_PUSH_PULL 0
#define EC_OPEN_DRAIN 1

// OSPEED
#define EC_LOW 0
#define EC_MEDIUM 1
#define EC_FAST 2
#define EC_HIGH 3

// PIN
#define LED_A0 0
#define LED_A1 1
#define LED_B0 0
#define LED_C1 1
```

```
#define LED_PB9     9
#define LED_PA5     5
#define LED_PA6     6
#define LED_PA7     7
#define LED_PB6     6
#define LED_PC7     7
#define LED_PA9     9
#define LED_PA8     8
#define LED_PB10    10
#define BUTTON_PIN 13
#define LED_PIN 5

#define Direction_PIN 2
#define PWM_PIN PA_0

void GPIO_init(GPIO_TypeDef *Port, int pin, unsigned int mode);
void GPIO_write(GPIO_TypeDef *Port, int pin, unsigned int Output);
int  GPIO_read(GPIO_TypeDef *Port, int pin);
void GPIO_mode(GPIO_TypeDef* Port, int pin, unsigned  int mode);
void GPIO_ospeed(GPIO_TypeDef* Port, int pin, unsigned int speed);
void GPIO_otype(GPIO_TypeDef* Port, int pin, unsigned int type);
void GPIO_pupd(GPIO_TypeDef* Port, int pin, unsigned int pupd);

void sevensegment_init(void);
void sevensegment_decoder(uint8_t  num);

void sevensegment_display_init(void);
void sevensegment_display(uint8_t  num);
void LED_UP(uint8_t  num);

void LED_toggle();

void mcu_init(GPIO_TypeDef* Port, int pin);
```

GPIO_init()

Initializes GPIO pins with default setting and Enables GPIO Clock. Mode: In/Out/AF/Analog

```
void GPIO_init(GPIO_TypeDef *Port, int pin, int mode);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **mode**: INPUT(0), OUTPUT(1), AF(02), ANALOG (03)

Example code

```
GPIO_init(GPIOA, 5, OUTPUT);
GPIO_init(GPIOC, 13, INPUT); //GPIO_init(GPIOC, 13, 0);
```

GPIO_mode()

Configures GPIO pin modes: In/Out/AF/Analog

```
void GPIO_init(GPIO_TypeDef *Port, int pin, int mode);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **mode**: INPUT (0), OUTPUT (1), AF(02), ANALOG (03)

Example code

```
GPIO_mode(GPIOA, 5, OUTPUT);
```

GPIO_write()

Write the data to GPIO pin: High, Low

```
write(GPIO_TypeDef *Port, int pin, int output);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **output**: LOW(0), HIGH(1)

Example code

```
GPIO_write(GPIOA, 5, 1);  // 1: High
```

GPIO_read()

Read the data from GPIO pin

```
int  GPIO_read(GPIO_TypeDef *Port, int pin);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15

Example code

```
GPIO_read(GPIOC, 13);
```

GPIO_ospeed()

Configures output speed of GPIO pin : Low, Mid, Fast, High

```
void GPIO_ospeed(GPIO_TypeDef* Port, int pin, int speed);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **speed**: LOW_SPEED(0), MID_SPEED(1), FAST_SPEED(2) , HIGH_SPEED(3)

Example code

```
GPIO_ospeed(GPIOA, 5, 2);  // 2: FAST_SPEED
```

GPIO_otype()

Configures output type of GPIO pin: Push-Pull / Open-Drain

```
void GPIO_otype(GPIO_TypeDef* Port, int pin, int type);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **type**: PUSH_PULL(0), OPEN_DRAIN(1)

Example code

```
GPIO_otype(GPIOA, 5, 0);  // 0: Push-Pull
```

GPIO_pupdr()

Configures Pull-up/Pull-down mode of GPIO pin: No Pull-up, Pull-down/ Pull-up/ Pull-down/ Reserved

```
void GPIO_pupdr(GPIO_TypeDef* Port, int pin, int pupd);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15
- **pupd**: NO_PUPD(0), PULL_UP(1), PULL_DOWN(2), RESERVED(3)

Example code

```
GPIO_pupdr(GPIOA, 5, 0);  // 0: No Pull-up, Pull-down
```

sevensegment_init()

Initializes sevensegment number with GPIO OUTPUT

```
void sevensegment_init(void);
```

Example code

```
sevensegment_display_init();
```

sevensegment_decoder()

Configures sevensegment with decoder

```
void sevensegment_decoder(uint8_t  num);
```

Parameters

- num: number of rows

Example code

```
sevensegment_decoder(cnt % 10);
```

sevensegment_display_init()

Initializes sevensegment number with GPIO OUTPUT

```
void sevensegment_display_init(void);
```

Example code

```
sevensegment_display_init();
```

sevensegment_display()

Sevensegment numbers are displayed

```
void sevensegment_display(uint8_t  num);
```

