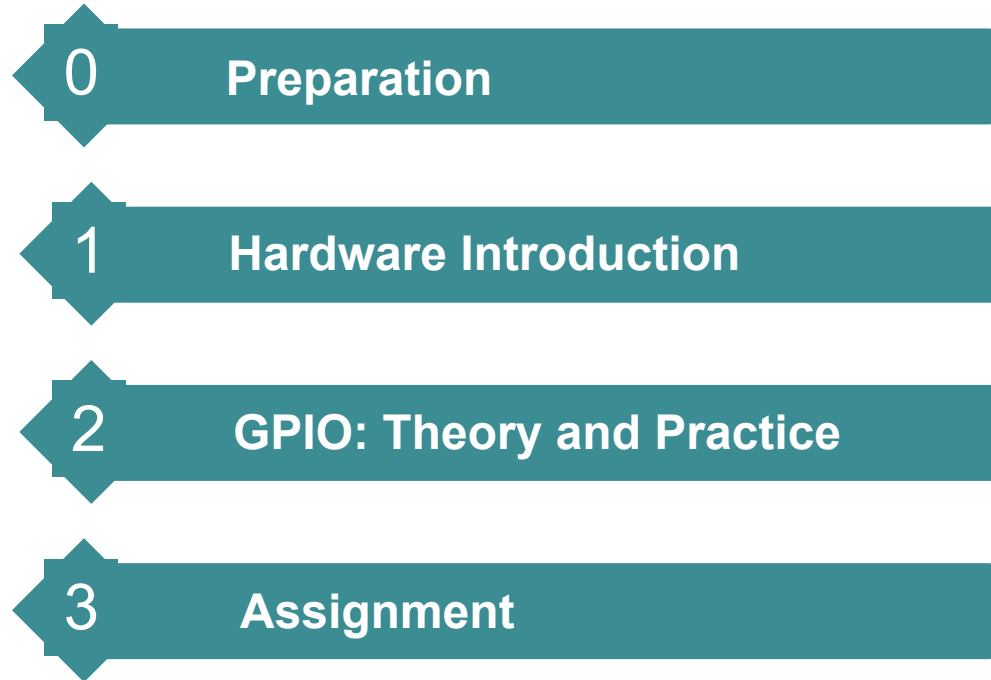




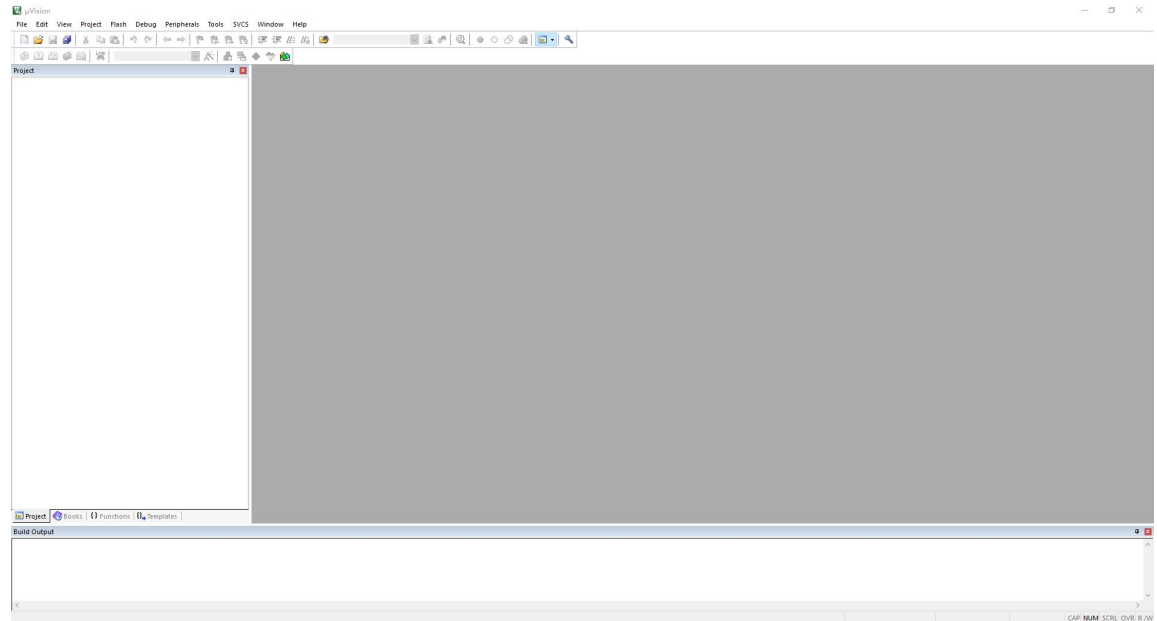
0 - STM32 Introduction

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 developmented 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.

