Homework 1

URL

DEBUG µVision User's Guide (arm.com)

FWLIB STM32标准外设软件库 - STMicroelectronics

芯片文档 STMCU中文官网

Week 1

The introduction to STM32

Microcontroller

> A **microcontroller** (**MCU** for *microcontroller unit*) is a small computer on a single VLSI integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. Program memory in the form of ferroelectric RAM, NOR flash, or OTP ROM is also often included on the chip, as well as a small amount of RAM. Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications consisting of various discrete chips.
>
> *From Wikipedia*

STM32

> **STM32** is a family of **32-bit** microcontroller integrated circuits by STMicroelectronics. The STM32 chips are grouped into related series that are based around the same 32-bit **ARM** processor core, such as the Cortex-M33F, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, or Cortex-M0. Internally, each microcontroller consists of the processor core, static RAM, flash memory, debugging interface, and various peripherals.
>
> *From Wikipedia*

The STM32 chip can be roughly divided into two parts: **kernel** + **on-chip peripherals**. The core is designed by ARM, and other manufacturers match the peripherals and produce chips based on the internal design. The kernel and on-chip peripherals are connected through **Bus matrix-S**. The on-off of each bit of each peripheral is controlled by the **register**. The essence of programming is to use registers to control the on and off of the circuit.

Library Functions

> The STM32F10x Standard Peripherals Library is a complete package, consisting of device drivers for all of the standard device peripherals, for STM32 Value line(High, Medium, and Low), Connectivity line, XL-, High-, Medium- and Low- Density Devices 32-bit Flash microcontrollers.
>
> This library is a firmware package that contains a collection of routines, data structures, and macros covering the features of STM32 peripherals. It includes a description of the device drivers plus a set of examples for each peripheral. The firmware library allows any device to be used in the user application without the need for an in-depth study of each peripheral's specifications.
>
> Using the Standard Peripherals Library has two advantages: it saves significant time that would otherwise be spent in coding, while simultaneously reducing application development and integration costs.
>
> *From STM32F10x Standard Peripherals Library Manual*

Development environment

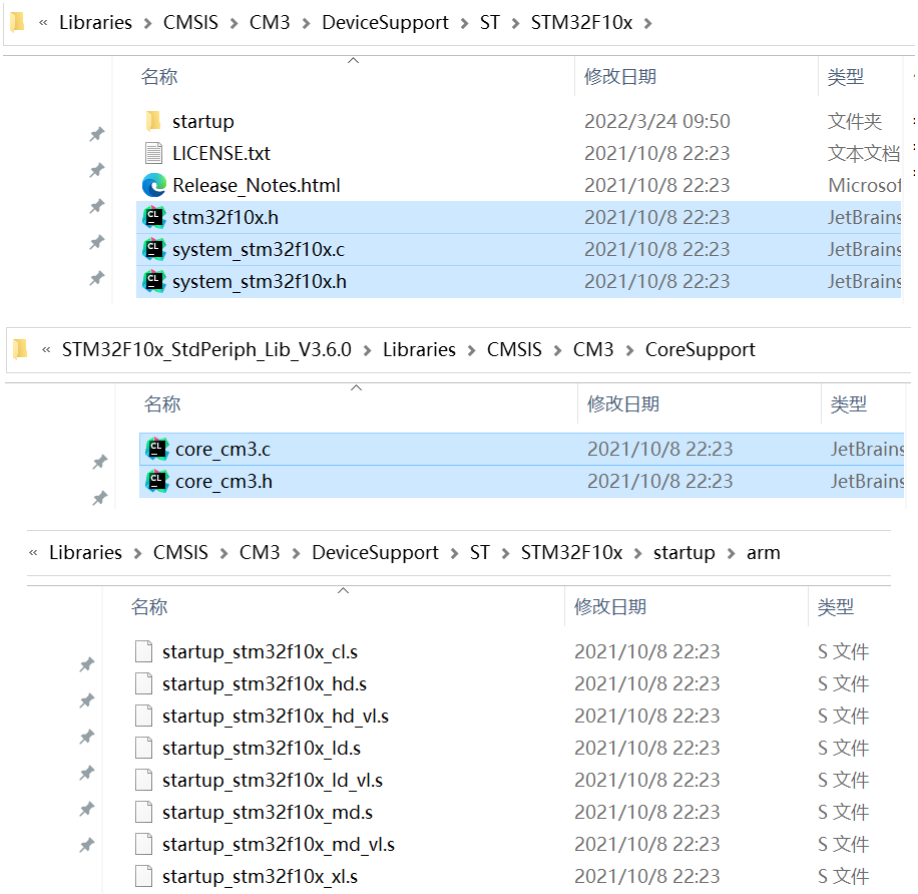
MDK536 - Microcontroller Development Kit

