****

**“跳一跳”物理助手**

**设**

**计**

**报**

**告**

[摘 要 3](#_Toc21253)

[第一章、 方案比较、设计与论证 4](#_Toc1969)

[1.1方案优势与比较 4](#_Toc24784)

[1.1.1方案优势 4](#_Toc28088)

[1.1.2 方案比较 5](#_Toc17697)

[1.2方案设计与论证 5](#_Toc12212)

[1.2.1 设计意义 5](#_Toc7173)

[1.2.2 设计步骤 5](#_Toc6053)

[第二章、理论分析与计算 5](#_Toc21754)

[2.1 系统原理框图 5](#_Toc30350)

[2.2单片机IAP15W4K58S4原理 6](#_Toc1678)

[第三章、硬件部分 7](#_Toc13324)

[3.1 LCD1602液晶显示模块 7](#_Toc8197)

[3.1.1 LCD1602的简述 7](#_Toc6125)

[3.1.2 LCD1602的管脚 8](#_Toc28973)

[3.2 SYN6288语音合成模块 9](#_Toc1330)

[3.2.1 SYN6288 语音合成芯片IC引脚图 9](#_Toc453)

[3.2.2 SYN6288 语音合成芯片引脚定义 10](#_Toc23129)

[3.3 AD转换模块 12](#_Toc15648)

[3.3.1 AD转换原理 12](#_Toc2292)

[第四章、软件设计及流程图 12](#_Toc23535)

[4.1设计的软件环境简介 12](#_Toc18118)

[4.1.1 Keil\_c51 12](#_Toc1371)

[4.1.2 Altium Designer 软件 13](#_Toc3329)

[4.2设计流程图 15](#_Toc10701)

[4.3主程序设计 15](#_Toc32357)

[4.4液晶显示函数 25](#_Toc27926)

[4.5 语音函数 29](#_Toc21318)

[4.6 LCD1602.h头文件 32](#_Toc25312)

[第五章、测试方法与仪器，测试数据及测试结果分析 33](#_Toc8896)

[5.1 单片机的硬件仿真 33](#_Toc14215)

[5.2 AD采集数据包正确率测试 33](#_Toc23210)

[5.3 通过串口通讯助手调试串口通信模块 35](#_Toc28873)

[参考文献 36](#_Toc1030)

# 摘 要

本设计是将完成宏晶杯竞赛题目，实现基于单片机的AD采集模块、伺服模块、液晶显示模块、语音模块构成的微信小游戏“跳一跳”物理助手。

本设计以IAP15W4K58S4单片机为控制核心，分为四个模块构成。其中AD采集模块采用线性电阻RuilongMaker和单片机的AD采集功能构成，该模块实现了“距离-电阻差-电位差-时间差”的转换，然后伺服模块实现跳一跳的模拟操作，并通过液晶实时显示步数以及语音实时播报步数。

# 方案比较、设计与论证

## 1.1方案优势与比较

### 1.1.1方案优势：

我们的微信小游戏“跳一跳”物理助手有以下优势：

（1） 实现跳一跳过程的模拟操作，降低人工干预的误差。

（2） 采用高精度线性电阻，提高精度。

（3） 液晶屏实时显示跳动步数。

（4） 语音模块实时播报步数。

### 1.1.2 方案比较

采用直滑电位器测距有助于提高精确度。若采用其他方式，则精度不足。

## 1.2方案设计与论证

### 1.2.1 设计意义

该项目实现了微信小游戏“跳一跳”的模拟人工运行，并且精度高于大多数人的技术，可以实现“物理助手”的功能。而其语音播报和液晶显示的功能，也方便了用户的使用。

### 1.2.2 设计步骤

针对微信小游戏“跳一跳”物理助手，制定以下方案及步骤：

第一步，根据设计目的构想设计的原理图框架，学习设计中要用到的知识，如编码解码，单片机C语言编程设计，1602液晶显示，语音模块设计，使用的芯片引脚工作原理， Altium Designer软件使用，焊锡的使用等。

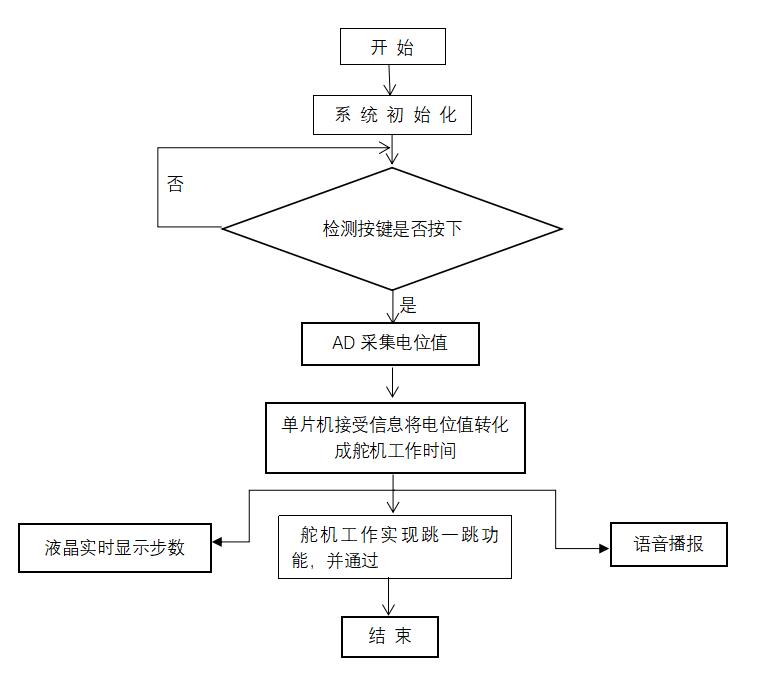
第二步，对硬件模块进行设计。如AD采集模块、线性电阻、1602液晶显示模块、语音模块。在Altium Designer中绘制原理图。

