UserGuide

Doc Ref: UG035 Rev. A

Generation date: Sep 21, 2026

FullScale

Analog Solutions

DB-MIPI-A1 Driver - User Guide

Driver version 1.1

Online documentation :

https://docs.fullscale-labs.com/UG035/v1.1/

Contents

1 References 1

2 Abbreviation table 2

3 Introduction 3

4 Driver Context 4
4.1 How the Driver Appears in Linux 4
4.2 Media Format and Link Assumptions 4

5 Getting Started 5

6 Control Model 6

7 Control Reference 7

8 Using v4l2-ctl 11

9 Using the V4L2 ioctl API 12

10 Revision history 18

11 Disclamer 19

1 References

Driver version: 1.1

Document reference	UG035
Document revision	A
Generation date	21 sept.. 2026

Written by	F. DESCAMPS	08 Apr. 2026
Validated by	-	-
Approved by	K. TOUTAIN	09 Apr. 2026

2 Abbreviation table

API	Application Programming Interface
CSI	Camera Serial Interface
I2C	Inter-Integrated Circuit
ISP	Image Signal Processor
SPI	Serial Peripheral Interface
V4L2	Video4Linux2

3 Introduction

This guide introduces the DB-MIPI-A1 Linux driver and explains how to interact with its custom V4L2 controls from user space.

The driver is intended to be installed from the Debian package provided for the platform (refer to document `UG034` for instructions). Once installed and enabled, it registers a V4L2 sub-device node and exposes board-specific controls for temperature acquisition, test pattern handling, sensor resolution, sensor clock generator setup, shutter management, and specific register access of Proxy, Sensor, Sensor ISP.

This document is focused on user-space usage. It gives the operational context needed to understand where the controls live, how they are expected to be used, and which access patterns are important during bring-up and validation.

4 Driver Context

4.1 How the Driver Appears in Linux

When the package is installed on a Raspberry Pi 4 and the corresponding overlay is enabled, the board appears in the Linux media graph as a V4L2 sub-device. User-space control access therefore happens on a device such as `/dev/v4l-subdev0`.

> **Note**
>
> The custom controls documented here are exposed on the V4L2 sub-device node, not on a generic capture node such as `/dev/video0`.

4.2 Media Format and Link Assumptions

The driver reports the following media-bus format:

- Bus code: `MEDIA_BUS_FMT_RGB888_1X24` (with R = LSB, G = MSB and B = 0)
- Colorspace: `V4L2_COLORSPACE_SRGB`
- Frame width: `1536`
- Frame height: initialized to `0` and updated when the sensor `Y` resolution is configured
- Field: `V4L2_FIELD_NONE`

5 Getting Started

A typical deployment and validation sequence on Raspberry Pi 4 is:

1. Install the provided Debian package.
2. Enable the board device tree overlay.
3. Reboot.
4. Confirm that the sub-device node exists.
5. Inspect the media graph and the list of custom controls.
6. Issue control requests on the sub-device node.

Refer to the document `UG034` for instructions about first steps

Useful commands during validation are:

```
ls /dev/v4l-subdev*  # => Get exact subdevice node file (and confirm that the driver
# has been properly loaded)
v4l2-ctl --device=/dev/v4l-subdev0 --list-ctrls -L  # => List available controls
media-ctl -p  # => Confirm that the camera subdevice driver is properly linked to CSI-2 driver
```

6 Control Model

The public API defines several main control styles:

- Read-only volatile controls used to fetch live state and measurements
- Button controls used to trigger one-shot actions
- Integer read or write controls used to read hardware registers values or to set them
- Two-element unsigned arrays used for register writes
- A menu control used to report shutter position

A few practical rules are worth keeping in mind:

- `V4L2_CID_CUSTOM_S_ADCVTEMP_INIT` should be triggered before reading the ADC VTemp control
- I2C proxy controls only makes sense if the connected proxy has i2c connection
- Sensor ISP controls only makes sense if the connected sensor embeds an ISP
- For sensor resolution updates, user should set `X` then `Y` then `apply`
- For sensor clock generator updates user should set the coefficient, then `apply config`, then `output enable`
- The Proxy i2c, Sensor ISP and Sensor read controls use a two-step access pattern: write the register address first, then read the same control back to obtain the value

7 Control Reference

Sensor temperature

`V4L2_CID_CUSTOM_G_COLVTEMP_TEMPERATURE`

Returns the VTemp temperature measurement given by the proxy in the temperature columns.

Read-only unsigned 16-bit control.