Parameters

- num: number of rows

Example code

```
sevensegment_display(cnt % 10);
```

LED_UP()

Configures rising number by using LED

```
void LED_UP(uint8_t  num);
```

Parameters

- num: desired number to display

Example code

```
LED_UP(cnt % 10);
```

LED_toggle()

Configures blinking LED

```
void LED_toggle();
```

Example code

```
LED_toggle();
```

mcu_init()

Initialize basic setting : GPIO_pupd(EC_NONE), GPIO_otype(EC_PUSH_PULL), GPIO_ospeed(EC_FAST)

```
void mcu_init(GPIO_TypeDef* Port, int pin);
```

Parameters

- **Port:** Port Number, GPIOA~GPIOH
- **pin**: pin number (int) 0~15

Example code

```
mcu_init(GPIOA, LED_PIN);
```

Example Code

```
#include "ecSTM32F411.h"

#define LED_PIN 5
#define BUTTON_PIN 13
```

```
// Initialiization
void setup(void) {
    RCC_PLL_init();
    SysTick_init();
    GPIO_init(GPIOA, LED_PIN, OUTPUT);
}

int main(void) {
    setup();

    while(1){
        delay_ms(500);
        GPIO_write(GPIOA, LED_PIN, LOW);
        delay_ms(500);
        GPIO_write(GPIOA, LED_PIN, HIGH);
    }
}
```

EXTI

Header File

```
#include "stm32f411xe.h"

#define FALL 0
#define RISE 1
#define BOTH 2

void EXTI_init(GPIO_TypeDef *Port, int pin, int trig, int priority);
void EXTI_enable(uint32_t pin);     // mask in IMR
void EXTI_disable(uint32_t pin);     // unmask in IMR
uint32_t is_pending_EXTI(uint32_t pin);
void clear_pending_EXTI(uint32_t pin);
```

EXTI_init()

Initializes EXTI pins with default setting : FALL, RISE, BOTH

```
void EXTI_init(GPIO_TypeDef *Port, int pin, int trig, int priority);
```

Parameters

- Port: Port Number, GPIOA~GPIOH
- pin: pin number (int) 0~15
- trig: Configure Trigger edge
- priority: set priority of conducting code

Example code

```
EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
```

EXTI_enable()

Mask pin in IMR

```
void EXTI_enable(uint32_t pin)
```

Parameters

- pin: pin number (int) 0~15

Example code

```
EXTI_enable(BUTTON_PIN);
```

EXTI_disable()

Unmask pin in IMR

```
void EXTI_disable(uint32_t pin)
```

Parameters

- pin: pin number (int) 0~15

Example code

```
EXTI_disable(BUTTON_PIN);
```

is_pending_EXTI()

Check EXTI pending

```
uint32_t is_pending_EXTI(uint32_t pin);
```

Parameters

- pin: pin number (int) 0~15

Example code

```
if (is_pending_EXTI(BUTTON_PIN) && time<1000000) {
        LED_toggle();
}
```

clear_pending_EXTI

Clear EXTI pending(Make 0)

```
void clear_pending_EXTI(uint32_t pin);
```

Parameters

- pin: pin number (int) 0~15

Example code

```
clear_pending_EXTI(BUTTON_PIN);
```

Example code

```
//EXTI for Pin 13
void EXTI15_10_IRQHandler(void) {
    time++;
    if (is_pending_EXTI(BUTTON_PIN) && time<1000000) {
        LED_toggle();
        sevensegment_display(count % 10);
        //clear_pending_EXTI(BUTTON_PIN);
        count++;
        if(count > 9) count = 0;
    }
}

void setup(){
    RCC_PLL_init();
    SysTick_init();
    EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);
}
```

Systick

Header File

```
#include "ecSTM32F411.h"

extern volatile uint32_t msTicks;
void SysTick_init(void);
void SysTick_Handler(void);
void SysTick_counter();
void delay_ms(uint32_t msec);
void SysTick_reset(void);
uint32_t SysTick_val(void);
void SysTick_enable(void);
void SysTick_disable(void);
```

SysTick_init()

Initializes SysTick with default setting : set priority, enable interrupt in NVIC

```
void SysTick_init(void);
```

Example Code

```
void setup(void){
    SysTick_init();
}
```

SysTick Handler()

Acts as an interrupt service routine (ISR) for the SysTick timer interrupt.

```
void SysTick_Handler(void);
```

Example Code

```
void SysTick_Handler(void){}
```

SysTick_counter()

Increase msTicks by one.

```
void SysTick_counter();
```

Example Code

```
SysTick_counter();
```

delay_ms()

Generates a delay for a specified time period.

```
void delay_ms(uint32_t msec);
```

Parameter

- msec: The larger the number, the longer the delay

Example Code

```
delay_ms(1000);
```

SysTick_reset()

Configures the value to 0

```
void SysTick_reset(void);
```

Example Code

```
while(){
    delay_ms(1000);
    SysTick_reset();
}
```

SysTick_val()

