LED Sequence Version 3.0

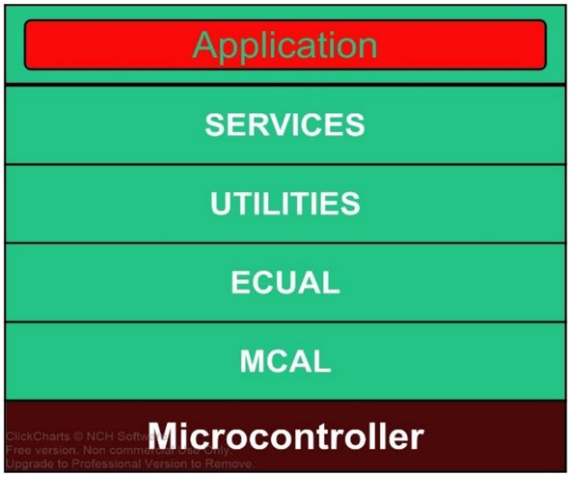
Assigned By : Mohamed Adel Abdelsalam

Firstly: Project Description:

1. *Description*
    1. ***Hardware Requirements***
        1. Four LEDs (**LED0**, **LED1**, **LED2**, **LED3**)
        2. **Two** buttons (**BUTTON0** and **BUTTON1**)
    2. ***Software Requirements***
        1. Initially, all LEDs are OFF
        2. Once **BUTTON0** is pressed, **LED0** will blink with **BLINK_1** mode
        3. Each press further will make another LED blinks **BLINK_1** mode
        4. At the **fifth press**, **LED0** will changed to be **OFF**
        5. Each **press further** will make only one LED is **OFF**
        6. This will be repeated forever
        7. The sequence is described below
            1. Initially (OFF, OFF, OFF, OFF)
            2. Press 1 (BLINK_1, OFF, OFF, OFF)
            3. Press 2 (BLINK_1, BLINK_1, OFF, OFF)
            4. Press 3 (BLINK_1, BLINK_1, BLINK_1, OFF)
            5. Press 4 (BLINK_1, BLINK_1, BLINK_1, BLINK_1)
            6. Press 5 (OFF, BLINK_1, BLINK_1, BLINK_1)
            7. Press 6 (OFF, OFF, BLINK_1, BLINK_1)
            8. Press 7 (OFF, OFF, OFF, BLINK_1)
            9. Press 8 (OFF, OFF, OFF, OFF)
            10. Press 9 (BLINK_1, OFF, OFF, OFF)
        8. When BUTTON1 has pressed the blinking on and off durations will be changed
            1. No press → **BLINK_1** mode (**ON**: 100ms, **OFF**: 900ms)
            2. First press → **BLINK_2** mode (**ON**: 200ms, **OFF**: 800ms)
            3. Second press → **BLINK_3** mode (**ON**: 300ms, **OFF**: 700ms)
            4. Third press → **BLINK_4** mode (**ON**: 500ms, **OFF**: 500ms)
            5. Fourth press → **BLINK_5** mode (**ON**: 800ms, **OFF**: 200ms)
            6. Fifth press → **BLINK_1** mode
        9. **USE EXTERNAL INTERRUPTS**

Secondly: Layered architecture:

1- Microcontroller
2- MCAL
3- ECUAL
4- UTILITIES
5- SERVICES
6- Application



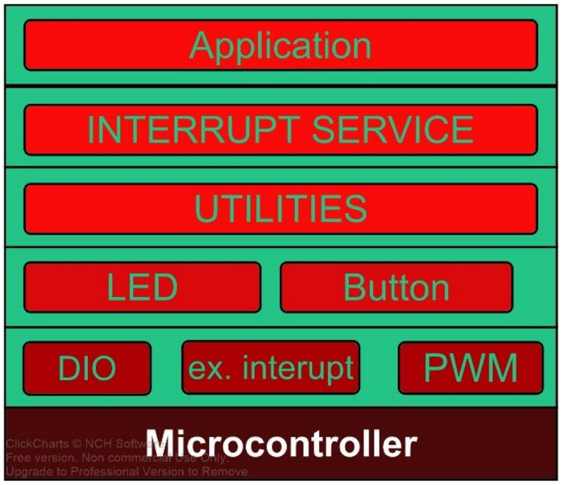
Thirdly : System modules:

