ECM-XF Sample Code User Guide

(for STM32 F4/H7 & Nuvoton M487 board)

NEXTW Technology CO., LTD.

V.05

08.17.2021

STM32 Sample Code List 3

Nuvoton Sample Code List 4

STM32 Sample Setup Instruction (NUCLEO-F401RE) 5

STM32 H7 Pin Configuration 13

Nuvoton Sample Setup Instruction 14

Modify Sample Code to Your Application Code 16

Maintance Log 22

ECM-XF Sample Code Guide

The sample codes you download have different types of them to use. This introduction will tell you how to pick the right one and modify them into usable project code.

The first thing you need to verify which one you want to use based on the MCU you are using. The sample codes are based on STM32 F4 boards, STM32 H7 boards or Nuvoton M487 board. If your project will not run on these boards, you need to figure out how to hook up the MCU and the ECM-XF chips via SPI connection.

STM32 Sample Code List

The following projects are for STM32 boards with STM32 IDE.

- STF4DAC_ADC
  The project is to test DAC and ADC functions on the ECM-XF chips.
- STF4Drive_IO
  This project shows 2 drives with a motor on each of them and an IO link together.
- STF4Drive
  A motor with a drive operation.
- STF4EEPROM
  The project shows the EEPROM data on the slave.
- STF4GPIO
  The project for testing GPIO pins on the ECM-XF chip.
- STF4Homing
  This project shows how to operate homing on a drive and a motor.
- STF4HSP
  The NEXTW HSP with 2 steppers operating demo.
- STF4HSP_A
  The NEXTW HSP with automatically 402 state machine transfer.
- STF4HSP_IO
  The NEXTW HSP and an IO operating demo.
- STF4IO
  The demo with only Junction and IO connection.
- STF4PP
  The demo of profile position mode.
- STF4QEI
  The QEI is an encoder signal that can read from the ECM-XF chip pins.
- STH7Drive_IO
  Similar with STF4Drive_IO. The difference is this project runs on STM32 H7 boards.

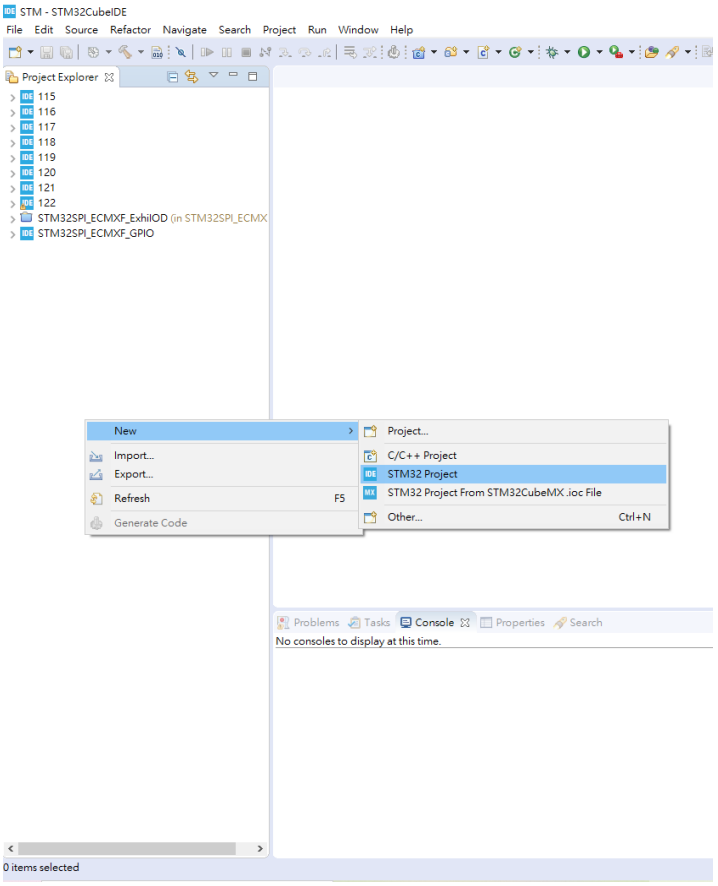
Nuvoton Sample Code List

The following projects are for Nuvoton M487 boards with NuEclipse.

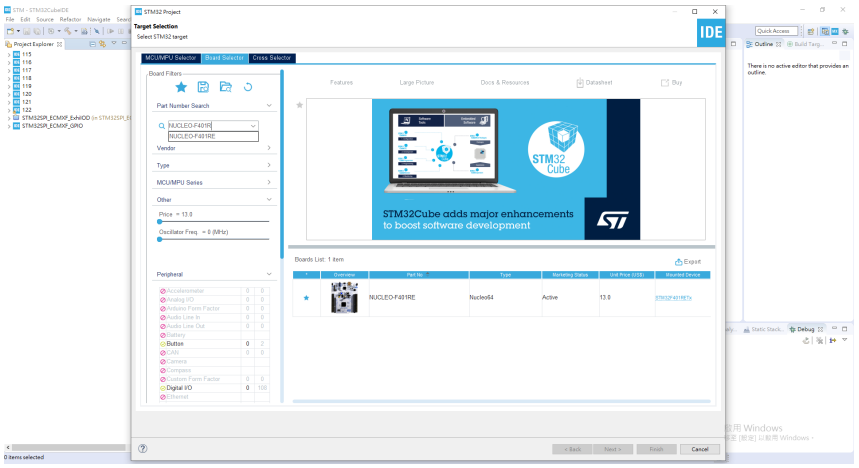
- NuDAC_ADC
  The project is to test DAC and ADC functions on the ECM-XF chips.
- NuDrive_IO
  This project shows a drive with a motor and an IO link together.
- NuDrive
  A motor with a drive project.
- NuEEPROM
  The project shows the EEPROM on the slave.
- NuGPIO
  The project for testing GPIO pins on the ECM-XF chip.
- NuHoming
  This project shows how to operate homing on a drive and a motor.
- NuHSP
  The NEXTW HSP with 2 steppers operating demo.
- NuHSP_A
  The NEXTW HSP with automatically 402 state machine transfer.
- NuHSP_IO
  The NEXTW HSP and an IO operating demo.
- NuIO
  The demo with only IO connection.
- NuPP
  The demo of profile position mode.
- NuQEI
  The QEI is an encoder signal that can read from the ECM-XF chip pins.

STM32 Sample Setup Instruction (NUCLEO-F401RE)

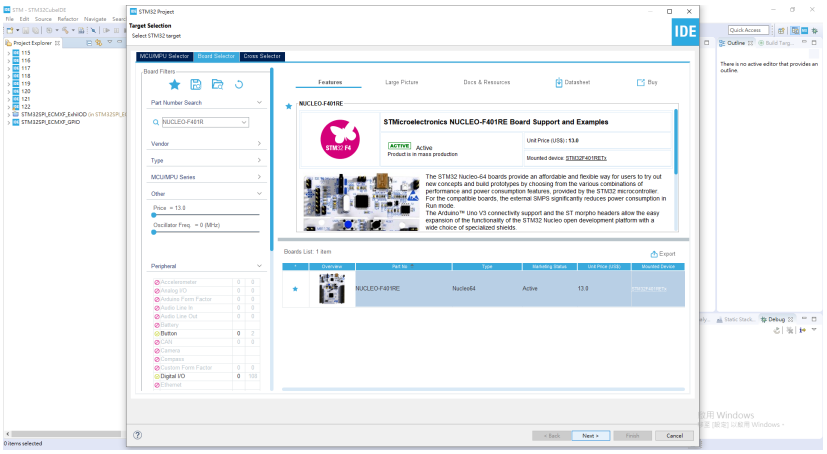
1. Open STM32Cube IDE.
2. Right click and select “STM32 Project”.



