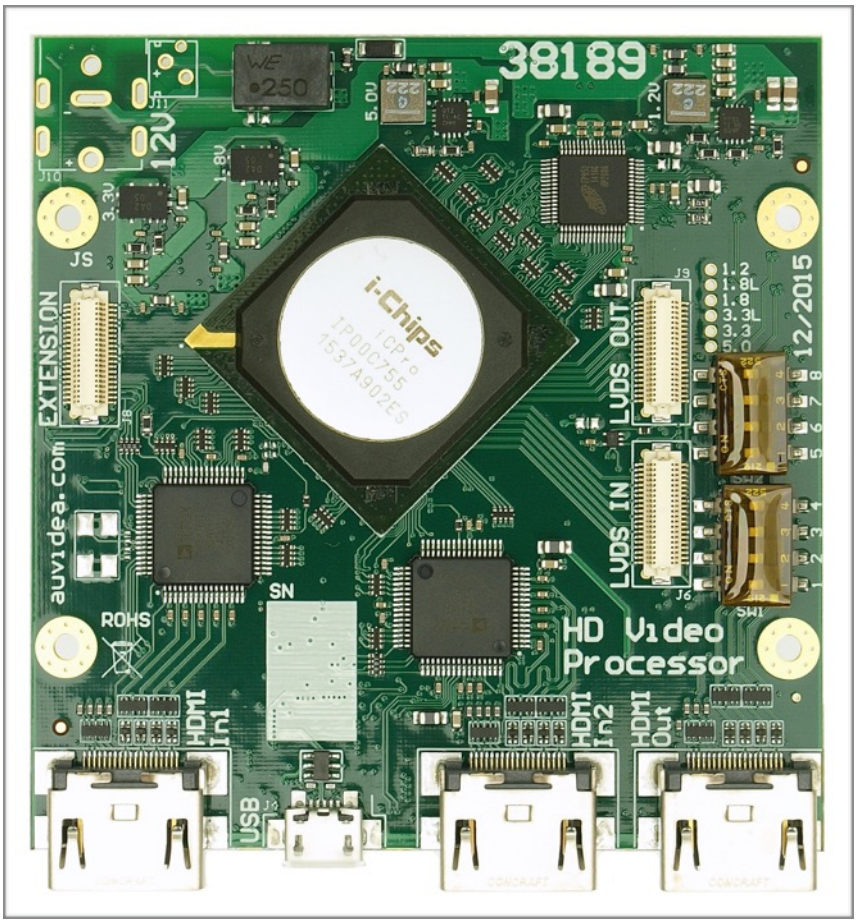
V100 HD Video Processor

Technical Reference Manual



Version 1.1.5

November 2016

Auvidea GmbH
Kellerberg 3
D-86920 Denklingen

Tel: +49 8243 7714 622
info@auvidea.com
www.auvidea.com

Copyright Notice

Introduction

V100 HD Video Processor

General purpose HD video processor for 2 inputs and 1 output. Add-on modules for additional inputs and outputs and audio mixing are planned.

background graphic + **video 1** + **video 2**

Background graphic image, video 1 and video 2 are merged live into a combined output video. Three examples are shown on the right:

- scaling (x and y independently)
- cropping
- de-interlacing
- flexible window positioning (with optional overlap)
- dynamic OSD alterations by local host computer (like Raspberry Pi) via SPI interface

Applications

- 3D video mixing (side by side or top/bottom)
- video scaling for Raspberry Pi (with B101 HDMI to CSI-2 module)
- lecture recording (mixing of presenter and presentation video)
- interactive OSD for motor sports (driver, track and data)

Technical details

- 2 HDMI inputs up to 1080p60
- HDMI output up to 1080p60
- scaling, cropping, de-interlacing, format conversion
- full graphic OSD with 256 indexed colors
- PiP, PoP, side-by-side, 3D view, and up to 20 configuration presets
- UART remote control interface for import of OSD graphics (BMP file) and control of video window configuration

V100 rev 3 improvements (38189-3)

- TTL UART jack on the front

Firmware Releases

Release 0.8.1 (August 2016)

- please issue „lmem 1“ and „osd off“ commands before uploading an OSD BMP file via XMODEM

Terminal software for UART console

OS X El Capitan (Version 10.11.4)
minicom version 2.2

Windows 7

Tera Term (4.90 and 4.92 tested)
Extra PuTTY 0.29_RC2

Debian (Linux debian 3.2.0-4-686-pae #1 SMP Debian 3.2.73-2+deb7u3 i686 GNU/Linux)

minicom Version 2.6.1

Timing

The V100 supports various output timings (video resolutions):

active: the visible resolution of the video (horizontal, vertical)
total: the total resolution of the video (including front and back porch)
e.g. 1650 x 750 x 60 = 74,250,000 Hz = pixel frequency

input timing	resolution	vic	active	total	command
576p50	720x576 p (50 Hz)	17	[720,576]	[864,625]	c 17
720p50	1280x720 p (50 Hz)	19	[1280,720]	[1980,750]	c 19
720p60	1280x720 p (60 Hz)	4	[1280,720]	[1650,750]	c 4
1080p24	1920x1080 p (24 Hz)	32	[1920,1080]	[2750,1125]	c 32
1080p25	1920x1080 p (25 Hz)	33	[1920,1080]	[2640,1125]	c 33
1080p30	1920x1080 p (30 Hz)	34	[1920,1080]	[2200,1125]	c 34
1080p50	1920x1080 p (50 Hz)	31	[1920,1080]	[2640,1125]	c 31
1080p60	1920x1080 p (60 Hz)	16	[1920,1080]	[2200,1125]	c 16

Example:

```
c 31                              - set the video output to 1080p50
```

a mixer with custom presets

The V100 supports 21 (0 to 20) configuration sets. The V100 comes with some configuration sets pre-programmed, so the system works out of the box. Each configuration set includes a full screen OSD graphic (1920x1080) with 256 colors and its color palette, and all input and output parameters.

These configuration sets may be customized and saved, so up to 19 custom configuration sets may be stored in the V100. All sets may be backed up and restored. In this manner custom configurations of the V100 may be cloned to other V100 modules.

Configuration sets can be selected by the following methods:

1. first the Auvidea welcome display is shown for 10 seconds. It shows 2 preview windows (16 by 9) of both video inputs and displays the input resolution and format.
2. a specific configuration set may be defined as the default configuration set, which comes up 10 seconds after the initial Auvidea welcome display.
3. with a UART connection the configuration set may be switched (and customized)
4. the 8 DIP switches on the V100 may be used to select the default configuration set (future option)
5. a DMX interface allows the live control of parameters and the selection of configuration sets with a DMX light control table (future option)
6. a WiFi interface allows the remote live control of parameters and the selection of configuration sets (future option)

So the V100 could be set up as a remote controllable video mixer. So it can be switched between multiple presets:

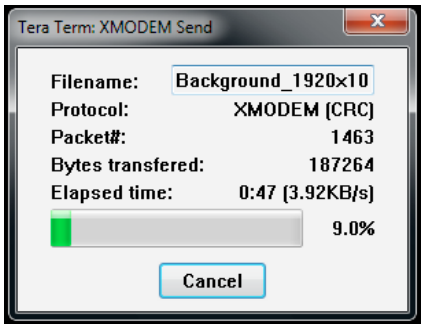
1. video 1 full screen
2. video 2 full screen
3. video 1 in a window to the left of video 2
4. video 1 in a window to the right of video 2
5. video 1 as picture in picture inside of video 2
6. video 1 and video 2 side by side (3D mode)
7. selection of a custom configuration set
8. display just a computer graphic without live video (pause and break display
9. .. plus many more

Tutorial 1

Create a new configuration set

Create an new configuration set and store it in the internal flash as set 20. This configuration set has a full screen background image and 2 video windows (1080p inputs).

