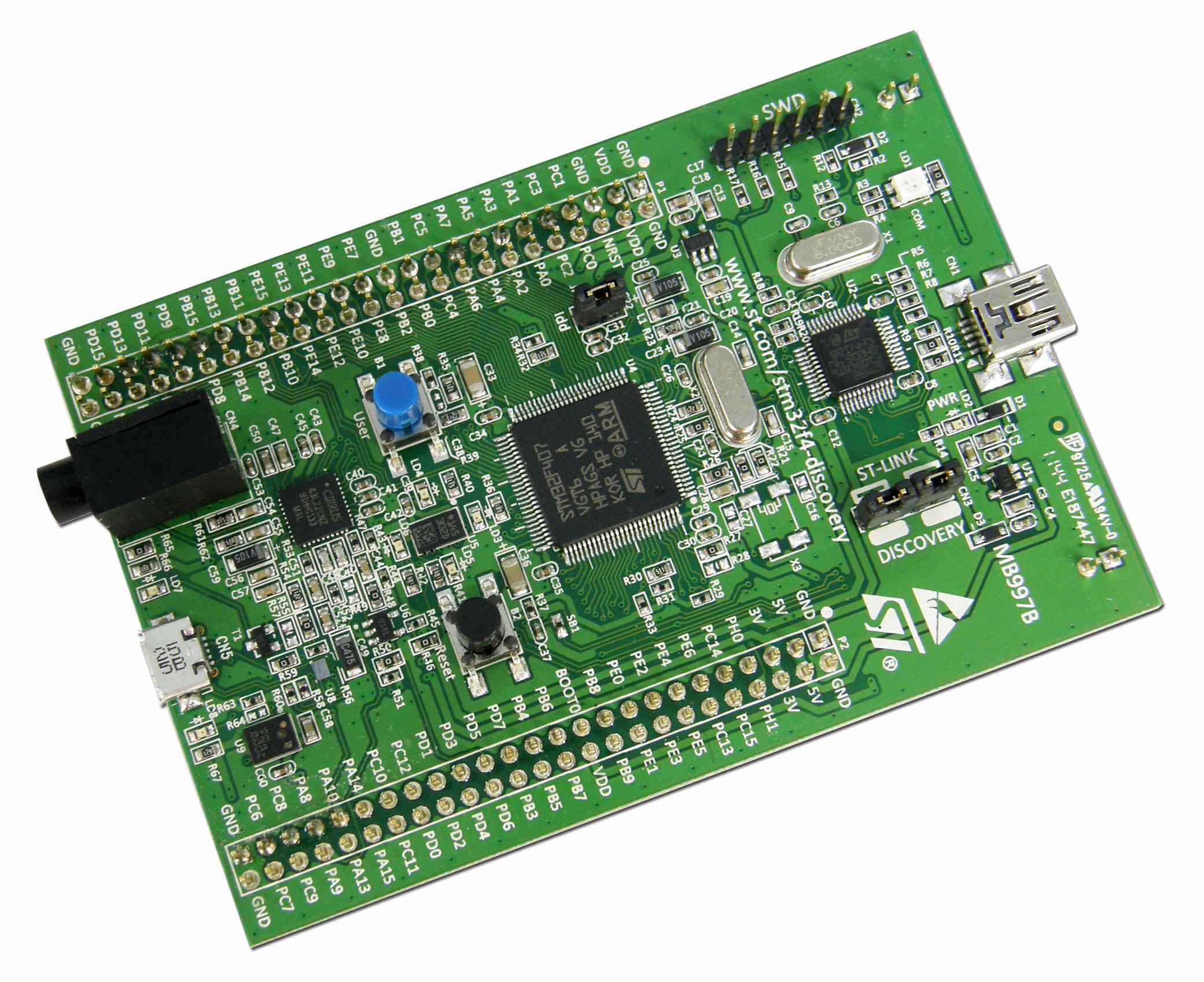
24/3/2017



ARM PROGRAMMING TUTORIAL STM32F4 DISCOVERY

ARM PROGRAMMING TUTORIAL – STM32F4 DISCOVERY

BUI VIET HA – FGA.LSI

**CONTENT**

[INTRODUCTION 1](#_Toc478455904)

[PART I: GETTING STARTED 2](#_Toc478455905)

[CHAPTER 1: SETTING KEIL C v5 AND STM32CubeMX 3](#_Toc478455906)

[CHAPTER 2: SETTING STM32CubeMX 5](#_Toc478455907)

[PART II: STM32F4 DISCOVERY TUTORIAL 8](#_Toc478455908)

[CHAPTER 1: GENERAL PURPOSE INPUT OUTPUT GPIO 8](#_Toc478455909)

[1.1. Register Description: 8](#_Toc478455910)

[1.2. Creating new Project: 8](#_Toc478455911)

[1.3. Configuration: 9](#_Toc478455912)

[1.4. Code: 9](#_Toc478455913)

[CHAPTER 2: EXTERNAL INTERRUPT 11](#_Toc478455914)

[2.1. Register description : 11](#_Toc478455915)

[2.2.Configuration 11](#_Toc478455916)

[2.3. CODE : 12](#_Toc478455917)

[CHAPTER 3: TIMER/COUNTER 13](#_Toc478455918)

[3.1. Register description: 13](#_Toc478455919)

[3.2. Configuration : 13](#_Toc478455920)

[3.3. Timer Handling 14](#_Toc478455921)

[3.4. CODE 14](#_Toc478455922)

[CHAPTER 4: UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSCEIVER (UART) 15](#_Toc478455923)

[4.1. Register description 15](#_Toc478455924)

[4.2. Configuration 15](#_Toc478455925)

[4.3: Code : 16](#_Toc478455926)

[CHAPTER 5: SERIAL PERIPHERAL INTERFACE (SPI) 18](#_Toc478455927)

[5.1. SPI Interface introduction: 18](#_Toc478455928)

[5.3. CODE 19](#_Toc478455929)

# INTRODUCTION

This reference manual targets application developers. It provides complete information on how to use the STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx microcontroller memory and peripherals.  
The STM32F405xx/07xx, STM32F415xx/17xx, STM32F42xxx and STM32F43xxx constitute a family of microcontrollers with different memory sizes, packages and peripherals.  
For ordering information, mechanical and electrical device characteristics please refer to the datasheets.  
For information on the ARM Cortex®-M4 with FPU core, please refer to the Cortex®-M4 with FPU Technical Reference Manual.

# PART I: GETTING STARTED

In this part, you will learn some basic knowledge about ARM programming. The tutorial uses STM32F4-DISCOVERY development board (Figure 1) and Keil uvision 5 and STM32CubeMX.

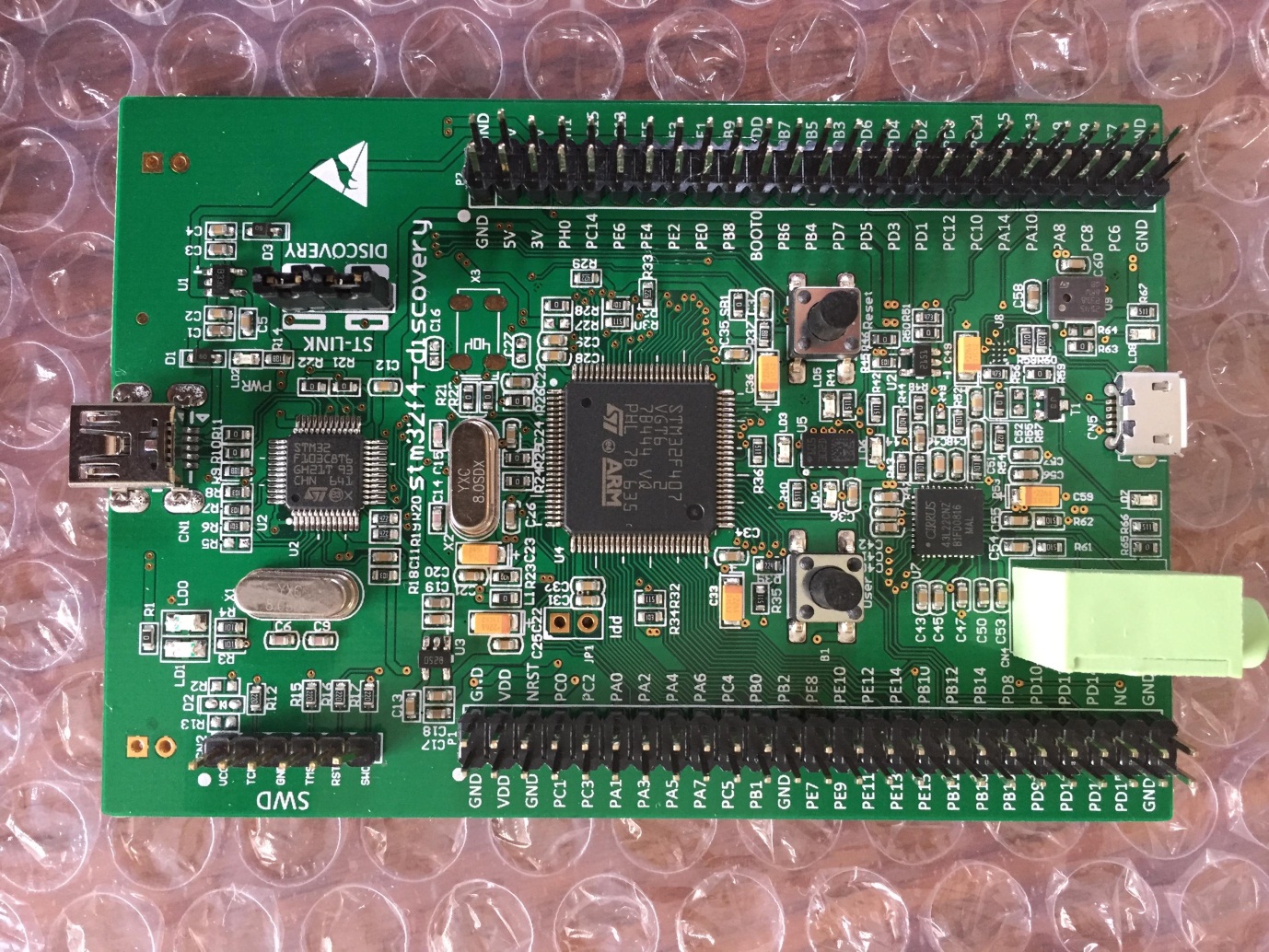
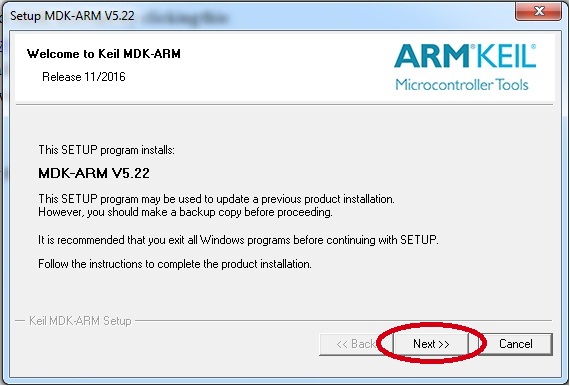


Figure 1. STM32F407-Discovery development kit.

