

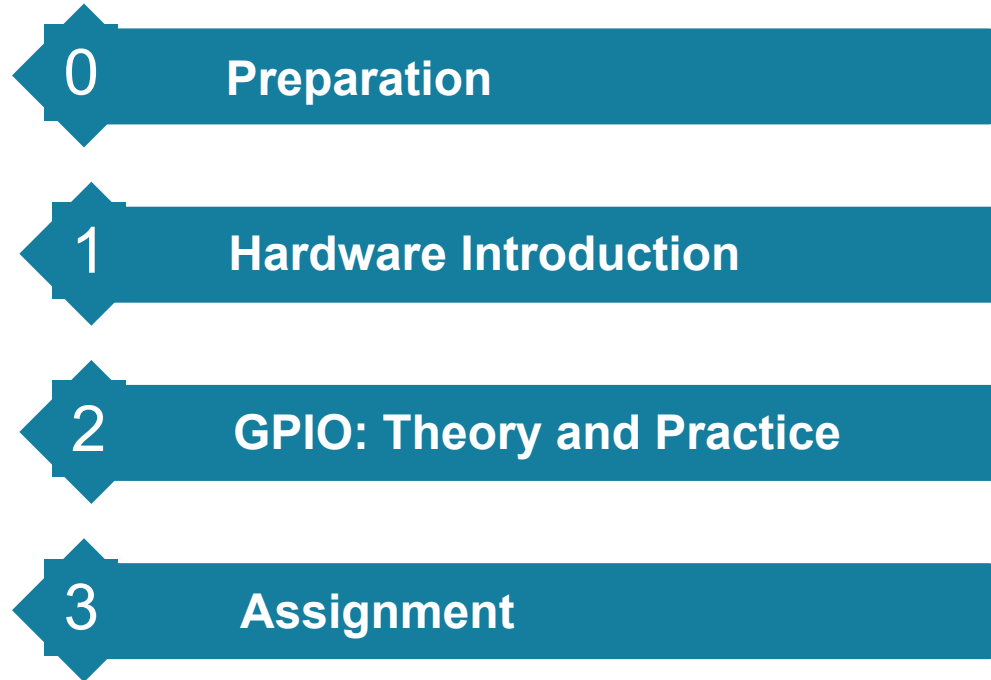


---

# 0 - STM32 Introduction

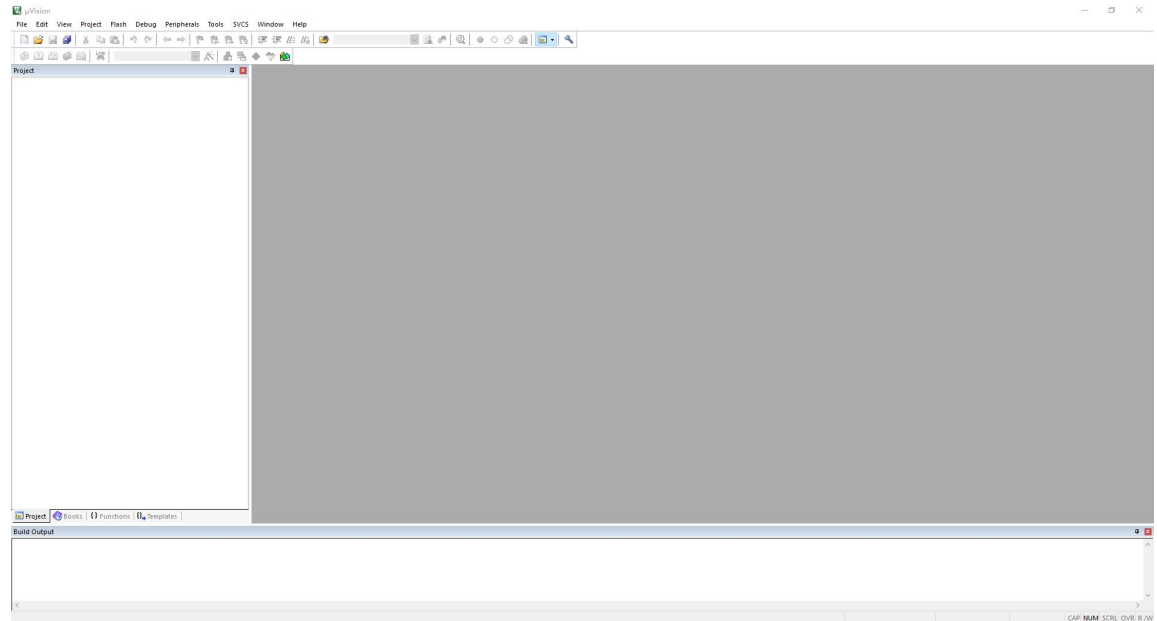
---

System Design and Intelligent Manufacture  
2019 Spring

- 
- 0** Preparation
  - 1** Hardware Introduction
  - 2** GPIO: Theory and Practice
  - 3** Assignment

# What is “MDK”?









RealView MDK is developed by Keil. It can provide an environment for processor based on Cortex, ARM7, ARM9.



# Tips

1. Installation path should be in English.
2. System user name should be in English.
3. Don't install multiple MDK(Keil) in the same path.
4. Don't forget to load package of chip.

# Resources

 Cpp_refcard.pdf	2019/3/8 18:49	WPS PDF 文档	25 KB
 Explorer STM32F4_V2.2_SCH.pdf	2015/7/8 12:35	WPS PDF 文档	803 KB
 RealPlayer_16.0.6.7.exe	2019/3/5 10:00	应用程序	41,027 KB
 ST MCU 最新选型手册_201603.pdf	2016/9/22 10:53	WPS PDF 文档	12,537 KB
 STM32F4xx英文参考手册.pdf	2015/11/16 17:43	WPS PDF 文档	23,634 KB
 STM32F4xx中文参考手册.pdf	2014/7/18 8:49	WPS PDF 文档	21,092 KB
 STM32F4开发指南-库函数版本_V1.1.pdf	2016/10/15 16:10	WPS PDF 文档	51,866 KB
 STM32F407ZGT6数据手册.pdf	2014/4/10 12:54	WPS PDF 文档	2,188 KB

