**[STM32 ADC学习](http://www.cnblogs.com/hnrainll/archive/2011/01/18/1937888.html)**

1. **什么时候开启AFIO时钟（AFIO意为备用IO，即需要重映射的IO）。**

首先为什么要开启时钟？

答：因为要对寄存器进行读写！而在STM32中对寄存器的读写都是要打开寄存器对应的时钟才可以的【就像人一样，有了跳动的脉搏手臂才能有能量才能进行各种动作】。然后就什么时候AFIO时钟开启（所有时钟都是这样）就清楚了：当需要对“AFIO时钟管理的寄存器”进行读写时AFIO时钟打开！当然不对“AFIO时钟管理的寄存器”读写时也可以打开AFIO时钟，此时只是白白增加能耗、写无用代码（白白浪费程序存储器空间）而已！

接下来：跟AFIO相关的寄存器有哪些呢？

答：根据《STM32中文参考手册\_V10》有：①事件控制寄存器(AFIO\_EVCR)、②复用重映射和调试I/O 配置寄存器(AFIO\_MAPR)、③外部中断配置寄存器1(AFIO\_EXTICR1)、④外部中断配置寄存器2(AFIO\_EXTICR2)、⑤外部中断配置寄存器3(AFIO\_EXTICR3)、⑥外部中断配置寄存器4(AFIO\_EXTICR4)。

应用：在usart1串口通信没有端口重映射时为什么不打开AFIO时钟？PA9、PA10作为通用I/O口也复用了，为什么不打开为什么不打开复用时钟？

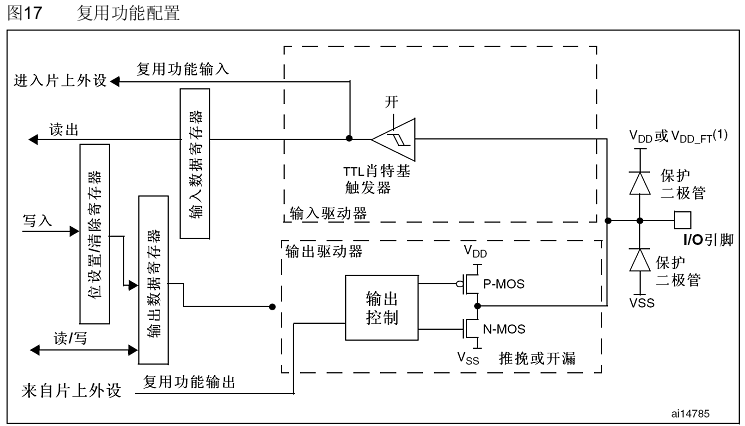
答：AFIO全称：Alternate function I/O  alternate备用的、替代的、交替的，所以AFIO可翻译为“备用功能I/O”，原英文手册上解释很清楚：[《STM32英文手册》P152]

To optimize the number of peripherals available for the 64-pin or the 100-pin or the 144-pin package, it is possible to remap some alternate functions to some other pins. This is

achieved by software, by programming the AF remap and debug I/O configuration register(AFIO\_MAPR) on page 159. In this case, the alternate functions are no longer mapped to

their original assignations.

可见AFIO就设计用来remap some alternate functions to some other pins的，所以根本就没有什么“复用”之说，看来都是翻译惹的祸！另外看一下I/O复用时的情况就更清楚了

[STM32中文参考手册\_V10》P109]：  
 

可见，当设置“PA9为GPIO\_Mode\_AF\_PP（只是一个常数0x18）、PA10为GPIO\_Mode\_IN\_FLOATING(只是一个常数0x04)”时PA9、PA10就已经直接跟串口的相应寄存器连接上了，而串口的寄存器不需要AFIO时钟、PA9/PA10也不许要AFIO时钟，同时有没有对AFIO相应的寄存器操作所以根本不用开启AFIO时钟！

1. **ADC时钟，采样间隔设置。**
2. **ADC工作模式，规则通道、注入通道含义，各通道工作模式。**
3. **ADC运行触发模式。**
4. **双ADC模式。**
5. **ADC多通道连续采样模式，采用中断方式（中断无法区分那个通道）。**