# CHAPTER 1: SETTING KEIL C v5 AND STM32CubeMX

* Download MDK software by clicking this: <https://www.keil.com/demo/eval/arm.htm>.
* Fill the information and formation follow steps to get download link.
* When downloading done, open file MDK.exe then follow instructions to set up IDE.



*Figure 1.1: Setup Keil uVision 5 IDE.*

* Finishing this, open file Keil uVision 5 and a window would be shown up:

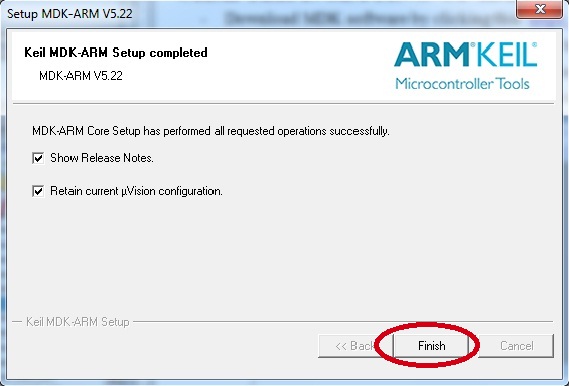


Figure1.2 : Pack setup completed window

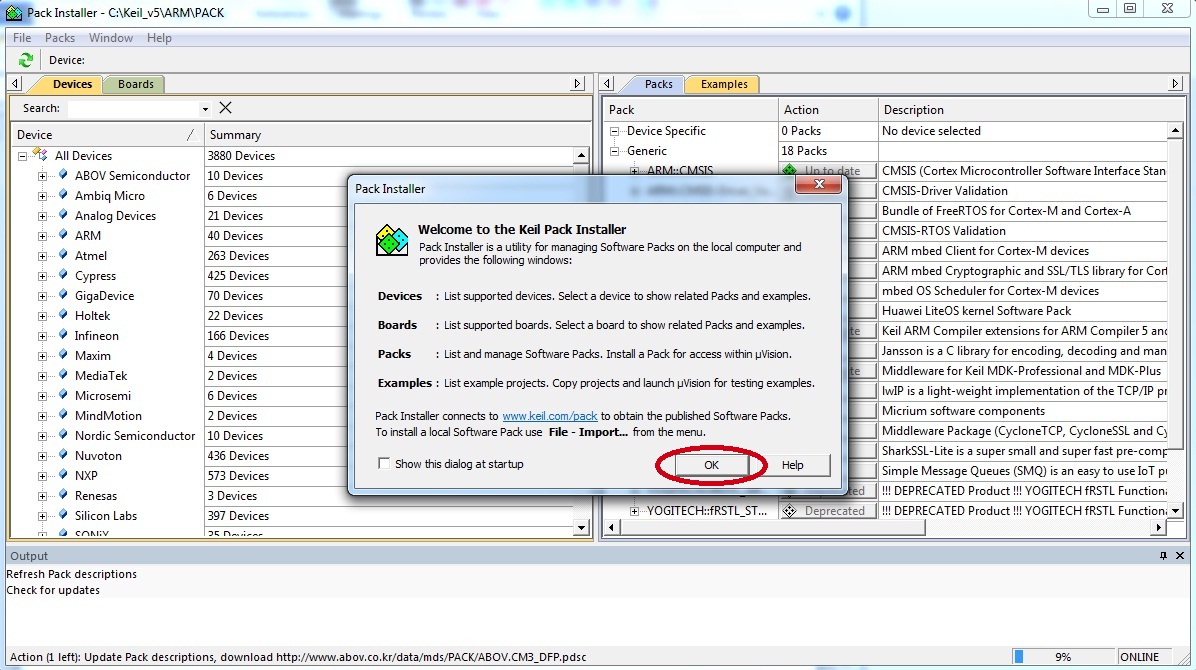
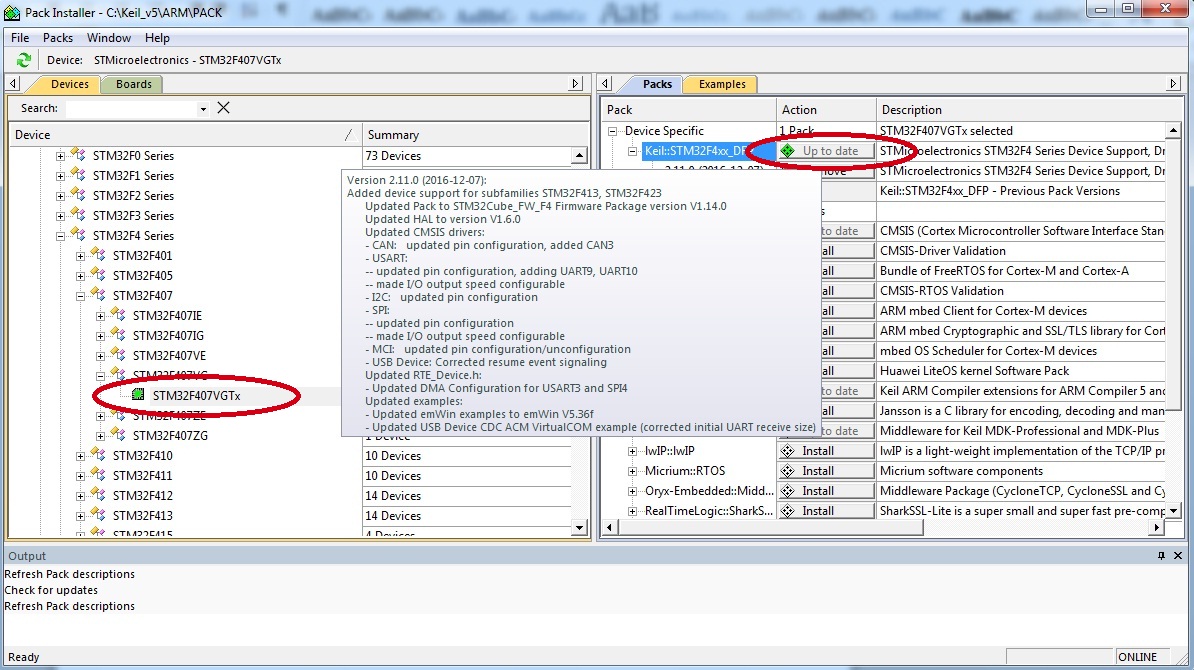


Figure 1.3 : Pack installer window

After the MDK Core installation is complete, the [Pack Installer](http://www.keil.com/mdk5/packinstaller) is started automatically, which allows you to add supplementary Software Packs. As a minimum, you need to install a Software Pack that supports your target microcontroller device.

On the **Devices** tab, select the microcontroller you wish to use. Then press the **Install** button of the corresponding Device Family Pack (DFP) on the **Packs** tab:



*Figure 1.4: Pack Installer window.*

# CHAPTER 2: SETTING STM32CubeMX

To install STM32CubeMX, follow the steps below:

1. Download STM32CubeMX installation package from [http://www.st.com/en/development-tools/stm32cubemx.html?sc=stm32cubemx](www.st.com/stm32cubemx)

2. Extract (unzip) stm32cubemx.zip whole package into the same directory.

3. Check your access rights and launch the installation wizard:

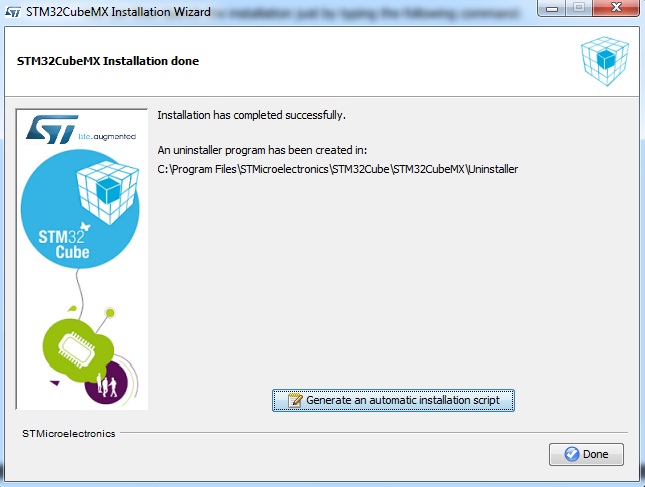
a) Make sure you have administrators rights.

b) Double-click the SetupSTM32CubeMX-VERSION.exe file to launch the installation wizard :



**Auto-install mode :**

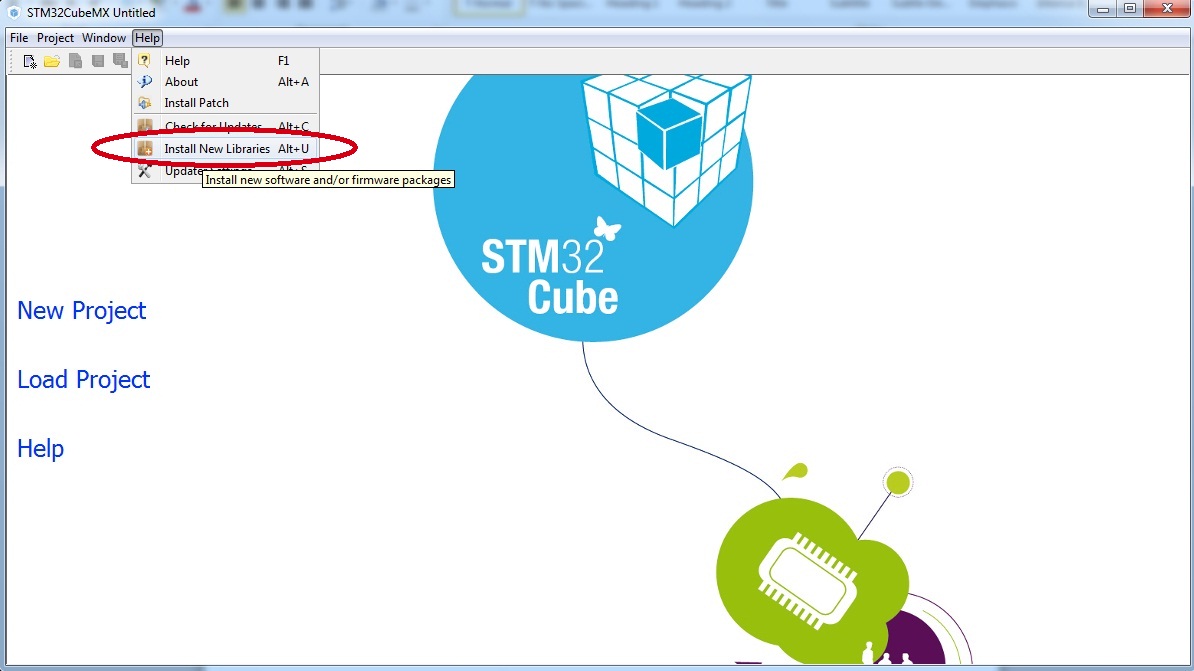
At end of an installation, performed either using STM32CubeMX graphical wizard or console mode, it is possible to generate an auto-installation script containing user installation preferences :



**Downloading new libraries**

To download new libraries, follow the steps below:

1. **Select Help > Install New Libraries** to open the **New Libraries Manager** window.

