**《嵌入式系统课程设计报告》**

**设计题目：基于STM32F103的示波器制作**

**专业班级：**

**学号：**

**姓名：**

**指导教师：**

## 设计任务和要求

1. 采用STM32F103单片机和2.8寸液晶屏显示；
2. 能显示外部输入的波形，频率1KHz。
3. 显示的波形在x、y方向上可以调整。
4. LCD显示清晰，波形无明显失真，不能有明显的残影、乱码、抖动。

# 设计方案分析

## 软件方案的比较与选择

### 测试波形产生方案

（1）正弦波和三角波的产生是用Timer（定时器）、DAC（Digital to Analog Converter，数字模拟转换器）、DMA（直接存储器存取）这些外设配合使用产生。可以设置频率。默认PA4输出三角波，PA5输出正弦波。

（2）方波的产生是由PWM（Pulse Width Modulation，脉冲宽度调制）产生的，通过配置PWM的占空比为50%就可实现方波的输出，默认PA3输出方波

由于使用DMA控制DAC输出和定时器输出PWM不会占用CPU核心资源，不影响开发其他功能，故选用此方案。

### 示波器ADC采集的设计方案

（1）方案一：通过定时器中断控制进行单次AD转换

优点：代码逻辑简单且容易理解。

缺点：AD采样速率较慢无法测量高于1KHz的波形，且频繁占用CPU资源产生中断。

（2）方案二：使用DMA配合定时器更新事件进行AD转换

优点：使用DMA[5]和定时器3的更新事件来控制AD转换可以测量更高频率的波形，且占用CPU资源少，功耗降低。

缺点：代码逻辑复杂

为了更好的使用性能和锻炼自己编程能力，此处选用方案二使用DMA配合定时器3更新事件进行AD转换。

### 波形显示设计方案

（1）方案一：每一帧绘图前先将屏幕清空再显示

优点：代码逻辑简单

缺点：界面刷新速度较慢，实时性低，用户视觉体验较差。

（2）方案二：绘制波形时清除每一个像素点上次显示的内容[6]

优点：刷新速度远大于方案一且没有残影，视觉体验较好。

缺点：代码逻辑复杂，编程难道较大需要占用更多的内存空间保存上一次的点的信息

为了更好的视觉体验和锻炼自己编程能力，此处选用方案二绘制波形时清除每一个像素点上次显示的内容

# 硬件电路设计

# 程序设计与分析

系统软件设计主要包括各个组件的初始化、DMA读取ADC、绘制网格、字符显示、绘制波形、清除波形等程序设计。LCD液晶显示模块正常工作前，需要对其进行初始化，主要包括复位液晶、确定显示对比度、偏压比等参数，才能实现绘制网格、字符、波形以及其他显示功能。

在向液晶显示模块写入数据或命令时，要注意写入的时序，严格按照时序编写驱动程序才能正常工作。设计程序时，为了方便操作，可以将相关引脚的操作用宏定义封装起来。

## 主函数

主函数中对各个外设进行了初始化，绘制开机动画之后，将示波器核心部分的ADC1，定时器3，DMA1进行初始化，初始化完成之后启动DMA通道转换AD结果，当转换结果数量达到设定值后产生DMA中断，在DMA中断里进行绘制波形等相应的处理，处理完成后再开始DMA通道进行转换。DMA中断中用LED0灯的亮灭判断程序是否在运行。流程图如图 4.1主函数流程图所示。



图 4.1主函数流程图

对示波器始化代码如下：

int main(void)

{

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_2);

Stm32\_Clock\_Init(SYSCLOCK/8); //配置系统频率

delay\_init(SYSCLOCK); //延时函数初始化

uart\_init(SYSCLOCK,9600); //串口初始化

TestWave\_Init(); //测试波形输出初始化

LED\_Init();

KEY\_Init();

LCD\_Init();

Logo(); //显示启动界面

DSO\_Init(); //示波器核心部分初始化

run\_sta=1;

while(1);

}

## 测试信号输出

由于实际使用示波器时不一定有函数发生器来检查示波器自身是否正常工作，所以为了示波器自检的方便，设计了三种1kHz的函数信号通过DAC配合DMA和定时器进行输出，充分利用了STM32F103RCT6丰富的外设功能。正弦波和三角波测试信号分别通过DAC1和DAC2进行输出，方波信号则是通过定时器的PWM输出功能，输出占空比为50%的PWM即可得到想要的方波。由于正弦波和三角波是通过DMA控制DAC输出，方波是使用定时器的PWM输出功能进行输出，两种输出过程在初始化之后并没有使用CPU核心进行控制，所以不会影响后续示波器功能的程序编写。

