

Avenger96 Camera Interface User Manual

STM32MP1 Avenger96 Reference Design

Date: January 20, 2022 | Version 1.0



The Solutions People

Five Years Out

CONTENTS

1	INTRODUCTION	4
1.1	Purpose of the Document.....	4
1.2	About the System	4
1.3	Intended Audience.....	5
1.4	Prerequisites	5
2	ENVIRONMENT SETUP	6
2.1	Steps to build Yocto Image using build script.	6
2.2	Download firmware package	8
2.3	Flash the firmware image to SD Card in LINUX HOST PC	8
2.4	Flash the firmware image to SD Card in Windows HOST PC	9
2.5	Hardware Installation.....	10
3	CAMERA DEMO	11
3.1	Live stream from camera on HDMI display.....	11
3.2	Capture image from camera	11
3.3	Validation of ISP controlled feature of AP1302:.....	13
3.3.1	Resolution, FPS and Data format	13
3.3.2	Brightness	14
3.3.3	Contrast	14
3.3.4	Saturation	14
3.3.5	Gamma	14
3.3.6	Exposure	14
3.3.7	Gain	15
3.3.8	White Balance.....	15
4	LIMITATION	16
5	REFERENCES	17

FIGURES

Figure 1:	STM32MP1 Avenger96 board	4
Figure 2:	Win32 Disk Imager for flashing SD Card	9
Figure 3 :	Avenger96 AP1302 setup	10
Figure 4:	Live stream in HDMI Display	11
Figure 5:	Image captured using AP1302.....	12

ACRONYMS AND ABBREVIATIONS

Definition/Acronym/Abbreviation	Description
cd	Change directory
scp	Secure copy over the network
BSP	Board Support Package
USB	Universal Serial Bus
HW	Hardware
FW	Firmware
NV RAM	Non-Volatile Random-Access Memory
API	Application Programming Interface
V4L2	Video for Linux second version
ISP	Image signal processor
SD	Secure Digital
HDMI	High-Definition multimedia interface
LTS	Long Term Support
UART	universal asynchronous receiver-transmitter
PC	Personal computer
FAT	File Allocation Table
FPS	Frames per second

1 INTRODUCTION

1.1 Purpose of the Document

Purpose of this document is to help developers flash firmware and demonstrate camera interface on STM32MP1-Avenger96 firmware. For demo, we have used AP1302 ISP with either one of the following camera sensors:

- SRT-Vision96-AR0430 mezzanine (AP-vision-AR0430)
- SRT-Vision96-ARX3A0 mezzanine (AP-vision-ARX3A0)
- SRT-Vision96-AR1335 mezzanine (AP-vision-AR1335)

1.2 About the System

This system is based on STM32MP1 processor and supporting multiple interfaces. This can facilitate for Human-Machine Interface experience.

