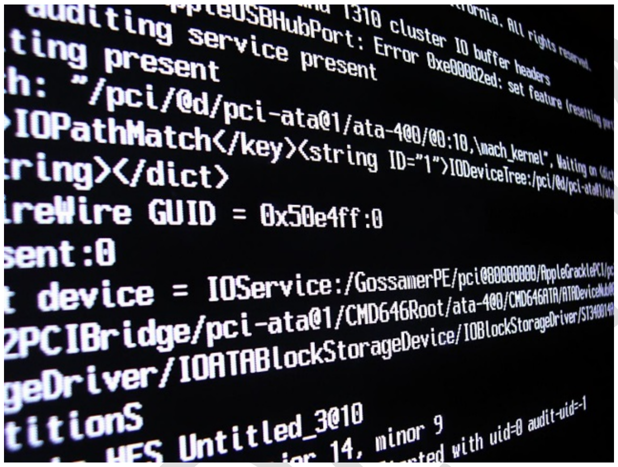


```
auditing service present
ting present
h: "/pci/@d/pci-ata@1/ata-4@0/@0:10,\mach_kernel"
IOPathMatch</key><string ID="1">IODeviceTree:/pci/@d/pci-ata@1/
ring></dict>
reWire GUID = 0x50e4ff:0
sent:0
t device = IOService:/GossamerPE/pci@80000000/AppleGracklePCI/
2PCIBridge/pci-ata@1/CMD646Root/ata-4@0/CMD646ATA/ATADeviceNub@
geDriver/IOATABlockStorageDevice/IOBlockStorageDriver/
titionS
HFS Untitled_3@10
```

# PROGRAMMER'S GUIDE

**Revised 2016-12-19**

# Table of Contents

# Introduction

This is the full list of commands that can be sent to a Just Add Power device to perform various functions, diagnostics, and modes.

Commands are backwards-compatible. This means that 2G commands function on all 2G+ and 2G+AVP pieces, but 2G+AVP commands do **NOT** function on 2G+ or 2G devices.

All 2G/2G+/2G+AVP commands function on 3G+AVP devices. Commands in the 3G+AVP section are exclusive to 3G+AVP and will not work on any 2G models.

All commands are **CASE-SENSITIVE** and should be used exactly as shown in this manual unless otherwise mentioned.

End command with a carriage-return to execute.

Commands in `Typewriter Font` are exact commands or feedback from a device and can be used for parsing.

This document is **CONFIDENTIAL** and contains information that could cause more harm than good in the wrong hands.

Absolute power corrupts absolutely, but it also rocks absolutely.

## Directory

Commands are executed from the default directory – `/usr/local/bin` – unless otherwise stated.

# All Devices

## Basic Functions

### `reset_to_default.sh`

- Returns the Just Add Power device to factory default settings. **DOES NOT** reset EDID or USB settings.

### `capture_edid.sh`

- Copies the EDID from a display attached to a Receiver and forwards that EDID to the connected Transmitter. Transmitter replaces stored EDID with the new EDID handed to it. See the EDID Management guide for more information.

### `hdmi_off.sh`

- Logical HDMI unplug. **TRANSMITTER ONLY**

### `hdmi_on.sh`

- Logical HDMI plug in. **TRANSMITTER ONLY**

### `reboot`

- Reboots the device

## Diagnostics & Monitoring

### `cat /var/ast_device_status`

- Checks the state of video transmission on Transmitter or Receiver. Returns status
    - Receiver:
        - `s_idle`
            - Receiver is **NOT** receiving video from a Transmitter
        - `s_srv_on`
            - Receiver is receiving video from a Transmitter
        - Example

```
/usr/local/bin # cat /var/ast_device_status
s_srv_on
/usr/local/bin #
```

- Transmitter:
    - `s_idle`
        - Receiver connection is not currently allowed (loopback mode)
    - `s_attaching`
        - Transmitter is connecting or video source not available
    - `s_srv_on`
        - all functions are ready or working
    - Example

```
/usr/local/bin # cat /var/ast_device_status
s_srv_on
/usr/local/bin #
```