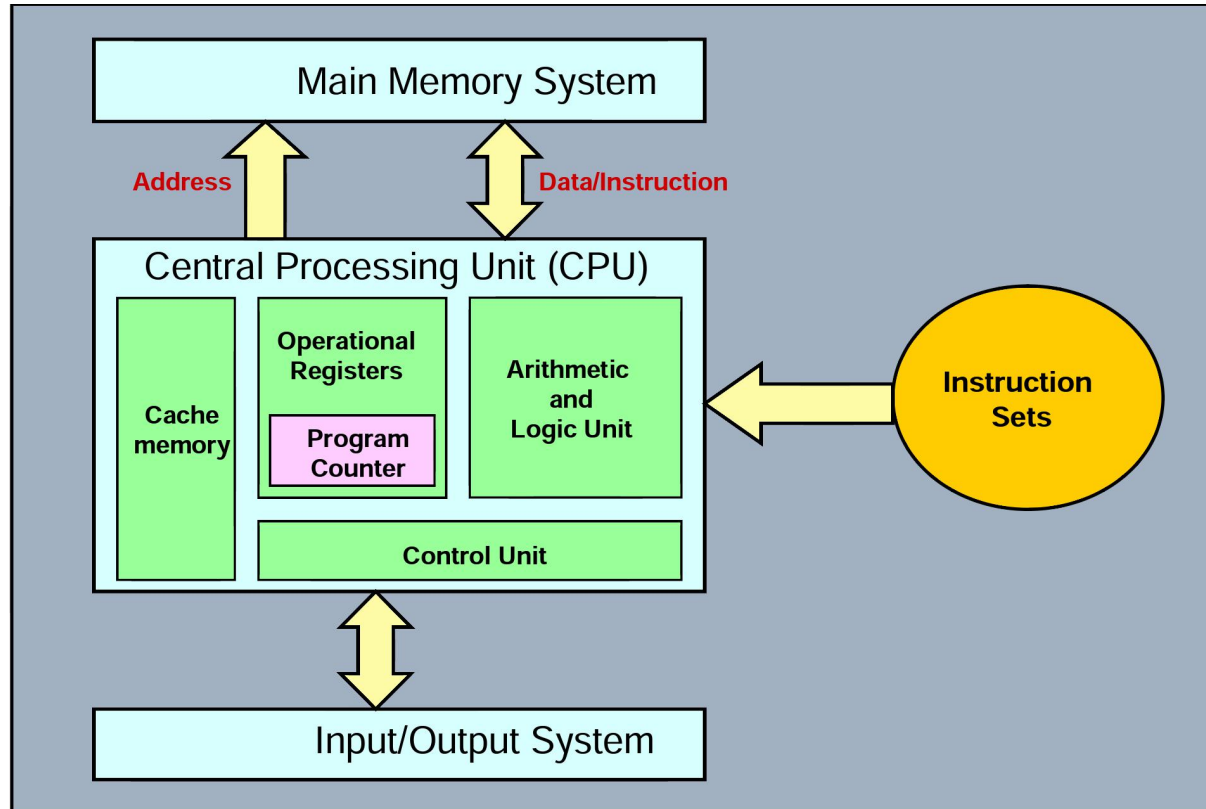
<https://www.stmcu.com.cn/>

<http://www.stmcu.org.cn/module/forum/forum.php>

<http://bbs.21ic.com/icfilter-typeid-35-226.html>

<http://firebbs.cn/forum.php>

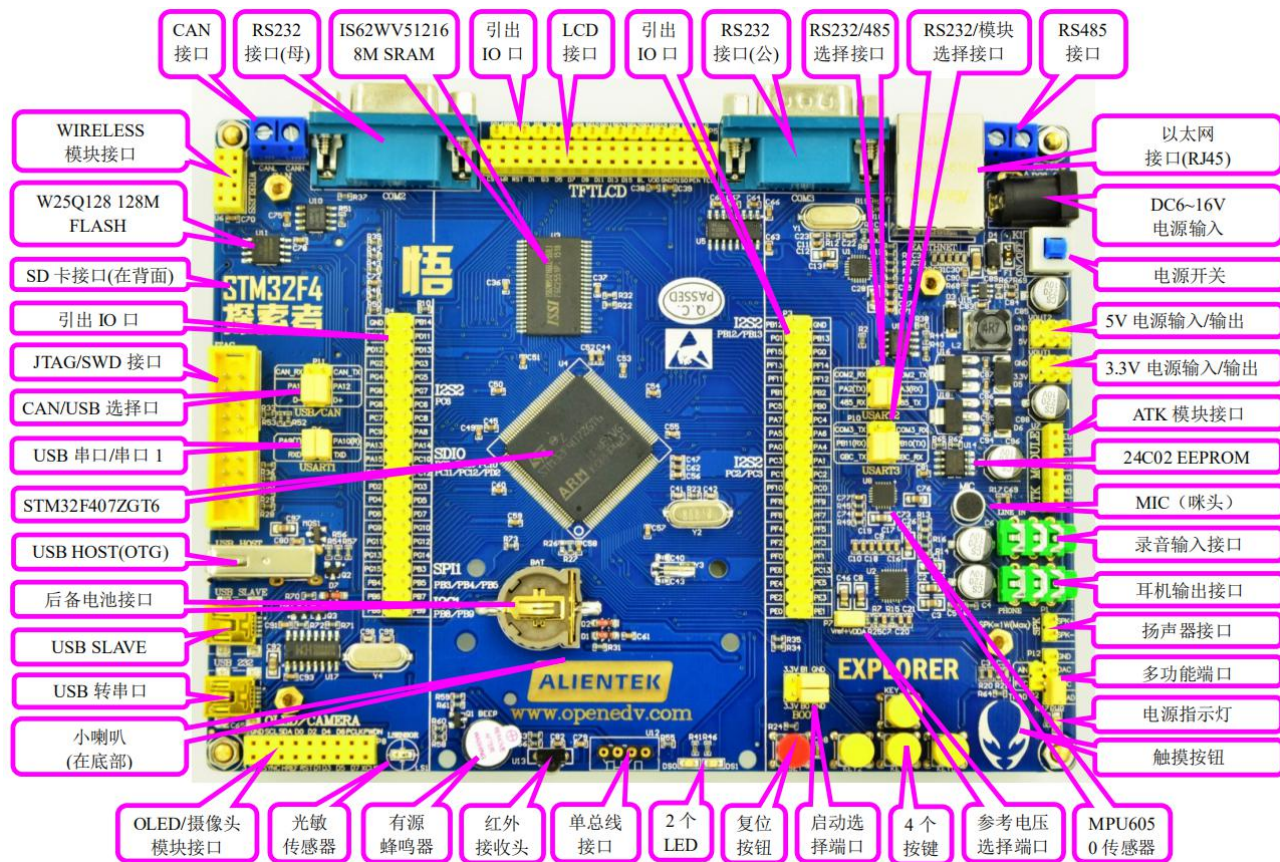
