FS-K51 多功能语音模块

用户说明文档

版本号: V1.1

简介

FS-K51 语音模块是雍异科技开发的多功能语音播放产品，它具有成本低，可重复烧写，接口丰富，使用方便，功能全面，音质出众，二次开发方便等优点。它通过 USB 下载器方便高效的下载语音，通过按键、高低电平、脉冲、RS485/232 信号等方式触发语音播放，板载 25W 大功率功放系统，适合在多种场合中使用。

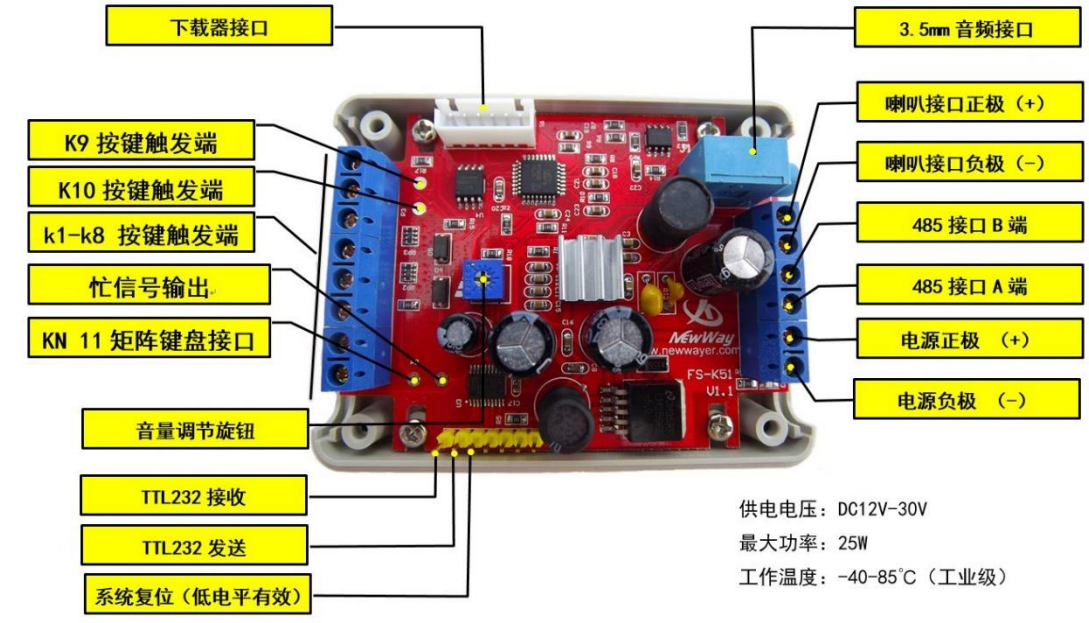
产品特性

- 语音触发方式可选，可以通过 10 个按键触发 10 段语音，也可以通过 3*8 矩阵键盘触发 24 段语音，或者也可以通过 RS485/232 给模块指令触发 200 段语音，能够通过指令设置音量的大小，设置循环播放等等，使用灵活。
- 触发电平可选，播放方式可选，通过上位机软件可以设置高（低）电平触发、脉冲触发，可以选择循环播放或单次播放。
- 工作方式可设置，您可以让它工作在普通模式，你也可以让它工作在 MP3 模式，通过给相对应的管脚信号就能控制播放、停止、上一首、下一首、音量等功能。
- 采样频率可调节，满足不同场合需求。设置采样频率可以控制语音最大长度以及音质，最长能够播放 1000 秒(15

分钟)的语音（6000HZ/16 位语音信号）。

- 全部采用工业级芯片，正规厂家做机器焊接，强抗干扰，特别适合在恶劣环境下工作，工作温度-40℃至+80℃。
- 通过 USB 下载器直接从电脑上位机程序下载语音，高效快捷，无失真，适合大规模使用。
- 板载 25W 大功率功放，足以满足普通需求。
- 12-30V 直流压供电，带防反接功能，方便与大部分系统集成。

产品接口



📖 模块后面丝印层很清楚的标注了每一个接口，用户使用时可以参考使用。

接口说明：

- **TTL232 接口：**TTL232 信号输入的接口，可以接到单片机上面用于通过命令控制语音模块,使用很方便
- **RS485：**通讯接口为 RS485 信号的输入口，可以连接到 RS485 设备上通过指令控制语音模。
- **喇叭接口：**可以接无源的扬声器（喇叭）。
- **3.5mmy 音频输出接口：**输出语音模块的语音信号，可以通过 3.5mm 录音线接到有源音箱或者功放上面,特别注意的如果此接口接了录音线，板载功放将不再继续工作。
- **音量调节旋钮：**可以调节板载功放的输出音量。
- **按键触发接口：**可以通过与触发公共端地端子(GND)相连从而触发语音的播放，或者接入 3*8 的矩阵键盘
- **下载接口：**可以通过 USB 下载器通过电脑下载语音。
- **复位接口：**接到 GND 可以使得模块复位。

标注说明：

按键触发口

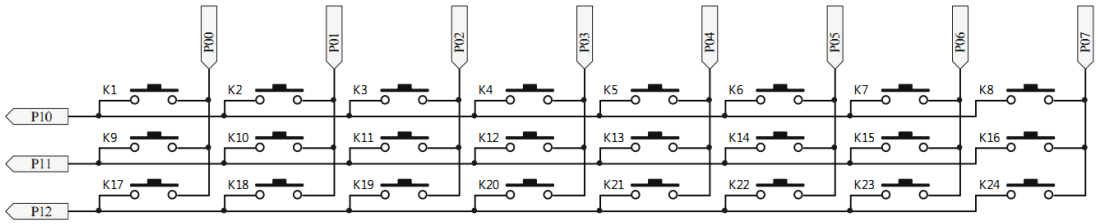
K1～K10 被定义为控制端口按键触发控制端，所对应控制端口分别为 P00～P07、P10 和 P11。通过 K1～K10 拉低相关控制端口的电平或往相关控制端口输入脉冲，就可达到

控制触发语音播放的效果。按键 K0～K9 的触发模式可以被设置为无效按键、脉冲可重复触发、脉冲不可重复触发、电平保持不可循环、电平保持可循环、电平非保持可循环、上一曲不循环、下一曲不循环、上一曲可循环、下一曲可循环、停止、播放/暂停、音量+、音量-及播放/停止等 15 种触发方式。

标号	说明
K1	按键 1 触发口，矩阵键盘 P00 口
K2	按键 2 触发口，矩阵键盘 P01 口
K3	按键 3 触发口，矩阵键盘 P02 口
K4	按键 4 触发口，矩阵键盘 P03 口
K5	按键 5 触发口，矩阵键盘 P04 口
K6	按键 6 触发口，矩阵键盘 P05 口
K7	按键 7 触发口，矩阵键盘 P06 口
K8	按键 8 触发口，矩阵键盘 P07 口
K9	按键 9 触发口，矩阵键盘 P010 口
K10	按键 10 触发口，矩阵键盘 P11 口
K_N_11	矩阵键盘 P12 口

3*8 矩阵键盘接口

矩阵键盘按照如下方式连接，可以通过矩阵键盘触发 24 段语音播放



TTL 串口

RX:模块 TTL 串口接收端

TX:模块 TTL 串口发送端

GND:模块地

❋ 特别注意，TTL 串口不是 RS232 串口，如果使用 RS232 串口与本模块通讯，需要一个电平转换模块（比如 MAX232 模块）转换电平后方可与模块通讯，如不进行电平转换，模块可能损坏