Handbooks and 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

BBS and Pages

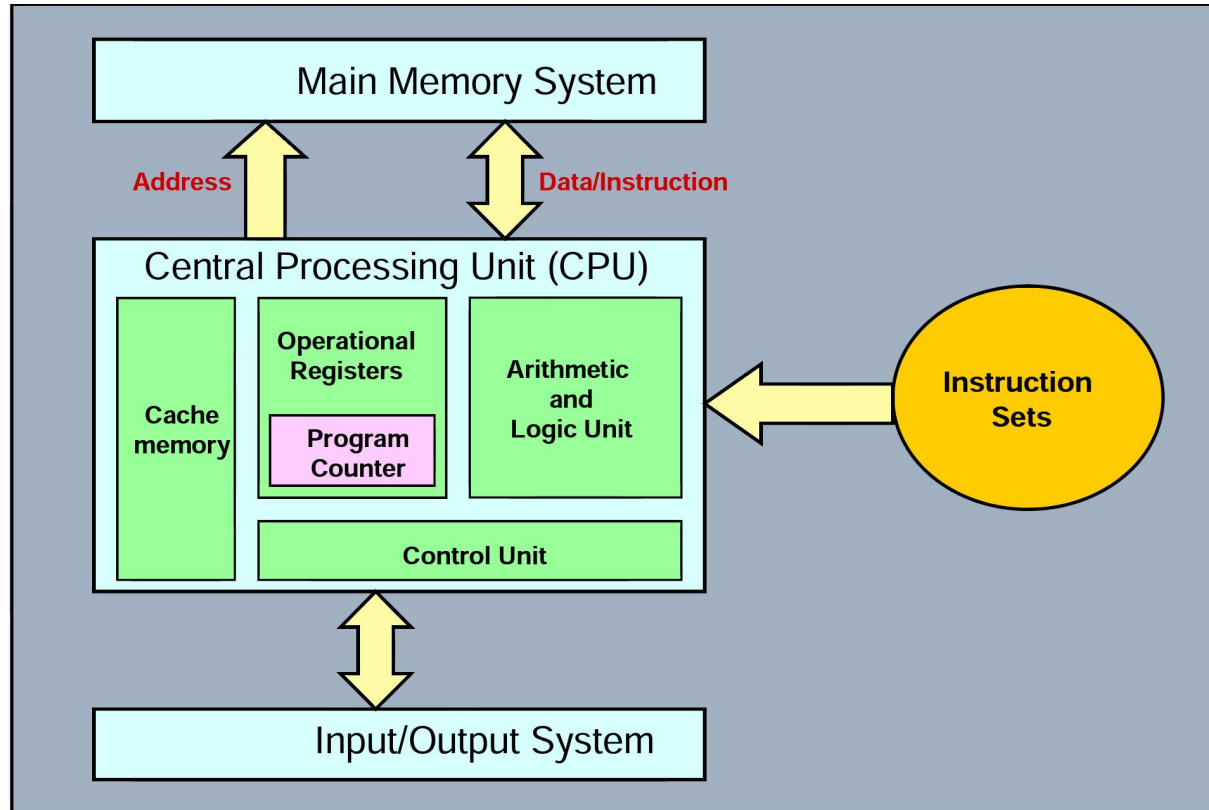
<https://www.stmcu.com.cn/>