第三步，对系统软件进行设计。如主函数程序设计、初始化程序设计、延时子程序设计、液晶显示子程序设计、外部中断服务子程序设计、语音模块子程序设计、AD采集模块子程序设计。

# 第二章、理论分析与计算

## 2.1 系统原理框图

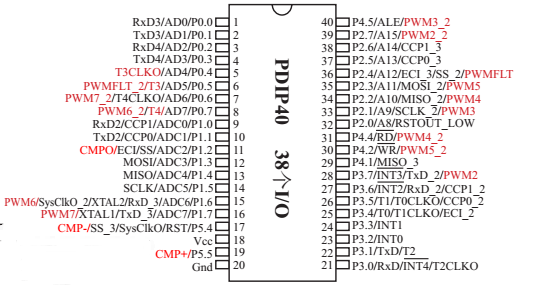
根据单片机的要求初步绘制出系统原理框图如图所示。



## 2.2单片机IAP15W4K58S4原理

IAP15W4K58S4系列单片机是单时钟/机器周期（1T）的单片机，是宽高压/高速/高可靠/低功耗/超强抗干扰的新一代8051单片机，采用第九代加密技术，无法解密，指令完全兼容传统8051，但速度快8-12倍。内部集成高精度R/C时钟，ISP编程时5MHz~35MHz宽范围可设置，可彻底省掉昂贵的晶振和外部复位电路。8路10位PWM，8路高速10位A/D转换（30万次/秒），内置4K字节大容量SRAM，4组独立的高速异步串行通信端口（UART1/URAT2/URAT3/URAT4），一组高速同步串行通信端口SPI，针对多串行口通信/电机控制/强干扰场合，内置比较强，功能更强大。

单片机IAP15W4K58S4管脚说明：



# 第三章、硬件部分

## 3.1 LCD1602液晶显示模块

### 3.1.1 LCD1602的简述

LCD1602是一种工业字符型液晶，能够同时显示16x02即32个字符。LCD1602液晶显示的原理是利用液晶的物理特性，通过电压对其显示区域进行控制，即可以显示出图形。

简介：工业字符型液晶，能够同时显示16x02即32个字符。（16列2行）

注：为了表示的方便 ，后文皆以1表示高电平，0表示低电平。

1602液晶也叫1602字符型液晶，它是一种专门用来显示字母、数字、符号等的点阵型液晶模块。它由若干个5X7或者5X11等点阵字符位组成，每个点阵字符位都可以显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此所以它不能很好地显示图形（用自定义CGRAM，显示效果也不好）。

1602LCD是指显示的内容为16X2,即可以显示两行，每行16个字符液晶模块（显示字符和数字）。

市面上字符液晶大多数是基于HD44780液晶芯片的，控制原理是完全相同的，因此基于HD44780写的控制程序可以很方便地应用于市面上大部分的字符型液晶。

### 3.1.2 LCD1602的管脚

1602采用标准的16脚接口，其中：

第1引脚：GND为电源地

第2引脚：VCC接5V电源正极

第3引脚：V0为液晶显示器对比度调整端，接正电源时对比度最弱，接地电源时对比度最高（对比度过高时会 产生“鬼影”，使用时可以通过一个10K的电位器调整对比度）。

第4引脚：RS为寄存器选择，高电平1时选择数据寄存器、低电平0时选择指令寄存器。

第5引脚：RW为读写信号线，高电平(1)时进行读操作，低电平(0)时进行写操作。

第6引脚：E(或EN)端为使能(enable)端,高电平（1）时读取信息，负跳变时执行指令。

第7～14引脚：D0～D7为8位双向数据端。

第15～16脚：空脚或背灯电源。第15引脚背光正极，第16引脚背光负极。

特性：3.3V或5V工作电压，对比度可调，内含复位电路，提供各种控制命令,如：清屏、字符闪烁、光标闪烁、显示移位等多种功能，有80字节显示数据存储器DDRAM，内建有192个5X7点阵的字型的字符发生器CGROM，8个可由用户自定义的5X7的字符发生器CGRAM

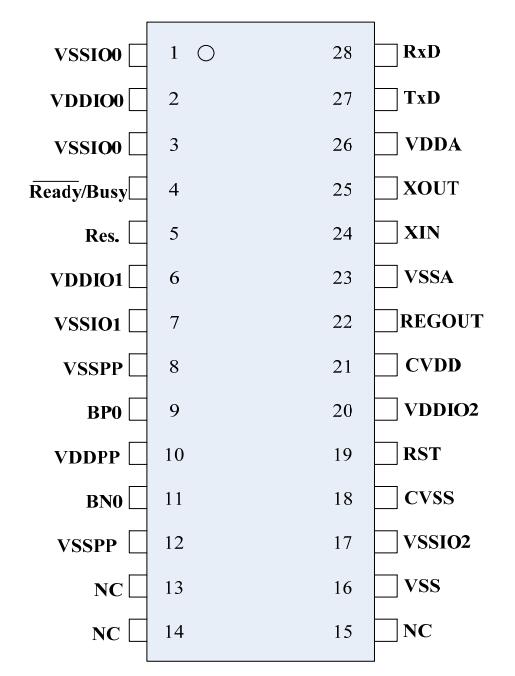
特征应用：微功耗、体积小、显示内容丰富、超薄轻巧，常用在袖珍式仪表和低功耗应用系统中。