### `cat /sys/devices/platform/videoip/State`

- Returns the status of the Transmitter or Receiver with regards to the HDMI signal being sent to it. Returns state
    - Receiver:
        - `WAITING_HOST_MODE`
            - video source not available or not showing
        - `OPERATING`
            - video source available
        - Example

```
/usr/local/bin # cat /sys/devices/platform/videoip/State
OPERATING
/usr/local/bin #
```

- Transmitter:
    - `DETECTING_MODE` = video source not available or not showing
    - `OPERATING` = video source available
    - Example

```
/usr/local/bin # cat /sys/devices/platform/videoip/State
OPERATING
/usr/local/bin #
```

## `cat /sys/devices/platform/display/monitor_info`

- On a Receiver, returns the status of the HDMI connection to the display
  - If HDMI is **NOT** attached
    - `attached=n`
  - If HDMI is attached

    -
      ```
      attached=y
      EDID=
      ```
      [EDID information]
- On a2G+ or 2G+AVP Transmitter, returns the status of the HDMI **Out** connection
  - If HDMI is **NOT** attached (or unit is 2G)
    - `attached=n`
  - If HDMI is attached

    -
      ```
      attached=y
      EDID=
      ```
      [EDID Information]

## `cat /sys/devices/platform/display/timing_info`

- Returns the timing, type, HDCP status, and color depth of the video being sent by a Transmitter or to a Receiver
  - Timing is the resolution and horizontal and vertical sync polarities
  - Type is the input signal format and screen ratio
  - HDCP is ‘y’ for yes or ‘n’ for no
  - Color depth is a single number representing the color depth
  - Example feedback (text after equal sign will change based on signal)

```
/usr/local/bin # cat /sys/devices/platform/display/timing_info
timing=[13] 1920x1080p@60Hz H+ V+
type=HDMI 16:9
HDCP=y
color depth=0
/usr/local/bin #
```

## EDID

**`capture_edid.sh`**

- Send to a Receiver to instruct it to grab the EDID of the attached display and send back to the Transmitter it is showing
- Same as performing EDID Capture on the web interface

## Image Pause

**`echo $a > /sys/devices/platform/videoip/pause`**

- **Receiver ONLY**
- Freezes the last frame of video on the display. Audio will continue. Unpausing will remove the frozen frame and restore live video. Does **NOT** pause the video output from the source device.
- $a = 0 or 1
  - 1 = pause the video
  - 0 = unpause the video
- Examples:
  - `echo 1 > /sys/devices/platform/videoip/pause`
  - `echo 0 > /sys/devices/platform/videoip/pause`

## Image Pull

**`echo $a $b > /dev/videoip;cat /dev/videoip > /www/pull.bmp`**

- Captures the frame being shown by the Transmitter/Receiver as a Bitmap and puts it in the web interface folder labeled as "pull.bmp". The image is viewable in the web interface of the device in a frame, or on its own page at [ipaddress]/pull.bmp
- $a and $b are whole-number variables
  - $a = horizontal pixel resolution of the image to be captured
    - Valid values are integers from 64 – 1080
  - $b = 0 or 1
    - 0 is high priority. More processor resources used but higher image quality
    - 1 is low priority. Less processor resources used but lower image quality
- Examples:
  - echo 320 1 > /dev/videoip;cat /dev/videoip > /www/pull.bmp
    - Capture a single 320-pixel-wide frame at low priority
  - echo 1080 0 > /dev/videoip;cat /dev/videoip > /www/pull.bmp
    - Capture a single 1080-pixel-wide frame at high priority. May affect video quality.

