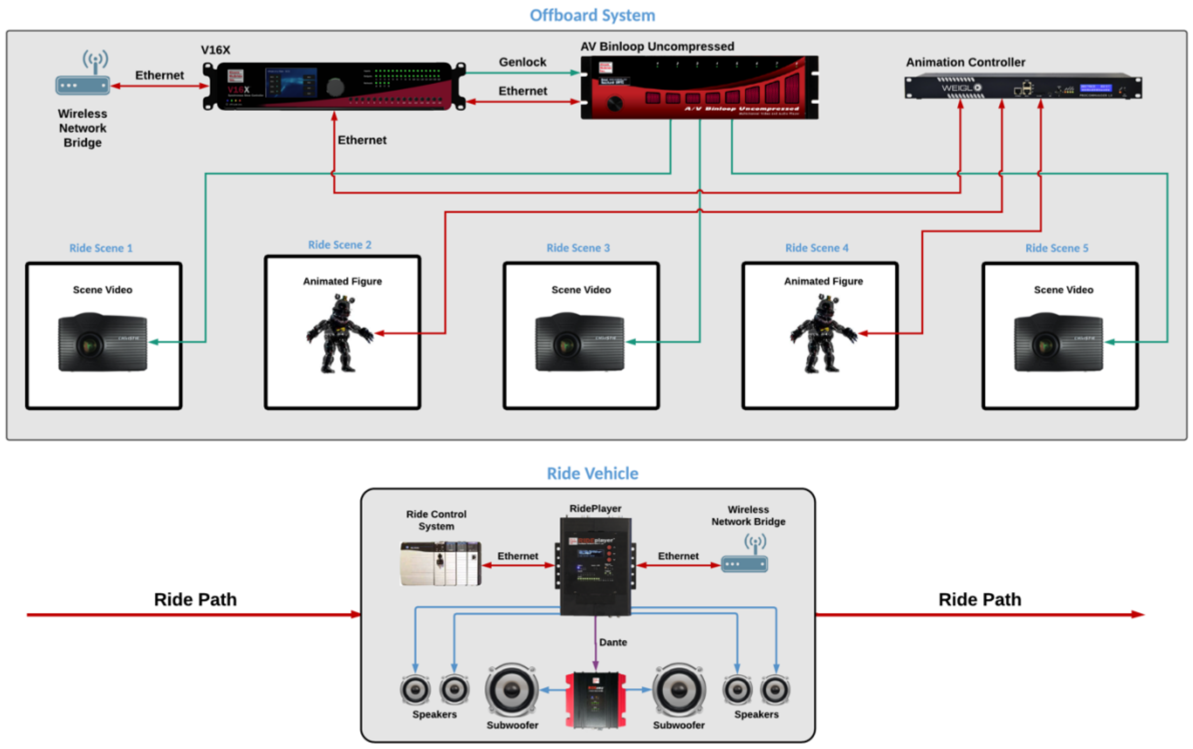
Dark Ride Application



Overview

Dark Rides are often the main attractions of major theme parks. They incorporate everything from highly themed scenery, props, video, audio, animatronics, lighting, and ride systems. The tricky part is getting these systems to synchronize nicely with one another to provide a great guest experience. This is especially difficult considering that most ride vehicles have no hard-wired connection to the rest of the ride system. This application note demonstrates how revolutionary products such as V16X, RidePlayer, RideAmp, and the AV Binloop Uncompressed make these types of applications a breeze to implement.

How It’s Done

Introduction

A typical dark ride centers around the ride system itself. Rides are generally programmed to follow a specific path along a track and sometime even have a built-in motion base that moves around to thrill guests. As fascinating as that may be, a moving vehicle is not enough to make a fun dark ride. For this you need other elements like audio, video, lighting, and animated figures that coordinate with the ride system to provide an immersive experience.

The biggest challenges for building these types of attractions is ensuring the synchronization of the various types of gear with the ride system. This is difficult because ride systems aren’t really designed with audio-visual-sensory experience in mind. They are designed with safety in mind because it’s generally frowned upon to maim and/or kill guests.

This is where it is nice to have a system that is uniquely designed to coordinate with the ride system as well as all the other aspects of the attraction in a tightly synchronized manner. This involves communication with the PLC devices that operate the ride as well as a precise synchronization method that works over wireless connections.

This guide explains how to use the V16X and RidePlayer to coordinate synchronization between the ride vehicles and the offboard systems. It also ties in our AV Binloop Uncompressed to provide video playback for the projections commonly used in modern dark rides.

Integrating the System

As you can see from the application diagram, we're going to implement a dark ride that incorporates a ride system, video projection, animation control, and onboard audio. By monitoring data within the onboard ride control PLC, each RidePlayer is able to sense its location (scene) along the ride path. When the vehicle reaches specific locations within the ride, RidePlayer coordinates synchronous playback of onboard audio with the offboard video and animation control for that location.

System Components

Let's look at the gear we have designed into this system and its role in implementing the dark ride.

