EMBEDDED PROJECT

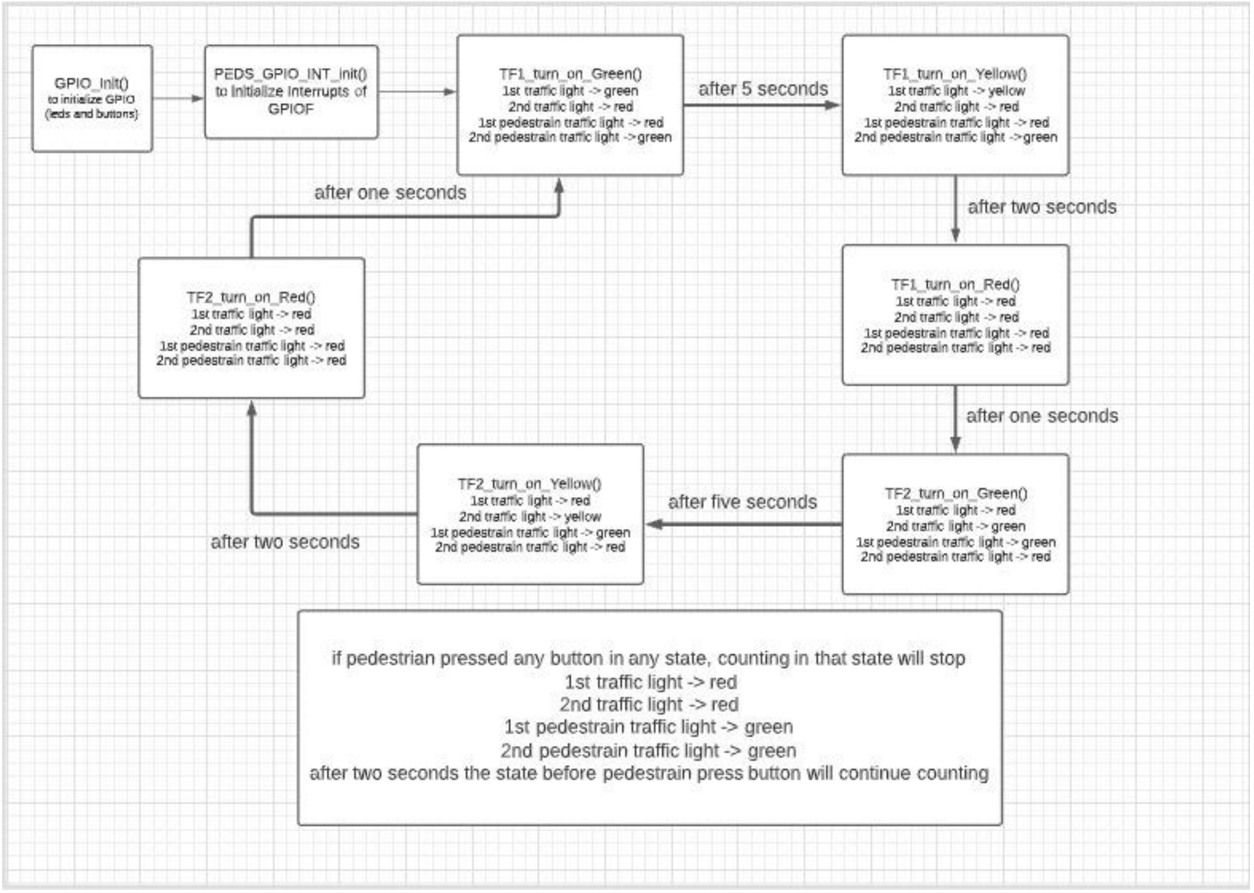
Traffic Light System

TEAM 11

MAHMOUD MOHASSEB	|| 20P2787
AHMED AMR BEHAIRY	|| 18P5837
OMAR KHALED OSSMAN	|| 18P1498
FARAH ESSAM	|| 18P3448
TASNEEM OSAMA	|| 18P3208

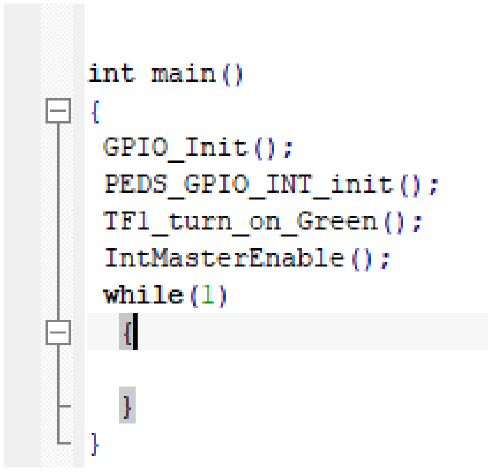
Design document that may include flow chart and description of files and functions used.

