第三章 AD 转换

本章的内容分两部分，第一是 AD 的单通道转换，第二是 AD 的多通道转换。首先先将单通道转换。

STM32 中自带的 AD 最大的转换频率是 14MHZ，共有 16 个转换通道，每个转换通道对应的管脚如下表所示。

管脚名	默认复用功能
PF6	ADC3_IN4
PF7	ADC3_IN5
PF8	ADC3_IN6
PF9	ADC3_IN7
PF10	ADC3_IN8
PC0	ADC123_IN10
PC1	ADC123_IN11
PC2	ADC123_IN12
PC3	ADC123_IN13
PA3	ADC123_IN3
PA6	ADC12_IN6
PA7	ADC12_IN7
PC4	ADC12_IN14
PC5	ADC12_IN15
PB0	ADC12_IN8
PB1	ADC12_IN9

注：ADC123_IN10 表明 PC0 管脚可以作为 AD1，AD2，AD3 的第 10 通道。

下面我们将 PC0 配置成 AD1 的通道 10 为例进行讲解。

3.1 首先我们应将 PC0 设置成模拟输入：

```
#include "adc.h"
/*为何定义 ADC1_DR_Address    为((u32)0x40012400+0x4c)
，因为存放 AD 转换结果的寄存器的地址就是 0x4001244c*/
#define    ADC1_DR_Address        ((u32)0x40012400+0x4c)
/*定义变量 ADC_ConvertedValue,放 AD1 通道 10 转换的数据*/
__IO uint16_t ADC_ConvertedValue;

static void ADC1_GPIO_Config(void)
{
  GPIO_InitTypeDef    GPIO_InitStructure;
   /* Enable ADC1 and GPIOC clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1                    |
                                  RCC_APB2Periph_GPIOC,ENABLE);
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 ;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
```

