第四讲 白驹过隙

《庄子·知北游》中提到“人生天地之间，若白驹之过郤，忽然而已。” 时间像白色骏马在细小的缝隙前跑过一样过得极快。 可以用稍纵即逝来形容，时间是连续的，模拟量与数字量最大的区别也在于连续，本讲讲结合时钟中断，完成对 ADC 的控制与读取。记住！时间很宝贵！稍纵即逝！人生没有第二次！

本讲学习目标：

1、了解定时器时钟体系。

2、熟练配置 ADC。

3、完成简单的定时采样程序。

A：感悟 STM32 学习过程

时间也过得极快 STM32F103 的资源正在被我主讲的稀里糊涂学 STM32 侵蚀，我已经通过前三讲将 STM32 整体体系、软件分层、时钟源分配、GPIO 控制、NVIC 体系结构、外部中断、串口中断一一深入的讲明，除了讲义以外聊天记录中还涉及了很多旁路知识，上完本讲，今天希望大家能细细的思考下，现在的 STM32，是否还有我讲课之前的那么混沌？是否自己对其有了新的认识。

在这里先打个招呼，笔者最近世事缠身，疲于分身做 N 件事，尤其是新唐 M0 相关的文档以及手册的重新规整与学习笔记的整理，所以时间掌控的不是很好，讲义的速度也慢了下来，每周六开课，到周四的时候才来得及写讲义，对朋友们来说是很抱歉，过意不去。在此报以歉意，这小小的 QQ 群三位管理员中 Second Life 进步的是最快的，在我引导下已经可以使用从未学过的 VB 来编写串口程序了，打通上位机真的我替他高兴，他自身的努力也

是息息相关的。在这里我也要感谢他以及另外群友对我课程的反馈。

重新回顾一下我们对应一个新的芯片的学习过程：

一．理清其芯片结构，清楚知道其架构背景以及学习资源

二．了解其编程环境与调试方式

三．了解时钟体系与 GPIO 结构，对其简单控制并能有效的将这一过程映射到其他外设中

四．熟能生巧，将所学融会贯通。

第一讲学习中是对应整体架构的理解，不仅对芯片本身，而且对其资源（注意：资源不只是外设，包括相关的 Datasheet、固件库函数等都是芯片的配套资源）应有整体理解。

以下是我按照上面对软件结构的理解，建立的一套工程模板。原因是如果使用库，那么建立工程时会很繁琐，如果有了一套适宜自己的模板，那么相信开发起来会很顺手。

第二讲的学习中，我是先列出了相关的寄存器，然后对相应库函数进行的归类和分析，将相关函数对应程序的位置找到，了解函数的输入输出，对“剪刀加浆糊”是很有帮助的。

知道了这几个时钟，我们来认识一下相关的几个寄存器：

时钟控制寄存器(RCC_CR)	时钟配置寄存器(RCC_CFGR)
APB2 外设复位寄存器(RCC_APB2RSTR)	APB1 外设复位寄存器(RCC_APB1RSTR)
AHB 外设时钟使能寄存器(RCC_AHBENR)	APB2 外设时钟使能寄存器(RCC_APB2ENR)
APB1 外设时钟使能寄存器(RCC_APB1ENR)	AHB 外设时钟复位寄存器(RCC_AHBRSTR)

部分对应库函数：

函数名	rcc.c	函数作用
RCC_DeInit	00216	将外设 RCC 寄存器重设为缺省值
RCC_HSEConfig	00269	设置外部高速晶振（HSE）

第三讲中有这么两段，可以看见这是对函数以及结构体的总结，其实是对寄存器到程序的一个升华。希望能学会这种方式方法。

软件配置：

1、NVIC_Configuration(void)——用以配置中断组与中断优先级；（参看第二版库中文说明）

设置优先级分组：先占优先级和从优先级	NVIC_PriorityGroupConfig
使能或者失能指定的 IRQ 通道	NVIC_InitStructure.NVIC_IRQChannel
设置了成员 NVIC_IRQChannel 中的先占优先级	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority
设置了成员 NVIC_IRQChannel 中的从优先级	NVIC_InitStructure.NVIC_IRQChannelSubPriority
设置了中断功能（失能/使能）	NVIC_InitStructure.NVIC_IRQChannelCmd