RidePlayer – Onboard Synchronous Audio Player and Show Controller

This product is designed to endure the high-vibration environments of ride vehicles and to provide many features to reduce the need for auxiliary equipment aboard the vehicle. Some of the key features include 16 channels of audio output as well as onboard show control. Other useful features like DSP, amplification, network audio, power monitoring, and advanced synchronization are all rolled into a nice compact, rugged, and energy efficient package to make this product the ultimate onboard audio and control solution.

For this dark ride system, we are using a RidePlayer unit for each ride vehicle to manage onboard playback and control. Each RidePlayer will maintain very precise clock synchronization with the offboard V16X using NTP, and will also communicate with an onboard PLC to trigger the AV experience based upon vehicle location.

RideAmp-350Q – High-powered Onboard Amplifier

The built-in amplifiers of RidePlayer are powerful enough to drive the main speakers on most vehicles, but louder or larger speakers (i.e. subwoofers, transducers) benefit from high-power amplification.

RideAmp-350Q delivers up to 4 channels of amplification, each capable of up to 350W of output power.   This product is also designed to endure high-vibration environments and operate from a DC power source, making it ideal for onboard audio applications. To ensure a nice clean audio signal, RideAmp leverages the digital Dante/AES67 network audio interface of RidePlayer. For vehicle configurations that require multiple amplifiers, this connection can be easily daisy-chained without the need for additional network switch hardware.

PLC – Ride Control System

It's very typical for dark ride vehicles to have an onboard PLC that is responsible for controlling all vehicle motion. This system is often precisely aware of the vehicle's operating status and position along the ride path. RidePlayer can easily access this information for the purpose of triggering the show experience based on the vehicle's location. For the sake of this example, we'll simulate the use of an Allen Bradley ControlLogix PLC since they are so commonly used for this type of application.

V16X – Synchronous Show Controller

This device primarily manages the offboard systems, but also coordinates with the RidePlayer units onboard the vehicles to ensure that both systems synchronize perfectly with one another. In this example, the V16X will distribute a master clock to all RidePlayers using NTP to ensure everyone is triggering based on the same reference.

A/V Binloop Uncompressed – Multi-channel Synchronous Video Player

The purpose of this unit is to provide multiple channels of uncompressed playback for the various scenes of this dark ride. Content is stored on solid-state media drives as uncompressed Targa sequences. This product physically connects to each projector using a 3G-SDI connection to provide video without the need for extension devices. This device also connects to the offboard RidePlayer via Ethernet so that video clips can be played at the appropriate time and synchronized with other devices in the system. Genlock is provided from the offboard RidePlayer as well to ensure perfectly synchronized timing between the show control timeline and video playback

Animation Controller

It's very common to have many animated figures in dark ride attractions. For this example, we will perform this control using a Weigl ProCommander that is triggered from the V16X.

Network Infrastructure

Although it is not shown in the system diagram, it is implied that a system like this would consist of network switches, routers, and wireless bridges as needed. A reliable wireless connection between the offboard system and the ride vehicles is essential to consistent operation. WIFI may not always be the best tool for this, but there are several wireless networking technologies that are well-suited for reliable low-latency networking.

Implementing Control

Properly implementing the show control interface between the onboard RidePlayer and offboard V16X systems is critical to synchronous AV for this application. Both of these devices are configured and programmed using our WinScript Live software.

Show Control Programming

We need to create two show control scripts to properly implement this dark ride application; one for the offboard V16X and another for the onboard RidePlayer. The ride vehicle script will be responsible for monitoring the ride control PLC tags for triggers and coordinating synchronous playback for both the onboard and offboard systems.

These two scripts have been included with this application note and are appropriately named:

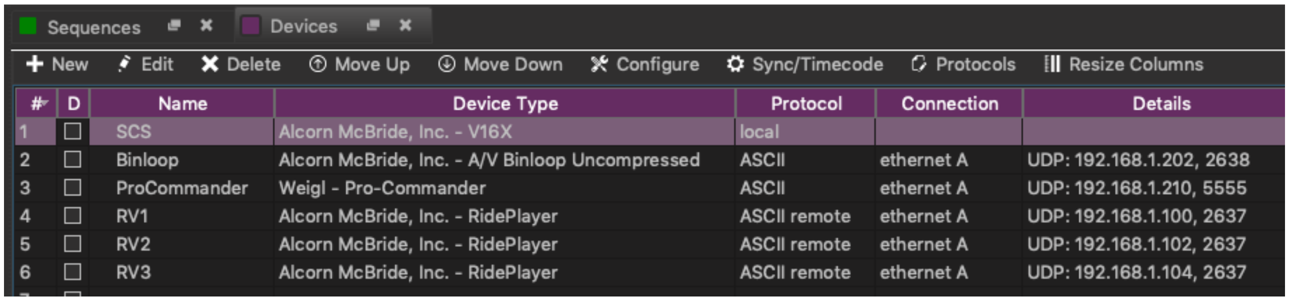
- **Dark Ride – Offboard.WSL**
- **Dark Ride – RV.WSL**

Although Alcorn McBride goes through great effort to make this programming significantly easier than many other control systems, there is a learning curve to using WinScript Live, V16X, and RidePlayer. If you're looking to learn more about using these tools, Alcorn McBride offers free training in the form of interactive in-person classes and online courses.

