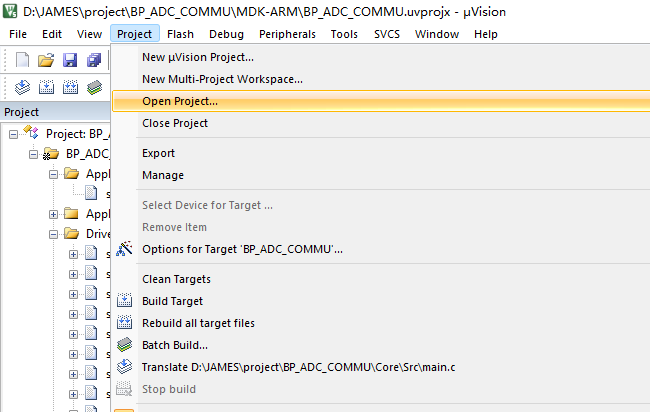
# Multi-ADC on Blue Pill (bp)

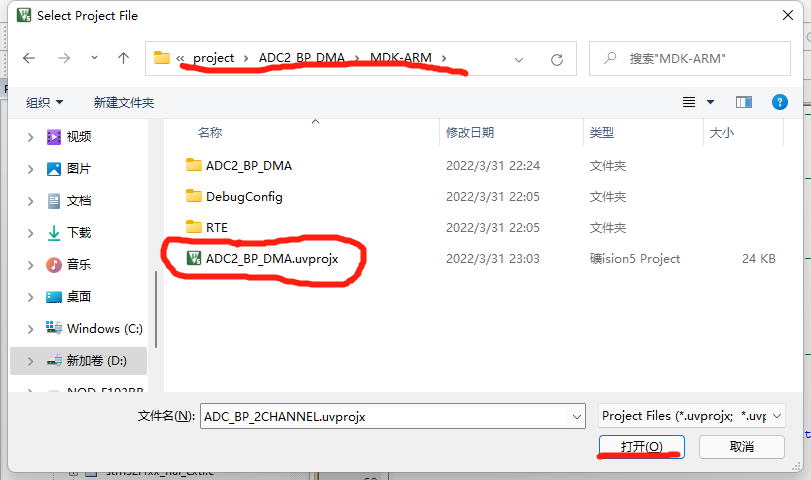
The code is in ADC2\_BP\_DMA.zip at

<https://github.com/Coin-Hou/Arduino_project_for_load_bank/tree/main/ADC_SAMPLING_BLUE-PILL>

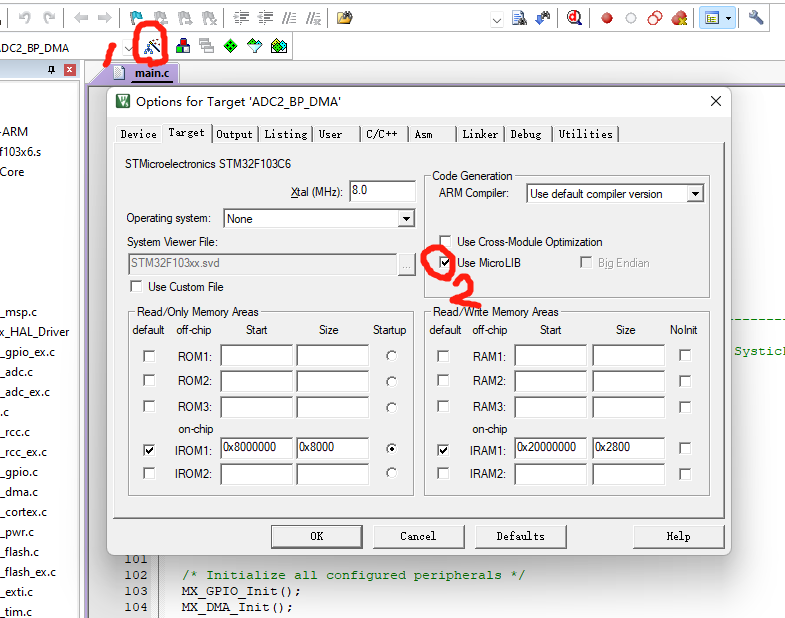
After download and unzip the .zip file, you can open the project on Keil by

* Open the project

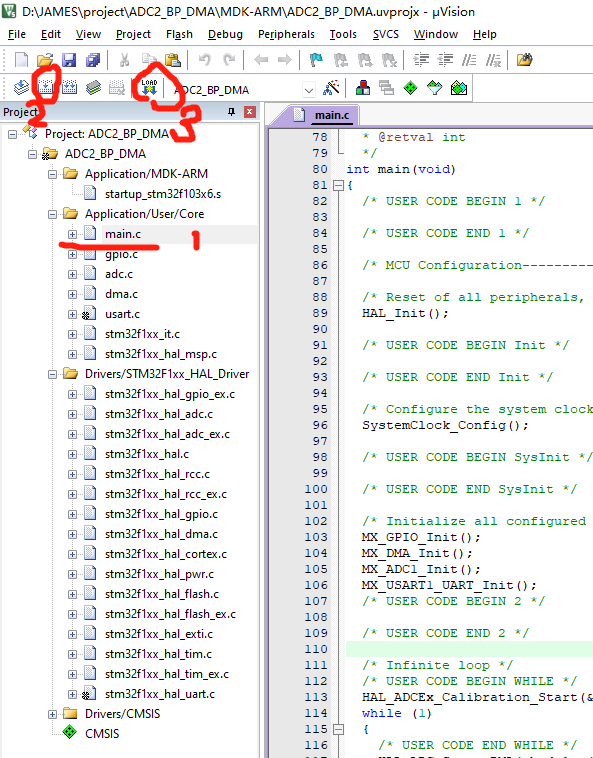




* Check the USE MicroLIB



* Open the main.c, build and download the program



By using DMA for multi-ADC sampling (continuous mode) , the bp can automatically sample the voltage and save it in the register.

There's an another multi-ADC sampling strategy (discontinuous mode) which requires to open the ADC conversion and close that in each single conversion.

I have tried the both methods and find that continuous mode has a steady performance than discontinuous mode. The bp will stuck as the runtime increases in discontinuous mode.

