```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);

```

```
 67          L298_IN1_Reset();
 68          L298_ENA_Set();
 69
 70          self->MotorTimer = timestamp + self->MotorRunTime * 1000;
 71      }
 72  }
 73
 74
 75  /*  */
 76  void L298_CloseWindow(SysData_TypeDef *self){
 77
 78      if(self->WindowState == Closed || self->PauseTimer > timestamp){
 79          L298_Process(self);
 80          return;
 81      }
 82
 83      if(self->MotorState == Run_WClose){
 84
 85        if(self->MotorTimer < timestamp){
 86              L298_MotorStop(self);
 87              self->WindowState = Closed;
 88              self->PauseTimer = timestamp + self->MotorDelayTime * 1000;
 89        }
 90
 91      }else{
 92
 93          self->MotorState = Run_WClose;
 94          L298_EnablePWM(self);
 95
 96          L298_IN1_Set();
 97          L298_ENA_Set();
 98
 99          self->MotorTimer = timestamp + self->MotorRunTime * 1000;
100      }
101  }
102
103
104  /*  */
105  void L298_MotorStop(SysData_TypeDef *self){
106
107      L298_IN1_Set();
108
109      LL_TIM_DisableCounter(TIM16);
110      LL_TIM_DisableAllOutputs(TIM16);
111      LL_TIM_CC_DisableChannel(TIM16, LL_TIM_CHANNEL_CH1);
112
113      self->MotorState = Stopped;
114
115      self->MotorDriverOffTimerCounter = timestamp + 10000;
116  }
117
118
119  /*  */
120  static void L298_EnablePWM(SysData_TypeDef *self){
121
122      uint16_t speed = self->MotorSpeed;
123
124      if(self->MotorState == Run_WClose ) speed = LL_TIM_GetAutoReload(TIM16)- self->
MotorSpeed;
125
126      LL_TIM_OC_SetCompareCH1(TIM16, speed);
127
128      LL_TIM_EnableCounter(TIM16);
129      LL_TIM_EnableAllOutputs(TIM16);
130      LL_TIM_CC_EnableChannel(TIM16, LL_TIM_CHANNEL_CH1);
131
```

```
132  }
133
134
135
```