`V4L2_CID_CUSTOM_G_ADCVTEMP_TEMPERATURE`

Returns the VTemp temperature measurement given by the board embedded ADC.

Read-only unsigned 16-bit control. This readout is meaningful only after the ADC has been initialized.

`V4L2_CID_CUSTOM_S_ADCVTEMP_INIT`

Initializes the embedded ADC VTemp.

Button control.

Test Pattern Controls

`V4L2_CID_CUSTOM_G_TEST_PATTERN_ACTIVATED`

Returns whether the board test pattern is currently enabled.

Read-only boolean control.

`V4L2_CID_CUSTOM_S_TEST_PATTERN_ON`

Enables the board test pattern generator.

Button control.

`V4L2_CID_CUSTOM_S_TEST_PATTERN_OFF`

Disables the board test pattern generator.

Button control.

Sensor resolution

These controls informs the board of the currently connected sensor’s resolution.

`V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_X`

Sets the sensor active X resolution.

Unsigned 16-bit write control.

`V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_Y`

Sets the sensor active Y resolution.

The driver updates the reported media-bus frame height from this value.

Unsigned 16-bit control.

V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_APPLY

Applies the sensor resolution previously configured through the X and Y controls.

Button control.

Sensor Clock

These controls configure the embedded clock generator so that it produces the desired sensor clock.

V4L2_CID_CUSTOM_G_SENSOR_CLKGEN_CONFIG_DONE

Returns whether the sensor clock generator configuration has been applied.

Read-only boolean control.

V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_M_DIV_COEF

Sets the sensor clock generator M divider coefficient.

Unsigned 8-bit control.

V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_N_DIV_COEF

Sets the sensor clock generator N divider coefficient.

Unsigned 16-bit control.

V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_OUT_DIV_COEF

Sets the sensor clock generator output divider coefficient.

Unsigned 8-bit control.

V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_APPLY_CONFIG

Applies the sensor clock generator configuration.

Button control.

V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_OUTPUT_ENABLE

Enables the sensor clock generator output.

Button control.

Shutter controls

V4L2_CID_CUSTOM_G_SHUTTER_POSITION

Returns the current shutter position.

Read-only menu control. The returned value matches one of the `V4L2_SHUTTER_POSITION_*` constants below.

V4L2_CID_CUSTOM_S_SHUTTER_OPEN

Opens the shutter.

Button control.

V4L2_CID_CUSTOM_S_SHUTTER_CLOSE

Closes the shutter.

Button control.

Spi Proxy arbitrary register write

These controls can be used to write any Proxy register accessible through SPI

V4L2_CID_CUSTOM_S_SPI_PROXY_REGISTER

Writes a Proxy register through the SPI bus.

Two-element unsigned 16-bit control. Element 0 is the SPI proxy register address and element 1 is the value to write.

I2C Proxy arbitrary register access

These controls can be used to read or write any Proxy register accessible through I2C

V4L2_CID_CUSTOM_G_I2C_PROXY_REGISTER

Reads a Proxy register through the I2C bus.

Unsigned 8-bit control. User space first writes the register address to this control, then reads the same control back to obtain the register value.

V4L2_CID_CUSTOM_S_I2C_PROXY_REGISTER

Writes a Proxy register through the I2C bus.

Two-element unsigned 16-bit control. Element 0 is the register address and element 1 is the value to write.

Sensor ISP arbitrary register access

These controls can be used to read or write any Sensor ISP register

V4L2_CID_CUSTOM_G_SENSOR_ISP_REGISTER

Reads a Sensor ISP register through a two-step access pattern.

Unsigned 16-bit control. User space first writes the register address to this control, then reads the same control back to obtain the register value.

V4L2_CID_CUSTOM_S_SENSOR_ISP_REGISTER

Writes a Sensor ISP register.

Two-element unsigned 16-bit control. Element 0 is the register address and element 1 is the value to write.

Sensor arbitrary register access

These controls can be used to read or write any Sensor register

`V4L2_CID_CUSTOM_G_SENSOR_REGISTER`

Reads a sensor register through a two-step access pattern.

Unsigned 8-bit control. User space first writes the register address to this control, then reads the same control back to obtain the register value.

`V4L2_CID_CUSTOM_S_SENSOR_REGISTER`

Writes a sensor register.

Two-element unsigned 16-bit control. Element 0 is the register address and element 1 is the value to write.

Shutter constant values

`V4L2_SHUTTER_POSITION_CLOSED`

Shutter position value for a closed shutter.

`V4L2_SHUTTER_POSITION_OPEN`

Shutter position value for an open shutter.