Flow Chart


GPIO_Init()
to initialize GPIO (leds and buttons)
PEDS_GPIO_INT_init()
to Initialize Interrupts of GPIOF
TF1_turn_on_Green()
1st traffic light -> green
2nd traffic light -> red
1st pedestrain traffic light -> red
2nd pedestrain traffic light -> green
after 5 seconds
TF1_turn_on_Yellow()
1st traffic light -> yellow
2nd traffic light -> red
1st pedestrain traffic light -> red
2nd pedestrain traffic light -> green
after two seconds
TF1_turn_on_Red()
1st traffic light -> red
2nd traffic light -> red
1st pedestrain traffic light -> red
2nd pedestrain traffic light -> red
after one seconds
TF2_turn_on_Green()
1st traffic light -> red
2nd traffic light -> green
1st pedestrain traffic light -> green
2nd pedestrain traffic light -> red
after five seconds
TF2_turn_on_Yellow()
1st traffic light -> red
2nd traffic light -> yellow
1st pedestrain traffic light -> green
2nd pedestrain traffic light -> red
after two seconds
TF2_turn_on_Red()
1st traffic light -> red
2nd traffic light -> red
1st pedestrain traffic light -> red
2nd pedestrain traffic light -> red
after one seconds
if pedestrian pressed any button in any state, counting in that state will stop
1st traffic light -> red
2nd traffic light -> red
1st pedestrain traffic light -> green
2nd pedestrain traffic light -> green
after two seconds the state before pedestrain press button will continue counting

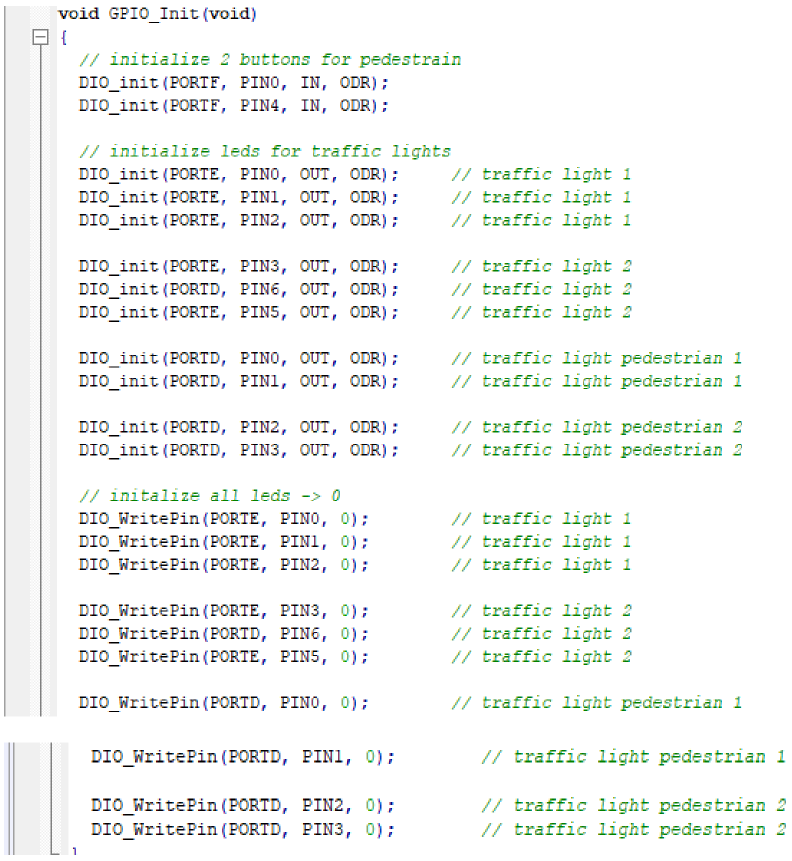
Code documentation

We will start as usual from the main function and we will step into every function whenever they are called.



```c
int main()
{
 GPIO_Init();
 PEDS_GPIO_INT_init();
 TF1_turn_on_Green();
 IntMasterEnable();
 while(1)
  {

  }
}
```

