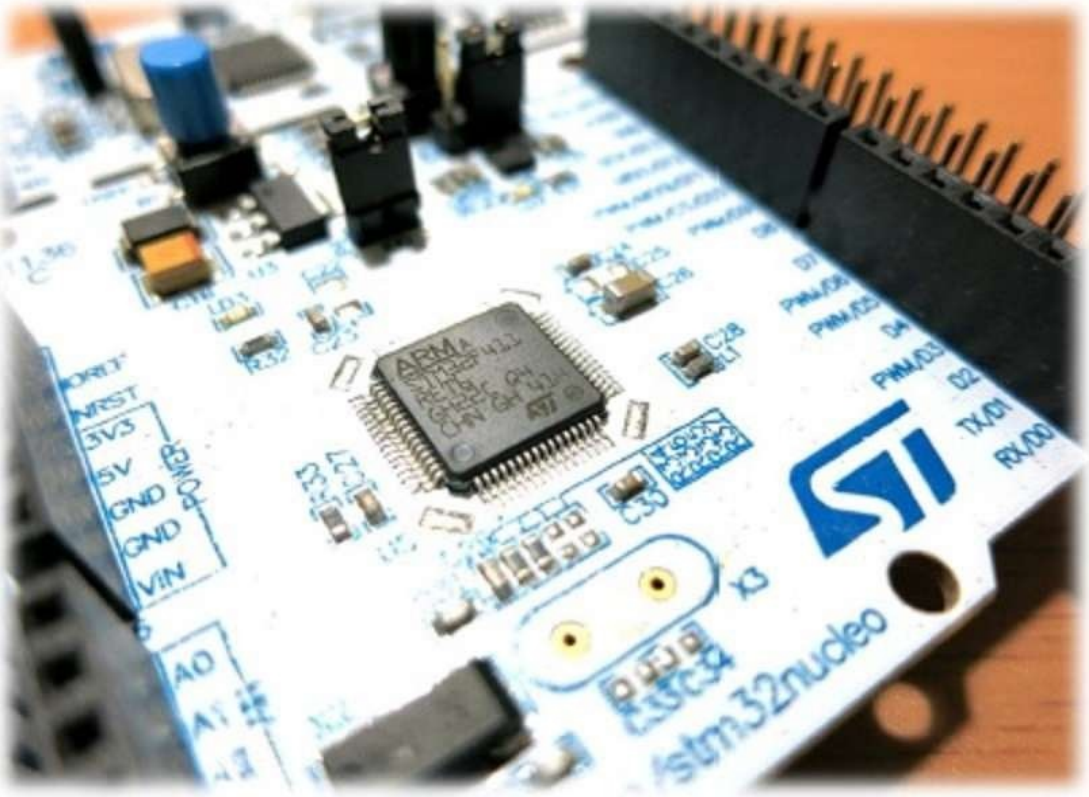
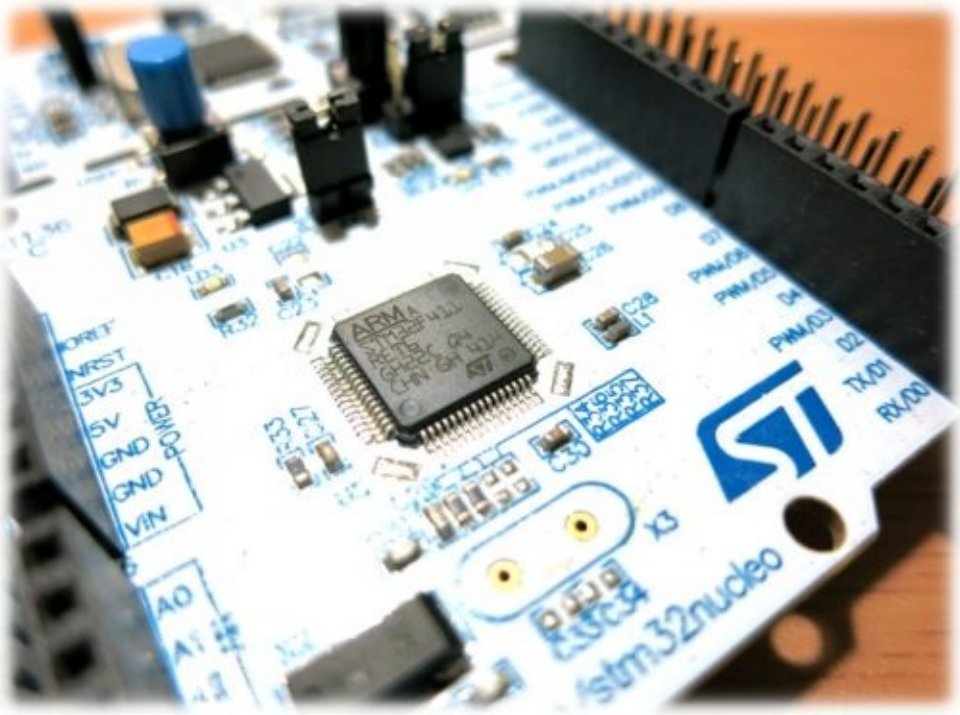
# Introduction to STM32 ARM Microcontroller with STM HAL-Library & SW4STM32




For those who don't want to use mbed.
This book includes explanations about key peripherals and major functions, in addition, building a free ARM offline development environment, automatic program generation CubeMX, FreeRTOS introduction.
This book is optimal for ST microcontroller beginners!

CRESCENT

# Introduction to STM32 ARM Microcontroller with STM HAL-Library & SW4STM32




For those who don’t want to use mbed.
This book includes explanations about key peripherals and major functions, in addition, building a free ARM offline development environment, automatic program generation CubeMX, FreeRTOS introduction.
This book is optimal for ST microcontroller beginners!

CRESCENT

# Index

1. Introduction

A microcontoller that is installed in electronic devices are close to home. Among the microcontollers, the STM32 microcontroller is one of the most popular ARM-based microcontrollers used in various products, with many lineups. However, they are not only complicated and difficult to set up, but also difficult to build integrated development environment. The STM32 microcontroller is difficult for beginner.

In recent years it has become easy to program with simple code without building integrated development environment with online compiler called mbed. However, since mbed is an online compiler, there is resistance to using work codes in the cloud, regardless of play. Meanwhile, at the end of 2015, ST unveiled the free integrated development environment, System Workbench for STM 32 (SW4STM32) , and the offline integrated development environment can be easily used for free.

Although the environment that can use the STM32 microcontoller easily is prepared, books of the ST microcontoller in the world are explained using libraries called STD libraries, and how it is used differs from HAL library promoted by ST in recent years. HAL library is insufficient in information.

With this background, I felt the limits of processing speed of PIC microcontoller that I have used for over 5 years, decided to shift from PIC microcontoller to STM 32 microcontoller in 2016. Based on the experience at that time, I compiled this book as an introduction to STM32 microcontoller.

Unlike general introductory books, we chose the order of chapters rather than every function of microcontoller, in order that engineers who start many built-in systems work. Also, as a target, it is targeted at those who have learned basic operation of PC and can understand C language at minimum.

*Sample code will be released sequentially on the following link.
URL https://github.com/meerstern/stm32introductory

## 2. Environment Development

### 2-1. Download of integrated development environment

System Workbench for STM32 (SW4STM32) can be used free of charge, but user registration is necessary to download. Registration is required separately from the STM account.

* Since the login screen is not https as of March 2017, watch out for using account names and passwords carefully.

URL http://www.openstm32.org/HomePage


After creating an account and logging in, click "System Workbench for STM32" on the left side and click "Downloading the System Workbench for STM32 installer".


Please download the installer according to your environment, 32 bit environment or 64 bit environment.

It can be used not only on Windows but also on Mac OS and Linux.

## 2-2. Install of integrated development environment

This time we will explain about installing on Windows 7 machine.

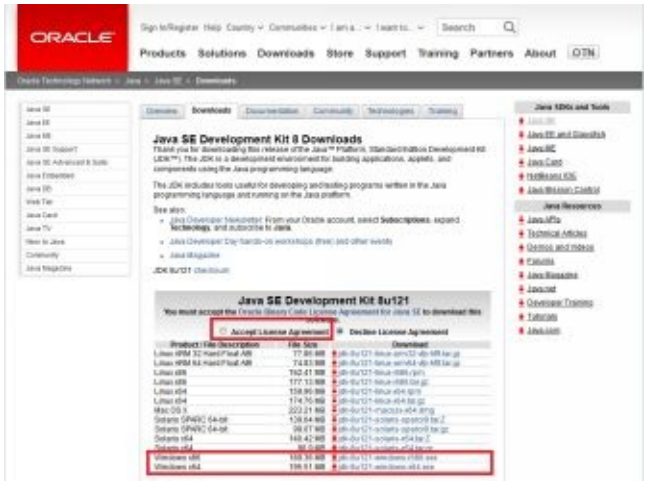
### 2-2-1. Install JavaSE

Since SW4STM32 is an Eclipse-based integrated development environment, installation of JavaSE is required. If JavaSE is already installed, please proceed to the next section. When you started installing the downloaded SW4STM32, installation is stopped automatically when JavaSE is not installed. Then, the installation of JavaSE is urged, but this time JavaSE will be installed first.

URL

http://www.oracle.com/technetwork/java/javase/downloads/index.html


If you can agree after confirming the license information, select "Accept License Agreement" at the top of the download link, the download link becomes effective and downloading becomes possible. Please also select from 32 bits or 64 bits for Windows depending on the environment and download it.



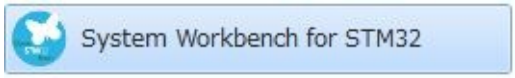

After downloading the installer of JavaSE, click the executable file and install it.

### 2-2-2. Install SW4STM32

Run the install executable file of SW4STM32 with administrator privileges and install it. On the way, the ST-Link driver installation screen will appear. Click "Install" to install the ST-Link driver.

2-3. SW4STM32 Settings

Click SW4STM32 shortcut link and start SW4STM32.




If the error message "Java was started but returned exit code = 13" has appeared and it can not be started up, please associate SW4STM32 with JavaSE as follows. In particular, error seems to occur in Windows 7 environment in many cases.


Java error message example

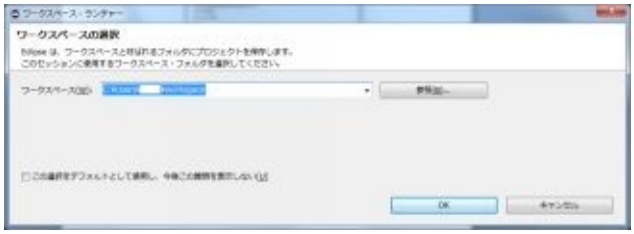
If an error message like the one above appears, if it can not be started, modify the Eclipse configuration file. Open C: ¥ Ac6 ¥ SystemWorkbench ¥ eclipse.ini with notepad etc. and add the following two lines on the first line.

**-vm**

**C:/Program Files/Java/jdk1.8.0_121/bin/ javaw.exe**

※ "jdk1.8.0_121" in the red letter differs depending on the version of JavaSE installed. In the case of Windows, check the version where the directory exists in Explorer etc., and add it by replacing the red letter part as necessary.

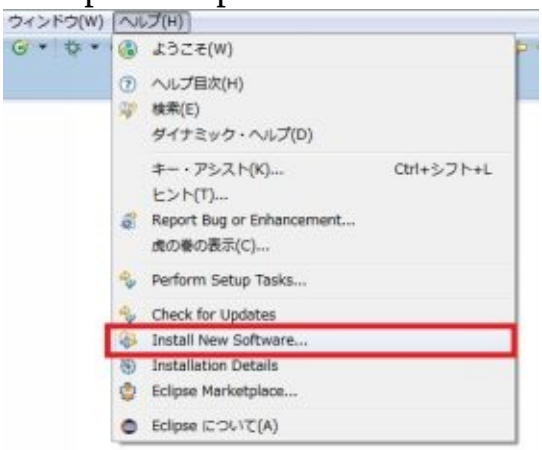
When you start up, the workspace selection screen will be displayed. By default, the "workspace" folder directly under the user folder is the working directory.



2-4. Eclipse Localization

There are several ways to make Eclipse Japanese, but this time we will use Japanese to make it Babel. If you want to use it in English, please proceed to the next section.

Click "Install New Software" from Eclipse "Help".

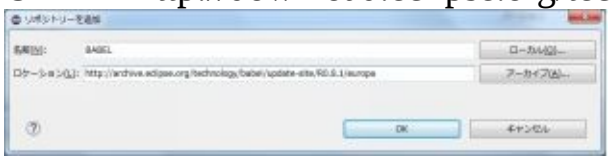



Click "Add" from the installation screen.

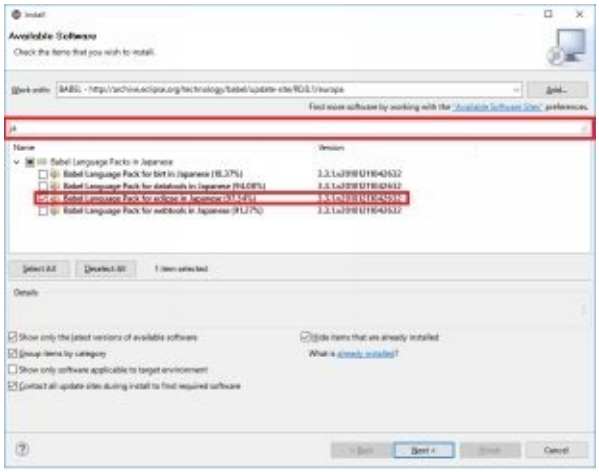
Check the link for the latest Mars from the site of Babel below.
http://www.eclipse.org/babel/downloads.php


Enter "BABEL" for the name, put the latest URL for MARS in the location and click "OK". In this time, the Eclipse is MARS. Please check and use your Eclipse version.

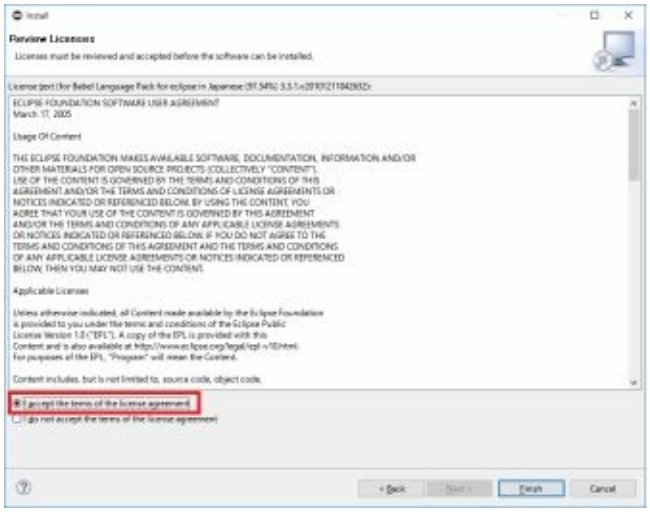
URL http://download.eclipse.org/technology/babel/update-site/R0.14.1/mars



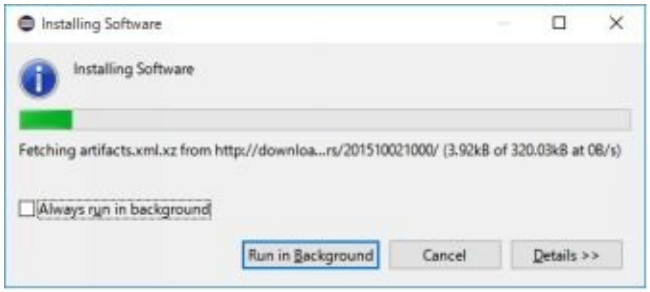
Please patiently wait. A reading progress bar will appear in the lower right bar of the Eclipse application and a list will be retrieved.



When the list is displayed, enter "ja" in the search field and search for Japanese. In that, check "Babel Language Pack for eclipse in Japanese" and click "Next". Check the license, check "accept", and click "Finish".

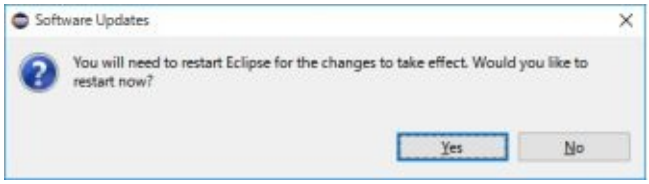


Download and installation will start.




Please check if licensing or security confirmation screen appears again during installation.

When the installation is completed, a confirmation message of software restart will be displayed. Please click "Yes" and restart.

