AX58101 GpioAio User Guide

Revision 1.00
September 23rd, 2025

Revision History

Revision	Date	Description
1.00	2025/09/23	Initial release

CONTENT

List of Figure

List of Table

1. Introduction

This document introduces how to setup entire environment for GPIO / AIO application on AX58101 evaluation platform (NUCLEO-F303RE + AX58101-EVB-SSPDI boards). The demo application of EtherCAT MainDevice side is developed using PLC (Programmable Logic Controller) language running on BECKHOFF TwinCAT engineering (XAE). The IDE relies on Microsoft Visual Studio 2010 / 2013. We can execute the application through the IDE or pure BECKHOFF runtime package (XAR). The firmware part of EtherCAT SubDevice side is developed using C language through ARM KEIL MDK (Microcontroller Development Kit).

The firmware mainly handles the tasks below:
- BECKHOFF SSC (SubDevice Stack Code)
- Hardware access of SSC through SPI interface at MCU side
- Simple object form for CIA-401 digital/analog IO
- Peripherals (GPIO / ADC / DAC) accessed by EtherCAT PD (Process Data)

The PLC at EtherCAT MainDevice side mainly handles the tasks below:
- PD mapping / handling by PLC for GPIO / AIO
- A demo GUI

2. Requirements

Before starting to setup the environment, the user should prepare some software and hardware. Of course, they are all running based on general desktop PC with windows 10 / 11 operation systems. The software and hardware requirements are listed as following:

[Software at EtherCAT MainDevice side]

- Windows 10 / 11 operation system
- Microsoft Visual Studio 2010 / 2013
- Beckhoff TwinCAT XAE v3.1.4022.28 or later
  You can download the BECKHOFF TwinCAT here

Figure 2-1

[Software at EtherCAT SubDevice side]

- STM32CubeMX version 6.14.1 (for development only)
  You can download the STM32CubeMX here.

Figure 2-2

- In this reference design, two types of Integration Development Environment (IDE) could be supported, you can choose one for the development.
  - ARM KEIL MDK micro-Vision 5 or later (for development only)

Figure 2-3

- STM32 CubeIDE version 1.19.0 or later (for development only)
  You can download the STM32CubeIDE here.

Figure 2-4

- STM32CubeProgrammer
  You can download the STM32CubeProgrammer here.

Figure 2-5

- SSC Tool V5.13 (for development only)
  You can download the SSC Tool from EtherCAT Technology Group (ETG) here

Figure 2-6

[Hardware]

- ST NUCLEO-F303RE Control Board x 1 PCS

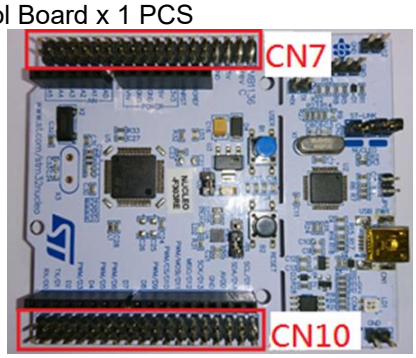


Figure 2-7

- AX58101-EVB-SSPDI-1 V1.0 EVB x 1 PCS
  For more details, please refer to the link below and the following chapter.
  ASIX official link: https://www.asix.com.tw/

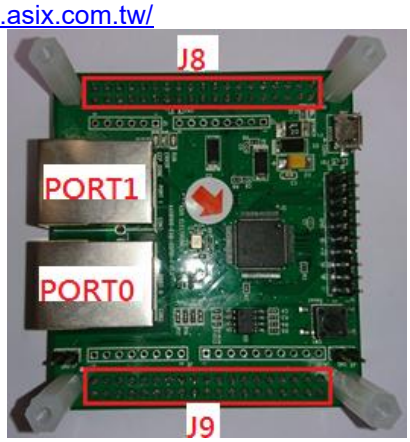


Figure 2-8

3. Board Description

3-1 AX58101-EVB Evaluation Board

AX58101-EVB is an EtherCAT SubDevice Controller (ESC) Evaluation Board with AX58101 IC, please refer to ASIX official website as below for more detail.
https://www.asix.com.tw/en/product/IndustrialEthernet/EtherCAT/AX58101

The AX58101-EVB-SSPDI is an evaluation board designed for the AX58101 IC and includes the following components:

- J8 – Pins that provides GPIO or additional connections, typically interfacing with other devices or expansion boards.
- J9 – Pins that provides GPIO or additional connections, like J8.
- ESC Port0 – Port0 of the ESC
- ESC Port1 – Port1 of the ESC
- Reset – A reset button of AX58101

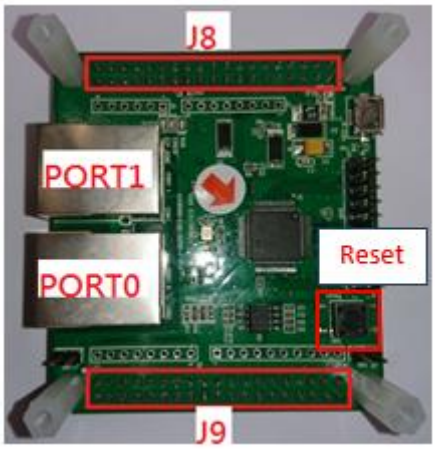


Figure 3-1

3-2 NUCLEO-F303RE

The STM32 Nucleo-64 board offers an affordable platform for prototyping with STM32 microcontrollers, combining performance and low power. It supports ARDUINO® Uno V3 and ST morpho headers for expansion and includes an integrated ST-LINK debugger/programmer. Free software libraries and examples are available via the STM32Cube MCU Package. For more details, visit the official product page:
https://www.st.com/en/evaluation-tools/nucleo-f303re.html

The NUCLEO-F303RE functions as the control board for the AX58101 GPIO / AIO application and provides the following components:

- USB – ST-LINK USB mini-B connector
- CN7 – ST morpho connector
- CN10 – ST morpho connector
- USER – User button
- Reset – Reset button

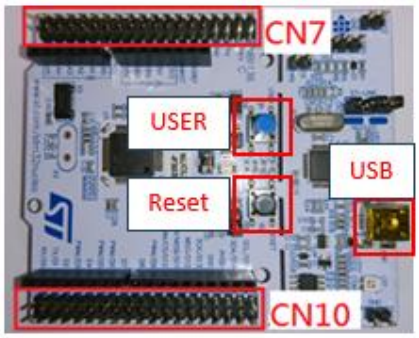
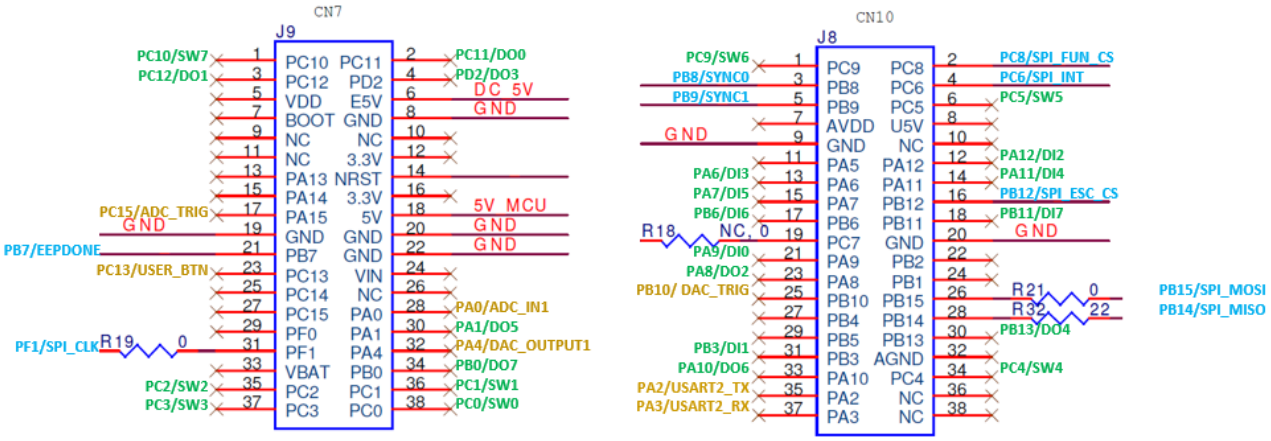


Figure 3-2

Pinout



4. Environment Setup

4-1 Hardware Connectivity

Please refer to the diagram below to see the combination of the NUCLEO-F303RE and AX58101-EVB-SSPDI.

J8 (AX58101-EVB-SSPDI) connects to CN10 (NUCLEO-F303RE), and J9 (AX58101-EVB-SSPDI) connects to CN7 (NUCLEO-F303RE).

Connect the "**EtherCAT Port 0**" and "**USB Port**" to your host:

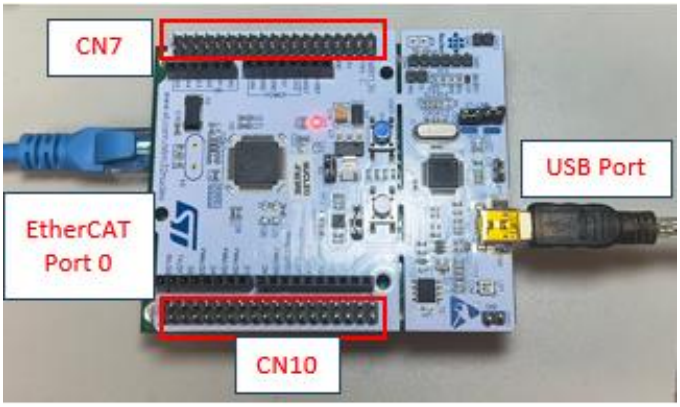


Figure 4-1

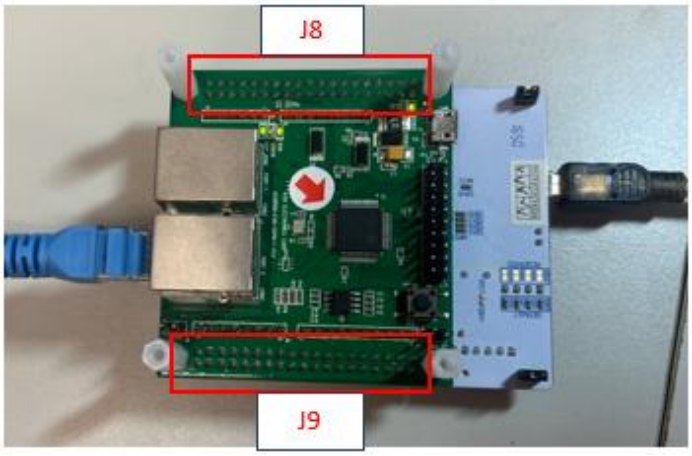


Figure 4-2

5. Building Reference Design Begin with STM32CubeMX

Here will introduce the procedure to build up the complete source code package, please follow the steps below to do this.

Note: Sections 5-1 to 5-4 provide a step-by-step guide starting from creating a new project. If you only need to generate the SSC code, you may skip these sections and go directly to Section 5-5.

5-1 Create New Project in STM32CubeMX

Step 1: Execute the STM32CubeMX, then click "**ACCESS TO MCU SELECTOR**" on main page.

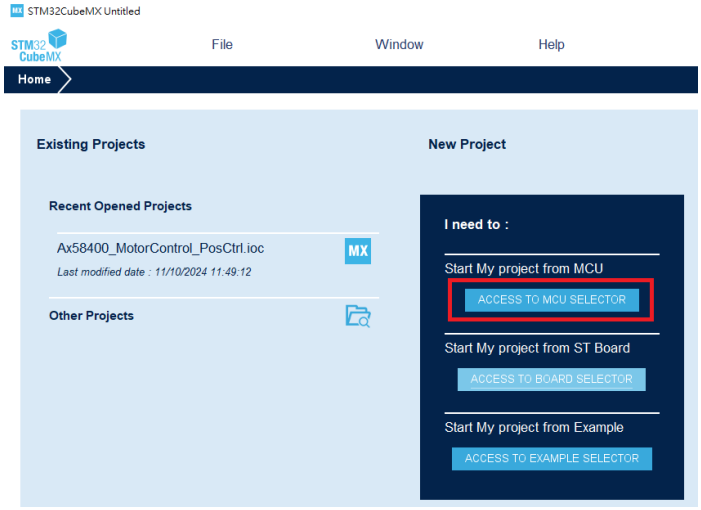


Figure 5-1

Step 2: Searching part number "**STM32F303RET6**", then select it on "**MCUs/MPU List**" and press "**Start Project**" to create project.

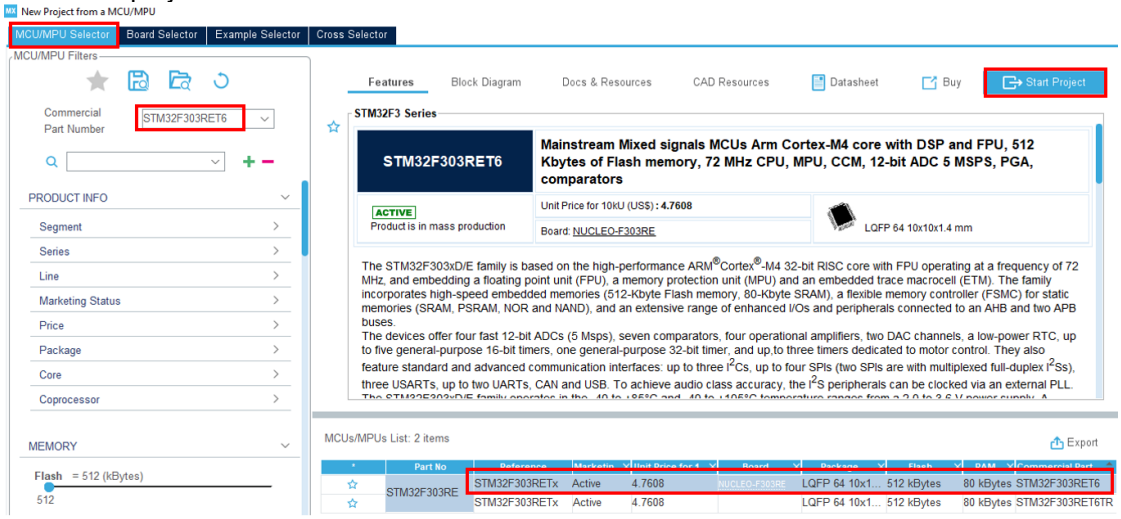


Figure 5-2

Note: To keep the corrected reference path in BSP, please save the project folder in the path below.
BSP_ROOT\SampleCode

5-2 Modify Peripheral Settings in STM32CubeMX

Step 1: Disable the "**Pinout & Configuration > System Core > RCC > High Speed Clock (HSE)**" setting as the figure below.