## Hardware

PA0-PA9->ADC1/2\_IN0- ADC1/2\_IN9 for ADC sampling

PA9->RX (for USART)

PA10->TX (for USART)

GND->SWD 3

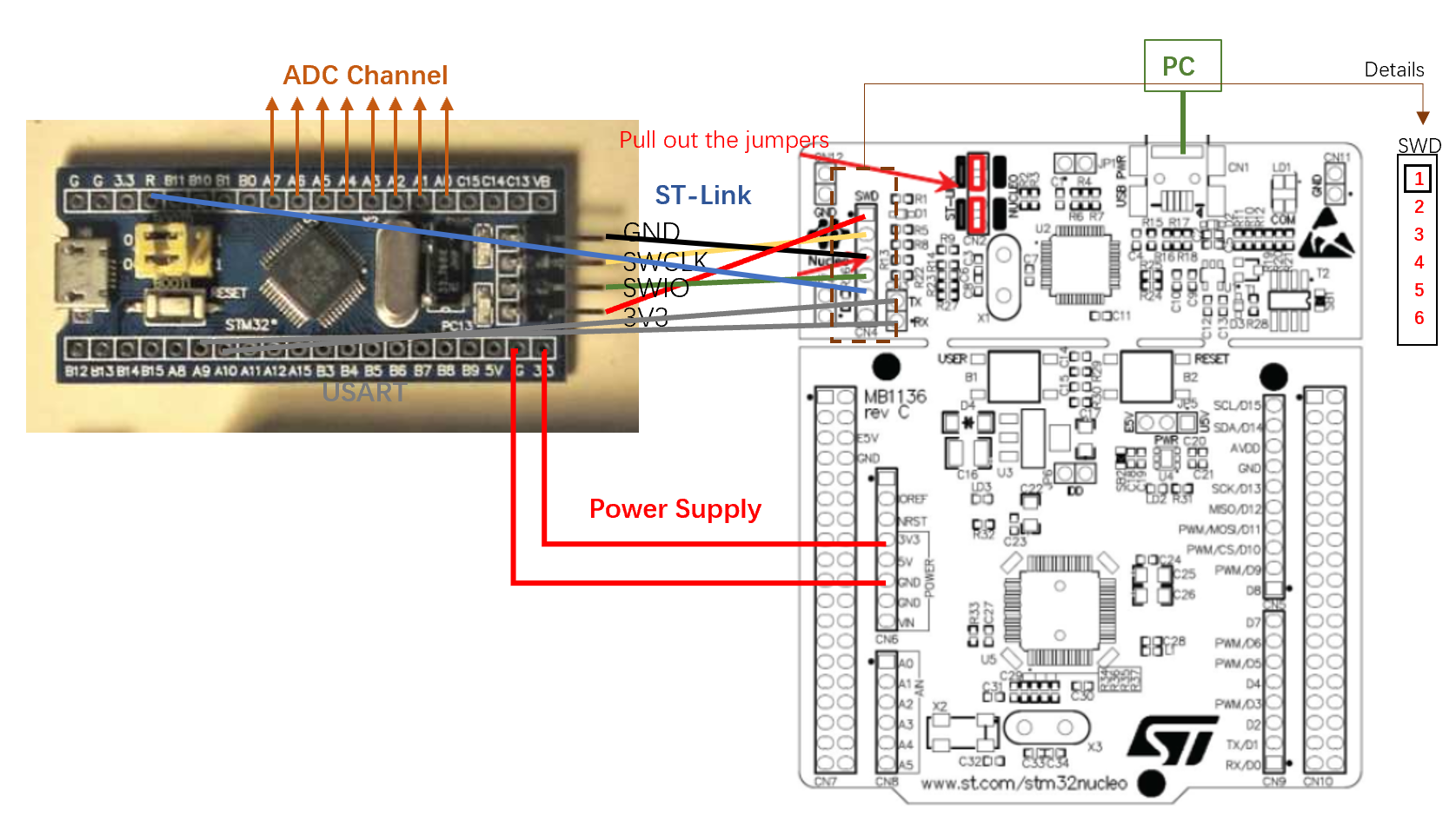
SWCLK->SWD 2

SWIO->SWD 4

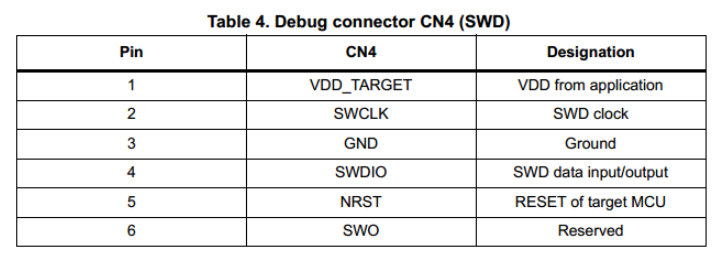