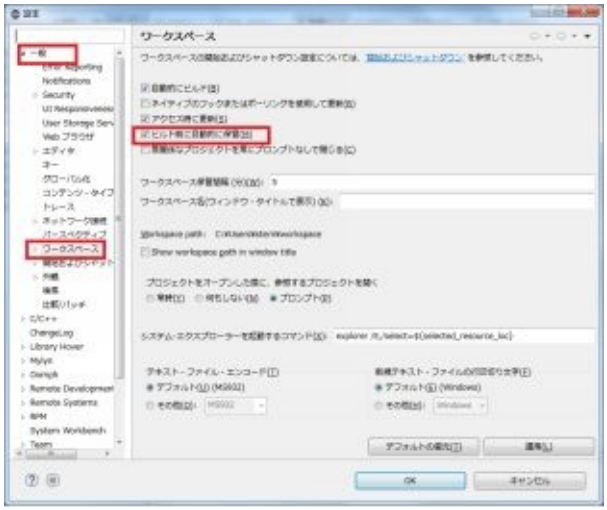



After rebooting, it is completed when it is Japaneseized and started up.

## 2.5. Auto save setting before build

In the default setting, unless you save the changed contents at build time, the change contents will not be changed. By activating the pre-build auto save setting, changes will be automatically saved at build time and you can prevent mistakes due to forgetting to save.

Check "Save automatically before build" in "Window" → "Settings" → "General" → "Workspace" and click "Apply".



## 2-6. Add Explorer Link

By default Eclipse does not have an explorer link to access files located locally from the project. Follow the procedure below to add a link.

Choose Execute → External Tools → Configure External Tool. Double click on the program and add a new one.

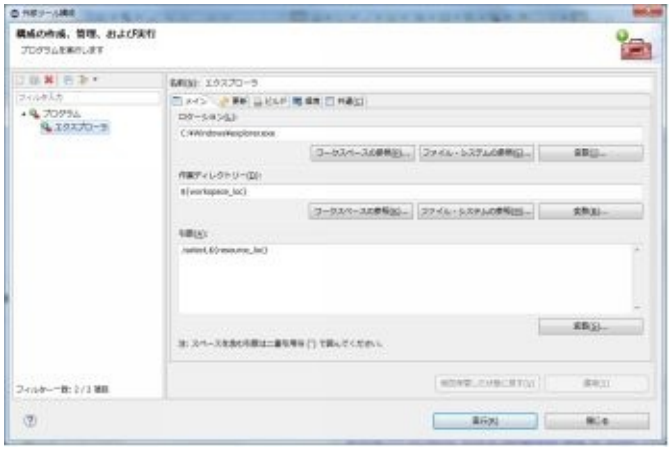
Name: Explorer

Location: C: ¥ Windows ¥ explorer.exe

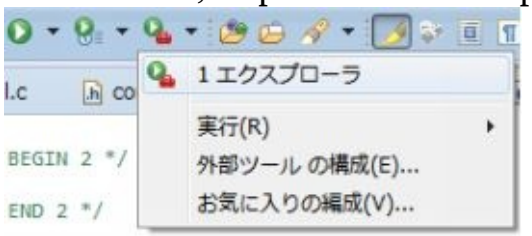
Working directory: $ {workspace_loc}

Arguments: / select, $ {resource_loc}

Uncheck "Build before launch" on the "Build" tab and uncheck "Assign to console" on standard input / output of "Common" tab.



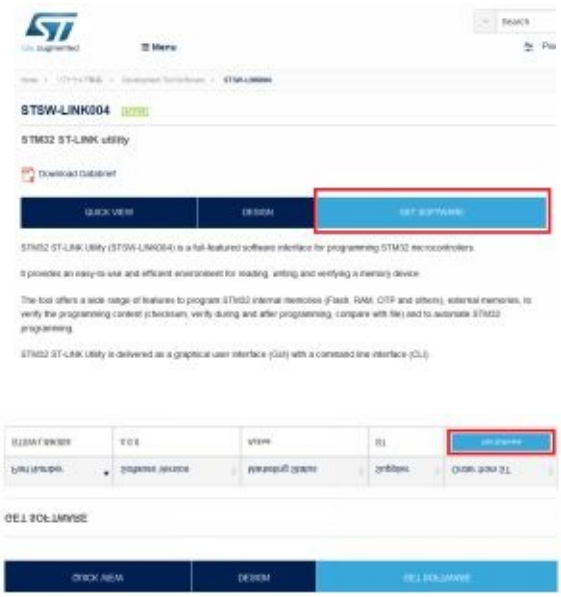
When you click the External Execution button, Explorer will be displayed.



2-4. Download firm writing software

Download the writer software ST-Link Utility for writing to the STM32 microcontoller. If you have not created an STM account, please create it before downloading.

URL http://www.st.com/ja/embedded-software/stsw-link004.html



2-5. Install firm writing software

Run the installation executable of ST-Link Utility with administrator privileges and install it. On the way, the ST-Link driver installation screen will appear. Click "Install" to install the ST-Link driver.

## 2-6. Download CubeMX

CubeMX is software (board support package, BSP) that automatically generates the minimum necessary code for the microcontoller. Click "Get Software" from the URL below to download the software.

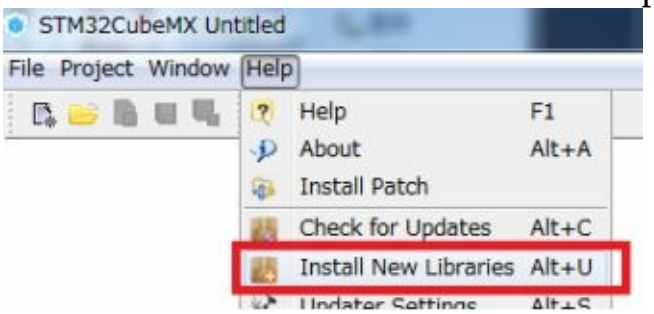
URL
http://www.st.com/ja/development-tools/stm32cubemx.html

2-7. install CubeMX

Run the installation executable file with administrator privileges and install it.

2-8. Install CubeMX Package

In order to automatically generate the code with CubeMX, it is necessary to have a package of the corresponding ST microcontoller series. Launch CubeMX and click "Help" → "Install New libraries".




Packages available for each series of ST microcontoller are displayed. Although older versions of packages are also available, check the latest version of the package and click "Install Now", especially for no reason.

Because the board to be used at this time is STM32F4 and STM32F3, check the latest version of each and click "Install Now".



Check the required packages and click "Install Now" to automatically install the packages from the Internet.

However, if there is no network environment or the network is not stable, it is possible to add the package offline. Download the package of STM32F3 and STM32F4 from "Obtaining software" at the following URL.

URL http://www.st.com/ja/embedded-software/stm32cubef3.html

http://www.st.com/ja/embedded-software/stm32cubef4.html

Launch CubeMX, click "Help" → "Install New libraries", click "From Local", select the downloaded zip file, and offline package installation starts.

## 2-9. Information CubeMX Package

If you add CubeMX package, you can use CubeMX to generate code automatically, you can also use related libraries such as sample code, FAT library, USB library, FreeRTOS and so on.

The CubeMX library is installed by default in the following directory. Please refer to the sample code etc. by all means.

C: ¥ Users ¥ (user name) ¥ STM32Cube ¥ Repository

# 3. About STM32 Microcontroller

## 3-1. STM32 Microcontroller Selection

Although the ST selling the STM 32 microcontoller also sells other 8-bit microcontollers, we will explain about the STM 32 microcontoller because the mainstay is STM 32 which is an ARM-based 32-bit microcontoller.

STM32 is a processor for built-in called CortexM among ARM processors, and the series differs from Cortex A used for smartphones and others. Cortex A is an ARM processor that exceeds 1 GHz because it is used with OS and applications. Meanwhile, CortexM is embedded in appliances and automotive equipment, so it's low cost and low power consumption.

Among the Cortex M series, it is further divided into M0, M3, M4, M7, H7, but in STM it is divided into three types of STM 32F series, STM 32L series, and STM 32H series according to the application. The STM32F series is for general use, the STM 32L series for low power consumption applications, and the STM32H series for high performance high end use.

In addition, since the code using the HAL library described this time is very portable, it is possible to use almost the same function in different series even though it is very portable. Therefore, it is also possible to use different series depending on the application.

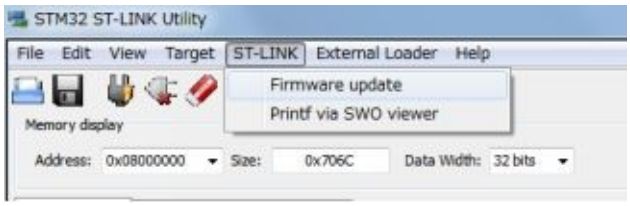
## 3-2. ST Microcontroller Introduction

Nucleo board is recommended to start STM32 microcontoller. In this document, we explain using Nucleo board using STM 32 F 411 of STM 32 F 4 and STM 32 F 303 of STM 32 F 3 among STM 32 F series, which is inexpensive and easy to obtain at the Internet and electronic parts stores.

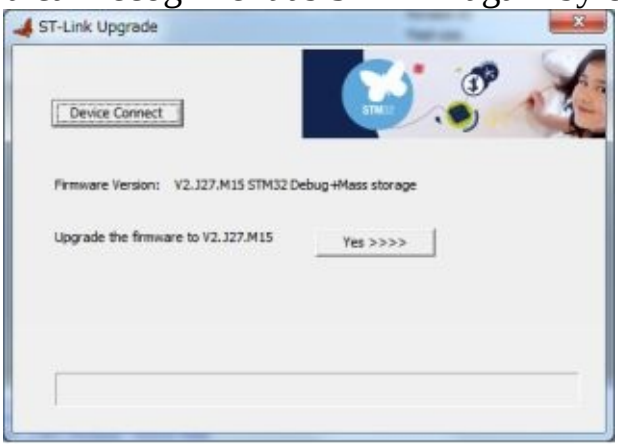
3-3. After Getting Nucleo Board

First, check the version of firmware on ST-Link with ST-Link Utility, and make sure to update if version is old. The initial LED program also does not work properly wheh the firmware version of ST-Link is old. It moved smoothly if it was the latest farm.

Connect the Nucleo board and the PC with the USB cable and start the ST-Link Utility. Select "Firmware update" from "ST-Link" on the menu bar.




Click "Device Connect", and if the firmware version of ST-Link is old, click "Yes >>>>" and update the firmware. After updating, you can recognize it as ST-Link again by connecting the USB cable again.



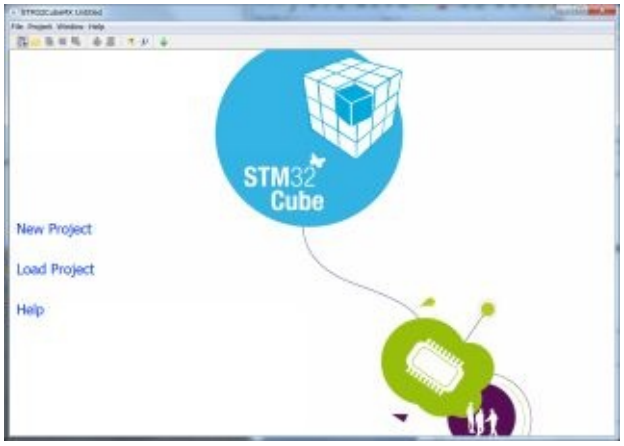
# 4. Lighting LED

## 4-1. Auto Code Generation with CubeMX

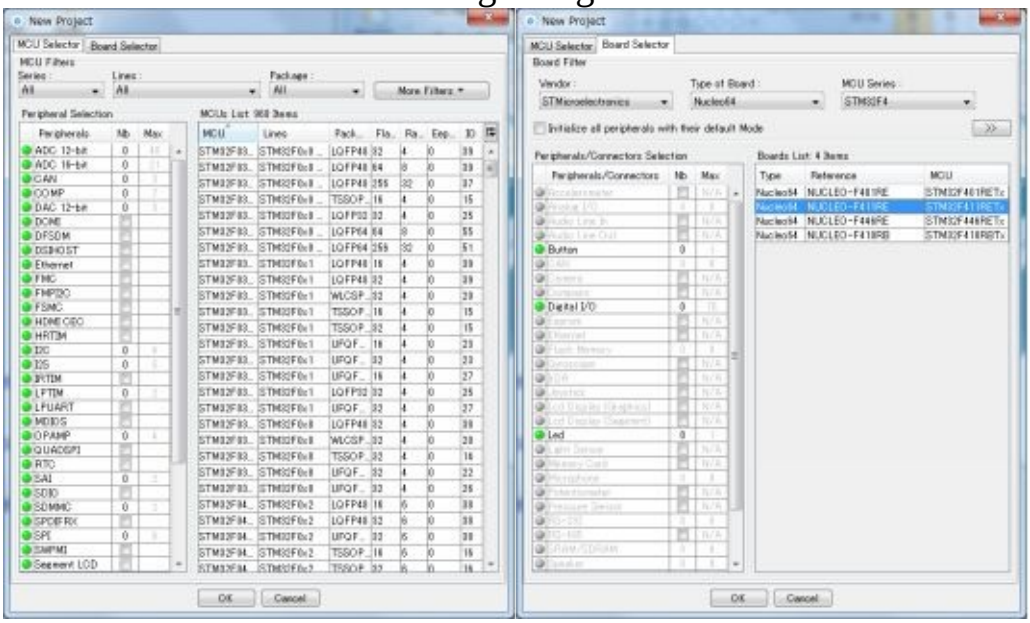
Writing a microcontoller port and peripheral initialization code from 0 makes it very painful. CubeMX is a tool to automate it. If you use this, the development is very easy. Since code such as initial setting is generated automatically, it is possible to drastically reduce the trouble of writing mistakes and setting codes.

### 4-1-1. Board Selection

Click "New Project" to display the microcontoller selection screen.




Since we are using the STM 32 F 411 Nucleo board this time, click on the "Board Selector" tab above and select the STM 32 F 411 Nucleo board. The tab of "MCU Selector" is used for incorporating a microcontoller as a standalone board into a self-made board or an embedded board without using the Nucleo board. Even if you select the microcontoller on the Nucleo board, it is almost the same, but in the case of the Nucleo board, it is better to select "Board Selector" because the LEDs on the Nucleo board and peripheral circuits are defined from the beginning.



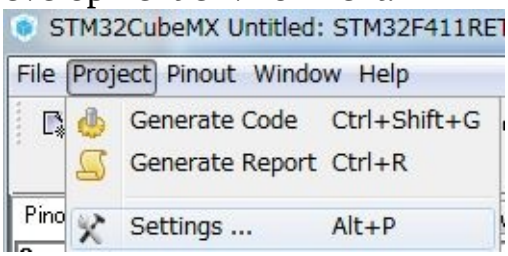

When selecting the STM32F411 Nucleo board Type of Board "Nucleo 64", MCU Series "STM 32 F 4" When selected, the board is narrowed down and displayed. For this time, select the second "NUCLEO - F411RE" from the top and press "OK".

### 4-1-2. Setting of CubeMX

GreenLED is already defined on the PA5 pin of the middle microcontoller. Therefore, in the case of LED lighting, that is all.

Click "Project Settings" and set the project name and project settings for development in the

development environment.




Project name "NucleoF 411 - LED" was selected. The default folder is set to "workspace" in the user folder. Toolchain / IDE select "SW4STM32".

Although it is not a problem when generating code with CubeMX only for the first time, since the added code sometimes disappears when re-generating the code with CubeMX, from the "Code Generator" tab Generated Files "Backup check the "previously generated files" and set up to take a backup. When you press "OK" button, code will be generated automatically.

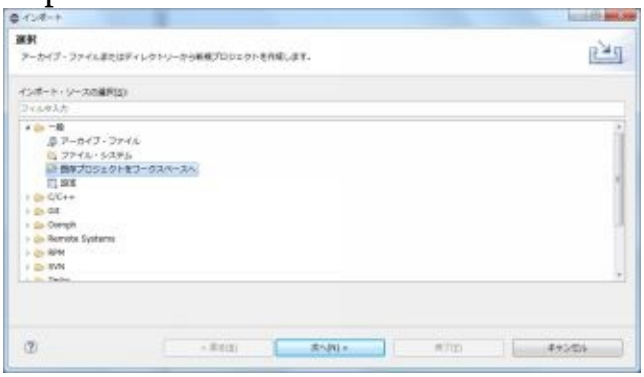
Click "Generate project report files" to output the current settings as PDF data. Finally, save the setting of CubeMX by pressing save button. Make sure that the automatically generated code is in the project folder.

