简易频率特性测试仪（E题）

**摘要：** 该简易频率特性测试仪采用STM32F103ZET6核心板，主要由自制正交扫描信号源、RLC被测网络、乘法器、低通滤波等功能模块组成；测试数据包含信号的幅频特性及相频特性，于彩屏TFT上显示。本系统利用高速DDS（AD9854）模块实现信号发生电路，能输出双端口的正交信号，并可控制设置其幅值与频率。

被测网络采用RLC串联谐振电路，达到要求的网络通带中心频率。将DDS产生的正交信号和被测网络信号通过乘法器AD835模块，经LTC1564低通滤波，利用单片机上的片上AD进行采样，最终在彩屏上显示输出，完成对信号的幅频和相频的简单测试。

**关键词：** STM32；DDS；乘法器；滤波；TFT

**Abstract:** The simple frequency tester is based on high-performance STM32F103ZET6 controller, and it mainly consists of self-maid signal generator, RLC network, multipliers, low-pass filter and other function blocks. The test data contains amplitude-frequency characteristics and phase-frequency characteristics, and will be displayed on TFT. The system uses high-speed DDS (AD9854) block as signal circuit, generating the dual-port quadrature signals, whose amplitude and frequency can be controlled and set. The test network is made of RLC circuit, which is able to achieve the standardized center frequency. The quadrature signals generated by DDS and the signals under test pass through the multiplier AD835 and the LTC1564 low-pass filter, and will be sampled by the 12-bit on-chip ADC. The final output is displayed on the TFT to complete the test of amplitude-frequency and phase-frequency.

**Keyword:** STM32; DDS; multiplier; filter; TFT

一、方案选择与论证**·······································3**

1.1 正交扫频信号源的设计与选择·································3

1.2 乘法器模块的设计与选择·····································3

1.3 低通滤波器的设计与选择·····································3

1.4 ADC 模块的设计与选择·······································3

1.5 被测网络的设计与选择·······································4

二、电路设计·············································4

2.1 正交扫描信号源设计电路·····································4

2.2 模拟乘法器设计电路·········································4

2.3 低通滤波器设计电路·········································4

2.4 末级电压抬升设计电路·······································5

三、系统设计·············································5

3.1 方案总体设计描述···········································5

3.2 理论分析与计算·············································6

3.2.1 正交扫描信号发生器的设计·······························6

3.2.2 低通滤波器的设计·······································6

3.2.3 RLC 被测网络的设计·····································6

3.2.4 特性曲线显示···········································7

四、软件设计·············································7

五、系统测试·············································8

5.1 测试方法··················································8

5.2 测试工具··················································8

5.3 各模块测试过程············································8

5.3.1 扫频信号发生器的测试··································8

5.3.2 乘法器模块的测试······································9

5.3.3 输入输出阻抗的测试····································9

5.3.4 RLC 被测网络的测试····································9

六、结论················································10

参考文献·················································10

附录·····················································10

附录一：元件清单··················································10

附录二：AD9854扫描信号发生器设计电路······························11

附录三：系统总体设计电路···········································11

附录四：主要程序清单···············································12

1. 方案选择与论证

**1.1 正交扫频信号源的设计与选择**

方案一：采用程控锁相环频率合成方案。

方案说明：锁相环频率合成是将高稳定度和高精度的标准频率经过算术运算产生同样稳定度和精确度的大量离散频率，在一定程度上解决了既要频率稳定精确，又要频率在较大范围内可变的问题。但该方案采用多次积分电路，这种具有惰性特性的电路误差大，不能满足幅频曲线和相频曲线的输出要求。

方案二：采用AD9854芯片搭建DDS模块电路，通过控制AD9854寄存器可产生编程控制的、高精度的频率信号。

方案说明：AD9854是AD公司生产的DDS专用芯片，其拥有300MHz的内部时钟频率，可以进行FSK、BPSK、PSK、chirp、AM等的操作。 AD9854数字合成器是一个利用DDS技术、两个内部高速高性能正交DACs控制数字可编程输入输出的综合器件。当给AD9854加上一个确定时钟时，它可以生成高度稳定、频率-相位-幅度可编程的正弦信号，频率分辨率很高、抗干扰能力强、灵敏度高、实用性强。