RS485 通讯端口

485A:模块 485 通讯 A 端

485B:模块 485 通讯 B 端

GND:模块地

电源接口

VCC：12-30V 电源正极

GND:电源负极

❋ 特别注意，由于功放需要一个相对较大的电流，再给模块供电时，电流一定要够，不然会出现模块音量不够的情况，推荐使用 12V/2A 以上的电源。

喇叭接口

SP+：喇叭接口正极

SP-：喇叭接口负极

❋ 喇叭分正负接入

软件说明

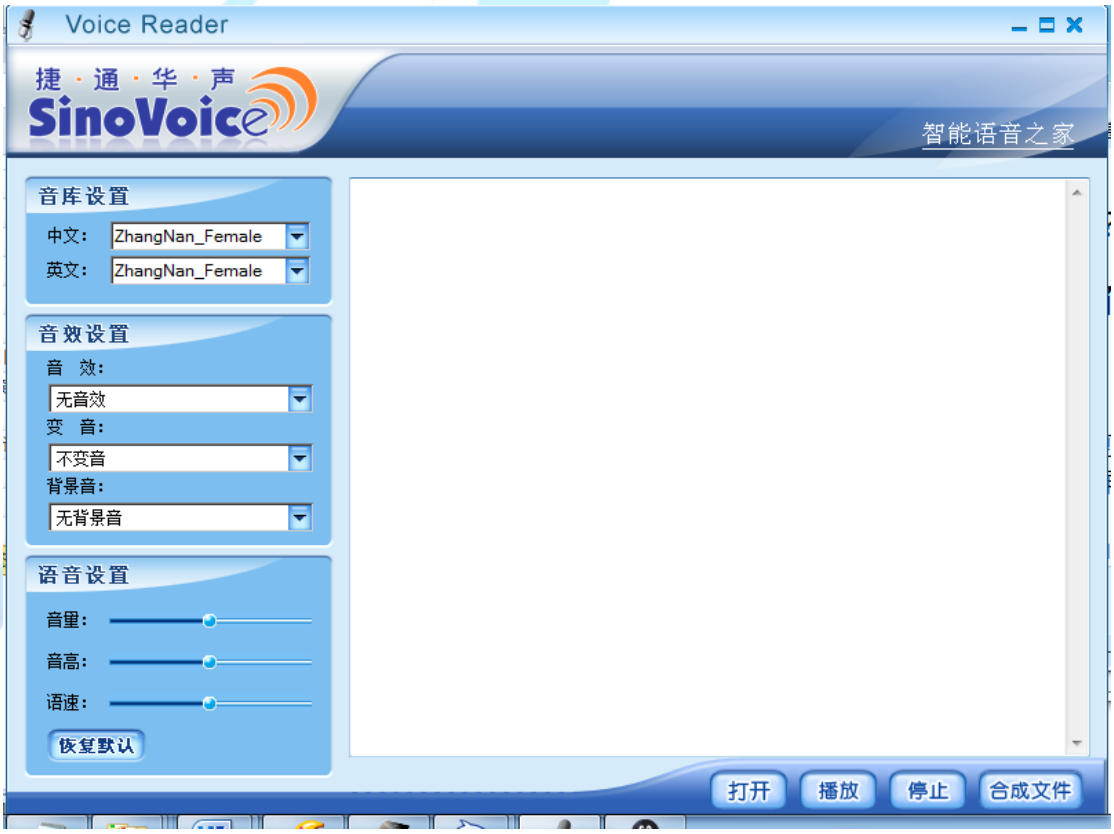
在本节中，我们将完成一个语音应用工程来说明语音软件的使用方法。

音频文件的获取

声音的获取有很多方式，可以自己录制或者通过**文字-语音**软件进行获取，在我们的附送软件中，我们提供了三款该类型软件，软件的使用方法大同小异，在此，我们介绍 VoiceReader 的使用方法

打开**赠送资料\文字语音软件\VoiceReader_2013**，出现如

下界面

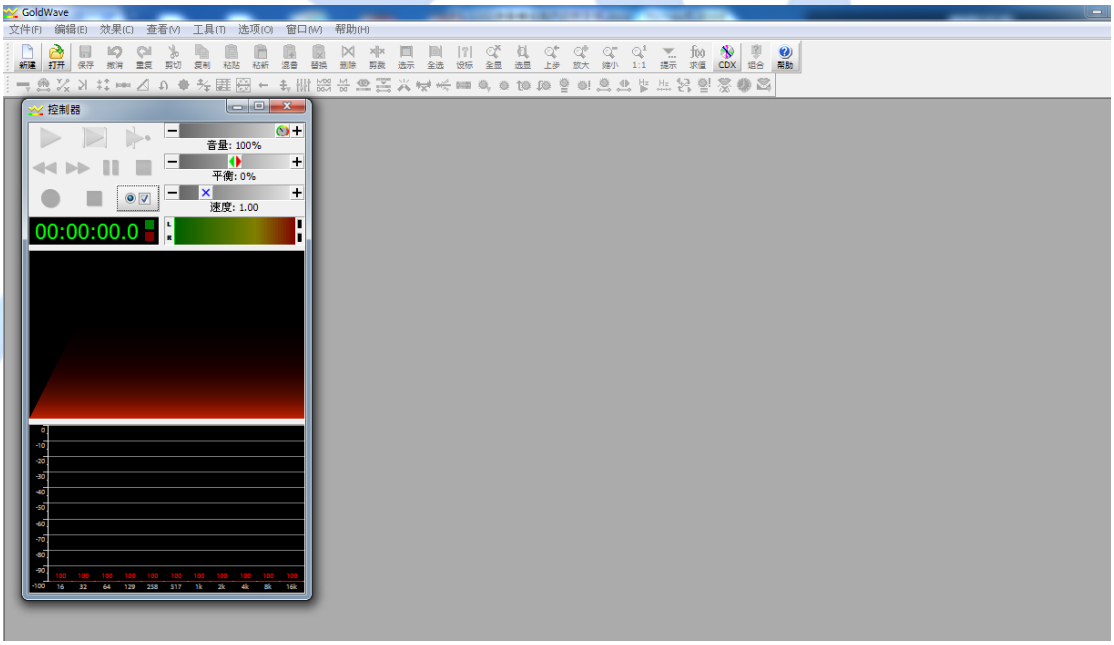


输入**你好，欢迎使用 FS-K51 语音模块**点击合成文件，即可完成语音生成。

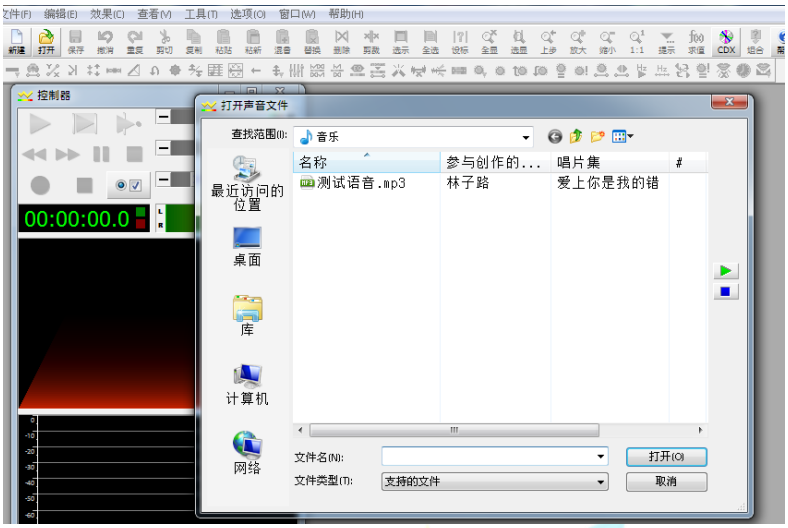
音频文件的准备

FS-K51 语音模块支持 6000Hz、8000Hz、10000Hz、12000Hz、14000Hz、16000Hz、18000Hz、20000Hz、22000Hz 等这几种采样率的音频，请务必要转换成以上提到的这几种类型的采样率。一般来说采样频率越高音效越好，采样频率越低，音效越差，改变采样频率可以通过专业的语音处理软件改变，在产品赠送资料中，我们已经赠与您一款专业的音频处理软件 GoldWave，下面我们将基于该软件处理得到我们所需要的语音。

