Lesson 6 - ADC and Embedded C

Norman McEntire
norman.mcentire@gmail.com

Contents

- Introduction to ADC and Embedded C
- STM32CubeMX and ADC Code Generation
- Tour of ADC
- TrueStudio and ADC
    - Read ADC value

https://www.st.com/content/ccc/resource/technical/document/user_manual/63/a8/8f/e3/ca/a1/4c/84/DM00173145.pdf/files/DM00173145.pdf/jcr:content/translations/en.DM00173145.pdf

life.augmented

UM1884
User manual

Description of STM32L4/L4+ HAL and low-layer drivers

Introduction

STMCube™ is STMicroelectronics's original initiative to ease developers' life by reducing development efforts, time and cost. STM32Cube covers the STM32 portfolio.

STM32Cube Version 1.x includes:

- The STM32CubeMX, a graphical software configuration tool that allows generating C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as STM32CubeL4 for STM32L4 series and STM32L4+ series)
  - The STM32Cube Hardware Abstraction Layer (HAL), an STM32 abstraction layer embedded software ensuring maximized portability across the STM32 portfolio. The HAL is available for all peripherals.
  - The low-layer APIs (LL) offering a fast light-weight expert-oriented layer which is closer to the hardware than the HAL. The LL APIs are available only for a set of peripherals.

Key Sections From Manual

Introduction to ADC and Embedded C

28.2.1 ADC Peripheral Features

- ADC = Analog to Digital Conversion
- **12-bit**, 10-bit, 8-bit, 6-bit resolution
- Interrupt at end of conversion
- Single or continuous conversion
- Programmable sample time
- External trigger (timer or EXTI)

28.2.2 How to use driver - Part 1

- Step 1. Enable ADC Interface
  - __HAL_RCC_GPIOx_CLK_ENABLE()
- Step 2. Configure ADC Pins
  - HAL_GPIO_Init()
- Step 3. (If using input for interrupts). Configure NVIC IRQ Priority mapped to the EXTI line
  - HAL_NVIC_SetPriority()
  - HAL_NVIC_EnableIRQ()

NVIC = Nested Vector Interrupt Controller

HAL ADC Data Types

Drivers/STM32L4xx_HAL_Driver/Inc/**stm32l4xx_adc.h**

ADC_OversamplingTypeDef

- typedef struct {
    uint32_t Ratio;
    uint32_t RightBitShift;
    uint32_t TriggeredMode;
    uint32_t OversamplingStopReset;
} ADC_OversamplingTypeDef

ADC_InitTypeDef

- typedef struct {
    uint32_t ClockPrescaler;
    uint32_t Resolution;
    uint32_t DataAlign;
    uint32_t ScanConvMode;
    uint32_t EOCSelection;
    FunctionalState LowPowerAutoWait;
    FunctionalState ContinuousConvMode;
    uint32_t NbrOfConversion;
    FunctionalState DiscontinuousConvMode;
    uint32_t NbrOfDiscConversion;
    uint32_t ExternalTrigConv;
    uint32_t ExternalTrigConvEdge;
    FunctionalState DMAContinuousRequests;
    uint32_t Overrun;
    FunctionalState OversamplingMode;
    ADC_OversamplingTypeDef Oversampling;
} ADC_OversamplingTypeDef

ADC_ChannelConfTypeDef

- typedef struct {
    uint32_t Channel;
    uint32_t Rank;
    uint32_t SamplingTime;
    uint32_t SingleDiff;
    uint32_t OffsetNumber;
    uint32_t Offset;
} ADC_ChannelConfTypeDef

ADC_AnalogWDGConfType Def

- ADC Analog Watchdog
- typedef struct {
    uint32_t WatchDogNumber;
    uint32_t WatchDogMode;
    uint32_t Channel;
    FunctionalState ITMode;
    uint32_t HighThreshold;
    uint32_t LowThreshold;
} ADC_AnalogWDGConfTypeDef

HAL ADC State Machine States of ADC **Global Scope**

- HAL_ADC_STATE_RESET
- HAL_ADC_STATE_READY
- HAL_ADC_STATE_BUSY_INTERNAL
- HAL_ADC_STATE_TIMEOUT
- Example
  - if (HAL_ADC_GET_STATE(hadc1) & HAL_ADC_STATE_READY != 0) { … }

HAL ADC State Machine States of ADC **Errors**

- HAL_ADC_STATE_ERROR_INTERNAL
- HAL_ADC_STATE_ERROR_CONFIG
- HAL_ADC_STATE_ERROR_DMA

HAL ADC State Machine

States of ADC **Group Regular**

- HAL_ADC_STATE_REG_BUSY
- HAL_ADC_STATE_REG_EOC
  - End of Conversion
- HAL_ADC_STATE_REG_OVR
  - Overrun
- HAL_ADC_STATE_REG_EOSMP
  - End of Sample

HAL ADC State Machine
States of ADC **Analog Watchdogs**

- These are watchdogs for out-of-window occurrence
- HAL_ADC_STATE_AWD1
- HAL_ADC_STATE_AWD2
- HAL_ADC_STATE_AWD3

ADC_HandleTypeDef

- typedef struct {
  ADC_TypeDef *instance; //Register base address
  ADC_InitTypeDef Init;
  DMA_HandleTypeDef *DMA_Handle;
  HAL_LockTypeDef Lock;
  __IO uint32_t State;
  __IO uint32_t ErrorCode;
  ADC_InjectionConfigTypeDef InjectionConfig;
#if (USE_HAL_ADC_REGISTER_CALLBACKS = 1)
…
#endif
} ADC_HandleTypeDef

ADC_RESOLUTION

- ADC_RESOLUTION_12B
- ADC_RESOLUTION_10B
- ADC_RESOLUTION_8B
- ADC_RESOLUTION_6B

ADC_DATAALIGN

- ADC_DATAALIGN_RIGHT
- ADC_DATAALIGN_LEFT

ADC_CHANNEL

- ADC_CHANNEL_0
- ADC_CHANNEL_1
- ADC_CHANNEL…
- ADC_CHANNEL_VREFINT
    - Internal voltage reference
- ADC_CHANNEL_TEMPSENSOR
    - Internal temperature sensor
- ADC_CHANNEL_VBAT
    - VBAT voltage through a divider ladder of factor 1/3 to have VBAT always below VDDA

HAL ADC Functions

HAL ADC Initialization/DeInitialization

- HAL_StatusTypeDef HAL_ADC_Init(ADC_HandleTypeDef *hadc)

- HAL_StatusTypeDef HAL_ADC_DeInit(ADC_HandleTypeDef *hadc)

- void HAL_ADC_MspInit(ADC_HandleTypeDef *hadc)

- void HAL_ADC_MspDeInit(ADC_HandleTypeDef *hadc)

HAL ADC Callbacks

- HAL_StatusTypeDef HAL_ADC_RegisterCallback(ADC_HandleTypeDef *adc, HAL_ADC_CallbackIDTypeDef CallbackID, pADC_CallbackTypeDef pCallback)

- HAL_StatusTypeDef HAL_ADC_UnRegisterCallback(ADC_HandleTypeDef *adc, HAL_ADC_CallbackIDTypeDef CallbackID)

HAL ADC
Blocking Mode - **Polling**

- HAL_StatusTypeDef HAL_ADC_Start(ADC_HandleTypeDef *hadc)

- HAL_StatusTypeDef HAL_ADC_Stop(ADC_HandleTypeDef *hadc)

- HAL_StatusTypeDef HAL_ADC_PollForConversion(ADC_HandleTypeDef *hadc, uint32_t Timeout)

- HAL_StatusTypeDef HAL_ADC_PollForEvent(ADC_HandleTypeDef *hadc, uint32_t EventType, uint32_t Timeout)

HAL ADC
Non Blocking Mode - **Interrupt**

- HAL_StatusTypeDef
  HAL_ADC_Start_**IT**(ADC_HandleTypeDef *hadc)

- HAL_StatusTypeDef
  HAL_ADC_Stop_**IT**(ADC_HandleTypeDef *hadc)

HAL ADC
Non Blocking Mode - **DMA**

- HAL_StatusTypeDef HAL_ADC_Start_**DMA**(ADC_HandleTypeDef *hadc, uint32_t *data, uint32_t length)

- HAL_StatusTypeDef HAL_ADC_Stop_**DMA**(ADC_HandleTypeDef *hadc)

HAL ADC
Retrieve Conversion Value (Polled and Interrupt Driven)

- uint32_t HAL_ADC_GetValue(ADC_HandleTypeDef *hadc)

HAL ADC
IRQ Handler

- void HAL_ADC_IRQHandler(ADC_HandleTypeDef *hadc)

HAL ADC
Peripheral Control

- HAL_StatusTypeDef HAL_ADC_ConfigChannel(ADC_HandleTypeDef *hdac, ADC_ChannelConfTypeDef *sConfig)

- HAL_StatusTypeDef HAL_ADC_AnalogWDGConfig(ADC_HandleTypeDef *hadc, ADC_AnalogWDGConfTypeDef *AnalogWDGConfig)

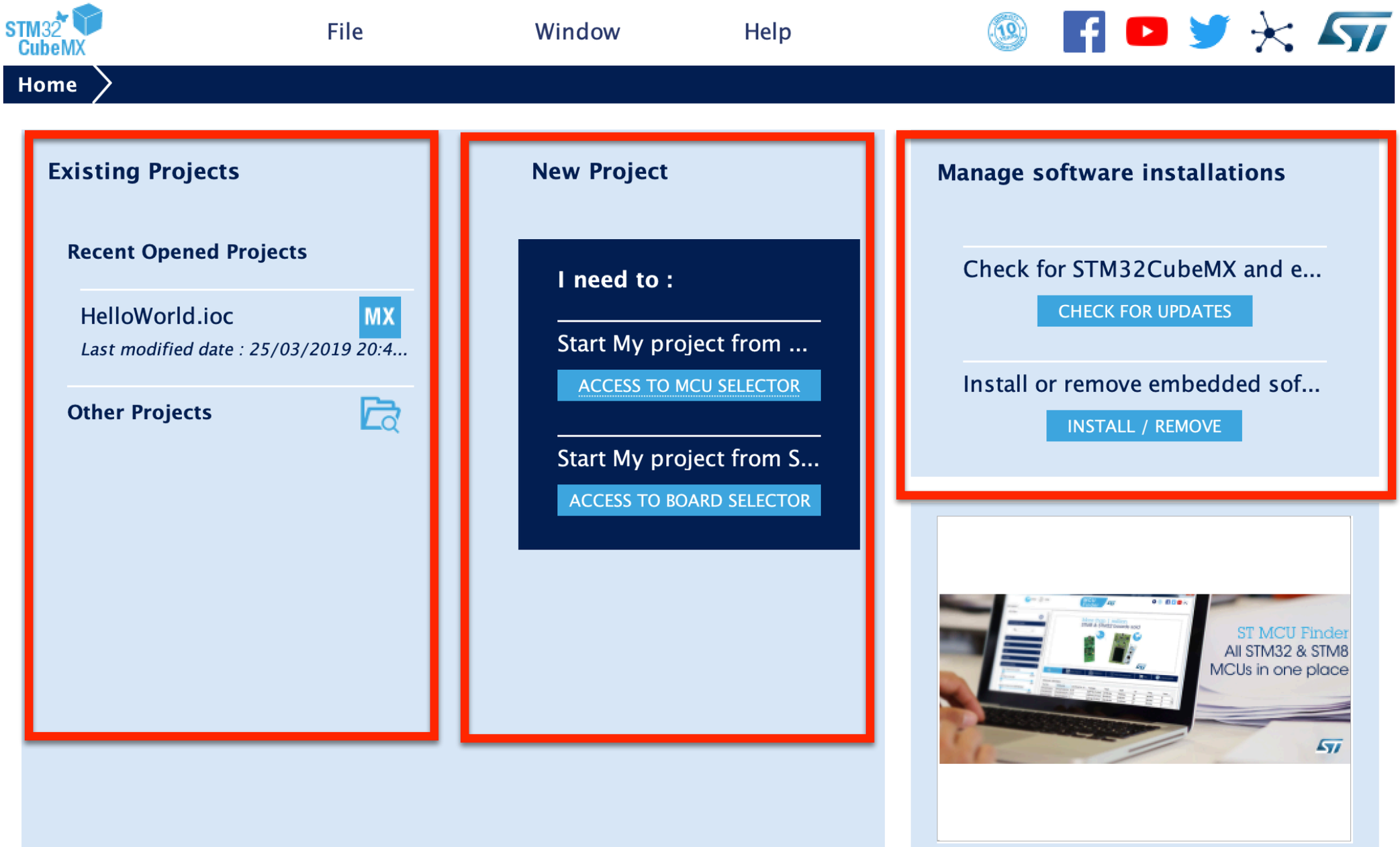
HAL ADC
Other Functions

- uint32_t HAL_ADC_GetState(ADC_HandleTypeDef *hadc)