3V3->SWD 1

R->SWD 5

St-link和bluepill连接引脚，迁移一下以前的文档，jumper要拔掉pull out，蓝色线颜色不一样，代码load到git上

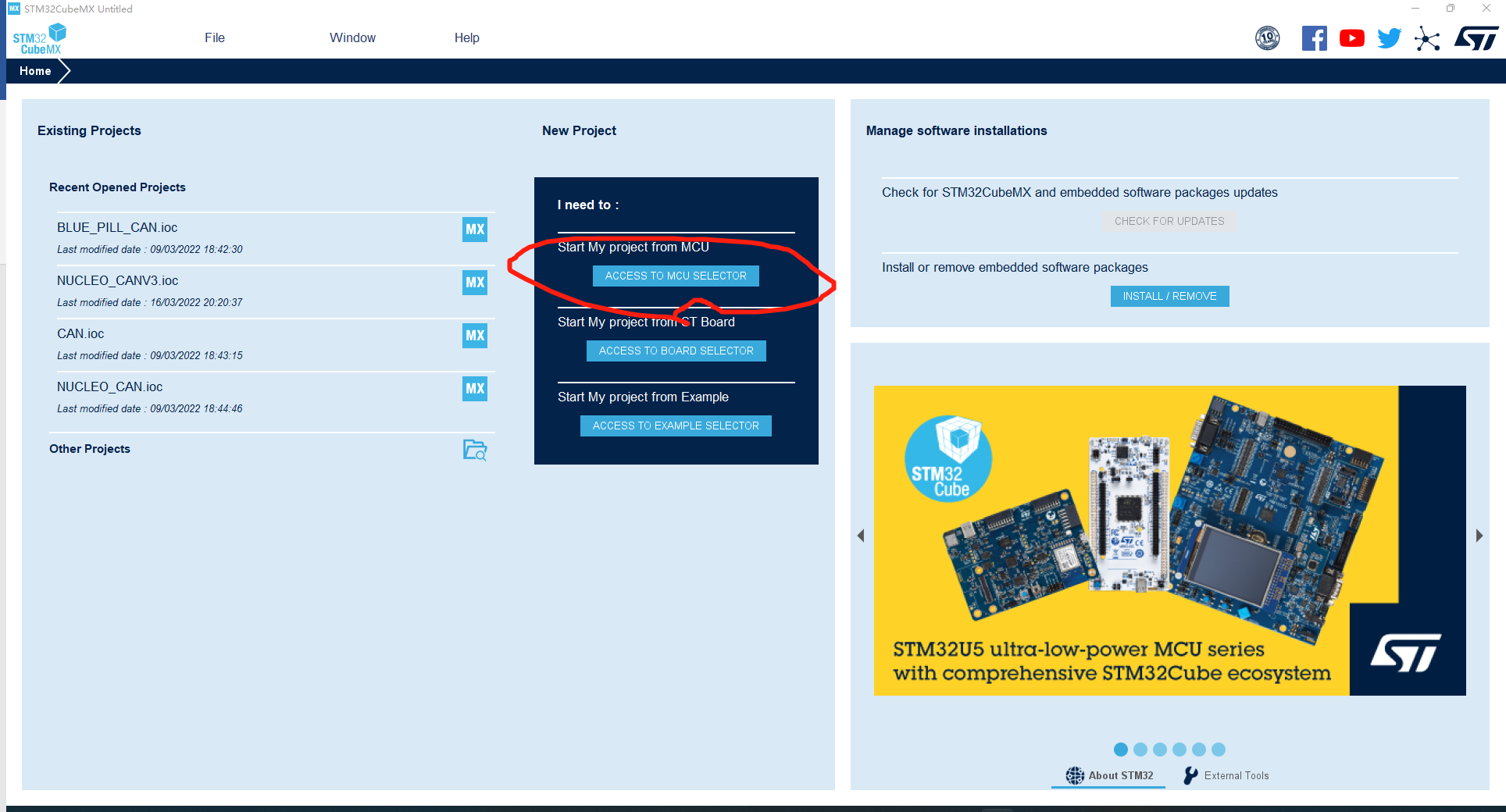


Details for the PINs of SWD on the ST-LINK

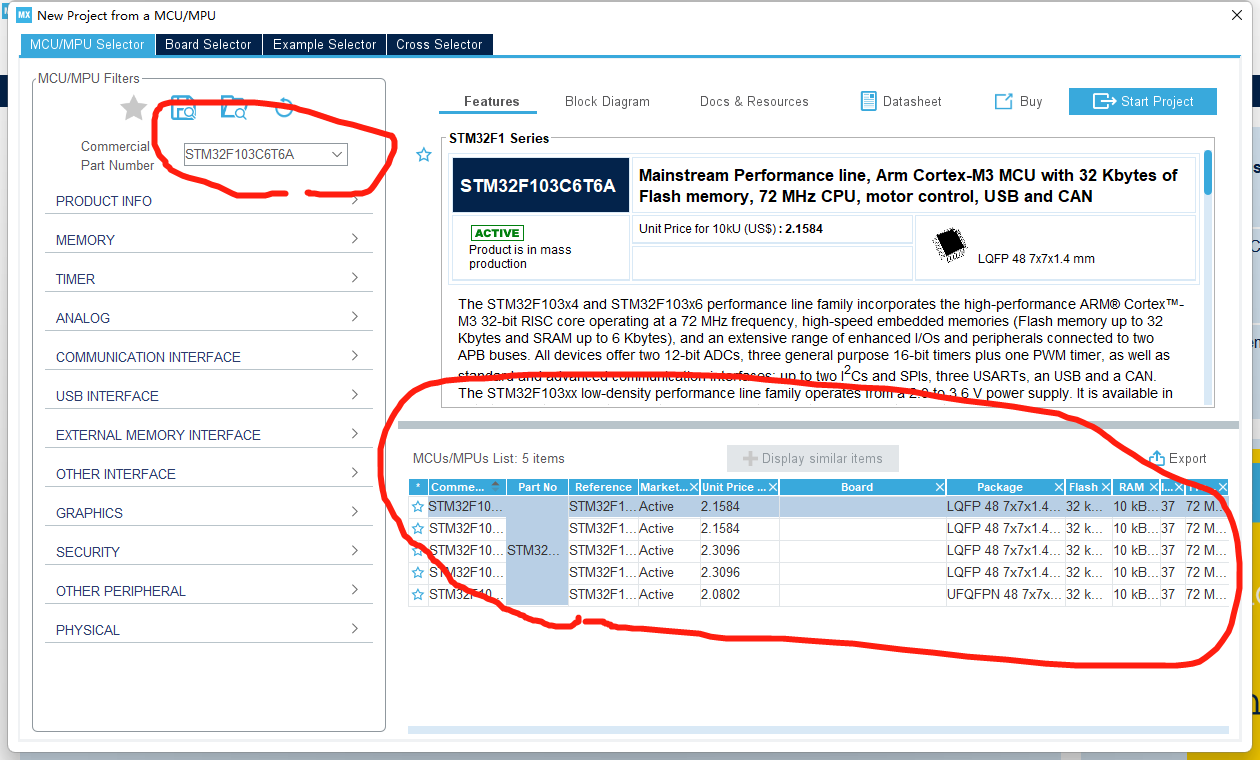