# Hardware Introduction



# Knowledge

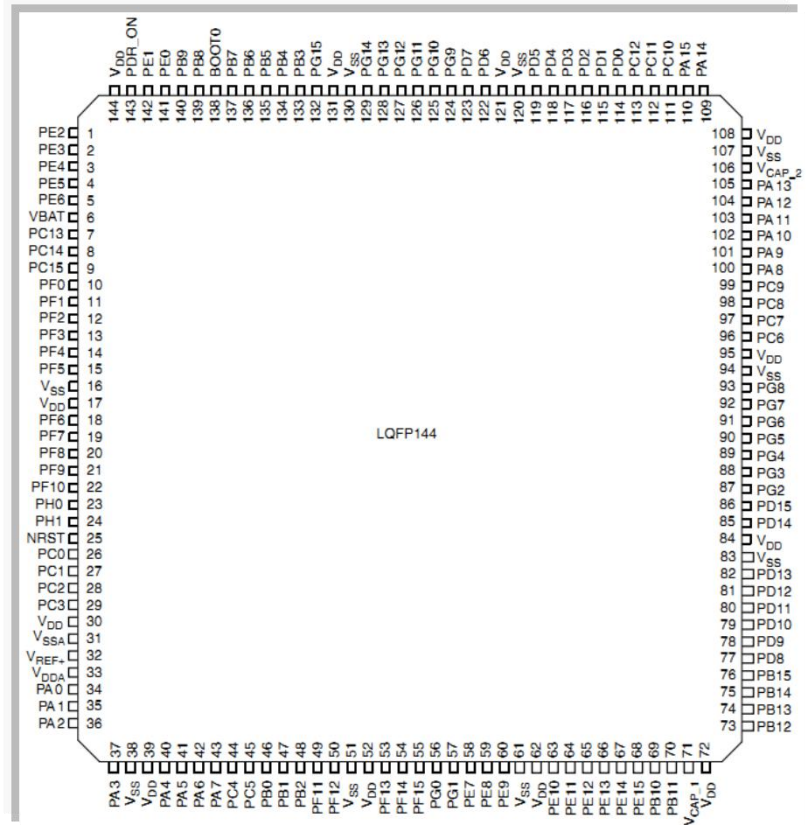
	Knowledge
CSE	Computer Architecture, Embedded Programming, C++ Programming, Operating System, Fundamentals of Compiling
EEE	Fundamentals of Circuits, Artificial Circuits, Digital Circuits
ME	/