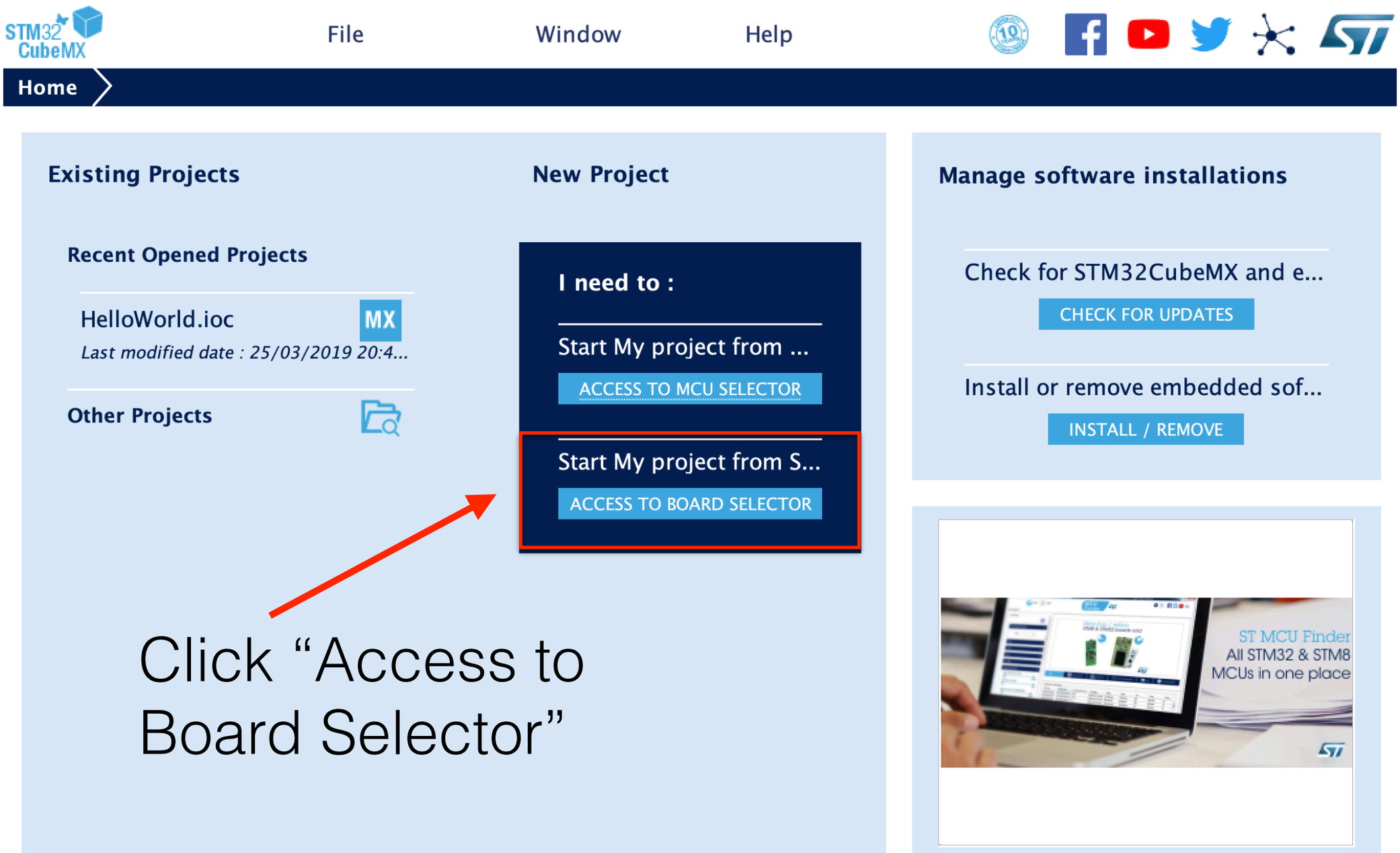
- uint32_t HAL_ADC_GetError(ADC_HandleTypeDef *hadc)

STM32CubeMX and Generation of ADC Code

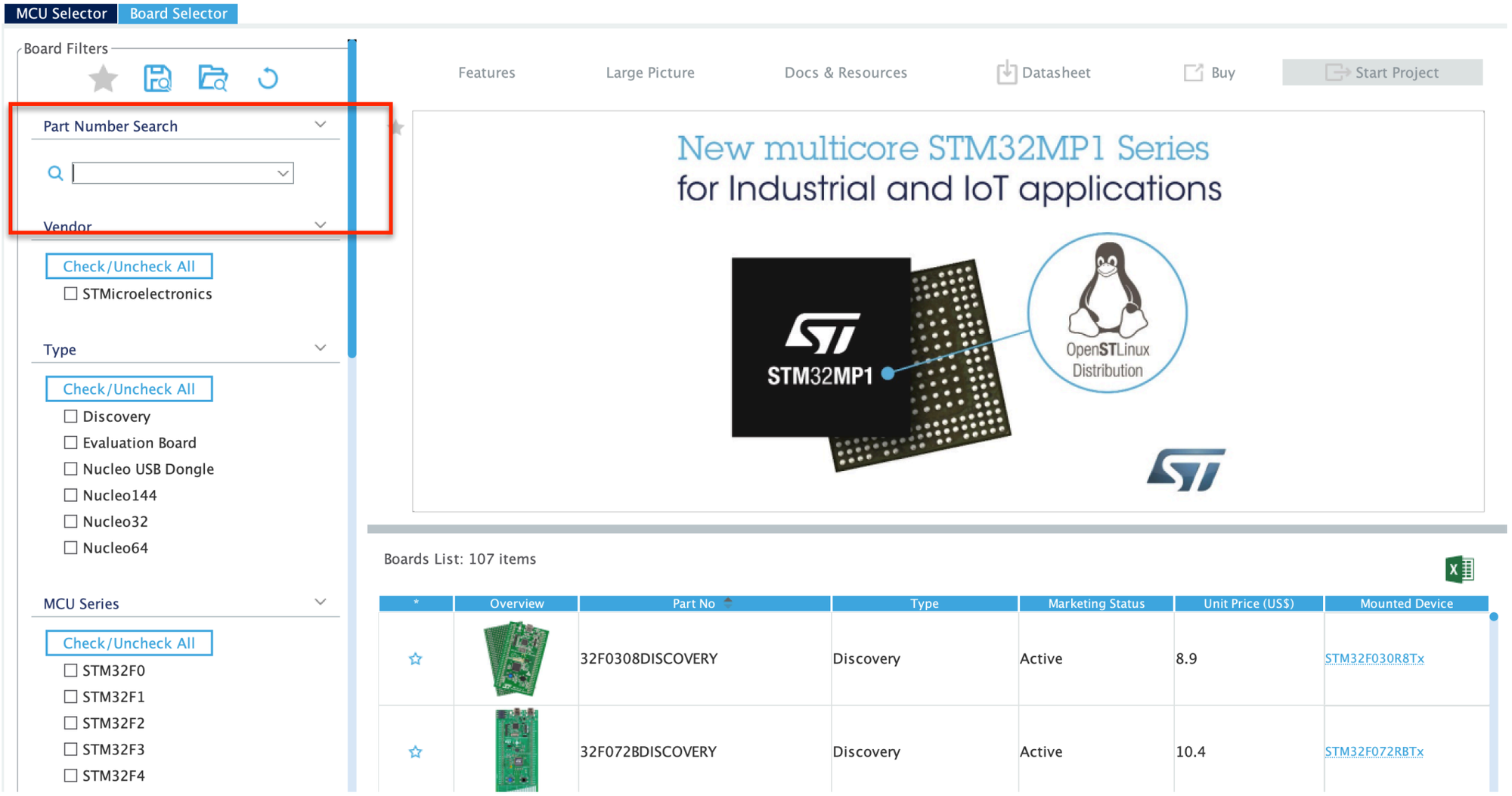
Step: Startup STM32CubeMX



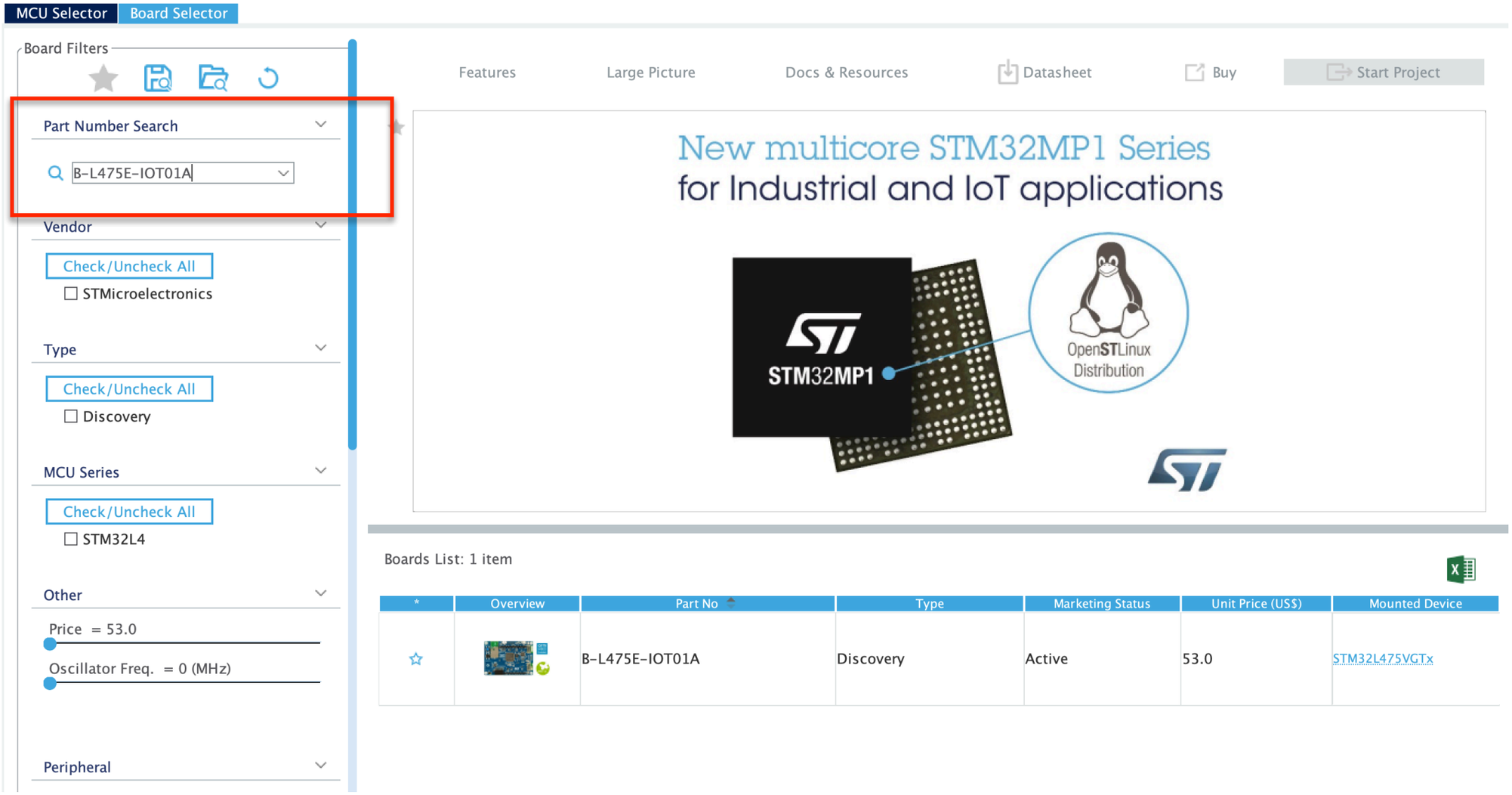
Step: Click on “Access to Board Selector”