## Software

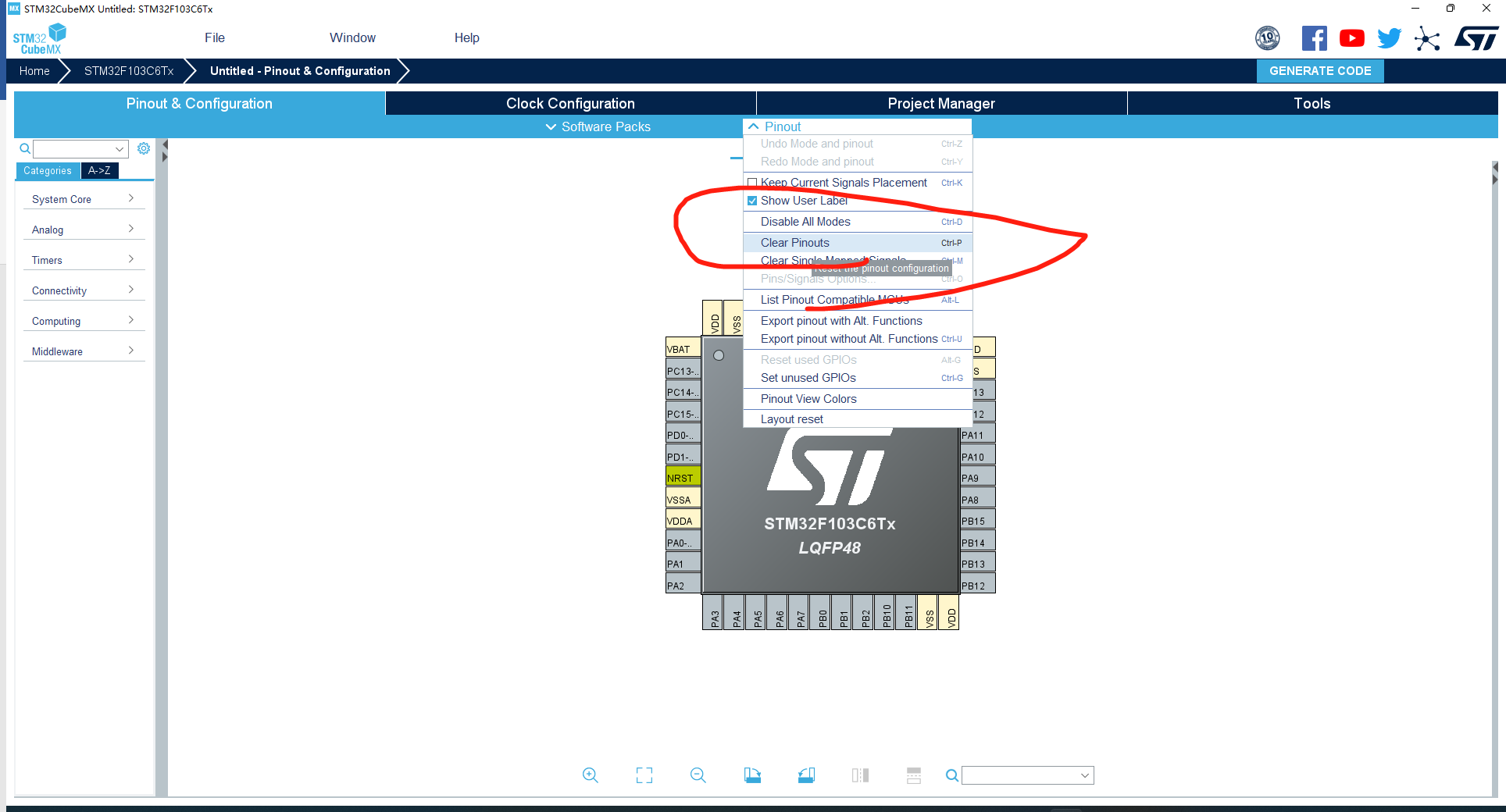
1. Open STM32CUBEMX, click access to mcu selector



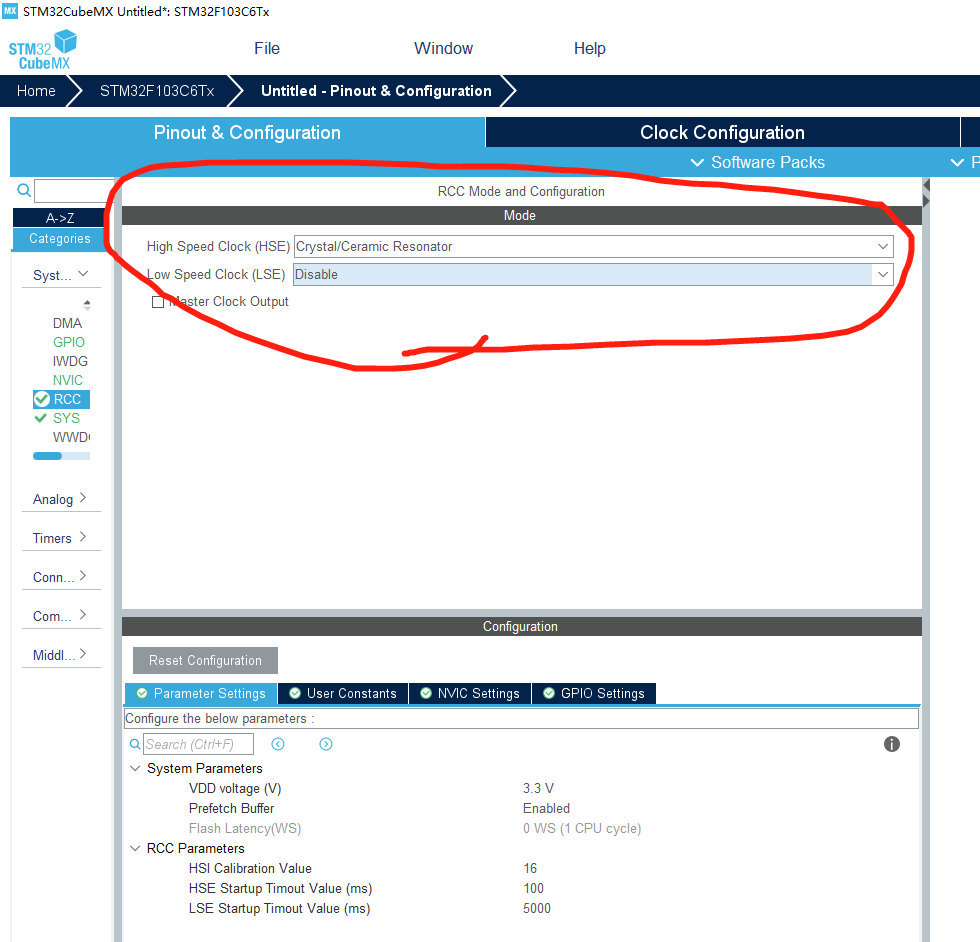
1. Select MCU STM32F103C6T6A, then click start project