**`astparam s pull_on_boot $a_$b_$c;astparam save;sleep 1;reboot`**

- Sets a value in memory for Image Pull to use on startup
- $a, $b, $c are variables
  - $a = horizontal pixel resolution of the image to be captured
    - Valid values are integers from 64 – 1080
  - $b = pull priority
    - 0 = high priority. Guarantees a full frame capture but causes video degradation.
    - 1 = low priority. Image will sometimes have "tears" but does not affect video. Recommended.
  - $c = Number of seconds to wait between image pulls.
    - Must be an integer value
    - 0 = as fast as possible

**`astparam s pull_on_boot n;astparam save;sleep 1;reboot`**

- Prevents Image Pull from running on startup
- Default state

**`start_image_pull $a $b $c`**

- Start Image Pull and kill all other Image Pull processes running
- $a, $b, $c are variables
  - $a = horizontal pixel resolution of the image to be captured
    - Valid values are integers from 64 – 1080
  - $b = pull priority
    - 0 = high priority. Guarantees a full frame capture but causes video degradation.
    - 1 = low priority. Image will sometimes have "tears" but does not affect video. Recommended.
  - $c = Number of seconds to wait between image pulls.
    - Must be an integer value
    - 0 = as fast as possible

**`stop_image_pull`**

- Stop all Image Pull processes currently running

## IR Manager

**`irm.sh $bank $command $mod`**

- Teach or send an IR command to a specific bank & command combination
- $bank, $command, $mod are variables.
  - $bank = the bank that the IR command is stored in. Corresponds to ports on the IR Manager box. Valid numbers are 1 through 4.
  - $command = the command in that bank to send. Valid numbers are 1 through 32.
  - $mod = 'w' or 'x'. 'w' puts that bank in learning mode and will learn the next command that is sent at it. 'x' executes the command.

- Examples:
  - **`irm.sh 3 25 w`**
    - Sets bank 3, command 25 to learn the next IR signal sent to it
  - **`irm.sh 2 10 x`**
    - Send the IR command stored in bank 2, command 10

## Link Reset and Stop

**`e e_reconnect`**

- Interrupts and re-establishes network signal. Causes all video and USB services to restart.

**`e e_stop_link`**

- Interrupts the network signal. Device will behave as though it is not connected to a network.
- Use **`e e_reconnect`** to reconnect

## Link Watchdog

**`astparam w screen_state_watch n;astparam save ro;sleep 1;reboot`**

- Disables aggressive checking for HDMI connection
- Default setting

**`astparam w screen_state_watch y;astparam save ro;sleep 1;reboot`**

- Enables aggressive checking for HDMI connection reset
- Enable if HDMI signal loses audio or drops after a set period of time because of source device going to “sleep”

## On-Screen Display

On-Screen Display creates a 640x480 floating frame textbox in the center of the screen. The output resolution of the Receiver determines the relative size of the floating frame on the screen (640x480 appears smaller on a 1080p picture than it does on a 720p picture).

### *Commands*

**`osd_debug.sh`**

- Shows debug information about the current status of a JAP Receiver on the display. Disables after 60 seconds.

**`export OSD_STR='$string'`**

- Creates the text to be shown when On-Screen Display is turned on. Use **`osd_on.sh`** to enable OSD.
- $string is a string variable of any kind

**`export OSD_FONT_SIZE=$size`**

- Sets the font size to use when On-Screen Display is turned on. Use **`osd_on.sh`** to enable OSD.
- $size is a whole-number representing the font size

**`export OSD_FONT_COLOR='0xffffffff'`**

- Sets the font color to use when On-Screen Display is turned on. Use **`osd_on.sh`** to enable OSD.
- 0xffffffff = RGB color format (0xAARRGGBB)
  - AA = hex value for alignment of text within the
    - 00 = left-aligned
    - ff = centered
  - RR = hex value for red coloring
  - GG = hex value for green coloring
  - BB = hex value for blue coloring
- Default alignment is centered and default color is green (0xff00ff00)

**`export OSD_TRANSPARENT=$trans`**

- Sets the transparency of the On-Screen Display text when turned on. Use `osd_on.sh` to enable OSD.
- $trans is a whole-number from 1 (nearly invisible) to 15 (opaque)

