**USB数据采集卡设计**

# 一、工程介绍

这个工程主要是用STM32内置的ADC等进行原理性的尝试。具体做法是利用外部按键触发定时器2，定时器2主要是用于防止按键抖动的，定时器2计时15ms后进行中断，开启定时器1，定时器1外部触发ADC采样，采样结果经过DMA传输到一个Buffer，然后利用USB将Buffer中的数据发送出去。

这个工程没有别的意义，只是线熟悉ADC、DMA、定时器等外设，同时进行初步的原理验证，为后续外接高速ADC，高速数据采集打下基础。

工程能使用LL库的尽量使用了LL库，只有USB HS使用HAL库。

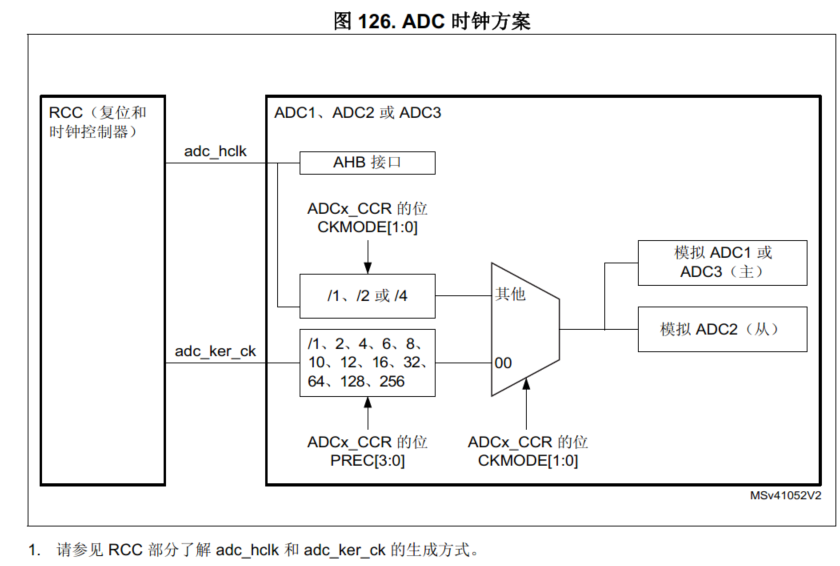
# 二、STM32CUBEMX配置

## 1，ADC部分配置

1. 通道配置：ADC可以选择单端采样和差分采样。通道由快速和慢速之分，采样时间有一点限制，应当参照数据手册中ADC特性规定的最短采样时间要求。
2. EXTI Conversion Trigger：好像是由外部信号触发ADC采样。

3，Mode: 好像ADC1只有Independent模式，不管了。

4，Clock Prescaler: 时钟预分频。ADC时钟有两个方案，一种是使用HCLK，一种是使用



Adc\_ker\_ck，这个是可以通过时钟数配置的，不过不管是哪个始终，都可以配置输入时钟的分频系数，这个值就是Clock Prescaler。不过这两个时钟有区别，Adc\_hclk的优点是无论选择哪种AHB时钟方案，都可以达到ADC的最大时钟频率，而adc\_ker\_clk的优点是绕过了时钟域重新同步，*如果ADC由定时器触发，并且要求ADC精确触发，可以时钟此选项，重新同步两个时钟域会为触发时刻带了不确定性。*我这里使用的是第一种方案，后续可以改成第二种进行实验。

5，Resolution：分辨率，这个没什么好说的，但是要注意ADC的采样频率和采样分辨率也有关。

6，Scan Conversion Mode: 这个是扫描转换模式，如果有多个通道，开启扫描模式会把所有的通道都转换一遍。

7，Continuous Conversion Mode: 这个是连续转换模式，转换完成之后会继续转换。

8，Discontinuous Conversion Mode: 这个是不连续转换模式，不管了

9，End Of Conversion Selection: 这个是转换完成选择，可以选择转换一次后完成，也可以选择转换一个通道序列后完成

10，Overrun behavior: 数据溢出操作，可以保留，可以覆盖

11，Left Bit Shift：数据怎么存放，左移几位。

12，Conversion Data Management Mode: 可以用DMA循环传送，假设我要传512个数据，DMA循环模式就是传完512个数据之后继续重新开始传512个，而DMA One Shot就是传512个数据之后就结束，也可以不用DMA直接存在DR寄存器里。

13，Low Power Auto Wait: 低功耗相关吧，不太了解。

14，Enable Regular Conversion： 开启常规转换，转换模式有常规转换和注入转换，具体可以百度，这个很多资料。

15，Enable Regular Oversampling：过采样模式，这个好像是为了平滑数据采用的，具体参照手册

16，Number Of Conversion：转换通道的数量

17，External Trigger Conversion Source，ADC外部触发源，我这里是定时器1更新信号触发，还有比较触发等选项，具体可以参照《RM0443》表192。

