Application Note

# ATK-MO1053 模块使用说明

## 高性能音乐播放器模块

使用说明

正点原子

广州市星翼电子科技有限公司

修订历史

| 版本 | 日期 | 原因 |
| --- | --- | --- |
| V1.0 | 2022/06/25 | 第一次发布 |
| V1.1 | 2023/03/07 | 新增阿波罗 F429 与 F767 硬件连接描述 |

# 目 录

# 1，硬件连接

## 1.1 正点原子 MiniSTM32F103 开发板

ATK-MO1053 模块可通过杜邦线与正点原子 MiniSTM32F103 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| MiniSTM32F103 开发板 | PA11 | PA12 | PA6 | PA7 | PA5 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| MiniSTM32F103 开发板 | PA4 | PA8 | - | 5V | GND |

表 1.1.1 ATK-MO1053 模块与 MiniSTM32F103 开发板连接关系

## 1.2 正点原子精英 STM32F103 开发板

ATK-MO1053 模块可通过杜邦线与正点原子精英 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| 精英 STM32F103 开发板 | PE6 | PC13 | PA6 | PA7 | PA5 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| 精英 STM32F103 开发板 | PF6 | PF7 | - | 5V | GND |

表 1.2.1 ATK-MO1053 模块与精英 STM32F103 开发板连接关系

## 1.3 正点原子战舰 STM32F103 开发板

ATK-MO1053 模块可通过杜邦线与正点原子战舰 STM32F103 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| 战舰 STM32F103 开发板 | PE6 | PC13 | PB14 | PB15 | PB13 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| 战舰 STM32F103 开发板 | PF6 | PF7 | - | 5V | GND |

表 1.3.1 ATK-MO1053 模块与战舰 STM32F103 开发板连接关系

## 1.4 正点原子探索者 STM32F407 开发板

ATK-MO1053 模块可通过杜邦线与正点原子探索者 STM32F407 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| 探索者 STM32F407 开发板 | PC0 | PF6 | PB4 | PB5 | PB3 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| 探索者 STM32F407 开发板 | PB11 | PB10 | - | 5V | GND |

表 1.4.1 ATK-MO1053 模块与探索者 STM32F407 开发板连接关系

## 1.5 正点原子 MiniSTM32H750 开发板

ATK-MO1053 模块可通过杜邦线与正点原子 MiniSTM32H750 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| MiniSTM32H750 开发板 | PE6 | PC3 | PB14 | PB15 | PB13 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| MiniSTM32H750 开发板 | PC0 | PE1 | - | 5V | GND |

表 1.5.1 ATK-MO1053 模块与 MiniSTM32H750 开发板连接关系

## 1.6 正点原子阿波罗 STM32F429 开发板

ATK-MO1053 模块可通过杜邦线与正点原子阿波罗 STM32F429 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| 阿波罗 STM32F429 开发板 | PA4 | PI11 | PF8 | PF9 | PF7 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| 阿波罗 STM32F429 开发板 | PB11 | PB10 | - | 5V | GND |

表 1.6.1 ATK-MO1053 模块与阿波罗 STM32F429 开发板连接关系

## 1.7 正点原子阿波罗 STM32F767 开发板

ATK-MO1053 模块可通过杜邦线与正点原子阿波罗 STM32F767 开发板进行连接，具体的连接关系，如下表所示：

| 模块对应开发板 | 连接关系 | | | | |
|---|---|---|---|---|---|
| ATK-MO1053 模块 | RST | DREQ | SO | SI | SCK |
| 阿波罗 STM32F767 开发板 | PA4 | PI11 | PB14 | PB15 | PB13 |
| **模块对应开发板** | **连接关系** | | | | |
| ATK-MO1053 模块 | XDCS | XCS | 3.3V | 5V | GND |
| 阿波罗 STM32F767 开发板 | PB11 | PB10 | - | 5V | GND |

