LTOS – Led blink Example with NUCLEO-F401RE

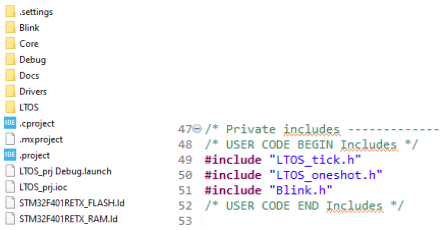
Adding LTOS to your project is so simply. You should only copy & pas LTOS folder into your project direction. LTOS folder has 3 subfolders:

- docs : documents about LTOS
- src : Source files
- inc : Header files

Following steps explain LTOS-LedBlink project in LedBlinkPrj_NucleoF401RE folder. This project created with STM32CubeIDE version: 1.7.0

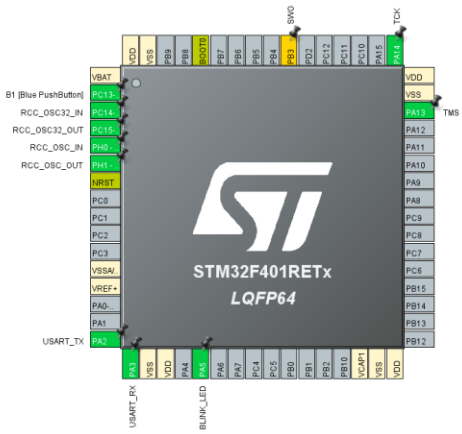
1- LTOS Folder

In LedBlinkPrj_NucleoF401RE project folder created by STM32CubeIDE wrt project specifications. After project created, LTOS folder just added into project folder and added #include directives into main.h file.



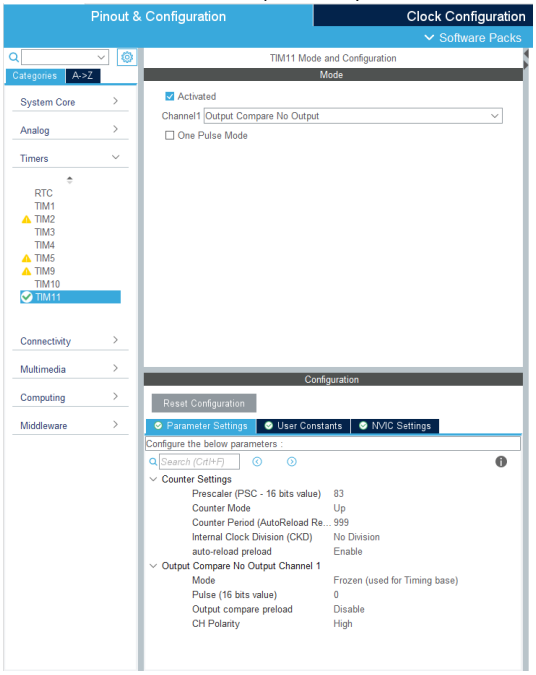
2- Blink LED

In Nucleo-F401RE development board, PA.5 pin connected to User-LED (yellow led). So, PA5 used as a BLINK_LED in the example project.



3- TIMER

LTOS-tick has to be increased periodically wrt LTOS resolution. In this example we used TIMER11 to increase LTOS-tick every 1ms.



- TIM11 feed by APB2 Peripheral CLK. So, its clock frequency is 84MHz.
- To decrease TIM11's clock frequency to 1MHz, prescaler value set to 83 (84MHz / (83+1) = 1MHz).
- To get 1ms periodic interrupt from TIM11, the Auto Reload Register value set to 999 (1MHz / (999+1) = 1kHz).
- TIM11 global interrupt enabled and LTOS_tickIncrease function call added inside TIM1_TRIG_COM_TIM11_IRQHandler function to get interrupt periodically.

```
/**
  * @brief This function handles TIM1 trigger and commutation interrupts and TIM11 global interrupt
  */
void TIM1_TRG_COM_TIM11_IRQHandler(void)
{
  /* USER CODE BEGIN TIM1_TRG_COM_TIM11_IRQn 0 */
    if(LL_TIM_IsActiveFlag_UPDATE(TIM11)) {
        LL_TIM_ClearFlag_UPDATE(TIM11);
        LTOS_tickIncrease();
    }
  /* USER CODE END TIM1_TRG_COM_TIM11_IRQn 0 */

  /* USER CODE BEGIN TIM1_TRG_COM_TIM11_IRQn 1 */

  /* USER CODE END TIM1_TRG_COM_TIM11_IRQn 1 */
}
```

LTOS-tick will be increased every 1ms.

4- Blink Task

In this step we will create our blink task and attach it periodically.

- First step is allocation oneshot to use it as a caller of blink task.
- Second step is attach our task to oneshot.

```
/**
  * @brief  Init Blink Task
  * @param  None
  * @retval None
  */
void BlinkInit(void)
{
    blinkFlag = 0;

    blinkOS = LTOS_oneshotAlloc();
    if(blinkOS) {
        LTOS_oneshotAttach(blinkOS, (os_callback_t)BlinkTASK, 0, BLINK_OFF_TIME);
    }
}
```

- Blink task is ready. It will be called when it attached with requested time.

```
/**
  * @brief  Blink Task
  * @param  argument
  * @retval None
  */
void BlinkTASK(uint32_t arg)
{
    tick_t tout;

    blinkFlag ^= 1;

    if(blinkFlag) {
        LL_GPIO_SetOutputPin(BLINK_LED_GPIO_Port, BLINK_LED_Pin);
        tout = BLINK_ON_TIME;
    } else {
        LL_GPIO_ResetOutputPin(BLINK_LED_GPIO_Port, BLINK_LED_Pin);
        tout = BLINK_OFF_TIME;
    }

    LTOS_oneshotAttach(blinkOS, (os_callback_t)BlinkTASK, 0, tout);
}
```

5- LTOS Run

We are ready to use LTOS. It is simple. We should only call LTOS_run endless function.

```
  /* USER CODE BEGIN 2 */
  BlinkInit();
  LTOS_run();
  /* USER CODE END 2 */
```

This function never return until memory crash error occurred while LTOS running.