1- Specify system modules/drivers:

- DIO, EX. INT, PWM, LED, BUTTON, APPLICATION

2- Assign each module to its related layer:

- By drawing



Fourthly: APIs:

1- DIO APIs:

```
void DIO_init (uint8_t portNumber,uint8_t pinNumber,uint8_t direction);
void DIO_write (uint8_t portNumber,uint8_t pinNumber,uint8_t value);
void DIO_read (uint8_t portNumber,uint8_t pinNumber,uint8_t *data);
void DIO_toggle (uint8_t portNumber,uint8_t pinNumber);
```

2- External interrupt APIs:

```
void INT_VECT(void) __attribute__ ((signal,used));
void SIE(void);
void CLI(void);
void INT_SENSE(uint8_t inerrupt_number,uint8_t sense);
void EX_INT_Enable(uint8_t inerrupt_number);
void EX_INT_Disable(uint8_t inerrupt_number);
void EX_INT0_SET_CALLBACK (void (*copyFuncptr) (void));
void EX_INT0_SET_CALLBACK (void (*copyFuncptr) (void));
void EX_INT_init(uint8_t interrupt , uint8_t sense);
```

3- LED APIs:

```
void LED_init (uint8_t port, uint8_t pin);
void LED_on (uint8_t port, uint8_t pin);
void LED_off (uint8_t port, uint8_t pin);
void LED_toggle (uint8_t port, uint8_t pin);
```

4- BUTTON APIs:

```
void BUTTON_init (uint8_t buttonport, uint8_t buttonpin);

void BUTTON_read (uint8_t buttonport, uint8_t buttonpin, uint8_t *value);
```

5- PWM APIs:

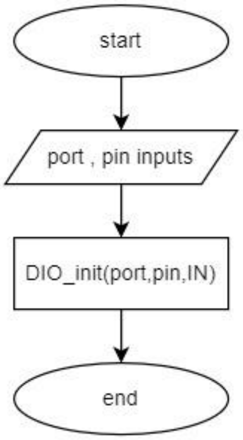
```
void PWM_Stop (void);
void PWM_init (void);
void PWM_start (uint8_t duty_percent);
void PWM_set (uint8_t duty_percent , uint8_t blinks);
```

6- APPLICATION APIs:

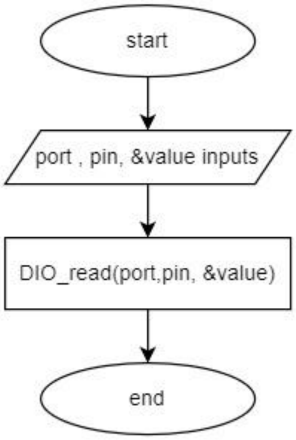
```
void APP_init(void);                         void APP_start(void);
void APP_stop(void);                       void EX_INT0_ISR(void);
                     void EX INT1 ISR (void);
```

Fifthly: Flowcharts APIs:

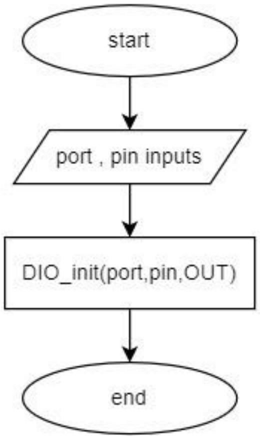
BUTTON_init(port,pin)