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

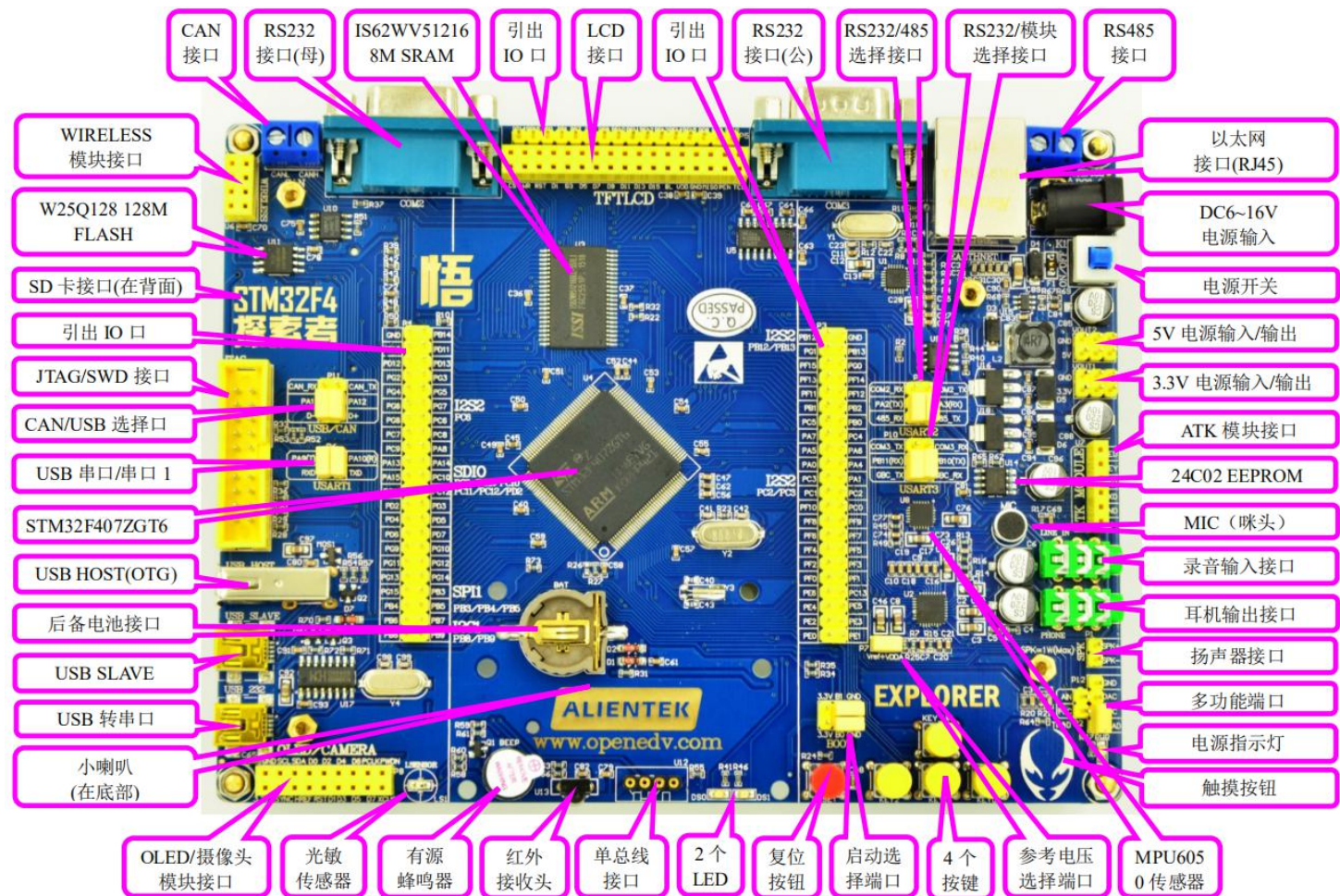
<http://bbs.21ic.com/icfilter-typeid-35-226.html>