# Explorer

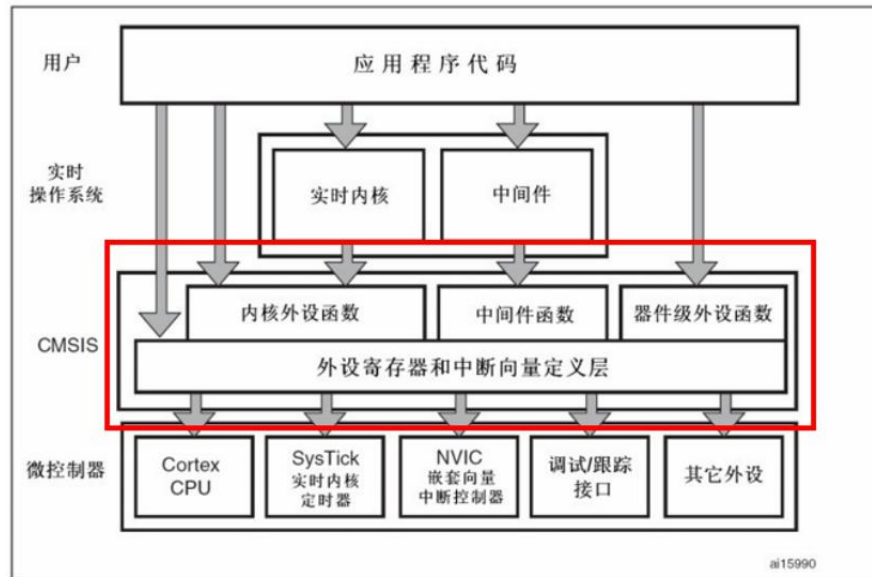
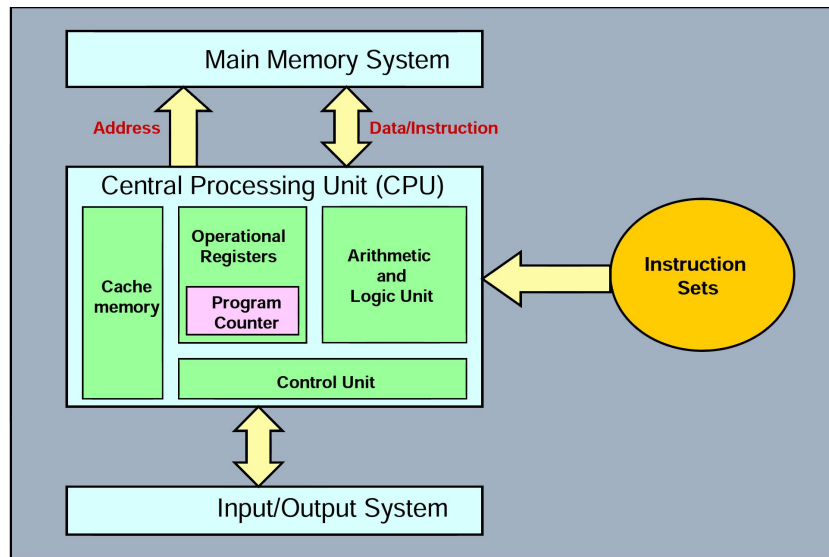