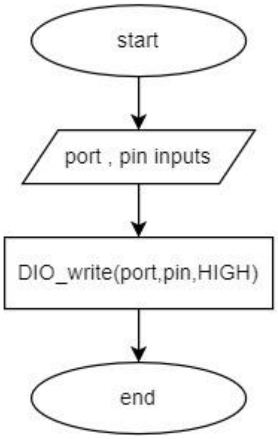
BUTTON_read(port,pin,&value)



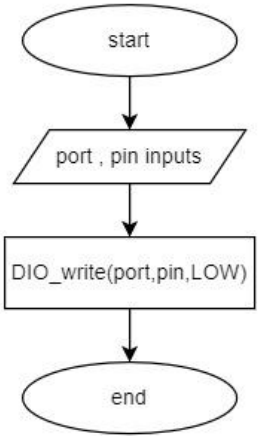
LED_init(port,pin)



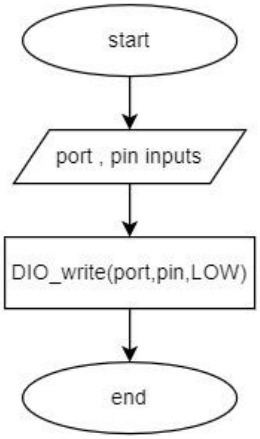
LED_on(port,pin)



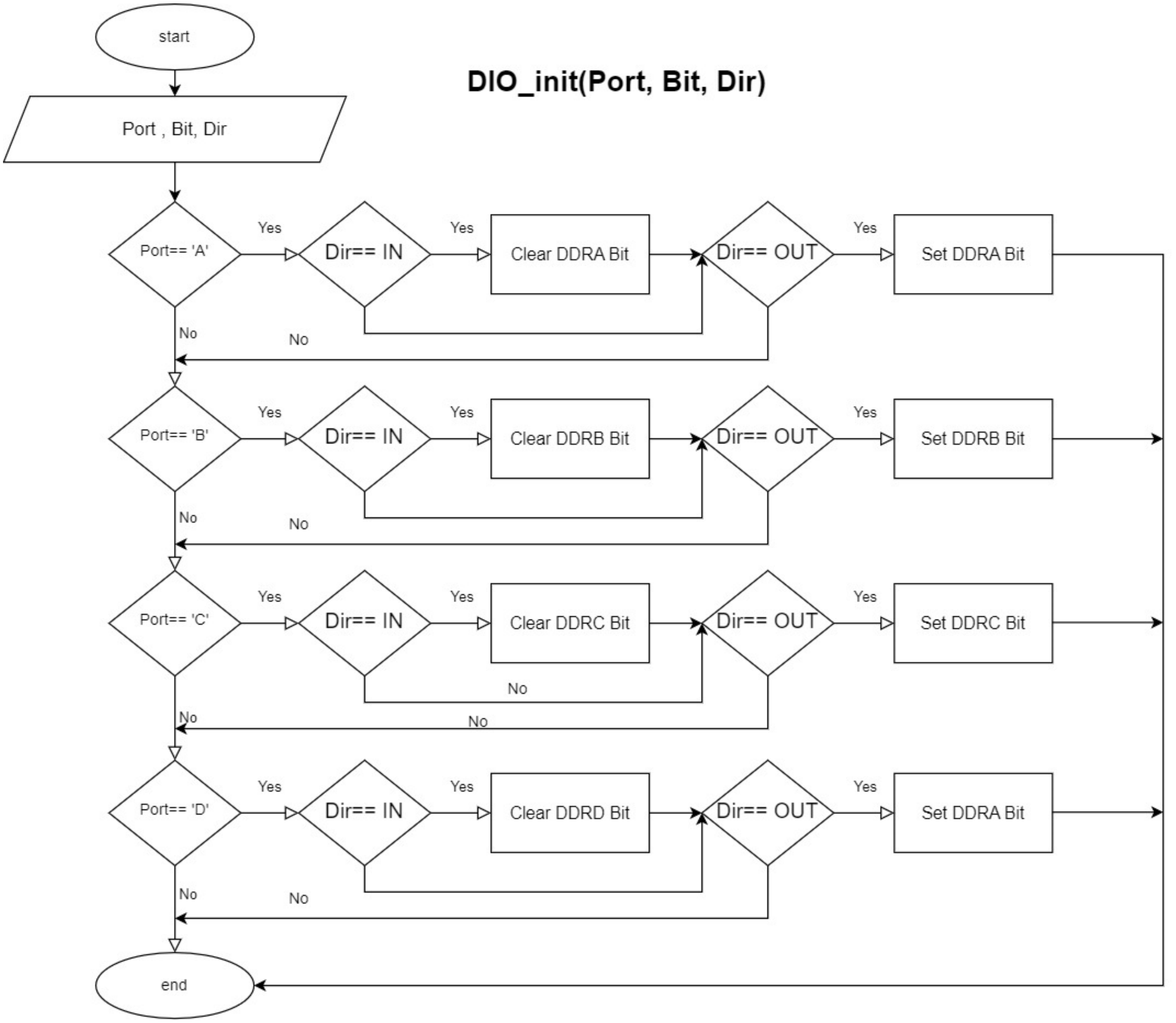
LED_off(port,pin)



