hi

why 'HAL_I2C_IsDeviceReady' is always 'BUSY'?

using board : Nucleo-F446RE , MPU6050
Library : HAL

here is my source code ----------

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  *
  * COPYRIGHT(c) 2016 STMicroelectronics
  *
  * Redistribution and use in source and binary forms, with or without modification,
  * are permitted provided that the following conditions are met:
  *   1. Redistributions of source code must retain the above copyright notice,
  *      this list of conditions and the following disclaimer.
  *   2. Redistributions in binary form must reproduce the above copyright notice,
  *      this list of conditions and the following disclaimer in the documentation
  *      and/or other materials provided with the distribution.
  *   3. Neither the name of STMicroelectronics nor the names of its contributors
  *      may be used to endorse or promote products derived from this software
  *      without specific prior written permission.
  *
  * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
  * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
  * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
  * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
  *
  ******************************************************************************
  */
/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

/* USER CODE BEGIN Includes */
#include "tlcd.h"
#include "my_mpu6050.h"
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
DMA_HandleTypeDef hdma_i2c1_rx;
DMA_HandleTypeDef hdma_i2c1_tx;

TIM_HandleTypeDef htim2;
TIM_HandleTypeDef htim3;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
#define DATA  10
uint8_t Rx_Buffer2[DATA];
uint8_t Rx_Data2[2];
uint8_t Rx_Indx2;
uint8_t Transfer_cplt2;
__IO uint8_t i2c_data[20];


volatileuint32_t count=0;