1. start V100 and establish a UART connection to a host PC. For this example we connect the V100 via a DeLOCK USB to UART cable to a Windows PC. On the Windows PC the application Tera Term is installed. Please see the Tera Term installation for details



2. send „Background_1920x1080_255c.bmp“ via XMODEM to the V100.
   - enter „xmodem“ in the Tera Term console. The V100 will respond with a sires of Cs to indicate that ii is waiting for a transfer.
   - open „file - transfer - XMODEM - send..“ and select „Background_1920x1080_255c.bmp“ file. The transfer dialog will open (see screenshot on the right). The transfer will take a few minutes. As the file name indicates this BMP file has 255 indexed colors (0 to 254).

```
>xmodem
CCCCC
transfer complete
1920x1080 pixel transmitted      - this should match the size of the BMP image
```

3. define the OSD image size and transparency color (255). The entire color range is 0 to 255. The OSD is full screen and transparency is turned on. Any pixel in the OSD with a color 255 will become transparent. As the background image only has 255 color, so colors from 0 to 254, none of the background pixels will be transparent. You should now see the background image full screen.

```
>osd 0 1920 0 1080 1 255
```

4. define the first video window (video 1). The video window size is 300x500 pixels and the starting position from the top left corner is 100/100 (horizontal/vertical). The „paint“ command overwrites the OSD image with a rectangular window and fills it with a specific index color. Please use the „paint“ command with caution, as this in not reversible. If you make a mistake, the OSD needs to be re-transferred. Here we want this to become transparent, so the color is 255. In this example the input and output properties are identical. This means that this video window just shows a windows into video 1. Video 1 is not scaled. It is just cropped so that it fits the 300x500 output window. Alternatively select one of the automatic input configuration modes.

```
>paint 100 300 100 500 255       - write a transparent window
>i 1 100 300 100 500             - define the video 1 input properties
>o 1 100 300 100 500             - define the video 1 output properties
```

5. define the second video window (video 2). The video window size is 1300x800 pixels and the starting position from the top left corner is 500/100 (horizontal/vertical) so it is positioned to the right of window 1. Again input and output properties are the same. This again just performs cropping and no scaling.

```
>paint 500 1300 100 800 255       - write a transparent window
>i 2 500 1300 100 800             - define the video 2 input properties
>o 2 500 1300 100 800             - define the video 2 output properties
```

6. now lets change video 2 settings so the video windows show the full video 1. So now it should be scaled and cropped to fit the 1300x800 video window. Please note that the aspect ratio of the video input (1920x1080 = 16 by 9 = 1.77) is different from the output window (1300x800 = 1.625). The V100 can fit video 2 into this window keeping pixels square, when the inputs and outputs are adjusted correctly. First lets crop of 350 pixels on the left side of video 2. So the horizontal size is 1920 - 350 = 1570. The height of the input video is computed by the formula: **height = <input hsize> / <output hsize> * <output vsize> = 1570 / 1300 * 800 = 966**

```
>i 2 350 1570 0 966               - define scaling and cropping
```

7. save this configuration set in flash location #20.

```
>wmem 20                          - save configuration set
```

8. restore the new configuration set.

```
>lmem 0                                 - load init screen
>lmem 20                          - load the new configuration set
```

9. now you have created your first custom configuration set. Congratulations.

Tutorial 2

Add logo overlay (2 colors)

Download an Auvidea logo and display it on top of video 1 (full screen). The „logo_256x116_2c.bmp“ is a BMP file with 2 indexed colors. Size: 256x116 pix. File size: 3774 byte. As this file is very small it can be downloaded very quickly (for test purposes).

1. send „logo_256x116_2c.bmp“ via XMODEM to the V100. Please have a look at tutorial 1 for details.

```
>xmodem
CCCCC
transfer complete
256x116 pixel transmitted        - this should match the size of the BMP image
```

2. define the OSD image size and transparency color (1). The image has two colors. Color 0 is the dark read logo and color 1 is the white background of the logo. We want to turn the background transparent. You should now see the logo in the top left corner of the display.

```
>osd 0 256 0 116 1 0             - turns the logo color transparent
>osd 0 256 0 116 1 255           - the logo is opaque
>osd 0 256 0 116 1 1             - turns the white background transparent
```

3. define the first video window (video 1) as full screen.

```
>full
```

4. now you should see a logo with transparent background overplayed over the video.

Tutorial 3

Add logo overlay (multiple colors)

Download an Auvidea logo and display it on top of video 1 (full screen). The „logo_256x116.bmp“ is a BMP file with multiple indexed colors. Size: 256x116 pixel. File size: 14954 byte.

1. send „logo_256x116.bmp“ via XMODEM to the V100. Please have a look at tutorial 1 for details.

```
>xmodem
CCCCC
transfer complete
256x116 pixel transmitted        - this should match the size of the BMP image
```

2. next we need to determine the color index for the white background. You may get this data from your paint application, which generated this image. But it is also possible to determine the color from the V100, now that we have downloaded the image file and its color palette. Please use „rcp“ to print out the color palette in the V100. Then search for the index color of white - while is FF for red, green and blue. This is easy to find, as it is the first entry in the color palette table. It is marked red below. The numbers are in hexadecimal format (FF = 255).

```
>rcp
RED color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 99 33 99 66 66 CC 99 66 FF CC 00 E4 74 89
 10: 9D A6 B1 B8 C0 C4 D4 EB 7A 89 90 DD E1 DA F5 B1
 20: F0 F7 F2 EC E6 CD D8 FA FD E6 ED 1D 24 22 32 2C
 30: 2B 3B 37 4E 43 40 5C 5A F6 80 FE 2C 2B E1 D5 F7
 40: D9 91 ED DC DB C1 70 96 95 8D 89 88 85 A7 A4 A2
 50: 99 98 6D 68 7B 79 76 74 72 6F 6E 8A 86 80 7C 9A
 60: 6F 6D 6B 69 67 81 EE FA C8 F7 D2 C2 B2 FD FB F9
 70: F5 F3 E3 D9 D1 CB C4 FE DF FD EB F0 FA F8 FB FB
 80: FC FB F6 FE FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: B3 57 CA 77 55 DB 5F 3F 63 2D 8B F6 10 3C B0 00
GREEN color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 33 33 99 66 00 99 66 33 CC 66 00 D7 15 2A
 10: 56 67 74 83 93 9F B6 E1 23 37 46 C4 D0 CA ED 89
 20: E3 F0 EB E7 DD BD CF F7 FB E1 EA 1C 23 21 31 2B
 30: 2A 3A 36 4D 42 3F 5B 59 F5 7F FD 2C 2B E1 D5 F7
 40: D9 91 EE DD DC C2 72 98 97 8F 8B 8A 87 A9 A6 A4
 50: 9B 9A 70 6B 7E 7C 79 77 75 72 71 8D 89 83 7F 9D
 60: 73 71 6F 6D 6B 85 F2 FE CB F9 D4 C4 B4 FE FC FA
 70: F6 F4 E4 DA D2 CC C5 FF E0 FC DE E9 F5 F3 F7 F8
 80: FA F9 F4 FD FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
```

```
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: 34 3A EB CF A8 CF 65 82 DF 61 14 95 B1 59 C7 00
BLUE color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 33 33 99 66 00 99 66 33 CC 66 00 D8 1E 33
 10: 5E 6E 7A 89 98 A3 B9 E2 2E 41 50 C7 D2 CC EE 8F
 20: E5 F1 EC E8 DF C1 D2 F8 FC E4 EC 1D 24 22 32 2C
 30: 2B 3B 37 4E 43 40 5C 5A F6 80 FF 2D 2C E3 D7 F8
 40: DA 92 EF DE DD C3 73 99 98 90 8C 8B 88 AA A7 A5
 50: 9C 9B 71 6C 7F 7D 7A 78 76 73 72 8E 8A 84 80 9E
 60: 74 72 70 6E 6C 86 F3 FE CB F9 D4 C4 B4 FE FC FA
 70: F6 F4 E4 DA D2 CC C5 FE DF FB DE E9 F5 F3 F7 F8
 80: FA F9 F4 FD FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: B0 DF E2 97 46 6E F3 8B 50 1E C1 AF F6 E8 64 00
```

3. define the OSD image size and transparency color as determined above (0). We want to turn the background transparent. You should now see the color logo in the top left corner of the display.