Keil uVision5 - IDE

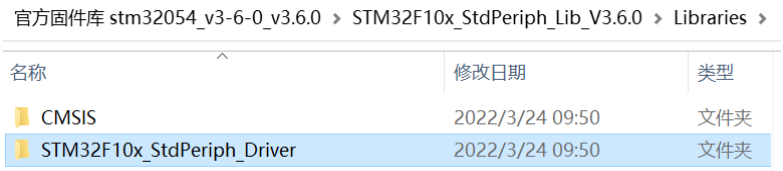
Create project

// Files

Start:



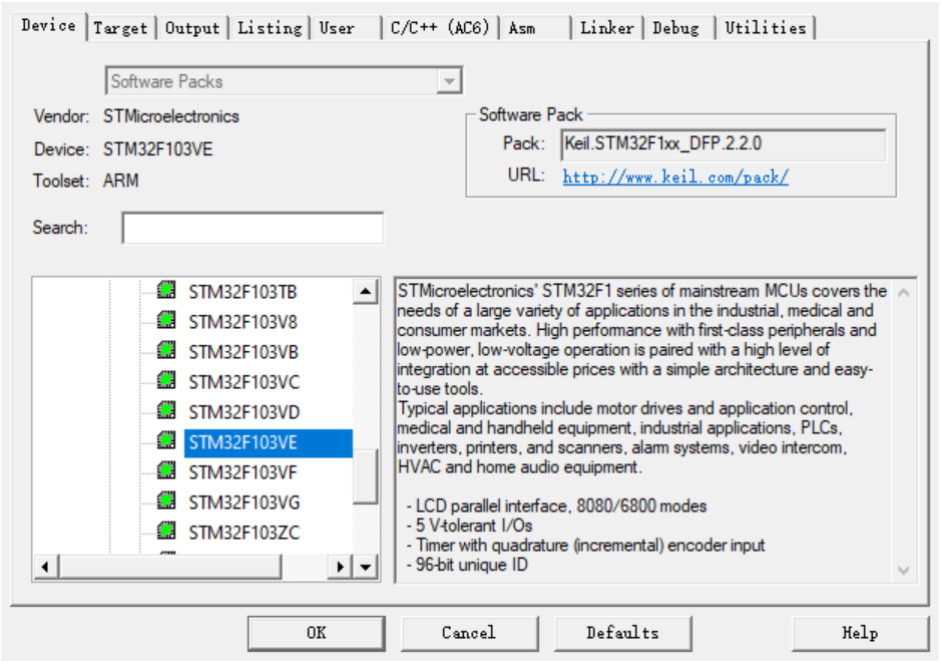
Libraries:



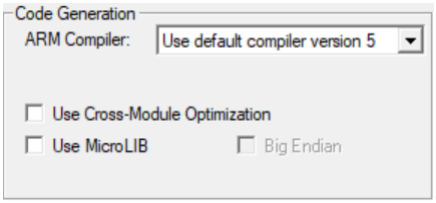
User:

// Settings

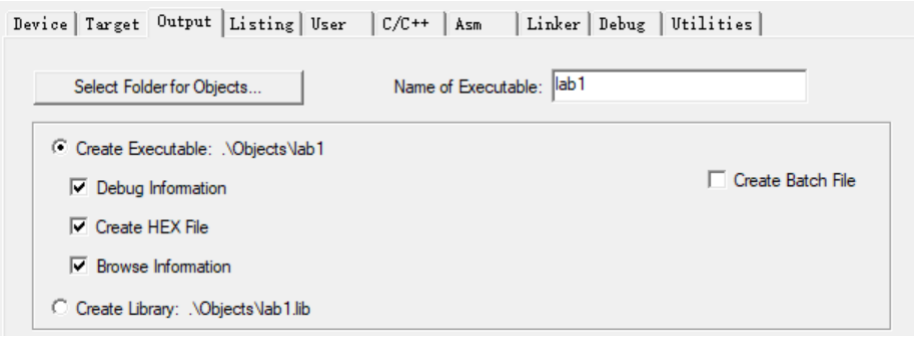
Choose type:



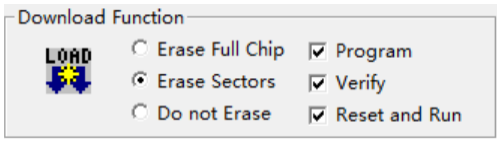
Compiler:



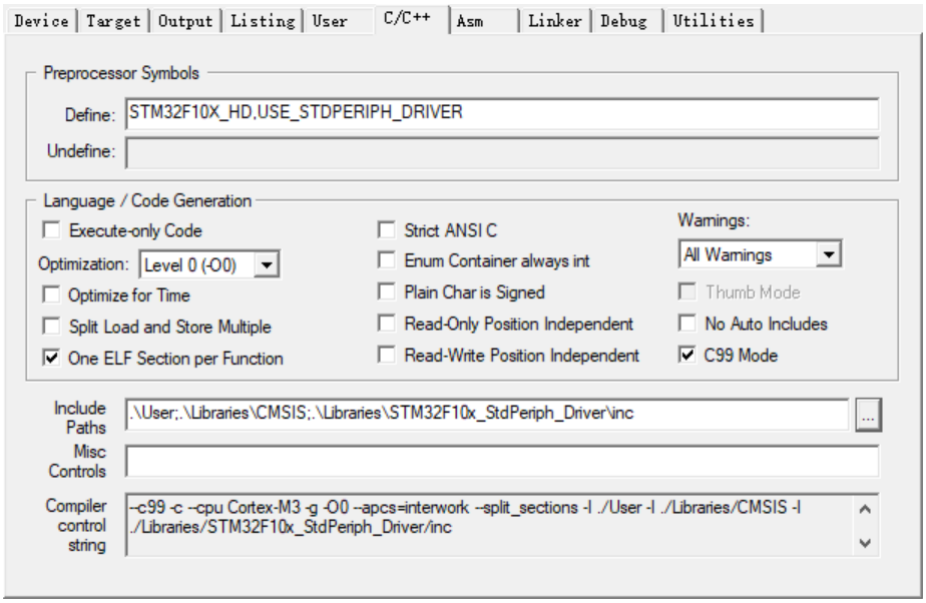
Output:



Debugger:



Language:



Programming

Register Programming:

1. Turn on the clock for the GPIO port

7.3.7 APB2 peripheral clock enable register (RCC_APB2ENR)

Address: 0x18

Reset value: 0x0000 0000

Access: word, half-word and byte access

No wait states, except if the access occurs while an access to a peripheral in the APB2 domain is on going. In this case, wait states are inserted until the access to APB2 peripheral is finished.

Note: When the peripheral clock is not active, the peripheral register values may not be readable by software and the returned value is always 0x0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved										TIM11 EN	TIM10 EN	TIM9 EN	Reserved		
										rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART 1EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	Res.	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

2. Configure the output mode and speed of the GPIO port

0x4001 2000 - 0x4001 23FF	GPIO Port G
0x4001 1C00 - 0x4001 1FFF	GPIO Port F
0x4001 1800 - 0x4001 1BFF	GPIO Port E
0x4001 1400 - 0x4001 17FF	GPIO Port D
0x4001 1000 - 0x4001 13FF	GPIO Port C
0x4001 0C00 - 0x4001 0FFF	GPIO Port B
0x4001 0800 - 0x4001 0BFF	GPIO Port A

Section 9.5 on page 194

9.2.1 Port configuration register low (GPIOx_CRL) (x=A..G)

Address offset: 0x00

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

9.2.2 Port configuration register high (GPIOx_CRH) (x=A..G)

Address offset: 0x04

Reset value: 0x4444 4444

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF15[1:0]		MODE15[1:0]		CNF14[1:0]		MODE14[1:0]		CNF13[1:0]		MODE13[1:0]		CNF12[1:0]		MODE12[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF11[1:0]		MODE11[1:0]		CNF10[1:0]		MODE10[1:0]		CNF9[1:0]		MODE9[1:0]		CNF8[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

表17 端口位配置表

配置模式		CNF1	CNF0	MODE1 MODE0	PxODR寄存器
通用输出	推挽(Push-Pull)	0	0	01	0 或 1
	开漏(Open-Drain)		1	10	0 或 1
复用功能输出	推挽(Push-Pull)	1	0	11	不使用
	开漏(Open-Drain)		1	见表18	不使用
输入	模拟输入	0	0	00	不使用
	浮空输入		1		不使用
	下拉输入	1	0		0
	上拉输入				1

表18 输出模式位

MODE[1:0]	意义
00	保留
01	最大输出速度为10MHz
10	最大输出速度为2MHz
11	最大输出速度为50MHz

3. Control ODR Register

9.2.4 Port output data register (GPIOx_ODR) (x=A..G)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy:** Port output data (y= 0 .. 15)

These bits can be read and written by software and can be accessed in Word mode only.

