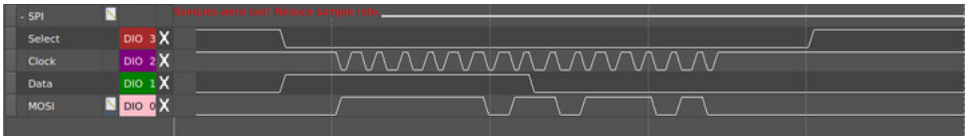
Hello! I new to programming within the embedded world(and C in general) and I have been stuck on this problem for some time now.

The problem is that I want to use MPU9250 with SPI by using HAL, but I can't find out why I only receive the value 255 from the IMU.

I am using a Nucleo-F767ZI board and MPU9250 + 280 for this project.

This is how the signals look like:



Yellow: MISO, Green: MOSI

I don't really understand why Data(MISO) is outputting in this matter.

I am trying to read the WHO_AM_I register of the MPU9250 which is supposed to be 0x71 not 0xFF.

The source code: (mpu9250.c)

```
#include "mpu9250.h"

void mpu_init(void)
{
	// Initialize SPI
	MX_SPI1_Init();
	// Reset MPU
	mpu_write_register(PWR_MGMT_1, 0x01);
	// Power on MPU
	mpu_write_register(PWR_MGMT_1, 0x00);
	// Turn off I2C mode and enable SPI mode
	mpu_write_register(USER_CTRL, 0x10);
	// Set MPU sample rate
	mpu_write_register(SMPLRT_DIV, 0x07);
	// Enable gyro with full scale range
	mpu_write_register(GYRO_CONFIG, 0x01);
	// Write to ACCEL_CONFIG register
	mpu_write_register(ACCL_CONFIG, 0x01);
}

HAL_StatusTypeDef mpu_write_register(uint8_t wr_register, uint8_t wr_data)
{
	uint8_t data[2] = {wr_register, wr_data};
	// Set chip select low to enable MPU
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);
	// Transmit register and data to MPU
	HAL_StatusTypeDef spi_result = HAL_SPI_Transmit(&hspi1, data, 2, 100);
	// Set chip select to high to disable MPU
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);
	return spi_result;
}

HAL_StatusTypeDef mpu_read_byte(uint8_t rd_register, uint8_t *rd_data)
{
	uint8_t data_rd_bit = rd_register | 0x80;
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_RESET);
	HAL_StatusTypeDef spi_result = HAL_SPI_TransmitReceive(&hspi1, &data_rd_bit, rd_data, 2, 100);
	HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);

	return spi_result;
}

static void MX_SPI1_Init(void)
```

```
{

  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_HIGH;
  hspi1.Init.CLKPhase = SPI_PHASE_2EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_64;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
  hspi1.Init.CRCPolynomial = 7;
  hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
  //hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}
????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
```

main.c

```
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "stm32f7xx_hal.h"
#include "mpu9250.h"
#include <stdlib.h>

UART_HandleTypeDef huart3;

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_USART3_UART_Init(void);

int main(void)
{
  /* 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();
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_6, GPIO_PIN_SET);
  MX_USART3_UART_Init();

  mpu_init();

  char msg[15];
  uint8_t data[1] = {0};

  while (1)
  {
       if(mpu_read_byte(WHO_AM_I, data) != HAL_OK) {
            sprintf(msg, "Error\r\n");
       } else {
            sprintf(msg, "Who am i: %d\r\n", data[0]);
       }
       HAL_UART_Transmit(&huart3, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
       HAL_Delay(500);
  }
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct;

    /**Configure the main internal regulator output voltage
    */
  __HAL_RCC_PWR_CLK_ENABLE();

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

    /**Initializes the CPU, AHB and APB busses clocks
    */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = 16;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 96;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV6;
  RCC_OscInitStruct.PLL.PLLQ = 2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
```

```
  }

    /**Initializes the CPU, AHB and APB busses clocks
    */
  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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

  PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_USART3;
  PeriphClkInitStruct.Usart3ClockSelection = RCC_USART3CLKSOURCE_PCLK1;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

    /**Configure the Systick interrupt time
    */
  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

    /**Configure the Systick
    */
  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

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

/* USART3 init function */
static void MX_USART3_UART_Init(void)
{

  huart3.Instance = USART3;
  huart3.Init.BaudRate = 19200;
  huart3.Init.WordLength = UART_WORDLENGTH_8B;
  huart3.Init.StopBits = UART_STOPBITS_1;
  huart3.Init.Parity = UART_PARITY_NONE;
  huart3.Init.Mode = UART_MODE_TX_RX;
  huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart3.Init.OverSampling = UART_OVERSAMPLING_16;
  huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
  huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
  if (HAL_UART_Init(&huart3) != HAL_OK)
  {
    _Error_Handler(__FILE__, __LINE__);
  }

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
     PC1   ------> ETH_MDC
     PA1   ------> ETH_REF_CLK
     PA2   ------> ETH_MDIO
     PC4   ------> ETH_RXD0
     PC5   ------> ETH_RXD1
     PB13   ------> ETH_TXD1
     PA8   ------> USB_OTG_FS_SOF
     PA9   ------> USB_OTG_FS_VBUS
     PA10   ------> USB_OTG_FS_ID
     PA11   ------> USB_OTG_FS_DM
     PA12   ------> USB_OTG_FS_DP
     PG11   ------> ETH_TX_EN
     PG13   ------> ETH_TXD0
*/
static void MX_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();
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOG_CLK_ENABLE();


  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, LD3_Pin|LD2_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(USB_PowerSwitchOn_GPIO_Port, USB_PowerSwitchOn_Pin, GPIO_PIN_RESET);

  /*Configure GPIO pin : USER_Btn_Pin */
  GPIO_InitStruct.Pin = USER_Btn_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USER_Btn_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : RMII_MDC_Pin RMII_RXD0_Pin RMII_RXD1_Pin */
  GPIO_InitStruct.Pin = RMII_MDC_Pin|RMII_RXD0_Pin|RMII_RXD1_Pin;
```

```
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : RMII_REF_CLK_Pin RMII_MDIO_Pin */
  GPIO_InitStruct.Pin = RMII_REF_CLK_Pin|RMII_MDIO_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA4 */
  GPIO_InitStruct.Pin = GPIO_PIN_6;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : RMII_TXD1_Pin */
  GPIO_InitStruct.Pin = RMII_TXD1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(RMII_TXD1_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : LD3_Pin LD2_Pin */
  GPIO_InitStruct.Pin = LD3_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(GPIOB, &GPIO_InitStruct);

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

  /*Configure GPIO pin : USB_OverCurrent_Pin */
  GPIO_InitStruct.Pin = USB_OverCurrent_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USB_OverCurrent_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : USB_SOF_Pin USB_ID_Pin USB_DM_Pin USB_DP_Pin */
  GPIO_InitStruct.Pin = USB_SOF_Pin|USB_ID_Pin|USB_DM_Pin|USB_DP_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : USB_VBUS_Pin */
  GPIO_InitStruct.Pin = USB_VBUS_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(USB_VBUS_GPIO_Port, &GPIO_InitStruct);

  /*Configure GPIO pins : RMII_TX_EN_Pin RMII_TXD0_Pin */
  GPIO_InitStruct.Pin = RMII_TX_EN_Pin|RMII_TXD0_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF11_ETH;
  HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error occurrence.
  * @param  file: The file name as string.
  * @param  line: The line in file as a number.
  * @retval None
  */
void _Error_Handler(char *file, int line)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  while(1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#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,
      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
   /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/**
  * @}
  */

/**
  * @}
  */

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

Thanks for all the help!