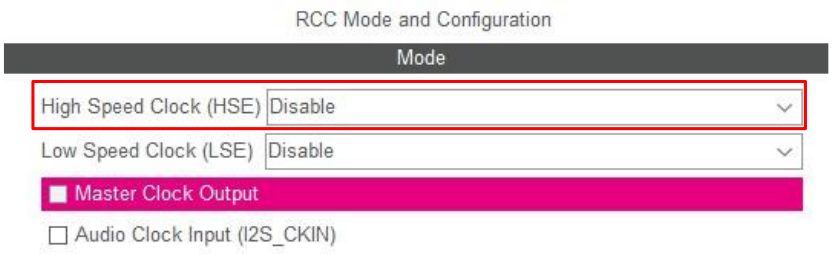
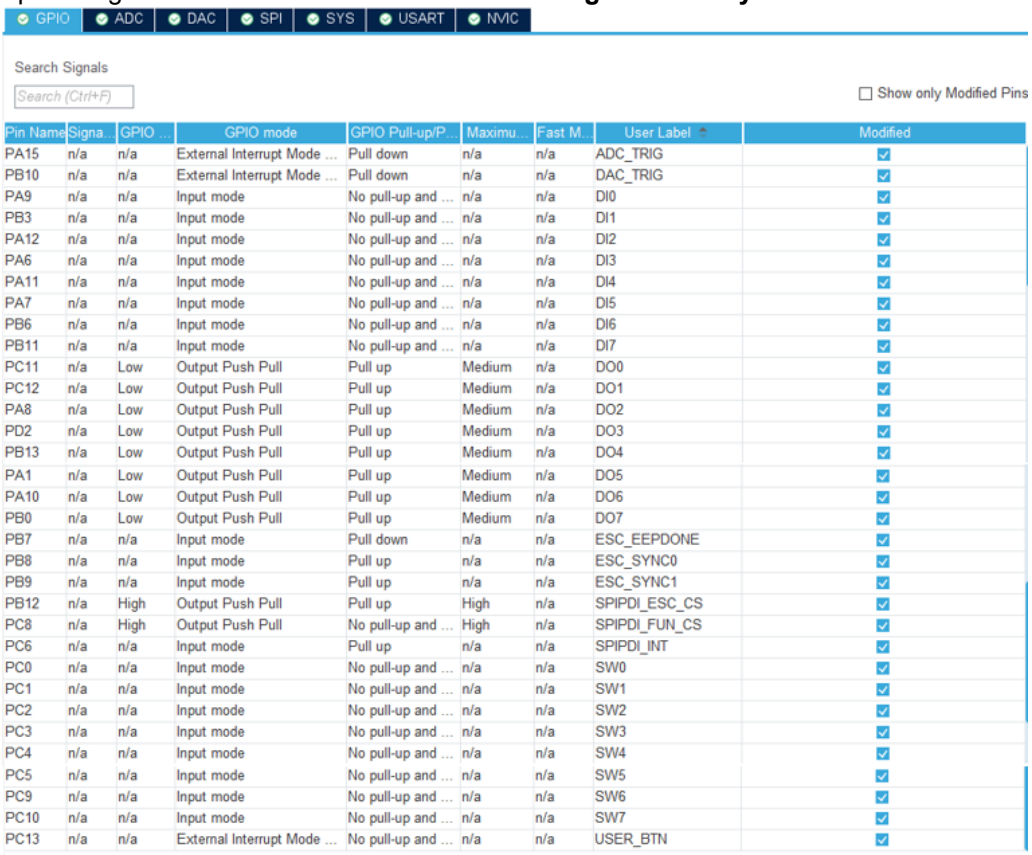


Figure 5-3

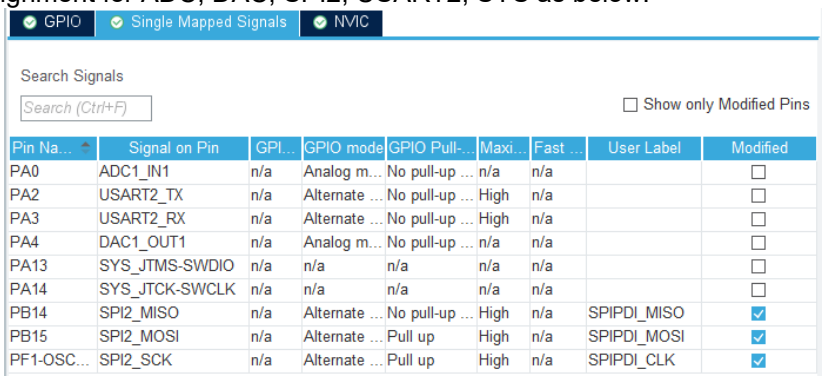
Step 2: Add pin assignment as the below in "**Pinout & Configuration > System Core > GPIO > GPIO**".



Pin Name	Signa...	GPIO ...	GPIO mode	GPIO Pull-up/P...	Maximu...	Fast M...	User Label	Modified
PA15	n/a	n/a	External Interrupt Mode ...	Pull down	n/a	n/a	ADC_TRIG	☑
PB10	n/a	n/a	External Interrupt Mode ...	Pull down	n/a	n/a	DAC_TRIG	☑
PA9	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI0	☑
PB3	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI1	☑
PA12	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI2	☑
PA6	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI3	☑
PA11	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI4	☑
PA7	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI5	☑
PB6	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI6	☑
PB11	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	DI7	☑
PC11	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO0	☑
PC12	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO1	☑
PA8	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO2	☑
PD2	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO3	☑
PB13	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO4	☑
PA1	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO5	☑
PA10	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO6	☑
PB0	n/a	Low	Output Push Pull	Pull up	Medium	n/a	DO7	☑
PB7	n/a	n/a	Input mode	Pull down	n/a	n/a	ESC_EEPDONE	☑
PB8	n/a	n/a	Input mode	Pull up	n/a	n/a	ESC_SYNC0	☑
PB9	n/a	n/a	Input mode	Pull up	n/a	n/a	ESC_SYNC1	☑
PB12	n/a	High	Output Push Pull	Pull up	High	n/a	SPIPDI_ESC_CS	☑
PC8	n/a	High	Output Push Pull	No pull-up and ...	High	n/a	SPIPDI_FUN_CS	☑
PC6	n/a	n/a	Input mode	Pull up	n/a	n/a	SPIPDI_INT	☑
PC0	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW0	☑
PC1	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW1	☑
PC2	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW2	☑
PC3	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW3	☑
PC4	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW4	☑
PC5	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW5	☑
PC9	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW6	☑
PC10	n/a	n/a	Input mode	No pull-up and ...	n/a	n/a	SW7	☑
PC13	n/a	n/a	External Interrupt Mode ...	No pull-up and ...	n/a	n/a	USER_BTN	☑

Figure 5-4

Step 3: Add pin assignment for ADC, DAC, SPI2, USART2, SYS as below.



Pin Na...	Signal on Pin	GPI...	GPIO mode	GPIO Pull-...	Maxi...	Fast ...	User Label	Modified
PA0	ADC1_IN1	n/a	Analog m...	No pull-up ...	n/a	n/a		☐
PA2	USART2_TX	n/a	Alternate ...	No pull-up ...	High	n/a		☐
PA3	USART2_RX	n/a	Alternate ...	No pull-up ...	High	n/a		☐
PA4	DAC1_OUT1	n/a	Analog m...	No pull-up ...	n/a	n/a		☐
PA13	SYS_JTMS-SWDIO	n/a	n/a	n/a	n/a	n/a		☐
PA14	SYS_JTCK-SWCLK	n/a	n/a	n/a	n/a	n/a		☐
PB14	SPI2_MISO	n/a	Alternate ...	No pull-up ...	High	n/a	SPIPDI_MISO	☑
PB15	SPI2_MOSI	n/a	Alternate ...	Pull up	High	n/a	SPIPDI_MOSI	☑
PF1-OSC...	SPI2_SCK	n/a	Alternate ...	Pull up	High	n/a	SPIPDI_CLK	☑

Figure 5-5

Step 4: Enable ADC1 as the settings highlighted in red.

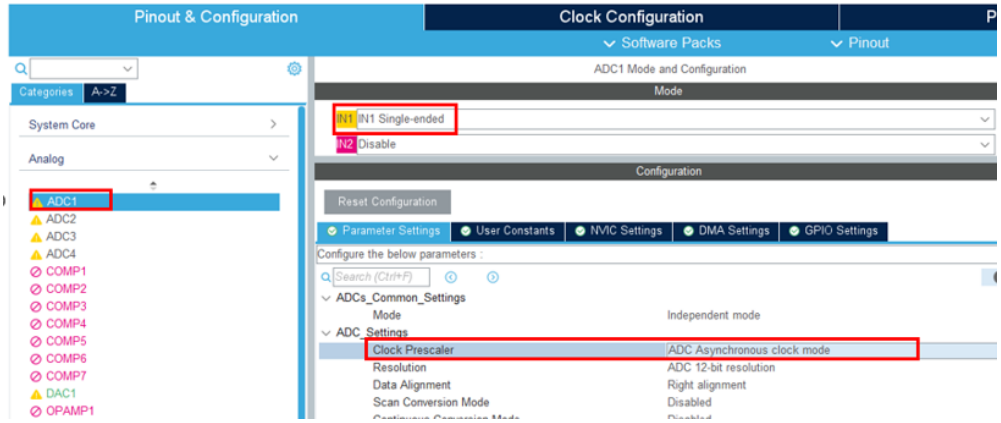


Figure 5-6

Step 5: Enable DAC with the setting highlighted in red.

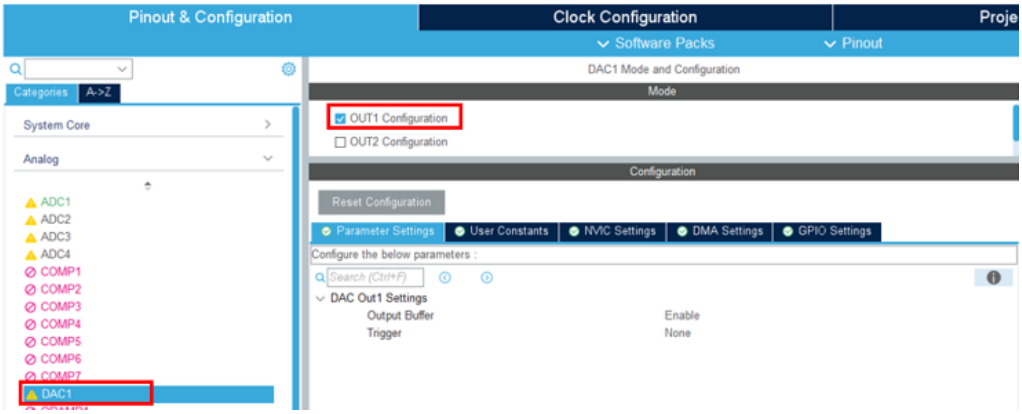


Figure 5-7

Step 6: Activated TIM6 & TIM7 as the setting below.

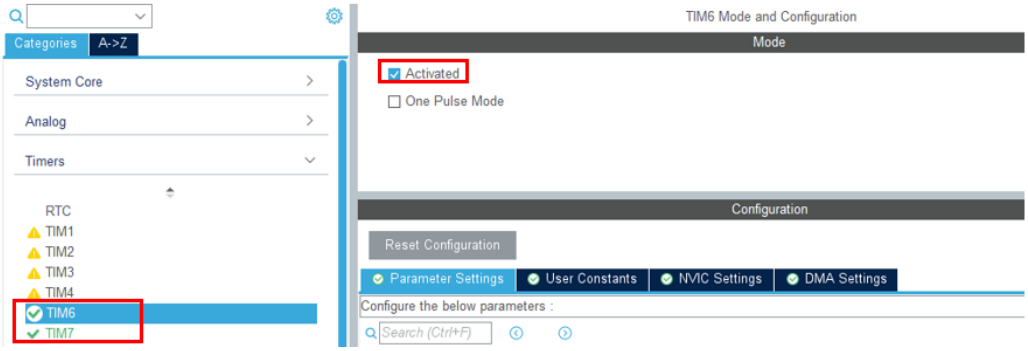


Figure 5-8

Step 7: Enable SPI2 with the setting below.

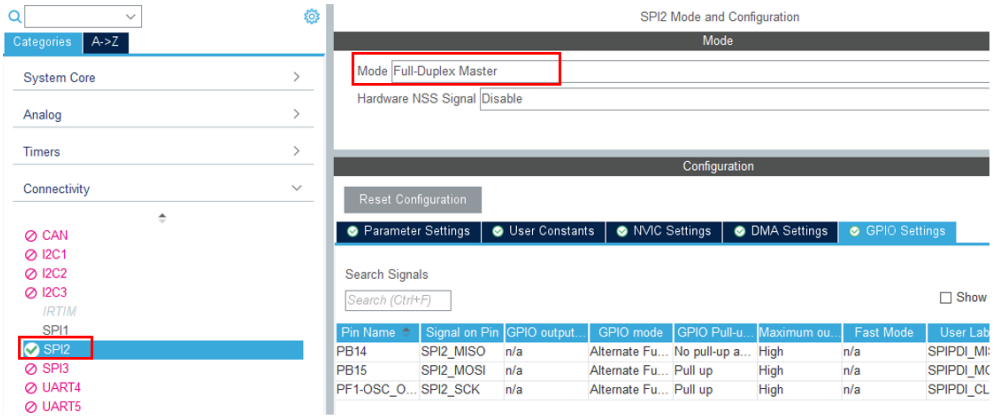


Figure 5-9

Step 8: Enable USART2 for debug console as the setting below.

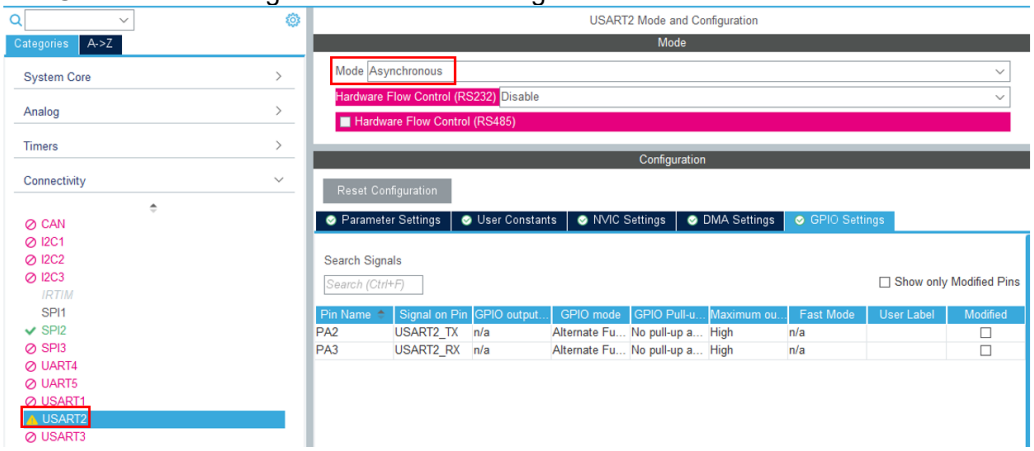


Figure 5-10

Step 9: Modify the NVIC setting as below.

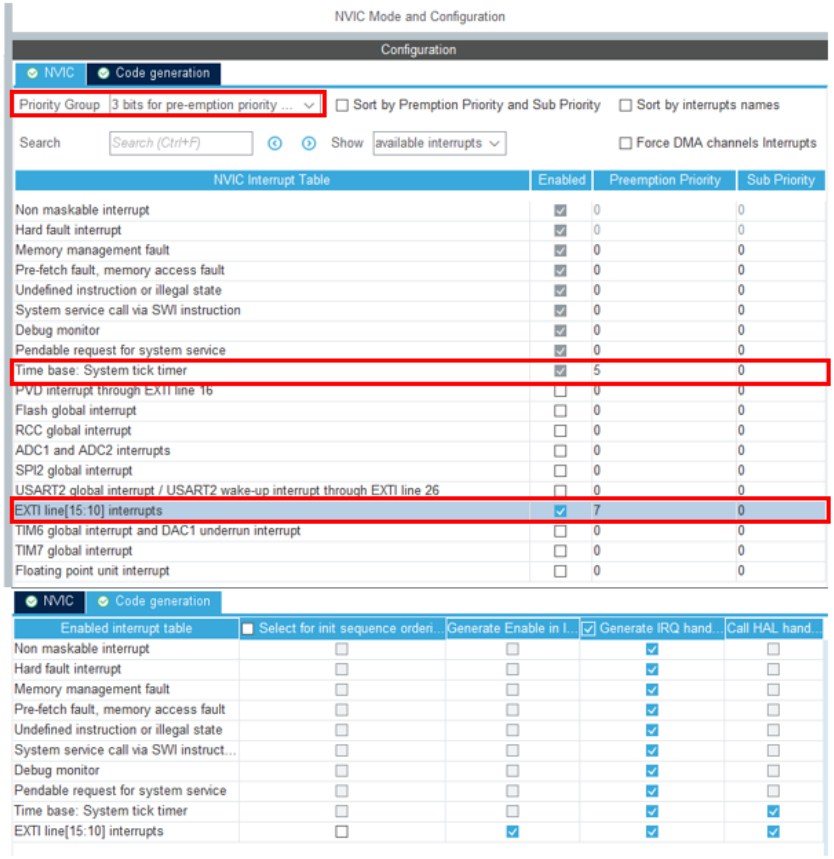


Figure 5-11

Step 10: Adjusting the clock tree in "**Clock Configuration**" page.
Press "**NO**" to ignore auto solver process, then modify the settings as the red highlighted below.

