浙江大学 实验报告

专业：电子信息工程
姓名：冯静怡
学号：3220104119
地点：紫金港东三 406
日期：2024 年 6 月 12 日

课程名称：微机原理及应用实验　　指导老师：胡斯登
实验名称：FFT　　同组学生：陈亦乔

1 实验目的

1. 掌握单片机 AD 采样的使用方法 (DMA)；
2. 掌握单片机 FFT 的使用方法。

2 实验原理与思路

2.1 ADC（Analog to Digital Converter）

原理： ADC（模数转换器）的作用是将连续的模拟信号转换为离散的数字信号。STM32 中的 ADC 模块通常具有以下特性：

- 分辨率：12 位分辨率, 可以产生 0 到 4095 之间的数字值。
- 采样率：指每秒钟 ADC 可以进行多少次转换，本实验通过设置 TIM3，通过 TIM3 计时器溢出从而调用 ADC。
- 通道：设置 GPIO 口进行 AD 转换。
- 转换模式：包括单次转换、连续转换和扫描模式，本实验采用单次转换。

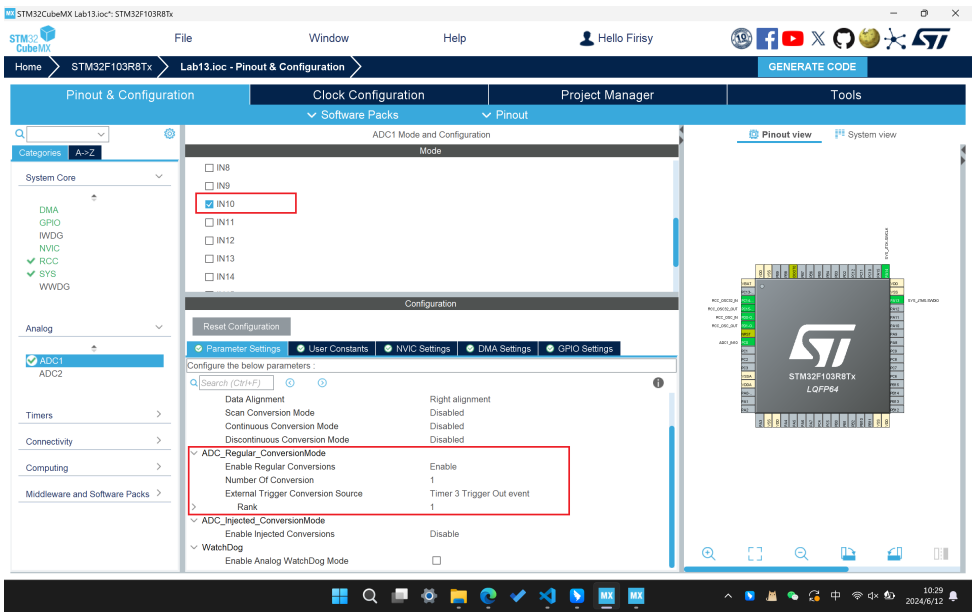


图 1: AD 转换设置

工作流程：

- 设置 ADC 参数，包括分辨率、采样时间、通道等。
- 启动 ADC 进行采样。
- 将采样得到的模拟信号转换为数字信号。

2.2 DMA（Direct Memory Access）

原理： DMA（直接内存访问）模块允许外设（如 ADC）在不经过 CPU 的情况下，直接将数据传输到内存或从内存传输到外设，从而提高数据传输效率，减轻 CPU 负担。

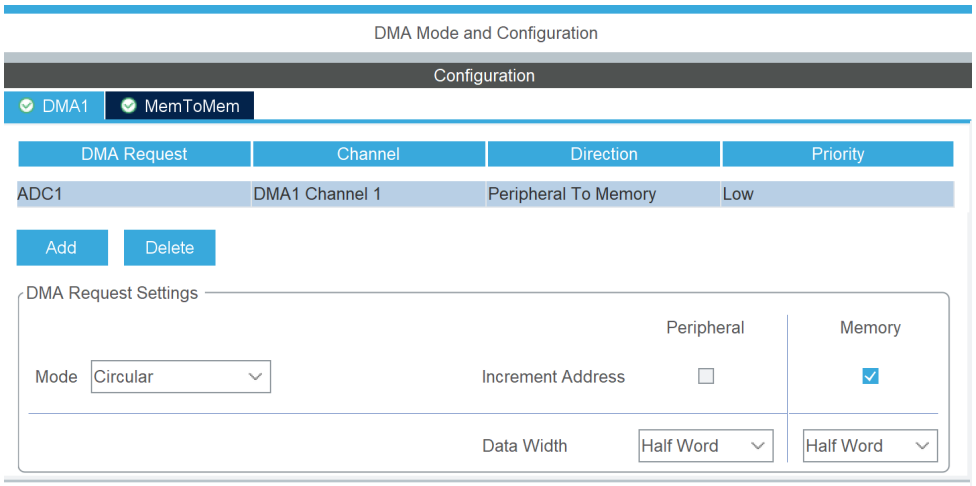


图 2: DMA 传输

工作流程：

- 配置 DMA 传输方向（如从外设到内存），模式为循环，数据长度为半字即 16 位。
- 设置 DMA 的源地址和目标地址。
- 启动 DMA，使其在 ADC 完成数据转换后自动将数据传输到内存。

