Blackmagicdesign

Installation and Operation Manual

Blackmagic 3G-SDI Shield for Arduino

February 2020

English, 日本語, Français, Deutsch, Español, 中文, 한국어, Русский, Italiano, Português and Türkçe.

Languages

To go directly to your preferred language, simply click on the hyperlinks listed in the contents below.

Welcome

Thank you for purchasing your new Blackmagic 3G-SDI Shield for Arduino.

We are always interested in new technologies and are excited by all the creative ways our SDI products can be used. With your 3G-SDI Shield for Arduino, you can now integrate the Arduino into your SDI workflow to get more control options with your Blackmagic Design equipment.

For example, ATEM switchers can control Blackmagic URSA Mini and Blackmagic Studio Cameras via data packets embedded in the SDI signal. If you are not running an ATEM switcher, but you would still like the ability to control your Blackmagic cameras, you can build custom control solutions with your 3G-SDI Shield for Arduino. The shield gives you the SDI platform to build upon, so you can loop the program return feed from your switcher, through the shield, and into the program input on your Blackmagic Cameras.

Writing the code to send the commands to the camera is easy and all the supported commands are included in this manual.

You can control the cameras using a computer, or you can add buttons, knobs and joysticks to your shield and build dynamic hardware controllers for adjusting features such as lens focus and zoom, aperture settings, pedestal and white balance control, the camera's powerful built in color corrector, and much more. Building your own custom controller is useful for production, but it's also a lot of fun!

We are excited by this technology and would love to hear about any SDI controllers you have built for your 3G-SDI Shield for Arduino!

This instruction manual contains all the information you need to start using your Blackmagic 3G-SDI Shield for Arduino. Please check the support page on our website at www.blackmagicdesign.com for the latest version of this manual and for updates to your shield's internal software. Keeping your software up to date will ensure you get all the latest features! When downloading software, please register with your information so we can keep you updated when new software is released. We are continually working on new features and improvements, so we would love to hear from you!

Grant Petty
CEO Blackmagic Design

Contents

# Blackmagic 3G-SDI Shield for Arduino

Getting Started

Attaching and Soldering Headers

Your Blackmagic 3G-SDI Shield for Arduino is supplied with 4 stackable headers, including two 8 pin headers, a 10 pin, and a 6 pin header. Headers are bridging connectors used to mount your shield to the Arduino board, and because they are stackable you can attach other shields on top with additional components, such as control buttons, knobs and joysticks. The header layout supports mounting to Arduino boards with an R3 footprint, such as the Arduino UNO.

To attach the headers to your shield:

1 Insert the pins of each header through the corresponding pin holes on each side of your Blackmagic 3G-SDI Shield. Refer to the illustration below for the header layout arrangement.

0 - Serial RX
1 - Serial TX

A5 (I^{2}C) SCL
A4 (I^{2}C) SDA

(I^{2}C) SDA
(I^{2}C) SCL

Blackmagicdesign
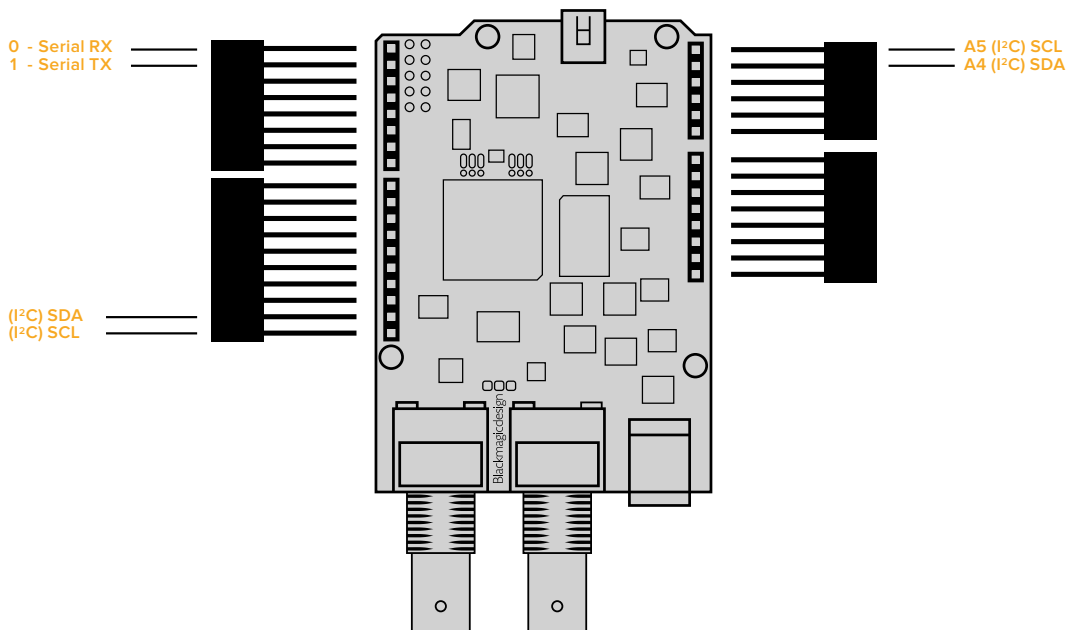
NOTE When connecting to the shield, communication is via I^{2}C or Serial. We recommend I^{2}C as this enables the serial monitor to be used and makes all other pins available. Select the communication mode when defining the BMDSDIControl object in the sketch. Refer to the 'Communicating with your Blackmagic 3G-SDI Shield for Arduino' section for more information.

2 Solder the base of each header pin to the underside of your shield. Make sure the solder on each pin creates a firm join with the pin hole, but does not touch the solder on nearby pins.

Mounting to the Arduino Board

Now that your headers are soldered to your shield, you can mount the 3G-SDI shield to your Arduino board.