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
voidSystemClock_Config(void);
voidError_Handler(void);
staticvoidMX_GPIO_Init(void);
staticvoidMX_DMA_Init(void);
staticvoidMX_TIM2_Init(void);
staticvoidMX_TIM3_Init(void);
staticvoidMX_USART2_UART_Init(void);
staticvoidMX_I2C1_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
```

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart);
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */
voidHAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
uint8_t i;

HAL_GPIO_TogglePin(GPIOA, GPIO_PIN_5);

if(huart->Instance==USART2)
{
if(Rx_Indx2==0)
{
for(i=0;i<DATA;i++) Rx_Buffer2[i] = 0;
}
if(Rx_Data2[0]!=0x0d)
{
Rx_Buffer2[Rx_Indx2++]=Rx_Data2[0];
}
else
{
Rx_Indx2 = 0;
Transfer_cplt2 = 1;
}
HAL_UART_Receive_IT(&huart2, Rx_Data2, 1);
HAL_UART_Transmit(&huart2, Rx_Data2, 1,1000);
}
}

voidHAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
if(htim->Instance==TIM2) {
// HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
// HAL_UART_Transmit(&huart2, (uint8_t *)"Start!!!!!\r\n", 12,1000);
}
elseif(htim->Instance==TIM3) {

}
}

/* USER CODE END 0 */

intmain(void)
{

  /* USER CODE BEGIN 1 */
char s1[17] = "**1234567890**";
char s2[17] = "**ABCDEFGHIJ**";

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* Configure the system clock */
  SystemClock_Config();

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_TIM2_Init();
  MX_TIM3_Init();
  MX_USART2_UART_Init();
  MX_I2C1_Init();

  /* USER CODE BEGIN 2 */
while(HAL_UART_Receive_IT(&huart2, Rx_Data2, 1)!=HAL_OK);

// Activate TIMER Interrupt
while(HAL_TIM_Base_Start_IT(&htim2)!=HAL_OK);
while(HAL_TIM_Base_Start_IT(&htim3)!=HAL_OK);

//  i2c1 setting
uint8_t whoAmI[2];
// whoAmI[0] = RA_PWR_ADDR;
// whoAmI[1] = RA_PWR_MGMT_1;
uint8_t  pdata=0x00;


while(HAL_I2C_IsDeviceReady(&hi2c1,MPU6050,10,1000)!=HAL_OK);

HAL_I2C_Master_Transmit_DMA(&hi2c1,MPU6050,&pdata,1);
HAL_I2C_Mem_Write_DMA(&hi2c1,MPU6050>>1,RA_PWR_ADDR,1,&pdata,1);
HAL_UART_Transmit(&huart2, (uint8_t *)"\r\n11111!!!!!\r\n", 14,1000);

/*
while(HAL_I2C_GetState(&hi2c1)!=HAL_I2C_STATE_READY);
HAL_UART_Transmit(&huart2, (uint8_t *)"\r\n22222!!!!!\r\n", 14,1000);

while(HAL_I2C_Master_Receive_DMA(&hi2c1,MPU6050,whoAmI,2)!=HAL_OK);
```

```
  HAL_UART_Transmit(&huart2, (uint8_t *)"\r\n33333!!!!!\r\n", 14,1000);
  */
  //HAL_I2C_Master_Receive_DMA(&hi2c1,0x68,(uint8_t *)i2c_data,7);
  // HAL_I2C_Mem_Read_DMA(&hi2c1,0x69,0x3b,14,(uint8_t *)i2c_data,14);
  //while(HAL_I2C_Master_Receive_IT(&hi2c1,i2c_buffer,1) != HAL_OK );

  // ADC - DMA
  char sss[10];
  HAL_UART_Transmit(&huart2, (uint8_t *)"\r\nStart!!!!!\r\n", 14,1000);
  sprintf(sss,"%2x,%2x\r\n",whoAmI[0],whoAmI[1]);
  HAL_UART_Transmit(&huart2, sss, 10,1000);

  // LCD??? ?? GPIOC? ??.
  TLCD_Init(GPIO_PIN_10,GPIO_PIN_11,GPIO_PIN_0,GPIO_PIN_1,GPIO_PIN_2,GPIO_PIN_3);

  TLCD_Puts(1,s1);
  TLCD_Puts(2,s2);

  HAL_Delay(3000000);

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  TLCD_Puts(1," *Init    Gyro*");
  TLCD_Puts(2," *InIt   Motor*");
  HAL_Delay(1000000);
  TLCD_Puts(1," *InIt   Motor*");
  TLCD_Puts(2," *Init    Gyro*");
  HAL_Delay(1000000);
  HAL_UART_Transmit(&huart2, (uint8_t *)"\r\nwhile!!!!!\r\n", 14,1000);

    }
    /* USER CODE END 3 */

  }

  /** System Clock Configuration
  */
  voidSystemClock_Config(void)
  {

    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;

    __HAL_RCC_PWR_CLK_ENABLE();

    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
    RCC_OscInitStruct.HSEState = RCC_HSE_ON;
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
    RCC_OscInitStruct.PLL.PLLM = 4;
    RCC_OscInitStruct.PLL.PLLN = 180;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
    RCC_OscInitStruct.PLL.PLLQ = 2;
    RCC_OscInitStruct.PLL.PLLR = 2;
    if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
    {
      Error_Handler();
    }

    if (HAL_PWREx_EnableOverDrive() != HAL_OK)
    {
      Error_Handler();
    }

    RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                  |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
    RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
    RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
    RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
    RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
    if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5) != HAL_OK)
    {
      Error_Handler();
    }

    HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000000);

    HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

    /* SysTick_IRQn interrupt configuration */
    HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
  }

  /* I2C1 init function */
  staticvoidMX_I2C1_Init(void)
  {
```

```
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 100000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }

}

/* TIM2 init function */
staticvoidMX_TIM2_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim2.Instance = TIM2;
  htim2.Init.Prescaler = 9000;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 4999;
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  if (HAL_TIM_Base_Init(&htim2) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim2, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim2, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }

}

/* TIM3 init function */
staticvoidMX_TIM3_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim3.Instance = TIM3;
  htim3.Init.Prescaler = 9000;
  htim3.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim3.Init.Period = 9999;
  htim3.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  if (HAL_TIM_Base_Init(&htim3) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim3, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim3, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }

}

/* USART2 init function */
staticvoidMX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
```

```
    Error_Handler();
  }

}

/**
  * Enable DMA controller clock
  */
staticvoidMX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __HAL_RCC_DMA1_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA1_Stream0_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream0_IRQn);
  /* DMA1_Stream6_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA1_Stream6_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA1_Stream6_IRQn);

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
*/
staticvoidMX_GPIO_Init(void)
{

  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOC_CLK_ENABLE();
  __HAL_RCC_GPIOH_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin : B1_Pin */
  GPIO_InitStruct.Pin = B1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : PC0 PC1 PC2 PC3
                           PC8 PC9 PC10 PC11 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                        |GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pin : LD2_Pin */
  GPIO_InitStruct.Pin = LD2_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3
                        |GPIO_PIN_8|GPIO_PIN_9|GPIO_PIN_10|GPIO_PIN_11, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  None
  * @retval None
  */
voidError_Handler(void)
{
  /* USER CODE BEGIN Error_Handler */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler */
}

#ifdef USE_FULL_ASSERT

/**
   * @brief Reports the name of the source file and the source line number
```

```
    * where the assert_param error has occurred.
    * @param file: pointer to the source file name
    * @param line: assert_param error line source number
    * @retval None
    */
  void assert_failed(uint8_t* file, uint32_t line)
  {
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
      ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */

  }

  #endif

  /**
    * @}
    */

  /**
    * @}
  */

  /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```