

SDWb_Lua 视频播放器案例说明

本案例利用 Lua 脚本制作了一个控制串口屏顺序循环播放 U 盘/TF 卡内视频的视频播放器，可以控制 TF 卡中视频播放/暂停、播放结束、播放上一曲、播放下一曲和调节音量。测试时需要将视频文件复制到 TF 卡/U 盘里插到串口屏上。视频必须以数字开头命名放置在 U 盘/TF 根目录里，如图 1 所示。

名称	修改日期	类型	大小
0.avi	2024/9/12 19:01	视频剪辑	25,995 KB
1.avi	2024/9/12 19:02	视频剪辑	21,336 KB

图 1. 视频文件

一. 界面设计

如图 2 所示，背景图片中央位置是视频播放区域，在背景图片底部位置，放置了四个按钮控件，控制播放/暂停、上一曲、下一曲，结束，一个变量图标控件配合“播放/暂停”按钮显示当前按钮的功能。一个拖动调节控件和滑块刻度控件以及进度条控件来控制音量大小，除此之外还有一个变量图标配合显示音量的状态，当音量值为 0 时，该图标控件将显示“静音”图标，音量值不为 0 时显示“音量”图标。

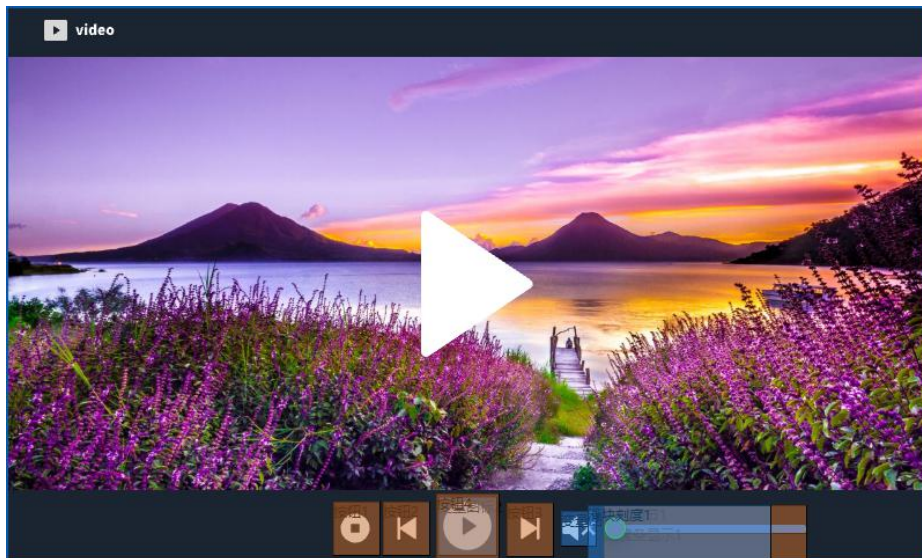


图 2 视频播放器界面

二. Lua 脚本编程

借助 Lua 脚本实现上述视频播放器功能，实际也是通过写寄存器来控制播放

视频的，视频播放寄存器如图 3 所示。0x60-0x6E 寄存器为视频播放相关的寄存器，每个寄存器的功能在表格中都有相应的说明。

寄存器地址	定义	R/W	字节长度	说明
0x60-0x67	Play_Avi_Set	R/W	1	0x5A:申请设置播放器参数
	Avi_Type	W	1	0x00: 单曲播放 VGUS 屏内视频 (默认模式) 0x01: 单曲循环播放 VGUS 屏内视频 0x02: 顺序循环播放 VGUS 屏内视频 0x03: 单曲播放 TF 卡内视频 0x04: 单曲循环播放 TF 卡内视频 0x05: 顺序循环播放 TF 卡内视频 注: 视频文件扩展名必须是*.avi; 单曲播放时文件名必须为阿拉伯数字,如“123.avi”; 顺序播放时文件名可以为字母+数字,如“wuhan123.avi”。
	Play_Position	W	4	视频窗口左上角坐标位置 (XH, XL, YH, YL) 注: (0, 0, 0, 0)表示居中显示。
	Play_Avi_Num	W	2	通过视频文件名选择播放视频曲目, 最多允许 65536 个视频; 仅用于单曲播放, 顺序播放时无效。
0x68-0x69	Vol_Adj_En	W	1	0x5A:申请调整播放视频音量
	Vol	W	1	播放视频音量值, 范围 0x00-0x3F, 上电默认值是 0x3F。
0x6a	Play_Control	W	1	0x5A: 播放/暂停 对于单曲播放方式, 当播放完当前视频后, 系统自动跳回到当前图片界面。
0x6b	Play_Stop	W	1	0x5A: 停止 执行停止播放视频后, 系统自动跳回到当前图片界面, 也可以按照按钮跳转。
0x6c	Play_Next	W	1	0x5A: 播放下一首
0x6d	Play-Prev	W	1	0x5A: 播放前一首
0x6e	Play_Status	R	1	0x00=空闲; 0x01=播放中; 0x02=暂停。

图 3 视频播放寄存器表

2.1 控制视频播放/暂停