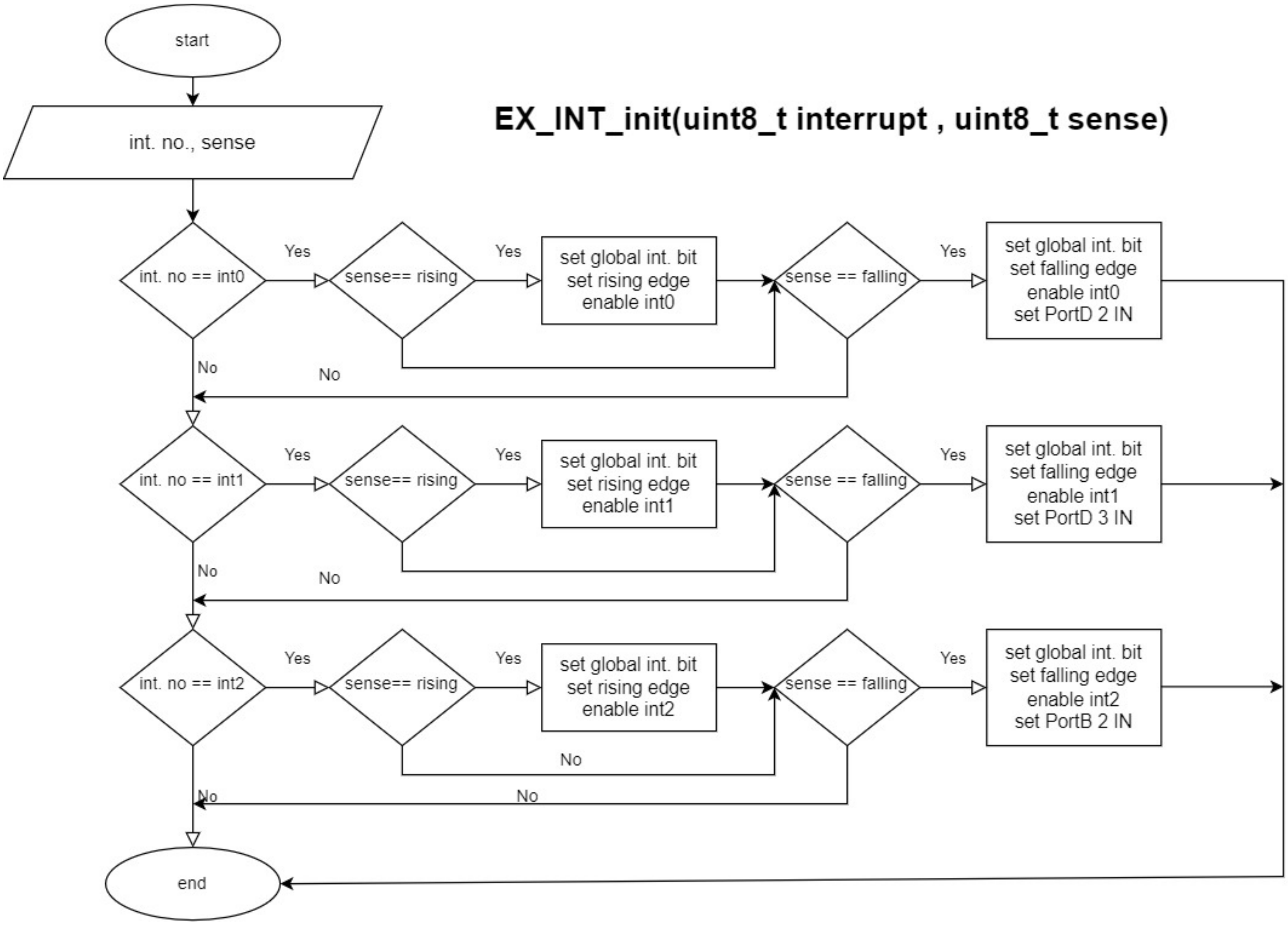
LED_toggle(port,pin)

