**How to configure and use the PSSI on the STM32U5**

**Introduction :**

The PSSI stands for Parallel synchronous slave interface and was introduced in our high performance STM32U5 family. The PSSI can be used as a fast interface between two STM32 microcontrollers or between an STM32 and an MPU or FPGA. This article will show how to configure and use this interface on a STM32U575.

**Prerequisites :**

* **Hardware :** 
  + Micro USB cable used to power the Nucleo board from a host machine and to load the code into the STM32
  + 2x Nucleo-U575ZI-Q



* + connection cables
* **Software :** 
  + STM32CubeMX
  + STM32CubeIDE

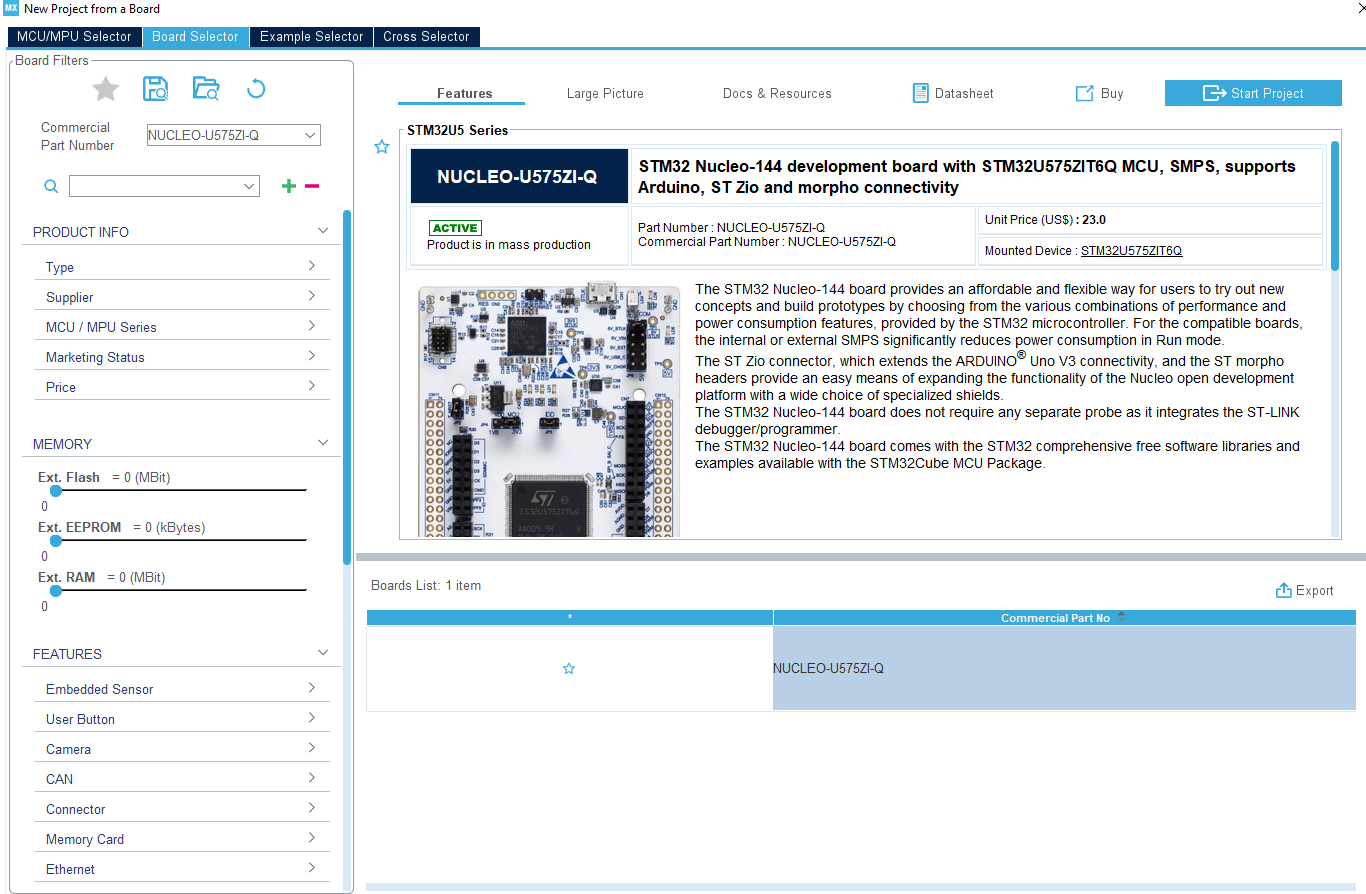
**Steps :**

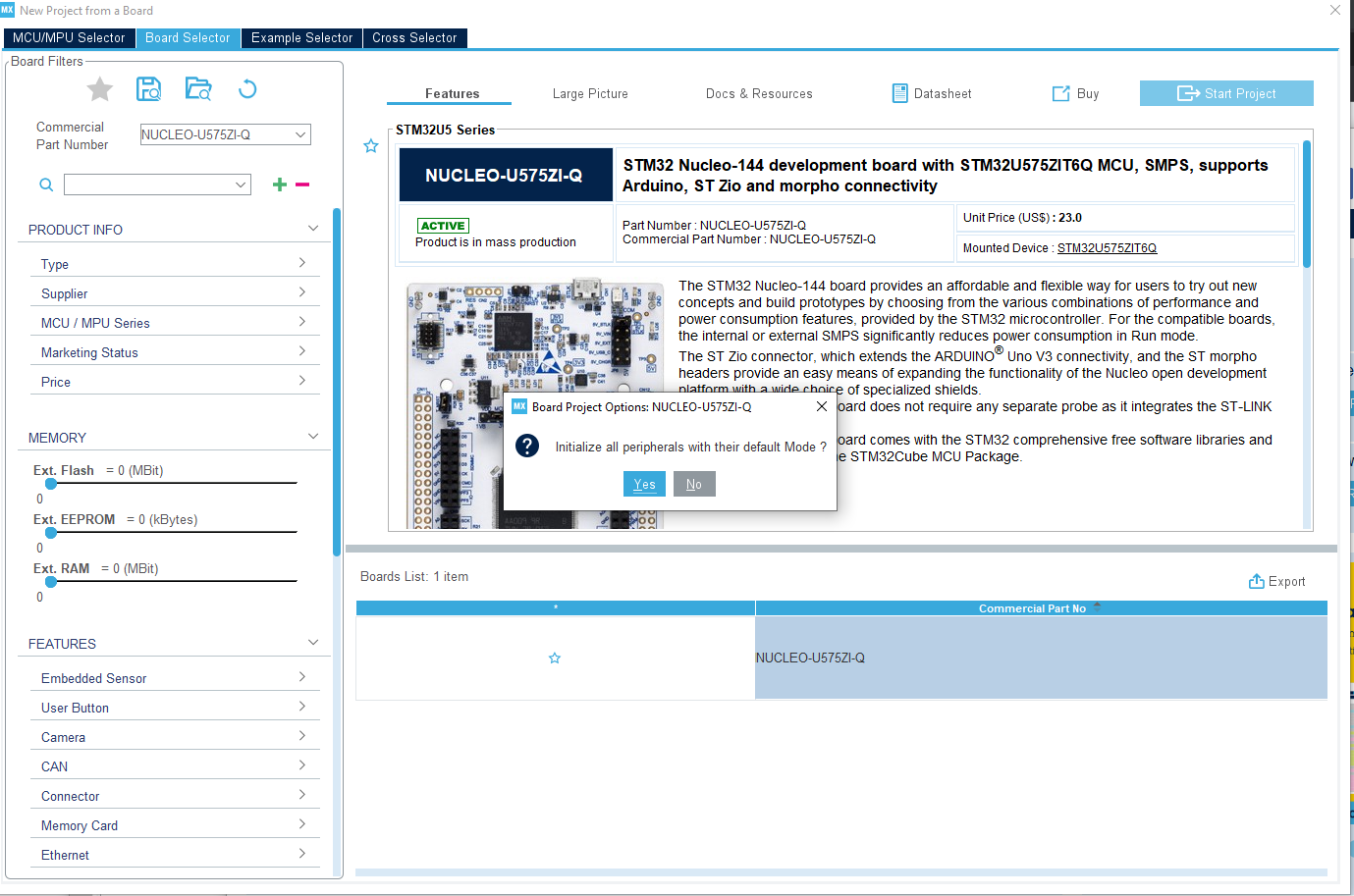
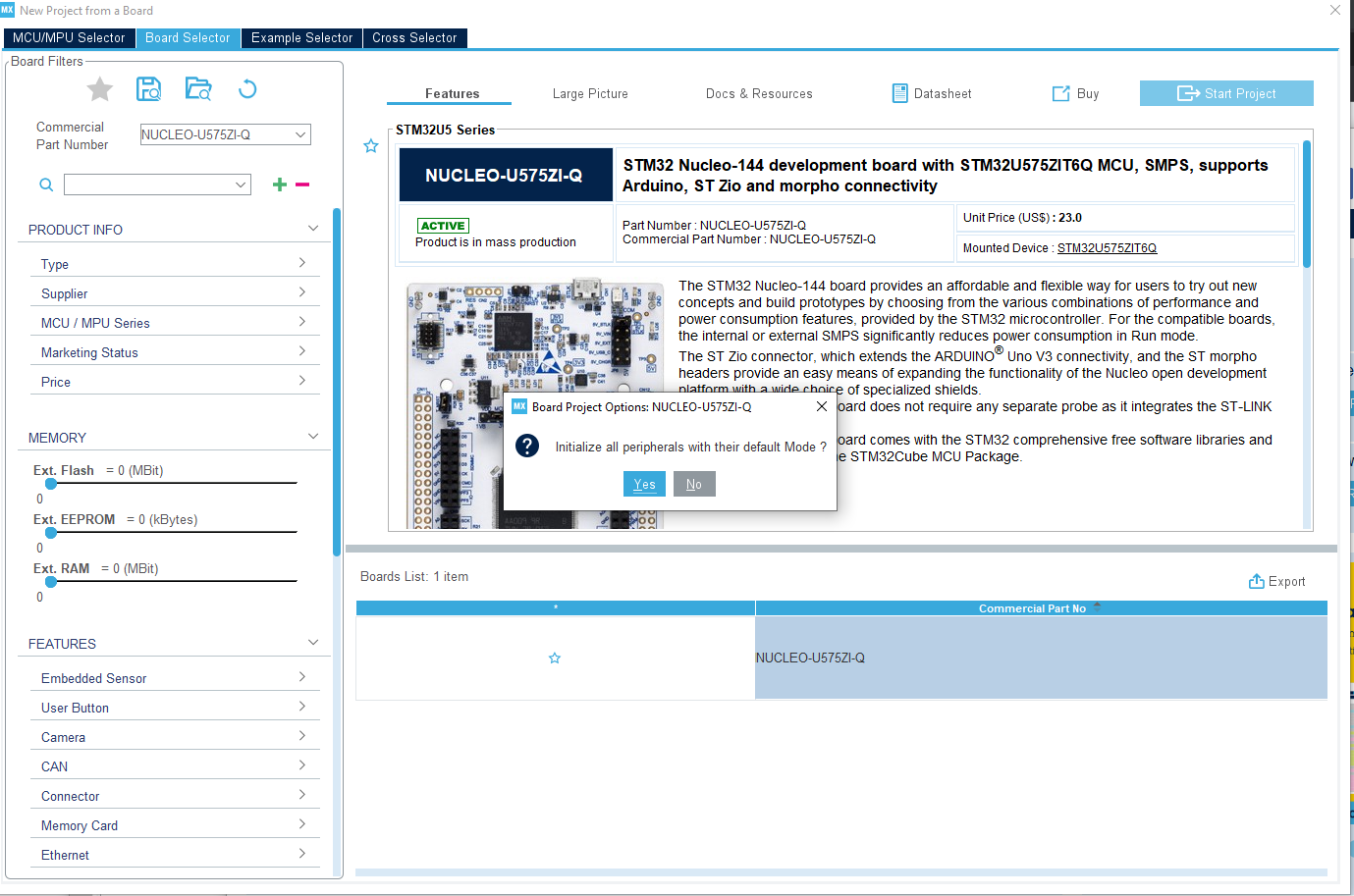
1. Open STM32CubeMX and select “ ACCESS TO BOARD SELECTOR“