3.2 设置完端口后下一步当然是对 AD 进行初始化：

这里需要补充一个知识点 DMA，DMA 就相当与 CPU 的一个秘书，他的作用就是帮 CPU 减轻负担的。说的再具体点就是帮 CPU 来转移数据的。我们都知道，AD 每次转换结束后会将转换的结果放到一个固定的寄存器里，以往我们如果想将该寄存器中的值赋给某一变量时会用到赋值语句，如果不用 DMA，则赋值语句便要 CPU 来完成，CPU 本来就要忙着处理其他事情，现在还要来解决赋值语句这么简单的问题，肯到会蛋疼。所以需要 DMA 这个秘书来帮他解决这个问题。由于 DMA 只是个秘书，所以比较笨，你只有把任务交代清楚了她才能很好的完成任务。那么怎样来给 DMA 吩咐任务呢，聪明的人肯定想到了，那当然是“DMA_Init(DMA1_Channel1, &DMA_InitStructure)”这个函数啦。下面就来一步步的来给 DMA 交代任务。

```
/* 函数名：ADC1_Mode_Config
* 描述 ：配置 ADC1 的工作模式为 MDA 模式
* 输入 ：无
* 输出 ：无
* 调用 ：内部调用
*/
static void ADC1_Mode_Config(void)
{
  DMA_InitTypeDef    DMA_InitStructure;
  ADC_InitTypeDef    ADC_InitStructure;
  /* 将与 DMA 有关的寄存器设我初始值 */
  DMA_DeInit(DMA1_Channel1);
  /*定义 DMA 外设基地址，这里的 ADC1_DR_Address 是用户自己定义的，即为存放转换结果的寄存器 ，他的作用就是告诉 DMA 取数就到 ADC1_DR_Address 这里来取。*/
  DMA_InitStructure.DMA_PeripheralBaseAddr =ADC1_DR_Address;
  /* 定义内存基地址，即告诉 DMA 要将从 AD 中取来的数放到 ADC_ConvertedValue 中 */
  DMA_InitStructure.DMA_MemoryBaseAddr =(u32)&ADC_ConvertedValue;
  /*定义 AD 外设作为数据传输的来源，即告诉 DMA 是将 AD 中的数据取出放到内存中，不能反过来*/
  DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
  /*指定 DMA 通道的 DMA 缓存的大小,即告诉 DMA 开辟几个内存空间，由于我们只取通道 10 的 AD 数据所以只需开辟一个内存空间*/
  DMA_InitStructure.DMA_BufferSize = 1;
  /*设定寄存器地址固定，即告诉 DMA，只从固定的一个地方取数*/
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  /*设定内存地址固定，即每次 DMA，，只将数搬到固定的内存中*/
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
/*设定外设数据宽度，即告诉 DMA 要取的数的大小*/
  DMA_InitStructure.DMA_PeripheralDataSize                    =
  DMA_PeripheralDataSize_HalfWord;
```

```
/*设定内存的的宽度*/
DMA_InitStructure.DMA_MemoryDataSize                                    =
DMA_MemoryDataSize_HalfWord;
/*设定 DMA 工作再循环缓存模式，即告诉 DMA 要不停的搬运，不能偷懒*/
DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
/*设定 DMA 选定的通道软件优先级*/
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
DMA_Init(DMA1_Channel1, &DMA_InitStructure);
/* Enable DMA channel1，CPU 有好几个 DMA 秘书，现在只用 DMA1_Channel1
   这个秘书*/
 DMA_Cmd(DMA1_Channel1, ENABLE);
/*设置 ADC 工作在独立模式*/
ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
/*规定 AD 转换工作在单次模式，即对一个通道采样*/
ADC_InitStructure.ADC_ScanConvMode = DISABLE ;
/*设定 AD 转化在连续模式*/
ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
/*不使用外部促发转换*/
ADC_InitStructure.ADC_ExternalTrigConv = ADC_ExternalTrigConv_None;
/*采集的数据在寄存器中以右对齐的方式存放*/
ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
/*设定要转换的 AD 通道数目*/
ADC_InitStructure.ADC_NbrOfChannel = 1;
ADC_Init(ADC1, &ADC_InitStructure);
/*配置 ADC 时钟，为 PCLK2 的 8 分频，即 9MHz*/
RCC_ADCCLKConfig(RCC_PCLK2_Div8);
/*配置 ADC1 的通道 11 为 55.5 个采样周期 */
ADC_RegularChannelConfig(ADC1,                ADC_Channel_10,                1,
ADC_SampleTime_55Cycles5);
      /* Enable ADC1 DMA */
 ADC_DMACmd(ADC1, ENABLE);
/* Enable ADC1 */
 ADC_Cmd(ADC1, ENABLE);
/*复位校准寄存器 */
 ADC_ResetCalibration(ADC1);
/*等待校准寄存器复位完成 */
 while(ADC_GetResetCalibrationStatus(ADC1));
/* ADC 校准 */
 ADC_StartCalibration(ADC1);
/*  等待校准完成*/
 while(ADC_GetCalibrationStatus(ADC1));
/*  由于没有采用外部触发，所以使用软件触发 ADC 转换 */
 ADC_SoftwareStartConvCmd(ADC1, ENABLE);
```

}

配置完以上的程序，那么 AD 每转换一次，DMA 都会将转换结果搬到变量 ADC_ConvertedValue 中，而不需用每次都用赋值语句来取值 AD 转换的值。

