Hi. I try to multi channel ADC with DMA.

I use mini-stmf4 board made by mikro Elektronika.

I want to sampling 250Hz and send the data to UART.

So, I want to use single ADC mode.

The ADC1 - ch 0,1,2,3 are used.

The channel0 is no problem. But Another channels output '0' frequently. (input is not zero)

Why the '0' value is output between normal output?

My code attached below.

```
/**
  ******************************************************************************
  * File Name          : main.c
  * Description        : Main program body
  ******************************************************************************
  *
  * COPYRIGHT(c) 2017 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 "FNIRS_LIB.h"
#include "STM32F4_MINI_LIB.h"
/* USER CODE END Includes */

/* Private variables ---------------------------------------------------------*/
ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */
/* Private variables ---------------------------------------------------------*/
uint32_t value_adc[4];
uint32_t value_dac=0;


#define F_sample 150
#define Rise_time 80
#define falling_time 20
#define Sampling_rate 250 //Fs, 1000, 100,

int Fs= 1000/Sampling_rate;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_DMA_Init(void);
static void MX_ADC1_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_USB_OTG_FS_USB_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----------------------------------------------*/
void delay_us(uint32_t sysclk,uint32_t val);
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)  //for printf

/* USER CODE END PFP */

/* USER CODE BEGIN 0 */
PUTCHAR_PROTOTYPE
{
  HAL_UART_Transmit(&huart1, (uint8_t *)&ch, 1, 0xFFFF);
  return ch;
}
/* USER CODE END 0 */

int main(void)
{

  /* USER CODE BEGIN 1 */

  /* 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_ADC1_Init();
  MX_USART1_UART_Init();
  MX_USB_OTG_FS_USB_Init();

  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
 stm32_LED_STAT_ON();
 //HAL_ADC_Start_DMA(&hadc1,(uint32_t*)value_adc,4);

  while (1)
  {

  HAL_ADC_Start_DMA(&hadc1,(uint32_t*)value_adc,4);

  ENABLE_MAX4581();   //GPIO output for MUX
  ENABLE_MAX4581_PIN_A();   //GPIO output for MUX
  DISABLE_MAX4581_PIN_B();   //GPIO output for MUX
  DISABLE_MAX4581_PIN_C();   //GPIO output for MUX
  //
  //HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_SET);
  //
  for(int i=0; i<Fs; i++)
  {
  delay_us(168000,Rise_time);
  }
  HAL_GPIO_WritePin(GPIOA,GPIO_PIN_4,GPIO_PIN_SET);
    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,F_sample);
  }
   HAL_GPIO_WritePin(GPIOA,GPIO_PIN_4,GPIO_PIN_RESET);
   for(int i=0; i<Fs; i++)
  {
  delay_us(168000,falling_time);
  }
  //
  //HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_RESET);
  //
  DISABLE_MAX4581();

    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,Rise_time+falling_time);
  delay_us(168000,F_sample);
  }

  ENABLE_MAX4581();   //GPIO output for MUX
  DISABLE_MAX4581_PIN_A();   //GPIO output for MUX
```

```
  DISABLE_MAX4581_PIN_B();   //GPIO output for MUX
  DISABLE_MAX4581_PIN_C();    //GPIO output for MUX
  HAL_GPIO_WritePin(GPIOC,GPIO_PIN_10,GPIO_PIN_SET);
    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,Rise_time);
  }
  HAL_GPIO_WritePin(GPIOC,GPIO_PIN_11,GPIO_PIN_SET);
    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,F_sample);
  }
    HAL_GPIO_WritePin(GPIOC,GPIO_PIN_11,GPIO_PIN_RESET);
    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,falling_time);
  }
  HAL_GPIO_WritePin(GPIOC,GPIO_PIN_10,GPIO_PIN_RESET);
  DISABLE_MAX4581();
    for(int i=0; i<Fs; i++)
  {
  delay_us(168000,Rise_time+falling_time);
  delay_us(168000,F_sample);
  }


  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

  }
  /* USER CODE END 3 */

}

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

  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

  __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 = 10;
  RCC_OscInitStruct.PLL.PLLN = 210;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 7;
  HAL_RCC_OscConfig(&RCC_OscInitStruct);

  RCC_ClkInitStruct.ClockType = 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;
  HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_5);

  HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

  HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

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

/* ADC1 init function */
void MX_ADC1_Init(void)
{

  ADC_ChannelConfTypeDef sConfig;

    /**Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)
    */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION12b;
  hadc1.Init.ScanConvMode = ENABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 4;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = EOC_SEQ_CONV;
  HAL_ADC_Init(&hadc1);

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_15CYCLES;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_1;
  sConfig.Rank = 2;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_2;
  sConfig.Rank = 3;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);