Default directory
C: ¥ Users ¥ (user name) ¥ workspace ¥ NucleoF 411 – LED

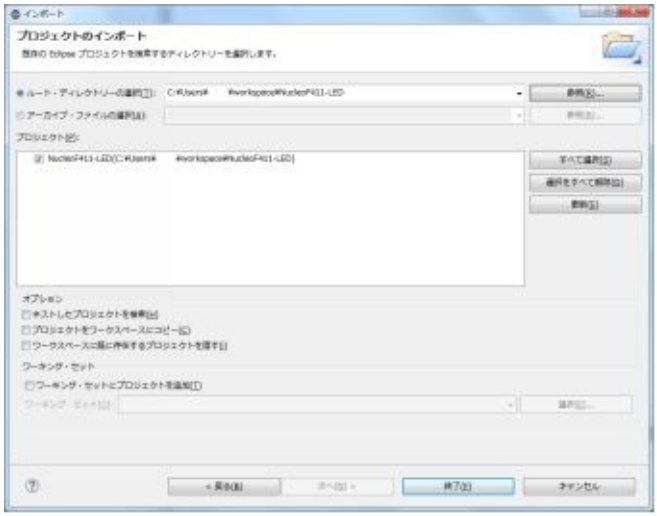
4-2. Import Project to SW4STM32

Add projects automatically generated by CubeMX to SW4STM32. Select "Import" from "File".


Select "Existing project to workspace" and click "Next".

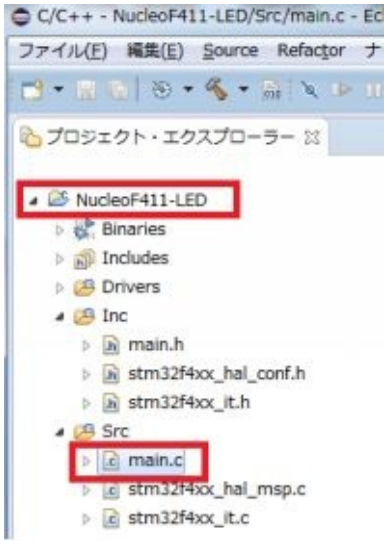


Click "Browse", select the folder "NucleoF 411-LED" of the project that was generated earlier, and click "OK". The project is automatically recognized and the "Project" column is added. Click "Finish" to complete the import.



4-3. Coding

When you click on a project folder added in the Project Explorer, there is a source code automatically generated by CubeMX. Among them, I write additional code in the main.c file and I will do LED lighting.




Click main.c to open the editor screen and edit the code. As for the code automatically generated by CubeMX, the part to be edited by the user is predetermined as follows.

If you add comments to other places or delete comments, CubeMX overwrites it when you generate the code again, and the edited part is deleted. When IO setting or peripheral functions are added or deleted later, they may be generated again by CubeMX. At that time, if you fill in the decided part in advance, the code will remain when you generate it again with CubeMX and you can continue coding as it is.

```
/* USER CODE BEGIN XX */
Write Your Code Here!
/* USER CODE END XX */
```

Let's add the code of LED lighting soon. Add the following code below "USER CODE BEGIN 3" in the main.c file around 90 lines.

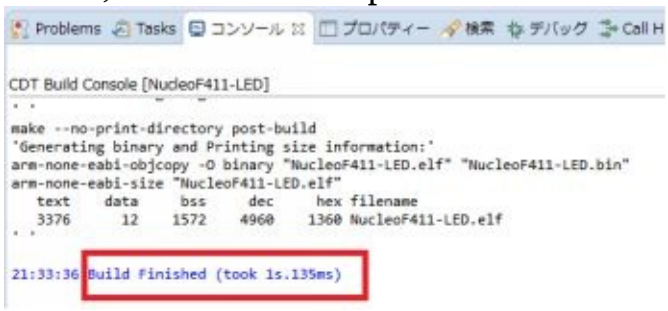
```
  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin);
HAL_Delay(100);
HAL_GPIO_WritePin(LD2_GPIO_Port,
                  LD2_Pin,
                  GPIO_PIN_SET);
HAL_Delay(100);
HAL_GPIO_WritePin(LD2_GPIO_Port,
                  LD2_Pin,
                  GPIO_PIN_RESET);
HAL_Delay(100);
```

```
}
  /* USER CODE END 3 */
```

The LED on the Nucleo board has already been added as LD2. HAL_GPIO_TogglePin is a function that inverts OFF when IO is ON and ON when it is OFF. The first argument is the GPIO group and the second argument is the pin number in the group. HAL_Delay is a function to wait the number ms specified by the argument. HAL_GPIO_WritePin is a function to set IO ON / OFF. The first argument is the GPIO group and the second argument is the pin number in the group. Set the third argument ON or OFF.

For definitions and functions in HAL, you can jump to the definition part by clicking the function or definition character string while holding down "Ctrl". Let's check how the LD2_Pin and GPIO_PIN_SET are defined by the jump function. Alternatively, you can enter a function name halfway and press "Ctrl" + spacebar to display the candidate functions.

After adding code, start building with "Ctrl" + "B". When "Build Finished" is displayed on the console screen, the build is completed. If an error occurs, check the error contents and correct the code.



4-4. Write Firmware

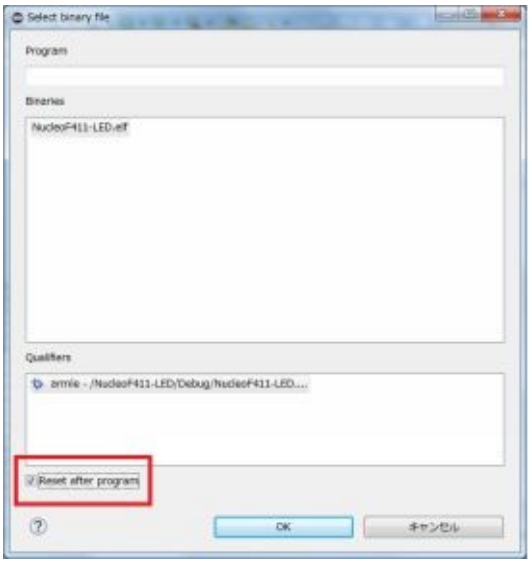
There are some types of microcontoller writing method for Nucleo board. It is a method using the mbed function and a method using the ST-Link Utility. It is easy and easy to use the mbed function, but the microcontroller is limited only to mbed-compatible microcontollers.

4-4-1. Write from Eclipse

Connect the USB cable to the PC to the USB terminal of the Nucleo board to be written.

Right-click the project folder of Project Explorer and select "Target" → "Program Chip".

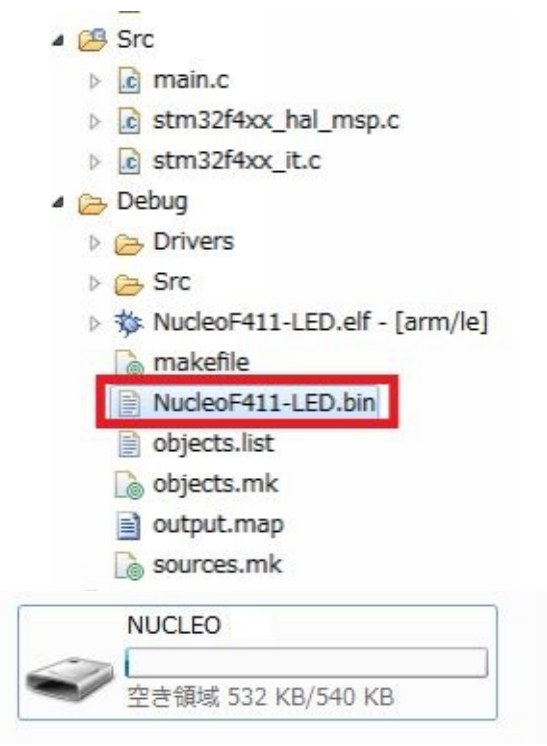

Check "Reset after program" and click "OK". Writing is started, the LED flashes as it works.




4-4-2. Write with mbed function

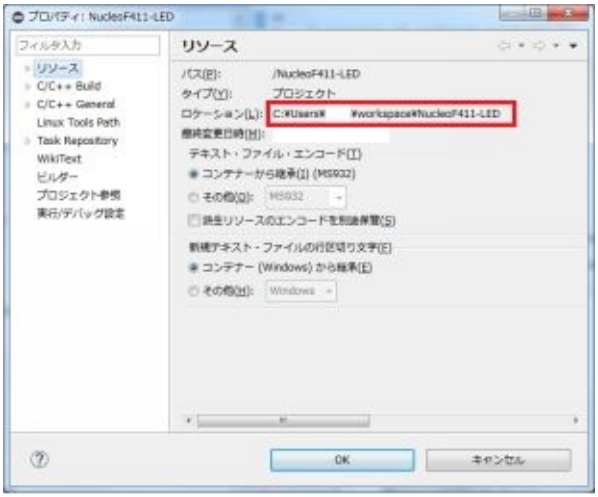
When the build is completed, a compiled Bin file is created in the Debug folder. In this example, NucleoF 411 - LED.bin file is generated. Also connect the USB cable to the PC. After building is completed, right-click the NucleoF 411 - LED.bin file in the Project Explorer and choose Copy.

When Nucleo board is recognized normally, it is recognized as USB memory, paste the previously copied NucleoF 411 - LED.bin file just under the recognized drive. Writing is automatically done to the microcontoller, and the LED flashes as it works.

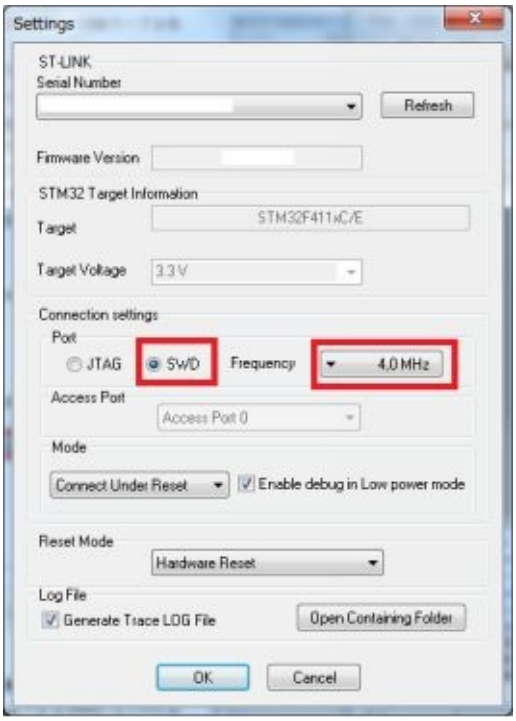



4-4-3. Write with ST-Link Utility

When the build is completed, a compiled . Bin file is created in the Debug folder. In this example NucleoF 411 - LED.bin file is generated. Also connect the USB cable to the PC to the USB terminal. Right-click the project in the Project Explorer and select Properties. Click "Resource" in the Properties window and check the directory of the file.




Activate ST-Link Utility. Click "Target" → "Settings" and confirm that the write setting is SWD.




Click the "Program & Verify" icon.


Click "Browse" and select NucleoF 411 - LED.bin in the "Debug" folder of the directory you confirmed earlier. Click "Start" button to execute the writing. When it works, the LED on the Nucleo board flashes.

# 5. Debug

Debugging of the microcontoller is possible in real time using OpenOCD and ST-Link installed together with SW4STM32. There is another way to debug using UART and printf function.

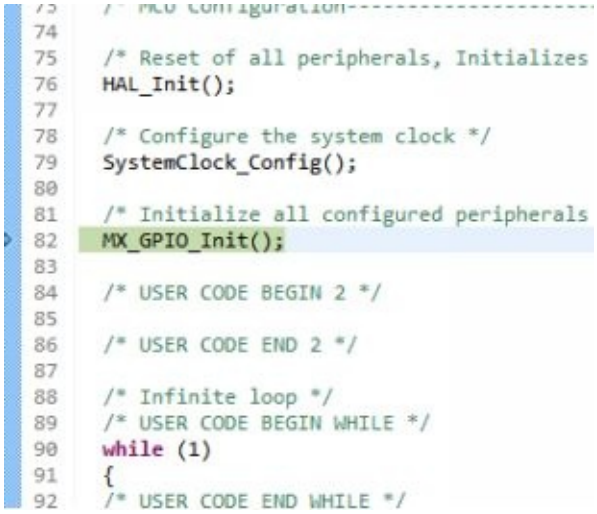
5-1. Debug with OpenOCD

By using OpenOCD, it is possible to debug in real time step by step while confirming the value of variable and processing.

Add a project to OpenOCD. Click the pull-down menu to the right of the debug button and select "Debug" -> "Ac6 STM32 C / C ++ Application".

It automatically switches to the debug screen and debugging is started. If an error occurs, disconnect the ST-Link Utility. If you still get an error, disconnect the Nucleo board's USB cable and connect it again.


When debugging is started, a part of the code is selected green as follows, you can step in with the "F5" key, step out with the "F6" key, and resume with "F8". Also, if necessary, you can double-click the left part of the line number to make a break.




To stop, click the stop button.

To return to the project screen from the debug screen, click the "C / C ++" button.

5-2. Debug with UART

For debugging using UART, serial output is performed using the UART port on the microcontoller. In the case of the Nucleo board, the output destination is recognized as a COM port when the USB terminal is connected to the PC, so it can be confirmed as a character string by software such as teraterm. It can also be imported from the UART port on the Nucleo board using the USB serial conversion adapter.

5-2-1. CubeMX Setting

If you want to use the already created project as it is, open the * .ioc file in the project. In order to use TX, RX of Arduino terminal of Nucleo board this time, set port of USART 2 of "Pinout" tab from "disable" to "Asynchronous" in case of STM32F4 NUCLEO. For the NUCLEO board of STM 32 F 3, set UART 1 to "Asynchronous".


Click the "USART 2" button on the "Configuration" tab and make detailed settings. Select "9600 bps" for "Baud Rate" and 8 bits for "Word Length".


Also, when using the STM32F3 series UART with CubeMX please pay attention to Word Length setting in UART setting. In CubeMX default setting, it is 7bit, and in order to perform general UART communication it needs to be 8bit.

After setting, clicking the Generate button on the CubeMX menu bar will automatically generate the code reflecting this setting.

5-2-2. Add Code

Add the following function to "USER CODE BEGIN 0" in main.c.

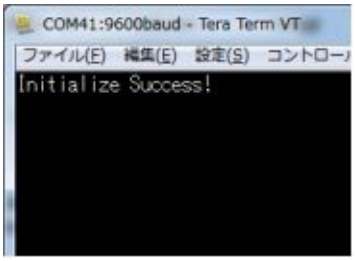
```
/* USER CODE BEGIN 0 */
```

```
int _write( int file, char *ptr, int len )
{
  HAL_UART_Transmit(&huart2,
                    (uint8_t*) ptr,
                    len, 1000);
  return len;
}
/* USER CODE END 0 */
```

Add the above function and add the following printfoutput to the "USER CODE BEGIN 2" part before the while (1) in the main function.

```
/* USER CODE BEGIN 2 */
  printf("Initialize Success!¥n");
/* USER CODE END 2 */
```

After adding code, after building with "Ctrl" + "B", write to the microcontoller. In addition, you can confirm that it is output by opening the COM port of NUCLEO board at 9600 bps with teraterm and pressing the reset button on the board.




5-2-3. Use of Floating Point

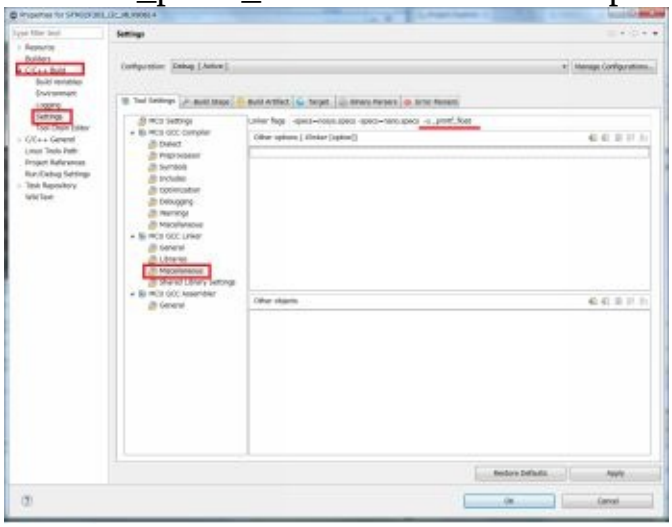
To use floating point in printf, project setting of SW4STM32 is required separately.