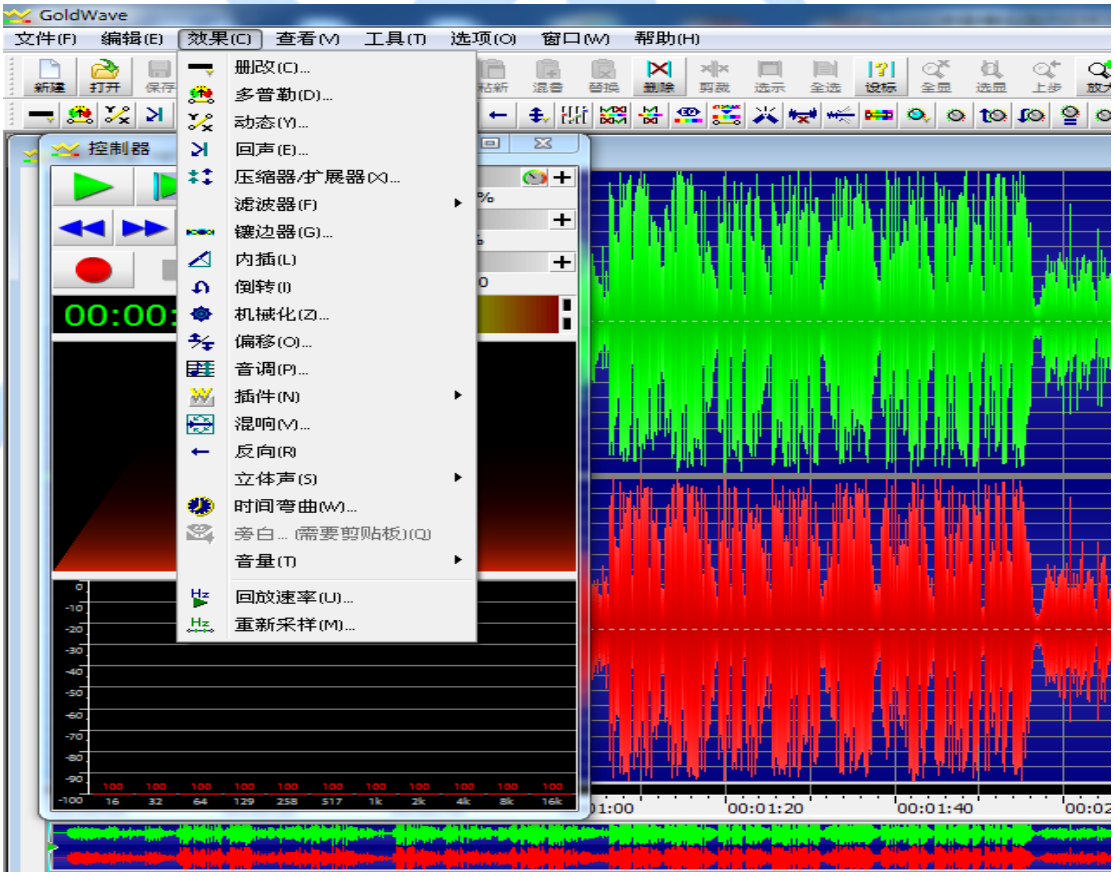
1. 打开音频处理软件 GoldWave.exe（赠送资料\音频处理软件）

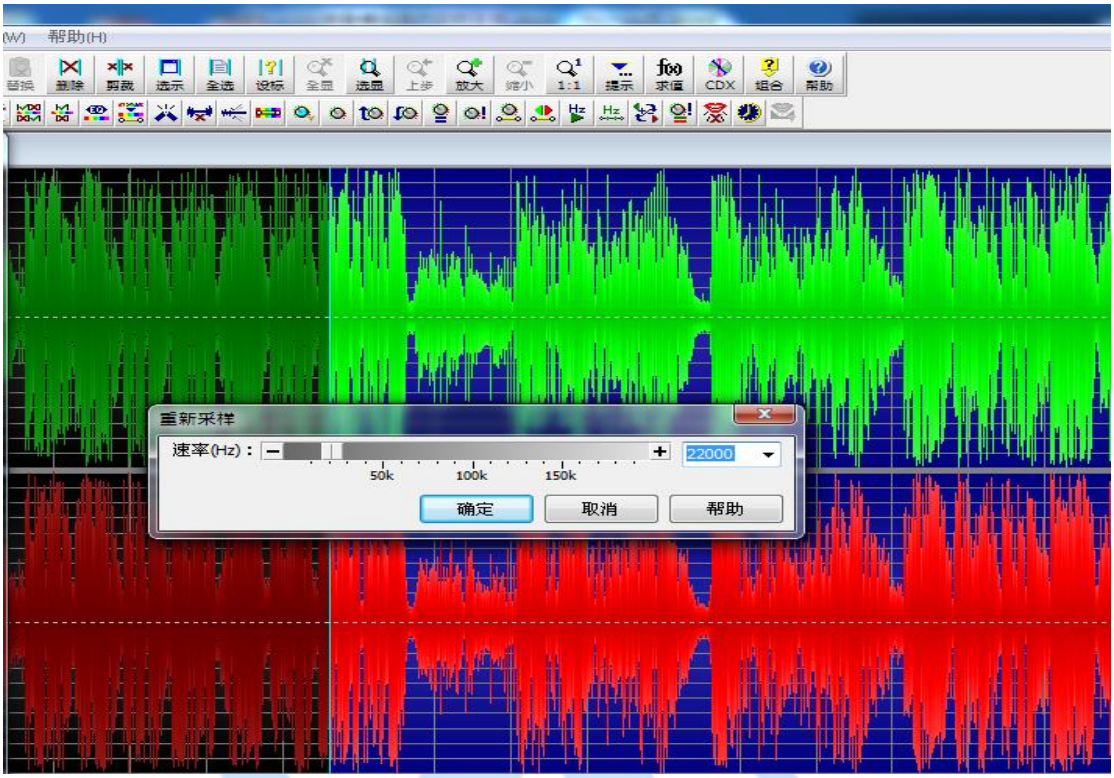


2. 点击打开按钮，选择我们所需要处理的音频文件，此例中我们选择“测试语音.mp3”