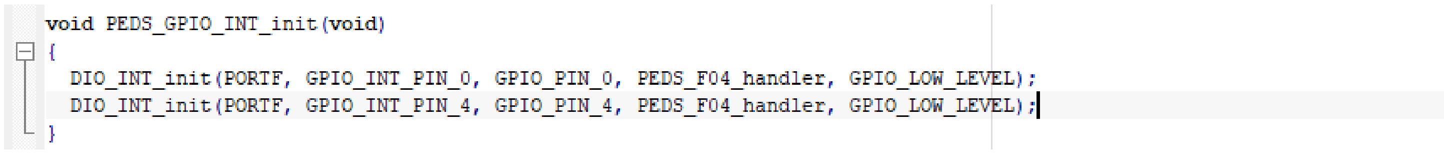
So as a start we will initialize the GPIO ports and pins that we'll use for the traffic light along with the switches that the pedestrians will use to pass the street. We will use 10 pins where each one represents one LED light in certain traffic light



```c
void GPIO_Init(void)
{
  // initialize 2 buttons for pedestrain
  DIO_init(PORTF, PIN0, IN, ODR);
  DIO_init(PORTF, PIN4, IN, ODR);

  // initialize leds for traffic lights
  DIO_init(PORTE, PIN0, OUT, ODR);     // traffic light 1
  DIO_init(PORTE, PIN1, OUT, ODR);     // traffic light 1
  DIO_init(PORTE, PIN2, OUT, ODR);     // traffic light 1

  DIO_init(PORTE, PIN3, OUT, ODR);     // traffic light 2
  DIO_init(PORTD, PIN6, OUT, ODR);     // traffic light 2
  DIO_init(PORTE, PIN5, OUT, ODR);     // traffic light 2

  DIO_init(PORTD, PIN0, OUT, ODR);     // traffic light pedestrian 1
  DIO_init(PORTD, PIN1, OUT, ODR);     // traffic light pedestrian 1

  DIO_init(PORTD, PIN2, OUT, ODR);     // traffic light pedestrian 2
  DIO_init(PORTD, PIN3, OUT, ODR);     // traffic light pedestrian 2

  // initalize all leds -> 0
  DIO_WritePin(PORTE, PIN0, 0);        // traffic light 1
  DIO_WritePin(PORTE, PIN1, 0);        // traffic light 1
  DIO_WritePin(PORTE, PIN2, 0);        // traffic light 1

  DIO_WritePin(PORTE, PIN3, 0);        // traffic light 2
  DIO_WritePin(PORTD, PIN6, 0);        // traffic light 2
  DIO_WritePin(PORTE, PIN5, 0);        // traffic light 2

  DIO_WritePin(PORTD, PIN0, 0);        // traffic light pedestrian 1
  DIO_WritePin(PORTD, PIN1, 0);        // traffic light pedestrian 1

  DIO_WritePin(PORTD, PIN2, 0);        // traffic light pedestrian 2
  DIO_WritePin(PORTD, PIN3, 0);        // traffic light pedestrian 2
}
```

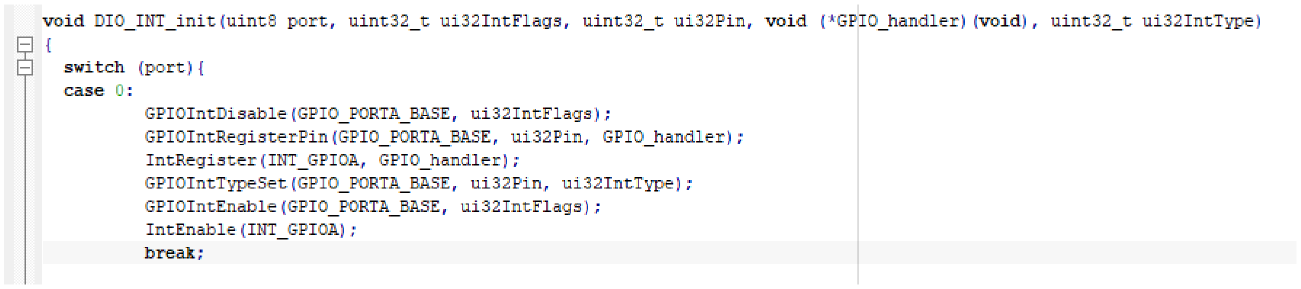
We used the functions of the DIO.C files we implemented in the very early labs to initialize the LEDs and switches.