Devices – Offboard

A great first step when writing any script is to configure the list of devices that will be connected to the V16X. This involves browsing the comprehensive library of devices in the WinScript library by manufacturer and model number, choosing the device, and then configuring the physical connection to the device (i.e. Ethernet, Serial, etc.).

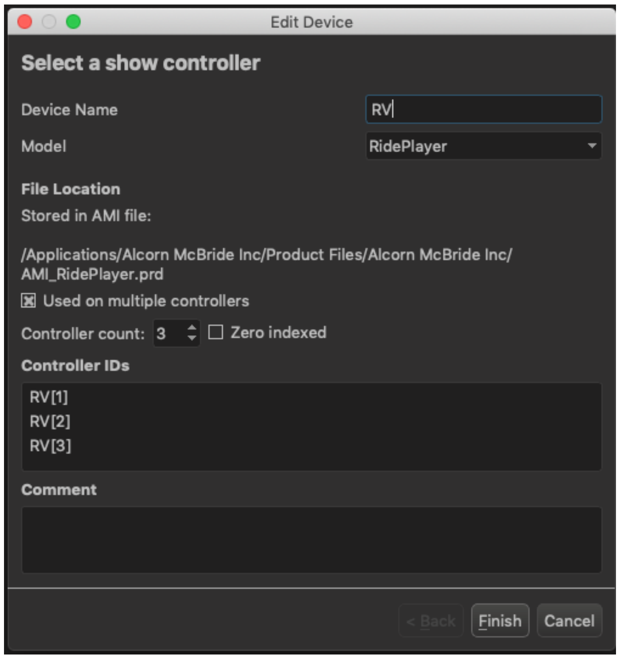
The V16X must connect with the Binloop, the animation controller, as well as each of the RidePlayers in the ride vehicle fleet:



#	D	Name	Device Type	Protocol	Connection	Details
1	☐	SCS	Alcorn McBride, Inc. - V16X	local		
2	☐	Binloop	Alcorn McBride, Inc. - A/V Binloop Uncompressed	ASCII	ethernet A	UDP: 192.168.1.202, 2638
3	☐	ProCommander	Weigl - Pro-Commander	ASCII	ethernet A	UDP: 192.168.1.210, 5555
4	☐	RV1	Alcorn McBride, Inc. - RidePlayer	ASCII remote	ethernet A	UDP: 192.168.1.100, 2637
5	☐	RV2	Alcorn McBride, Inc. - RidePlayer	ASCII remote	ethernet A	UDP: 192.168.1.102, 2637
6	☐	RV3	Alcorn McBride, Inc. - RidePlayer	ASCII remote	ethernet A	UDP: 192.168.1.104, 2637

Devices – Ride Vehicle

We will configure the RidePlayer script so that it can be easily deployed to multiple vehicles. This configuration method uses a single device name (i.e. "RV") that can be referenced throughout the script. When it comes time to deploy this script to multiple vehicles, individual RidePlayers will be uniquely identified using this name combined with an array index (i.e. "RV[1]", "RV[2]", etc.). These identifiers make it clear which vehicle systems are operating and whether they are up to date with the most recent script and media content. From the connection screen, you can also choose to deploy updated content to individual units one at a time, or to multiple vehicles simultaneously. This approach avoids a tremendous amount of unnecessary duplication by managing large fleets of vehicles with a single script.



Select a show controller

Device Name	RV
Model	RidePlayer

File Location
Stored in AMI file:

/Applications/Alcorn McBride Inc/Product Files/Alcorn McBride Inc/AMI_RidePlayer.prd

☒ Used on multiple controllers

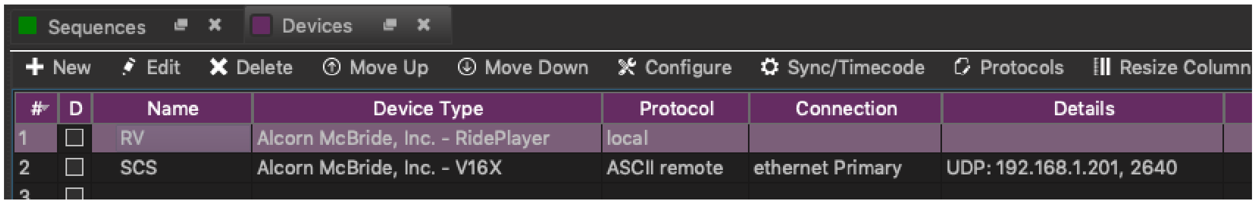
Controller count: 3 ☐ Zero indexed

Controller IDs

RV[1]
RV[2]
RV[3]