Assign a value to SysTick

```
uint32_t SysTick_val(void);
```

Example Code

```
SysTick_val() = 1;
```

SysTick_enable()

Make to enable interrupt in NVIC

```
void SysTick_enable(void);
```

Example Code

```
SysTick_enable();
```

SysTick_disable()

Make to disable interrupt in NVIC

```
void SysTick_disable(void);
```

Example Code

```
SysTick_disable();
```

Example Code

```
int main(void) {
    // Initialiization --------------------------------------------------------
        setup();//EXTI for Pin 13

    // Inifinite Loop ---------------------------------------------------------
    while(1){
        sevensegment_display(count % 10);
        delay_ms(1000);
        count++;
        if (count >9) count =0;
        SysTick_reset();
    }
}

//EXTI for Pin 13
void EXTI15_10_IRQHandler(void) {
    if (is_pending_EXTI(BUTTON_PIN)) {
        count = 0;
        sevensegment_display(count % 10);
        clear_pending_EXTI(BUTTON_PIN);
    }
}

void setup(void)
{
    RCC_PLL_init();
    SysTick_init();
    sevensegment_init();
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);
}
```

Timer

Header File

```
#include "ecSTM32F411.h"

// ICn selection according to CHn
#define FIRST 1
#define SECOND 2

// Edge Type
#define IC_RISE 0
#define IC_FALL 1
#define IC_BOTH 2

// IC Number
#define IC_1 1
#define IC_2 2
#define IC_3 3
#define IC_4 4
```

```
/* Timer Configuration */
void TIM_init(TIM_TypeDef *TIMx, uint32_t msec);
void TIM_period_us(TIM_TypeDef* TIMx, uint32_t usec);
void TIM_period_ms(TIM_TypeDef* TIMx, uint32_t msec);
void TIM_period(TIM_TypeDef* TIMx, uint32_t msec);

void TIM_UI_init(TIM_TypeDef* TIMx, uint32_t msec);
void TIM_UI_enable(TIM_TypeDef* TIMx);
void TIM_UI_disable(TIM_TypeDef* TIMx);

uint32_t is_UIF(TIM_TypeDef *TIMx);
void clear_UIF(TIM_TypeDef *TIMx);

/* Input Capture*/
void ICAP_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);
void ICAP_init(PinName_t pinName);
void ICAP_setup(PinName_t pinName, int ICn, int edge_type);
void ICAP_counter_us(PinName_t pinName, int usec);
uint32_t ICAP_capture(TIM_TypeDef* TIMx, uint32_t ICn);

uint32_t is_CCIF(TIM_TypeDef *TIMx, uint32_t CCnum);  // CCnum= 1~4
void clear_CCIF(TIM_TypeDef *TIMx, uint32_t CCnum);
```

TIM_init()

Initializes TIM with default setting and set CNT period and direction

```
void TIM_init(TIM_TypeDef *TIMx, uint32_t msec);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- msec: Adjust the desired duration by the timer.

Example Code

```
TIM_init(TIM3, 1);
```

TIM_period_us()

Configuring duration by TIM peripherals of microcontrollers with usec

```
void TIM_period_us(TIM_TypeDef* TIMx, uint32_t usec);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- usec: Allowable range for usec: 1~1,000

Example Code

```
TIM_period_us(TIM1, 200);
TIM_period_us(TIM2, 200);
```

TIM_period_ms()

Configuring duration by TIM peripherals of microcontrollers with msec

```
void TIM_period_ms(TIM_TypeDef* TIMx, uint32_t msec);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- msec: allowable range for msec: 1~2,000

Example Code

```
TIM_period_ms(TIM3,100);
```

TIM_period()

Same as TIM_period_ms()

```
void TIM_period(TIM_TypeDef* TIMx, uint32_t msec);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- msec: allowable range for msec: 1~2,000

Example Code

```
TIM_period(TIM3,100);
```

TIM_UI_init()

Initializes TIM with default setting, enable IRQ and set priority

```
void TIM_UI_init(TIM_TypeDef* TIMx, uint32_t msec);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- msec: allowable range for msec: 1~2,000

Example Code

```
TIM_UI_init(TIM3, 1);
```

TIM_UI_enable()

Enable Timer Update Interrupt

```
void TIM_UI_enable(TIM_TypeDef* TIMx);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)

Example Code

```
TIM_UI_enable(TIM4);
```

TIM_UI_disable()

Disable Timer Update Interrupt

```
void TIM_UI_disable(TIM_TypeDef* TIMx);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)

Example Code

```
TIM_UI_disable(TIM4);
```

is_UIF()

Configure to update Interrupt

```
uint32_t is_UIF(TIM_TypeDef *TIMx);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)