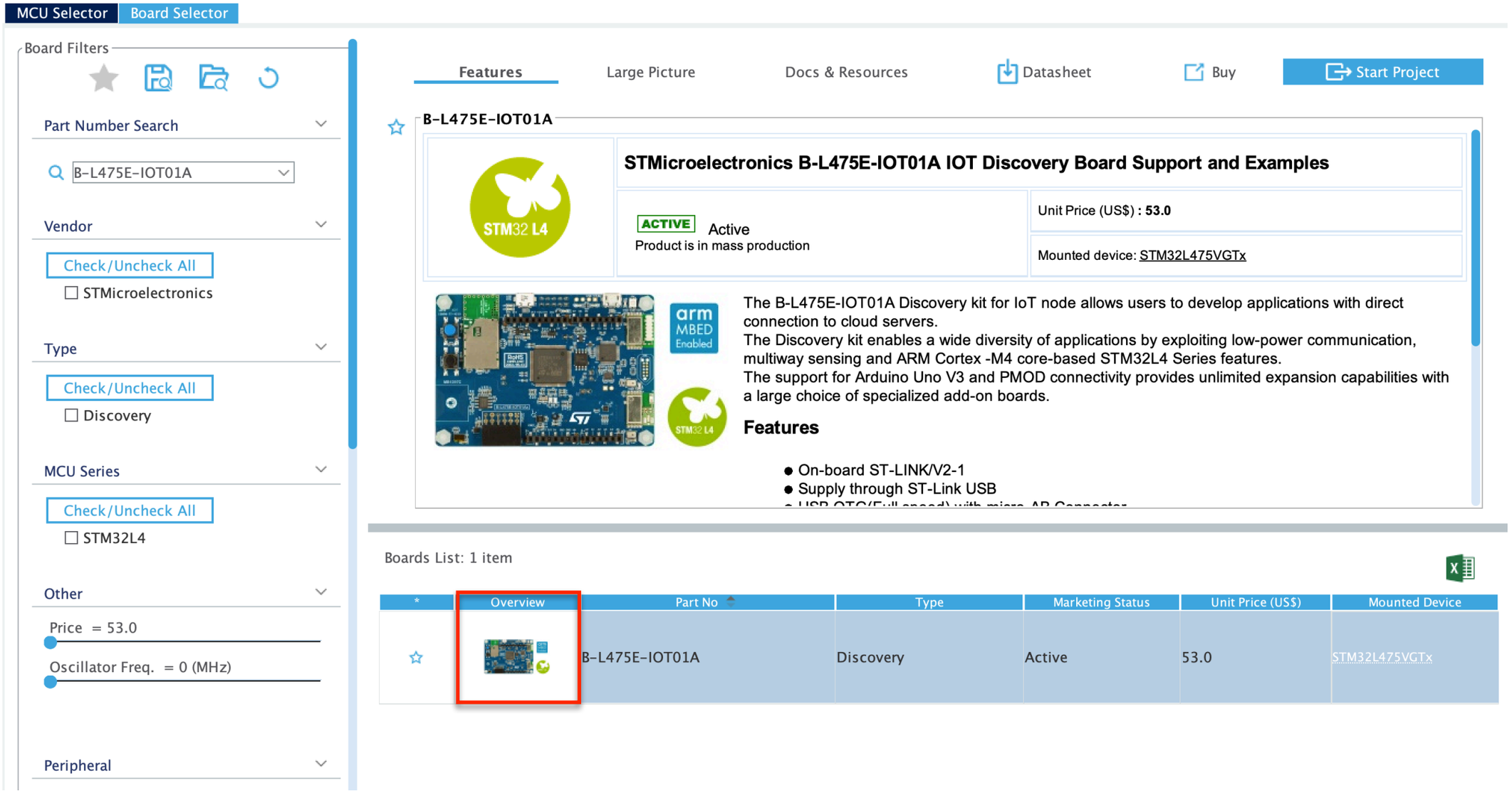
Step: Observe “Part Number Search”



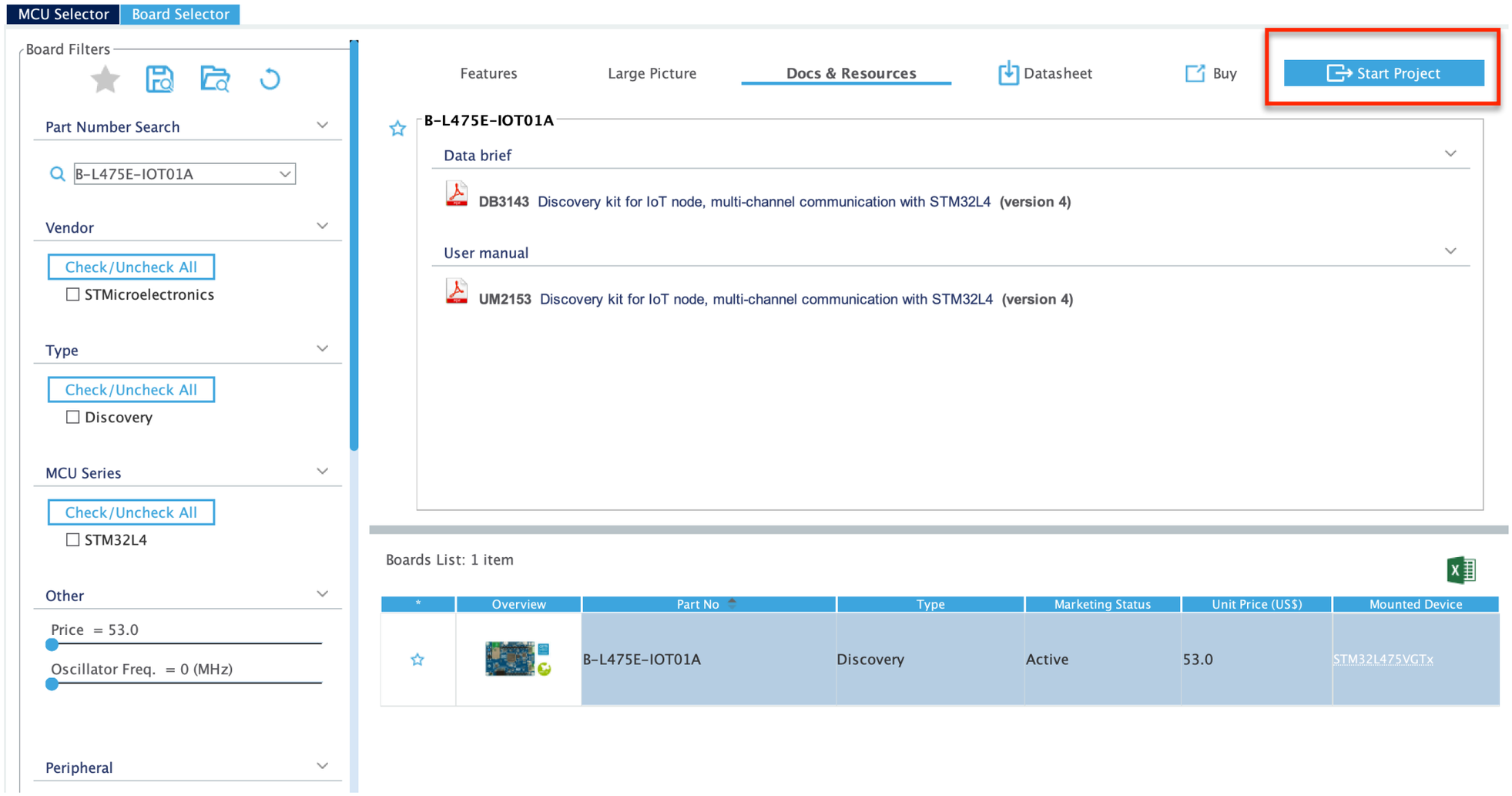
Step: Enter ”B-L475E-IOT01A”



