STM]32H750 USB DAQ设计

## ADC部分设计

**1，ADC部分配置**

ADC代码生成仍然是依靠STM32CUBEMX生成，采用LL库。下面说明STM32CUMEMX界面的设置。

通道选用ADC1的IN3 Single-ended，也就是单通道采样，不是差分采样。为什么选ADC1的IN3通道呢？因为这个通道是快速通道。如图1-1所示

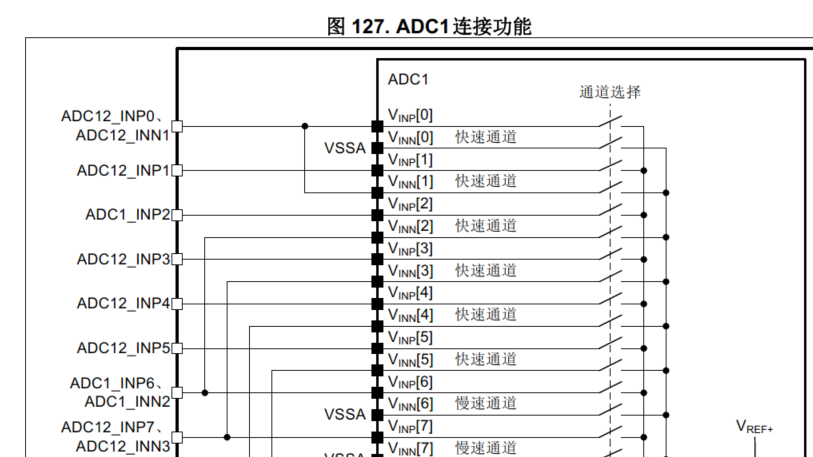


图1-1 STM32H750开发手册第807页

外部中断转换触发关掉；独立模式；异步时钟一分频；16bit分辨率；{说明：ADC分辨率越高，采样时间越长，最后要根据采样效果来决定具体的采样位数}；关掉扫描转换模式；开启连续转换模式{连续转换模式可能也不是一个通道多次ADC采样，可能是把多个通道进行一次ADC采样，后续要注意}；转换结束选择End of sequence of conversion；Overrun Behaviour选择Overrun data preserved， 否则会造成数据丢失的；Left Bit Shift选择No Bit shift；Conversion Data Management Mode选项中，DMA是灰色的，需要手动在DMA选项中Add一个DMA，这样就可以在Conversion Data Management Mode中选择DMA Circular Mode；然后在Low Power Auto Wait中选择Disabled；Enable Regular Conversions选择Enabled， Enable Regular Oversampling选择Disable，过采样是为了平滑数据的，不过不需要。Number of Conversion选择1，Number of Conversion不是转换的次数，而是转换通道的次数。

为了保证定时器的采样频率固定，External trigger Conversion Source选择Timer 1 Trigger Out event.

External Trigger Conversion Edge选择Trigger detection on the rising edge。

Rank中的Channel一般是自动和选用通道对应；

Sampling Time越大，采样效果越好，不过为了保证采样速度，这里选择2.5个cycle。

**2，DMA配置**

DMA Setting里面，Mode选择Circular{改为了Normal，因为DMA传输完N个ADC数据后，我在手动开始}。Data Width都选用Half Word，因为都是16个bit。其它的基本上不用动，也不需要FIFO。