注意串口的 GPIO 复用设置：

```
/* A9 USART1_Tx */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;          //推挽输出-TX
GPIO_Init(GPIOA, &GPIO_InitStructure);
/* A10 USART1_Rx  */
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;//浮空输入-RX
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

以上是对 STM32 学习过程的回顾，是否有想过什么是最优的学习？其实就是最适合自己，也有可能你在我所讲的学习过程中一无所获，但是请不要放弃，只要坚持不懈你一定会找到属于自己的学习方式。而我本人是属于比较笨的，比较死板的学习，所以我觉得我的学习方式应该能适合大多数人。

B：STM32-ADC 简单设置与使用

通过上面的回顾，本讲的 B 知识点，利用以下资源独立完整的对一个资源进行解析，希望能学会过程，死学是没用的。

第一步：了解软硬件结构与资源

第二步：了解其相关寄存器及函数作用

第三步：寄存器到程序的升华

资源的联想与积累：

1、硬件部分资源片段：

A： 100 脚和 144 脚封装：为了确保输入为低压时获得更好精度，用户可以连接一个独立的外部参考电压 ADC 到 VREF+和 VREF-脚上。在 VREF+的电压范围为 2.4V～VDDA。

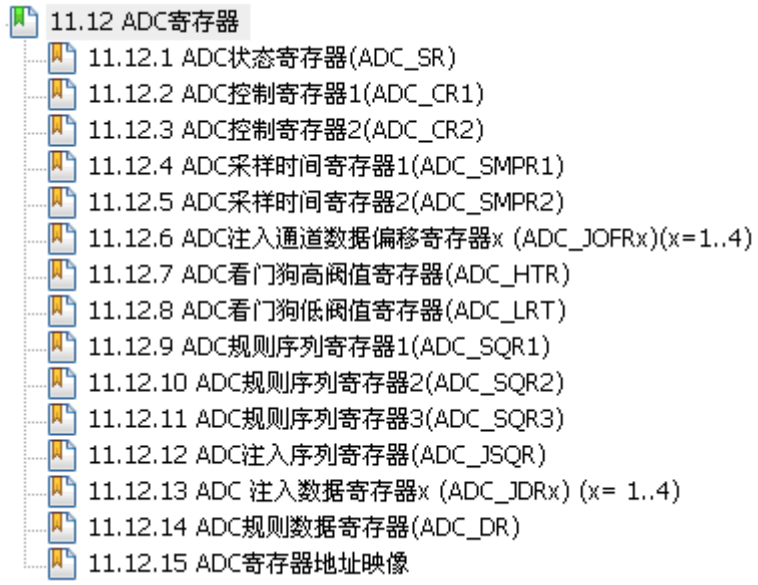
64 脚或更少封装：没有 VREF+和 VREF-引脚，他们在芯片内部与 ADC 的电源(VDDA)和地(VSSA)相联。

B： 时钟：

C： 特性：12 位 ADC 是一种逐次逼近型模拟数字转换器。它有多达 18 个通道，可测量 16 个外部和 2 个内部信号源。各通道的 A/D 转换可以单次、连续、扫描或间断模式执行。ADC 的结果可以左对齐或右对齐方式存储在 16 位数据寄存器中。

2、寄存器与库函数部分的资源片段：

A：相关寄存器：



B：库函数资源片段：

结构体：

```
typedef struct
{
  uint32_t ADC_Mode;
  FunctionalState ADC_ScanConvMode;
  FunctionalState ADC_ContinuousConvMode;
  uint32_t ADC_ExternalTrigConv;
  uint32_t ADC_DataAlign;
  uint8_t ADC_NbrOfChannel;
}ADC_InitTypeDef;
```

操作函数：