Moving on to the next line in main we will call the initialization of pedestrians interrupts where we initialize interrupt for GPIO port F on both switches 0 & 4 using the DIO_INT_INIT function.



```c
void PEDS_GPIO_INT_init(void)
{
  DIO_INT_init(PORTF, GPIO_INT_PIN_0, GPIO_PIN_0, PEDS_F04_handler, GPIO_LOW_LEVEL);
  DIO_INT_init(PORTF, GPIO_INT_PIN_4, GPIO_PIN_4, PEDS_F04_handler, GPIO_LOW_LEVEL);
}
```

This DIO function takes (port, flags, pin, handler, type -> high/low level or failing/rising edge) as parameters

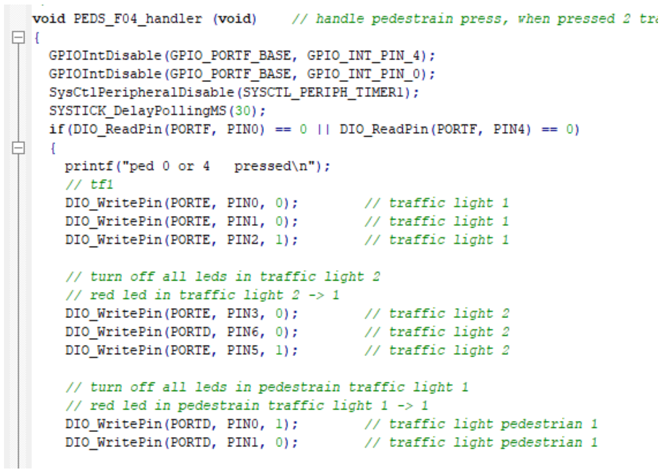


```c
void DIO_INT_init(uint8 port, uint32_t ui32IntFlags, uint32_t ui32Pin, void (*GPIO_handler)(void), uint32_t ui32IntType)
{
  switch (port){
  case 0:
          GPIOIntDisable(GPIO_PORTA_BASE, ui32IntFlags);
          GPIOIntRegisterPin(GPIO_PORTA_BASE, ui32Pin, GPIO_handler);
          IntRegister(INT_GPIOA, GPIO_handler);
          GPIOIntTypeSet(GPIO_PORTA_BASE, ui32Pin, ui32IntType);
          GPIOIntEnable(GPIO_PORTA_BASE, ui32IntFlags);
          IntEnable(INT_GPIOA);
          break;
```

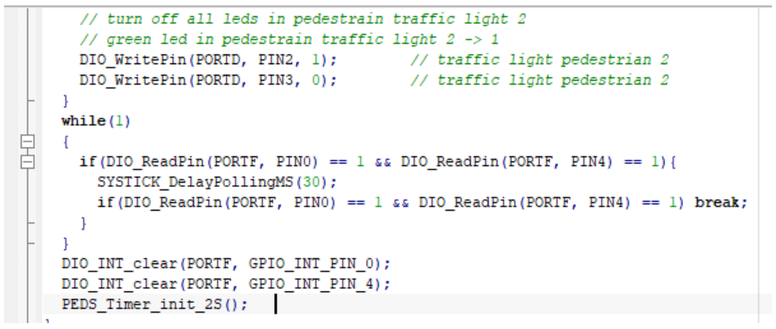
Figure 1- snippet from the function

This function disables the interrupt on the port selected and then do the following configuration to select pin responsible for it and assign this interrupt to a handler and after all configurations are set we will enable back the interrupt.

Now we are done with initializing the interrupts, we will need to handle these interrupt so the handler function which will be called upon interrupt is the PEDS_F04_handler().