“播放/暂停”按钮对应的按钮控件以及变量控件属性如图 4 所示，按钮控件以及变量图标地址为 0x0001，变量 0 和 1 对应“播放”和“暂停”图标，图标指示“播放/暂停”按键当前的功能，所以当点击该按钮控制视频播放后将显示“暂停”图标，当控制视频暂停后将显示“播放”图标。



图 4. 播放/暂停按钮对应的按钮控件以及变量图标控件

“播放/暂停”按钮控件的按键键码设置为 1，用于触发触摸回调函数 `callback_touch (pic_id,key_code,touch_state)`。其对应的代码如下：

```
function callback_touch(pic_id,key_code,touch_state)
--播放/暂停按钮
if pic_id==0 and key_code==1 and touch_state==2 then
--设置播放模式为顺序循环播放TF卡视频，设置视频左上角坐标为(1, 44)
vgus_reg_write(0x60,8,{0x5A,0x05,0x00,0x01,0x00,0x2C,0x00,0x00})
vgus_reg_write(0x6A,1,{0x5A})
vgus_timer_start(1,0,0,200)
end
```

进入触摸回调函数，首先通过写寄存器函数 `vgus_reg_write(reg_addr, write_len, write_table)` 将数据表 `{0x5A,0x05,0x00,0x01,0x00,0x2C,0x00,0x00}` 的 8 个字节数据依次写入 `0x60-0x67` 寄存器中，对照图 3 的视频播放寄存器表，可以看出数据表从左往右，`0x5A` 表示申请播放视频，`0x05` 表示播放模式设置为顺序循环播放 TF 卡/U 盘视频，`0x00,0x01,0x00,0x2C` 表示设置视频播放左上角坐标为 `(1, 44)`，`0x00,0x01` 表示播放视频文件 `0.AVI`。然后往 `0x6A` 寄存器里写入 `0x5A`，控制视频“播放/暂停”接下来用 `vgus_timer_start(timer_id,tmr_mode, count_mode, time out)` 函数，开启定时器 1，定时模式为单次模式，计数方式为向上计数，定时器超时时间设置为 `200ms`，定时器 1 对应的功能是读取当前视频播放状态，然后根据视频播放状态来控制“播放/暂停”的按钮对应的图标的状态，定时时间设置为 `200ms` 是因为屏读取视频并播放需要一定时间，延时一段时间后再去读取当前播放状态会更加准确。定时器 1 对应的代码如下：

```

--定时器1功能
if timer_id==1 then
    vgus_reg_read(0x6E,1,reg_table)
    playback_status=reg_table[1]
    if playback_status==0x00 or playback_status==0x02 then
        --播放状态为空闲或者暂停时显示暂停图标
        vgus_vp_var_write(0x0001,0,0)
    else
        --当视频处于播放中时，控制视频停止，同时显示播放图标
        vgus_vp_var_write(0x0001,0,1)
    end
end
end