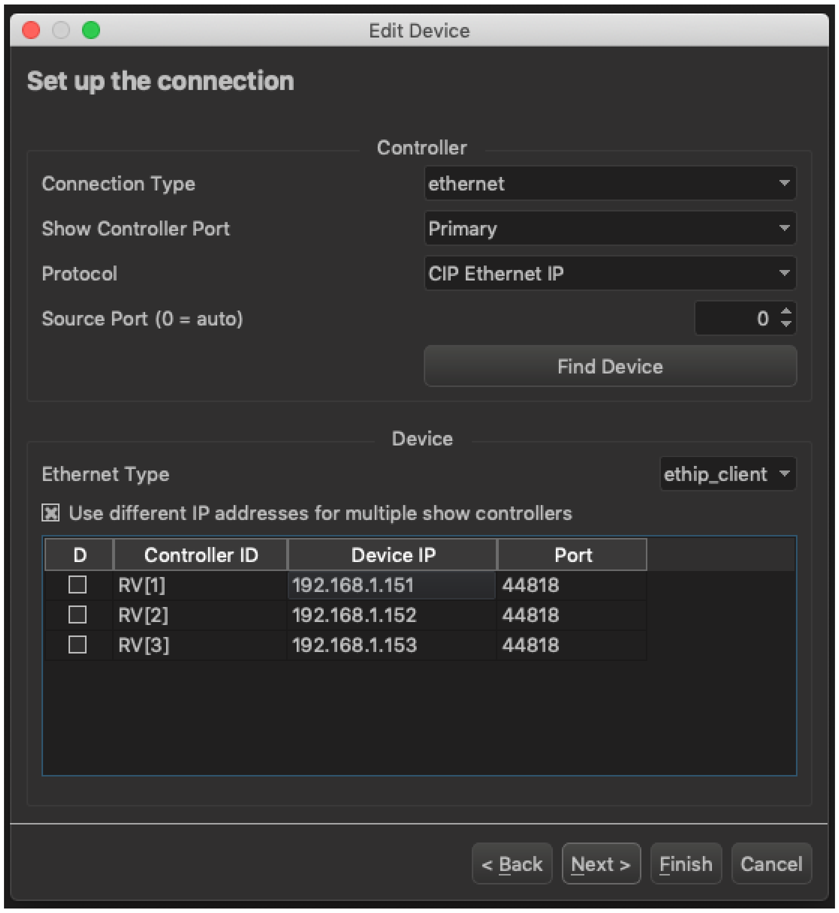
Comment

Next, we must configure the V16X so that the RidePlayers can communicate with it:



#	D	Name	Device Type	Protocol	Connection	Details
1	☐	RV	Alcorn McBride, Inc. - RidePlayer	local		
2	☐	SCS	Alcorn McBride, Inc. - V16X	ASCII remote	ethernet Primary	UDP: 192.168.1.201, 2640
3	☐					

Finally, we need to configure the PLC connections. What's unique about this is that each ride vehicle is likely to have a PLC with its own unique IP address. It's our job to identify which PLC is associated with each specific RidePlayer unit:



Set up the connection

Controller

Connection Type: ethernet

Show Controller Port: Primary

Protocol: CIP Ethernet IP

Source Port (0 = auto): 0

Device

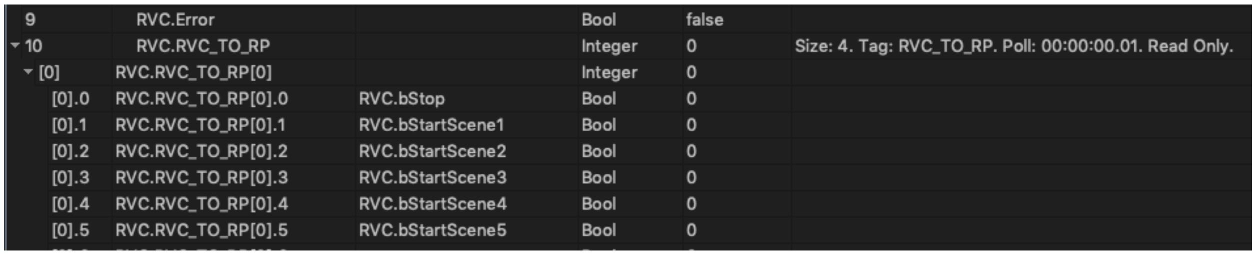
Ethernet Type: ethip_client

☒ Use different IP addresses for multiple show controllers

D	Controller ID	Device IP	Port
☐	RV[1]	192.168.1.151	44818
☐	RV[2]	192.168.1.152	44818
☐	RV[3]	192.168.1.153	44818

PLC Tag Configuration

The script needs to be configured to scan the PLC tags so that we can trigger based on ride vehicle position. It's common for these types of triggers to be associated with individual bits within INT or DINT tags. To make these trigger bits easier to keep track of, we can utilize the Alias feature to assign them with descriptive names:

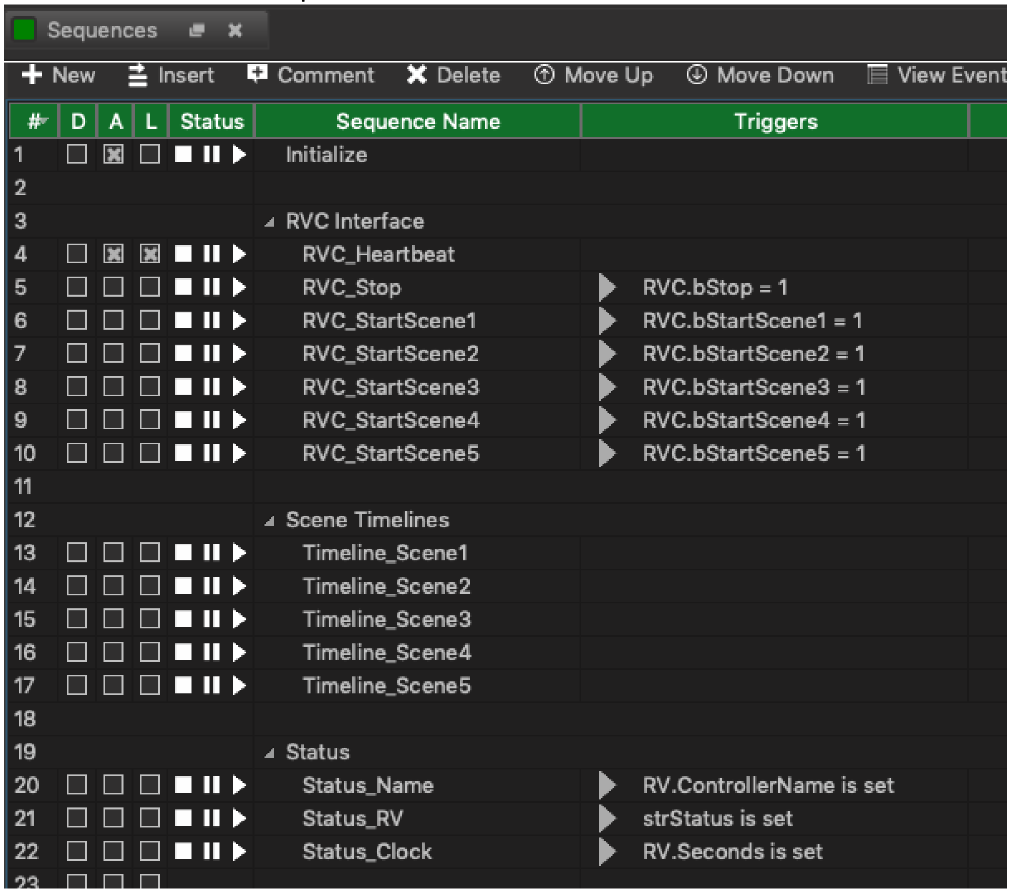


9	RVC.Error		Bool	false	
▾ 10	RVC.RVC_TO_RP		Integer	0	Size: 4. Tag: RVC_TO_RP. Poll: 00:00:00.01. Read Only.
▾ [0]	RVC.RVC_TO_RP[0]		Integer	0	
[0].0	RVC.RVC_TO_RP[0].0	RVC.bStop	Bool	0	
[0].1	RVC.RVC_TO_RP[0].1	RVC.bStartScene1	Bool	0	
[0].2	RVC.RVC_TO_RP[0].2	RVC.bStartScene2	Bool	0	
[0].3	RVC.RVC_TO_RP[0].3	RVC.bStartScene3	Bool	0	
[0].4	RVC.RVC_TO_RP[0].4	RVC.bStartScene4	Bool	0	
[0].5	RVC.RVC_TO_RP[0].5	RVC.bStartScene5	Bool	0	

Sequences – Ride Vehicle

Sequences are the heart of the show control script and contain all of the functional events that are programmed. Let's take a moment to walk through the key sequences contained in the Ride Vehicle script.

Here's what the RV sequences look like:



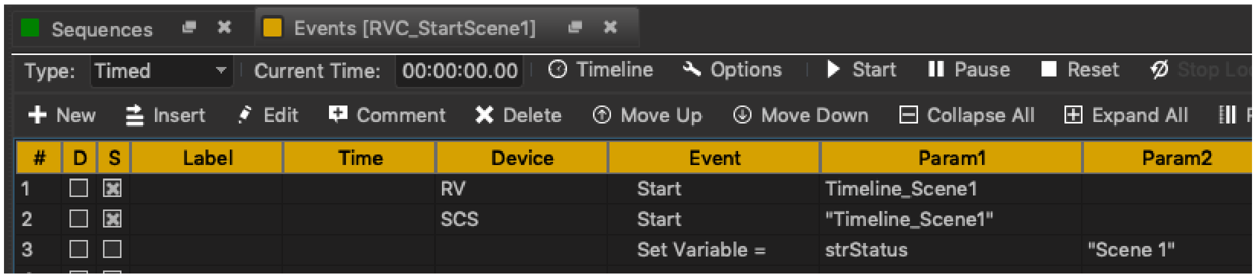
#	D	A	L	Status	Sequence Name	Triggers
1	☐	☒	☐	■ ❚❚ ▶	Initialize	
2						
3					◢ RVC Interface	
4	☐	☒	☒	■ ❚❚ ▶	RVC_Heartbeat	
5	☐	☐	☐	■ ❚❚ ▶	RVC_Stop	▶ RVC.bStop = 1
6	☐	☐	☐	■ ❚❚ ▶	RVC_StartScene1	▶ RVC.bStartScene1 = 1
7	☐	☐	☐	■ ❚❚ ▶	RVC_StartScene2	▶ RVC.bStartScene2 = 1
8	☐	☐	☐	■ ❚❚ ▶	RVC_StartScene3	▶ RVC.bStartScene3 = 1
9	☐	☐	☐	■ ❚❚ ▶	RVC_StartScene4	▶ RVC.bStartScene4 = 1
10	☐	☐	☐	■ ❚❚ ▶	RVC_StartScene5	▶ RVC.bStartScene5 = 1
11						
12					◢ Scene Timelines	
13	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene1	
14	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene2	
15	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene3	
16	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene4	
17	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene5	
18						
19					◢ Status	
20	☐	☐	☐	■ ❚❚ ▶	Status_Name	▶ RV.ControllerName is set
21	☐	☐	☐	■ ❚❚ ▶	Status_RV	▶ strStatus is set
22	☐	☐	☐	■ ❚❚ ▶	Status_Clock	▶ RV.Seconds is set
23	☐	☐	☐			