Right-click the project folder displayed in the Project Explorer and select Properties. You can also select the project folder and display it with "Alt" + "Enter".

Linker flags of "C / C ++ Build" → "Settings" → "Tool Settings" → "MCU GCC Linker" → "Miscellaneous"

"-specs = nosys.specs -specs = nano.specs"

Add "-u _printf_float". Please set to "-specs = nosys.specs -specs = nano.specs -u _printf_float".



After the above setting, if you build again, the setting will be reflected.

# 6. SPI Communication

There is SPI (Serial Peripheral Interface) as a serial interface used for communication of sensors, memories, SD cards, peripheral devices such as AD converters and DA converters. The SPI consists of four signal lines (CS: Cable Select, SCK: Serial Clock, MISO: Master In Slave Out, MOSI: Master Out Slave In). Multiple devices can be connected by preparing CS for each peripheral device and sharing other SCK, MOSI, MISO. A typical SPI device can communicate with that device by pulling CS low. Therefore, it is possible to communicate by sending / receiving a signal from SCK, MISO, MOSI by setting CS of CS to HIGH and CS to LOW only when you want to communicate.

The STM32 microcontoller has an SPI controller built in. Although the SPI controller has master mode and slave mode, this time the general STM32 microcontoller is in master mode and the method of reading and writing the SPI connected peripheral device as a slave will be explained.

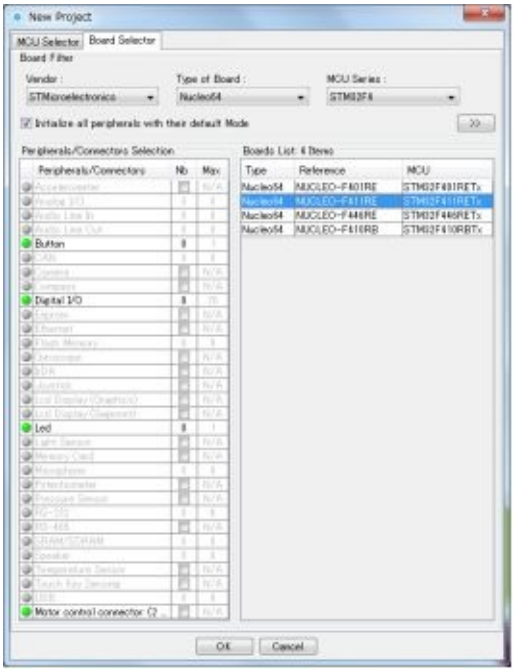
6-1. Code Generation with CubeMX

6-1-1. Board Select

Start CubeMX. Select the board you have. This time we will explain about using NUCLEO - F 411RE.

Click New Project and select "Board Selector" tab from the "New Project" screen.

Select "Nucleo 64" for Type of Board and "STM 32 F 4" for MCU Series. When the types of boards are narrowed down, select "NUCLEO - F411RE" and click the "OK" button.




6-1-2. CubeMX Setting

In the case of NUCLEO board, SPI 1 is invalid and it can not be selected because the SCK / D 13 pin used in the SPI is assigned to the LED port LD 2 in its initial state. Set the assignment of LD2 in the right pin assignment to "Reset_State" and release it, then select "Full-Duplex Master" with SPI 1 in the left column to activate it.

Click PA5 to which LD2 of the port diagram is assigned, and select SPI1_SCK from GPIO_Output.

This time we will send and receive SPI 8bit data. If the slave device is 16 bit data, it is possible to set it to 16 bit in addition to sending and receiving 8 bit data twice.

Change the pin of CS to "GPIO_Output".

6-1-3. SPI Wiring

Connect four signal lines (CS: Cable Select, SCK: Serial Clock, MISO: Master In Slave Out, MOSI: Master Out Slave In) for the SPI communication to the slave device. Connect the power supply VDD and ground GND separately from the signal line to the slave device. Since the port used in the SPI is 5 V tolerant, it is possible to connect the slave device directly without a voltage level conversion circuit even if it is a 5 V power supply.

STM32F411RE Port written as FT of item of Pin definitions in datasheet is 5V tolerant.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| - | 19 | - | 28 | - | VDD | I | FT |
| 14 | 20 | G6 | 29 | M3 | PA4 | I/O | FT |
| 15 | 21 | F5 | 30 | K4 | PA5 | I/O | FT |
| 16 | 22 | F4 | 31 | L4 | PA6 | I/O | FT |
| 17 | 23 | F3 | 32 | M4 | PA7 | I/O | FT |

6-1-4. SPI Coding

Functions of SPI communication use HAL_SPI_TransmitReceive. The first argument passes the SPI structure in "Private variables" around the first 40th line. The second argument passes the array pointer of the transmission data as. The third argument passes the array pointer of the received data. The fourth argument passes the data size. The fifth argument is the timeout time.

```
HAL_SPI_TransmitReceive(
        SPI_HandleTypeDef *hspi,
        uint8_t *pTxData,
        uint8_t *pRxData,
        uint16_t Size,
        uint32_t Timeout)
```

For SPI send only, use HAL_SPI_Transmit. The first argument passes the SPI structure. The second argument passes the array pointer of the transmission data. The third argument passes the data size. The fourth argument is the timeout time.

```
HAL_SPI_Transmit(
        SPI_HandleTypeDef *hspi,
        uint8_t *pData,
        uint16_t Size,
        uint32_t Timeout)
```

When receiving only with SPI, use HAL_SPI_ Receive. The first argument passes the SPI structure. The second argument is the array pointer of the received data. The third argument is the data size. The fourth argument is the timeout time.

```
HAL_SPI_Receive(
        SPI_HandleTypeDef *hspi,
        uint8_t *pData,
        uint16_t Size,
        uint32_t Timeout)
```

Here is an example of description when reading the magnetic encoder AS5048A made by AMS

throughSPI.

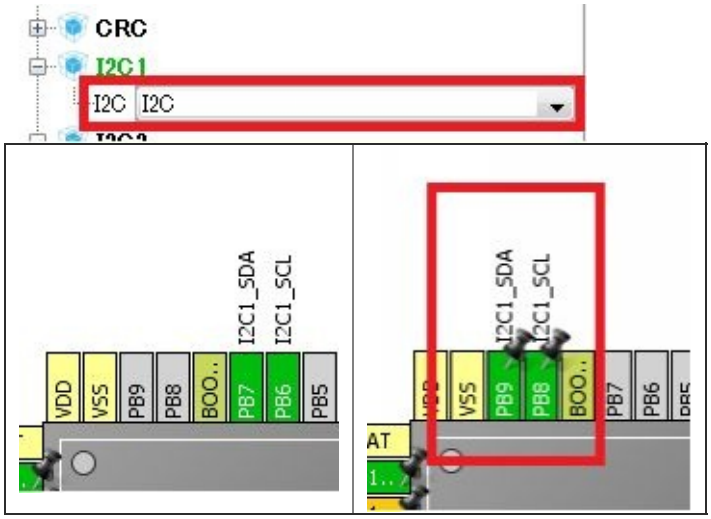
```
                    uint16_t sData[2];
                    uint16_t rData[2];
                    uint16_t res;

                    //CS LOW
                    HAL_GPIO_WritePin(ENC_CS1_GPIO_Port,
                    ENC_CS1_Pin,0);
                    //Make send data
sData[0]=0xFF;
sData[1]=0xFF;
HAL_SPI_TransmitReceive(&hspi1,
                                sData,
                                rData,2,100);
//Change to 16bit data
res =(rData[0]<<8)+rData[1];
res = 0x3FFF&res;
//CS HIGH
                    HAL_GPIO_WritePin(ENC_CS1_GPIO_Port,
                    ENC_CS1_Pin,1);
```

# 7. I2C Communication

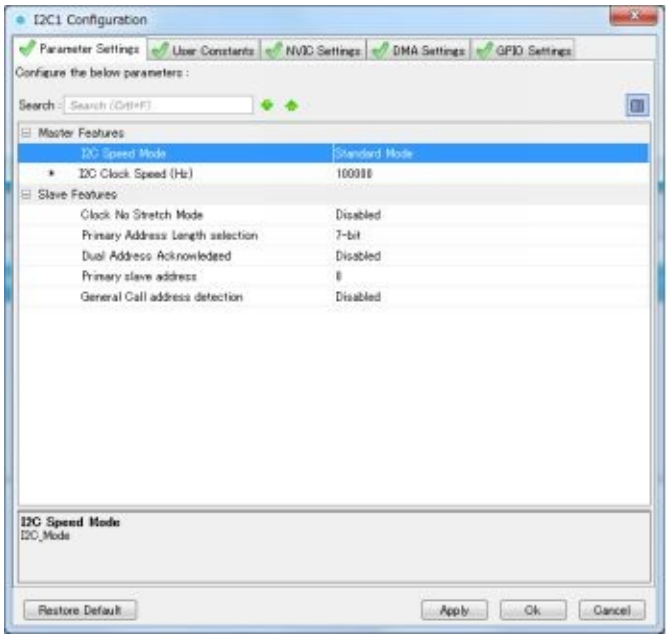
As with the SPI communication, I 2 C (Inter Integrated Circuit) is available as a serial interface used for sensors and others. I2C consists of two signal lines (SCL: Serial Clock Line, SDA: Serial Data Line). The clock is transmitted by the master device (this time STM 32 is the master), and the data is transmitted and received with one signal line. I 2 C has a device address for each device, and it communicates with slave address + data during communication. Unlike SPI communication, there is no CS. The slave address is specified and the device with the matched address responds.
The STM32 microcontroller has an I2C controller built in. Although the I2C controller has master mode and slave mode, this time the general STM32 microcontoller is in master mode and the method of reading / writing the I2C connected peripheral device as a slave will be explained.

## 7-1. CubeMX Setting

Set the port of I2C1 in the "Pinout" tab from "disable" to "I2C". I2C1 is automatically assigned to the right pin assignment setting screen. Although it does not matter, since it is different from the Arduino port of Nucleo board, click "PB9" on the pin assignment setting screen and select "I2C1_SDA". Likewise, click "PB8" and select "I2C1_SCL". With this change, SCL and SDA of Arduino port of NUCLEO board can be used as I2C1.




Click the "I 2 C 1" button on the "Configuration" tab to make detailed settings. We will use it as it is in the initial setting this time. If necessary, change the I2C Speed Mode etc. of "Parameter Settings" in the detailed setting. Also, when you check "GPIO Settings" in the detailed setting, if you enable I2C, Pull-up inside the microcontoller is enabled by default. Therefore, it is possible to omit the external pull-up resistor required for I2C. However, depending on the device, the internal pull-up resistor is weak, and if it can not communicate, an external pull-up resistor is required separately.


I2C1 Configuration
Parameter Settings
User Constants
NVIC Settings
DMA Settings
GPIO Settings
Configure the below parameters :
Search :
Search (Ctrl+F)
Master Features
I2C Speed Mode
Standard Mode
I2C Clock Speed (Hz)
100000
Slave Features
Clock No Stretch Mode
Disabled
Primary Address Length selection
7-bit
Dual Address Acknowledged
Disabled
Primary slave address
0
General Call address detection
Disabled
I2C Speed Mode
I2C_Mode
Restore Default
Apply
Ok
Cancel

## 7-2. I2C Wiring

Connect two signal lines (SCL: Serial Clock, SDA: Serial Data) for I2C communication to the slave device. Connect the power supply VDD and ground GND separately from the signal line to the slave device. It is possible to omit the external pull-up resistor by enabling the internal pull-up. However, depending on the device, the internal pull-up resistor is weak, and if it cannot communicate, an external pull-up resistor is required separately. Connect a resistance of about 4.7 kΩ to SCL and SDA, respectively, and pull up at 3.3 V.

7-3. I2C Coding

Since I2C communication exchanges bidirectionally with one data signal line, coding is generally more complicated than SPI. In the HAL library, functions that can easily handle complicated processes peculiar to I2C are prepared, so it is very easy to communicate with I2C.

When communicating with the I2C slave device, specify the address of the slave. Slave address such as device data sheet is written with 7 bit length excluding R / W bit. On the other hand, the I2C slave address used in the function argument of the HAL library is passed as 7 bits + R / W 8 bits. Therefore, for example, when the 7 bit address is 0x5 A, be careful to specify the address used in the HAL library as (0 x 5 A << 1) as the 8 bit address including the R / W bit. However, since the value of the R / W bit changes at the time of reading and writing in HAL functions, it is unnecessary to change the address.

To write data to the slave device, use the HAL_I2C_Mem_Write function. The first argument passes the I2C structure. It specifies the slave address as the second argument, the register address of the data to be written to the third argument, the register address length as the fourth argument, the data to be written to the fifth argument, the data length to write to the sixth argument, and the timeout time as the seventh argument.

```
HAL_I2C_Mem_Write(
        I2C_HandleTypeDef *hi2c,
        uint16_t DevAddress,
        uint16_t MemAddress,
        uint16_t MemAddSize,
        uint8_t *pData,
        uint16_t Size,
        uint32_t Timeout)
```

To read data from the slave device, use the HAL_I2C_Mem_Read function. The first argument passes the I2C structure. It specifies the slave address as the second argument, the register address of the data to be written to the third argument, the register address length as the fourth argument, the data to be written to the fifth argument, the data length to write to the sixth argument, and the timeout time as the seventh argument.

```
HAL_I2C_Mem_Read(
        I2C_HandleTypeDef *hi2c,
        uint16_t DevAddress,
        uint16_t MemAddress,
        uint16_t MemAddSize,
        uint8_t *pData,
        uint16_t Size,
        uint32_t Timeout)
```

Below is a code example when actually reading the temperature of the object from infrared temperature sensor MLX90614.

```
#define MLX90614_I2CADDR (0x5A<<1)
#define MLX90614_TOBJ1 0x07
#define MLX90614_TIMEOUT 50
```

```
uint8_t ret[3];
int16_t res;
int16_t temp;
```

```
float tempf;
HAL_I2C_Mem_Read(
                &hi2c1,
                MLX90614_I2CADDR,
                MLX90614_TOBJ1,
                I2C_MEMADD_SIZE_8BIT,
                (uint8_t*) ret,
                3,
                MLX90614_TIMEOUT);
res=ret[0]+(ret[1]<<8);
temp=res;
temp=2*temp;
temp=temp-27315;
tempf =temp/100.0f;
printf(“ObjectTemp: %2.2f ¥n”,tempf);
```

# 8. Timer

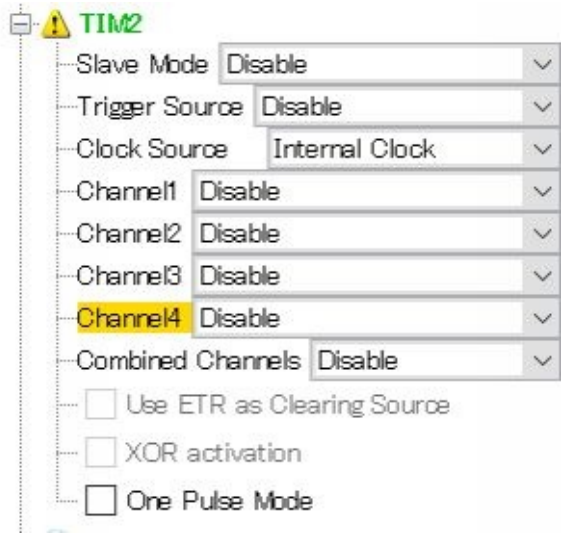
The timer function can realize functions such as an interval timer that processes at regular intervals, and an output compare that outputs High / Low signals at arbitrary time intervals.

8-1. Interval Timer

8-1-1. CubeMX Setting

Use interval timer when processing at fixed intervals. For the STM32F411 except for the special timer, eight timers are prepared, but this time we will explain the case where timer 2 is used as interval timer.

Set "Clock Source" of TIM 2 of "Pinout" tab to "Internal Clock".




Timers 2, 3, 4, and 5 are supplied from the internal clock APB1 from the block diagram of the data sheet. When checking "Clock Configuration", confirm that the default setting of the timer clock of APB 1 is set to 84 MHz.

