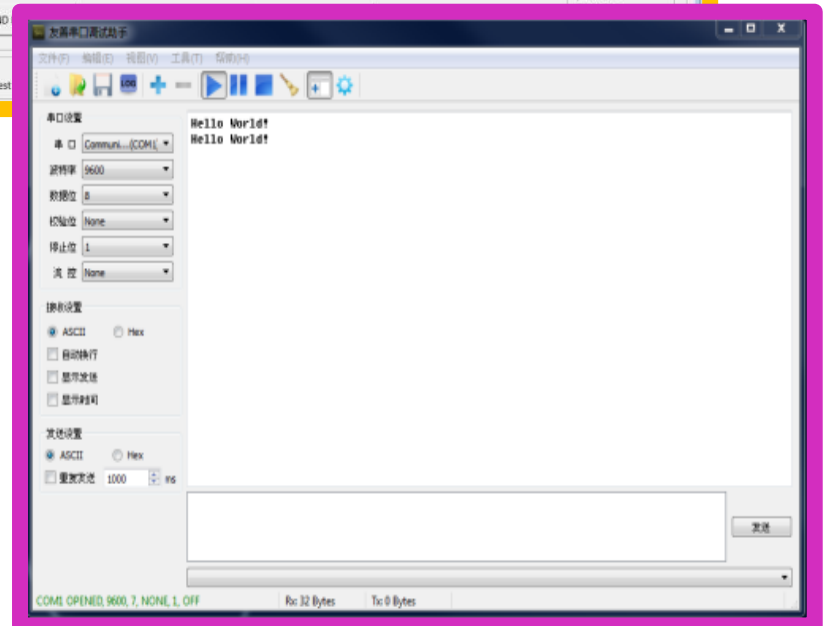
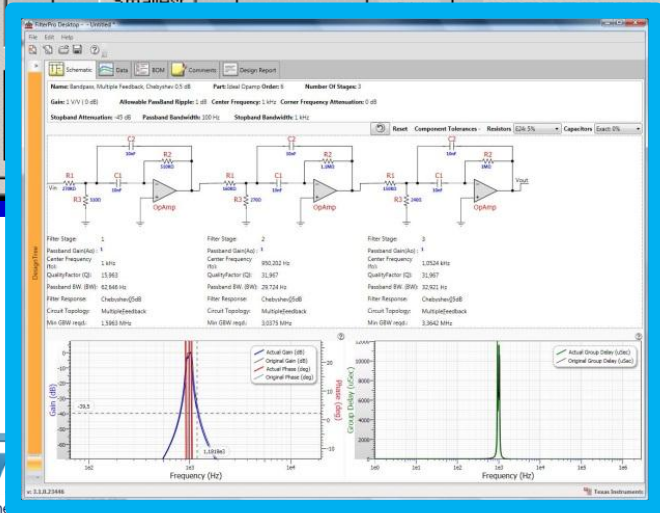
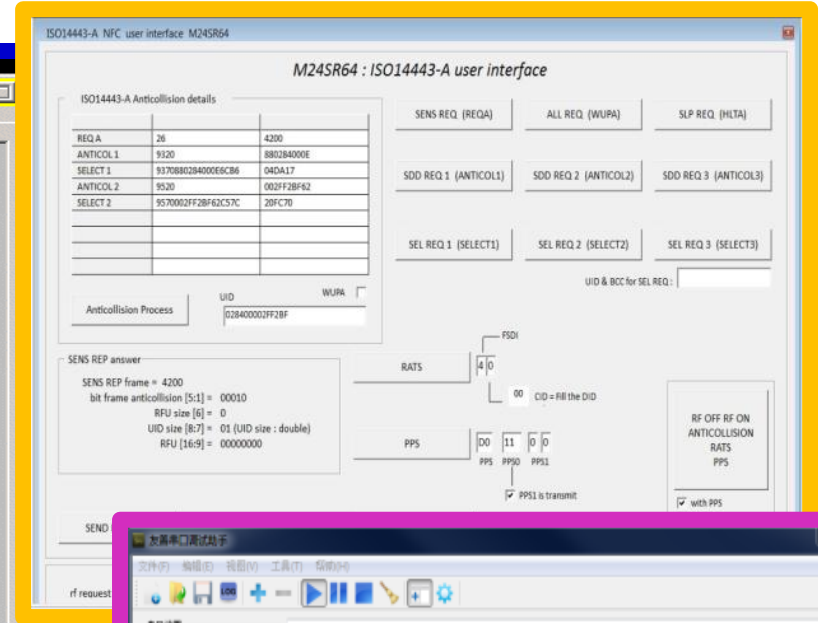
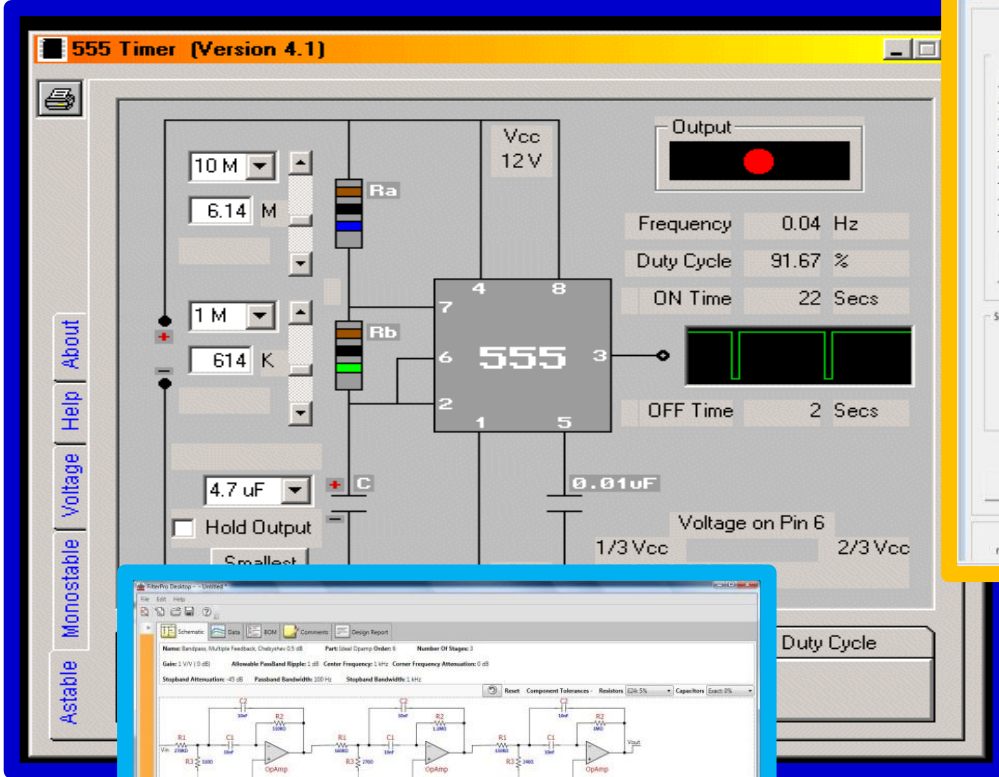


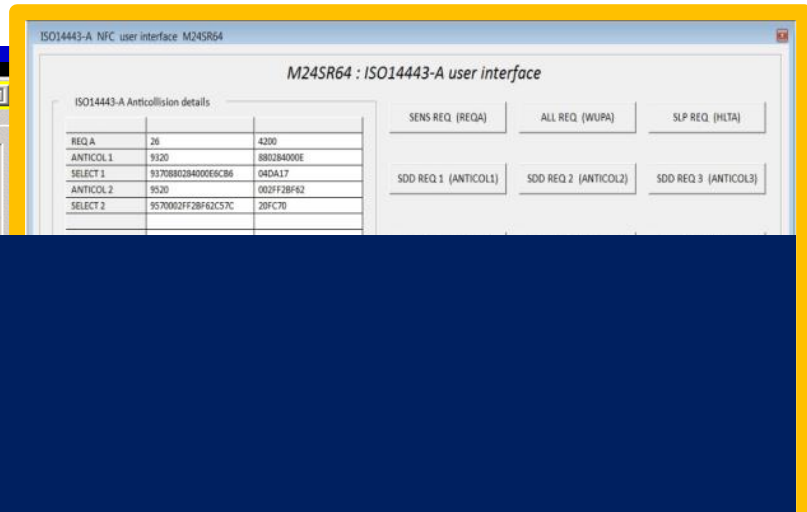
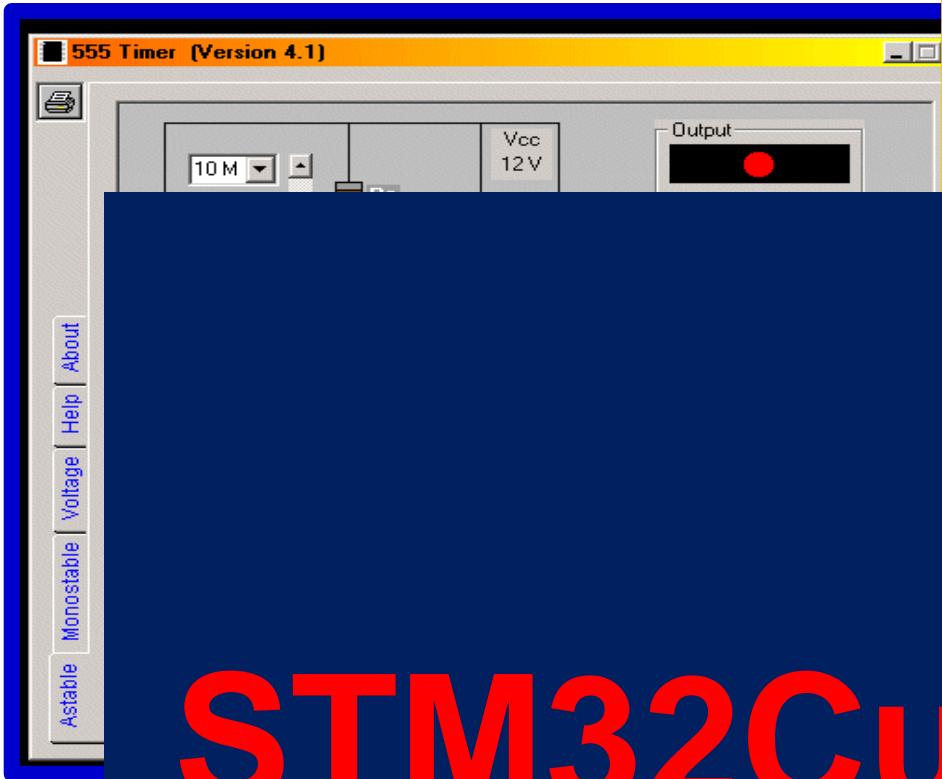
# STM32Cube™

2014年STM32Cube全国路演



# 你是否使用过这些软件？





# STM32Cube™ ...

# 如何保证软件的兼容性？



**STM32 F0**  
入门型

**STM32 F1**  
主流型

**STM32 F2**  
高性能 MCU

**STM32 F3**  
模拟+ DSP

**STM32 F4**  
高性能  
DSP

**STM32 L1**  
超低功耗

**STM32 L0**  
超低功耗  
入门型

48 MHz  
1.8 to 3.6V  
面向8位、16  
位应用

24 ~ 72 MHz  
2.0 to 3.6V  
具有最宽的产  
品线分布

120 MHz  
1.7 to 3.6V  
高性能系列

72 MHz  
1.8V 或者 2.0  
~ 3.6V  
带DSP指令  
更强模拟功能

168 ~ 180 MHz  
1.7 to 3.6V  
超高性能  
带DSP指令

32 MHz  
1.65 to 3.6V  
具有宽泛的产  
品线分布

32 MHz  
1.71 ~ 3.6V  
面向8位、16位  
低功耗应用

## ST的32位MCU平台

CM0

CM3

CM4

CM0

CM0+

# 如何保证软件的兼容性？

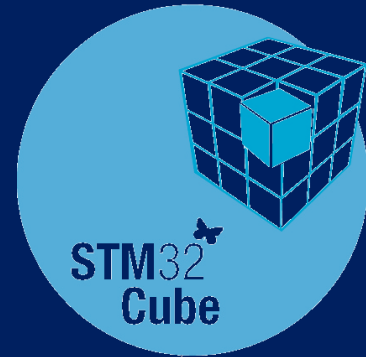
STM3

STM32  
入门型

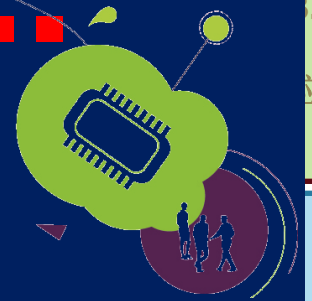
48 MHz  
1.8 to 3.3V  
面向8位、  
16位应用

L0  
低功耗型

100 MHz  
1.8 to 3.3V  
面向16位应用



# STM32Cube™



CM0

CM3

CM4

CM0

CM0+



完备的软件集合： 驱动+中间件+工具软件 6

**USB**    **File System**    **RTOS**

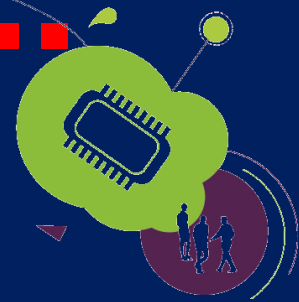
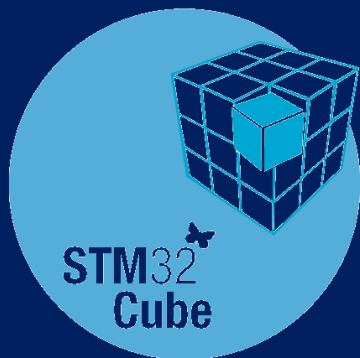
**TCP IP** 引脚分配工具    **Audio**