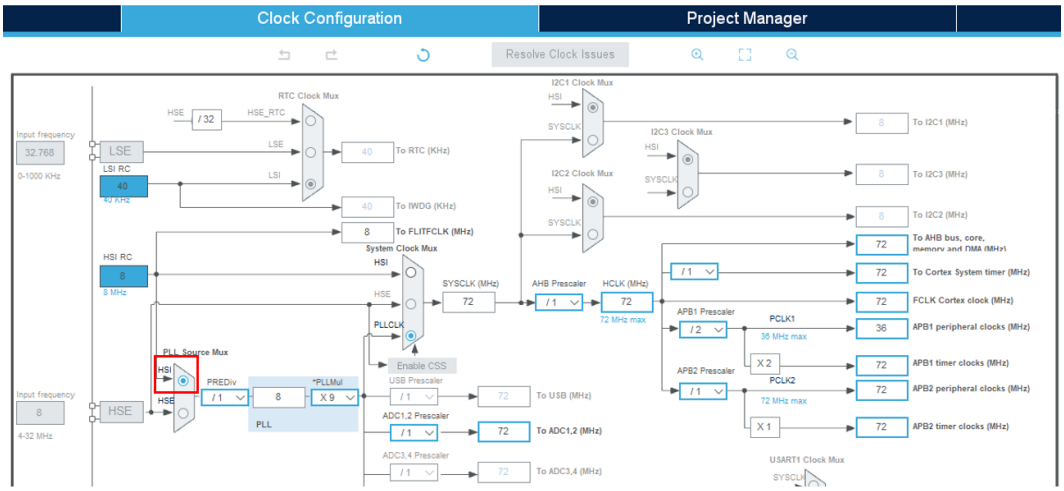


Figure 5-12

5-3 Generate Source Code in STM32CubeMX and MDK-ARM

Step 1: In "**Project Manager > Project**" page, extended the minimum heap size to **0x1000**, and select Firmware Package as "**STM32Cube FW_F3 V1.11.2**".

- **For Keil MDK IDE:**

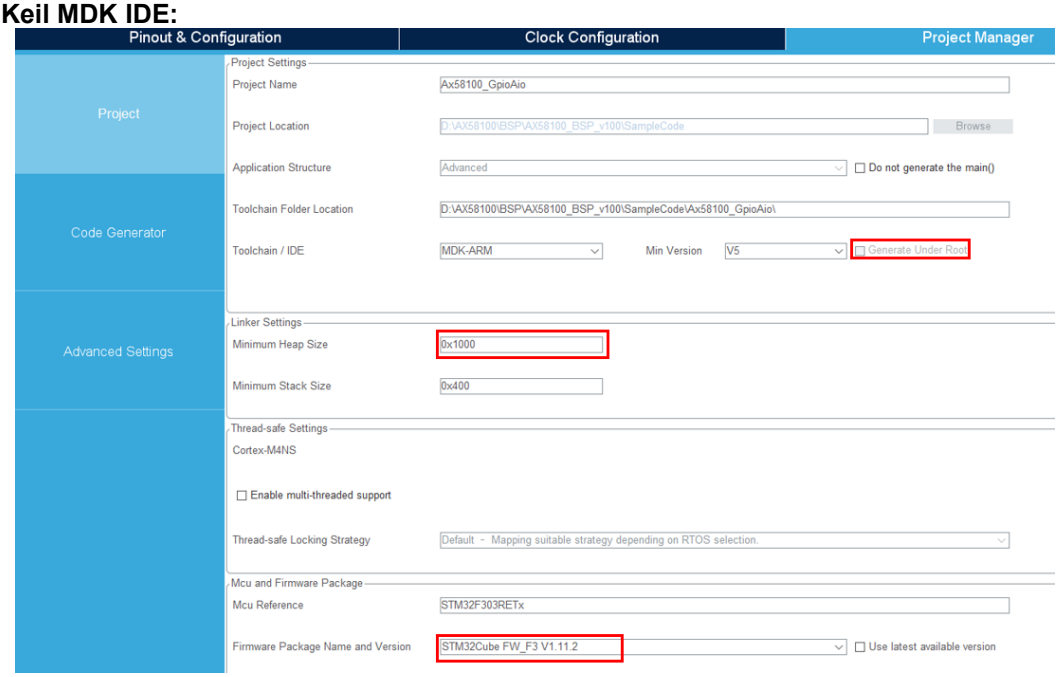


Figure 5-13

- **For STM32CubeIDE:**

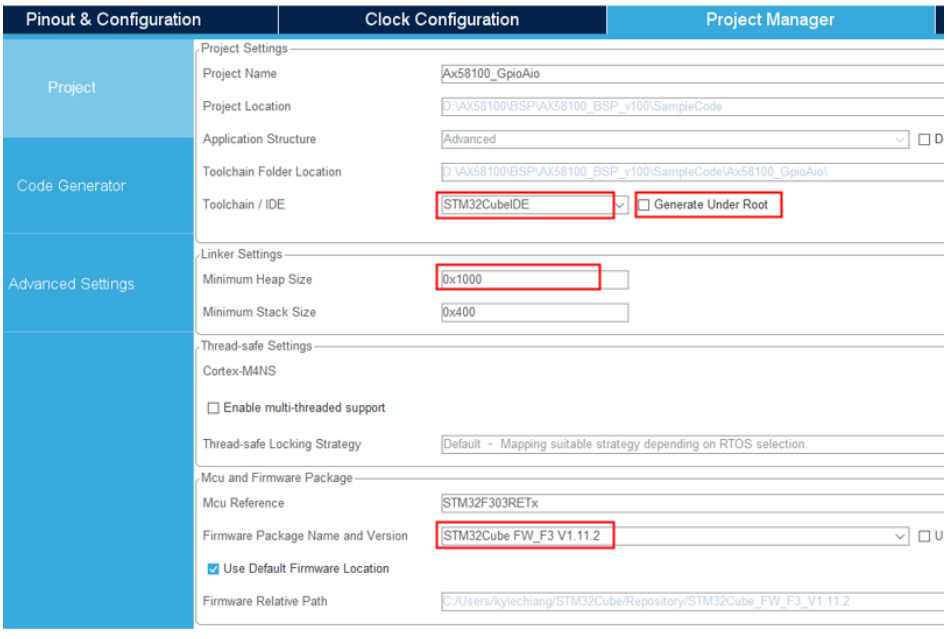


Figure 5-14

Step 2: In "**Project Manager > Code Generator**" page, select Copy all used libraries into the project folder.

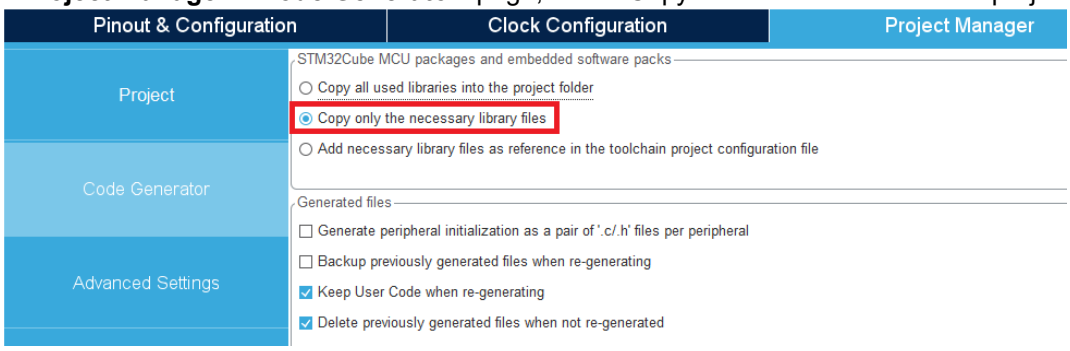


Figure 5-15

Step 3: In "**Project Manager > Advanced Settings**" page, disable initial code generation as below.

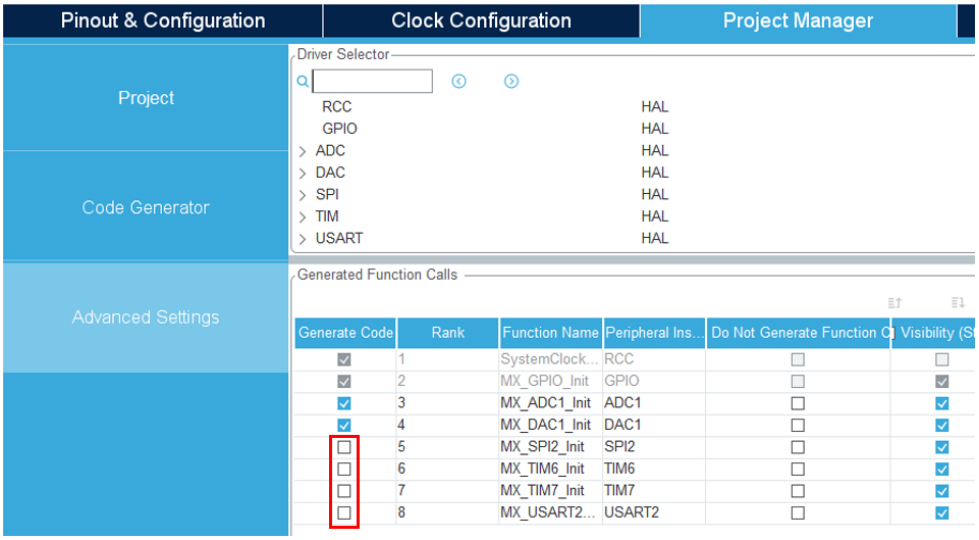


Figure 5-16

Step 4: Generates the source code.

5-4 Source Code Adjustment

Step 1: Add the AX58101 source package.
Copy the folders as below to your STM32CubeMX project folder.
BSP_ROOT\SampleCode\Ax58101_GpioAio\AX58101
BSP_ROOT\SampleCode\Ax58101_GpioAio\Binary
BSP_ROOT\SampleCode\Ax58101_GpioAio\For_SSC_Tool
BSP_ROOT\SampleCode\Ax58101_GpioAio\For_TwinCAT
Copy the folders as below to your root folder.

Step 2: Adjust the Vector Table Address.
As highlighted in the figure below, modify file content in the path:
BSP_ROOT\SampleCode\Ax58101_GpioAio\Core\Src\system_stm32f3xx.c

```
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)8000000) /*!< Default value of the Internal os
                                              This value can be provided and ac
#endif /* HSI_VALUE */

/*!< Uncomment the following line if you need to relocate your vector Table in
     Internal SRAM. */
/* #define VECT_TAB_SRAM */
#define VECT_TAB_OFFSET  0x4000 /*!< Vector Table base offset field.
                                 This value must be a multiple of 0x200. */
/**
  * @}
  */

  ● ● ●

void SystemInit(void)
{
/* FPU settings ------------------------------------------------------------*/
#if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
  SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full Access */
#endif

  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table Relocation in
Internal SRAM */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH *
#endif /* USER_VECT_TAB_ADDRESS */
}
```

Figure 5-17

Step 3: Adjust the Linker setting.

- **For Keil MDK IDE:**

Skip this step.

- **For STM32CubeIDE:**

BSP_ROOT\SampleCode\Ax58101_GpioAio\STM32CubeIDE\STM32F303RETX_FLASH.ld

```
/* Memories definition */
MEMORY
{
  CCMRAM    (xrw)    : ORIGIN = 0x10000000,   LENGTH = 16K
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 64K
  FLASH    (rx)    : ORIGIN = 0x8004000,   LENGTH = 246K
}
```

Figure 5-18

Step 4: Modify source code.
As highlighted in the figure below, modify file content in the path:
BSP_ROOT\SampleCode\Ax58101_GpioAio\Core\Src\main.c

```
/* Private includes ----------------------------
/* USER CODE BEGIN Includes */
#include "AX58101_Hw.h"
#include "applInterface.h"
#include "ax58101.h"
#include "AX58101_GPIO_8Bit_AIO_16Bit.h"
#include "bootmode.h"
#include "ax58101_FoeAppl.h"

#include "ecat_def.h"
#include "ax58101_utils.h"
/* USER CODE END Includes */

    ●       ●       ●

/* Private macro -------------------------------
/* USER CODE BEGIN PM */
#define DO_LEVEL_REVERSE              0

#define ADC_STATE_INIT                0
#define ADC_STATE_WAIT_CONVERSION     1
#define ADC_STATE_CONVERSION_FINISH   2
#define ADC_STATE_WAIT_NEXT           3

#define ADC_NEXT_TIMEOUT              1//ms
/* USER CODE END PM */

    ●       ●       ●

/* USER CODE BEGIN PV */
u8 virtualAdioEnable;
/* USER CODE END PV */
```

Figure 5-19

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_ADC1_Init();
  MX_DAC1_Init();
  /* USER CODE BEGIN 2 */
#if (AX58101_DEBUG_ENABLE)
  AX_UART_Init();
  CONSOLE_Init();
  DBG_PRINT("%s Firmware v%s\r\n", DEVICE_NAME, DEVICE_SW_VERSION);
#endif //#if (AX58101_DEBUG_ENABLE)

#if (SSC_STACK_ENABLE)
  if (HW_Init())
  {
      HW_Release();
  }
  else
  {
      MainInit();
      GpioAio_Init();
      AX58101_FoeInit();
  }
#endif //#if (SSC_STACK_ENABLE)

    ADIO_Init();
  /* USER CODE END 2 */
```

Figure 5-20

```
/* USER CODE BEGIN 4 */
/*
 * -----------------------------------------------------------------------------
 * Function Name: void GetDigitalInput(void)
 * Purpose:
 * Params:  None
 * Returns: None
 * Note:
 * -----------------------------------------------------------------------------
 */
u8 GetDigitalInput(void)
{
    u8  val = 0;

    /* Handle input */
    val |= HAL_GPIO_ReadPin(DI0_GPIO_Port, DI0_Pin) ? 0x01 : 0x00;
    val |= HAL_GPIO_ReadPin(DI1_GPIO_Port, DI1_Pin) ? 0x02 : 0x00;
    val |= HAL_GPIO_ReadPin(DI2_GPIO_Port, DI2_Pin) ? 0x04 : 0x00;
    val |= HAL_GPIO_ReadPin(DI3_GPIO_Port, DI3_Pin) ? 0x08 : 0x00;
    val |= HAL_GPIO_ReadPin(DI4_GPIO_Port, DI4_Pin) ? 0x10 : 0x00;
    val |= HAL_GPIO_ReadPin(DI5_GPIO_Port, DI5_Pin) ? 0x20 : 0x00;
    val |= HAL_GPIO_ReadPin(DI6_GPIO_Port, DI6_Pin) ? 0x40 : 0x00;
    val |= HAL_GPIO_ReadPin(DI7_GPIO_Port, DI7_Pin) ? 0x80 : 0x00;
    return val;

} /* End of GetDigitalInput() */

/*
 * -----------------------------------------------------------------------------
 * Function Name: void DigitalOutputHandler(void)
 * Purpose:
 * Params:  None
 * Returns: None
 * Note:
 * -----------------------------------------------------------------------------
 */
void SetDigitalOutput(u8 val)
{

#if (DO_LEVEL_REVERSE)
    /* Reverse the output level */
    val = ~val;
#endif //#if (DO_LEVEL_REVERSE)

    /* Handle output */
    HAL_GPIO_WritePin(DO0_GPIO_Port, DO0_Pin, ((val & BIT_0) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO1_GPIO_Port, DO1_Pin, ((val & BIT_1) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO2_GPIO_Port, DO2_Pin, ((val & BIT_2) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO3_GPIO_Port, DO3_Pin, ((val & BIT_3) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO4_GPIO_Port, DO4_Pin, ((val & BIT_4) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO5_GPIO_Port, DO5_Pin, ((val & BIT_5) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO6_GPIO_Port, DO6_Pin, ((val & BIT_6) ? GPIO_PIN_SET :
GPIO_PIN_RESET));
    HAL_GPIO_WritePin(DO7_GPIO_Port, DO7_Pin, ((val & BIT_7) ? GPIO_PIN_SET :
GPIO_PIN_RESET));