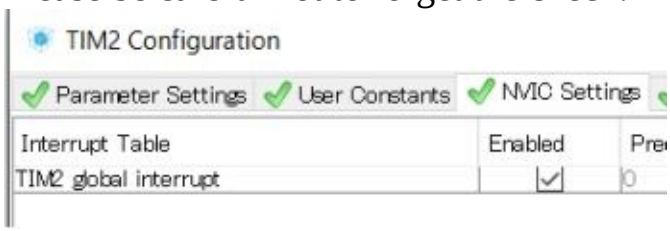

Click "TIM 2" on the "Configuration" tab to make detailed settings.

On the "Parameter Settings" tab, set the time interval for interrupts. Divide the clock frequency by the setting value of "Prescaler" and lower it to 1, which is an integer (set value -1). It counts up to the setting value of "Counter Period", and an interrupt occurs when the set value is reached.

It is set at 1 us interval and an interrupt is generated every 500 µs. "Prescaler" is set to 83 from 84 - 1 from the timer clock 84 MHz of APB 1 confirmed earlier. In order to generate an interrupt every 500 µs, set 499 of 500 - 1 to "Counter Period".

However, the timer condition changes depending on the setting of the clock. There is also a method to set the timer condition to be the same regardless of the clock setting on the code. In this case, since it is set on the code, the values of "Prescaler" and "Counter Period" of CubeMX can be left at the initial value of 0.

Activate the interrupt on the "NVIC Settings" tab. To activate, check "Enabled" of "TIM 2 global interrupt". If you forget this check, no interrupt will be generated even if the specified time is reached. Please be careful not to forget the check.




8-1-2. Calc Interrupt Interval

TIM 2, 3, 4, 5, 6, and 7 are clocked by APB 1, and TIM 1 and 8 are supplied by APB 2. It depends on the timer used.
If the internal clock division CKD, the prescaler divide ratio prescaler, the count cycle Period, and the timer clock APB 1 (MHz) are used, the interrupt interval IT (us) is given by the following formula.

$$IT(us) = \frac{(CKD + Prescaler + 1)(Period + 1)}{APB1(MHz)}$$

$$= \frac{(1 + 83 + 1)(499 + 1)}{84MHz} \times 10us$$

8-1-3. Coding

When an interrupt occurs in the case of an interval timer, the function HAL_TIM_PeriodElapsedCallback to be executed is added to the main.c file. Although TIM 2 was activated this time, the same function is called also when using other interval timer, so condition branch from the instance when called. In the following example in the conditional branch, LED is changed by TIM 2 interrupt.

```
/* USER CODE BEGIN PFP */
void HAL_TIM_PeriodElapsedCallback(
            TIM_HandleTypeDef *htim){
   if (htim->Instance==TIM2){
   //TIM2 Task
      HAL_GPIO_TogglePin(LD2_GPIO_Port,LD2_Pin);
    }
   else if (htim->Instance==TIM3){
      //TIM3 Task
  }
}
/* USER CODE END PFP */
```

To start the interval timer, execute the interrupt start function HAL_TIM_Base_Start_IT to start the interval timer.

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

Conversely, when stopping the interval timer, stop by the interrupt stop function HAL_TIM_Base_Stop_IT.

```
HAL_TIM_Base_Stop_IT(&htim2);
```

I will show you how to set the timer condition regardless of the clock setting on the code. The clock setting is acquired by setting the SystemCoreClock variable to the Prescaler setting in the timer initialization function MX_TIM2_Init () of main.c. The timer condition can be set in us. The following is an example when the timer condition is set to 10 μs in units of 1 μs.

```
htim2.Instance = TIM2;
htim2.Init.Prescaler =
      (SystemCoreClock / 1000000) - 1;  //
1us Tick
```

```
htim2.Init.CounterMode =
      TIM_COUNTERMODE_UP;
htim2.Init.Period = 10-1; // 10us (100kHz)
htim2.Init.ClockDivision=
      TIM_CLOCKDIVISION_DIV1;
```

The following is an example when the timer condition is set to 10 ms in units of 1 ms.

```
htim2.Instance = TIM2;
htim2.Init.Prescaler =
      (SystemCoreClock/1000000)*1000-1;//
1ms Tick
htim2.Init.CounterMode =
      TIM_COUNTERMODE_UP;
htim2.Init.Period = 10-1; // 10ms (100Hz)
htim2.Init.ClockDivision=
TIM_CLOCKDIVISION_DIV1;
```

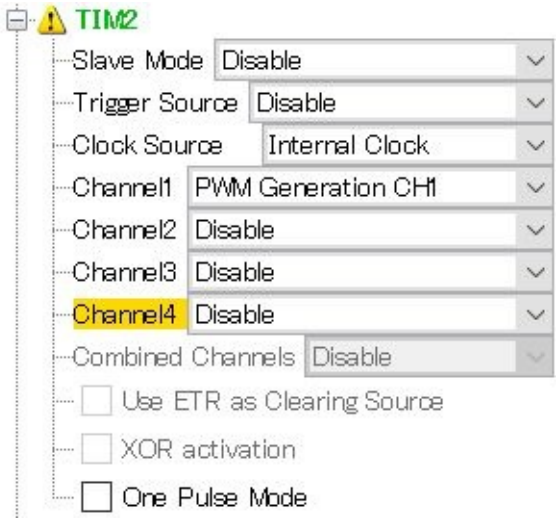
8-2. Output Compare

8-2-1. CubeMX Setting

PWM (Pulse Width Modulation) is available as a function used for motor control and LED brightness control. PWM repeats on and off at high speed and changes the speed and brightness of the motor by changing the on / off ratio (duty ratio).

PWM is available with the output compare function. This time we will explain the case where timer 2 is used as output compare. In addition, multiple channels can be set as PWM output and different duty ratios can be set. This time I will explain how to use only 1 channel.

Set "Clock Source" of TIM 2 of "Pinout" tab to "Internal Clock" and set "Channel 1" to "PWM Generation CH 1".




Click "TIM 2" on the "Configuration" tab to make detailed settings.

On the "Parameter Settings" tab, set the PWM cycle and PWM resolution. Divide the clock frequency by the setting value of "Prescaler" with the setting value of "Prescaler" and lower it to 1, which is an integer (set value -1). The setting value of "Counter Period" determines the resolution of PWM. The frequency of PWM is determined by the setting value of "Prescaler" and the setting value of "Counter Period".

This time we will explain the case of the resolution of 1024 stages of duty ratio at about 40 kHz. "Prescaler" is set to 1 of 84 / 42-1 from the timer clock 84 MHz of APB 1. To generate an interrupt every 1024 steps of the duty ratio resolution, set 1024 to "Counter Period".

It is not necessary to set the interrupt on the "NVIC Settings" tab for output compare.

When PWM frequency is 15 kHz or less, switching frequency of PWM as audible sound from motor and switching element is heard as noise, therefore 20 kHz or more is preferable. However, when the frequency is high, such as 100 kHz, noise may increase due to counter electromotive force etc. of the motor or the switching element may not respond. It is necessary to adjust the PWM frequency according to the motor and parts to be used.

8-2-2.Calc PWM Frequency

TIM 2, 3, 4, 5, 6, and 7 are clocked by APB 1, and TIM 1 and 8 are supplied by APB 2. It depends on the timer used.

The internal clock division CKD, the prescaler divide ratio prescaler, the count cycle Period, and the timer clock APB 1 (MHz), the PWM frequency Frq (KHz) is given by the following formula.

$$Frq(kHz) = \frac{APB1(MHz)}{CKD \times (Prescaler+1) \times (Period+1)}$$

$$= \frac{84MHz}{1 \times (1+1) \times (1024+1)} = 40kHz$$

8-2-3. Coding

PWM starts with the PWM start function HAL_TIM_PWM_Start.

```
HAL_TIM_PWM_Start(&htim2,
TIM_CHANNEL_1);
```

To change the PWM duty ratio, use the PWM set function __HAL_TIM_SET_COMPARE.

```
                    __HAL_TIM_SET_COMPARE(     &htim2,
                TIM_CHANNEL_1,
                                duty0);
```

As with the interval timer, regardless of the clock setting, when setting the timer condition, set it within the timer initialization function MX_TIM2_Init () of main.c as shown below.

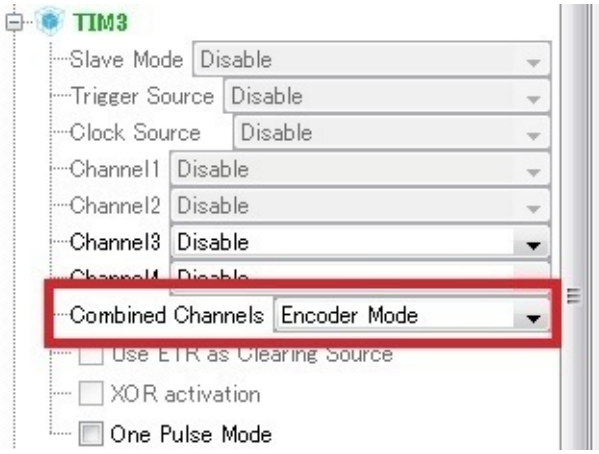
```
htim2.Instance = TIM2;
htim2.Init.Prescaler =
      (SystemCoreClock/1000000)/42-1;
htim2.Init.CounterMode=
      TIM_COUNTERMODE_UP;
htim2.Init.Period =1024;
    htim2.Init.ClockDivision=
    TIM_CLOCKDIVISION_DIV1;
```

8-3. Read From Quadrature Encoder

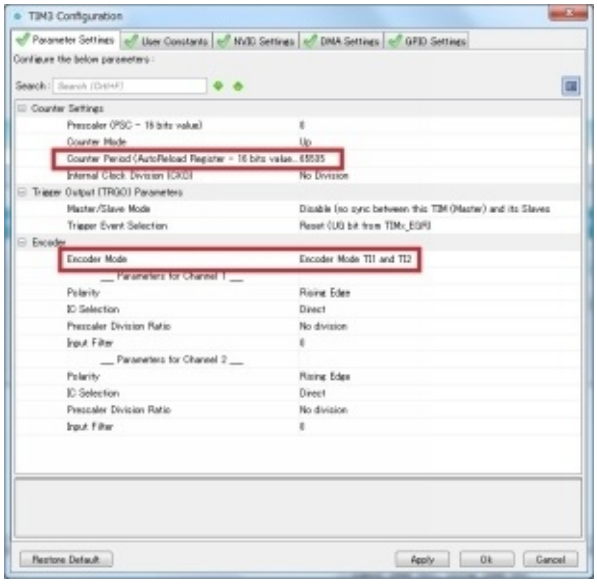
One of timer functions has a reading function of a quadrature encoder.

8-3-1. CubeMX Setting

When "Combined Channels" is set to "Encoder Mode" in the item of "TIM *" on the "Pinout" tab of CubeMX, ports to connect phase A and phase B are automatically assigned. In this time TIM 3 was set to encoder mode.




It is reset to 0 when counting up to the value set in Period. By setting EncoderMode to TI1 and TI2, it is multiplied by 4 (it counts at both the rising and falling edges of Phase A and Phase B).




8-3-2. Coding

To start counting the encoder value, use the HAL_TIM_Encoder_Start function. The direction of rotation can be obtained with 0 or 1 of the __HAL_TIM_IS_TIM_COUNTING_DOWN function. It is 0 when the count value is incremented and 1 when the count value decreases. In addition, the encoder count value is acquired with TIM 3 -> CNT.

```
        HAL_TIM_Encoder_Start(&htim3,
        TIM_CHANNEL_ALL);
printf("Initialized Success!!¥n");

while (1){
  HAL_Delay(300);
```

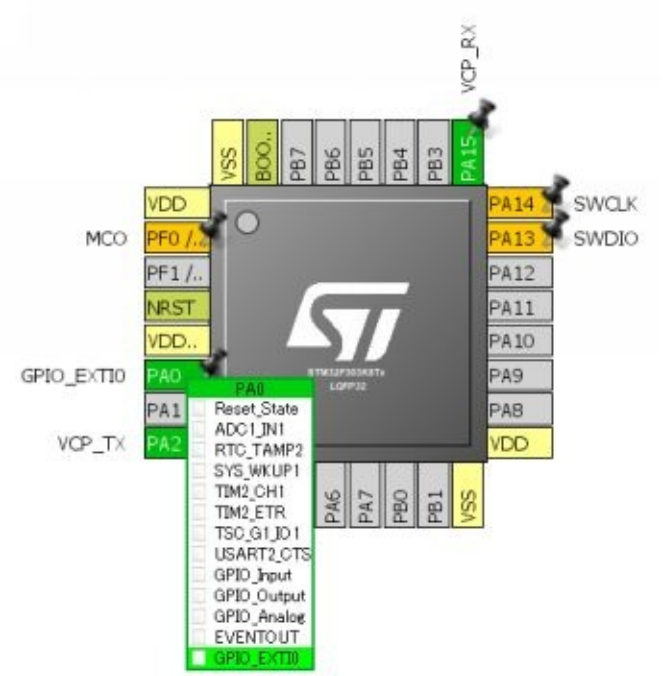
```
    int8_t uwDirection =
    __HAL_TIM_IS_TIM_COUNTING_DOWN(
      &htim3);
    uint16_t cnt = TIM3->CNT;
    printf("CNT:%d,                    Drc:%d
¥n",cnt,uwDirection);
}
```

# 9. External Event Trigger

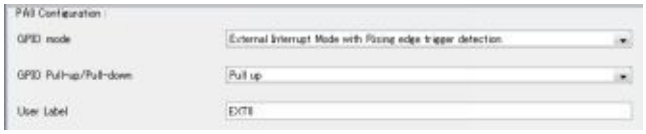
A timer is internally counted and generates an event when it reaches a certain time, it is called an internal event. On the other hand, external event interrupt is used for user's button operation, sensor reading completion, data reception, etc. This chapter explains how to generate external events from GPIO. Trigger that generates an event can set the rising edge, falling edge, or both of the input signal.

9-1. CubeMX Setting

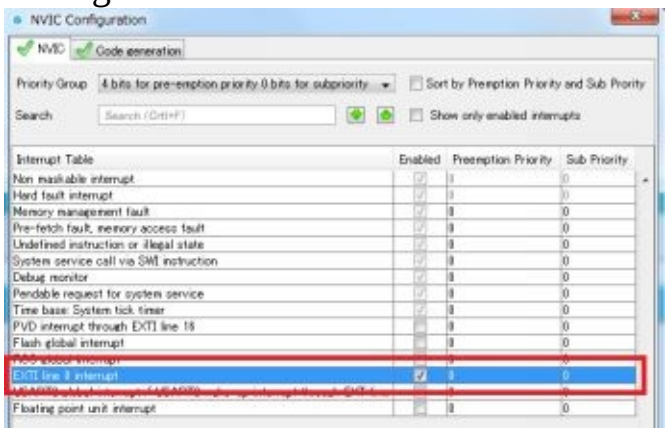
This time NUCLEO F 303 K 8 was used as the target board. The GPIO that generates the external event interrupt is PA0. Select "GPIO_EXTI *" with the GPIO that you set on the "Pinout" tab of CubeMX.




Next, click the "GPIO" button on the "Configuration" tab to open the detailed setting screen. Select "External Interrupt Mode with rising edge trigger detection" to generate an event at the rising edge of the external signal in "GPIO mode". Select "Falling edge" to generate an event at the falling edge, or "Rising / Falling" to generate an event at both. Also, selecting "Pull up" with "GPIO Pull-up / Pull-down" enables internal pull-up, eliminating an external pull-up resistor. Therefore, it is possible to generate an event by attaching a switch to PA0 and pushing the switch with only wiring that drops one side to GND.




In order to generate an external event interrupt, click on the "NVIC" button on the "CubeMX Configuration" tab to set it. Check "EXTI line *" on the "NVIC" tab to activate the interrupt.



9-2. Coding

Add the external event interrupt function HAL_GPIO_EXTI_Callback as a global function. This function is called when an external event interrupt occurs. In this example, since external event interrupt is set in PA0, processing is performed when an interrupt occurs in GPIO_PIN_0.

```
void
HAL_GPIO_EXTI_Callback(uint16_t
GPIO_Pin)
{
 if(GPIO_Pin == GPIO_PIN_0)
 {
        //Do something in interrupt!
            printf("EXT0 Interrupt!!¥n");
 }
}
```

External event interrupt occurs when executed and PA 0 is dropped to GND. Since chattering prevention processing and circuits are not put in, interrupts will enter many times with one button operation. In the case of buttons, measures against chattering are required.

# 10. AD Conversion

Use AD conversion when reading the voltage value of the sensor output with voltage. Most ST microcontrollers have 12 bit resolution and 1 Msps maximum performance. In other words, it can read 4096 divisions (about 0.0008 V resolution when the reference voltage is 3.3 V) and 1 million times per second.

