TPP Programmer's Guide

Vertige™

Ref. VRC300

For version v05.01.01

ANALOG WAY®
Pioneer in Analog, Leader in Digital

Table of contents

1 INTRODUCTION 5
1.1 References 5
1.2 Notices 5
2 CONTROLLING VERTIGE™ 6
2.1 Introduction 6
2.2 Physical interfaces 6
2.3 Protocol 6
2.4 Command principle and structure 7
2.5 Commands sequencing 10
2.6 Command indexes and Command values 11
2.7 Multiple controllers 11
3 VERTIGE™ AUTOMATION BEST PRACTICES 12
4 COMMON VERTIGE™ USE CASES 13
4.1 Establishing a connection with a Vertige™ 14
4.2 Setting up a “workspace” 17
4.3 Loading a Preset onto Preview and TAKING to Program 18
4.4 Changing a layer content 20
4.5 Running a macro 21
4.6 Working with sequences 23
5 NOTES 25
5.1 Using this document 25

Pictures index

Picture 1: Write command example 7
Picture 2: Read command example 8
Picture 3: Valid answer structure 8
Picture 4: Error answer example 9
Picture 5: Write sequence 10
Picture 6: Read sequence 10
Picture 7: Example of connection establishment 16
Picture 8: Example of workspace loading 18
Picture 9: Example of Load Preset and Take 19
Picture 10: Example of changing a layer content 20
Picture 11: Example of macro running 21
Picture 12: Example of sequence actions 24
Picture 13: PDF reader, Previous and Next page buttons 25

1 INTRODUCTION

This document provides information and guidance to control a Vertige™ directly from controllers. A basic knowledge of the device is necessary.

TPP stands for Third Party Protocol.

1.1 References

(on ANALOG WAY web site)

- VRC300_TPP_variables_for_v05-01-01.xls (Vertige™ v05.01.01 TPP command set)
- Vertige™ v05.01.01 firmware updater
- Vertige™ User Manual
- Vertige™ Quick Start Guide

1.2 Notices

Pictures and drawings are non-contractual.

Specifications are subject to change without prior notice.

2 CONTROLLING VERTIGE™

2.1 Introduction

The Vertige™ is usually manually controlled by an operator, but a programming interface is also provided for automation applications.

A good practice is to manually setup the Vertige™ and then control it with a few basic commands like “PRESET_LOAD” and “TRANSITION_TAKE”.

2.2 Physical interfaces

Vertige™ can be controlled through one of its rear Ethernet **RJ45** plugs:

- labeled “**ETHERNET #1**” and “**ETHERNET #2**”
- 10/100/1Gbps compatible
- **auto-MDIX** (which avoid need of crossover cable to connect it directly to a computer)

2.3 Protocol

Supported protocol is **TCP/IP;** Vertige™ IP address and port can be set up in its settings menu.

Default values are:

- Protocol: **TCP**
- DHCP client: **no**
- IP address: **none**
- IP mask: **255.255.255.0**
- Gateway: **none**
- TPP port: **10600**

Note: The Vertige™ TPP server can handle at most 5 clients simultaneously.

2.4 Command principle and structure

2.4.1 Vertige™ control principle

Vertige™ functionalities are controlled through **commands**. Those commands allow **reading** or **writing** in device **registers**.

Vertige™ TPP interface could be considered as a state machine, controlled by sending commands to read or write its registers. Writing into registers modifies machine state. Current state of the machine is always available by reading its registers.

Registers structure and size can range from a simple bit, up to multidimensional array of 32 bits words, and also string. Note that reading or writing a value into a multidimensional register requires providing indexes in addition to the **register value**. Vertige™'s TPP commands use from 0 to 3 indexes values.

Each register have a unique name, only made of **letters**, usually five, upper case or lower case. (one exception having only one letter)

A command is made only of displayable ASCII characters (ranging from 0x21 up to 0x7E) and is ended with a line feed character (LF) (ASCII 0x0A) that will be represented hereafter with the $^{L}_{F}$ symbol.

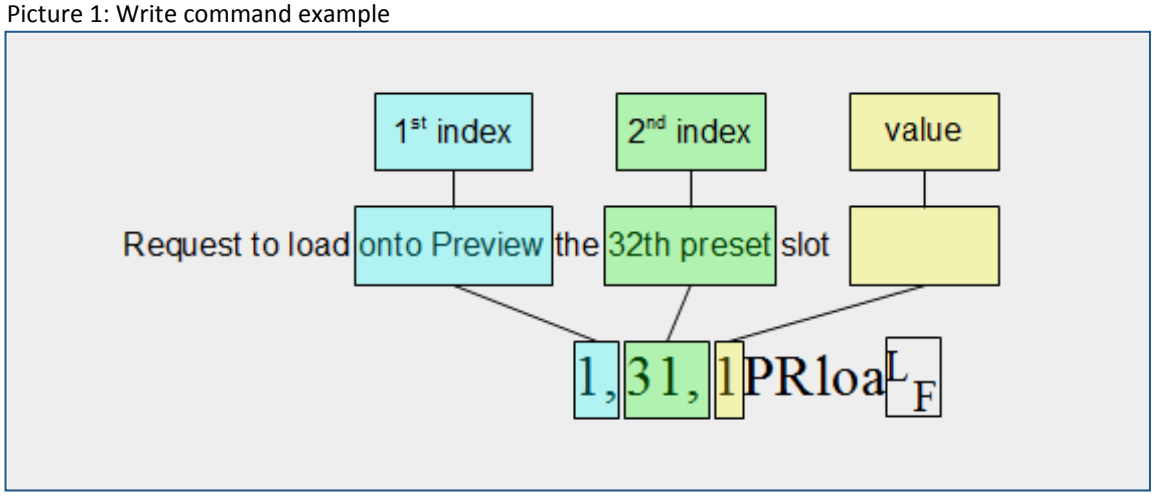
Commands are of 2 types: read commands or write commands, using the same syntax.

- a write command is made of indexes values, followed by the register value, the register name and ended by the $^{L}_{F}$ character.
- a read command uses exactly the same syntax, except the register value that is omitted.

2.4.2 Write command structure

A write command is made of numeric values separated by a comma, followed by a group of up to 5 letters defining the command and is ended with LF. (ASCII 0x0A)

Picture 1: Write command example



The last numerical field is the value to be written in the register.

The first numerical fields are "indexes values", specifying on which dimension the command relates. The number of indexes can range from 0 to 3 depending on the command.