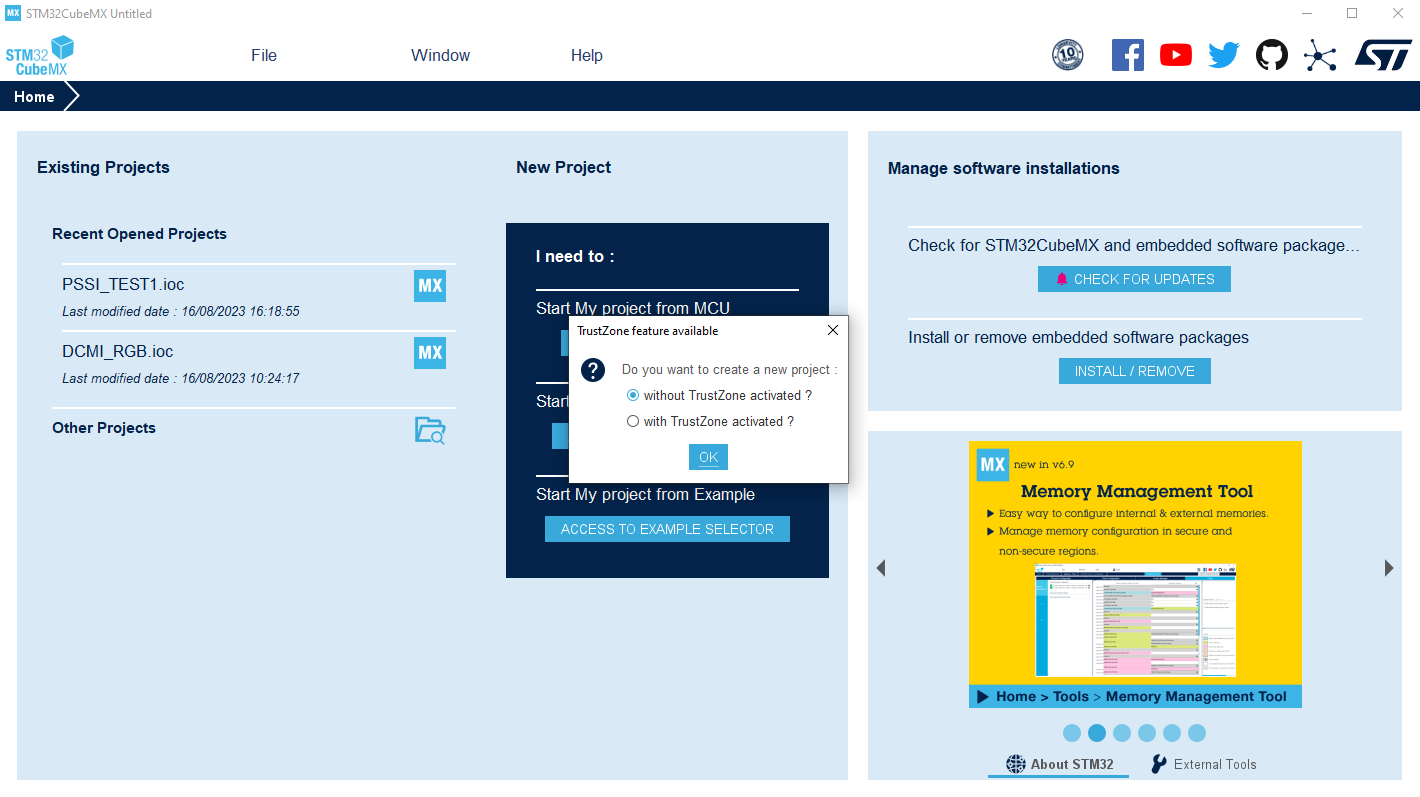
A screenshot of a computer

Description automatically generated

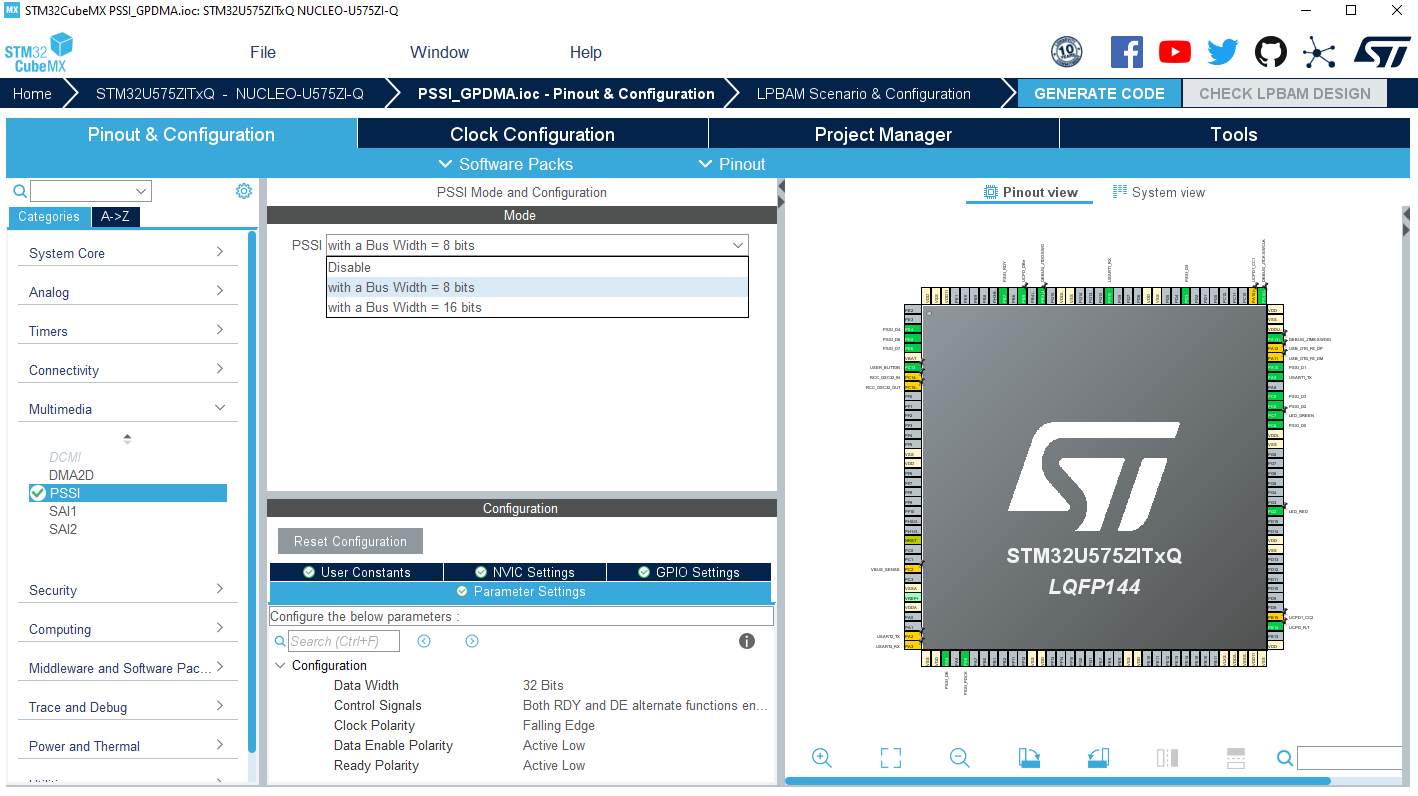
1. Select the correct board that you are working on. For this example, we are working with the “NUCLEO-U575ZI-Q”



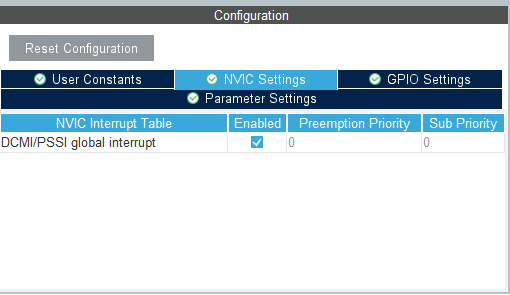
1. Start the project without initializing any peripherals.  
     
   