Example Code

```
if(is_UIF(TIM4)){                     // Update interrupt
        uint32_t ovf_cnt = 0;                                 // overflow
count
        clear_UIF(TIM4);                                  // clear update
interrupt flag
    }
```

clear_UIF()

clear interrupt (make 0)

```
void clear_UIF(TIM_TypeDef *TIMx);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)

Example Code

```
clear_UIF(TIM3);
```

ICAP_pinmap()

Used in the functions input capture and compare

```
void ICAP_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);
```

Parameter

- pinName: Port and Pin number
- TIMx: It can be used as a pin (TIM1~TIM4)
- chN: Channel

ICAP_init()

Initializes TIM with default setting

```
void ICAP_init(PinName_t pinName);
```

Parameter

- pinName: Port and Pin number

Example Code

```
ICAP_init(ECHO);
```

ICAP_setup()

Configure Selecting TIx-ICy and Edge Type

```
void ICAP_setup(PinName_t pinName, int ICn, int edge_type);
```

Parameter

- pinName: Port and Pin number
- ICn: Capture channel

- edge_type: Rising edge/ Falling edge

Example Code

```
ICAP_setup(ECHO, 1, IC_RISE);
ICAP_setup(ECHO, 2, IC_FALL);
```

ICAP_counter_us()

Time span for one counter step by usec

```
ICAP_counter_us(PinName_t pinName, int usec);
```

Parameter

- pinName: Port and Pin number
- usec: allowable range for usec: 1~1,000

Example Code

```
ICAP_counter_us(ECHO, 10);
```

ICAP_capture()

Not recommended, rather use

```
uint32_t ICAP_capture(TIM_TypeDef* TIMx, uint32_t ICn);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- ICn: Capture channel

Example Code

```
ICAP-capture(TIM3, IC_1);
```

is_CCIF()

This is used in the interrupt handler to check if the interrupt flag is on.

```
uint32_t is_CCIF(TIM_TypeDef *TIMx, uint32_t CCnum);  // CCnum= 1~4
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- CCnum: CCIF channel number

Example Code

```
    if(is_CCIF(TIM4, 1)){                                   // TIM4_Ch1 (IC1)
Capture Flag. Rising Edge Detect
        time1 = TIM4->CCR1;                                 // Capture TimeStart
        clear_CCIF(TIM4, 1);                 // clear capture/compare interrupt
flag
    }
```

clear_CCIF()

It is used to clear the interrupt flag

```
void clear_CCIF(TIM_TypeDef *TIMx, uint32_t CCnum);
```

Parameter

- TIMx: It can be used as a pin (TIM1~TIM4)
- CCnum: CCIF channel number

Example Code

```
clear_CCIF(TIM4, 1);
```

Example Code

```
void TIM4_IRQHandler(void){
    if(is_UIF(TIM4)){                     // Update interrupt
        uint32_t ovf_cnt = 0;                                   // overflow
count
        clear_UIF(TIM4);                                    // clear update
interrupt flag
    }

    if(is_CCIF(TIM4, 1)){                                   // TIM4_Ch1 (IC1)
Capture Flag. Rising Edge Detect
        time1 = TIM4->CCR1;                                 // Capture TimeStart
        clear_CCIF(TIM4, 1);                 // clear capture/compare interrupt
flag
    }

    else if(is_CCIF(TIM4, 2)){                                  // TIM4_Ch2
(IC2) Capture Flag. Falling Edge Detect
        time2 = TIM4->CCR2;                                 // Capture TimeEnd
        timeInterval = (time2-time1+(TIM4->ARR+1)*ovf_cnt)*1e-2;    // (10us *
counter pulse -> [msec] unit) Total time of echo pulse
        ovf_cnt = 0;                          // overflow reset
        clear_CCIF(TIM4,2);                                 // clear
capture/compare interrupt flag
    }
}
```

PWM

Header File

```
#include "stm32f411xe.h"

/* PWM Configuration using PinName_t Structure */
/* PWM initialization */
// Default: 84MHz PLL, 1MHz CK_CNT, 50% duty ratio, 1msec period
void PWM_init(PinName_t pinName);
void PWM_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);

/* PWM PERIOD SETUP */
// allowable range for msec:  1~2,000
void PWM_period(PinName_t pinName,  uint32_t msec);
void PWM_period_ms(PinName_t pinName,  uint32_t msec);  // same as PWM_period()
// allowable range for usec:  1~1,000
void PWM_period_us(PinName_t pinName, uint32_t usec);

/* DUTY RATIO SETUP */
// High Pulse width in msec
void PWM_pulsewidth(PinName_t pinName, uint32_t pulse_width_ms);
void PWM_pulsewidth_ms(PinName_t pinName, uint32_t pulse_width_ms);  // same as
void PWM_pulsewidth
void PWM_pulsewidth_us(PinName_t pinName, uint32_t pulse_width_us);
// Duty ratio 0~1.0
void PWM_duty(PinName_t pinName, float duty);
```

PWM_init()

Initializes PWM with default setting : 84MHz PLL, 1MHz CK_CNT, 50% duty ratio, 1msec period