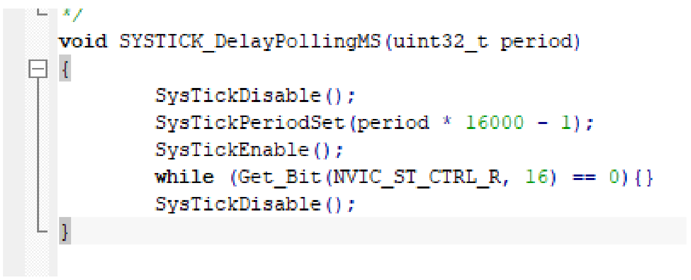


```c
void PEDS_F04_handler (void)    // handle pedestrain press, when pressed 2 tr
{
  GPIOIntDisable(GPIO_PORTF_BASE, GPIO_INT_PIN_4);
  GPIOIntDisable(GPIO_PORTF_BASE, GPIO_INT_PIN_0);
  SysCtlPeripheralDisable(SYSCTL_PERIPH_TIMER1);
  SYSTICK_DelayPollingMS(30);
  if(DIO_ReadPin(PORTF, PIN0) == 0 || DIO_ReadPin(PORTF, PIN4) == 0)
  {
    printf("ped 0 or 4   pressed\n");
    // tf1
    DIO_WritePin(PORTE, PIN0, 0);        // traffic light 1
    DIO_WritePin(PORTE, PIN1, 0);        // traffic light 1
    DIO_WritePin(PORTE, PIN2, 1);        // traffic light 1

    // turn off all leds in traffic light 2
    // red led in traffic light 2 -> 1
    DIO_WritePin(PORTE, PIN3, 0);        // traffic light 2
    DIO_WritePin(PORTD, PIN6, 0);        // traffic light 2
    DIO_WritePin(PORTE, PIN5, 1);        // traffic light 2

    // turn off all leds in pedestrain traffic light 1
    // red led in pedestrain traffic light 1 -> 1
    DIO_WritePin(PORTD, PIN0, 1);        // traffic light pedestrian 1
    DIO_WritePin(PORTD, PIN1, 0);        // traffic light pedestrian 1
```



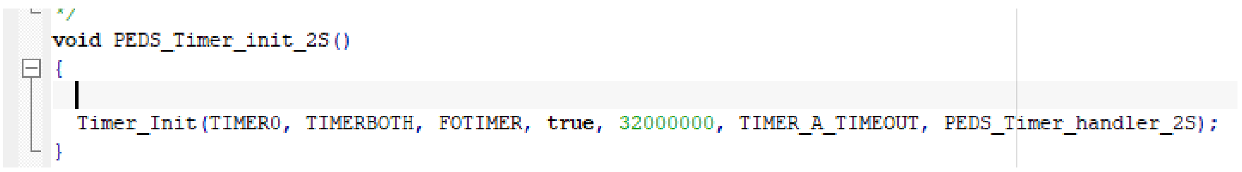
```
    // turn off all leds in pedestrain traffic light 2
    // green led in pedestrain traffic light 2 -> 1
    DIO_WritePin(PORTD, PIN2, 1);         // traffic light pedestrian 2
    DIO_WritePin(PORTD, PIN3, 0);         // traffic light pedestrian 2
  }
  while(1)
  {
    if(DIO_ReadPin(PORTF, PIN0) == 1 && DIO_ReadPin(PORTF, PIN4) == 1){
      SYSTICK_DelayPollingMS(30);
      if(DIO_ReadPin(PORTF, PIN0) == 1 && DIO_ReadPin(PORTF, PIN4) == 1) break;
    }
  }
  DIO_INT_clear(PORTF, GPIO_INT_PIN_0);
  DIO_INT_clear(PORTF, GPIO_INT_PIN_4);
  PEDS_Timer_init_2S();
```

We will start by disabling the interrupts on the switch as we are already in the interrupt state and we will wait 30 secs polling delay using systick module. The function SYSTICK_DelayPollingMS from SystickTimer.c is a function that use systick timer to count seconds using polling.