18，External Triger Conversion Edge，外部触发边沿选择，没什么好说的

Rank:

19，Channel 这个是ADC的通道，一般自动配置好了

20，Sampling Time: 采样时间，一般是多少多少周期

21，Offset Number，主要是配置ADC数据的保存形式，可以参照手册。

22，Offset Signed Saturation：这个不知道是啥，一般Disable就行了

23，Analog Watchdog xxx 全部失能，不知道有啥用。

24，中断就使能DMA1 Stream0，不用使能ADC，因为ADC转换完成后，数据就自动被DMA传输走。

2，TIM1配置

1. Slave Mode：一般使能的时候是为了配合外部信号，比如计数外部按键，配合主定时器这种情况时，要使能，其它的情况一般都是主模式。
2. Trigger Source，触发源，但是这里作为主定时器使用，不用触发源
3. Clock Source，一般都是Internal Clock，其它时钟还没有研究。
4. Channel 123456，以及其他，没有研究，全部Disable。
5. Rrescaler，这个就是时钟预分频，假设要一百分频，需要将其设为99。
6. Counter Mode 可以往上计数，往下计数，中间计数，具体可百度
7. Internal Clock Division 不知道是干啥的，一般选择No Division
8. Repetition Counter，不知道干啥的，也没用
9. Master/Slave Mode，选Disable(Trigger input effect not delayed
10. Trigger Event Selection TRGO，就是外部触发1，这个选择Update Event，其它还有Reset，比较等触发源，这个选择Update是因为计数器计数满一轮之后会溢出，这时候就会产生更新时间，正好是周期的，于是选择更新时间作为ADC的输入触发源。

11，Trigger Event Selection TRGO2，这个触发源2没有用的，随便设置一个就行。

12，NVIC Setting里没有任何中断选中，因为不需要。

3，TIM2配置

TIM2主要是用于解决按键抖动问题的，就是按键按下后会立刻进入EXTI中断，然后在EXTI中断中开启TIM2，计时15ms后进入TIM2中断，在TIM2中断中开启采集功能。TIM2就不往上放了，主要区别设置了Prescaler，因为15ms还是挺长的。然后开启了TIM2中断罢了

4，DMA配置

开始配置ADC的时候，DMA选项是灰色的，因为需要手动添加一个DMA后才能进行配置。

1. DMA Request： ADC1 就是DMA请求发起的对象
2. DMA Stream，一个DMA有不同的Stream，这个随便选一个就好了。
3. Direction，可以选择Peripheral to Memory，也可以Memory to Peripheral，也可以Memory to Memory，这里是把ADC数据传到一个Buffer中，所以选择外设到存储
4. Priority，因为总线资源有限，如果有多个DMA的话，就需要利用优先级来判断是哪个DMA请求总线，就要用到优先级。
5. Mode选择Normal，Normal就是传输一次完成之后就停止，而Circular传输任务完成后会重新开始，比如传输512个数据，Circular传输512个数据后会重头开始，而Normal模式就会直接停止
6. Data Width，因为我配置的ADC是16bit分辨率，这里我就设为了Half Word，16bit
7. 其它的都没用的，不在解释
8. DMA选项里面没有中断配置，需要在NVIC选项卡之中进行配置。

5，CORTEX\_M7

在这个选项卡里面我把I-Cache打开了，把D-Cache关掉了。Cache本意是用来加快指令和数据的读取，但是DMA和D-Cache一起打开的时候容易造成数据不一致的问题。具体可以参考文章[Cache和DMA一致性 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/109919756)

6，USB HS

我这里用的USB HS，速度能够达到480MBps，但是STM32 USB HS没有内置PHY（USB物理接口芯片），因此选用微雪的USB3300 PHY模块。然后把Internal DMA IP关掉，其它正常配置即可。选用Class为Communication Device Class Virtual Port Com，其它默认。微雪的USB3300我还设计了一块PCB板，因为杜邦线已经不太行了。

