You are required to implement a LED blinking project with a watchdog supervision capability. The **LED blinking capability** is handled through two software components. The first one is the LEDMgr and the second one is the GPIO.

# GPIO

## Required Functions:

* GPIO\_Init

It initializes GPIO configuration for the pin that used to control the LED.

* GPIO\_Write

It writes a specific value (0 or 1) to that pin.

# LEDMgr

## Required Functions:

* LED\_Init

It initializes the LED component internal variables.

* LED\_Manage

It manages the LED blinking actions using the GPIO\_Write function.

The **Watchdog management capability** is handled through two different components the WDGDrv and the WDGM. The WDGDrv shall implement a complete driver for the window watchdog peripheral in your STM32 microcontroller.

# WDGDrv

## Required Functions:

* WDGDrv\_Init

It configures the watchdog driver support the following features:

* + Set the Maximum timeout value to 50ms.
  + Disable the window mode.
  + Enable the early interrupt feature.
  + Activate the watchdog.
* WDGDrv\_IsrNotification

It checks the following conditions. If both of them are satisfied, it shall refresh the watchdog timer otherwise it will leave it to reset. The conditions are:

1. WDGM\_MainFunction is not stuck.
2. The WDGM State set by the WDGM\_MainFunction is OK

# WDGM

The WDGM shall provide supervision for the availability of the LEDM software entity.

## Required Functions:

* WDGM\_Init

It initializes the WDGM Internal variables

* WDGM\_ProvideSuppervisionStatus

It shall provide the Status of the LEDM entity to the WDGDrv.

* WDGM\_MainFunction

It shall check the number of calls of the LEDM\_MainFunction within a 100ms period. If the number of calls is between 8 and 12 then the status is OK. Otherwise, the status is not OK. The WDGM\_MainFunction shall be called periodically every 20ms.

* WDGM\_AlivenessIndication

It shall be called from the LEDM\_Manage function to detect its call at the correct timing.

# Notes

* The **LED\_Mange** shall be called from a super loop every 10ms and shall manage the LED blinking periodicity to be 500ms for each stage. **You can use the standard Delay function to manage the timing.**
* The WDGM state shall be known by checking the return of the function WDGM\_PovideSuppervisionStatus.

# Requirements:

You are required to provide the following:

1. Source code of the different components you provide.
2. Simulation file for your test.
3. A screen recording for the simulation of the following scenarios.
   1. **Positive scenario** that checks the periodicity of the LED Blinking, Call of the LEDM\_Manage, Call of the WDGM\_MainFunction and refreshment of the WDGDrv.  
      - You can provide the timing evidence by using test pins toggle on the oscilloscope.
   2. **Negative scenarios**:
      1. One that comments the call of the WDGM\_MainFunction and checks that the watchdog reset occurred after 50ms.
      2. One that comments the call of the WDGM\_AlivenessIndication from the LEDM\_Manage while the WDGM\_MainFunction is executed periodically and checks that the watchdog reset occurs after 100ms.
      3. One that changes the periodicity of the call of the LEDM\_MainFunction to be every 5ms and checks that watchdog reset occurs after 100ms.

The header files of the components are attached, you are not allowed to change the APIs prototype or add new APIs.

# The Flow

1. **Initialization**:
   * First, the GPIO pin that controls the LED is set up.
   * Then, the LED management is initialized, setting up any internal variables needed.
   * The watchdog driver is initialized, configuring it to reset the system if something goes wrong.
   * Finally, the watchdog manager is initialized to start supervising the LED management component.
2. **Main Loop**:
   * The main loop of the program runs indefinitely.
3. **LED Blinking**:
   * Every 10 milliseconds, the LED\_Manage function is called.
   * If it has been 500 milliseconds since the last toggle:
     + The LED state is changed: if it was on, it is turned off, and if it was off, it is turned on.
   * This toggle of the LED involves writing a value (0 or 1) to the GPIO pin.
   * The WDGM\_AlivenessIndication function is called to notify the watchdog manager that the LED management function is running.
4. **Watchdog Supervision**:
   * Every 20 milliseconds, the WDGM\_MainFunction is called.
   * This function checks how many times the LED\_Manage function has been called in the last 100 milliseconds.
     + If the LED\_Manage function has been called between 8 and 12 times, everything is fine, and the status is set to OK.
     + If it hasn't been called the correct number of times, the status is set to NOK.
5. **Watchdog Refresh**:
   * Periodically (usually handled by an interrupt in real systems), the WDGDrv\_IsrNotification function is called.
   * This function checks if:
     + The watchdog manager function is running correctly and is not stuck.
     + The status reported by the watchdog manager is OK.
   * If both conditions are met, the watchdog timer is refreshed to prevent the system from resetting.
   * If either condition is not met, the watchdog timer is not refreshed, and the system will reset after 50 milliseconds.

Inside the main function (in main.c):

* first, we'll initialize the LED, WDGDrv, WDGM
* in the loop of the main function
* we will call the "LED\_Manage" every 10ms, but the LED state only changes after 500ms. And inside the "LED\_Manage", we call the WDGM\_AlivenessIndication to count the number of calls.
* Then, we will call WDGM\_MainFunction every 20ms to check that the LED\_Manage is called from 8 to12 times in the last 100ms.

Lastly, the WDGM\_ProvideSuppervisionStatus is called (every 50ms which is the timeout intervel) within the WDGDrv\_IsrNotificationreturns to get OK (the LED\_Manage was called 8-12 times) or NOK. So the IsrNotification decides whether to reset the watchdog or the system.