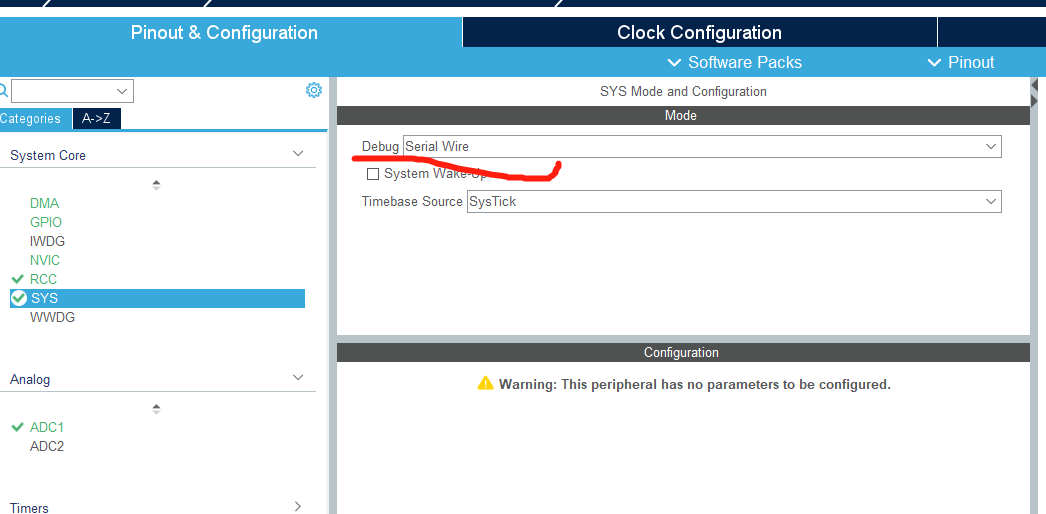
1. Click Pinout, choose clear pinouts



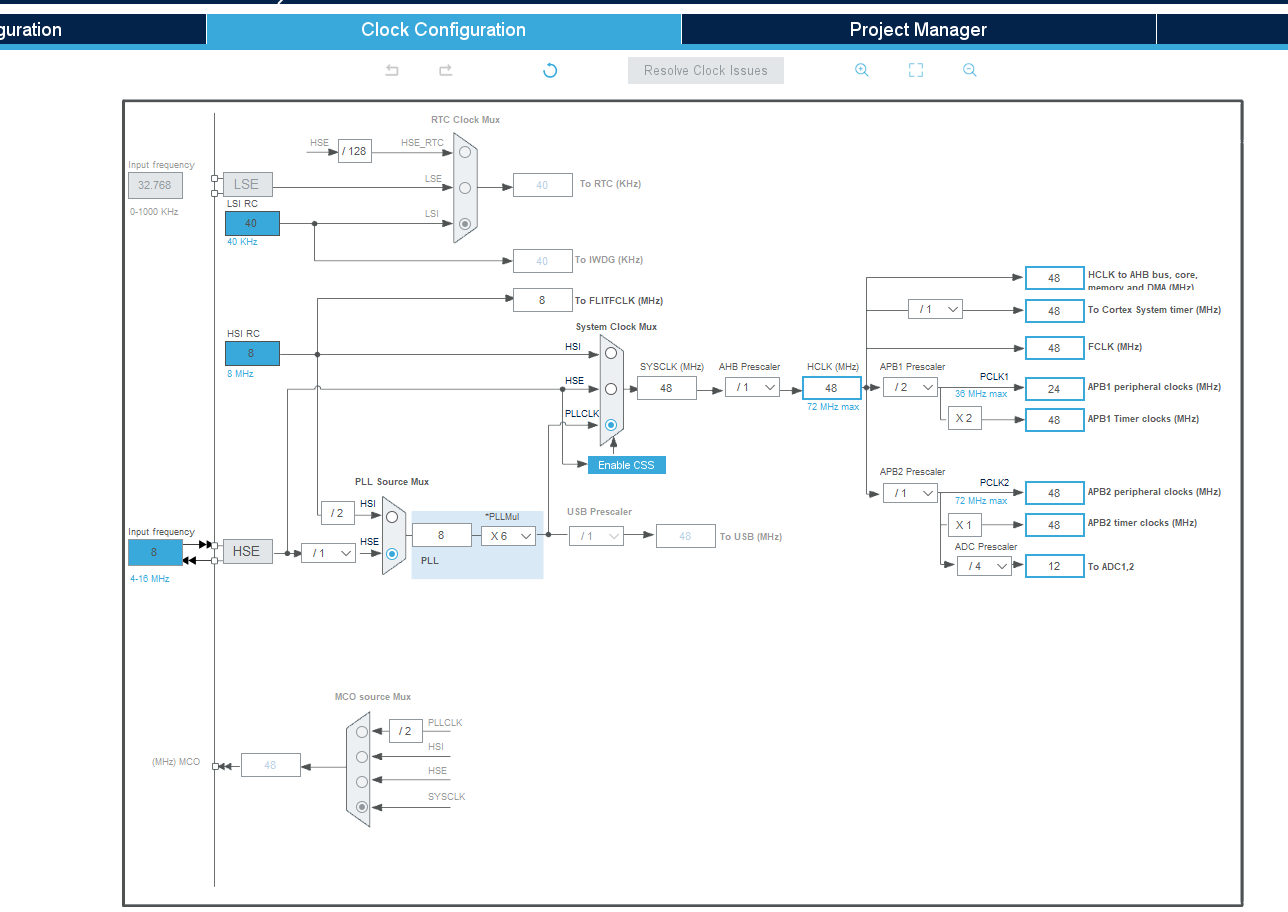
1. Configure RCC