```
void PWM_init(PinName_t pinName);
```

Parameter

- pinName: Port and Pin number

Example Code

```
PWM_init(TRIG);         // PA_6: Ultrasonic trig pulse
```

PWM_pinmap()

```
void PWM_pinmap(PinName_t pinName, TIM_TypeDef **TIMx, int *chN);
```

Parameter

- pinName: Port and Pin number
- TIMx: It can be used as a pin (TIM1~TIM4)
- chN: Channel

PWM_period()

Configuring duration by PWM peripherals of microcontrollers with msec

```
void PWM_period(PinName_t pinName,  uint32_t msec);
```

Parameter

- pinName: Port and Pin number
- msec: allowable range for msec: 1~2,000

Example Code

```
PWM_period(TRIG, 50);
```

PWM_period_ms()

same as PWM_period()

```
void PWM_period_ms(PinName_t pinName,  uint32_t msec);
```

Parameter

- pinName: Port and Pin number
- msec: allowable range for msec: 1~2,000

Example Code

```
PWM_period_MS(TRIG, 50);
```

PWM_period_us()

Configuring duration by PWM peripherals of microcontrollers with Usec

```
void PWM_period(PinName_t pinName,  uint32_t msec);
```

```
void PWM_period_us(PinName_t pinName, uint32_t usec);
```

Parameter

- pinName: Port and Pin number
- usec: allowable range for usec: 1~1,000

Example Code

```
PWM_period_us(TRIG, 50000);    // PWM of 50ms period. Use period_us()
```

PWM_pulsewidth()

Control PWM pulsewidth in milli-second

```
void PWM_pulsewidth(PinName_t pinName, uint32_t pulse_width_ms);
```

Parameter

- pinName: Port and Pin number
- pulse_width_ms: Milli-second pulsewidth

Example Code

```
PWM_pulsewidth(PWM_PIN,(float)duty/period);
```

PWM_pulsewidth_ms()

same as void PWM_pulsewidth

```
void PWM_pulsewidth_ms(PinName_t pinName, uint32_t pulse_width_ms);
```

Parameter

- pinName: Port and Pin number
- pulse_width_ms: Milli-second pulsewidth

Example Code

```
PWM_pulsewidth_ms(PWM_PIN, 50);
```

PWM_pulsewidth_us()

Control PWM pulsewidth in micro-second

```
void PWM_pulsewidth_us(PinName_t pinName, uint32_t pulse_width_us);
```

Parameter

- pinName: Port and Pin number
- pulse_width_us: Micro-second pulsewidth

Example Code

```
PWM_pulsewidth_ms(PWM_PIN, 50);
```

PWM_duty()

Duty is between 0 and 1 and is adjustable as a function.

```
void PWM_duty(PinName_t pinName, float duty);
```

Parameter

- pinName: Port and Pin number
- duty: Duty ratio 0~1.0

Example Code

```
PWM_duty(PWM_PIN,(float)duty);
PWM_duty(PWM_PIN,(float)0);
```

Example Code

```
int main(void) {
    // Initialiization --------------------------------------------------------
  setup();

    // Inifinite Loop ---------------------------------------------------------
    while(1){
        if(State == 1){
        PWM_duty(PWM_PIN,(float)duty);
        }
        else if(State == -1){
            PWM_duty(PWM_PIN,(float)0);
        }
    }
}

    void TIM3_IRQHandler(void){
        if(is_UIF(TIM3)){           // Check UIF(update interrupt flag)
            count++;
            if (count > 2000){
                if(State == 1){
                    flag *= -1;
                    if(flag == 1)   {duty = 0.25;}
                    else if(flag == -1) {duty = 0.75;}
                count = 0;
                }
            }
            clear_UIF(TIM3);        // Clear UI flag by writing 0
        }
    }
```

Stepper

Header File

```
#include "ecSTM32F411.h"

//State mode
#define HALF 0
#define FULL 1

/* Stepper Motor */
//stepper motor function
typedef struct{
    GPIO_TypeDef *port1;
    int pin1;
    GPIO_TypeDef *port2;
    int pin2;
    GPIO_TypeDef *port3;
    int pin3;
    GPIO_TypeDef *port4;
    int pin4;
    uint32_t _step_num;
} Stepper_t;

void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2,
GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);
void Stepper_setSpeed(long whatSpeed, uint32_t mode);
void Stepper_step(uint32_t steps, uint32_t direction, uint32_t mode);
void Stepper_stop(void);
void Stepper_pinOut (uint32_t state, uint32_t mode);
```

Stepper_init()

Initializes PWM with default setting : GPIO, myStepper

```
void Stepper_init(GPIO_TypeDef* port1, int pin1, GPIO_TypeDef* port2, int pin2,
GPIO_TypeDef* port3, int pin3, GPIO_TypeDef* port4, int pin4);
```

Parameter

- port1: A port
- pin1: A pin
- port2: B port
- pin2: B pin
- port3: C port
- pin3: C pin
- port4: D port
- pin4: D pin

Example Code