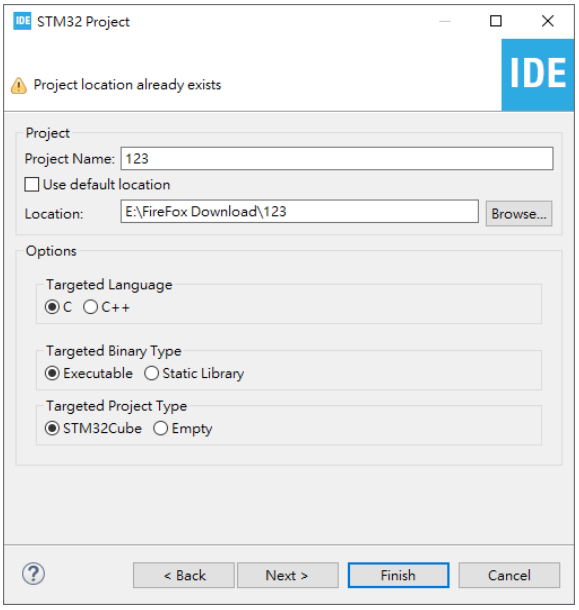
3. Select the board you have. For example, the selected board is NUCLEO-F401RE.



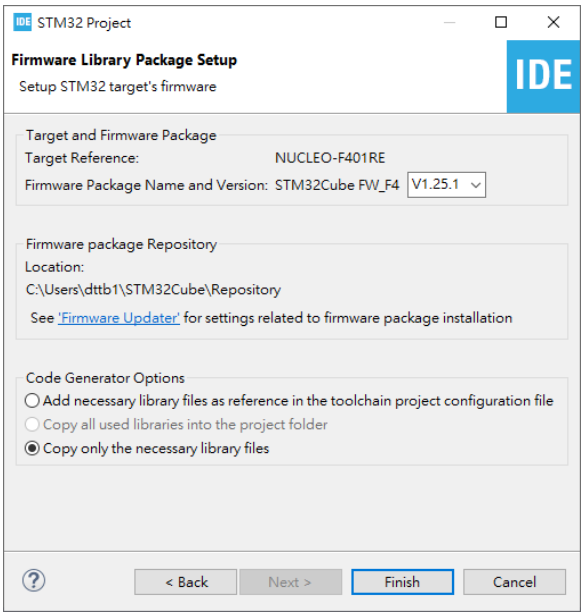
4. Then click “Next” to continue setup.



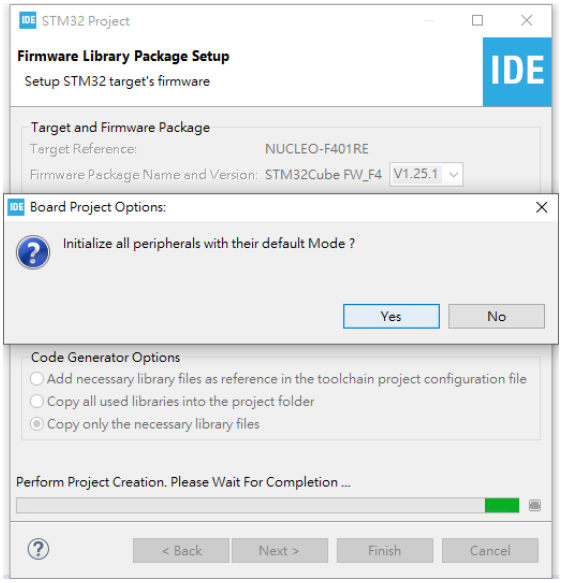
5. Insert the project name and decide project location.



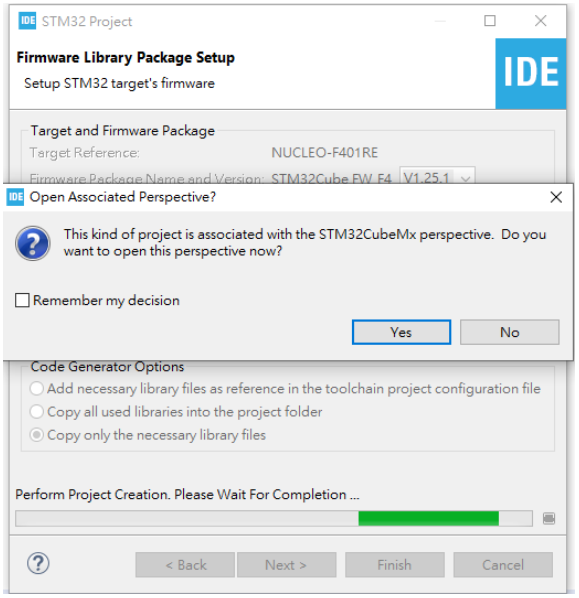
6. Choose firmware then click “Finish”. If you don’t have the firmware you are selected, the system will process to download the firmware.



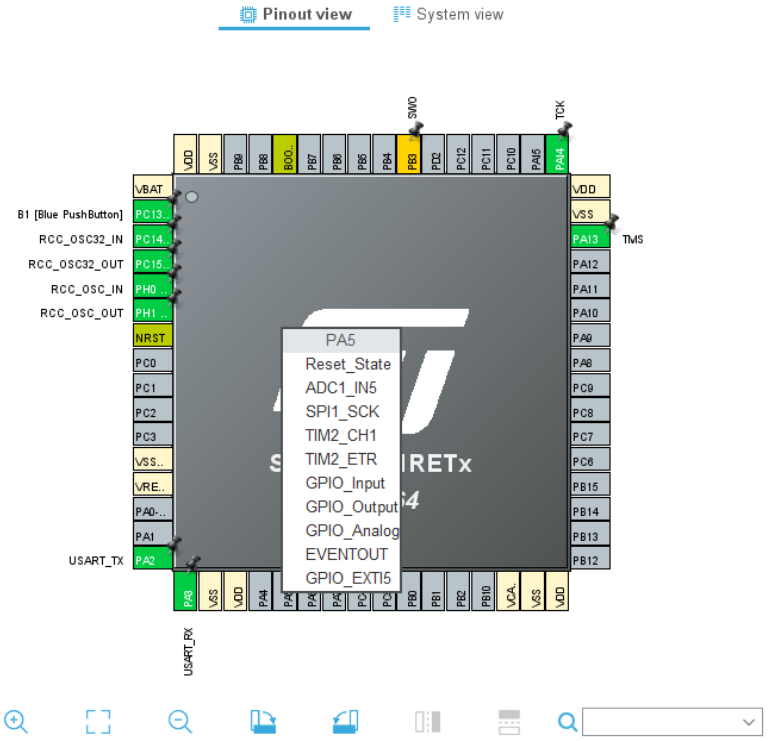
7. Initialize all peripherals with their default Mode? click “Yes”.