时钟配置工具    **Driver**    **GUI**

完备的软件集合： 驱动+中间件+工具软件

7

STM32Cube™





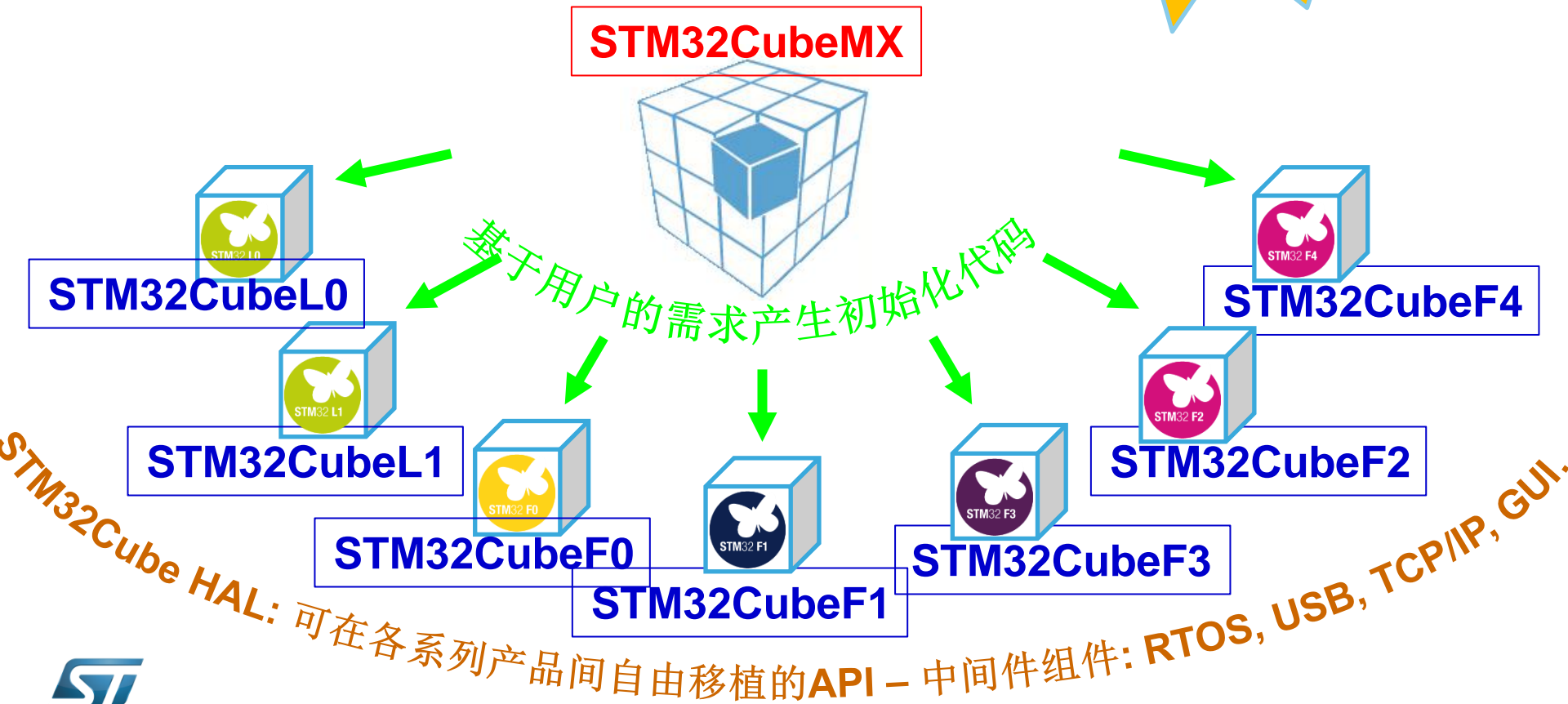
# STM32Cube™

## 加速您的设计



- STM32Cube™组件由两部分组成
  - PC端的图形化配置工具：[STM32CubeMX](#)
  - 基于STM32上的完备固件集合：[STM32Cube库](#)

均可从ST官网  
免费下载



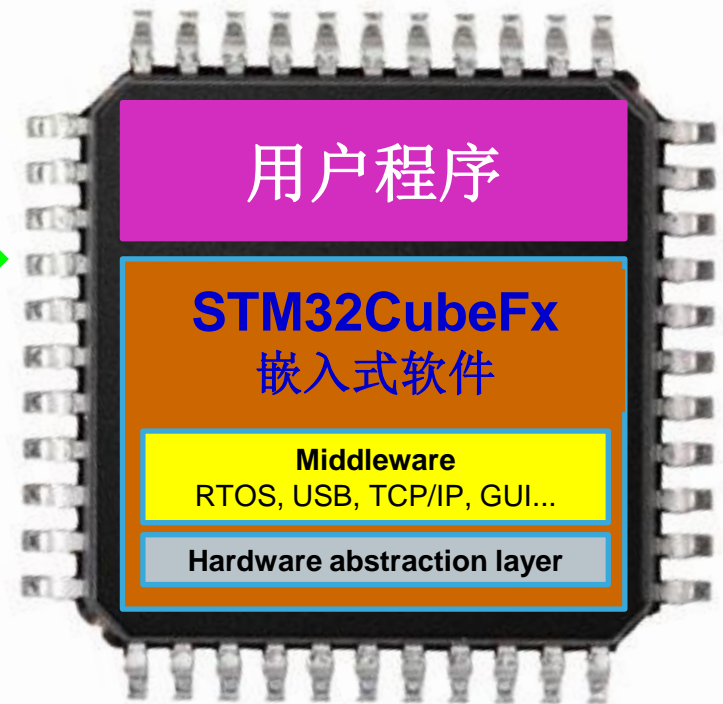
- STM32Cube™组件由两部分组成
  - PC端的图形化配置工具：[STM32CubeMX](#)
  - 基于STM32上的完备固件集合：[STM32CubeFx](#)

均可从ST官网  
免费下载

## STM32CubeMX



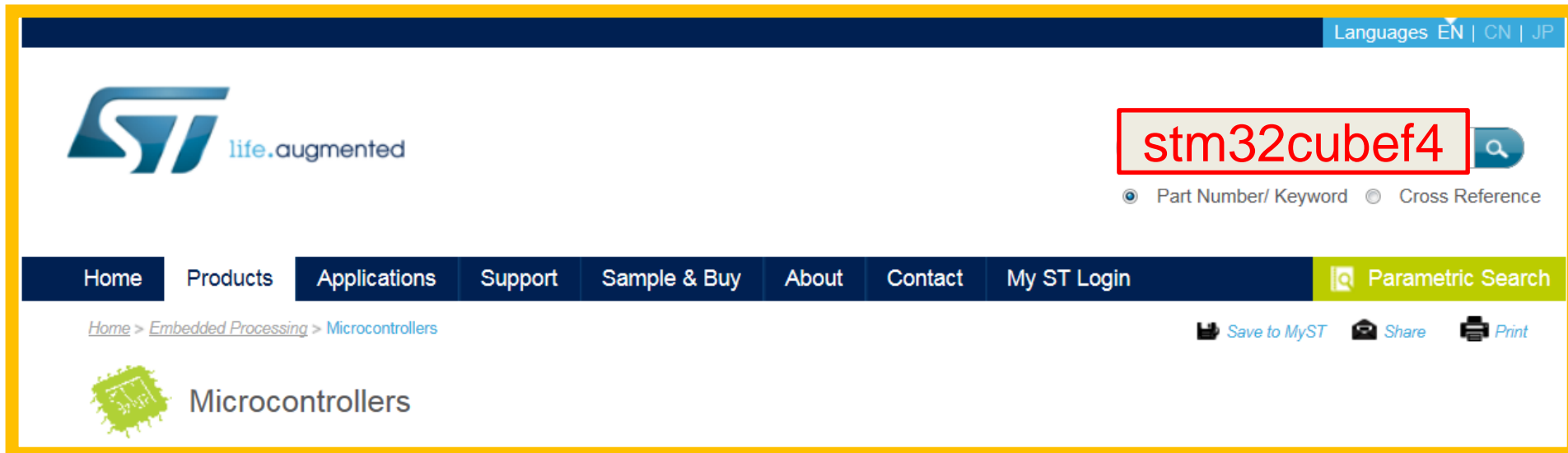
根据用户的需求  
生成工程项目及  
初始化代码



# 可在官网免费下载

11

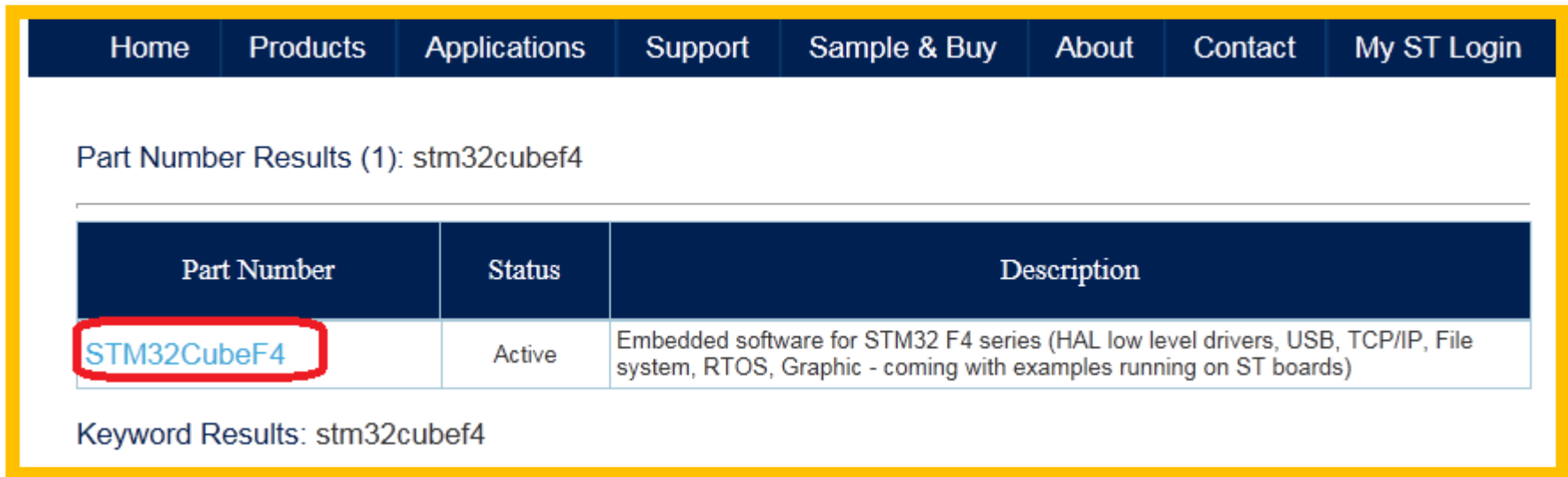
- 一键下载：获得该家族**唯一**且**完备**的固件库



The screenshot shows the STMicroelectronics website interface. At the top right, there are language options: "Languages EN | CN | JP". The ST logo and "life.augmented" tagline are on the left. A search bar in the top right contains the text "stm32cubef4" and has a search icon. Below the search bar, there are radio buttons for "Part Number/ Keyword" (selected) and "Cross Reference". A navigation menu below the search bar includes "Home", "Products", "Applications", "Support", "Sample & Buy", "About", "Contact", and "My ST Login". To the right of the menu is a "Parametric Search" button. Below the menu, there is a breadcrumb trail: "Home > Embedded Processing > Microcontrollers". On the right side, there are icons for "Save to MyST", "Share", and "Print". At the bottom left, there is a "Microcontrollers" category icon and text.

**st**官网任意页面的搜索栏输入：  
**“stm32cubef4”**

- 一键下载：获得该家族**唯一**且**完备**的固件库



The screenshot shows the ST website navigation bar with links: Home, Products, Applications, Support, Sample & Buy, About, Contact, and My ST Login. Below the navigation bar, the search results for 'stm32cubef4' are displayed. The results show one part number, 'STM32CubeF4', which is highlighted with a red box. The status is 'Active' and the description is 'Embedded software for STM32 F4 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards)'. Below the table, the keyword results are also shown as 'stm32cubef4'.