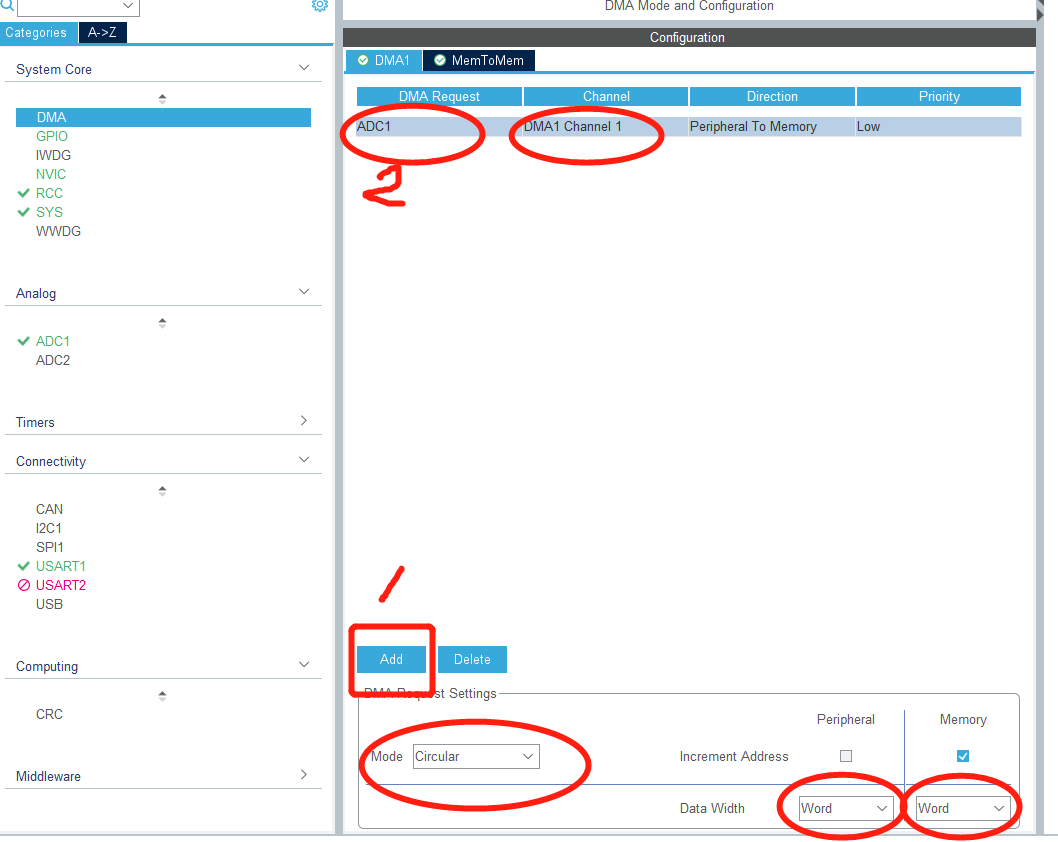
1. Configure SYS (for debug)



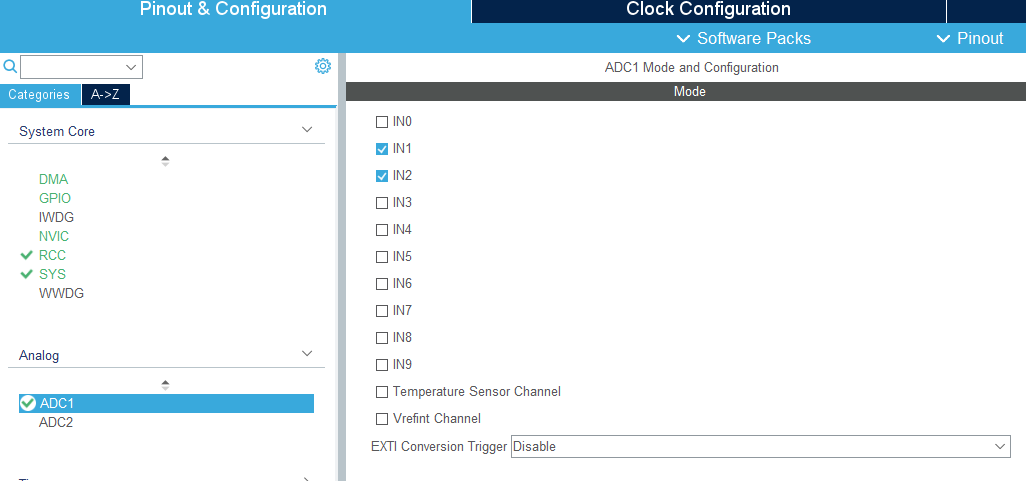
1. Clock configuration, the maximum of ADC clock is 14MHz.