AD conversion is set to perform AD conversion at an arbitrary timing in software (manual reading without trigger), setting to read at fixed intervals (timer trigger automatic reading) as a timer, transfer the acquisition value directly to memory without going through the MPU. You can select settings such as setting (automatic reading of DMA transfer).

When loading the value of the potentiometer (variable resistance) etc. only once at the time of turning on the power supply etc. of the microcontoller etc., there is no trigger manual reading. In reading the value many times in succession, it uses the timer trigger reading etc. to perform FFT or averaging processing.

10-1. Software Trigger

10-1-1. CubeMX Setting

In performing AD conversion at arbitrary timing on software, manual reading without trigger is performed. This time I will explain the method of AD conversion of two channels.

Check "IN 0" and "IN 1" in ADC 1 of the "Pinout" tab. PA0 and PA1 are automatically assigned as IO for AD conversion. For the STM32F3 series, select "Single-ended" instead of checking.


Click "ADC 1" on the "Configuration" tab to make detailed settings. In the "Parameter Settings" tab, detailed setting of AD conversion is done, but this time it is fine as it is for manual reading with no trigger.

10-1-2. Coding

It alternately repeats AD conversion of two channels and reads. Function to set the channel to be converted before AD conversion 1 HAL_ADC_ConfigChannel is used. The AD conversion starts with the HAL_ADC_Startfunction 2 . The HAL_ADC_PollForConversionfunction 3 requests AD conversion start on the software. HAL_ADC_Stopfunction 4 to stop AD conversion. This is a series of flows. Next, AD conversion is performed for the other channel in the same way.

```
ADC_ChannelConfTypeDef sConfig;
uint32_t adcValue0,adcValue1;
while(1){
  sConfig.Channel = ADC_CHANNEL_0;
    sConfig.Rank = 1;
  sConfig.SamplingTime =
    ADC_SAMPLETIME_56CYCLES;//->F4 Series
    //          ADC_SAMPLETIME_1CYCLE_5//->F3
Series
    if (HAL_ADC_ConfigChannel(&hadc1,      //1
        &sConfig) != HAL_OK){}
  HAL_ADC_Start(&hadc1);      //2

if(HAL_ADC_PollForConversion(&hadc1,1000)//3
          == HAL_OK) {
     adcValue0 = HAL_ADC_GetValue(&hadc1);
    }
  HAL_ADC_Stop(&hadc1);//4

  sConfig.Channel = ADC_CHANNEL_1;
  if (HAL_ADC_ConfigChannel(&hadc1,
        &sConfig) != HAL_OK){}
    HAL_ADC_Start(&hadc1);
  if (HAL_ADC_PollForConversion(&hadc1, 1000)
          == HAL_OK) {
       adcValue1 = HAL_ADC_GetValue(&hadc1);
```

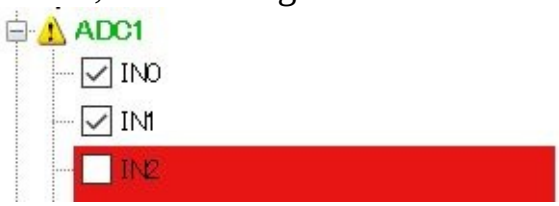
```
    }
  HAL_ADC_Stop(&hadc1);
  printf("AD0: %d, AD1: %d ¥n",
                adcValue0,adcValue1);
   HAL_Delay(300);
}
```

10-2. Timer Trigger

10-2-1. CubeMX Setting

Timer trigger reading is used for automatic reading at fixed intervals that are timers. This time, we will explain the method of reading two channels with 1 ms by timer with DMA transfer. When reading multiple channels, it is common to use DMA transfer.

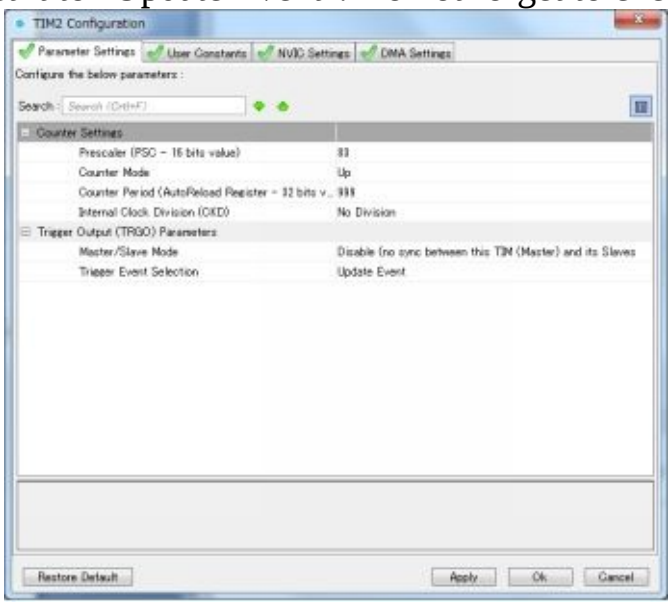
Check "IN 0" and "IN 1" in ADC 1 of the "Pinout" tab. PA0, PA1 are automatically assigned as IO for AD conversion. For the STM32F3 series, select "Single-ended" instead of checking.




Also, set "Clock Source" of TIM 2 to "Internal Clock".

Click "TIM 2" on the "Configuration" tab to make detailed settings. In "Parameter Settings" tab, set "prescaler" to 83 of 84-1 from 84 MHz of timer clock of APB 1. In order to generate an interrupt every 1 ms, set 9000 of 1000-1 to "Counter Period". Also, "Trigger Event Selection"

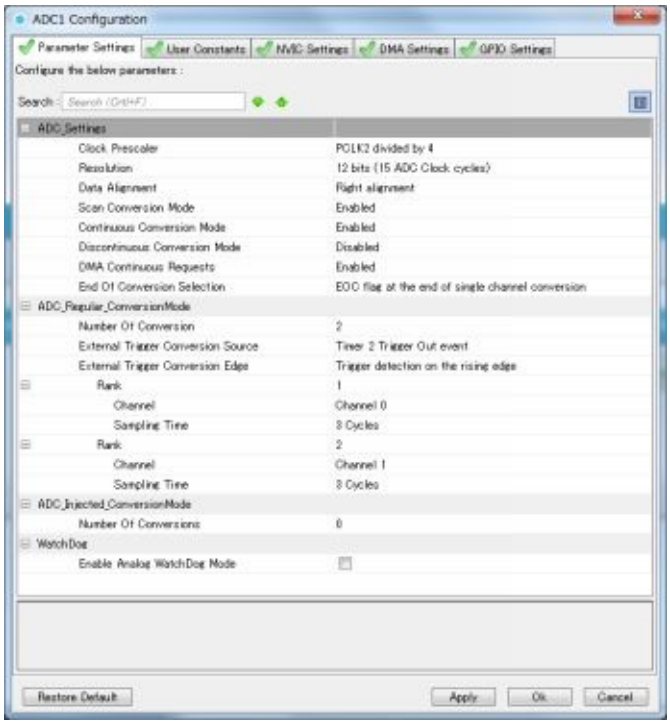
Set it to "Update Event". Do not forget to check "TIM 2 global interrupt" on the "NVIC Settings" tab.




Click "ADC 1" on the "Configuration" tab to make detailed settings. On the "Parameter Settings" tab, you will make detailed AD conversion settings. Set "Scan Conversion Mode" to "Enabled" and "Continuous Conversion Mode" to "Enabled". Set "DMA Continuos Request" to "Enable". Set "External Trigger Conversin Source" to "Timer 2 Trigger Out event" and set "External Trigger Conversion Edge" to

Set it to "Trigger detection on the rising edge". Also, set "Number of Conversion" to 2 and set the added "Rank" to "Channel 1".

Click "Add" on the tab of "DMA Settinngs" and select "ADC 1" from "Select" in the add column. For "Priority", select "Medium". Also change "DMA Request Settings" to "Circular".




10-2-2. Coding

AD conversion is executed after the timer interrupt, and the function HAL_ADC_ConvCpltCallback to be called when the translation for the specified buffer is completed is added as a global function. Also, define the buffer for DMA transfer as a global variable (Important!).

```
/* USER CODE BEGIN 0 */
#define LENGTH_SAMPLES    16
enum{ ADC_BUFFER_LENGTH=
LENGTH_SAMPLES };
uint16_t
g_ADCBuffer[ADC_BUFFER_LENGTH];
void HAL_ADC_ConvCpltCallback(
            ADC_HandleTypeDef*
AdcHandle){
        //Do something after AD conversion
}
/* USER CODE END 0 */
```

Before starting AD conversion, set the buffer and transfer count of the DMA transfer with the HAL_ADC_Start_DMA function, and start the timer with the HAL_TIM_Base_Start_IT function.

```
/* USER CODE BEGIN 2 */
                    HAL_ADC_Start_DMA(     &hadc1,
                    (uint32_t*)&g_ADCBuffer,
                                ADC_BUFFER_LENGTH);
  HAL_TIM_Base_Start_IT(&htim2);
/* USER CODE END 2 */
```

DMA transfer is transferred to the buffer alternately when 2 channels are used for IN0 and IN1.

```
while (1) {
```

```
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
        printf("ADC in0: %d, in1: %d ¥n",
                    g_ADCBuffer[0],g_ADCBuffer[1]);
        HAL_Delay(300);
  }
/* USER CODE END 3 */
```

In this time, nothing is processed by the HAL_ADC_ConvCpltCallback function, but the buffer is overwritten when the specified AD conversion is completed in the DMA buffer, so please enter the process of copying to another variable once in the HAL_ADC_ConvCpltCallback function.

10-3. Input Impedance

The AD converter that performs AD conversion has a sample hold circuit inside, and charges the capacitor with the input voltage. The smaller the impedance of the circuit that inputs the voltage to the AD converter is, the more current flows through the capacitor of the AD converter, so AD conversion can be started faster. Therefore, it is necessary to change the conversion time of the AD converter according to the impedance of the circuit that inputs the voltage to the AD converter. The following table shows the relationship between conversion time ts and input impedance Rsrc extracted from the STM32F373 data sheet.

Table 61. $R_{SRC}$ max for $f_{ADC}$ = 14 MHz(1)

| $T_s$ (cycles) | $t_s$ (µs) | $R_{SRC}$ max (kΩ) |
|---|---|---|
| 1.5 | 0.11 | 0.4 |
| 7.5 | 0.54 | 5.9 |
| 13.5 | 0.96 | 11.4 |
| 28.5 | 2.04 | 25.2 |
| 41.5 | 2.96 | 37.2 |
| 55.5 | 3.96 | 50 |
| 71.5 | 5.11 | 50 |
| 239.5 | 17.1 | 50 |

1. Guaranteed by design.

If conversion time is set to be long, conversion can be performed even with larger input impedance, but much time is required for AD conversion. Therefore, in general, when the input impedance is large, input voltage is input to the AD converter after lowering the input impedance via a voltage follower circuit or the like with an operational amplifier.

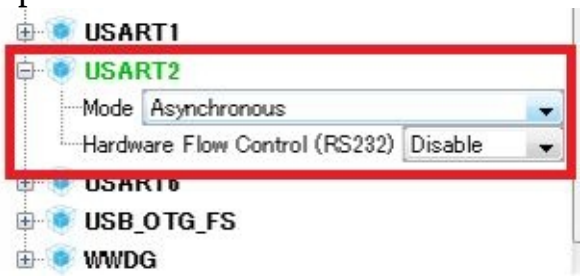
# 11. UART Communication

UART is already used for debugging, but UART communication is used for communication with PCs and other microcontollers.

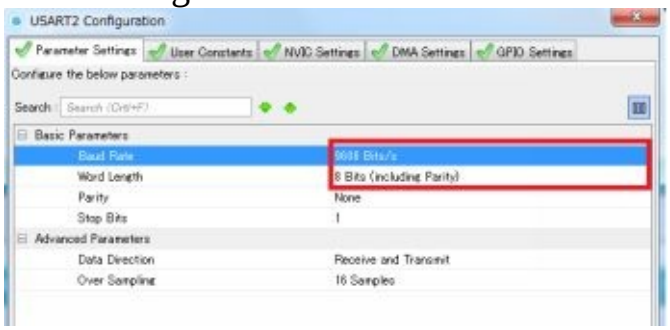
11-1. UART Transfer

11-1-1. CubeMX Setting

Set the port of USART 2 of "Pinout" tab from "disable" to "Asynchronous" in case of the NUCLEO board of STM 32 F 4. For the NUCLEO board of STM 32 F 303, set UART 1 to "Asynchronous". These port is connected the UART port of ST-Link.




Click the "USART 2" button on the "Configuration" tab and make detailed settings. Select "9600 bps" for "Baud Rate" and 8 bits for "Word Length".




11-1-2. Coding

When sending with UART, use the HAL_UART_Transmit function. Pass the UART structure as the first argument. Specify the data to be written to the second argument, the data length written to the third argument, and the timeout time for the fourth argument.

```
uint8_t sData[2];
sData[0]=10;
sData[1]=11;
HAL_UART_Transmit( &huart2,
                   sData,
                   2,
                   1000);
```

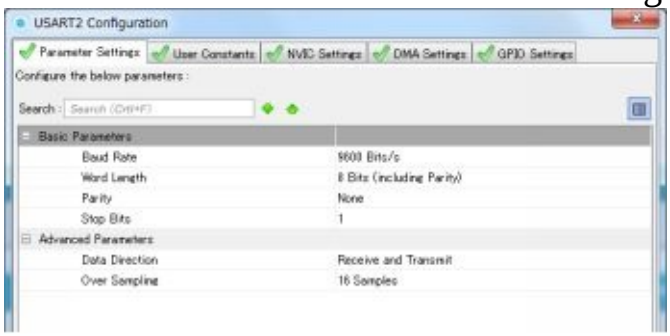
11-2. UART Recieve

UART reception is used DMA transfer. Reception by interrupts is also possible, but there is a possibility of data leak, etc., so we explain how to use DMA transfer.

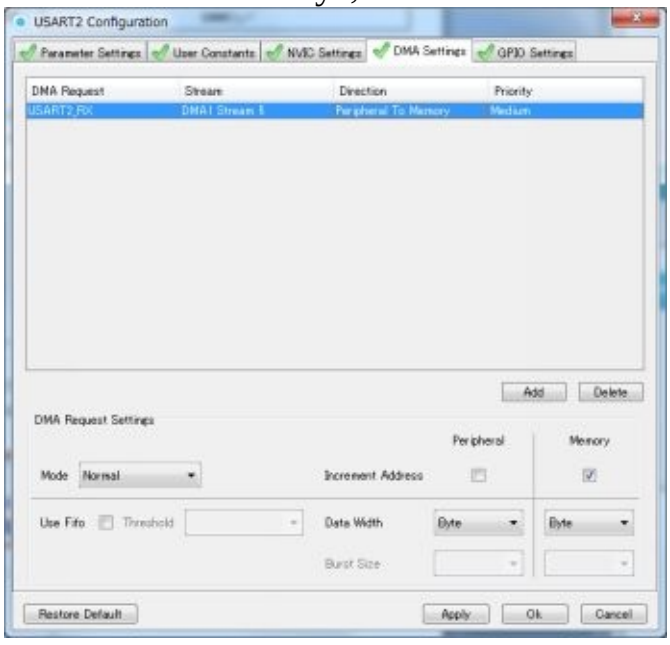
11-2-1. CubeMX Setting

Set the port of USART 2 of "Pinout" tab from "disable" to "Asynchronous". For the NUCLEO board of STM 32 F 303, set UART 1 to "Asynchronous".


Click the "USART 2" button on the "Configuration" tab and make detailed settings. Select "9600 bps" for "Baud Rate" and 8 bits for "Word Length".




Click "Add" on the tab of "DMA Settinngs" and select "UART 2 _ RX" from "Select" in the add column. For "Priority", select "Medium".




11-2-2. Coding

Each time one character is received, DMA transfer is performed to UART1_Buff, and an example of coding is shown in UART1_Data after an interrupt occurs. The DMA interrupt is started with the HAL_UART_Receive_DMA function. Pass the UART structure as the first argument. Specify the DMA buffer as the second argument and the DMA buffer data length as the third argument.

```
uint8_t UART1_Data[30]; //defined as global val
```

```
uint8_t UART1_Len; //defined as global val
uint8_t UART1_Buff[1]; //defined as global val
```

```
HAL_UART_Receive_DMA( &huart1,
            (uint8_t*)& UART1_Buff, 1);
```

Add the function HAL_UART_RxCpltCallback called after receiving the specified number of characters as a global function. Execute the interrupt start function again after interrupt processing.

