This is designing for my smart control system

I will try my best to complete it

When i finished it ,i will leave byd and find a new job

And i hope i can find some one that time

Be brave and confident

2020-09-12

2020-09-15

串口调试助手不能接收到数据。

1.串口调试工具的问题，傻逼工具打开后不能关闭，导致串口占用，使得不能接收数据。

2.开发板flash下载之后，需要重新上电再次打开串口才能接收到数据。

问题遗留：

1.按理说，使用ST\_LINK下载程序之后，不需要重新上电程序也能正常跑起来，但现在似乎不符合常理。

2.代码设置为先printf("hhhh"),之后while(1)间隔半秒输出hhhhi++,上电之后，按理说,程序已经开始跑起来了，在点击调试助手打开串口通道的过程中，log应该进入i++阶段的输出了，但每次都是打开串口通道，才开始输出 hhhh\hhhh0\hhhh1...

3.这里使用的是串口中断输出，每次收发是否都要触发中断，对于系统的影响如何，除此之外，是否可以采用DMA等其他方式输出串口log.

参考资料：

https://blog.csdn.net/heda3/article/details/80602287

version:

20200915\_001

2020-09-15

上传github:

1.添加ssh,创建github代码仓

2.git remote add origin git@github.com:michaelliao/learngit.git

3.git branch -M master

4.git push -u origin master(第一次上传）

5.git push origin master(修改上传）

参考资料：

https://www.liaoxuefeng.com/wiki/896043488029600/898732864121440

version:

20200915\_002

2020-09-21

添加CAN通信，可以通过周立功USBCAN\_II接收到CAN数据。

待解决问题：

1.stm32的系统时钟配置问题,野火和正点使用的时钟频率不相同，需进一步学习。

2.can\_test接收到的数据与系统发送的数据不相同，按照程序设定，应该是间隔一秒发送一次Id为1314的扩展帧，但接收到的数据毫无规律可言，而且接收到的数据极快，并没有一秒发送一次的效果。

version:

20200921\_003

2020-10-03

添加了EEPROM的读取代码。

待解决问题：

1.正点原子使用的是软件GPIO模拟IIC，而野火使用的是硬件IIC，不能直接将其代码修改移植，同时应当分析一下二者之间各自的优势与缺点。

2.正点原子还采用了一种FLASH模拟EEPROM的方式，应分析一下二者之间各自的特点以及运用场景。

version:

20201003\_004

2020-10-07

解决CAN报文发送失败的问题。

1.在正点原子的代码上移植了野火的代码，在对CAN管脚初始化的时候没有使能管脚时钟--CAN\_TX\_GPIO\_CLK\_ENABLE()，而是沿用了--\_\_HAL\_RCC\_CAN1\_CLK\_ENABLE()，可能导致管脚没有初始化无用。

2.系统时钟的设置，正点使用的系统时钟为180MHz，而野火注释标注的是216MHz，实际使用的依然还是180MHz，在对波特率的计算的时候造成了偏差，CAN时钟APB1等于系统时钟除以APB1分频系数，野火CAN时钟应为45Mhz，而不是标注的54MHz。

3.将上述问题修正之后，进行CAN报文发送，串口数据显示，前三帧数据发送正常，从第四帧开始提示发送错误，修正Can\_Handle.Init.NART=ENABLE，使能禁止报文自动重传功能，应该是发送的报文每一帧都是一样的，导致CAN的三个mailbox一致，再次发

送相同的报文时禁止自动重传，导致发送失败。

CAN时钟以及波特率的计算：

bitrate = APB1 / (1 + BS1 + BS2) / Prescaler;

APB1 = SYSCLK / APB1CLKDivider;

SYSCLK = PLLCLK = 外部时钟 / PLLM \* PLLN / PLLP = 25 / 25 \* 360 / 2 = 180MHz;

version:

20201007\_005

2020-10-07

解决CAN报文接收失败、系统卡死的问题,替换掉正点老旧sdk。

1.Can\_Handle.Init.NART=ENABLE 与报文发送失败无关，将其修改为DISABLE之后发送依然成功，还需继续学习相关寄存器的使用。

2.系统卡死的原因是 can接收中断没有配置正确。CAN\_RX\_IRQ 在 stm32f4xx\_it.c中应该有定义 void CAN\_RX\_IRQHandler(void)，正点原子的代码比较老旧，并没有找到相应的内容，将其替换为野火的代码后测试可以通过。