```
Stepper_init(GPIOB, 10, GPIOB, 4, GPIOB, 5, GPIOB, 3);      // Stepper GPIO pin
initialization
```

Stepper_setSpeed()

Set the stepper motor to RPM

```
void Stepper_setSpeed(long whatSpeed, uint32_t mode);
```

Parameter

- whatSpeed: RPM
- mode: FULL, HALF

Example Code

```
Stepper_setSpeed(14, FULL);         // set stepper motor speed in FULL mode
```

Stepper_step()

One rotation is 2048 steps

```
void Stepper_step(uint32_t steps, uint32_t direction, uint32_t mode);
```

Parameter

- steps: Number of steps
- direction: Clock, Clockwise
- mode: FULL, HALF

Example Code

```
Stepper_step(2048, 0, FULL);        //Step 30x64 Direction 0 or 1 Mode in FULL
mode
```

Stepper_stop()

Stop the step motor

```
void Stepper_stop(void);
```

Example Code

```
Stepper_stop();
```

Stepper_pinOut()

Divide the status according to the mode (FULL, HALF)

```
void Stepper_pinOut (uint32_t state, uint32_t mode);
```

Parameter

- state: GPIO_write : Port - Pin (A, B, C, D)
- mode: FULL, HALF

Example Code

```
int main(void){
     // Initialization
    setup();
    Stepper_step(2048, 0, FULL);        //Step 30x64 Direction 0 or 1 Mode in
FULL mode
    // Infinite Loop
    while(1){;}
}

void setup(void){
    RCC_PLL_init();     // System Clock = 84MHz
    SysTick_init();     // SysTick init
    EXTI_init(GPIOC, BUTTON_PIN, FALL, 0);  // External Interrupt Setting
    GPIO_init(GPIOC, BUTTON_PIN, INPUT);        //  GPIOC BUTTON PIN
initialization
    Stepper_init(GPIOB, 10, GPIOB, 4, GPIOB, 5, GPIOB, 3);      // Stepper GPIO
pin initialization
    Stepper_setSpeed(14, FULL);         // set stepper motor speed in FULL mode
    //Stepper_setSpeed(14, HALF);           // set stepper motor speed in HALF
mode
}

void EXTI15_10_IRQHandler(void){
    if (is_pending_EXTI(BUTTON_PIN)) {       // pendig 1
        Stepper_stop();
        clear_pending_EXTI(BUTTON_PIN);         // cleared by writing '1'
    }
}
```

UART

Header File

```
#include "ecSTM32F411.h"
#include <stdio.h>
#include "string.h"

#define POL 0
#define INT 1
```

```
// You can modify this
#define BAUD_9600   9600
#define BAUD_19200  19200
#define BAUD_38400  38400
#define BAUD_57600  57600
#define BAUD_115200 115200
#define BAUD_921600 921600

// Configuration UART 1, 2 using default pins
void UART1_init(void);
void UART2_init(void);
void UART1_baud(uint32_t baud);
void UART2_baud(uint32_t baud);

// USART write & read
void USART1_write(uint8_t* buffer, uint32_t nBytes);
void USART2_write(uint8_t* buffer, uint32_t nBytes);
uint8_t USART1_read(void);
uint8_t USART2_read(void);

// RX Inturrupt Flag USART1,2
uint32_t is_USART1_RXNE(void);
uint32_t is_USART2_RXNE(void);

// private functions
void USART_write(USART_TypeDef* USARTx, uint8_t* buffer, uint32_t nBytes);
void USART_init(USART_TypeDef* USARTx, uint32_t baud);
void UART_baud(USART_TypeDef* USARTx, uint32_t baud);
uint32_t is_USART_RXNE(USART_TypeDef * USARTx);
uint8_t USART_read(USART_TypeDef * USARTx);
void USART_setting(USART_TypeDef* USARTx, GPIO_TypeDef* GPIO_TX, int pinTX,
GPIO_TypeDef* GPIO_RX, int pinRX, uint32_t baud);
void USART_delay(uint32_t us);
```

UART1_init()

Configuration UART 1 using default pins

```
void UART1_init(void);
```

Example Code

```
UART1_init();
```

UART2_init()

Configuration UART 2 using default pins

```
void UART2_init(void);
```

Example Code

```
UART2_init();
```

UART1_baud()

Configure Baud-rate of USART1

```
void UART1_baud(uint32_t baud);
```

Parameter

- baud: Baud rate of USART 1

Example Code

```
UART1_baud(BAUD_9600);
```

UART2_baud()

Configure Baud-rate of USART2

```
void UART2_baud(uint32_t baud);
```

Parameter

- baud: Baud rate of USART 2

Example Code

