Design of W-Player Program based on LabVIEW

Wang Jing

North China Electric Power University, Baoding, Hebei 071000, China;

ww170108@163.com

Abstract

This article uses the LabVIEW development tools to implement the player's design simulation, including reading the player file list, starting playback, pausing playback, and transforming songs. It embodies the principle, design method and implementation skills of virtual instrument based on LabVIEW.

Keywords

LabVIEW, player.

1. Introduction

The virtual instrument W-player is mainly used to play music files in a stored computer. It is based on general-purpose computer hardware and operating system to achieve music functions. The default easy-to-recognize format is WAV format in the VI programming. So at design time importing the sound file information vi by reading the playlist. The designed music player interface includes playing song names, volume levels, playback progress bars and stop buttons.

Program

2. Program flow design

Program flow design: set the path, determine whether the path is valid → invalid error; valid, execute the next stage program → call the media player, display the song name, set the stop function button → end.

2.1 Event structure

The event structure in this program is mainly used to respond to the interface. Including switching songs forward, switching songs backwards, stopping playback, exiting programs, etc. When the user clicks the corresponding toggle button with the mouse and exits the program button the event structure detects that an event occurs and jumps according to the event that occurred. Execute the program to the corresponding branch, complete the corresponding action and respond to the events occurring on the interface.

3. Program structure

The program mainly adopts the programming mode combining sequential structure, cyclic structure and event structure. The sequential structure is mainly used to execute the program in the corresponding order. The sequence structure is mainly used to obtain the next or a previous song and stop the playback of the current song in the switching of the song part .The loop structure is used to process data processing related to WAV file reading. The event structure is used to implement the response to the interface. When the user performs a click-and-movement operation on the interface the event structure captures the events that occur correspondingly.

3.1 Description of the programming structure

3.1.1 Initialization

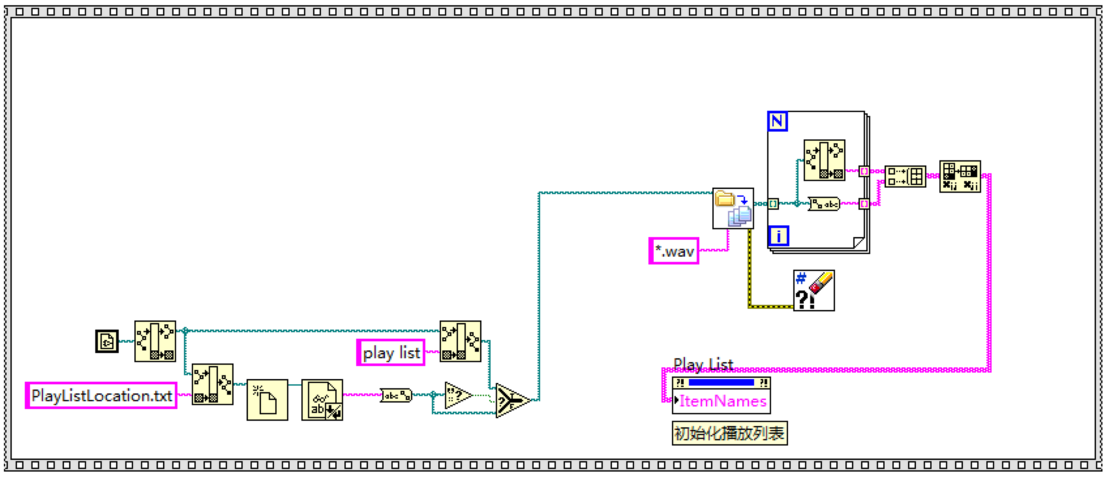

Figure 3-1 Initialization structure

The initialization program mainly obtains the address of the program and splits the address then obtains the address of the folder where the playlist is located. And then obtains the names of all the play files by reading all the wav play files in the folder where the playlist is located. And save it in the playlist control for easy display on the interface.