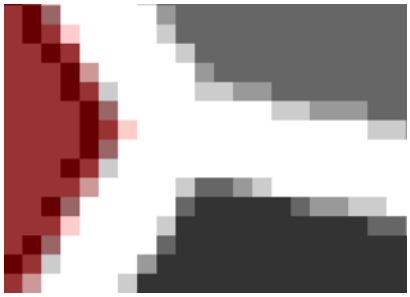
```
>osd 0 256 0 116 1 0                    - turns the logo color transparent
```

4. define the first video window (video 1) as full screen.

```
>full
```

Now you should see a logo with transparent background overplayed over the video. You may note the white edges of the logo. This is due to the fact that the image was created with Adobe Photoshop and „aliasing of the edges“ was enabled. Please see the enlarged portion of the logo and the grey pixels around the edges. These will show up opaque. Only true white pixel are turned transparent.

In tutorial 2 there was no white edge. If you would like to remove this edge then please make sure that you create the image with aliasing turned off.

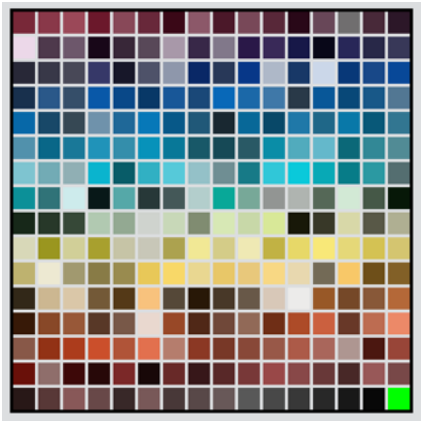


Tutorial 4

Custom shape video window

The standard procedure to define the location and size of the video window is to use the „paint“ command. You may use multiple paint commands to define a single video window which consists of multiple rectangular areas.

Alternatively you may use your favorite paint application to define the video window already when creating the OSD graphic.



Here it is important to create an image which uses 255 index colors for the background and a unique index color (like 255) for the video windows(s).

Here we show how to create such an image with Adobe Photoshop Elements 11.

1. open the background image and resize it to 1920x1080 pixels.
2. open dialog „image - mode - indexed colors ..“ and specify 255 colors. This will convert the image from RGB to indexed colors. A custom color palette will be created. See on the right.
3. open this color palette in the dialog „image - mode - color palette“. The box on the lower right should currently be grayed out, indicating that this color index is not used. Click on this box and define a unique and extreme color such as bright green in our example. This color is red=0, green=255 (FF) and blue=0.
4. use any drawing tools and create any window shape. There is just one limitation. The V100 can put 2 video windows underneath. They may overlap each other. But your custom video windows must fall into those rectangular video windows.
5. save as BMP file.
6. upload with XMODEM.
7. define the OSD size and transparency color. OSD is full screen. The transparent color is the color in the palette box on the bottom right (index: 255). Do not use the „paint“ command, as the video windows are already defined with a unique index color.

```
>osd 0 1920 0 1080 1 255                  – turns green custom shapes transparent
```

8. define the first video window (video 1) as full screen. Alternatively define custom video windows as shown in the previous tutorials.

```
>full
```

Tutorial 5

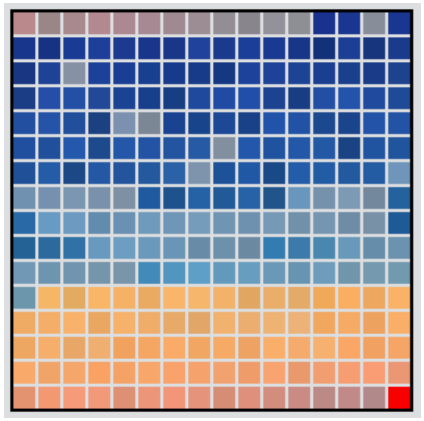
2 custom shape video windows

Here the procedure is very similar to the previous tutorial. use a paint application like Adobe Photoshop Elements 11 and define the video windows (1080p inputs) with a unique index color. Here we used also the last index (255). We defined it as bright red: red=255 (FF), green=0 (00) and blue=0 (00).

Please notice that the color palette here is very different. Here is is optimized for this specific background image, which mainly has orange and blue colors.

1. upload the BMP file with XMODEM.
2. define the OSD size and transparency color. OSD is full screen. The transparent color is the color in the palette box on the bottom right (index: 255).

```
>osd 0 1920 0 1080 1 255
```



3. define the properties for the first video window (video 1).

```
>i 1 760 500 0 1080          - centers video 1 (1080p)
>o 1 50 500 0 1080
>win 1                       - read back properties
50,500,00,1080,760,500,0,1080
```

4. define the properties for the 2nd video window (video 2).

```
>i 2 310 1300 40 1000        - centers video 2 (1080p)
>o 2 580 1300 40 1000
>win 2                       - read back properties
580,1300,40,1000,310,1300,40,1000
```

How to compute scaling and cropping manually?

If the video 2 input is changed from 1080p to 720p, the video input settings need to be adjusted, so that video 2 is cropped and then upscaled. First we like to crop the 10 top lines and the 10 bottom lines, so <vsize> of the input is 720-10-10 = 700. Pixels should stay square, so the aspect ratio of the input window and the output window must be the same.
<hsize>/<vsize> = <hsize>/700 = 1300/1000 => <hsize> = 1300/1000*700 = 910.
Last the input video window should be centered in the source video. The 720p video input provides 1280 pixel horizontally. So <hstart> = (1280 - <hsize>) / 2 = (1280 - 910) / 2 = 185.

```
>i 2 185 910 10 700          - centers video 2 (720p input)
```

Alternatively the automatic „zoom“ mode could be used to fit video 1 and video 2 into their output video windows.

```
>zoom 1
>zoom 2
```

Input

Automatic input configuration

In a lecture recording system you may want to connect one HDMI input to a camera for the presenter. Here the resolution will be fixed and may be set by the manual „i“ command. The second HDMI input may connect to the HDMI output of the presentation system (e.g. a notebook). In some setups changing speakers may bring their notebooks which all provide different formats and resolutions. The V100 can be configured with 3 automatics modes to automatically adjust the video properties to dynamically scale the output of the notebook into the presentation window.

`fill <ch>`

Completely fills the output window with the input video. If the aspect ratio of input and output window are not the same then the video will be distorted (the aspect ratio of the horizontal to the vertical size will be adjusted) so it fits into the output window. Please use this mode, if the video should not be cropped or black borders should not be added.