```
UART2_baud(BAUD_9600);
```

USART1_write()

USART1 write to Tera Term

```
void USART1_write(uint8_t* buffer, uint32_t nBytes);
```

Parameter

- buffer: String
- nBytes: Number of characters

Example Code

```
USART1_write(&PC_data,1);       // TX to USART1  (BT)
```

USART2_write()

USART2 write to Tera Term

```
void USART2_write(uint8_t* buffer, uint32_t nBytes);
```

Parameter

- buffer: String
- nBytes: Number of characters

Example Code

```
USART2_write(&PC_data,1);       // TX to USART2  (PC)
```

UART1_read()

USART1 read data from bluetooth

```
uint8_t USART1_read(void);
```

Example Code

```
BT_data = USART1_read();
```

UART2_read()

Read data from serial communication

```
uint8_t USART2_read(void);
```

Example Code

```
PC_data = USART2_read();        // RX from UART2 (PC)
```

is_USART1_RXNE()

Check Interrupt flag of USART1

```
uint32_t is_USART1_RXNE(void);
```

Example Code

```
void USART1_IRQHandler(){                // USART2 RX Interrupt : Recommended
    if(is_USART1_RXNE()){
        GPIO_init(GPIOA, LED_PIN, OUTPUT);
        if(BT_data == 'L')  GPIO_write(GPIOA, LED_PIN, 1);
        else if(BT_data == 'H') GPIO_write(GPIOA, LED_PIN, 0);
        BT_data = USART1_read();

        printf("RX: %c \r\n",BT_data); // TX to USART2(PC)
    }
}
```

is_USART2_RXNE()

Check Interrupt flag of USART2

```
uint32_t is_USART2_RXNE(void);
```

Example Code

```
void USART2_IRQHandler(){                // USART2 RX Interrupt : Recommended
    if(is_USART2_RXNE()){
        PC_data = USART2_read();         // RX from UART2 (PC)

        USART2_write(&PC_data,1);        // TX to USART2  (PC)    Echo of
keyboard typing
        USART1_write(&PC_data,1);        // TX to USART1  (BT)

    }
}
```

USART_write()

USART write to Tera Term

```
void USART_write(USART_TypeDef* USARTx, uint8_t* buffer, uint32_t nBytes);
```

Parameter

- USARTx: USART1 or USART2
- buffer: String
- nBytes: Number of characters

Example Code

```
USART_write(USART1,(uint8_t*)"MOD : ", 6);
USART_write(USART1, &mode, 1);
```

USART_init()

Configuration UART using default pins

```
void USART_init(USART_TypeDef* USARTx, uint32_t baud);
```

Parameter

- USARTx: USART1 or USART2
- baud: Baud rate of USART

Example Code

```
USART_init(USART1, BAUD_9600);
```

USART_baud()

Configure Baud-rate of USART

```
void UART_baud(USART_TypeDef* USARTx, uint32_t baud);
```

Parameter

- USARTx: USART1 or USART2
- baud: Baud rate of USART

Example Code

```
UART_baud(USART1,BAUD_9600);
```

is_USART_RXNE()

Check Interrupt flag of USART

```
uint32_t is_USART_RXNE(USART_TypeDef * USARTx);
```

Parameter

- USARTx: USART1 or USART2

Example Code

```
is_USART_RXNE();
```

USART_read()

Read data of USART

```
uint8_t USART_read(USART_TypeDef * USARTx);
```

Parameter

- USARTx: USART1 or USART2

Example Code

```
R_1 = USART_read(USART1);
R_2 = USART_read(USART2);
```

USART_setting()

Configure the setting for communication between UsSARTs

```
void USART_setting(USART_TypeDef* USARTx, GPIO_TypeDef* GPIO_TX, int pinTX,
GPIO_TypeDef* GPIO_RX, int pinRX, uint32_t baud);
```

Parameter

- USARTx: USART1 or USART2
- GPIO_TX: Communication port TX with RX
- pinTX: Communication pin TX with RX
- GPIO_RX: Communication port RX with TX
- pinRX: Communication pin TX with RX
- baud: Baud rate of USART

Example Code

```
USART_setting(USART1,GPIOA, 9, GPIOA, 10, 9600);
```

USART_delay()

Control the USART delay

```
void USART_delay(uint32_t us);
```

Parameter

- us: control delay time in micro-second

Example Code

```
void setup(void){
    RCC_PLL_init();
    SysTick_init();

    // USART2: USB serial init
    UART2_init();
    UART2_baud(BAUD_9600);

    // USART1: BT serial init
    UART1_init();
    UART1_baud(BAUD_9600);
}

int main(void){
    setup();
    printf("MCU Initialized\r\n");

    while(1){
        if (bReceive == 1){
            printf("PC_string: %s\r\n", PC_string);
            bReceive = 0;
        }
    }
}

void USART2_IRQHandler(){               // USART2 RX Interrupt : Recommended
    if(is_USART2_RXNE()){
        PC_data = USART2_read();        // RX from UART2 (PC)