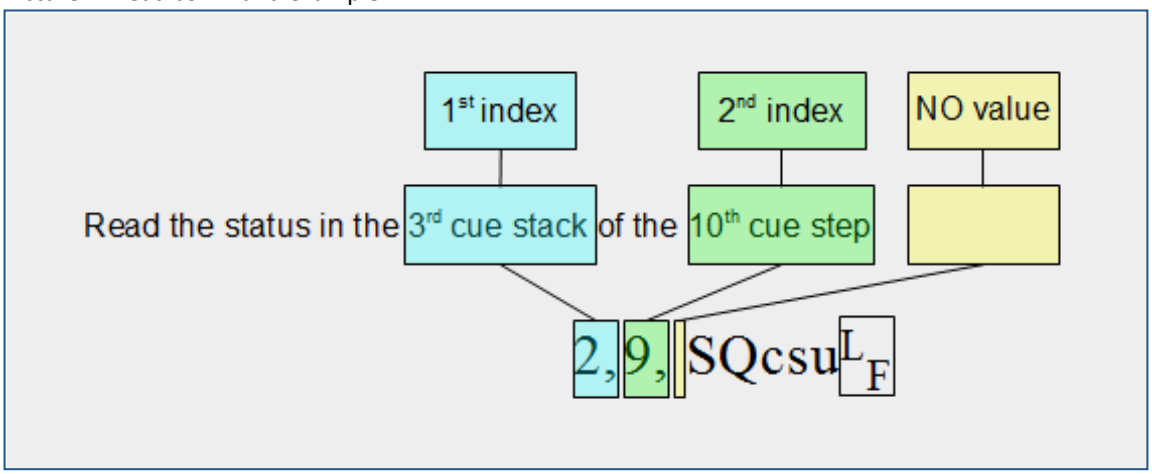
(details in chapter §Command indexes and Command values)

Each command shall be ended with the $^{L}_{F}$ character. (ASCII 0x0A)

2.4.3 Read command structure

A read command follows the same structure than the write command, simply with the value field omitted. Please note that an index value is always followed by a comma.

Picture 2: Read command example

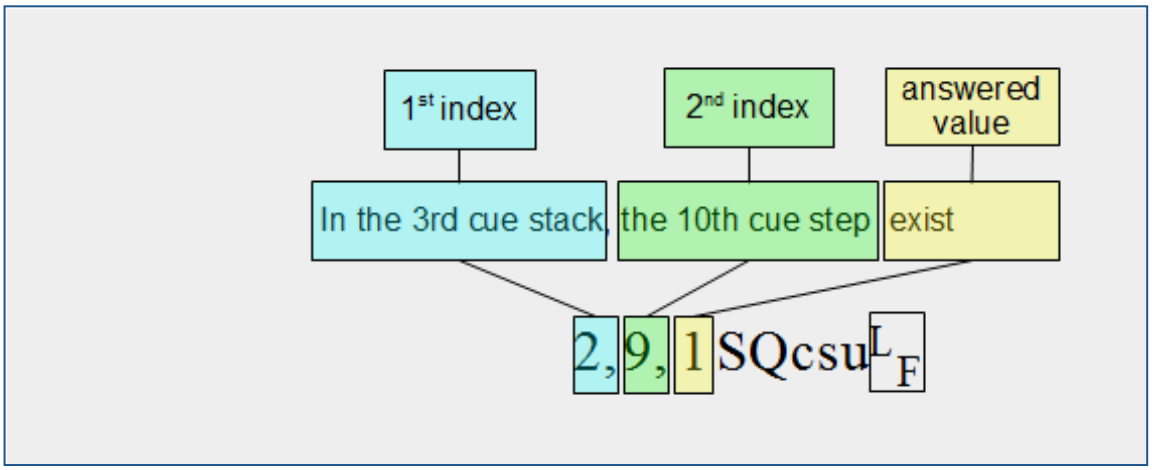


2.4.4 Valid answer structure

When a read or write command is **valid**, the device answers, giving the current register value. The answer structure is symmetrical to the write or read command.

An answer is made of a group of letters (most often the same as the command) followed by numeric values separated by comma, and ended with CR LF characters. (ASCII 0x0D and 0x0A)

Picture 3: Valid answer structure

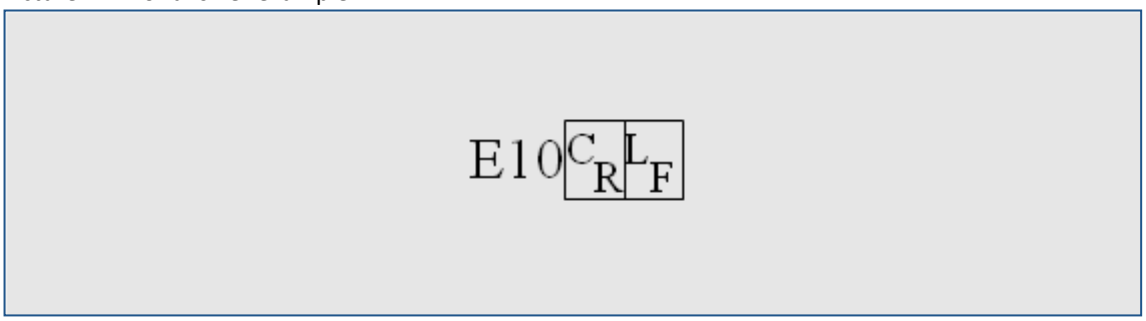


Answer starts generally by the same group of letters than in the initial read or write command, followed by indexes values, then register value and ending with CR LF characters.

2.4.5 Error answer

When an **invalid** read or write command string is received, the device immediately answers with one of the following error string.

Picture 4: Error answer example



Error message structure:

An error message is made of the capital letter E followed by a 2 digits value depending on the error and is ended with CR LF characters. (ASCII 0x0D and 0x0A)

Here are returned error code and conditions covered:

- E10: means “register name error”. It is usually due to a command field (i.e. five letters) that does not match any legal command string.
- E11: means “index value out of range”. It is usually due to a wrong index value.
- E12: means “index number error”. It is usually due to an incorrect number of indexes, too or not enough.
- E13: means “value out of range”. It is usually due to a wrong value in a writing command.

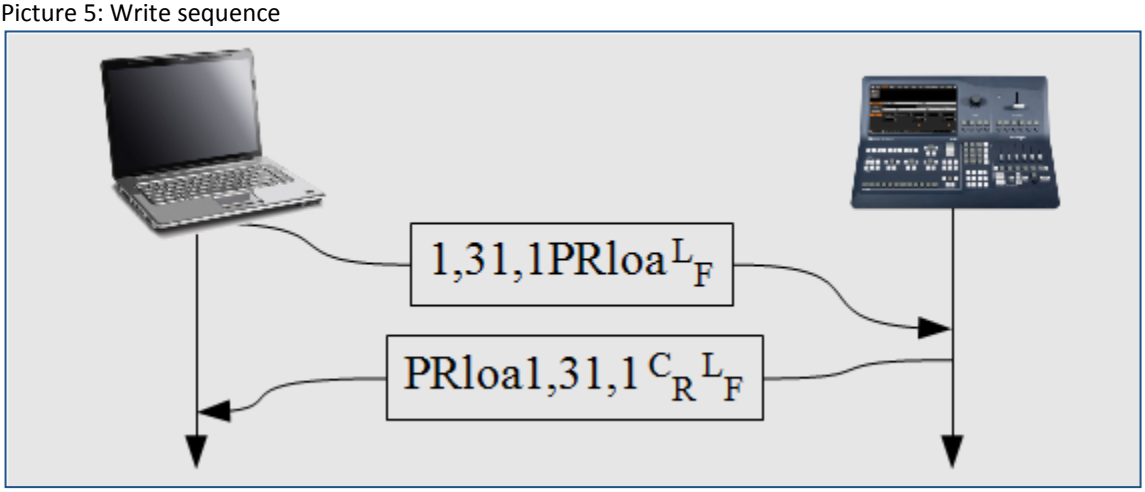
2.5 Commands sequencing

A complete command sequence is made of two parts: first, a read or write command issued by the controller, second, the answer of the device. The answer can be used as an acknowledgment. Because the processing of received commands is asynchronous, the answer time is nor constant nor predictable, but on the other side, this allows to send multiple commands in advance.