表 1.7.1 ATK-MO1053 模块与阿波罗 STM32F767 开发板连接关系

# 2，实验功能

## 2.1 ATK-MO1053 调试实验

### 2.1.1 功能说明

模块通过 SPI 接口来接受输入的音频数据流，它可以是一个系统的从机，也可以作为独立的主机。这里我们只把它当成从机使用。我们通过 SPI 口向 ATK-MO1053 不停的输入音频数据，它就会自动帮我解码了，然后从输出通道输出音乐，这时我们接上耳机就能听到所播放的歌曲了。ATK-MO1053 通过 7 根信号线同主控芯片连接，分别是：XCS、XDCS、SCK、SI、SO、DREQ 和 RST。其中 RST 是 VS1053 芯片的复位信号线，低电平有效。DREQ 是一个数据请求信号，用来通知主机，ATK-MO1053 可以接收数据与否。SCK、SI(MOSI)和 SO(MISO)则是 ATK-MO1053 的 SPI 接口，他们在 XCS 和 XDCS 的控制下面来执行不同的数据通信。另外，ATK-MO1053 需要外部提供 5V/3.3V 供电，推荐采用 5V 供电，这样，总共需要 9 根线来连接。

### 2.1.2 源码解读

打开本实验的工程文件夹，能够在./Drivers/BSP 目录下看到 ATK_MO1053 子文件夹，该文件夹中就包含了 ATK-MO1053 模块的驱动文件，如下图所示：

```
./Drivers/BSP/ATK_MO1053/
|-- atk_mo1053.c
|-- atk_mo1053.h
|-- patch_flac.h
|-- patch_flac_1.h
|-- patch_spec.h
|-- patch_spec_1.h
|-- vs10xx_1.c
`-- vs10xx_1.h
```

图 2.1.2.1 ATK-MO1053 模块驱动代码

在./BSP/SPI 目录下存放这 ATK-MO1053 模块 SPI 接口的驱动文件，如下图所示：

```
./Drivers/BSP/SPI/
|-- spi.c
|-- spi.h
|-- spi_1.c
`-- spi_1.h
```

图 2.1.2.2 ATK-MO1053 模块 SPI 驱动代码

#### 2.1.2.1 ATK-MO1053 模块接口驱动

在图 2.1.2.2 中，spi.c 和 spi.h 是开发板与 ATK-MO1053 模块通讯而使用的 SPI 驱动文件，关于 SPI 的驱动介绍，请查看正点原子各个开发板对应的开发指南中 SPI 对应的章节。

#### 2.1.2.2 ATK-MO1053 模块驱动

在图 2.1.2.1 中，atk_mo1053.c 和 atk_mo1053.h 是 ATK-MO1053 模块的驱动文件，包含了 ATK-MO1053 模块初始化等 API 函数，下面介绍几个重要的 API 函数。

**1. 函数 atk_mo1053_soft_reset()**

该函数用于软件复位 ATK-MO1053 模块，具体的代码，如下所示：

```
/**
 * @brief       ATK_MO1053 软复位
 * @param       无
 * @retval      无
 */
void atk_mo1053_soft_reset(void)
{
    uint8_t retry = 0;

    while (VS10XX_DQ == 0);                                 /* 等待软件复位结束 */

    atk_mo1053_spi_read_write_byte(0Xff);                   /* 启动传输 */
    retry = 0;

    while (atk_mo1053_read_reg(SPI_MODE) != 0x0800)         /* 软件复位,新模式 */
    {
        atk_mo1053_write_cmd(SPI_MODE, 0x0804);             /* 软件复位,新模式 */
        delay_ms(2);                                        /* 等待至少 1.35ms */

        if (retry++ > 100)
        {
            break;
        }
    }

    while (VS10XX_DQ == 0)
        ;                                                   /* 等待软件复位结束 */

    retry = 0;

    while (atk_mo1053_read_reg(SPI_CLOCKF) != 0X9800)
    /* 设置 ATK_MO1053 的时钟,3 倍频 ,1.5xADD */
    {
        atk_mo1053_write_cmd(SPI_CLOCKF, 0X9800);
    /* 设置 ATK_MO1053 的时钟,3 倍频 ,1.5xADD */

        if (retry++ > 100)
        {
            break;
        }
```

```
    }

    delay_ms(20);
}
```