`V4L2_SHUTTER_POSITION_MOVING`

Shutter position value for a shutter currently moving.

8 Using v4l2-ctl

Control names shown by `v4l2-ctl` are derived from the driver strings and are typically normalized with underscores. Always trust the exact names returned by `--list-ctrls -L` on your target.

Typical examples are:

```
# Query current custom controls
v4l2-ctl --device=/dev/v4l-subdev0 --list-ctrls -L

# Read temperatures
v4l2-ctl --device=/dev/v4l-subdev0 --get-ctrl='g_colvtemp_temperature'
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_adcvtemp_init=1'
v4l2-ctl --device=/dev/v4l-subdev0 --get-ctrl='g_adcvtemp_temperature'

# Enable or disable the board test pattern
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_test_pattern_on=1'
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_test_pattern_off=1'

# Configure and apply sensor resolution
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_sensor_resolution_x=1288'
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_sensor_resolution_y=1024'
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='s_sensor_resolution_apply=1'

# Read shutter state
v4l2-ctl --device=/dev/v4l-subdev0 --get-ctrl='g_shutter_position'

# Select a Sensor ISP register to read, then fetch it
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='g_sensor_isp_register=0x0001'
v4l2-ctl --device=/dev/v4l-subdev0 --get-ctrl='g_sensor_isp_register'

# Select a sensor register to read, then fetch it
v4l2-ctl --device=/dev/v4l-subdev0 --set-ctrl='g_sensor_register=0x0000'
v4l2-ctl --device=/dev/v4l-subdev0 --get-ctrl='g_sensor_register'
```

Warning

Register-read and register-write interactions depend on the exact support level of the `v4l2-ctl` version available on the target, especially for array-valued controls. For robust production use, a dedicated application based on the V4L2 ioctl API is often more convenient than shell commands alone.

9 Using the V4L2 ioctl API

The public header provides the control identifiers that should be used by applications.

The example below illustrates a one-shot button command and a Proxy register write.

```c
#include <fcntl.h>
#include <linux/videodev2.h>
#include <stdint.h>
#include <stdio.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

#include "db-mipi-a1_public.h"

static int trigger_action(int fd, unsigned int id)
{
   struct v4l2_control ctrl;

   memset(&ctrl, 0, sizeof(ctrl));
   ctrl.id = id;
   ctrl.value = 1;
   return ioctl(fd, VIDIOC_S_CTRL, &ctrl);
}

static int write_proxy_register(int fd, uint16_t reg, uint16_t value)
{
   uint16_t payload[2] = {reg, value};
   struct v4l2_ext_control ext_ctrl;
   struct v4l2_ext_controls ext_ctrls;

   memset(&ext_ctrl, 0, sizeof(ext_ctrl));
   memset(&ext_ctrls, 0, sizeof(ext_ctrls));

   ext_ctrl.id = V4L2_CID_CUSTOM_S_SPI_PROXY_REGISTER;
   ext_ctrl.size = sizeof(payload);
   ext_ctrl.p_u16 = payload;

   ext_ctrls.count = 1;
   ext_ctrls.controls = &ext_ctrl;

   return ioctl(fd, VIDIOC_S_EXT_CTRLS, &ext_ctrls);
}

int main(void)
{
   int fd = open("/dev/v4l-subdev0", O_RDWR);

   if (fd < 0)
      return 1;

   // Enable test pattern
   if (trigger_action(fd, V4L2_CID_CUSTOM_S_TEST_PATTERN_ON) < 0)
      perror("enable test pattern");

   // Change GMS register value
   // /!\ NOTE : Check your Proxy/Sensor datasheet for correct values
   if (write_proxy_register(fd, 0x0005, 0x00D3) < 0)
      perror("write Proxy register");

   close(fd);
   return 0;
}
```