综上论证比较，我们选用方案二作为正交扫频信号源的实现方式。

**1.2 乘法器模块的设计与选择**

方案一：采用晶体管等分立元件搭建差分对模拟乘法器电路，通过两个晶体管输出的电压乘积项达到频率变换作用。

方案说明：电路结构冗繁复杂，稳定性差，不易调节。

方案二：直接采用模拟乘法器AD835芯片。

方案说明：AD835是一款电压输出型四象限模拟乘法器，带宽高达250MHz，很适合宽带调制和解调应用。且AD835需要的外围电路非常少，配置方便。

综上论证比较，我们选用方案二作为模拟乘法器的实现方法。

**1.3 低通滤波器的设计与选择**

方案一：采用分立元件，搭建LC低通滤波器。

方案说明：虽然电路结构简单，但精度不高，稳定性差。

方案二：采用高精低噪运算放大器OP07搭建截止频率为5KHz的二阶有源低通滤波器。

方案说明：可利用滤波器专用设计软件Filter Solutions设计，得到二阶低通有源滤波器电路的连接图和频率响应曲线。

方案三：采用LTC1564芯片搭建截止频率为10KHz的十阶低通滤波器。

方案说明：LTC1564是凌力尔特公司的数控滤波器芯片，可采用经典10KHz的十阶低通滤波器电路结构，设计简单，滤波效果非常好，稳定性高。

综上论证比较，我们选用方案三作为低通滤波器的实现方法。

**1.4 ADC模块的设计与选择**

方案一：采用ADC芯片搭建模数转换电路。

方案说明：可选取合适的ADC芯片，采用经典模块电路搭建。

方案二：直接利用STM32片上的12位高速ADC模块。

方案说明：本系统选用的核心板STM32F103ZET6上，自带3个12位模数转换器，且转换时间可达1us，多达21个输入通道。

由于单片机上的ADC已可以达到实验采样要求，故我们选用方案二作为ADC模块的设计方法，可减少外部电路的焊接。

**1.5被测网络的设计与选择**

被测网络按照实验要求采用RLC串联谐振网络设计，图示如下：

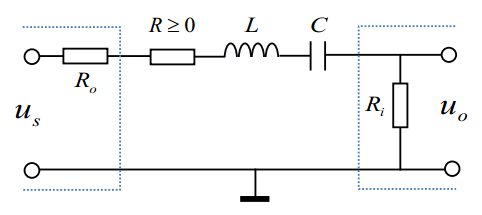


图1.1 RLC被测网络电路图

RLC被测网络的元器件值详见电路理论分析。

1. 电路设计

**2.1 正交扫描信号源设计电路**

正交扫描信号源采用AD9854芯片设计，电路详见附图。

**2.2模拟乘法器设计电路**

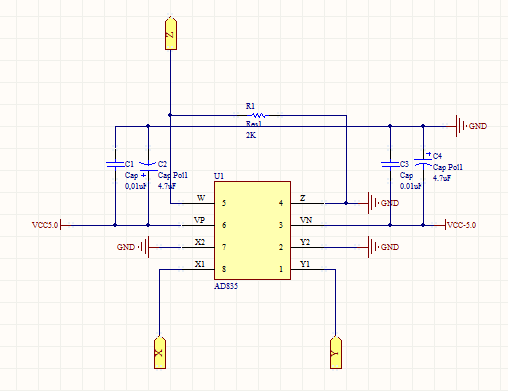
模拟乘法器采用AD835芯片设计，电路如下：

图2.1 乘法器电路

**2.3低通滤波器设计电路**

低通滤波器采用LTC1564芯片设计，搭建构成十阶的可达10KHz截止频率的电路，图示如下：

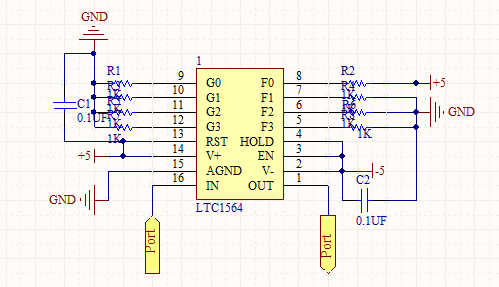


图2.2 低通滤波器电路

**2.4末级电压抬升设计电路**

由于本系统设计检测被测网络信号为正弦波形，电压有负值，接入ADC采样前设计一电压抬升电路，图示如下：

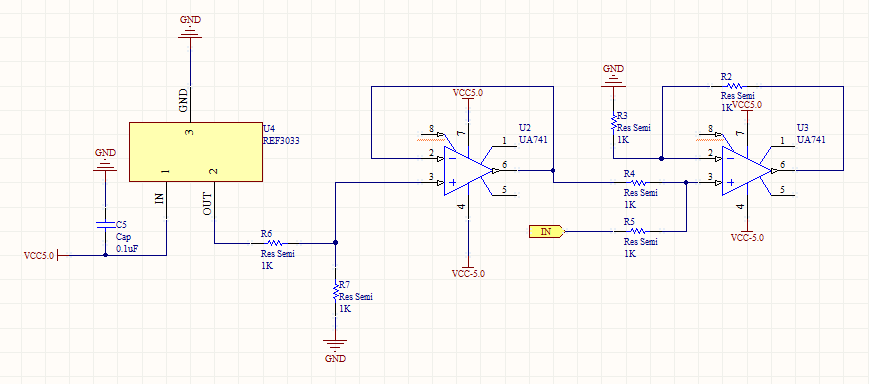


图2.3 末级电压抬升电路

1. 系统设计

**3.1 方案总体设计描述**

本方案首先设计一个正交扫描信号源，可产生相位互为正交的双端口正弦信号，可设置信号频率的步进单位。制作一个RLC串联谐振电路作为被测网络，且网络通带中心频率为20Mhz。自制信号源产生的两路正弦信号和被测网络信号通过乘法器模块，由低通滤波模块滤除高频分量，得到被测网络的频率，由ADC采样信号并输出显示。