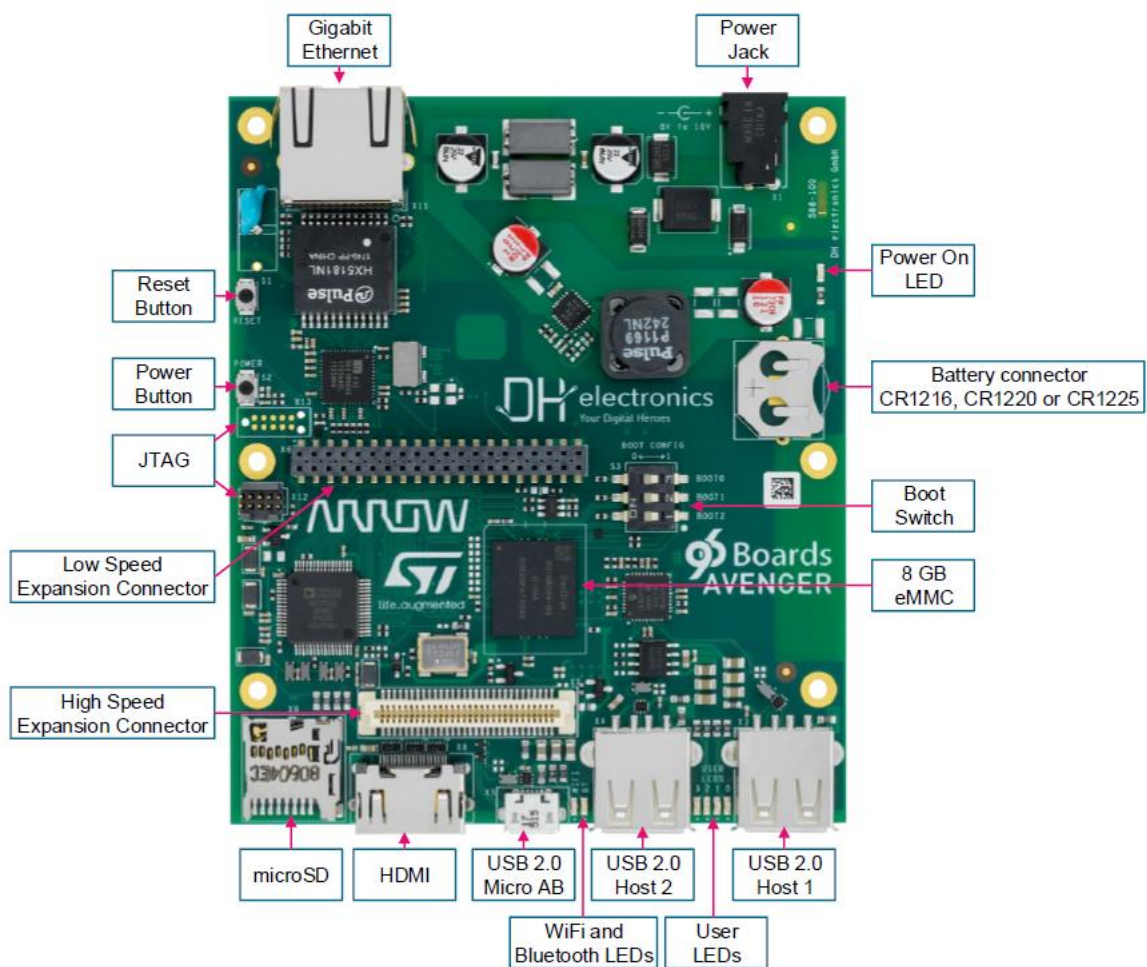


Figure 1: STM32MP1 Avenger96 board

1.3 Intended Audience

This document is for developers and end-users who want to understand/flash/demonstrate camera interface on STM32MP1-Avenger96 firmware.

1.4 Prerequisites

Below are the list of hardware and software needed to demonstrate the Camera interface on STM32MP1-Avenger96 Board:

- x86 host system having Linux Ubuntu 16.04 or 18.04 LTS installed (to build Yocto image)
- Linux PC (Minicom for serial console, Optimal 16GB-RAM, Processor:Octa-core)
- Install Essential Yocto Project host packages
- Basic understanding of Linux commands
- Setup will require following:
 - Avenger96 Board
 - SD-card -32GB
 - UART debug cable - [UART Serial - 96Boards](#)
 - Power Supply - [Power Accessories - 96Boards](#)
 - One of the following mezzanine cards
 - SRT-VISION96-AR1335
 - SRT-VISION96-AR0430
 - SRT-VISION96-ARX3A0
 - HDMI Display
 - USB Keyboard
 - USB Mouse

2 ENVIRONMENT SETUP

2.1 Steps to build Yocto Image using build script.

A release package [Avenger96](#) is available which contains build script, prebuilt-image and all the packages, BSP changes and the required patches for Avenger96 firmware. User need to download release package first to build image for Avenger96.