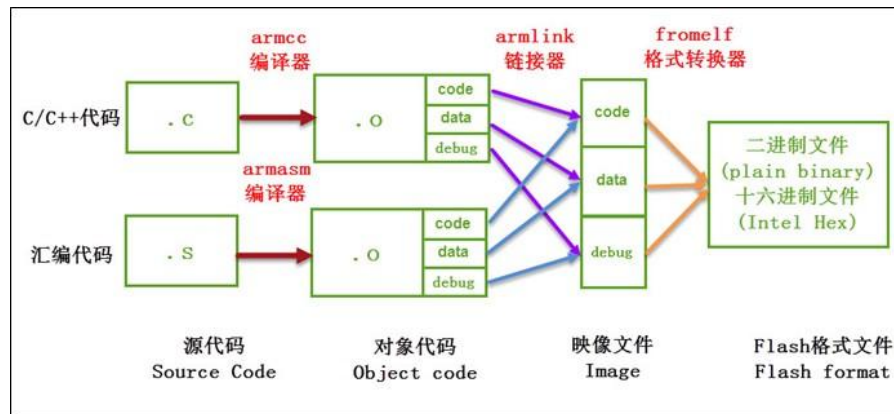
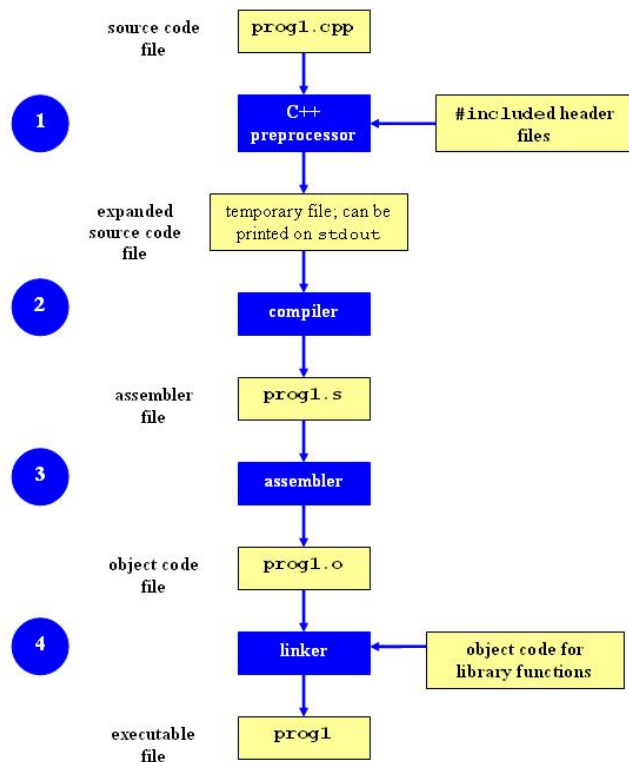
# Chip



[illegible]



# Programming



Details:

<https://www.cnblogs.com/firege/p/5806134.html>

# Simple C++

## C++ QUICK REFERENCE

### PREPROCESSOR

```
// Comment to end of line
/* Multi-line comment */
// Insert standard header file
// Insert file in current directory
// Replace X with some text
// Replace F(1,2) with 1+2
#define F(a,b) a+b
#define X \
    some text
// Line continuation
#undef X // Remove definition
#if defined(X) // Conditional compilation (#ifdef X)
#else // optional (#ifndef X or #if !defined(X))
#endif // Required after #if, #ifdef
```

### LITERALS

```
255, 0377, 0xff // Integers (decimal, octal, hex)
2147483647L, 0x7fffffff // Long (32-bit) integers
123.0, 1.23e2 // double (real) numbers
'a', '\a', '\x61' // Character (literal, octal, hex)
'\n', '\\', '\'', '\"' // Newline, backslash, single quote, double quote
"string\n" // Array of characters ending with newline and \0
"hello" "world" // Concatenated strings
true, false // bool constants 1 and 0
```

