```c
#include "motor.h"
#include "io.h"



extern volatile uint32_t timestamp;



static void L298_EnablePWM(SysData_TypeDef *self);


/*  */
void L298_Init(SysData_TypeDef *self){

    L298_IN1_Set();
    L298_ENA_Set();

    LL_TIM_DisableCounter(TIM16);
    LL_TIM_DisableAllOutputs(TIM16);
    LL_TIM_CC_DisableChannel(TIM16, LL_TIM_CHANNEL_CH1);

    L298_CloseWindow(self);

    self->MotorDriverOffTimerCounter = timestamp + 10000;
}


/*  */
void L298_Process(SysData_TypeDef *self){

    if(self->MotorTimer < timestamp){

        if(self->MotorState != Stopped) L298_MotorStop(self);

        if(self->MotorDriverOffTimerCounter < timestamp){
            L298_ENA_Reset();
        }

    }


}


/*  */
void L298_OpenWindow(SysData_TypeDef *self){

    if(self->WindowState == Opened || self->PauseTimer > timestamp){
        L298_Process(self);
        return;
    }

    if(self->MotorState == Run_WOpen){

      if(self->MotorTimer < timestamp){
            L298_MotorStop(self);
            self->WindowState = Opened;
            self->PauseTimer = timestamp + self->MotorDelayTime * 1000;
      }

    }else{

        self->MotorState = Run_WOpen;
        L298_EnablePWM(self);

        L298_IN1_Reset();
        L298_ENA_Set();

        self->MotorTimer = timestamp + self->MotorRunTime * 1000;
    }
}


/*  */
void L298_CloseWindow(SysData_TypeDef *self){

    if(self->WindowState == Closed || self->PauseTimer > timestamp){
        L298_Process(self);
        return;
    }

    if(self->MotorState == Run_WClose){

      if(self->MotorTimer < timestamp){
            L298_MotorStop(self);
            self->WindowState = Closed;
            self->PauseTimer = timestamp + self->MotorDelayTime * 1000;
      }

    }else{

        self->MotorState = Run_WClose;
        L298_EnablePWM(self);

        L298_IN1_Set();
        L298_ENA_Set();

        self->MotorTimer = timestamp + self->MotorRunTime * 1000;
    }
}


/*  */
void L298_MotorStop(SysData_TypeDef *self){

    L298_IN1_Set();

    LL_TIM_DisableCounter(TIM16);
    LL_TIM_DisableAllOutputs(TIM16);
    LL_TIM_CC_DisableChannel(TIM16, LL_TIM_CHANNEL_CH1);

    self->MotorState = Stopped;

    self->MotorDriverOffTimerCounter = timestamp + 10000;
}


/*  */
static void L298_EnablePWM(SysData_TypeDef *self){

    uint16_t speed = self->MotorSpeed;

    if(self->MotorState == Run_WClose ) speed = LL_TIM_GetAutoReload(TIM16)- self->MotorSpeed;

    LL_TIM_OC_SetCompareCH1(TIM16, speed);

    LL_TIM_EnableCounter(TIM16);
    LL_TIM_EnableAllOutputs(TIM16);
    LL_TIM_CC_EnableChannel(TIM16, LL_TIM_CHANNEL_CH1);

```

```
}
```