To build Avenger96 firmware on LINUX HOST PC, user will follow below steps:

- Download new repo for Avenger96 board based on kernel version Linux dh-stm32mp1-dhcor-avenger96 5.10.74 release [1]

```
$ sudo apt-get install repo
$ git config --global user.name "Your Name"
$ git config --global user.email "Your Email"
$ git config --global user.email "Your Email"
$ git config --list
```

- Download or clone release [Avenger96](#), it contains as shown below:

```
Avenger96/Kernel_5_10_74
├── Avenger96_L5_10_74_Rel_1_0_patches
├── readme.md
├── Software_Docs
│   ├── ei_Camera_User_Guide_STM32MP1_Avenger96_L5_10_Rel_1_0.pdf
│   ├── ei_OnSemiCamModule_96B_AV96_ReleaseNote.pdf
│   └── ei_OnSemiCamModule_96B_AV96_TestCases.xlsx
├── Yocto_build_manual_steps_avenger96.txt
└── yocto_build_setup_Avenger96.sh
```

- Run the build script to setup the Yocto environment on the LINUX based Host PC

```
$ cd Avenger96/Kernel_5_10_74/
$ sudo chmod 755 yocto_build_setup_Avenger96.sh
$ . yocto_build_setup_Avenger96.sh
```



Note: [For building firmware image, it will take ~10 hours to download all packages and build, the time may vary based on your HOST PC configurations]

- After successful build the final SD Card image is available at below location:
<root>/ dhcom-yocto-bsp/build/tmp/deploy/images/dh-stm32mp1-dhcor-avenger96/
- Filename should be dh-image-demo-dh-stm32mp1-dhcor-avenger96.wic.xz which is soft link of original build image file **dh-image-demo-dh-stm32mp1-dhcor-avenger96-<TIMESTAMP>.rootfs.wic.xz**
- If user want to clean a previous build image and want to run it again then first clean it with command "cleanall" or "cleansstate"

```
$: bitbake dh-image-demo -c cleanall
$: bitbake -v dh-image-demo (If user want to turn on verbose)
```

- If user want to clean any package, then that can be done with command “cleanall” or “cleansstate”

```
$: bitbake <PACKAGE_NAME> -c cleanall
$: bitbake <PACKAGE_NAME>
e.g.
$: bitbake linux-stm32mp1-dhsom -c cleanall
$: bitbake linux-stm32mp1-dhsom (Build linux kernel only)

Same way
$: bitbake u-boot-stm32mp1-dhsom -c cleanall
$: bitbake u-boot-stm32mp1-dhsom (Build uboot code only)

$: bitbake opencv -c cleanall
$: bitbake opencv (Build opencv package)

$: bitbake python3-django -c cleanall
$: bitbake python3-django (Build django python package for python3)
```

2.2 Download firmware package

- Download the provided SD Card (wic.xz) image on Linux PC (host system)
- Open terminal in Host PC from left desktop panel or using keyboard shortcut (ctrl + t)
- From command terminal traverse to the location where firmware has been downloaded using cd command
- use ls command to verify the image downloaded
- Verify md5 check sum of downloaded image with given md5sum
- Extract the provided. xz image using unxz command, which will take couple of minutes.
- Once done, will end with. wic image in the same directory and can again be verified using ls -l command. Steps are as following:

```
$ cd /home/user/download/stm32images/
$ ls -l
$ md5sum <image name>.wic.xz
$ unxz -c <image>. wic.xz
$ ls -l
```

2.3 Flash the firmware image to SD Card in LINUX HOST PC

- Plug in micro SD card into x86 Linux Host PC
- Verify the node created for SD card inside /dev directory
ls -l /dev/sd*
- Open terminal and traverse to the location where downloaded firmware image is stored using cd command
- Ensure the extracted firmware image's file format is .wic using ls -l command
- Use below command for flashing, if the SD card's entry in Linux is /dev/sdX

```
$ sudo dd of=/dev/sdX bs=1M iflag=fullblock oflag=direct conv=fsync status=progress; sync;
```

