STM32基础知识

[TOC]

注意

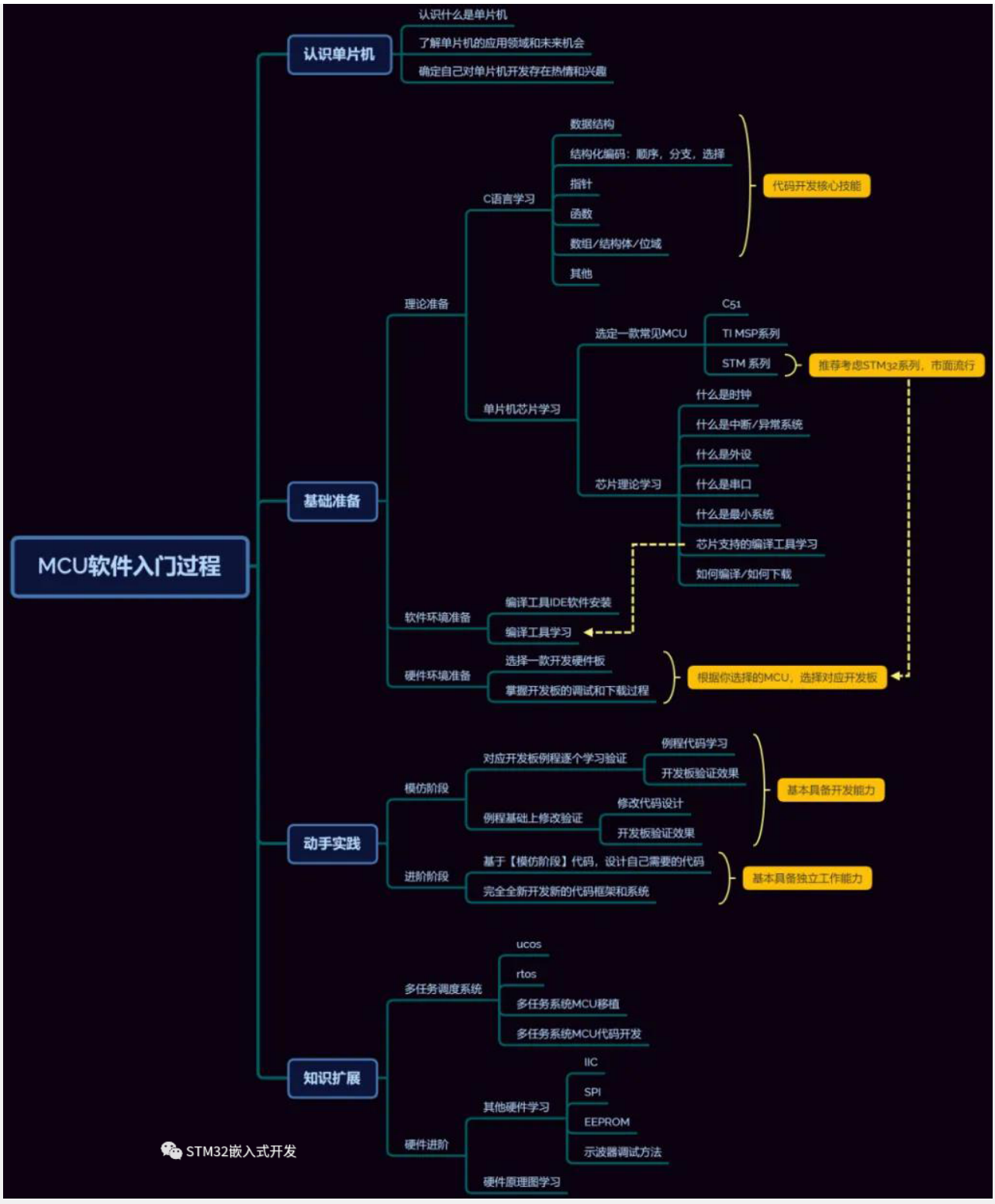
-

STM32学习路线与推荐文章

https://mp.weixin.qq.com/s/bUF9goso0xDaP9WvGf0xZA

单片机基础知识点

对STM32 MCU理论知识的学习和理解很重要，以下是入门一款单片机的学习路线仅供参考。



相关文章👉EEPROM和Flash这样讲，我早就懂了 👉单片机中的RAM vs ROM 👉STM32CubeMX教程-GPIO输入之外部中断 👉C语言操作寄存器的常见手法 👉学习STM32单片机，绕不开的串口 👉详解CAN总线 👉详解STM32的时钟系统，收藏了

STM32工作流程

专业术语并不会多难，记住基本不会忘，就算忘了回顾下也简单，在迈过这个坎后，下一步就是理解芯片执行流程和我们要做什么才能让芯片工作起来。

- 在上电后，芯片复位启动。

- MCU从FLASH通过指令/数据总线分别读取指令和数据，配合解析执行，其中在通过RAM和通用寄存器(也就是R0~R12的内部寄存器)处理可变数据。
- 根据执行的指令，配置和操作外设的的寄存器，从而驱动对应外设实现具体的功能。
- 配合上层组合逻辑，结合外设功能，执行具体应用的全部需求。

  上述就是芯片工作的流程，而作为单片机开发者，实现的代码就是放在FLASH中的程序，由官方提供的SDK包和用户裁剪实现定义的外设驱动组成，按照上述流程就包含:

- 芯片的启动过程中对堆栈的初始化
- 系统相关外设的处理(如时钟) => 这部分由厂商提供，我们把文件放入项目即可，以STM32为例就是启动文件startup_xxx.s和系统初始化的SystemInit函数
- 配置应用需要的外设模块，如果你看网上的教学视频，都是从这一步直接开始的，特别是STM32基于STM32CubeMx的实现，会告诉你这么写是满足要求的

如果你不是愿意深究的人倒也无所谓，当你深究这东西的时候，就会发现没有人告诉为什么要这么写，迷惑自然就来了，其实这部分就是嵌入式所谓的经验成分。

> 相关文章👉STM32代码的启动过程 👉详解STM32启动文件 👉STM32CubeMX介绍 👉STM32实例-用按键控制串口发送数据，文末附代码 👉详解STM32单片机的堆栈

如大部分外设模块都要进行时钟使能，GPIO就要配置引脚的上拉/下拉，推挽/开漏，USART配置模式，波特率，奇偶校验，停止位，数据位，还要考虑对应GPIO的端口复用，I2C就要考虑主从模式，根据需要的传输速率确定模块时钟，SDRAM除了考虑引脚复用，时序，还要计算刷新时间来保证数据的稳定，这些知识理论上都是长期在工作中积累出来的。

在项目需求确定后，根据芯片和器件选型，就能够确定硬件PCB的绘图和布置，功能需求也能确定，如访问外部EPPROM，那么I2C就确定为主模式，并根据EPPROM的读写时序和芯片的性能，考虑到效率和稳定的平衡，就可以确定I2C的速率，考虑布局和其它需求，I2C对应的复用引脚也能确定。

事实上，==对于真实项目，都是从需求=>项目计划书=>软硬件框架=>软件驱动==

上层应用逻辑实现

配合外设的应用逻辑实现其实才是项目中最复杂的部分，包含中断前后台系统的交互，可能存在RTOS应用和协议移植，GUI的管理操作，和其它模块如wifi模块，蓝牙模块的交互，不过这部分属于进阶内容，当你对外设模块有一定认知后，这部分应该也会有自己的学习方法，这里我就不在赘述。

> 推荐下相关参考文章👉一种STM32框架，实现RTOS的部分功能 👉STM32平衡小车，文末附代码、电路图等资源获取方式 👉使用STM32CubeMx工具，写FreeRTOS的demo程序

这只是我对单片机问题关于自己理解的总结，单片机虽然并不需要创新研发未知的技术，但却涉及从芯片底层到软件接口的全部实现，包含对现代计算机技术中向微处理器方向的全部精华技术，也是需要不断学习才能不被时代抛弃。

先从STM32理论知识入手，拿实例代码在实验板验证，来加深对外设、协议等理论知识的理解。过程可能很慢，但是基础知识肯定会打的很扎实，相信对后面开发会有很大的帮助。

STM32芯片的内部架构

STM32芯片主要由内核和片上外设组成，STM32F103采用的是Cortex-M3内核，内核由ARM公司设计。STM32的芯片生产厂商ST，负责在内核之外设计部件并生产整个芯片。这些内核之外的部件被称为核外外设或片上外设，如 GPIO、USART（串口）、I2C、SPI 等。

![图片](../../../知识汇总/0 - 个人学习/5 - Linux驱动学习/assets/640.webp)

芯片内部架构示意图

芯片内核与外设之间通过各种总线连接，其中驱动单元有 4 个，被动单元也有 4 个，具体如上图所示。可以把驱动单元理解成是内核部分，被动单元都理解成外设。

ICode 总线

ICode总线是专门用来取指令的，其中的I表示Instruction（指令），指令的意思。写好的程序编译之后都是一条条指令，存放在 FLASH中，内核通过ICode总线读取这些指令来执行程序。

DCode总线

DCode这条总线是用来取数的，其中的D表示Data（数据）。在写程序的时候，数据有常量和变量两种。常量就是固定不变的，用C语言中的const关键字修饰，放到内部FLASH当中。变量是可变的，不管是全局变量还是局部变量都放在内部的SRAM。

系统System总线

我们通常说的寄存器编程，即读写寄存器都是通过系统总线来完成的，系统总线主要是用来访问外设的寄存器。

DMA总线

DMA总线也主要是用来传输数据，这个数据可以是在某个外设的数据寄存器，可以在SRAM，可以在内部FLASH。

因为数据可以被Dcode总线，也可以被DMA总线访问，为了避免访问冲突，在取数的时候需要经过一个总线矩阵来仲裁，决定哪个总线在取数。

内部的闪存存储器Flash

内部的闪存存储器即FLASH，编写好的程序就放在这个地方。内核通过ICode总线来取里面的指令。

内部的SRAM

内部的SRAM，是通常所说的内存，程序中的变量、堆栈等的开销都是基于内部SRAM，内核通过DCode总线来访问它。

FSMC

FSMC的英文全称是Flexible static memory controller（灵活的静态的存储器控制器）。通过FSMC可以扩展内存，如外部的SRAM、NAND-FLASH和NORFLASH。但FSMC只能扩展静态的内存，不能是动态的内存，比如就不能用来扩展SDRAM。

AHB

从AHB总线延伸出来的两条APB2和APB1总线是最常见的总线，GPIO、串口、I2C、SPI 这些外设就挂载在这两条总线上。这个是学习STM32的重点，要学会对这些外设编程，去驱动外部的各种设备。