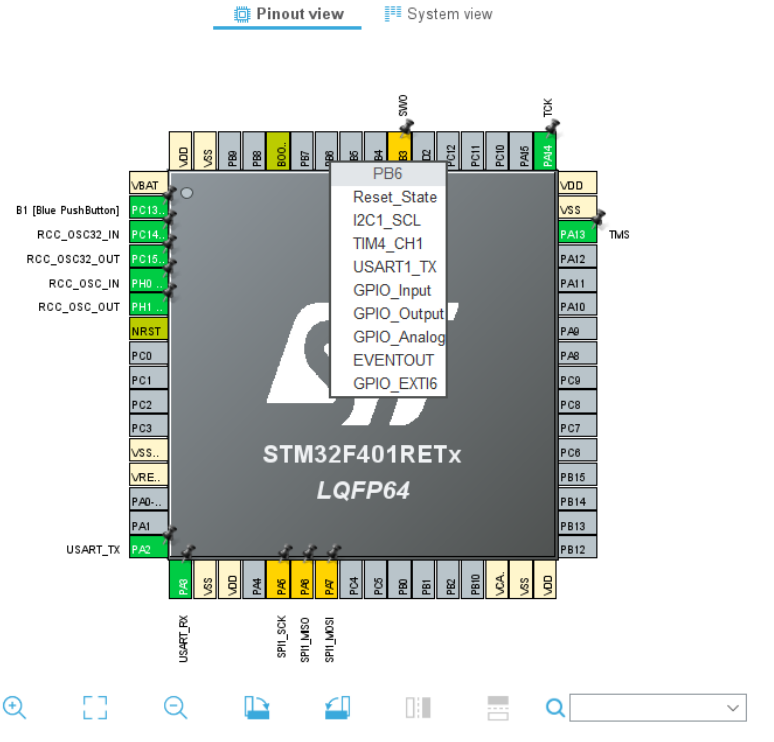


8. Open the STM32CubeMx for pinout define.

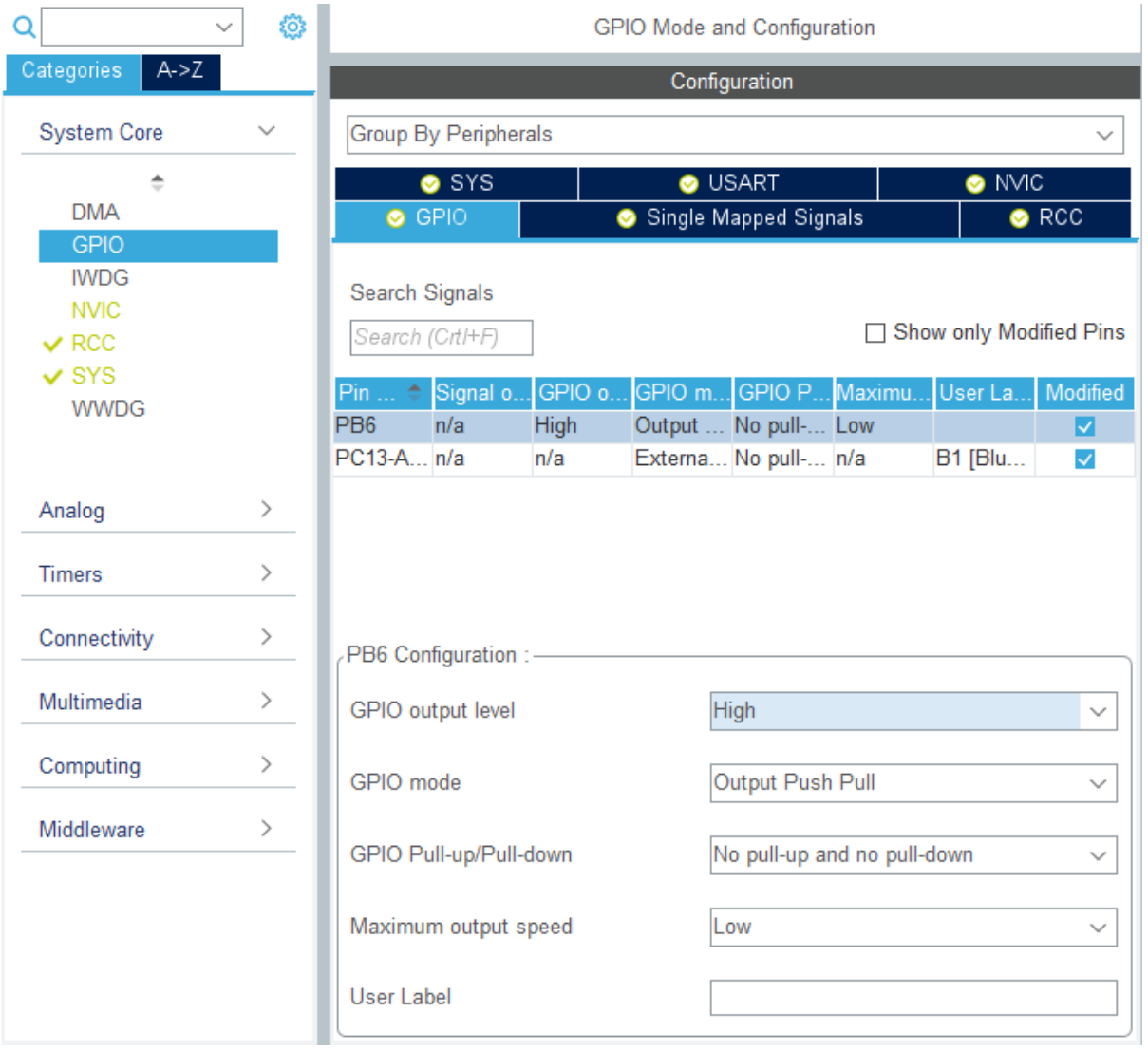


9. In pinout view, choose the SPI1_SCK at PA5, SPI1_MISO at PA6, SPI1_MOSI at PA7, GPIO_Output at PB6.

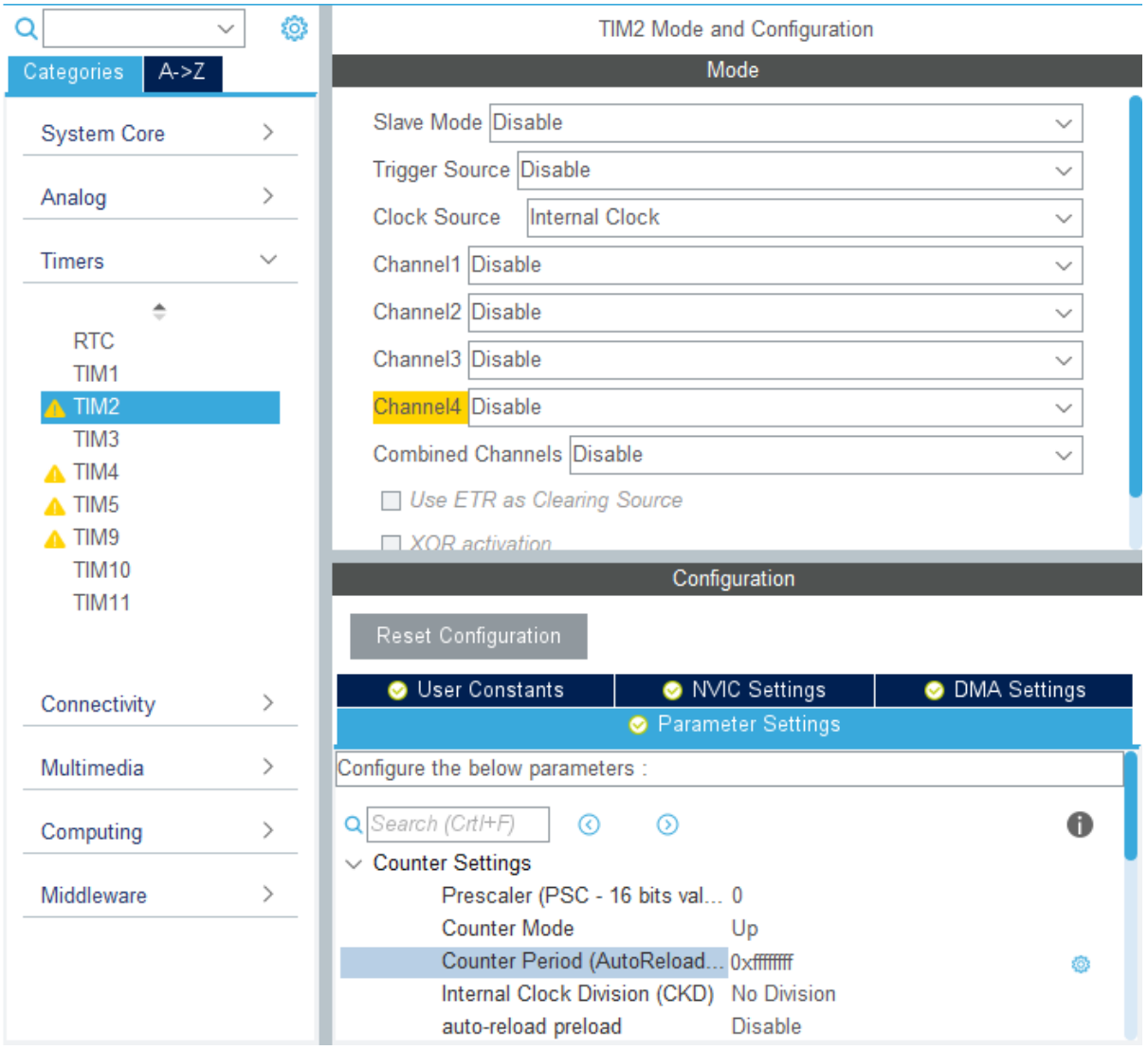