} /* End of SetDigitalOutput() */
```

Figure 5-21

```
/*
 * ------------------------------------------------------------------------
 * Function Name: u16 GetAnalogInput(void)
 * Purpose:
 * Params:  None
 * Returns: None
 * Note:
 * ------------------------------------------------------------------------
 */
u16 GetAnalogInput(void)
{
    static u32 adcState = ADC_STATE_INIT, tickstart=0;
    u32 tmp16;
    static u16 val=0;

    if (adcState == ADC_STATE_INIT)
    {
        /* Start conversion */
        if (HAL_ADC_Start(&hadc1) != HAL_OK)
        {
            HAL_ADC_Stop(&hadc1);
            return val;
        }
        adcState = ADC_STATE_WAIT_CONVERSION;
    }
    else if (adcState == ADC_STATE_WAIT_CONVERSION)
    {
        /* Ignore handle, just wait conversion complete */
        if (hadc1.Instance->ISR & ADC_FLAG_EOC)
        {
            adcState = ADC_STATE_CONVERSION_FINISH;
        }
    }
    else if (adcState == ADC_STATE_CONVERSION_FINISH)
    {
        /* Get the conversion result of the sample module 0 */
        val = (u16)HAL_ADC_GetValue(&hadc1);
        tickstart = HAL_GetTick();
        adcState = ADC_STATE_WAIT_NEXT;
    }
    else if (adcState == ADC_STATE_WAIT_NEXT)
    {
        /* Wait a while for next conversion */
        tmp16 = HAL_GetTick();
        if (tmp16 >= tickstart)
        {
            if ((tmp16 - tickstart) >= ADC_NEXT_TIMEOUT)
            {
                adcState = ADC_STATE_INIT;
            }
        }
        else
        {
            tickstart = tmp16;
        }
    }

    return val;
} /* End of GetAnalogInput() */
```

Figure 5-22

```
/*
 * ------------------------------------------------------------------------
 * Function Name: void SetAnalogOutput(void)
 * Purpose:
 * Params:  None
 * Returns: None
 * Note:
 * ------------------------------------------------------------------------
 */
void SetAnalogOutput(u16 val)
{
    u32 tmp16, tickstart=0;

    tmp16 = val & 0x0FFF;

    /* Start A/D conversion */
    HAL_DAC_SetValue(&hdac1, DAC_CHANNEL_1, DAC_ALIGN_12B_R, tmp16);
    HAL_DAC_Start(&hdac1, DAC_CHANNEL_1);

    /* Wait convertion done */
    tickstart = HAL_GetTick();
    while ((HAL_GetTick() - tickstart) < 3)
    {
        if (0 == READ_BIT(hdac1.Instance->SWTRIGR, DAC_SWTRIGR_SWTRIG1))
        {
            break;
        }
    }
} /* End of SetAnalogOutput() */

/*
 * ------------------------------------------------------------------------
 * Function Name: u8 GetSwitchID(void)
 * Purpose: Get current digit of "Rotary Dip Switch" of AX58400-EXB-ADIO
 * Params:  None
 * Returns: Current digit
 * Note:
 * ------------------------------------------------------------------------
 */
u8 GetSwitchID(void)
{
    u8  currentSwitchID = 0;

    currentSwitchID |= HAL_GPIO_ReadPin(SW0_GPIO_Port, SW0_Pin)  ? BIT_0 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW1_GPIO_Port, SW1_Pin)  ? BIT_1 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW2_GPIO_Port, SW2_Pin)  ? BIT_2 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW3_GPIO_Port, SW3_Pin)  ? BIT_3 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW4_GPIO_Port, SW4_Pin)  ? BIT_4 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW5_GPIO_Port, SW5_Pin)  ? BIT_5 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW6_GPIO_Port, SW6_Pin)  ? BIT_6 : 0;
    currentSwitchID |= HAL_GPIO_ReadPin(SW7_GPIO_Port, SW7_Pin)  ? BIT_7 : 0;

    return (~currentSwitchID);
} /* End of GetSwitchID() */
```

Figure 5-23

```
/*
 * ---------------------------------------------------------
 * Function Name: HAL_GPIO_EXTI_Callback
 * Purpose:
 * Params:
 * Returns:
 * Note:
 * ---------------------------------------------------------
 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == DAC_TRIG_Pin)
    {
    /* Detected DAC trigger signal */
        MCU_GPIO_EXTI_CLEAR_FLAG(DAC_TRIG_Pin);
    }

    if (GPIO_Pin == ADC_TRIG_Pin)
    {
        /* Detected ADC trigger signal */
        MCU_GPIO_EXTI_CLEAR_FLAG(ADC_TRIG_Pin);
    }

    if (GPIO_Pin == USER_BTN_Pin)
    {
        /* Detected user button signal */
        MCU_GPIO_EXTI_CLEAR_FLAG(USER_BTN_Pin);
    }
} /* End of HAL_GPIO_EXTI_Callback() */

/* USER CODE END 4 */
...
```

Figure 5-24

As highlighted in the figure below, modify file content in the path:
BSP_ROOT\SampleCode\Ax58101_GpioAio\Core\Src\stm32f3xx_it.c

```
/* Includes ------------------------------------------
#include "main.h"
#include "stm32f3xx_it.h"
/* Private includes ----------------------------------
/* USER CODE BEGIN Includes */
#include "ax58101_utils.h"
/* USER CODE END Includes */