A good practice is to check commands acknowledgment before sending new block of commands.

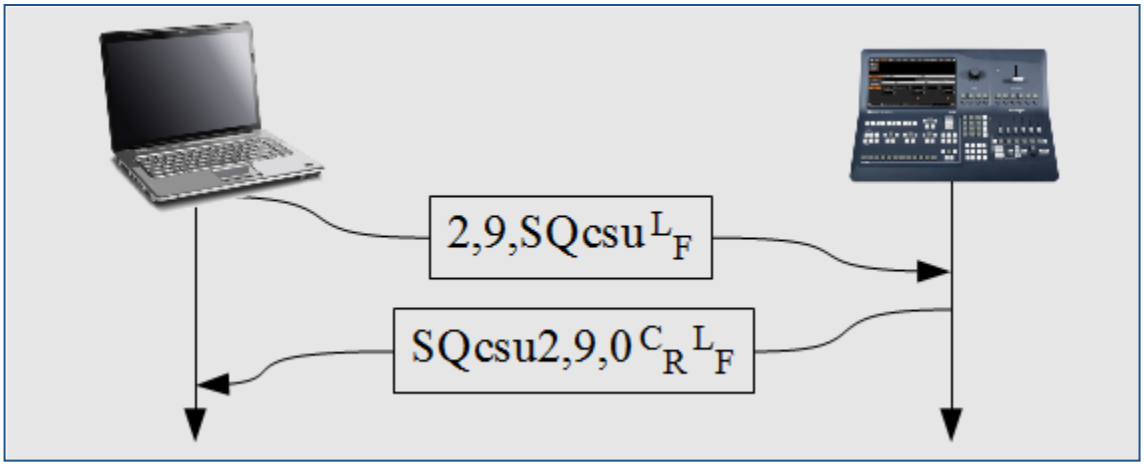
2.5.1 Write sequence

Picture 5: Write sequence



2.5.2 Read sequence

Picture 6: Read sequence



2.6 Command indexes and Command values

As explained in chapter §Vertige_control_principle, Vertige™ commands allow reading or writing values in multidimensional registers. For these, indexes values must be supplied. For simple registers (without dimension), reading or writing commands don't need indexes values.

Indexes values: Depending on the command, you can have to specify from 0 to 3 indexes values. They indicate on which dimension the command relates. For example, the "*TRANSITION_SCENE_TAKE*" command which requests a scene to Take Preview to Program, requires an index value to indicate the desired scene.

No wildcard exists; all required indexes values shall be supplied. Some indexes values have names starting with "*DIM_*", meaning dimension. For example, the "MACRO_NAME" command giving the name of a macro, always requires a "DIM_MACRO_SLOT" index value indicating the slot number of the macro.

Indexes values are detailed in the "VRC300_TPP_variables_for_v05-00-11.xls" document.

Command value: This is the register value. In a write command, it indicates the new value that you want to be applied. In a read answer, it indicates the current state of the command (current register value). A write command is **only** distinguished from a read command due to the presence of the numerical or string value just before the command letters.

A command value can be numerical or string:

- Numerical value should be integer (no space, no decimal part nor engineer notation) and made only of digits
- A string is made of displayable 7bits ASCII characters, surrounded by quotation mark, with a length up to 255 characters. (e.g. "My first show")

A value written in a register remains until modified by a new write command or by the device itself. This allows options to be written only once.

All registers have a ***default value***, noted in the detailed tables.

You must be careful on value range, which depends on multiple factors, like device type, device configuration or current situation. Value range have names starting with "ENUM_", else, if no enumeration name exist, value must be comprise between given "min value" and "max value".

Commands values are detailed in the "VRC300_TPP_variables_for_v05-00-11.xls" document.

2.7 Multiple controllers

Multiple controllers are allowed, limited to 5, with TCP protocol.

No priority exists, in case of simultaneous writing of the same command, the device applies both, one after the other. In all cases, controllers must take into account the last answer received.

User must particularly be careful with compound commands. For example the "WORKSPACE_TRYLOAD" command requires sending at least 3 other commands (WORKSPACE_PICK_USER, WORKSPACE_PICK_SHOW, WORKSPACE_PICK_ITERATION) before sending the request. Multiple controllers using those commands simultaneously would cause unpredictable results.

3 VERTIGE™ AUTOMATION BEST PRACTICES

Even though Vertige™ has been designed to be used manually in many ways, when used in automation, it is advisable to use the following working methods in order to have a controller's software as simple as possible and independent of future show evolutions.

- Workspace:
  - Create a password protected user account. You will use it to store the baseline of all the different setup: create shared shows, with as much iteration as you need.
  - Create another user account, without password, which will be used for TPP automation.
  - Create an empty show per setup, following a naming pattern (e.g. Setup1, Setup2…); then use the Transfer function of the Vertige™ to copy an iteration of a shared show from the protected user account into this one.
- Presets:
  - Set the transition filter when saving a preset. It will be restored when loading a preset, allowing the controller to simply use the PRESET_LOAD and TRANSITION_TAKE pair of commands.
- Macros:
  - Use macros to store series of commands instead of coding them in controller software.

4 COMMON VERTIGE™ USE CASES

The following common actions can be remotely controlled:

- Establishing a connection with a Vertige™, comprising:
  - Socket opening
  - Device type checking
  - Command set version checking
  - Vertige™ registers readback
- Setting up a “workspace”, comprising:
  - User name selection
  - Show name selection
  - Iteration number selection
  - Workspace loading
- Loading a Preset onto Preview and TAKING to Program
- Changing a layer content
- Running a macro
- Working with sequences, comprising
  - Initializing the sequencer
  - Playing a sequence
  - Stopping a sequence
  - Resuming the sequencer

4.1 Establishing a connection with a Vertige™

4.1.1 Usage

This example gives you the proper way to establish the connection with a Vertige™ device. It is made of four recommended steps: socket opening, device type checking, command set version checking and Vertige™ registers read back.

4.1.2 Summary of the commands sequence

- socket opening — initial step, TCP/IP connection.
- device type checking — verifying that the expected device (Vertige™) is connected.
- command set version checking — verifying that controller driver and Vertige™ TPP version matches.
- Vertige™ registers read back — retrieving the current Vertige™ state.

4.1.3 Detailed commands sequence

- socket opening: As indicated in §CONTROLLING VERTIGE chapter, TCP/IP must be used to control Vertige™. The device acts as a server, and accepts connections as soon as it is ready. Once the connection established, the device sends the **SYSTEM_CLIENT_CONNECTED** status, giving the number of connected controllers.

  Answer: SYclc<value>$^{C}_{R}{}^{L}_{F}$ The numerical <value> gives the connected controllers number.

- device type checking: This read only command gives the device type.