```

通过 `vgus_reg_read(reg_addr, read_len, read_table)` 读取存储播放状态的寄存器 0x6E，将读取到的值赋给定义的一个名为“playback_status”的变量，用该变量代表播放状态。然后对该变量进行判断，当其值为 0x00 或者 0x02 时，表示空闲或者暂停时，此时“播放/暂停”按钮应该对应播放功能，用 `vgus_vp_var_write(vp_addr,var_type,var_value)` 函数往 0x0001 地址里写入变量值 0，显示“播放”的图标。当其值为 0x01 时，代表视频处于播放中。然后通过 `vgus_vp_var_write(vp_addr,var_type,var_value)` 函数往 0x0001 里写入变量值 1，显示“暂停”图标。

2.2 控制播放上一曲

“上一曲”按键的控制属性如图 5 所示，



图 5 “上一曲”按钮控件属性

“上一曲”按键键码设置为 2。点击该按键后触发触摸回调函数，其对应的代码如下：

```

--上一曲按钮
if pic_id==0 and key_code==2 and touch_state==2 then
--播放上一曲
vgus_reg_write(0x6C,1,{0x5A})
--开启定时器1
vgus_timer_start(1,0,0,200)
end

```

当触发该按键后进入触摸回调函数，首先通过 `vgus_reg_write(reg_addr, write_len, write_table)` 往 0x6C 寄存器中写入 0x5A 控制播放上一曲。然后同样的开启定时器 1，根据视频播放状态来改变播放/暂停按钮对应的图标。之所以也需要开启定时器 1 是因为当视频处于暂停时，按上一曲或者下一曲视频会变为播放的状态，此时“播放/暂停”按钮对应的图标需要变为“暂停”图标。

2.3 控制播放下一曲

“下一曲”按钮的控件属性如图 6 所示。



图 6 “下一曲”按钮控件属性

“下一曲”按钮的按键键码设置为 3。点击该按键后触发触摸回调函数，其对应的代码如下：

```

--下一曲按钮
if pic_id==0 and key_code==3 and touch_state==2 then
--播放下一曲
vgus_reg_write(0x6D,1,{0x5A})
--开启定时器1
vgus_timer_start(1,0,0,200)
end

```

当触发该按键后进入触摸回调函数，首先通过 `vgus_reg_write(reg_addr, writ`

e_len, write_table)往 0x6D 寄存器中写入 0x5A 控制播放下一曲。然后开启定时器 1。

2.4 控制视频结束播放

“结束”按钮的控件属性如图 7 所示



图 7 “结束”按钮控件属性

“结束”按钮的按键键码设置为 4，点击该按键后触发触摸回调函数，其对应的代码如下：

```
--结束按钮
if pic_id==0 and key_code==4 and touch_state==2 then
    vgus_reg_write(0x6B,1,{0x5A})
    vgus_timer_start(1,0,0,200)
end
```

当触发“结束”按钮按键后进入触摸回调函数，首先通过 vgus_reg_write(reg_addr, write_len, write_table)函数往 0x6B 寄存器中写入 0x5A 控制播放结束，然后开启定时器 1。

2.5 调节视频播放音量

音量调节采用拖动调节的调节方式实现，官网案例工程有类似的拖动调节的案例，视频教程里也有相应的说明，界面设计方面这里就不再赘述。这里通过拖动调节的按钮来触发 Lua 脚本的触摸回调函数，读取到需要改变的音量值，然后将其写到控制视频播放音量的寄存器 0x68-0x69 中，达到改变音量的效果。拖动调节的控件属性如图 8 所示：