/* Private typedef -----------------------------------
/* USER CODE BEGIN TD */

    ●       ●       ●

  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
  /* USER CODE BEGIN SysTick_IRQn 1 */
#if (AX58101_DEBUG_ENABLE)
    CONSOLE_TimeTick();
#endif //#if (AX58101_DEBUG_ENABLE)
  /* USER CODE END SysTick_IRQn 1 */
}
```

Figure 5-25

5-5 Generate EtherCAT SubDevice Stack Code

Due to the ETG SubDevice Stack Code (SSC) license limitation, the user must re-generate the source code by the SSC tool locally, about how to get the SSC tool and how to generate targeted source code, please refer to the user guide as below.

BSP_ROOT\Document\AX58x00_SSC_Tool_Configuration_Import_UserGuide_vXXX.pdf

Assume the SSC tool has been installed, the related files needed are as below.

SSC tool project file:
BSP_ROOT \SampleCode\Ax58101_GpioAio\For_SSC_Tool\AX58101_GPIO_8Bit_AIO_16Bit.esp

Configuration file for custom import:
BSP_ROOT\SampleCode\Ax58101_GpioAio\For_SSC_Tool\Import\Configuration\AsixAx58101GpioAio Configurations.xml

Excel file for application import:
BSP_ROOT\SampleCode\Ax58101_GpioAio\For_SSC_Tool\Import\Configuration\files\AX58101_GPIO_ 8Bit_AIO_16Bit.xlsx

5-6 IDE Setting Adjustment

- **For Keil MDK IDE:**

Double clicks the file as below to open project.
BSP_ROOT\SampleCode\Ax58101_GpioAio\MDK-ARM\Ax58101_GpioAio.uvproj

Step 1: Add source code both on dual cores.
Clock on "**Project > Manage > Project Items…**" to open "**Manage Project Items**" dialog, then add groups and files as the figure below.

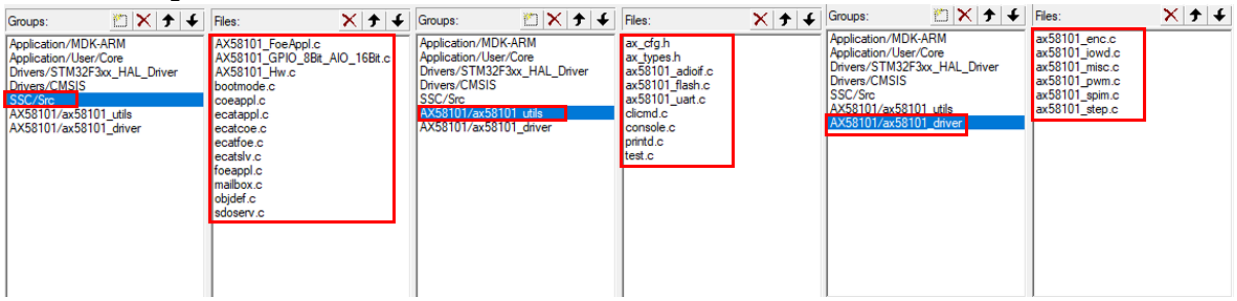


Figure 5-26

Step 2: Add include path both on dual cores.
Click on "**Project > Options for Target 'xxxx'… > C/C++ > Include Path**" to open "**Folder Setup**" dialog, then add paths as the figure below.

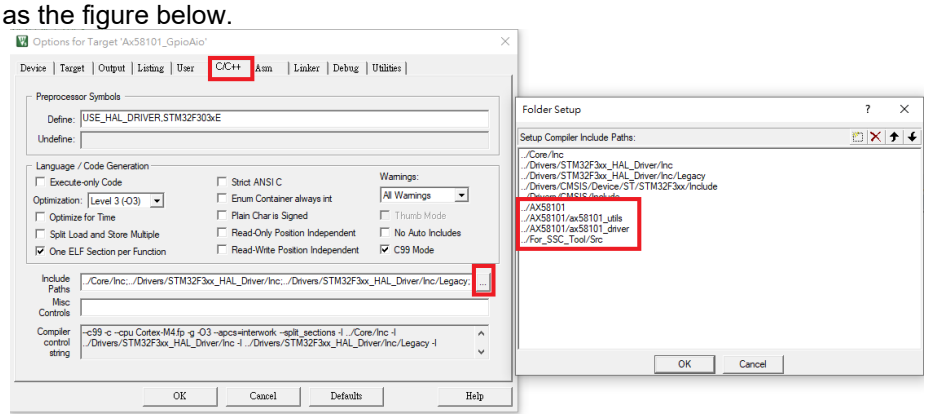


Figure 5-27

Step 3: Setup Utilities.
Click on "**Project > Options for Target 'xxxx'… > Utilities > Settings**", then select "**Flash Download**" page to add "**Programming Algorithm**" as the value below.

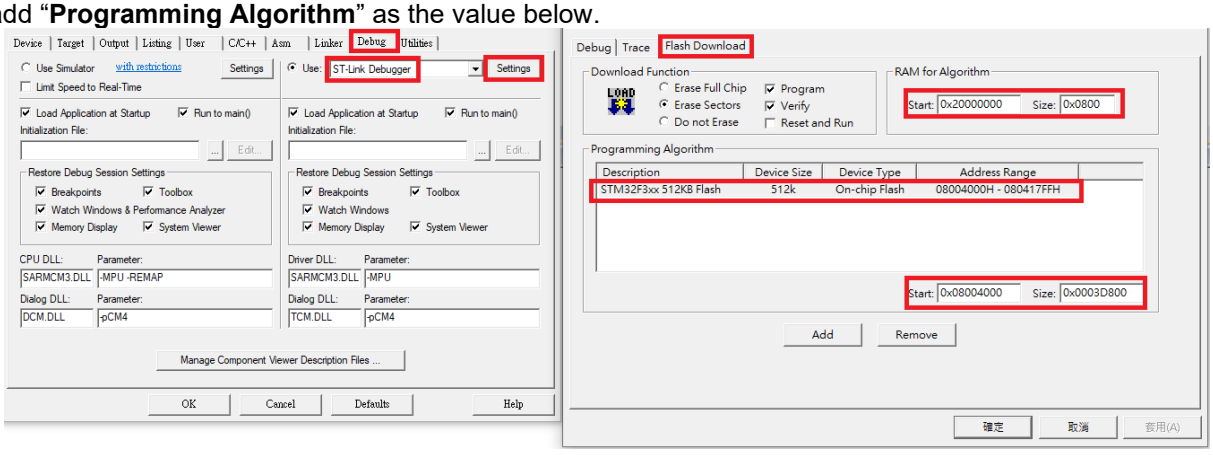


Figure 5-28

Start = 0x08004000, Size = 0x0003D800

Step 4: Change "**Linker**" settings.
Find the setting of "**Project > Options for Target 'xxxx'… > Linker**", then checked the "**Use Memory Layout from Target Dialog**".

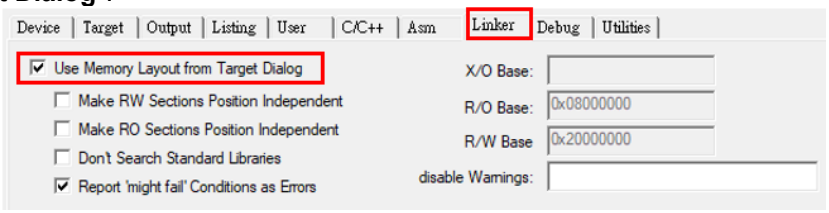


Figure 5-29

Step 5: Add post build actions.
Find the setting of "**Project > Options for Target 'xxxx'… > User > After Build/Rebuild**", then add actions as the value below.

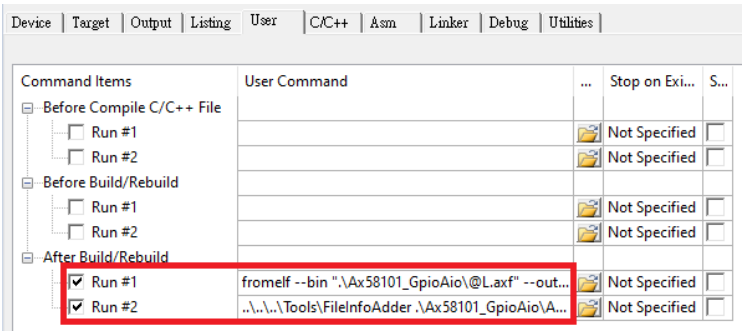


Figure 5-30

Run #1: **fromelf --bin ".\Ax58101_GpioAio\@L.axf" --output ".\Ax58101_GpioAio\@L.bin"**
Run #2:
..\..\..\Tools\FileInfoAdder .\Ax58101_GpioAio\Ax58101_GpioAio.bin ..\For_TwinCAT\FoE\AX58101_GpioAio.efw

Step 6: Modify "**Target Options**".
Find the setting of "**Project > Options for Target 'xxxx'… > Target > Read/Write Memory Areas**", then modify it as the value below.

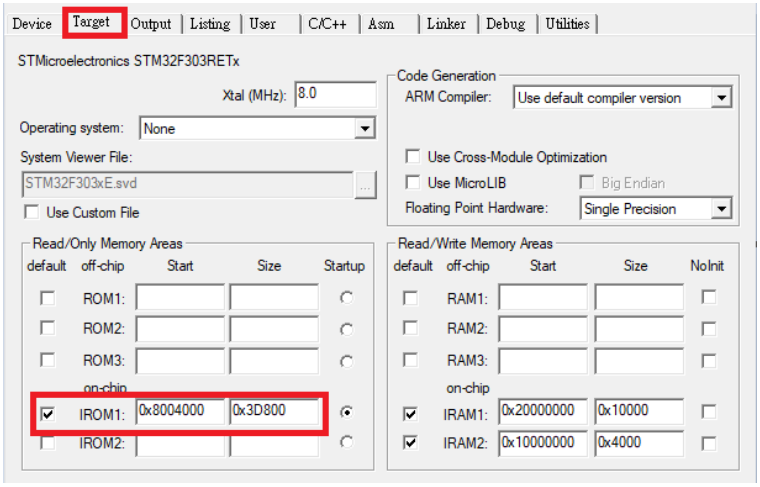


Figure 5-31

IROM1 Start = 0x8004000, Size = 0x3D800

Now, entire building procedure has done for Keil MDK IDE.

- **For STM32CubeIDE:**

Double clicks the file as below to open project.
BSP_ROOT\SampleCode\Ax58101_GpioAio\STM32CubeIDE\.project

Step 1: Add source code.
On "**Project Explorer**", add virtual folders and linking the source files.

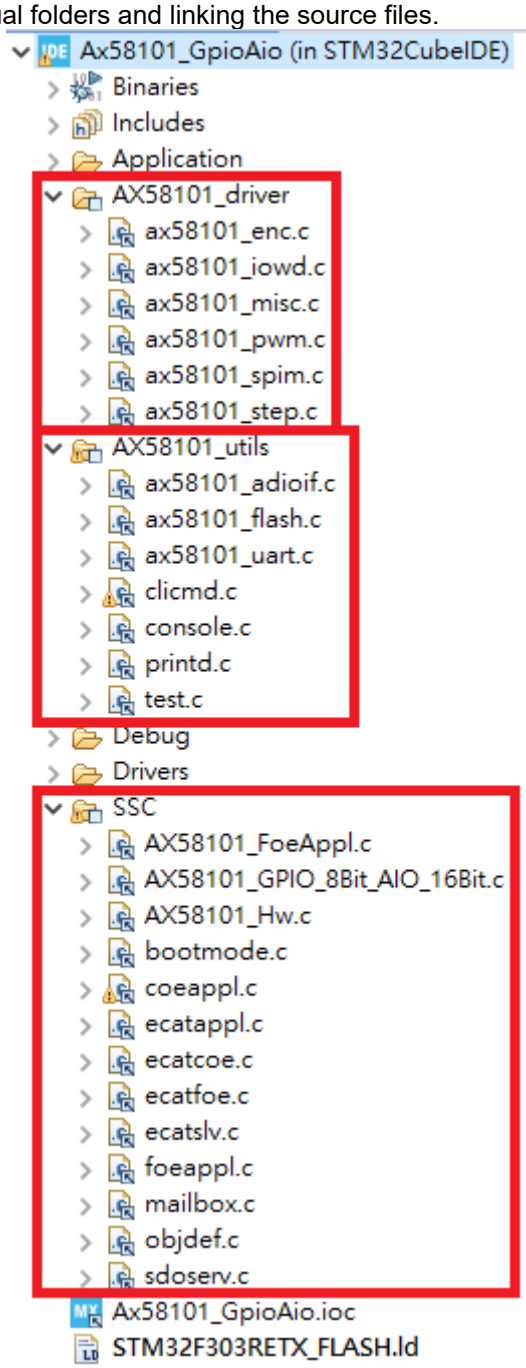


Figure 5-32

- **Add Virtual Folder:** Right click on target project, enter new folder name and select "**New > Folder > Folder is not located in the file system (Virtual Folder)**".

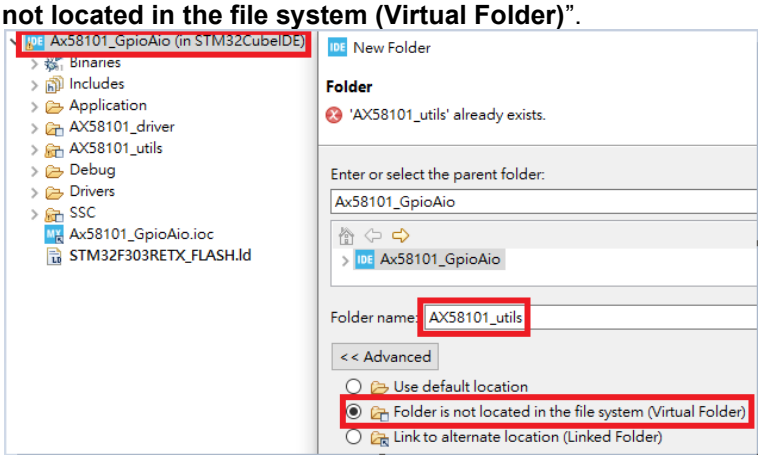


Figure 5-33

- **Add File Link:** Select and drag the files onto the created virtual folder.

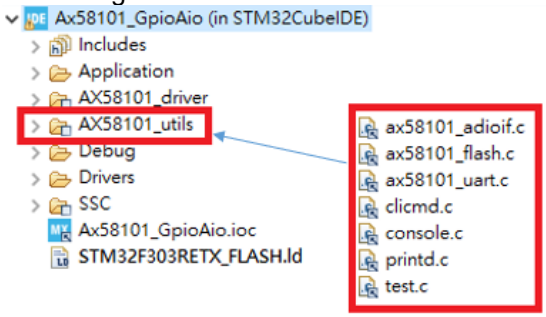


Figure 5-34

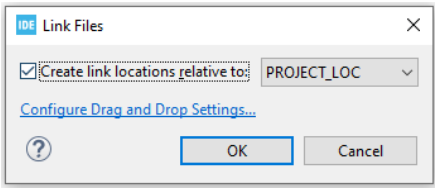


Figure 5-35

Step 2: Enable binary & Hex output files conversion on dual cores.
Select target project, then find setting on "**Project > Properties > C/C++ Build > Settings > Tool Settings > MCU Post build outputs**", then give the setting as highlighted as below.

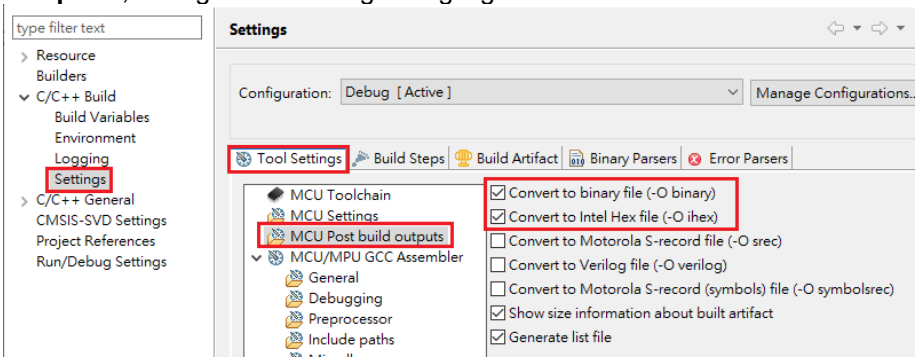


Figure 5-36

Step 3: Add include path.
Select target project, then find setting on "**Project > Properties > C/C++ Build > Settings > Tool Settings > MCU/MCP GCC Compiler > Include paths**", then add paths as the figure below.

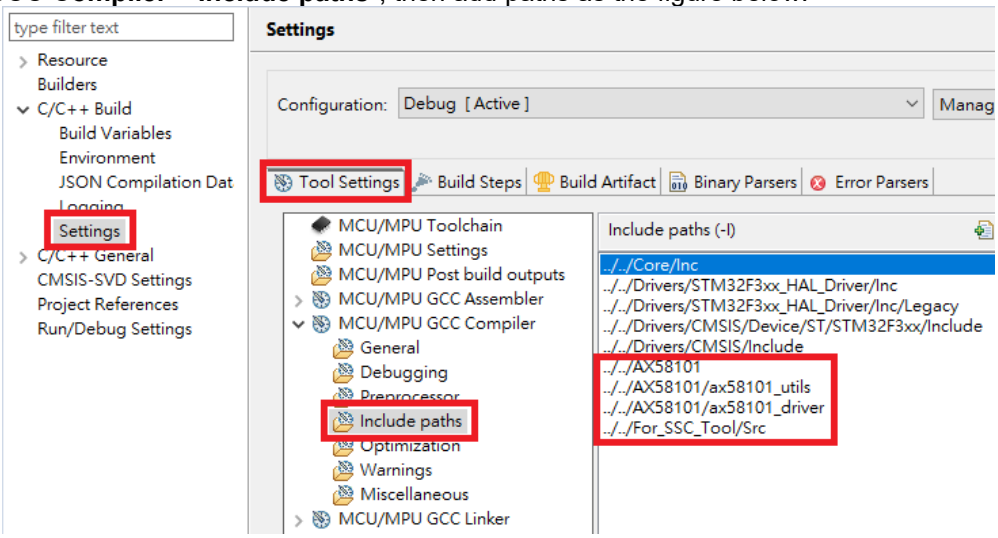


Figure 5-37

Step 4: Add post build actions.
Find the setting of "**Project > Properties > C/C++ Build > Settings > Build Steps > Post-build steps**", then add actions as the value below.

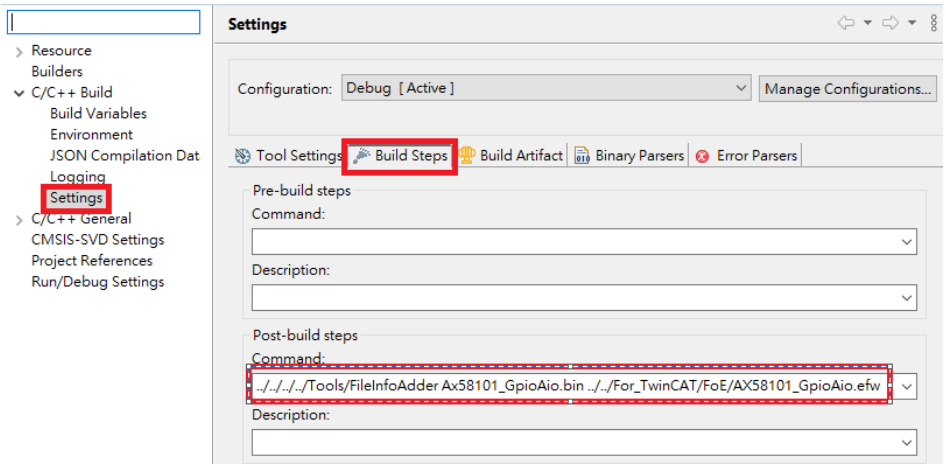


Figure 5-38

Post-build steps Command:
../../../../Tools/FileInfoAdder Ax58101_GpioAio.bin ../../For_TwinCAT/FoE/AX58101_GpioAio.efw

Now, entire building procedure has done for STM32CubeIDE.

So far, the entire project source has been ready for building firmware binary.

6. Build Up and Download the Firmware

Here will introduce how to build up full firmware source in your local IDE.

6-1 Build up Firmware Binary

- **For Keil MDK IDE:**

Step 1: Double clicks on “Ax58101_GpioAio.uvprojx” file in the folder below to open project.
BSP_ROOT\SampleCode\Ax58101_GpioAio\MDK-ARM.

Step 2: Execute “**Rebuild all target files**” under “**Menu Bar > Project**”.

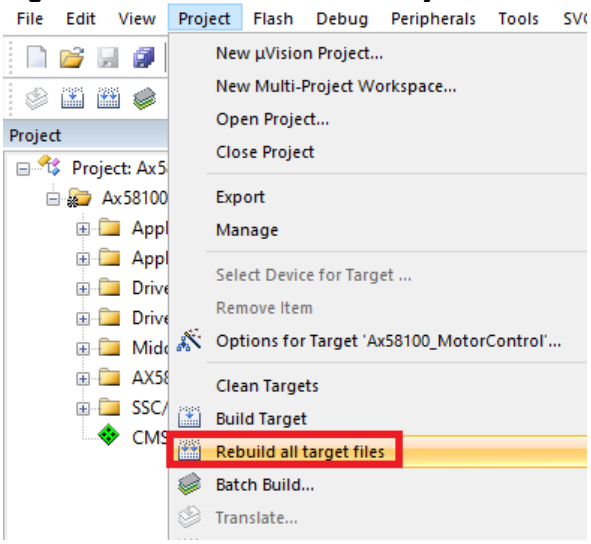


Figure 6-1

- **For STM32CubeIDE:**

Step 1: Double clicks the file as below to open project.
BSP_ROOT\SampleCode\Ax58101_GpioAio\STM32CubeIDE\.project

Step 2: Select the project node and execute “**Build Project**” under “**Menu Bar > Project**”.

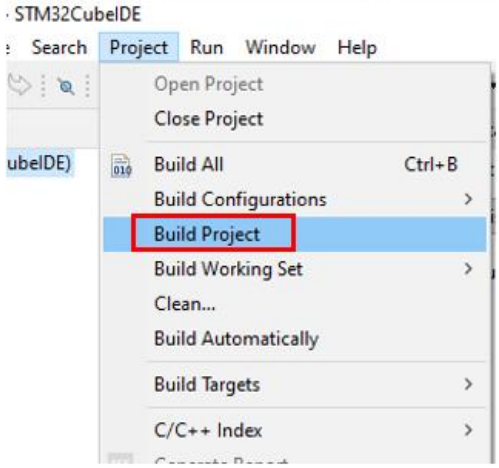


Figure 6-2

6-2 Download Bootloader Firmware

Step 1: Open STM32CubeProgrammer utility and connect to the MCU board.

Step 2: Select the "**Erasing & Programming**" page, then click "**Browse**" button to select bootloader firmware binary file in the path below.
BSP_ROOT\SampleCode\ Ax58101_Bldr\Binary\Ax58101_Bldr_vX.X.X

Please ensure the "**Start address**" is 0x08000000.

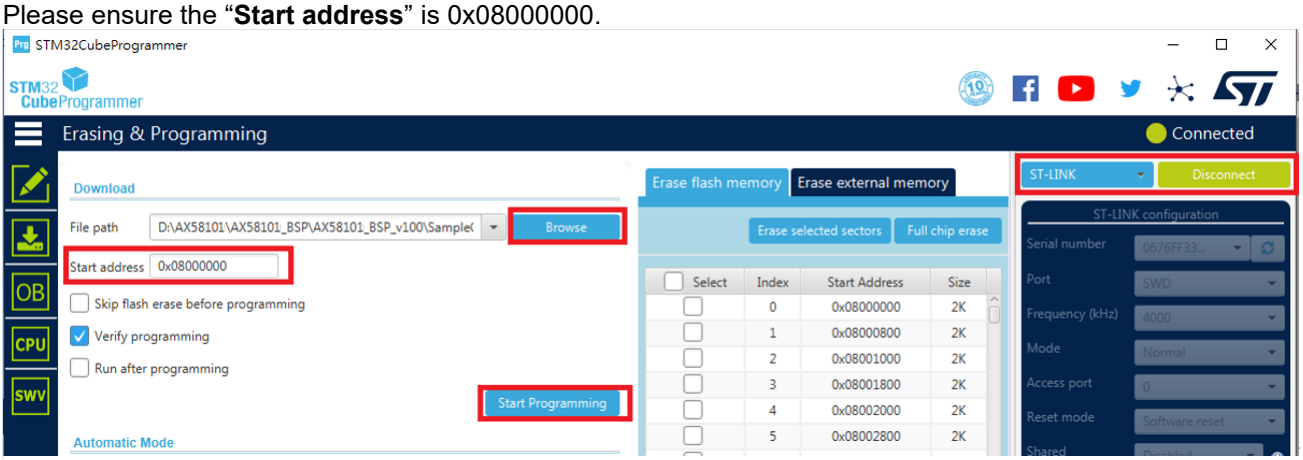


Figure 6-3

Step 3: Press "**Start Programming**" button to download bootloader firmware.

6-3 Download GPIO / AIO Firmware

- **For Keil MDK IDE:**

Step 1: Click "**Menu bar > Flash > Download**" to download firmware into MCU.

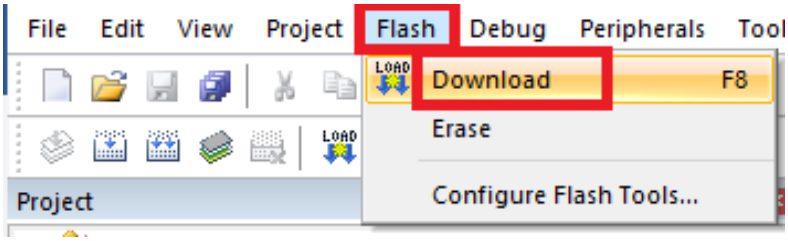


Figure 6-4

- **For STM32CubeIDE:**

Step 1: Click "**Menu bar > Run > Run As > STM32 C/C++ Application**" to download firmware into MCU.

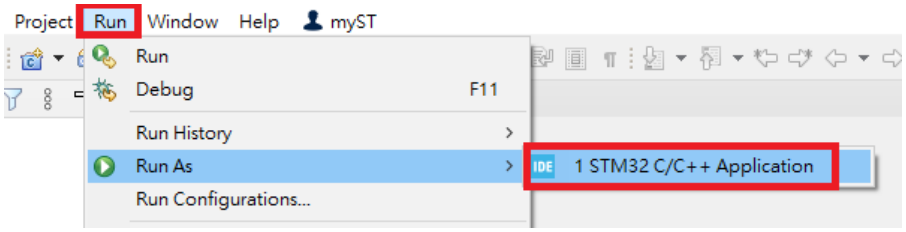


Figure 6-5

- **Using STM32CubeProgrammer Utility:**

Step 1: Similarly, please ensure the utility has connected to target MCU board.