  Syntax: ?$^{L}_{F}$

  Answer: DEV<value>$^{C}_{R}{}^{L}_{F}$ The numerical <value> gives the connected device type. Other values match other Analog Way devices.

<value>	device
513	Vertige™
514	Rackmount Control Unit

- command set version checking: This read only command gives the "VERSION_COMMAND" number. When a new firmware version is released, if TPP is modified, then the "VERSION_COMMAND" is increased.

  It is recommended to check that this value matches the one expected by the controller.

  Syntax: VEcmdLF

  Answer: VEcmd4$^{C}_{R}{}^{L}_{F}$ The current value is **4** for this firmware version.

- Vertige™ registers read back: This step is recommended to initialize the controller. Various methods exist depending on controller software architecture.

  To ease this initialization step, the device features the SYSTEM_DIESE command to enumerate (read back) all its registers current values. This produces a huge amount of data that can saturate the controller. A command parameter allows reducing this volume by sending only register values different from their default value. If the volume is still too high, the controller should enumerate himself all the required registers, at its own pace.

  - Read back using SYSTEM_DIESE command:

    At first, the controller should wait for a possible current SYSTEM_DIESE command to finish.

sending: SYdie$^{L}_{F}$ Controller ask the current state of the SYSTEM_DIESE register.

Answer: SYdie<value>$^{C\ L}_{R\ F}$ The controller must wait that the numerical <value> be equals to **0**, meaning that no enumeration is running.

sending: 1SYdie$^{L}_{F}$ or 2SYdie$^{L}_{F}$ When <value> is equal to 1, the device will enumerate all registers values, for all indexes values in case of multidimensional registers. This can produce a huge amount of data. The end of the enumeration is signaled when the SYSTEM_DIESE register automatically returns to the value 0, meaning that the controller must wait until receiving SYdie0$^{C\ L}_{R\ F}$.

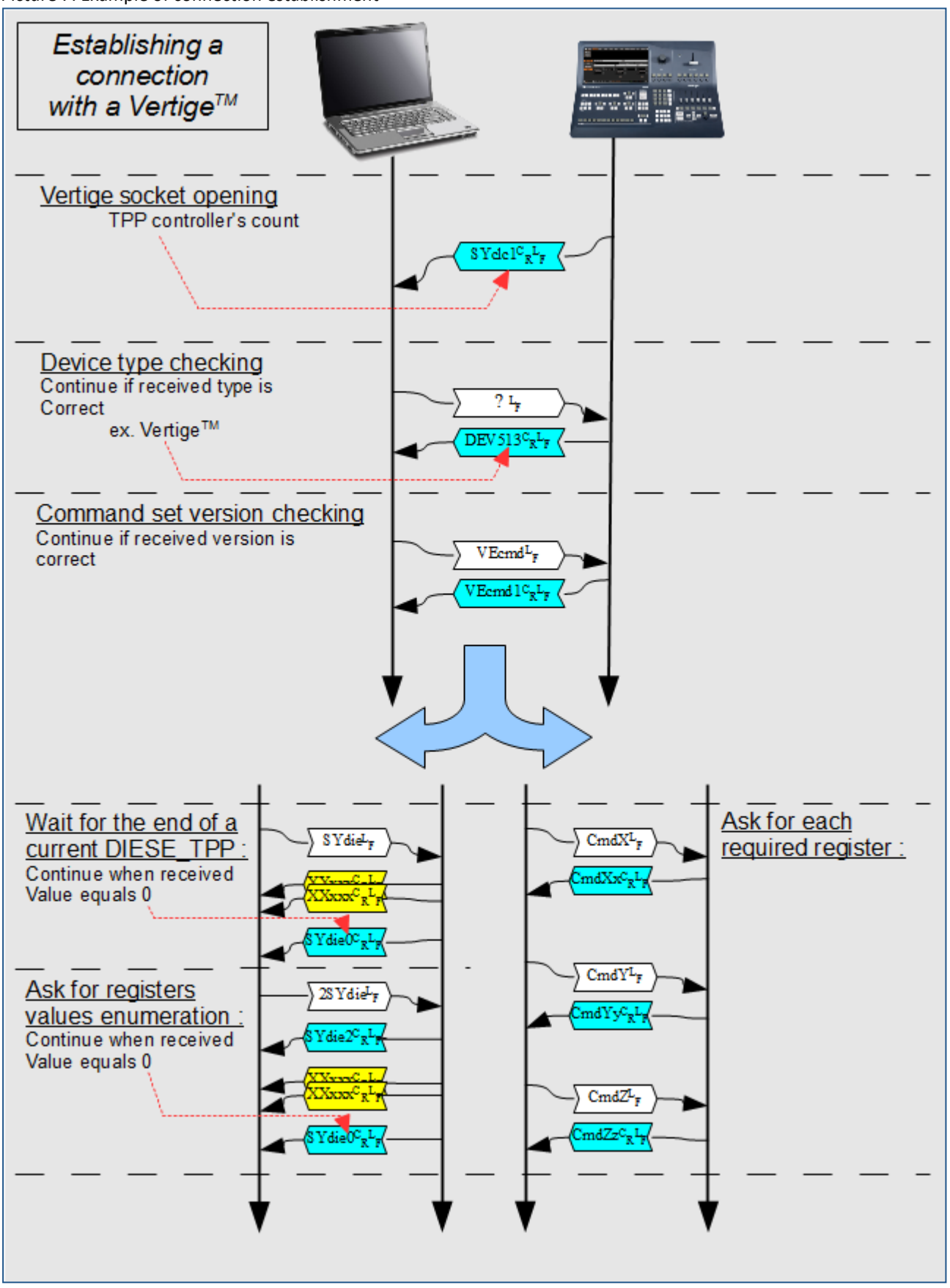
When <value> is equal to 2, the device works the same way, except that it will not enumerate registers having their default value, reducing the amount of received data.

- Registers read back managed by the controller:

  The controller should read all used registers, slowly enough to avoid being saturated, issuing as many read commands as needed.

4.1.4 Example of connection establishment

Picture 7: Example of connection establishment



4.2 Setting up a "workspace"

4.2.1 Usage

Once the connection properly established between the controller and the Vertige™, a "workspace" should be established, allowing using presets, macros or sequences. To load a workspace, three information should be provided: an existing ***show name***, an existing ***iteration number*** of this show and a ***user name*** allowed reading this show (not necessarily the owner if the show is shared). Once the workspace loaded, the Vertige™ tries to establish a connection with all related devices.

The Vertige™ automatically unloads any current active workspace when attempting to load another one.

4.2.2 Summary of the commands sequence

- Set the user name (may be the "Guest" user or another, allowed to read the show)
- Set the show name (name of an existing show)
- Set the show iteration number (existing number in the show)
- Try to load the show
- Wait a successful pick status (meaning that the correct iteration of the show has been loaded)
- Wait a successful workspace status (meaning successful connection of devices)

As explained at §Vertige control principle, values written in registers remain active until changed. This allows sending only once the parameters that don't change while power remains on. This implies also that order is not important but parameters must be settled before executing the WORKSPACE_TRYLOAD command.