- Above command will take couple of minutes or more (depending upon PC config) to flash onto the SD card
- Once done remove and insert the SD card again, two drives will get mounted if the above command is successful, named <boot> and <rootfs>
- Eject (safely remove) SD card from host PC and plug it into board's SD card slot

2.4 Flash the firmware image to SD Card in Windows HOST PC

- Plug in micro-SD card into x86 Windows Host PC
- Install win32 Disk Imager (<https://sourceforge.net/projects/win32diskimager/>)
- Format SD card with FAT file system.
- Plug SD card with card reader. It will show drive like "E:"
- Download appropriate production image *.wic.xz
- Extract *.wic.xz image using WinZip or 7-zip. It will create *.wic image.
- Run Win32 Disk Imager
- Select .wic image file and target drive i.e. E: for input Image File. (see below figure)

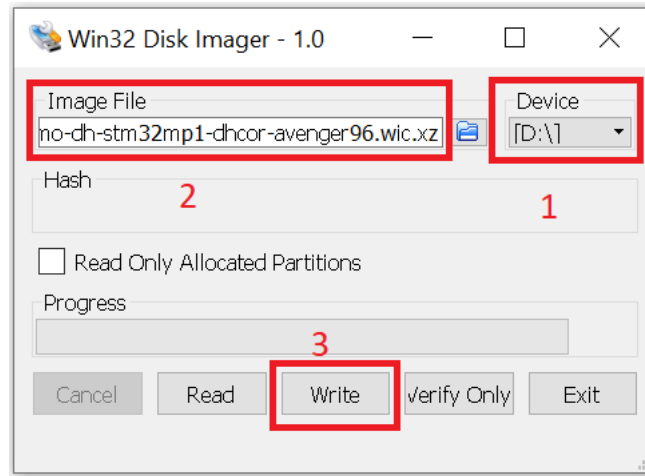


Figure 2: Win32 Disk Imager for flashing SD Card

- Click on Write
- After successful transfer, success message will pop up and there will be around 63.8 MB FAT partition. Inside this partition, one will find different DTB files and Image

2.5 Hardware Installation

- Place hardware board on a clean anti-static surface
- Insert flashed SD card to X9 SD card slot
- Attach HDMI display to X8 HDMI Connector
- Attach Ethernet cable to board's X11 Ethernet connector
- Attach USB Mouse and Keyboard to X4 and X3 USB Connectors
- Provide 12V-2A power supply (provided with board) to board on X1 Power Jack connector.
- Top view of the board after all the connections is shown in Figure 3



Figure 3 : Avenger96 AP1302 setup

3 CAMERA DEMO

3.1 Live stream from camera on HDMI display

- To watch live stream over the HDMI, connect HDMI Display, USB keyboard and Mouse.
- Attach any one of the above mentioned-camera-mezzanine modules, to high-speed expansion connector as shown in Figure <figure to be attached>
- Make sure the dtb environment is set with stm32mp157a-dhcor-avenger96.dtb
- Power up the board
- From HDMI display, open command prompt of the target
- Then apply below command in the command prompt of the target.

```
$ gst-launch-1.0 v4l2src device=/dev/video0 ! videoconvert ! videoscale ! autovideosink
```

- And user will be able to see following camera live stream on HDMI display:

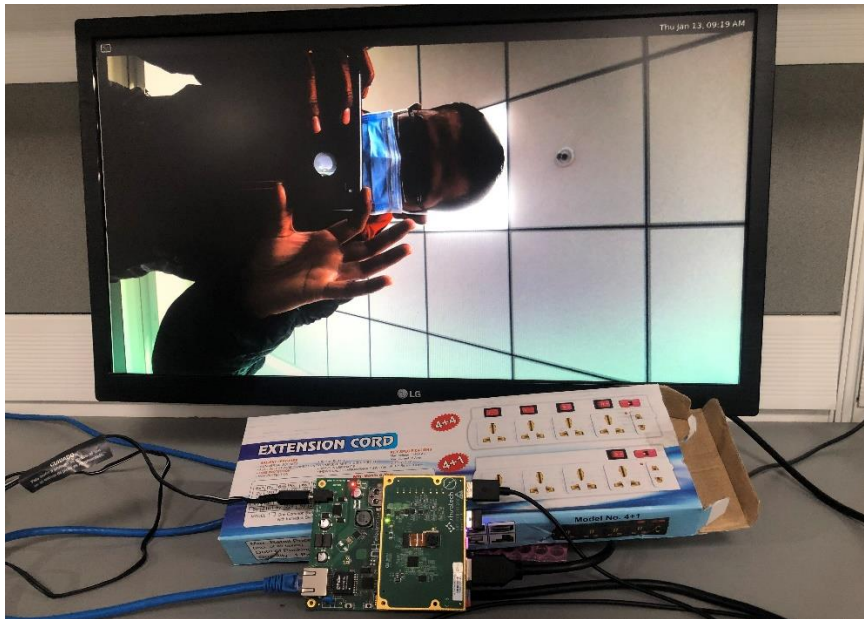


Figure 4: Live stream in HDMI Display

3.2 Capture image from camera

- To capture image from the camera, connect HDMI Display, USB keyboard and Mouse.
- Attach any one of the above mentioned-camera-mezzanine modules, to high-speed expansion connector as shown in Figure <figure to be attached>
- Make sure the dtb environment is set with stm32mp157a-dhcor-avenger96.dtb
- Ensure Ethernet is plugged-in to get image from board to local x86 host pc
- Power up the board
- From HDMI display, open command prompt of the target
- Then apply below command in the command prompt of the target.

```
$ gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=1 ! jpegenc ! filesink location=test0.jpg
```

- Above command will capture image named test0.jpg in /home/root/ location
- Copy image from board to local PC using below command

```
$ scp test0.jpg <user name of host pc >@<ip of host pc>:/home/user/Desktop
```

- Go to local PC's /home/user/Desktop and watch image into image viewer to verify captured image from board's camera