## 3.2 SYN6288语音合成模块

### 3.2.1 SYN6288 语音合成芯片IC引脚图

SYN6288语音合成芯片具有28个引脚

其引脚图如下：



### 3.2.2 SYN6288 语音合成芯片引脚定义

SYN语音合成芯片具有28个引脚，各引脚具有不同的含义

其引脚定义图如下：



## 3.3 AD转换模块

### 3.3.1 AD转换原理

AD转换，即模数转换。顾名思义，就是把模拟信号转换成数字信号。主要包括积分型、逐次逼近型、并行比较型/串并行型、Σ-Δ调制型、电容阵列逐次比较型及压频变换型。

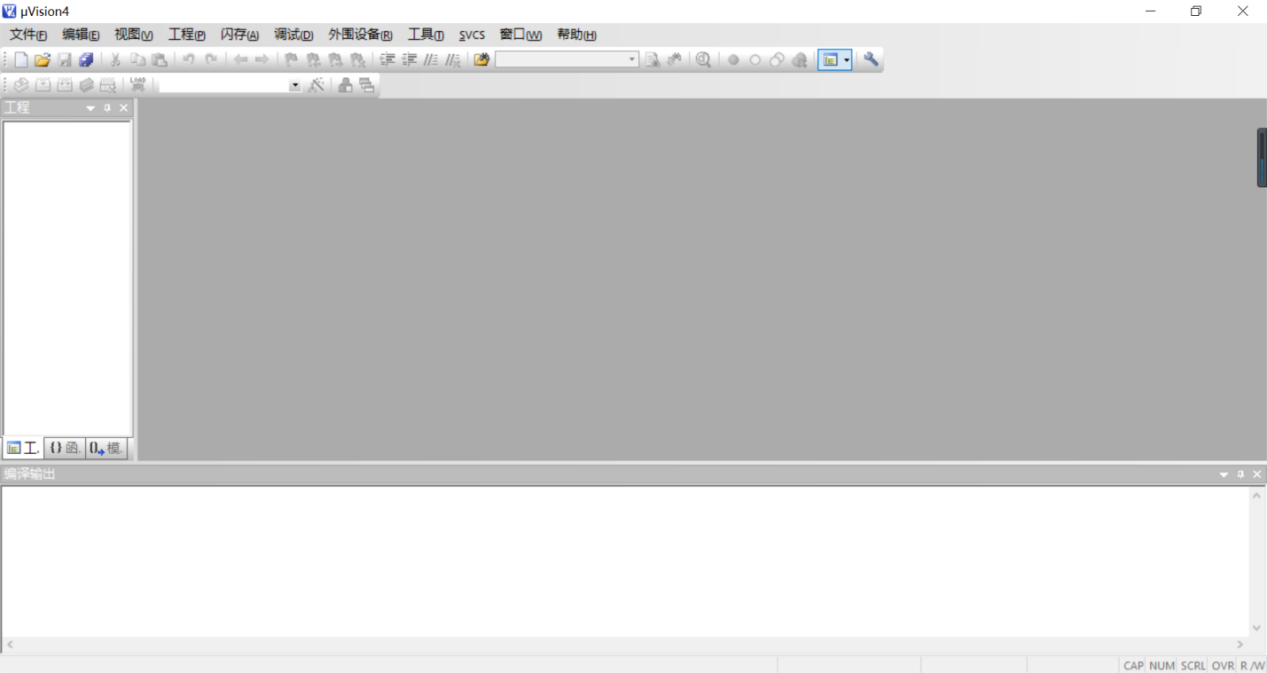
A/D转换器是用来通过一定的电路将模拟量转变为数字量。模拟量可以是电压、电流等电信号，也可以是压力、温度、湿度、位移、声音等非电信号。但在A/D转换前，输入到A/D转换器的输入信号必须经各种传感器把各种物理量转换成电压信号。