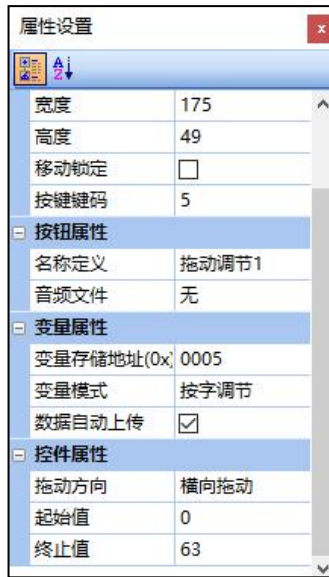


图 8 拖动调节按钮控件属性

当手触发拖动调节的按钮调节当前地址的数值时，需要将当前地址的这个数值读取出来，然后发送给控制音量的寄存器改写音量。这里在 Lua 脚本里定义了一个函数 `volume_set()`，用于将地址的数值读取出来再写入 0x68-0x69 寄存器，其代码如下：

```
function volume set()
    --读取当前需要设置的音量值
    vol=vgus_vp_var_read(0x0005, 0)
    write_table[1]=0x5A
    write_table[2]=vol
    --往0x68寄存器里写入改写的音量值
    vgus_reg_write(0x68,2,write_table)
    if vol==0 then
        --音量为0，显示静音图标
        vgus_vp_var_write(0x0010,0,0)
    else
        --音量不为0，显示音量图标
        vgus_vp_var_write(0x0010,0,1)
    end
end
```

如代码所示，首先通过 `vgus_vp_var_read(vp_addr, var_type)` 函数，读取拖动调节控件对应的变量地址 0x0005 中的数值赋值给“vol”变量，然后通过 `vgus_reg_write(reg_addr, write_len, write_table)` 函数，往 0x68 寄存器里写入 0x5A，表示申请写入音量值，往 0x69 寄存器写入读取到的音量值“vol”。除此之外还需要根据当前音量值来显示音量状态图标。音量状态对应的变量图标控件属性如图 9 所示。

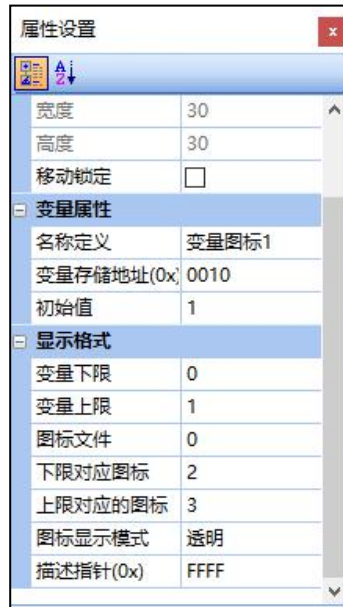


图9 “音量状态”变量图标控件属性

变量图标控件的变量地址为 0x0010，当音量值“vol”为 0 时，往地址里 0x0010 里写入 0，显示“静音”图标，当音量值不为 0 时，往地址 0x0010 里写入 1，显示“音量”图标。

如图所示，拖动调节按钮的按键键码为 5，当拖动该控件触发触摸回调函数后，其代码如下：

```
-- 触发调节音量滑动调节按钮
if pic_id == 0 and key_code == 5 then
    --持续触发拖动调节按钮
    if touch_state == 3 then
        --开启定时器2，实时读取需要修改的音量值，同步改写寄存器中音量值
        vgus_timer_start(2,1,0,80)
```

当持续触发拖动调节时，需要实时读取地址的值，并写入音量寄存器中，所以这里打开定时器 2，计时模式为周期模式，计数方式为向上计数，定时器超时时间为 80ms，意味着 80ms 读一次音量并将读取的数值写入音量寄存器。其定时器 2 对应的 Lua 代码如下：

```
--定时器2功能：读取需要改写的音量值，将音量值写入音量寄存器中
if timer_id==2 then
    volume_set()
end
```

如代码所示，只需要在定时器回调函数中调用之前定义好的设置音量的 volume_set() 函数即可。

当松开拖动调节按钮后，其代码如下：


```
--松开滑动按钮
elseif touch_state==2 then
    --读取最终音量值，将寄存器音量值改写为该值
    volume_set()
    --关闭定时器2
    vqus_timer_stop(2)
end
end
```

同样调用一次设置音量的 `volume_set()` 函数，获取到最终调节的音量值写入音量寄存器，然后再关闭定时器 2，音量调节完成。