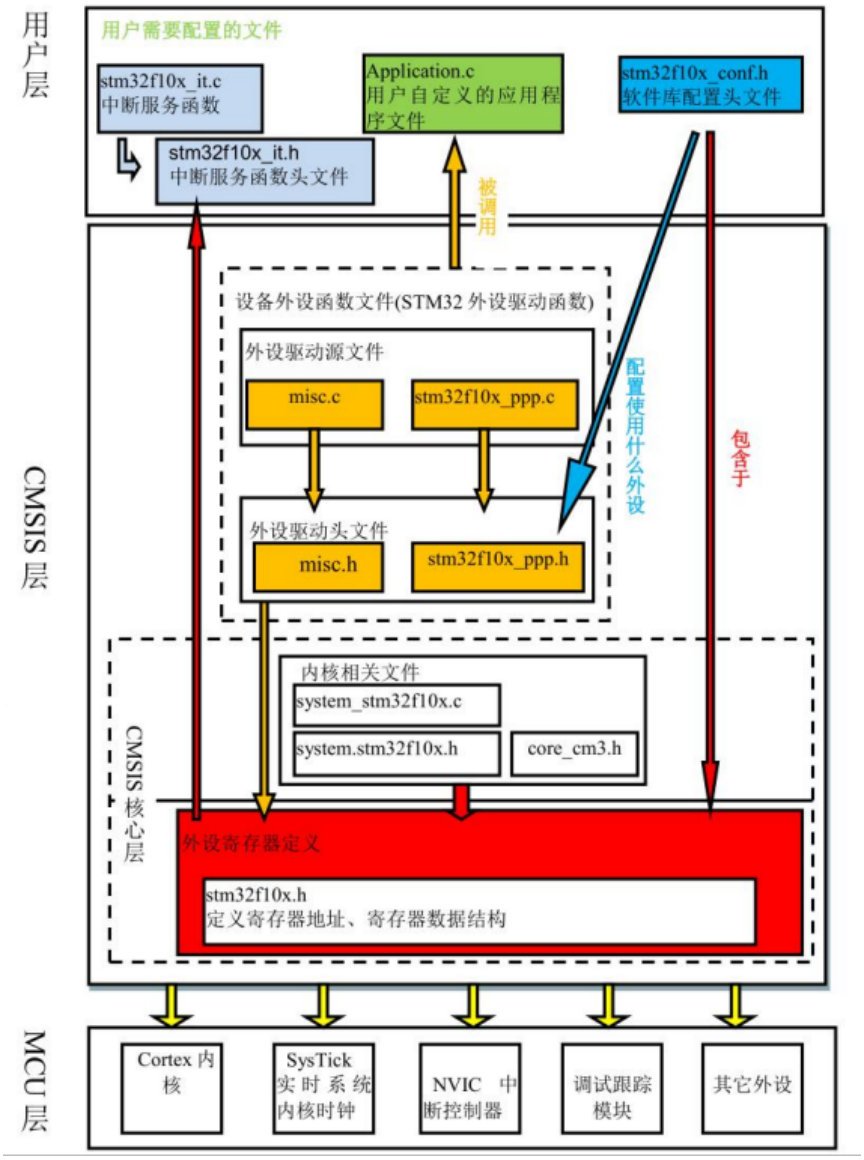
Note: For atomic bit set/reset, the ODR bits can be individually set and cleared by writing to the GPIOx_BSRR register (x = A .. G).

// ODR | BSRR | BRR

Firmware library programming:

// as follows

Analysis



STM32F10x固件库文件分析		
启动文件	startup_stm32f10x_hd.s	汇编编写的启动文件
外设相关	stm32f10x.h	外设寄存器定义
	system_stm32f10x.h	1、用于系统初始化
	system_stm32f10x.c	2、配置系统时钟
	stm32f10x_xxx.h	外设固件库头文件,如GPIO，ADC，I2C，SPI等
	stm32f10x_xxx.c	外设的固件库文件,如GPIO，ADC，I2C，SPI等
内核相关	core_cm3.h	内核寄存器定义
	core_cm3.c	控制内核外设的相关函数，用的非常少
	misc.h	跟NVIC和SysTick相关的函数
	misc.c	
用户相关	main.c	main函数存在的地方
	stm32f10x_it.h	用户编写的中断服务函数都放在这里
	stm32f10x_it.c	