`fit <ch>`

Preserves the original aspect ratio by adding black bars (letterbox / pillarbox) to the left and right side or to the top and bottom.

`zoom <ch>`

Zooms the input video so it completely covers the output window. If the aspect ratio is not the same, then a part of the input video is cut off, to make it fit.

`none <ch>`

Leaves the the size of the input video untouched

Manual input configuration

`i <ch> <hstart> <hsize> <vstart> <vsize>`

Manual input configuration. Please use this mode, if the input resolution is fixed.

Parameters

ch	1 .. 2	video 1 or video 2
hstart	0 .. 9999	hstart=0: left side
hsize	0 .. 9999	maximal the horizontal size of the output (1280 or 1920)
vstart	0 .. 9999	vstart=0: top
vsize	0 .. 9999	maximal the vertical size of the output (720 or 1080)

Output

Basic output modes

full	video 1 displayed full size
sbs	video 1 and video 2 side by side (3D mode)
pip	picture in picture
pip tl	top left
pip tr	top right
pip bl	bottom left
pip br	bottom right
swap	swap video 1 and video 2

Properties of the output video window

o <ch> <hstart> <hsize> <vstart> <vsize>

Define the size and the position of the video output window. For a full screen video please see the samples below. For video window examples please have a look at the tutorials.

```
>o 1 0 1920 0 1080                - 1080p output (1920x1080)
>o 1 0 1280 0 720                 - 720p output (1280x720)
```

Parameters

ch	1 .. 2	video 1 or video 2
hstart	0 .. 9999	hstart=0: left side
hsize	0 .. 9999	maximal the horizontal size of the output (1280 or 1920)
vstart	0 .. 9999	vstart=0: top
vsize	0 .. 9999	maximal the vertical size of the output (720 or 1080)

OSD

The V100 supports a full screen OSD (on screen display) graphic overlay up the 1920x1080 with 256 colors. The OSD graphic file may be prepared on a PC and then downloaded via xmodem and the UART connection to the V100. The graphic file must be saved in the BMP format with up 240 indexed colors. The top 16 colors are reserved for internal use in the V100. By default color 255 is the transparency color.

xmodem

Upload OSD image file (BMP file with 1, 2, 4 or 8 bits/pixel - up to 255 colors). If the graphic is shown upside down please correct the BMP setting in the PC application. Adobe Photoshop Elements 11 saves the right orientation by default.

osd off

Disables OSD graphic display. To enable OSD please use the command below.

osd <hstart> <hsize> <vstart> <vsize> <transparancy> <color>

Enables OSD display. <hstart> is the horizontal start of the OSD display. 0 means the OSD starts on the left side of the video display. <hsize> is the horizontal size and it must match the size of the downloaded BMP file. For a 1080p display the horizontal size is 1080. <vstart> is the vertical start. 0 is the top. <vsize> is the vertical size of the OSD. If transparency is enabled then any pixel with the color of 255 (0xFF) will show the video placed underneath. After download the OSD graphic please use the paint command to paint the video window with the transparency color (255).
Example: osd 0 1920 0 1080 1 255 (this shows a full screen OSD on a 1080p display)

paint <hstart> <hsize> <vstart> <vsize> <color>

Fills a defined rectangular area with a specific color. This command is used to define a video window, after a new OSD graphic file has been downloaded into the V100.
Example: osd 100 400 100 300 255 (this creates a video window of 400x300 pixel at starting location 100/100)

Parameters

hstart	0 .. 1920	
hsize	0 .. 1920	
vstart	0 .. 1080	
vsize	0 .. 1080	
transparancy	0 or 1	0 = disabled, 1 = enabled
color	0 .. 255	transparancy key color (default:255)

Configuration memory

The V100 features 64 MBytes of on-board Flash memory. This is used to store configuration sets. 21 sets are supported. Each set is 11x256 kByte = 2816 kByte. A configuration set defines all parameters for a given configuration including the full screen OSD graphic up to a resolution of 1920x1200 pixel and its color palette.

To define and store a new configuration set please use any UART API instructions to download the OSD and to define any video display mode including the full custom definition of 2 video windows. Once the configuration is complete it may be saved to Flash with the „wmem“ command.

The V100 is shipped with a preprogrammed Flash with a set of sample configuration sets.

The entire Flash memory may be backed up and restored. Please see the backup section for more details.

lmem <number>

Load configuration set from Flash memory. Set 0 is reserved for the Auvidea welcome display. It is the default display which is shown after power up.

```
>lmem 1             - please select a set from 0 to 20
```

rmem

Lists memory locations utilized. They are marked with an „x“.

```
>rmem
 0: x
 1: x
 2: x
 3: x
 4: x
 5: x
 6: free
 7: free
 8: x
 9: free
10: x
11: free
12: free
13: free
14: free
15: free
16: free
17: free
18: free
19: x
20: x
```

wmem <number>

Write configuration set to Flash memory. The store operation takes just a few seconds.

```
>wmem 19
... writing flash memory - please wait!
9... finish
```

If the Flash location is already occupied, please confirm with <ctrl> y. Stop with any other key.

```
>wmem 20
storage space 20 is already occupied!
overwrite?
          yes: ^Y       no: anykey
storage will be erased - please wait
... (0) erasing flash memory now - please wait!
... (1) erasing flash memory now - please wait!
... (2) erasing flash memory now - please wait!
... (3) erasing flash memory now - please wait!
... (4) erasing flash memory now - please wait!
... (5) erasing flash memory now - please wait!
... (6) erasing flash memory now - please wait!
... (7) erasing flash memory now - please wait!
... (8) erasing flash memory now - please wait!
... (9) erasing flash memory now - please wait!
... (10) erasing flash memory now - please wait!
... writing flash memory - please wait!
... (10) erasing flash memory now - please wait!
... finish
```

Parameter

number	*0 .. 20*

MISCELLANEOUS

v prints current firmware version

```
>v
```

t prints timing

```
>t
HDMI IN CH1  1024x 768p85   RGB
HDMI IN CH2  1920x1080i50   RGB
HDMI OUT     1920x1080p60   RGB
```

td prints timing (more detailed)

```
>td
                HDMI IN CH1    VP IN CH1
interlaced                0            -
hstart                  208          201
active width           1024          800
vstart                    36           36
active height            768          768
total width             1376         1375
total height             808          805
field rate                85           85
piclk               94507000     94152800
hsync                     96            -
vsync                      3            -
colorspace       RGB LIMITED            -

                HDMI IN CH2    VP IN CH2
interlaced                1            -
hstart                  148          141
active width           1920          800
vstart                    20           20
active height          1080         1080
total width            2640         2639
total height           1125          557
field rate               50           50
piclk              74257000     73524000
hsync                     44            -
vsync                      5            -
colorspace       RGB LIMITED            -

                   HDMI OUT       VP OUT
active width              -         1920
active height             -         1080
total width            2200         2200
total height           1125         1125
active width              -          800
active height             -          451
color in                RGB            -
color out           RGB 444   YUV444/RGB
hsync                     -           44
vsync                     -            5
```