## 启动界面和示波器界面设计

为增加作品美观，对示波器启动时和运行时的界面做了相应的美化，启动时界面显示海南大学LOGO和姓名学号，示波器主程序界面可显示运行状态，波形平移位置，背景网格和每网格对应电压和时间。

### 运行界面

示波器主界面通过输出字符来显示相应的字母和数组，调整好对应坐标进行显示。波形位置显示是通过符号[-]|按照一定组合成字符串即可得到这种效果。网格是通过画点实现的，横线每隔5像素画一个点，纵向每隔20像素画一条横线。竖线每隔5像素画一个点，横向每隔20像素画一条竖线。画完点阵之后在外部画一个矩形实线框起来，横纵坐标中心位置画十字实线表示中点。整体效果如图 4.2所示

画网格核心部分代码如下：

for(x = 20;x < 320;x += 20) //画列点

{

for(y = 20;y <220;y += 5)

LCD\_Fast\_DrawPoint(x,y,0XAAAA);

}

for(y = 20;y < 220; y += 20) //画行点

{

for(x = 0 ; x < 320 ; x += 5)

LCD\_Fast\_DrawPoint(x,y,0xAAAA);

}

## ADC模数转换

### ADC 初始化

先进行 GPIO 口的初始化，使能引脚 PA1 上的 AFIO 时钟，配置引脚PA1 为模拟输入通道，即采集待测信号的通道。再进行 ADC 的初始化，设置独立的工作模式，模数转换在定时器3更新事件触发下进行，开启 ADC 通道数1，初始化外设 ADC1并使能 ADC1，配置ADC转换规则。使能ADC的DMA请求。流程图与程序如下：



图 4.4初始化程序流程图

ADC初始化核心代码：

void DSO\_ADC\_Init(void)

{

RCC\_APB2PeriphClockCmd(ADC1|GPIOA,ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_1;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AIN;

GPIO\_Init(GPIOA, &GPIO\_InitStructure);

ADC\_InitStructure.ADC\_Mode = ADC\_Mode\_Independent; ADC\_InitStructure.ADC\_ScanConvMode = DISABLE ;

ADC\_InitStructure.ADC\_ContinuousConvMode = DISABLE; ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_T3\_TRGO;

ADC\_InitStructure.ADC\_NbrOfChannel = 1;

ADC\_Init(ADC1, &ADC\_InitStructure);

ADC\_RegularChannelConfig(ADC1, ADC\_Ch\_1, 1, ADC\_SampleTime\_7Cycles5);

ADC\_DMACmd(ADC1, ENABLE);

ADC\_Cmd(ADC1, ENABLE);

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

}

### DMA和定时器配置

由于ADC采集数据是通过DMA传输的，所以要相应的配置DMA的参数，核心的主要有设置源地址为ADC1->DR寄存器，内存地址为存放AD转换结果的数组，将传输方向设置为外设到内存，并设置buffersize为要转换的数量，由于ADC->DR寄存器是固定的，所以设置外设地址不自增，存放结果时是往数组里一个个放的，所以内存地址自增，内存数据大小配置为半字。配置完这些参数后初始化DMA并使能DMA通道1的传输完成中断，后续波形显示和计算都在DMA通道1中断里处理。由于ADC初始化时设置了通过定时器3的更新事件触发，所以要对定时器3进行相应的配置，按照正常流程配置好分频系数，重装载值，并设置定时器3为输出更新事件源。这样每次定时器3计数到ARR的值后就触发ADC一次转换，通过改变定时器3的ARR即可控制采样周期。

DMA和定时器初始化核心部分代码:

DMA\_InitStructure.DMA\_PeripheralBaseAddr = (uint32\_t)(&ADC1->DR);

DMA\_InitStructure.DMA\_MemoryBaseAddr = (u32)&ADC\_Value;

DMA\_InitStructure.DMA\_DIR = DMA\_DIR\_PeripheralSRC;

DMA\_InitStructure.DMA\_BufferSize = 600;

DMA\_InitStructure.DMA\_PeripheralInc = DMA\_PeripheralInc\_Disable;