First, there is a sequence called **Initialize** which does nothing more that fill the front-panel display of the RidePlayer with some text. This is common practice since the RidePlayer display is a handy way to display status information to operators and maintenance staff.

Next, we have the **RVC_Heartbeat** sequence which simply toggles a bit within a PLC tag. On the PLC side, this bit is typically associated with a 'Watchdog' timer to verify the show system (RidePlayer) is operating normally.

Then we have our **RVC_Stop** and **RVC_StartScene** sequences. These sequences are quite simple. They are triggered by the PLC tag bits we configured earlier. When a scene is entered the tag bit changes to a value of 1, and the appropriate sequence is triggered as a result.

The triggered sequence then schedules two timelines to start synchronously; one in the offboard V16X and one within RidePlayer.



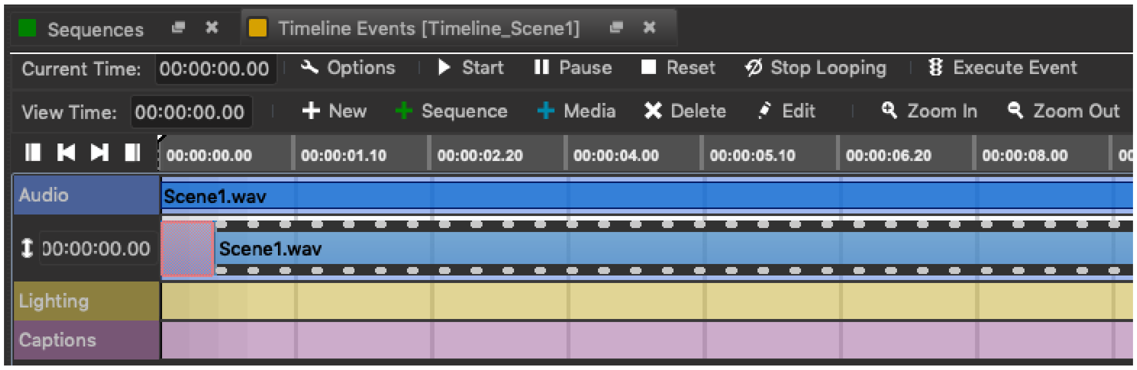
Type: Timed

Current Time: 00:00:00.00

#	D	S	Label	Time	Device	Event	Param1	Param2
1	☐	☒			RV	Start	Timeline_Scene1	
2	☐	☒			SCS	Start	"Timeline_Scene1"	
3	☐	☐				Set Variable =	strStatus	"Scene 1"

Notice that the checkboxes in the ‘S’ (Sync) column are enabled. This means that these two sequences will be scheduled to trigger simultaneously. When this synchronous method is used, there is a consistent ‘Event Sync Delay’ that occurs before these events run. For this example, that delay is configured to 16 frames (approximately half a second).

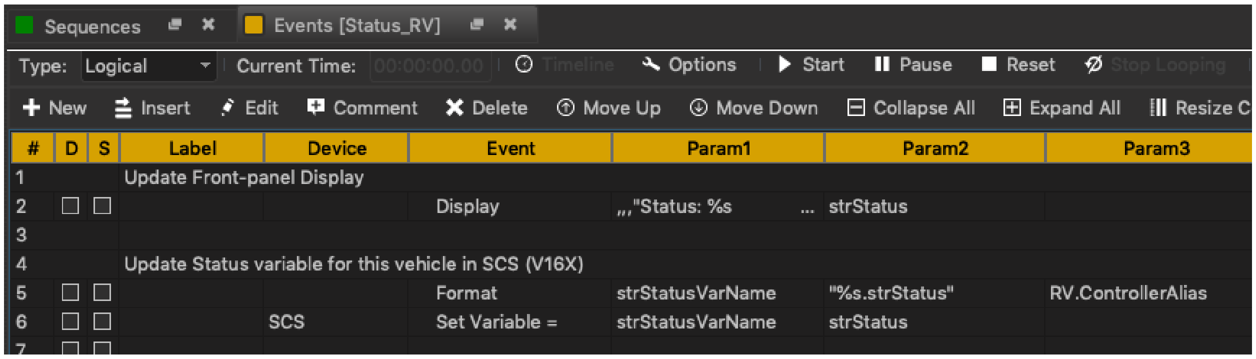
The **Timeline_Scene** sequences are timelines that trigger audio playback onboard. Other timed events like onboard video, effects, captions, etc. would be inserted into this timeline as needed.