Part Number	Status	Description
<a href="#">STM32CubeF4</a>	Active	Embedded software for STM32 F4 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards)

搜索结果第一条就是你想要的！  
点击超级链接...

# STM32CubeF4 STM32CubeF4的资源页面

Embedded software for STM32 F4 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards)

● Active

STM32Cube™ is an STMicroelectronics original initiative to ease developers' life by reducing development efforts, time and cost. STM32Cube™ covers STM32 portfolio.

STM32Cube™ includes the STM32CubeF4 which is a graphical software configuration tool that allows generating C initialization code using graphical wizards.

It also embeds a comprehensive software platform, delivered per series (such as STM32CubeF4 for STM32F4 series). This platform includes the STM32Cube HAL (an STM32 abstraction layer embedded software, ensuring maximized portability across STM32 portfolio), plus a consistent set of middleware components (RTOS, USB, TCP/IP and graphics). All embedded software utilities come with a full set of examples.

STM32CubeF4 gathers in one single package all the generic embedded software components required to develop an application on STM32F4 microcontrollers. Following STM32Cube™ initiative, this set of components is highly portable, not only within STM32F4 series but also to other STM32 series.

Brochure

Description

Version

Size

 STM32 F4 series - High-performance Cortex-M4 MCU 2,645 KB

## Get Software

[Top](#)

Part Number	Version	Marketing Status	Order From ST
STM32CubeF4	1.3.0	Active	<a href="#">Download</a>

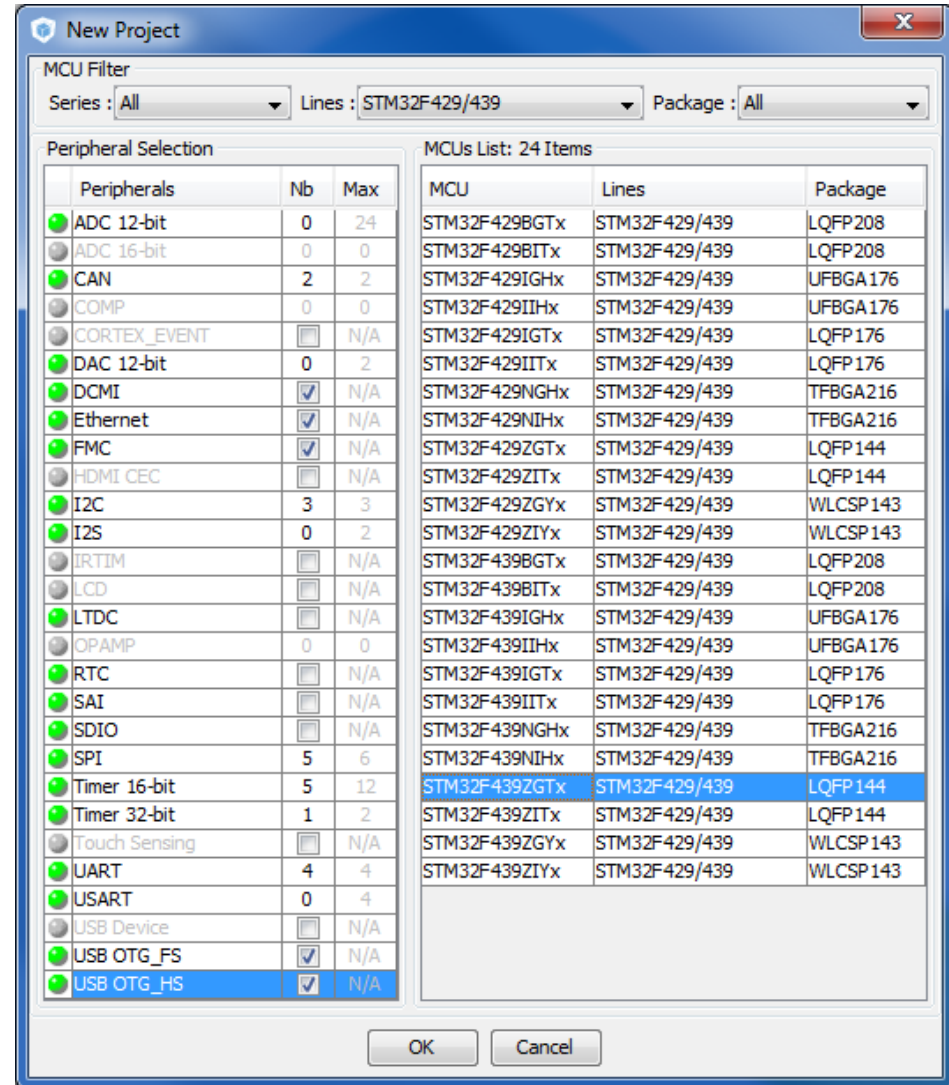


# STM32Cube™的第一部分 STM32CubeMX

# 配置工具：STM32CubeMX

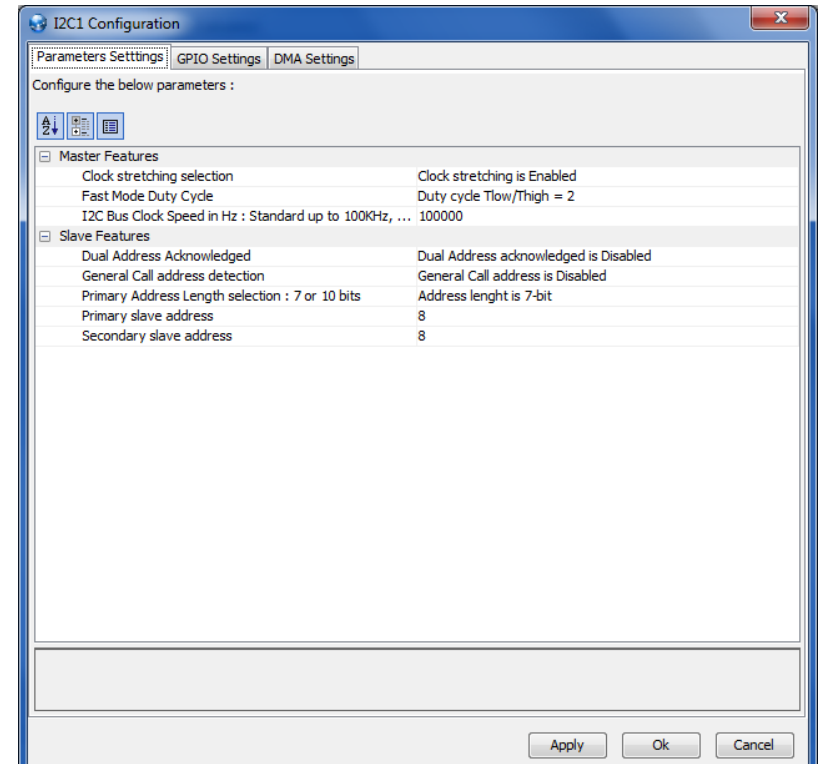
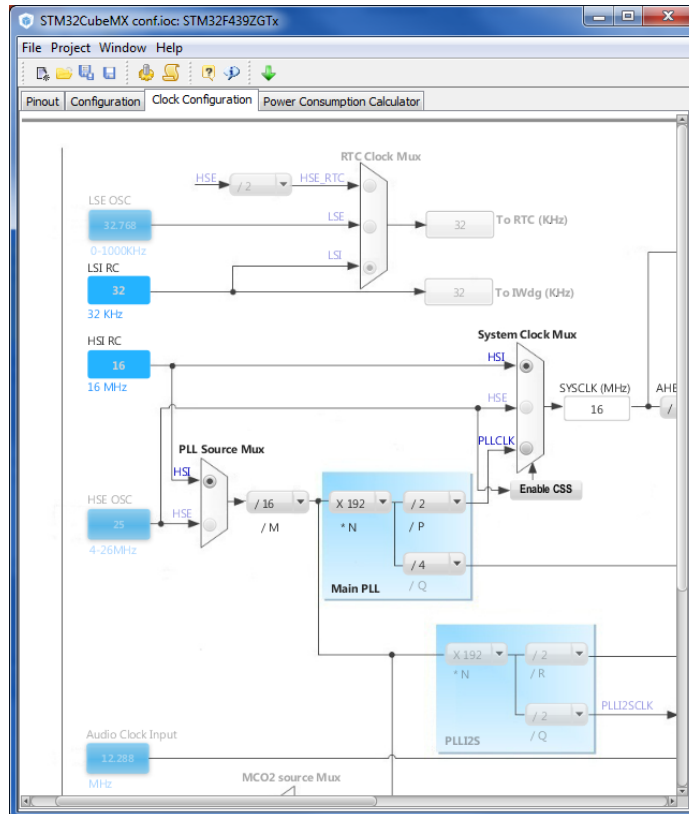
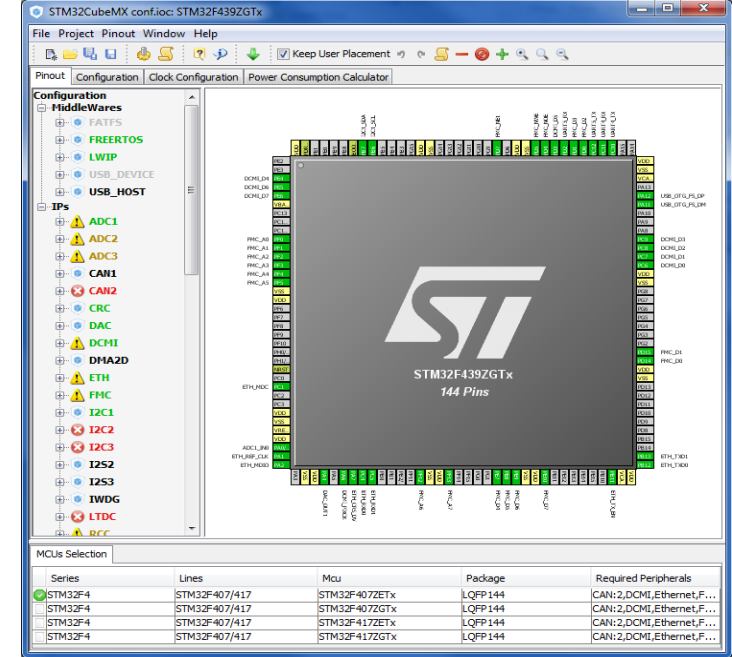
## • 选择MCU

- 通过型号、外设、封装进行过滤



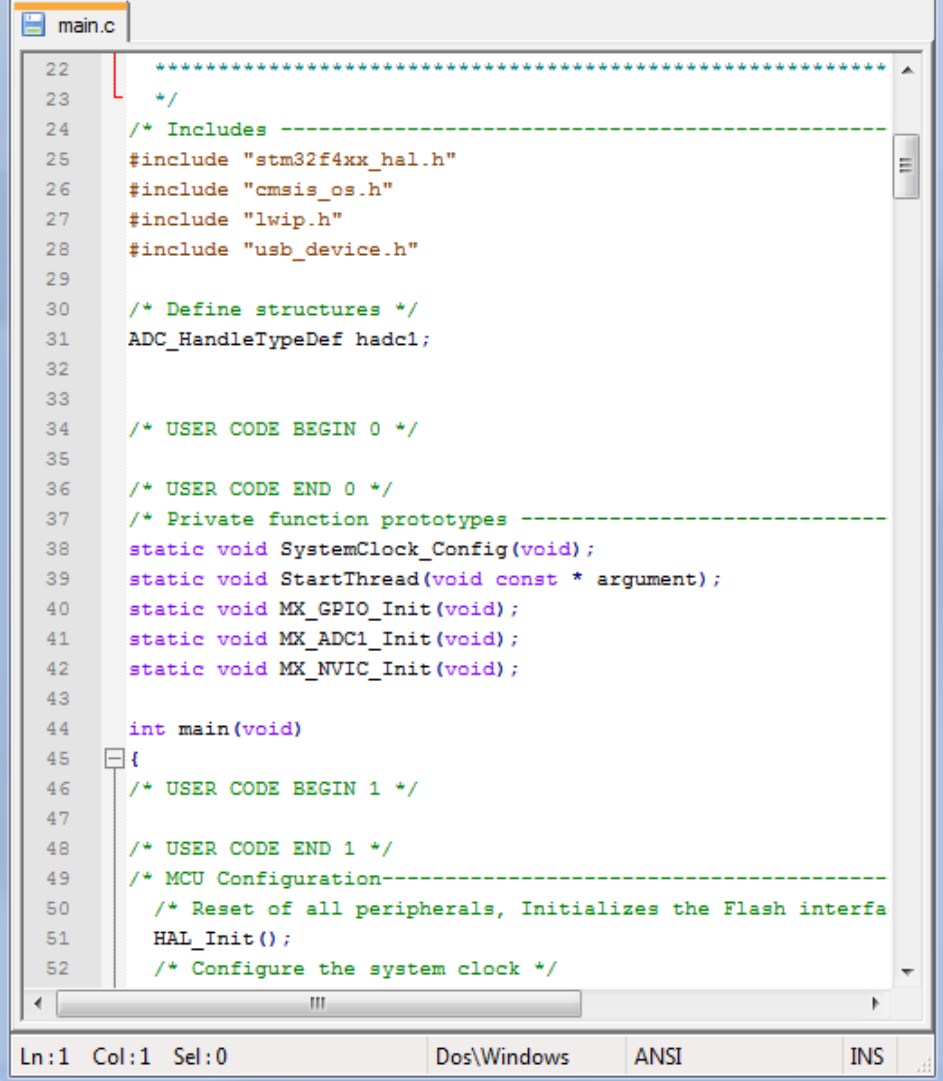
## 通过向导配置MCU

- 管脚分配和配置
- 时钟树配置
- 外设参数和中间件配置





- 生成初始化代码
  - 直接生成工程
  - 支持IAR, Keil, TrueStudio
- 两种形式的版本
  - 独立的PC端程序
  - 也可作为Eclipse插件运行