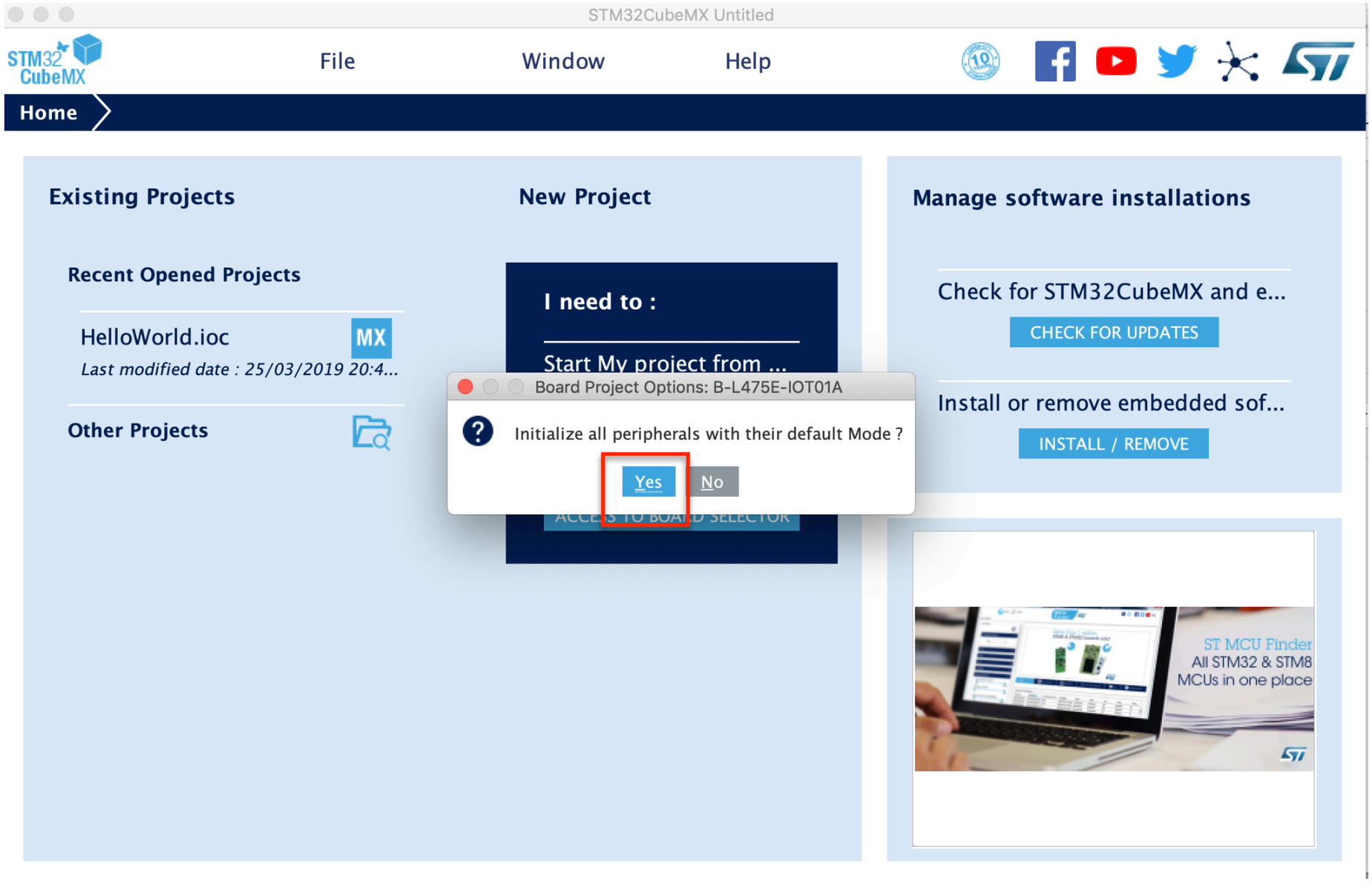
Step: Click on Image, and observe “Features”



Step: Click on “Start Project”

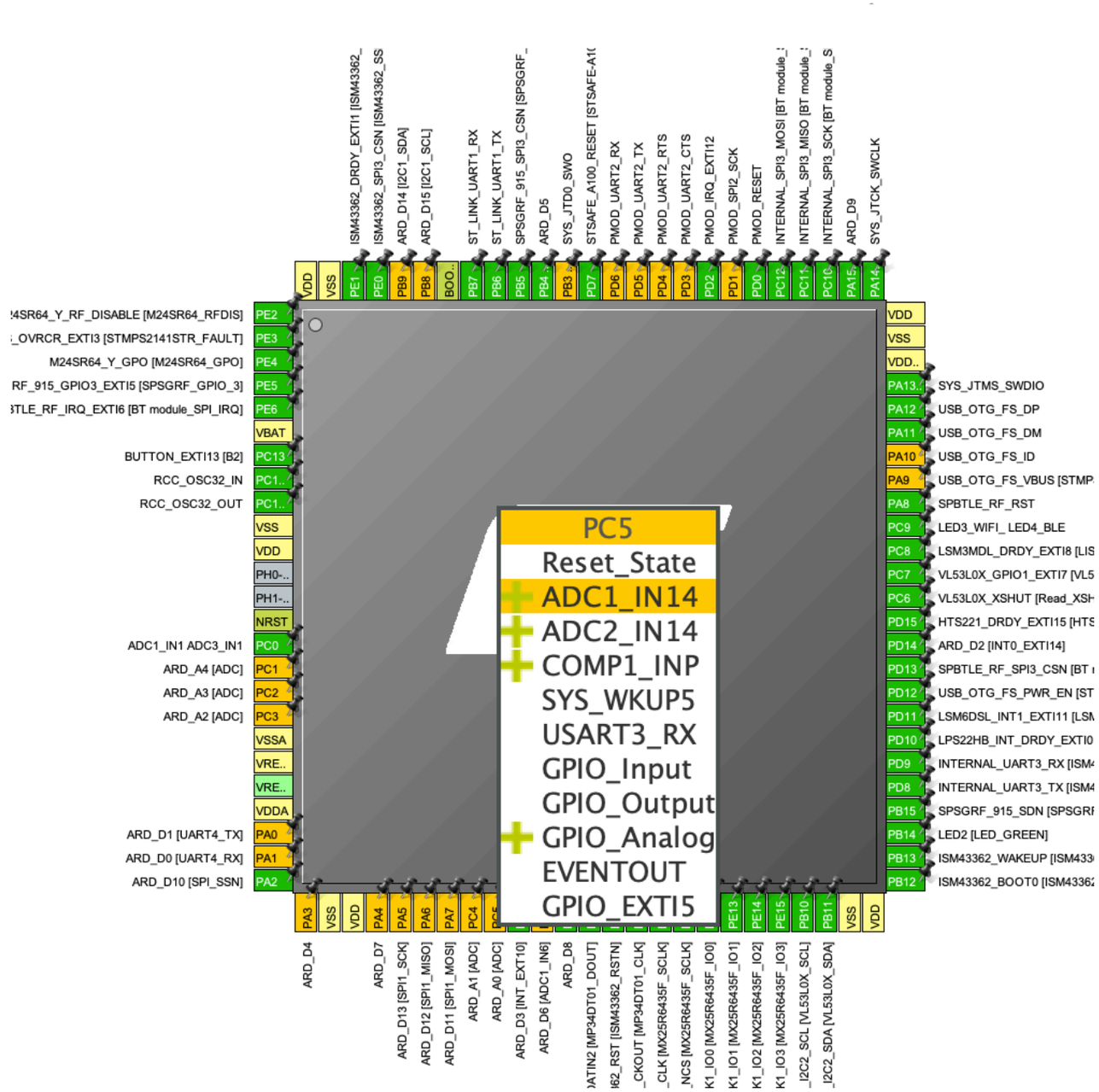


Step: Click on “Yes” (Initialize all …with default mode)