Carefully holding each side of the shield, align the header pins with your Arduino board's headers and gently push the pins into the header slots. Be careful not to bend any of the pins while mounting the shield.
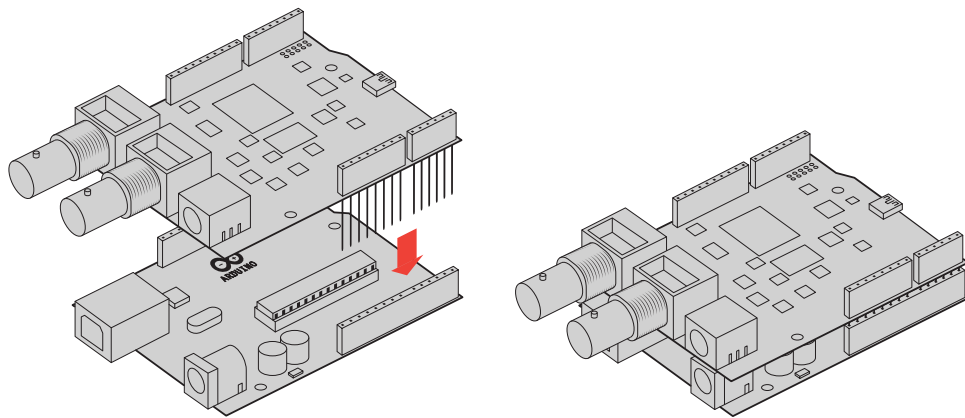
With all pins plugged in, the connection between the Blackmagic shield and the Arduino board should be firm and stable.

Plugging in Power

To power your Blackmagic 3G-SDI Shield for Arduino, simply plug in a 12V power adapter into the 12V power input on your Blackmagic shield.

NOTE Plugging power into the Arduino board will not provide sufficient power to the Blackmagic shield, however, powering the Blackmagic shield will provide power to the Arduino as well, so make sure power is connected to your Blackmagic shield only.

Connecting to SDI Equipment

With power supplied, you can now plug your Blackmagic 3G-SDI Shield into your SDI equipment. For example, to plug into a switcher and a Blackmagic URSA Mini:

1 Plug the program output from your switcher to the Blackmagic 3G-SDI Shield's SDI input.

2 Plug your Blackmagic 3G-SDI Shield's SDI output into the 'program' SDI input marked PGM on your Blackmagic URSA Mini.

A connection diagram is provided below.

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini
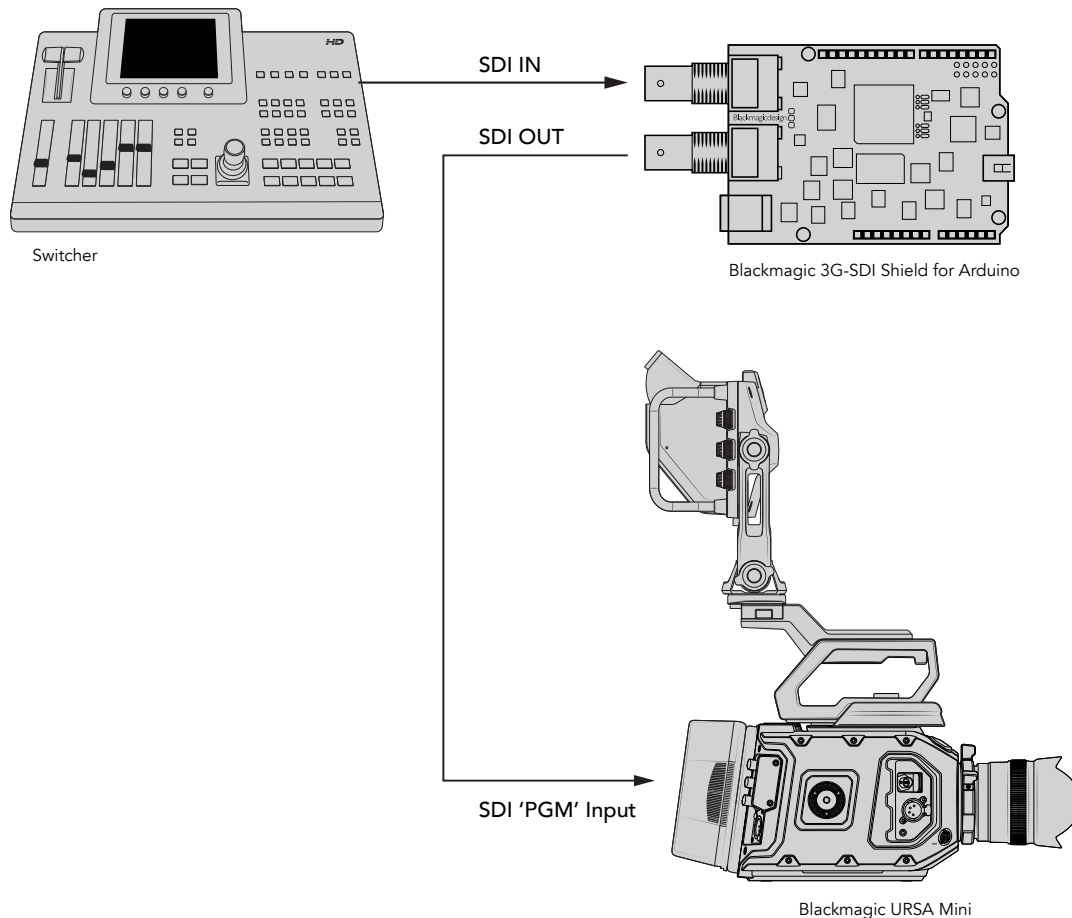
That's all there is to getting started!

Now that your shield is mounted to the Arduino board, powered, and connected to your SDI equipment, you can install the internal software and library files, program the Arduino software and begin using the shield to control your equipment.

Continue reading the manual for information on how to install the shield's internal software, and where to install the Arduino library files so the shield can communicate with your Arduino.

> **TIP** You can also use your Blackmagic 3G-SDI Shield for Arduino to control other Blackmagic Design products, such as Blackmagic MultiView 16. For example, when your shield is connected to input 16, you can display a tally border on the multi view. When using Blackmagic URSA Mini cameras, make sure the grid overlay is turned on to enable the tally display on the camera viewfinder. Refer to your Blackmagic camera manual for more information.