**3，ADC时钟配置**

STM32H750的ADC最高时钟就是80MHz，因此设置其为80MHz。

**4，定时器配置**

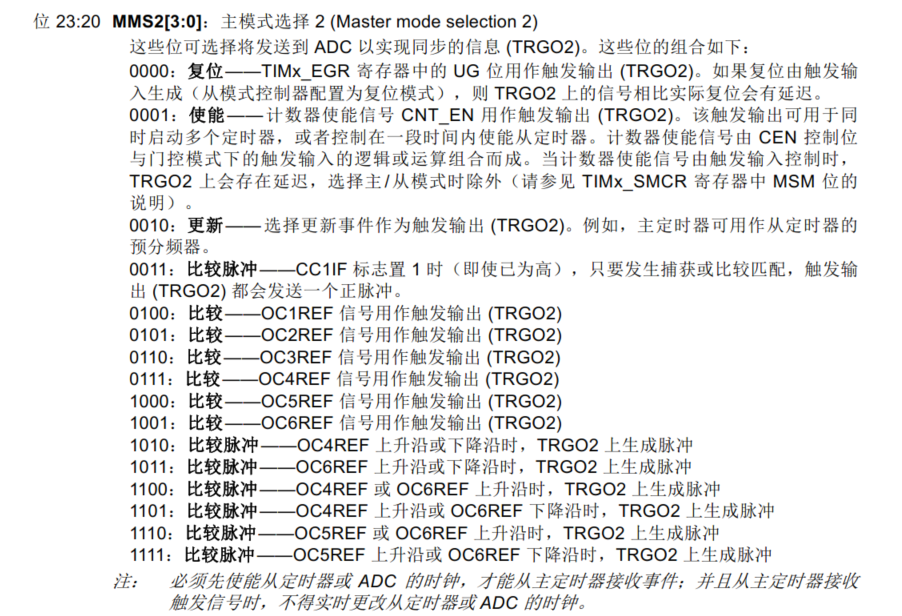
但是STM32H750的定时器时钟挂在APB2\_GRP1上，也就是APB2的Timer Clock，目前设置为200MHz。先让定时器以1us时间中断一次，也就是说1MHz，而定时器时钟为200MHz，那么Counter Period应当设置为200-1。

Repetition Counter就是定时器溢出多少次之后触发中断或者其他事件， 这里设置为0，也就是说定时器溢出了立刻触发中断。

Auto-reload preload失效时，自动重装载寄存器写入新值后，计数器立刻产生技术溢出，然后开始新的技术周期，而使能后，计数器完成旧的计数后，再开始新的计数周期。这里不是很重要。

Master/Slave模式，也就是主从模式，一般发出触发信号的定时器为主模式，接收触发信号的模式为从模式。这里直接就是主模式

Trigger Event Selection TRGO：

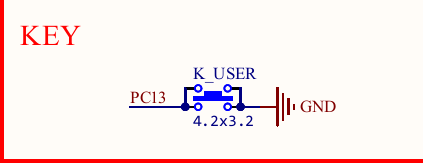


**5，中断优先级配置**

我把ADC的中断优先级设为了2\_2，DMA中断优先级设为了2\_1，ADC采完数据直接送到DMA，DMA完成之后中断，优先级高一点，先把ADC的中断取消掉。按键优先级1\_3。

**6，外部按键中断配置**

按键输入引脚是PC13，然而按键没有加任何上拉和下拉电阻。于是配置成外部中断后，



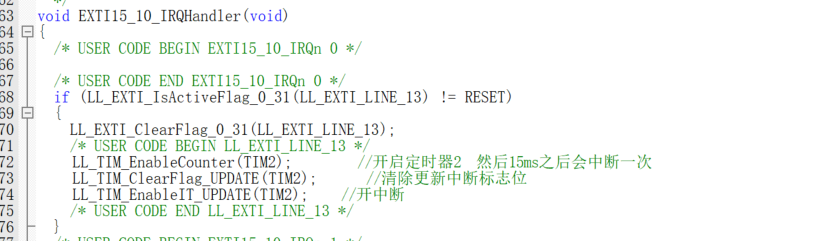
需要把GPIO口上拉一下，然后配置下降沿触发中断即可，这个中断优先级高一点吧。而且为了保证按下按键，按键不会抖动多次进入中断，因此对按键处理做一下小小的改进。

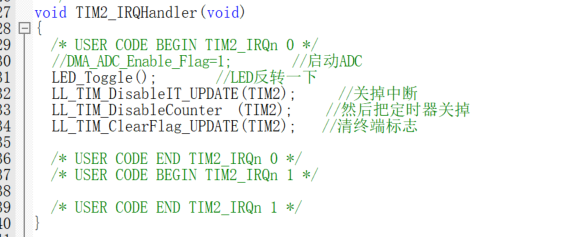
通过外部中断开启定时器，定时器定时20ms后进入中断，在此中断函数里读取按键的状态，则没有问题。

这里开启定时器2，定时器2挂在APB1总线上，200MHz，设置分频系数为20000-1,这样分频后的时钟频率为200MHz/20K = 10KHz，也就是说1个周期0.1ms，这样设置Counter Period为150-1即可，配置优先级为2\_2，然后生成代码。而且不用配置TRGO