r 0x36

Print out EDID for video input 1. This is the base (128 Byte) and extended EDID (128 Byte) for input 1. The bytes are shown in hexadecimal format 00=0 and FF=255. The EDID specifies the video resolutions supported by the V100. When customized it allows the connected video source to be forced into a specific mode.

```
>r 0x36
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00: 00 FF FF FF FF FF FF 00 41 A5 01 00 00 00 00 00
10: 1A 17 01 03 80 10 09 78 0A EE 91 A3 54 4C 99 26
20: 0F 50 54 21 08 00 45 40 61 40 71 40 81 40 81 C0
30: 81 00 81 80 95 00 00 00 00 FE 00 56 33 5F 30 00
40: 00 00 00 00 00 00 00 00 01 1D 00 BC 52 D0 1E 20
50: B8 28 55 40 A0 5A 00 00 00 1E 00 00 00 FD 00 18
60: 3C 1A 51 11 00 0A 20 20 20 20 20 20 00 00 00 FC
70: 00 50 43 4F 52 44 45 52 0A 5F 36 0A 20 20 01 19
80: 02 03 22 F6 4A 93 84 05 14 03 12 A0 A1 90 9F 23
90: 09 07 07 83 01 00 00 67 03 0C 00 10 00 80 22 E2
A0: 00 2A 01 1D 80 D0 72 1C 16 20 10 2C 25 80 A0 5A
B0: 00 00 00 9E 01 1D 80 18 71 1C 16 20 58 2C 25 00
C0: A0 5A 00 00 00 9E 01 1D 00 72 51 D0 1E 20 6E 28
D0: 55 00 A0 5A 00 00 00 1E 8C 0A D0 90 20 40 31 20
E0: 0C 40 55 00 A0 5A 00 00 00 1E 8C 0A D0 8A 20 E0
F0: 2D 10 10 3E 96 00 A0 5A 00 00 00 1E 00 00 00 12
```

r 0x37

Print out EDID for video input 2.

```
>r 0x37
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00: 00 FF FF FF FF FF FF 00 41 A5 01 00 00 00 00 00
10: 1A 17 01 03 80 10 09 78 0A EE 91 A3 54 4C 99 26
20: 0F 50 54 21 08 00 45 40 61 40 71 40 81 40 81 C0
30: 81 00 81 80 95 00 00 00 00 FE 00 56 33 5F 30 00
40: 00 00 00 00 00 00 00 00 01 1D 00 BC 52 D0 1E 20
50: B8 28 55 40 A0 5A 00 00 00 1E 00 00 00 FD 00 18
60: 3C 1A 51 11 00 0A 20 20 20 20 20 20 00 00 00 FC
70: 00 50 43 4F 52 44 45 52 0A 5F 36 0A 20 20 01 19
80: 02 03 22 F6 4A 93 84 05 14 03 12 A0 A1 90 9F 23
90: 09 07 07 83 01 00 00 67 03 0C 00 10 00 80 22 E2
A0: 00 2A 01 1D 80 D0 72 1C 16 20 10 2C 25 80 A0 5A
B0: 00 00 00 9E 01 1D 80 18 71 1C 16 20 58 2C 25 00
C0: A0 5A 00 00 00 9E 01 1D 00 72 51 D0 1E 20 6E 28
D0: 55 00 A0 5A 00 00 00 1E 8C 0A D0 90 20 40 31 20
E0: 0C 40 55 00 A0 5A 00 00 00 1E 8C 0A D0 8A 20 E0
F0: 2D 10 10 3E 96 00 A0 5A 00 00 00 1E 00 00 00 12
```

audio <off/on>

Enable and disable audio. When audio is on, the green LED on the back of the V100 will light up.

rcp

Print color palette. It is printed in 16+1 lines with 16 entries in each line. The first 16 lines are the 256 color palette entries from 0 to 255 (decimal) or 00 to FF (hexadecimal). The last line is reserved for future use. Please ignore. The first entry (color index 0) is marked red in the printout below.

```
>rcp
RED color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 99 33 99 66 66 CC 99 66 FF CC 00 E4 74 89
 10: 9D A6 B1 B8 C0 C4 D4 EB 7A 89 90 DD E1 DA F5 B1
 20: F0 F7 F2 EC E6 CD D8 FA FD E6 ED 1D 24 22 32 2C
 30: 2B 3B 37 4E 43 40 5C 5A F6 80 FE 2C 2B E1 D5 F7
 40: D9 91 ED DC DB C1 70 96 95 8D 89 88 85 A7 A4 A2
 50: 99 98 6D 68 7B 79 76 74 72 6F 6E 8A 86 80 7C 9A
 60: 6F 6D 6B 69 67 81 EE FA C8 F7 D2 C2 B2 FD FB F9
 70: F5 F3 E3 D9 D1 CB C4 FE DF FD EB F0 FA F8 FB FB
 80: FC FB F6 FE FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: B3 57 CA 77 55 DB 5F 3F 63 2D 8B F6 10 3C B0 00
GREEN color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 33 33 99 66 00 99 66 33 CC 66 00 D7 15 2A
 10: 56 67 74 83 93 9F B6 E1 23 37 46 C4 D0 CA ED 89
 20: E3 F0 EB E7 DD BD CF F7 FB E1 EA 1C 23 21 31 2B
 30: 2A 3A 36 4D 42 3F 5B 59 F5 7F FD 2C 2B E1 D5 F7
 40: D9 91 EE DD DC C2 72 98 97 8F 8B 8A 87 A9 A6 A4
 50: 9B 9A 70 6B 7E 7C 79 77 75 72 71 8D 89 83 7F 9D
 60: 73 71 6F 6D 6B 85 F2 FE CB F9 D4 C4 B4 FE FC FA
 70: F6 F4 E4 DA D2 CC C5 FF E0 FC DE E9 F5 F3 F7 F8
 80: FA F9 F4 FD FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: 34 3A EB CF A8 CF 65 82 DF 61 14 95 B1 59 C7 00
BLUE color palette
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
 00: FF CC 33 33 99 66 00 99 66 33 CC 66 00 D8 1E 33
 10: 5E 6E 7A 89 98 A3 B9 E2 2E 41 50 C7 D2 CC EE 8F
 20: E5 F1 EC E8 DF C1 D2 F8 FC E4 EC 1D 24 22 32 2C
 30: 2B 3B 37 4E 43 40 5C 5A F6 80 FF 2D 2C E3 D7 F8
 40: DA 92 EF DE DD C3 73 99 98 90 8C 8B 88 AA A7 A5
 50: 9C 9B 71 6C 7F 7D 7A 78 76 73 72 8E 8A 84 80 9E
 60: 74 72 70 6E 6C 86 F3 FE CB F9 D4 C4 B4 FE FC FA
 70: F6 F4 E4 DA D2 CC C5 FE DF FB DE E9 F5 F3 F7 F8
 80: FA F9 F4 FD FE FD FC FB F8 F6 F5 F4 F3 F1 EE ED
 90: EC E9 E8 E7 E6 E4 E3 E2 E0 DF DE DD DC DB D9 D8
 A0: D7 D6 D4 D3 D1 D0 CE CC CB C8 C7 C5 C2 C1 BF BD
 B0: B9 B7 B5 B4 B3 B2 B0 AE AB AA A7 A6 A5 A4 A2 9F
 C0: 9D 9B 98 97 94 8F 8C 8B 89 86 85 84 82 7E 7C 7B
 D0: 79 76 73 71 70 6F 6E 6B 68 66 65 64 60 5F 5D 5B
 E0: 58 57 56 55 54 52 51 4E 4B 4A 48 47 45 44 28 26
 F0: 00 49 47 44 43 40 3D 3A 38 35 2F 2B 28 25 22 00
100: B0 DF E2 97 46 6E F3 8B 50 1E C1 AF F6 E8 64 00
```

h prints manual (future command)

Backup and restore

The entire Flash memory may be backed up and restored. For this feature the V100 is connected via the extension header to a Raspberry Pi. Jumper wires from the extension header adapter to the RPi connect the UART (serial console) and the SPI bus. The UART connection is used for issuing commands. The SPI interface facilitates a fast data transfer (up to 10 MByte/s) between the V100 and the RPi.