Software Installation

Installing Internal Software

Blackmagic Shield for Arduino Setup is used to update your shield's internal software. The internal software communicates with the Arduino board, and controls the board using Arduino library files. These library files are installed with the setup software and all you need to do is copy the folder containing the files and paste it into your Arduino application folder. You can find information about the library files and how to install them in the next section of this manual.

We recommend downloading the latest Blackmagic Shield for Arduino software and updating your shield so you can benefit from new features and improvements. The latest version can be downloaded from the Blackmagic Design support center at www.blackmagicdesign.com/support

To install the internal software using Mac OS X:

1   Download and unzip the Blackmagic Shield for Arduino software.

2   Open the resulting disk image and launch the Blackmagic Shield for Arduino installer. Follow the on screen instructions.

3   After installing the latest version of Blackmagic Shield for Arduino installer, power your Blackmagic shield and connect it to your computer via a USB cable.

4   Now launch the setup utility and follow any onscreen prompt to update your shield's internal software. If no prompt appears, the internal software is up to date and there is nothing further you need to do.

To install the internal software using WIndows:

1   Download and unzip the Blackmagic Shield for Arduino software.

2   You should see a Blackmagic Shield for Arduino folder containing this manual and the Blackmagic Shield for Arduino installer. Double-click the installer and follow the onscreen prompts to complete the installation.

3   After installing the latest version of the Blackmagic Shield for Arduino installer, power your Blackmagic shield and connect it to your computer via a USB cable.

4   Now launch the setup utility and follow any onscreen prompt to update your shield's internal software. If no prompt appears, the internal software is up to date and there is nothing further you need to do.

Installing Arduino Library Files

The programs written to control your Arduino are called sketches and your Blackmagic 3G-SDI Shield for Arduino uses Arduino library files that help make writing sketches easier. After installing your shield's setup software, the library files are installed into a folder named 'Library'. All you need to do now is copy the folder containing the library files and paste it into your Arduino libraries folder.

> **NOTE** The Arduino IDE software needs to be closed when installing libraries.

To install the library files on Mac OS X:

1   Open 'Blackmagic Shield for Arduino' in your 'applications' folder.

2   Open the 'Library' folder and right click/copy the folder named: BMDSDIControl.

3   Now go to your computer's 'documents' folder and open the Arduino folder.

4   You will see a sub-folder named 'libraries'. Paste the BMDSDIControl folder into the 'libraries' folder.

To install the library files on Windows:

1   Open the Programs/ Blackmagic Shield for Arduino folder.

2   You will now see a subfolder named 'Library'. Open this folder and then right click/copy the folder named: BMDSDIControl.

3   Now go to your computer's 'documents' folder and open the Arduino folder.

4   You will see a sub-folder named 'libraries'. Paste the BMDSDIControl folder into the 'libraries' folder.

That's all you need to do to install the Blackmagic Design library files on your computer. When running the Arduino software, you will now also have Blackmagic Design example sketches to choose from.

Simply go to the 'file' drop down menu in the Arduino software menu bar, and select 'examples'. Now select BMDSDIControl and you will see a list of example sketches you can use.

With the library files stored in the correct folder, your shield can now use them to communicate with the Arduino board. All you need to do is program the Arduino IDE software. Refer to the 'Programming Arduino Sketches' section for more information.

> **NOTE** If an updated library file with examples is released in the future, you will need to delete the old BMDSDIControl folder and replace it with the new folder using the method described above.

Blackmagic Shield for Arduino Setup
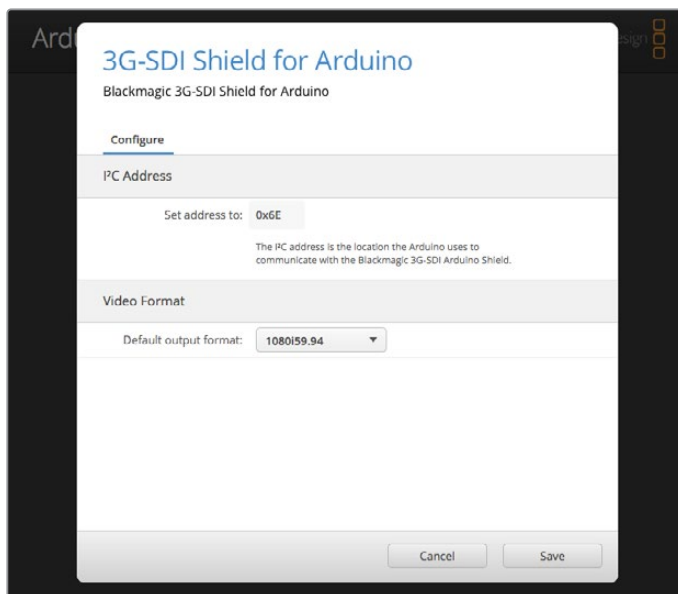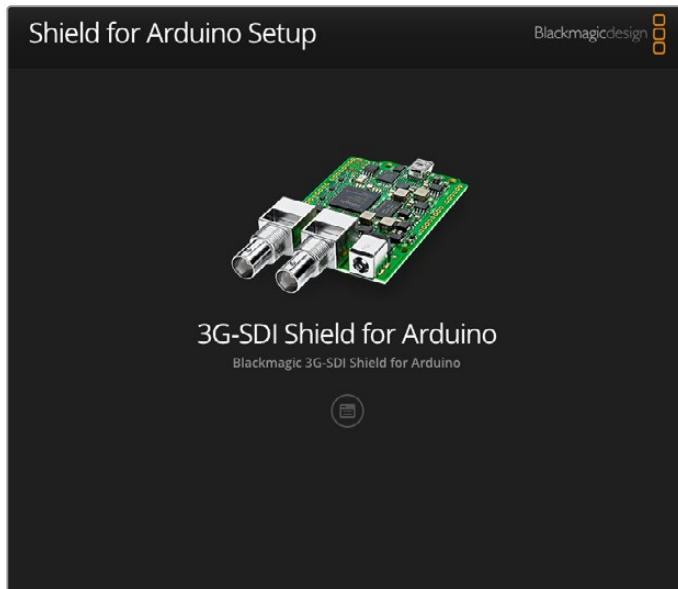
The Blackmagic Shield for Arduino Setup software lets you change settings on your shield such as the I²C address and video output format.