4.2.3 Detailed commands sequence

- **Set the user name:** This command gives Vertige™ the name of a user allowed to read the workspace's show. It can be the show's creator or anyone else if the show is "shared".

  Syntax: <name value>WKpus$^{L}_{F}$

  Example: "john doe"WKpus$^{L}_{F}$

- **Set the show name:** This command gives Vertige™ the name of a recorded show.

  Syntax: <name value>WKpsh$^{L}_{F}$

  Example: "def. show"WKpsh$^{L}_{F}$

- **Set the show iteration number**: This command gives Vertige™ an existing iteration number of the show.

  Syntax: <value>WKpit$^{L}_{F}$ <value> can be from 1 up to 4294967295.

- **Try to load the workspace**: Once all previous parameters are set, start the load process.

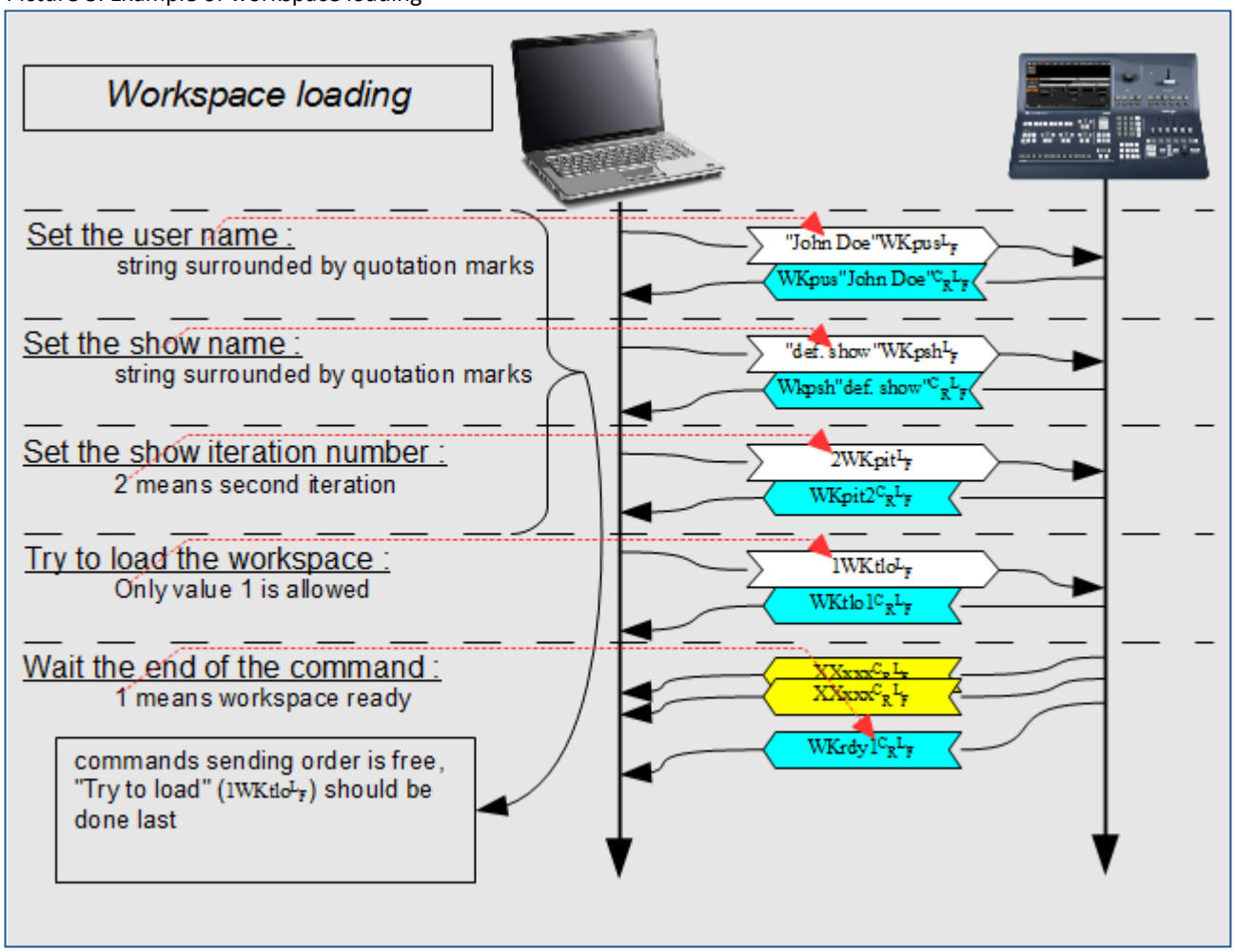
Syntax: 1WKtlo$^{L}_{F}$ Only value 1 is allowed. Device will immediately acknowledge the command, will unload any current workspace, and will load the requested show and iteration. Then it will try to connect all workspace devices and issue a WORKSPACE_READY value 1 when done.

- **Wait the end of the command**: Once loaded (check PICK_STATUS), you can wait for the WORKSPACE_READY status.

  Answer: WKrdy1$^{C\ L}_{R\ F}$ Wait this answer for a few seconds max.

4.2.4 Example of workspace loading

Picture 8: Example of workspace loading



4.3 Loading a Preset onto Preview and TAKING to Program

4.3.1 Usage

As recommended in chapter 3, you should manually record presets that embed scene filters. In the controller, use the “PRESET_LOAD” command to load them on PREVIEW and then issue the “TRANSITION_TAKE” command to transition the active scenes (non-filtered) onto the PROGRAM.

4.3.2 Summary of the commands sequence

- Load a Preset (containing the correct transition scene filter)
- Transition scenes to Program (using the transition scene filter)

4.3.3 Detailed commands sequence

- **Load a Preset**: This command allows to load a Preset on Program or Preview.

  Syntax: <dest. value>,<slot value>,1PRloa$^{L}_{F}$ <1/100s delay>

  The <dest. value> can be 0 for a Program destination or 1 for a Preview destination.

  The <slot value> can be a value ranging from 0 for the first Preset to 199 for the 200th Preset.

  Answer: PRloa<dest. value>,<slot value>,1$^{C\ L}_{R\ F}$

  Example 1: 1,0,1PRloa$^{L}_{F}$ load the 1st Preset onto Preview.

  Example 2: 0,199,1PRloa$^{L}_{F}$ load the 200th Preset onto Program.