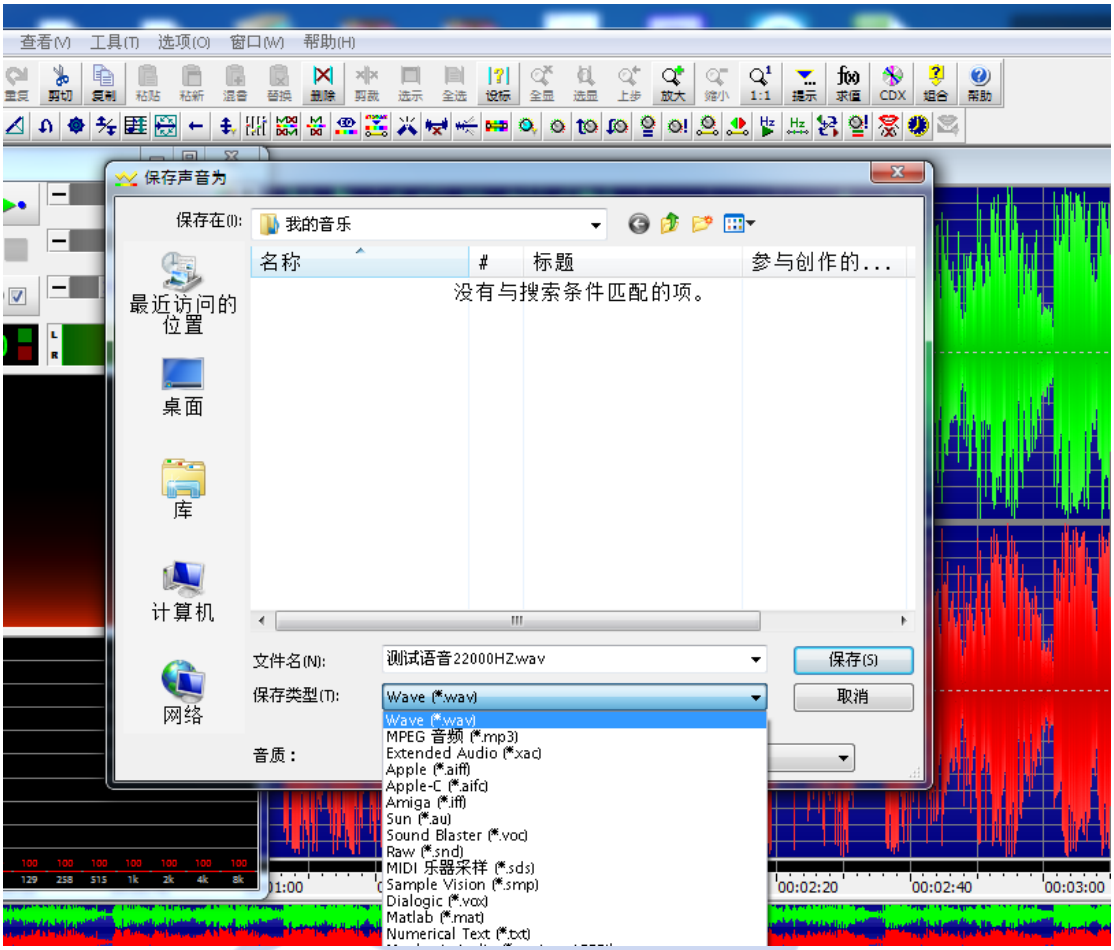


3. 点击菜单栏中效果->重新采样输出我们所期望的采样频率，这里我们输出最高的采样频率 22000HZ，点击确定，采样转换完成后将有一个警告，忽略即可。





4. 等待计算机完成采样后我们点击菜单栏上面文件->另存为按钮，把刚刚转换出来的语音保存，这里我们将文件名保存为“测试语音 22000HZ.wav”，应当特别注意的是 FS-1000 下载软件只支持 WAV 格式，所以保存类型我们需要选择 WAV。



音频文件的下载

准备工作

1. 安装上位机软件“语音模块下载软件”

2. 按照上节提到的步骤完成适合频率音频的转换。

3. 使用 6 芯的接线把 USB 下载器与 FS-K51 模块连接起来

4. 给 FS-K51 模块提供一个 12V-30V 的电压。

上位机软件的使用方法

通过上位机的设置我们可以让 FS-1000 工作在不同的工作模式。

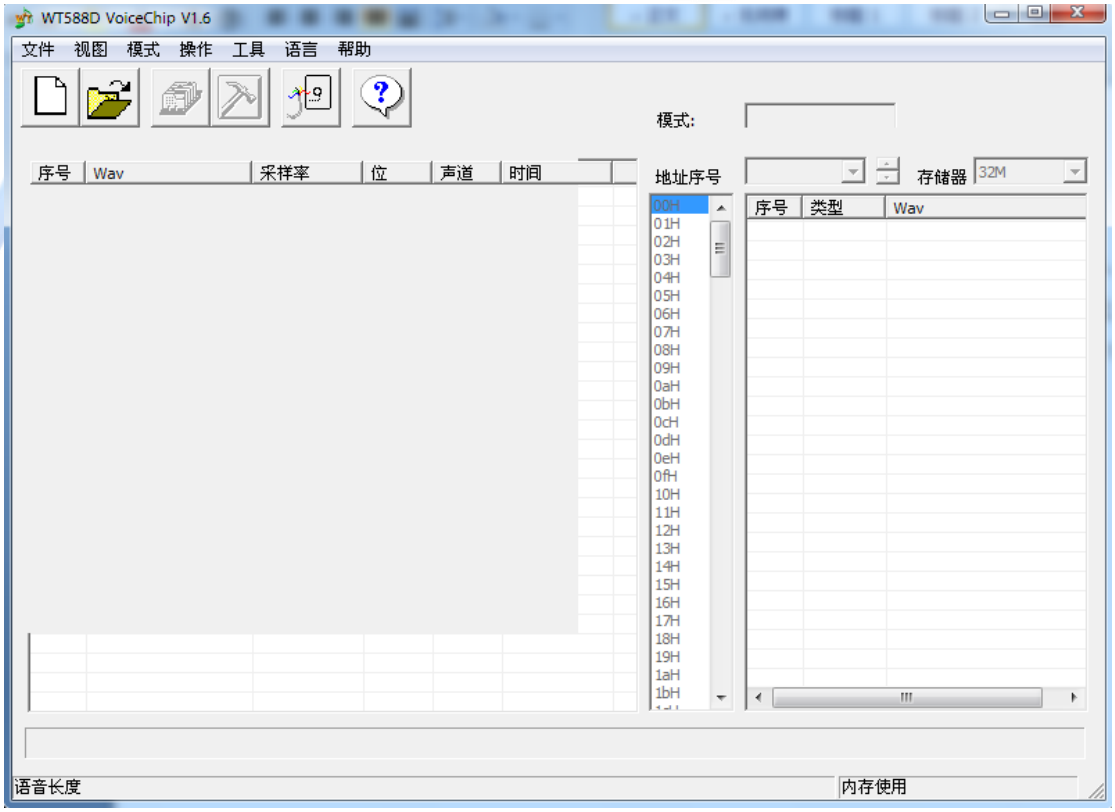
设置为按键模式

通过上位机的设置可以让模块工作在按键模式，此为最常见的应用模式，这种模式下，我们可以通过按键、电平脉冲、以及电平触发语音的播放。

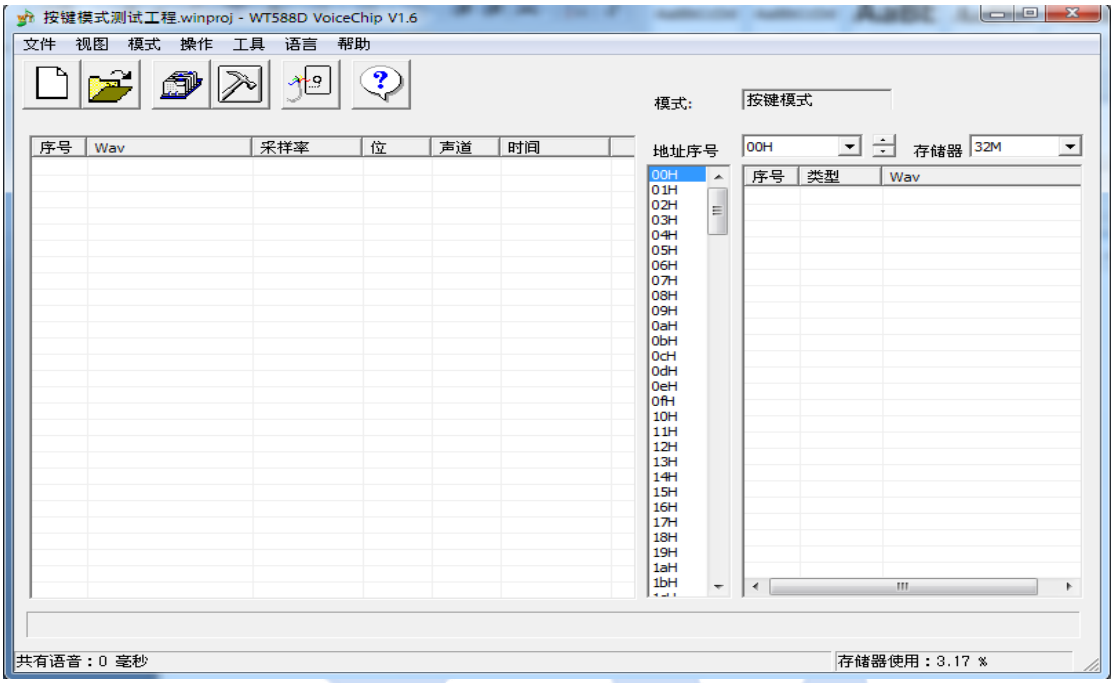
📖 小技巧：如果需要上电自动播放，可以设置成按键形式，然后再把一个按键接到地即可。