With Blackmagic Shield for Arduino Setup installed on your computer, you can now change settings for your shield, such as the 'I²C address', which identifies your shield so the Arduino board can communicate with it, and the 'video format', which sets the output format for your shield.

I²C Address

In very rare cases, there is a potential for another shield mounted to your Blackmagic shield to share the same I²C address as your shield's default address which will create a conflict. If this occurs, you can change your shield's default address setting.

The default address for your shield is 0x6E, however, you can choose from a range of addresses between 0x08 and 0x77.

To change the address for your shield:

1   Launch Blackmagic Shield for Arduino Setup and click on your shield's 'settings' icon.

2   In the 'set address to:' edit box, type the address you wish to use.

3   Click 'save'.

Video Format

The default output format is selected in the setup utility for when no input is connected. When an input is detected, the output will follow the same format as the input. If this input is removed the output will revert to the default output format selected in the utility. You can change the video format by clicking in the 'default output format' drop down menu and selecting the format you want.

You can choose from the following video output formats:

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Programming Arduino Sketches

The programs, or sketches, written into the Arduino software are very easy to write! Sketches are written using common 'C' programming language. When programming your sketches using commands from the Studio Camera Control Protocol, the shield embeds these commands into the SDI output which lets you control your Blackmagic URSA Mini or Blackmagic Studio Cameras.

All supported commands are included in the Studio Camera Control Protocol section of this manual so you can take the commands from the protocol and use them in your sketch.
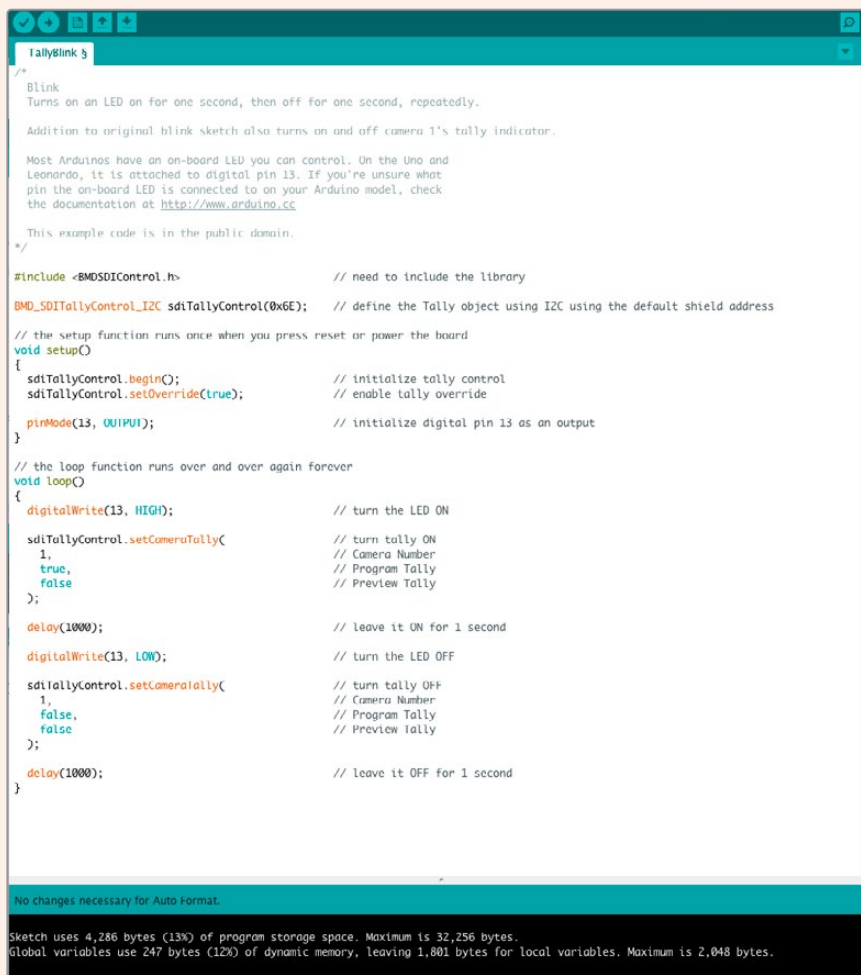
Testing your Blackmagic Shield and Library Installation

After everything is connected as described in the 'Getting Started' section and you have installed the setup software and library files, you'll want to check that your shield is communicating with the Arduino board and that everything is working as it should.

A fast way is to open and run the supplied tally blink example sketch.

To do this:

1 Launch the Arduino IDE software.

2 Go to the 'tools' menu and select the Arduino board and Port number.

3 From the 'File' menu, select 'Examples/BMDSDIControl' and choose the sketch named 'TallyBlink'.

4 Upload the sketch to your board.

The Tally Blink example sketch is a fast and easy way to test your Blackmagic 3G-SDI Shield for Arduino. Raw data can be sent to your shield via I²C using commands from the Studio Camera Protocol document, but we have also provided custom libraries to make programming sketches much easier.

> **NOTE** Make sure your Blackmagic Camera's tally number is set to 1. Also, ensure the camera's grid overlay is turned on to enable the tally display on the camera viewfinder. Refer to your Blackmagic camera manual for more information.

You should now see the tally light on your Blackmagic Studio Camera blink once every second. If you see the tally light blinking you can be sure your Blackmagic shield is communicating with the Arduino and everything is working properly.

If the tally light is not blinking, check that your Blackmagic camera's tally number is set to 1 and the camera's grid overlay is turned on.