3.1.2 Event Structure

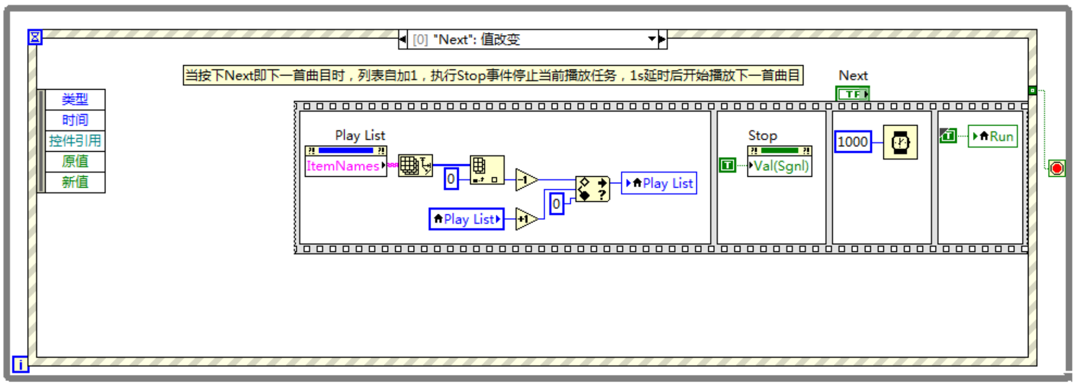

Figure 3-2 Event structure

The event structure is used to implement the response to the interface. When the user performs a click-and-movement operation on the interface the event structure captures the events that occur correspondingly. The function of the program event structure is included. In addition to direct operations on WAV files such as music switching, pause, stop, etc. In the above figure is the displayed time structure the program waits for the interface to switch the song event. When the user presses the button to switch to the next or switch back to the previous one. The event structure corresponds to the user's operation and enters the corresponding operation. The event branch performs the operation. The above figure shows the operation response of the system when the next button is pressed. The program first reads the playlist control to get the name and position of all the played songs and then selects the next song in the playlist to play thus achieving the function of switching songs.

3.1.3 Playback control loop

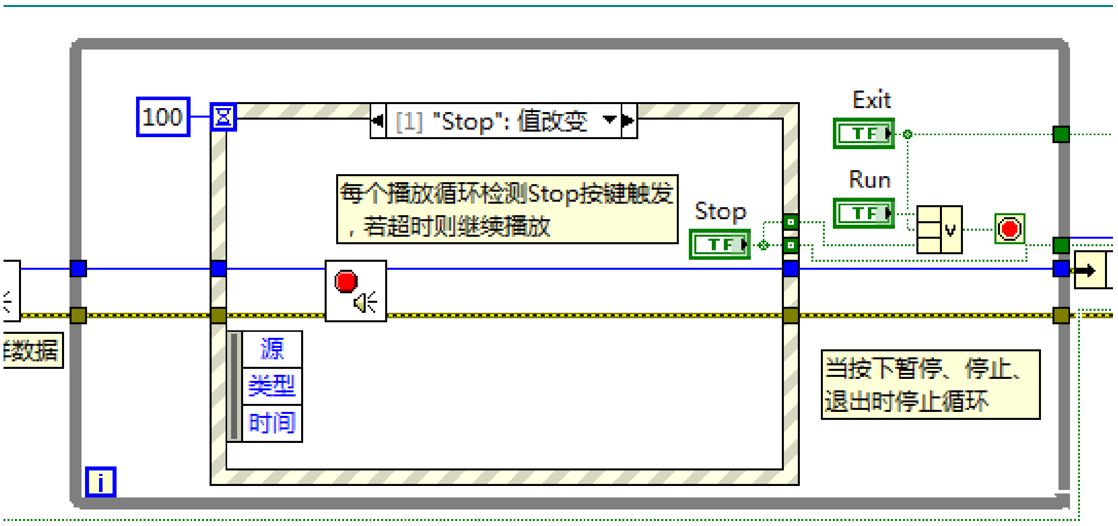
100
[1] "Stop": 值改变
每个播放循环检测Stop按键触发，若超时则继续播放
Stop
Exit
Run
源
类型
时间
数据
当按下暂停、停止、退出时停止循环

Figure 3-3 Inner loop structure

Inner loop: There is an event structure in the inner loop . This event structure has only two branches: the timeout branch and the stop control value change branch. The timeout time is 100ms and the program waits for the stop value change on the interface in this loop. The event occurs. When the stop control value changes within 100ms, that is, when the user presses the event structure detects the occurrence of the event. The program jumps to the branch and executes the stop sound output play function thereby stopping the play. Effect; when the stop control value does not change in 100ms the event structure enters the timeout branch and is used to execute some default programs. Outside the event structure when the exit button and the run button value have a value of true the program exits the current inner loop.