具体实现步骤：

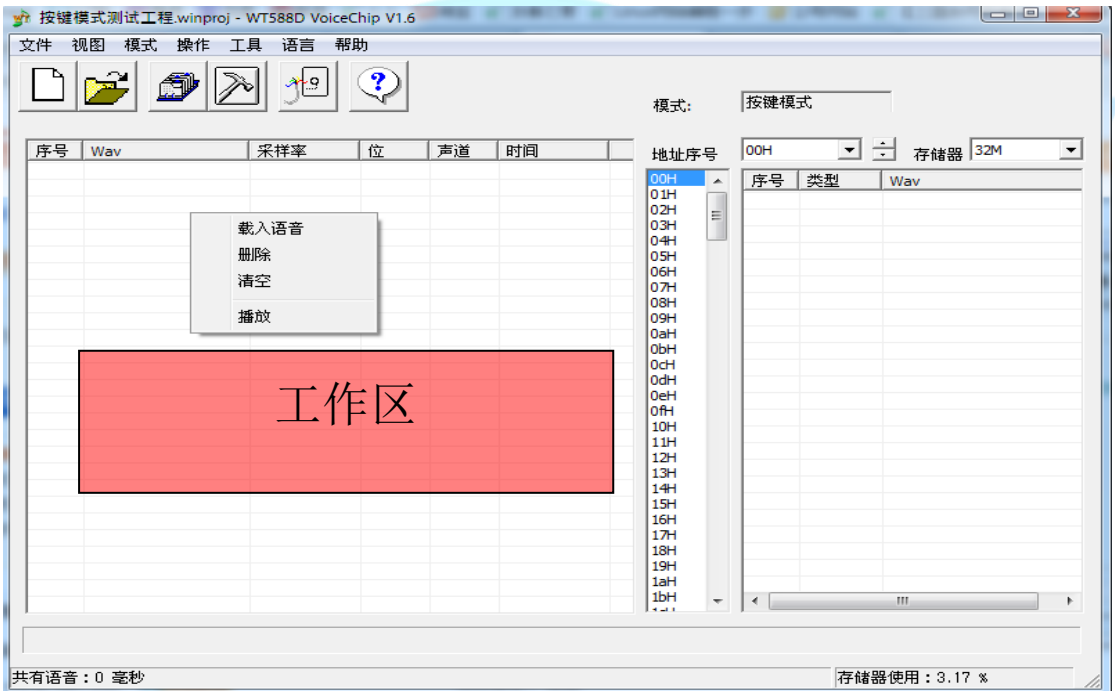
1. 打开 WT588D VoiceChip Beta 1.6.exe 软件，出现如下界面



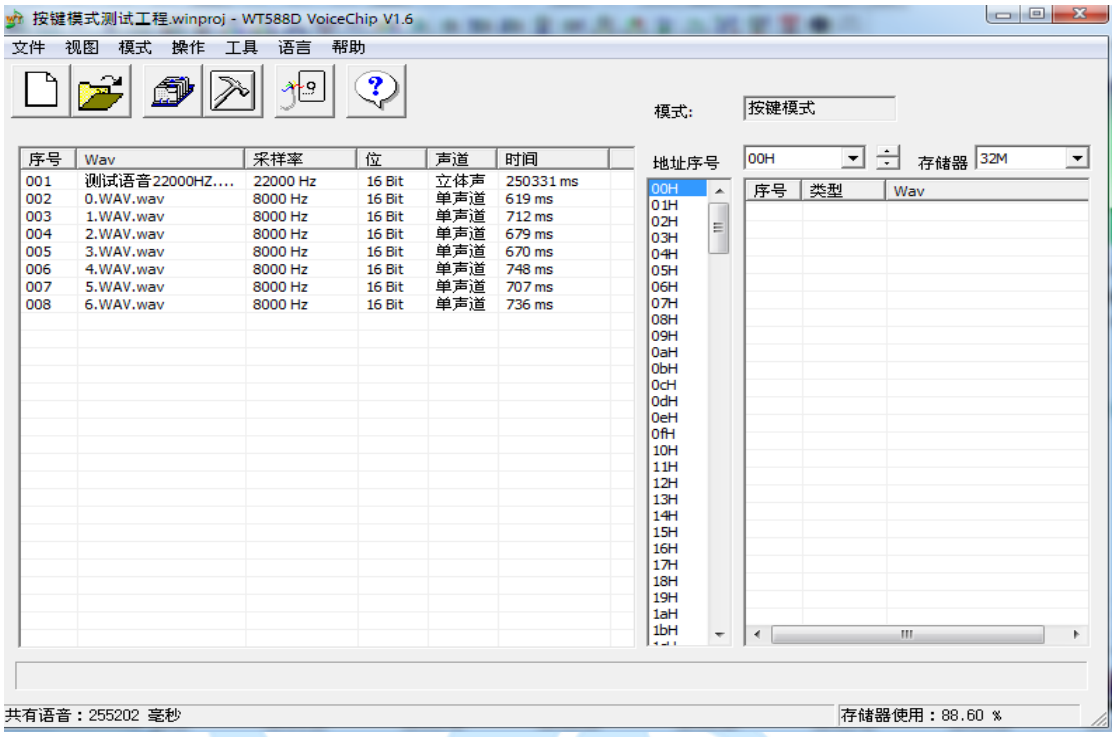
2. 新建一个语音工程，点击菜单栏文件->新建，输入工程名称为“按键模式测试工程”，点击确定按钮，出现如下界面：