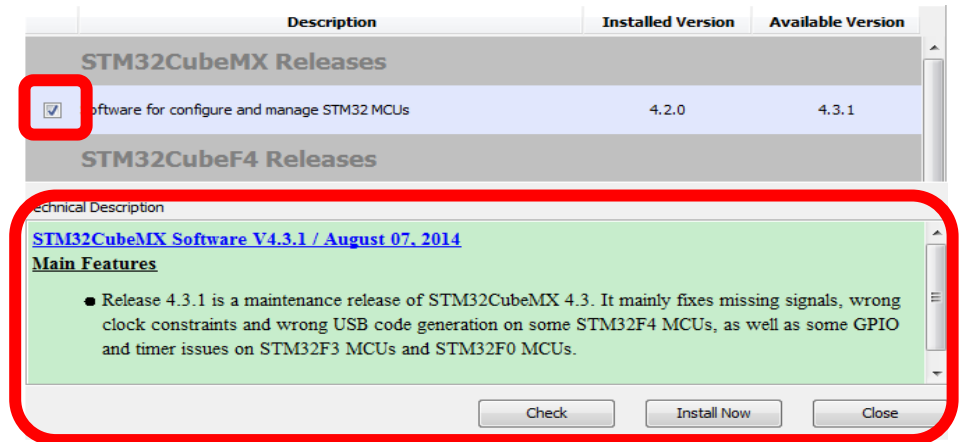
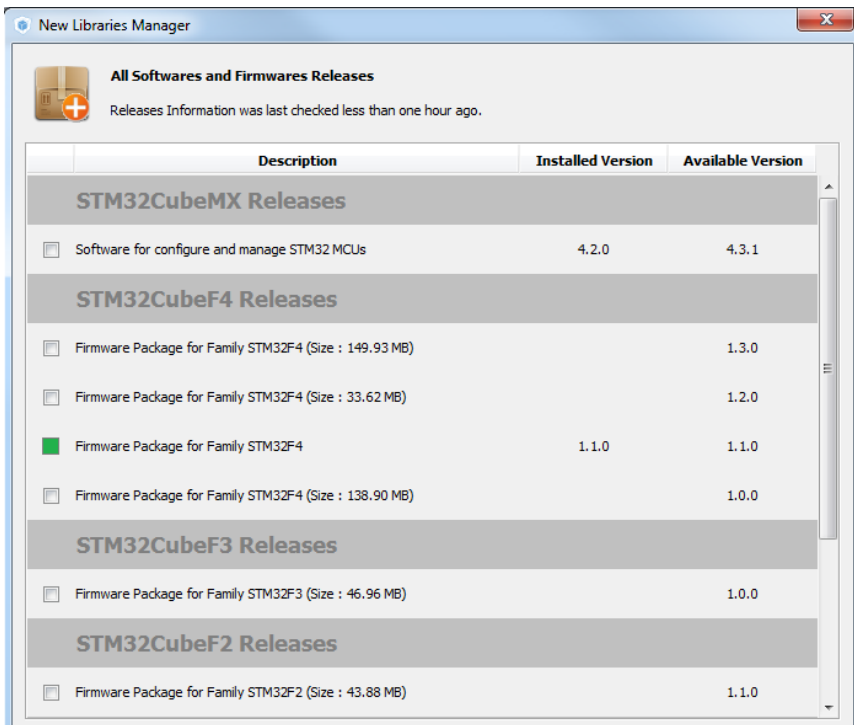


```
main.c
22  /*-----*/
23  */
24  /* Includes -----*/
25  #include "stm32f4xx_hal.h"
26  #include "cmsis_os.h"
27  #include "lwip.h"
28  #include "usb_device.h"
29
30  /* Define structures */
31  ADC_HandleTypeDef hadc1;
32
33
34  /* USER CODE BEGIN 0 */
35
36  /* USER CODE END 0 */
37  /* Private function prototypes -----*/
38  static void SystemClock_Config(void);
39  static void StartThread(void const * argument);
40  static void MX_GPIO_Init(void);
41  static void MX_ADC1_Init(void);
42  static void MX_NVIC_Init(void);
43
44  int main(void)
45  {
46  /* USER CODE BEGIN 1 */
47
48  /* USER CODE END 1 */
49  /* MCU Configuration-----*/
50  /* Reset of all peripherals, Initializes the Flash interfa
51  HAL_Init();
52  /* Configure the system clock */
```