```

```
    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_3;
  sConfig.Rank = 4;
  HAL_ADC_ConfigChannel(&hadc1, &sConfig);

}

/* USART1 init function */
void MX_USART1_UART_Init(void)
{

  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  HAL_UART_Init(&huart1);

}

/* USB_OTG_FS init function */
void MX_USB_OTG_FS_USB_Init(void)
{

}

/**
  * Enable DMA controller clock
  */
void MX_DMA_Init(void)
{
  /* DMA controller clock enable */
  __DMA2_CLK_ENABLE();

  /* DMA interrupt init */
  HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);

}

/** Configure pins as
        * Analog
        * Input
        * Output
        * EVENT_OUT
        * EXTI
     PA5   ------> COMP_DAC2_group
     PA9   ------> USB_OTG_FS_VBUS
     PA10   ------> USB_OTG_FS_ID
     PA11   ------> USB_OTG_FS_DM
     PA12   ------> USB_OTG_FS_DP
*/
void MX_GPIO_Init(void)
{
```

```
  GPIO_InitTypeDef GPIO_InitStruct;

  /* GPIO Ports Clock Enable */
  __GPIOC_CLK_ENABLE();
  __GPIOH_CLK_ENABLE();
  __GPIOA_CLK_ENABLE();
  __GPIOB_CLK_ENABLE();

  /*Configure GPIO pins : PC13 PC10 PC11 PC12 */
  GPIO_InitStruct.Pin = GPIO_PIN_13|GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

  /*Configure GPIO pins : PA4 PA13 PA14 */
  GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_13|GPIO_PIN_14;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pin : PA5 */
  GPIO_InitStruct.Pin = GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PB0 PB1 PB13 PB5 */
  GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_13|GPIO_PIN_5;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

  /*Configure GPIO pin : PA9 */
  GPIO_InitStruct.Pin = GPIO_PIN_9;
  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : PA10 PA11 PA12 */
  GPIO_InitStruct.Pin = GPIO_PIN_10|GPIO_PIN_11|GPIO_PIN_12;
  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_HIGH;
  GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
void delay_us(uint32_t sysclk,uint32_t val)
 {

  uint32_t us_tick,cur,last;

  switch(sysclk)
```

```
    {
     case 48000: us_tick = 48; break;
     case 80000: us_tick = 80; break;
     case 16000: us_tick = 16; break;
     case 32000: us_tick = 32; break;
     default : us_tick = sysclk / 1000;
    }
    val *= us_tick;
    last = SysTick->VAL;
      while(1)
      {
       cur = SysTick->VAL;
          if(cur > last) {
           if(last + (SysTick->LOAD + 1 - cur) >= val) break;
          }
          else if(last - cur >= val) break;
      }
  }
 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
  {
   uint8_t upper= 0xBE;
   uint8_t lower= 0xD7;
   uint8_t buff[8];
   //uint8_t CRC_buff=0;
  uint32_t val[4];
    val[0]=value_adc[0];
    val[1]=value_adc[1];
   val[2]=value_adc[2];
   val[3]=value_adc[3];

   buff[0]=val[0]>>8;
   buff[1]=0xFF&val[0];
   buff[2]=val[1]>>8;
   buff[3]=0xFF&val[1];
   buff[4]=val[2]>>8;
   buff[5]=0xFF&val[2];
   buff[6]=val[3]>>8;
   buff[7]=0xFF&val[3];

    HAL_UART_Transmit(&huart1, &upper, 1, HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart1, &lower, 1, HAL_MAX_DELAY);
    HAL_UART_Transmit(&huart1, &buff[0], 8, HAL_MAX_DELAY);
```

```
  HAL_ADC_Stop_DMA(&hadc1);
 }

/* USER CODE END 4 */

#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****/

void HAL_ADC_MspInit(ADC_HandleTypeDef* hadc)
{

GPIO_InitTypeDef GPIO_InitStruct;
if(hadc->Instance==ADC1)
{
/* USER CODE BEGIN ADC1_MspInit 0 */

/* USER CODE END ADC1_MspInit 0 */
/* Peripheral clock enable */
__ADC1_CLK_ENABLE();

/**ADC1 GPIO Configuration
PA0-WKUP ------> ADC1_IN0
PA1 ------> ADC1_IN1
PA2 ------> ADC1_IN2
PA3 ------> ADC1_IN3
*/
GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3;
GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
```

```
/* Peripheral DMA init*/

hdma_adc1.Instance = DMA2_Stream0;
hdma_adc1.Init.Channel = DMA_CHANNEL_0;
hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
hdma_adc1.Init.PeriphDataAlignment = DMA_PDATAALIGN_WORD;
hdma_adc1.Init.MemDataAlignment = DMA_MDATAALIGN_WORD;
hdma_adc1.Init.Mode = DMA_CIRCULAR;
hdma_adc1.Init.Priority = DMA_PRIORITY_VERY_HIGH;
hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
hdma_adc1.Init.FIFOThreshold = DMA_FIFO_THRESHOLD_FULL;
hdma_adc1.Init.MemBurst = DMA_MBURST_SINGLE;
hdma_adc1.Init.PeriphBurst = DMA_PBURST_SINGLE;
HAL_DMA_Init(&hdma_adc1);

__HAL_LINKDMA(hadc,DMA_Handle,hdma_adc1);

/* USER CODE BEGIN ADC1_MspInit 1 */

/* USER CODE END ADC1_MspInit 1 */
}

}

void HAL_ADC_MspDeInit(ADC_HandleTypeDef* hadc)
{

if(hadc->Instance==ADC1)
{
/* USER CODE BEGIN ADC1_MspDeInit 0 */

/* USER CODE END ADC1_MspDeInit 0 */
/* Peripheral clock disable */
__ADC1_CLK_DISABLE();

/**ADC1 GPIO Configuration
PA0-WKUP ------> ADC1_IN0
PA1 ------> ADC1_IN1
PA2 ------> ADC1_IN2
PA3 ------> ADC1_IN3
*/
HAL_GPIO_DeInit(GPIOA, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);

/* Peripheral DMA DeInit*/
HAL_DMA_DeInit(hadc->DMA_Handle);
}
/* USER CODE BEGIN ADC1_MspDeInit 1 */

/* USER CODE END ADC1_MspDeInit 1 */

}
```

EDIT