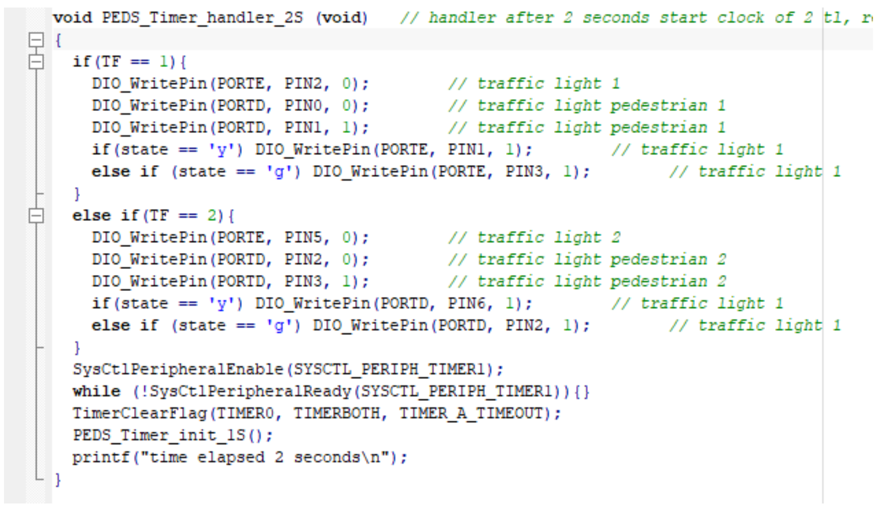
```
*/
void SYSTICK_DelayPollingMS(uint32_t period)
{
        SysTickDisable();
        SysTickPeriodSet(period * 16000 - 1);
        SysTickEnable();
        while (Get_Bit(NVIC_ST_CTRL_R, 16) == 0){}
        SysTickDisable();
}
```

SYSTICK_DelayPollingMS takes time as miliseconds until the switch bottom is stable and then we will handle all interrupt cases in the if condition. And we will end the function by clearing the interrupt flag for both the switches and then start the timer 2 seconds for the pedestrians to pass using the function PEDS_Timer_init_2S().



```
*/
void PEDS_Timer_init_2S()
{

  Timer_Init(TIMER0, TIMERBOTH, FOTIMER, true, 32000000, TIMER_A_TIMEOUT, PEDS_Timer_handler_2S);
}
```

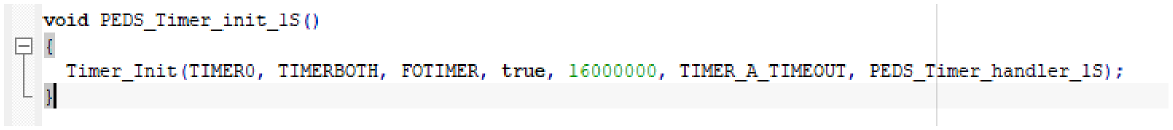
This function is to initialize timer for 2 seconds for pedestrian to cross street. It calls Timer_Init() from Timer.c. We use timer0 as full width one shot. Timer_Init() takes (timerBase, timer, configuration, bstall, value of ticks, flag and handler which called after 2 seconds) as parameters. As the handler of this timer will be called upon time elapsing, we will step into its function which is called PEDS_Timer_handler_2S().



```
void PEDS_Timer_handler_2S (void)   // handler after 2 seconds start clock of 2 tl, r
{
  if(TF == 1){
    DIO_WritePin(PORTE, PIN2, 0);        // traffic light 1
    DIO_WritePin(PORTD, PIN0, 0);        // traffic light pedestrian 1
    DIO_WritePin(PORTD, PIN1, 1);        // traffic light pedestrian 1
    if(state == 'y') DIO_WritePin(PORTE, PIN1, 1);        // traffic light 1
    else if (state == 'g') DIO_WritePin(PORTE, PIN3, 1);        // traffic light 1
  }
  else if(TF == 2){
    DIO_WritePin(PORTE, PIN5, 0);        // traffic light 2
    DIO_WritePin(PORTD, PIN2, 0);        // traffic light pedestrian 2
    DIO_WritePin(PORTD, PIN3, 1);        // traffic light pedestrian 2
    if(state == 'y') DIO_WritePin(PORTD, PIN6, 1);        // traffic light 1
    else if (state == 'g') DIO_WritePin(PORTD, PIN2, 1);        // traffic light 1
  }
  SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER1);
  while (!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER1)){}
  TimerClearFlag(TIMER0, TIMERBOTH, TIMER_A_TIMEOUT);
  PEDS_Timer_init_1S();
  printf("time elapsed 2 seconds\n");
}
```

Handler function of PEDS_Timer_init_2S() to return to state before pedestrain press button. We have two global variables one for state and another for which traffic light pedestrain press button. If 1st traffic light we turn off red led for 1st traffic light and depending on state we turn on led, if g -> turn on green, if y -> turn on yellow. For 1st pedestrian traffic light we turn off green led and turn on red led and same if 2nd traffic light. When pedestrian press button, we turn off clock of timer 1 to resume counting after 2 seconds (pedestrian cross street) .We turn on clock for timer 1 after 2 seconds then clear flag of timer 0 and initialize timer 0 to count for 1 second so we enable GPIO Interrupt (BOUNS).

