

VOICE CONTROL WITH APPLE TV

INTRODUCTION

In our examples that show voice control of the Apple TV, we use the voice-to-text capabilities of the Crestron TSW-xx52/60 panels and TSR-302 remote to achieve two things. First of all, we provide a mechanism for the user to fill in the text entry boxes for things like the Apple TV search functions. This avoids the tedious up/down/left/right that cannot be avoided using other control techniques.

The second feature is voice control for moving around playing media. This requires parsing the input string to extract a time value from the input, and passing this on to the Apple TV so that it can jump to the requested point. Anything beyond trivial absolute positions requires that we know the current position, so that relative jumps can be made, and this is also possible because the Apple TV module offers this feedback.

For this application note, we have provided a SIMPL+ module which performs some rudimentary pattern matching for a limited set of commands:

- "go to <absolute time>"
- "go to start"
- "go forward <relative time>"
- "go back <relative time>"

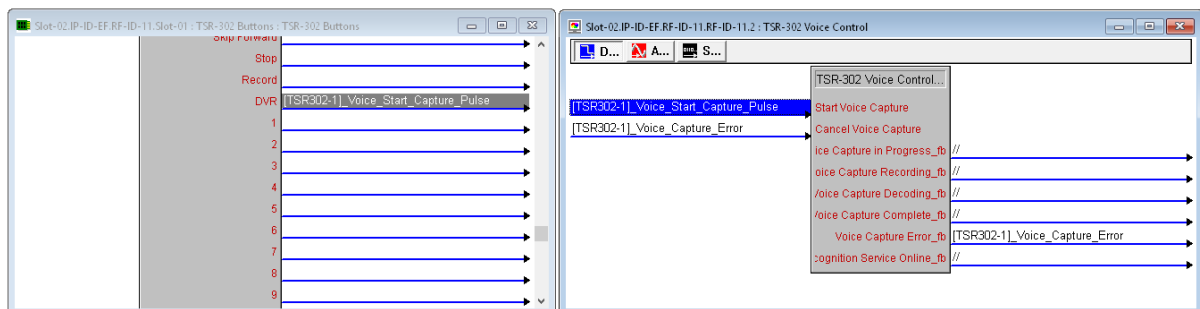
This SIMPL+ module can, of course, be extended to accommodate a broader range of commands, alternative phrasing or additional languages, but that is not discussed in this application note.

The application note describes one way that a suitable panel or remote (in our case, the TSR-302) can be programmed to capture and process the voice commands. In any practical situation, there will naturally be more code, such as cross points to handle source control, button routing and so on, but we have omitted all of this for clarity.

CAPTURING THE SPEECH TEXT

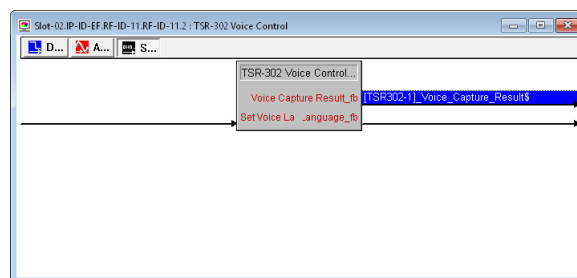
First of all, we need to address the matter of actually bringing the speech into the control system. This can present problems in itself, where the speech-to-text service isn't available, or the capture process gets "stuck". The first issue is addressed through correct DNS settings in the remote, and a reasonable connection to the internet (as the speech-to-text feature requires a connection). To mitigate against the second issue, the example code "resets" the voice capture process whenever an error is encountered.

Within your SIMPL windows program, add the Voice Control device extender to the touch panel, and assign a button to start the capture process. Then tie the Voice Capture Error Fb to the Cancel Voice Capture (rising edge triggered) signal, so that if anything goes wrong, the capture is abandoned.



You can tie the various feedback signals to the UI to inform the user what is going on. We have only tested this using manual voice capture, but it should also work using the "always listen" mode.