If you need further assistance, please visit the Blackmagic Design support center at www.blackmagicdesign.com/support. Refer to the help section of this manual for more information on the different ways you can get help setting up your shield.

LED Indicators

Your Blackmagic 3G-SDI Shield for Arduino has six indicator LEDs that confirm activity on your shield such as power, UART, I²C and SPI communication, plus indicators to show when tally and camera control overrides are enabled.
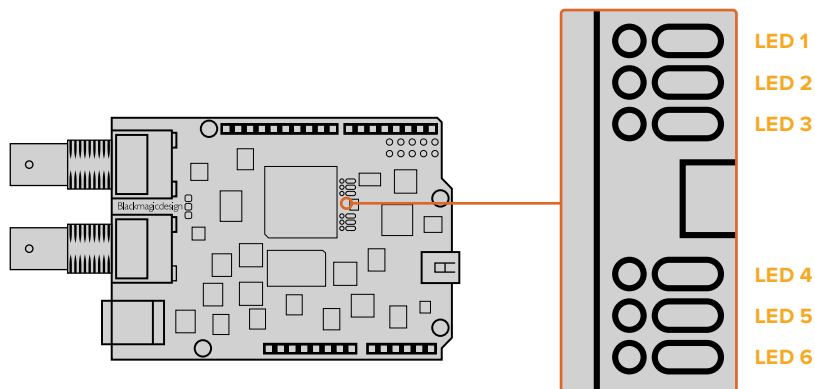
LED 1 - System Active
Illuminates when power is connected to the shield.

LED 2 - Control Overrides Enabled
Illuminates if you have enabled camera control in your Arduino sketch.

LED 3 - Tally Overrides Enabled
Illuminates if you have enabled tally in your Arduino sketch.

LED 5 - I²C Parser Busy
Illuminates when communication is detected between your shield and the Arduino using the I²C protocol.

 LED 6 - Serial Parser Busy
Illuminates when UART communication is detected.

When your Blackmagic shield is booting, the power indicator will remain off and LEDs 3, 4 and 5 will indicate the following activity.

LED 3 - Application image loading

LED 4 - EEPROM initializing

LED 5 - Memory check in progress

After a successful boot, the power LED will turn on and all LEDs will resume their standard functions during operation.

In the rare case of a boot failure, all LEDs except for the failed activity will flash rapidly so you can identify the cause of the failure.

Attaching Shield Components

If you want to build your own hardware controller, you can create a new shield with buttons, knobs and a joystick for more tactile, hands on control. Simply mount the custom shield to your Blackmagic 3G-SDI Shield for Arduino by plugging it into your shield's header slots. There is no limit to the types of controllers you can build. You can even replace the circuitry in an old CCU with your own custom Arduino solution for an industry standard camera control unit.
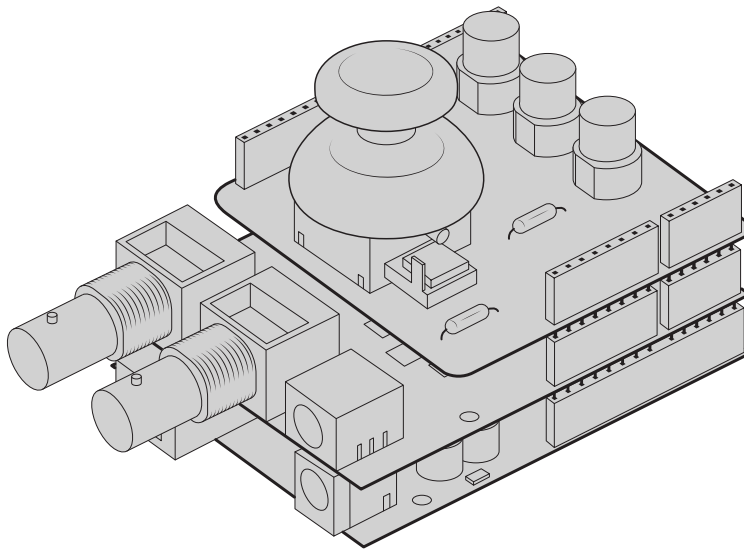
You can create your own hardware controller and plug it into your Blackmagic 3G-SDI Shield for Arduino for more interactive and refined control.

Communicating with your Blackmagic Shield for Arduino

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I²C or Serial. We recommend I²C because of the low pin count and it frees up the serial monitor. This also allows you to use more I²C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I²C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                shieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial    sdiCameraControl;
BMD_SDITallyControl_Serial     sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                shieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
Video	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Confi-guration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDddddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, ddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDddddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, dddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header		command						data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1. Standard mode (100 kbit/s)
2. Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
-------------------- |----------------------------
 A4                  | Serial Data (SDA)
 A5                  | Serial Clock (SCL)
```

I²C Protocol (Writes):

(START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

(START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
-------------------- |----------------------------
 IO1                 | Serial Transmit (TX)
 IO0                 | Serial Receive (RX)
```

UART Protocol (Writes):

[0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

[0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name        | R/W   | Register Description
-------------------- |----------   |-----  |-----------------------------
0x0000 - 0x0003      | IDENTITY    | R     | Hardware Identifier
0x0004 - 0x0005      | HWVERSION   | R     | Hardware Version
0x0006 - 0x0007      | FWVERSION   | R     | Firmware Version
                     |             |       |
0x1000               | CONTROL     | R/W   | System Control
                     |             |       |
0x2000               | OCARM       | R/W   | SDI Control Override Arm
0x2001               | OCLENGTH    | R/W   | SDI Control Override Length
0x2100 - 0x21FE      | OCDATA      | R/W   | SDI Control Override Data
                     |             |       |
0x3000               | ICARM       | R/W   | SDI Control Incoming Arm
0x3001               | ICLENGTH    | R     | SDI Control Incoming Length
0x3100 - 0x31FE      | ICDATA      | R     | SDI Control Incoming Data
                     |             |       |
0x4000               | OTARM       | R/W   | SDI Tally Override Arm
0x4001               | OTLENGTH    | R/W   | SDI Tally Override Length
0x4100 - 0x41FE      | OTDATA      | R/W   | SDI Tally Override Data
                     |             |       |
0x5000               | ITARM       | R/W   | SDI Tally Incoming Arm
0x5001               | ITLENGTH    | R     | SDI Tally Incoming Length
0x5100 - 0x51FE      | ITDATA      | R     | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

```
[ IDENTITY ]
31        0
```

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7           0
```

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7           0
```

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

```
[ RESERVED ] [ OVERRIDE OUTPUT ] [ RESET TALLY ] [ OVERRIDE TALLY ] [
OVERIDE CONTROL ]
7          4            3              2               1            0
```

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Help

Getting Help

Your Blackmagic 3G-SDI Shield for Arduino is a developers tool designed for you to develop independently based on your custom requirements.