Current Time: 00:00:00.00

View Time: 00:00:00.00

The last series of **Status** sequences that update the front-panel display with status information, and also update important status variables within the V16X. These sequences are triggered whenever a variable value changes. For example, when the ride vehicle enters scene 1 the display will be updated with the text “Status: Scene 1” and a variable is also set within the V16X to keep it up-to-date with the RidePlayer’s current status.



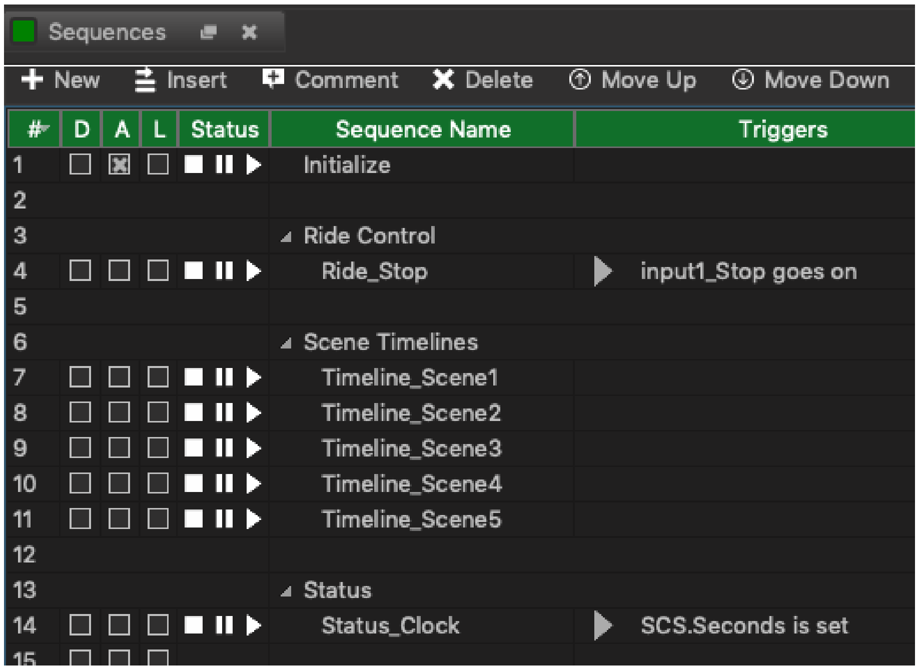
Type: Logical

Current Time: 00:00:00.00

#	D	S	Label	Device	Event	Param1	Param2	Param3
1			Update Front-panel Display					
2	☐	☐			Display	,,,"Status: %s ...	strStatus	
3								
4			Update Status variable for this vehicle in SCS (V16X)					
5	☐	☐			Format	strStatusVarName	"%s.strStatus"	RV.ControllerAlias
6	☐	☐		SCS	Set Variable =	strStatusVarName	strStatus	

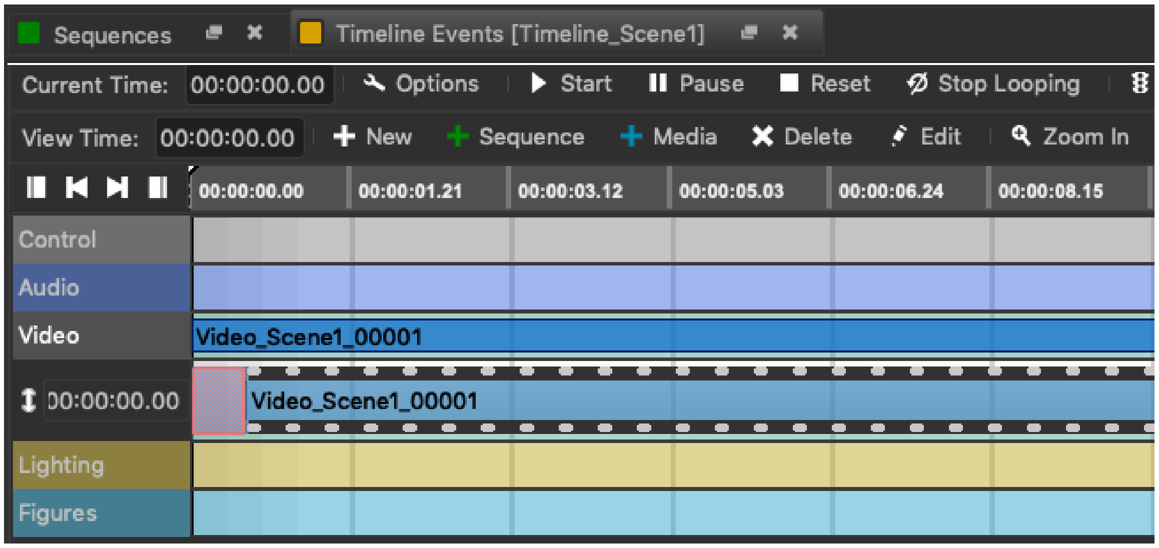
Sequences - Offboard

The sequences of the offboard V16X are even more simple in this example. The only major functional sequences are the **Timeline** sequences that are dedicated to each scene. These sequences are triggered directly from the ride vehicle RidePlayer when scene triggers are encountered.