Modify LED to PC13

```
// 打开GPIOC端口的时钟
*( unsigned int * )0x40021018 |= ( 1<<4 );

// 配置IO口PC13为推挽输出，速率为10M
*( unsigned int * )0x40011004 &= 0xFF0FFFFF;
*( unsigned int * )0x40011004 |= 0X00100000;

// 控制ODR寄存器
*( unsigned int * )0x4001100C &= ~( 1<<13 ); //LED on
//*( unsigned int * )0x4001100C |= ( 1<<13 ); //LED off
```

Week 2

Configure PC13 as a pull-up input pin by 2 methods.

1. Firmware library programming.

```
RCC->APB2ENR |= ( (1)<<4 );
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin= GPIO_Pin_13;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz; //似乎无影响？
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_ResetBits(GPIOC, GPIO_Pin_13);
```

2. Register programming.

```
*( unsigned int * )0x40021018 |= ( (1)<<4 );
*( unsigned int * )0x40011004 &= 0xFF0FFFFF;
*( unsigned int * )0x40011004 |= 0x00800000;
*( unsigned int * )0x4001100C &= ~( 1<<13 );
```

Week 3

Write out the assignment process of each step in the function `GPIO_Init(GPIOB, &GPIO_InitStructure)`

```
GPIO_InitTypeDef GPIO_InitStructure;
GPIO_InitStructure.GPIO_Pin= GPIO_Pin_2;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD ;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

1. Incoming Parameters

   Incoming `GPIO_TypeDef* GPIOx`, `GPIO_InitTypeDef* GPIO_InitStruct`

```
void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)
{
  uint32_t currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;
  uint32_t tmpreg = 0x00, pinmask = 0x00;
  /* Check the parameters */
  assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
  assert_param(IS_GPIO_MODE(GPIO_InitStruct->GPIO_Mode));
  assert_param(IS_GPIO_PIN(GPIO_InitStruct->GPIO_Pin));
```

   * `GPIOx`

```
#define GPIOA               ((GPIO_TypeDef *) GPIOA_BASE)
#define GPIOB               ((GPIO_TypeDef *) GPIOB_BASE)
#define GPIOC               ((GPIO_TypeDef *)
#define GPIOD               ((GPIO_TypeDef *)
#define GPIOE               ((GPIO_TypeDef *)
#define GPIOF               ((GPIO_TypeDef *)
#define GPIOG               ((GPIO_TypeDef *)
```

声明位置: stm32f10x.h

定义:

```
#define GPIOB_BASE            (APB2PERIPH_BASE + 0x0C00)
```

   * `GPIO_InitStruct`

```
typedef struct
{
  uint16_t GPIO_Pin;             /*!< Specifies the GPIO pins to be configured.
                                      This parameter can be any value of @ref GPIO_pins_define */

  GPIOSpeed_TypeDef GPIO_Speed;  /*!< Specifies the speed for the selected pins.
                                      This parameter can be a value of @ref GPIOSpeed_TypeDef */

  GPIOMode_TypeDef GPIO_Mode;    /*!< Specifies the operating mode for the selected pins.
                                      This parameter can be a value of @ref GPIOMode_TypeDef */
}GPIO_InitTypeDef;
```

`GPIO_Pin` is `GPIO_Pin_2`

`GPIO_Mode` is `GPIO_Mode_IPD`

`GPIO_Speed` is `GPIO_Speed_10MHz`

*Check the parameters

2. GPIO Mode Configuration