# 第四章、软件设计及流程图

## 4.1设计的软件环境简介

### 4.1.1 Keil\_c51

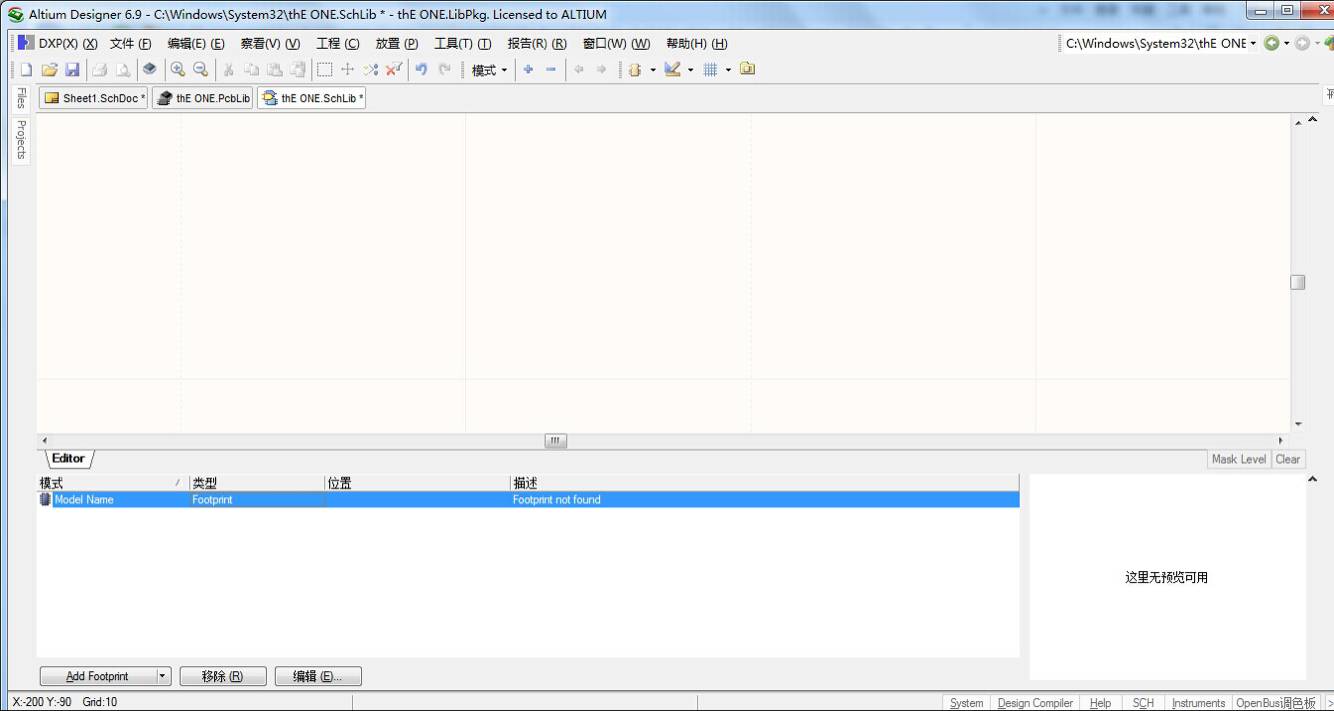
Keil C51是美国Keil Software公司出品的51系列兼容单片机C语言软件开发系统，与汇编相比，C语言在功能上、结构性、可读性、可维护性上有明显的优势，因而易学易用。用过汇编语言后再使用C来开发，体会更加深刻。 Keil C51软件提供丰富的库函数和功能强大的集成开发调试工具，全Windows界面。另外重要的一点，只要看一下编译后生成的汇编代码，就能体会到Keil C51生成的目标代码效率非常之高，多数语句生成的汇编代码很紧凑，容易理解。在开发大型软件时更能体现高级语言的优势。下面详细介绍Keil C51开发系统各部分功能和使用。Keil软件界面如图3-1-1：



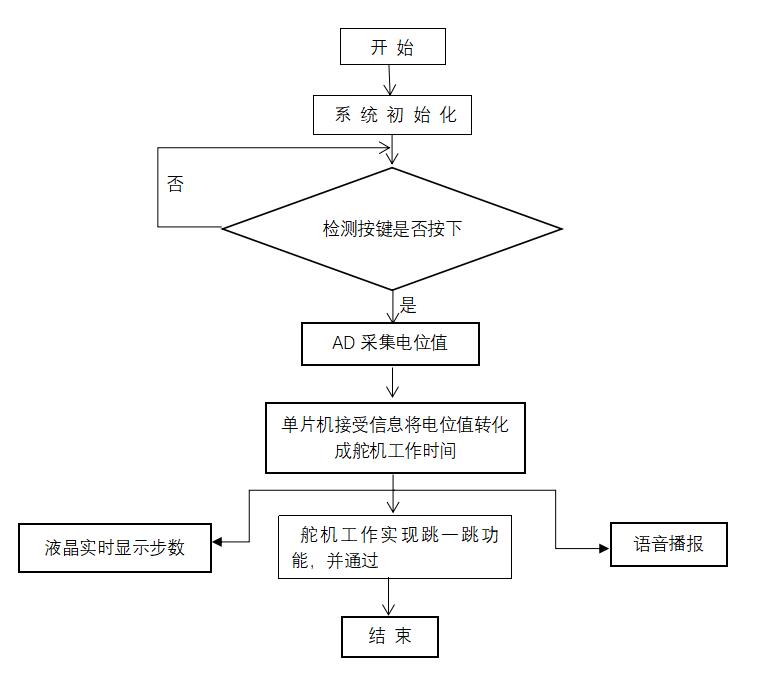
该软件是一款集编程和仿真于一体的软件，它支持汇编、C语言及二者的混合编程。

### 4.1.2 Altium Designer 软件

Altium Designer是原Altium Designer软件开发商Altium公司推出的一体化的电子产品开发系统，主要运行在Windows操作系统。这套软件通过把原理图设计、电路仿真、PCB绘制编辑、拓扑逻辑自动布线、[信号完整性分析](http://baike.baidu.com/item/%E4%BF%A1%E5%8F%B7%E5%AE%8C%E6%95%B4%E6%80%A7%E5%88%86%E6%9E%90" \t "http://baike.baidu.com/_blank)和设计输出等技术的完美融合，为设计者提供了全新的设计解决方案，使用该软件提高了电路图设计的质量和效率。