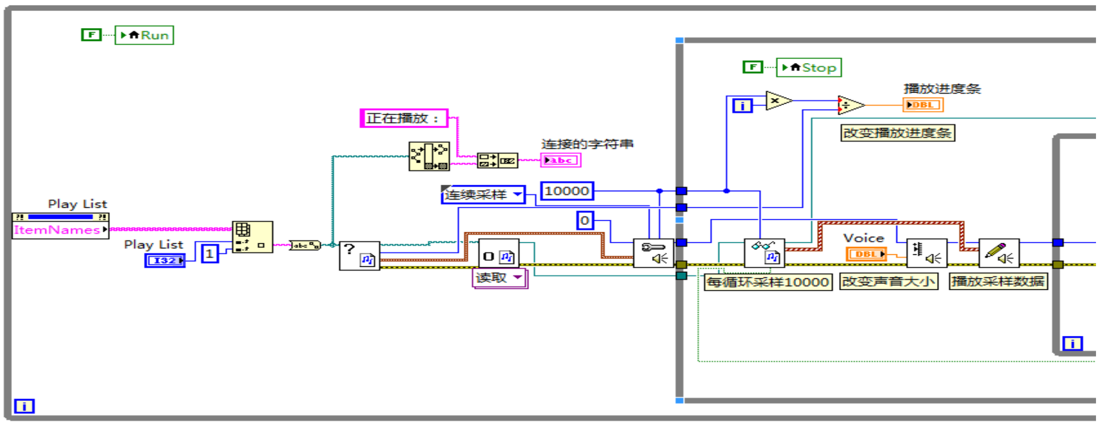

Figure 3-4 Outer loop structure (1)

The outer loop consists of two parts. The upper part is the first part. The main execution content of the program is to read the names of all the play files in the playlist and index the wav play file name corresponding to the line specified by the playlist control. The path is input to the sound file information .vi. The format information of the playback file is obtained through this VI and the total number of samples per channel. The music file currently needed to be played is opened and the reference handle of the sound file is obtained. After that the format of the sound output is configured to prepare for outputting sound. This configuration process needs to set the sampling type, device ID, number of samples per channel and the root of the sound file is information. After the program enters the sub-loop the sound file to be played is read, the data information to be played is obtained and the size of the playback sound can be adjusted and adjusted, and finally the write sound output is output to output the information of the audio file to the playback device. At the same time the progress bar option is set to update the current playback progress in real time and display on the front panel interface to increase the user-friendly interface so that the user can understand the current playback progress.

