## Bootloader

Mcu从地址0x00000000开始运行厂家特定程序，然后跳转到0x08000000处运行用户程序代码，于是可以在用户程序代码处做flash分区，划分boot和app两个层次，boot用于远程升级和跳转app层使用，app层用于整个程序功能的运行。

boot层：(16K+2K 该2K为统一管理变量的内存) 0x08000000 ~ 0x08004800

1. 看门狗、滴答定时器、串口初始化
2. 循环判断当前是否需要更新程序
3. 如不需要则跳转到第5步
4. 接收远程升级文件数据：

* 接收数据并写入FLASH
* 写入成功后判断在app的栈顶地址0x08004800是否存在0x20000000 ~ 0x20001FFFF（RW-DATA+ZI-DATA）的字节数据
* 进行中断失能(除NMI、HardFault、复位)
* 重新设置中断向量表为app的地址0x08004800

1. 跳转app层

app层：(256K-18K) 拥有自己的中断向量表，执行程序代码

## Scatter File

分散加载文件：ARM的配置文件，告诉链接器怎么生成HEX、AXF、BIN，将HEX放在FLASH哪个位置，也可以将FLASH中的代码拷贝到RAM上执行。内部分为加载域与执行域，若使用.sct文件则需要关闭keil中linker中的use memory layout from target dialog，在bootloader这一节中的2K内存就是由.sct决定的。

## UART

串口溢出中断：

原因：

1. 被高优先级打断没有即使处理，致使缓冲区满溢
2. 当前缓冲区中的数据没有及时取出又有新数据来了

处理：

在接收中断处理之前需要对溢出中断进行判断处理，依次读取SR、DR寄存器或者清除串口溢出中断并取出数据

## IIC

两根线：时钟线SCK、数据线SDA

信号：

1. 起始信号：SCK高电平，SDA自高向低的跳变
2. 终止信号：SCK高电平，SDA自低向高的跳变
3. 应答信号：发送完一帧8位数据后，在第9个脉冲释放数据线，接收器应答信号

若SDA为1，则为有效应答，表示成功收到数据

若SDA为0，则为非应答，表示未成功收到数据

## SPI

四根线：MOSI、MISO、SCK、CS

四种工作模式SCK(CPOL、CPHA)：

CPOL=0，CPHA=0，SCK空闲为低电平，在上升沿获取数据，下降沿发送数据

CPOL=0，CPHA=1，SCK空闲为低电平，在下降沿获取数据，上升沿发送数据

CPOL=1，CPHA=0，SCK空闲为高电平，在下降沿获取数据，上升沿发送数据

CPOL=1，CPHA=1，SCK空闲为高电平，在上升沿获取数据，下降沿发送数据