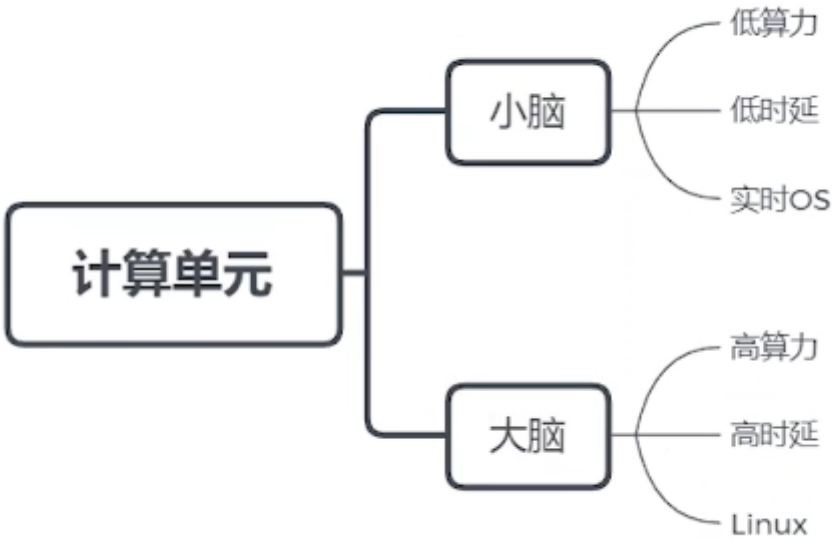
STM32入门

什么是单片机，单片机和电脑有什么不同



单片机 —— 机器人的小脑

```
算力低 低延迟 实时系统 控制各个模块单元
```

小电脑PC —— 机器人的大脑

```
算力高 高延迟 Linux 负责感知，思考，决策
```

STM32概述

1. 什么是STM32

1. ST——意法半导体，是一家半导体公司。可能大家经常听说ARM，ARM与STM32公司的关系可以简述为：STM32板子的内核由ARM设计，外设例如GPIO IIC FLASH等由ST公司生产。
2. M——Microelectronics的缩写，即微控制器。
3. 32——32bit的意思，表示这是一个32bit的微控制器。

2. STM32芯片分类

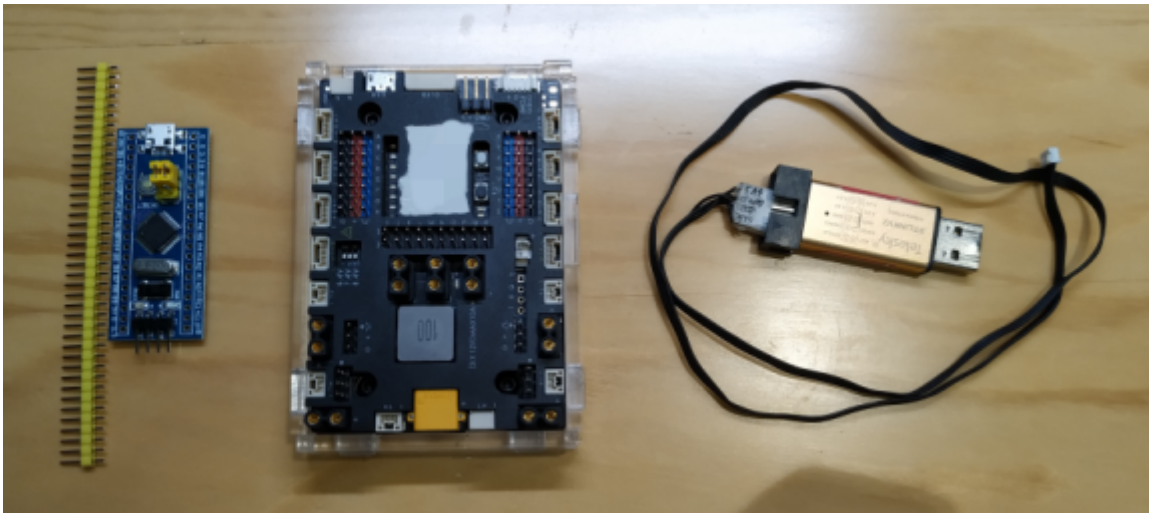
表格 4-2 STM8 和 STM32 分类

CPU 位数	内核	系列	描述
32	Cortex-M0	STM32-F0	入门级
		STM32-L0	低功耗
	Cortex-M3	**STM32-F1**	**基础型，主频 72M**
		STM32-F2	高性能
		STM32-L1	低功耗
	Cortex-M4	STM32-F3	混和信号
		STM32-F4	**高性能，主频 180M**
		STM32-L4	低功耗
	Cortex-M7	STM32-F7	高性能
8	超级版 6502	**STM8S**	**标准系列**
		STM8AF	标准系列的汽车应用
		STM8AL	低功耗的汽车应用
		STM8L	低功耗