Note that the `db-mipi-a1_public.h` file should be placed next to the above file with the following content:

```
/*****************************************************************************
 *                     ^                                                     *
 *                     |  @                                                  *
 *                @    _____   @    _ _ ____              _                  *
 *                |   |  ___|   _| | / ___|  ___ __ _| | ___                *
 *                |   | |_  | | | | \___ \ / __/ _` | |/ _ \               *
 *             @  |   |  _|| |_| | | |___) | (_| (_| | |  __/               *
 *             |  |   |_|   \__,_|_|_|____/ \___\__,_|_|\___|               *
 *             ------------------------------------------------->          *
 *                     |             @  |     |     |    @                  *
 *                     |                |     |     |                        *
 *                                      @     |     @                        *
 *                                            @                              *
 * File: db-mipi-a1_public.h                                                 *
 * Project: db-mipi-a1-driver                                                *
 * Created 21/08/2025 by François Descamps                                   *
 *                                                                           *
 * Copyright (c) 2025 - 2026 FullScale SARL. All Rights Reserved.            *
 *                                                                           *
 * This project is licensed under GNU General Public License v2.0 only.      *
 * See the LICENSE file for details.                                         *
 *****************************************************************************/

#ifndef DB_MIPI_A1_PUBLIC_H
# define DB_MIPI_A1_PUBLIC_H

# include <linux/v4l2-controls.h>

/**
 * @name Sensor temperature
 */

/**
 * @brief Returns the VTemp temperature measurement given by the proxy
 * in the temperature columns.
 *
 * Read-only unsigned 16-bit control.
 */
# define V4L2_CID_CUSTOM_G_COLVTEMP_TEMPERATURE (V4L2_CID_USER_BASE + 0x0000)

/**
 * @brief Returns the VTemp temperature measurement given by the board
 * embedded ADC.
 *
 * Read-only unsigned 16-bit control. This readout is meaningful only
 * after the ADC has been initialized.
 */
# define V4L2_CID_CUSTOM_G_ADCVTEMP_TEMPERATURE (V4L2_CID_USER_BASE + 0x0100)

/**
 * @brief Initializes the embedded ADC VTemp.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_ADCVTEMP_INIT (V4L2_CID_USER_BASE + 0x0101)

/**
 * @name Test Pattern Controls
 */

/**
 * @brief Returns whether the board test pattern is currently enabled.
 *
 * Read-only boolean control.
 */
# define V4L2_CID_CUSTOM_G_TEST_PATTERN_ACTIVATED (V4L2_CID_USER_BASE + 0x0200)
```

(continues on next page)

(continued from previous page)

```
/**
 * @brief Enables the board test pattern generator.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_TEST_PATTERN_ON (V4L2_CID_USER_BASE + 0x0201)

/**
 * @brief Disables the board test pattern generator.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_TEST_PATTERN_OFF (V4L2_CID_USER_BASE + 0x0202)

/**
 * @name Sensor resolution
 *
 * @brief These controls informs the board of the
 * currently connected sensor's resolution.
 */

/**
 * @brief Sets the sensor active X resolution.
 *
 * Unsigned 16-bit write control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_X (V4L2_CID_USER_BASE + 0x0300)

/**
 * @brief Sets the sensor active Y resolution.
 *
 * The driver updates the reported media-bus frame height from this value.
 *
 * Unsigned 16-bit control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_Y (V4L2_CID_USER_BASE + 0x0301)

/**
 * @brief Applies the sensor resolution previously configured through the X and Y
 * controls.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_RESOLUTION_APPLY (V4L2_CID_USER_BASE + 0x0302)

/**
 * @name Sensor Clock
 *
 * @brief These controls configure the embedded clock generator
 * so that it produces the desired sensor clock.
 */

/**
 * @brief Returns whether the sensor clock generator configuration has been applied.
 *
 * Read-only boolean control.
 */
# define V4L2_CID_CUSTOM_G_SENSOR_CLKGEN_CONFIG_DONE (V4L2_CID_USER_BASE + 0x0400)

/**
 * @brief Sets the sensor clock generator M divider coefficient.
 *
 * Unsigned 8-bit control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_M_DIV_COEF (V4L2_CID_USER_BASE + 0x0401)
```

(continues on next page)

(continued from previous page)

```
/**
 * @brief Sets the sensor clock generator N divider coefficient.
 *
 * Unsigned 16-bit control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_N_DIV_COEF (V4L2_CID_USER_BASE + 0x0402)

/**
 * @brief Sets the sensor clock generator output divider coefficient.
 *
 * Unsigned 8-bit control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_OUT_DIV_COEF (V4L2_CID_USER_BASE + 0x0403)

/**
 * @brief Applies the sensor clock generator configuration.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_APPLY_CONFIG (V4L2_CID_USER_BASE + 0x0404)

/**
 * @brief Enables the sensor clock generator output.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_CLKGEN_OUTPUT_ENABLE (V4L2_CID_USER_BASE + 0x0405)

/**
 * @name Shutter controls
 */

/**
 * @brief Returns the current shutter position.
 *
 * Read-only menu control. The returned value matches one of the
 * ``V4L2_SHUTTER_POSITION_*`` constants below.
 */
# define V4L2_CID_CUSTOM_G_SHUTTER_POSITION (V4L2_CID_USER_BASE + 0x0500)

/**
 * @brief Opens the shutter.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_SHUTTER_OPEN (V4L2_CID_USER_BASE + 0x0501)

/**
 * @brief Closes the shutter.
 *
 * Button control.
 */
# define V4L2_CID_CUSTOM_S_SHUTTER_CLOSE (V4L2_CID_USER_BASE + 0x0502)

/**
 * @name Spi Proxy arbitrary register write
 *
 * @brief These controls can be used to write any Proxy register
 * accessible through SPI
 */

/**
 * @brief Writes a Proxy register through the SPI bus.
 *
 * Two-element unsigned 16-bit control. Element 0 is the SPI proxy register
 * address and element 1 is the value to write.
```