Step 2: Select the "**Erasing & Programming**" page, then click the "**Browse**" button to select GPIO / AIO firmware binary file from one of the following paths.

- **For Keil MDK IDE:**
  BSP_ROOT\SampleCode\Ax58101_GpioAio\MDK-ARM\Ax58101_GpioAio\
  Ax58101_GpioAio.bin
- **For STM32CubeIDE:**
  BSP_ROOT\SampleCode\Ax58101_GpioAio\STM32CubeIDE\Debug\Ax58101_GpioAio.bin
- **For Binary folder:**
  BSP_ROOT\\SampleCode\Ax58101_GpioAio\Binary\Ax58101_GpioAio.bin

Please ensure the "**Start address**" is 0x08004000.

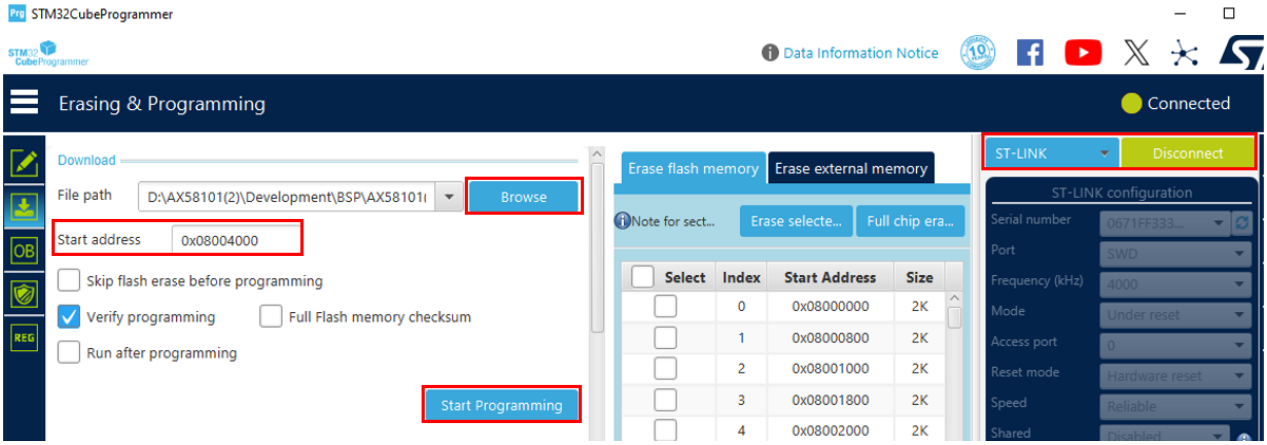


Figure 6-6

Step 3: Press "**Start Programming**" button to download GPIO / AIO firmware.

6-4 Check Firmware Status

After upgrading the firmware, you can verify that the firmware upgrade has been successfully programmed. Please refer to the following steps.

Users need to use "**Tera Term**" console tool or another console connection tool.
The default setting of UART port as following:

Baud rate: **115200**
Data: **8 bit**
Parity: **none**
Stop: **1 bit**
Flow control: **none**

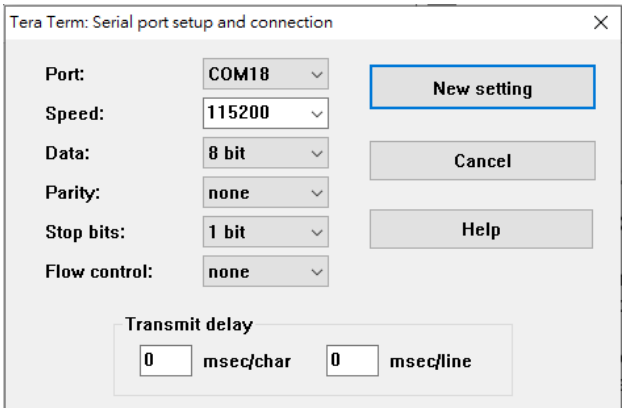


Figure 6-7

Open the COM port. If no characters appear in COM port, you can press "**Reset**" button to confirm again. Below screen shows the firmware of the EtherCAT SubDevice Stack working successfully.

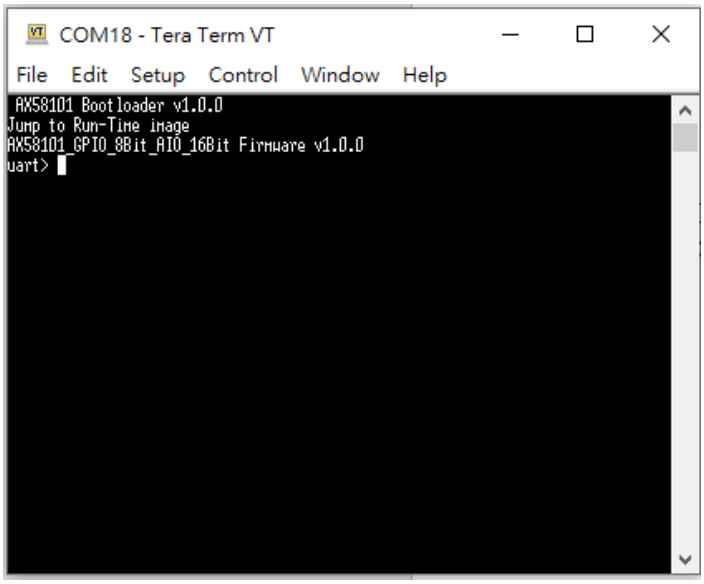


Figure 6-8

7. Basic Operation of TwinCAT

Please refer to the file “**AX58x00_TwinCAT_UserGuide**” Chapter 3.

8. PLC Application

Please refer to the file “**AX58x00_TwinCAT_UserGuide**” Chapter 5.

9. Flash Memory Allocation and FoE Upgrading

To support “File Access over EtherCAT” (FoE) feature, this reference design has the flash memory allocation on the STM32F303RE as below.

Flash area	Flash memory addresses	Size (bytes)	Name
Main memory	0x0800 0000 - 0x0800 07FF	2 K	Page 0
	0x0800 0800 - 0x0800 0FFF	2 K	Page 1
	0x0800 1000 - 0x0800 17FF	2 K	Page 2
	0x0800 1800 - 0x0800 1FFF	2 K	Page 3
	0x0800 3800 - 0x0800 3FFF		Page 7
	0x0800 4000 - 0x0800 47FF 0x0804 1000 - 0x0804 17FF		Page 8 Page 130
	0x0804 1800 - 0x0804 1FFF 0x0807 F000 - 0x0807 F7FF		Page 131 Page 254
	0x0807 F800 - 0x0807 FFFF	2 K	Page 255
Information block	0x1FFF D800 - 0x1FFF F7FF	8 K	System memory
	0x1FFF F800 - 0x1FFF F80F	16	Option bytes

Bootloader area (Page 0~7)

Runtime image area (Page 8~130)

Flash buffer area (Page 131~254)

Figure 9-1

- Bootloader area (Address=0x08000000, Size=16Kbytes):
  The Bootloader firmware must be pre-burn in this area, it’ll responsible for coping the received new firmware from “Flash buffer area” into “Runtime image area”, then executes the new firmware.

- Runtime image area (Address=0x08004000, Size=246Kbytes):
  Used to run application firmware (as the GPIO / AIO firmware in here), it is also responsible for store the received .efw file into “Flash buffer area” via FoE data transfer.

- Flash buffer area (Address=0x08041800, Size=247Kbytes):
  Used to store entire .efw file.

The .efw file is dedicate use for FoE transfer, it’s a firmware binary file with 32-bytes header pre-fix. Every time the project build done, the .efw file will auto be generated in the path below by “**FileInfoAdder.exe**” utility.

BSP_ROOT\SampleCode\Ax58101_GpioAio\For_TwinCAT\FoE\AX58101_GpioAio.efw

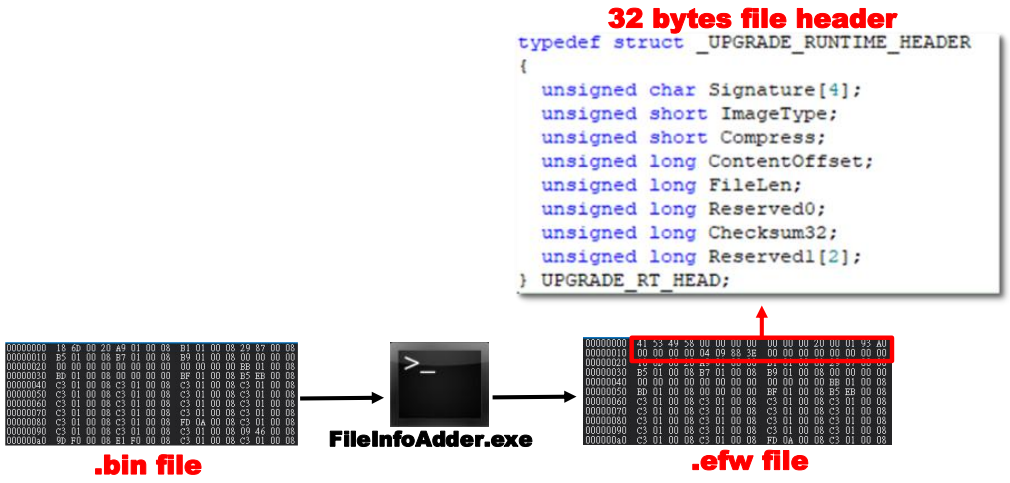


Figure 9-2

9-1 Firmware Upgrade

Here assume the EtherCAT network has established and the device has been under OP mode, then please follow the steps below to do firmware upgrade via FoE.

Step 1: Right click on the online device that you want to upgrade firmware, then select "**Firmware Update**".

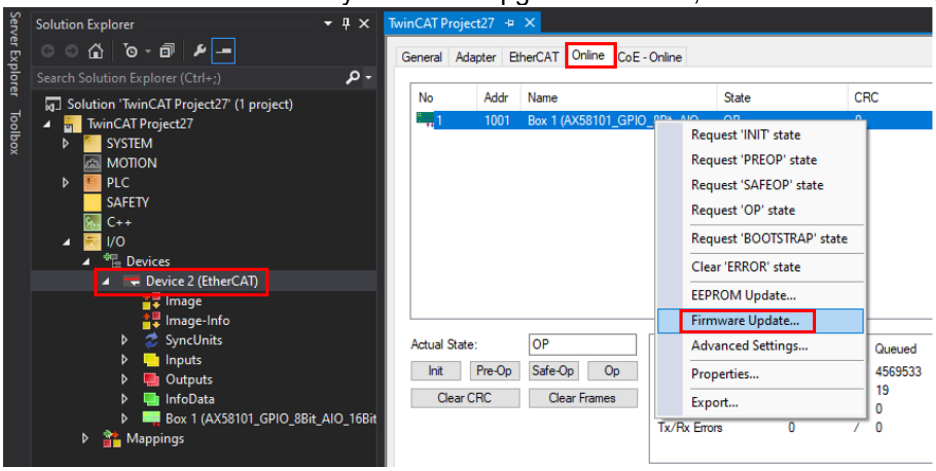


Figure 9-3

Step 2: Select firmware binary file for FoE in the path below.
"**BSP_ROOT\SampleCode\Ax58101_GpioAio\For_TwinCAT\FoE\AX58101_GpioAio.efw**"

Step 3: Click OK to start firmware upgrading.

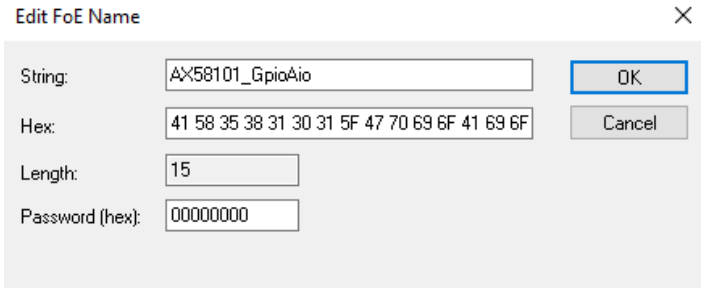


Figure 9-4

Step 4: Wait the process complete, the dialog will pop up as below if upgrading is successfully done.

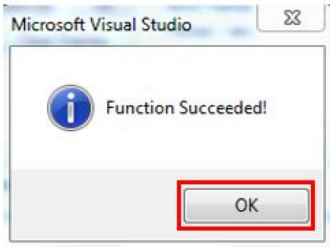


Figure 9-5

10. Virtual ADIO Mode

This example not only provides physical pins for connecting external devices but also offers virtual commands, making GPIO / AIO evaluation and testing more convenient. Virtual ADIO commands must be used in conjunction with the TwinCAT PLC AxGpioAio example. This section can be referenced in Chapter 8.

If users need to connect external devices, the pinout can be referenced in Section 3.2.

10-1 Console Command Usage

Usage		
adio [-dsb] [-ve DecValue] [-vsw HexValue] [-vdi HexValue] [-vai HexValue]		
Parameters	Available value	Description
-dsb	N/A	Enables the dashboard display to show real-time test status or results. Ctrl+C is used to exit.
-ve	0: Disable (Default) 1: Enable	This command is used to enable virtual ADIO Mode. Once enabled, the -vsw, -vdi, and -vai commands can be used to provide virtual values to their respective parameters. These values will be reflected in the corresponding fields on the TwinCAT PLC AxGpioAio operation interface. If disabled, use the physical peripheral pins instead.
-vsw	0~FF	Sets the virtual switch ID state. The parameter must be in hexadecimal format. virtual ADIO Mode must be enabled.
-vdi	0~FF	Sets the virtual digital input state. The parameter must be in hexadecimal format. virtual ADIO Mode must be enabled.
-vai	0~FFF	Sets the virtual analog input value. The parameter must be in hexadecimal format and limited to 12-bit ADC data (0x000 to 0xFFF). virtual ADIO Mode must be enabled.
Examples:		
In virtual ADIO Mode, the digital I/O value is set to 0xff and enable the dashboard: adio -ve 1 -vdi ff -dsb		
In virtual ADIO Mode, the analog I/O value is set to 0x123 and enable the dashboard: adio -ve 1 -vai 123 -dsb		
In virtual ADIO Mode, analog I/O value is set to 0x123, the digital IO value is set to 0x45 and enable the dashboard: adio -ve 1 -vai 123 -vdi 45 -dsb		
In physical ADIO Mode, enable the dashboard: adio -ve 0 -dsb		

Table 10-1

11. Switch ID Description

The Switch ID in this application is used for identification function of TwinCAT MainDevice. User can specify an ID at EtherCAT SubDevice side and setup identification at TwinCAT MainDevice side. TwinCAT MainDevice will check if the ID of the SubDevice matched with the expected identification value during startup of the SubDevice. If you want to know how to setup identification, please follow below example.

You can find out the “IdentificationReg134” setting in ESI file after SSC Tool generated source files and ESI file through import ASIX’s configuration file.