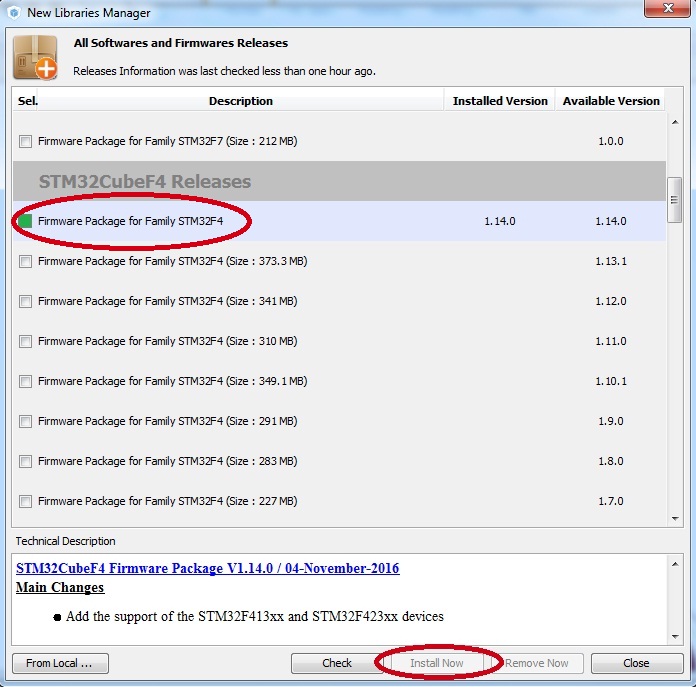


If the installation was performed using STM32CubeMX, all the packages available for download are displayed along with their version including the version currently installed on the user PC (if any), and the latest version available from <http://www.st.com>.

If no Internet access is available at that time, choose “Local File”. Then, browse to select the zip file of the desired STM32Cube firmware package that has been previously downloaded from st.com. An integrity check is performed on the file to ensure that it is fully supported by STM32CubeMX.

The package is marked in green when the version installed matches the latest version available from <http://www.st.com>.

2. Click the checkbox to select a package then **“Install Now”** to start the download.



# PART II: STM32F4 DISCOVERY TUTORIAL

# CHAPTER 1: GENERAL PURPOSE INPUT OUTPUT GPIO

## 1.1. Register Description:

See STM32F4 Reference manualchapter 8.

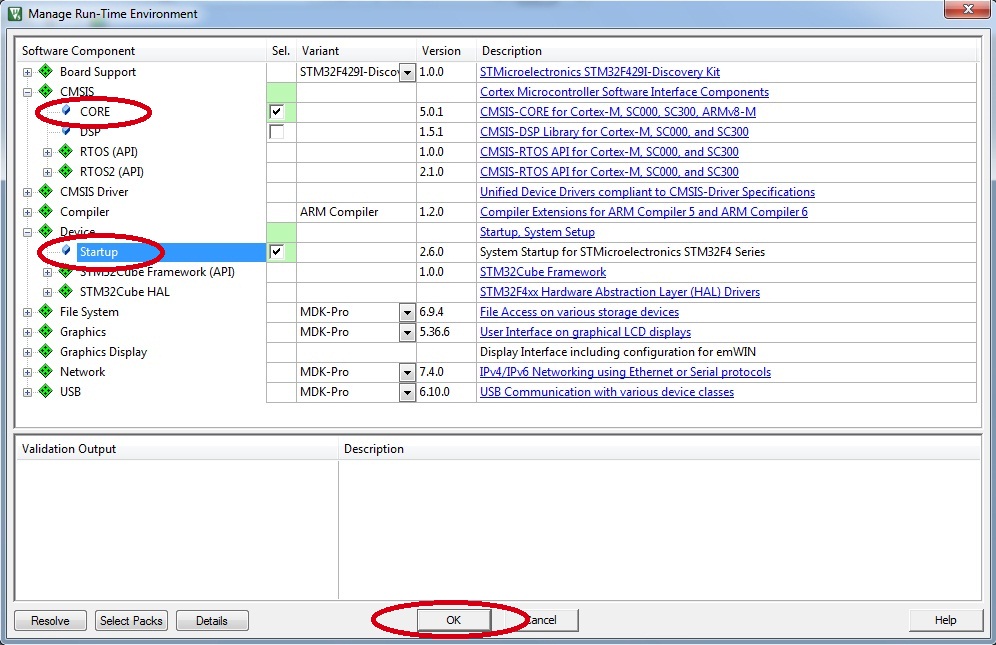
## 1.2. Creating new Project:

1. Open Keil µVision → Project → New → µVision Project.

2. Select device-> Search: STM32F407VGTx -> OK.

3. Add startup code file: *startup*

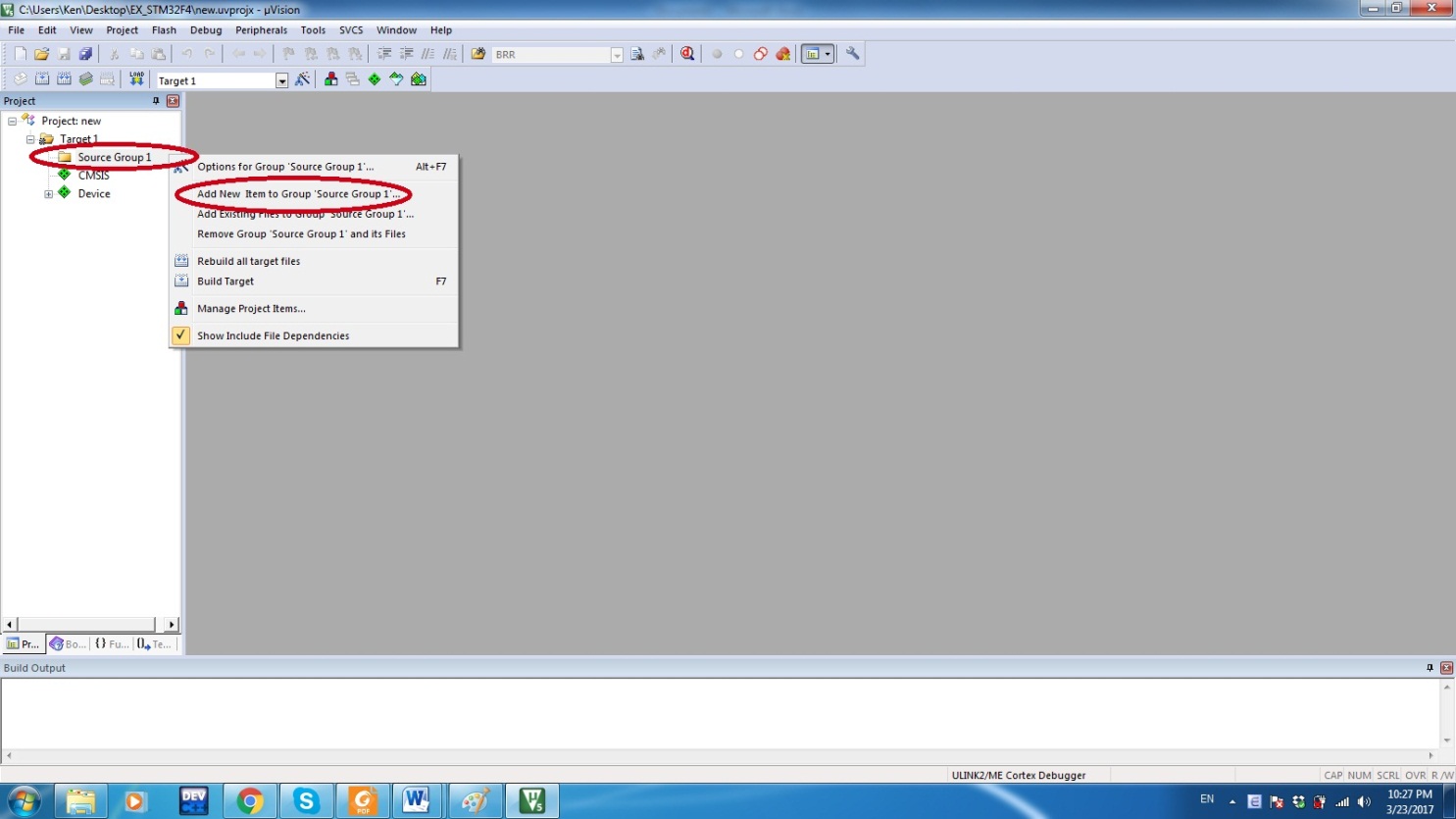
4. Add CMSIS code file: *CORE*



5. Create/Add User file: *main.c,…*

*- Click right mouse to Source group 1 – select Add New item to group 1…*

6. Configure *Include Path.*

**

## 1.3. Configuration:

1. Configure RCC register : enable clock for peripheral that we use.

- Set bit 3 for AHB1ENR register to 1 : enable clock for GPIOD

2. Configure GPIO register :

- MODER register : set mode output for PD12,PD13,PD14,PD15.

- OSPEEDR register : set high speed for PD12,PD13,PD14,PD15.

- OTYPER register : set mode push pull for PD12,PD13,PD14,PD15.

- Modify GPIO by setting RCC and GPIO registers (details in datasheet chapter 7 and chapter 8).

## 1.4. Code:

//main.c

#include <stm32f4xx.h>

int i;

// Create delay function

void delay\_ms (int ms)

{

while (ms-- > 0)

{

volatile int x = 500;

while (x-- > 0);

}

}

int main()

{

**/\* SystemCoreClockUpdate() updates the SystemFrequency variable \*/**

SystemCoreClockUpdate();

//enable clock for GPIOD : RCC->AHB1ENR register

RCC->AHB1ENR |= 1<<3;

//select mode output for pin GPIOD from MODER register

GPIOD->MODER |= 1<<24; //ouput mode for PD12

GPIOD->MODER |= 1<<26; //ouput mode for PD13

GPIOD->MODER |= 1<<28; //ouput mode for PD14

GPIOD->MODER |= 1<<30; //ouput mode for PD15

// select speed output for pin GPIOD

GPIOD->OSPEEDR |= 0x0002<<24; // set high speed for pin PD12

GPIOD->OSPEEDR |= 0x0002<<26; // set high speed for pin PD13

GPIOD->OSPEEDR |= 0x0002<<28; // set high speed for pin PD14

GPIOD->OSPEEDR |= 0x0002<<30; // set high speed for pin PD15

// select mode push pull for output for PD12, PD13,PD14,PD15

GPIOD->OTYPER &= ~(1<<15 | 1<<12 | 1<<13 | 1<<14);

While(1)

{

//Reset all port output data for GPIOD

GPIOD->ODR &= ~0xFFFF;

//Set pin PD15 output data

GPIOD->ODR |= 1<<15;

//Delay 1s

Delay\_ms(1000);

//Reset all port output data for GPIOD

GPIOD->ODR &= ~0xFFFF;

//Set pin PD12 output data

GPIOD->ODR |= 1<<12;

//Delay 1s

Delay\_ms(1000);

//Reset all port output data for GPIOD

GPIOD->ODR &= ~0xFFFF;

//Set pin PD13 output data

GPIOD->ODR |= 1<<13;

//Delay 1s

Delay\_ms(1000);

//Reset all port output data for GPIOD

GPIOD->ODR &= ~0xFFFF;

//Set pin PD14 output data

GPIOD->ODR |= 1<<14;

//Delay 1s

Delay\_ms(1000);

}

}

# CHAPTER 2: EXTERNAL INTERRUPT

## 2.1. Register description :

- RCC register and GPIO register.