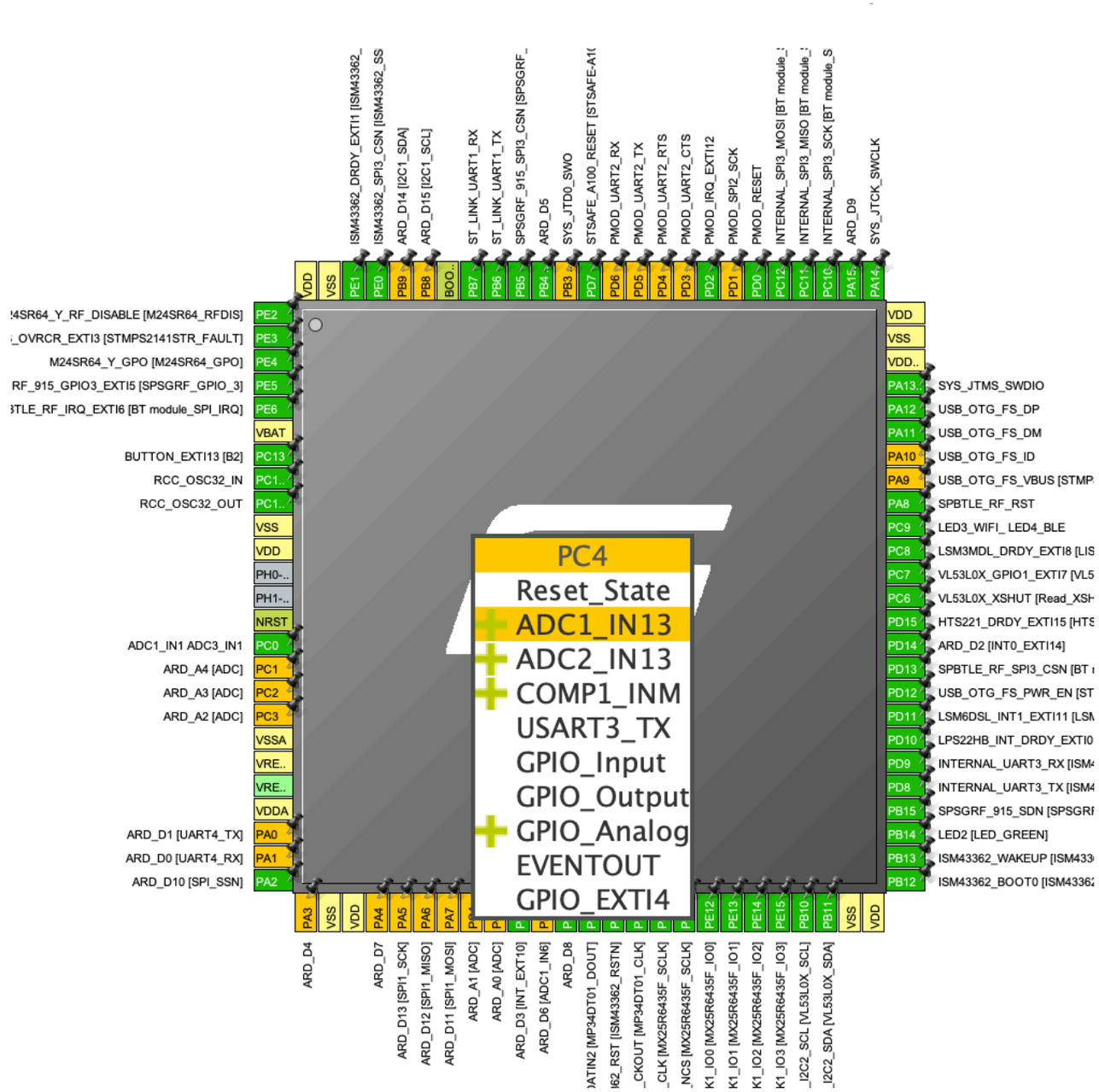
Step: Pin View

PC5 - ADC1_IN14 - ARD_A0

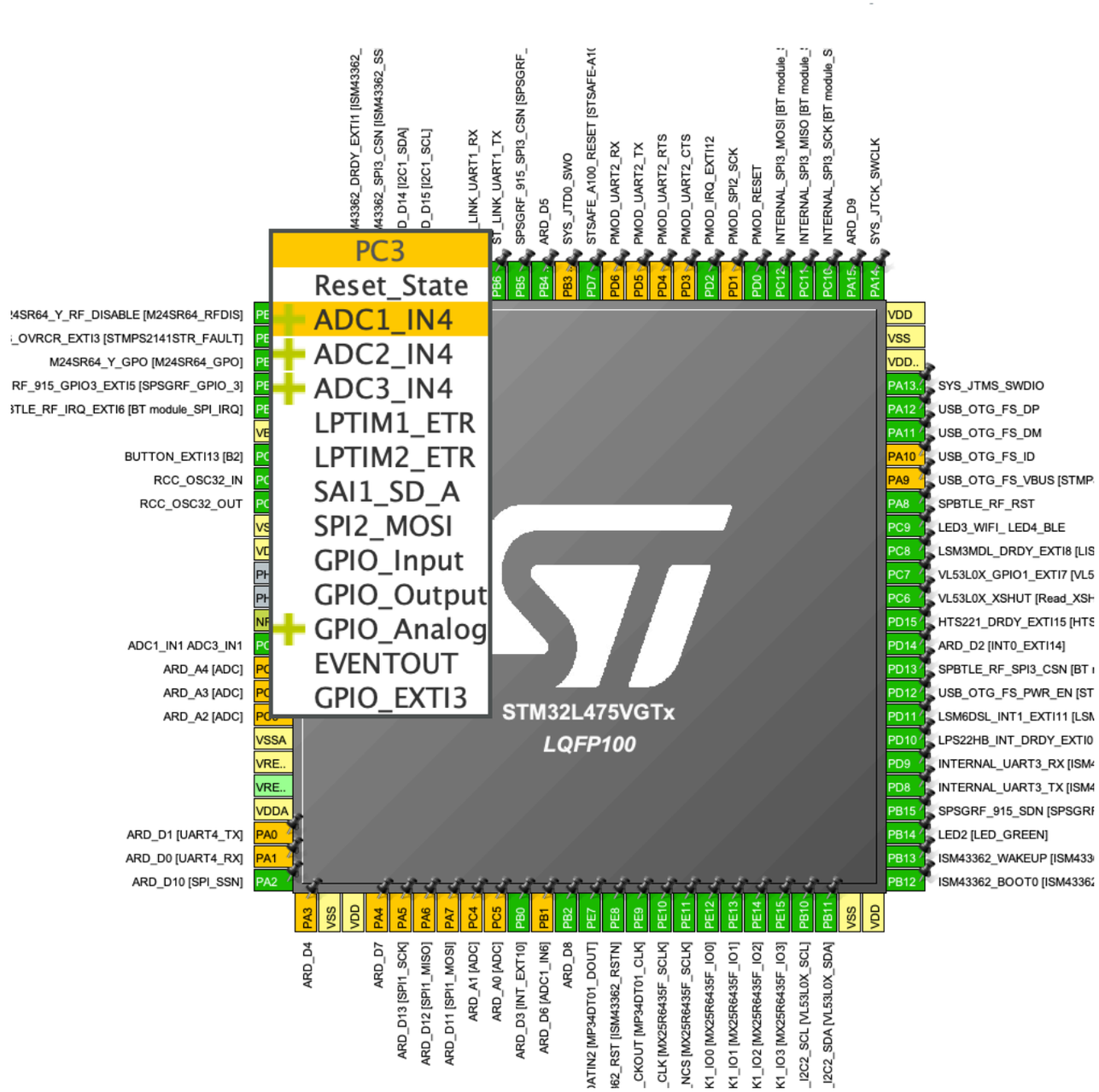


Step: Pin View

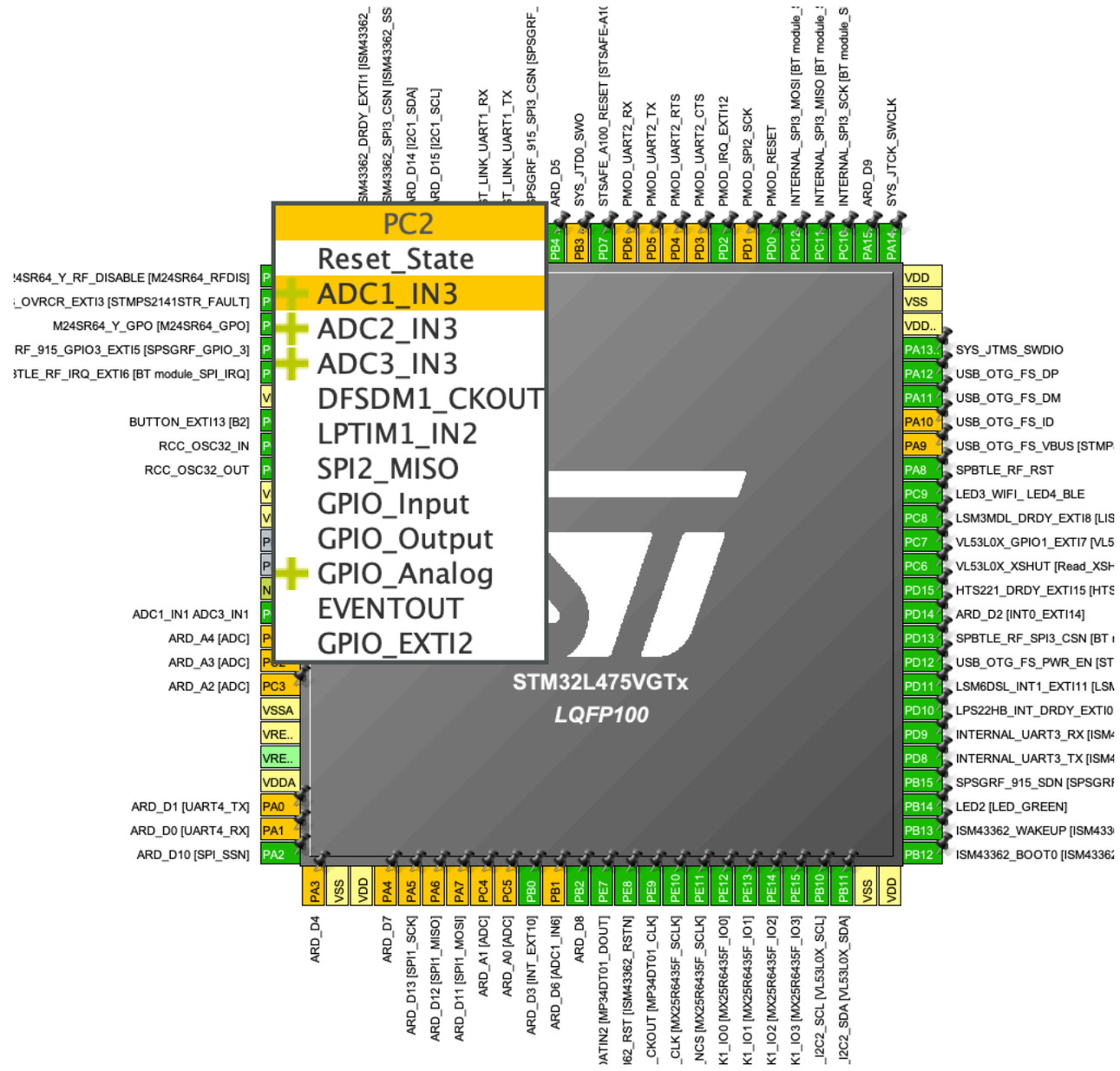
PC4 - ADC1_IN13 - ARD_A1



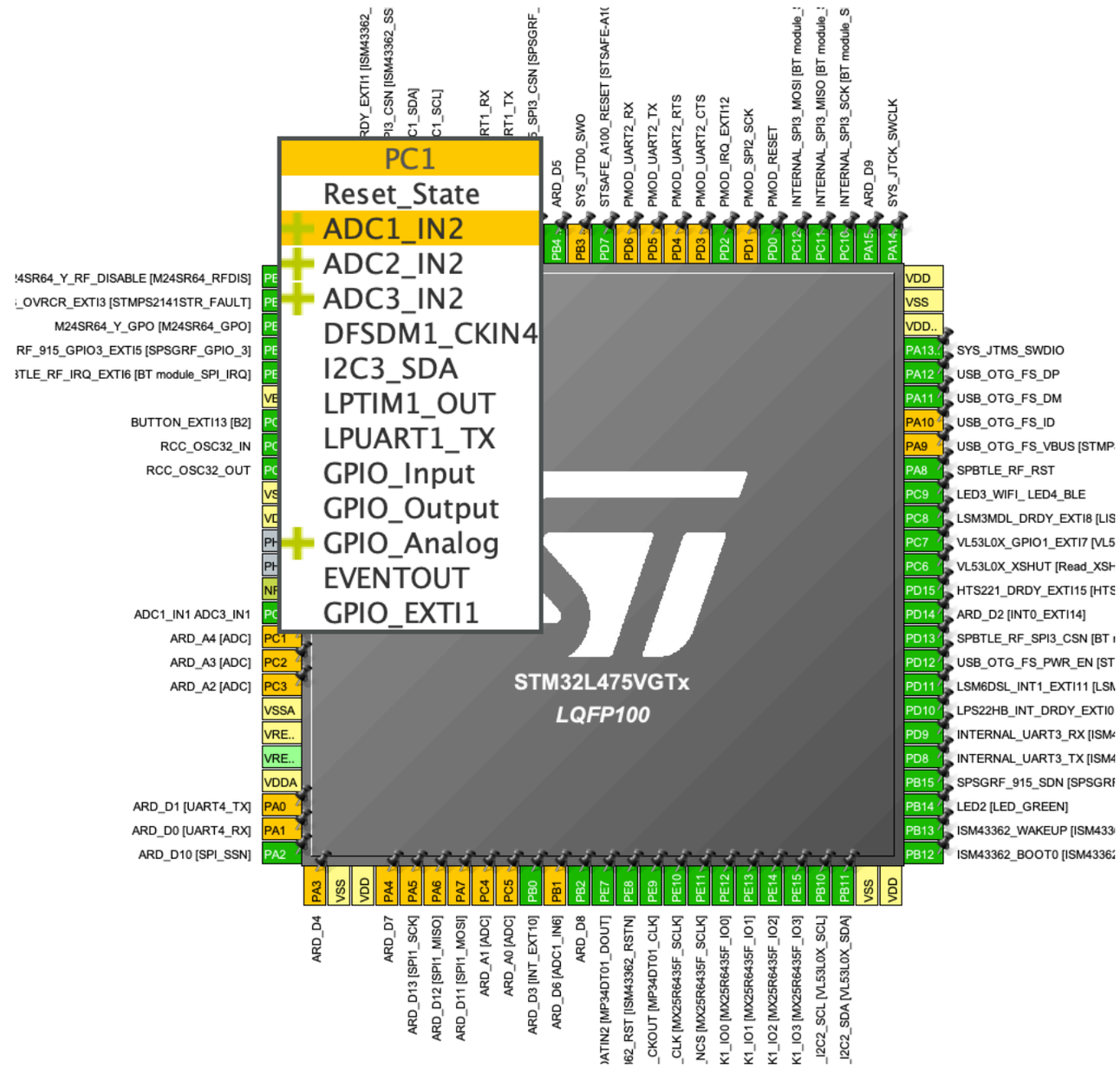
Step: Pin View
PC3 - ADC1_IN4 - ARD_A2