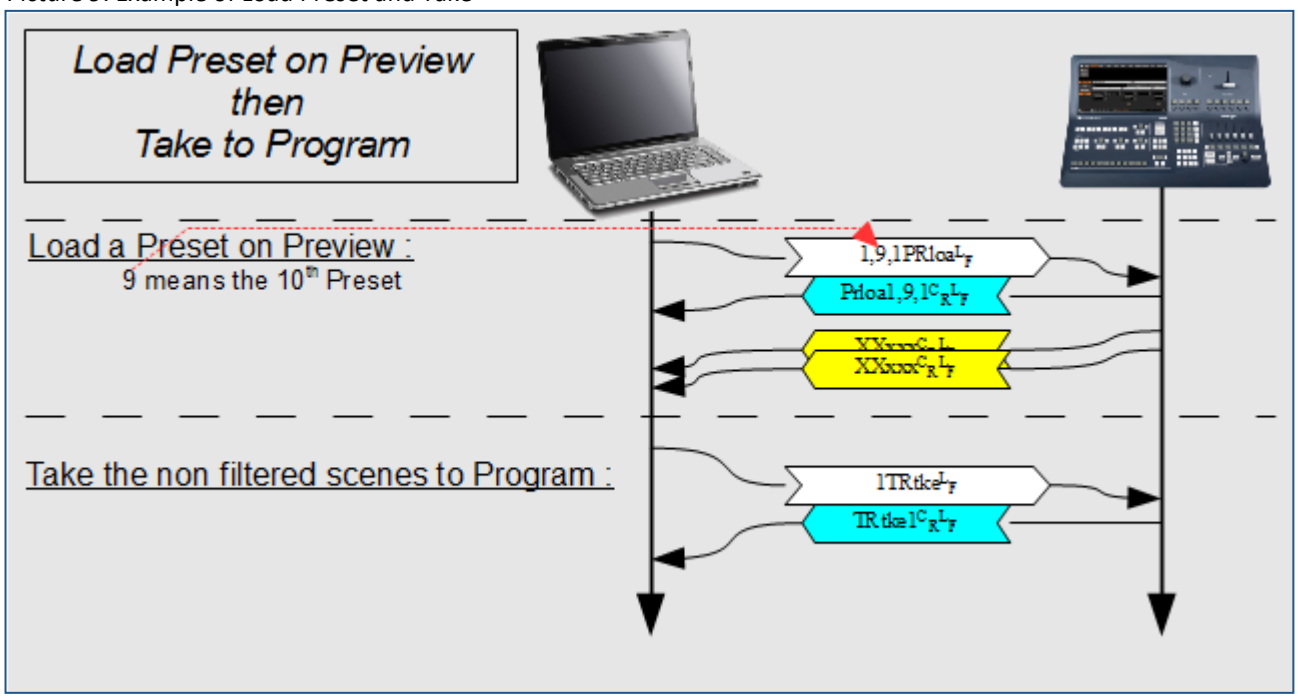
- **Transition active scenes (non-filtered) Preview to Program**: This command requests the Vertige™ to transition the Preview of active scenes (not filtered) onto Program.

Syntax: 1TRtke$^{L}_{F}$ <1/100s delay> Only value 1 is allowed. Vertige™ will immediately acknowledge the command and then will apply the transitions to the active scenes (non-filtered).

Answer: TRtke1$^{C\ L}_{R\ F}$

4.3.4 Example of Load Preset on Preview then Take to Program

Picture 9: Example of Load Preset and Take



4.4 Changing a layer content

4.4.1 Usage

Please refer to the user manual for source configuration and usage on Vertige™.

The PRESET_LAYERFILL_SOURCE register is used to change the source displayed in a layer. It should be noted that other commands should be used for a native background layer or for a “Cut and Fill” layer.

Usually changes are made on Preview.

4.4.2 Detailed commands sequence

Syntax: <destination>,<scene pos.>, <layer>,<source> $PRlfs^{L}_{F}$

The <destination> is an index value: 0 for a Program destination or 1 for a Preview destination.

The <scene> is an index value ranging from 0 for the 1st scene, up to 23 for the last.

The <layer> is an index value ranging from 0 for the first layer, up to 71.

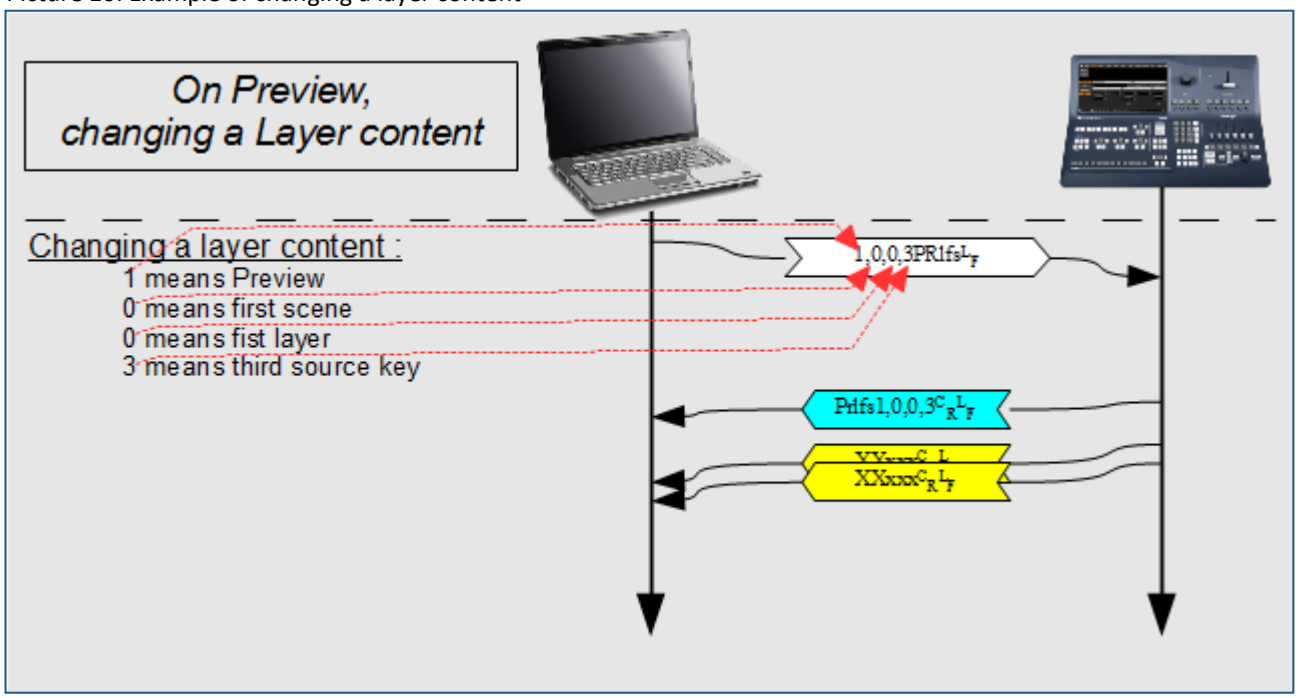
The <source> is the register value giving the source: 0 for none and from 1 up to 240 for source key.

Answer: PRlfs <destination>,<scene pos.>, <layer>,<source> $^{C\ L}_{R\ F}$

Example 1: $1,0,0,3PRlfs^{L}_{F}$ on preview, on the leftmost scene position, in the first layer assigns the 3rd source key.

4.4.3 Example of changing a layer content

Picture 10: Example of changing a layer content



4.5 Running a macro

4.5.1 Usage

Please refer to the user manual for macro creation on the Vertige™.

Using macro created and stored in the Vertige™ is a good method to avoid modifying the controller's software.

Four TPP commands are related to macros:

- MACRO_RUN is used to start a macro. If the macro is already running, the Vertige™ aborts the macro then restarts it from the first step.
- MACRO_STATUS is used to know if a macro is running.
- MACRO_USED is used to know if a macro exists.
- MACRO_NAME is used to retrieve the name of a macro.