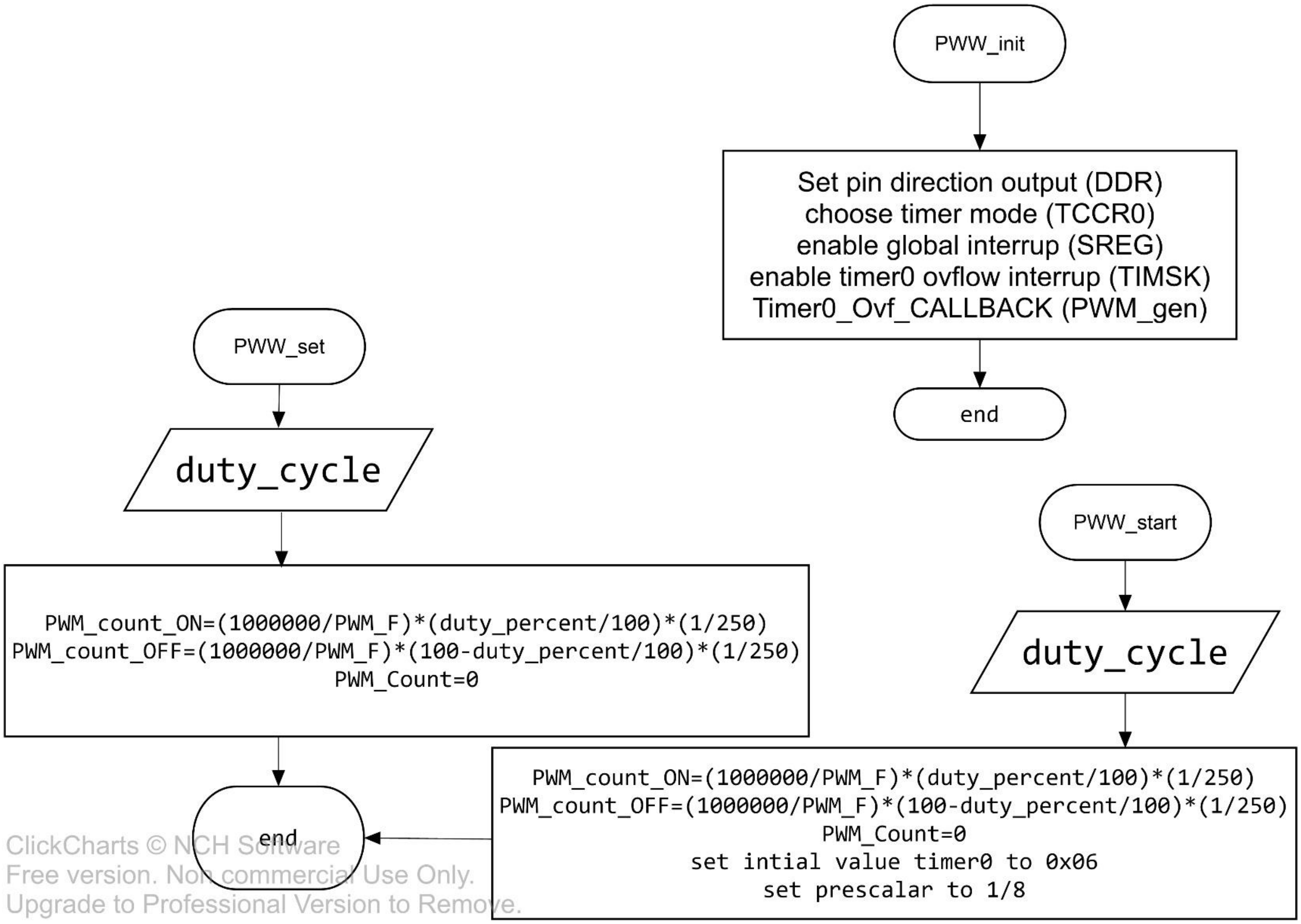