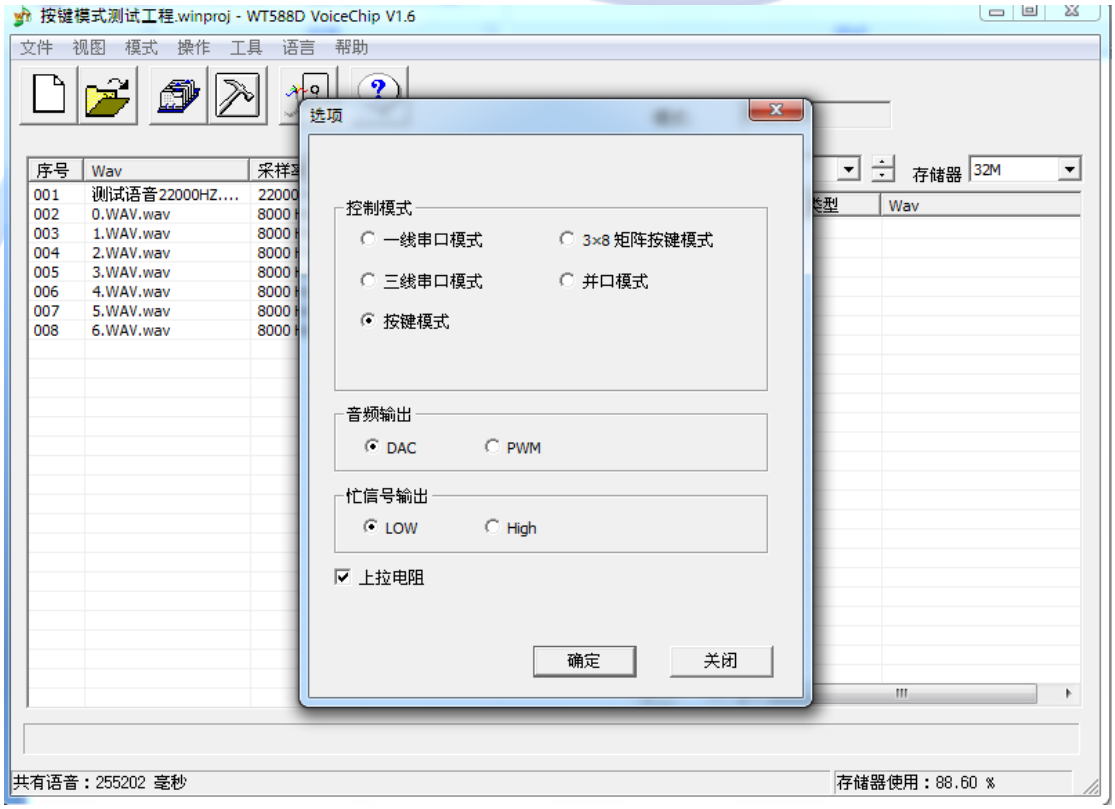
3. 在工作区点击右键->载入语音把已经编辑好的语音添加到工作区



在此，我们添加入刚刚编辑好的测试语音和一些数字语音

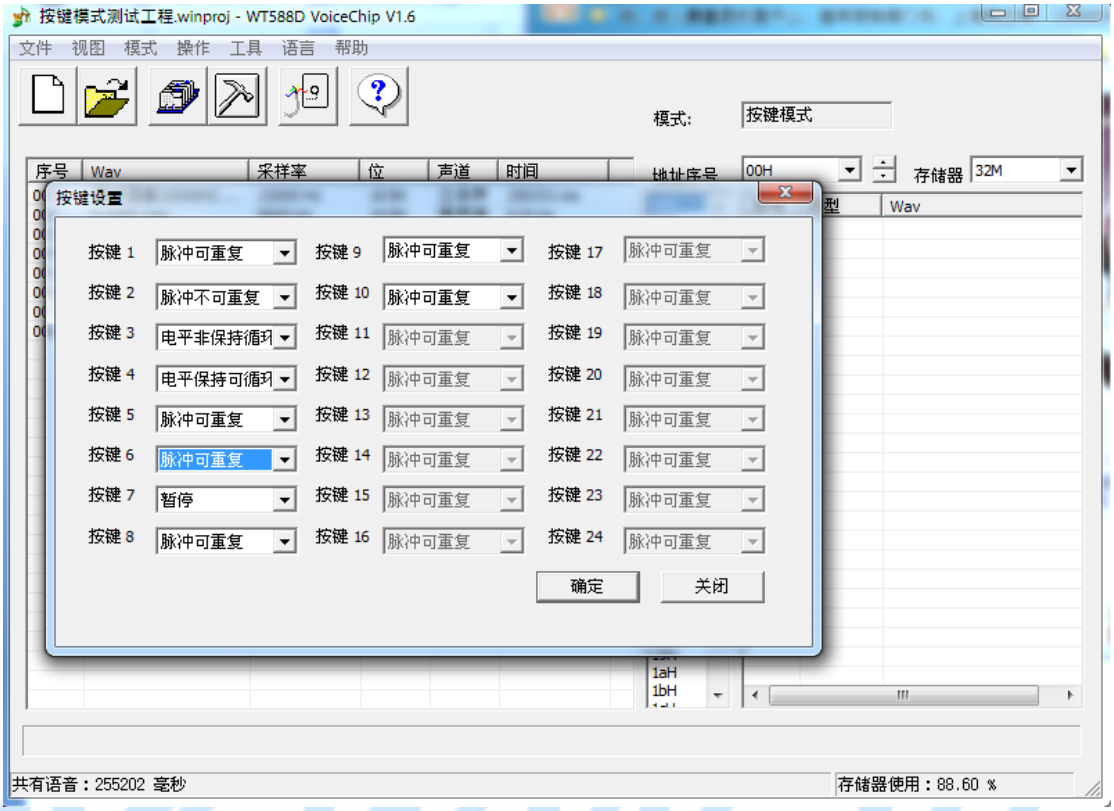


4. 点击**操作->设置**，设置模块的工作状态，在这里，我们设置 FS-1000 工作在按键模式，音频输出为 DAC 模式，**特别注意，FS-K51 不支持 PWM 输出，如果设置为此输出将使得输出的音质失真较大，严重是还会损坏模块，使用时应该特别注意。**具体设置为如下图所示：