3.在参考手册之中有这么一段话：要注意如果设置了接收报文中断，必须要在中断内调用HAL\_CAN\_Receive\_IT 函数读取接收FIFO 的内容，因为只有这样才能清除该FIFO 的接收中断标志，如果不在中断内调用它清除标志的话，一旦接收到报文，STM32 会不

断进入中断服务函数，导致程序卡死。

待解决问题：

1.CAN通信中各种寄存器的使用。

2.形如 stm32f4xx\_it.c 之类的文件的使用。

version:

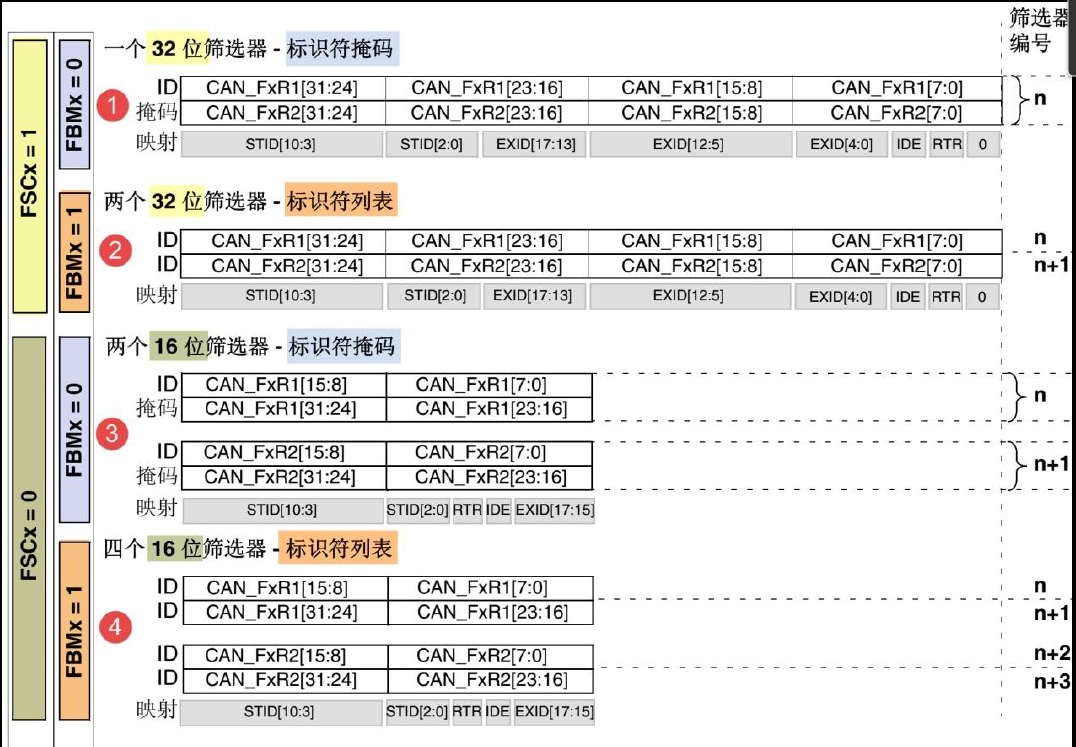
20201007-006

2020-10-08

解决520报文接收失败的问题。

1.原代码中接收的是1314的扩展帧，需要将其修改为接收520的标准数据帧，对CAN筛选器进行配置，将CAN\_ID\_EXT 修改为CAN\_ID\_STD，0x1314<<3修改为0x520<<3，并在接收中断函数中将pRxMsg->Id、IDE修改为标准数据帧，验证不通过。

2.查阅资料如下：



StdId的起始位置为第22位，ExtId的起始位置是第4位，因此需要左移的位数应该是21位，测试通过。

待解决问题：

1.16位筛选器模式的使用。

2.标识符列表的使用。

version:

20201008-007