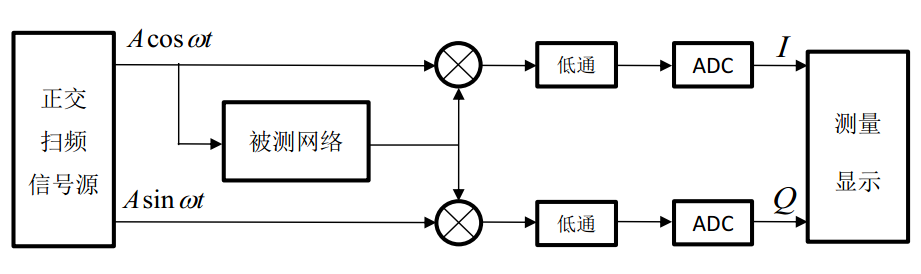


图3.1 简易频率特性测试仪系统原理示意图

该方案在硬件电路设计上主要包含了AD9854正交扫描信号发生器、AD835模拟乘法器、LTC1564低通滤波器、末级抬升电压电路及RLC串联谐振被测网络五个模块，通过逐个模块调试至整个系统电路连贯调试，完成基本测试实验要求。在软件设计上利用了ARM编程工具和STM32核心开发板，软件主要包含了对AD9854、串口通信、片上ADC采样及彩屏控制部分。其中，ADC模块采用STM32板上自带12位高速ADC，减少了部分硬件电路的设计，容易实现。

**3.2理论分析与计算**

3.2.1 正交扫描信号发生器的设计

本设计的扫描信号发生器以AD9854芯片采用DDS技术，结合单片机程控，以产生两路正交正弦信号。

设计中设置AD9854为单音模式，不仅可以调制所需频率的频率源信号，而且可根据需要设置输出信号的幅度、频率和相位等，即通过改变ROM表输出的幅度控制字实现幅度控制，通过改变相位累加器输入端的频率控制字实现频率控制，通过改变相位累加器输出端的相位控制字实现相位控制，从而完成相应的幅度、频率和相位的调制。

设相位累加器的位数为N，相位控制字的值为FN，频率控制字的位数为M，频率控制字的值为FM，AD9854内部工作时钟为FC，则最终合成信号的频率可由式（3-1）来决定，合成信号的相位由式（3-2）来决定。

(3-1)

(3-2)

当AD9854的参考时钟频率为270MHz时，

其频率分辨率计算：

相位控制精度计算：

幅度控制范围计算：

得到频率分辨率接近，相位控制精度达0.022，幅度控制范围为72dB，完全可达到实验要求。

3.2.2 低通滤波器的设计

设计中为得到被测网络信号的频率特性，需要将经过乘法器后的高频分量滤除，由于被测网络中心频率在20MHz，故我们选用凌力尔特公司的LTC1564芯片，采用经典电路，通过Multisim软件搭建电路并进行仿真调试，设计截止频率为10KHz的低通滤波器。

3.2.3 RLC被测网络的设计

RLC串联电路由电阻R、电感L及电容C串联构成，利用Multisim软件设计被测网络电路。

反映RLC电路频率特性的参数有谐振频率f0、通频带宽BW和品质因数Q，其定义如下：

（3-3）

（3-4）

(3-5)

其中，计算通频带式（3-4）中的和分别是回路电流由最大值增加和减少3dB时所对应的上限频率和下限频率；计算品质因数式（3-5）中的。实验要求电路的中心频率为20MHz，即=20MHz，要求品质因数为4，可得到电路的通频带宽为5MHz。

3.2.4 特性曲线显示

1）幅频特性曲线

采用STM32片上ADC芯片，通过DMA通道测量信号电压值，当ADC转换结束以后，读取ADC\_DR寄存器中的结果，通过下面的公式转换电压：

（3-6）

设系统输出的两路正交信号为：

（3-7）

（3-8）

经ADC双通道采样后，电压幅值计算：

（3-9）

据经电压转换后，计算电压增益，公式如下：

（3-10）

在彩屏上显示幅频特性曲线，以线性频率（Hz）为横坐标，以电压增益（dB）为纵坐标，显示出被测信号的幅频特性曲线。

2）相频特性曲线

两路正交信号经ADC双通道采样后，相位计算：

（3-11）

由于信号相位计算需要函数算法，比较复杂，故采用查表法程序。在彩屏上显示相频特性曲线，以线性频率（Hz）为横坐标，以相移（）为纵坐标，显示出被测信号的相频特性曲线。