First the Raspberry Pi must be configured so that the minicom application can communicate with the V100. The Raspberry Pi will not send commands automatically to facilitate login:

```
$ sudo systemctl stop serial-getty@ttyAMA0.service    - Raspberry Pi 2
$ sudo systemctl stop serial-getty@ttyS0.service      - Raspberry Pi 3
```

Two Python scripts are provided to automate the backup and restore.

```
$ sudo python backup.py                  - backup all configuration sets
$ sudo python restore.py                 - restore the configuration sets
```

Each configuration set is stored in one file in the config directory. Each file is named „conf0“ to „conf20“ and is 2,883,584 Byte (2816 kByte) in size.

To run the Python script the UART port must be available for access by the Python script. Please terminate any console session ahead of time.

backup.py

```
#!/usr/bin/env python

# This file is part of Project 38189
# Copyright (C) 2016 Auvidea GmbH
#
# Execute backup.py to prepare new configuration files.
# This will read all configuration sets in the Flash of the V100.
# They are saved on the RPi.
# Compatible with V100 firmware version 0.7.7 and better

import serial
import spidev

i = ii = j = rdtgo = rbusy = rempty = data = 0
SOH  = '\x01'
ETX  = '\x03'
EOT  = '\x04'
ACK  = '\x06'
NACK = '\x15'

def rdram():
    global j
    wad = 0x6A00000
    tx = [0x4f] * 1025
    tx = list(tx)
    f = open('config/conf'+str(j),'w')
    j = j+1
    spi = spidev.SpiDev()
    spi.open(0,0)
```

```
    spi.max_speed_hz = 10000000
    for i in range(0,2816):
        wad1 = wad&0xff
        wad2 = (wad>>8)&0xff
        wad3 = (wad>>16)&0xff
        wad4 = (wad>>24)&0xff
        spi.xfer([0x80, 0x00])
        spi.xfer([0xC7, wad1, 0xC8, wad2, 0xC9, wad3, 0xCA, wad4])
        rdt = spi.xfer([0x50, 0xFF])
        rdtgo = (rdt[1]|0x01)
        spi.xfer([0xD0, rdtgo])
        while True:
            rbusy = spi.xfer([0x50, 0xFF])
            if (rbusy[1]&0x08) != 1:
                break
        spi.xfer([0xD0, (rbusy[1]&0xFE)])
        spi.xfer([0x80, 0x40])
        data = spi.xfer2(tx)
        del data[0]
        newFileByteArray = bytearray(data)
        f.write(newFileByteArray)
        wad = wad + 1024
    spi.xfer([0x80, 0x00, 0xC7, 0x00, 0xC8, 0x00, 0xC9, 0x00, 0xCA, 0x00])
    f.close()

port = serial.Serial("/dev/ttyS0", baudrate=115200, timeout=3.0)
port.write('\x0e')
port.readline()
port.write("rdc\r")
rx = port.read()
while rx != EOT:
    if rx==ETX:
        print "ACK"
        rdram()
        port.write(ACK)
        rx = port.read()
    else:
        print "CANCEL"
        port.write('\x19')
        quit()
print "EOT"
port.write('\x19')
quit()
```

Please edit the UART device to be used in the Python scripts. UART device with various Raspberry Pi models:

- Raspberry Pi 2: /dev/ttyAMA0
- Raspberry Pi 3: /dev/ttyS0

restore.py

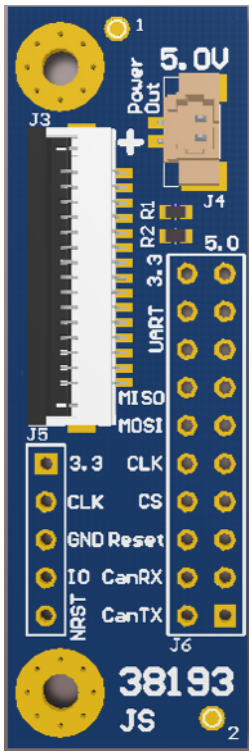
```
#!/usr/bin/env python

# This file is part of Project 38189
# Copyright (C) 2016 Auvidea GmbH
#
# execute restore.py to restore the backup configuration sets
# it reads the cons files and writes to data into the Flash of the V100
```

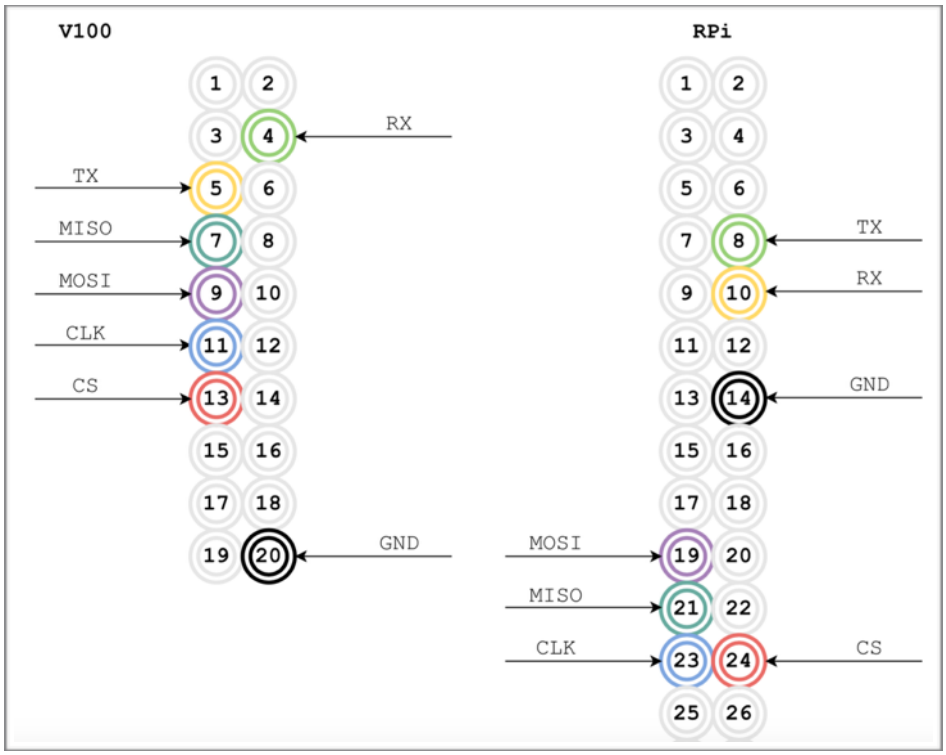
```
# Compatible with V100 firmware version 0.7.7 or better

import serial
import spidev

i = ii = j = rdtgo = rbusy = rempty = data = 0
SOH  = '\x01'
ETX  = '\x03'
EOT  = '\x04'
ACK  = '\x06'
NACK = '\x15'

def wrram():
    global j
    f = open('config/conf'+str(j),'rb')
    j = j+1
    spi = spidev.SpiDev()
    spi.open(0,0)
    spi.max_speed_hz = 10000000
    spi.xfer([0x80, 0x00])
    spi.xfer([0xCB, 0x00, 0xCC, 0x00, 0xCD, 0xA0, 0xCE, 0x06])
    rdt = spi.xfer([0x50, 0xFF])
    rdtgo = (rdt[1]|0x01)
    spi.xfer([0xD0, rdtgo])
    while True:
        rbusy = spi.xfer([0x50, 0xFF])
        if (rbusy[1]&0x08) != 1:
            break
    spi.xfer([0xD0, (rbusy[1]&0xFE)])
    spi.xfer([0x80, 0x40])
    for i in range(0,2816):
        tx = list(f.read(1024))
        tx = [ord(n) for n in tx]
        tx.insert(0,0xCF)
        spi.xfer(tx)
    spi.xfer([0x80, 0x00, 0xCB, 0x00, 0xCC, 0x00, 0xCD, 0x00, 0xCE, 0x00])
    f.close()

port = serial.Serial("/dev/ttyS0", baudrate=115200, timeout=3.0)
port.write('\x0e')
port.readline()
port.write("wdc\r")
rx = port.read()
while rx != EOT:
    if rx==ETX:
        wrram()
        print "ACK"
        port.write(ACK)
        rx = port.read()
    else:
        print "CANCEL"
        port.write('\x19')
        quit()
print "EOT"
port.write('\x19')
quit()
```