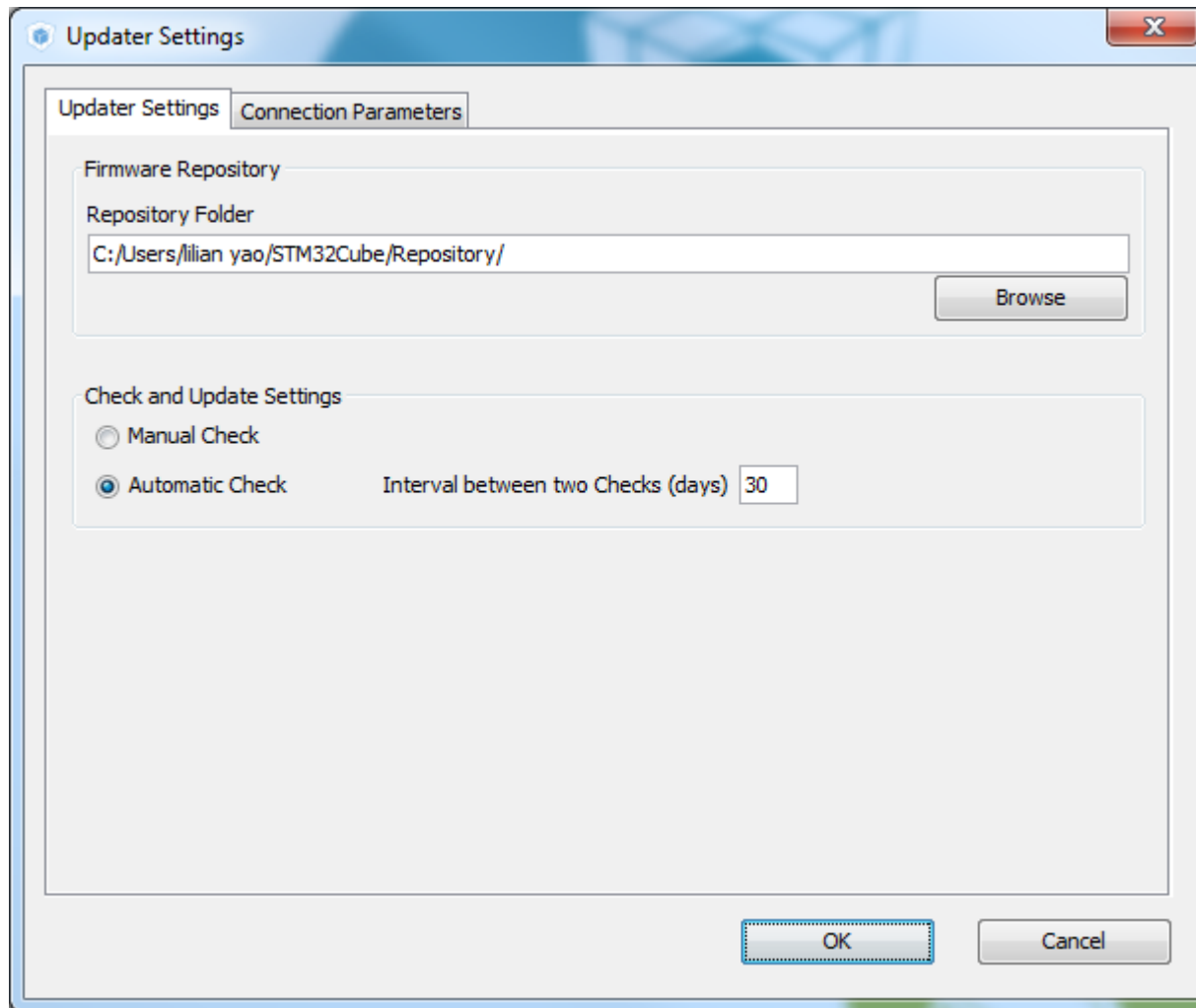
Ln:1 Col:1 Sel:0 Dos\Windows ANSI INS

# 配置工具：STM32CubeMX

- 支持版本检测



- 支持软件自动更新

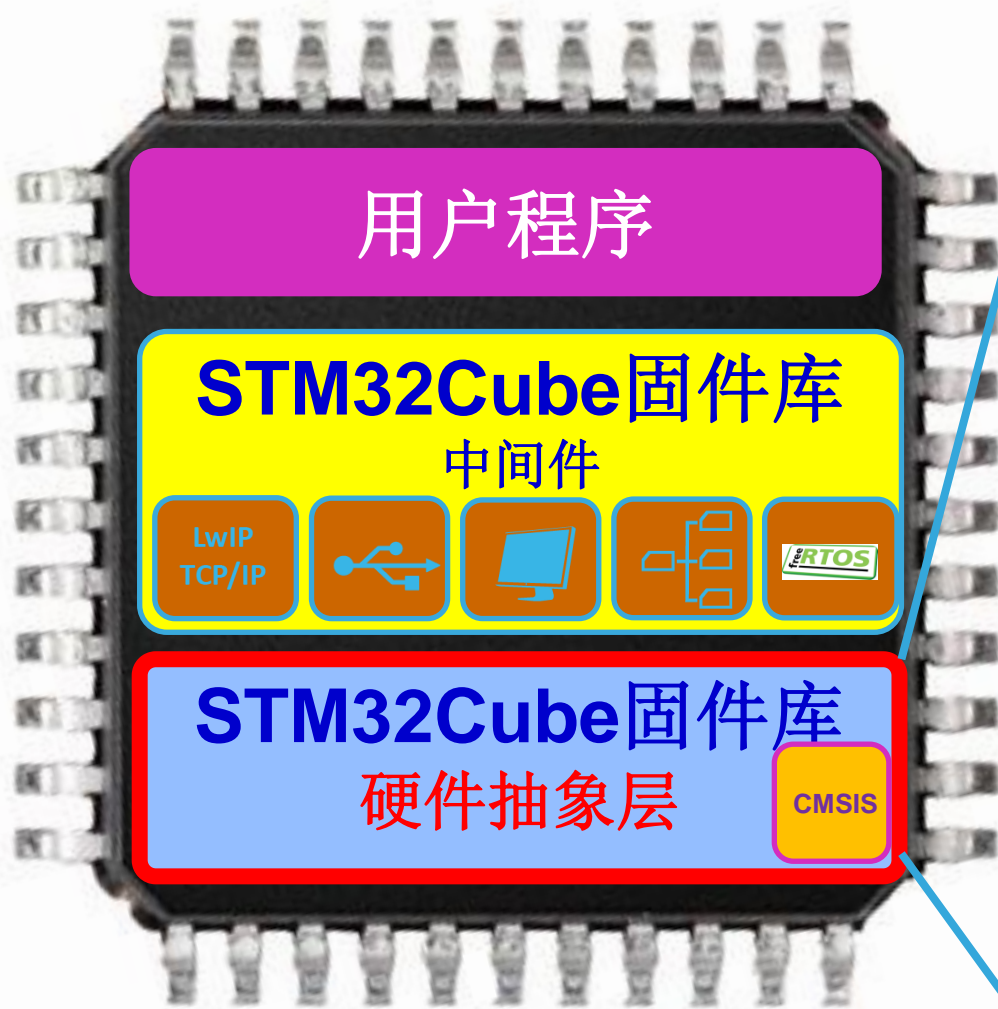




# STM32Cube™的第二部分

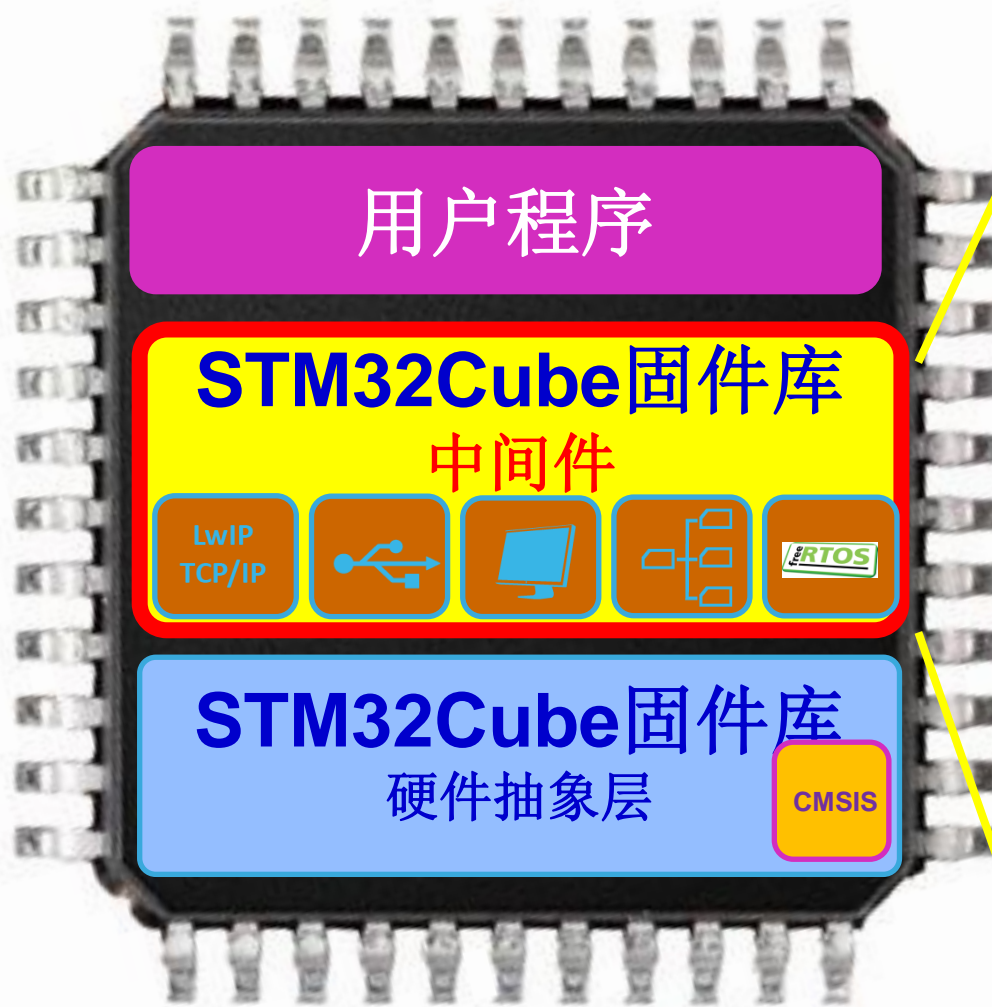
## STM32Cube固件库

# 完备的软件集合：STM32Cube固件库



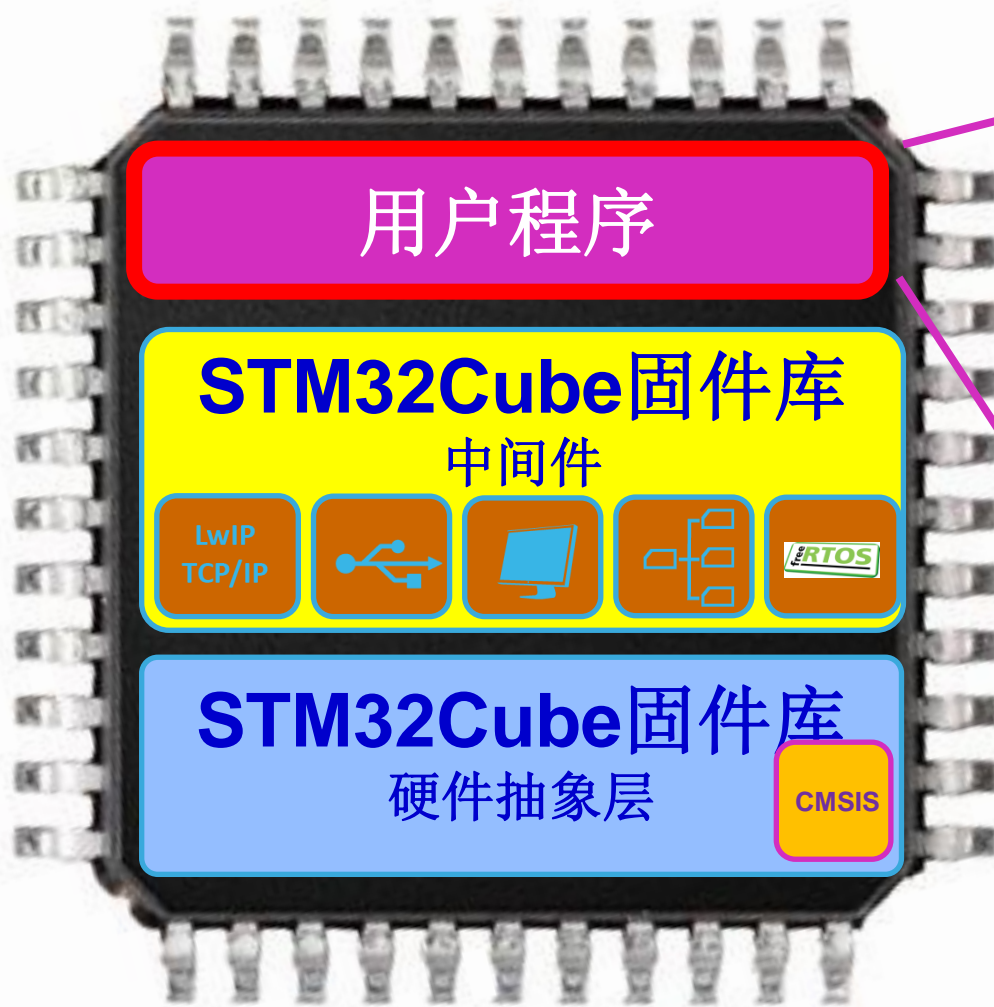
- **STM32硬件抽象层驱动**
  - 友好的接口更易于移植
- **支持所有的外设**
  - 包括Core,
  - Ethernet, USB, SDIO等
- **代码可靠性强**
  - 经过CodeSonar测试
- **丰富的例程**
  - 超过150个例程
- **支持STM32全系列产品**
  - 已支持F2, F3, F4, F0, L0
  - 将会支持F1, L1

# 完备的软件集合：STM32Cube固件库



- **TCP/IP协议栈**
  - LwIP、http、DHCP...
- **USB library**
  - ST开发的USB主机/设备库
- **GUI**
  - STemWin (ST和Segger)
- **文件系统**
  - FatFS
- **实时操作系统**
  - FreeRTOS (遵循CMSIS-RTOS标准)
- **超过50个例程**

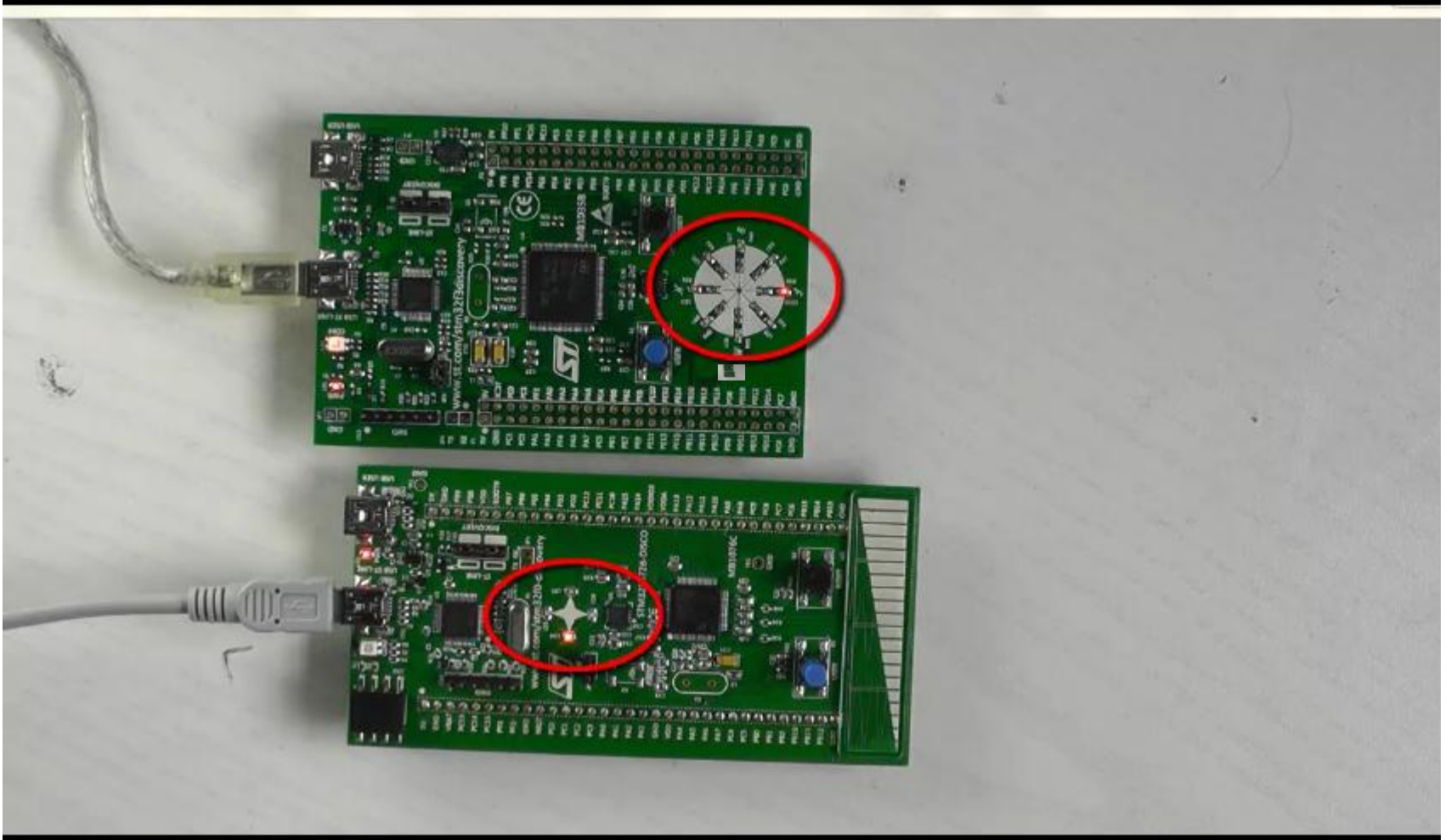
# 完备的软件集合：STM32Cube固件库



- 用户只需要专注于应用开发

```
/* USER CODE BEGIN 1 */  
void user_code();  
/* USER CODE END 1 */  
/* The code will be  
kept upon  
generation */
```

# 来看一段视频...





# F0和F3平台上的应用程序

25

```
main.c f0
/* USER CODE BEGIN 3 */
/* Infinite loop */
while (1)
{
#define GPIO_USER_DEFINE      GPIOC
#define GPIO_PIN_LEFT        GPIO_PIN_8
#define GPIO_PIN_UP          GPIO_PIN_6
#define GPIO_PIN_RIGHT       GPIO_PIN_9
#define GPIO_PIN_DOWN        GPIO_PIN_7

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_LEFT);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_UP);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_RIGHT);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_DOWN);
HAL_Delay(100);
}
/* USER CODE END 3 */

/** System Clock Configuration
*/
```

1. 通过宏定义  
屏蔽平台差别

```
Project - IAR Embedded Workbench IDE
File Edit View Project Tools Window Help
st_jump_usr_app
main.c
/* USER CODE BEGIN 3 */
/* Infinite loop */
while (1)
{
#define GPIO_USER_DEFINE      GPIOE
#define GPIO_PIN_LEFT        GPIO_PIN_15
#define GPIO_PIN_UP          GPIO_PIN_9
#define GPIO_PIN_RIGHT       GPIO_PIN_11
#define GPIO_PIN_DOWN        GPIO_PIN_13

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_LEFT);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_UP);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_RIGHT);
HAL_Delay(100);

HAL_GPIO_TogglePin(GPIO_USER_DEFINE, GPIO_PIN_DOWN);
HAL_Delay(100);
}
/* USER CODE END 3 */

/** System Clock Configuration
*/
```