**`osd_on.sh`**

- Turns on the OSD message using all *export* values previously set
- Use **`osd_off.sh`** to turn message off

**`osd_off.sh`**

- Turns off the OSD message.
- $sec is a whole-number variable
  - $sec = number of seconds to wait before activating osd_off.sh
  - Add an ampersand (&) to the end of the time variable to have it run in the background.
- Example:
  - **`osd_off.sh 15&`**

## Reboot

**`reboot`**

- Reboots the device

## Receiver HDMI Signal

**`astparam s v_turn_off_screen_on_pwr_save y;astparam save;sleep 1;reboot`**

- Cause the Receiver to automatically stop showing video after 10 seconds of no video from a Transmitter
- Requires device reboot to apply

**`astparam s v_turn_off_screen_on_pwr_save;astparam save;sleep 1;reboot`**

- Cause the Receiver to permanently show debug screen when no video is received from Transmitter
- Requires device reboot to apply
- Default state

**`echo avmute_on > /sys/devices/platform/it6613/ctrl`**

- Cause Receiver to immediately output a black screen
- Send **`echo setup_video > /sys/devices/platform/it6613/ctrl`** to resume video

**`echo shutdown > /sys/devices/platform/it6613/ctrl`**

- Cause Receiver to immediately stop video output
- Attached display will respond as if there is no HDMI cable connected
- Send **`echo setup_video > /sys/devices/platform/it6613/ctrl`** to resume video

**`echo setup_video > /sys/devices/platform/it6613/ctrl`**

- Immediately enable video output for Receiver
- Default state

## Remove Diagnostic Text on Receiver

Changes the behavior of the white diagnostic text shown when a Receiver is not showing video from a Transmitter.

**`astparam s ui_show_text n;astparam save;sleep 1;reboot`**

- Diagnostic text will **NOT** show on the debug screen of the Receiver
- Requires device reboot to apply
- WARNING: This setting will need to be changed to troubleshoot any problems.

**`astparam s ui_show_text;astparam save;sleep 1;reboot`**

- Diagnostic text will show on debug screen of the Receiver
- Requires device reboot to apply
- Default state

## Send to All Receivers

**`commandallrx.sh $command`**

- Sends `$command` to all attached Receivers in an installation.
- `$command` is a variable.
  - `$command` is any other command that can be run on a JAP device. Multi-word commands should be enclosed in quotation marks.

- Examples:
  - **`commandallrx.sh osd_debug.sh`**
    - Sends `osd_debug.sh` to all Just Add Power Receivers
  - **`commandallrx.sh "e e_vw_enable_0_0_0_0"`**
    - Sends the single-screen command to all Just Add Power Receivers

## Serial

**`stty –F /dev/ttyS0 $baud`**

- Set the baud rate of the serial port to match the baud rate of the receiving device
- $baud is a whole-number variable
- $baud = 9600, 19200, 38400, 115200, etc.

**`printf "$string" > /dev/ttyS0`**

- $string is a string variable
  - $string = string to be sent out serial port of the Just Add Power device
  - Can be ASCII or hex
    - Hex pairs must be preceded with "\x"

## Slideshow

**`slideshow.sh $sec $slides`**

- Rotate through a slideshow of the images loaded onto the Just Add Power Receiver
- $sec and $slides are whole-number variables
  - $sec = number of seconds to show each slide
  - $slides = number of slides to show in the rotation before stopping

## Transmitter HDMI Signal

**`hdmi_off.sh`**

- Disable the HDMI input port on Transmitter
- Transmitter will behave as if HDMI cable is not connected
- Send **`hdmi_on.sh`** to enable HDMI video

**`hdmi_on.sh`**

- Enable the HDMI input port on Transmitter
- Default state

## Transmitter Video Quality Mode

**`astparam s ast_video_quality_mode $a;astparam save;sleep 1;save`**

- $a is a variable with values -1, 0, 1, 2, 3
  - -1 = Dynamic adjustment based on content being shown
    - Default value
  - 0 = Codec optimized for static images
  - 3 = Codec optimized for constant motion
  - 1 or 2 = Codecs for intermediate content