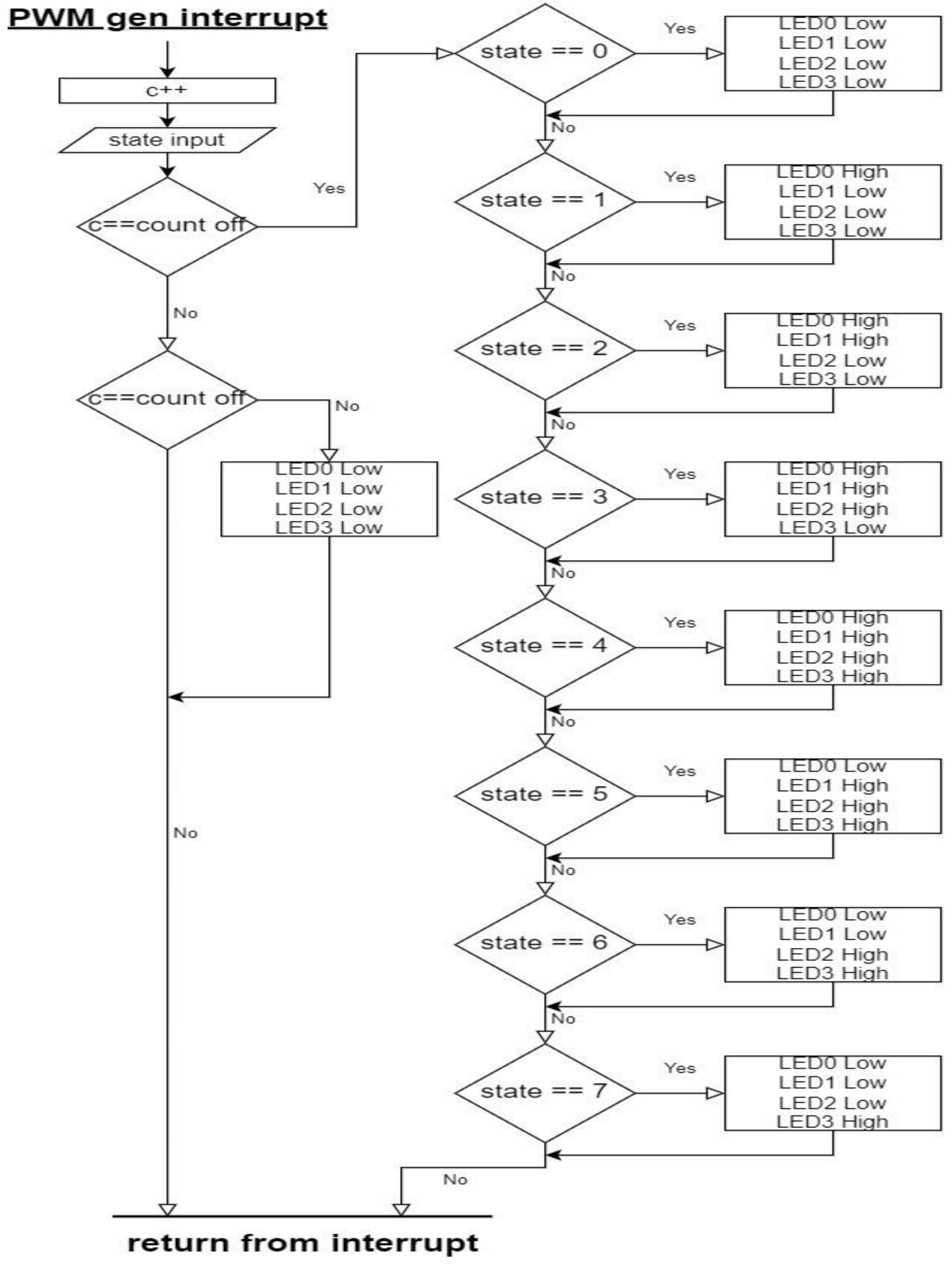
DIO_init(Port, Bit, Dir)