2. 调用HAL API操作MCU硬件模块  
→ HAL\_GPIO\_TogglePin()  
→ HAL\_Delay()



# STM32Cube固件库的**HAL**

## 特性

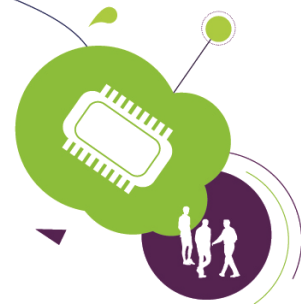
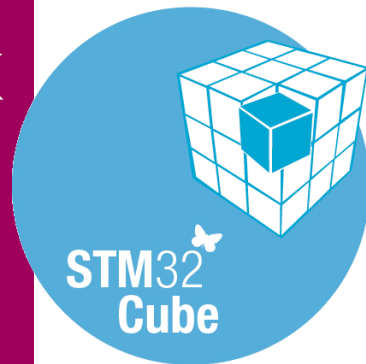
两种API类型

三种编程模型

安全性

支持模块多实例

灵活的回调机制



## 目的

对标准外设固件库的替换

提供友好易用的API

提供高度可移植的API

# HAL特性1：两种API

# 特性

## 两种API类型

### HAL通用API

普遍适用于所有STM32产品

## 通用性

### HAL扩展API

专用于特定家族或者  
特定产品线的STM32产品

## 扩展性

- 为程序的模块化设计带来便利
- 跨平台的共用外设的API
  - 初始化函数
    - 时钟，外设等自定义/缺省初始化
  - IO操作函数
    - 交互式数据通讯都以读写的方式进行访问
  - 控制函数
    - 动态配置外设参数
  - 状态函数
    - 返回运行状态及出错信息

stm32f4xx\_hal\_ppp.c/.h

## API类型举例

HAL\_USART\_Init()  
HAL\_USART\_DeInit()  
HAL\_SPI\_Receive()  
HAL\_SPI\_Receive\_IT()  
HAL\_USART\_Transmit\_DMA()  
HAL\_ADC\_ConfigChannel()  
HAL\_RTC\_SetAlarm()  
HAL\_I2C\_GetState()  
HAL\_I2C\_GetError()

## HAL通用API

初始化  
函数

I/O操作  
函数

控制  
函数

状态  
函数

- 兼顾STM32各系列产品的特有功能和扩展性能
- 提高HAL驱动的扩展性
  - 某个家族系列特有的特性
  - 家族内部不同产品线之间不同的特性
  - 某个产品线特有的外设

stm32f4xx\_hal\_ppp\_ex.c/.h

例如：

stm32f4xx\_hal\_adc.c

stm32f4xx\_hal\_adc\_ex.c

## HAL扩展API

专用于某个家族的功能API

专用于某个产品线的功能API

## HAL特性2：三种编程模型

# 特性

## 三种编程模型

### 轮询

### 中断

### DMA

## 外设句柄

HAL中实现了三种编程模型，用户调用相应API的即可

每个外设实例有自己的句柄，作为HAL API的函参

- 该结构用于保存具体外设实例的所有参数
  - 初始化配置参数
  - I/O缓冲区（可选）
  - 外设状态
  - DMA通道句柄

句柄内容随外设不同可能有所不同

ADC1的句柄	
Field	类型
<b>*Instance</b>	ADC_TypeDef
<b>Init</b>	ADC_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
<b>*DMA_Handle</b>	DMA_HandleTypeDef

USART2的句柄	
Field	类型
<b>*Instance</b>	USART_TypeDef
<b>Init</b>	USART_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
hdmatx	DMA_HandleTypeDef*
hdmarx	DMA_HandleTypeDef*



- 该结构用于保存具体外设实例的所有参数

- 初始化配置参数
- I/O缓冲区（可选）
- 外设状态
- DMA通道句柄

句柄内容随外设不同可能有所不同

ADC1的句柄	
Field	类型
*Instance	ADC_TypeDef
Init	ADC_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
*DMA_Handle	DMA_HandleTypeDef

USART2的句柄	
Field	类型
*Instance	USART_TypeDef
Init	USART_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
hdmatx	DMA_HandleTypeDef*
hdmarx	DMA_HandleTypeDef*

- 该结构用于保存具体外设实例的所有参数

- 初始化配置参数
- I/O缓冲区（可选）
- 外设状态
- DMA通道句柄

句柄内容随外设不同可能有所不同

### ADC1的句柄

Field	类型
*Instance	ADC_TypeDef
Init	ADC_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
*DMA_Handle	DMA_HandleTypeDef

### USART2的句柄

Field	类型
*Instance	USART_TypeDef
Init	USART_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
hdmatx	DMA_HandleTypeDef*
hdmarx	DMA_HandleTypeDef*

- 该结构用于保存具体外设实例的所有参数

- 初始化配置参数
- I/O缓冲区（可选）
- 外设状态
- DMA通道句柄**

句柄内容随外设不同可能有所不同

### ADC1的句柄

Field	类型
*Instance	ADC_TypeDef
Init	ADC_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
<b>*DMA_Handle</b>	<b>DMA_HandleTypeDef</b>

### USART2的句柄

Field	类型
*Instance	USART_TypeDef
Init	USART_InitTypeDef
pTxBuffPtr	uint8_t*
pRxBuffPtr	uint8_t*
TxXferSize	uint16_t
RxXferSize	uint16_t
TxXferCount	uint16_t
RxXferCount	uint16_t
NbrOfCurrentConversionRank	__IO uint32_t
Lock	HAL_LockTypeDef
State	USART_StateTypeDef
ErrorCode	uint8_t
<b>hdmatx</b>	<b>DMA_HandleTypeDef*</b>
<b>hdmarx</b>	<b>DMA_HandleTypeDef*</b>

- 以ADC转换为例

```
/*##-1- Configure the ADC peripheral #####/  
  AdcHandle.Instance = ADCx;  
  AdcHandle.Init.ClockPrescaler = ADC_CLOCKPRESCALER_PCLK_DIV2;  
  AdcHandle.Init.Resolution = ADC_RESOLUTION12b;  
  
  .....  
  if(HAL_ADC_Init(&AdcHandle) != HAL_OK)  
  { Error_Handler(); }  
/*##-2- Configure ADC regular channel #####/  
  sConfig.Channel = ADCx_CHANNEL;  
  
  .....  
  if(HAL_ADC_ConfigChannel(&AdcHandle, &sConfig) != HAL_OK)  
  { Error_Handler(); }
```

```
/*##-3- Start the conversion process #####/  
if(HAL_ADC_Start(&AdcHandle) != HAL_OK)  
{ Error_Handler(); }
```

轮询模式

```
/*##-3- Start the conversion process #####/  
if(HAL_ADC_StartIT(&AdcHandle) != HAL_OK)  
{ Error_Handler(); }
```

中断模式

```
/*##-3- Start the conversion process #####/  
if(HAL_ADC_StartDMA(&AdcHandle, &uhADCxConvertedValue, 1) != HAL_OK)  
{ Error_Handler(); }
```

DMA模式

# 三种工作模式 (2)

- 转换结果的处理

```
/*##-4- Wait for the end of conversion #####/ 轮询模式
HAL_ADC_PollForConversion(&AdcHandle, 10);
if(HAL_ADC_GetState(&AdcHandle) == HAL_ADC_STATE_EOC_REG)
{
    /*##-5- Get the converted value of channel ##*/
    uhADCxConvertedValue = HAL_ADC_GetValue(&AdcHandle);
}
/* Infinite loop */
while(1);
```

```
/* Infinite loop */
while(1);

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* AdcHandle)
{ /* Get the converted value of regular channel */
    uhADCxConvertedValue = HAL_ADC_GetValue(AdcHandle);
}
```

```
/* Infinite loop */  
while(1);
```

**DMA模式**

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* AdcHandle)  
{ /* Turn LED1 on: Transfer process is correct */  
  BSP_LED_On(LED1);  
}
```

```
/*###-3- Start the conversion process #####/
```

**DMA模式**

```
if(HAL_ADC_StartDMA(&AdcHandle,&uhADCxConvertedValue,1) != HAL_OK)  
{ Error_Handler(); }
```

# 特性

安全性

## HAL特性3： 安全性

39

保护锁

对资源性操作增加  
操作锁的保护

超时机制

轮询模式下的API  
增加超时机制，避免  
无限等待

- 以USART发送API为例

```
HAL_StatusTypeDef HAL_UART_Transmit
(UART_HandleTypeDef *huart, uint8_t *pData, uint16_t Size, uint32_t Timeout)
{
    /* Process Locked */
    __HAL_LOCK(huart);
    .....

    if(UART_WaitOnFlagUntilTimeout(huart, UART_FLAG_TXE, RESET, Timeout) != HAL_OK)
    {
        return HAL_TIMEOUT;
    }
    huart->Instance->DR = (*pData++ & (uint8_t)0xFF);

    if(UART_WaitOnFlagUntilTimeout(huart, UART_FLAG_TC, RESET, Timeout) != HAL_OK)
    {
        return HAL_TIMEOUT;
    }

