本文档知识临时性的，以下记录一些我的写程序的规范化、固定下来的东西，下次正式搞的时候加上，作为以后所有工程起步的模板！

* 新想法（补充：对于IAP，搞完外设后最后弄）：对于IAP程序，需要两部检验，第一步需要加密通讯（非对称加密，IAP程序内部定义一个密钥，用于解密密文，密文的内容就是一段特定的我staok的名字和邮箱再加上用特定的我的算法生成的一串随机数），如果成功才能开始下固件，一边下一边MD5（如果MD5慢或者麻烦科研STM32硬件CRC32）校验程序是否正确（是否有被改），如果再次成功，则才开始执行固件
* ~~外设驱动库以后全部用HAL库去写，F1-F4全支持，省心，建工程时直须参考原子手册的建工程部分和历程里的模板即可。基础外设可以先看例程，移植，看不明白的地方再去看手册，基本外设写完后，再查STM32\_HAL库用户手册做API补充，这个手册有初始化流程说明，对比例程看是否缺东西，最后再看原子手册里面的API补充。~~最终支持的有：LittlevGL（可以学这个先！），~~STemWin~~、USB、FATFS、~~freeRTOS~~、LWIP，用预编译控制！这些库如GUI和网络和文件系统等等，再单独弄一个配置文件！F4的DSP（包括FFT和滤波等等）
* 最后看看码云和github上面我的star项目
* 早晚，再把硬石和安富莱（都已经下载好，只需看硬盘里的）的例程都过一遍，有很多不一样的有价值的东西，除了例程，前者还有电机部分，后者还有DSP部分！
* 再添加的高级组件：~~一阶滤波（API写在sys.h里提供使用）滤波~~，线性回归（一阶次和任意次），各种加密算法，音频编解码，modbus，MQTT（里面可能包含cjson），JPEG GIF BMP解析+BMP编码，这些外加库，放在与上面GUI、LWIP、FATFS库所在的同一个文件（名叫高级组件.c）里统一管理是否启用，这些外添库源程序单独建一个文件夹放
* ~~基础模板建好后，复制一个，在此基础上添加原子的例程代码，即所有的外设驱动代码，均写明所有API和其参数。~~
* **借鉴程序风格和框架的有：麒麟座、X-Bot等，本地化“等着移植的”里面的所有程序，~~本地化我以前做过的所有程序~~**
* 做两个模板，~~一个是裸机的，一个基于FreeRTOS的~~，F429和F103各一个
* ~~STC15的裸机和RTOS模板各一个，51的RTOS模板先尝试移植FreeRTOS！RTXtiny算了，以后再要学RTT，就学太多了。~~

~~更新：单片机模板不用分裸机和操作系统，写一个模板，但是用于编译控制用裸机还是用操作系统。~~

* STM8、~~ESP8266~~、~~328pArduuino~~、NRF5x、~~K210~~等做到API库可查可用即可。
* 关于程序规范：~~不自己写库，用HAL库！！~~

1. 所有工程模板，留一个文件做一个 引脚和外设使用的MAP：

/\*用于控制\*/

#define w25q32\_CLK PBout(13)

以此类推

~~RCC\_OscInitTypeDef RCC\_OscInitStructure;~~

~~RCC\_OscInitStructure.OscillatorType=RCC\_OSCILLATORTYPE\_HSE;   
RCC\_OscInitStructure.HSEState=RCC\_HSE\_ON;   
RCC\_OscInitStructure.HSEPredivValue=RCC\_HSE\_PREDIV\_DIV1;~~

~~…~~

~~改为：~~

~~RCC\_OscInitTypeDef RCC\_OscInitStructure = {~~

~~.OscillatorType=RCC\_OSCILLATORTYPE\_HSE;   
 .HSEState=RCC\_HSE\_ON;~~

~~.HSEPredivValue=RCC\_HSE\_PREDIV\_DIV1;~~

~~}；~~

~~其他结构体初始化以此类推。~~

1. ~~HAL库的delay函数不要用，原子的delay挺好的移植了，原子的sys也移植了，构建一个最小实现的模板。~~
2. ~~写一个：~~

~~print(uart1,”out:\t%d.2”,i);~~

~~具体实现：sprintf合成字符串，根据选择的串口，用字符串发送函数往对应串口发送，形参写法或许可以参考：~~

~~或者：先用sprintf合成字符串，然后发送字符串两条函数，最好是想办法合成一个函数API。~~

1. ~~所有外设提前写好配置函数，只是不调用，按需要调用初始化！，其他中断服务函数等但要写好。（GPIO的配置、读写和寄存器的改变入出模式（毕设里面的）罗列~~、定时器（在103RB系列还差TIM1有待补充，在F429里面补充TIM1\8和TIM567这几个定时器）、~~五个串口~~、~~一个MCO PA8，看门狗~~、~~SPI~~、~~软件I2C~~，DMA，支持FSMC驱动LCD、RAM和FLASH，~~AD~~ DA，~~RTC~~等待），~~如，串口1的初始化函数，里面包括时钟开启、串口配置、中断配置和引脚配置，再加上中断函数标明，DMA配置，使能失能的函数和优先级设置等API罗列等，全面一些，一劳永逸。串口改编：串口详细看原子F1 HAL文档的187页开始。~~

可选参数在每个函数的上方用注释标清楚，形如：

/\*\_\_\_\_\_\_\_\_\_\_函数名\_\_\_\_\_\_\_\_\_\_\_\_\_\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*描述：