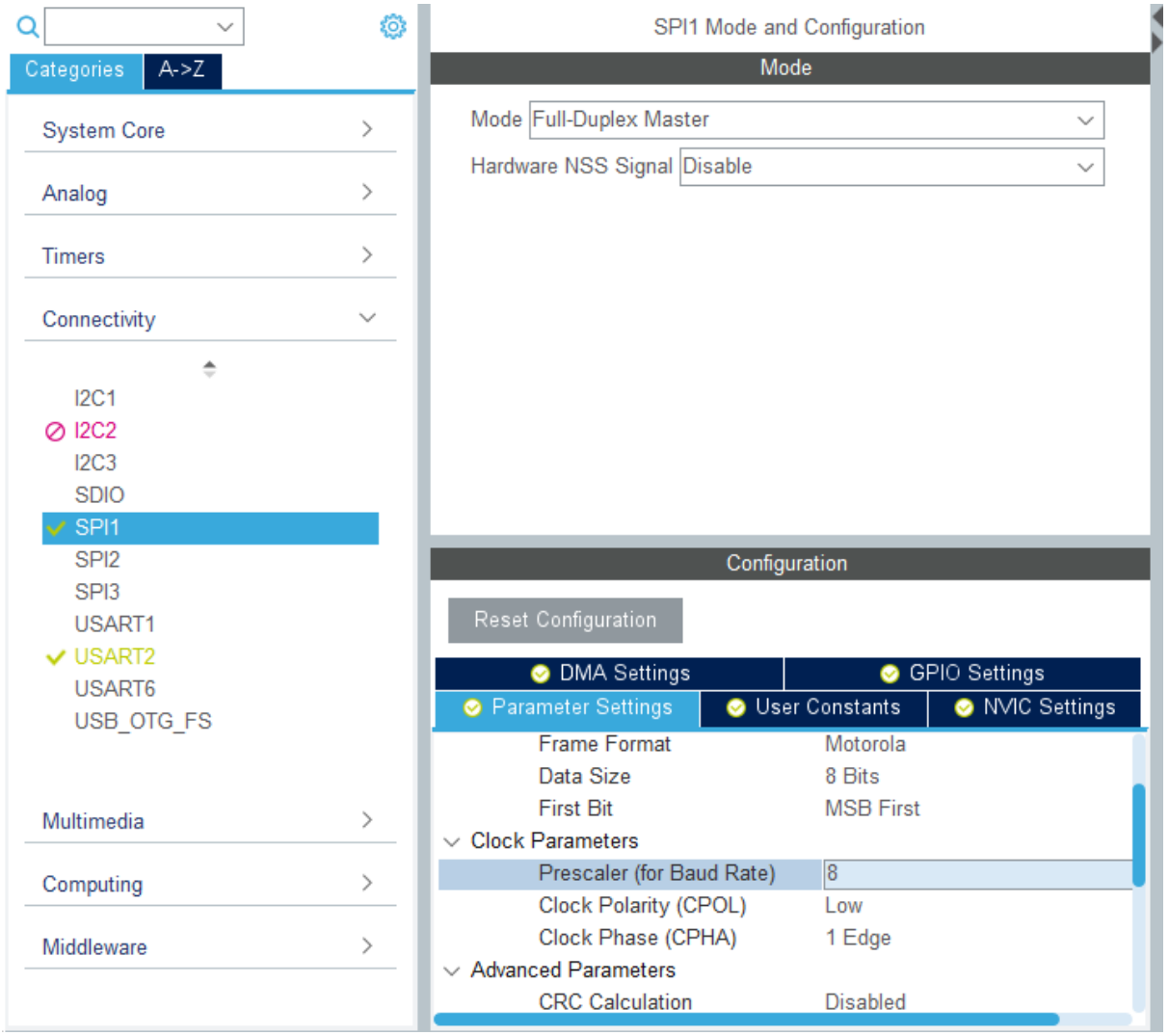
10. Then click “System Core” for further GPIO setting, change “GPIO output level” to “High”.



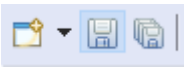
11. Move to “TIM2” in “Timers”. Define “Clock Source” as “Internal Clock” and input “0xFFFFFFFF” in “Counter Period”.