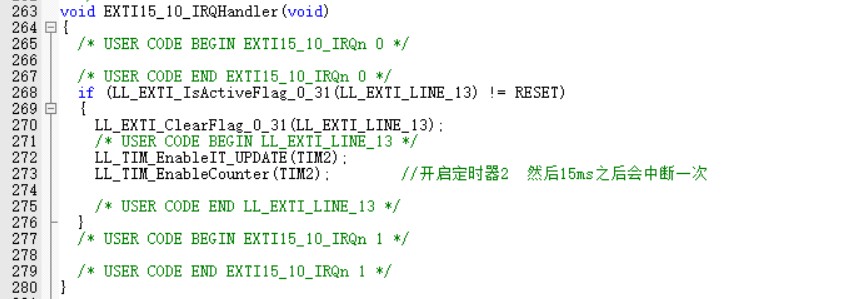
7，其它

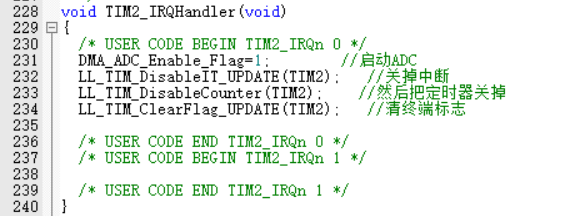
优先级，GPIO，EXTI这些都非常简单，时钟数也是非常的清晰明了，这里主要提一下定时器TIM1和TIM2的时钟频率。定时器主要挂在APB1和APB2总线上，具体可以参照STM32H7B3xQ HAL User Manual的Bus模块。比如这里TIM1挂在APB2总线上，TIM2挂在APB1总线上。

# 三、工程调试过程

## 1，定时器2老是自己进中断

答：不过这个定时器2老是自己进中断，我还没有使能它，他就自己进中断，很烦。一看原来是UIF自动置位了，难道是我的UIE开的太早了？应该是找到原因了，我对定时器2进行一丁点配置，更新标志就置位了，即使定时器2是关着的也是如此。看样子只能一般关中断一遍清除标志了。然后在按键中断里开定时器2和中断，然后在定时器2中断里把定时器2中断失能掉就应该没问题。现在解决了，但是没有完全解决，按一下按键之后，定时器2会中断两次，检查了一下是清标志语句的位置的问题，最终是这样处理的





## 2，定时器无法触发ADC转换

原来尽管是设置外部硬件触发ADC，仍然要调用LL\_ADC\_REG\_StartConversion函数，这个函数调用之后，ADC会等待外部触发到来，触发到来之后才会进行AD转换。

## 3，DMA老是自己中断

原来的原因可能是我不知道何时把DMA设置成循环模式了，应当改一下。而且每次我还没有按按键他就会自动进入中断，原来是我把DMA\_ADC\_Enable\_Flag手动设为1了，还是在KEIL调试中利用指定值断点调试出来的。

但是解决这个之后，程序刚开始还是会进入DMA中断，我要检查一下DMA中断配置。原来是因为ADC自己启动了，然后DMA跟着就传输，导致了中断。ADC为什么启动呢？因为我把启动ADC的函数放在main函数中了，damn it！浪费好多时间。这么明显的函数调用都没有发现。

## 4，DMA无法进入中断了

经过调整之后，一切都很正常，就是ADC数据无法DMA传输了，需要再次进行检查。，不知怎么回事进不去DMA中断了。发现有中断信号，就是进不去，而且DMA的PAR寄存器里的外设地址为0x10000000，很烦。要检查外设地址配置，原来我设置的地址是ADC->DR，这样是不对的，这样读到的数据是ADC寄存器里的DR的数据，而不是DR的地址，应当改成&ADC->DR。修改完外设的地址后，就可以进入中断了。然后多次进入定时器2中断的方法也解决了，不仅要清楚中断标志，最好把定时器直接在中断里关掉。

## 5，ADC Buffer的数据变一次就再也不变了

每次传输ADC\_Buffer的数据不变问题解决了，原来真的要重新对DMA传输数据量重新赋值，赋值之后再使能就没问题了。我被手册骗了。

## 6，Buffer数据总有后几位的数据是0

关于数组后面全是0的问题，我把ADC Buffer数组长度设置成16，就只有前两个位有数据，其它全是0，设为17则全部有数据了。设成32，只有前18个位有数据。设置成33时，全都有数据了。而且验证了数据是正确的。

现在最主要的问题就是解决为什么我所设置的传输数据量的值影响到我DMA实际传输的值了。我要仔细找找线索。最后发现了每次ADC传输停的位置都是16的倍数，于是我想着是不是目标地址也会造成影响，于是我把ADC\_Buffer的地址从0x240001DC改为了0x24000200上，这样一下子就可以了，我也不明白到底是哪个地方出了问题。好，目前来说把传输数量设置成512个数据也没有任何问题。那么511个呢？511个也没有问题。初步推测问题出在内部SRAM保护或者是FLASH保护上了，因为超过了一个页，后续求证。**最后发现了，原来是DCache的问题，需要开透写功能，或者直接把D-Cache关掉。**