### DECLARATIONS

```
int x; // Declare x to be an integer (value undefined)
int x=255; // Declare and initialize x to 255
short s; long l; // Usually 16 or 32 bit integer (int may be either)
char c='a'; // Usually 8 bit character
unsigned char u=255; signed char s=-1; // char might be either
unsigned long x=0xffffffff; // short, int, long are signed
float f; double d; // Single or double precision real (never unsigned)
bool b=true; // true or false, may also use int (1 or 0)
int a, b, c; // Multiple declarations
int a[10]; // Array of 10 ints (a[0] through a[9])
int a[]={0,1,2}; // Initialized array (or a[3]={0,1,2}; )
int a[2][3]={1,2,3},{4,5,6}; // Array of array of ints
char s[6]="hello"; // String (6 elements including '\0')
int* p; // p is a pointer to (address of) int
char* s="hello"; // s points to unnamed array containing "hello"
void* p=NULL; // Address of untyped memory (NULL is 0)
int& r=x; // r is a reference to (alias of) int x
enum weekend {SAT,SUN}; // weekend is a type with values SAT and SUN
enum weekend day; // day is a variable of type weekend
enum weekend {SAT=0,SUN=1}; // Explicit representation as int
enum {SAT,SUN} day; // Anonymous enum
typedef String char*; // String s; means char* s;
```

```
const int c=3; // Constants must be initialized, cannot assign to
const int* p=a; // Contents of p (elements of a) are constant
int* const p=a; // p (but not contents) are constant
const int* const p=a; // Both p and its contents are constant
const int& cr=x; // cr cannot be assigned to change x
```

### STORAGE CLASSES

```
int x; // Auto (memory exists only while in scope)
static int x; // Global lifetime even if local scope
extern int x; // Information only, declared elsewhere
```

### STATEMENTS

```
x=y; // Every expression is a statement
int x; // Declarations are statements
; // Empty statement

{ // A block is a single statement
    int x; // Scope of x is from declaration to end of block
    a; // In C, declarations must precede statements
}
if (x) a; // If x is true (not 0), evaluate a
else if (y) b; // If not x and y (optional, may be repeated)
else c; // If not x and not y (optional)

while (x) a; // Repeat 0 or more times while x is true

for (x; y; z) a; // Equivalent to: x; while(y) {a; z;}

do a; while (x); // Equivalent to: a; while(x) a;

switch (x) { // x must be int
    case X1: a; // If x == X1 (must be a const), jump here
    case X2: b; // Else if x == X2, jump here
    default: c; // Else jump here (optional)
}
break; // Jump out of while, do, or for loop, or switch
continue; // Jump to bottom of while, do, or for loop
return x; // Return x from function to caller
try { a; } // If a throws a T, then jump here
catch (T t) { b; } // If a throws something else, jump here
catch (...) { c; }
```

### FUNCTIONS

```
int f(int x, int); // f is a function taking 2 ints and returning int
void f(); // f is a procedure taking no arguments
void f(int a=0); // f() is equivalent to f(0)
f(); // Default return type is int
inline f(); // Optimize for speed
f() { statements; } // Function definition (must be global)
T operator+(T x, T y); // +b (if type T) calls operator+(a, b)
T operator-(T x); // -a calls function operator-(a)
T operator++(int); // postfix ++ or -- (parameter ignored)
extern "C" { void f(); } // f() was compiled in C
```



# Programming in STM32

10 mins

## 输入：

模拟输入-获得外部的模拟信号，输入不经过输入数据寄存器

浮空输入-输入完全由外部决定，此时IO的电平状态未知

上拉输入-外部无输入时，为高电平

下拉输入-外部无输入时，为低电平

## 输出：

开漏输出-输出0时为GND，输出1时，由外接上拉电阻决定

推挽输出-输出0时为GND，输出1时为VCC

复用开漏输出-片内外设功能（TX1, MOSI, MISO, SCK, SS）

复用推挽输出-片内外设功能（I2C的SCL, SDA）



# Code

```
void LED_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStructure;
    RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOF, ENABLE); //使能 GPIOF 时钟

    //GPIOF9,F10 初始化设置
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9 | GPIO_Pin_10; //LED0 和 LED1 对应 IO 口
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT; //普通输出模式
    GPIO_InitStructure.GPIO_OType = GPIO_OType_PP; //推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz; //100MHz
    GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_UP; //上拉

    GPIO_Init(GPIOF, &GPIO_InitStructure); //初始化 GPIO

    GPIO_SetBits(GPIOF,GPIO_Pin_9 | GPIO_Pin_10); //GPIOF9,F10 设置高, 灯灭
}
```