Hello Sir. I have a problem Sir in adding manually ADC(using DMA). First, i've generated ADC(using DMA) using STM32CubeMX. After that, i plan to adding manually ADC inside the main.c
But, it didn't work at all.
Could you help me, how to add ADC(using DMA) manually in the main.c ?
Oh yeah, i've done these things, i need to read 11 number of conversion ADC. Then i plan to add 1 more, just 1 more.
So i add :

```
  /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
  */
sConfig.Channel = ADC_CHANNEL_11;
sConfig.Rank = 12;
if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
{
  Error_Handler();
}
```

Then i change these 3 :

```
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();

  /* USER CODE BEGIN 2 */
HAL_ADC_Start_DMA(&hadc1,(uint32_t *)ADC_buffer,10);
  /* USER CODE END 2 */

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

  /* USER CODE BEGIN 3 */
```

and

```
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();

  /* USER CODE BEGIN 2 */
HAL_ADC_Start_DMA(&hadc1,(uint32_t *)ADC_buffer,10);
  /* USER CODE END 2 */

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

  /* USER CODE BEGIN 3 */
```

and

```
41  /* Private variables ---------------------------------------------------------*/
42  ADC_HandleTypeDef hadc1;
43  DMA_HandleTypeDef hdma_adc1;
44
45  /* USER CODE BEGIN PV */
46  /* Private variables ---------------------------------------------------------*/
47  uint8_t ADC_buffer[10];
48  /* USER CODE END PV */
49
50  /* Private function prototypes -----------------------------------------------*/
51  void SystemClock_Config(void);
52  void Error_Handler(void);
53  static void MX_GPIO_Init(void);
54  static void MX_DMA_Init(void);
55  static void MX_ADC1_Init(void);
56
57  /* USER CODE BEGIN PFP */
58  /* Private function prototypes -----------------------------------------------*/
59
```