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，我对定时器进行一丁点配置，更新标志就置位了，即使定时器是关着的也是如此。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和中断，然后在定时器中断里把中断关掉就应该没问题。这样解决了一点，但是没有完全解决，之前是一直在中断，现在是按一次按键定时器中断两次。

最后终于发现是清标志语句位置的问题，后来最终位置是这样的。





## 生成代码修改

**1，DMA部分**

它在MX\_DMA\_Init中仅仅初始化了时钟，在MX\_ADC1\_Init中真正进行初始化，但是初始化过程少了很多部分，包括DMA传输数据大小，传输起始地址这些东西，因此需要进一步修改。具体相关的修改在DMA\_ADC.c文件中。

看了一下，LL\_ADC\_REG\_SetTDataTransferMode应该挺重要的。

不过DMA有循环模式和普通模式，循环模式是传输任务结束之后重新开始传输，比如我要传512个byte，传完之后自动重新开始传，而普通模式是传输一次就停止。

**2，ADC部分**

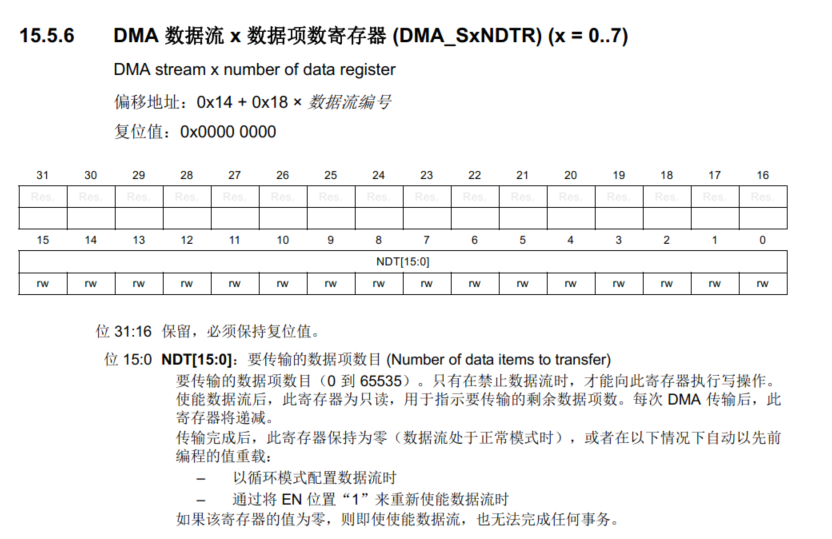
定时器无法触发AD转换，目前正在调试。。

注意RM0443的第888页，EXTSEL寄存器。

原来外部硬件触发ADC，仍然要调用LL\_ADC\_REG\_StartConversion函数，调用之后在下一次触发时才会进行AD转换。

最后功能实现了，每次按按钮的时候都会有数据传输，确实也是收到1024个字节，但是每次按下按钮收到的ADC数据一致，让人不得不怀疑ADC只启动了一次，然后DMA每次传输的数据都没有变。调试发现确实ADC数据从来没有变过，需要查看一下是DMA没有把数据传过来，还是说ADC就没有再启动。

经过调试发现ADC数据确实在变换，应该是DMA传输出错。每次DMA传输完成，DNA\_SxNDTR肯定就已经清零了，应当重新赋值{然后发现并不是这样，在文档RM0433的568页，寄存器是这样描述的}



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。

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。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。

但是收到的数据仍然每次都一样，而且最后几位都是0，应当再检查，先检查ADC数据。把ADC buffer的容量设置的很小，那么数组中的数据全是0，很奇怪的现象。

每次传输ADC\_Buffer的数据不变问题解决了，原来真的要重新对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。

关于数组后面全是0的问题，我把ADC Buffer数组设置成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。

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，只要我方为了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。

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。

不过既然把I-Cache和D-Cache关掉之后一切正常了，那么我不用手动设置ADC\_Buffer的地址在0x24000200上，是不是也行呢？答案是：行！

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。

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。

STM32有128KB的Flash，1MB的SRAM：包括192KB的TCM RAM（64KB的ITCM RAM和128KB的DTCM RAM），以及高达512KB的SRAM和备份与的4KB的SRAM。内存分配可以在KEIL中进行分配。如果不是不知道I-Cache和D-Cache的作用，我一天就搞完了。