```
<Info>
    <StateMachine>
        <Timeout>
            <PreopTimeout>2000</PreopTimeout>
            <SafeopOpTimeout>9000</SafeopOpTimeout>
            <BackToInitTimeout>5000</BackToInitTimeout>
            <BackToSafeopTimeout>200</BackToSafeopTimeout>
        </Timeout>
    </StateMachine>
    <Mailbox>
        <Timeout>
            <RequestTimeout>100</RequestTimeout>
            <ResponseTimeout>2000</ResponseTimeout>
        </Timeout>
    </Mailbox>
    <IdentificationReg134>true</IdentificationReg134>
</Info>
```

Figure 11-1

This example provides both physical pins and the ability to set virtual values to simulate the Rotary Dip Switch through Console commands. However, the physical pins require an actual rotary switch to be connected externally.

The diagram below provides a pinout reference for SW0-7.
Note: To use the SW0–SW7 pinout, each pin must be connected to DC_5V with an external pull-up resistor (recommended 4.7 kohm). Since the pinout is active low, for example, to set the Switch ID to 0x05, pull SW0 and SW2 low.

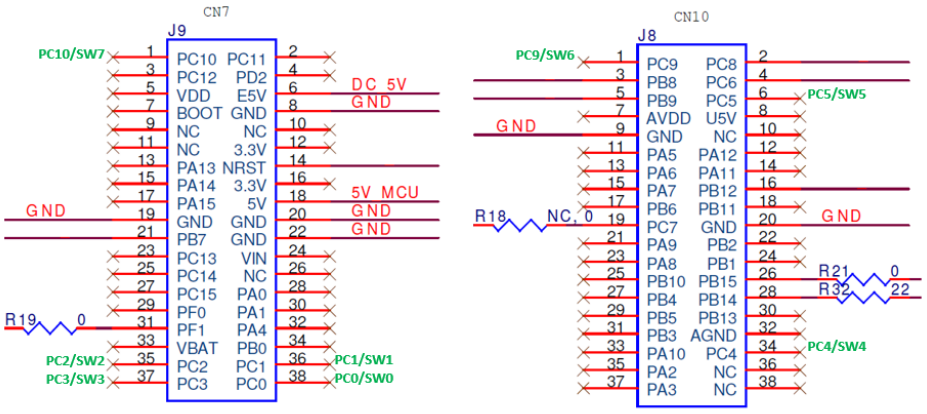


Figure 11-2

This setting will make TwinCAT MainDevice enable the identification function for the SubDevice. Select the SubDevice, right-click and select “**Advanced Settings…**”.

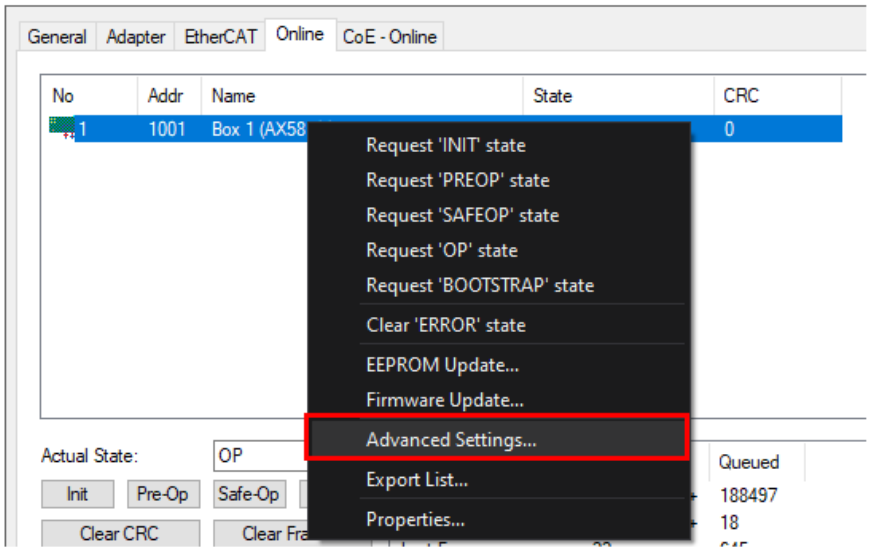


Figure 11-3

Go to “**General -> Behavior**” and turn on the “**Check Identification**”.

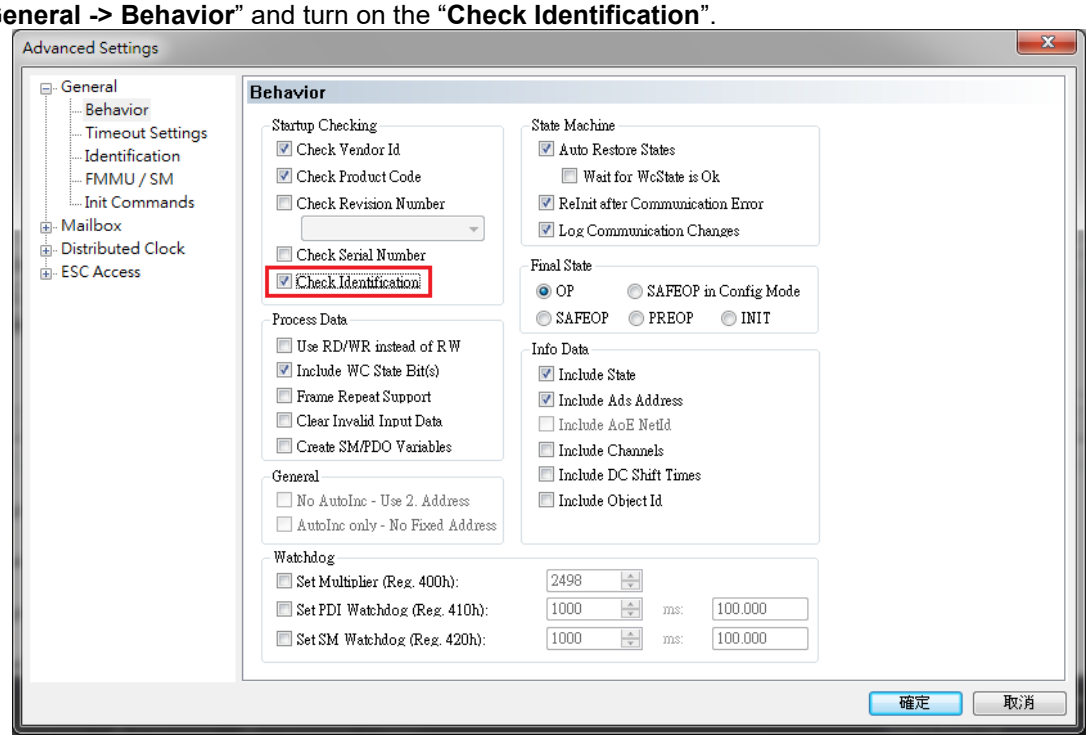


Figure 11-4

Confirm current identification ADO (Address Offset) is “**(ADO 0x0134)**”.
Don’t forget to assign the value in decimal. The value “**55**” is just an example here.

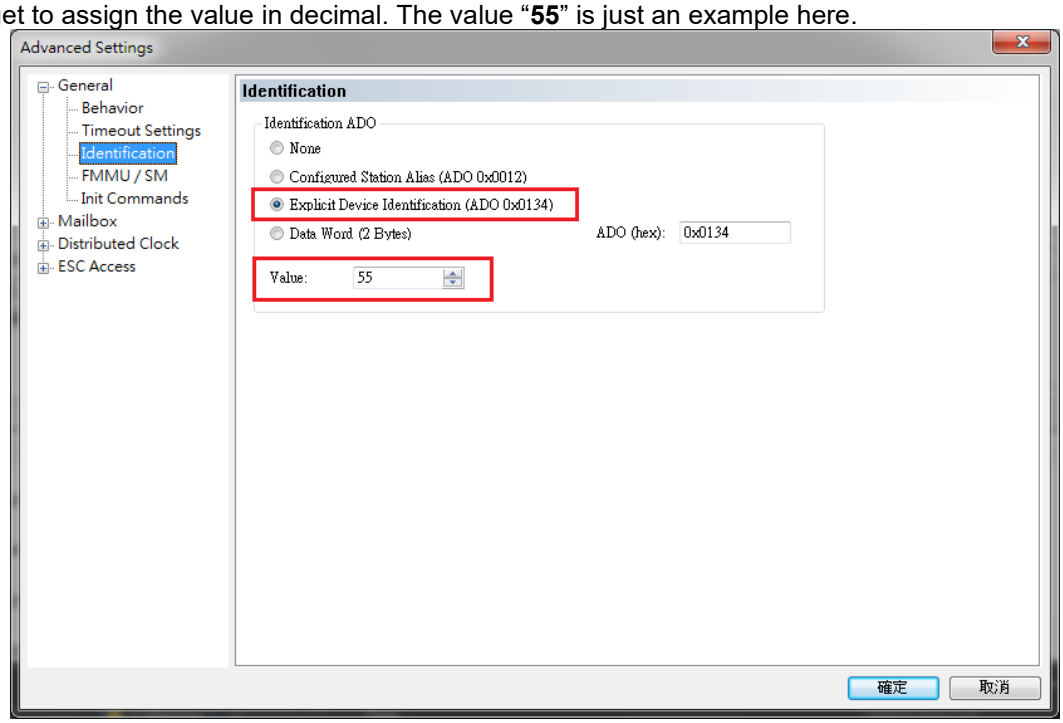


Figure 11-5

Of course, you should also set up the same value by simulating the Rotary Dip Switch through Console commands like following:

Assuming users are already familiar with executing commands via the console, otherwise, users can refer to Section 6.4 and Chapter 10.

Command: adio -ve 1 -vsw 37 -dsb

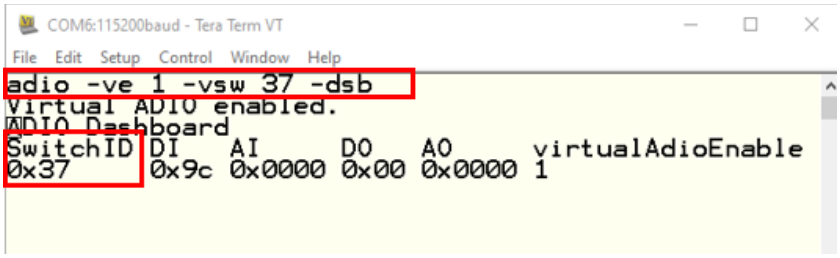


Figure 11-6

Click the "**Reload Devices**" button, and the state will change to "**OP**."

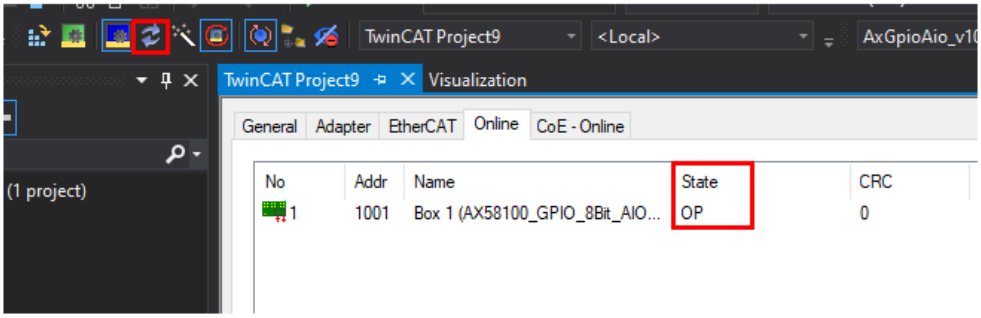


Figure 11-7

Clicked "**Activate Configuration**".

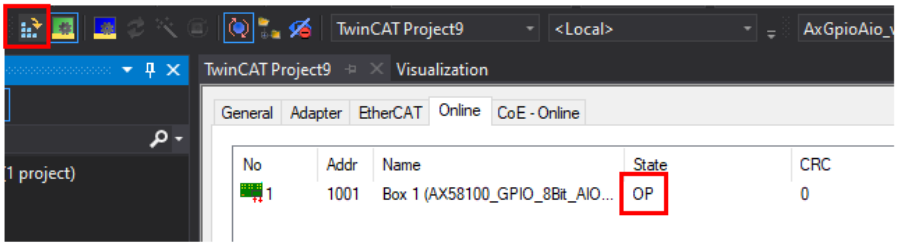


Figure 11-8

Since this example does not have an actual Rotary Dip Switch connected, we can observe that the value of ESC 0x012 will be a random value, differing each time the system is powered on. This behavior is normal. Next, we will arrange the commands to observe this behavior.

Go to "**ESC Access -> Memory**" in "**Advanced Settings**" and check out the address 0x0012.

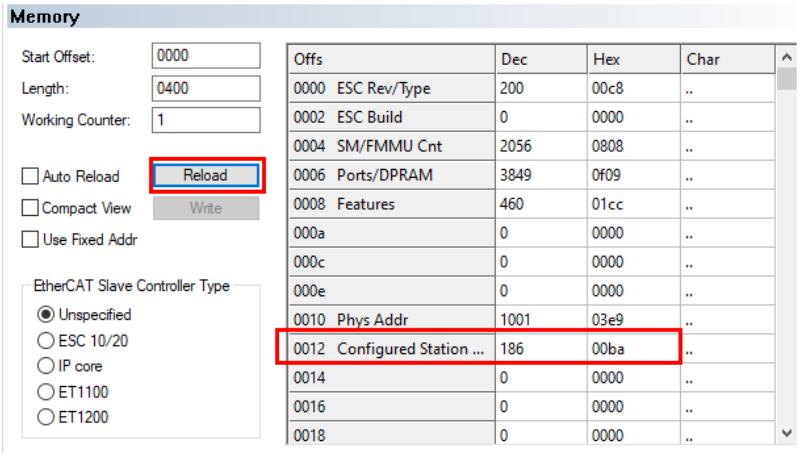


Figure 11-9

You can append an EtherCAT command to TwinCAT MainDevice for confirm the second address.
Click “**Device N (EtherCAT) -> Append EtherCAT Cmd**”.

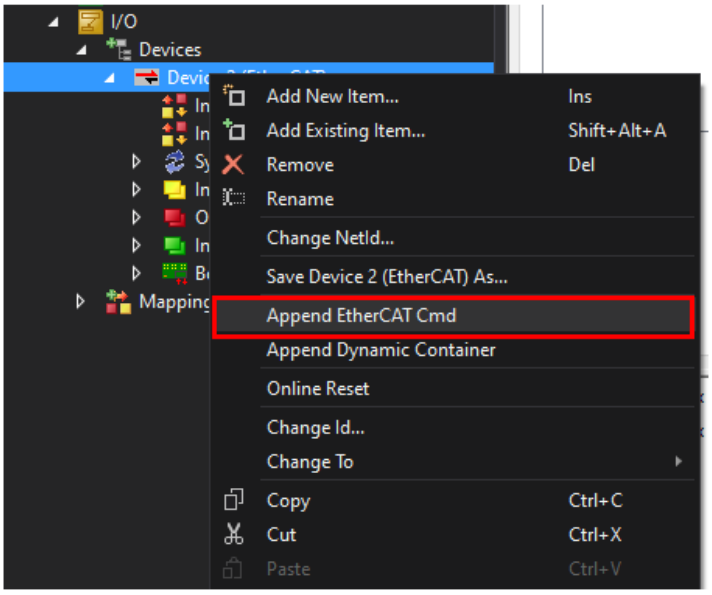


Figure 11-10

Please assign “**Command**” and “**Type**”.
I demonstrate “**FPRD**” and “**UINT**” in this example.

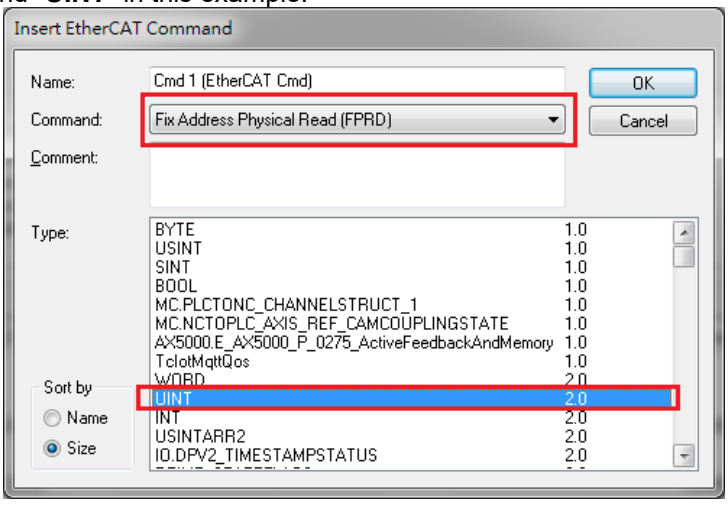


Figure 11-11

Double-click “**Cmd 1**” and setup “**SubDevice Address**” and “**Address Offset**”.
I setup SubDevice address = 0xba and address offset = 0x12 in this example.

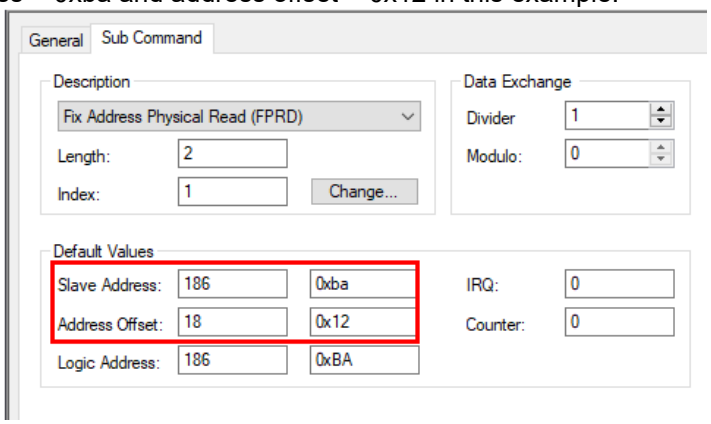


Figure 11-12

Clicked “**Activate Configuration**”.

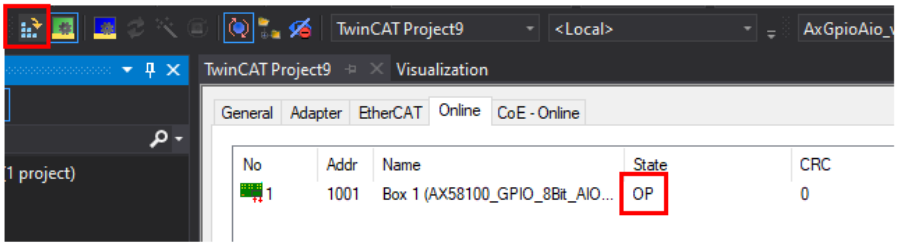


Figure 11-13

If you expect that the SubDevice can also use this Switch ID value as second SubDevice address. You just turn on “**Seconds address**” bit (bit8 = 1) in ESC register 0x0102.

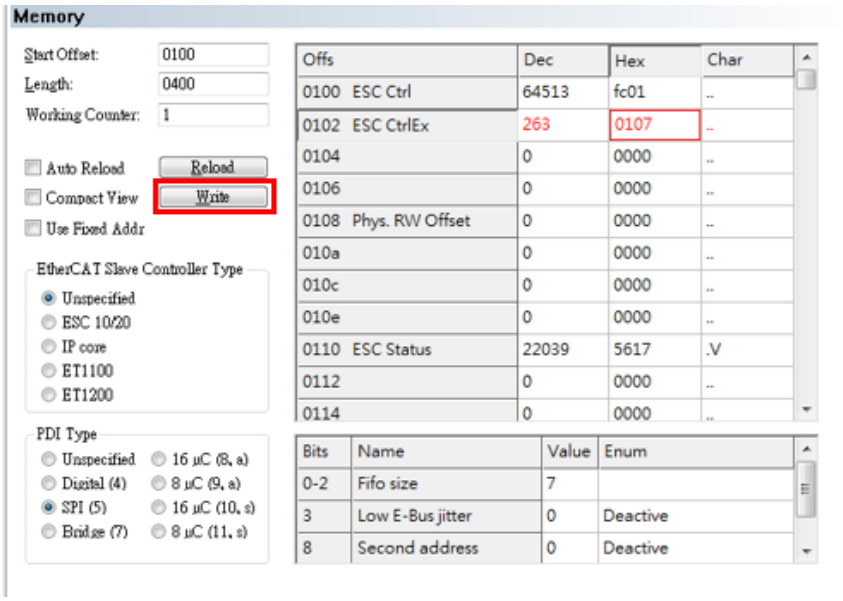


Figure 11-14

Click “**Device N (EtherCAT) -> Cmd 1 -> DATA**”, you will see the command get the value of ESC register 0x0012 back successfully. It is correct, the “**0x00ba**” in UINT data type.

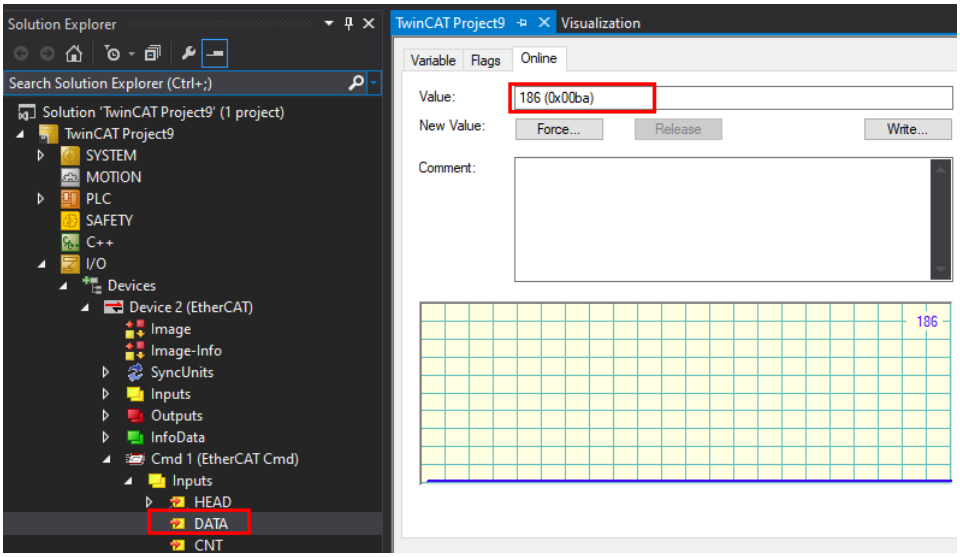


Figure 11-15

12. Shortest EtherCAT Cycle Evaluation

We tried to decrease the TwinCAT cycle to evaluate shortest cycle of AX58101 GPIO / AIO.
Shortest workable cycle of AX58101 GPIO / AIO is "**250us**" now.
This reference design allocates TX-PDO (3 bytes) and RX-PDO (3 bytes) totally.
Please refer to below figures.

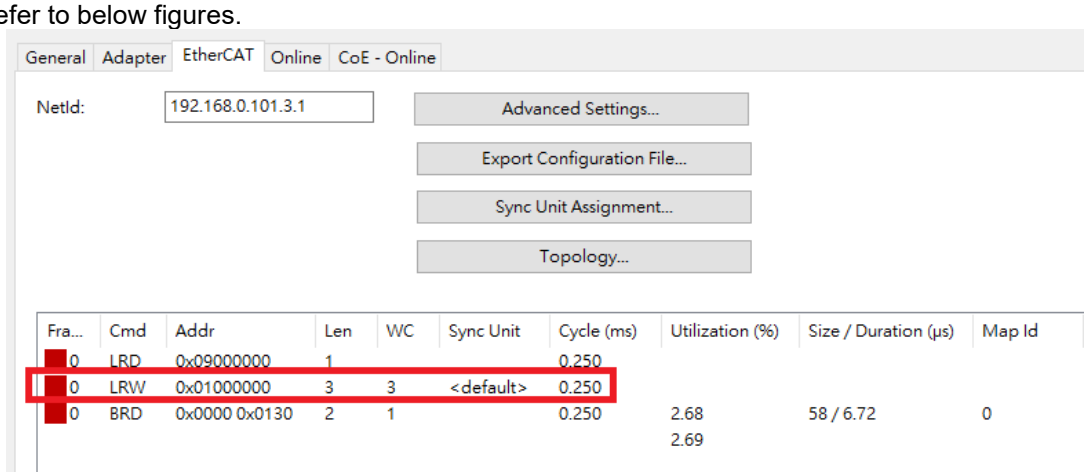


Figure 12-1

Under this condition, the "**Lost Frames**" and "**Tx/Rx Errors**" does not count.

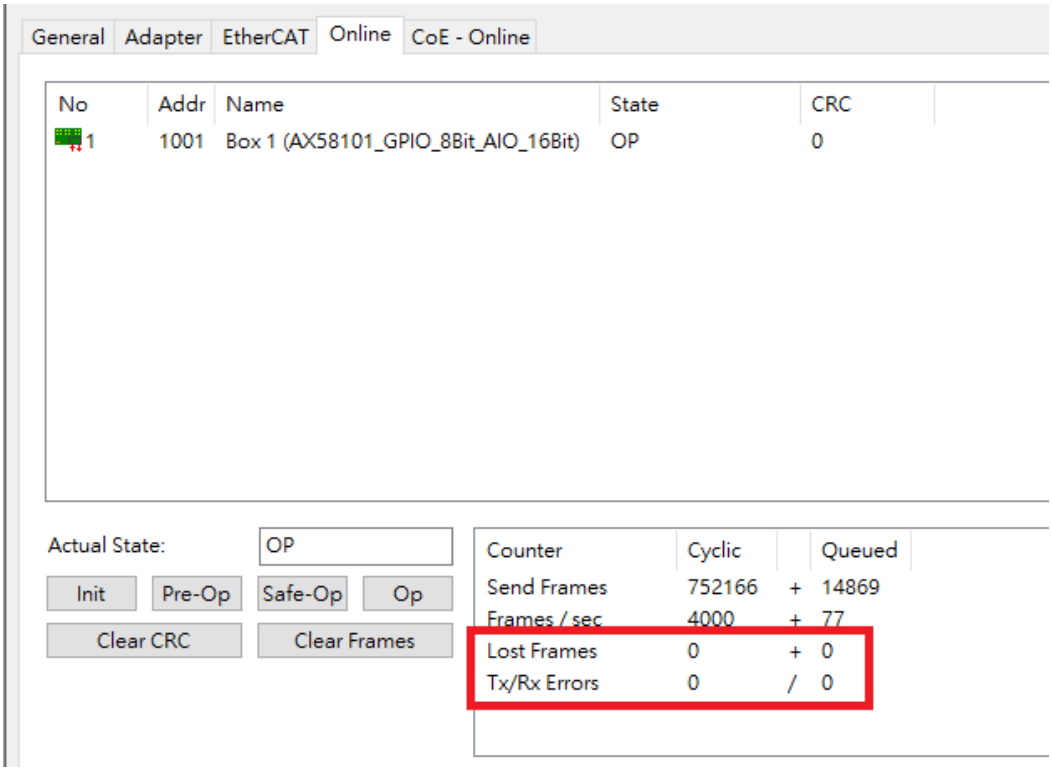


Figure 12-2

The SubDevice stack also does not indicate any errors in SM output parameter / SM input parameter objects of CoE-Online table.

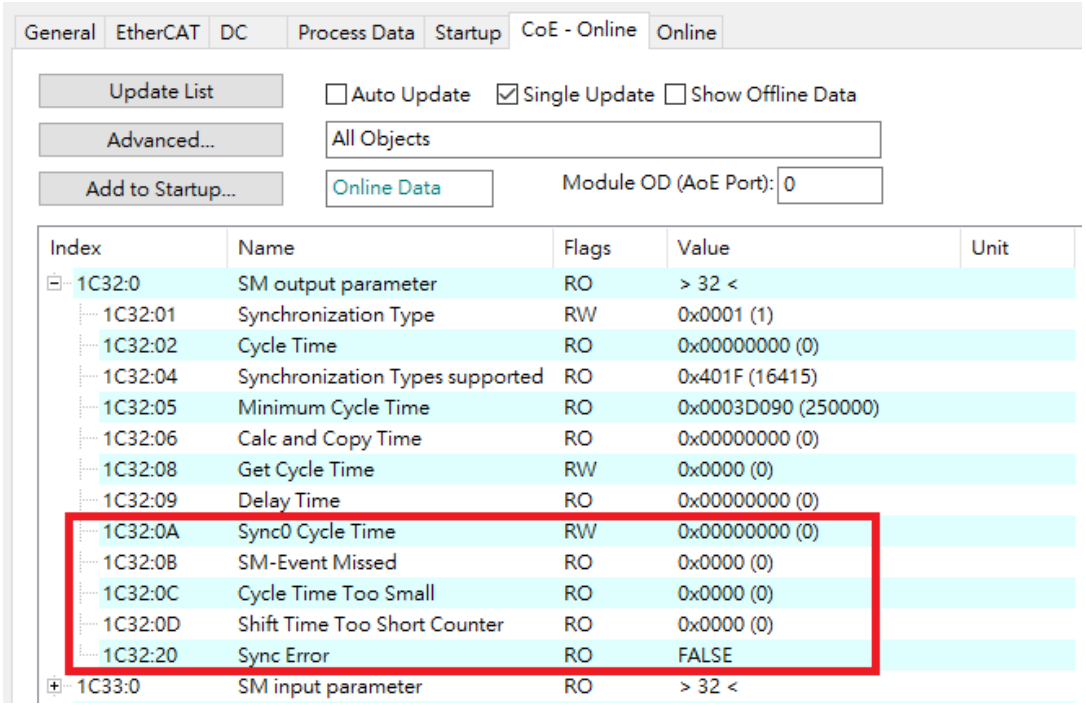


Figure 12-3

Evaluation Board License Agreement