该函数比较简单，先读取 ATK-MO1053 模块的模式，然后再发送软件复位指令，软件复位指令执行结束后，再设置时钟，待时钟配置完成后，即完成软件复位。

**2. 函数 atk_mo1053_read_reg()**

该函数用于写 ATK-MO1053 模块的寄存器，具体的代码，如下所示：

```
/**
 * @brief       ATK_MO1053 读寄存器
 * @param       address : 寄存器地址
 * @retval      读取到的数据
 */
uint16_t atk_mo1053_read_reg(uint8_t address)
{
    uint16_t temp = 0;

    while (VS10XX_DQ == 0);                         /* 非等待空闲状态 */

    atk_mo1053_spi_speed_low();                     /* 低速 */
    VS10XX_XDCS(1);
    VS10XX_XCS(0);
    atk_mo1053_spi_read_write_byte(VS_READ_COMMAND); /* 发送 ATK_MO1053 的读命令
*/
    atk_mo1053_spi_read_write_byte(address);        /* 地址 */
    temp = atk_mo1053_spi_read_write_byte(0xff);    /* 读取高字节 */
    temp = temp << 8;
    temp += atk_mo1053_spi_read_write_byte(0xff);   /* 读取低字节 */
    VS10XX_XCS(1);
    atk_mo1053_spi_speed_high();                    /* 高速 */
    return temp;
}
```