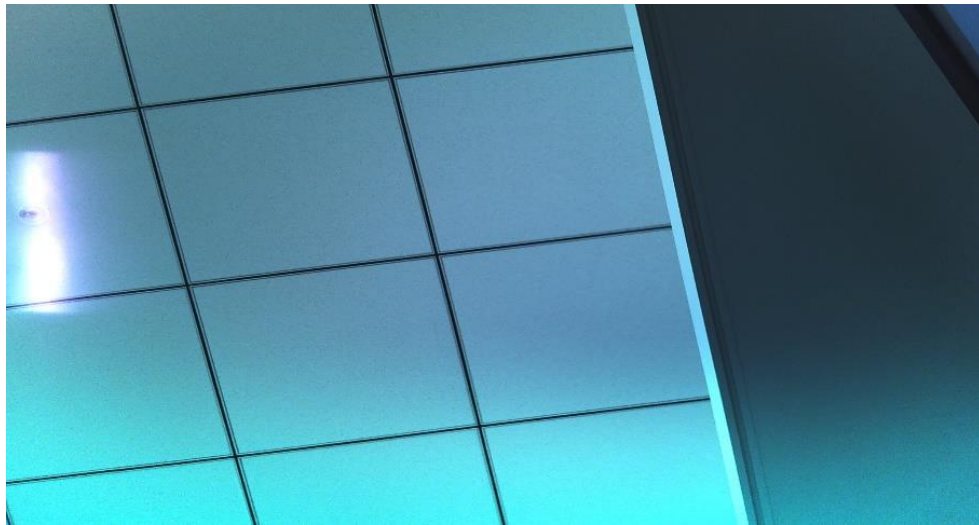


Figure 5: Image captured using AP1302

3.3 Validation of ISP controlled feature of AP1302:

User can check the available v4l2 controls related to the camera from the user space using the below command:

```
root@dh-stm32mp1-dhcor-avenger96:~# v4l2-ctl -L

User Controls

    brightness 0x00980900 (int) : min=256 max=65535 step=256 default=256 value=256
    contrast 0x00980901 (int) : min=256 max=65535 step=256 default=256 value=256
    saturation 0x00980902 (int) : min=256 max=65535 step=256 default=4096 value=4096
    gamma 0x00980910 (int) : min=256 max=65535 step=256 default=4096 value=4096
    exposure 0x00980911 (int) : min=0 max=12 step=1 default=12 value=12
    gain 0x00980913 (int) : min=256 max=65535 step=256 default=256 value=256

Camera Controls

    white_balance_auto_preset 0x009a0914 (menu) : min=0 max=9 default=1 value=1
        0: Manual
        1: Auto

Image Processing Controls

    link_frequency 0x009f0901 (intmenu): min=0 max=0 default=0 value=0
        0: 160000000 (0x9896800)
```

User will be able to get and set the values for each of the controls using following format.

- To get control value, Run the following command:

```
$ v4l2-ctl --get-ctrl <control_name>
```

- To set control value, Run the following command:

```
$ v4l2-ctl --set-ctrl <control_name> =<control_value>
```

3.3.1 Resolution, FPS and Data format

User can check the current resolution, FPS and data format of AP1302 using the following command.

```
$ v4l2-ctl --list-formats-ext -d /dev/video0
```

```
ioctl: VIDIOC_ENUM_FMT
Type: Video Capture
```

```
[0]: 'UYVY' (UYVY 4:2:2)
    Size: Discrete 2316x1746
    Interval: Discrete 0.333s (3.000 fps)
```

3.3.2 Brightness

- To get the brightness value, Run the following command:

```
$ v4l2-ctl --get-ctrl brightness
```

- To set the brightness value, Run the following command:

```
$ v4l2-ctl --set-ctrl brightness=4096
```

3.3.3 Contrast

- To get the brightness value use the following command

```
$ v4l2-ctl --get-ctrl contrast
```

- To set the brightness value use the following command

```
$ v4l2-ctl --set-ctrl contrast=4096
```

3.3.4 Saturation

- To get the saturation value use the following command

```
$ v4l2-ctl --get-ctrl saturation
```

- To set the saturation value use the following command

```
$ v4l2-ctl --set-ctrl saturation=4096
```

3.3.5 Gamma

- To get the saturation value use the following command

```
$ v4l2-ctl --get-ctrl gamma
```

- To set the gamma value use the following command

```
$ v4l2-ctl --set-ctrl gamma=4096
```

3.3.6 Exposure

- To get the exposure value use the following command

```
$ v4l2-ctl --get-ctrl exposure
```

- To set the exposure value use the following command

```
$ v4l2-ctl --set-ctrl exposure=1
```

3.3.7 Gain

- To get the gain value use the following command

```
$ v4l2-ctl --get-ctrl gain
```

- To set the gain value use the following command

```
$ v4l2-ctl --set-ctrl gain=4096
```

3.3.8 White Balance

- To get the White balance value use the following command

```
$ v4l2-ctl --get-ctrl white_balance_auto_preset
```

- To set the White balance value use the following command

```
$ v4l2-ctl --set-ctrl white_balance_auto_preset =1
```

4 LIMITATION

The theoretical throughput rate of MIPI CSI-2.1 interface of about 2.5 Gbyte/s is not possible due to the hardware limitations in the STM32MP1. Refer section 3.3 in the reference [2].

5 REFERENCES

[1] https://github.com/dh-electronics/dhcom_stm32mp1-bsp-platform

[2] https://www.st.com/resource/en/application_note/an5470-stm32mp1-series-interfacing-with-a-mipi-csi2-camera-stmicroelectronics.pdf