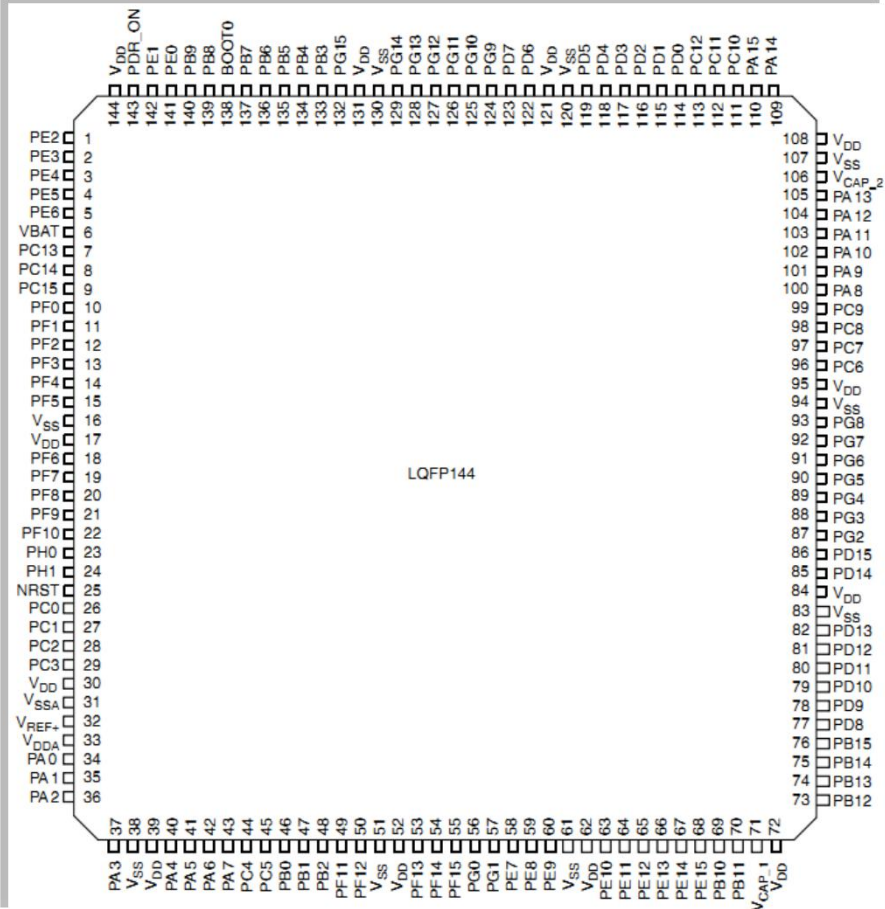
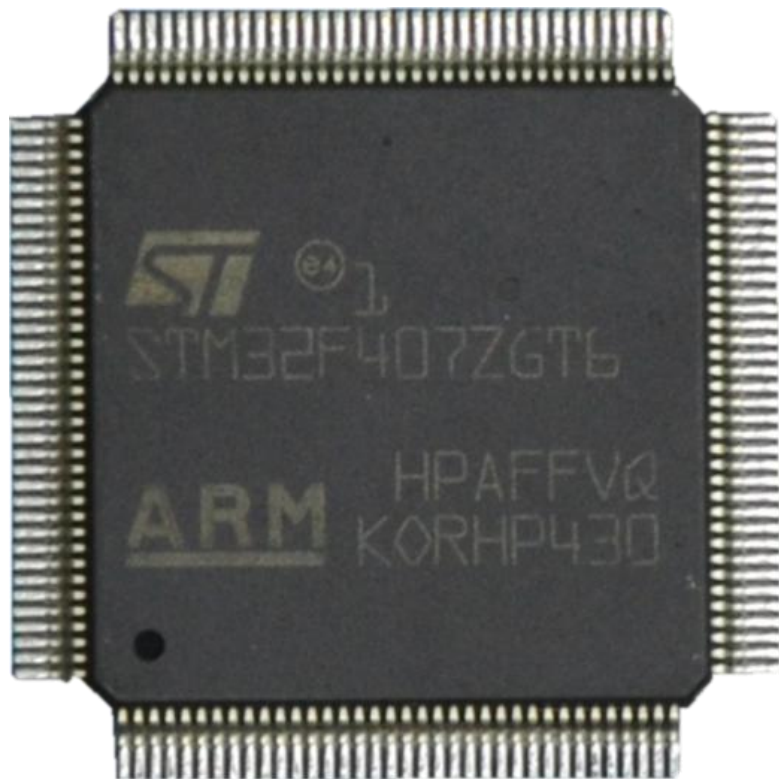
<http://firebbs.cn/forum.php>