1. 软件设计

在本设计中，由STM32F103ZET6核心板实现程序的控制。其中，软件主要完成了AD9854的信号发生控制、Usart串口通信、ADC双通道采样、彩屏TFT显示的功能。系统框图及流程如下：

开始

STM2初始化

STM1初始化

彩屏初始化显示开机界面

控制AD9854扫描信号发生器

Uasrt串口通信

ADC双通道采样

结束

彩屏TFT显示特性曲线

图4.1 系统软件流程框图

1. 系统测试

**5.1 测试方法**

本系统的测试过程中，模拟电路采用由前端到后端，数字电路采用先仿真再调试的方法，从逐个模块测试完成再到模块连接调试，直至整个系统电路完成测试要求。

**5.2 测试工具**

1）直流稳压源

2) 安捷伦数字示波器

3）函数信号发生器

4）数字频谱仪

5）数字万用表

**5.3 各模块测试过程**

5.3.1 扫频信号发生器的测试

使用直流稳压源对AD9854制成的扫描信号发生器进行正负5V供电，可承载电流不超过1A，将扫描信号发生器连接单片机，下载控制程序，通过数字示波器显示输出的I路及Q路信号，现列出一组典型实验数据。

表5.1 扫描信号发生器测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 理论电压（Vpp/V） | 实际电压（Vpp/V） | 理论频率（f/MHz） | 实际频率（f/MHz） |
| I路信号 | 2.00 | 1.98 | 20.0 | 20.0 |
| Q路信号 | 2.00 | 1.96 | 20.0 | 20.0 |

I、Q两路正弦信号呈正交关系，示波器上测量计算两路信号相位呈90度。输出电压和频率误差小于5%。通过频谱仪接入电路输出信号，扫描信号发生器完全可产生1M-40MHz范围的信号，精度可达10-4；且频率步进可调，达100KHz精度。

5.3.2 乘法器模块的测试

利用数字函数发生器产生两路正交正弦信号，接入乘法器模拟电路，现列出一组典型信号测试结果：

两路输入电压峰峰值均为Vp-p=2.00V，频率均为F=20MHz。

表5.2 乘法器测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 理论电压（Vpp/V） | 实际电压（Vpp/V） | 理论频率（f/MHz） | 实际频率（f/MHz） |
| I路信号 | 2.00 | 1.98 | 20.0 | 20.0 |
| Q路信号 | 2.00 | 2.00 | 20.0 | 20.0 |
| 输出信号 | 1.00 | 0.98 | 40.0 | 39.9 |

综上数据可看出，模拟乘法器模块工作正常，输出信号误差小于2%。

5.3.3 输入输出阻抗的测试

利用数字万用表测试该系统的输入阻抗和输出阻抗，即测量被测网络两端的Ri和RO，可达50Ω。

5.3.4 RLC被测网络的测试

连接RLC被测网络，利用调节频率法测量电路的谐振频率。电路输入端接入数字函数发生器，保持信号源输出电压US不变，改变信号发生器的频率，将电路输出端接入数字频谱仪观察其输出的电压值。当输出电压UR的读数达到最大值（即电流达到最大值）时，其所对应的频率值即为谐振频率或中心频率。

设置数字函数发生器输出电压为2.0V，调节信号频率，取一组调节频率法测试结果列示如下：

表5.3 RLC被测网络测试结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 信号源频率（f/MHz） | 17.0 | 18.0 | 19.0 | 20.0 | 21.0 | 22.0 | 23.0 |
| 测量电压（U0/V） | 1.32 | 1.58 | 1.80 | 1.98 | 1.78 | 1.54 | 1.30 |

由上表数据可得，RLC网络的中心频率为20MHz，误差小于5%；且其通频带宽约在17.5MHz-22.5MHz范围内，品质因数约4。

1. 结论

本设计很好的满足了题目要求，对各模块和整体系统的测试基本完成。在设计中对于由AD9854搭建的扫描信号发生器花费了很多调试时间。通过本次设计，我们不仅熟练掌握各种仪器设备的使用调节，同时很大程度上提高了动手能力和处理问题的能力，受益匪浅！

参考文献

1. 黄智伟 全国大学生电子设计竞赛制作实训 北京航空航天大学出版社
2. 黄智伟 全国大学生电子设计竞赛技能训练 北京航空航天大学出版社
3. 谢嘉奎，宣月清. 电子线路（非线性部分） 高等教育出版社
4. 童诗白，华成英. 模拟电子技术基础（第四版） 高等教育出版社