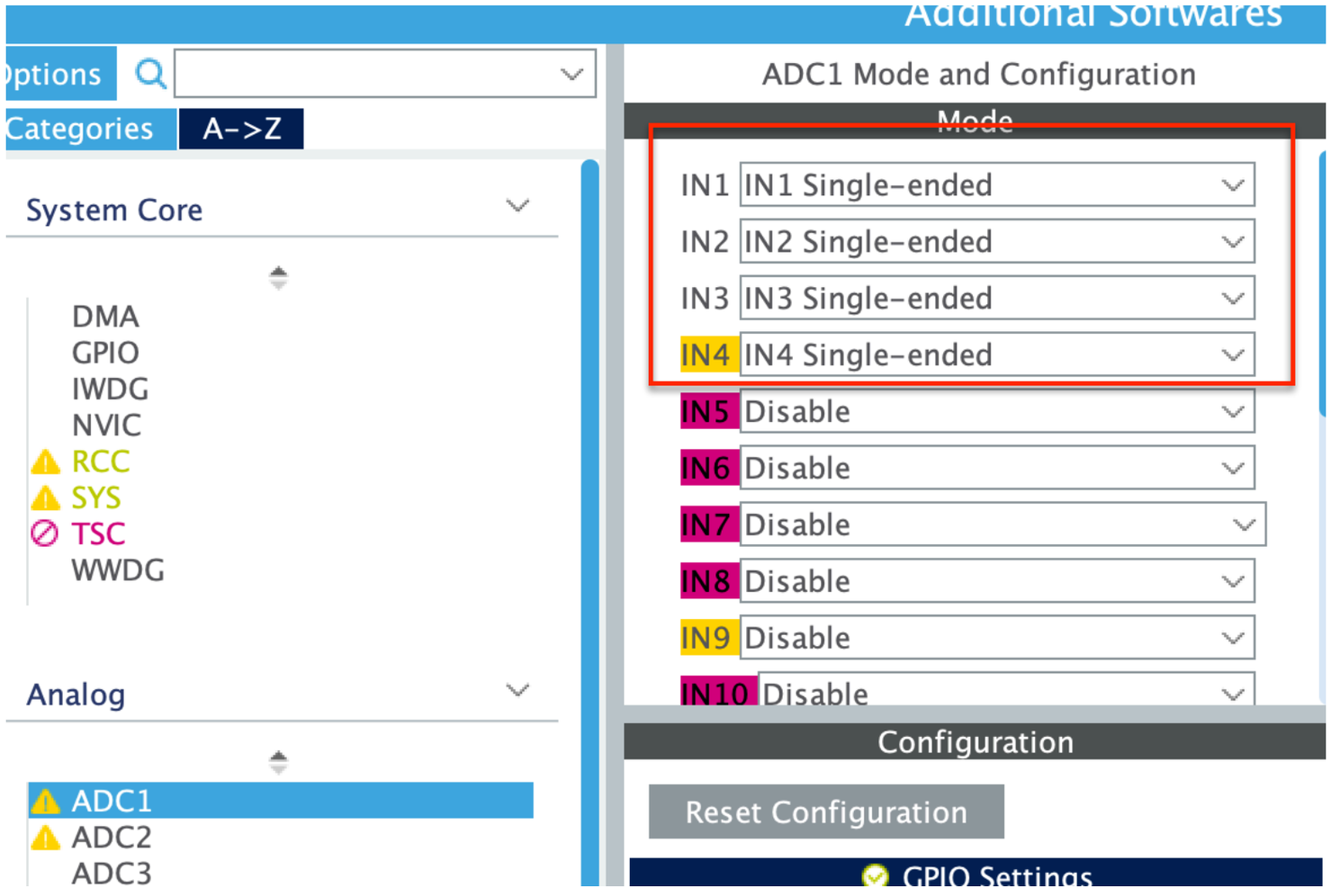
Step: Pin View
PC2 - ADC1_IN3 - ARD_A3



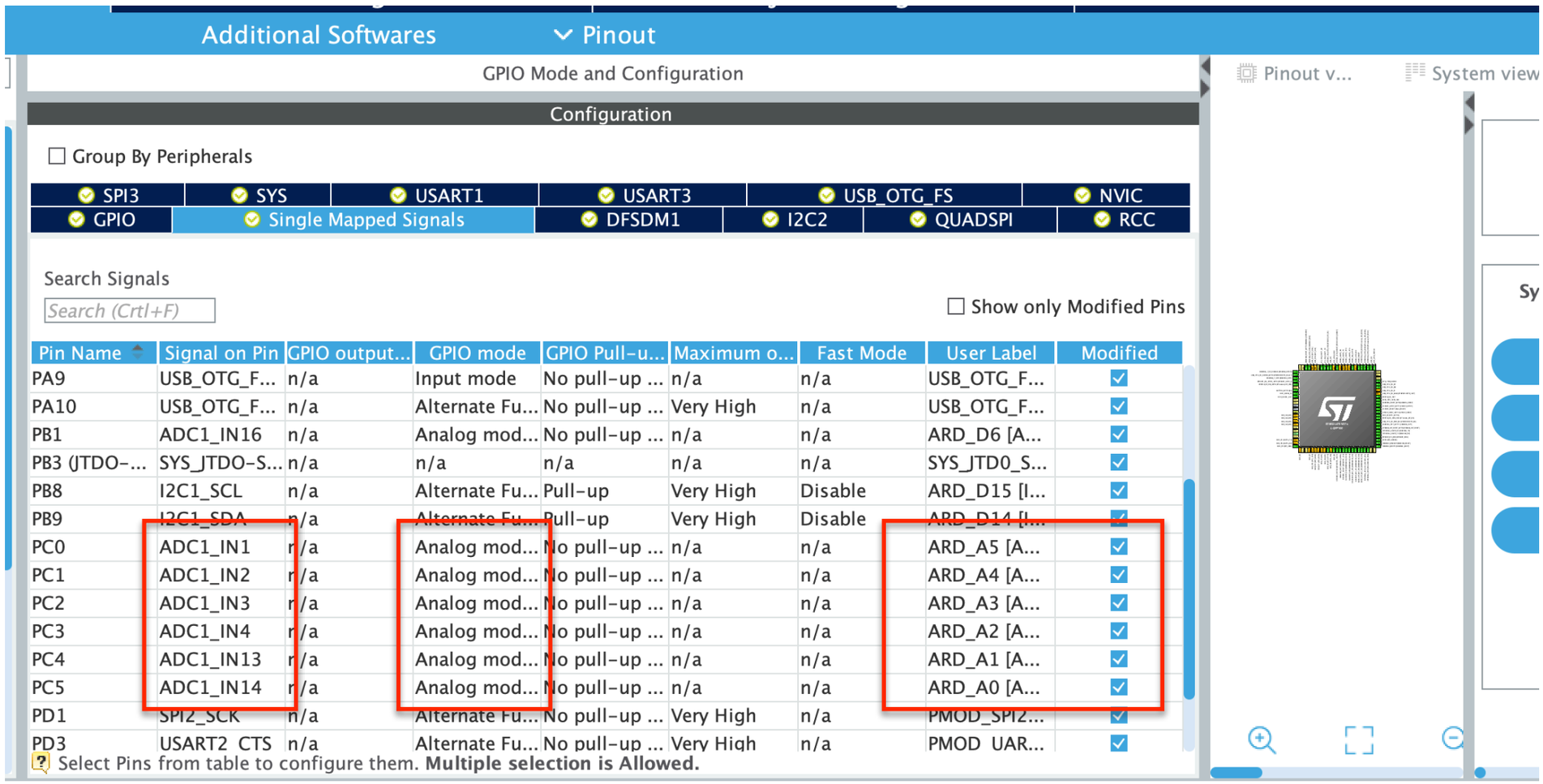
Step: Pin View
PC1 - ADC1_IN2 - ARD_A4



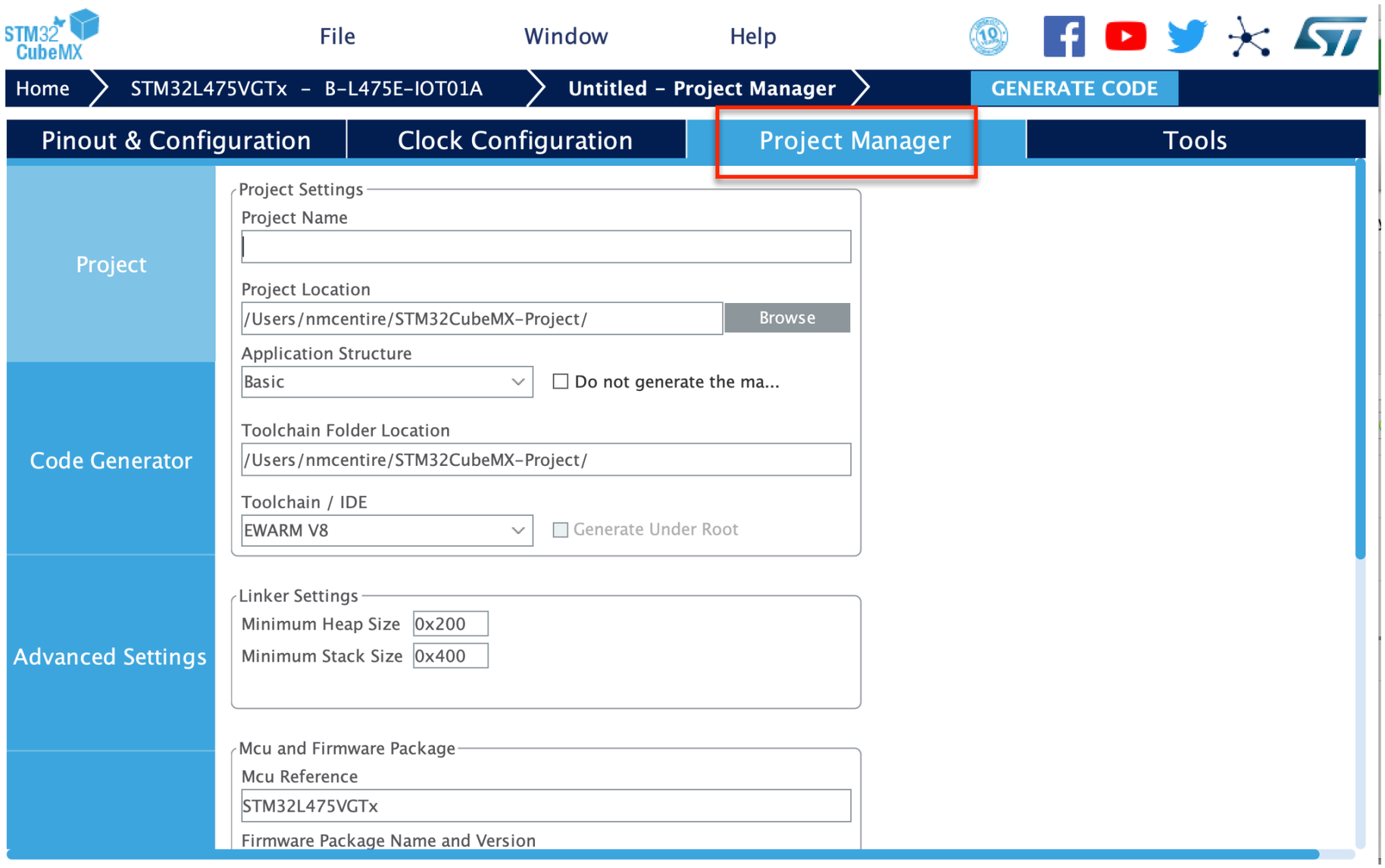
Step: Enable 4 channels for ADC1



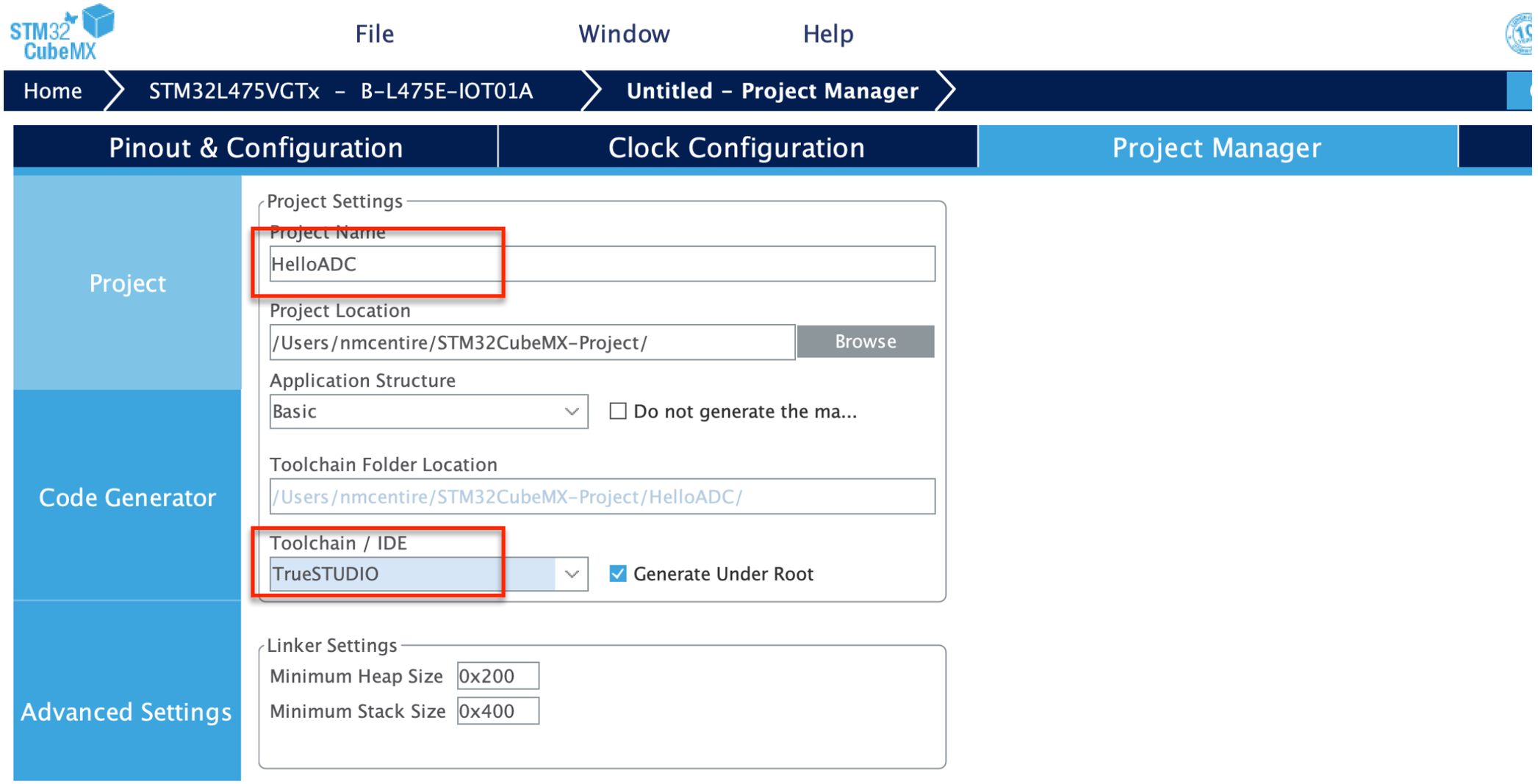
Step: System View, GPIO, Single Mapped Signals