And here's my code :

```
001. /**
002.   ******************************************************************************
003.   * File Name          : main.c
004.   * Description        : Main program body
005.   ******************************************************************************
006.   *
007.   * COPYRIGHT(c) 2016 STMicroelectronics
008.   *
009.   * Redistribution and use in source and binary forms, with or without modification,
010.   * are permitted provided that the following conditions are met:
011.   *   1. Redistributions of source code must retain the above copyright notice,
012.   *      this list of conditions and the following disclaimer.
013.   *   2. Redistributions in binary form must reproduce the above copyright notice,
014.   *      this list of conditions and the following disclaimer in the documentation
015.   *      and/or other materials provided with the distribution.
016.   *   3. Neither the name of STMicroelectronics nor the names of its contributors
017.   *      may be used to endorse or promote products derived from this software
018.   *      without specific prior written permission.
019.   *
020.   * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
021.   * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
022.   * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
023.   * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
024.   * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
025.   * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
026.   * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
027.   * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
028.   * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
029.   * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
030.   *
031.   ******************************************************************************
032.   */
033. /* Includes ------------------------------------------------------------------*/
034. #include "main.h"
035. #include "stm32f4xx_hal.h"
036.
037. /* USER CODE BEGIN Includes */
038.
039. /* USER CODE END Includes */
040.
041. /* Private variables ---------------------------------------------------------*/
```

```
042. ADC_HandleTypeDef hadc1;
043. DMA_HandleTypeDef hdma_adc1;
044.
045. /* USER CODE BEGIN PV */
046. /* Private variables ---------------------------------------------------------*/
047. uint8_t ADC_buffer[11];
048. /* USER CODE END PV */
049.
050. /* Private function prototypes -----------------------------------------------*/
051. void SystemClock_Config(void);
052. void Error_Handler(void);
053. static void MX_GPIO_Init(void);
054. static void MX_DMA_Init(void);
055. static void MX_ADC1_Init(void);
056.
057. /* USER CODE BEGIN PFP */
058. /* Private function prototypes -----------------------------------------------*/
059.
060. /* USER CODE END PFP */
061.
062. /* USER CODE BEGIN 0 */
063.
064. /* USER CODE END 0 */
065.
066. int main(void)
067. {
068.
069.   /* USER CODE BEGIN 1 */
070.
071.   /* USER CODE END 1 */
072.
073.   /* MCU Configuration--------------------------------------------------------*/
074.
075.   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
076.   HAL_Init();
077.
078.   /* Configure the system clock */
079.   SystemClock_Config();
080.
081.   /* Initialize all configured peripherals */
082.   MX_GPIO_Init();
083.   MX_DMA_Init();
084.   MX_ADC1_Init();
085.
086.   /* USER CODE BEGIN 2 */
087. HAL_ADC_Start_DMA(&hadc1,(uint32_t *)ADC_buffer,11);
088.   /* USER CODE END 2 */
089.
090.   /* Infinite loop */
091.   /* USER CODE BEGIN WHILE */
092.   while (1)
093.   {
094.   /* USER CODE END WHILE */
095.
096.   /* USER CODE BEGIN 3 */
097.
098.   }
099.   /* USER CODE END 3 */
100.
101. }
102.
103. /** System Clock Configuration
104. */
105. void SystemClock_Config(void)
106. {
107.
```

```
  RCC_OscInitTypeDef RCC_OscInitStruct;
  RCC_ClkInitTypeDef RCC_ClkInitStruct;

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

  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**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_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

    /**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_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }

    /**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);
}

/* ADC1 init function */
static 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_CLOCK_SYNC_PCLK_DIV2;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  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 = 12;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
```

```
    Error_Handler();
  }

    /**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_3CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**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;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**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;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**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;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_4;
  sConfig.Rank = 5;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_5;
  sConfig.Rank = 6;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

    /**Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
    */
  sConfig.Channel = ADC_CHANNEL_6;
  sConfig.Rank = 7;
```

```
237.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
238.   {
239.     Error_Handler();
240.   }
241.
242.     /**Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
243.     */
244.   sConfig.Channel = ADC_CHANNEL_7;
245.   sConfig.Rank = 8;
246.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
247.   {
248.     Error_Handler();
249.   }
250.
251.     /**Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
252.     */
253.   sConfig.Channel = ADC_CHANNEL_8;
254.   sConfig.Rank = 9;
255.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
256.   {
257.     Error_Handler();
258.   }
259.
260.     /**Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
261.     */
262.   sConfig.Channel = ADC_CHANNEL_9;
263.   sConfig.Rank = 10;
264.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
265.   {
266.     Error_Handler();
267.   }
268.
269.     /**Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
270.     */
271.   sConfig.Channel = ADC_CHANNEL_10;
272.   sConfig.Rank = 11;
273.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
274.   {
275.     Error_Handler();
276.   }
277.
278.     /**Configure for the selected ADC regular channel its corresponding rank in the sequencer
and its sample time.
279.     */
280.   sConfig.Channel = ADC_CHANNEL_11;
281.   sConfig.Rank = 12;
282.   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
283.   {
284.     Error_Handler();
285.   }
286. }
287.
288. /**
289.   * Enable DMA controller clock
290.   */
291. static void MX_DMA_Init(void)
292. {
293.   /* DMA controller clock enable */
294.   __HAL_RCC_DMA2_CLK_ENABLE();
295.
296.   /* DMA interrupt init */
297.   /* DMA2_Stream0_IRQn interrupt configuration */
298.   HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
299.   HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);
300.
```

```
301. }
302.
303. /** Configure pins as
304.         * Analog
305.         * Input
306.         * Output
307.         * EVENT_OUT
308.         * EXTI
309. */
310. static void MX_GPIO_Init(void)
311. {
312.
313.   GPIO_InitTypeDef GPIO_InitStruct;
314.
315.   /* GPIO Ports Clock Enable */
316.   __HAL_RCC_GPIOC_CLK_ENABLE();
317.   __HAL_RCC_GPIOA_CLK_ENABLE();
318.   __HAL_RCC_GPIOB_CLK_ENABLE();
319.   __HAL_RCC_GPIOD_CLK_ENABLE();
320.
321.   /*Configure GPIO pin Output Level */
322.   HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15, GPIO_PIN_RESET);
323.
324.   /*Configure GPIO pins : PD12 PD13 PD14 PD15 */
325.   GPIO_InitStruct.Pin = GPIO_PIN_12|GPIO_PIN_13|GPIO_PIN_14|GPIO_PIN_15;
326.   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
327.   GPIO_InitStruct.Pull = GPIO_NOPULL;
328.   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
329.   HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
330.
331. }
332.
333. /* USER CODE BEGIN 4 */
334.
335. /* USER CODE END 4 */
336.
337. /**
338.   * @brief  This function is executed in case of error occurrence.
339.   * @param  None
340.   * @retval None
341.   */
342. void Error_Handler(void)
343. {
344.   /* USER CODE BEGIN Error_Handler */
345.   /* User can add his own implementation to report the HAL error return state */
346.   while(1)
347.   {
348.   }
349.   /* USER CODE END Error_Handler */
350. }
351.
352. #ifdef USE_FULL_ASSERT
353.
354. /**
355.    * @brief Reports the name of the source file and the source line number
356.    * where the assert_param error has occurred.
357.    * @param file: pointer to the source file name
358.    * @param line: assert_param error line source number
359.    * @retval None
360.    */
361. void assert_failed(uint8_t* file, uint32_t line)
362. {
363.   /* USER CODE BEGIN 6 */
364.   /* User can add his own implementation to report the file name and line number,
365.     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
366.   /* USER CODE END 6 */
367.
368. }
```

```
369.
370. #endif
371.
372. /**
373.   * @}
374.   */
375.
376. /**
377.   * @}
378. */
379.
380. /************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```

Thanks before :)