Hardware Introduction

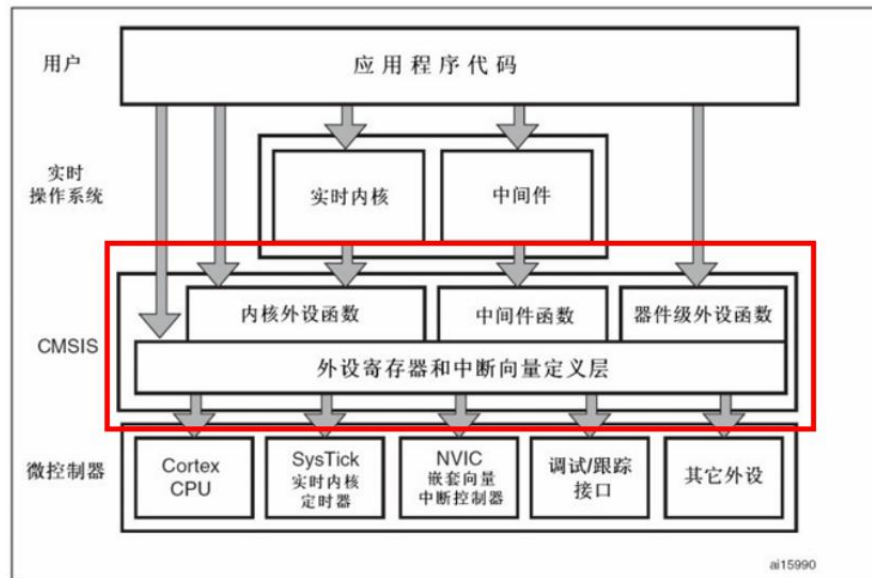
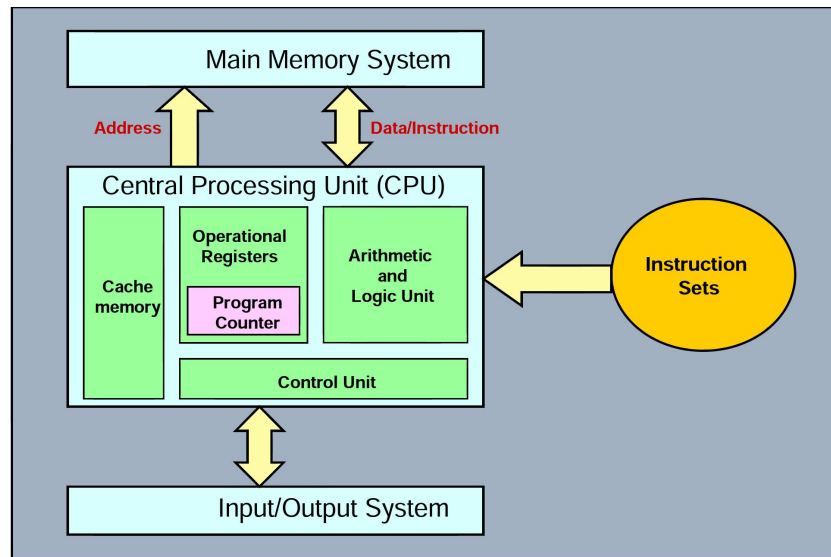


	Knowledge
CSE	Computer Architecture, Embedded Programming, C++ Programming, Operating System, Fundamentals of Compiling
EEE	Fundamentals of Circuits, Artificial Circuits, Digital Circuits
ME	/