After the interrupt time is elapsed we will have function to initialize timer for 1 second to enable interrupt of GPIOF because we turned it off when pedestrain press button. This function is named as PEDS_Timer_init_1. It calls Timer_Init() from Timer.c. We use timer 0 as full width one shot. Timer_Init() takes (timerBase, timer, configuration, bstall, value of ticks, flag and handler which called after 1 second).



```
void PEDS_Timer_init_1S()
{
  Timer_Init(TIMER0, TIMERBOTH, FOTIMER, true, 16000000, TIMER_A_TIMEOUT, PEDS_Timer_handler_1S);
}
```

After we initialized the interrupts and gave a brief explanation on how it will be implemented, we will move on line further in the main and we will start lighting up the LEDS using function TF1_turn_on_Green();

first we turned off interrupt of two buttons during writing on leds and turned them on again.

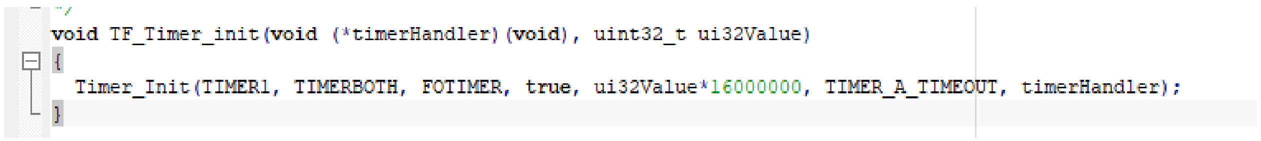
1st traffic light -> green

2nd traffic light -> red

1st pedestrain traffic light -> red

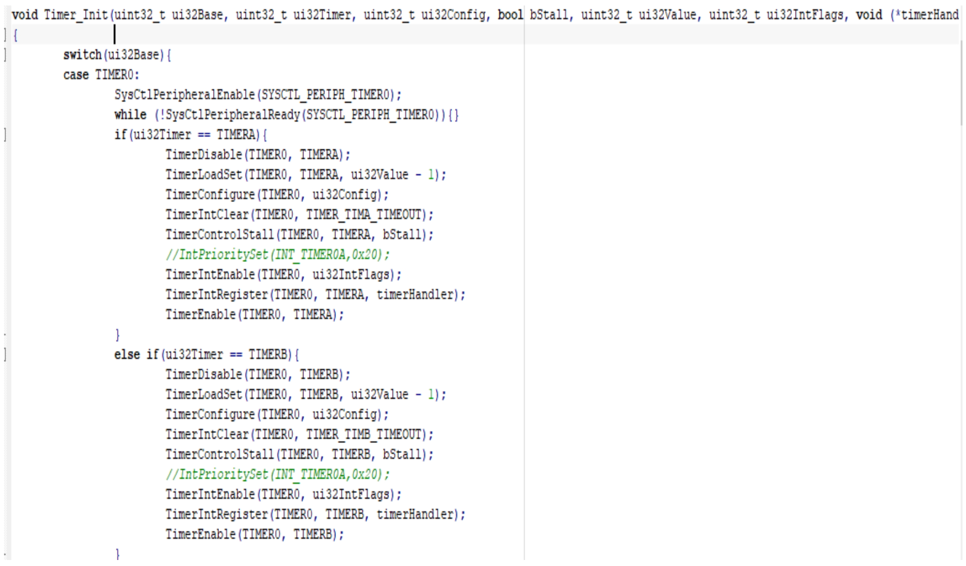
2nd pedestrain traffic light -> green

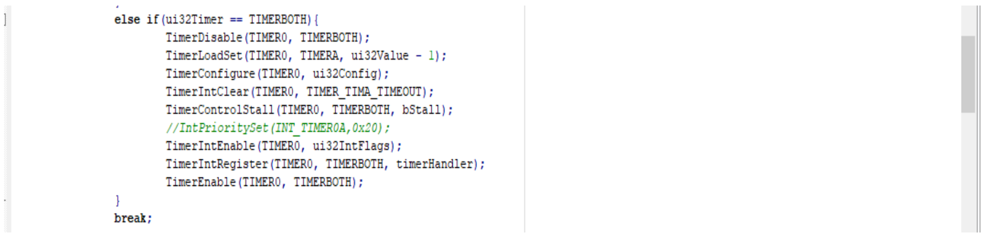
And update the global variables state and TF with the new values and call the function TF_Timer_init(TF1_turn_on_Yellow, 5);. This function that initialize timer for two traffic lights.it takes handler that will be called after ui32value elapsed



```
void TF_Timer_init(void (*timerHandler)(void), uint32_t ui32Value)
{
  Timer_Init(TIMER1, TIMERBOTH, FOTIMER, true, ui32Value*16000000, TIMER_A_TIMEOUT, timerHandler);
}
```

This function will call the Timer.c function which is Timer_Init. This function that initialize timer. It takes base timer, timer(a, b or both), configuration, bstall, value of ticks, flags, handler.





```
void Timer_Init(uint32_t ui32Base, uint32_t ui32Timer, uint32_t ui32Config, bool bStall, uint32_t ui32Value, uint32_t ui32IntFlags, void (*timerHand
{
        switch(ui32Base){
        case TIMER0:
                SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
                while (!SysCtlPeripheralReady(SYSCTL_PERIPH_TIMER0)){}
                if(ui32Timer == TIMERA){
                        TimerDisable(TIMER0, TIMERA);
                        TimerLoadSet(TIMER0, TIMERA, ui32Value - 1);
                        TimerConfigure(TIMER0, ui32Config);
                        TimerIntClear(TIMER0, TIMER_TIMA_TIMEOUT);