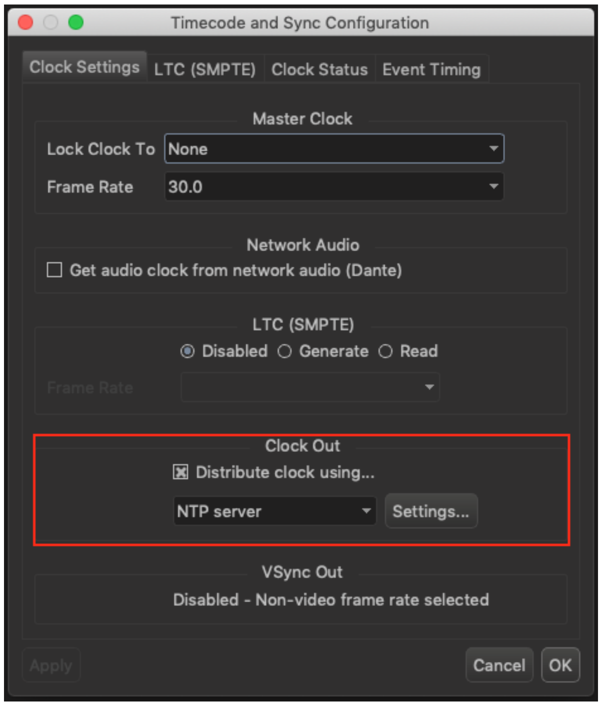
#	D	A	L	Status	Sequence Name	Triggers
1	☐	☒	☐	■ ❚❚ ▶	Initialize	
2						
3					◢ Ride Control	
4	☐	☐	☐	■ ❚❚ ▶	Ride_Stop	▶ input1_Stop goes on
5						
6					◢ Scene Timelines	
7	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene1	
8	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene2	
9	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene3	
10	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene4	
11	☐	☐	☐	■ ❚❚ ▶	Timeline_Scene5	
12						
13					◢ Status	
14	☐	☐	☐	■ ❚❚ ▶	Status_Clock	▶ SCS.Seconds is set
15	☐	☐	☐			

These timeline sequences trigger video playback and animation control on the offboard system.



Clock Configuration – Offboard

Last, but not least, we need to configure both the V16X and RidePlayers to synchronize their clocks with one another. For this application, the V16X functions as the clock master and distributes this clock as an NTP Server.


Timecode and Sync Configuration

Clock Settings | LTC (SMPTE) | Clock Status | Event Timing

Master Clock
Lock Clock To: None
Frame Rate: 30.0

Network Audio
☐ Get audio clock from network audio (Dante)

LTC (SMPTE)
[✓ Disabled] Generate Read
Frame Rate

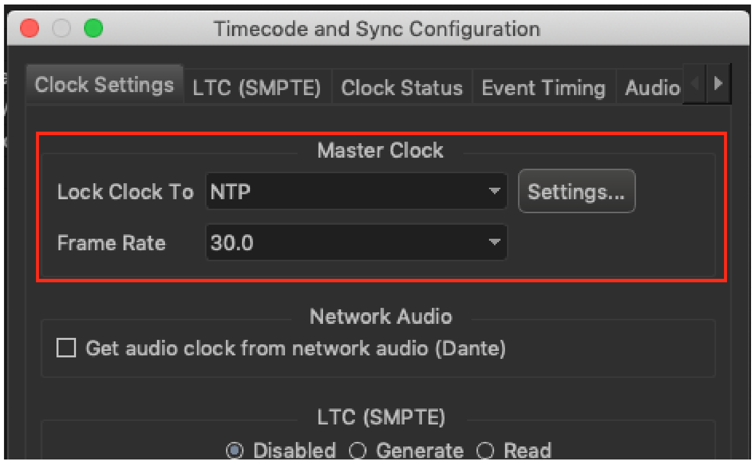
Clock Out
☒ Distribute clock using...
NTP server | Settings...

VSync Out
Disabled - Non-video frame rate selected

Apply | Cancel | OK

Clock Configuration – Ride Vehicle

The RidePlayers operate as NTP clients so they can lock to the V16X NTP Server.


Timecode and Sync Configuration

Clock Settings | LTC (SMPTE) | Clock Status | Event Timing | Audio

Master Clock
Lock Clock To: NTP | Settings...
Frame Rate: 30.0

Network Audio
☐ Get audio clock from network audio (Dante)

LTC (SMPTE)
[✓ Disabled] Generate Read

With this configuration and proper networking infrastructure, these products can synchronize clocks with sub-millisecond precision to ensure the scene timelines are triggered with incredible accuracy.

Conclusion

This application note can serve as a starting point in implementing your own dark ride application. Keep in mind that it's easy to scale the system to include as many ride vehicles and scenes as you need. You can also do WAY more with the offboard system than just trigger video playback and animation control. It's common to control and monitor things like queue line displays, lighting, projection, DSPs, etc.

Now it's time for you to implement your own project with the V16X, RidePlayer and the AV Binloop Uncompressed. Please don't forget that we are here to help you so feel free to contact us with questions.