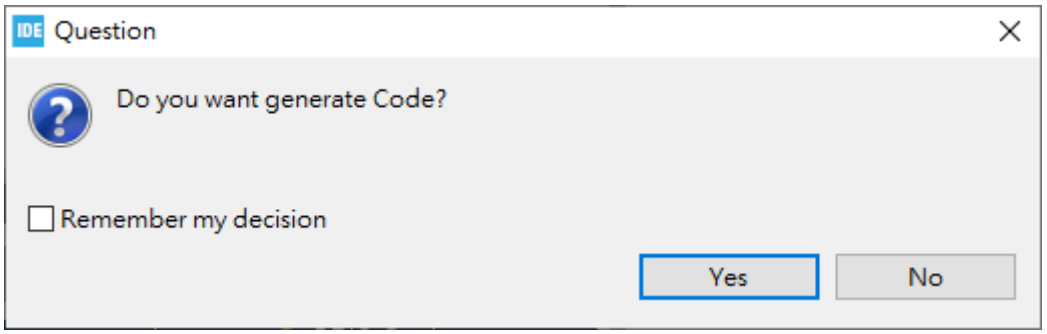


12. Switch to “SPI1” in “Connectivity”, choose mode into “Full-Duplex Master” and set “2~8” for “Prescaler (for Baud Rate).



13. After finishing all settings, press the button to save and generate the code.





14. If you find the following code setup by system in your Private variables in main.c means you got all the sample code function needed.

```
/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi1;

TIM_HandleTypeDef htim2;

UART_HandleTypeDef huart2;
```

15. Open “main.h” and add sample code show as below:

```
/* USER CODE BEGIN Includes */
#include "platform.h"
#include "EcmUsrDriver.h"
/* USER CODE END Includes */


/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */
#ifndef PRINTF
#define PRINTF( str, ...) \
    do{ \
        int n; \
        n = sprintf( printbuf, (str), ##__VA_ARGS__); \
        HAL_UART_Transmit( &huart2, (uint8_t *)printbuf, n, 0xffffffff); \
    }while(0)
#endif
#ifndef GETCHAR
#define GETCHAR userGetchar
#endif
/* USER CODE END EM */


/* USER CODE BEGIN EFP */
extern UART_HandleTypeDef huart2;
extern char printbuf[];
/* USER CODE END EFP */
```

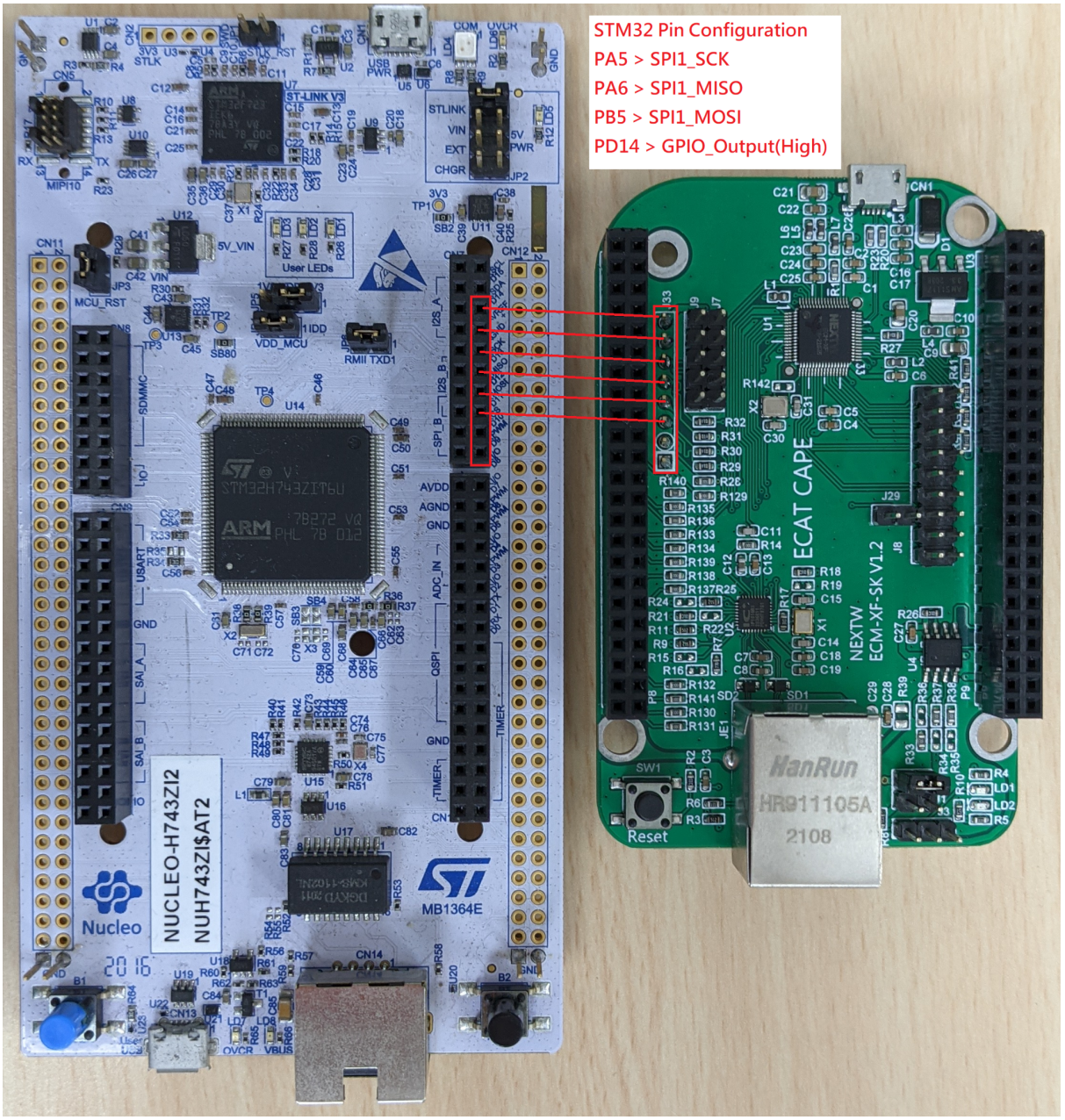
16. Open “main.c” and add setting show as following pictures:

```
/* USER CODE BEGIN PFP */
char printbuf[128];
int main_ini(void);
/* USER CODE END PFP */


/* USER CODE BEGIN 2 */
main_ini();
/* USER CODE END 2 */


/* USER CODE BEGIN TIM2_Init 2 */
HAL_TIM_Base_Start(&htim2);
/* USER CODE END TIM2_Init 2 */
```

17. Copy the files: EcmDriver.h, EcmUsrDriver.h, PdoDefine.h, platform.h, and Utility.h(partial) into your corresponding “Inc” file.
18. Copy the files: crc32.c, EcmUsrDriver.c, main_ini.c(main code but some have other names), platform.c and Utility.c(partial) into your corresponding “Src” file.
19. Ready to run sample code.