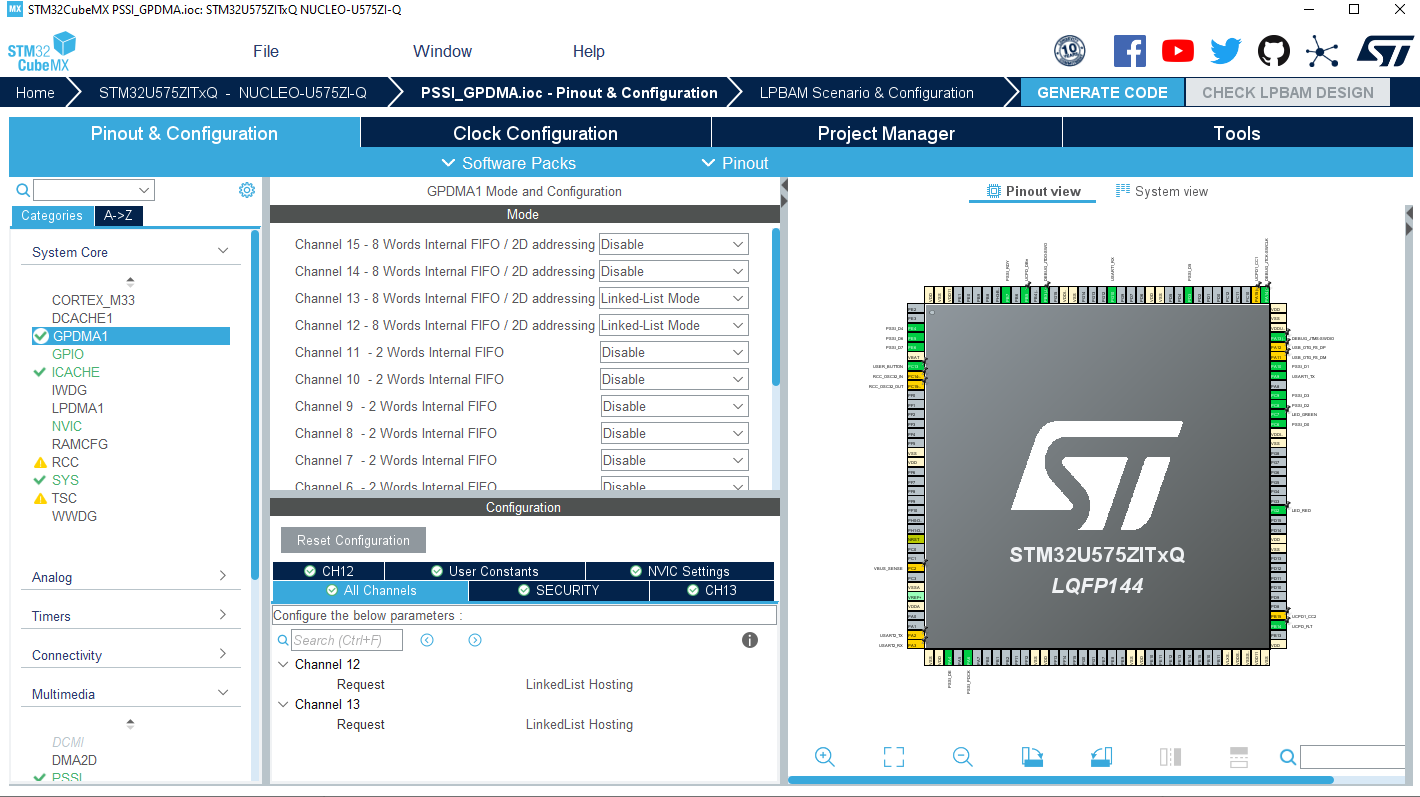

1. Go to multimedia and select “PSSI” and configure it to the following :

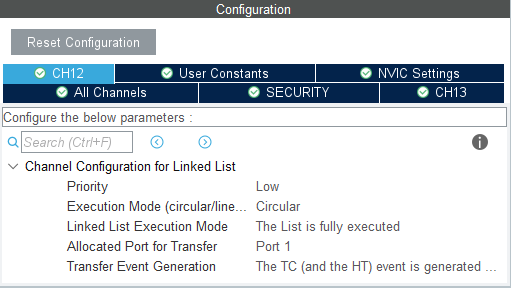


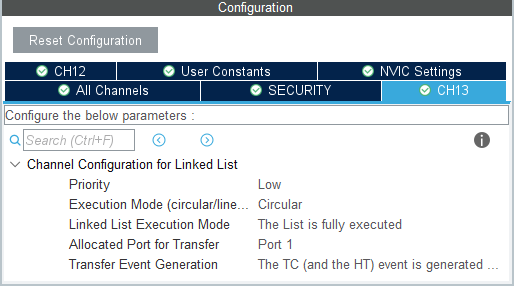
1. enable the NVIC Interrupt of the PSSI from the NVIC Settings .

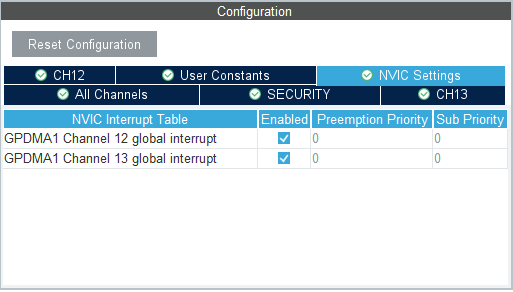


1. Go to System Core and configure two GPDMA channels one for transmission and the second for recieving as follows(In this exemple we went with channel 12 and 13 note that if 2D addressing are available they must be used as refernced in the application note):