DMA\_InitStructure.DMA\_MemoryInc = DMA\_MemoryInc\_Enable;

DMA\_Init(DMA1\_Channel1, &DMA\_InitStructure);

DMA\_ClearITPendingBit(DMA1\_IT\_GL1);

DMA\_ITConfig(DMA1\_Channel1, DMA\_IT\_TC, ENABLE);

NVIC\_Init(DMA1\_Channel1\_IRQn);

DMA\_Cmd(DMA1\_Channel1, ENABLE);

TIM\_TimeBaseStructure.TIM\_Prescaler = SYSCLOCK/2-1; //分频系数

TIM\_TimeBaseStructure.TIM\_Period = 1;

TIM\_TimeBaseInit(TIM3, &TIM\_TimeBaseStructure); //初始化定时器3

TIM\_ARRPreloadConfig(TIM3, ENABLE); //允许TIM3定时重载

TIM\_SelectOutputTrigger(TIM3, TIM\_TRGOSource\_Update);

//选择TIM3的UPDATA事件为触发源

TIM\_Cmd(TIM3, ENABLE);

### 获得ADC采样值

由于ADC转换是通过DMA完成，每次DMA中断时，可以直接从ADC\_Value 数组内获取12位的AD转换结果。AD转换流程为定时器3产生Updata事件然后ADC转换一次，每次AD转换时间为7个ADC时钟周期，重复上述步骤600次后，得到AD转换结果600个然后产生DMA中断，在DMA中断内即可进行数据的处理和计算



图 4.5 ADC采样流程

## 波形绘制与数据显示的处理

### 获取ADC采样值并画出波形

ADC采样值由DMA进行存储，当ADC转换完成数量达到DMA设置的数量后，产生DMA中断，再DMA中断里，通过if(run\_sta==1)判断程序是否运行。如果条件不成立，表示按下了暂停键，后面的程序都不执行。如果程序运行，则进入DSO\_DrawWare();函数进行波形绘制，波形绘制时将ADC转换的全部结果从con\_t（计算频率时确定值）位置取出320个值。再将这320个12位AD转换结果通过计算变成屏幕上点的坐标，计算好了两个点时通过画线函数将两点连成一条线，并将点坐标保存到buff2数组里供清除上一次波形时用。整个逻辑流程如图 4.6所示



图 4.6绘制波形流程

绘制波形核心代码如下:

void DSO\_DrawWare(void)//画波形图

{

for(n = 0;n<lcddev.width;n++)

buff[n] = ADC\_Value[offset+con\_t+n];

for(n = 1;n < (lcddev.width-2);n++)

{

buff[n] = Ypos2 = ((lcddev.height-35) - (Yinit + (double)(buff[n] \*

0.0397\*avg)));//转换坐标

//清除上一帧显示波形的这一条线并画出当前波形对应的线

DSO\_DrawLine\_COLOR(n ,buff2[n] , n+1 ,buff2[n+1],BLACK);

DSO\_DrawLine\_COLOR(n ,Ypos1 , n+1 ,Ypos2,YELLOW);

Ypos1 = Ypos2 ;

}

for(n=1;n<lcddev.width;n++)

buff2[n]=buff[n-1];

}

在程序运行的时候，为了防止残影，在每次连线时都将上一帧波形这两点的连线清除[7]，再绘将当前波形数据两点连线，最后呈现出来的图案就是所要显示的波形。上一帧波形的坐标点信息存放在buff2数组里，通过这种方式绘制波形不会出现残影和闪屏。且刷新速度非常快，从而达到动态刷新的效果。从计算波形是找到的触发电平位置开始绘制波形，可以将波形的一个或者多个周期静态的显示在界面上方便观察，且市场上的商用级示波器都是这样的显示方式。

### 计算频率

计算频率时先设置一个触发电压，从采集到的数据中找到一个点，该点特征是该点前面比的点比触发电压小后面的点比触发电压大[8]，将这个点标记为con\_t，再按照相同的方法往后找下一个这样的点记为con\_t2，求出两点间隔多少个点，再乘以采样间隔timebase就可以求出周期，即可算出频率。

计算频率核心代码如下:

void DSO\_GetFreq(void)

{

for(n = 100;n<NCollect;n++)

if(ADC\_Value[n] < max\_data && ADC\_Value[n+2] > max\_data)

{

if(n>(NCollect-lcddev.width)) con\_t=100;

else con\_t=n;

break;