3. STM32F103C8T6 && STM32F427IIH

命名规则：

1. STM32：ST公司开发的32位微控制器。
2. F：代表flash，ST公司只开发了F系列的产品。
3. 103：代表增强型系列。
4. C：代表48pin，R表示64，V代表100，Z表示144。
5. 8：代表主闪存存储器（flash）容量，8表示64k字节。
6. T：表示QFP封装方式，这个是最常用的封装。
7. 6：表示工作温度，在-40~85°C。



4. 一块开发板的组成部分

USB接口。
boot选项：选择板子的启动模式，boot0为0时则为竹山村存储器启动方式，即flash启动方式，一般情况下我们用的都是flash启动，所以正常情况下不要改boot。
复位按键
主控芯片
石英晶振
RTC晶振
源指示灯
LED灯
SWD调试接口
.....

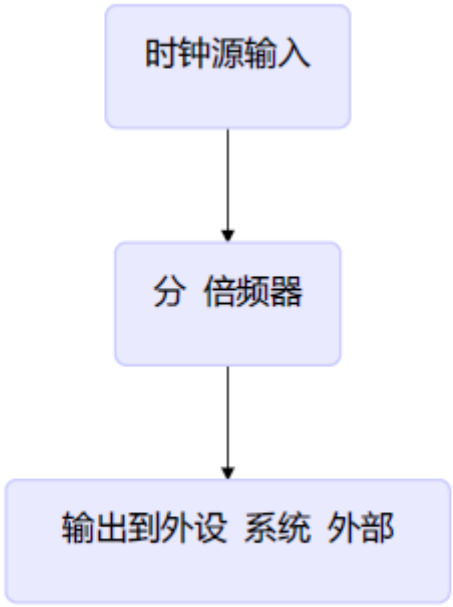
HAL库编程

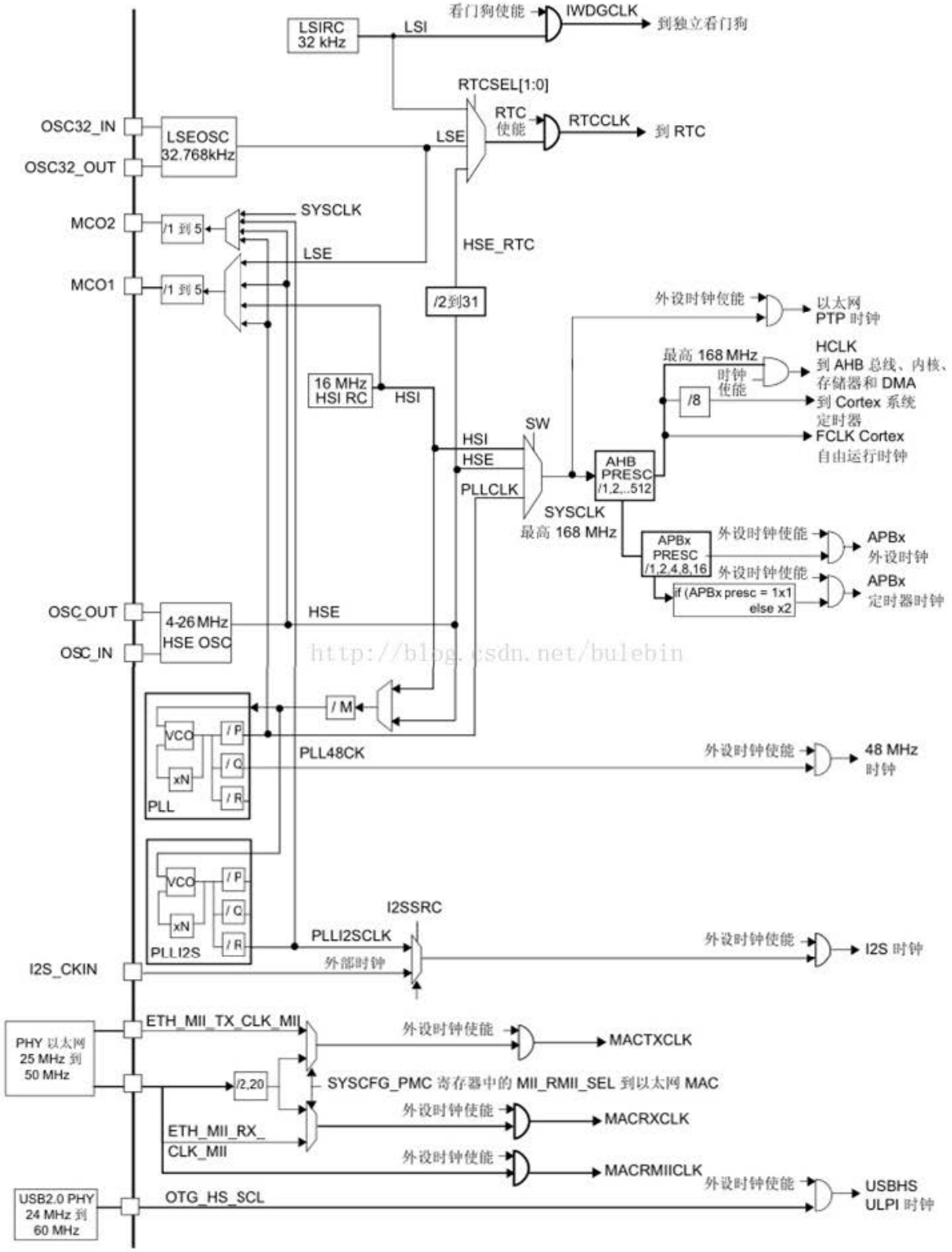
STM32编程主要有三种，分别为寄存器、标准库和HAL库。我们目前只涉及HAL库编程，另外两种方式有时间的话也
可以去研究。
寄存器开发可以通过直接操作寄存器来实现功能，STM32的寄存器非常多，如果要用寄存器开发方式，需要不断地翻
阅芯片的数据手册，会很麻烦。但它的优点是更底层，可以更容易理解本质。
标准库，挺好用的，但有个现实，**ST官方不再更新标准固件库。**
而HAL库，全称Hardware Abstraction Layer(硬件抽象层)是目前ST公司主推的开发方式。它比标准库更加抽象、简
洁。它往下接底层硬件，网上给用户提供API。HAL库的使用大大节省了用户开发程序所花的时间。（虽然写程序还
是很费时间）

```
/* Tips:
Hal库的函数都非常有特色 我们可以通过猜函数名+代码联想的方式来进行快速高效的编程开发
大部分函数命名方式为 HAL_<外设名>_<功能>
比如:*/
HAL_GPIO_WritePin        //在I/O口写入电平
HAL_UART_Receive_DMA     //以DMA形式接收串口消息
HAL_TIM_Base_Start_IT    //开启定时器中断
```