- SYSCFG register : external interrupt configuration register.

- EXTI register.

- Details in datasheet : chapter 7- RCC register, chapter 8- GPIO register, chapter 9- System configuration controller and chapter 12 – Interrupts and events.

2.2.Configuration **:**

**2.2.1. Initialization :**

1. Configuration pin for Button : RCC\_Register

- Enable clock for pin button : enable PA0 for button 2(B2) use RCC register.

- select mode input for pin Button.

- Set speed, pull-up,pull-down or no pull-up, no pull-down…

2. Select source input for EXTI external interrupts : SYSCFG\_Register.

- Use SYSCFG register .

- Select port and pin for EXTI external interrupt.

3.Select Mask or Unmask for interrupt :

- Use EXTI\_IMR register.

- Select mask or unmask for interrupt on line.

- Set bit MR0 to interrupt request from line 0 is not mask : MR0=1.

4.Select mode interrupt : EXTI\_MODE

- Select mode interrupt Rising trigger selection register : EXTI\_RTSR register.

- Set bit TR0 to rising trigger enable : TR0=1

5. Set priority for interrupt :

NVIC\_SetPriority(EXTI0\_IRQn,1);

6.Enable NVIC interface

NVIC\_EnableIRQ(EXTI0\_IRQn);

**2.2.2. Interrupt handling :**

- Delay time.

- Set bit PR to clear interrupt 0.

- Add user code here.

## 2.3. CODE :

// setup pin PA0\_Button 2 to EXTI0 \_ LINE 0

void setup\_EXTI0\_PA0(void)

{

//enable clock for GPIOA

RCC->AHB1ENR |= 0x0001;

//select source input for EXTI external interrupt

SYSCFG->EXTICR[1] &= ~(0x000F);

SYSCFG->EXTICR[1] |= SYSCFG\_EXTICR1\_EXTI0\_PA;

// select unmask unterrupt

EXTI->IMR |= 0x0001;

// select mode interrupt trigger to rising edge

EXTI->RTSR |= 0x0001;

//set priority for the interrupt

NVIC\_SetPriority(EXTI0\_IRQn,1);

// enable Nvic interface

NVIC\_EnableIRQ(EXTI0\_IRQn);

}

/\* i is the number press button

int i ;

void EXTI0\_IRQHandler(void)

{

delay\_ms(500);

//Clear interrupt 0

EXTI->PR |= 0x0001;

i++;

}

void toggleLED(void)

{

switch(i%4)

{

case 0 : // Press button 0

blink\_led(15);

break;

case 1 : // Press button 1

blink\_led(12);

break;

case 2 : //press button 2

blink(13);

break;

case 3 : //press button 3

blink(14);

break;

}

delay\_ms(200);

}

int blink\_led(int i)

{

GPIOD->ODR &= ~(0xFFFF);

GPIOD->ODR |= 1<<i;

return i;

}

# CHAPTER 3: TIMER/COUNTER

## 3.1. Register description:

**-** RCC register, GPIO register, TIM2 register.

## 3.2. Configuration :

1. Register APB1ENR :

- Set bit TIM2EN to enable Timer/counter2, power/clock control.

2. Register CR1 (TIM2 Register) :

- Set bit CEN to 1 : counter enable, CEN=1.

- Set bit ARPE to 1 : ARPE=1 set auto-reload preload.

- Set bit DIR to 0: counter used as upcounter.

- Set bit8-9 to choose value Clock division for Peripheral Clock.

3. Register PSC ,TIM2 register :

- Configure Prescaler is 41999.

4.Register CNT :

- Configure CNT register : counter value is 0.

5.Register ARR :

- Configure ARR register : set auto reload is 199.

6. Register DIER :

- Set bit UIE to enable : UIE =1 update interrupt enable.

7. Enable interrupt for TIM2 :

- NVIC\_EnableIRQ(TIM2\_IRQn)

3.3. Timer Handling **:**

1. Register SR in TIM2 register :

- Reset interrupt flag in TIM.2

- Check the LED interface state

- If the LED interface state is on : turn it off.

- else turn it on.

3.4. CODE **:**

**//main.c**

#include <stm32f4xx.h>

#include <timer2.h>

void Tim2\_config(void)

{

//enable clock for Timer 2

RCC->APB1ENR |= 0x0001;

/\* -- Set TIM2 to 10khz(0.1ms) interrupt --

++ Down timer 1mhz with prescaler

timer\_tick\_frequency = Timer\_default\_frequency / (prescaller\_set + 1)

=> 1000000 = 84000000 / (prescaller\_set + 1)

=> prescaller = 83

++ Down timer 10khz with timer period

tim\_frequency = timer\_tick\_frequency / (TIM\_Period + 1)

==> 10000 = 1000000 /(TIM\_Period +1)

==> TIM\_Period = 1000000 / 10000 – 1

==> TIM\_Period = 99

++ Frequency(update event) = (clock)/((PSC+1)\*(ARR+1)) \*/

//set Prescaler

TIM2->PSC = 41999;

//set counter

TIM2->CNT = 0;

//set Auto reload

TIM2->ARR = 199;

// enable TIM.2 couter

TIM2->CR1 |= 0x0001;

TIM2->CR1 |= 1<<7; // enable auto-reload

TIM2->CR1 &= ~(1<<4); // counter used as upcounter

//enable interrupt on update event

TIM2->DIER |= 0x0001;

//enable interrupt for TIM.2

NVIC\_EnableIRQ(TIM2\_IRQn);

}

void TIM2\_IRQHandler(void)

{

//Reset interrupt flag in TIM.2

TIM2->SR = 0x0000;

//check the led interface state

if((GPIOD->ODR & (GPIO\_ODR\_OD15 | GPIO\_ODR\_OD12|GPIO\_ODR\_OD13|GPIO\_ODR\_OD14)) != 0 )

//if the led interface state is on, turn it off

GPIOD->ODR &= ~(0xF000);

//else turn it on

else GPIOD->ODR |= (0x000F<<12);

}

# CHAPTER 4: UNIVERSAL ASYNCHRONOUS RECEIVER/TRANSCEIVER (UART)

4.1. Register description**:**

- RCC register : AHB1ENR, APB1ENR, APB2ENR register.

- GPIO register : MODER, PUPDR, OTYPER, OSPEEDR, AFR register.

- USART register : CR1 register, BRR register, DR…

4.2. Configuration **:**

4.2.1. UART Initialization :

1. RCC register :

- Enable bit 1 to configure GPIOA.

- Enable bit 17 to configure USART\_2.

- Enable bit 14 to configure SYSCFG.

2. MODER register :

- Enable bit 4 and 5 to configure PA.2 (TX) : Alternate function mode.

- Enable bit 6 and 7 to configure PA.3 (RX) : Alternate function mode.

3. OTYPER register : enable bit 2 to configure PA.2 : output push pull.

4. PUPDR register : enable bit 6 and 7 to configure PA.3 to input floating.

5. OSPEEDR register :

- Set PA.2 and PA.3 to high speed.

6. AFR register :

- Set PA.2 and PA.3 to alternate function AF7.

7. CR1 register :

- Enable bit 2 and 3 to configure pin TX and RX for usart.2

- Set bit 15 to 0: select oversampling mode bye 16.

- Set bit 12 to 0 : set Word length data : 1 start bit,8 data bits, n bit stop.

8. BRR register :

- Set baudrate for usart to : 9600.

9. CR1 register : enable bit 13 to 1 : enable USART 2

4.2.2. UART Send:

- Write data needs to be sent to DR (8bits).

## 4.3: Code :

#include <stm32f4xx.h>

void USART2\_Config(void)

{

// enable clock for GPIOA, PA.2 and PA.3

RCC->AHB1ENR |= 0x0001;

// configuration PA.2 mode output push-pull

GPIOA->MODER |= 2<<4; // set PA.2 to alternate function mode

GPIOA->OTYPER &= ~(1<<2); // set PA.2 output push-pull

GPIOA->PUPDR |= 1<<4; // set PA.2 to pull up

// configuration PA.3 mode input floating

GPIOA->MODER |= 2<<6; //set PA.3 to alternate function mode

GPIOA->OTYPER &= ~(1<<3);// set PA.3 output push-pull

GPIOA->PUPDR |= 3<< 6; // set PA.3 to input floating

// set speed for GPIOA, PA.2 and PA.3

GPIOA->OSPEEDR |= 2<<4; // set high speed to PA.2

GPIOA->OSPEEDR |= 2<<6; // set high speed to PA.3

//set PA.2 and PA.3 to alternate function AF7

GPIOA->AFR[0] |= 7<<8;

GPIOA->AFR[0] |= 7<<12;

// enable clock for USART.2

RCC->APB1ENR |= 1<<17;

//enable system configuration controller clock : SYSCFG

RCC->APB2ENR |= 1<<14;

// set PA.2 to TX and PA.3 to RX

USART2->CR1 |= 1<<2; // enable RX for USART2

USART2->CR1 |= 1<<3; // enable TX for USART2

// select oversampling mode by 16

USART2->CR1 &= ~(1<<15);

// set Word length data: 1 start bit,8 data bits, n stop bit

USART2->CR1 &= ~(1<<12);

// Set stop bits : 1 stop bit

USART2->CR2 &= ~ (0x0002<<12);

// set baudrate

USART2->BRR = (42000000/(16\*9600)); // set baudrate to 9600

//enable USART.2

USART2->CR1 |= 1<<13;

}

void USART\_SendData( char data)

{

while(!(USART2->SR & USART\_SR\_TXE));

USART2->DR = (data &0xFF) ;

}

void USART\_SendString(char \*s)

{

// Send a string

while (\*s)

{

USART\_SendData(\*s++);

}

}

unsigned char USART\_ReceiveData()

{

while((USART2->SR & USART\_SR\_RXNE)==0);

return( USART2->DR & 0xFF);