4.5.2 Detailed commands sequence

- **Running a macro**:

Syntax: <macro nbr>,1MRrun$^{L}_{F}$

<macro nbr> is the macro number, starting with value 0.

Only value 1 is allowed. Vertige™ will immediately acknowledge the command and then will start or restart the macro, send the running status for the macro number and finally the stopped status at the end.

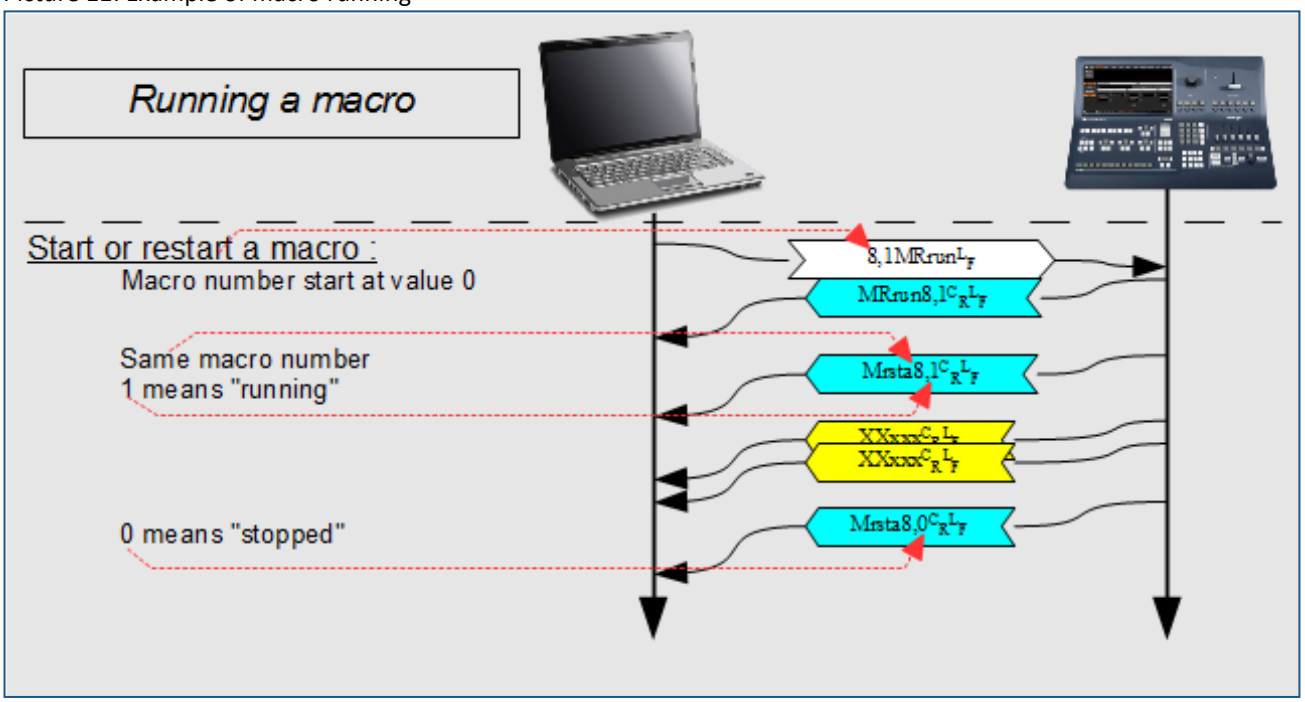
Answer:

MRrun<macro nbr>,1$^{C\ L}_{R\ F}$ MRsta<macro nbr>,1$^{C\ L}_{R\ F}$ ... MRsta<macro nbr>,0$^{C\ L}_{R\ F}$

Multiple macros can be run at the same time.

4.5.3 Example of macro running

Picture 11: Example of macro running



4.5.4 Tips

- **Running an empty macro**:

  When trying to run an empty macro, only the command acknowledgement is immediately sent. As the macro is empty, its status remain “stopped”, the MACRO_STATUS answer is not changed.

- **Detecting an empty macro**:

  The MACRO_USED command can be used to know if a macro exists.

  Syntax: <macro nbr>,MRusd$^{L}_{F}$

  <macro nbr> is the macro number, starting with value 0.

  Answer if a macro is defined: MRusd<macro nbr>,1$^{C\ L}_{R\ F}$

  Answer if a macro do not exist: MRusd<macro nbr>,0$^{C\ L}_{R\ F}$

4.6 Working with sequences

4.6.1 Usage

Please refer to the user manual for sequence creation on the Vertige™.

Typical TPP actions are:

- **Initializing the sequencer**: When the sequencer is running (after a SEQUENCE_PLAY, SEQUENCE_NEXT or SEQUENCE_PREVIOUS command) the SEQUENCE_PICK_PLAYHEAD value (play head starting position) is no more used. So it is important to check the SEQUENCE_STATUS value before sending the SEQUENCE_PICK_PLAYHEAD command.
- Setting the "play head" position: This gives Vertige™ the cue stack and cue step starting point (play head position) to be used for the next playback, triggered by the following commands: SEQUENCE_PLAY, SEQUENCE_NEXT or SEQUENCE_PREVIOUS. Once one of these commands has been sent, the sequencer will be in "running" state and it will not use this starting point until it return in "stopped" state, either at the end of the sequence or following a SEQUENCE_STOP command.
- **Playing a sequence**.
- **Stopping a sequence**.
- **Resuming the sequencer**, Allows to continue the playback when the sequencer is waiting for a manual trigger.

4.6.2 Detailed commands sequence

- **Initializing the sequencer**: Unless special case, it is safe to issue first to the sequencer a SEQUENCE_STOP command in order to be able to setup the starting play head position.

Syntax: 1SQsto$^{L}_{F}$

Only value 1 is allowed. Vertige™ will immediately acknowledge the command and then will stop the sequencer.

- **Setting the "play head" position**:

The starting position is given as a string using the following format: *cue stack number*, then a **dot**, then the *cue step number*.

Syntax: <CueStack CueStep>SQppl$^{L}_{F}$

Example: "0.1"SQppl$^{L}_{F}$ set the starting position to the second cue step of the first cue stack.

- **Playing a sequence**: Usually, the sequence will automatically run to the end with a SEQUENCE_PLAY command.

Please note that an internal delay of 0.15s is applied between each executed cue step.

In order to avoid infinite run of sequence, the "jump to" action of a cue step use a repeat value comprised between 1 and 10. After this count down, the "jump to" no more operates.

Syntax: 1SQpla$^{L}_{F}$

The device will immediately acknowledge the command and then enter the "running" state, other status may also be sent.

- **Stopping a sequence**: After issuing a SEQUENCE_STOP command or when the end of the sequence is reached, the sequencer returns to the stopped state.

  Please note that when restarting through one of the 3 possible commands, the sequencer will not start from the last cue step but will start from the one indicated by the SEQUENCE_PICK_PLAYHEAD value. During the "running" state, the current play head position can be retrieved with the SEQUENCE_PLAYHEAD_STATUS.

  Syntax: 1SQsto$^{L}_{F}$

  The device will immediately acknowledge the command and then enter the "stopped" state, other status may also be sent.