**ADC\_SR中只有JEOC、EOC、AWD三个与中断相关的标志位，所以无法区别发生中断时，当前采样的通道。**

1. **ADC模拟看门狗功能**

**当模拟通道电压采样值低于（ADC\_LTR）或高于阈值（ADC\_HTR）时，ADC状态寄存器ADC\_SR中AWD标志位被置位，如果使能相应中断位（ADC\_CR1的AWDIE位），则触发模拟看门狗中断。**

**模拟看门狗通道选择：**

**通过ADC\_CR1寄存器中的AWDSGL、AWDEN、JAWDEN和AWDCH[4：0]选择所需通道。**

12位ADC是一种逐次逼近型模拟数字数字转换器。它有多达18个通道，可测量16个外部和2个内部信号源。

ADC的输入时钟不得超过14MHZ，它是由PCLK2经分频产生。

如果被ADC转换的模拟电压低于低阀值或高于高阀值，AWD模拟看门狗状态位被设置。

**ADC通常要与DMA一起使用 这里只是简单的用库配置ADC 不断扫描来实现ADC的应用。**

**首先配置GPIO与ADC的时钟：**

ADC\_InitTypeDef  ADC\_InitStructure;   
GPIO\_InitTypeDef GPIO\_InitStructure;

RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_ADC1,ENABLE);   
RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB,ENABLE);

GPIO\_InitStructure.GPIO\_Pin  =GPIO\_Pin\_1;   
GPIO\_InitStructure.GPIO\_Mode =GPIO\_Mode\_AIN;   
GPIO\_Init(GPIOB,&GPIO\_InitStructure); //默认速度为两兆

**配置ADC的运行：**

ADC\_InitStructure.ADC\_Mode              = ADC\_Mode\_Independent;  //独立模式   
ADC\_InitStructure.ADC\_ScanConvMode      =DISABLE;      //连续多通道模式   
ADC\_InitStructure.ADC\_ContinuousConvMode =ENABLE;      /**/连续转换（前一次ADC转化完成后，进行下一次转化），单次转化模式（ADC转化完成后，ADC停止工作）。**  
ADC\_InitStructure.ADC\_ExternalTrigConv  = ADC\_ExternalTrigConv\_None; //转换不受外界决定   
ADC\_InitStructure.ADC\_DataAlign         =ADC\_DataAlign\_Right;   //右对齐   
ADC\_InitStructure.ADC\_NbrOfChannel      =1;       //扫描通道数   
ADC\_Init(ADC1,&ADC\_InitStructure);   
ADC\_RegularChannelConfig(ADC1,ADC\_Channel\_9, 1,ADC\_SampleTime\_1Cycles5); //通道X,采样时间为1.5周期,1代表规则通道第1个这个1是啥意思我不太清楚只有是1的时候我的ADC才正常。   
ADC\_Cmd  (ADC1,ENABLE);             //使能或者失能指定的ADC   
ADC\_SoftwareStartConvCmd(ADC1,ENABLE);//使能或者失能指定的ADC的软件转换启动功能

[](http://images.cnblogs.com/cnblogs_com/hnrainll/201101/201101180042507152.gif)   
这里我用的是ADC1的9通道 PB1引脚。

也用一些默认的配置函数同GPIO 的一样例如： ADC\_StructInit

**ADC\_InitTypeDef structureADC\_InitTypeDef定义于文件“stm32f10x\_adc.h”：**

typedef struct

{

u32 ADC\_Mode; FunctionalState ADC\_ScanConvMode; FunctionalStateADC\_ContinuousConvMode; u32 ADC\_ExternalTrigConv; u32ADC\_DataAlign; u8 ADC\_NbrOfChannel;

} ADC\_InitTypeDef

**注意：为了能够正确地配置每一个ADC通道，用户在调用ADC\_Init()之后，必须调用ADC\_ChannelConfig()来配置每个所使用通道的转换次序和采样时间。**

**然后就是不停的读；**

u16 TestAdc(void)   
{   
u16 adc;   
while(ADC\_GetFlagStatus(ADC1, ADC\_FLAG\_EOC)==RESET); //检查制定ADC标志位置1与否 ADC\_FLAG\_EOC 转换结束标志位   
adc=ADC\_GetConversionValue(ADC1);

returnadc;//返回最近一次ADCx规则组的转换结果   
}

**这个程序的8位单片机风格很重，真正的ADC一定要放在DMA或是中断之中。**

**转自：**[**http://ntn314.blog.163.com/blog/static/16174358420103563814664/**](http://ntn314.blog.163.com/blog/static/16174358420103563814664/)