附录

**附录一：元件清单**

本设计系统所用的主要元件清单如下所示：

序号 型号 器件封装 说明

1 AD835 DIP 乘法器

2 LTC1564 SSOP 8阶低通滤波器

3 REF3033 SSOP 3.3V基准电压

4 ua741 DIP 通用运放

5 OPA727 SSOP 通用运放

6 若干电阻电容

表7.1 系统元件清单列表

**附录二：AD9854扫描信号发生器设计电路**

AD9854的高速DDS模块硬件原理图如下所示：

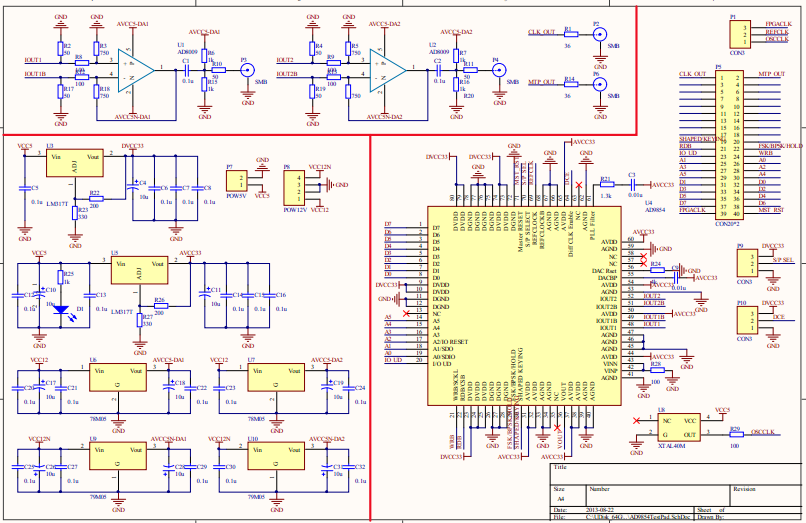


图7.1 AD9854扫描信号发生器硬件原理图

**附录三：系统总体设计电路**

本系统总体硬件设计原理图如下所示：

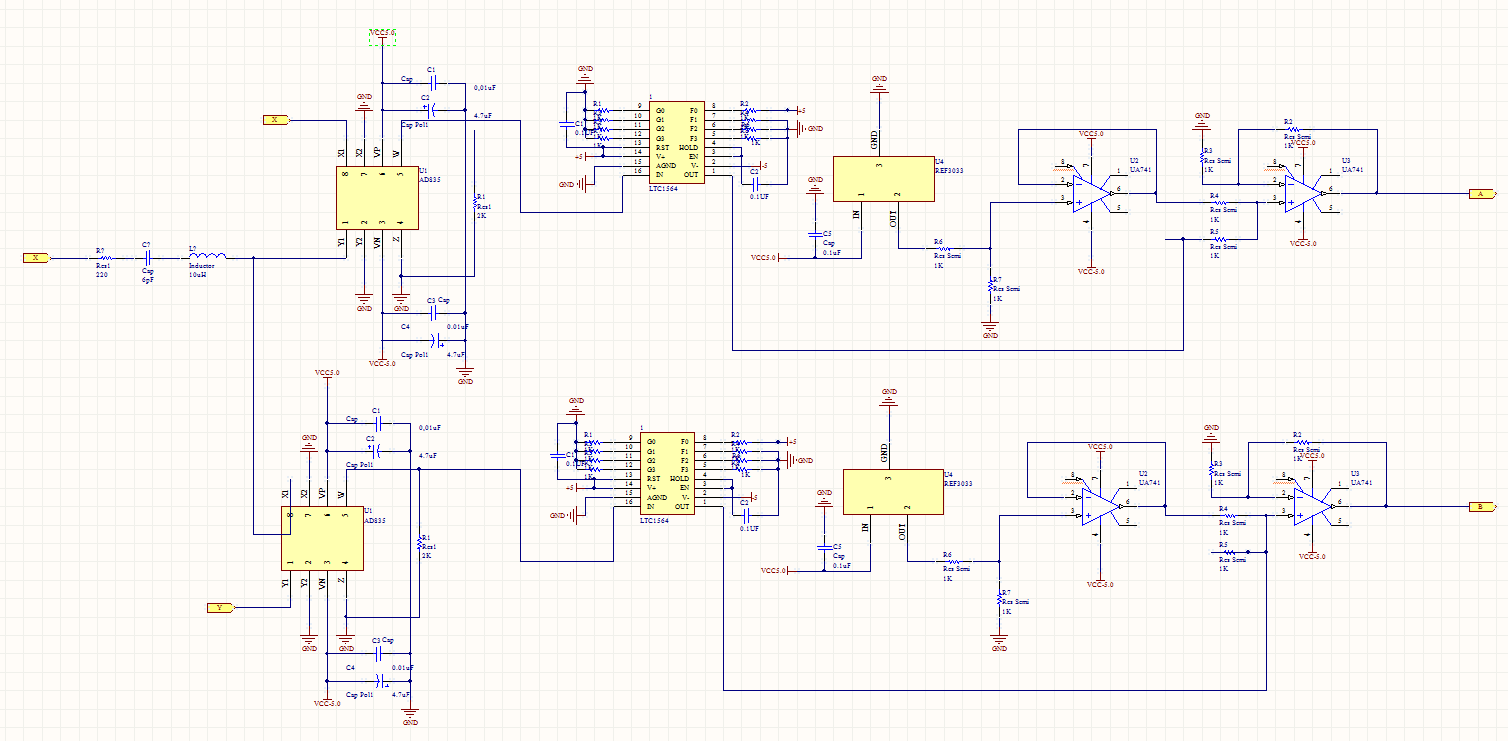


图7.2 系统总体硬件原理图

**附录四：主要程序清单**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

AD9854控制程序：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SetAD9854Frequency(uchar ucFreWordAdd,ulong ulFreqVal)

{

uchar FreqWord[6];

uchar Count=6,Adress;

ulong FreqBuf;

Adress = ucFreWordAdd;

FreqBuf = (ulFreqVal\*FRETMP\_Low);

FreqWord[0] = (FreqBuf%0x100)&0xff;

FreqBuf /= 0x100;

FreqBuf += (ulFreqVal\*FRETMP\_MIDDLE);

FreqWord[1] = (FreqBuf%0x100)&0xff;

FreqBuf /= 0x100;

FreqBuf += (ulFreqVal\*FRETMP\_HIGH);

FreqWord[2] = (FreqBuf%0x100)&0xff;

FreqBuf /= 0x100;

FreqWord[3] = (FreqBuf%0x100)&0xff;

FreqBuf /= 0x100;

FreqWord[4] = (FreqBuf%0x100)&0xff;

FreqWord[5] = (FreqBuf/0x100)&0xff;

for( Count=6; Count;)

{

AD9854\_WriteByte(Adress++,FreqWord[--Count]);

AD9854\_WR\_0;

AD9854\_WR\_1;

}

AD9854\_UDCLK\_1;

AD9854\_UDCLK\_0;

}

void SetAD9854Amplitude(uchar ucChannel,uint uiAmpVal)

{

uchar AmpWord[2];

uint AmpBuf;

AmpWord[0] = (u8)(((u16)(uiAmpVal\*4096/5))%256);

AmpWord[1] = (u8)(((u16)(uiAmpVal\*4096/5))/256);

if( ucChannel==I )

{

AD9854\_WR\_0;

AD9854\_WR\_1;

AD9854\_WR\_0;