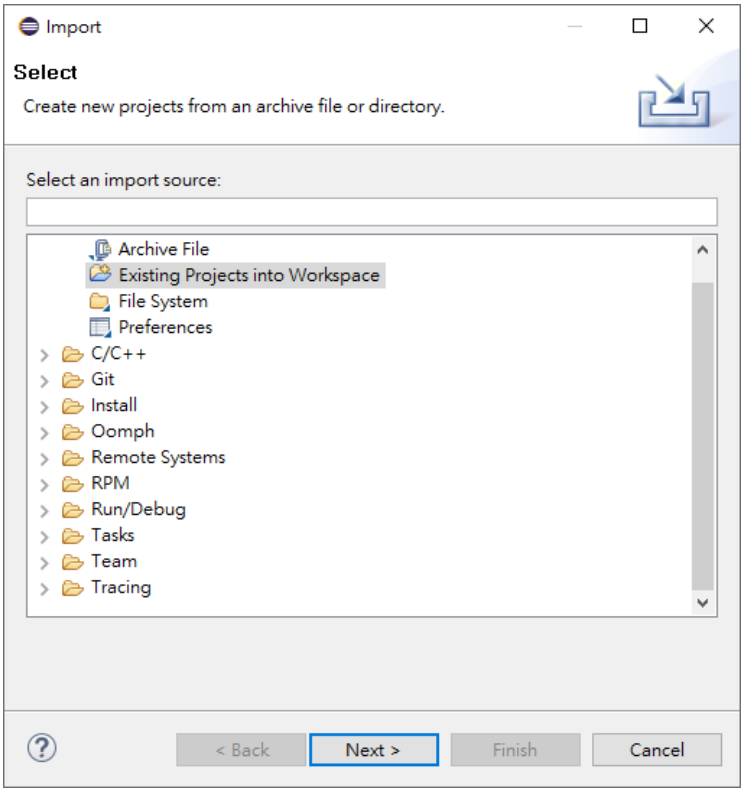
STM32 H7 Pin Configuration

STM32 automatically generate main.c and main.h are not compatible with F4 version. Please use H7 sample to modify. The platform file including platform.c and platform.h are independent for H7 series also not compatible with F4 version. The main_ini.c for applications in the F4 version are all available to replace in the H7 version.

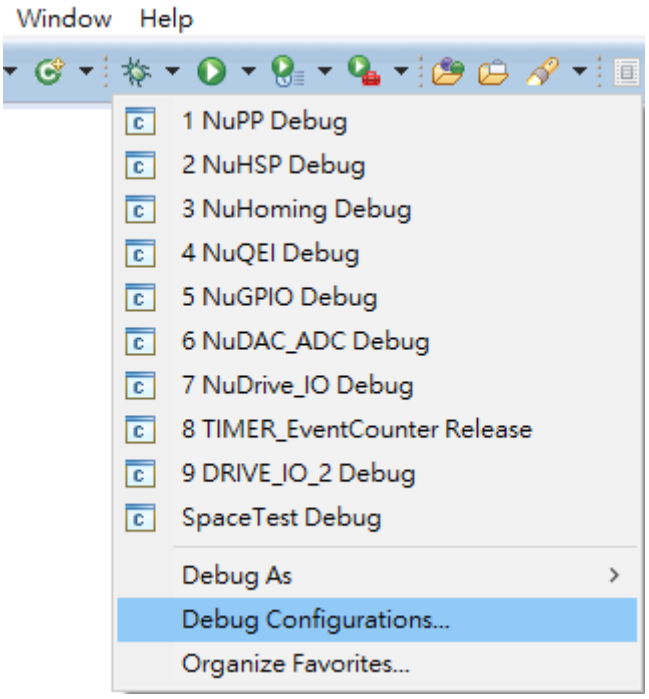


Nuvoton Sample Setup Instruction

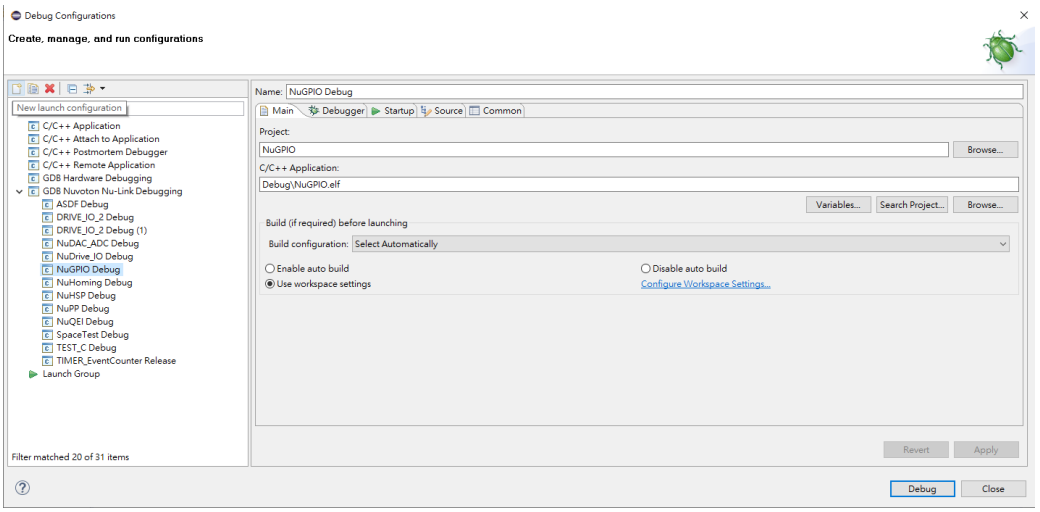
1. Choose your OS and download “Nueclipse GCC”(open source IDE) from Nuvoton official website: https://www.nuvoton.com/products/microcontrollers/arm-cortex-m4-mcus/m487-ethernet-series/?group=Software&tab=2
2. Download Nu-Link Keil driver from the same page.
3. Download M480_BSP_CMSIS
4. Install IDE and Keil driver.
5. Download the sample code and put them into the workspace location.
6. Open Eclipse and right click the area in Project Explorer. Choose import.
7. In the import window, select General > existing project into workspace.



8. Select the root directory which the sample project you want to open. The project will appear in the middle of the window. Then click Finish.
9. After importing the project successfully, right click or press the icon on the toolbar to build the project. The project allows build in Debug mode and Release Mode. It is recommended to build in Debug mode.



10. With building success, continue to set Debug configuration or Run configuration. Click the icon on the upper left corner, New launch configuration, and it will show the same project name as you build before. Then click Debug. It would automatically write the code into the chip.



Modify Sample Code to Your Application Code

After opening files correctly, please follow the sequence to modify code into your own application.

The sample for modification using STFDrive and NuDrive as base and some other sample will be mentioned when needed.