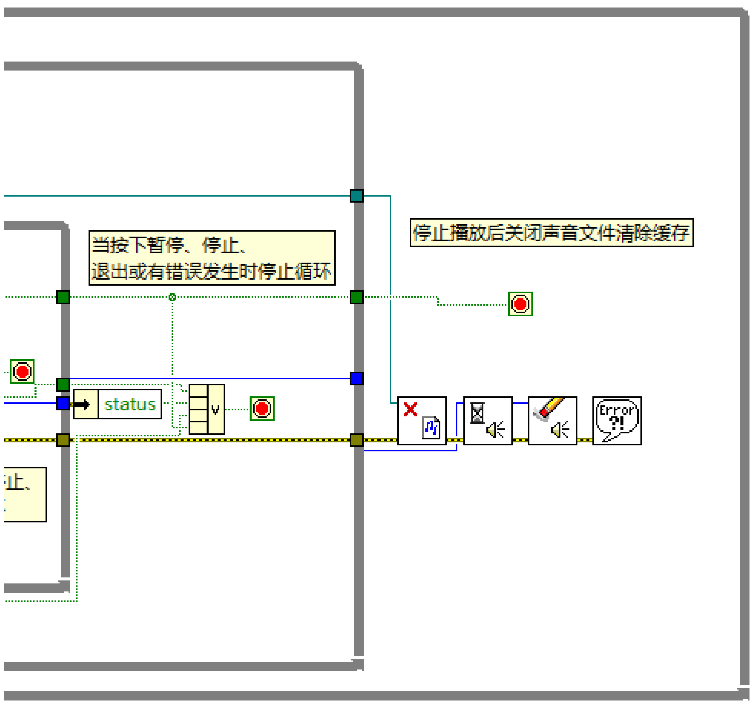

Figure 3-5 Outer loop structure (2)

The second part is that performing the subsequent response operation after the stop play control is pressed and closing the sound file that has been opened, avoiding the access permission of the sound file, causing other programs to access the file, and then setting the sound output. To wait for the mode and clear the sound output the device stops playing the sound and the cached content is cleared. The task returns to the default state and the resources related to the task are cleared so as to avoid wasting computer resources and finally set the error handling. It is used to handle errors and finally clear the error.

3.1.4 Implementation of the currently playing track display

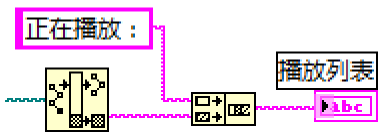

Figure 3-6 Display of the currently playing track

Use the split path control to split the "file name" and the connection string to connect the "playing" constant to display the currently playing track. Add a string display control on the front panel to display the currently playing track.

3.1.5 Path and its transformation implementation

Path-to-string conversion lookup: In the block diagram, follow the steps: function-programming-string-string/array/path conversion-path-to-string conversion to find the path to string conversion function 。

Implementation of the conversion: When the path of the file is obtained through the path operation function the path data type to the string data type can be converted by the path-to-string conversion function thereby achieving the result of displaying the path.

3.2 Overall structure of the program

Integrate the various module programs described earlier in this chapter, and add an incorrect file path error. After finishing the improvement, the overall program structure diagram shown in Figure 3-5 is obtained: it contains an initialization sequence structure block diagram. A block diagram of an event structure for responding to interface song switching, a loop structure for playing music and stopping music playback, and an embedded self-loop structure and event structure.