AD9854\_WR\_1;

}

if( ucChannel==Q )

{

AD9854\_WR\_0;

AD9854\_WR\_1;

AD9854\_WR\_0;

AD9854\_WR\_1;

}

AD9854\_UDCLK\_1;

AD9854\_UDCLK\_0;

}

void SetAD9854Phase(uchar ucPhaAdjRegAdd,uint uiPhaVal)

{

uchar PhaWord[2];

uint PhaBuf;

PhaBuf = (uiPhaVal\*0x2d);

PhaWord[0] = (PhaBuf%0x100)&0xff;

PhaWord[1] = (PhaBuf/0x100)&0xff;

AD9854\_WriteByte(ucPhaAdjRegAdd,PhaWord[1]);

AD9854\_WR\_0;

AD9854\_WR\_1;

AD9854\_WriteByte(ucPhaAdjRegAdd+1,PhaWord[0]);

AD9854\_WR\_0;

AD9854\_WR\_1;

AD9854\_UDCLK\_1;

AD9854\_UDCLK\_0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Uarst串口初始化程序：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void USART1\_Configuration(void)

{

USART\_InitTypeDef USART\_InitStructure;

USART\_ClockInitTypeDef USART\_ClockInitStructure;

USART\_InitStructure.USART\_BaudRate = 115200;

USART\_InitStructure.USART\_WordLength = USART\_WordLength\_8b;

USART\_InitStructure.USART\_StopBits = USART\_StopBits\_1;

USART\_InitStructure.USART\_Parity = USART\_Parity\_No;

USART\_InitStructure.USART\_HardwareFlowControl = USART\_HardwareFlowControl\_None;

USART\_InitStructure.USART\_Mode = USART\_Mode\_Rx | USART\_Mode\_Tx;

/\* Configure the USART1 \*/

USART\_Init(USART1, &USART\_InitStructure);

USART\_ClockInitStructure.USART\_Clock = USART\_Clock\_Disable;

USART\_ClockInitStructure.USART\_CPOL = USART\_CPOL\_Low;

USART\_ClockInitStructure.USART\_CPHA = USART\_CPHA\_2Edge;

USART\_ClockInitStructure.USART\_LastBit = USART\_LastBit\_Disable;