## 4.2设计流程图

****

## 4.3主程序设计

主程序如下：

#include <stc15.h>

#include <string.h>

#include <math.h>

#include "LCD1602.H"

#define MAIN\_Fosc 11059200UL //定义主时钟

#define PWM\_DUTY 221184 //定义PWM的周期，数值为时钟周期数，假如使用11.0592MHZ的主频，PWM频率为20HZ。

#define PWM\_HIGH\_MIN 32 //限制PWM输出的最小占空比。用户请勿修改。

#define PWM\_HIGH\_MAX (PWM\_DUTY-PWM\_HIGH\_MIN) //限制PWM输出的最大占空比

typedef unsigned char u8;

typedef unsigned int u16;

typedef unsigned long u32;

sbit P\_PWM = P3^5; //定义PWM输出引脚。

sbit key = P3^2;

u16 pwm; //定义PWM输出高电平的时间的变量

u16 PWM\_high,PWM\_low; //中间变量，用户请勿修改。

void delay\_ms(unsigned int ms);

void delay(unsigned long mycnt);

void LoadPWM(u16 i);

void PWM\_Init(void);

void UartInit(void);

void UartInit3(void);

void UART\_Send\_Byte(unsigned char dat);

void UART\_Send\_Buff(const char\* src);

void voice(char \*text);

unsigned char Long\_Str(long dat,unsigned char \*str);

void port\_mode() // 端口模式

{

P0M1=0x00; P0M0=0x00;

P1M1=0x00; P1M0=0x00;

P2M1=0x00; P2M0=0x00;

P3M1 &= ~(1 << 5); P3M0 |= (1 << 5); //P3.5 设置为推挽输出

P4M1=0x00; P4M0=0x00;

P5M1=0x00; P5M0=0x00;

P6M1=0x00; P6M0=0x00;

P7M1=0x00; P7M0=0x00;

}

void main()

{

/\*\*\*\*\*\*\*\*\*\*\*\*舵机/AD的的变量\*\*\*\*\*\*\*\*\*\*\*\*/

char a[11];

char a1[11];

char a2[11];

unsigned char adc\_status;

long temp\_buf1=0;

long temp\_buf2=0;

long T1=0;

long T2=0;

long time=0;

/\*\*\*\*\*\*\*\*\*\*\*\*语音函数的变量\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char length;

unsigned char ecc = 0;

unsigned int i=0;

unsigned char ta[11];

char text[] = {"\0"};

/\*\*\*\*\*\*\*\*\*\*\*\*LCD的变量\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char xPos,yPos; // X坐标、Y坐标

unsigned char \*s="Step=";

unsigned char DispBuf[9]; // 存放4个待发送ASCII码

unsigned long TestDat=0; // 临时变量

/\*\*\*\*\*\*\*\*\*\*\*\*记步变量step\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned long step = 0;

port\_mode(); // IO口工作模式设置

/\*\*\*\*\*\*\*\*\*\*\*\* LCD初始化 \*\*\*\*\*\*\*\*\*\*\*/

delay\_ms(100); // 等待LCD1602上电时内部复位

LCD1602\_Init();

SetCur(CurFlash); // 开光标显示、闪烁，NoCur——有显示无光标,NoDisp——无显示，

// CurNoFlash——有光标但不闪烁 ，CurFlash——有光标且闪烁

xPos=0; // xPos表示水平右移字符数(0-15)

yPos=1; // yPos表示垂直下移字符数(0-1)

WriteString(0,0,"JUMP-AND-JUMP"); // X坐标、Y坐标、字符串，屏幕左上角为坐标原点，水平：0-15，垂直：0-1

WriteString(xPos,yPos,s); // X坐标、Y坐标、字符串，屏幕左上角为坐标原点

xPos=5;

yPos=1;

Long\_Str(TestDat,DispBuf); // 同第4章8节"利用串口调试程序"

WriteString(xPos,yPos,DispBuf);

/\*\*\*\*\*\*\*\*\*\*\*\* PWM/串口初始化 \*\*\*\*\*\*\*\*\*\*\*/

UartInit();

UartInit3();

length = strlen(text);

P\_PWM = 0;

TR0 = 0; //停止计数

ET0 = 1; //允许中断

PT0 = 1; //高优先级中断

TMOD &= ~0x03; //工作模式,0: 16位自动重装

AUXR |= 0x80; //1T

TMOD &= ~0x04; //定时

INT\_CLKO |= 0x01; //输出时钟

TH0 = 0;

TL0 = 0;

TR0 = 1; //开始运行

EA = 1;

pwm = PWM\_DUTY / 25;

LoadPWM(pwm);

key = 1;

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

while(1)

{

if(!key)

{

delay(20);

if(!key)

{

P1ASF=0x01;

ADC\_CONTR=0x80; //打开AD电源

delay(10);

ADC\_CONTR=0xe0; //设置转换速度和通道1

CLK\_DIV|=0x20;

ADC\_CONTR|=0x08;

adc\_status=0;

while(adc\_status==0)