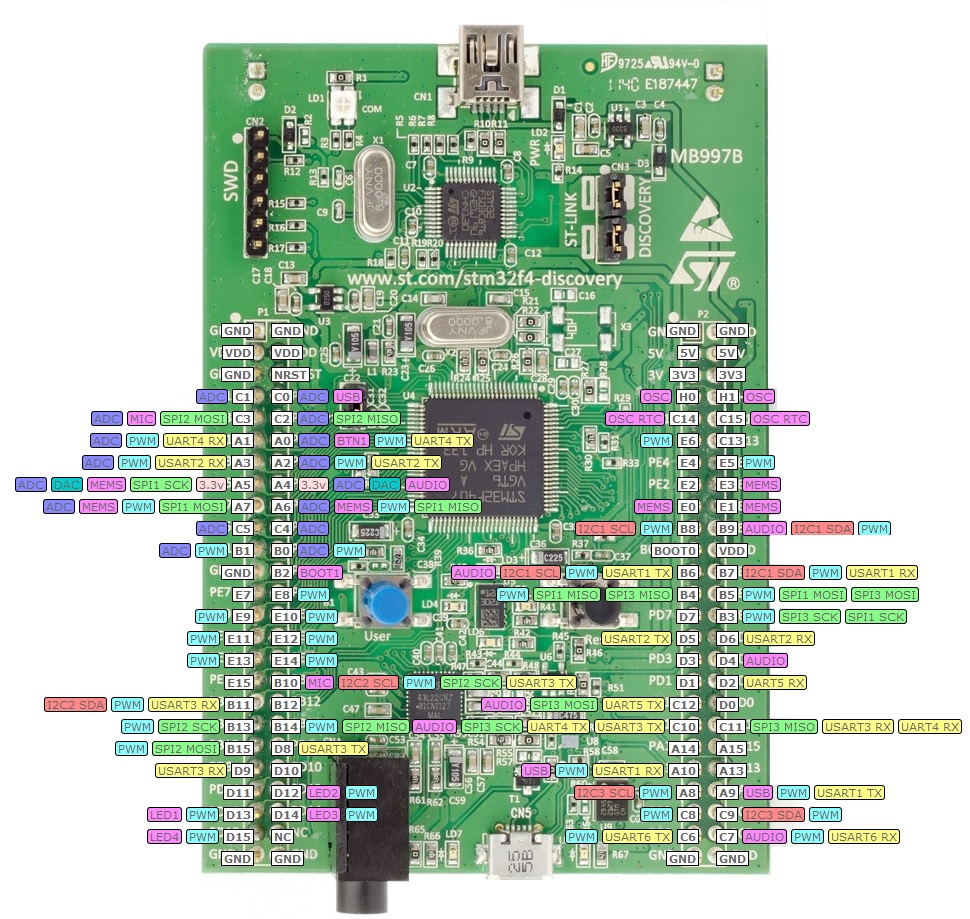
}

# CHAPTER 5: SERIAL PERIPHERAL INTERFACE (SPI)

This chapter will introduce how to use SPI peripheral and display data on LCD TFT SPI 1,44” 128\*128, 16bit color.

5.1. SPI Interface introduction:  
https://en.wikipedia.org/wiki/Serial\_Peripheral\_Interface\_Bus  
5.2 Connecting LCD TFT SPI 1,44” 128\*128 with STM32F4 Discovery:

First, let take a look at STM32F4 discovery pin map :

****

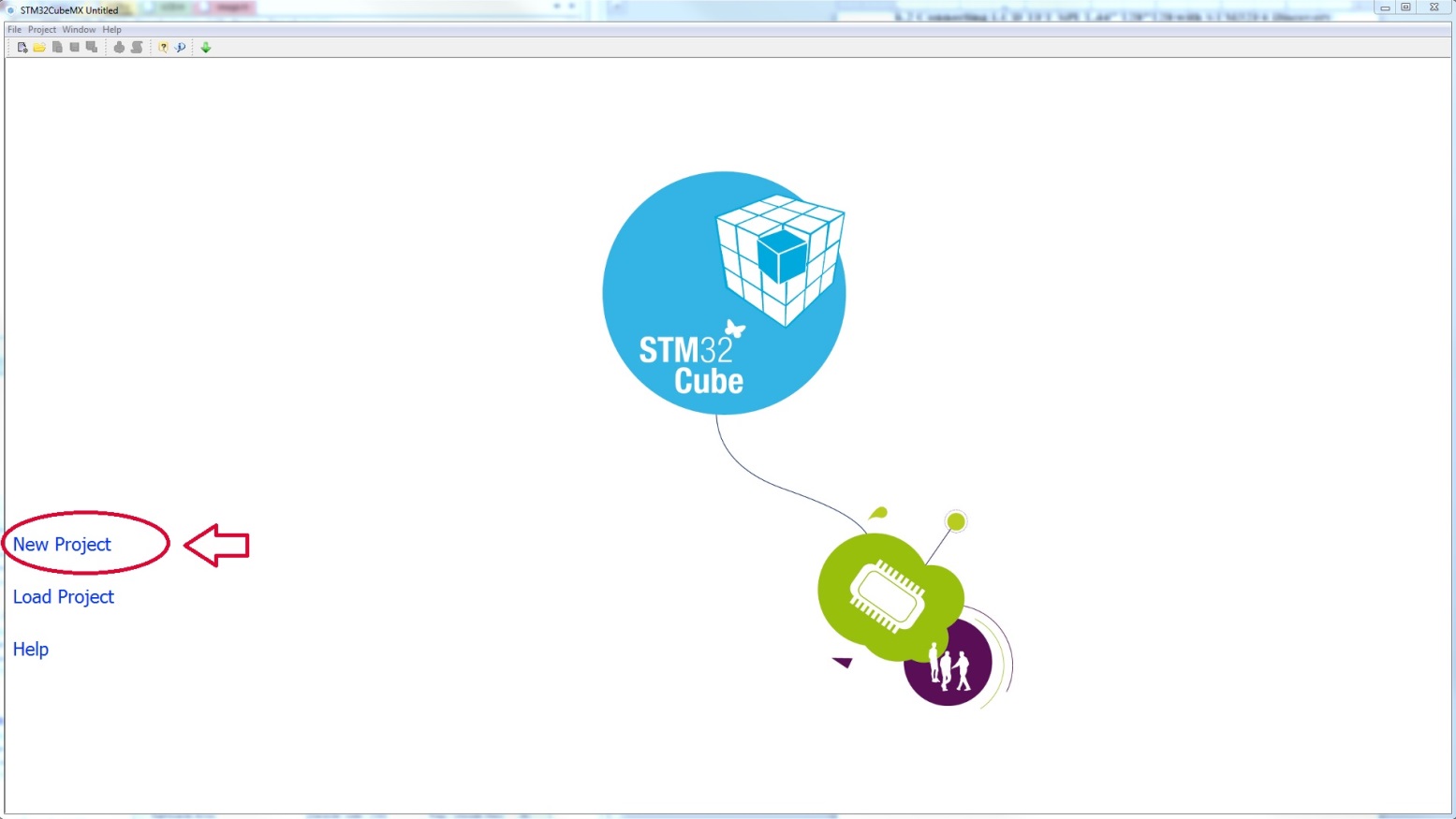
**Then read LCD you find : LCD TFT 1,44” SPI 128\*128.**