                        TimerControlStall(TIMER0, TIMERA, bStall);
                        //IntPrioritySet(INT_TIMER0A,0x20);
                        TimerIntEnable(TIMER0, ui32IntFlags);
                        TimerIntRegister(TIMER0, TIMERA, timerHandler);
                        TimerEnable(TIMER0, TIMERA);
                }
                else if(ui32Timer == TIMERB){
                        TimerDisable(TIMER0, TIMERB);
                        TimerLoadSet(TIMER0, TIMERB, ui32Value - 1);
                        TimerConfigure(TIMER0, ui32Config);
                        TimerIntClear(TIMER0, TIMER_TIMB_TIMEOUT);
                        TimerControlStall(TIMER0, TIMERB, bStall);
                        //IntPrioritySet(INT_TIMER0A,0x20);
                        TimerIntEnable(TIMER0, ui32IntFlags);
                        TimerIntRegister(TIMER0, TIMERB, timerHandler);
                        TimerEnable(TIMER0, TIMERB);
                }

                else if(ui32Timer == TIMERBOTH){
                        TimerDisable(TIMER0, TIMERBOTH);
                        TimerLoadSet(TIMER0, TIMERA, ui32Value - 1);
                        TimerConfigure(TIMER0, ui32Config);
                        TimerIntClear(TIMER0, TIMER_TIMA_TIMEOUT);
                        TimerControlStall(TIMER0, TIMERBOTH, bStall);
                        //IntPrioritySet(INT_TIMER0A,0x20);
                        TimerIntEnable(TIMER0, ui32IntFlags);
                        TimerIntRegister(TIMER0, TIMERBOTH, timerHandler);
                        TimerEnable(TIMER0, TIMERBOTH);
                }
                break;
```

Figure 2- snippet form the code

Now we saw how the code will transit in the functions during the case where Traffic Light 1 transit from Green to Yellow and this is a summary of what happened:

Update states of the 4 traffic lights -> timer start according to the states -> timer elapse -> timer handler which is the color of the next state.

This transitions will go on according to the flow chart provided in the first section.

Drive Links

One project that has all functions distributed on .c and .h files

https://drive.google.com/file/d/1zfOkR9mgWX6VFVJHXu4tn1PZAc4nc_nI/view?usp=sharing

Video file of maximum 5 minutes showing operation and project features

https://drive.google.com/file/d/1MDAfkZMJz7k4QnXbhxcYyTxSJ1nTe3WD/view?usp=sharing