{

adc\_status=ADC\_CONTR&0x10;

}

ADC\_CONTR=ADC\_CONTR&0xe7;

temp\_buf1=(ADC\_RES<<8)+ADC\_RESL;

delay(10);

P1ASF=0x02;

ADC\_CONTR=0x80; //打开AD电源

delay(10);

ADC\_CONTR=0xe1; //设置转换速度和通道2

CLK\_DIV|=0x20;

ADC\_CONTR|=0x08;

adc\_status=0;

while(adc\_status==0)

{

adc\_status=ADC\_CONTR&0x10;

}

ADC\_CONTR=ADC\_CONTR&0xe7;

temp\_buf2=(ADC\_RES<<8)+ADC\_RESL;

delay(10);

time =temp\_buf1-temp\_buf2;

if(time>0)

{

time = 1.8\*time;

}

if(time<0)

{

time = -time;

time = 2.2\*time;

}

Long\_Str(time,a);

Long\_Str(temp\_buf1,a1);

Long\_Str(temp\_buf2,a2);

UART\_Send\_Buff(a1);

UART\_Send\_Buff(".");

UART\_Send\_Buff(a2);

UART\_Send\_Buff(".");

UART\_Send\_Buff(a);

UART\_Send\_Buff(".");

pwm = PWM\_DUTY / 35;

LoadPWM(pwm); //舵机按下

delay\_ms(time);

delay\_ms(5);

pwm = PWM\_DUTY / 25;

LoadPWM(pwm); //舵机抬起

delay(20);

xPos=5;

yPos=1;

step++;

TestDat = step;

Long\_Str(TestDat,DispBuf);

WriteString(xPos,yPos,DispBuf);

Long\_Str(step,ta);

for(i=0;i<11;i++)

{

text[i]=ta[i];

}

voice(text);

while(!key);

}

}

}

}

void delay(unsigned long mycnt)

{

while(mycnt--);

}

//========================================================================

// 函数: void delay\_ms(unsigned char ms)

// 描述: 延时函数。

// 参数: ms,要延时的ms数, 这里只支持1~255ms. 自动适应主时钟.

//========================================================================

void delay\_ms(unsigned int ms)

{

unsigned int i;

do{

i=MAIN\_Fosc/9600;

while(--i); //96T per loop

}while(--ms); //--ms ms=ms-1

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 计算PWM重装值函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LoadPWM(u16 i)

{

u16 j;

if(i > PWM\_HIGH\_MAX) i = PWM\_HIGH\_MAX; //如果写入大于最大占空比数据，则强制为最大占空比。

if(i < PWM\_HIGH\_MIN) i = PWM\_HIGH\_MIN; //如果写入小于最小占空比数据，则强制为最小占空比。

j = 65536UL - PWM\_DUTY + i; //计算PWM低电平时间

i = 65536UL - i; //计算PWM高电平时间

EA = 0;

PWM\_high = i; //装载PWM高电平时间

PWM\_low = j; //装载PWM低电平时间

EA = 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Timer0中断函数\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void timer0\_int (void) interrupt 1

{

if(P\_PWM)

{

TH0 = (u8)(PWM\_low >> 8); //如果是输出高电平，则装载低电平时间。

TL0 = (u8)PWM\_low;

}

else

{

TH0 = (u8)(PWM\_high >> 8); //如果是输出低电平，则装载高电平时间。

TL0 = (u8)PWM\_high;

}

}

## 4.4、液晶显示函数

液晶显示函数如下：

#include "LCD1602.H"

void delay2uS () // 22.1184MHz

{

unsigned char t=9;

while(--t);

}

// 正常读写操作之前检测LCD控制器状态

// 读状态时序：RS=0，RW=1，E=1，判断忙完毕后释放总线

void WaitIdle()

{

unsigned char tmp;

RS=0; // 命令

RW=1; // 读取

DPORT=0xff; // 为接收数据作准备

\_nop\_(); // 短暂延时

E=1; // 使能LCD1602

delay2uS(); // LCD1602在E为高电平区间输出数据到端口

for(;;)

{

tmp=DPORT; // 将数据端口上的值赋给tmp

tmp&=0x80; // 最高位为1时表示液晶模块正忙，不能对其进行操作

if( tmp==0) // 其余6位表示内部当前显示地址，无实际用途。

break;

}

E=0; // 释放总线

}

// 向LCD1602液晶写入一字节数据，dat-待写入数据值

// 写数据时序：RS=1，RW=0，D7——D0 = 数据，E=正脉冲，液晶在脉冲下降沿采样数据。

void LcdWriteDat(unsigned char dat)

{

WaitIdle(); // 等待LCD1602空闲

RS=1; // 数据

RW=0; // 写

DPORT=dat; // 将待写数据送到数据端口

\_nop\_(); // 短暂延时

E=1; // 使能LCD1602

delay2uS(); // LCD1602在E为高电平区间读取数据端口上的值

E=0; // 关闭LCD1602使能,释放总线

}

// 向LCD1602液晶写入一字节命令，cmd-待写入命令值

// 写命令时序：RS=0，RW=0，D7——D0 = 数据，E=正脉冲，液晶在脉冲下降沿采样数据。

void LcdWriteCmd(unsigned char cmd)