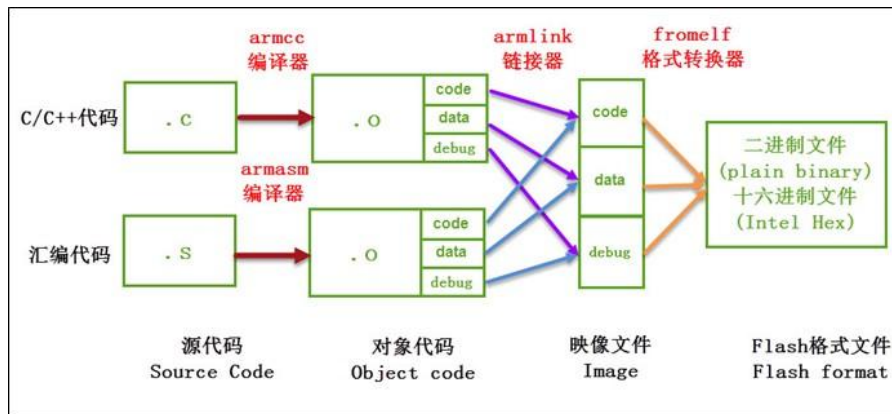
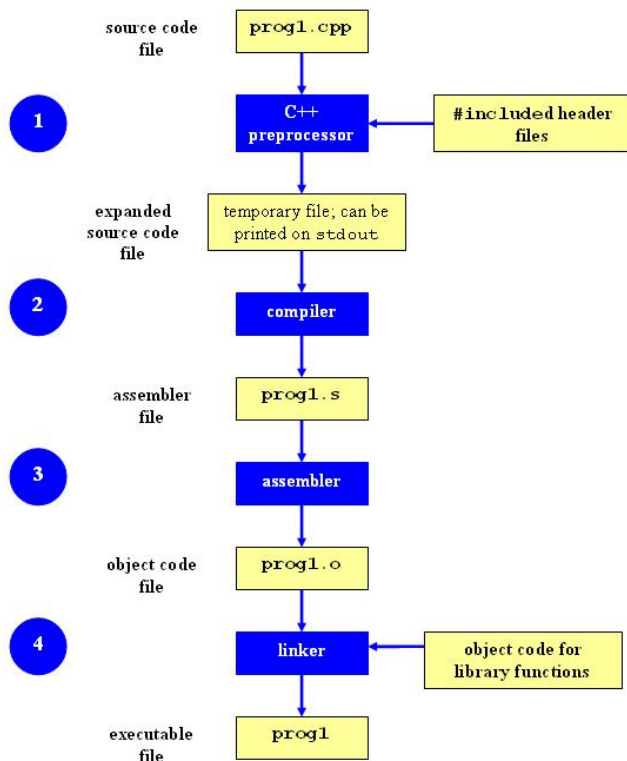