#### 2.1.2.3 实验测试代码

**1. 函数 demo_modular_test()**

该函数用于与 ATK-MO1053 模块建立通讯，并调用相关函数进行上一首歌的切换与下一首歌的切换操作。具体的代码，如下所示：

```
/**
 * @brief       ATK-MO1053 模块测试
 * @param       无
 * @retval      无
 */
static void demo_modular_test(void)
{
    uint8_t i = 0;
```

```
uint8_t res;
DIR mp3dir;                                     /* 目录 */
FILINFO *mp3fileinfo;                           /* 文件信息 */
char *pname;                                    /* 带路径的文件名 */
uint16_t totmp3num;                             /* 音乐文件总数 */
uint16_t curindex;                              /* 图片当前索引 */
uint8_t key;                                    /* 键值 */
uint16_t temp
uint16_t *mp3offsettbl;                         /* 音乐索引表 */
while (f_opendir(&mp3dir, "0:/MUSI              /* 打开图片文件夹 */
{
    lcd_show_string(10, 160, 200, 16, 16, "MUSIC file folder error", RED);
    delay_ms(200);
    lcd_fill(10, 160, 240, 206, WHITE);         /* 清除显示 */
    delay_ms(200);
}

totmp3num = audio_get_tnum("0:/MUSIC");         /* 得到总有效文件数 */

while (totmp3num == NULL)                       /* 音乐文件总数为 0 */
{
    lcd_show_string(10, 160, 200, 16, 16, "No music files", RED);
    delay_ms(200);
    lcd_fill(10, 190, 240, 146, WHITE);         /* 清除显示 */
    delay_ms(200);
}

mp3fileinfo = (FILINFO *)mymalloc(SRAMIN, sizeof(FILINFO));
                                                /* 为长文件缓存区分配内存 */
pname = mymalloc(SRAMIN, 2 * FF_MAX_LFN + 1); /* 为带路径的文件名分配内存 */
mp3offsettbl = mymalloc(SRAMIN, 2 * totmp3num);
/* 申请 2*totmp3num 个字节的内存，用于存放音乐文件索引 */

while (mp3fileinfo == NULL || pname == NULL || mp3offsettbl == NULL)
 /* 内存分配出错 */
{
 lcd_show_string(10, 160, 200, 16, 16, "memory allocation failed", RED);
 delay_ms(200);
 lcd_fill(10, 190, 240, 146, WHITE);            /* 清除显示 */
    delay_ms(200);
}

atk_mo1053_reset();
atk_mo1053_soft_reset();
```

```
    vsset.mvol = 220; /* 默认设置音量为 220 */
    demo_audio_vol_show((vsset.mvol - 100) / 5);
    /* 音量限制在: 100~250, 显示的时候, 按照公式(vol-100)/5, 显示, 也就是 0~30 */

    /* 记录索引 */
    res = f_opendir(&mp3dir, "0:/MUSIC");          /* 打开目录 */

    if (res == FR_OK)
    {
        curindex = 0;                              /* 当前索引为 0 */

        while (1)                                  /* 全部查询一遍 */
        {
            temp = mp3dir.dptr;                    /* 记录当前 offset */
            res = f_readdir(&mp3dir, mp3fileinfo); /* 读取目录下的一个文件 */

            if (res != FR_OK || mp3fileinfo->fname[0] == 0)
            {
                break;                             /* 错误了/到末尾了,退出 */
            }

            res = exfuns_file_type(mp3fileinfo->fname);

            if ((res & 0XF0) == 0X40)              /* 取高四位,看看是不是音乐文件 */
            {
                mp3offsettbl[curindex] = temp;     /* 记录索引 */
                curindex++;
            }
        }
    }

    curindex = 0;                                  /* 从 0 开始显示 */
    res = f_opendir(&mp3dir, (const TCHAR *)"0:/MUSIC"); /* 打开目录 */

    while (res == FR_OK)                           /* 打开成功 */
    {
        dir_sdi(&mp3dir, mp3offsettbl[curindex]);  /* 改变当前目录索引 */
        res = f_readdir(&mp3dir, mp3fileinfo);     /* 读取目录下的一个文件 */

        if (res != FR_OK || mp3fileinfo->fname[0] == 0)
        {
            break;                                 /* 错误了到末尾了,退出 */
        }
        strcpy((char *)pname, "0:/MUSIC/");        /* 复制路径(目录) */
```

```
        strcat((char *)pname, (const char *)mp3fileinfo->fname);
                                                    /* 将文件名接在后面 */
        lcd_fill(10, 180, lcddev.width, 190 + 16, WHITE);
                                                    /* 清除之前的显示 */
        text_show_string(10, 180, lcddev.width - 30, 16, mp3fileinfo->fname, 16,
0, RED);                                          /* 显示歌曲名字 */
        printf("*********************************\r\n");
        printf("Playing: %s     Kilobit rate: 1411Kbps\r\n", mp3fileinfo->fname);
/* printf 语句中打印的是：正在播放以及比特率 */
        demo_audio_index_show(curindex + 1, totmp3num);
        printf("\r\n");
        key = audio_play_song(pname);               /* 播放这个 MP3 */

         if (key == KEY1_PRES)                      /* 上一曲 */
            {
                if (curindex)
                {
                    curindex--;
                }
                else
                {
                    curindex = totmp3num - 1;
                }
            }
            else if (key == KEY0_PRES)              /* 下一曲 */
            {
                curindex++;

                if (curindex >= totmp3num)
                {
                    curindex = 0;                   /* 到末尾的时候，自动从头开始 */
                }
            }
            else
            {
                break;                              /* 产生了错误 */
            }

            delay_ms(10);
            i++;

            if (i == 25)
            {
                LED0_TOGGLE();
```

```
            i = 0;
        }
    }
    myfree(SRAMIN, mp3fileinfo);  /* 释放内存 */
    myfree(SRAMIN, pname);        /* 释放内存 */
    myfree(SRAMIN, mp3offsettbl); /* 释放内存 */
}
```

从上面的代码中可以看出，该函数实现 ATK-MO1053 模块音乐播放控制，这个函数的执行逻辑是：先查询一遍外接储存卡卡中是否存在 mp3 音乐文件，若存在则进行下一步：为长文件缓存区分配内存以及带路径的文件名分配内存等操作，若是不存在则提示错误。同样的在 LCD 屏幕上以及串口助手上也会显示相应的字样。当初始化成功后便可以选择执行“上一首歌曲的切换”和“下一首歌曲的切换”，当播放完外界存储设备里的歌曲时会自动从头开始。