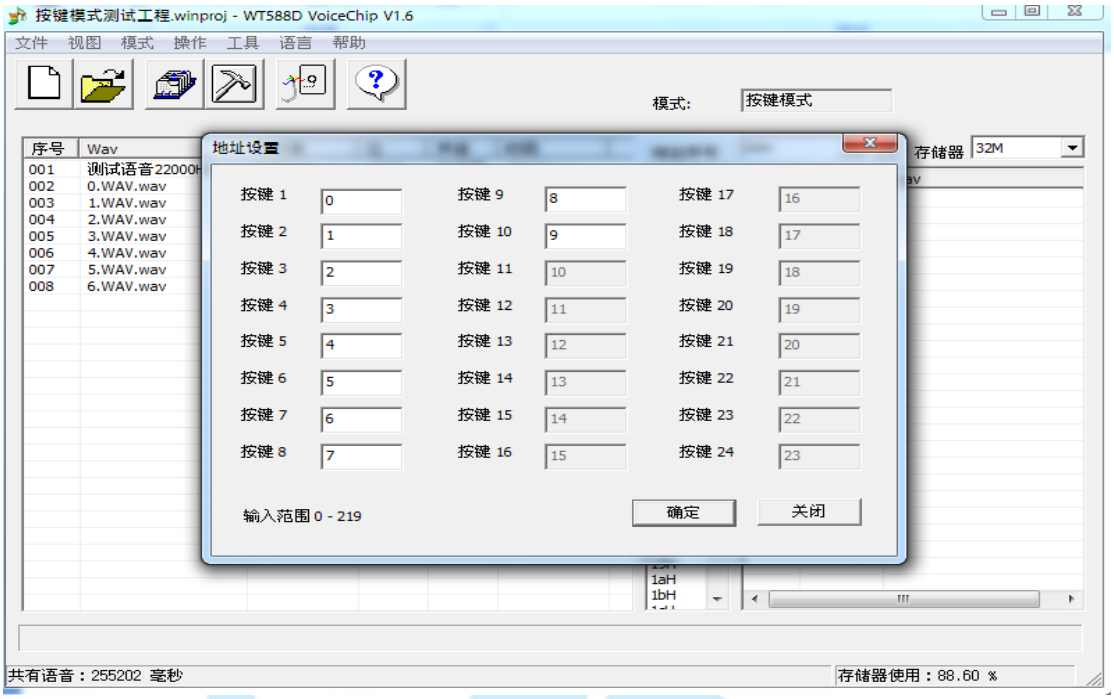


点击确定，完成设置。

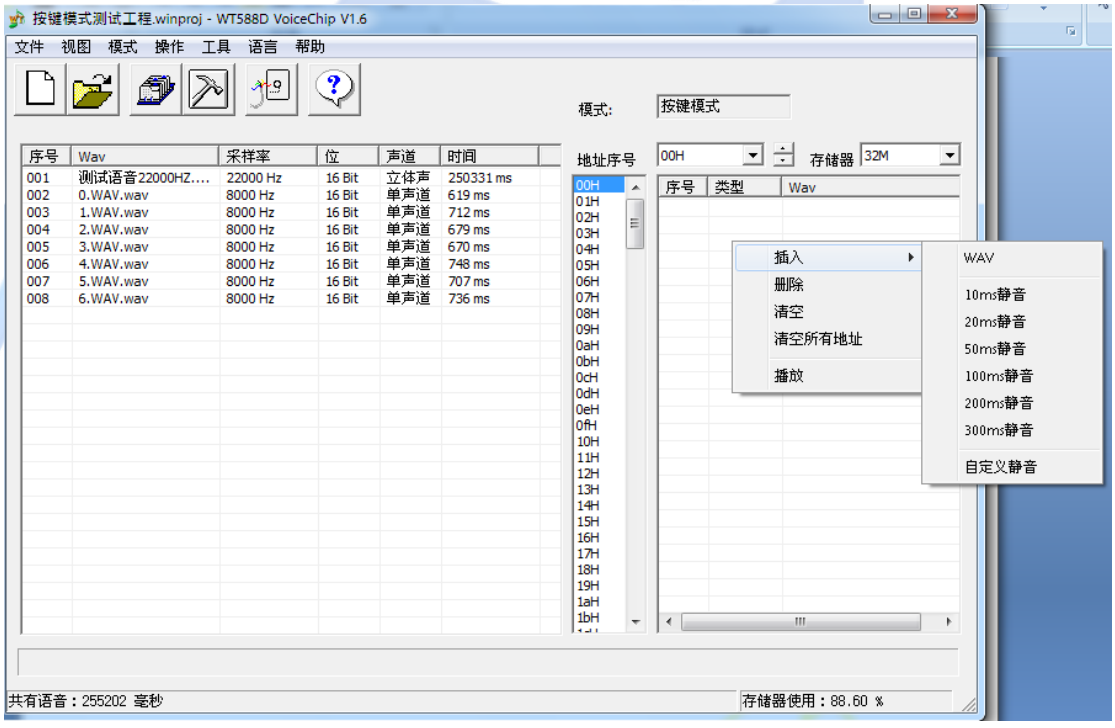
5. 点击**操作->按键设置**，按键的触发方式，这里可以选择各种触发方式，用户可以按照需求设置，这里就不一一例举，在此我们把按键 1 设置脉冲可重复，按键 2 设置成脉冲不可重复，按键 3 设置成电平非保持循环，按键 4 设置为电平保持可循环，按键 5 设置为脉冲可重复，按键 6 设置为脉冲可，按键 7 设置为暂停。具体设置为下图所示：



6. 点击**操作->地址设置**，设置按键触发的地址，在这里，我们用默认设置就好。



7. 在**地址序号**的下拉框中选择相应的地址，这个地址就是上一步按键所设置的地址，我们选中一个地址，在右边工作区点击**右键->插入->WAV**，插入我们处理好的语音。



在这里，我们在

地址 00H 插入“测试语音 22000HZ.wav”

地址 01H 插入“0.wav”和“1.wav”

地址 02H 插入“2.wav”

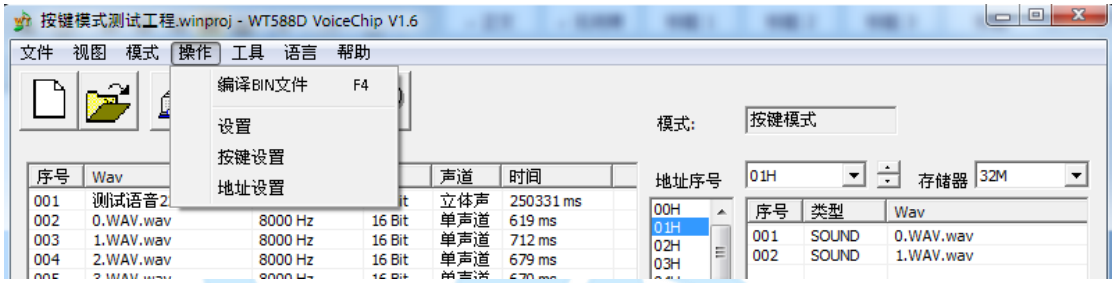
地址 03H 插入“3.wav”

地址 04H 插入“4.wav”

地址 05H 插入“5.wav”

地址 00H 插入“6.wav”

8. 上述设置全部添加完成后，整个工程设置基本完成，但是在下载之前我们需要编译工程，点击菜单栏操作->编译，稍等片刻后完成对工程的编译。



9. 通过 MINI-USB 线让电脑和 FS-K51 语音模块相连，并且给 FS-K51 供一个 12V 的直流电，点击菜单栏工具->下载，然后点击连接，一键下载，完成下载工作。

设置为一线模式(MODBUS-RTU 模式)

设置为一线串口模式可以使得模块工作在 RS485 或者 232 信号触发模式(TTL 电平)或者使用红外信号触发，这样您就可以通过发送指令触发语音播放、控制，如果您选择的是 MODBUS-RTU 控制的方式，则可使用标准的 MODBUS-RTU 协议进行控制。

协议出厂默认参数：

- 通讯接口： RS-485，TTL232。
- 通讯格式： 1 个起始位，8 个数据位，无校验，1 个停止位。
- 波特率： 9600（可软件更改） 。

- 地址： 1（可软件更改） 。
- 通讯方式： 监控主机与本装置采用一对一（或一对多）主从查询方式
- 数据协议： MODBUS-RTU

2. MODBUS 协议帧格式

数据格式：标准的 MODBUS 协议 RTU 方式，16 进制编码，若数字不是一个字节，则高字节在前低字节在后，主从式传输，每一帧有固定的格式，包括几个字节的数据，主机发送帧每个字节的意义如下：

从机地址（1Byte）	命令（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	数据区（nByte）	CRC 校验（2Byte）
addr	cmd	regH regL	numH numL	d0-dn	

若需要返回数据，从机返回帧每个字节的意义如下：

从机地址（1Byte）	命令（1Byte）	数据长度（1Byte）	数据区（nByte）	CRC 校验（2Byte）
addr	cmd	len	d0-dn	crcH crcL

若无需返回数据，从机返回帧每个字节的意义如下：

从机地址（1Byte）	命令（1Byte）	寄存器地址（2Byte）	寄存器数目（2Byte）	CRC 校验（2Byte）
addr	cmd	regH regL	numH numL	crcH crcL

●**从机地址** 要询问采集卡的地址。每个采集卡可以有 1-247 个地址，若主机询问的地址为 0，则为广播式，每个从机都要响应命令，但是无需回复报文。 （A 版本中无广播功能）

●**命令** 用来指示从机要执行的功能。

●**寄存器地址** 对应命令下需要访问的寄存器的起始地址，有两个字

节，高字节在前。不同命令下可以有相同的地址，但并不是指向同样的寄存器。

●**寄存器数目** 对应命令下需要访问的寄存器的数目，有两个字节，高字节在前。若数目为 0，则填充 0x00 0x00。

●**数据区** 每个命令下数据的意义不同，参考下面各条命令的详细解释。若无数据，则此处没有，无需填充 0。

●**数据长度** 数据区的字节数。

●**CRC 校验** 用来校验一帧数据的正确性，防止传输过程中出错。CRC 校验有两个字节，高字节在前，所校验的字节包括本帧 CRC 前面所有的字节，CRC 校验字产生的算法参考后面附录。

3. MODBUS 协议命令、详细解释、示例

注意：以下所有命令中的数据为 16 进制，地址为默认的 01。

（1）读保持寄存器（可以读出系统设置）（0x03）

读保持寄存器可以读出系统的设置，比如系统的地址、通讯参数等。系统寄存器有如下定义：

寄存器地址(十六进制)	说明(十六进制)
0000（读写）	模块地址
0001（读写）	系统波特率有如下取值： 0000:1200 0001：2400 0002：4800 0003：9600

	0004：19200 0005：28800 0006：38400 0007：57600 0008：115200
0002（读写）	奇偶校验，取值如下： 0000：无奇偶校验 0004：偶校验 0006：奇校验
0003（读写）	0000：8 位数据格式 0010：9 位数据格式
0004（只写）	执行指令 FIFO 填充寄存器
0005（只写）	执行指令，立即执行寄存器

例如，如果模块地址是 1，波特率是 9600，无奇偶校验，数据位是 8 位我们如果想全部读出这些寄存器，做如下操作：

主机发送： 01 03 00 00 00 03 05 CB

从机响应： 01 03 0A 00 01 00 03 00 00 00 00 00 0A 9A 21

请求（主机发送）十六进制		响应（从机发送）十六进制	
从机地址	01	从机地址	01
功能	03	功能	03
起始地址 Hi	00	字节数	0A
起始地址 Lo	00	寄存器值 Hi（00）	00