## Video Wall

**`e e_vw_enable_$a_$b_$c_$d`**

- $a, $b, $c, $d are zero-based, whole-number variables from 0 through 15 (0_0_0_0 = single screen)
    - $a = number of rows in the entire wall
    - $b = number of columns in the entire wall
    - $c = row position of this screen
    - $d = column position of this screen

**`e e_vw_moninfo_$a_$b_$c_$d`**

- $a, $b, $c, $d are whole-number variables. These values are set as part of the initial video wall setup procedure and ONLY need to be input once.
    - $a = video screen width (viewable portion)
    - $b = frame width (physical frame)
    - $c = video screen height (viewable portion)
    - $d = frame height (physical frame)

**`e e_vw_delay_kick_$a`**

- Fixes "tearing" in video wall lower rows. Equal resolutions have equal tearing values
- $a is a whole-number variable
    - $a = tearing delay in microseconds (1000 microseconds = 1 millisecond)

**`vw_scale.sh`**

- Scales a video wall screen to remove the appropriate number of pixels for video wall displays. Corrects any "duplication" of pixels shown from one screen to another.
- Pair with commandallrx.sh to perform on all screens at once.
- If this command is performed, no other scaling or shifting commands are needed.
- Present in firmware A5.13d and later.

**`e e_vw_h_scale_$a`**

- Scales a video wall screen horizontally by $a pixels.
- $a is a whole-number variable
    - $a = number of pixels to scale horizontally

**`e e_vw_v_scale_$a`**

- Scales a video wall screen vertically by $a pixels.
- $a is a whole-number variable
    - $a = number of pixels to scale vertically

**`e e_vw_h_shift_l_$a`**

- Shifts a video wall image to the left by $a * 8 pixels.
- $a is a whole-number variable
    - $a = number of pixels shifted to the left * 8

**`e e_vw_h_shift_r_$a`**

- Shifts a video wall image to the right by $a * 8 pixels.
- $a is a whole-number variable
  - $a = number of pixels shifted to the right * 8

**`e e_vw_v_shift_u_$a`**

- Shifts a video wall image up by $a * 8 pixels.
- $a is a whole-number variable
  - $a = number of pixels shifted up * 8

**`e e_vw_v_shift_d_$a`**

- Shifts a video wall image down by $a * 8 pixels.
- $a is a whole-number variable
  - $a = number of pixels shifted down * 8

## 1080i Scaling

```
astparam s v_1080i_to_720p $a;astparam save;sleep 1;reboot
```

- $a is a variable with value of 'y' or 'n'
  - n = convert 1080i video to 1080p video
    - Default value
  - y = convert 1080i video to 720p video

# 2G+

## Audio Delay

**AUD_DLY_SLOW $a**

- Increases the audio delay value by $a. Increases by 1 if $a not specified. Use AUD_DLY_SAVE_VAL to save new value.
- $a is a whole-number variable from 0 to 25

**AUD_DLY_FAST $a**

- Decreases the audio delay value by $a. Decreases by 1 if $a not specified. Use AUD_DLY_SAVE_VAL to save new value.
- $a is a whole-number variable from 0 to 25

**AUD_DLY_SET_VALUE $a**

- Sets the audio delay value to $a. Use AUD_DLY_SAVE_VAL to save new value.
- $a is a whole-number variable from 0 to 25

**AUD_DLY_SAVE_VAL**

- Saves current audio delay value to flash memory.

**AUD_DLY_RESTORE_VAL**

- Restores audio delay value last saved to flash memory.

**AUD_DLY_READ_CURRENT**

- Displays current audio delay value in flash memory.