<https://www.elecrow.com/download/ILI9163DS_V018_2008-08-15-14-13-43-798.pdf>

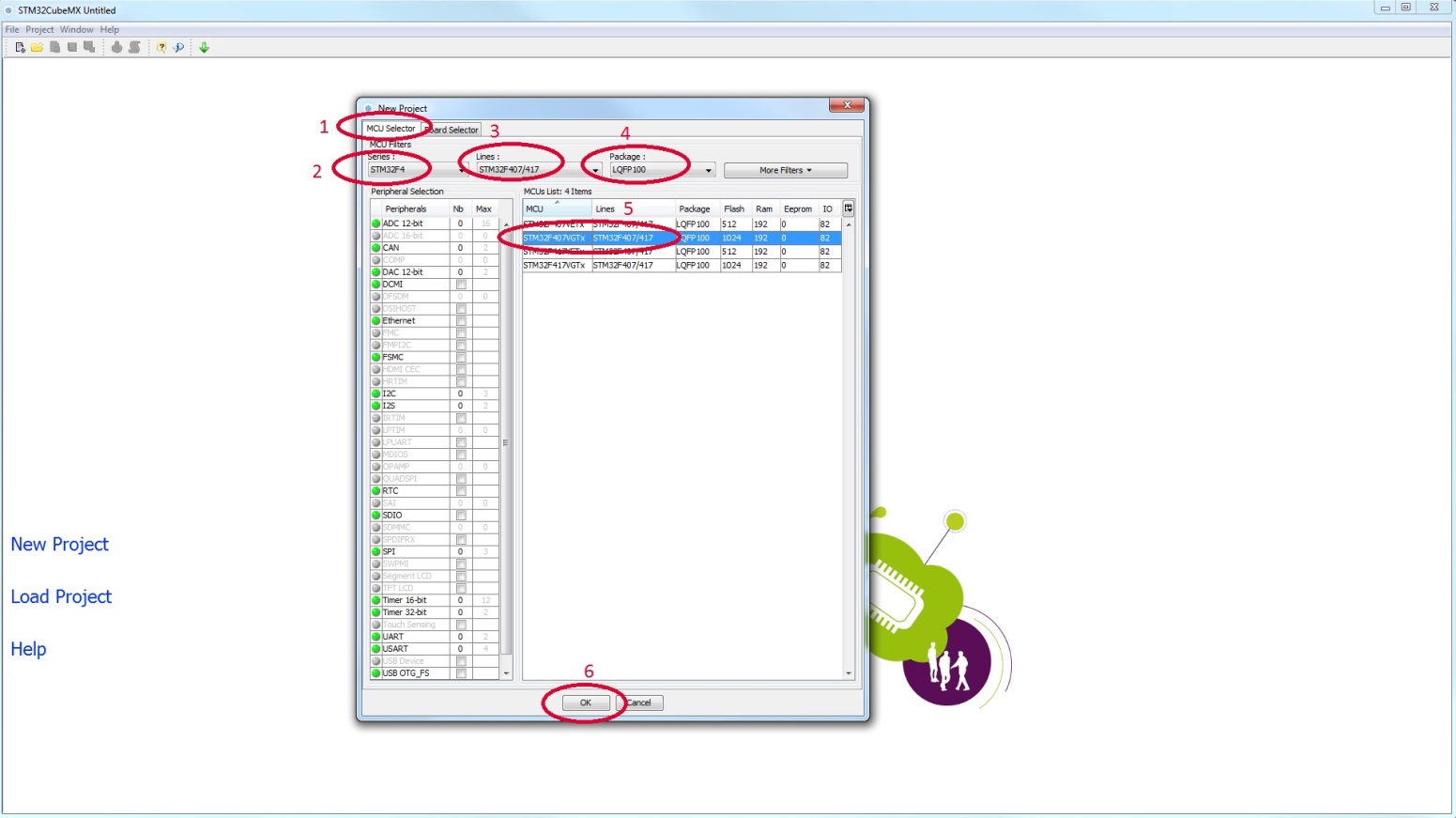
5.3. CODE **:**

**5.3.1. Configure STM32CubeMX to generate code :**

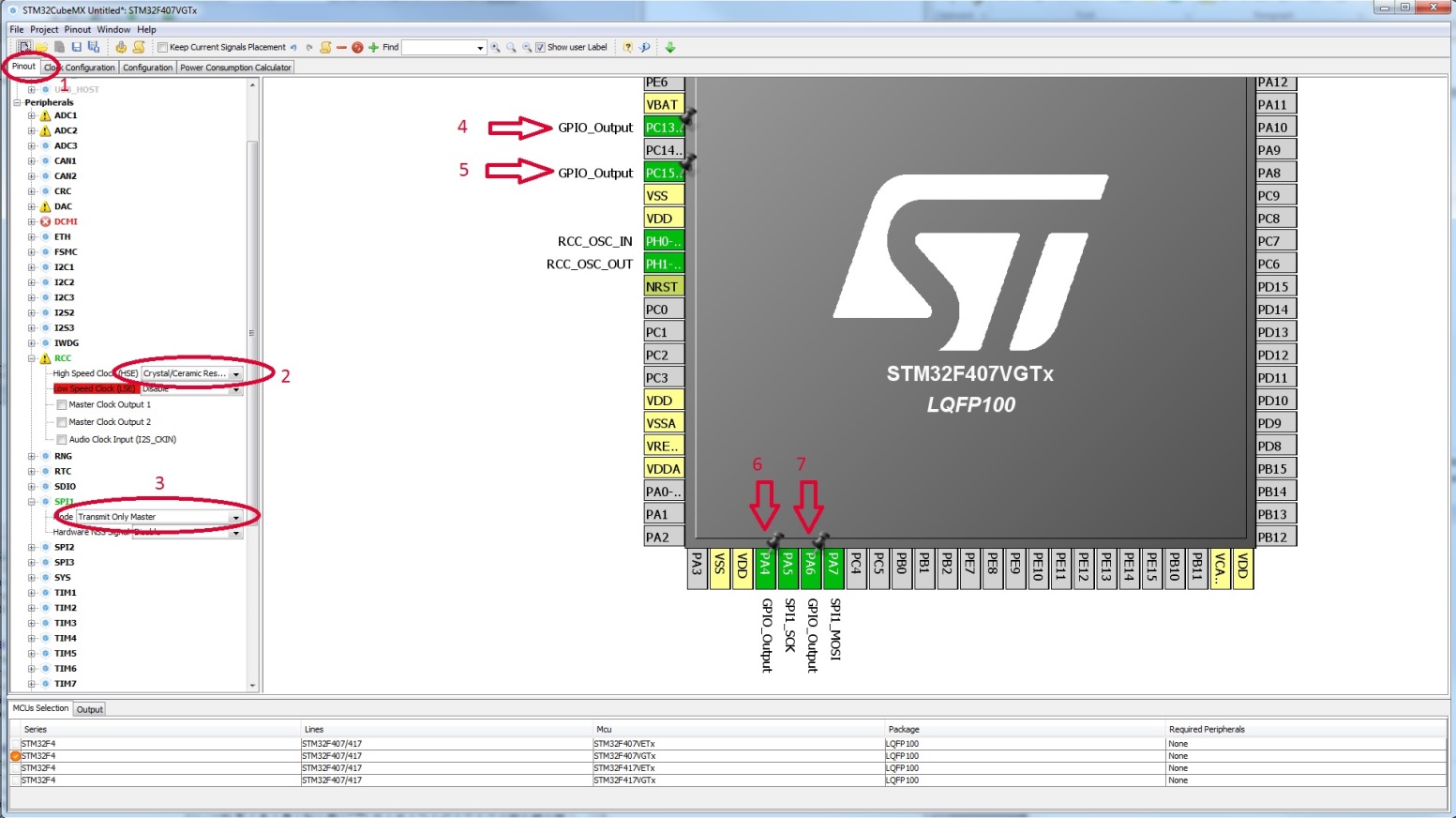
- Open STM32CubeMX → New Project :



2. Choose MCU Selector → At Series Choose STM32F4 → At Lines choose STM32F407/417 → At Package choose LPQFP100 → At MCUs list choose STM32F407VGTx → OK.



3.Pinout setting :



* 1. Setting pinout.
* 2. Select RCC : High speed clock (HSE) : Crystal/Ceramic Resonator
* 3. Select SPI1 : select mode Transmit only Master.
* 4. Select Pin PC13 to PIN LIGHT for LCD.
* 5. Select Pin PC15 to PIN RESET for LCD.
* 6. Select pin PA4 to PIN CS for LCD.
* 7. Select pin PA6 to PIN AO for LCD.
* PA5 : SCK\_SPI1 and SCK\_LCD.
* PA7 : MOSI\_SPI1 and SDA\_LCD.

4. Clock Configuration :

- 1. Click to Clock configuration.

- 2. Select input frequency is 8MHz : because use quartz oscillator external.

- 3. Select HSE : select HSE oscillator clock.

- 4. Select PLLM : PLLM/8

- 5. Select PLLN : PLLN\*336.

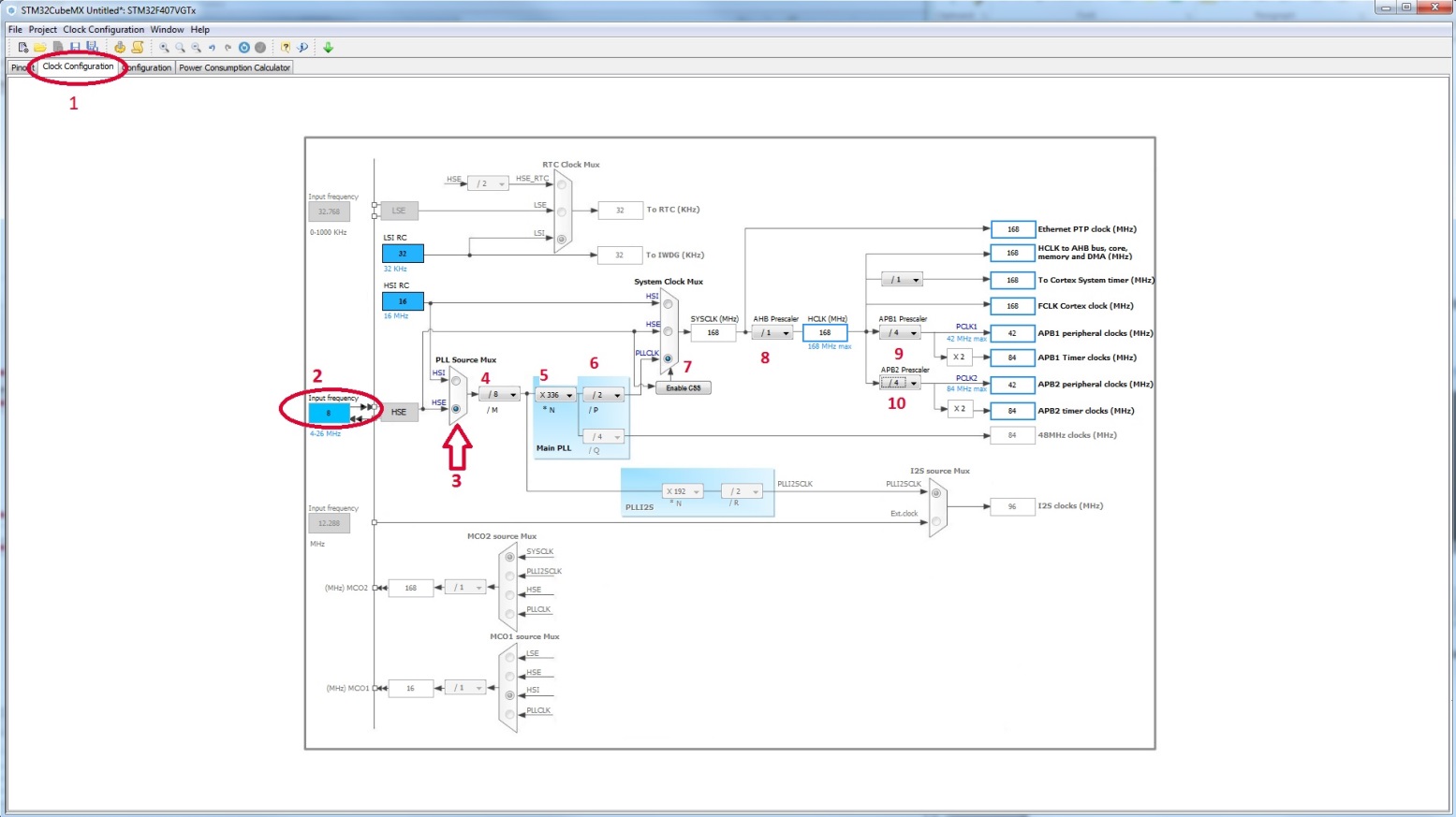
- 6. Select PLLP : PLLP/2.

- 7. Select PLLCLK

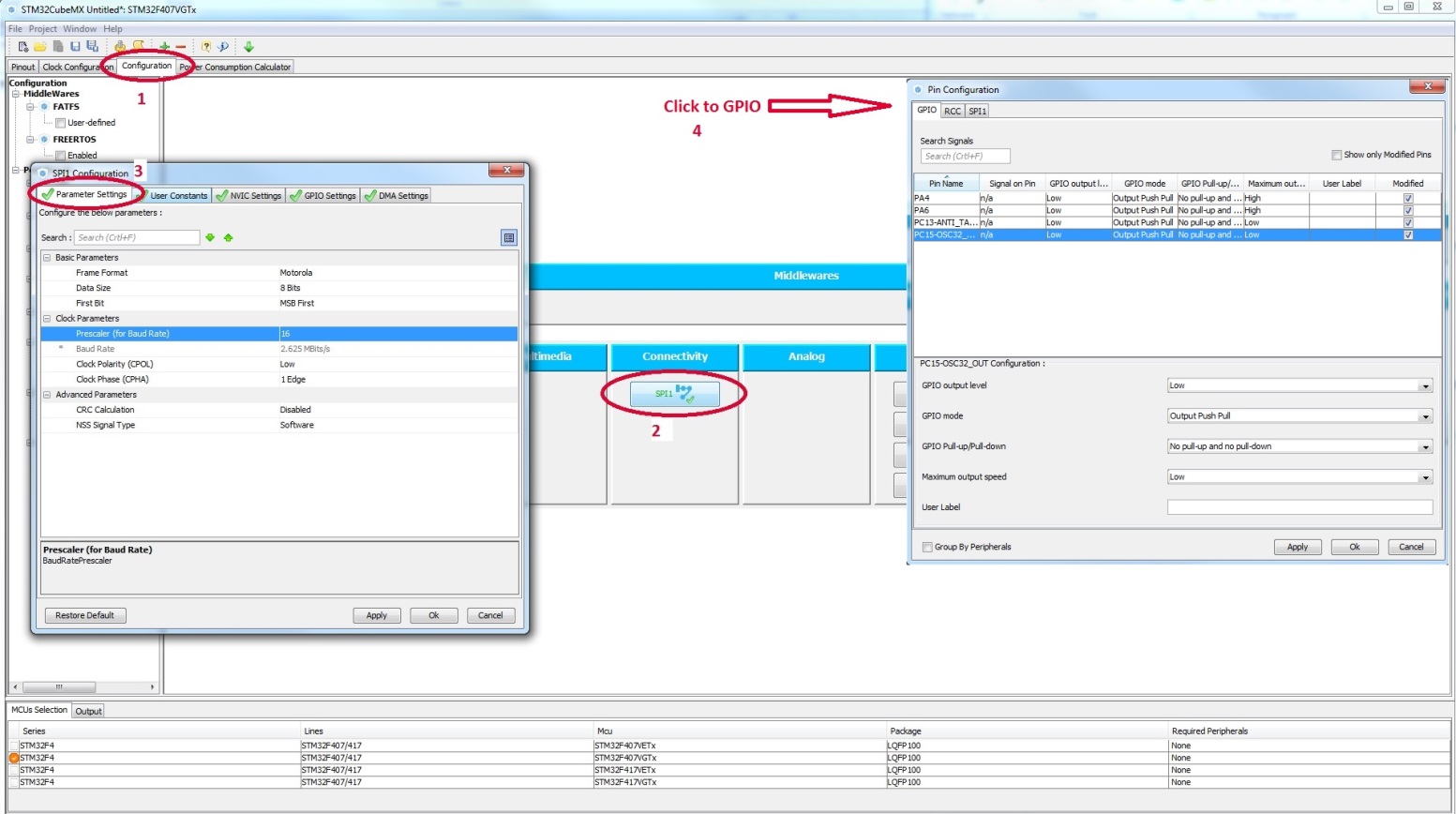
- 8. Select AHB Prescaler : div 1 for HCLK max speed :168MHz.