RPi to V100 cable

7 jumper cables (J6 of the 38193) may connect the RPi GPIO header and the V100. 3 cables are for the UART (RX, TX, and GND) and 4 cables are for SPI (MISO, MOSI, CLK, and CS). The SPI connection is just required for backup and restore. It is used as the data rate on the SPI bus (20 Mbit/s) is 200x higher than on the UART (115200 bit/s) connection. On the RPi pin 1 is marked by a square pad on the bottom side. If you have any questions or doubt please search the web for „raspberry GPIO“.

In the future the new 38192 adapter will connect to the 38193 via an FFC cable (J3).





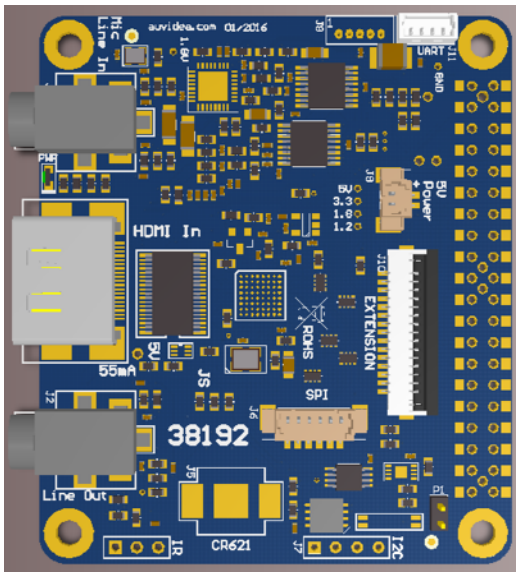
38192 interface module for RPi

This new interface module is designed for the Raspberry Pi. I integrates:

- B101
- analog audio in/out and audio mixer
- extension connector to V100

The V100 is an ideal companion product to the B101 and RPi as it can convert any video input to the 1080p25 resolution supported by the B101.

Status: first samples Q4’2016



UART to USB cable

UART connection

The V100 and the host computer may be connected via a UART cable, which is plugged into the UART jack on the front of the V100. This requires a 3.5mm to USB cable (3.3V TTL UART). We recommend the DeLOCK 83114 Converter USB 2.0 male > Serial-TTL 3.5mm stereo jack 1.8m (3.3V) cable (www.delock.com). Auvidea can provide this cable together with the V100.

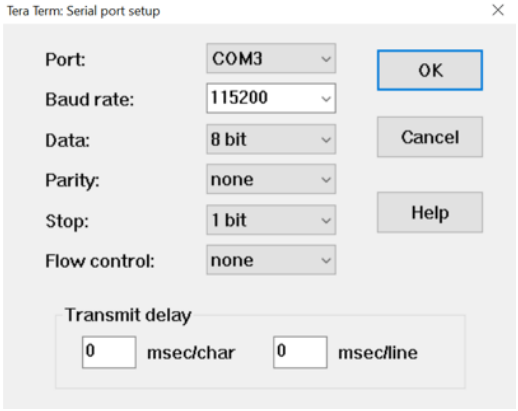
- 3.5mm jack
- 1. GND, 2. RXD, 3. TXD
- chip set: FTDI 232RL
- 1.8 m length
- 115200 baud, 8N1
- driver: http://www.delock.com/produkte/G_83114/merkmale.html

Tera Term (PC)

We have tested this terminal application (Tera Term 4.92) on Windows 7 and 10. Please use the DeLOCK USB to UART cable to connect to the V100.

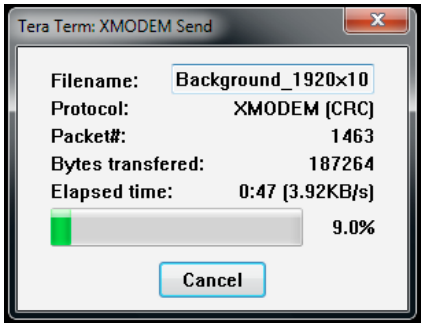
Attention

The V100 has one UART interface but two connectors. One is the 3.5mm jack in the front, the other is the pin header on the 38193 interface board. Please only use one at a time. If you use the USB cable, then the pin header must not be connected!



1. please install the FTDI 232RL driver. Windows 10 can do this automatically. Please check the device manager that this hardware was installed successfully.
2. The FTDI 232RL driver creates a virtual COM port. The COM number may vary from system to system. Please go to setup - serial port and configure COM port number and baud rate (115200).
3. now you should be able to communicate with the V100. Test the „v“ command.

```
v                                   - print the version
```



4. to download an OSD file, please use the XMODEM protocol which is supported by the V100 and Tera Term. Please see tutorial 1. First issue the „xmodem“ command to the V100. It will respond by sending a sequence of „C“ characters. Then start the XMODEM transfer on the Tera Term by selecting the menu item: „file - transfer - XMODEM - send..“. Then select the file to be transmitted. The V100 can receive BMP files with indexed colors (1, 2 4 or 8 bit) only. JPEG, PNG, TIFF or PSD files are not supported. The BMP file must a maximum of 255 colors, as by default the 256th color is reserved for transparency.
5. please note that the serial connection only transfers up to 115200 bits/s. As the interface requires approx. 11 bits for each byte transferred, the theoretical data rate is about 10kByte/s. Tera Term typically transfers 4kByte/s. A typical full screen OSD image is approx. 2 MByte in size. So it takes approx. 500 seconds (8 minutes) to transfer.
6. for interactive download of OSD image updates we recommend to use the internal SPI connection to the V100 as this is approx. 200x faster. Our backup and restore scripts transfer complete configuration sets (2.8 MByte each) in just a few seconds.

minicom (Raspberry Pi)

We have tested this terminal application (minicom) on the Raspberry Pi. Please use the DeLOCK USB to UART cable to connect to the V100 or connect the Pi to the pin header on the 38193 interface module on the V100.

Attention

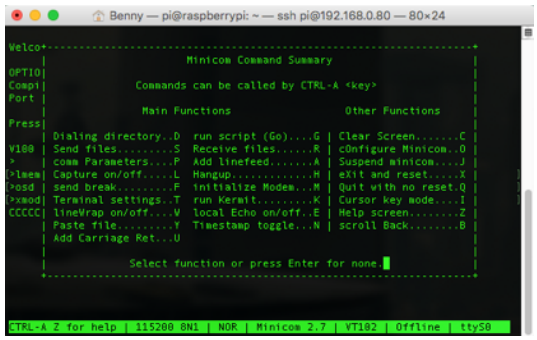
The V100 has one UART interface but two connectors. One is the 3.5mm jack in the front, the other is the pin header on the 38193. Please use only one for the connection. I you use both at the same time, hardware could get damaged.