USART\_ClockStructInit(&USART\_ClockInitStructure);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ADC采样程序：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void DAC\_Configuration(void)

{

DAC\_InitTypeDef DAC\_InitStructure;

DAC\_Cmd(DAC\_Channel\_1,ENABLE);

DAC\_StructInit(&DAC\_InitStructure);

DAC\_InitStructure.DAC\_Trigger=DAC\_Trigger\_Software;

DAC\_InitStructure.DAC\_WaveGeneration=DAC\_WaveGeneration\_None;

DAC\_InitStructure.DAC\_LFSRUnmask\_TriangleAmplitude=DAC\_TriangleAmplitude\_4095;

DAC\_InitStructure.DAC\_OutputBuffer=DAC\_OutputBuffer\_Enable;

DAC\_Init(DAC\_Channel\_1,&DAC\_InitStructure);

DAC\_Cmd(DAC\_Channel\_1,ENABLE);

//DAC\_SetChannel1Data(DAC\_Align\_12b\_R, 0x0000);

DAC\_SoftwareTriggerCmd(DAC\_Channel\_1,ENABLE);

}

void ADC\_Configuration(void)

{

ADC\_InitTypeDef ADC\_InitStructure;

/\* Configure the ADC1 \*/

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

ADC\_InitStructure.ADC\_ScanConvMode = ENABLE;

ADC\_InitStructure.ADC\_ContinuousConvMode = ENABLE;

ADC\_InitStructure.ADC\_ExternalTrigConv = ADC\_ExternalTrigConv\_None;

ADC\_InitStructure.ADC\_DataAlign = ADC\_DataAlign\_Right;

ADC\_InitStructure.ADC\_NbrOfChannel = 1;

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

/\* ADC2 regular channel5 configuration \*/

ADC\_RegularChannelConfig(ADC1, ADC\_Channel\_3, 1, ADC\_SampleTime\_1Cycles5);

ADC\_DMACmd(ADC1,ENABLE);

/\* Enable ADC2 \*/

ADC\_Cmd(ADC1, ENABLE);

/\* Enable ADC2 reset calibaration register \*/

ADC\_ResetCalibration(ADC1);

/\* Check the end of ADC2 reset calibration register \*/

while(ADC\_GetResetCalibrationStatus(ADC1));

/\* Start ADC2 calibaration \*/

ADC\_StartCalibration(ADC1);

/\* Check the end of ADC2 calibration \*/

while(ADC\_GetCalibrationStatus(ADC1));

/\* Start ADC2 Software Conversion \*/

ADC\_SoftwareStartConvCmd(ADC1, ENABLE);

}

void DMA\_Configuration(void)

{

DMA\_InitTypeDef DMA\_InitStructure;

DMA\_DeInit(DMA1\_Channel1);

DMA\_InitStructure.DMA\_PeripheralBaseAddr = ADC1\_DR\_Address;

DMA\_InitStructure.DMA\_MemoryBaseAddr = (u32)&ADCConvertedValue;

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

DMA\_InitStructure.DMA\_BufferSize =1024;

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

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

DMA\_InitStructure.DMA\_PeripheralDataSize = DMA\_PeripheralDataSize\_HalfWord;

DMA\_InitStructure.DMA\_MemoryDataSize = DMA\_MemoryDataSize\_HalfWord;

DMA\_InitStructure.DMA\_Mode = DMA\_Mode\_Circular;

DMA\_InitStructure.DMA\_Priority = DMA\_Priority\_High;

DMA\_InitStructure.DMA\_M2M = DMA\_M2M\_Disable;

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

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

/\* Enable DMA1 channel1 \*/

DMA\_Cmd(DMA1\_Channel1, ENABLE);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TFT彩屏部分算法程序：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCD\_Writ\_Bus(char VH,char VL) //并行数据写入函数

{

datah\_change(VH); // LCD\_DataPortH=VH;

datal\_change(VL); // LCD\_DataPortL=VL;

GPIO\_ResetBits(LCD\_GRP, LCD\_WR); //LCD\_WR=0;

GPIO\_SetBits(LCD\_GRP, LCD\_WR); //LCD\_WR=1;

}

void Lcd\_Write\_Com(char VH,char VL) //发送命令-8位参数

{

GPIO\_ResetBits(LCD\_RS\_GRP, LCD\_RS);// LCD\_RS=0;

LCD\_Writ\_Bus(VH,VL);

}

void Lcd\_Write\_Data(char VH,char VL) //发送数据-8位参数

{

GPIO\_SetBits(LCD\_RS\_GRP, LCD\_RS);// LCD\_RS=1;

LCD\_Writ\_Bus(VH,VL);

}

void Lcd\_Write\_Com\_Data(int com,int val) //发送数据命令

{

Lcd\_Write\_Com(com>>8,com);

Lcd\_Write\_Data(val>>8,val);

}

void Address\_set(unsigned int x1,unsigned int y1,unsigned int x2,unsigned int y2)

