```
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void TickTimer_timeout_handler(void * p_context)
{
  // the ISR is visited once a mSec
  UNUSED_PARAMETER(p_context);
  CumeTickCounter++;
  if(CounterOnFlag == true)
  {
    CounterOnFlag = false;
    NRF_TIMER2->TASKS_CLEAR = 1;
    NRF_TIMER2->TASKS_START = 1;
  }
  else if(CumeTickCounter > 60000)  // 60 seconds
  {
    NRF_TIMER2->TASKS_STOP = 1;
    // CC[0] SHOULD have 14,640 counts: every sec is 244 counts (1,000,000 / 2^12) * 60 seconds
    NRF_TIMER2->TASKS_CAPTURE[0] = 1;
    // BUT it has around 14,751 counts. This translates to 420,000 extra counts per minute (0.4
    seconds!)
    CumeTickCounter = 0;
    CounterOnFlag = true;
  }
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

void timers_init(void)
{
  // Initialize timer module.
  APP_TIMER_INIT(APP_TIMER_PRESCALER, APP_TIMER_MAX_TIMERS, APP_TIMER_OP_QUEUE_SIZE, false);

  // Create timers.
//-----------------------------------------------------------------------------------------------------
  Global_err_code = app_timer_create(&m_battery_timer_id,
                                     APP_TIMER_MODE_REPEATED,
                                     BatteryMeasurementTimeoutHandler);
  APP_ERROR_CHECK(Global_err_code);
//-----------------------------------------------------------------------------------------------------
  Global_err_code = app_timer_create(&m_ambient_timer_id,
                                     APP_TIMER_MODE_REPEATED,
                                     AmbientMeasurementTimeoutHandler);
  APP_ERROR_CHECK(Global_err_code);
//-----------------------------------------------------------------------------------------------------
  Global_err_code = app_timer_create(&m_1mSec_timer_id,
                                     APP_TIMER_MODE_REPEATED,
                                     TickTimer_timeout_handler);
  APP_ERROR_CHECK(Global_err_code);
```

```
//------------------------------------------------------------------------------------------------
--
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~

/**@brief Function for starting the application timers.
 */
#define TIMER_TICK_1mSEC_INTERVAL APP_TIMER_TICKS(1, APP_TIMER_PRESCALER)     /**< 1 mSec timer
tick (ticks). */
void application_timers_start(void)
{
  // Start application timers
//------------------------------------------------------------------------------------------------
--
  Global_err_code = app_timer_start(m_battery_timer_id, BATTERY_LEVEL_MEAS_INTERVAL, NULL);
  APP_ERROR_CHECK(Global_err_code);
//------------------------------------------------------------------------------------------------
--
  Global_err_code = app_timer_start(m_ambient_timer_id, AMBIENT_LEVEL_MEAS_INTERVAL, NULL);
  APP_ERROR_CHECK(Global_err_code);
//------------------------------------------------------------------------------------------------
--
  Global_err_code = app_timer_start(m_1mSec_timer_id, TIMER_TICK_1mSEC_INTERVAL, NULL);
  APP_ERROR_CHECK(Global_err_code);
//------------------------------------------------------------------------------------------------
--
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~

/* USING SD110, SDK7.2.0 */

int main(void)
{
  timers_init();
  gpiote_init();
  gpio_init();

  // CONFIG GPIOTE
  nrf_gpio_cfg_input(DIAG00, NRF_GPIO_PIN_PULLDOWN);
  NRF_GPIOTE->EVENTS_IN[1] = 0;
  NRF_GPIOTE->CONFIG[1] = (GPIOTE_CONFIG_MODE_Event << GPIOTE_CONFIG_MODE_Pos) |
                          (DIAG00 << GPIOTE_CONFIG_PSEL_Pos) |
                          (GPIOTE_CONFIG_POLARITY_LoToHi << GPIOTE_CONFIG_POLARITY_Pos);
  // CONFIG TIMER 2
  NRF_TIMER2->TASKS_STOP = 1;
  NRF_TIMER2->MODE = TIMER_MODE_MODE_Counter;                 // Set the timer in Counter Mode
  NRF_TIMER2->BITMODE = TIMER_BITMODE_BITMODE_16Bit;          // Set counter to 16 bit
  resolution (max for T2)
  NRF_TIMER2->PRESCALER = 0;
```

```
  NRF_TIMER2->TASKS_CLEAR = 1;                                // clear the task first to be
  usable for later
  NRF_TIMER2->CC[0] = 0;
  // CONFIGURE PPI CHANNEL 0 TO COUNT UP THE TIMER COUNTER ON EVERY GPIOTE EVENT (244Hz clock)
  NRF_PPI->CH[0].EEP = (uint32_t)&NRF_GPIOTE->EVENTS_IN[1];
  NRF_PPI->CH[0].TEP = (uint32_t)&NRF_TIMER2->TASKS_COUNT;
  NRF_PPI->CHEN = (PPI_CHEN_CH0_Enabled << PPI_CHEN_CH0_Pos); // enable ppi channel 0

  // Start the usual stuff but NOT the BLE
  application_timers_start();
  ble_stack_init();
  InitNVStorage();
  gap_params_init();
  services_init();
  advertising_init();
  conn_params_init();
  sec_params_init();

  while(1);
}

//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~
```