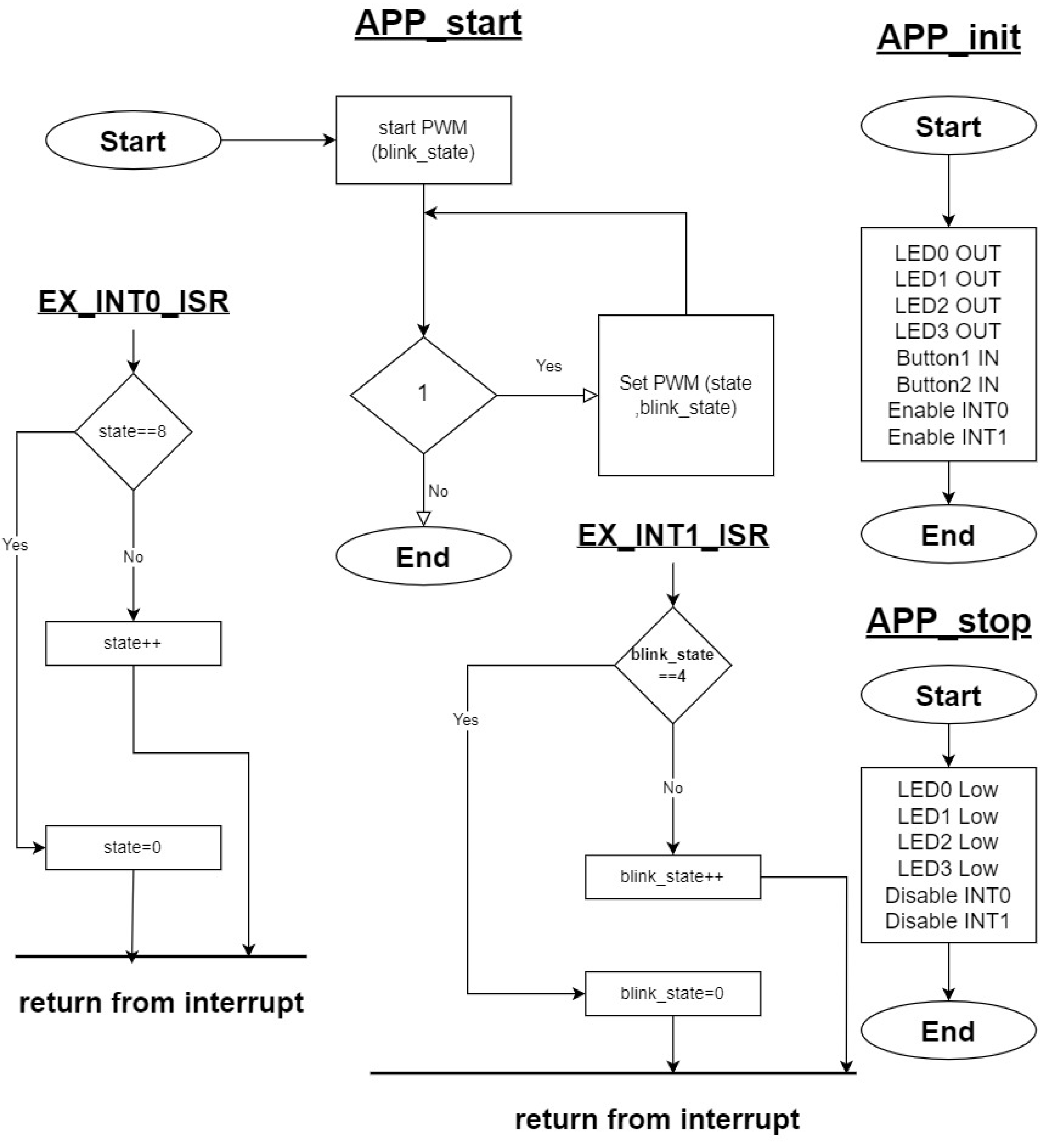


EX_INT_init(uint8_t interrupt , uint8_t sense)




PWW_init
Set pin direction output (DDR)
choose timer mode (TCCR0)
enable global interrup (SREG)
enable timer0 ovflow interrup (TIMSK)
Timer0_Ovf_CALLBACK (PWM_gen)
end
PWW_set
duty_cycle
PWM_count_ON=(1000000/PWM_F)*(duty_percent/100)*(1/250)
PWM_count_OFF=(1000000/PWM_F)*(100-duty_percent/100)*(1/250)
PWM_Count=0
end
PWW_start
duty_cycle
PWM_count_ON=(1000000/PWM_F)*(duty_percent/100)*(1/250)
PWM_count_OFF=(1000000/PWM_F)*(100-duty_percent/100)*(1/250)
PWM_Count=0
set intial value timer0 to 0x06
set prescalar to 1/8


PWM gen interrupt
c++
state input
c==count off
Yes
No
c==count off
No
LED0 Low
LED1 Low
LED2 Low
LED3 Low
No
state == 0
Yes
LED0 Low
LED1 Low
LED2 Low
LED3 Low
No
state == 1
Yes
LED0 High
LED1 Low
LED2 Low
LED3 Low
No
state == 2
Yes
LED0 High
LED1 High
LED2 Low
LED3 Low
No
state == 3
Yes
LED0 High
LED1 High
LED2 High
LED3 Low
No
state == 4
Yes
LED0 High
LED1 High
LED2 High
LED3 High
No
state == 5
Yes
LED0 Low
LED1 High
LED2 High
LED3 High
No
state == 6
Yes
LED0 Low
LED1 Low
LED2 High
LED3 High
No
state == 7
Yes
LED0 Low
LED1 Low
LED2 Low
LED3 High
No
return from interrupt


APP_start
Start
start PWM
(blink_state)
1
Yes
Set PWM (state
,blink_state)
No
End
APP_init
Start
LED0 OUT
LED1 OUT
LED2 OUT
LED3 OUT
Button1 IN
Button2 IN
Enable INT0
Enable INT1
End
EX_INT0_ISR
state==8
Yes
No
state++
state=0
return from interrupt
EX_INT1_ISR
blink_state
==4
Yes
No
blink_state++
blink_state=0
return from interrupt
APP_stop
Start
LED0 Low
LED1 Low
LED2 Low
LED3 Low
Disable INT0
Disable INT1
End