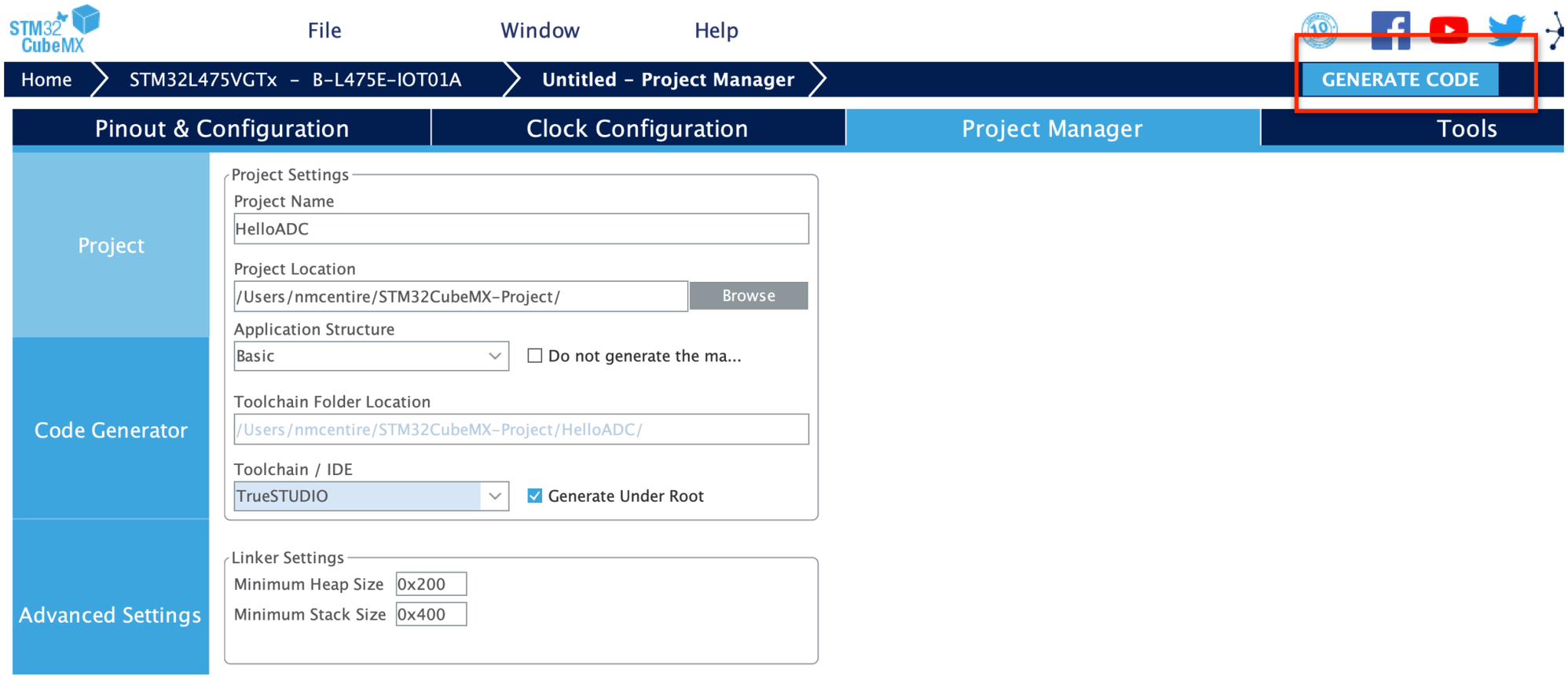
Step: Observe “Project Manager”



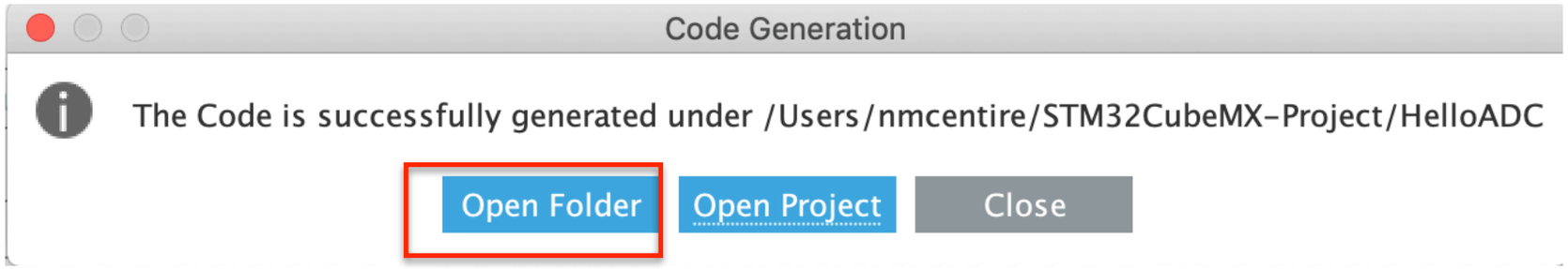
Step: Enter “HelloADC” for Project Name, TrueStudio for IDE



Step: Click on “Generate Code”

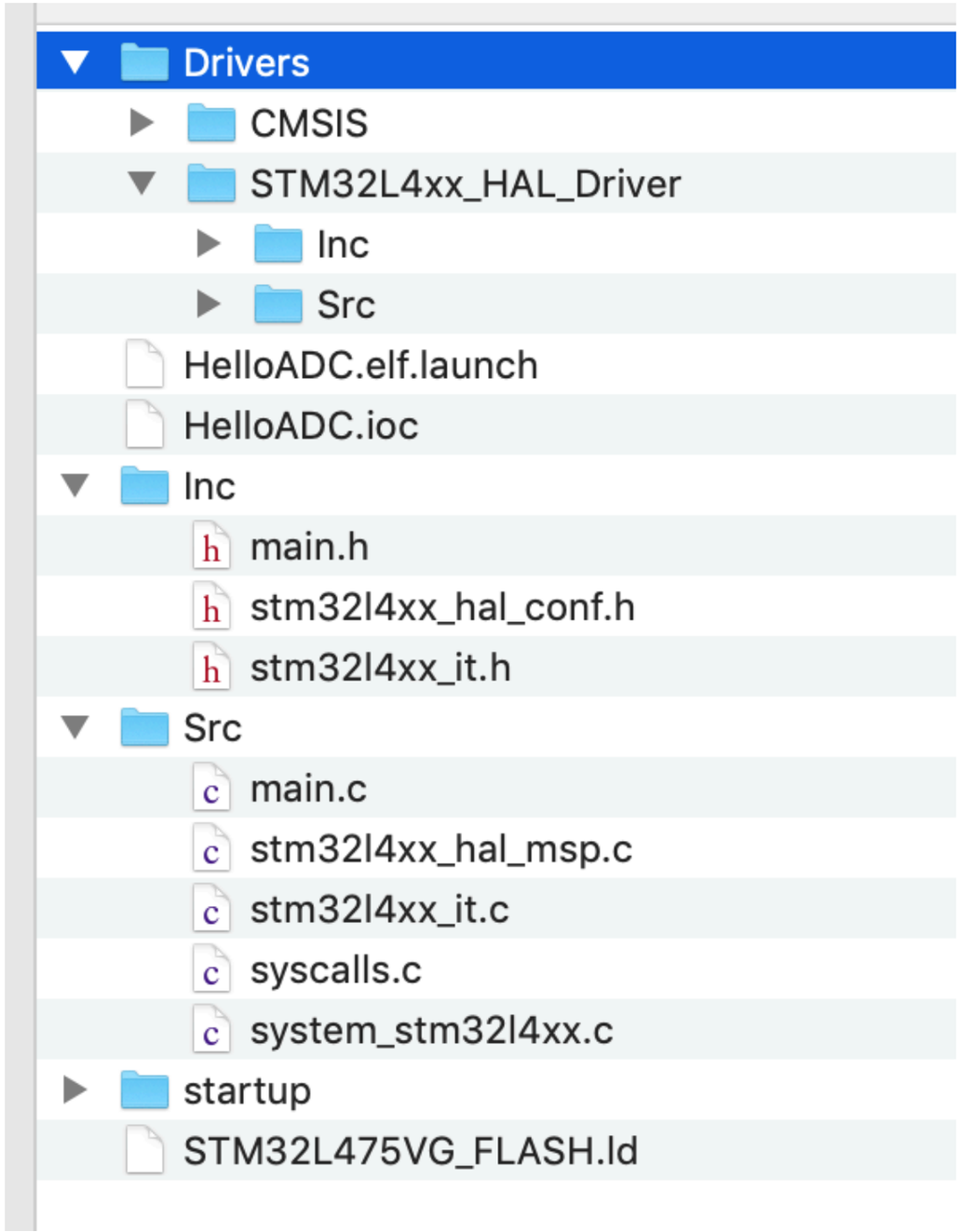


Step: Select “Open Folder”