{

WaitIdle(); // 等待LCD1602空闲

RS=0; // 命令

RW=0; // 写

DPORT=cmd; // 将命令码输出在数据端口上

\_nop\_(); // 短暂延时

E=1; // 使能LCD1602

delay2uS(); // LCD1602在E为高电平区间读取数据端口上的值

E=0; // 关闭LCD1602使能,释放总线

}

// 清屏命令：清除显示内容，将1602内部RAM全部填入空白的ASCII码20H

// 光标归位，将光标撤回到屏幕左上角的坐标原点

// 将1602内部显示地址设为0

void ClrLcd()

{

LcdWriteCmd(0x01);

}

// 内部函数用于设置显示字符起始坐标

void LcdPos(unsigned char xPos,unsigned char yPos)

{

unsigned char tmp;

xPos&=0x0f; // x位置范围是0~15

yPos&=0x01; // y位置范围是0~1

if(yPos==0) // 显示第一行

tmp=xPos; // 第一行字符地址从 0x00 开始

else

tmp=xPos+0x40; // 第二行字符地址从 0x40 开始

tmp|=0x80; // 设置 RAM 地址

LcdWriteCmd(tmp);

}

void SetCur(unsigned char Para) // 设置光标

{

switch(Para)

{

case 0:

{

LcdWriteCmd(0x08); break; // 关显示

}

case 1:

{

LcdWriteCmd(0x0c); break; // 开显示但无光标

}

case 2:

{

LcdWriteCmd(0x0e); break; // 开显示有光标但不闪烁

}

case 3:

{

LcdWriteCmd(0x0f); break; // 开显示有光标且闪烁

}

default:

break;

}

}

//在指定的行与列显示指定的字符，xpos:行，ypos:列，c:待显示字符

void WriteChar(unsigned char xPos,unsigned char yPos,unsigned char Dat)

{

LcdPos(xPos,yPos);

LcdWriteDat(Dat);

}

// 在液晶上显示字符串，xpos:行坐标，ypos:列坐标，str-字符串指针

void WriteString(unsigned char xPos,unsigned char yPos,unsigned char \*s)

{

unsigned char i=0;

LcdPos(xPos,yPos); // 起始坐标

while(s[i])

{

LcdWriteDat(s[i]);

i++;

if (i>=16) break; // 超出16个字符外的数据丢弃

}

}

// LCD 1602初始化

void LCD1602\_Init()

{

LcdWriteCmd(0x38); // 显示模式设置，设置16×2显示，5×7点阵，8位数据接口

LcdWriteCmd(0x08); // 显示关闭，不显示光标、光标不闪烁

LcdWriteCmd(0x01); // 显示清屏

LcdWriteCmd(0x06); // 显示光标移动位置

LcdWriteCmd(0x0c); // 显示开及光标设置

}

## 4.5 语音函数

语音函数如下：

#include <stc15.h>

#include <string.h>

void voice(char \*text)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*需要发送的文本\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

unsigned char headOfFrame[5];

unsigned char length;

unsigned char ecc = 0; //定义校验字节

unsigned int i=0;

length = strlen(text); //需要发送文本的长度

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*发送过程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

headOfFrame[0] = 0xFD ; //构造帧头FD

headOfFrame[1] = 0x00 ; //构造数据区长度的高字节

headOfFrame[2] = length + 3; //构造数据区长度的低字节

headOfFrame[3] = 0x01 ; //构造命令字：合成播放命令

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*带背景音乐\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

headOfFrame[4] = 0x21 ; //构造命令参数：编码格式为GBK

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*不带背景音乐\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//headOfFrame[4] = 0x01 ; //构造命令参数：编码格式为GBK

for(i = 0; i<5; i++) //依次发送构造好的5个帧头字节

{

ecc=ecc^(headOfFrame[i]); //对发送的字节进行异或校验

S3BUF = headOfFrame[i];

while (!(S3CON & 0x02)) {;} //等待发送中断标志位置位

S3CON &= 0xFD; //发送中断标志位清零

}

for(i = 0; i<length; i++) //依次发送待合成的文本数据

{

ecc=ecc^(text[i]); //对发送的字节进行异或校验

S3BUF = text[i];

while (!(S3CON & 0x02)) {;}

S3CON &= 0xFD;

}

S3BUF=ecc; //最后发送校验字节

while (!(S3CON & 0x02)) {;}

S3CON &= 0xFD;

}

## 4.6 LCD1602.h头文件

#include "stc15.h"