PdoDefine.h part:

1. Click and open the PdoDefine.h, then find the #define as the picture shows below to enter the drives count and IO counts. A drive might boost two motors, but here counts the drives, not motors(axes).

```
#define TEST_DRV_CNT            1
#define TEST_IO_CNT             0
```

2. After setting the slave counts, drag down to continue the PDO structure. The PDO structures are made by RxPDO and TxPDO, if the slave structures are not the same, define the different structure as you need and make a macro to conclude all the structures as the end.

```
typedef struct __attribute__((__packed__)) _axis_rxpdo_st_def_t
{
    uint16_t    u16CtlWord;     //  0x6x40 - Control word
    int32_t     n32Target;      //  0x6x7A - Target position
                                //  0x60FF - Target velocity
#ifdef TEST_HSP_DEV
    int32_t     n32Out;         //  0x6xFE
#endif
}AXIS_RXPDO_ST_DEF_T;
typedef struct __attribute__((__packed__)) _axis_txpdo_st_def_t
{
    uint16_t    u16StaWord;     //  0x6x41 - Status word
    int32_t     n32Actual;      //  0x6x64 - Actual position
                                //  0x6063 -
#ifdef TEST_HSP_DEV
    int32_t     n32In;          //  0x6xFD
#endif
}AXIS_TXPDO_ST_DEF_T;
```

STM32\main_ini.c or Nuvoton\main.c part:

1. First of all is the SPI communication frequency, STM32 series need to set the SPI frequency with STM32CubeMX shows before, and Nuvoton series can be define at “#define TEST_SPI_FREQ” or search “UserSys_Init()” and enter SPI frequency. The BASE_CYCTIME is EtherCAT cycle time.

```
#define TEST_SPI_FREQ           24000000
#define DC_ACTIVE_CODE          0x300
#define BASE_CYCTIME            1000000


UserSys_Init(TEST_SPI_FREQ);
```

2. In this sample code provide several settings for testing like multiple or divid the cycle time, and also provide RPM and PPR setting, etc.

```
/*
 * TEST_CYCTIME_DIVID
 * 1 : for TEST_CYCTIME_MULTI
 * 2 : 0.5ms
 * 4 : 0.25ms
 * 8 : 0.125ms
 *
 * TEST_CYCTIME_MULTI
 * 1 : for TEST_CYCTIME_DIVID
 * 2 : 2ms
 * 4 : 4ms
 * */
#define TEST_CYCTIME_DIVID      1
#define TEST_CYCTIME_MULTI      1
#define TEST_CYCTIME_NS         ((BASE_CYCTIME*TEST_CYCTIME_MULTI)/TEST_CYCTIME_DIVID)
#define ONE_SEC_CYC_CNT         (1000000000/TEST_CYCTIME_NS)
```

3. The next is FIFO setting, FIFO is a memory to store commands. They will be sent by the EtherCAT master depending on the cycle time. There is a scenario in which the drives operate the received commands faster than cycle time. It means the drives are idle and waiting for new commands. It causes the control mess and not as you expected. In order to deal with this scenario, we provide sending multiple commands in a cycle time to make the drives filled with commands. “TEST_PDO_TO_FIFO_ONCE” means the number of the commands sent in a cycle time. The value of it needs to be tested to find the optimal one.

```
#define TEST_PDO_TO_FIFO_ONCE   2
#define TEST_RXFIFO_CNT         40
#define TEST_TXFIFO_CNT         TEST_PDO_TO_FIFO_ONCE
```

4. After a bunch of settings, move to the “int main_ini()” or “int main()” in STM32 series or Nuvoton series. The first part is to reset and wipe out all the memories. to ensure the code works.

```
uint64_t u64Data;
int nStartFIFO = 0;
int i = 0, DrvIdx = 0, AxisIdx = 0, YasIdx = 0, nCnt1Sec=0, nRunTimeCnt=0;
int nVel = 0, nSumVel = 0, n32DO=0x5555;
int nret = 0, nServoState = 1;
int nLogStart = 0;
int n32CurPos[TEST_AXIS_CNT];
uint16_t u16LastSW[TEST_AXIS_CNT], u16StaWord[TEST_DRV_CNT][N_AXIS_IN_ONE_DRV];
uint32_t u32CycTimeCnt = 0, u32RunTimeCnt = 0, u32LogFifoCnt=0;
uint8_t u8LEDAxis = 0;
uint8_t u8LEDBit = 0;
uint8_t u8Version = 0, u8FifoCnt = 0, u8FifoCntMax = TEST_RXFIFO_CNT;
uint8_t u8State = 0, u8WkcErrCnt = 0, u8CrcErrCnt = 0, u8IsSlvAlive = 0;
uint8_t u8LastState = 0, u8LastWkcErrCnt = 0, u8LastCrcErrCnt = 0;
uint16_t u16RxPDOSize = 0, u16TxPDOSize = 0, u16SpiSize = 0;
int nDriveRxPDOSize = 0, nDriveTxPDOSize = 0;
int8_t SlaveCnt = 0;
RXPDO_ST_DEF_T  *pAllDevRx;
TXPDO_ST_DEF_T  *pAllDevTx;
AXIS_RXPDO_ST_DEF_T *pRxPDOAxis;
AXIS_TXPDO_ST_DEF_T *pTxPDOAxis;
memset(RxPDOData, 0, sizeof(RxPDOData));
memset(TxPDOData, 0, sizeof(TxPDOData));
memset(nPos, 0, sizeof(nPos));
```

5. The EtherCAT master's first command is “ECM_InitLibrary(&u16SpiSize)”. The command will set the SPI data size and return the IC firmware version. The “u16SpiSize” is the size of SPI data. If this parameter is 0 means the system will use the default setting is 112 Bytes.

```
u8Version = ECM_InitLibrary(&u16SpiSize);
```

6. If the checking process passes, the next is EtherCAT initialization “ECM_EcatInit(DCActCode, CycleTime)” and makes EtherCAT state into initialization.
   ECM_EcatInit(DCActCode, CycleTime):
   The value meaning of DCActCode: 0 is disable DC sync、0x300 is activate Sync0、0x700 is activate both Sync0 and Sync1
   The CycleTime unit is ns.
   ※This function is used for all the slaves to get the same DCActCode. If you want various DCActCode, please use ECM_CMD_ECAT_DCSYNC(Command Code: 50) to operate.

```
ECM_EcatInit(DC_ACTIVE_CODE, (BASE_CYCTIME*TEST_CYCTIME_MULTI) / TEST_CYCTIME_DIVID);
```

7. Use “ECM_StateCheck(Slave, ExpectedState, Timeout)” to take the state into the Pre-OP state and start to apply or configure the PDO mapping. If the slaves have provided default PDO structures you need, it will only need to apply the index. But if you want to use custom configuration. You need to take all the objects here to configure the structures. We provide 3 sets of PDO with 8 objects as default to let users configure. After configuration, make sure the “ECM_EcatReconfig()” is applied to reform your structures. You can use “ShowPDOConfig” to print and check your configuration after reconfiguration.
   ECM_StateCheck(Slave, ExpectedState, Timeout)
   If the value of the “Slave” parameter is 0xFF means “all slaves”.
   The “ExpectedState” is the state you want to switch(Pro-OP, Safe-OP, and OP state).
   The “Timeout” is the waiting time to switch objective state. If slaves cannot switch to the next state, try to extend the waiting time.

```
ConfigDrive(1, 0, (TEST_DRV_CNT - 1), 1, N_AXIS_IN_ONE_DRV, 0x1602, 0x1A02);
```