For the most up to date information about your shield, visit the Blackmagic Design online support pages and check the latest support material.

Blackmagic Design Online Support Pages

The latest manual, software and support notes can be found at the Blackmagic Design support center at www.blackmagicdesign.com/support.

Arduino Development Forum

If you have programming questions, you can get help from Arduino development forums on the Internet. There is a whole community of Arduino developers and many good quality forums where you can ask software questions, or even find a willing engineer to hire to implement your solution for you!

Blackmagic Design Forum

The Blackmagic Design forum on our website is a helpful resource you can visit for more information and creative ideas. This can also be a faster way of getting help as there may already be answers you can find from other experienced users and Blackmagic Design staff which will keep you moving forward. You can visit the forum at https://forum.blackmagicdesign.com

Checking the Software Version Currently Installed

To check which version of Blackmagic 3G-SDI Shield for Arduino Setup software is installed on your computer, open the About Blackmagic 3G-SDI Shield for Arduino Setup window.

- On Mac OS X, open Blackmagic 3G-SDI Shield for Arduino Setup from the Applications folder. Select About Blackmagic Shield for Arduino Setup from the application menu to reveal the version number.
- On Windows 7, open Blackmagic 3G-SDI Shield for Arduino Setup from your Start menu. Click on the Help menu and select About Blackmagic 3G-SDI Shield for Arduino Setup to reveal the version number.
- On Windows 8, open Blackmagic 3G-SDI Shield for Arduino Setup from the Blackmagic 3G-SDI Shield for Arduino Setup tile on your Start page. Click on the Help menu and select About Blackmagic Shield for Arduino Setup to reveal the version number.

How to Get the Latest Software Updates

After checking the version of Blackmagic 3G-SDI Shield for Arduino Setup software installed on your computer, please visit the Blackmagic Design support center at www.blackmagicdesign.com/support to check for the latest updates. While it is usually a good idea to run the latest updates, it is wise to avoid updating any software if you are in the middle of an important project.

Warranty

12 Month Limited Warranty

Blackmagic Design warrants that the Blackmagic 3G-SDI Shield for Arduino product will be free from defects in materials and workmanship for a period of 12 months from the date of purchase. If a product proves to be defective during this warranty period, Blackmagic Design, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, you the Customer, must notify Blackmagic Design of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. The Customer shall be responsible for packaging and shipping the defective product to a designated service center nominated by Blackmagic Design, with shipping charges pre paid. Customer shall be responsible for paying all shipping changes, insurance, duties, taxes, and any other charges for products returned to us for any reason.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Blackmagic Design shall not be obligated to furnish service under this warranty: a) to repair damage resulting from attempts by personnel other than Blackmagic Design representatives to install, repair or service the product, b) to repair damage resulting from improper use or connection to incompatible equipment, c) to repair any damage or malfunction caused by the use of non Blackmagic Design parts or supplies, or d) to service a product that has been modified or integrated with other products when the effect of such a modification or integration increases the time or difficulty of servicing the product. THIS WARRANTY IS GIVEN BY BLACKMAGIC DESIGN IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. BLACKMAGIC DESIGN AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. BLACKMAGIC DESIGN'S RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE WHOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR ANY INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER BLACKMAGIC DESIGN OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES. BLACKMAGIC DESIGN IS NOT LIABLE FOR ANY ILLEGAL USE OF EQUIPMENT BY CUSTOMER. BLACKMAGIC IS NOT LIABLE FOR ANY DAMAGES RESULTING FROM USE OF THIS PRODUCT. USER OPERATES THIS PRODUCT AT OWN RISK.
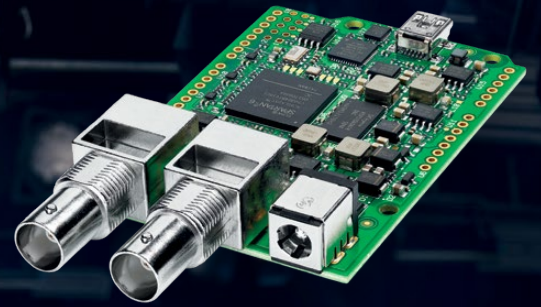
インストール/オペレーション マニュアル

Blackmagic 3G-SDI Shield for Arduino

2020年2月

日本語

ようこそ

このたびは新しいBlackmagic 3G-SDI Shield for Arduinoをお買い求めいただき誠にありがとうございました。

私たちは常に新しいテクノロジーに関心を持っており、弊社のSDI製品がクリエイティブに使用されていることを非常に嬉しく思っています。3G-SDI Shield for Arduinoを使用すれば、ArduinoをSDIワークフローに組み込んで、より多くのコントロールオプションをBlackmagic Design製品に追加でききます。

例えば、SDI信号にエンベッドしたデータパケット経由で、ATEMスイッチャーからBlackmagic URSA MiniやBlackmagic Studio Cameraをコントロールできます。また、ATEMスイッチャーを使用せずにBlackmagicカメラをコントロールしたい場合は、3G-SDI Shield for Arduinoを使ってカスタムコントロールソリューションを構築できます。同シールドはSDIプラットフォームとして使用できるので、スイッチャーからのプログラムリターンフィードを、シールドを通じてBlackmagicカメラのプログラム入力にループできます。