\*参数： 1、

2、

\*返回值： 1、

2、

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. ~~规则：等待发送结束等while死循环里面加上超时退出，超时退出后重发，重发超过五次则触发错误标志，让自检程序处理。~~
2. ~~规则：所有的外部按键等输入均用中断！~~
3. ~~规则：尽量不使用引脚重映射。~~
4. ~~模板：以定时器为例，类似HAL\_TIM\_Base\_MspInit和HAL\_TIM\_PeriodElapsedCallback等需要用户根据实际用了哪些定时器写的函数，我的模板里把所有定时器全部写上，留空等用。具体原子219页开始。~~
5. ~~单独一个文件：外设信息都注明，如~~高级定时器1、8~~（定时器中断，高级PWM（7路，具体在xxx引脚，重映射在xxx引脚，形式如HAL F1手册的225页的表）），通用定时器2-5（2、5为32位，定时器中断、普通PWM（4路，具体在xxx引脚，重映射在xxx引脚）、输入捕获等），普通定时器6、7（定时器中断，无PWM等），配置函数的所有参数和其意思。~~
6. ~~获取函数运行时间函数，get\_runtime((voi\*)founction){获取当前定时器8的值，运行函数，获取当前定时器8的值，返回差值}，超时返回错误，定时器8作为本功能的基准定时器，0.1ms周期，如果是16位定时器，则最大为65536x0.1ms时间，因为频率太快不宜做定时器中断。~~
7. ~~一些暂时难理解的外设先搁置：FSMC、CAN，对于DMA看多方资料~~
8. IAP可以作为一个单独的最小实现工程，注释写好要使用需要怎么设置MDK，和如何使用，可选的更新来源：USB（通过虚拟串口）、串口（可选串口2、3等，不用串口1）、SDIO控制的SD卡和SPI控制的无线模块。注：能不能用USB模拟读卡器把单片机的FLASH作为U盘显示在电脑上，然后网这个U盘复制bin程序文件，然后用FATFS取得该文件的地址然后控制运行。
9. ~~sprintf不适用标准库的，太占flash，用纯c写的那个，找找。~~
10. **~~可选的~~**~~基本三方组件（包含在模板中，通过是否编译）：RTOS（~~**~~通过是否使用freeRTOS的标志位来编译工程！在main里面通过标志位选择编译两套初始化写法~~**~~）、StemWin，LWIP，FATFS，USB从机（虚拟串口和读卡器）等等，另外包括各种协议（加密、校验、MQTT等），看看~~**~~麒麟座~~**~~都有啥全弄。这些库如GUI和网络和文件系统等等，再单独弄一个配置文件！~~
11. 基本外设处理完之后，加入常用IC支持，如：LCD、OLED、MEMS、595和165、DS18B20、DHT11、RTC、ADC、DAC等等，看NodeMCU固件支持啥芯片都。
12. 原子手册，HAL的F294比F1多了这些内容：**LTDC LCD**、硬件随机数、DAC、I2C还是软件模拟的没变、IO扩展IC、MPU9250（与6050区别不大，看价格选择，但是我的模板要支持）、摄像头章节的DCMI接口、**NAND FLASH章节**、相机章节新增JPEG编码、音乐播放使用**SAI接口**（录音章节同）、视频播放章节、FPU测试、DSP测试、USB章节全看、F429自带**以太网模块**。
13. F407比F1多的内容：硬件随机数、摄像头章节的DCMI接口、DAC、I2C还是软件模拟的没变、相机章节新增JPEG编码、音乐播放使用I2S接口（录音章节同）、视频播放章节、FPU测试、DSP测试、USB章节全看、F429自带以太网模块。

以下为有待添加的模板内容：

* 裸机：

~~开机外设初始化、开机自检、~~

~~定时器轮询框架、~~

~~循环自检程序框架、~~

~~蜂鸣器和按键(包括长按和双击检测，和等待按键函数：wait\_for\_key(key1,push或者longPush或者doublepPush))、（蜂鸣器看密码锁里面那个，按键：结构看菜单模板里面的，按住有效或者单次有效、长按和双击等检测先看看原子的实现，试试状态机检测，再好好写一个）~~

~~看门狗+FLASH存储开机次数(开机自检加上：可选如果复位了(通过检查开机次数判断)，卡在开机自检等待操作，也可选不必)、~~

~~串口，串口中断只接收信息，用一个函数组合内容：接收可选两种，一种环形FIFO实现，一种在接收中断实现定时接收，即从接收到第一个字节开始计时，然后往后每一个字节都串起来成字符串，直到过了超时时间没有接收时或者连续接收超过200字节时算作一次接收，结果是得到一串字符串，然后在处理函数中进一步解析看看是不是某种协议。~~

实现一个可选择的协议打包函数如：结尾自动加\t\n，modbus等等，形成自己的规范，可以通过串口发送、SPI发送等函数调用把数据打包后再发送、

~~以下两条复杂一些：~~

~~通过预编译，所有算法可以选择是否编译(如菜单模板、最小二乘、PID、加密、FPU、GUI等等)~~

~~通过预编译，所有外设写好初始化代码和摆放常用API，选择性编译~~

* ~~RTOS：仿照上面裸机的框架，写成RTOS的框架~~

~~开机外设初始化、开机自检、~~

~~创建开始任务，在开始任务里创建完成功能的任务、~~

~~按键、ADC和串口通过不同的队列消息传递数据到有关任务~~

~~串口中断只接收信息，用一个任务组合串口内容：~~

~~串口队列：队列长度（有几个队列项目）5个吧，队列项目大小50字节吧（即一条字符串），接收程序按照上面裸机的接收，单次接收形成一个字符串，作为一个队列项目放到串口队列中，供有关任务处理，串口字符串数据太大，可以以指针形式发送，char\*的结尾加上字符串结尾符号\0，用覆盖的方式入队，用取完清除的方式出队。~~

~~循环自检任务，用信号量把此任务与定时器中断同步。~~

~~其它内容与上面裸机一致。~~