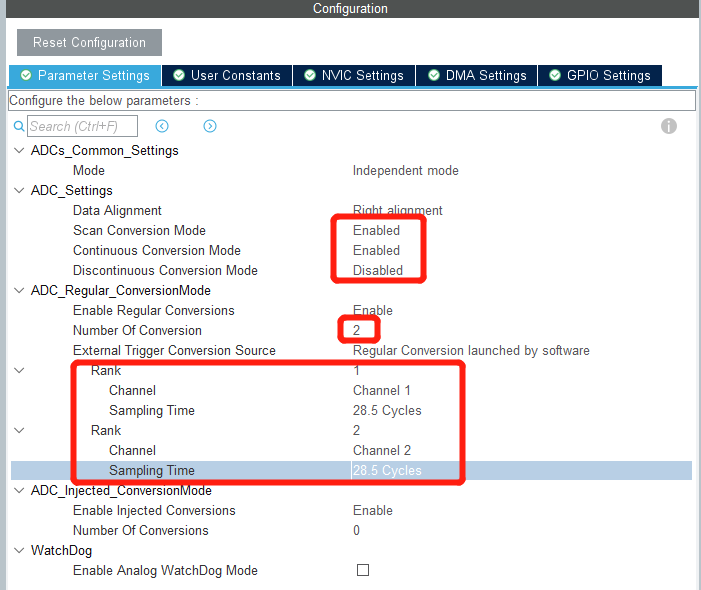
1. Config DMA



1. Config ADC, here I choose ADC1 and use 1/2 channel



Parameter settings for ADC1 IN1/IN2

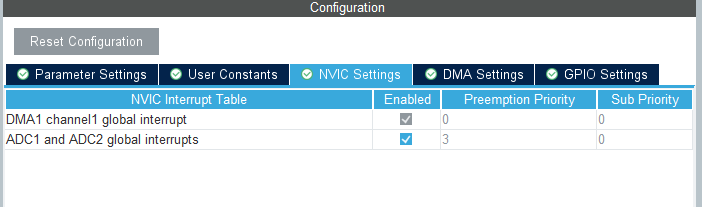


Sampling time for each channel

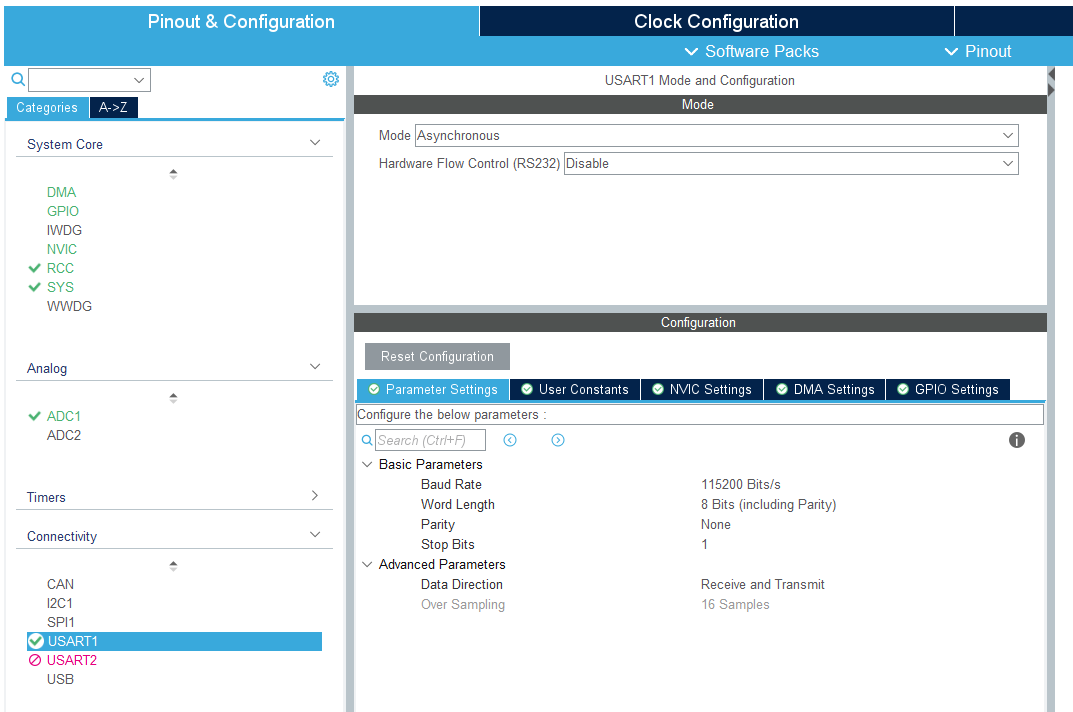
Number of ADC channels

For each conversion, it requires ***t=sampling time + 12.5 cycles***

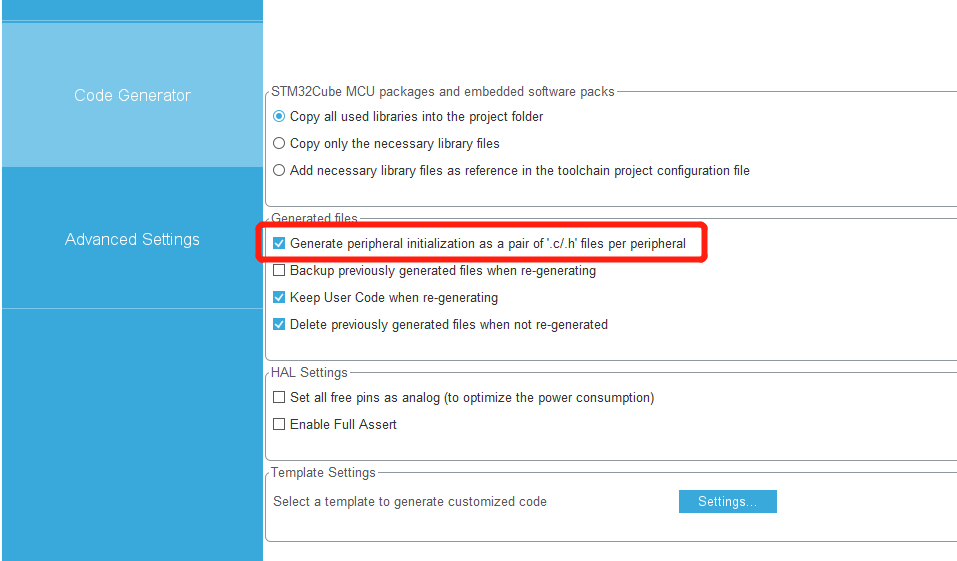
Set NVIC, it seems useless.