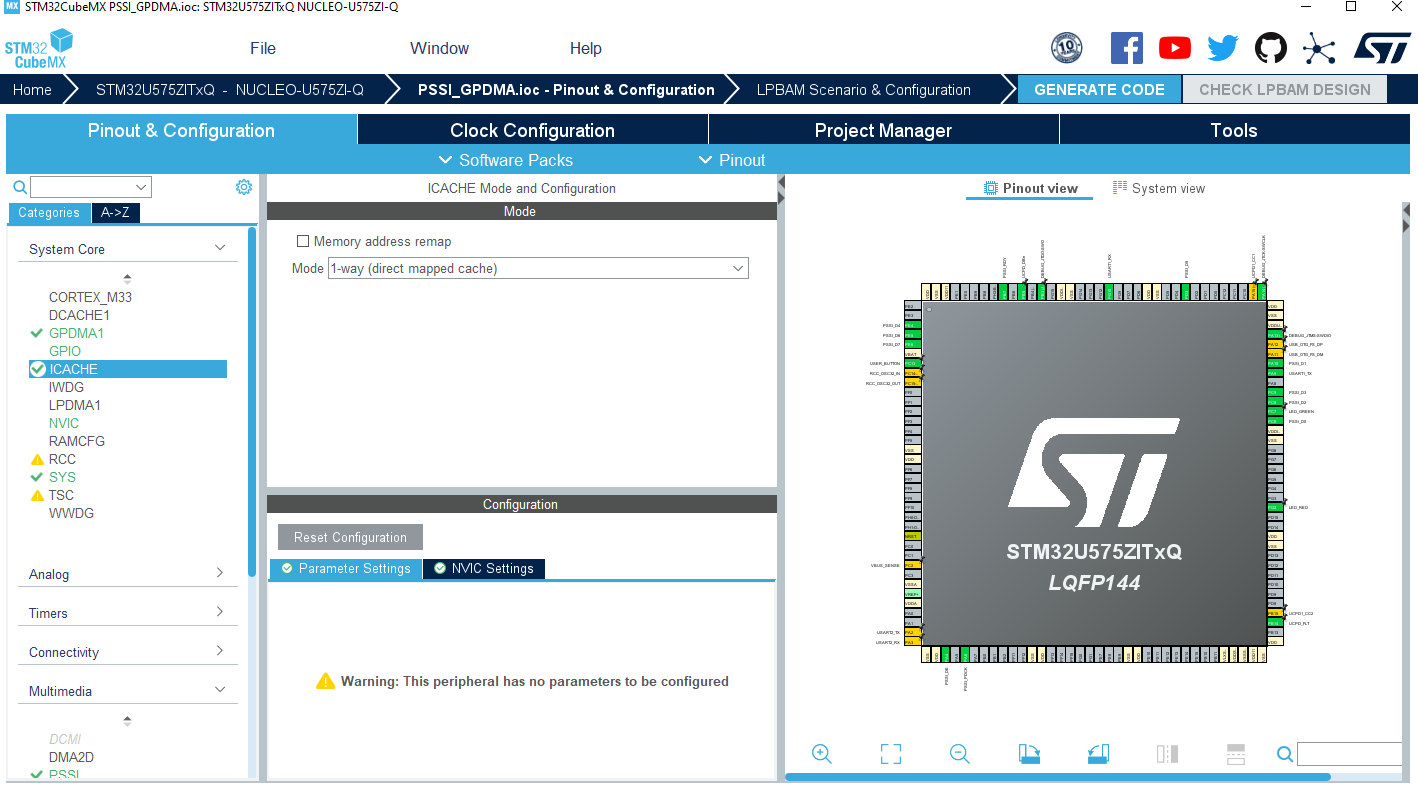




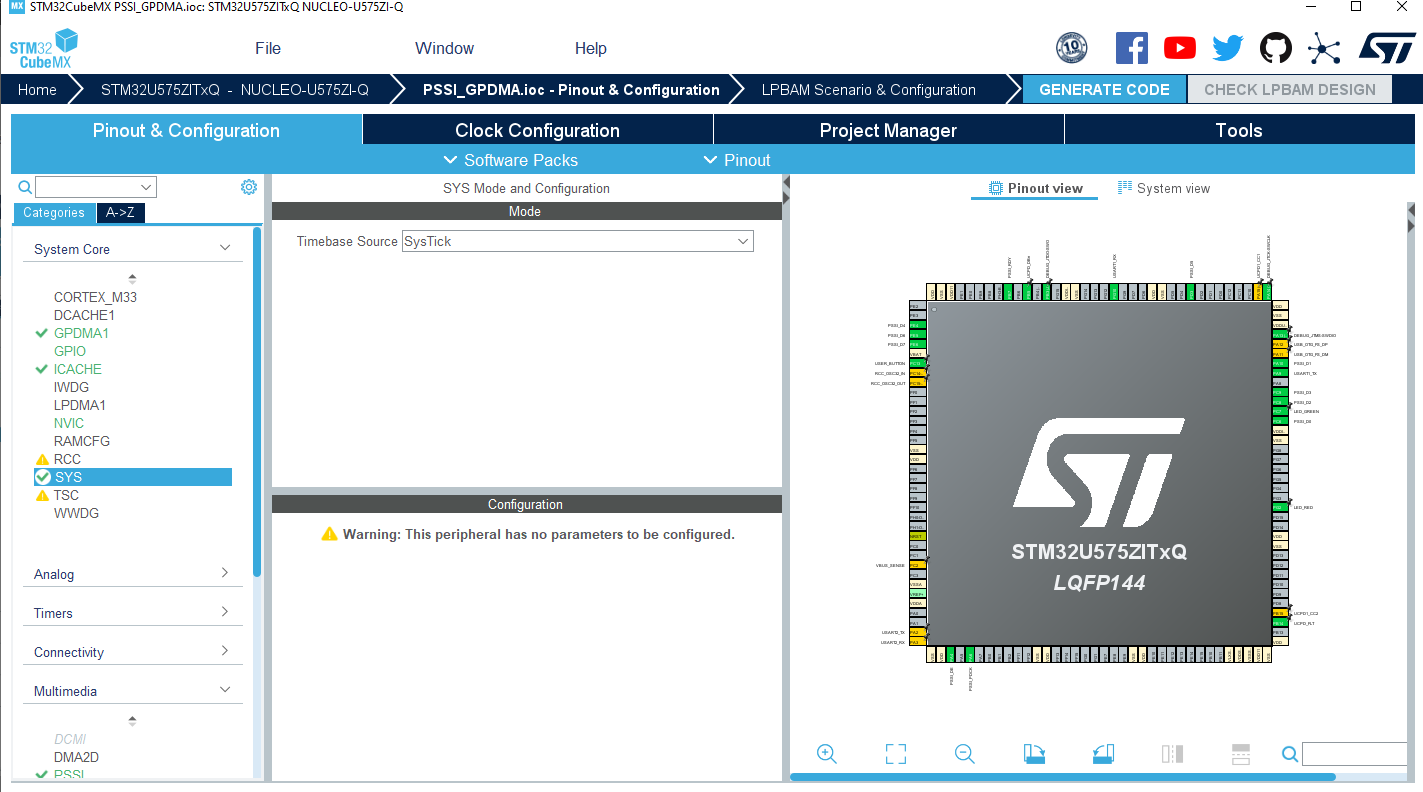




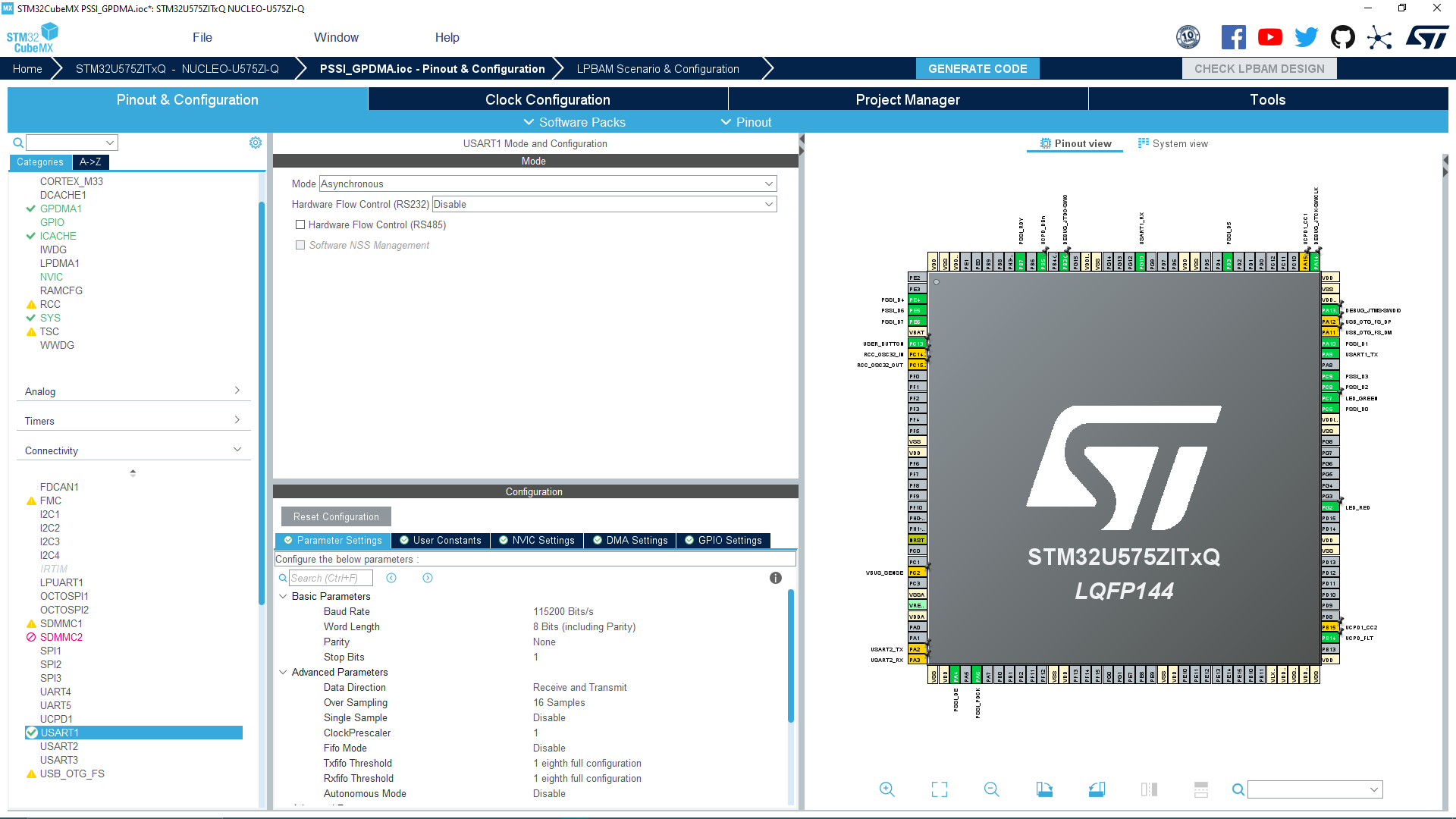
1. Enable the ICACHE for a performance boost :



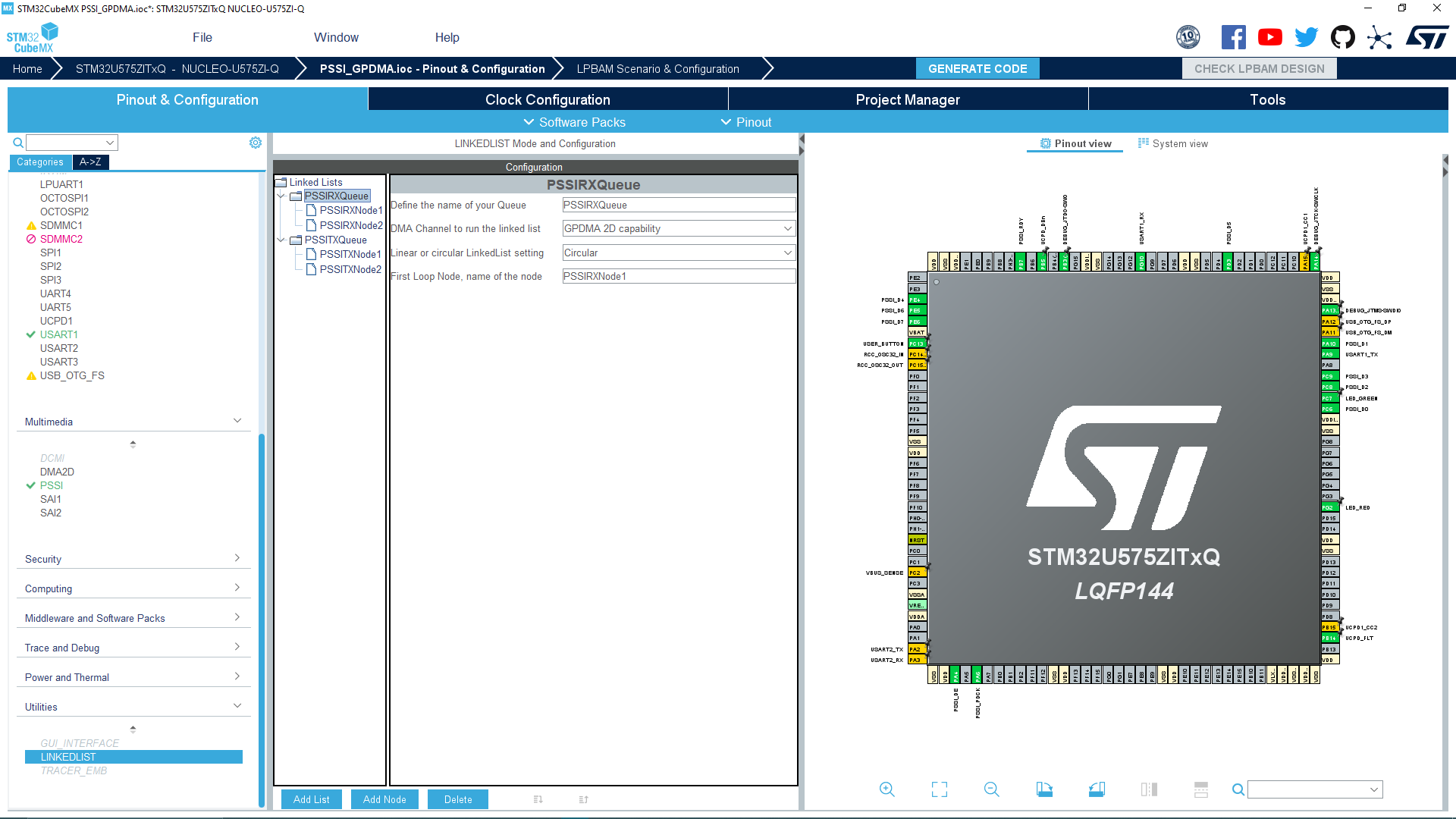
1. Change the time base source to SysTick to boost the performance of the AHB2 :