#include "intrins.h" // \_nop\_()函数需要

sbit RS = P4^1; // 根据实际硬件连接修改

sbit RW = P4^2; // 根据实际硬件连接修改

sbit E = P4^4; // 根据实际硬件连接修改

#define DPORT P2 // 根据实际硬件连接修改

#define NoDisp 0 // 无显示

#define NoCur 1 // 有显示无光标

#define CurNoFlash 2 // 有光标但不闪烁

#define CurFlash 3 // 有光标且闪烁

void LCD1602\_Init(); // 初始化

void ClrLcd(); // 清屏命令

void SetCur(unsigned char Para); //设置光标

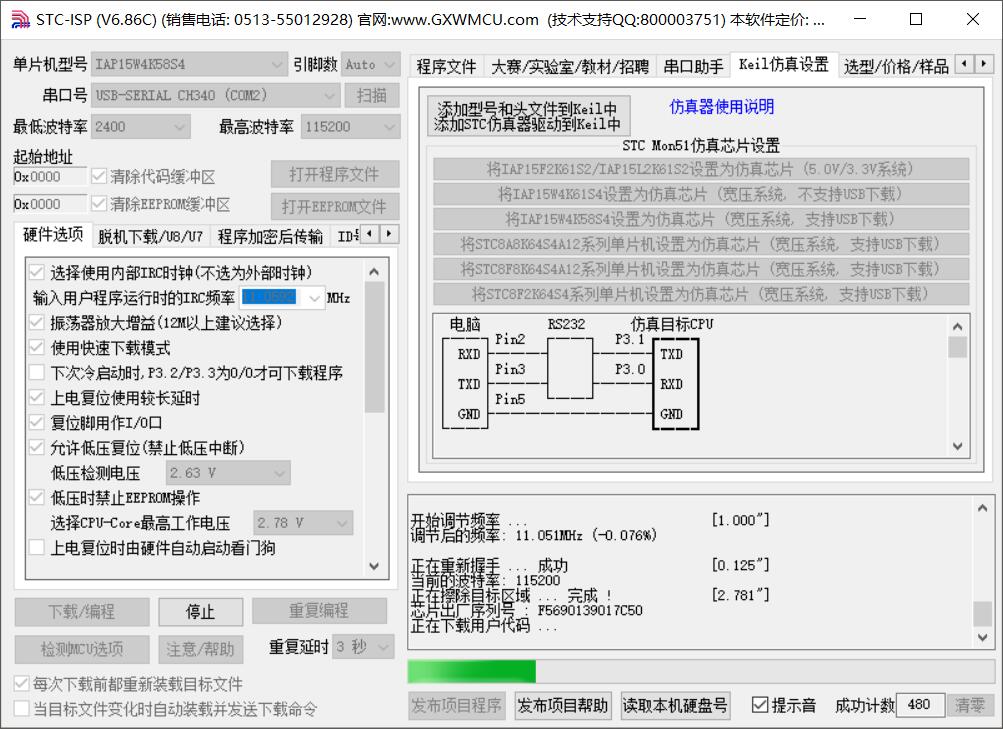
void WriteChar(unsigned char xPos,unsigned char yPos,unsigned char Dat); // 写1个字符

void WriteString(unsigned char xPos,unsigned char yPos,unsigned char \*s); // 写字符串

# 第五章、测试方法与仪器，测试数据及测试结果分析

## 5.1 单片机的硬件仿真

在连接实际硬件电路之前，我们在软件STC-ISP中进行了硬件仿真，调试了单片机的模拟工作状态，为单片机的实际工作做了准备与排障工作。



STC-ISP硬件仿真界面

## 5.2 AD采集数据包正确率测试

通过串口调试助手，使用两台计算机通过通讯模块收发100条数据。

其中AD数据为：

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

接收的数据为：

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

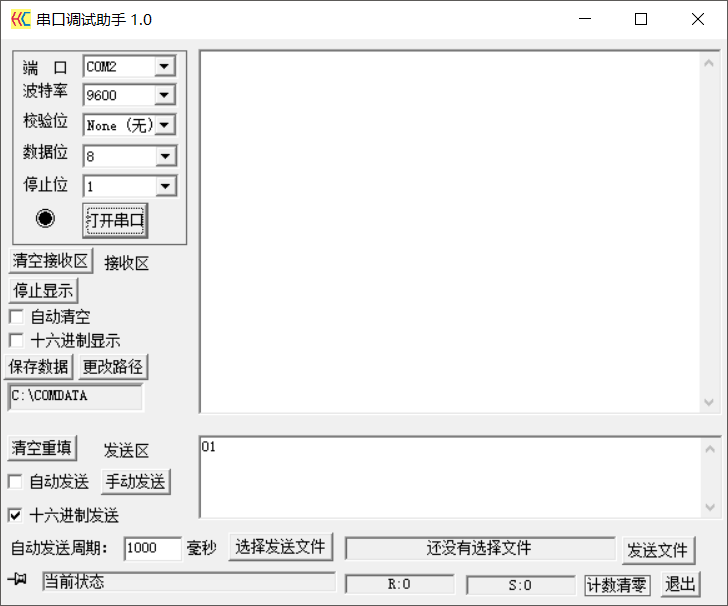
0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01 0x01

正确率为100%，满足通讯的需求标准。

## 5.3 通过串口通讯助手调试串口通信模块

在串口通信模块被连接上单片机之前，我们在计算机上通过串口通讯助手对该模块进行了调试，更好地设置了该模块的工作状态和工作方式，排除了与单片机连接后模块故障的初步可能。



# 参考文献

[1] 陈桂友，《单片机应用技术基础》，机械工业出版社，2015年8月第一版

[2] 郭天祥，《51单片机C语言教程》，电子工业出版社，2009年1月第一版

[3] 邵发， 《C/C++学习指南》，清华大学出版社，2016年1月第一版

[4] 叶林朋，《Altium Designer 14 原理图与PCB设计》，西安电子科技大学出版社，2015年5月第一版

[5] 丁向荣，《单片机原理与应用项目教程：基于STC15W4K58S4单片机》，清华大学出版社，2015年出版

[6] 徐爱钧，《Keil C51单片机高级语言应用编程技术》，电子工业出版社，2015年版