## CEC

**cec_tv_off.sh**

- Sends the CEC command to turn a display off

**cec_tv_on.sh**

- Sends the CEC command to turn a display on

**cec_audio_on.sh**

- Sends the CEC command to enable audio

**cec_audio_off.sh**

- Sends the CEC command to disable audio

**cec_watch_me.sh**

- Sends the CEC command to select HDMI 1

**CEC_SEND_BYTES $aa $bb $cc $dd**

- Sends the hex values entered as CEC codes.
- $aa, $bb, $cc, $dd are hex variables. (00 through ff)
    - Example:

```
CEC_SEND_BYTES ef 82 10 00
```

# 2G+AVP

## HDMI Audio

### `dd_main_on.sh`

- Enables the HDMI-In audio signal
- Works on Transmitter only
- Default state

### `dd_main_mute.sh`

- Mutes the HDMI-In audio signal
- Works on Transmitter only

## Line In

### `dd_line_on.sh`

- Enables the Line In port audio
- Works on Transmitter only

### `dd_line_mute.sh`

- Mutes the Line In port audio
- Works on Transmitter only
- Default state

## Microphone In

### `dd_mic_on.sh`

- Enables the Mic-In port audio
- Works on Transmitter only

### `dd_mic_mute.sh`

- Mutes the microphone port audio
- Works on Transmitter only
- Default state

## Stereo port audio adjust

**`vol_bth_up.sh`**

- Increases the left- and right-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_bth_dn.sh`**

- Decreases the left- and right-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_l_up.sh`**

- Increases the left-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_l_dn.sh`**

- Decreases the left-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_r_up.sh`**

- Increases the right-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_r_dn.sh`**

- Decreases the right-channel audio on the Stereo port by 1 unit
- Valid audio levels are 0-31

**`vol_bth_set.sh X`**

- Sets the left- and right-channel audio on the Stereo port to an exact value (X)
- Valid audio levels are 0 (lowest) to 31 (highest)
- Example:
  - **`vol_bth_set.sh 16`**

**`vol_r_set.sh X`**

- Sets the right-channel audio on the Stereo port to an exact value (X)
- Valid audio levels are 0 (lowest) to 31 (highest)
- Example:
  - **`vol_r_set.sh 16`**

**`vol_l_set.sh X`**

- Sets the left-channel audio on the Stereo port to an exact value (X)
- Valid audio levels are 0 (lowest) to 31 (highest)
- Example:
  - **`vol_l_set.sh 16`**

**`vol_mute.sh`**

- Mutes the Stereo port

**`vol_unmute.sh`**

- Unmutes the Stereo port

## Stereo port volume read

**`volume_read.sh`**

- Returns the audio-level adjustment on the Stereo port for both left- and right-channel
  - Format:
    - Example 1: “`Left: +4.5 db Right: +4.5 db`”
    - Example 2: “`Left: -39 db Right: -39 db`”

- Returns “`No version queuery`” if not a 2G+AVP Receiver or Transmitter
  - Can be used to logically identify if a unit is 2G+ AVP or not

# 3G+AVP

## Image Pop

The background assigned to the device via Image Push can be shown as an overlay on the screen. The first pixel in the upper-left corner will be used as the transparency color for the overlay. The image will be shown at 640 x 480 resolution, and will take up that many pixels on the display (example: if the display is 1920 x 1080, it will take up 640 x 480 of the screen. If the display is 3840 x 2160, it will still take up 640 x 480 pixels, but appear smaller because of the larger pixel resolution).

**`image_pop on`**

- Enable Image Pop and place the image in the center of the screen
- Image Pop will remain on until the *off* command is sent

**`image_pop on tl`**

- Enable Image Pop and place the image in the top-left corner of the screen
- Image Pop will remain on until the *off* command is sent

**`image_pop off`**

- Disable Image Pop

## On-Screen Display Enhanced

In addition to all the On-Screen Display commands present in 2G, 3G adds the ability to move the floating frame textbox to different areas of the screen.