PE1	P0	P00	P4	P04
PE3	1	P02	3	P06
PE5	5	P04	5	P08
PE7	7	P06	7	P10
PE9	9	P08	9	P12
PE11	11	P10	11	P14
PE13	13	P12	13	P16
PE15	15	P14	15	P18
PE17	17	P16	17	P20
PE19	19	P18	19	P22
PE21	21	P20	21	P24
PE23	23	P22	23	P26
PE25	25	P24	25	P28
PE27	27	P26	27	P30
PE29	29	P28	29	P32
PE31	31	P30	31	P34
PE33	33	P32	33	P36
PE35	35	P34	35	P38
PE37	37	P36	37	P40
PE39	39	P38	39	P42
PE41	41	P40	41	P44
PE43	43	P42	43	P46
PE45	45	P44	45	P48
PE47	47	P46	47	P50
PE49	49	P48	49	P52
PE51	51	P50	51	P54
PE53	53	P52	53	P56
PE55	55	P54	55	P58
PE57	57	P56	57	P60
PE59	59	P58	59	P62
PE61	61	P60	61	P64
PE63	63	P62	63	P66
PE65	65	P64	65	P68
PE67	67	P66	67	P70
PE69	69	P68	69	P72
PE71	71	P70	71	P74
PE73	73	P72	73	P76
PE75	75	P74	75	P78
PE77	77	P76	77	P80
PE79	79	P78	79	P82
PE81	81	P80	81	P84
PE83	83	P82	83	P86
PE85	85	P84	85	P88
PE87	87	P86	87	P90
PE89	89	P88	89	P92
PE91	91	P90	91	P94
PE93	93	P92	93	P96
PE95	95	P94	95	P98
PE97	97	P96	97	P100
PE99	99	P98	99	P102
PE101	101	P100	101	P104
PE103	103	P102	103	P106
PE105	105	P104	105	P108
PE107	107	P106	107	P110
PE109	109	P108	109	P112
PE111	111	P110	111	P114
PE113	113	P112	113	P116
PE115	115	P114	115	P118
PE117	117	P116	117	P120
PE119	119	P118	119	P122
PE121	121	P120	121	P124
PE123	123	P122	123	P126
PE125	125	P124	125	P128
PE127	127	P126	127	P130
PE129	129	P128	129	P132
PE131	131	P130	131	P134
PE133	133	P132	133	P136
PE135	135	P134	135	P138
PE137	137	P136	137	P140
PE139	139	P138	139	P142
PE141	141	P140	141	P144
PE143	143	P142	143	P146
PE145	145	P144	145	P148
PE147	147	P146	147	P150
PE149	149	P148	149	P152
PE151	151	P150	151	P154
PE153	153	P152	153	P156
PE155	155	P154	155	P158
PE157	157	P156	157	P160
PE159	159	P158	159	P162
PE161	161	P160	161	P164
PE163	163	P162	163	P166
PE165	165	P164	165	P168
PE167	167	P166	167	P170
PE169	169	P168	169	P172
PE171	171	P170	171	P174
PE173	173	P172	173	P176
PE175	175	P174	175	P178
PE177	177	P176	177	P180
PE179	179	P178	179	P182
PE181	181	P180	181	P184
PE183	183	P182	183	P186
PE185	185	P184	185	P188
PE187	187	P186	187	P190
PE189	189	P188	189	P192
PE191	191	P190	191	P194
PE193	193	P192	193	P196
PE195	195	P194	195	P198
PE197	197	P196	197	P200
PE199	199	P198	199	P202
PE201	201	P200	201	P204
PE203	203	P202	203	P206
PE205	205	P204	205	P208
PE207	207	P206	207	P210
PE209	209	P208	209	P212
PE211	211	P210	211	P214
PE213	213	P212	213	P216
PE215	215	P214	215	P218
PE217	217	P216	217	P220
PE219	219	P218	219	P222
PE221	221	P220	221	P224
PE223	223	P222	223	P226
PE225	225	P224	225	P228
PE227	227	P226	227	P230
PE229	229	P228	229	P232
PE231	231	P230	231	P234
PE233	233	P232	233	P236
PE235	235	P234	235	P238
PE237	237	P236	237	P240
PE239	239	P238	239	P242
PE241	241	P240	241	P244
PE243	243	P242	243	P246
PE245	245	P244	245	P248
PE247	247	P246	247	P250
PE249	249	P248	249	P252
PE251	251	P250	251	P254
PE253	253	P252	253	P256
PE255	255	P254	255	P258
PE257	257	P256	257	P260
PE259	259	P258	259	P262
PE261	261	P260	261	P264
PE263	263	P262	263	P266
PE265	265	P264	265	P268
PE267	267	P266	267	P270
PE269	269	P268	269	P272
PE271	271	P270	271	P274
PE273	273	P272	273	P276
PE275	275	P274	275	P278
PE277	277	P276	277	P280
PE279	279	P278	279	P282
PE281	281	P280	281	P284
PE283	283	P282	283	P286
PE285	285	P284	285	P288
PE287	287	P286	287	P290
PE289	289	P288	289	P292
PE291	291	P290	291	P294
PE293	293	P292	293	P296
PE295	295	P294	295	P298
PE297	297	P296	297	P300