```
ADC_RegularChannelConfig
ADC_Cmd
ADC_SoftwareStartConvCmd
ADC_GetFlagStatus
ADC_GetConversionValue
```

3、资源的整合和利用：

概念解析：

1：（间断模式）规则组与注入组：规则组由多达 16 个转换组成。规则通道和它们的

转换顺序在 ADC_SQRx 寄存器中选择。规则组中转换的总数应写入 ADC_SQR1 寄存器的 L[3:0]位中。

注入组由多达 4 个转换组成。注入通道和它们的转换顺序在 ADC_JSQR 寄存器中选择。注入组里的转换总数目应写入 ADC_JSQR 寄存器的 L[1:0]位中。如果 ADC_SQRx 或 ADC_JSQR 寄存器在转换期间被更改，当前的转换被清除，一个新的启动脉冲将发送到 ADC 以转换新选择的组。

2：单与多：

单次转换模式下，ADC 只执行一次转换。

在**连续**转换模式中，当前面 ADC 转换一结束马上就启动另一次转换。

3：中断与标志

ADC中断

中断事件	事件标志	使能控制位
规则组转换结束	EOC	EOCIE
注入组转换结束	JEOC	JEOCIE
设置了模拟看门狗状态位	AWD	AWDIE

过程描述：

RCC->GPIO->ADC->while(1){XXX}

实际操作：

ADC_Init	函数设置ADC相关模式的配置
ADC_RegularChannelConfig	设置对应ADC参数
ADC_Cmd	使能ADC
ADC_SoftwareStartConvCmd	开始ADC转换
ADC_GetFlagStatus	读取ADC转换标志位
ADC_GetConversionValue	读取ADC数据

以下程序源于网络，使用独立模式，单通道连续采样

```
void ADC_Configuration(void)
{
  ADC_InitTypeDef   ADC_InitStructure;
  GPIO_InitTypeDef  GPIO_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1, ENABLE);
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);

  /* PB1*/
  GPIO_InitStructure.GPIO_Pin  = GPIO_Pin_1;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  /* ADC1 */
  ADC_InitStructure.ADC_Mode               = ADC_Mode_Independent;       //独立模式
  ADC_InitStructure.ADC_ScanConvMode       = ENABLE;                     //连续多通道模式
  ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;                     //连续转换
  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_55Cycles5);
  //通道x,采用时间为55.5周期，1代表规则通道第1个
  ADC_Cmd    (ADC1, ENABLE);                 /* Enable ADC1                                */
  ADC_SoftwareStartConvCmd(ADC1,ENABLE);/* Start ADC1 Software Conversion     */ //使能转换开始
}
```

```
u16 TestAdc(void)
{
    u16 adc;
    if(ADC_GetFlagStatus(ADC1, ADC_FLAG_EOC)==SET)
    {
        adc=ADC_GetConversionValue(ADC1);
    }
    return adc;
}
```

这只是一种**简单的ＡＤＣ应用**，ＳＴＭ３２的ＡＤＣ可谓博大精深，方式有Ｎ种，甚至有利用此片上ＡＤＣ设计简易示波器的，虽然是等效采样，但是能做出来也是相当不错的，在此不能一一赘述，但是我真诚希望能掌握此种学习方式这样能对今后的ＳＴＭ３２乃至别的芯片的学习有着很大的帮助。

C：课后练习

第四讲旨在学习 STM32 学习过程，将 STM32 资源按照学习过程分析。课上内容十分多，远比讲义来得深，希望课后能对我课上深入强调的一些我对概念、模式的理解加以巩固，课后加以练习，完成串口输出 ADC 采样数据的过程。

另：下节课（第五讲）将会是一个整合，将之前所讲进行回顾与整合，利用串口完成一模拟数据的采样，可以说是承上启下的过程，希望能好好听下去。有兴趣的也可以试着编写下程序。**要求如下：**

下位机利用外部中断触发 ADC 采样，并利用串口将数据传回上位机或串口助手，PC 发送指令#OK 点亮 STM32 板上指示灯(绿)，若返回#ERROR 点亮 STM32 板上指示灯(红)。

D：听课笔记