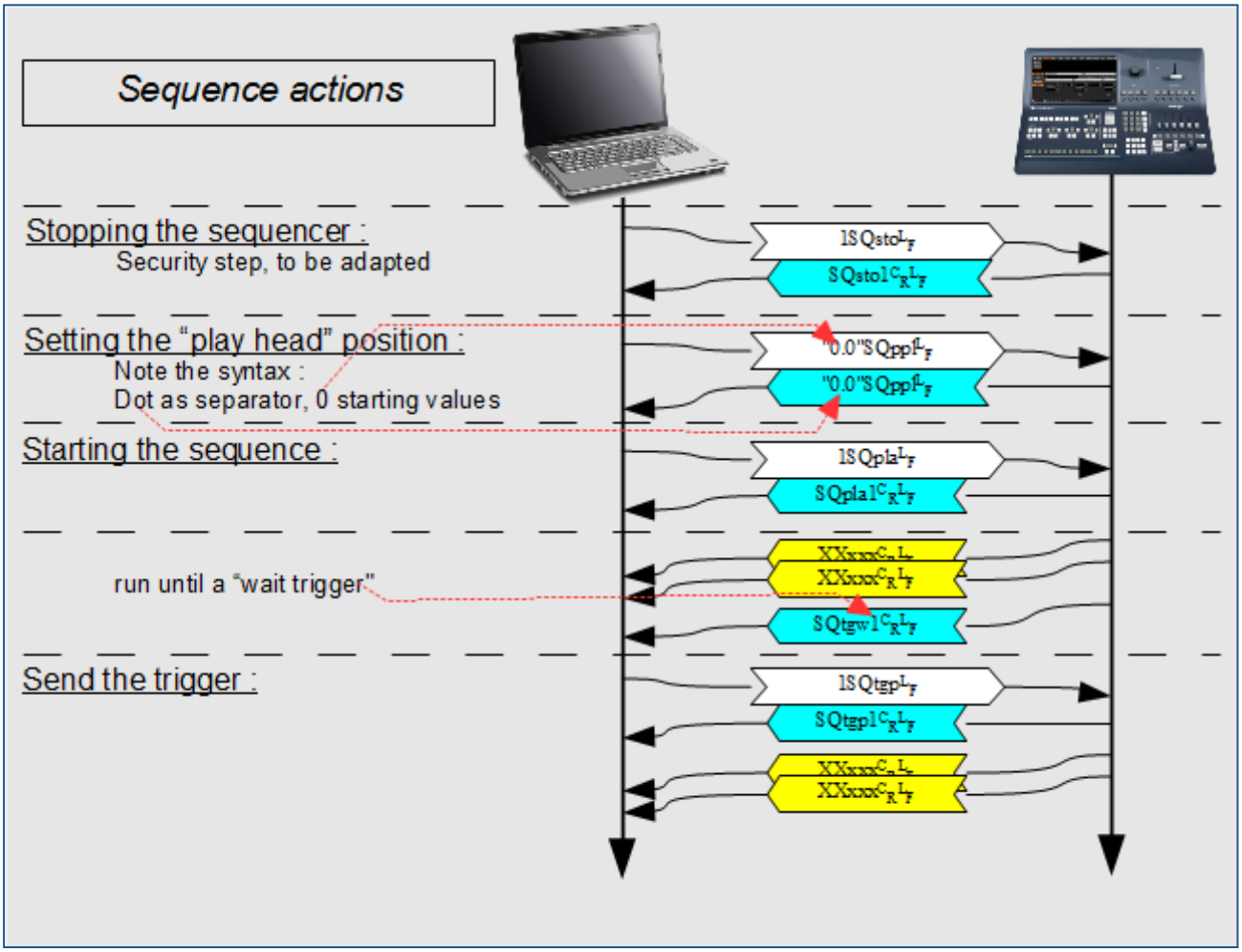
- **Triggering the sequencer**: During sequence running, the sequencer may pause, waiting for a manual trigger.

  Syntax: 1SQtgp$^{L}_{F}$

  The device will immediately acknowledge the command and then continue through the sequence.

4.6.3 Example of sequence actions

Picture 12: Example of sequence actions

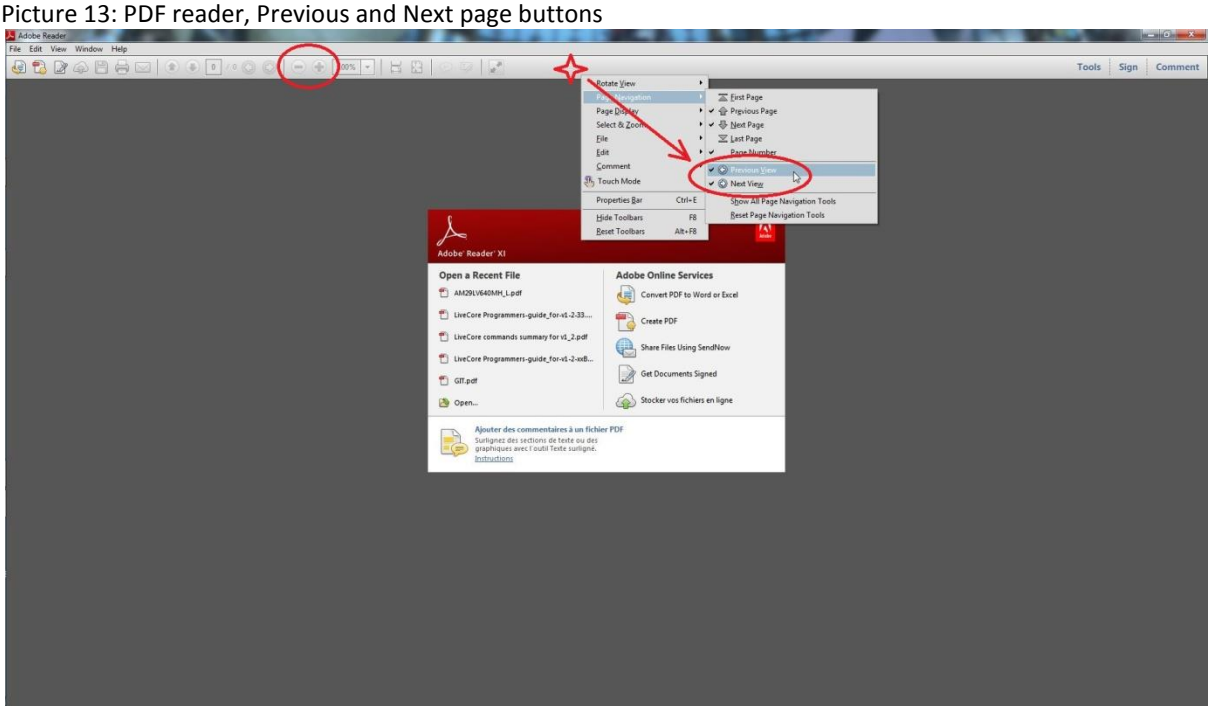


5 NOTES

5.1 Using this document

This document contains many internal links. You can improve your navigation by using the “previous page” function, as in the following example:

Picture 13: PDF reader, Previous and Next page buttons



Connect with us on