- 9. Select APB1 Prescaler: div 4.Because APB1 peripheral clock max speed is 42MHz.

- 10. Select APB2 Prescaler : div 4.



5. Configuration SPI and GPIO :



5.1. Click to Configuration.

5.2. Click to SPI1 :

5.3. Configure SPI1:

a. Parameter settings select :

+ Basic parameters :

- Frame format : Motorola.

- Data size : 8 bits.

- First bit : MSB first.

+ Clock parameters :

* Prescaler (for baud rate ) = 16.
* Clock polarity ( CPOL ) : low.
* Clock Phase ( CPHA ) : 1edge.

+ Advanced parameter :

* CRC calculation : Disable.
* NSS signal type : software.

5.4. Click to GPIO : configure GPIO

a.Configure Pin PA4 and PA6 :

- GPIO output level : low.

- GPIO mode : output push pull.

- GPIO pull-up/pull down : No pull-up and No pull-down.

- Maximum output speed : High.

b. Configure Pin PC13 and PC15 :

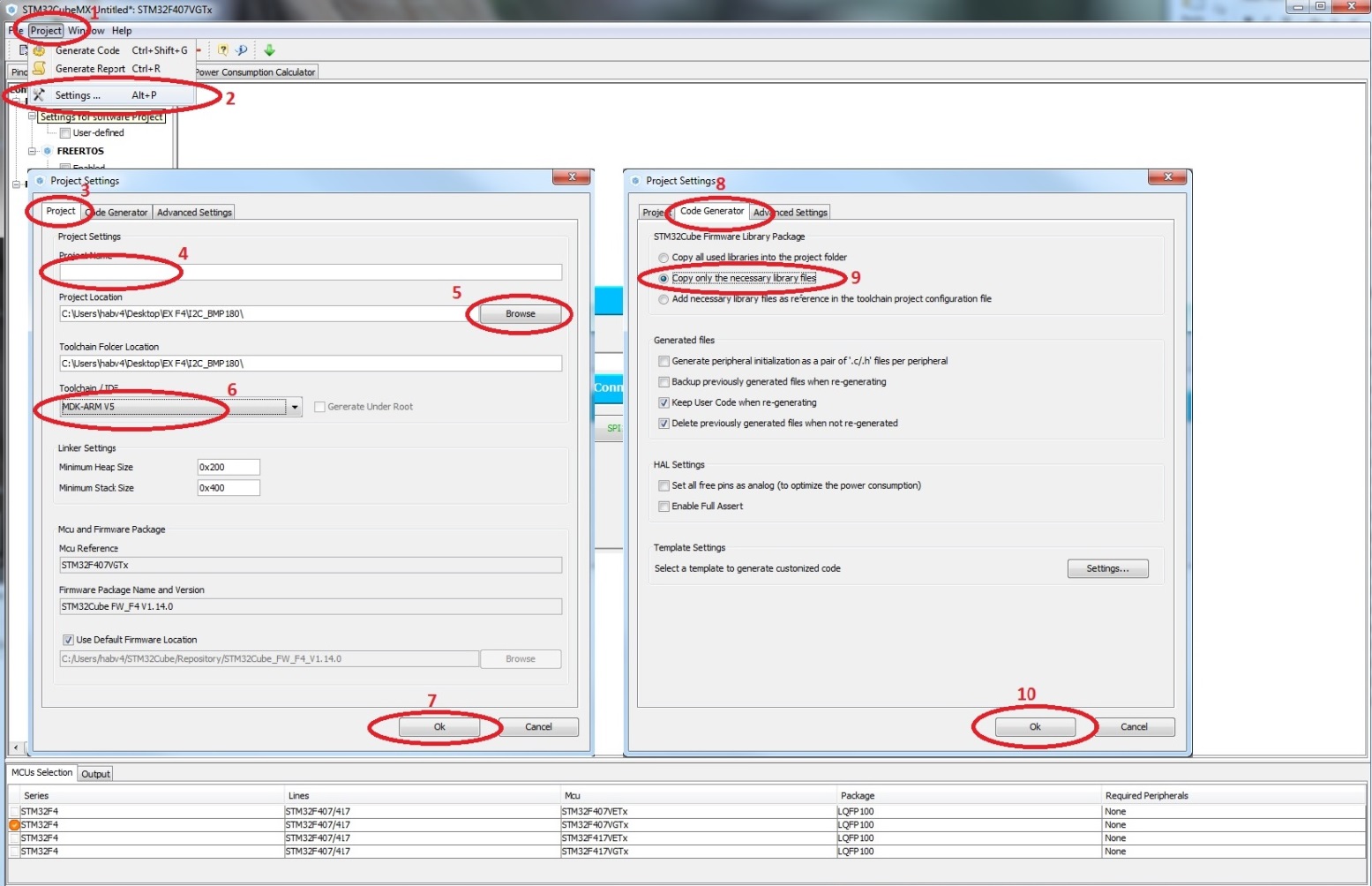
- GPIO output level : low.

- GPIO mode : output push pull.

- GPIO pull-up/pull down : No pull-up and No pull-down.

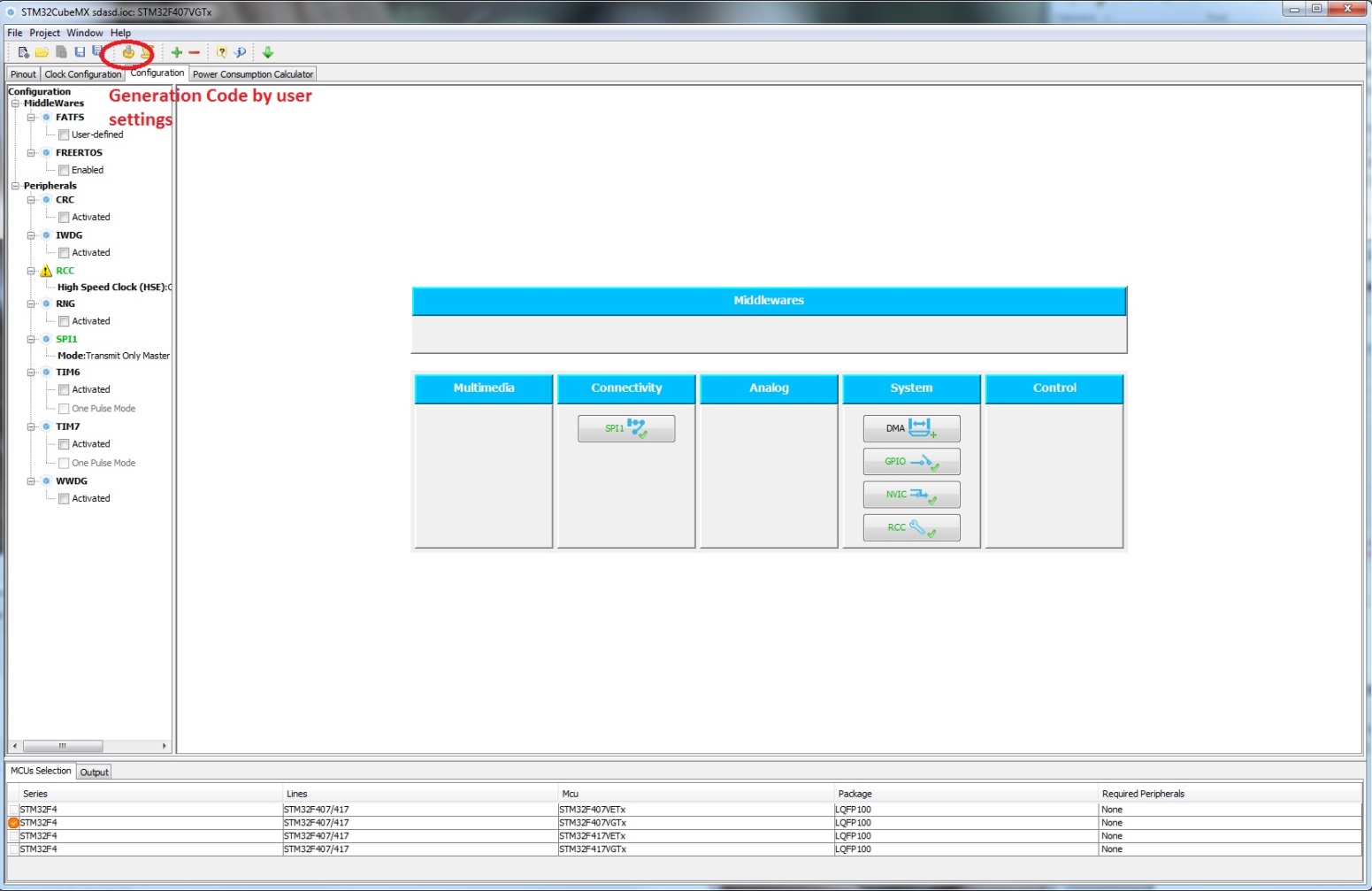
- Maximum output speed : LOW.

6. Setting Project :



-Configure by image.