        USART2_write(&PC_data,1);       // TX to USART2  (PC)    Echo of
keyboard typing
        USART1_write(&PC_data,1);       // TX to USART1  (BT)

    }
}


void USART1_IRQHandler(){               // USART2 RX Interrupt : Recommended
    if(is_USART1_RXNE()){
        GPIO_init(GPIOA, LED_PIN, OUTPUT);
        if(BT_data == 'L')  GPIO_write(GPIOA, LED_PIN, 1);
        else if(BT_data == 'H') GPIO_write(GPIOA, LED_PIN, 0);
        BT_data = USART1_read();

        printf("RX: %c \r\n",BT_data); // TX to USART2(PC)
    }
}
```

ADC

Header File

```
#include "ecSTM32F411.h"

// ADC trigmode
#define SW 0
#define TRGO 1

// ADC contmode
#define CONT 0
#define SINGLE 1

// Edge Type
#define RISE 1
#define FALL 2
#define BOTH 3

#define _DEFAULT 0

// ADC init
// Default:  one-channel mode, continuous conversion
// Default: HW trigger - TIM3 counter, 1msec
void ADC_init(PinName_t pinName);
void JADC_init(PinName_t pinName);


// Multi-Channel Scan Sequence
void ADC_sequence(PinName_t *seqCHn, int seqCHnums);
void JADC_sequence(PinName_t *seqCHn, int seqCHnums);

void ADC_start(void);
void JADC_start(void);

// flag for ADC interrupt
uint32_t is_ADC_EOC(void);
uint32_t is_ADC_OVR(void);
void clear_ADC_OVR(void);

// read ADC value
uint32_t ADC_read(void);

// Conversion mode change: CONT, SINGLE / Operate both ADC,JADC
void ADC_conversion(int convMode);
void ADC_trigger(TIM_TypeDef* TIMx, int msec, int edge);

// JADC setting
void JADC_trigger(TIM_TypeDef* TIMx, int msec, int edge);

// Private Function
void ADC_pinmap(PinName_t pinName, uint32_t *chN);
```

ADC_init()

Initializes ADC with default setting (trigmode 0 : SW, 1 : TRGO)

```
void ADC_init(PinName_t pinName);
```

Parameter

- pinName: Port and Pin number

Example Code

```
ADC_init(PB_0);
ADC_init(PB_1);
```

ADC_sequence()

```
void ADC_sequence(PinName_t *seqCHn, int seqCHnums);
```

Parameter

- seqCHn: Represents an array of pointers
- seqCHnums: Indicates the number of pins

Example Code

```
ADC_sequence(seqCHn, 2);    // ADC channel sequence setting
```

ADC_start()

Configure ADC starting

```
void ADC_start(void);
```

Example Code

```
ADC_start();
```

is_ADC_EOC()

Configure ADC interrupt flag

```
uint32_t is_ADC_EOC(void);
```

Example Code

```
if(is_ADC_EOC()){       // after finishing sequence
        if (flag==0)
            value1 = ADC_read();
        else if (flag==1)
            value2 = ADC_read();

        flag =! flag;       // flag toggle
    }
```

is_ADC_OVR()

Configure ADC overflow flag

```
uint32_t is_ADC_OVR(void);
```

Example Code

```
if(is_ADC_OVR())
    clear_ADC_OVR();
```

clear_ADC_OVR()

Configure ADC clear flag

```
void clear_ADC_OVR(void);
```

Example Code

```
if(is_ADC_OVR())
    clear_ADC_OVR();
```

ADC_read()

Configure ADC value reading

```
uint32_t ADC_read(void);
```

Example Code

```
if(is_ADC_EOC()){       // after finishing sequence
        if (flag==0)
            value1 = ADC_read();
        else if (flag==1)
            value2 = ADC_read();

        flag =! flag;       // flag toggle
    }
```

Example Code

```
int main(void) {
    // Initialiization --------------------------------------------------------
    setup();
    // Inifinite Loop ---------------------------------------------------------
    while(1){
        printf("value1 = %d \r\n",value1);
        printf("value2 = %d \r\n",value2);
        printf("\r\n");

        delay_ms(1000);
    }
}

// Initialiization
void setup(void)
{
    RCC_PLL_init();                         // System Clock = 84MHz
    UART2_init();                           // UART2 Init
    SysTick_init();                         // SysTick Init

    // ADC Init
    ADC_init(PB_0);
    ADC_init(PB_1);

    // ADC channel sequence setting
    ADC_sequence(seqCHn, 2);
}

void ADC_IRQHandler(void){
    if(is_ADC_OVR())
        clear_ADC_OVR();

    if(is_ADC_EOC()){       // after finishing sequence
        if (flag==0)
            value1 = ADC_read();
        else if (flag==1)
            value2 = ADC_read();

        flag =! flag;       // flag toggle
    }
}
```