カメラへのコマンド送信用のコードは簡単に書くことができ、すべての対応コマンドがこのマニュアルに記載されています。

また、コンピューターからカメラのコントロールも可能です。あるいは、ボタン、ノブ、ジョイスティックをシールドに追加して、ダイナミックなハードウェアコントローラーを構築することで、レンズフォーカス/ズーム、アパーチャー設定、ペデスタルおよびホワイトバランスコントロール、カメラのパワフルな内蔵カラーコレクターなどの機能を調整することも可能です。独自のカスタムコントローラーはプロダクションで便利なだけでなく、開発自体も面白い作業です！

このテクノロジーは拡張性が高く、多くの使用方法が考えられます。SDIコントローラーをカスタムビルドした際には、その内容をぜひお聞かせください！

このマニュアルには、Blackmagic 3G-SDI Shield for Arduinoを使用する上で必要な情報がすべて記載されています。弊社ウェブサイト www.blackmagicdesign.com/jp のサポートページでこのマニュアルの最新バージョンを確認し、シールドの内部ソフトウェアをアップデートしてください。ソフトウェアをアップデートすることで、常に最新の機能をお使いいただけます。ソフトウェアをダウンロードする際にユーザー登録していただければ、新しいソフトウェアのリリース時にお知らせいたします。常に新機能の開発および製品の改善に努めていますので、ユーザーの皆様からご意見をいただければ幸いです。

Grant Petty

Blackmagic Design CEO
グラント・ペティ

目次

Blackmagic 3G-SDI Shield for Arduino

はじめに

ヘッダーの取り付けおよびはんだ付け

Blackmagic 3G-SDI Shield for Arduinoには、積み重ね可能な4つのヘッダーが同梱されています。8ピンヘッダーが2つ、そして10ピン/6ピンヘッダーが1つずつです。ヘッダーは、Arduinoボードにシールドをマウントするためのブリッジコネクターです。積み重ねられるので、コントロールボタンやノブ、ジョイスティックなどの追加コンポーネントの付いた別のシールドをさらに取り付けることが可能です。ヘッダーレイアウトは、Arduino UNOなど、R3フットプリントのArduinoボードへのマウントをサポートします。

ヘッダーをシールドに取り付ける：

1   各ヘッダーのピンを、Blackmagic 3G-SDI Shieldシールドの各サイドにある、対応するピンホールに差し込みます。ヘッダーレイアウトの配置に関しては、以下の図を参照してください。
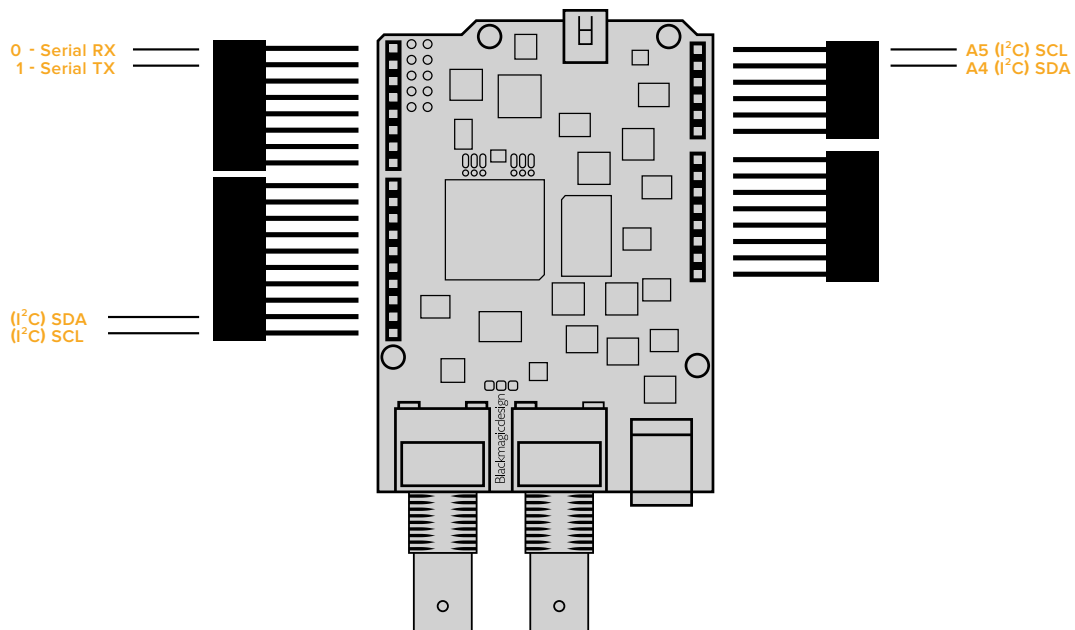
> **メモ** シールドと接続する際、通信はI2Cあるいはシリアル経由です。I2Cはシリアルモニターを有効にして、すべてのピンを使用できるため、これを推奨します。スケッチでBMDSDIControlオブジェクトを設定する際に通信モードを選択します。詳細は、「Blackmagic Shield for Arduinoとの通信」セクションを参照してください。