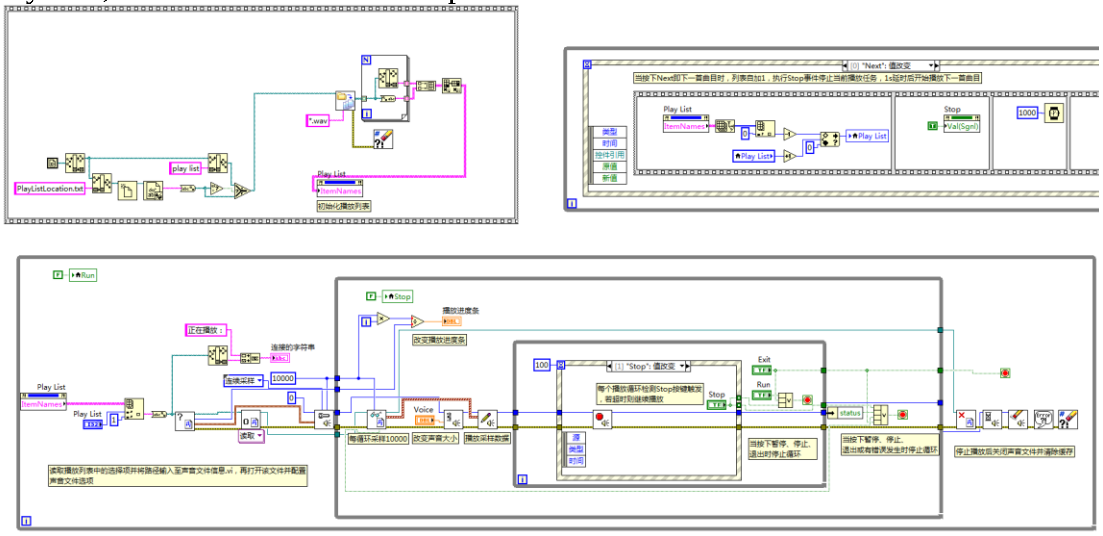

Figure 3-8 Overall block diagram

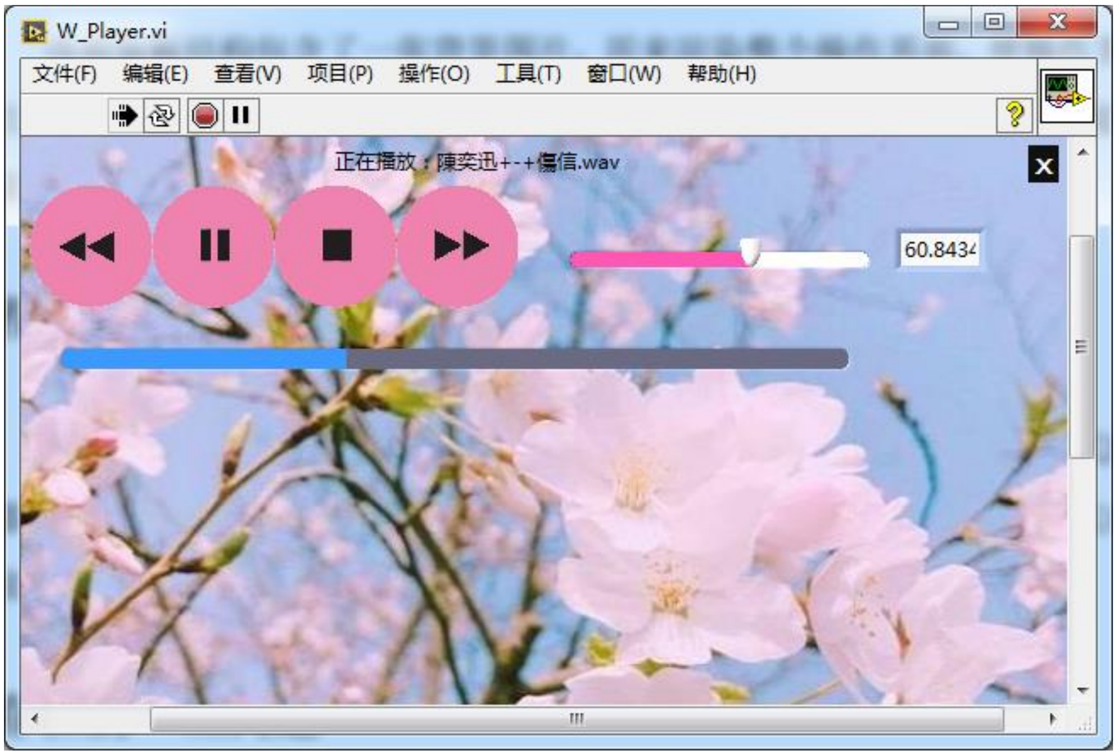
W_Player.vi
文件(F) 编辑(E) 查看(V) 项目(P) 操作(O) 工具(T) 窗口(W) 帮助(H)
正在播放：陳奕迅+-+傷信.wav
60.8434

Figure 3-9 Front panel

4. Conclusions

This program realizes the structural design, code programming and interface design of the simple music player program by using LabVIEW as the main programming language. The music player can realize the reading of the playlist, the playing of the music, the pause, the switching of the music to the previous one and the switching of the next function. The volume can be adjusted as needed in real time. In the structural design of the program the reuse of multiple structures such as sequential structure, cyclic structure and event structure is adopted. Writing code responds to different operations enables real-time response to user actions.

References

[1] Chen Fei. LabVIEW programming and project development practical tutorial, Xi'an: Xidian University Press, 2016.

[2] Hu Ganmiao. Design and application of labview virtual machine, Beijing: Tsinghua University Press, 2016.