PE299	299	P298	299	P302
PE301	301	P300	301	P304
PE303	303	P302	303	P306
PE305	305	P304	305	P308
PE307	307	P306	307	P310
PE309	309	P308	309	P312
PE311	311	P310	311	P314
PE313	313	P312	313	P316
PE315	315	P314	315	P318
PE317	317	P316	317	P320
PE319	319	P318	319	P322
PE321	321	P320	321	P324
PE323	323	P322	323	P326
PE325	325	P324	325	P328
PE327	327	P326	327	P330
PE329	329	P328	329	P332
PE331	331	P330	331	P334
PE333	333	P332	333	P336
PE335	335	P334	335	P338
PE337	337	P336	337	P340
PE339	339	P338	339	P342
PE341	341	P340	341	P344
PE343	343	P342	343	P346
PE345	345	P344	345	P348
PE347	347	P346	347	P350
PE349	349	P348	349	P352
PE351	351	P350	351	P354
PE353	353	P352	353	P356
PE355	355	P354	355	P358
PE357	357	P356	357	P360
PE359	359	P358	359	P362
PE361	361	P360	361	P364
PE363	363	P362	363	P366
PE365	365	P364	365	P368
PE367	367	P366	367	P370
PE369	369	P368	369	P372
PE371	371	P370	371	P374
PE373	373	P372	373	P376
PE375	375	P374	375	P378
PE377	377	P376	377	P380
PE379	379	P378	379	P382
PE381	381	P380	381	P384
PE383	383	P382	383	P386
PE385	385	P384	385	P388
PE387	387	P386	387	P390
PE389	389	P388	389	P392
PE391	391	P390	391	P394
PE393	393	P392	393	P396
PE395	395	P394	395	P398
PE397	397	P396	397	P400
PE399	399	P398	399	P402
PE401	401	P400	401	P404
PE403	403	P402	403	P406
PE405	405	P404	405	P408
PE407	407	P406	407	P410
PE409	409	P408	409	P412
PE411	411	P410	411	P414
PE413	413	P412	413	P416
PE415	415	P414	415	P418
PE417	417	P416	417	P420
PE419	419	P418	419	P422
PE421	421	P420	421	P424
PE423	423	P422	423	P426
PE425	425	P424	425	P428
PE427	427	P426	427	P430
PE429	429	P428	429	P432
PE431	431	P430	431	P434
PE433	433	P432	433	P436
PE435	435	P434	435	P438
PE437	437	P436	437	P440
PE439	439	P438	439	P442
PE441	441	P440	441	P444
PE443	443	P442	443	P446
PE445	445	P444	445	P448
PE447	447	P446	447	P450
PE449	449	P448	449	P452
PE451	451	P450	451	P454
PE453	453	P452	453	P456
PE455	455	P454	455	P458
PE457	457	P456	457	P460
PE459	459	P458	459	P462
PE461	461	P460	461	P464
PE463	463	P462	463	P466
PE465	465	P464	465	P468
PE467	467	P466	467	P470
PE469	469	P468	469	P472
PE471	471	P470	471	P474
PE473	473	P472	473	P476
PE475	475	P474	475	P478
PE477	477	P476	477	P480
PE479	479	P478	479	P482
PE481	481	P480	481	P484
PE483	483	P482	483	P486
PE485	485	P484	485	P488
PE487	487	P486	487	P490
PE489	489	P488	489	P492
PE491	491	P490	491	P494
PE493	493	P492	493	P496
PE495	495	P494	495	P498
PE497	497	P496	497	P500
PE499	499	P498	499	P502
PE501	501	P500	501	P504
PE503	503	P502	503	P506
PE505	505	P504	505	P508
PE507	507	P506	507	P510
PE509	509	P508	509	P512
PE511	511	P510	511	P514
PE513	513	P512	513	P516
PE515	515	P514	515	P518
PE517	517	P516	517	P520
PE519	519	P518	519	P522
PE521	521	P520	521	P524
PE523	523	P522	523	P526
PE525	525	P524	525	P528
PE527	527	P526	527	P530
PE529	529	P528	529	P532
PE531	531	P530	531	P534
PE533	533	P532	533	P536
PE535	535	P534	535	P53



all15990

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 X \
    some text
// Line continuation
#undef X // Remove definition
#ifdef(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', '\141', '\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[]="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

GPIO

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）

```

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
}

```