### *Commands*

**`osd_position.sh &position &x &y`**

- Changes the alignment, x-position, and y-position of the On-Screen Display floating frame textbox
- `$position` is a number from 0 – 9 and sets the position of the textbox on the screen
    - 0: Disable alignment overwrite. Use default behavior
    - 1: Top-Left
    - 2: Top
    - 3: Top-Right
    - 4: Left
    - 5: Center
    - 6: Right
    - 7: Bottom-Left
    - 8: Bottom
    - 9: Bottom-Right

- $x sets the number of horizontal pixels to offset the textbox from the position
    - 0 is the default
    - Negative values shift left
    - Positive values shift right

- $y sets the number of vertical pixels to offset the textbox from the position
    - 0 is the default
    - Negative values shift up
    - Positive values shift down

- Example:
    - `osd_position.sh 1 100 200`
        - Sets the textbox in the top-left and shifts it right 100 pixels and down 200 pixels

## Video Scalar (Receiver only)

Each 3G Receiver has a built-in scalar. These commands will change the scalar immediately to output the indicated resolution.

### *Pass-Through*

```
echo 0 > /sys/devices/platform/videoip/output_timing_convert
```

### *1080p60*

```
echo 80000010 > /sys/devices/platform/videoip/output_timing_convert
```

### *1080p50*

```
echo 8000001F > /sys/devices/platform/videoip/output_timing_convert
```

### *2160p30*

```
echo 8000005F > /sys/devices/platform/videoip/output_timing_convert
```

### *2160p25*

```
echo 8000005E > /sys/devices/platform/videoip/output_timing_convert
```

## Video Wall Rotation

All 2G video wall commands are functional on 3G devices. In addition to those commands, 3G has the ability to rotate, mirror, and shift the video wall image to create new configurations not possible on 2G.
‘$’ indicates a variable. CLI command:

```
e e_vw_rotate_$A
```

$A is a variable from 0 – 7

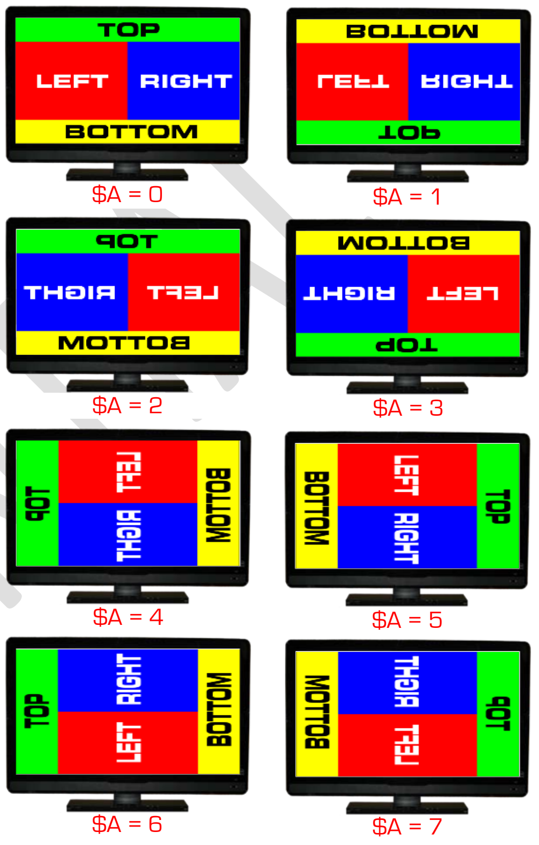
- 0 = normal
- 1 = vertical mirror
- 2 = horizontal mirror
- 3 = $180^{o}$ rotation
- 4 = $90^{o}$ clockwise rotation + horizontal mirror
- 5 = $90^{o}$ clockwise rotation
- 6 = $270^{o}$ clockwise rotation
- 7 = $270^{o}$ clockwise rotation + horizontal mirror

*Example 1:* Rotate the image $180^{o}$

```
e e_vw_rotate_3
```

*Example 2:* Rotate the image $90^{o}$ clockwise

```
e e_vw_rotate_5
```


TOP
LEFT
RIGHT
BOTTOM
$A = 0
$A = 1
$A = 2
$A = 3
$A = 4
$A = 5
$A = 6
$A = 7