   or

```
RxPDOConfig[i].SmaIdx = RxPDO_ASSIGN_IDX;
RxPDOConfig[i].PDOCnt = 1;
RxPDOConfig[i].MapIdx[0] = RxPDO_MAP_IDX;
RxPDOConfig[i].ObjsCnt[0] = 2;
SetPdoConfTbl(&RxPDOConfig[i], 0, 0, 0x6040, 0, 16); //control word  // 16 bits = 2 bytes for TEST_RXPDO_SIZE
// the 1st parameter is PDO_CONFIG_HEAD
// the 2nd parameter is 0 due to RxPDOConfig.PDOCnt = 1
// the 3rd parameter is 0 for the first object as RxPDOConfig.ObjsCnt[0] = 2
// the 4th parameter is a control word index 0x6040
// the 5th parameter is a sub-index for 0x6040
// the 6th parameter is the bit size for 4th parameter
SetPdoConfTbl(&RxPDOConfig[i], 0, 1, 0x607A, 0, 32); //target position // 32 bits = 4 bytes for TEST_RXPDO_SIZE
TxPDOConfig[i].SmaIdx = TxPDO_ASSIGN_IDX;
TxPDOConfig[i].PDOCnt = 1;
TxPDOConfig[i].MapIdx[0] = TxPDO_MAP_IDX;
TxPDOConfig[i].ObjsCnt[0] = 2;
SetPdoConfTbl(&TxPDOConfig[i], 0, 0, 0x6041, 0, 16); //status word // 16 bits = 2 bytes for TEST_TXPDO_SIZE
SetPdoConfTbl(&TxPDOConfig[i], 0, 1, 0x6064, 0, 32); //actual position // 32 bits = 4 bytes for TEST_TXPDO_SIZE
```

```
ECM_EcatReconfig();
```

8. Next is memory checking “ECM_CheckMEMSpace(TEST_PDO_FIFO_ONCE)”. This command will check all the memories with the commands count fill into a cycle time.

```
ECM_CheckMEMSpace(TEST_PDO_TO_FIFO_ONCE);
```

9. Use “ECM_StateCheck(Slave, ExpectedState, Timeout)” to get into Safe-OP state

```
ECM_StateCheck(0xFF, EC_STATE_SAFE_OP, 1000);
```

10. Use “ECM_StateCheck(Slave, ExpectedState, Timeout)” to get into OP state
11. Apply “ECM_CheckDCStable()” to check DC status.

```
ECM_CheckDCStable();
```

12. Use “ECM_Drv402SM_StateSet(Axes, ServoOn/OffState)” or “ECM_Drv402SM_Enable(Axes, Slaves)” to enable the 402 state machine to servo on state. Here is the way to automatically switch the state by using commands. The manual switch way will show in the next section below.

```
ECM_Drv402SM_StateSet((DrvIdx*N_AXIS_IN_ONE_DRV) + AxisIdx, SERVO_ON_STATE);

PdoExchangeAndGet402State(DrvIdx, AxisIdx, &u8State);
```

or

```
ECM_Drv402SM_Enable(0, 0);
```

13. Use “AlignmentPosition()” to align the motor position with the encoder position. Use “ECM_InitFIFO()” to initial FIFO, use “ClearFIFO(Direction)” to clear all the FIFO memories and use “ECM_EnableFIFO(Enable)” to enable FIFO.
ECM_ClearFIFO(0), 0 means both Tx Rx FIFO
ECM_EnableFIFO(1), 1:Enable、0:Disable

```
ECM_InitFIFO();
ECM_ClearFIFO(0);   //  0 for TX and RX both
PRINTF("ClearFIFO\r\n");
ECM_EnableFIFO(1);  //  Enable FIFO
PRINTF("EnableFIFO\r\n");
```

14. After all the configuration and preparation, the last while loop is the application part to exactly operate the movement of the motors. Here are separate into two parts, if the FIFO count is not exceeded to the maximum, continuing filling new commands, otherwise wait until the space. The application movement can be edited in the file “Utility.c".

Common Error in Sample Code
Common error when you operate the code and some solutions.

Stage	Show	Solution
ECM_Init_Library	wait ASYNC done timeout	Check the connection of the RJ45, the light on the hub is light up correctly
ECM_Init_Library	u8ErrorStatus 0x40	The first operation will pop up this error to mention the SPI data size

		has been modified. Nuvoton series press “enter” to continue
All the time	wait ASYNC done timeout	The waiting time is not enough or slaves show error
All the time	CRC Error	SPI transmission error. Check the SPI master and EtherCAT master IC connection

402 State Machine Automatically Switch and Manually Switch

- In STF4Drive/NuDrive sample code, using “ECM_Drv402SM_StateSet()” and “PdoExchangeAndGet402State()” to servo on the motors
- In others sample code use “ECM_Drv402SM_Enable()” to servo on
  The parameters in “ECM_Drv402SM_Enable()” are axes and slaves No.
  For a drive control a motor, the two drives code will be
  ECM_Drv402SM_Enable(0, 0);
  ECM_Drv402SM_Enable(1, 1);
  For a drive control two motors, the two drives code will be
  ECM_Drv402SM_Enable(0, 0); //the 0th axis, the slave No.0
  ECM_Drv402SM_Enable(1, 0); //the 1st axis, the slave No.0
  ECM_Drv402SM_Enable(2, 1); //the 2nd axis, the slave No.1
  ECM_Drv402SM_Enable(3, 1); //the 3rd axis, the slave No.1
- Manually switch please take STF4HSP or NuHSP as reference shows at the following parts to switch 402 state machines.

```
for(DrvIdx=0;DrvIdx<TEST_SLAVE_CNT;DrvIdx++){
    for(AxisIdx=0;AxisIdx<N_DRV_IN_ONE_SLV;AxisIdx++){
        j = 0;
        while(1){
            nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
            if(nret>0){
                u16LogStatus[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx] = pTxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16StaWord;

                PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx] = pTxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16StaWord & CIA402_SW_STATE_MASK;
                PRINTF("DrvIdx = %d, AxisIdx = %d, state = 0x%x\r\n", DrvIdx, AxisIdx, PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]);

                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_NOTREADYTOSWITCHON){
                    UserDelay(1000);
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_SWITCHEDONDISABLED){
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x6; //Control word: Shutdown = 0x6
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_READYTOSWITCHON){
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x7; //Control word: Switch on = 0x7
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_SWITCHEDON){
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0xF; //Control word: Enable operation = 0xF
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_OPERATIONENABLED){
                    j++;
                    if(j==3){
                        break;
                    }
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_QUICKSTOPACTIVE){
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x0; //Control word: Disable voltage = 0x0
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_FAULTREACTIONACTIVE){
                    UserDelay(1000);
                }
                if(PDOstate[(DrvIdx*N_DRV_IN_ONE_SLV)+AxisIdx]==CIA402_SW_FAULT){
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x0; //Control word: Fault reset = 0x0->0x80
                    nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
                    UserDelay(1000);
                    pRxPDOData->DRIVE_GROUP_0[DrvIdx].HSP[AxisIdx].u16CtrlWord = 0x80;
                    nret = ECM_EcatPdoFifoDataExchange(PDO_FIFO_DEFAULT_CNT, RxData, TxData, u16PDOSize, &u8FifoCnt, &u8WkcErrCnt, &u8CrcErrCnt);
                    UserDelay(1000);
                }
            }
        }
    }
}
```

Maintance Log

Version	Date	Description
01	02.09.2021	Combine ECM-XF-MCU User Guide, Sample Code Explanation, and STM32 Setup Instruction(NUCLEO-F401RE)
	02.17.2021	Modify Nuvoton Setup Instruction description
02	05.06.2021	Update STF4Drive & NuDrive
03	06.28.2021	Update sample code modify instruction
04	06.29.2021	Add function details at modification sequence
05	08.17.2021	Add STM32 H7 pin configuration