Push to Pair Button

****

Note: Clock Configuration

MCU CLOCK = MAX_FREQ = 64 MHz
Prescaler (PSC) = 64000-1
Auto Reload Register (ARR) = Counter Period = 1000-1

Period Elapsed = 1 second. (Periodic interrupt)

****

Define Button as GPIO EXTI
Configuration:
-> GPIO Pull-Up (Active Low)
-> GPIO Mode External Interrupt mode with falling/rising edge trigger detection.

****

Macro:

```
// button switch active low
#define BUTTON_PIN_DOWN 0
#define BUTTON_PIN_UP 1

#define SECURE_MODE_ON 1
#define SECURE_MODE_OFF 0
```

Global Variables:

```
// variable stored value of the counter when the button is pressed down
static uint16_t buttondown_TIMCNT = -1;

// variable stored value of boolean logic of *Secure pin*
static bool secureMode = true; // assume that true for testing
```

GPIO_EXTI Interrupt Falling (Button Down)

```
void HAL_GPIO_EXTI_Falling_Callback(uint16_t GPIO_Pin)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(GPIO_Pin);

  /* NOTE: This function should not be modified, when the callback is needed,
           the HAL_GPIO_EXTI_Falling_Callback could be implemented in the user file
  */

  // Debouncing button switch with 10 ms delay.
  HAL_Delay(10);

  // Declare startTime at 0
  const uint8_t startTime = 0;

  // External interrupt with falling edge trigger detection **button active low**
  // Check interrupt pin
  // Check secure mode status
  if( ( GPIO_Pin == BUTTON_Pin ) && ( secureMode == SECURE_MODE_ON ) )
  {

    // Clear EXTI flags.
    if(__HAL_GPIO_EXTI_GET_FALLING_IT(BUTTON_Pin) != RESET)
__HAL_GPIO_EXTI_CLEAR_FALLING_IT(BUTTON_Pin);

    // Reset Counter value variable to 0 every GPIO EXTI Callback.
    buttondown_TIMCNT = startTime;

    // Start timer (TIM2) with interrupt mode.
    HAL_TIM_Base_Start_IT(&htim2);

    // Toggle Led for indicating when the button is pressed.
    HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);

  }

}
```

GPIO_EXTI Interrupt Falling (Button Up)

```
void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin)
{

  /* Prevent unused argument(s) compilation warning */
  UNUSED(GPIO_Pin);
  /* NOTE: This function should not be modified, when the callback is needed,
     the HAL_GPIO_EXTI_Rising_Callback could be implemented in the user file
  */

  // Debouncing button switch with 10 ms delay.
  HAL_Delay(10);
```

```
  // External interrupt with Rising edge trigger detection **button active low**
  // Check interrupt pin
  // Check secure mode status
  if( ( GPIO_Pin == BUTTON_Pin ) && ( secureMode == SECURE_MODE_ON ) )
  {

      // Clear EXTI flags
      if(__HAL_GPIO_EXTI_GET_RISING_IT(BUTTON_Pin) != RESET)
__HAL_GPIO_EXTI_CLEAR_RISING_IT(BUTTON_Pin);

      // Stop timer (TIM2) with interrupt mode
      HAL_TIM_Base_Stop_IT(&htim2);

      // Toggle Led for indicating when the button is not pressed down
      HAL_GPIO_TogglePin(LED_GPIO_Port, LED_Pin);

      // Declare variables to define button operate time
      const uint8_t startTime = 0;
      const uint8_t shortTime = 2;
      const uint8_t longTime = 5;
      const uint8_t endTime = 15;

      // Check button down time counter conditions
      // startTime  <=  button-down-time-counter <   shortTime
      // shortTime  <=  button-down-time-counter <   longTime
      // longTime   <=  button-down-time-counter <=  endTime

      if( buttondown_TIMCNT >= startTime && buttondown_TIMCNT < shortTime )
      {
            buttondown_TIMCNT = startTime;
          // Quick press -> UART Command To BTM "kill/disconnect current connection"
            printf("Quick Press\n"); // debug msg
      }

      else if( buttondown_TIMCNT >= shortTime && buttondown_TIMCNT < longTime )
      {
            buttondown_TIMCNT = startTime;
          // Short press -> UART Command to BTM "Start BTM Pairing mode"
            printf("Short Press\n"); // debug msg
      }

      else if( buttondown_TIMCNT >= longTime && buttondown_TIMCNT <= endTime )
      {
            buttondown_TIMCNT = startTime;
            // Long press -> UART Command to BTM
                  "Delete all connection paired lists in non-volatile memory"
            printf("Long Press\n"); // debug msg
      }

      else if ( buttondown_TIMCNT > longTime )
      {
            buttondown_TIMCNT = startTime;
      }
```

```
   }
}
```

Timer Periodic Interrupt

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

  /*Prevent unused argument(s) compilation warning*/
  UNUSED(htim);

  if (htim->Instance == TIM2)
  {
    // Count up buttondown_TIMCNT variable every 1 sec period elapsed.
    buttondown_TIMCNT += 1;
  }

  /* NOTE : This function should not be modified, when the callback is needed,
          the HAL_TIM_PeriodElapsedCallback could be implemented in the user file
   */
}
```