## 7，不用USB发送Buffer数据，Buffer正常更新，用USB发送Buffer数据后，Buffer就无法更新。

上个问题解决完了，又出现了ADC\_Buffer的数据不变的问题了。但是我确实重新传输数据量进行赋值了，还有其它原因？明天再说吧。

找到原因了，就是因为我把USB发送的语句去掉了，把USB数据线拔下来了，就可以了，我在插上试一下。

1. 先插USB线，不进行数据传输，进行实验。实验完毕，数据正常。
2. 插上USB线，进行数据传输，进行实验。实验完毕，仅第一次能成功。看样子问题出在USB配置和DMA上，最有可能的是中断优先级出了问题，我调整了之后也没有解决。
3. 然后我插上USB线，USB传输别的东西而不是数据，数据正常。应当是数据拷贝到USB\_Buffer时出现了问题。

终于终于解决了，经过多次实验我发现，无论我拷贝没拷贝数据，只要我访问了ADC\_Buffer，那么DMA就不会再次更新，很显然是内存保护等相关的东西有点问题。然后我看到了一篇帖子。[STM32 DMA经常遇到的几个问题 (stmicroelectronics.cn)](https://shequ.stmicroelectronics.cn/thread-622821-1-1.html)

上面写到M7内核带cache，使用DMA传输时有时会遇到DMA访问到的数据不是实时的正确数据，因为这往往可能是因为DMA要访问的内存区域和CPU是共享的，同时又开启了相关区域的D-Cache属性，即CPU访问该内存区域数据时使用D-Cache，将内存数据拷贝到D-Cache。这是重点来了，我把I-Cache和D-Cache全部关掉，这样数据就全部能实时更新了。到此这个工作就基本完成了。[MMU、Icache、Dcache - 静之深 - 博客园 (cnblogs.com)](https://www.cnblogs.com/jingzhishen/p/4214917.html)，这篇文章也说，单核处理器使用DCache只会存在一个问题，即与DMA会数据不一致的问题。

## 8，调试时发送数据正常，直接运行时发送数据后面有好多0

又遇到了一个问题，实际程序运行时，DMA传输的数据后面会有好多0，调试时却没有问题，肯定是因为延时时间比ADC采样时间短，导致1ms后循环直接把DMA\_ADC关掉了，修改一下，把DMA\_TIM\_ADC\_Disable中的失能DMA关掉之后，一切正常。就是逻辑的问题，修改一下就可以了。

**9，数据只能传输4次，无论怎么按按钮，都无法开启第五次数据采集传输。**

不过又遇到了数据只能传输4次的问题，不过这个问题我很清楚，肯定是逻辑的问题，稍微修改一下就好了，不过我目前的解决方法就是把DMA\_TIM\_ADC\_Disbable放在DMA中断里，虽然用不了多少时间，可还是有点瑕疵。

**10，总结**

后面开发程序一定要先把I-Cache和D-Cache关掉后开发，功能开发完成后在开启I-Cache和D-Cache，这样好排查问题。

# 四、STM32CUBEMX生成代码调整

首先需要说明，CUBEMX生成的代码仅仅是配置外设，中断，使能，启动和其它的一些CUBEMX中没办法配置的东西都需要自己手动配置

## 1，ADC部分

1. 设置了ADC的数据传输模式为LL\_ADC\_REG\_DMA\_TRANSFER\_LIMITED
2. 进行了校正，注意校正时一定不要开启ADC，较正时我同时进行了偏移和线性度校正，总共需要16384个时钟周期
3. 然后等待校正完成。
4. 启动时我还要使用LL\_ADC\_REG\_StartConversion开始转换，然后ADC就会等待触发条件。
5. 最后进行使能。

## 2，TIM部分

1. Disable Counter，先把定时器关掉了
2. 到使用的时候我又把定时器打开就行了，没有别的

## 3，DMA部分

1. 先把双Buffer模式滚调了
2. 然后配置了一下优先级
3. 然后利用LL\_DMA\_ConfigAddresses配置了DMA的流，外设地址和存储器地址，和传输方向
4. 然后把DMA的传输完成中断开掉了，CUBEMX仅仅是在NVIC把DMA中断打开了，还需要自己配置DMA的相应中断。
5. 在DMA使能时，我先把DMA的相应中断标志请了，然后LL\_DMA\_SetDataLength配置DMA要传输的数据量，注意是数据量不是字节量。
6. 然后我对DMA进行了使能。**按道理来说，DMA单次传输，每次使能前都配置一下DMA的传输长度比较好，这一点和手册不相符。**

## 4，中断函数部分

中断函数都需要自己编写，注意清除中断标志即可，注意一下逻辑就行。