{

Lcd\_Write\_Com(0x00,0x46);Lcd\_Write\_Data(x2,x1);

Lcd\_Write\_Com(0x00,0x47);Lcd\_Write\_Data(y2>>8,y2);

Lcd\_Write\_Com(0x00,0x48);Lcd\_Write\_Data(y1>>8,y1);

Lcd\_Write\_Com(0x00,0x20);Lcd\_Write\_Data(x1>>8,x1);

Lcd\_Write\_Com(0x00,0x21);Lcd\_Write\_Data(y1>>8,y1);

Lcd\_Write\_Com(0x00,0x22);

}

//画点

//POINT\_COLOR:此点的颜色

void LCD\_DrawPoint(unsigned int x,unsigned int y,unsigned int POINT\_COLOR) //把函数体内的xy调换

{

if(x>319)x=319;else if(x<0)x=0;

if(y>239)y=239;else if(y<0)y=0;

Address\_set(y,x,y,x);//设置光标位置

Lcd\_Write\_Data(POINT\_COLOR>>8,POINT\_COLOR);

}

//画线

//x1,y1:起点坐标

//x2,y2:终点坐标

void LCD\_DrawLine(unsigned int x1, unsigned int y1, unsigned int x2, unsigned int y2,iu16 color)

{

u16 t;

int xerr=0,yerr=0,delta\_x,delta\_y,distance;

int incx,incy,uRow,uCol;

delta\_x=x2-x1; //计算坐标增量

delta\_y=y2-y1;

uRow=x1;

uCol=y1;

if(delta\_x>0)incx=1; //设置单步方向

else if(delta\_x==0)incx=0;//垂直线

else {incx=-1;delta\_x=-delta\_x;}

if(delta\_y>0)incy=1;

else if(delta\_y==0)incy=0;//水平线

else{incy=-1;delta\_y=-delta\_y;}

if( delta\_x>delta\_y)distance=delta\_x; //选取基本增量坐标轴

else distance=delta\_y;

for(t=0;t<=distance+1;t++ )//画线输出

{

LCD\_DrawPoint(uRow,uCol,color);//画点

xerr+=delta\_x ;

yerr+=delta\_y ;

if(xerr>distance)

{

xerr-=distance;

uRow+=incx;

}

if(yerr>distance)

{

yerr-=distance;

uCol+=incy;

}

}

}

//在指定位置显示一个汉字

void showhanzi(iu16 x,iu16 y,iu16 kuan,iu16 gao,u8 \*add,iu16 dcolor,iu16 bgcolor)

{

u8 i,j;

iu16 cyc;

Address\_set(x,y,x+kuan-1,y+gao-1);

for(j=0;j<gao;j++)

{

for(cyc=0;cyc<kuan/8;cyc++)

{

for(i=0;i<8;i++)

{

if((\*add&(1<<(i)))!=0)

{

Lcd\_Write\_Data(dcolor>>8,dcolor);

}

else

{

Lcd\_Write\_Data(bgcolor>>8,bgcolor);

}

}

add++;

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TFT触屏算法程序（略写）：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//触摸屏校准代码

#define tp\_pianyi 50 //校准坐标偏移量

#define tp\_xiaozhun 1000 //校准精度

void Touch\_Adjust(void)

{

float vx1,vx2,vy1,vy2;

u16 chx1,chx2,chy1,chy2;//默认像素点坐标为0时的AD起始值

struct tp\_pixu32\_ p[4];

u8 cnt=0;

cnt=0;

u8 ss[6];

Pant(0,0,239,319,GRAY);//清屏

Drow\_Touch\_Point(tp\_pianyi,tp\_pianyi);//画点1

while(1)

{

if(GPIO\_ReadInputDataBit(LCD\_TOUCH\_GRP,Penirq)==0)//按键按下了

{

if(Read\_TP\_Once())

{

p[cnt].x=tp\_pixad.x;

p[cnt].y=tp\_pixad.y;

cnt++;

}

switch(cnt)

{

case 1: while(!GPIO\_ReadInputDataBit(LCD\_TOUCH\_GRP,Penirq))

vx1=p[1].x>p[0].x?(p[1].x-p[0].x+1)\*1000/(LCD\_H-tp\_pianyi-tp\_pianyi):(p[0].x-p[1].x-1)\*1000/(LCD\_H-tp\_pianyi-tp\_pianyi);

{

cnt=0;

Pant(0,0,239,319,GRAY);//清屏

Drow\_Touch\_Point(tp\_pianyi,tp\_pianyi);//画点1

continue;

}

vx=(vx1+vx2)/2;

vy=(vy1+vy2)/2;

showzifustr(36,60,16,32,char\_3216,"vx=",RED,GRAY);

inttostr((int)vx,ss);

Read\_TP\_Once();

return;

}

}

}

}