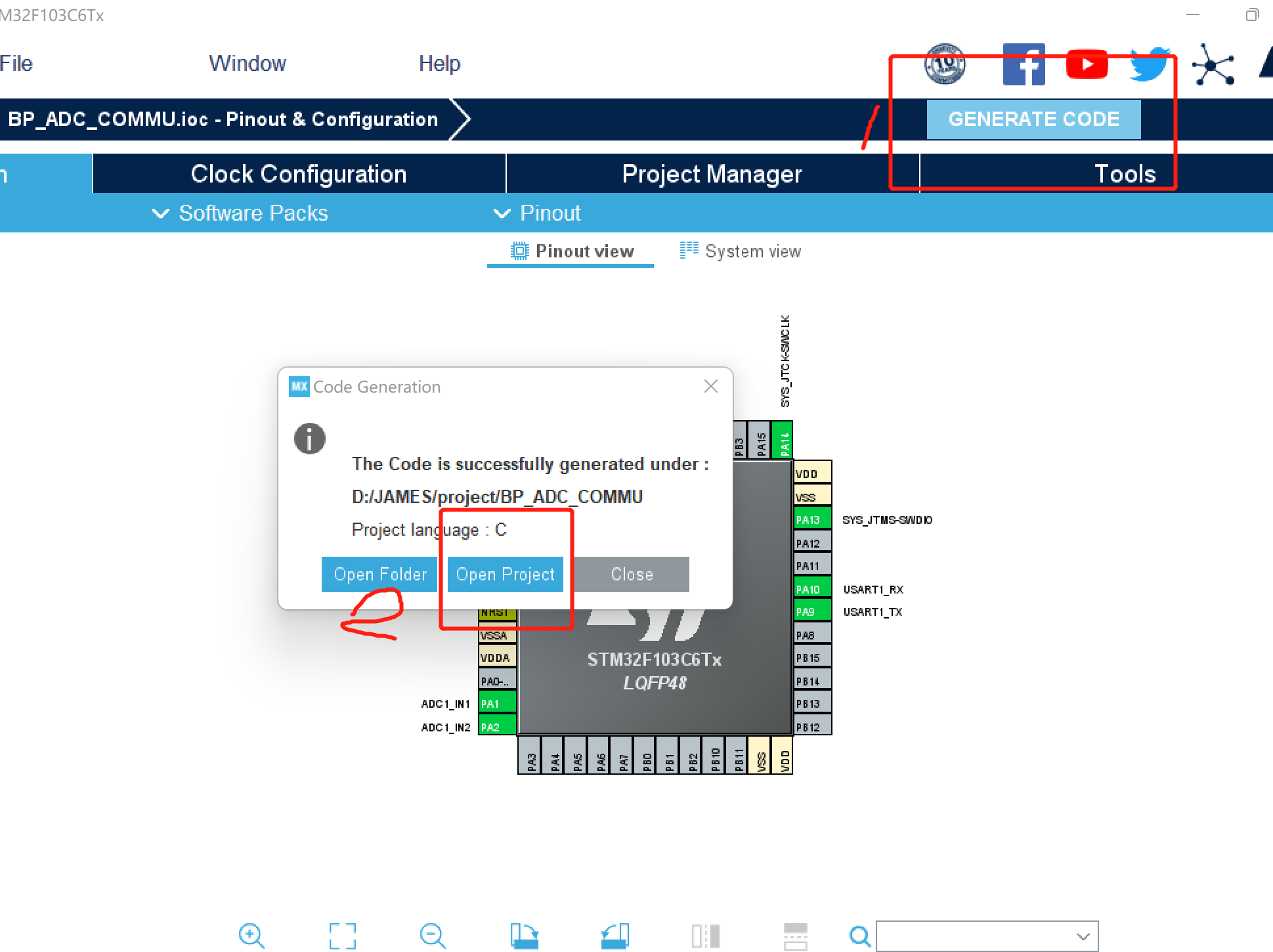
1. We also add USART



1. Generate the code, checking the option in the red box enables different functions (such as ADC, USART) of bp to generate .c files separately.



Click the GENERATE CODE and then click Open project. After that, we will see the STM32 code in Keil IDE.



1. The additional codes are shown below, with identified USER CODE REGION
2. /\* USER CODE BEGIN Includes \*/

#include "stdio.h"

/\* USER CODE END Includes \*/

1. /\* Private variables ---------------------------------------------------------\*/

/\* USER CODE BEGIN PV \*/

uint32\_t AD\_DMA[2];

/\* USER CODE END PV \*/

1. /\* USER CODE BEGIN PFP \*/

#ifdef \_\_GNUC\_\_

#define PUTCHAR\_PROTOTYPE int \_\_io\_putchar(int ch)

#else

#define PUTCHAR\_PROTOTYPE int fputc(int ch, FILE \*f)

#endif /\* \_\_GNUC\_\_ \*/

int fputc(int ch,FILE \*f)

{

HAL\_UART\_Transmit(&huart1,(uint8\_t \*)&ch,1,0xFFFF);

return ch;

}

/\* USER CODE END PFP \*/

1. /\* USER CODE BEGIN WHILE \*/

HAL\_ADCEx\_Calibration\_Start(&hadc1);

while (1)

{

/\* USER CODE END WHILE \*/

HAL\_ADC\_Start\_DMA(&hadc1, (uint32\_t\*)&AD\_DMA, 2); //

printf("AD\_DMA\_1 = %fV\r\n",AD\_DMA[0]\*3300/4096/1000.0);

printf("AD\_DMA\_2 = %fV\r\n",AD\_DMA[1]\*3300/4096/1000.0);

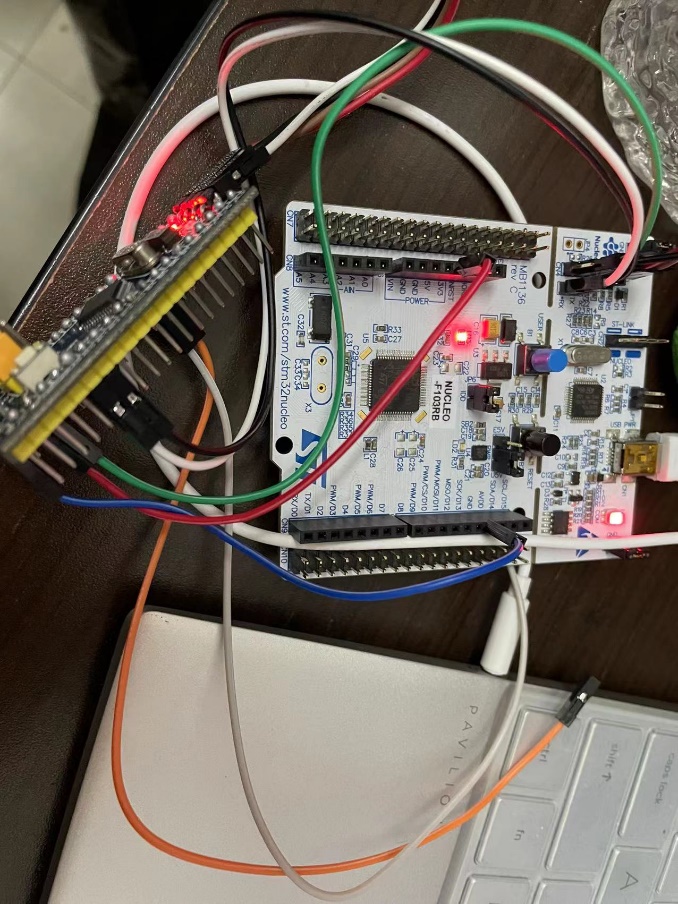
HAL\_Delay(1000);

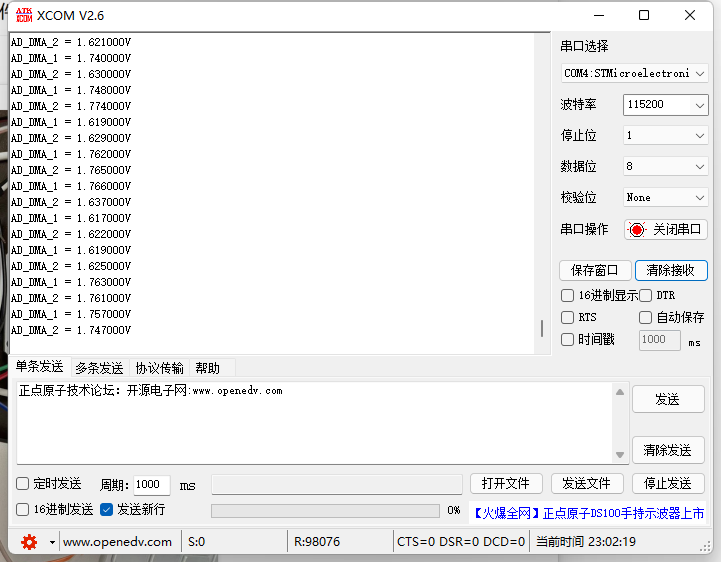
/\* USER CODE BEGIN 3 \*/

}

/\* USER CODE END 3 \*/

## Result





Voltage display on USAT