

Hi,

I have been trying to make the ADC work on an STM32L476RG nucleo board. No success so far, things don't work as expected.

I am testing the ADC + DMA by enabling 4 channels and configuring the ADC as follows:

The screenshot shows the STM32CubeMX software interface for configuring the ADC settings. The top navigation bar includes links for Parameter Settings, User Constants, NVIC Settings, DMA Settings, and GPIO Settings. Below the navigation bar, a search bar is present. The main configuration area is titled "Configure the below parameters:" and contains a tree view of configuration sections. The "Mode" section is expanded, showing the following settings:

Setting	Value
Mode	Independent mode
Clock Prescaler	Asynchronous clock mode divided by 64
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	End of sequence of conversion
Overrun behaviour	Overrun data overwritten
Low Power Auto Wait	Disabled

Other collapsed sections include ADC\_Common\_Settings, ADC\_Regular\_ConversionMode, ADC\_Injected\_ConversionMode, Analog Watchdog 1, and Analog Watchdog 2. A checkbox for "Enable Injected Conversions" is shown under the ADC\_Injected\_ConversionMode section.

For the DMA:

DMA Request	Channel	Direction	Priority
ADC1	ADC1	Peripheral To Memory	Medium

**DMA Request Settings**

Mode	Increment Address	Peripheral	Memory
Circular	<input type="checkbox"/>	<input checked="" type="checkbox"/>	
		Half Word	Word

**Apply    Ok    Cancel**

Here is main.c (very simple):

```
/*
*****
* File Name      : main.c
* Description    : Main program body
*****
** This notice applies to any and all portions of this file
* that are not between comment pairs USER CODE BEGIN and
* USER CODE END. Other portions of this file, whether
* inserted by the user or by software development tools
* are owned by their respective copyright owners.
*
* COPYRIGHT(c) 2018 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 "main.h"  
#include "stm32l4xx_hal.h"  
#include "adc.h"  
#include "dma.h"  
#include "i2c.h"  
#include "gpio.h"
```

```
/* USER CODE BEGIN Includes */
```

```
/* USER CODE END Includes */
```

```
/* Private variables ----- */
```

```
/* USER CODE BEGIN PV */  
/* Private variables ----- */  
uint32_t ADCResults1[4];  
  
uint8_t adcReady = 0;
```

```
/* USER CODE END PV */
```

```
/* Private function prototypes ----- */  
void SystemClock_Config(void);
```

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

```
/* USER CODE END PFP */
```

```
/* USER CODE BEGIN 0 */
```

```
/* 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();
```

```
/* USER CODE BEGIN Init */
```

```
/* USER CODE END Init */
```

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

```
/* USER CODE BEGIN SysInit */
```

```
/* USER CODE END SysInit */
```

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_I2C1_Init();
MX_ADC1_Init();
```

```
/* USER CODE BEGIN 2 */
```

```
HAL_ADC_Start_DMA(&hadc1, ADCResults1, 4);
/* USER CODE END 2 */
```

```
/* Infinite loop */
```

```
/* USER CODE BEGIN WHILE */
```

```

while (1)
{
/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
    HAL_Delay(200);
    if (adcReady)
    {
        printf("ADC results %d %d %d %d \r\n", ADCResults1[0], ADCResults1[1],
ADCResults1[2], ADCResults1[3]);
        adcReady = 0;
    }

}
/* USER CODE END 3 */
}

```

/\*\* System Clock Configuration

```

*/
void SystemClock_Config(void)
{

```

```

RCC_OscInitTypeDef RCC_OscInitStruct;
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_PeriphCLKInitTypeDef PeriphClkInit;

```

/\*\*Initializes the CPU, AHB and APB busses clocks

\*/

```

RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSISState = 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 = 1;
RCC_OscInitStruct.PLL.PLLN = 10;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
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.AHCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

```

```

if(HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_3) != HAL_OK)
{
    _Error_Handler(__FILE__, __LINE__);
}

PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_I2C1|RCC_PERIPHCLK_ADC;
PeriphClkInit.I2c1ClockSelection = RCC_I2C1CLKSOURCE_PCLK1;
PeriphClkInit.AdclkSelection = RCC_ADCCLKSOURCE_PLLSAI1;
PeriphClkInit.PLLSAI1.PLLSAI1Source = RCC_PLLSOURCE_HSI;
PeriphClkInit.PLLSAI1.PLLSAI1M = 1;
PeriphClkInit.PLLSAI1.PLLSAI1N = 8;
PeriphClkInit.PLLSAI1.PLLSAI1P = RCC_PLLP_DIV7;
PeriphClkInit.PLLSAI1.PLLSAI1Q = RCC_PLLQ_DIV2;
PeriphClkInit.PLLSAI1.PLLSAI1R = RCC_PLLR_DIV8;
PeriphClkInit.PLLSAI1.PLLSAI1ClockOut = RCC_PLLSAI1_ADC1CLK;
if(HAL_RCCEX_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
{
    _Error_Handler(__FILE__, __LINE__);
}

/**Configure the main internal regulator output voltage
 */
if(HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != 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);
}

/* USER CODE BEGIN 4 */
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    adcReady = 1;
}
/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.

```

```

 * @param None
 * @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,
   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\*\*\*\*/

I have tried with and without using the adcReady flag, same results.

Here is what the output looks like:

```
ARM Semihosting Console
ADC results 525 507 471 579
ADC results 443 447 448 436
ADC results 439 446 445 445
ADC results 523 508 723 580
ADC results 442 443 445 444
ADC results 4024 1149 438 439
ADC results 4024 1150 723 578
ADC results 526 506 465 574
ADC results 521 1150 725 580
ADC results 4024 1150 445 443
ADC results 440 504 462 449
ADC results 452 503 468 447
ADC results 522 1150 723 580
ADC results 443 445 444 439
ADC results 4024 446 445 443
ADC results 441 439 445 445
ADC results 4024 1149 724 439
ADC results 452 440 440 436
ADC results 526 506 724 574
ADC results 531 508 465 575
ADC results 523 503 723 579
ADC results 442 440 435 441
ADC results 524 503 471 455
ADC results 4025 441 439 444
ADC results 445 501 466 446
ADC results 4025 1151 444 441
ADC results 436 437 439 443
ADC results 526 1151 729 579
ADC results 452 441 442 447
ADC results 526 504 463 579
ADC results 447 447 466 445
ADC results 438 443 445 444
ADC results 4024 1147 723 577
ADC results 449 441 441 438
ADC results 441 443 442 438
ADC results 4025 441 441 440
ADC results 446 442 439 445
```

- I have connected my Channel 1 to 3.3V, which matches the >4000 values shown on the picture above. Channel 2 is connected to ground. Channels 3 and 4 are floating. The >4000 values are not shown in every row. Why?
- I have swapped ranks in the CubeMX configuration to see if I could get the value to move to other positions on the array but it has not made any difference.
- The only way to get the first value to be >4000 consistently is to stop the dma manually and call it again. The other values of the array always stay around those values, no matter what voltage I apply to the pins.
- Has it got something to do with sampling time and clocks? My ADC runs at 16 MHz. The STM32L476 runs at 80MHz. I have used different clocks, prescalers, and sampling times. Same results or worst case the program gets caught in the DMA routine when it's run too often.

Any help would be greatly appreciated, I have tried multiple ways but none has worked so far. Many examples are for the STM32F4 which have a slightly different DMA, or are for the STM32L4 family but only use one channel!

My next step will be to use interrupts instead of DMA, but the HAL library is not really made for multiple channels when not using DMA.

Thanks!