**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

\*/

}