}

for(n = con\_t+3;n<NCollect;n++)

if(ADC\_Value[n] < max\_data&&ADC\_Value[n+2] > max\_data)

{

con\_t1=n;

break;

}

freq = (2000000.0/((con\_t1 - con\_t)\*timebase));

}

### 计算电压峰峰值

先将最大值和最小值的电压初始化为0和1.6V，使用for循环，通过比较得出从原点往下采样分别得到的最大值和最小值[9]，用最大值减去最小值所得到的差值再进行转化，最终就能获得电压的峰峰值。

void DSO\_GetVpp(void)

{

u16 max=0; //最大值初始化为0

u16 min=2000; //最小值初始化为1.6V

u16 n=0;

for(n = 1;n<lcddev.width;n++)

{

if(buff[n] > max) max = buff[n];

if(buff[n] < min) min = buff[n];

}

Vpp = (float)(max - min)\*(3.3/4096.0); //将12位AD结果转化为电压值

}

## 按键功能

开发板按键WK\_UP 、KEY0 、KEY1分别对应的功能是 切换功能选项(长按暂停)、增加（放大）、减小（缩小）。按键通过外部中断的方式通知CPU进行处理，其中WK\_UP按键IO口默认下拉所以上升沿触发，其余都为下降沿触发，中断抢占优先级为WK\_UP> KEY0>KEY1。

void EXTI0\_IRQHandler(void)

{

u8 i=0,status=0;

delay\_ms(10); //消抖

if(WK\_UP==1)

{

for(;i<20;++i)

{

if(!WK\_UP)status=1;

delay\_ms(10);

}

if(status!=1)run\_sta=!run\_sta;

else key\_num++;

}

EXTI\_ClearITPendingBit(EXTI\_Line0); //清除EXTI0线路挂起位

}

WK\_UP按键刚按下时每隔10ms检测一次按下状态，当按下时间超过200ms则认为是在长按，此时将变量run\_sta取反，达到暂停示波器运行的功能。如果没有长按达200ms则认为是在切换KEY0、KEY1的功能，此时将按键功能选项变量key\_num变量自增切换到下一功能。KEY0、KEY1按键中断内根据key\_num%3的值来选择相应增加或减小的参数。

KEY0中断核心代码如下，KEY1类似:

void EXTI9\_5\_IRQHandler(void)

{

if(KEY0==0)

{

if(key\_num%KEYNUM==0)

{

DSO\_SetTimeBase(--timN%TIMEBASENUM);

}

else if(key\_num%KEYNUM==1)

{

DSO\_OffsetIncrement(INCREASE);

}

else if(key\_num%KEYNUM==2)

{

DSO\_AvgIncrement(INCREASE);

}

}

EXTI\_ClearITPendingBit(EXTI\_Line5); //清除LINE5上的中断标志位

}

## 波形显示的变化

波形的变化主要是X轴（时间轴）和Y轴（振幅）的变化，本设计中采用定时器3更新事件的产生时间来控制ADC采样周期从而达到改变时间轴的目的，采样时间间隔越长，波形越密集，相当于放大X轴的可视范围。

振幅的变化和X轴变化同理，在坐标转换的时候，将Y轴乘上一个数值表示其放大倍数。设置一个限幅的if 条件判断，不能超过屏幕的范围。

# 测试实验

## 实验平台

系统在以下环境进行功能测试：

（1）软件环境：Keil uVision5,等；

（2）硬件环境：正点原子ALIENTEK MiniSTM32 V3.41开发板、波形发生器YB32020、2.8寸TFTLCD液晶显示屏。

## 测试方法

首先将使用J-link将编写好的代码写入ALIENTEK MiniSTM32 V3.41板中，使用波形发生器YB32020输出1kHz，3V的正弦波、方波和三角波，和1kHz,1V的三角波，将信号发生器输出通过杜邦线接到PA1口即可测试。

## 测试数据、图片结论

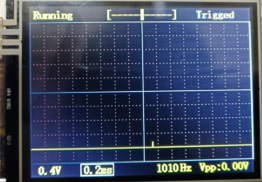
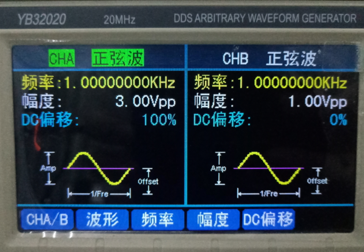
 