```
/*---------------------------- GPIO Mode Configuration -----------------------*/
  currentmode = ((uint32_t)GPIO_InitStruct->GPIO_Mode) & ((uint32_t)0x0F);
  if ((((uint32_t)GPIO_InitStruct->GPIO_Mode) & ((uint32_t)0x10)) != 0x00)
  {
    /* Check the parameters */
    assert_param(IS_GPIO_SPEED(GPIO_InitStruct->GPIO_Speed));
    /* Output mode */
    currentmode |= (uint32_t)GPIO_InitStruct->GPIO_Speed;
  }
```

Judging whether it is input or output. If it is output, set the transmission speed.

Now it has set the mode and speed of the GPIO

3. GPIO CRL/CHL Configuration

Copy register bits;

Get the port pins position;

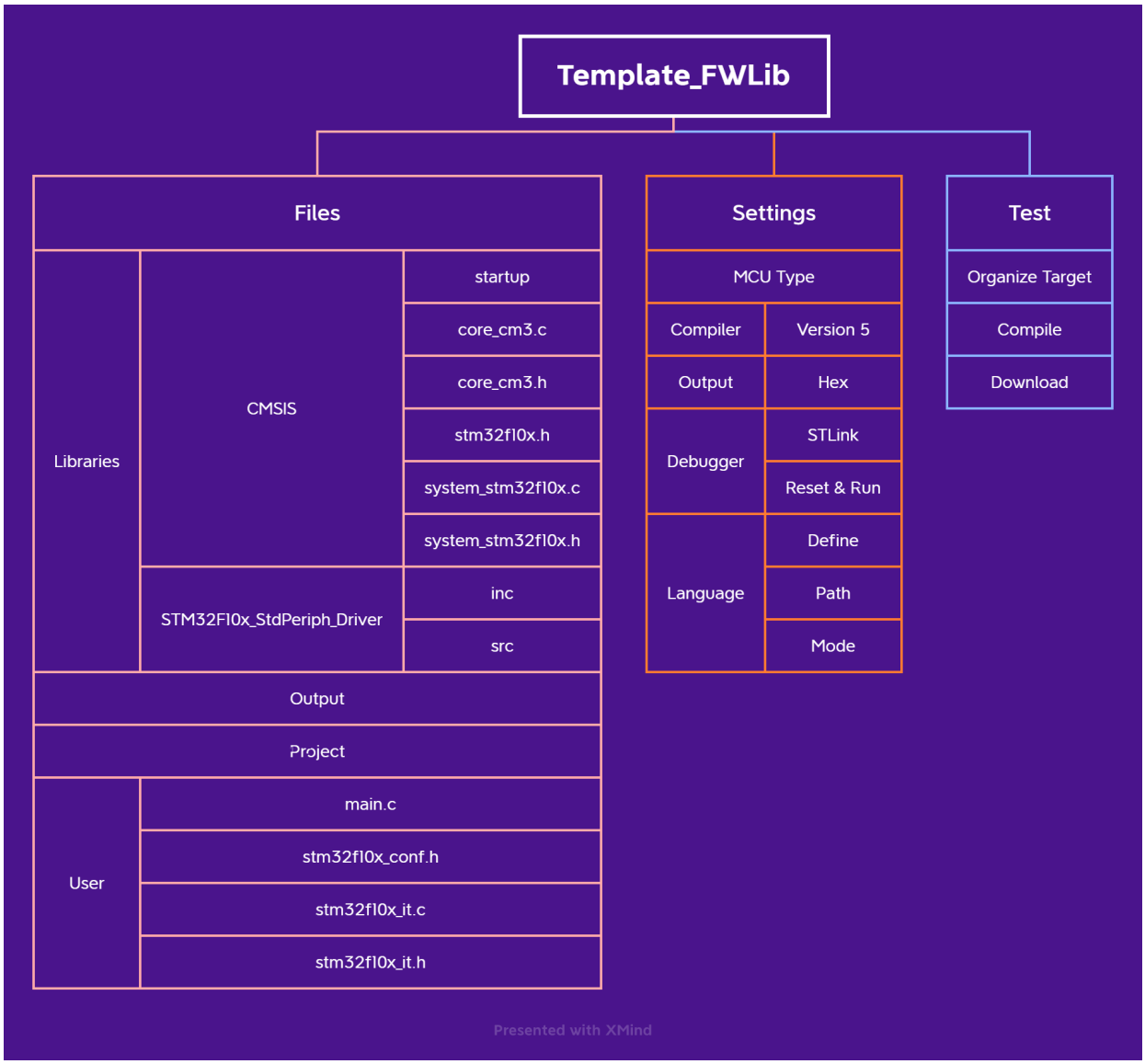
Clear the corresponding control register bits;

Write the mode configuration in the corresponding bits (above configuration `currentmode`);

Reset and set the corresponding ODR bit;

Write register bits.

Sort out the file structure of the official Firmware Library on page 27.


Template_FWLib
Files
Libraries
CMSIS
startup
core_cm3.c
core_cm3.h
stm32f10x.h
system_stm32f10x.c
system_stm32f10x.h
STM32F10x_StdPeriph_Driver
inc
src
Output
Project
User
main.c
stm32f10x_conf.h
stm32f10x_it.c
stm32f10x_it.h
Settings
MCU Type
Compiler
Version 5
Output
Hex
Debugger
STLink
Reset & Run
Language
Define
Path
Mode
Test
Organize Target
Compile
Download
Presented with XMind