输出数量 Hi	00	寄存器值 Lo（00）	01
输出数量 Lo	03	寄存器值 Hi（00）	00
crcH	05	寄存器值 Lo（00）	03
crcL	CB	寄存器值 Hi（00）	00
		寄存器值 Lo（00）	00
		crcH	EC
		crcL	B5

（2）写单个寄存器（设置系统的参数）（06）

寄存器说明已经在上面提到，特别注意的是，设置通讯参数时，响应不一定能接收到，所以设置完毕后最好给系统复位一下。

例如，当前模块的地址是 01，我们想设置到 02

主机发送： 01 06 00 00 00 02 08 0B

从机响应： 01 06 00 00 00 02 08 0B（响应完成后即可完成设置）

请求（主机发送）十六进制		响应（从机发送）十六进制	
从机地址	01	从机地址	01
功能	06	功能	06
寄存器地址 Hi	00	寄存器地址 Hi	00
寄存器地址 Lo	00	寄存器地址 Lo	00
寄存器值 Hi	00	寄存器值 Hi	00
寄存器值 Lo	02	寄存器值 Lo	02

crcH	08	crcH	08
crcL	0B	crcL	0B

特别注意：04,05 都可以播放语音，但是是不同的，04 是 FIFO 播放，你一次可以发多条播放指令，模块会依次播放你设置的指令，05 是立即播放，当写 05 寄存器时，如果前面有 FIFO 播放，那么 FIFO 播放会立即停止，马上播放 05 寄存器中的内容，并且会清空 FIFO。一个好的习惯是，一次写完所有 FIFO 播放，如果有下一次 FIFO 播放，首先给 05 寄存器写一个停止播放指令，这样可以防止 FIFO 溢出。

3. 暂时恢复出厂设置的方法

如果不知道模块的通讯参数，可以让模块暂时恢复出厂设置，重新设置模块的参数，恢复出厂设置的方法如下：

1. 模块断电，由于上面有大的电容，断电至少 5 秒钟
2. 把模块上面 HW 和旁边的 GND 相连
3. 给模块上电，此时，模块被动模式恢复到出厂的设置

- 通讯接口： RS-485，TTL232。
- 通讯格式： 1 个起始位，8 个数据位，无校验，1 个停止位。
- 波特率： 9600（可软件更改） 。
- 地址： 1（可软件更改） 。

- 通讯方式： 监控主机与本装置采用一对一（或一对多）主从查询方式
- 数据协议： MODBUS-RTU

4. 通过 MODBUS-RTU 协议重新设置模块的地址，通讯速率等等。设置完成后，模块会自动保存参数。
5. 给模块掉电 5 秒钟把模块上面 HW 和 GND 断开。
6. 重新给模块上电，完成设置

3. 例程 1:使用 MODBUS 协议播放单段语音

命令	功能	备注
0XE0-0XE7	音量调节	在语音播放或者待机状态发此命令可以调节 8 级音量，E0H 最小，E7H 音量最大。
0XF2	循环播放	在语音播放过程中发送此命令可循环播放当前地址语音。
0XFE	停止语音播放	停止播放语音命令。
0X00	播放语音	播放第 0 段语音
0X01	播放语音	播放第 1 段语音
0X02	播放语音	播放第 2 段语音
……	……	……
0XDA	播放语音	播放第 218 段语音
0XDB	播放语音	播放第 219 段语音

假设模块已经设置地址 0x01,9600 波特率，如果我们想播放第 0 段语音，我们做如下设置：

首先，模块需要工作在 1 线串口模式上面（非常重要），设置好后记得重新编译并下载。

写 04 寄存器，值是 0000，具体发送：

主机：01 06 00 04 00 00 C8 0B

从机响应：01 06 00 04 00 00 C8 0B

例程 2:使用 MODBUS 协议播放多段语音

命令	功能	备注
0XE0-0XE7	音量调节	在语音播放或者待机状态发此命令可以调节 8 级音量，E0H 最小，E7H 音量最大。
0XF2	循环播放	在语音播放过程中发送此命令可循环播放当前地址语音。
0XFE	停止语音播放	停止播放语音命令。
0X00	播放语音	播放第 0 段语音
0X01	播放语音	播放第 1 段语音
0X02	播放语音	播放第 2 段语音
……	……	……
0XDA	播放语音	播放第 218 段语音
0XDB	播放语音	播放第 219 段语音

假设模块已经设置地址 0x01,9600 波特率，如果我们想播放第 0、2、4、5、6 段语音（顺序播放），我们做如下设置：

首先，模块需要工作在 1 线串口模式上面（非常重要），设置好后记得重新编译并下载。

首先写 03 寄存器，值是 0000，具体发送：

主机：01 06 00 04 00 00 C8 0B

从机响应：01 06 00 04 00 00 C8 0B

同样的方式写入 03 寄存器，0002,0004,0005,0006，可以不用等待语音播放结束，连续写入即可

具体发送如下：

主机：01 06 00 03 00 02 F8 0B

从机响应：01 06 00 03 00 02 F8 0B

主机：01 06 00 03 00 04 78 09

从机响应：01 06 00 03 00 04 78 09

主机：01 06 00 04 00 00 C8 0B

从机响应：01 06 00 03 00 05 B9 C9

主机：01 06 00 03 00 06 F9 C8

从机响应：01 06 00 03 00 06 F9 C8

CRC 校验程序：

```
const unsigned int    auchCRCHi[] = {
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
```

```
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40,
0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1,
0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0,
0x80, 0x41, 0x00, 0xC1, 0x81, 0x40
} ;
/* CRC 低位字节值表*/
const unsigned int    auchCRCLo[] = {
0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06,
0x07, 0xC7, 0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD,
0x0F, 0xCF, 0xCE, 0x0E, 0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09,
0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9, 0x1B, 0xDB, 0xDA, 0x1A,
0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC, 0x14, 0xD4,
0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3,
0xF2, 0x32, 0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4,
0x3C, 0xFC, 0xFD, 0x3D, 0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A,
0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38, 0x28, 0xE8, 0xE9, 0x29,
0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF, 0x2D, 0xED,
0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60,
0x61, 0xA1, 0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67,
0xA5, 0x65, 0x64, 0xA4, 0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F,
0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB, 0x69, 0xA9, 0xA8, 0x68,
0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA, 0xBE, 0x7E,
0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
```

```
0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71,
0x70, 0xB0, 0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92,
0x96, 0x56, 0x57, 0x97, 0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C,
0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E, 0x5A, 0x9A, 0x9B, 0x5B,
0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89, 0x4B, 0x8B,
0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42,
0x43, 0x83, 0x41, 0x81, 0x80, 0x40
} ;

 unsigned int    crc16(unsigned char *puchMsg, unsigned int usDataLen)
{
      unsigned char uchCRCHi = 0xFF ; /*  高 CRC 字节初始化  */
      unsigned char uchCRCLo = 0xFF ; /*  低 CRC  字节初始化  */
      unsigned long uIndex ; /* CRC 循环中的索引  */
      while (usDataLen--) /*  传输消息缓冲区  */
      {
            uIndex = uchCRCHi ^ *puchMsg++ ; /*  计算 CRC */
            uchCRCHi = uchCRCLo ^ auchCRCHi[uIndex] ;
            uchCRCLo = auchCRCLo[uIndex] ;
      }
      return (uchCRCHi << 8 | uchCRCLo) ;
} //uint16 crc16(uint8 *puchMsg, uint16 usDataLen)
```

单段播放对应表:地址 01 写 05 寄存器（发送的数据是十六进制）

第 1 段：01 06 00 05 00 00 99 CB

第 2 段：01 06 00 05 00 01 58 0B

第 3 段：01 06 00 05 00 02 18 0A

第 4 段：01 06 00 05 00 03 D9 CA

第 5 段：01 06 00 05 00 04 98 08

第 6 段：01 06 00 05 00 05 59 C8

第 7 段：01 06 00 05 00 06 19 C9

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB

第一段：01 06 00 05 00 00 99 CB