图 5.1启动界面 图 5.2示波器基本界面

示波器启动界面如图 5.1所示，启动后的界面如图 5.1所示，其中左上角为示波器运行状态，中上方为波形水平移动标尺，右上为采样状态，右下为纵向每格电压，其后是每个格的时基，右下为测得频率和峰峰值。

将信号发生器YB32020设置输出正弦波，频率1KHz，幅度3.00Vpp，DC偏移100%测试效果如图 5.3所示



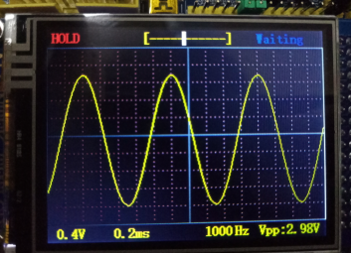
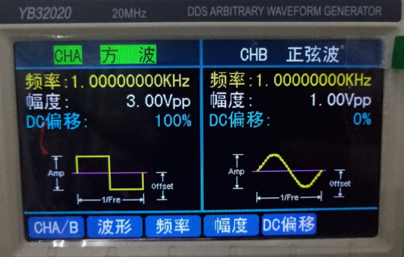


图 5.3正弦波输出和测试结果

图 5.3中测得正弦波频率为1000Hz，峰峰值Vpp为2.98V基本与信号发生器输出波形一致。波形纵向约占7.45格，峰峰值7.45\*0.4V=2.98V，横向一个周期占5格，周期5\*0.2ms=1ms。满足课程设计要求。

将信号发生器YB32020设置输出方波，频率1KHz，幅度3.00Vpp，DC偏移100%测试效果如图 5.4所示



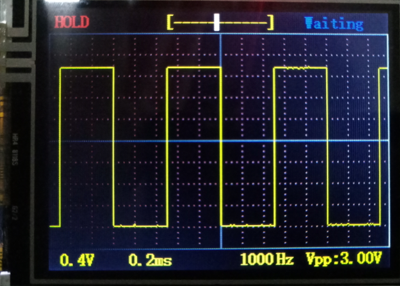


图 5.4方波输出和测试结果

图 5.4中测得方波频率为1000Hz，峰峰值Vpp为3.00V基本与信号发生器输出波形一致。波形纵向约占7.5格，峰峰值7.5\*0.4V=3.00V，横向一个周期占5格，周期5\*0.2ms=1ms。满足课程设计要求。

将信号发生器YB32020设置输出三角波，频率1KHz，幅度3.00Vpp，DC偏移100%测试效果如图 5.5所示。修改示波器水平时基后如图 5.6



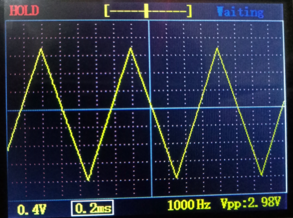


图 5.5三角波输出和测试结果

图 5.5中测得三角波频率为1000Hz，峰峰值Vpp为2.98V基本与信号发生器输出波形一致。波形纵向约占7.45格，峰峰值7.45\*0.4V=2.98V，横向一个周期占5格，周期5\*0.2ms=1ms

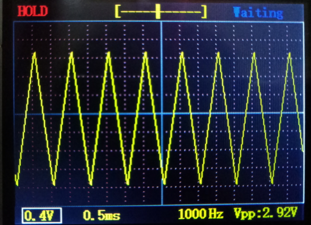


图 5.6将水平时基修改为0.5ms后效果

将信号发生器YB32020设置输出三角波，频率1KHz，幅度1.00Vpp，DC偏移100%测试效果如图 5.7所示。经过纵向缩放后的效果如图 5.8



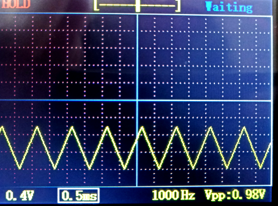


图 5.7方波输出和测试结果

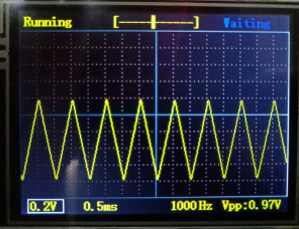
图 5.7中测得三角波频率为1000Hz，峰峰值Vpp为0.98V基本与信号发生器输出波形一致。波形纵向约占2.45格，峰峰值2.45\*0.4V=0.98V。

图 5.8缩放和水平移动后的三角波效果