    .....
    /* Process Unlocked */
    __HAL_UNLOCK(huart);
}
```

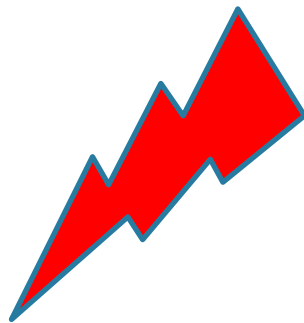


# 特性

灵活的回调机制

## HAL特性5: 灵活的回调机制

41



触发机制:

1. 初始化时
2. 中断事件
3. 错误

用户  
回调  
函数

外设初始化回调接口

事件回调接口

处理完成/出错回调

中断处理

- HAL库实现了各个外设的中断处理

- 用户必须在<stm32fxx\_it.c>中调用
- **HAL\_XXX\_IRQHandler**

- 用户回调函数

- 默认定义成“weak”属性，使用时再在应用代码中实现
- 三个类型的用户回调函数

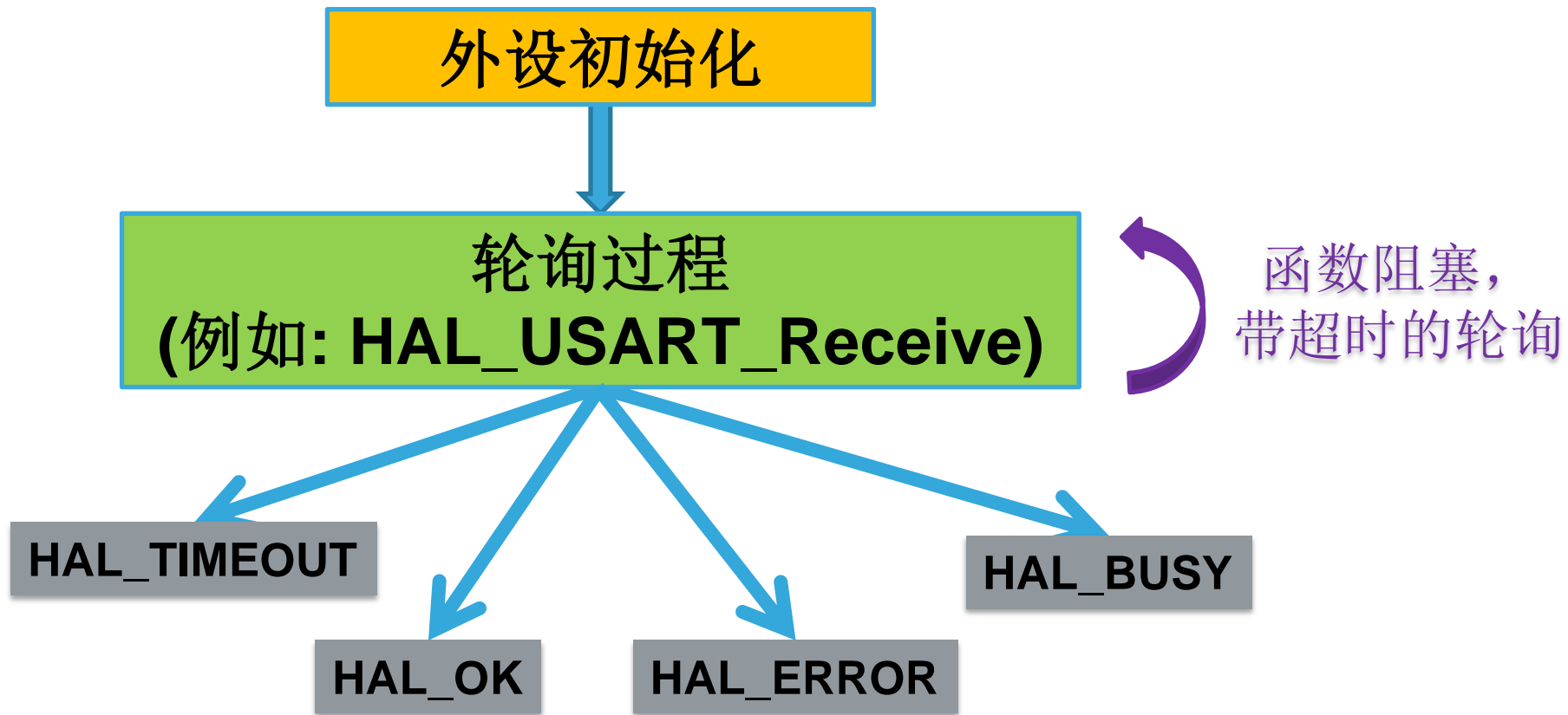
**Cube**将为您完成中断的查询和判断，用户只需关心如何处理中断和异常

回调函数类型	调用方式	作用
HAL_PPP_MspInit	被HAL_PPP_Init /DeInit调用	用户可在此做系统级初始化（GPIO、DMA、时钟）
HAL_PPP_ProcessCpltCallback HAL_PPP_ErrorCallback	使用中断或者DMA编程模型时	用户在此做任务完成后或出错清空下的处理
HAL_PPP_PeripEventCallback		处理过程中提供用户各个阶段的回调机会

回调函数类型	举例
HAL_PPP_MspInit	HAL_UART_MspInit()
HAL_PPP_ProcessCpltCallback HAL_PPP_ErrorCallback	HAL_UART_RxCpltCallback HAL_UART_ErrorCallback
HAL_PPP_PeripEventCallback	HAL_UART_RxHalfCpltCallback

# 编程模型和用户回调接口

## 阻塞型轮询调用过程



# 编程模型和用户回调接口 (2)

## 非阻塞型中断调用过程

外设和NVIC初始化

启动过程：结束时产生中断  
(例如: HAL\_USART\_Receive\_IT)

HAL\_OK

HAL\_ERROR

HAL\_BUSY

过程结束回调接口(例如:  
HAL\_USART\_RxCpltCallback)

过程出错回调接口(例如:  
HAL\_USART\_ErrorCallback)

PPP\_IRQHandler

HAL\_USART\_IRQHandler

# 编程模型和用户回调接口 (3)

## 非阻塞型DMA调用过程

外设和DMA初始化

启动过程：结束时产生DMA请求  
(例如: HAL\_USART\_Receive\_IT)

HAL\_OK

HAL\_ERROR

HAL\_BUSY

过程结束回调接口(例如:  
HAL\_DMA\_RxCpltCallback)

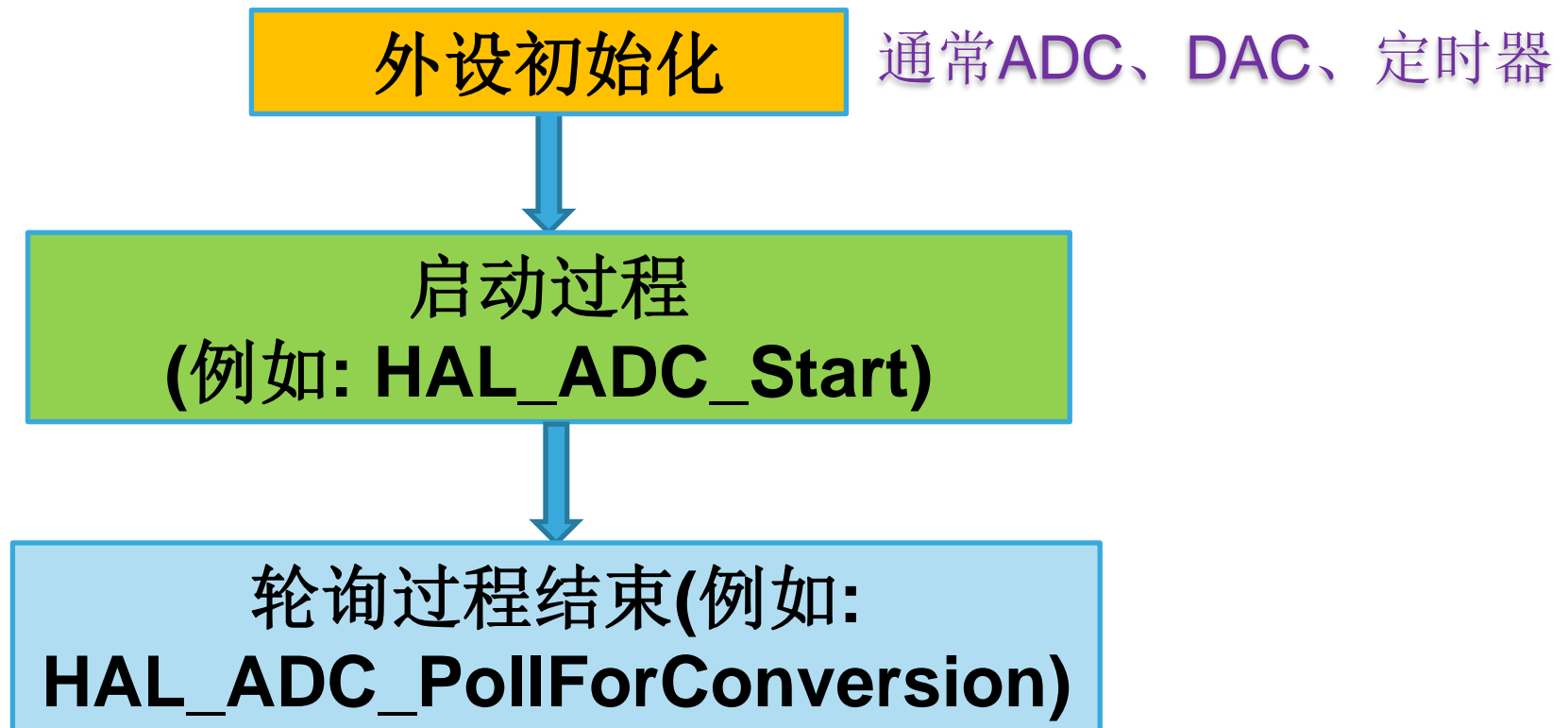
过程出错回调接口(例如:  