2.3 ADC 与 DMA 结合

1. 初始化设置 ADC 和 DMA。

```
1  HAL_ADCEx_Calibration_Start(&hadc1); //校准ADC，通过校准可以减少ADC转换中的误差。
2  HAL_ADC_Start_DMA(&hadc1,(uint32_t *)sampleData,FFT_NUM); //启动ADC，并使能DMA传输，将采样数据
       传输到内存中(sampleData)。
3  HAL_TIM_Base_Start(&htim3); //启动TIM3计时器，通过TIM3计时器溢出来触发ADC采样。
```

2. 设置 ADC-DMA 采样完成中断回调函数，当 ADC 采样完成时，停止 TIM3 计时器，停止 ADC 采样，并设置采样完成标志位。

```
1  void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef *hadc)
2  {
3      if(hadc == &hadc1) //ADC转换中断
4      {
5          HAL_TIM_Base_Stop(&htim3); //停止TIM3计时器
```

```
        HAL_ADC_Stop_DMA(&hadc1); //停止ADC-DMA
        isSampleReady = 1; //标志位置1
    }
}
```

3. 在主函数中，当采样完成标志位为 1 时，进行数据处理。

```
if(isSampleReady == 1) // 标志位置1
{
    dataIn[i] = sampleData[i]*3.3/4096; //处理输入数据转换为正确的浮点型
    /* 数据处理过程略 */
    isSampleReady = 0; //标志位清零
    HAL_ADC_Start_DMA(&hadc1,(uint32_t *)sampleData,FFT_NUM);//开始新一轮数据的读入
}
```

2.4 FFT（Fast Fourier Transform）

原理： FFT（快速傅里叶变换）是一种用于计算离散傅里叶变换（DFT）的高效算法。它将时域信号转换为频域信号，以分析信号的频率成分。

```
#include "arm_math.h"
#include "arm_const_structs.h"
```

使用以上两个 ARM 自带的数学库，执行 FFT 操作：

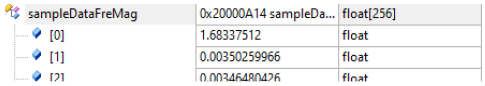
```
for(i=0;i<FFT_NUM;i++)
{
    fftIn[2*i] = sampleData[i]*3.3/4096; //处理输入数据，数组[2*i]为实数部分，数组[2*i+1]为虚数部分
    fftIn[2*i+1] = 0 ;
}
arm_cfft_f32(&arm_cfft_sR_f32_len256,fftIn,0,1); //库函数进行FFT
for(i=0;i<FFT_NUM;i++)
{
    sampleDataFreMag[i] = sqrtf(fftIn[2*i]*fftIn[2*i]+fftIn[2*i+1]*fftIn[2*i+1])/FFT_NUM; //将FFT结果转换为可读性的幅值结果
}
```

其中 `arm_cfft_f32()` 各个参数含义：

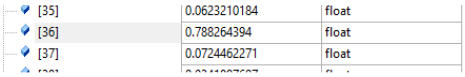
```
/*
* @details
* @brief Processing function for the floating-point complex FFT.
* @param[in] *S points to an instance of the floating-point CFFT structure.
* @param[in, out] *p1 points to the complex data buffer of size <code>2*fftLen</code>. Processing occurs in-place.
* @param[in] ifftFlag flag that selects forward (ifftFlag=0) or inverse (ifftFlag=1) transform.
* @param[in] bitReverseFlag flag that enables (bitReverseFlag=1) or disables (bitReverseFlag=0) bit reversal of output.
* @return none.
*/
```

3 实验结果

设置输入频率为 40kHz，1.65V 直流偏置，1.65V 的幅值，并将 `sampleDataFreMag` add to watch，得到以下两个数据显著不为 0.



(a) `sampleDataFreMag[0]`



(b) `sampleDataFreMag[36]`

图 3: 实验结果

通过计算可以得到:

TIM3 的 ARR 设置为 280，则单次间隔采样时间为

$$t = \frac{1}{72 \times 10^6} \times 280 = 3.9\mu S$$

读入数据为 256 个数据点，则频率分辨率为：

$$\Delta f = 1/t/256 = 1.004kHz$$

可以看到最后结果，直流偏置 `sampleDataFreMag[0]` $= 1.68V \approx 1.65V$，36 次谐波分量代表 36kHz，但并非 40kHz，可能由于频率分辨率不高的原因导致，`sampleDataFreMag[36]` $= 0.78 \approx 1.65V/2$，实验结果基本符合预期。