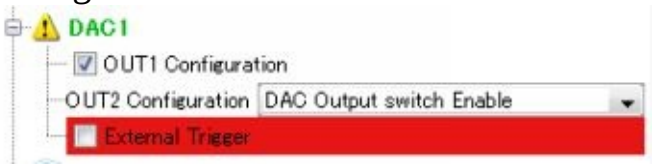
```
void HAL_UART_RxCpltCallback(
          UART_HandleTypeDef *UartHandle){
 if(UartHandle->Instance==USART1){
 switch(UART1_Buff [0]){
   case 0x0A:
   case 0x0D:
        //Do something in receiving return code
        break;
      default:
        UART1_Data[UART1_Len++]=UART1_Buff[0];
        if(UART1_Len >= 29)
{               UART1_Len = 0;//Buffer over
        }
        break;
   }
   HAL_UART_Receive_DMA( &huart1,
                (uint8_t*)& UART1_Buff, 1);
 }
}
```

# 12. DA Conversion

Use DA conversion when analog control of external equipment or simple sound is made. Since the STM32F411 does not have a built-in DA, this time I will explain using the STM32F303 with built-in 12-bit DA outputting 2 channels.

12-1. CubeMX Setting

Check "OUT1 Configuration" in DAC 1 of the "Pinout" tab and select "DAC Output switch Enable" for "OUT 2". Two DA outputs are automatically assigned to PA4 and PA5 of the right pin assignment diagram. By clicking "DAC1" on the "Configuration" tab, the initial value can be used for detailed setting.



12-2. Coding

Use the DA start function HAL_DAC_Start to start DA.

```
HAL_DAC_Start(&hdac1,
DAC_CHANNEL_1);
HAL_DAC_Start(&hdac1,
DAC_CHANNEL_2);
```

For this example, we will output a triangular wave as two channels. Use the function HAL_DAC_SetValue to set the value to DA.

```
   int out=0;

 while (1)
 {
   if(out>3600)out=0;
   else out=out+300;
   HAL_DAC_SetValue(&hdac1,
                                          DAC_CHANNEL_1,
                                          DAC_ALIGN_12B_R,
                                          out);
   HAL_DAC_SetValue(&hdac1,
                                          DAC_CHANNEL_2,
                                          DAC_ALIGN_12B_R,
                                          4096-out);
   printf("DA TEST: %d ¥n",i);
   HAL_Delay(200);
 }
```

# 13. Real Time Clock

Use Real Time Clock (Real Time Clock, RTC) when recording at a certain time, writing time information to the memory card, and so on. In general, it is common to use an RTC module with I2C connection externally, but in the case of STM32 microcontoller, many STM32 microcontollers of 48 pins or more have VBAT terminal. When 32.768 kHz crystal resonator and battery are connected, built-in RTC Functions can be used without parts.

In addition to the RTC, there is a backup register of about 64 bytes, and it is possible to hold the value with the battery. For example, it seems that it can be used for keeping state of equipment etc.

Once the RTC in the STM32 microcontoller is connected to the 32.768 kHz crystal oscillator on the board, the setting is retained even if another code is written, and the time function is in effect.

13-1. CubeMX Setting

Activate RTC on "Pinout" tab of STM32CubeMX and check "Activate Clock Source" and "Activate Calender".


Also, since 32.768 kHz crystal oscillator is used as the external oscillator, select "LSE" as "Crystal / Ceramic Resonator" with "RCC" on the "Pinout" tab. Make the setting to be installed.

13-2. RTC Wiring

Use a voltage of 1.6 to 3.6 V as the holding power supply. This time connect the 3 V coin battery such as CR 2032 to the VBAT terminal.

Attach 32.768 kHz crystal oscillator and oscillation stabilizing capacitor to PC 14 and PC 15.

13-3. Coding

The time and date are set using the HAL_RTC_SetTime function and the HAL_RTC_SetDate function using the structure.

For years of dates do not set the year as "2017" and set as a reference based on a certain year such as "2000".

```
RTC_TimeTypeDef sTime;
sTime.Hours = 11;
sTime.Minutes =12;
sTime.Seconds = 13;
sTime.SubSeconds = 0;
sTime.TimeFormat =
RTC_HOURFORMAT_24;
sTime.DayLightSaving =
RTC_DAYLIGHTSAVING_NONE;
sTime.StoreOperation =
RTC_STOREOPERATION_RESET;

RTC_DateTypeDef sDate;
sDate.WeekDay =
RTC_WEEKDAY_MONDAY;
sDate.Month =
RTC_MONTH_JANUARY;
sDate.Date = 15;
sDate.Year= 0;//  reference based year
HAL_RTC_SetTime(&hrtc, &sTime,
FORMAT_BCD);
HAL_RTC_SetDate(&hrtc, &sDate,
FORMAT_BCD);
```

When reading time and date, use the structure and use the HAL_RTC_GetTime function and HAL_RTC_GetDate function.

```
RTC_TimeTypeDef gTime;
RTC_DateTypeDef gDate;
HAL_RTC_GetTime(&hrtc, &gTime, FORMAT_BIN);
HAL_RTC_GetDate(&hrtc, &gDate, FORMAT_BIN);
printf("YY MM DD : %d , %d, %d,  ",
   gDate.Year,gDate.Month,gDate.Date);
printf("HH MM SS : %d , %d, %d ¥n",
   gTime.Hours,gTime.Minutes,gTime.Seconds);
```

Backup registers that can hold values with batteries use the write function HAL_RTCEx_BKUPWrite and the read function HAL_RTCEx_BKUPRead. It can hold up to 32 with 32 bit width.

```
uint32_t wdata=0x01234;
HAL_RTCEx_BKUPWrite(     &hrtc,
                         RTC_BKP_DR0,
                         wdata);
```

```
uint32_t rdata=HAL_RTCEx_BKUPRead(
                                &hrtc,
                                RTC_BKP_DR0);
```

# 14. Use of DSP

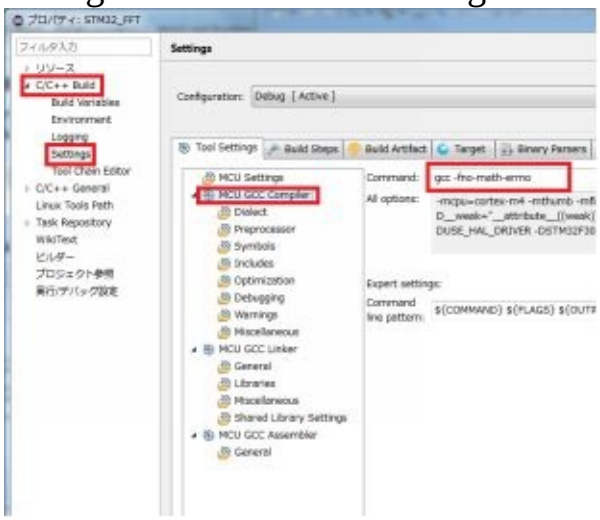
STM32F4 and STM32F3 have built-in DSP and FPU. It is possible to perform trigonometric functions and floating-point arithmetic at high speed.

14-1. MATH Library

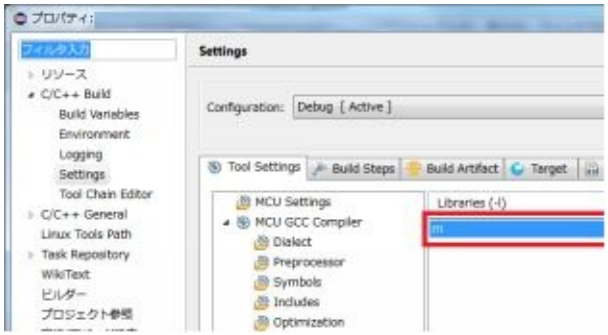
When using the Sqrt function, the pow function, etc., the compile error as shown below may occur and compilation may not be possible. In this case, change the build setting.

```
undefined reference to `__errno'
collect2.exe: error: ld returned 1 exit status
make: *** [*XXXX.elf] Error 1
```

From the Eclipse project
"C / C ++ Build" → "Settings" → "ToolSettings" → "MCU GCC Compiler"
Change "Command" field to "gcc - fno - math - errno".




In addition, from the Eclipse project
"C / C ++ Build" → "Settings" → "ToolSettings" → "MCU GCC Linker" in "Libraries"
m
Add.



14-2. FFT

I will explain FFT processing using DSP. Please refer to the "AD conversion" chapter for CubeMX setting for how to set AD conversion by timer interrupt.

14-2-1. Add DSP Library

In order to use the DSP library, add the following definition after #include "stm32f3xx_hal.h" in main.c. Add the definition under the header of the DSP library.

```
#define ARM_MATH_CM4
#include "arm_math.h"
#include "arm_const_structs.h"
```

The DSP library file is in the CubeMX library. By default it will be installed in the following directory. Place arm_cortexM4lf_math.lib in C: ¥ Users ¥ (user name) ¥ STM32Cube ¥ Repository ¥ STM32 Cube_FW_FXX_VX.X.X ¥ Drivers ¥ CMSIS ¥ Lib ¥ ARM directly under the project folder. In addition, there are arm_cortexM4b_math.lib,

arm_cortexM4bf_math.lib and arm_cortexM4l_math.lib, but this case("little endian", "fpu", "m4"), select arm_cortexM4lf_math.lib. Also, place libarm_cortexM4lf_math.a in C: ¥ Users ¥ (user name) ¥ STM32Cube ¥ Repository ¥ STM32 Cube_FW_FXX_VX.X.X ¥ Drivers ¥ CMSIS ¥ Lib ¥ GCC directly under the project folder.

In the properties of the project, add the following libraries carefully in the "Linker, Libraries" in "Settings", "Tool Settings" "MCU GCC Linker" of "C / C ++ Build".

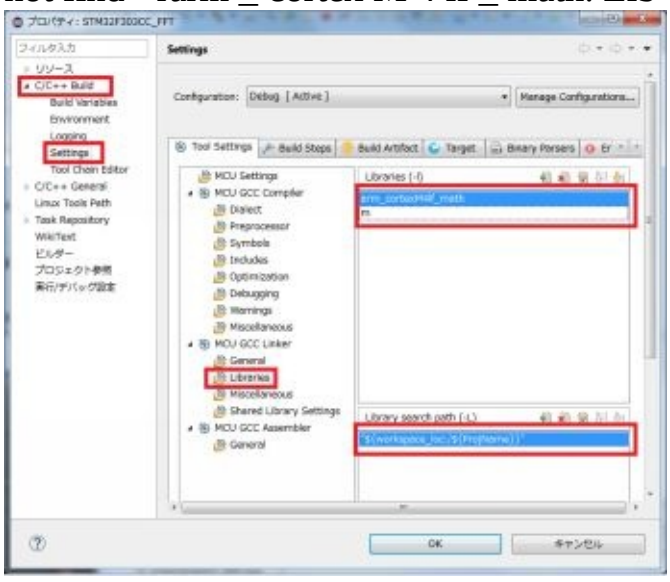
arm_cortexM4lf_math

m

When you select the project folder under the workspace in the Library search path, it is automatically specified as the relative directory below.

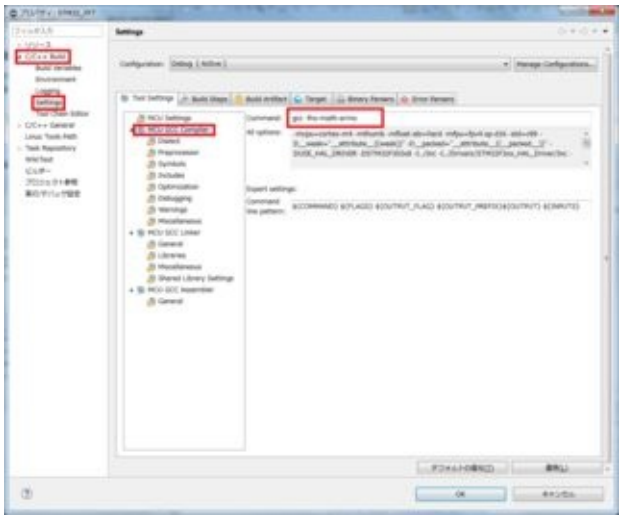
"$ {workspace_loc: / $ {ProjName}}"

If this setting is forgotten, a compile error such as "undefined reference to` arm _ cfft _ f 32 '"or" can not find - rarm _ cortex M 4 lf _ math. Lib "occurs.




To use the hard FPU, please add gcc argument to process exponential functions etc. From the Eclipse project

Change "Command" field of "C / C ++ Build" → "Settings" → "ToolSettings" → "MCU GCC Compiler" to "gcc - fno - math - errno".



14-2-2. Coding

Define the following definitions and global variables at the top of main.c.

```
#define LENGTH_SAMPLES     256
// Real part and imaginary part

#define ifftFlag       0
#define doBitReverse    1
#define fftSize
 LENGTH_SAMPLES/2
#define FFTLen arm_cfft_sR_q15_len128

q15_t FFT_Input[LENGTH_SAMPLES];
q15_t
FFT_Output[LENGTH_SAMPLES/2];

enum{ WAIT,RUN};
//AD Val is only real part =>size 128
enum{
ADC_BUFFER_LENGTH =
LENGTH_SAMPLES/2 };
//DMA transfer -> HalfWord:16bit Length
uint16_t
g_ADCBuffer[ADC_BUFFER_LENGTH];
```

As a global function, define an interrupt function after completion of AD conversion. After AD conversion, AD conversion is stopped and it assigns from the AD conversion buffer to the FFT input variable.

```
void HAL_ADC_ConvCpltCallback(
                    ADC_HandleTypeDef*
AdcHandle){
   HAL_TIM_Base_Stop_IT(&htim2);
   HAL_ADC_Stop_DMA(&hadc1);
```

```
   for(int
i=0;i<LENGTH_SAMPLES/2;i++){
       int tmp;
       tmp= g_ADCBuffer[i];
      //     Store real part and imaginary
part alternately in array

       FFT_Input[2*i]=tmp;         //Real
Part
       FFT_Input[2*i+1]=0.0;  //Imaginary
part
   }
   fft_status=RUN;
}
```

Perform FFT processing within the main function. After FFT processing, restart DMA transfer and AD conversion.

```
int main(void)
{
    //Do Initialize process
  while (1)
  {
              if(fft_status==RUN){
                 arm_cfft_q15(&FFTLen,
                              FFT_Input,
                              ifftFlag,
                              doBitReverse);
                 arm_cmplx_mag_q15(FFT_Input,
                                   FFT_Output,
                                   fftSize);
                 arm_max_q15(FFT_Output,
                               fftSize,
                               &maxValue,
                               &testIndex);
                              for(int
j=0;j<fftSize/2;j++){
                        printf("%d, ",FFT_Output[j]);
                             }
          fft_status=WAIT;
                 HAL_ADC_Start_DMA(&hadc1,
                             g_ADCBuffer,
                             ADC_BUFFER_LENGTH);
                 HAL_TIM_Base_Start_IT(&htim2);
               }
               else{
```

```
                __asm volatile("nop");
            }
    }
}
```

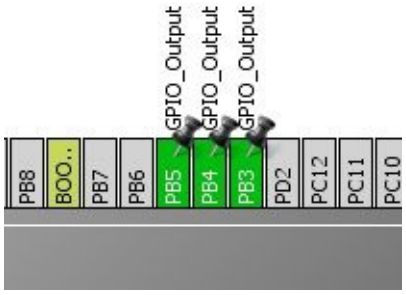
# 15. FreeRTOS

Since the high-end model of the ST microcontoller is equipped with a high-speed MPU and rich RAM, it is possible to perform various processing. When processing is several pieces, it is possible to divide the processing using multiple timers, but as more processing increases it becomes more difficult to separate processing by just the timer. In that case it is used RTOS (Real Time Operating System).
The task can be divided for each process and the OS side can automatically switch the task according to the priority order and process it. The larger the scale, the more RTOS needs to be introduced. In the case of the ST microcontoller, it is possible to easily introduce Free RTOS which is a kind of RTOS simply by checking on CubeMX.

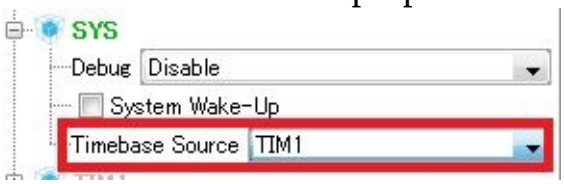
15-1. CubeMX Setting

Check "FREERTOS" on "Pinout" tab of STM32CubeMX.