**2. 函数 audio_play_song()**

保存 atk_mo1053.c 和 atk_mo1053.h 两个文件。把 atk_mo1053.c 加入到/BSP/ATK_MO1053 组下。然后我们打开 demo.c，该文件我们仅介绍一个函数。这里要介绍的是 audio_play_song 函数，该函数代码如下这个函数的功能是普通测试模式，具体的代码，如下所示：

```
/**
 * @brief       播放一曲指定的歌曲
 * @param       pname   : 带路径的文件名
 * @retval      播放结果
 *   @arg       KEY0_PRES , 下一曲
 *   @arg       KEY1_PRES , 上一曲
 *   @arg       其他      , 错误
 */
uint8_t audio_play_song(char *pname)
{
    FIL *fmp3;
    uint16_t br;
    uint8_t res, rval;
    uint8_t *databuf;
    uint16_t i = 0;
    uint8_t key;

    rval = 0;
    fmp3 = (FIL *)mymalloc(SRAMIN, sizeof(FIL));   /* 申请内存 */
    databuf = (uint8_t *)mymalloc(SRAMIN, 4096);   /* 开辟 4096 字节的内存区域 */

    if (databuf == NULL || fmp3 == NULL)rval = 0XFF ;/* 内存申请失败 */

    if (rval == 0)
    {
        atk_mo1053_restart_play();                  /* 重启播放 */
```

```
        atk_mo1053_set_all();                           /* 设置音量等信息 */
        atk_mo1053_reset_decode_time();                 /* 复位解码时间 */
        res = exfuns_file_type(pname);                  /* 得到文件后缀 */

        if (res == T_FLAC)                              /* 如果是 flac,加载 patch */
        {
            atk_mo1053_load_patch((uint16_t *)vs1053b_patch, VS1053B_PATCHLEN);
        }

        res = f_open(fmp3, (const TCHAR *)pname, FA_READ);      /* 打开文件 */

        if (res == 0)                                           /* 打开成功 */
        {
            atk_mo1053_spi_speed_high();                        /* 高速 */

            while (rval == 0)
            {
                res = f_read(fmp3, databuf, 4096, (UINT *)&br);
                                                    /* 读出 4096 个字节 */
                i = 0;

                do                                  /* 主播放循环 */
                {
                    if (atk_mo1053_send_music_data(databuf + i) == 0)
                                                    /* 给 VS10XX 发送音频数据 */
                    {
                        i += 32;
                    }
                    else
                    {
                        key = key_scan(0);

                        switch (key)
                        {
                            case KEY0_PRES:
                            {
                                rval=1;             /* 下一曲 */
                                break;
                            }
                            case KEY1_PRES:
                            {
                                rval=2;             /* 上一曲 */
                                break;
                            }
```

```
                    }
                    audio_msg_show(fmp3->obj.objsize);      /* 显示信息 */
                }
            } while (i < 4096);                 /* 循环发送 4096 个字节 */

            if (br != 4096 || res != 0)
            {
                rval = KEY0_PRES;
                break;                                      /* 读完了 */
            }
        }

        f_close(fmp3);
    }
    else
    {
        rval = 0XFF;                                        /* 出现错误 */
    }
}

myfree(SRAMIN, databuf);
myfree(SRAMIN, fmp3);
return rval;
}
```

该函数，就是我们解码音乐的核心函数了，该函数的参数为歌曲的路径+名字,比如:0:\MUSIC\加州旅馆.mp3。然后为文件结构体以及数据缓存区申请内存，之后重设 ATK-MO1053，然后根据歌曲的类型，如果是 flac 则加载 patch 文件（ATK-MO1053 解码 flac 要打补丁），然后打开这个文件，开始播放。播放的时候，设置 SPI 工作在高速模式（9Mhz 时钟），然后死循环发送音频数据给 ATK-MO1053，在死循环里面等待 DREQ 信号的到来，每次 VS10XX_DQ 变高，就向 ATK-MO1053 发送 32 个字节，直到整个文件读完。此段代码还包含了对按键的处理（暂停/播放、上一首、下一首）及当前播放的歌曲的一些状态（编号、码率、播放时间、总时间、歌曲名字等）的显示（通过 audio_msg_show 函数显示），播放完成后，关闭文件，然后释放内存，完成一首歌曲的播放。