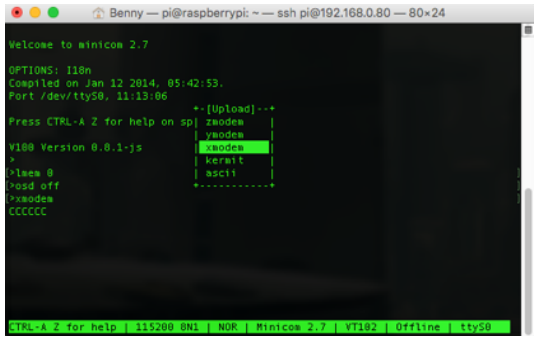
XMODEM

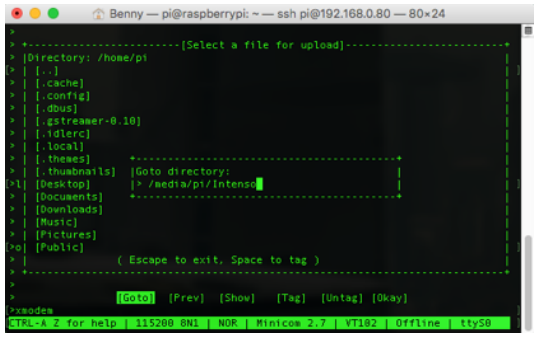
A BMP file may be uploaded with the minicom application on the Raspberry Pi. First enter „xmodem“ to put the V100 into receive mode. It will confirm this by putting out a series of „C“ characters

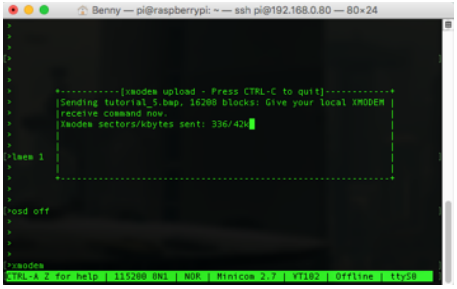
Press <ctrl>A and then „z“. This opens a dialog box. Press „s“ to select send a file. This opens the upload dialog. Please select „xmodem“. Please move to „Goto“ with the arrow keys and enter the path to the image directory. Please select one file to be transferred with the space bar.

The transfer will now start. The xmodem Upload dialog box will show the progress. When the transfer is completed the V100 will confirm the size of the BMP file uploaded- here 256x116 pixels.









Raspberry Pi video out

For testing purposes the Raspberry Pi may be used as a programmable video output generator. It supports 59 video modes (CEA) and 86 computer modes (DMT). Therefore the output resolution may be very flexibly adjusted. Up to date information may be retrieved from:

https://www.raspberrypi.org/documentation/configuration/config-txt.md

1. log into the Raspberry Pi with mouse, keyboard and monitor or with ssh remote access.

```
ssh pi@<ip address>                    - connect via ssh
password: raspberry
```

2. edit config.txt to modify the HDMI output settings

```
sudo nano /boot/config.txt             - edit config file
```

3. you may want to disable the EDID, to force the Pi into a specific output mode and therefore ignoring any EDID it may get from the V100 or another system it is connected to. Search for the line and edit it.

```
hdmi_ignore_edid=0xa5000080            - ignore EDID
```

4. next select the hdmi_group 1, to specify CEA output modes

```
hdmi_group=1
```

5. on the web site you find the complete list of CEA codes. Also they are listed in the timing table on page 6 of this document. Examples:

```
hdmi_mode=16                           - 1080p60
hdmi_mode=31                           - 1080p50
hdmi_mode=32                           - 1080p24
hdmi_mode=33                           - 1080p25
hdmi_mode=34                           - 1080p30
```

6. now save and close the config.txt file

```
<ctrl> o                               - save the file
<ctrl> x                               - close the file and editor
<ctrl> c                               - close without saving
```

7. reboot the Raspberry Pi so that the changes can take effect

8. the Raspberry may also be used to read the EDID of the V100, if the HDMI out of the Raspberry Pi is connected to one of the V100 HDMI inputs. Please make sure that the EDID is enabled in the config.txt file.

```
#hdmi_ignore_edid=0xa5000080           - edit the config.txt file, to read EDID
```

```
tvservice -d test.edid                 - save the EDID of the V100 to a file
hexdump test.edid                      - dump the EDID hex file
```

9.

DEBUGGING

This commands are mainly intended for internal use and debugging purposes.

`i`	probes I²C devices
`b <from> <to>`	prints register bank values of the HD video processor chip

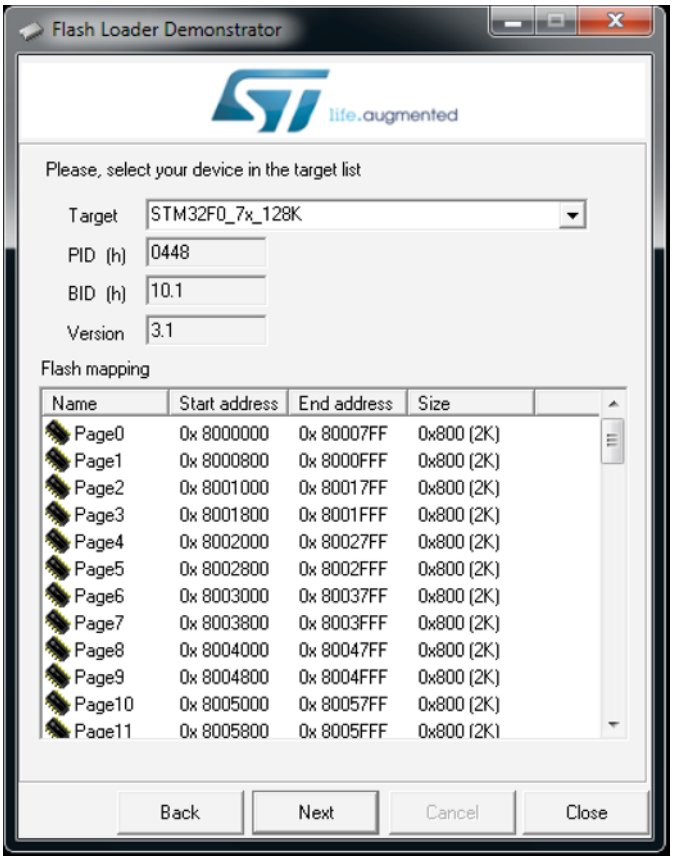
Parameters

from	*0 .. 28*
to	*0 .. 28*

Firmware upgrade

The firmware of the V100 may be upgraded via the UART connection. A UART connection to a Windows PC is required as the STM flash software only supports Windows at this time.

1. download and install the STM32 Flash loader demonstrator (Windows only) from http://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/flasher-stm32.html
2. connect V100 via USB-to-UART cable to your Windows PC
3. start terminal emulation software and connect with your V100 (FT232R USB UART) (USB Serial COM Port)
4. check firmware version with the API command „v“
   example: HD Video Processor Version 0.7.1
5. start bootloader with the API command „bootloader"
6. terminate terminal emulation program
7. start STM Flash Loader Demonstrator and program the micro controller
   - select the firmware bin file
   - select the target device: STM32F072
   - flash the device
8. restart V100
9. verify that new firmware has been programmed with the API command „v“



FAQ

1.

Disclaimer

Thank you for reading this manual. If you have found any typos or errors in this document or any bugs or issues in the software or API, please let us know.

The Auvidea Team