In the "Pinout" tab, set PB 3, PB 4, PB 5 of the pin assignment diagram to GPIO_OUTPUT.

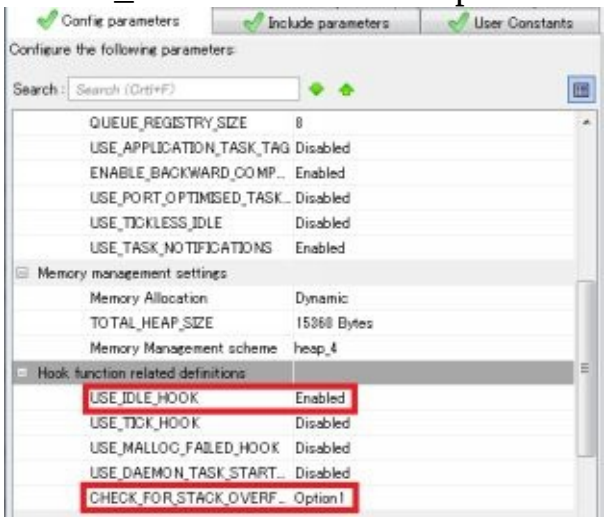



Also change "TimebaseSource" in "SYS" as the timer used for RTOS management from the initial value "Systick" to "TIMx". I chose "TIM 1" below. The selected timer is occupied by the RTOS, so it cannot be used for other purposes.

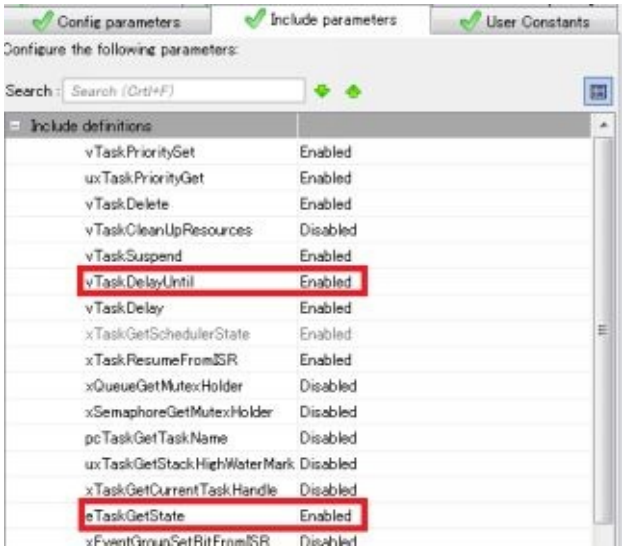



Click "FREERTOS" on the "Configuration" tab and make detailed settings.

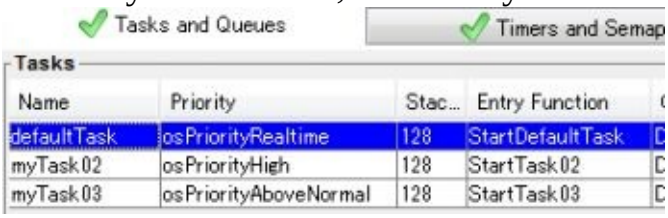

On the tab of "Config parameters", set "USE_IDLE_HOOK" of "Hook function related definitions" to "Enabled" and "CHECK_FOR_STACK_OVERFLOW" to "Option 1".




Also, set "vTaskDelayUntil" to "Enabled" and "eTaskGetState" to "Enabled" on the "Include parameters" tab.




Task settings are done from the "Tasks and Queues" tab. This time we will create three tasks. Since one task has been set by default, double click on "defaultTask" and set "Priority" to "osPriorityRealtime". Click "Add" to create "myTask02" and set "Priority" to "osPriorityHigh". Similarly click "Add", create "myTask03" and set "Priority" to "osPriorityAboveNormal".



15-2. Coding

In this example, we will execute the task of blinking the output of PB 3, PB 4, PB 5 of GPIO every 100 ms, 250 ms, 350 ms respectively.

```
void StartDefaultTask(void const * argument)
{
        uint32_t PreviousWakeTime = osKernelSysTick();
  for(;;){
            HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_3);
       osDelayUntil (&PreviousWakeTime, 100);
  }
}

void StartTask02(void const * argument)
{
        uint32_t PreviousWakeTime = osKernelSysTick();
  for(;;){
            HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_4);
       osDelayUntil (&PreviousWakeTime, 250);
  }
 }

void StartTask03(void const * argument)
{
        uint32_t PreviousWakeTime = osKernelSysTick();
  for(;;){
            HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
       osDelayUntil (&PreviousWakeTime, 350);
  }
}
```

# 16. Use of SD Card

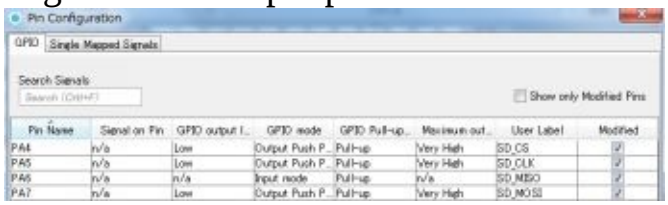
We will show you how to use SD card convenient for reading and writing data. This time we will introduce reading and writing of SD card via GPIO by using Chan's FatFS module which is a general library. Although I cannot expect the speed of reading and writing, I will show you how to communicate with an empty GPIO without using SPI.

16-1. CubeMX Setting

In the "Pinout" tab of the STM32CubeMX, set PA4, PA6, PA5 of the pin assignment diagram to GPIO_OUTPUT and PA7 to GPIO_INPUT.

* Since Chan's FatFS module is used, FatFS in the "Pinout" tab in CubeMX cannot be checked.

Also, enable Pull-up from "GPIO" on the "Configuration" tab as below and set the speed to "Very High" for the output pin.



16-2. FatFS Library Setting

16-2-1. Download FatFS

Download "Sample Project" on Chan's site "FatFs General FAT File System Module" below.

URL http://elm-chan.org/fsw/ff/00index_e.html

Use the code of the generic folder in the ffsample folder. Place the following files in Inc and Src in Workspace respectively.

diskio.h、 ff.c、 ff.h、 ffconf.h、 interger.h、 sdmm.c

16-2-2. Modify Library

Rewrite the library so that it can be used with STM32 and SW4STM32.

ffconf.h

```
   #define _FS_NORTC   1     //Disable
RTC
   #define _USE_LFN   0     //Disable
LFN
```

sdmm.c

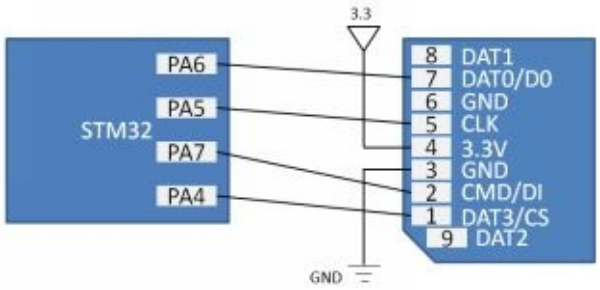
```
#define DO_INIT()
#define DO HAL_GPIO_ReadPin(GPIOA, GPIO_PIN_6)
#define DI_INIT()
#define     DI_H() HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_7,
                                     GPIO_PIN_SET)
#define DI_L() HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_7,
                                     GPIO_PIN_RESET)
#define CK_INIT()
#define CK_H()     HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_5,
                                     GPIO_PIN_SET)
#define CK_L()     HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_5,
                                     GPIO_PIN_RESET)

#define CS_INIT()
#define CS_H()     HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_4,
                                     GPIO_PIN_SET)
#define CS_L()     HAL_GPIO_WritePin(GPIOA,
                                     GPIO_PIN_4,
                                     GPIO_PIN_RESET)
```

Also comment out "PINB;" in dly_us (UINT n) function in "sdmm.c". Since the name of the (void) select function overlaps with other functions, replace it with another name that is appropriate (eg sselect). Together, change the 2 functions to the name replacing the part using the (void) select function in sdmm.c.

Comment out #include <avr / io.h> and change it to #include "stm32f3xx_hal.h" or #include "stm32f4xx_hal.h".

15-6. Wiring SD

Connect PA 4 to SD card CS, PA 5 to SD card CLK, PA 6 to SD card DO, PA 7 to SD card DI. Together connect 3.3 V power supply and GND.



16-4. Coding

In this example, mount the SD card in write mode and write a character string to "newfile.txt".

```
#include "ff.h"//Include
```

```
FATFS FatFs;
FIL Fil;
```

```
  int bw;
  if(f_mount(&FatFs, "", 1)==FR_OK){
        printf("Mount OK!! ¥n");
   }
     else{
    printf("Mount Fail!! ¥n");
    while(1);
     }
   if (f_open(&Fil,
                        "newfile.txt",
                        FA_WRITE |
FA_CREATE_ALWAYS) == FR_OK) {
        f_write(&Fil,
                    "It works!!!!!!!¥r¥n",
                    16,
                    &bw);
      f_close(&Fil);
  if(bw==16)printf("Write OK!! ¥n");
     while(1);
    }
        else{
          printf("Open Fail!! ¥n");
        while(1);
    }
```

# 17. Mounting PCB

We will introduce peripheral circuits and writing method when mounting the STM32 microcontoller on our own board.

17-1. Peripherals

In the case of making a board using an STM32 microcontoller, in addition to VDD and GND, the following terminals can be pulled out with terminals or connectors so that they can be used as needed.

· BOOT 0 pin for Boot mode switching
(BOOT 1 pin is also available depending on the type)
→ Pull down to Low normally.
Turn on the power by pulling up when rewriting the firmware.

· Write debug SWD terminal (SWDIO, SWCLK)
→ For ST-Link connection of firmware writing.

· Reset NRST
→ Normally released state. Reset with GND connection.
In order to prevent chattering, place a ceramic condenser of about 0.1 uF in parallel with the reset button.

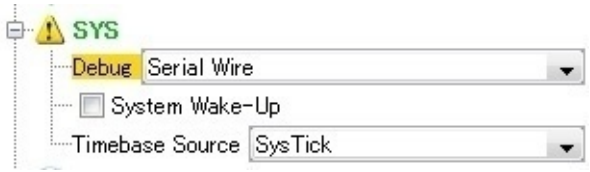
· VBAT terminal for holding real time clock
→ Connect a button battery of 3 V or the like.

17-2. Serial Debug

When serial debugging is invalid in the setting of CubeMX, when rewriting the farm, it is necessary to switch Boot mode by turning on the power with BOOT 0 set to High.

Although it is unintended rewriting prevention and security strengthening, it may be troublesome to switch jumpers and switches of Boot 0. In that case, select "Serial Wire" by Debug "SYS" item on CubeMX "Pinout" tab.

Activating serial debugging makes it possible to rewrite the farm without having to switch Boot 0. However, in this case, since the microcontoller can be accessed all the time through the SWD terminal and so on, Debug should be enabled for experimental applications other than writing the farm as a product.

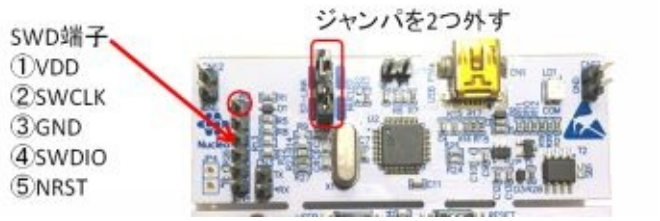


17-3. Write Firm

When writing to the microcontoller on the self-made board, write with ST-Link via the SWD terminal.

On the NUCLEO board, ST-Link is installed and it is possible to write to another STM microcontoller other than the microcontoller on the NUCLEO board. If there is a NUCLEO board or Discovery board, a dedicated programming writer is unnecessary.

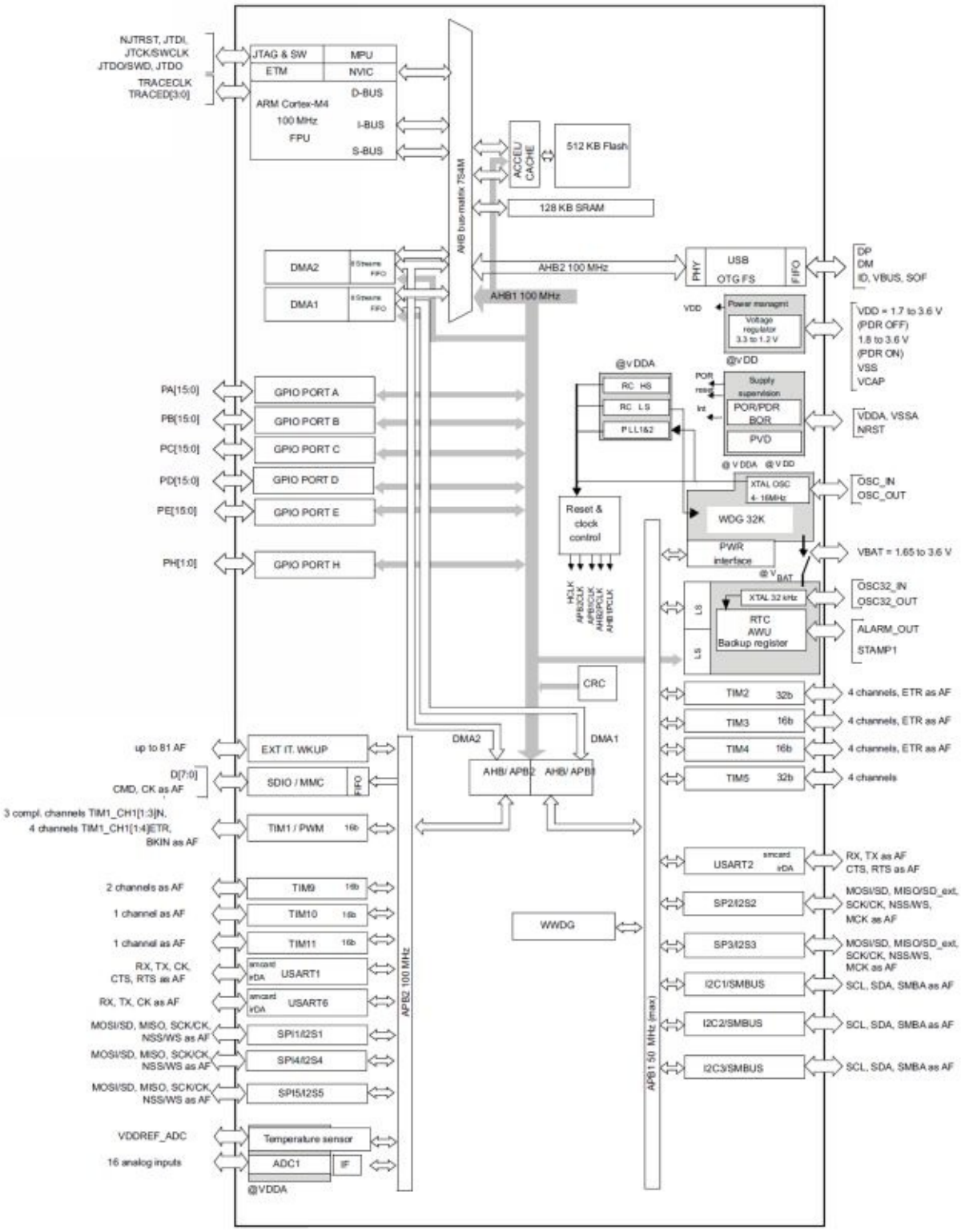
To write to other than the microcontoller on the NUCLEO board, remove the two jumpers shown below and connect the SWD terminal to the SWD terminal on your own board. At minimum, it is necessary to connect VDD, SWCLK, SWDIO and GND. VDD is not supplied from ST-Link, and ST-Link is used to monitor the voltage on its own board. Therefore, when writing, it is necessary to execute writing from ST-Link with power supply on self-made board.



# 18. Conclusion

Automatic program generation from ARM offline development environment creation CubeMX, introduction of FreeRTOS, introduction of major peripherals and main functions from use of SD card were introduced. Although some functions could not be introduced this time, if you learn how to use the development environment SW4STM32 and CubeMX itself, you will be able to implement while reading sample programs and data sheets in CubeMX. If you can master the powerful and inexpensive ARM microcontoller STM 32, please use the ARM microcontoller STM 32 and try creating a nice device that is not yet in the world.

# Appendix




STM32F411 Block Diagram (Extract from en.DM00115249.pdf)