**3. 函数 demo_run()**

```
/**
 * @brief       例程演示入口函数
 * @param       无
 * @retval      无
 */
void demo_run(void)
{
    uint8_t ret;
```

```
/* 初始化 ATK-MO1053 */
ret = atk_mo1053_init();
if (ret != 0)
{
    printf("ATK-MO1053 init failed!\r\n");
    while(1)
    {
        lcd_show_string(10, 97, 200, 16, 16, "MO1053 Error!!!", RED);
        delay_ms(500);
        lcd_show_string(10, 97, 200, 16, 16, "                ", WHITE);
        delay_ms(500);
    }
}

/* 初始化内部 SRAM 内存池 */
my_mem_init(SRAMIN);
/* 初始化 SRAM12 内存池(SRAM1+SRAM2) */
my_mem_init(SRAM12);
/* 初始化 SRAM4 内存池(SRAM4) */
my_mem_init(SRAM4);
/* 初始化 DTCM 内存池(DTCM) */
my_mem_init(SRAMDTCM);
/* 初始化 ITCM 内存池(ITCM) */
my_mem_init(SRAMITCM);

/* 为 fatfs 相关变量申请内存 */
exfuns_init();

f_mount(fs[0], "0:", 1);                    /* 挂载 SD 卡 */
ret = f_mount(fs[1], "1:", 1);              /* 挂载 FLASH */
if (ret == 0x0D)
{
    ret = f_mkfs("1:", 0, 0, FF_MAX_SS);
    /* 格式化 FLASH,1:,盘符;0,使用默认格式化参数 */
    if (ret == 0)
    {
        f_setlabel((const TCHAR *)"1:ALIENTEK");
        /* 设置 Flash 磁盘的名字为: ALIENTEK */
    }
    else
    {
        while (1)
        {
            LED0_TOGGLE();
```

```
            delay_ms(200);
        }
    }
}


/* ATK-MO1053 字库初始化 */
ret = fonts_init();
if (ret != 0)                                   /* 检查字库 */
{
    lcd_clear(WHITE);                           /* 清屏 */
    lcd_show_string(10, 50, 200, 16, 16, "ATOM@ALIENTEK", RED);
    lcd_show_string(10, 70, 200, 16, 16, "SD Card OK", RED);
    lcd_show_string(10, 90, 200, 16, 16, "Font Updating...", RED);
    fonts_update_font(10,110,16,(uint8_t*)"0:",RED);
}

printf("MO1053 OK\r\n");
lcd_show_string(10, 97, 200, 16, 16, "MO1053 OK", RED);

demo_atk_mo1053_memorytest_lcd_ui_init();
printf("Ram Test:0X%04X\r\n", atk_mo1053_ram_test());/* 打印 RAM 测试结果 */
demo_atk_mo1053_sinewavetest_lcd_ui_init();
atk_mo1053_sine_test();
lcd_show_string(10, 140, 200, 16, 16, "                ", WHITE);

while (1)
{
    demo_sensor_test();
    LED0_TOGGLE();
    delay_ms(500);
}
}
```

该函数先检测 SD 卡是不是在位（初始化 SD 卡），存在则继续，不存在在报错，然后检测外部 flash 是否存在字库，如果不存在字库，则报错（此时需要先下载汉字显示实验，更新字库，更新完字库后，再下载本实验），如果存在则继续下面的操作：对 ATK-MO1053 进行正弦测试和寄存器测试，随后，开始播放 MUSIC 文件夹下面的歌曲。最后，我们将 atk_mo1053_set_tone、atk_mo1053_set_volume 和 atk_mo1053_set_effect 等函数加入 usmart 控制，这样我们就可以利用 usmart 来设置 ATK-MO1053 的音效了。

### 2.1.3 实验现象

将 ATK-MO1053 模块按照第 1 节硬件连接中介绍的连接方式与开发板连接，并将实验代码编译烧录值开发板中，如果此时开发板连接 LCD，那么 LCD 显示的内容，如下图所示：