HAL\_USART\_ErrorCallback)

DMA\_Stream\_IRQ  
Handler

HAL\_DMA\_IRQHandler

# 编程模型和用户回调接口

## 非阻塞型启动过程



# STM32Cube库软件结构

**EVAL/Disk/Nuclo**  
评估板 Demo

中间件级别的例程

RTOS    USB    GUI

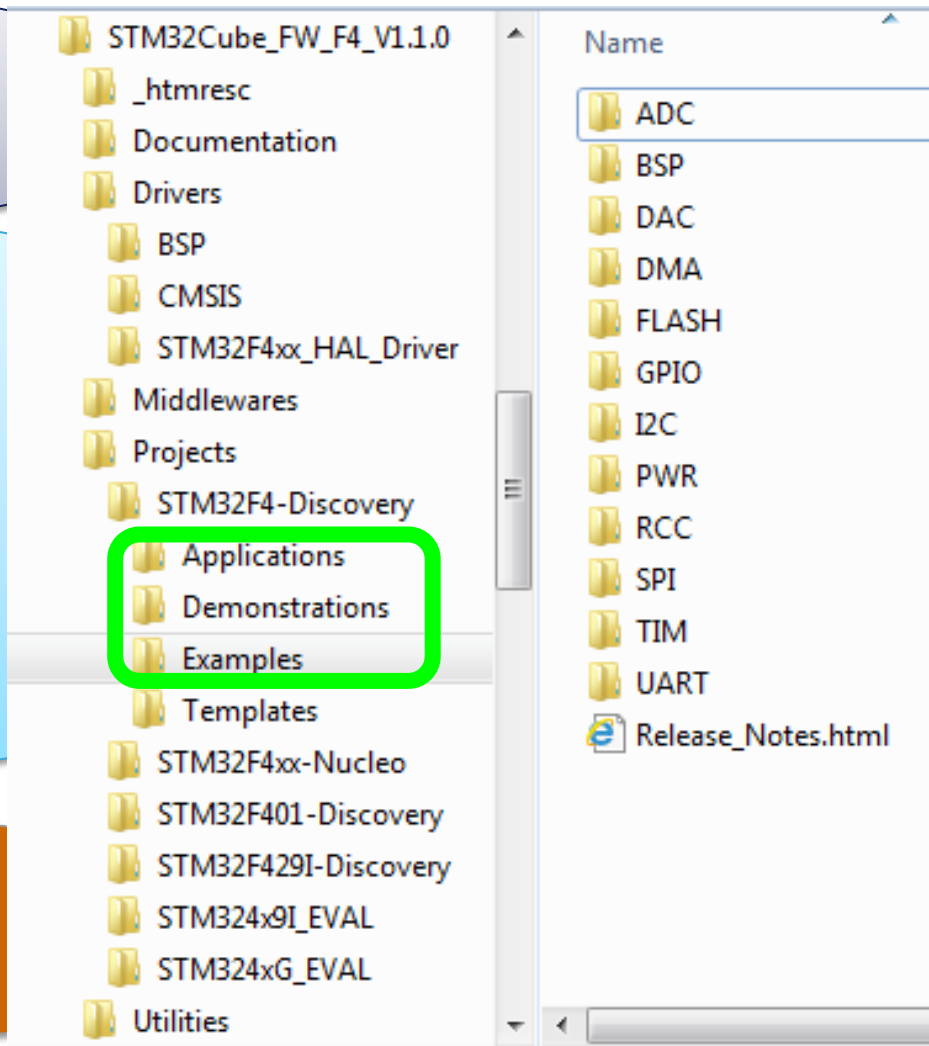
功能

HAL级别的示例

HAL 驱动库    BSP

CMSIS

STM32F0    STM32L1  
STM32F1    STM32L0  
STM32F2    STM32L4  
STM32F3    STM32L5





EVAL/Disk/Nuclo  
评估板Demo

中间件级别的例程

功能

RTOS

USB

GUI

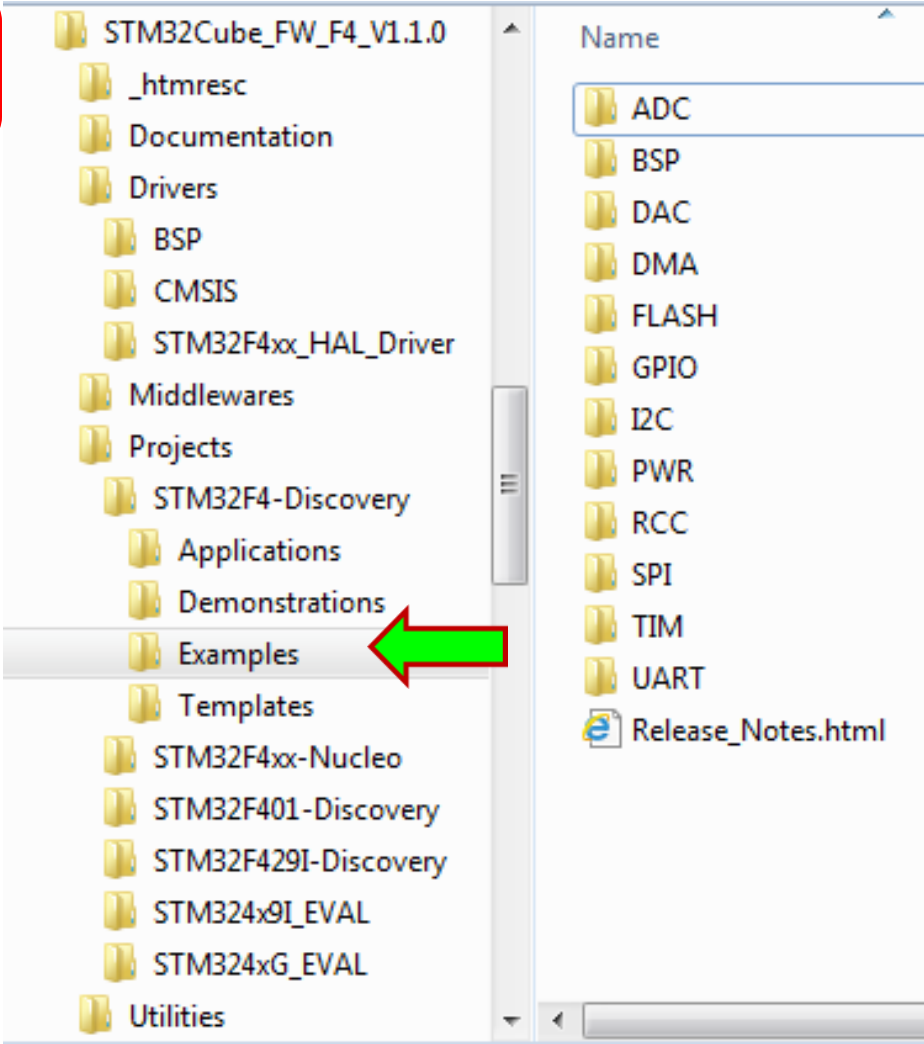
CMSIS

HAL级别的示例

HAL驱动库

BSP

STM32F4  
STM32F4  
STM32L1  
STM32L1  
STM32L1



EVAL/Disk/Nuclo  
评估板Demo

# 中间件级别的例程

RTOS

USB

GUI

功能

HAL级别的示例

HAL驱动库

BSP

CMSIS

STM32F0

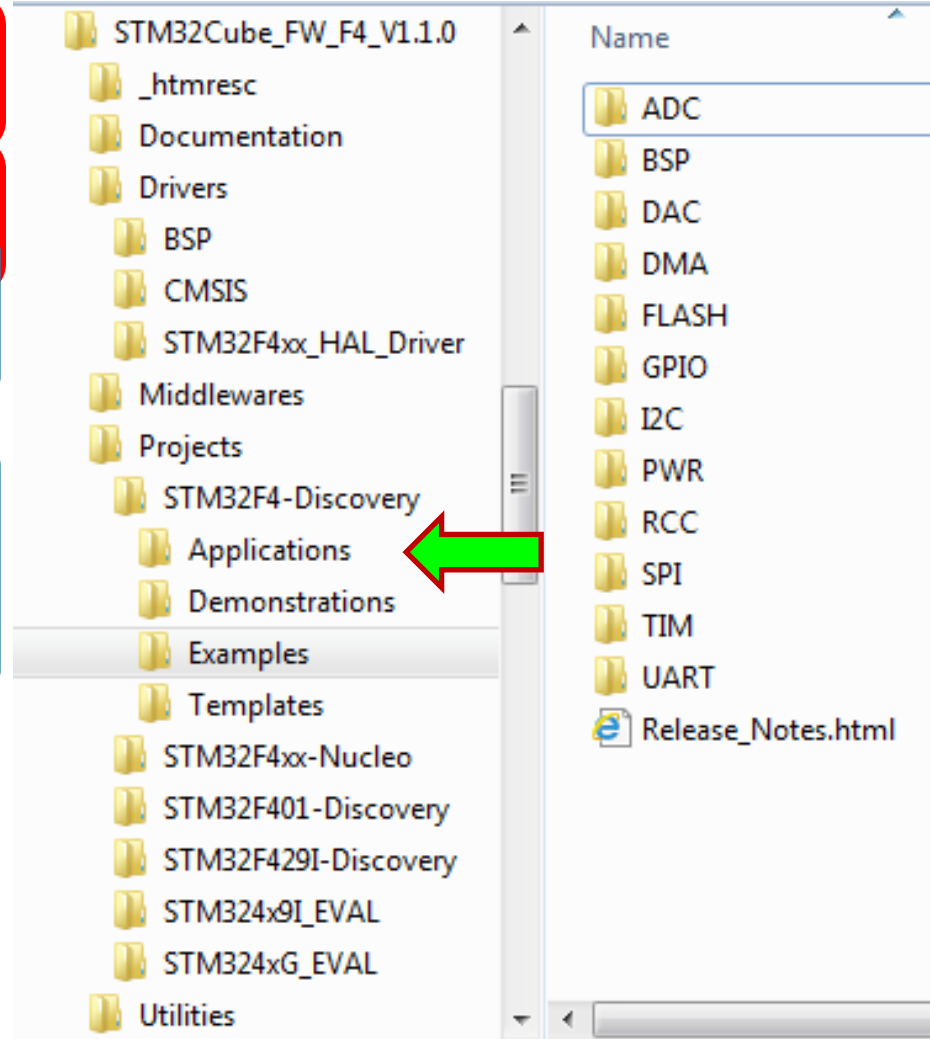
STM32L1

STM32F1

STM32L0

STM32F4

STM32L4



# EVAL/Disk/Nuclo 评估板 Demo

中间件级别的例程

RTOS

USB

GUI

功能

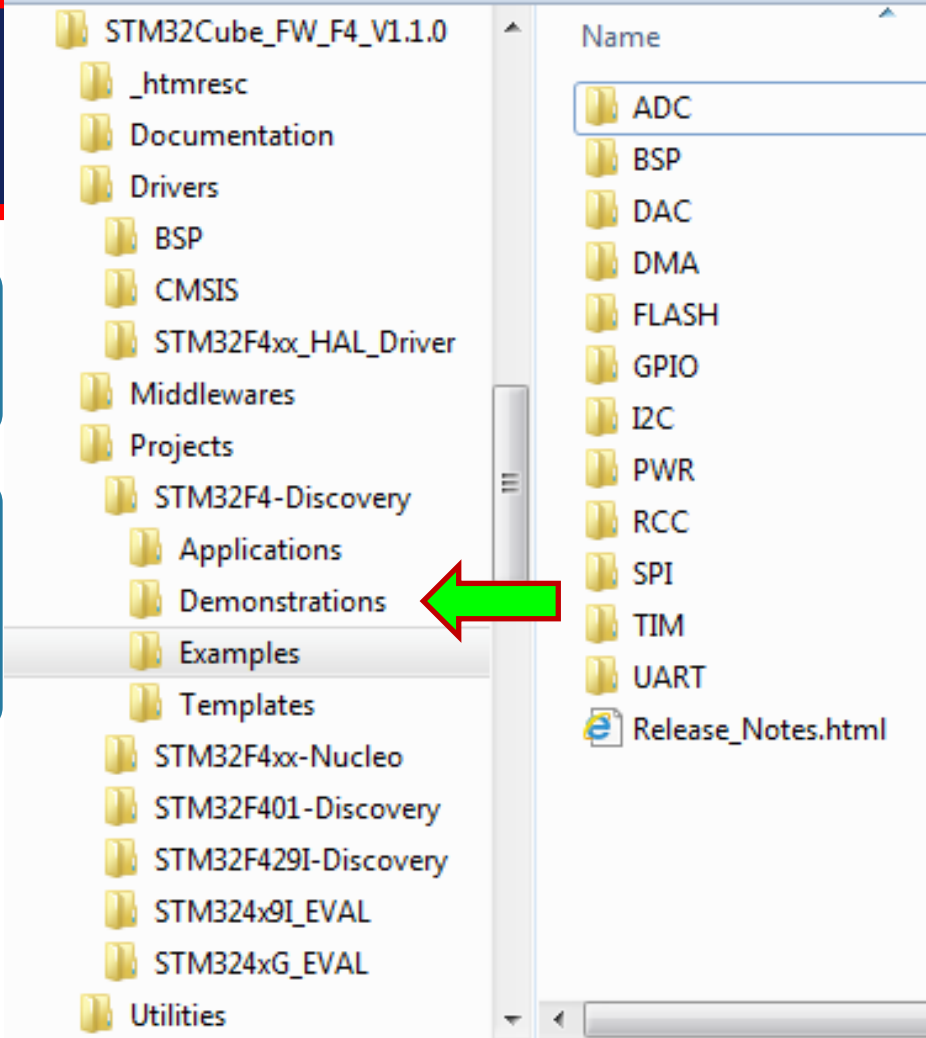
HAL级别的示例

HAL驱动库

BSP

CMSIS

STM32L1  
STM32L0  
STM32L4  
STM32L5



**EVAL/Disk/Nuclo**  
**评估板Demo**

**中间件级别的例程**

**RTOS** **U**

**超过150个例程**  
**供用户学习和参考**

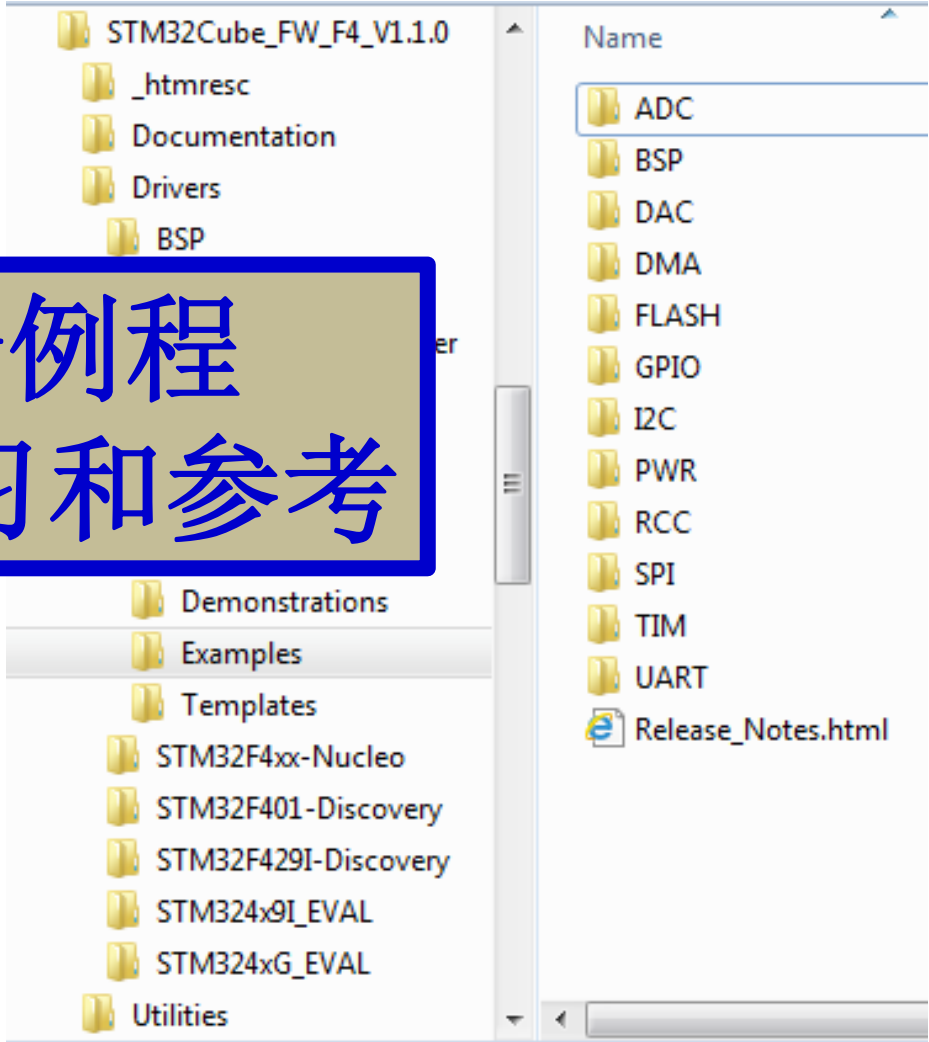
**HAL级**

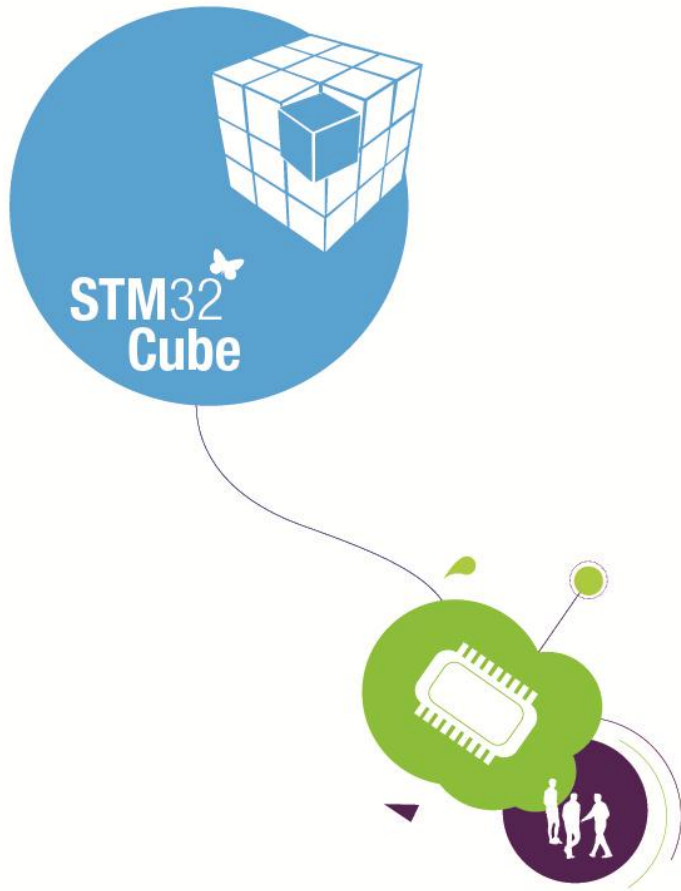
**HAL驱动库** **BSP**

**S**

**SIS**

**STM32F0** **STM32L1**  
**STM32F1** **STM32L0**  
**STM32F2** **STM32L4**  
**STM32F3** **STM32L5**





# 谢谢!

更多详情请访问

[www.st.com/stm32cube](http://www.st.com/stm32cube)

[www.stmcu.com.cn](http://www.stmcu.com.cn)

[www.stmcu.org](http://www.stmcu.org)