(continues on next page)

(continued from previous page)

```
 */
# define V4L2_CID_CUSTOM_S_SPI_PROXY_REGISTER (V4L2_CID_USER_BASE + 0x0600)

/**
 * @name I2C Proxy arbitrary register access
 *
 * @brief These controls can be used to read or write any Proxy register
 * accessible through I2C
 */

/**
 * @brief Reads a Proxy register through the I2C bus.
 *
 * Unsigned 8-bit control. User space first writes
 * the register address to this control, then reads the same control back to
 * obtain the register value.
 */
# define V4L2_CID_CUSTOM_G_I2C_PROXY_REGISTER (V4L2_CID_USER_BASE + 0x0700)

/**
 * @brief Writes a Proxy register through the I2C bus.
 *
 * Two-element unsigned 16-bit control. Element 0 is the register address and
 * element 1 is the value to write.
 */
# define V4L2_CID_CUSTOM_S_I2C_PROXY_REGISTER (V4L2_CID_USER_BASE + 0x0701)

/**
 * @name Sensor ISP arbitrary register access
 *
 * @brief These controls can be used to read or write any Sensor ISP register
 */

/**
 * @brief Reads a Sensor ISP register through a two-step access pattern.
 *
 * Unsigned 16-bit control. User space first writes
 * the register address to this control, then reads the same control back to
 * obtain the register value.
 */
# define V4L2_CID_CUSTOM_G_SENSOR_ISP_REGISTER (V4L2_CID_USER_BASE + 0x0800)

/**
 * @brief Writes a Sensor ISP register.
 *
 * Two-element unsigned 16-bit control. Element 0 is the register address and
 * element 1 is the value to write.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_ISP_REGISTER (V4L2_CID_USER_BASE + 0x0801)

/**
 * @name Sensor arbitrary register access
 *
 * @brief These controls can be used to read or write any Sensor register
 */

/**
 * @brief Reads a sensor register through a two-step access pattern.
 *
 * Unsigned 8-bit control. User space first writes
 * the register address to this control, then reads the same control back to
 * obtain the register value.
 */
# define V4L2_CID_CUSTOM_G_SENSOR_REGISTER (V4L2_CID_USER_BASE + 0x0900)

/**
```

(continues on next page)

(continued from previous page)

```
 * @brief Writes a sensor register.
 *
 * Two-element unsigned 16-bit control. Element 0 is the register address and
 * element 1 is the value to write.
 */
# define V4L2_CID_CUSTOM_S_SENSOR_REGISTER (V4L2_CID_USER_BASE + 0x0901)

/**
 * @name Shutter constant values
 */

/**
 * @brief Shutter position value for a closed shutter.
 */
# define V4L2_SHUTTER_POSITION_CLOSED            0

/**
 * @brief Shutter position value for an open shutter.
 */
# define V4L2_SHUTTER_POSITION_OPEN              1

/**
 * @brief Shutter position value for a shutter currently moving.
 */
# define V4L2_SHUTTER_POSITION_MOVING            2

#endif
```

10 Revision history

Date	Author(s)	Version	Comments
04/2026	F. DESCAMPS	Rev. A	First document release

11 Disclamer

FullScale reserves the right to make any changes without further notice to any products herein. FullScale makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does FullScale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including with out limitation consequential or incidental damages.

“Typical” parameters can and do vary in different applications. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts.

Use of FullScale products or services with statements different from or beyond the parameters stated by FullScale for that product or service voids all express and any implied warranties for the associated FullScale product or service and is an unfair and deceptive business practice. FullScale is not responsible or liable for any such statements.

FullScale products are not authorized for use as critical components in life support devices or systems without the express written approval of FullScale.

FullScale software and associated products cannot be used except strictly in accordance with an FullScale software license. The terms of the appropriate FullScale software license shall prevail over the above terms to the extent of any inconsistency.