图 2.1.3.1 LCD 显示内容一

这个界面是整个实验的初始界面，当开发板上电后会自动运行。图中显示的是对 RAM 的检测过程，ATK-MO1053 如果得到的值为 0x807F，则表明测试正常；同样的，ATK-MO1053 如果得到的值为 0X83FF 也表明测试正常，得到其它十六进制的结果表明测试失败。（该结果会以串口的形式返回，相应的函数和语句在 demo_run()中实现）


图 2.1.3.2 LCD 显示内容二

这个界面是整个实验的初始界面，当开发板上电后会自动运行。图中显示的是正弦波的检测过程。作为 MP3 播放模块，音频测试的参数有很多项，常规是使用正弦波作为测试信号源，特别是测失真时，要求测试信号源的质量要高。


图 2.1.3.3 LCD 显示内容三

上图是本次实验的主界面，图中显示了歌曲名字以及文件后缀，歌曲时间以及播放进度，音频的千比特率，曲库歌曲数目以及当前歌曲序号，当前音量。

同时，通过串口调试助手输出实验信息。主界面的信息显示，如下图所示：

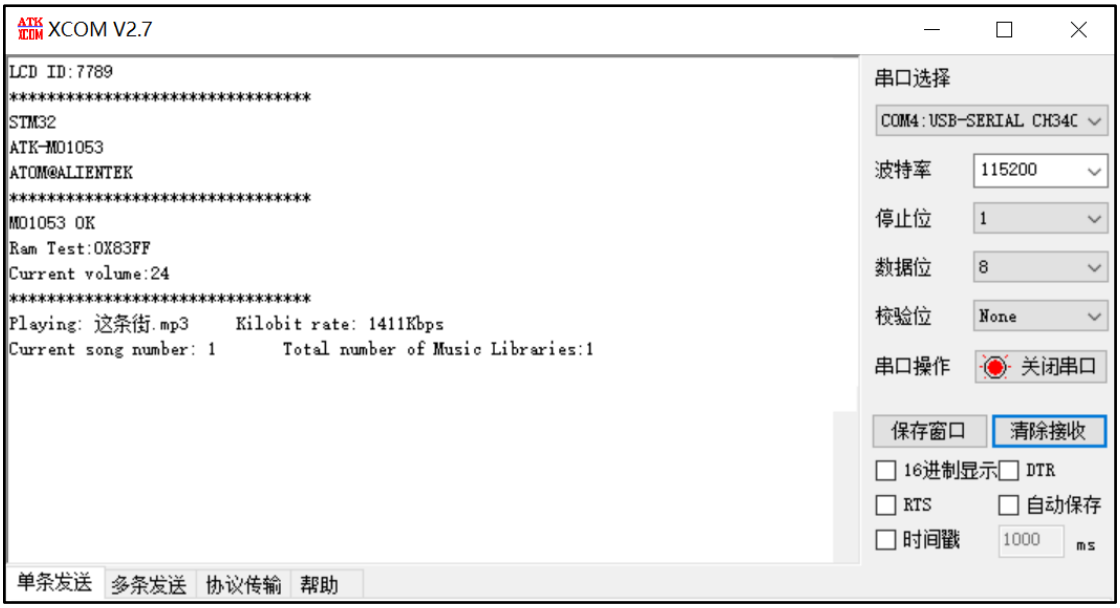


图 2.1.3.4 串口调试助手显示主界面信息

以上是串口助手上实时显示的实验信息。按下 KEY1 进行切换到上一首歌的操作，按下 KEY0 进行切换到下一首歌的操作。

# 3，其他

**1、 购买地址：**

天猫：https://zhengdianyuanzi.tmall.com

淘宝：https://openedv.taobao.com

**2、 资料下载**

模块资料下载地址：http://www.openedv.com/docs/modules/other/VS1053.html

**3、 技术支持**

公司网址：www.alientek.com

技术论坛：http://www.openedv.com/forum.php

在线教学：www.yuanzige.com

B 站视频：https://space.bilibili.com/394620890

传真：020-36773971

电话：020-38271790



正点原子