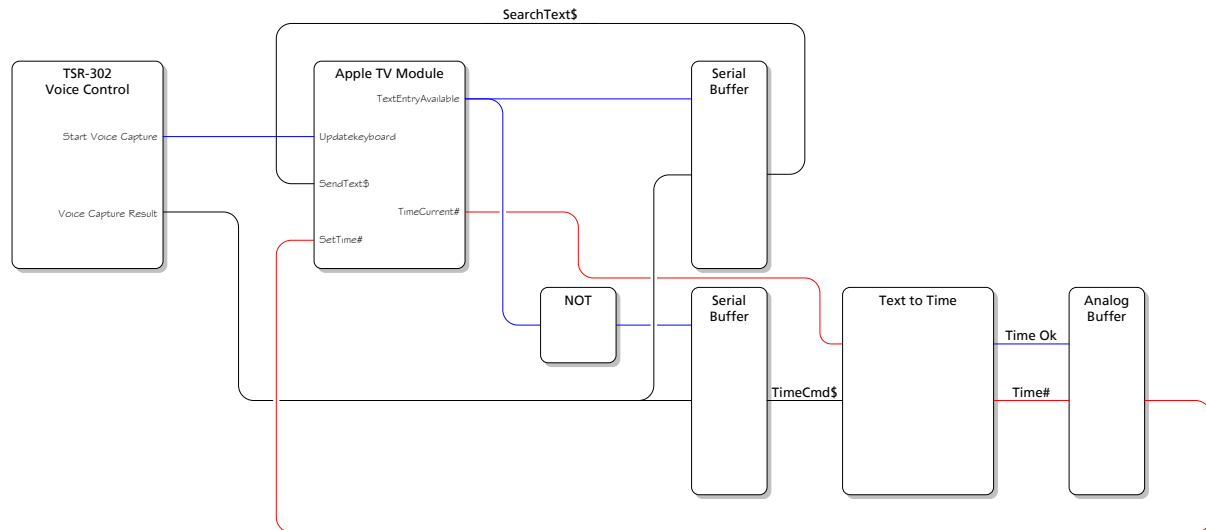
Finally, connect a signal to the Voice Capture Result Fb signal which is where the processed speech to text output will appear. It's worth testing this to ensure that you do in fact get text output.



We also route the "start capture" signal (defined as "[TSR302-1]_Voice_Start_Capture_Pulse" in this example) through to the Apple TV source control logic to "prime" the system for (possible) text input.

PROCESSING THE SPEECH TEXT

The diagram below shows an example logic flow, which will walk through. Read the diagram from the left (the TSR Voice Control module) to right, though – as you will see – some signals feed back into the Apple TV module.



1. First of all, the Start Voice Capture signal goes high. This instructs the Apple TV module to poll the device of its current text entry state. If and only if the Apple TV is ready for text input (i.e. it is displaying an on screen keyboard), the TextEntryAvailable signal will go high.
2. So, depending on the state of TextEntryAvailable (high or low), the NOT will ensure that one, or the other, of the two serial buffers will be enabled.
3. The top serial buffer is enabled when TextEntryAvailable is HIGH, and this means that when the Voice Capture Result is sent, it will be routed through the top serial buffer and back into the SendText\$ input on the module. The module is already prepared for the text entry and the spoken text will appear in the text edit box. You could also provide visual feedback to the user (using the TextEntryAvailable signal) to show them that the Apple TV has responded in the affirmative to text entry.
4. The lower serial buffer is enabled when TextEntryAvailable is LOW, and this then routes the spoken text through to the Natural Language Processer element. In our example, this provides Text to Time conversion. The Text to Time module takes an absolute time (in seconds) which is provided by the Apple TV feedback, and then applies the spoken command. If parsing is successful, the resulting absolute time is output and the analog buffer is triggered to pass the time back to the Apple TV on the SetTime# input, commanding the Apple TV to move jump to the indicated place in the media.

This example and the associated example modules (such as the Text to Time) is provided as-is.