第二部分：AD 多路采样

```
#include "adc.h"
#define    ADC1_DR_Address       ((u32)0x40012400+0x4c)
/*定义数组变量 ADC_ConvertedValue[2],分别放 AD1 通道 10 和 11 转换的数据*/
__IO uint16_t ADC_ConvertedValue[2];
/*
* 函数名：ADC1_GPIO_Config
* 描述 ：使能 ADC1 和 DMA1 的时钟,设置 PC0，PC1 为模拟输入
* 输入 ：无
* 输出 ：无
* 调用 ：内部调用
*/
static void ADC1_GPIO_Config(void)
{
  GPIO_InitTypeDef    GPIO_InitStructure;
   /* Enable DMA clock */
  RCC_AHBPeriphClockCmd(RCC_AHBPeriph_DMA1, ENABLE);
   /* Enable ADC1 and GPIOC clock */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1                    |
RCC_APB2Periph_GPIOC,ENABLE);
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0 |GPIO_Pin_1;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AIN;
   GPIO_Init(GPIOC, &GPIO_InitStructure);
}
/* 函数名：ADC1_Mode_Config
* 描述 ：配置 ADC1 的工作模式为 MDA 模式
* 输入 ：无
* 输出 ：无
* 调用 ：内部调用
*/
static void ADC1_Mode_Config(void)
{
  DMA_InitTypeDef    DMA_InitStructure;
  ADC_InitTypeDef    ADC_InitStructure;
```

```
    /* DMA channel1 configuration */
    DMA_DeInit(DMA1_Channel1);
    /*定义 DMA 外设基地址,即为存放转换结果的寄存器*/
    DMA_InitStructure.DMA_PeripheralBaseAddr =ADC1_DR_Address;
    /*定义内存基地址*/
    DMA_InitStructure.DMA_MemoryBaseAddr
=(u32)&ADC_ConvertedValue;
    /*定义 AD 外设作为数据传输的来源*/
    DMA_InitStructure.DMA_DIR = DMA_DIR_PeripheralSRC;
    /*指定 DMA 通道的 DMA 缓存的大小,即需要开辟几个内存空间，本
实验有两个转换通道，所以开辟两个*/
    DMA_InitStructure.DMA_BufferSize = 2;
    /*设定寄存器地址固定*/
    DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
    /*设定内存地址递加，即每次 DMA 都是将该外设寄存器中的值传到
两个内存空间中*/
    DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
  /*设定外设数据宽度*/
    DMA_InitStructure.DMA_PeripheralDataSize                    =
DMA_PeripheralDataSize_HalfWord;
    /*设定内存的的宽度*/
    DMA_InitStructure.DMA_MemoryDataSize                        =
DMA_MemoryDataSize_HalfWord;
    /*设定 DMA 工作再循环缓存模式*/
    DMA_InitStructure.DMA_Mode = DMA_Mode_Circular;
    /*设定 DMA 选定的通道软件优先级*/
    DMA_InitStructure.DMA_Priority = DMA_Priority_High;
    DMA_InitStructure.DMA_M2M = DMA_M2M_Disable;
    DMA_Init(DMA1_Channel1, &DMA_InitStructure);
    /* Enable DMA channel1 */
     DMA_Cmd(DMA1_Channel1, ENABLE);
    /*设置 ADC 工作在独立模式*/
    ADC_InitStructure.ADC_Mode = ADC_Mode_Independent;
  /* 规定 AD 转换工作在扫描模式，即对多个通道采样 */
    ADC_InitStructure.ADC_ScanConvMode = ENABLE ;
    /*设定 AD 转化在连续模式*/
    ADC_InitStructure.ADC_ContinuousConvMode = ENABLE;
    /*不使用外部促发转换*/
    ADC_InitStructure.ADC_ExternalTrigConv                      =
ADC_ExternalTrigConv_None;
    /*采集的数据在寄存器中以右对齐的方式存放*/
    ADC_InitStructure.ADC_DataAlign = ADC_DataAlign_Right;
    /*设定要转换的 AD 通道数目*/
    ADC_InitStructure.ADC_NbrOfChannel = 2;
```

```
    ADC_Init(ADC1, &ADC_InitStructure);
    /*配置 ADC 时钟，为 PCLK2 的 8 分频，即 9MHz*/
    RCC_ADCCLKConfig(RCC_PCLK2_Div8);
  /*配置 ADC1 的通道 10 和 11 的转换先后顺序以及采样时间为为 55.5 个采样周期 */
    ADC_RegularChannelConfig(ADC1, ADC_Channel_10, 1, ADC_SampleTime_55Cycles5);
    ADC_RegularChannelConfig(ADC1, ADC_Channel_11, 2, ADC_SampleTime_55Cycles5);
    /* Enable ADC1 DMA */
    ADC_DMACmd(ADC1, ENABLE);
    /* Enable ADC1 */
    ADC_Cmd(ADC1, ENABLE);
    /*复位校准寄存器 */
    ADC_ResetCalibration(ADC1);
    /*等待校准寄存器复位完成 */
    while(ADC_GetResetCalibrationStatus(ADC1));
    /* ADC 校准 */
    ADC_StartCalibration(ADC1);
    /* 等待校准完成*/
    while(ADC_GetCalibrationStatus(ADC1));
    /* 由于没有采用外部触发，所以使用软件触发 ADC 转换 */
    ADC_SoftwareStartConvCmd(ADC1, ENABLE);
  }
```

!!!!!单通道采样与多通道采样的不同点都在第二段程序中用红色标出来了，注意比较。

总结：DMA 就是一个无私奉献的搬运工，想将外设寄存器中的值放入内存中原本需要 CPU 来完成，现在 DMA 来帮 CPU 完成，这在一定程度上解放了 CPU.