my frame

system

- basic

```c
#define GPIOout(p)   BIT_ADDR((u32)p.reg + 12, p.bitnum)

inline u8 read_reg_bit(vul *reg, u8 bitnum){return (*reg & (1 << bitnum)) != 0;}

extern u8 sys_clock, sys_clock_pll;
void sys_init(u8 clock=8, u8 pll=9);

extern NVIC_priority default_priority, default_priority_H;
void NVIC_init(u8 channel, NVIC_priority priority);

//math
inline u32 put_bit(u32 num, u8 bits, u8 level){
        if(level) return num | (1 << bits);
        return num & (~(1 << bits));
}
inline u32 put16(u32 num, u8 bits, u32 n){
        u32 t = ~(0x0000000f << (bits * 4));
        return (num & t) | (n << (bits * 4));
}
inline u8 read_bit(u32 num, u8 b){
        return num & (1 << b);
}
inline int round(float num){
        return (int)num + (int)((num - (int)num) / 0.5);
}
inline float map(float x, float xmin, float xmax, float ymin, float ymax){
        return (x - xmin) / (xmax - xmin) * (ymax - ymin) + ymin;
}
/*inline u32 abs(int a){
        return (a >= 0)? a : -a;
}*/

//string
int to_int(u8* s, u8 len);
float to_float(u8* s, u8 len);

//timer
u32 millis();
u32 micros();
void delay(u32 ms);
void delay_us(u32 us);

void resetPinMode(pin p, IO_mode mode);
void pinMode(pin p, IO_mode mode, IO_level level=FLOAT);
```

- wdg

```
void iwdg_init(float tout_ms); //0.1 ~ 26000 ms
void iwdg_feed();

//but there's still interrupt issues
void wwdg_init(float tmin_ms, float tmax_ms, u8 EWI_enable=0); //0.113ms ~ 58.25ms
void wwdg_feed();
```

- exti

```
//same pin num share 1 interrupt, like PA1,PB1,PC1...
void attach_ITR(pin p, detect_mode dmode, void (*f)(void), NVIC_priority priority=default_priori
```

- timer

```
class timer{
    timer(TIM_TypeDef *tim);

        void init(tim_mode mode=continuous);
    void attach_ITR(void (*f)(void), tim_event event=update, NVIC_priority priority=default_pric

    void set_direction(u8 dir); //dir 0:up, 1:down
    void set_frequency(u32 f); //for not so accurate application
    void set_frequency(u16 arr, u16 psc);
    void set_channel_mode(TIM_CHx ch, tim_channel_mode mode, u8 pin_remap=0, u8 polarity=0);//ch
    void set_input_filter(TIM_CHx ch, tim_input_filter input_filter, u8 input_prescaler=0); //ir
    void set_polarity(TIM_CHx ch, u8 polarity); //0:none-inverted, output active high, input ris
    void set_pulsewidth(TIM_CHx ch, float pw); //pw:0 ~ 1, first set frequency because pw depenc

    void set_master_out(master_out_source m);
    /*ITR | 0 | 1 | 2 | 3
    *-----|---|---|---|---
    *TIM2 | 1 | 8 | 3 | 4
    *TIM3 | 1 | 2 | 5 | 4
    *TIM4 | 1 | 2 | 3 | 8
    *TIM5 | 2 | 3 | 4 | 8
    (slave_in_trigger_mapping)*/
    void set_slave_in(slave_in_source s);

    void enable_channel(TIM_CHx ch);
    void disable_channel(TIM_CHx ch);
        void enable();
    void disable();

    u16 get_cnt();
        u16 get_ccr(TIM_CHx ch);

};
```

- dma

```
class DMA_CH{
    DMA_CH(DMA_Channel_TypeDef* DMA_CHx);

    void init(u32 paddr,
        u32 maddr,
        dma_dir dir,
        u16 data_len,
        u8 pinc,
        u8 minc,
        data_size psize=bit_8,
        data_size msize=bit_8,
        u8 m2m=0,
                u8 circ=0,
        dma_priority priority=priority_medium);

        void attach_ITR(void (*f)(void), NVIC_priority priority=default_priority);
        void config(u32 paddr, u32 maddr, u16 data_len);
    void set_data_len(u16 len);

    void enable();
        void enable(u16 len);
        u8 transmitt_complete();
    void disable(); //auto reload counter

};
```

- adc

```
class ADC{
    ADC(ADC_TypeDef* adc);

    void init(u8 continuous=0, u8 dma_en=0);
    void attach_ITR(void (*f)(void), NVIC_priority priority=default_priority_H);
    void config_channel(u8 ch, pin p, adc_sample_t t);
    void add_scan_channel(u8 ch, pin p, adc_sample_t t);

    //single
    float read(u8 ch);
    float read_aver(u8 ch, u8 n);

    float value();
};
//TODO
```

- dac

```
class DAC_CH{
    DAC_CH(u8 dacn);

    void init();

    void set_waveform(u16* buf, u32 len, u32 freq, timer& tim);

    void write(u16 val); //0~4095

    void set_trigger(trig_sel tsel);
};
```

communication

- uart

```
class UART {
        UART(USART_TypeDef *uart=USART1);

        //recommend:
        //RECV_BY_TIME_SEPRAITON + rx:DMA
        //RECV_BY_LINE_ENDING + rx:ITR
        void enable(u8 pin_remap=0, recv_mode rmode=RECV_BY_TIME_SEPRAITON, io_t rx_way=ITR, io_
        void disable();


        //RECV_BY_LINE: \r\n(support block,itr)
        //RECV_BY_TIME_SEPRATION: data packet(support block,itr,dma)
        //RECV_IN_CIRCULAR_BUF: all data in buf(block not supported)
        u8 readline(u8 *buf, u8 *len, u32 timeout=1000); //return read success=1, fail=0
        u8 read(u8 *buf, u16 len);//circular buf, return read success=1, fail=0
        void printf(const char *fmt,...);
        void write(u8 val);
        void write(u8 *buf, u32 len);
        u8 write_available();

};
```