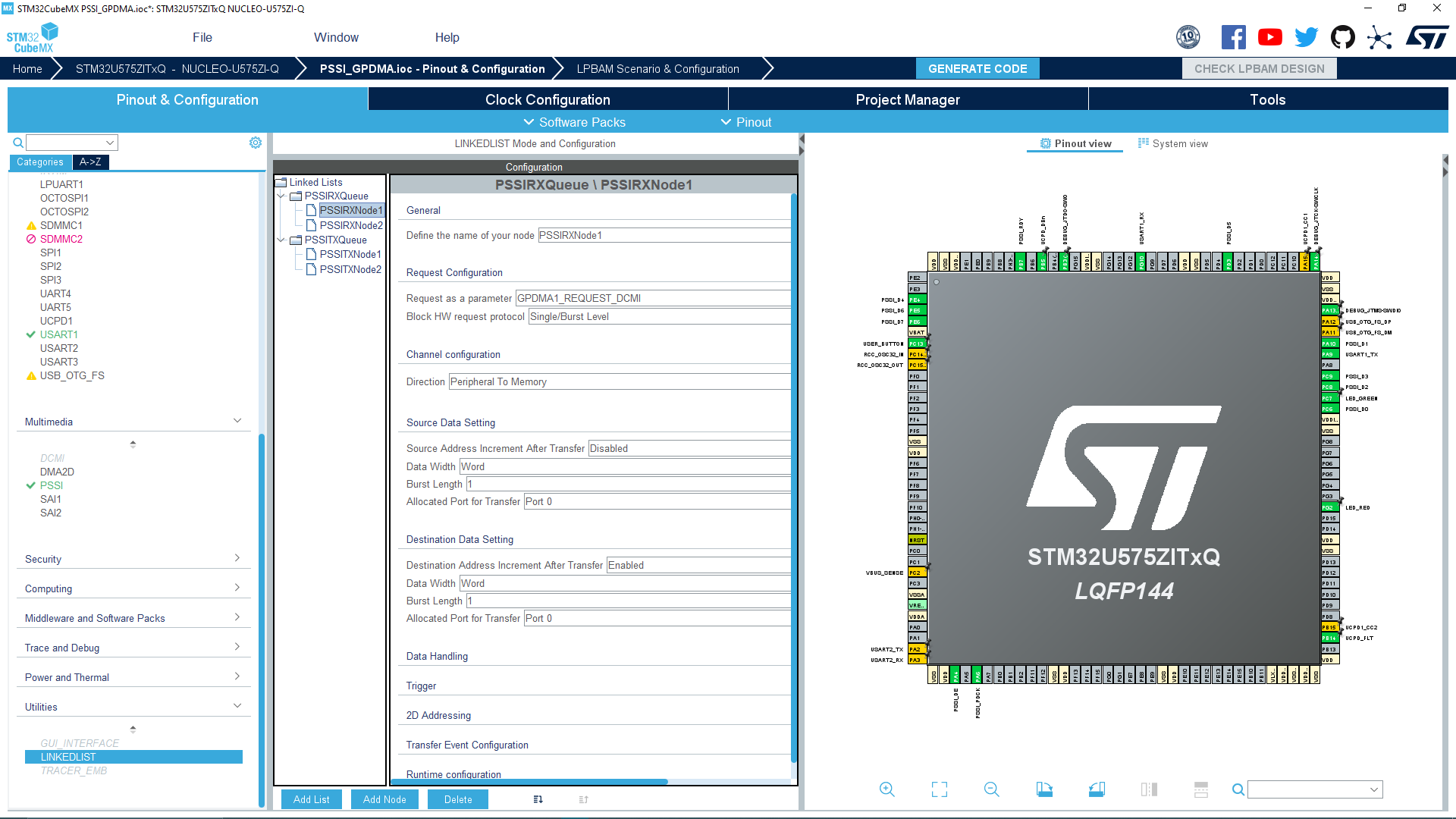


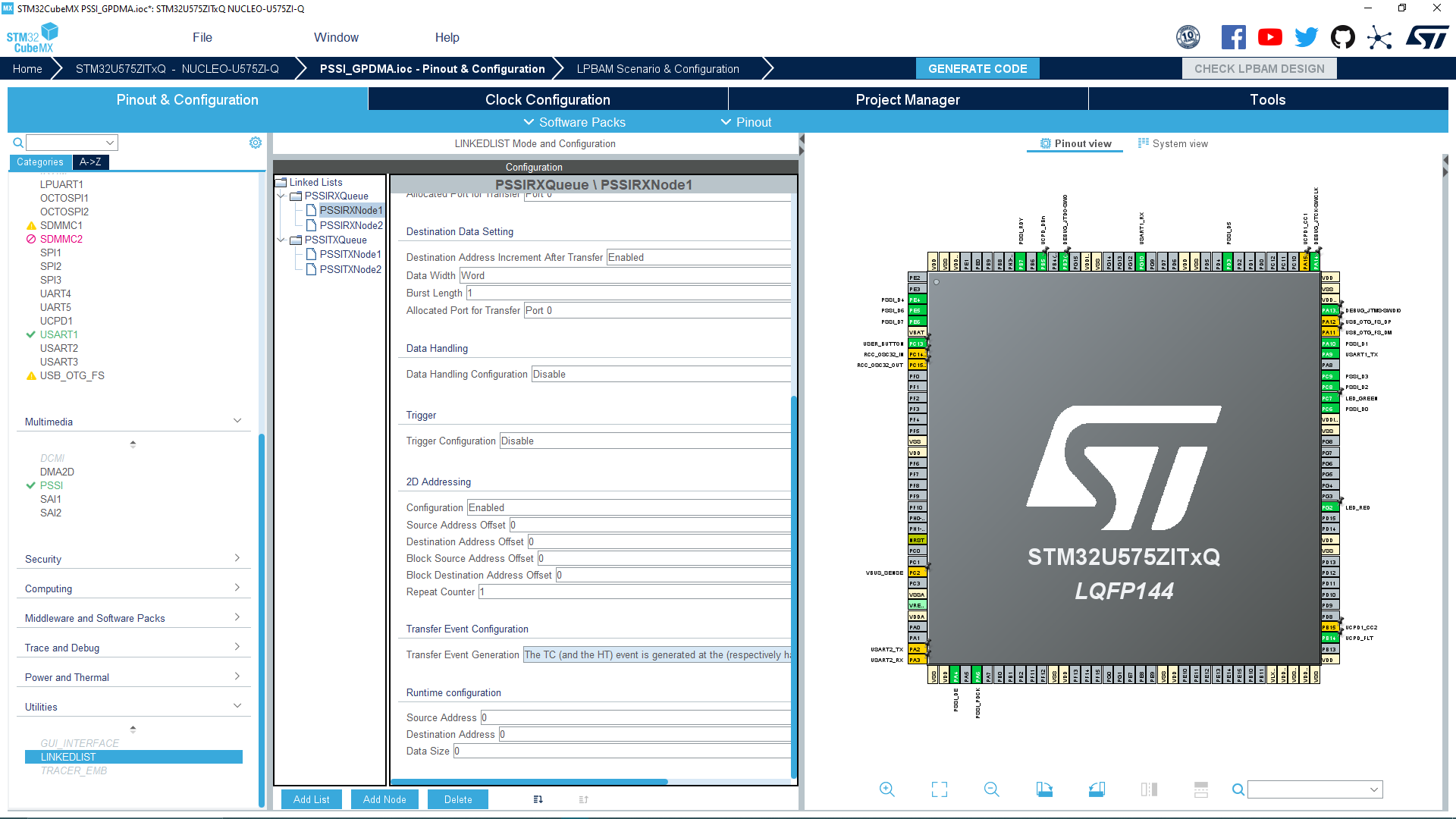
1. Enable the USART1 to display the transferred messages using the serial link via ST-Link with the following configuration:

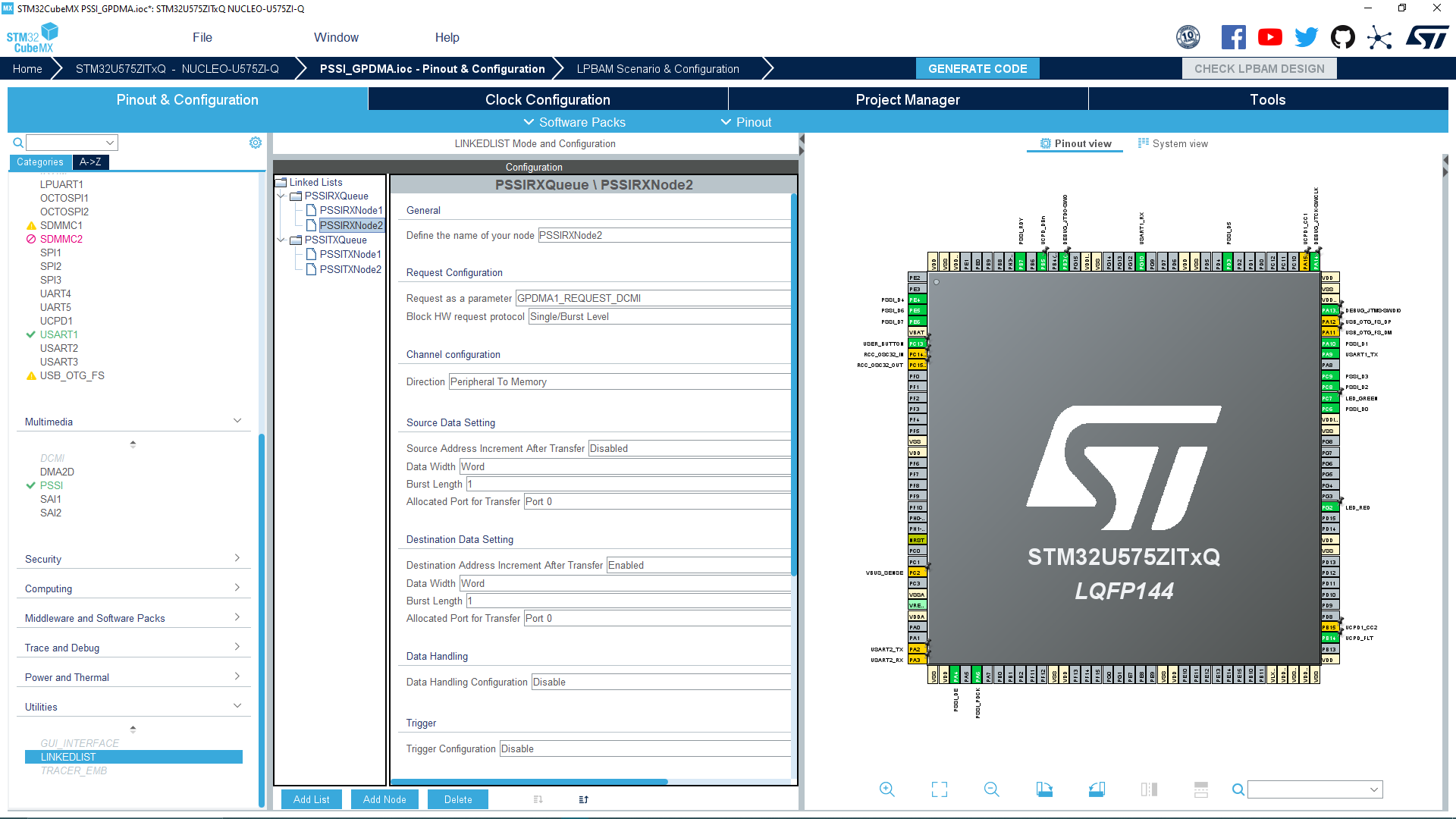


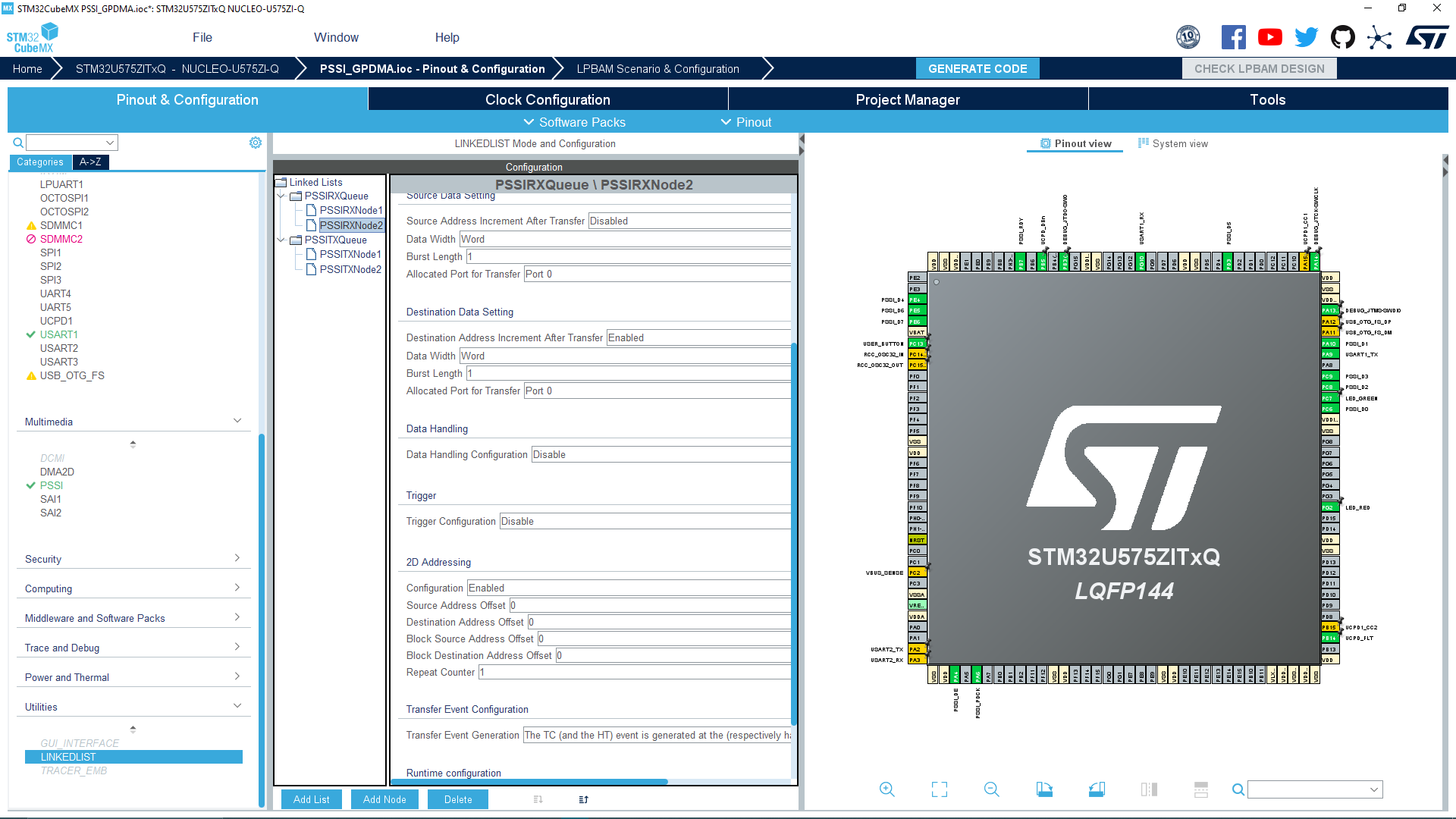
1. To ensure the transfer and receiving of data using PSSI you need to create two different linked lists with the following configurations each linked list will be linked to a channel of the GPDMA:

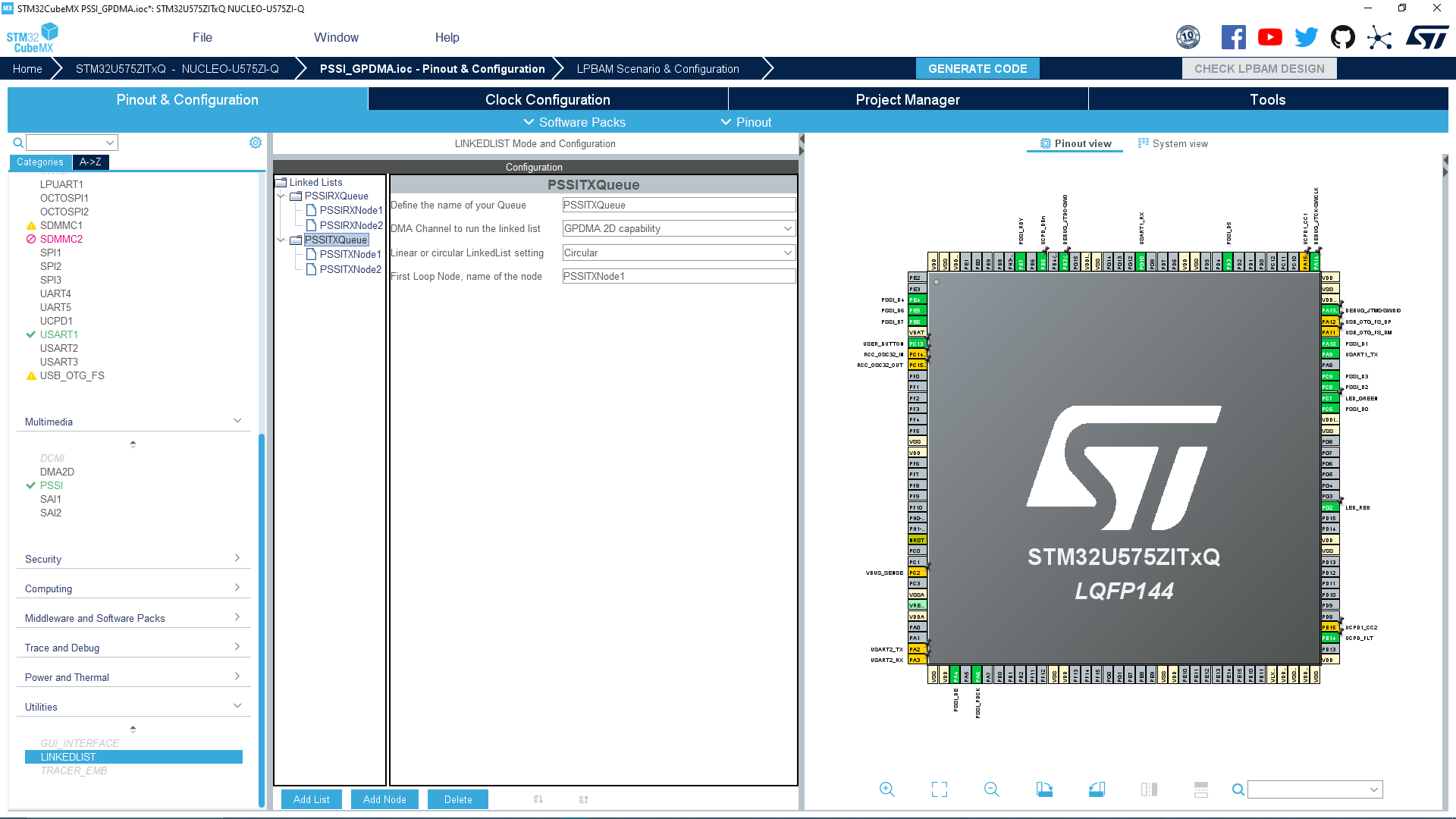


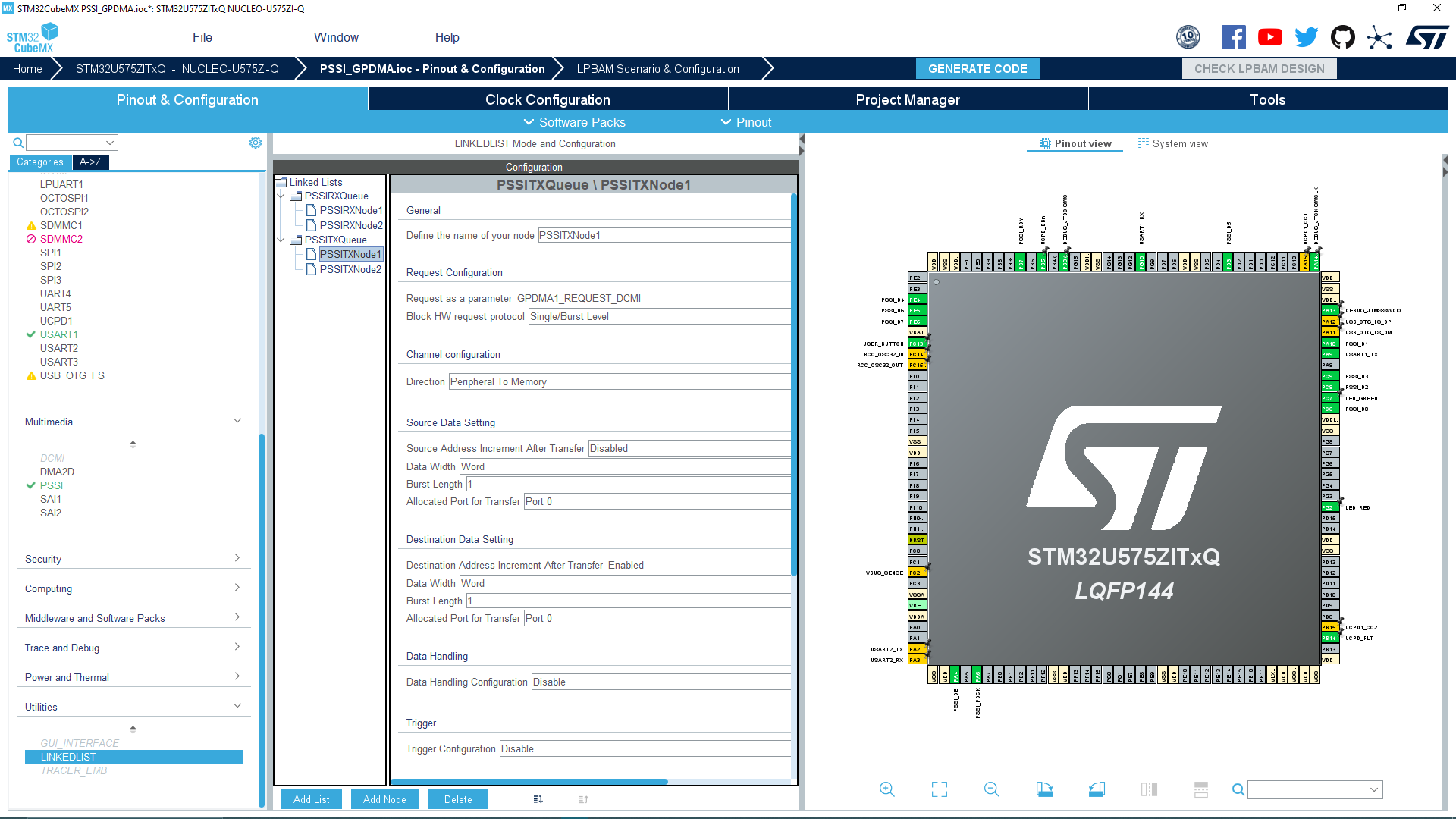


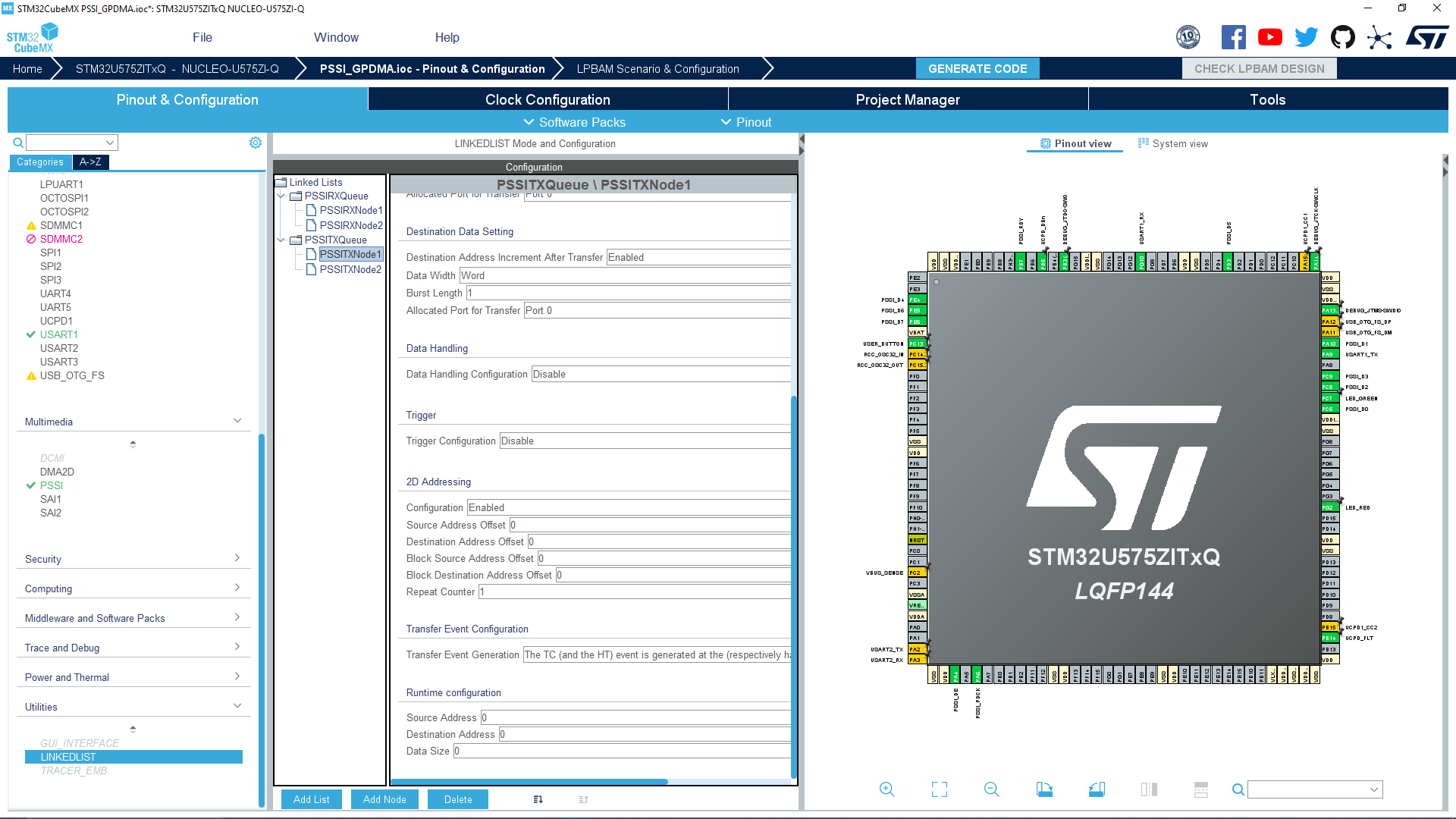


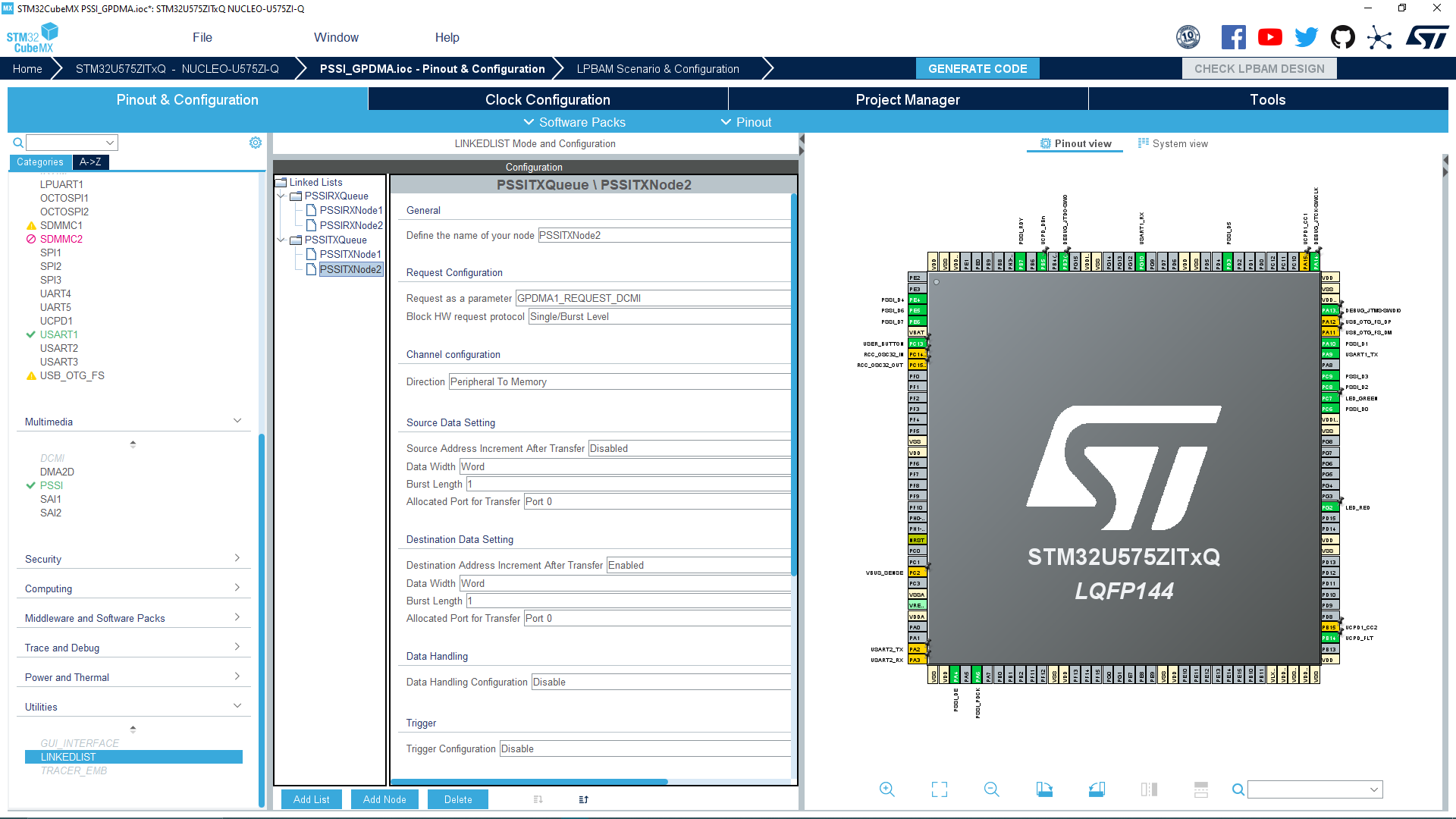


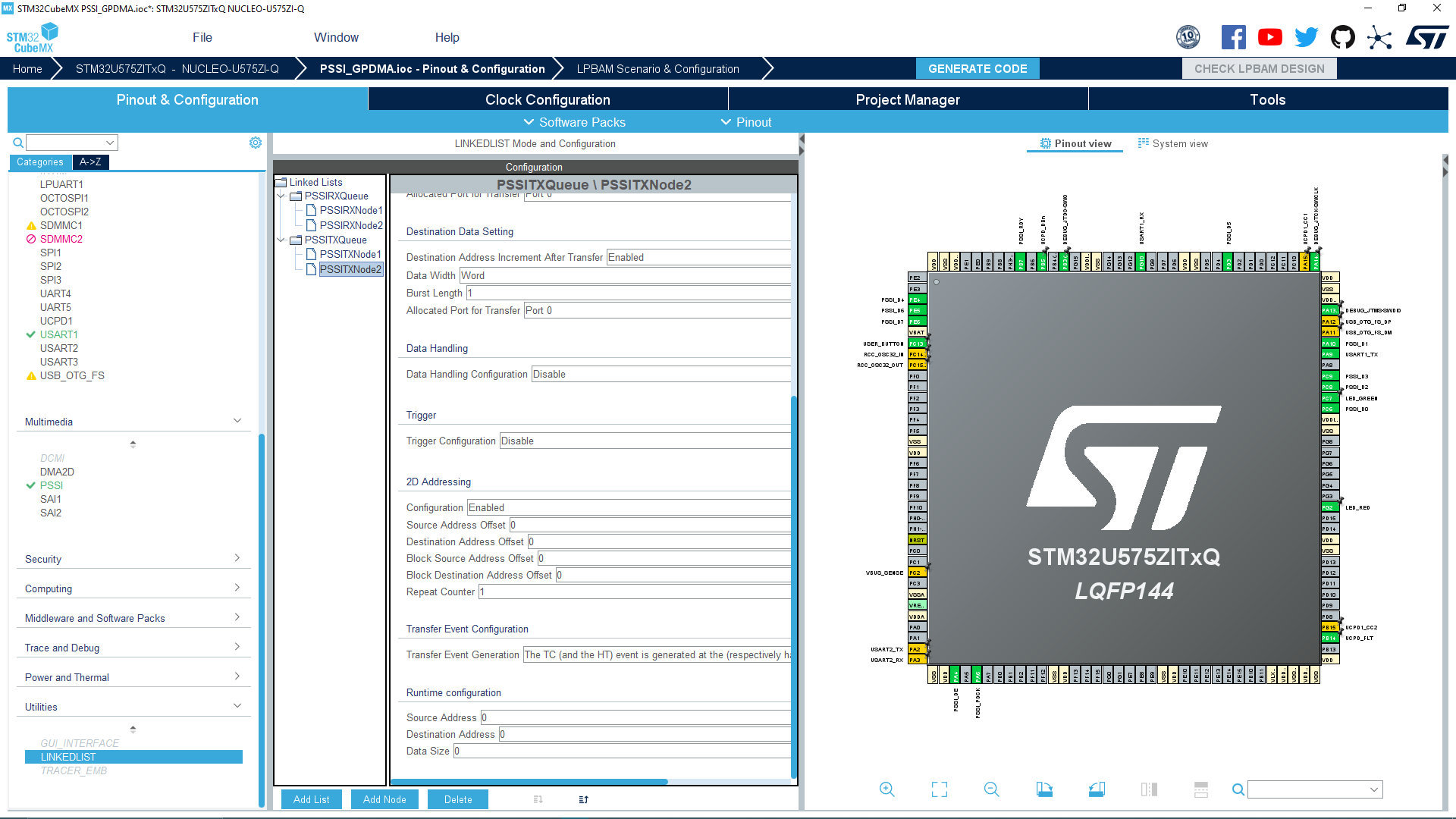




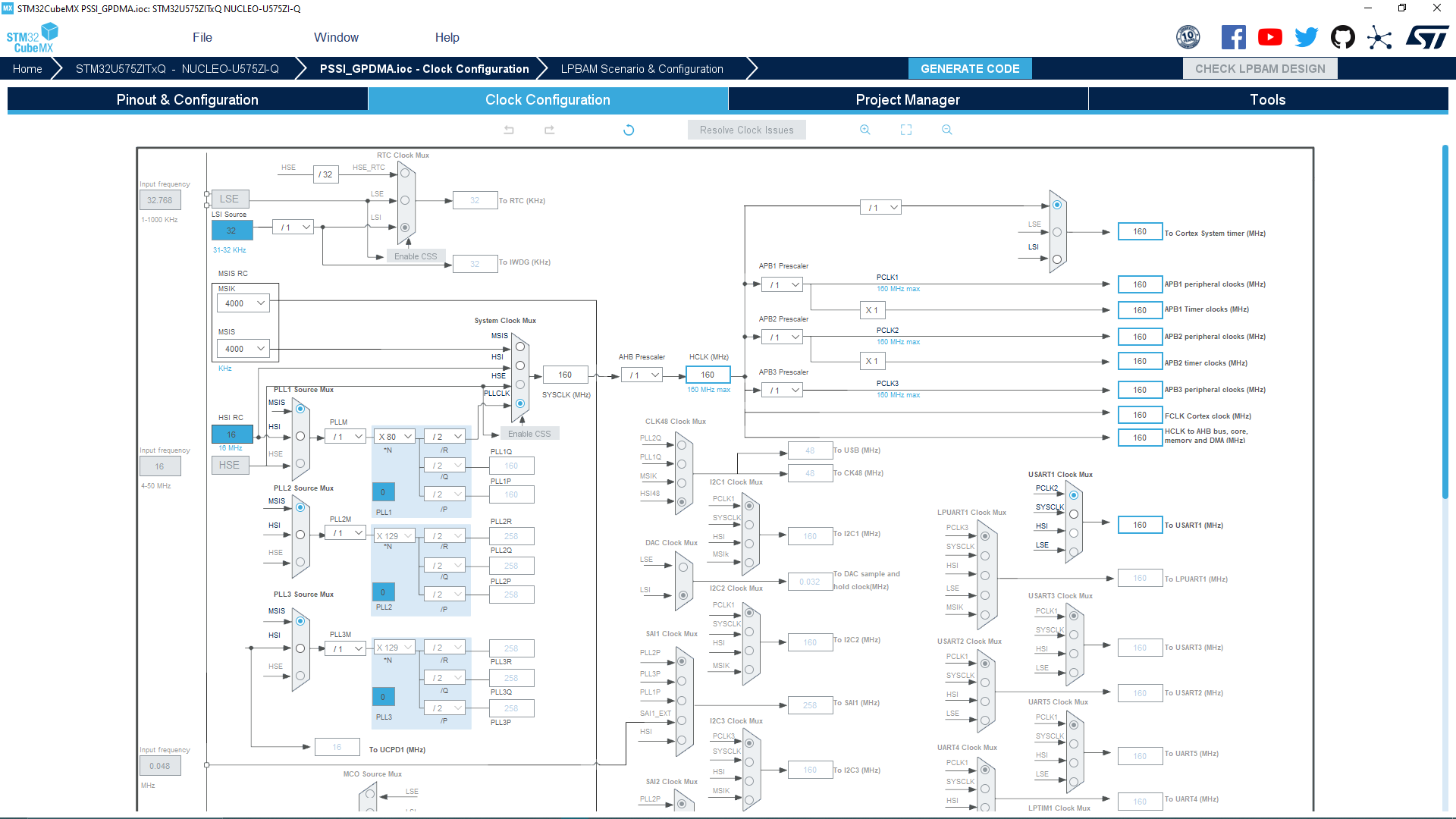








1. Edit the clock configuration to the following to obtain the maximum performance out of the GPDMA and the PSSI :

****

1. Clear all the unused pins and generate the project we used for this use case the STM32CubeIDE as a preferred toolchain :

**Code :**

1. In file “Core/Inc/main.h” add the following lines in the Private includes section:  
  
/\* USER CODE BEGIN Private defines \*/

/\* Definition for TIMx clock resources \*/

**#define** **TIMx** TIM1

**#define** **TIMx\_CLK\_ENABLE**() \_\_HAL\_RCC\_TIM1\_CLK\_ENABLE()

/\* USER CODE END Private defines \*/

2. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN Includes ” section:

/\* USER CODE BEGIN Includes \*/

**#include** "stdio.h"

**#include** <stdlib.h>

**#include** "string.h"

/\* USER CODE END Includes \*/

3. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN PTD” section(note that commenting or uncommenting the variable MASTER\_BOARD in this section determine if this is the code of the master or the slave):

/\* USER CODE BEGIN PTD \*/

**TIM\_HandleTypeDef** htim;

**TIM\_OC\_InitTypeDef** sConfig; /\* Timer Output Compare Configuration Structure declaration \*/

//#define MASTER\_BOARD

**#define** **TIMx** TIM1

**#define** **PERIOD\_VALUE** (uint32\_t)(1000 - 1) /\* Period Value \*/

**#define** **PULSE1\_VALUE** (uint32\_t)(PERIOD\_VALUE/2) /\* Capture Compare 1 Value \*/

/\* USER CODE END PTD \*/

4. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN PM” section:

/\* USER CODE BEGIN PM \*/

**uint32\_t** uhPrescalerValue = 0;

**uint32\_t** Request\_received = 0 ;

**uint32\_t** data\_cmp = 0 ;

**uint32\_t** i;

**uint32\_t** PSSI\_HAL\_PSSI\_TransmitComplete\_count = 0;

**uint32\_t** PSSI\_HAL\_PSSI\_ReceiveComplete\_count = 0;

**uint32\_t** PSSI\_HAL\_PSSI\_ErrorCallback\_count = 0;

**#ifndef** MASTER\_BOARD

//ALIGN\_32BYTES (char pData8\_S\_TRSMT[64] ="Hello from Slave"); /\* Data to transmit from Slave \*/

**#define** **ALIGN\_32BYTES** **\_\_attribute\_\_**((aligned(32)))

**char** pData8\_S\_TRSMT[64] ALIGN\_32BYTES = "Hello from Slave";

ALIGN\_32BYTES **char** pData8\_S\_RCV[64];

**#else**

ALIGN\_32BYTES (**char** pData8\_M\_RCV[64]);

ALIGN\_32BYTES (**char** pData8\_M\_TRSMT[64]); /\* Data to transmit from Master \*/

**#endif**

/\* USER CODE END PM \*/

5. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN PV” section:

**extern** **DMA\_QListTypeDef** PSSITXQueue;

**extern** **DMA\_QListTypeDef** PSSIRXQueue;

6. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN PFP” section:

/\* USER CODE BEGIN PFP \*/

**#ifdef** MASTER\_BOARD

**static** **void** Timer\_Config(TIM\_HandleTypeDef \*htim);

**static** **void** CLK\_On(**void**);

**static** **void** CLK\_Off(**void**);

**static** uint32\_t Fetch\_Slave\_Request(**char** Received\_Buffer[64]);

**#endif** /\*MASTER\_BOARD\*/

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

/\* USER CODE END PFP \*/

7. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN 0” section:

/\* USER CODE BEGIN 0 \*/

**int** **\_write**(**int** file, **char** \*ptr, **int** len) {

**HAL\_UART\_Transmit**(&huart1, (**uint8\_t**\*) ptr, len, HAL\_MAX\_DELAY);

**return** len;

}

/\* USER CODE END 0 \*/

8. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN 2” section:

/\* USER CODE BEGIN 2 \*/

**HAL\_DMAEx\_List\_LinkQ**(&handle\_GPDMA1\_Channel13, &PSSITXQueue);

\_\_HAL\_LINKDMA(&hpssi, hdmatx, handle\_GPDMA1\_Channel13);

**HAL\_DMAEx\_List\_LinkQ**(&handle\_GPDMA1\_Channel12, &PSSIRXQueue);

\_\_HAL\_LINKDMA(&hpssi, hdmarx, handle\_GPDMA1\_Channel12);

**#ifdef** MASTER\_BOARD

printf("Hello this is the master board \n\r");

HAL\_Delay(500);

/\* Configure Tim to generate the clock for PSSI slave \*/

Timer\_Config(&htim) ;

**while**(1)

{

CLK\_On();

**while**(Request\_received != 1)

{

**if**(HAL\_PSSI\_Receive\_DMA(&hpssi, (uint32\_t\*)pData8\_M\_RCV , **sizeof**(pData8\_M\_RCV)/4)!= HAL\_OK)

{

Error\_Handler();

}

**while**(PSSI\_HAL\_PSSI\_ReceiveComplete\_count !=1)

{

/\* wait until receiving data is complete \*/

}

**if**(Fetch\_Slave\_Request(pData8\_M\_RCV) == 1)

{

Request\_received = 1;

}

PSSI\_HAL\_PSSI\_ReceiveComplete\_count = 0 ;

}

/\* Start transmitting the received & modified data \*/

**if**(HAL\_PSSI\_Transmit\_DMA(&hpssi, (uint32\_t\*)pData8\_M\_TRSMT, **sizeof**(pData8\_M\_TRSMT)/4)!= HAL\_OK)

{

Error\_Handler();

}

HAL\_Delay(2);

**while**(PSSI\_HAL\_PSSI\_TransmitComplete\_count != 1)

{

/\* wait until transmit data is complete \*/

}

printf("Transmission completed\n\r");

HAL\_Delay(200);

HAL\_Delay(20);

/\* Transmit done \*/

Request\_received = 0 ;

PSSI\_HAL\_PSSI\_TransmitComplete\_count = 0 ;

CLK\_Off();

/\*Wait 1s before the next loop\*/

HAL\_Delay(1000);

}

**#else**

/\*PSSI slave Part\*/

**printf**("Hello this is the slave board \n\r");

**HAL\_Delay**(500);

/\* Start transmitting the data \*/

**if**(**HAL\_PSSI\_Transmit\_DMA**(&hpssi, (**uint32\_t**\*)(pData8\_S\_TRSMT), **sizeof**(pData8\_S\_TRSMT))!= *HAL\_OK*)

{

**Error\_Handler**();

}

**while**(PSSI\_HAL\_PSSI\_TransmitComplete\_count != 1)

{

/\* wait until transmit data is complete \*/

}

**HAL\_Delay**(2);

/\* Start receiving the data \*/

**if**(**HAL\_PSSI\_Receive\_DMA**(&hpssi, (**uint32\_t**\*)(pData8\_S\_RCV), **sizeof**(pData8\_S\_RCV)/4)!= *HAL\_OK*)

{

**Error\_Handler**();

}

**while**(PSSI\_HAL\_PSSI\_ReceiveComplete\_count != 1)

{

/\* wait until receiving data is complete \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Check first that the transmitted data is received \*/

**for**(i = 0; i < **strlen**(pData8\_S\_TRSMT) ;i++)

{

**if**(pData8\_S\_TRSMT[i] != pData8\_S\_RCV[i])

{

data\_cmp++;

**break**;

}

}

/\*if OK , check then, that the Master has modified the transmitted data\*/

**if**((data\_cmp == 0) && (**strstr**(pData8\_S\_RCV," Master") != 0))

{ /\* Infinite loop \*/

**while** (1)

{

**printf**("data has been modified\n\r");

**printf**("%s\n\r",pData8\_S\_RCV);

**HAL\_Delay**(1000);

}

}

**else** /\*KO\*/

{

**Error\_Handler**();

}

**#endif** /\*MASTER\_BOARD\*/

/\* USER CODE END 2 \*/

8. In file “Core/Src/main.c” add the following lines in the “USER CODE BEGIN 4” section:

**#ifdef** MASTER\_BOARD

**void** Timer\_Config(TIM\_HandleTypeDef \*htim)

{

/\* Compute the prescaler value to have TIM1 counter clock equal to 20000 Hz \*/

uhPrescalerValue = (uint32\_t)(250000000 / (20000000)) - 1;

htim->Instance = TIMx;

htim->Init.Prescaler = uhPrescalerValue;

htim->Init.Period = PERIOD\_VALUE;

htim->Init.ClockDivision = 0;

htim->Init.CounterMode = TIM\_COUNTERMODE\_UP;

htim->Init.RepetitionCounter = 0;

**if** (HAL\_TIM\_PWM\_Init(htim) != HAL\_OK)

{

/\* Initialization Error \*/

Error\_Handler();

}

/\*##-2- Configure the PWM channel #########################################\*/

/\* configuration for the channel \*/

sConfig.OCMode = TIM\_OCMODE\_PWM1;

sConfig.OCPolarity = TIM\_OCPOLARITY\_HIGH;

sConfig.OCFastMode = TIM\_OCFAST\_DISABLE;

sConfig.OCNPolarity = TIM\_OCNPOLARITY\_HIGH;

sConfig.OCNIdleState = TIM\_OCNIDLESTATE\_RESET;

sConfig.OCIdleState = TIM\_OCIDLESTATE\_RESET;

/\* Set the pulse value for channel 1 \*/

sConfig.Pulse = PULSE1\_VALUE;

**if** (HAL\_TIM\_PWM\_ConfigChannel(htim, &sConfig, TIM\_CHANNEL\_1) != HAL\_OK)

{

/\* Configuration Error \*/

Error\_Handler();

}

}

**static** **void** CLK\_On(**void**)

{

/\*Start PWM signals generation #######################################\*/

/\* Start channel 1 \*/

**if** (HAL\_TIM\_PWM\_Start(&htim, TIM\_CHANNEL\_1) != HAL\_OK)

{

/\* PWM Generation Error \*/

Error\_Handler();

}

}

**static** **void** CLK\_Off(**void**)

{

**if** (HAL\_TIM\_PWM\_Stop(&htim, TIM\_CHANNEL\_1) != HAL\_OK)

{

/\* PWM Generation Error \*/

Error\_Handler();

}

}

uint32\_t Fetch\_Slave\_Request(**char** Received\_Buffer[64])

{

uint32\_t j;

**char** REPLY[23] = ": received, from Master" ;

**if**(strstr(Received\_Buffer,"Slave") != 0)

{

**for**(i=0 ; i < strlen(Received\_Buffer) ; i++)

{

pData8\_M\_TRSMT[i] = Received\_Buffer[i];

}

**for**(j=0;j<strlen(REPLY);j++)

{

pData8\_M\_TRSMT[i] = REPLY[j];

i++;

}

**return** 1;

}

**else**

{

**return** 0;

}

}

**#endif** /\* MASTER\_BOARD \*/

**void** **HAL\_PSSI\_TxCpltCallback**(**PSSI\_HandleTypeDef** \*hpssi)

{

PSSI\_HAL\_PSSI\_TransmitComplete\_count++;

}

/\*\*

\* @brief PSSI receive complete callback

\* @par hppsi: PSSI HAL handle

\* @retval None

\*/

**void** **HAL\_PSSI\_RxCpltCallback**(**PSSI\_HandleTypeDef** \*hpssi)

{

PSSI\_HAL\_PSSI\_ReceiveComplete\_count++;

}

/\*\*

\* @brief PSSI transfer error callback

\* @par hppsi: PSSI HAL handle

\* @retval None

\*/

**void** **HAL\_PSSI\_ErrorCallback**(**PSSI\_HandleTypeDef** \*hpssi)

{

PSSI\_HAL\_PSSI\_ErrorCallback\_count++;

}  
  
  
  
  
  
9. In file “Core/Inc/stm32u5xx\_hal\_conf.h” uncomment the following line :  
  
**#define** **HAL\_TIM\_MODULE\_ENABLED**

10. Enjoy the demo.