7.Generation source CODE



**5.3.2. Add library then complete the project :**

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

**\* File : LCD.c**

**\* Date: 23 - March - 2017**

**\* Project: STM32F407 SPI to LCD example**

**\* Author: Bui Viet Ha**

**\* Description:**

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

#include <stm32f4xx.h>

#include <stm32f4xx\_hal.h>

#include <stm32f4xx\_hal\_gpio.h>

#include <font.h>

#include <sys.h>

#include <LCD.h>

void LIGHT\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_13, GPIO\_PIN\_SET);

}

void LIGHT\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_13, GPIO\_PIN\_RESET);

}

void REST\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_15, GPIO\_PIN\_SET);

}

void REST\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOC, GPIO\_PIN\_15, GPIO\_PIN\_RESET);

}

void CS\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, GPIO\_PIN\_SET);

}

void CS\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_4, GPIO\_PIN\_RESET);

}

void A0\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, GPIO\_PIN\_SET);

}

void A0\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_6, GPIO\_PIN\_RESET);

}

void SDA\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_SET);

}

void SDA\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_7, GPIO\_PIN\_RESET);

}

void SCK\_PinSet(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_SET);

}

void SCK\_PinReset(void)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

u16 BACK\_COLOR, POINT\_COLOR;

void LCD\_Write\_byte(uint8\_t data, uint8\_t lcd\_mode)

{

// Use SPI to LCD

if (lcd\_mode == LCD\_MODE\_CMD)

A0\_PinReset();

else

A0\_PinSet();

HAL\_SPI\_Transmit(&hspi1, &data, 1, 100);

}

void LCD\_Write\_Data8(uint8\_t data)

{

LCD\_Write\_byte(data, LCD\_MODE\_DATA);

}

void LCD\_Write\_Data16(uint16\_t data)

{

uint8\_t send\_data[2];

send\_data[0]=data >> 8;

send\_data[1]=data & 0x00ff;

A0\_PinSet();

HAL\_SPI\_Transmit(&hspi1,send\_data,2,100);

}

void LCD\_Write\_Command(uint8\_t data)

{

LCD\_Write\_byte(data, LCD\_MODE\_CMD);

}

void Address\_set(unsigned int x1,unsigned int y1,unsigned int x2,unsigned int y2)

{

LCD\_Write\_Command(0x2a);

LCD\_Write\_Data8(x1>>8);

LCD\_Write\_Data8(x1);

LCD\_Write\_Data8(x2>>8);

LCD\_Write\_Data8(x2);

LCD\_Write\_Command(0x2b);

LCD\_Write\_Data8(y1>>8);

LCD\_Write\_Data8(y1);

LCD\_Write\_Data8(y2>>8);

LCD\_Write\_Data8(y2);

LCD\_Write\_Command(0x2C);

}

void LCD\_Init(void)

{

LIGHT\_PinSet();

REST\_PinReset();

delayms(1);

REST\_PinSet();

delayms(1);

SPI\_ENABLE;

LCD\_Write\_Command(0x11); //Sleep out

delayms(120); //Delay 120ms

//------------------------------------ST7735S Frame Rate-----------------------------------------//

LCD\_Write\_Command(0xB1); // in normal mode full colors

LCD\_Write\_Data8(0x05);

LCD\_Write\_Data8(0x3A);

LCD\_Write\_Command(0x11);//Sleep exit

delayms (120);

//ST7735R Frame Rate

LCD\_Write\_Command(0xB1);

LCD\_Write\_Data8(0x01);

LCD\_Write\_Data8(0x2C); LCD\_Write\_Data8(0x2D);

LCD\_Write\_Command(0xB2);

LCD\_Write\_Data8(0x01); LCD\_Write\_Data8(0x2C); LCD\_Write\_Data8(0x2D);

LCD\_Write\_Command(0xB3);

LCD\_Write\_Data8(0x01); LCD\_Write\_Data8(0x2C); LCD\_Write\_Data8(0x2D);

LCD\_Write\_Data8(0x01); LCD\_Write\_Data8(0x2C); LCD\_Write\_Data8(0x2D);

LCD\_Write\_Command(0xB4); //Column inversion

LCD\_Write\_Data8(0x07);

//ST7735R Power Sequence

LCD\_Write\_Command(0xC0);

LCD\_Write\_Data8(0xA2); LCD\_Write\_Data8(0x02); LCD\_Write\_Data8(0x84);

LCD\_Write\_Command(0xC1); LCD\_Write\_Data8(0xC5);

LCD\_Write\_Command(0xC2);

LCD\_Write\_Data8(0x0A); LCD\_Write\_Data8(0x00);

LCD\_Write\_Command(0xC3);

LCD\_Write\_Data8(0x8A); LCD\_Write\_Data8(0x2A);

LCD\_Write\_Command(0xC4);

LCD\_Write\_Data8(0x8A); LCD\_Write\_Data8(0xEE);

LCD\_Write\_Command(0xC5); //VCOM

LCD\_Write\_Data8(0x0E);

LCD\_Write\_Command(0x36); //MX, MY, RGB mode

LCD\_Write\_Data8(0xC8);

//ST7735R Gamma Sequence

LCD\_Write\_Command(0xe0);

LCD\_Write\_Data8(0x0f); LCD\_Write\_Data8(0x1a);

LCD\_Write\_Data8(0x0f); LCD\_Write\_Data8(0x18);

LCD\_Write\_Data8(0x2f); LCD\_Write\_Data8(0x28);

LCD\_Write\_Data8(0x20); LCD\_Write\_Data8(0x22);

LCD\_Write\_Data8(0x1f); LCD\_Write\_Data8(0x1b);

LCD\_Write\_Data8(0x23); LCD\_Write\_Data8(0x37); LCD\_Write\_Data8(0x00);

LCD\_Write\_Data8(0x07);

LCD\_Write\_Data8(0x02); LCD\_Write\_Data8(0x10);

LCD\_Write\_Command(0xe1);

LCD\_Write\_Data8(0x0f); LCD\_Write\_Data8(0x1b);

LCD\_Write\_Data8(0x0f); LCD\_Write\_Data8(0x17);

LCD\_Write\_Data8(0x33); LCD\_Write\_Data8(0x2c);

LCD\_Write\_Data8(0x29); LCD\_Write\_Data8(0x2e);

LCD\_Write\_Data8(0x30); LCD\_Write\_Data8(0x30);

LCD\_Write\_Data8(0x39); LCD\_Write\_Data8(0x3f);

LCD\_Write\_Data8(0x00); LCD\_Write\_Data8(0x07);

LCD\_Write\_Data8(0x03); LCD\_Write\_Data8(0x10);

LCD\_Write\_Command(0x2a);

LCD\_Write\_Data8(0x00);LCD\_Write\_Data8(0x00);

LCD\_Write\_Data8(0x00);LCD\_Write\_Data8(0x7f);

LCD\_Write\_Command(0x2b);

LCD\_Write\_Data8(0x00);LCD\_Write\_Data8(0x00);

LCD\_Write\_Data8(0x00);LCD\_Write\_Data8(0x9f);

LCD\_Write\_Command(0xF0); //Enable test command

LCD\_Write\_Data8(0x01);

LCD\_Write\_Command(0xF6); //Disable ram power save mode

LCD\_Write\_Data8(0x00);

LCD\_Write\_Command(0x3A); //65k mode

LCD\_Write\_Data8(0x05);

LCD\_Write\_Command(0x29); //Display on

}

void LCD\_Clear(u16 Color)

{

uint16\_t i,j;

Address\_set(0,0,LCD\_W-1,LCD\_H-1);

for(i=0;i<LCD\_W;i++)

{

for (j=0;j<LCD\_H;j++)

{

LCD\_Write\_Data16(Color);

}

}

}

void showhanzi(unsigned int x,unsigned int y,unsigned char index)

{

unsigned char i,j;

unsigned char \*temp=hanzi;

Address\_set(x,y,x+31,y+31);

temp+=index\*128;

for(j=0;j<128;j++)

{

for(i=0;i<8;i++)

{

if((\*temp&(1<<i))!=0)

{

LCD\_Write\_Data16(POINT\_COLOR);

}

else

{

LCD\_Write\_Data16(BACK\_COLOR);

}

}

temp++;

}

}

void LCD\_ShowChar(u16 x,u16 y,u8 num,u8 mode)

{

u8 temp;

u8 pos,t;

u16 x0=x;

u16 colortemp=POINT\_COLOR;

if(x>LCD\_W-16||y>LCD\_H-16)return;

num=num-' ';

Address\_set(x,y,x+8-1,y+16-1);

if(!mode)

{

for(pos=0;pos<16;pos++)

{

temp=asc2\_1608[(u16)num\*16+pos];

for(t=0;t<8;t++)

{

if(temp&0x01)POINT\_COLOR=colortemp;

else POINT\_COLOR=BACK\_COLOR;

LCD\_Write\_Data16(POINT\_COLOR);

temp>>=1;

x++;

}

x=x0;

y++;

}

}else

{

for(pos=0;pos<16;pos++)

{

temp=asc2\_1608[(u16)num\*16+pos];

for(t=0;t<8;t++)

{

if(temp&0x01)LCD\_DrawPoint(x+t,y+pos);

temp>>=1;

}

}

}

POINT\_COLOR=colortemp;

}

u32 mypow(u8 m,u8 n)

{

u32 result=1;

while(n--)result\*=m;

return result;

}

//num: (0~4294967295);

void LCD\_ShowNum(u16 x,u16 y,u32 num,u8 len)

{

u8 t,temp;

u8 enshow=0;

num=(u16)num;

for(t=0;t<len;t++)

{

temp=(num/mypow(10,len-t-1))%10;

if(enshow==0&&t<(len-1))

{

if(temp==0)

{

LCD\_ShowChar(x+8\*t,y,' ',0);

continue;

}else enshow=1;

}

LCD\_ShowChar(x+8\*t,y,temp+48,0);

}

}

//num:(0~99);;

void LCD\_Show2Num(u16 x,u16 y,u16 num,u8 len)

{

u8 t,temp;

for(t=0;t<len;t++)

{

temp=(num/mypow(10,len-t-1))%10;

LCD\_ShowChar(x+8\*t,y,temp+'0',0);

}

}

void LCD\_ShowString(u16 x,u16 y,const u8 \*p)

{

while(\*p!='\0')

{

if(x>LCD\_W-16){x=0;y+=16;}

if(y>LCD\_H-16){y=x=0;}

LCD\_ShowChar(x,y,\*p,0);

x+=8;

p++;

}

}

void puts\_image(const unsigned char\* image\_arr)

{

uint32\_t i;

uint16\_t temp;

uint8\_t high\_byte,low\_byte;

Address\_set(0,32,128-1,160-1);

for(i=0;i<(128\*128);i++)

{

low\_byte=\*(image\_arr+i\*2);

high\_byte=\*(image\_arr+i\*2+1);

temp=((high\_byte)<<8|(low\_byte));

LCD\_Write\_Data16(temp);

}

}

void showimage()

{

int i,j,k;

for(k=2;k<4;k++)

{

for(j=0;j<3;j++)

{

Address\_set(40\*j,40\*k,40\*j+39,40\*k+39);

for(i=0;i<1600;i++)

{

LCD\_Write\_Data8(image[i\*2+1]);

LCD\_Write\_Data8(image[i\*2]);

}

}

}

}