By using this evaluation board or kit (together with all associated software, hardware and documentation provided by ASIX, collectively, the "Evaluation Board"), you ("You") agree to be bound by the terms and conditions of this Evaluation Board License Agreement ("Agreement "). Do not use the evaluation board until you have read and agreed to this agreement. Your use of the Evaluation Board constitutes your acceptance of this Agreement.

License:
ASIX Electronics (“ASIX”) grants you the right to use the accompanying evaluation board, which provides limited functionality, for the sole purpose of evaluating and testing ASIX products for your evaluation and testing purposes in a research and development environment. Under no circumstances should the evaluation board be assembled, directly or indirectly, as part of any of your products, as it has been developed for evaluation purposes only and has no direct functionality and is not a finished product.

Notice:
(1) The description of hardware circuit, software tools and other related information in this document are only used to illustrate the operation of application examples for ASIX Electronic products. You are solely responsible for the use of such circuits, software and information in the design of your equipment. ASIX Electronics is not responsible for any losses arising out of the use of the circuit, software or information by you or a third party.
(2) ASIX Electronics has taken reasonable care in describing the information in this document, but ASIX Electronics does not warrant that such information is error-free. ASIX Electronics disclaims all liability for any damages you may suffer as a result of errors or omissions in the information contained herein.
(3) ASIX Electronics shall not be liable for infringement of patents, copyrights or other intellectual property rights of third parties due to the use of ASIX Electronics products or the technical information described in this document. No license, express or implied, or otherwise, is granted under any patent, copyright or other intellectual property rights of ASIX Electronics or others.
(4) You may not alter, modify, copy or otherwise misappropriate any ASIX electronic product, whether in whole or in part. ASIX Electronics shall not be liable for any loss incurred by you or a third party as a result of altering, modifying, copying or otherwise misappropriating ASIX Electronics products.
(5) You should use the ASIX Electronic products described in this document within the range specified by ASIX Electronic, especially the maximum rating, operating power supply voltage range, mobile power supply voltage range, heat radiation characteristics, installation and other product characteristics. ASIX Electronics accepts no liability for malfunctions or damages resulting from use of ASIX Electronics products outside of such designations.
(6) When using ASIX electronic products, please comply with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including but not limited to the EU RoHS Directive. ASIX Electronics disclaims all liability for damage or loss resulting from your failure to comply with applicable laws and regulations.
(7) The buyer or distributor of ASIX electronic products is responsible for distributing, disposing of or otherwise placing the product to a third party, and notifying the third party of the contents and conditions specified in this document in advance, ASIX Electronics shall not be liable for any loss suffered as a result of authorized use of ASIX Electronics products.
(8) This document may not be reproduced or reproduced in any form, in whole or in part, without the prior written consent of ASIX Electronics.

DISCLAIMER

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of ASIX. ASIX may make changes to the product specifications and descriptions in this document at any time, without notice.

ASIX provides this document “as is” without warranty of any kind, either expressed or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

Designers must not rely on the absence or characteristics of any features or registers marked “reserved”, “undefined” or “NC”. ASIX reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. Always contact ASIX to get the latest document before starting a design of ASIX products.

TRADEMARKS

ASIX, the ASIX logo are registered trademarks of ASIX Electronics Corporation. All other trademarks are the property of their respective owners.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

ASIX Electronics Corporation

4F, No.8, Hsin Ann RD., Hsinchu Science Park,
Hsinchu, Taiwan, R.O.C.

TEL: +886-3-5799500
FAX: +886-3-5799558

Email: support@asix.com.tw
Web: https://www.asix.com.tw