时钟树系统介绍

思考两个问题：
1.时钟信号从哪里来?
2.时钟怎么供给STM32系统工作：





STM32F4时钟树

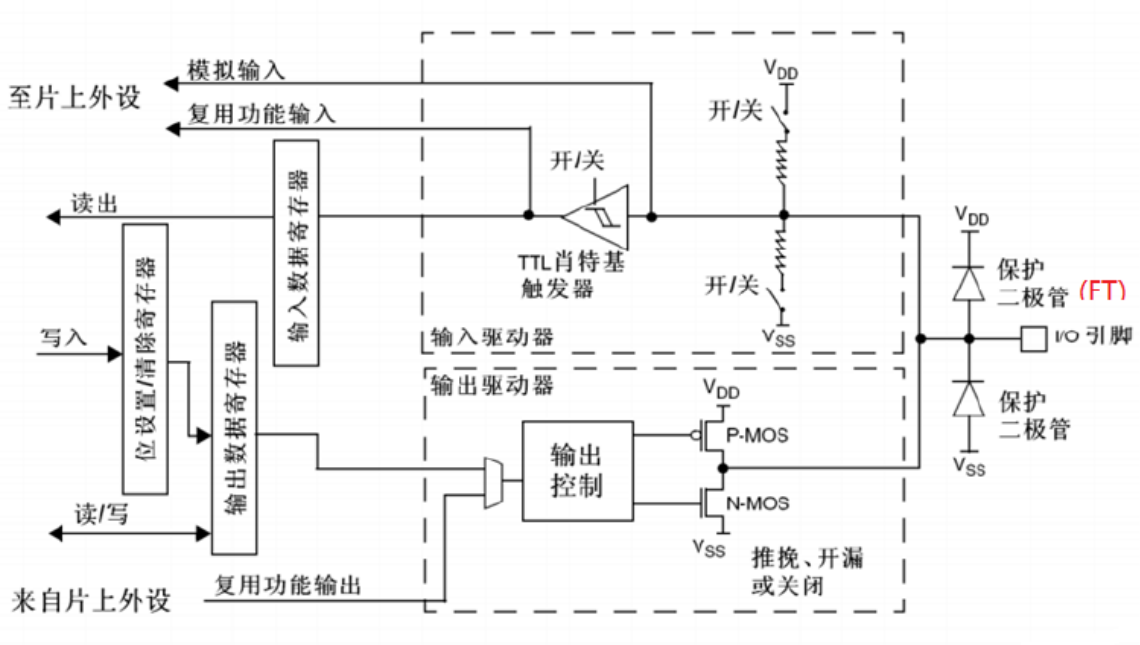
问题一、时钟信号从哪里来：时钟源。

> 分别为LSI、HSI、LSE、HSE。
> 其中HSE来自于外部晶振，频率为4~26MHz，精度高，一般就选用这个为时钟源。
> LSE来自于外部晶振，频率为32.768khz，一般用于RTC。
> HSI由内部RC振荡器产生，精度差。
> LSI位内部RC振荡器，用于独立看门狗时钟。

问题二、时钟怎么供给STM32的系统工作。

> 归纳为，看时钟信号流向，得到各个外设总线的时钟信号，使用外设时看外设挂载在哪一条总线上，使能这条总线时钟。

GPIO



GPIO支持4种输入模式（浮空输入、上拉输入、下拉输入、模拟输入）和4种输出模式（开漏输出、开漏复用输出、
推挽输出、推挽复用输出）。同时，GPIO还支持三种最大翻转速度（2MHz、10MHz、50MHz）。

1. GPIO_Mode_AIN 模拟输入
2. GPIO_Mode_IN_FLOATING 浮空输入
3. GPIO_Mode_IPD 下拉输入
4. GPIO_Mode_IPU 上拉输入
5. GPIO_Mode_Out_OD 开漏输出
6. GPIO_Mode_Out_PP 推挽输出
7. GPIO_Mode_AF_OD 复用开漏输出
8. GPIO_Mode_AF_PP 复用推挽输出

```
void HAL_GPIO_WritePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, GPIO_PinState
PinState)  //写入电平
void HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
//读取电平
void HAL_GPIO_TogglePin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
//翻转电平
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
//GPIO中断回调函数
```

同步/异步/阻塞/非阻塞/回调

https://www.cnblogs.com/IT-CPC/p/10898871.html

同步Sync & 异步 Async

同步和异步关注的是消息通信机制 (synchronous communication/ asynchronous communication)。
所谓同步，就是在发出一个调用时，在没有得到结果之前，该调用就不返回。但是一旦调用返回，就得到返回值了。
换句话说，就是由调用者主动等待这个调用的结果。
而异步则是相反，调用在发出之后，这个调用就直接返回了，所以没有返回结果。换句话说，当一个异步过程调用发出后，调用者不会立刻得到结果。而是在调用发出后，被调用者通过状态、通知来通知调用者，或通过回调函数处理这个调用。

阻塞 Blocking & 非阻塞 Non-blocking

阻塞和非阻塞关注的是程序在等待调用结果（消息，返回值）时的状态.
阻塞调用是指调用结果返回之前，当前线程会被挂起。调用线程只有在得到结果之后才会返回。
非阻塞调用指在不能立刻得到结果之前，该调用不会阻塞当前线程。

回调Callback

https://www.zhihu.com/question/19801131

回调函数(Callback)就是一个被作为参数传递的函数。 这个函数可能只是被预先定义了，但是没有内容，我们需要对其进行**重写（Rewrite）**，赋予这个函数特殊的功能。

在执行一项工作的某些时间点会调用对应回调函数，编写这些回调函数内容可以使得我们能在对应时间点做出实时反应，比如读取到编码器信息，串口传输完毕等时间点我们需要去做一些“工作”，这些读取到信息后在回调中进行的“工作”们统一称之为**解码(Decode)**

除此以外，类似于定时器中断这类回调函数中的编写的代码则更偏向于利用函数本身，而不再是传递的信息，我们会在定时器回调中进行稳定周期的控制函数编写。

STM32还有很多各式各样的回调函数，每个函数都有对应的功能，需要我们去探索发现它们的妙用。