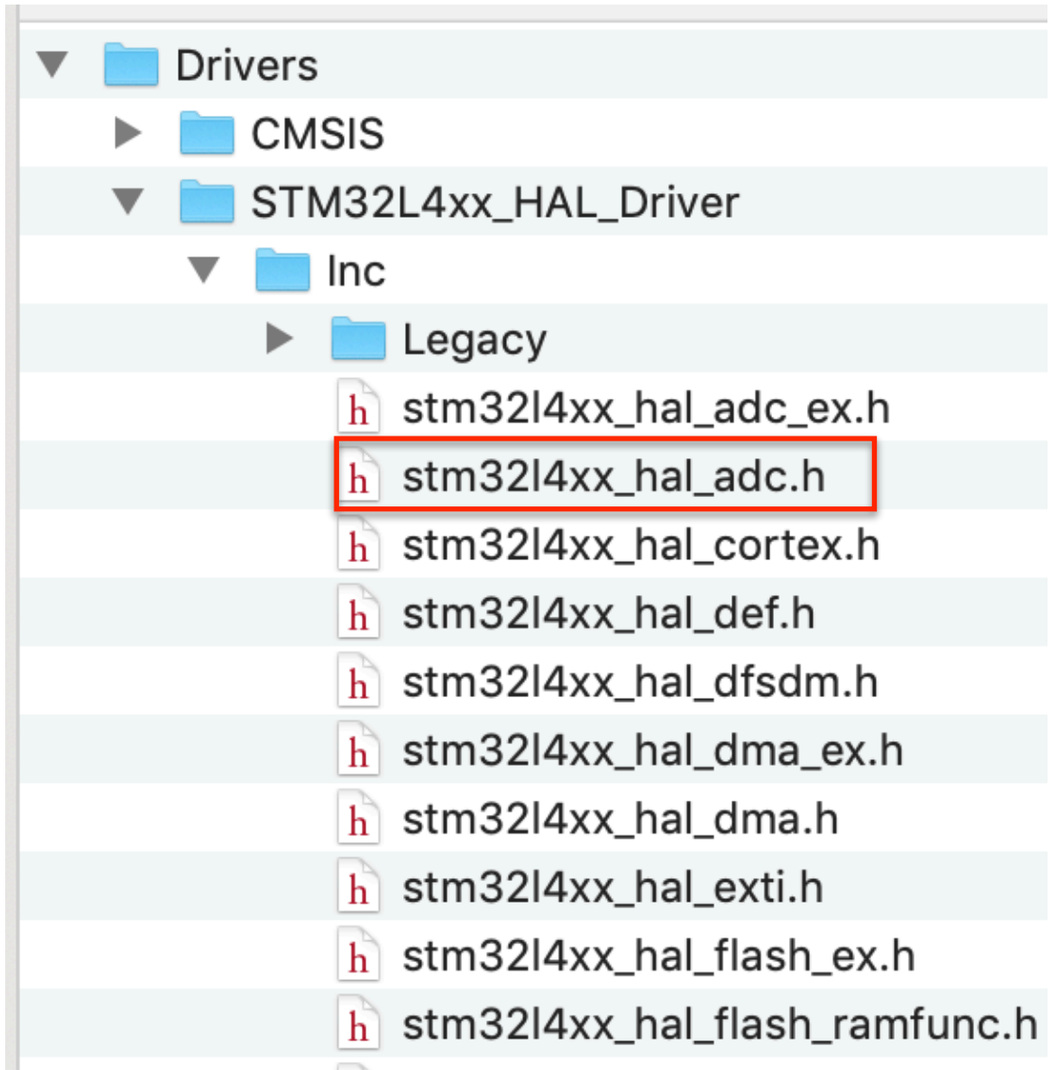


Tour of Generated Project

Step: View Open Folder

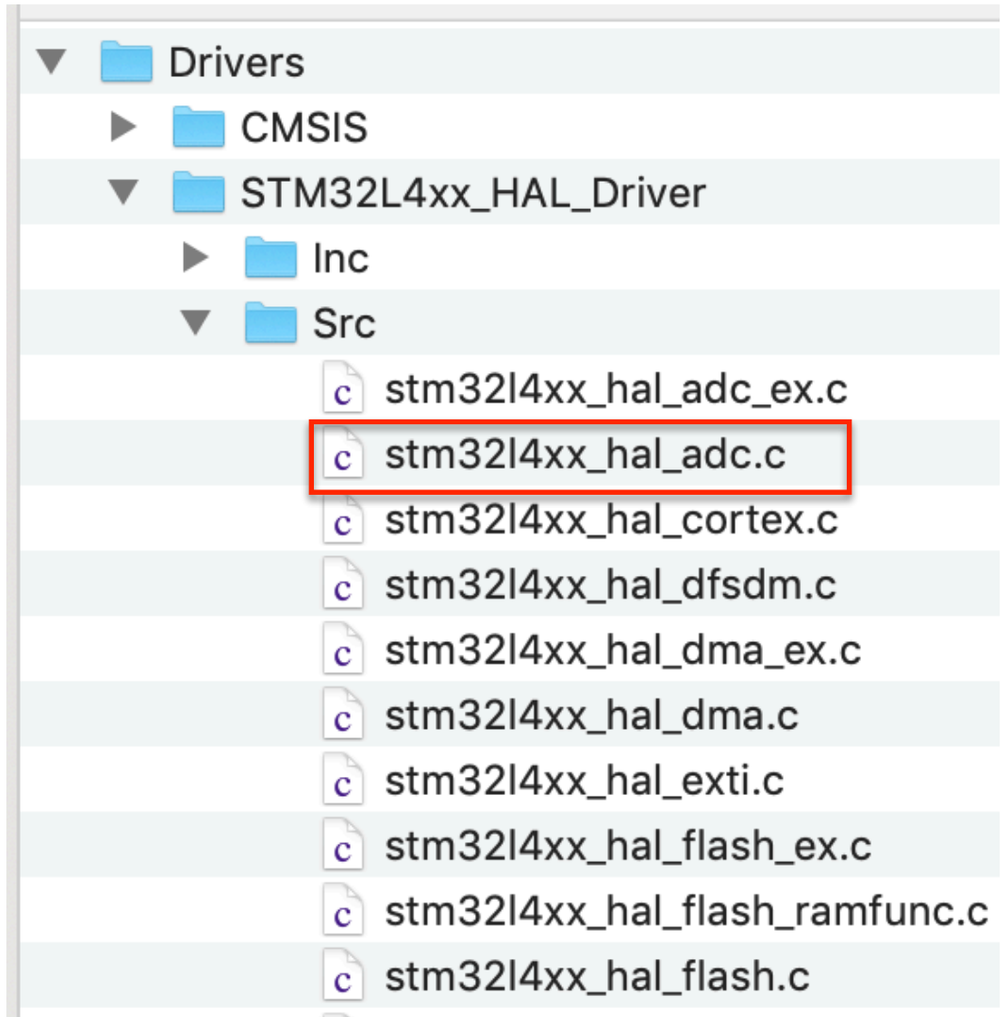


Step: View “Drivers/ STM32L4xx_HAL_Driver/Inc” directory



Observe stm32l4xx_hal_adc.h

Step: View “Drivers/ STM32L4xx_HAL_Driver/Src” directory



Observe stm32l4xx_hal_adc.c

main.c - Part 1

```
44  /* Private variables ---------
45  ADC_HandleTypeDef hadc1;
```

```
74  static void MX_ADC1_Init(void);
```

```
88  int main(void)
89  {
90    /* USER CODE BEGIN 1 */
```

main.c - Part 2

```
216  static void MX_ADC1_Init(void)
217  {
218
219    /* USER CODE BEGIN ADC1_Init 0 */
220
221  /* USER CODE END ADC1_Init 0 */
222
223  ADC_MultiModeTypeDef multimode = {0};
224  ADC_ChannelConfTypeDef sConfig = {0};
225
```

main.c - Part 3

```
229   /** Common config
230   */
231   hadc1.Instance = ADC1;
232   hadc1.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
233   hadc1.Init.Resolution = ADC_RESOLUTION_12B;
234   hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
235   hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
236   hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
237   hadc1.Init.LowPowerAutoWait = DISABLE;
238   hadc1.Init.ContinuousConvMode = DISABLE;
239   hadc1.Init.NbrOfConversion = 1;
240   hadc1.Init.DiscontinuousConvMode = DISABLE;
241   hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
242   hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
243   hadc1.Init.DMAContinuousRequests = DISABLE;
244   hadc1.Init.Overrun = ADC_OVR_DATA_PRESERVED;
245   hadc1.Init.OversamplingMode = DISABLE;
246   if (HAL_ADC_Init(&hadc1) != HAL_OK)
247   {
248     Error Handler();
```

main.c - Part 4

```
250   /** Configure the ADC multi-mode
251   */
252   multimode.Mode = ADC_MODE_INDEPENDENT;
253   if (HAL_ADCEx_MultiModeConfigChannel(&hadc1, &multimode) != HAL_OK)
254   {
255     Error_Handler();
256   }
```

main.c - Part 5

```
257   /** Configure Regular Channel
258   */
259   sConfig.Channel = ADC_CHANNEL_1;
260   sConfig.Rank = ADC_REGULAR_RANK_1;
261   sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
262   sConfig.SingleDiff = ADC_SINGLE_ENDED;
263   sConfig.OffsetNumber = ADC_OFFSET_NONE;
264   sConfig.Offset = 0;
265   if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
266   {
267     Error_Handler();
268   }
```

main.c - Part 6

```
589   /*Configure GPIO pins : PC0 ARD_A4_Pin ARD_A3_Pin ARD_A2_Pin
590                            ARD_A1_Pin ARD_A0_Pin */
591   GPIO_InitStruct.Pin = GPIO_PIN_0|ARD_A4_Pin|ARD_A3_Pin|ARD_A2_Pin
592                         |ARD_A1_Pin|ARD_A0_Pin;
593   GPIO_InitStruct.Mode = GPIO_MODE_ANALOG_ADC_CONTROL;
594   GPIO_InitStruct.Pull = GPIO_NOPULL;
595   HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

Summary so far…

- STM32CubeMX generates GPIO Code
- HAL Driver Code
- Drivers/STM32L4xx_HAL_Driver/Inc
  - stm32l4xx_hal_adc.h
- Drivers/STM32L4xx_HAL_Driver/Src
  - stm32l4xx_hal_adc.c

Use TrueStudio
To Read the value of
Arduino A0 (analog read)

Step: Edit main.c - Part 1

```
static void MX_ADC1_Init(void)
{

  /** Configure Regular Channel
  */
  sConfig.Channel = ADC_CHANNEL_14;
  sConfig.Rank = ADC_REGULAR_RANK_1;
  sConfig.SamplingTime = ADC_SAMPLETIME_2CYCLES_5;
  sConfig.SingleDiff = ADC_SINGLE_ENDED;
  sConfig.OffsetNumber = ADC_OFFSET_NONE;
  sConfig.Offset = 0;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
```

ARD_A0 => PC5 => ADCX1_IN14

Step: Edit main.c - Part 2

```
88  int main(void)
89  {
```

```
uint32_t adcResult1;
HAL_StatusTypeDef halResult1;

int main(void)
{

    HAL_ADC_Start(&hadc1);
    halResult1 = HAL_ADC_PollForConversion(&hadc1, 100);
    adcResult1 = HAL_ADC_GetValue(&hadc1);


    HAL_Delay(200);
```

Summary

- Introduction to ADC and Embedded C
- STM32CubeMX and ADC Code Generation
- Tour of ADC
- TrueStudio and ADC
    - Read ADC value