2   各ヘッダーピンの底部をシールドの下面にはんだ付けします。各ピンのはんだがピンホールにしっかり接合され、周辺のピンに触れていないことを確認します。

Arduinoボードへのマウント

ヘッダーをシールドにはんだ付けしたら、次はこの3G-SDIシールドをArduinoボードにマウントします。

シールドの両サイドを注意深く持ち、ヘッダーピンをArduinoボードのヘッダーと揃えてピンをヘッダースロットへゆっくりと差し込みます。シールドをマウントする際に、ピンが曲がらないように注意してください。
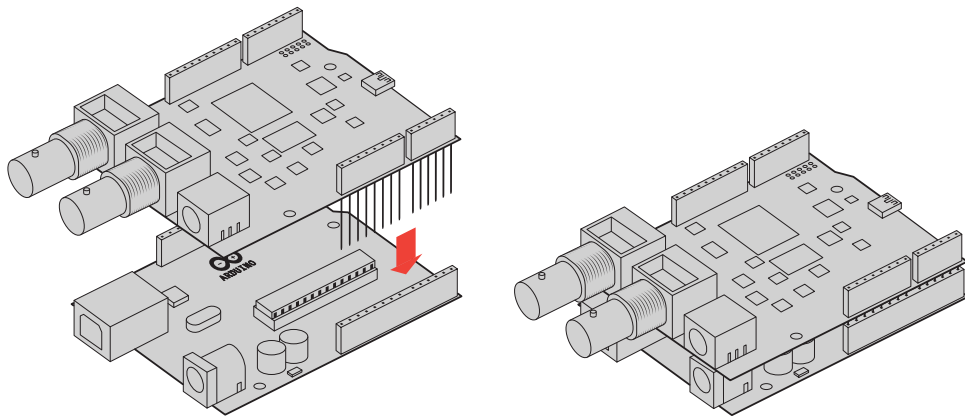
すべてのピンが差し込まれると、BlackmagicシールドとArduinoボードはしっかりと接続され固定されます。

電源の接続

Blackmagic 3G-SDI Shield for Arduinoに電源を入れるには、12V電源アダプターをBlackmagicシールドの12V電源入力に差し込みます。

メモ　Arduinoボードに電源を接続しても、Blackmagicシールドには十分な電力が供給されませんが、Blackmagicシールドに電源を接続すればArduinoボードにも給電されるので、電源がBlackmagicシールドに接続されていることを確認してください。

SDI機器への接続

電源を接続したら、次にBlackmagic 3G-SDI Shieldを スイッチャーとBlackmagic URSA MiniなどのSDI
機器に接続します。

1 スイッチャーからのプログラム出力をBlackmagic 3G-SDI ShieldのSDI入力に接続します。

2 Blackmagic 3G-SDI ShieldのSDI出力をBlackmagic URSA MiniのプログラムSDI入力 (PGM)
に接続します。

以下の接続図を参照してください。

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini
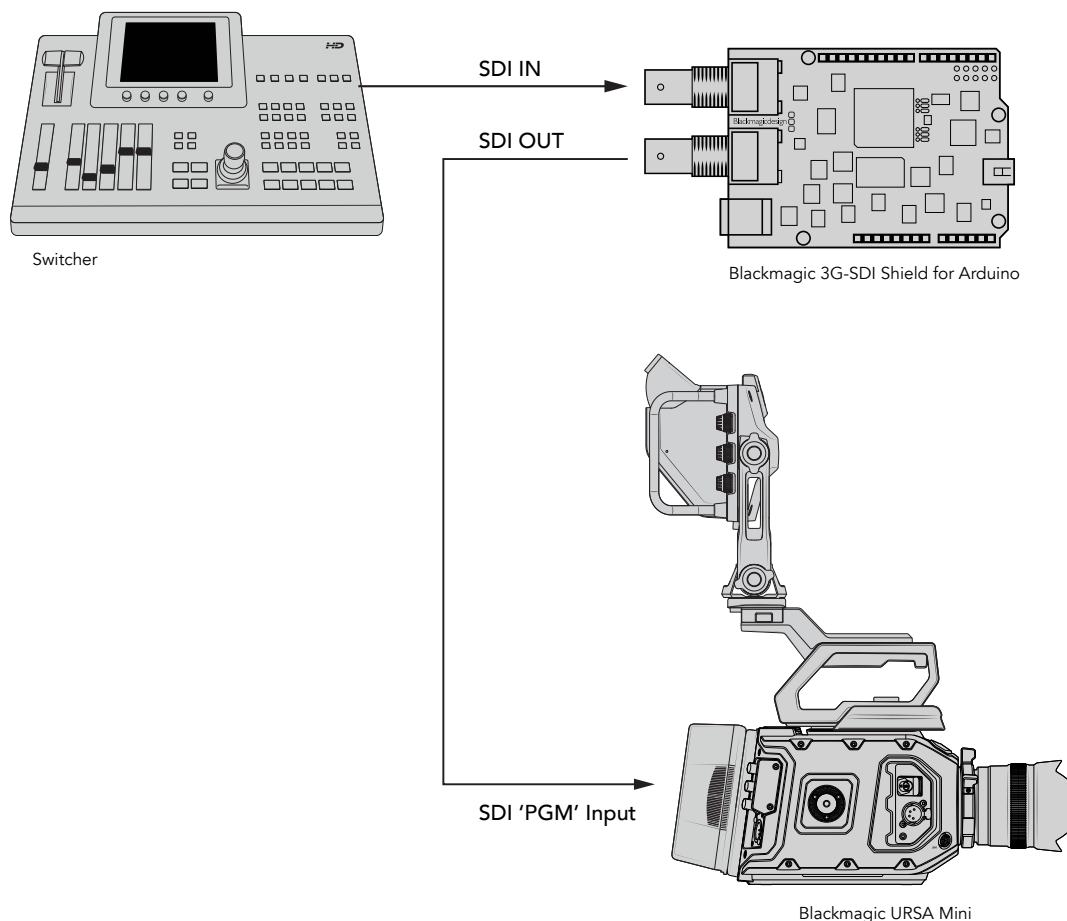
最初に必要な作業はこれだけです！

ここまでの作業でシールドがArduinoボードにマウントされ、電源およびSDI機器に接続されました。これで内
部ソフトウェアおよびライブラリファイルのインストール、Arduinoソフトウェアのプログラム作成が可能とな
り、シールドを使ったコントロールを開始できます。

シールドとArduinoを通信可能にするためのシールドの内部ソフトウェアのインストール方法およびArduino
ライブラリファイルのインストール場所に関しては、同マニュアルを読み進めてください。

作業のこつ Blackmagic 3G-SDI Shield for Arduinoは、Blackmagic MultiView 16などの他
Blackmagic Design製品をコントロールすることもできます。例えば、シールドを入力16に接続す
ると、マルチビューでタリーボーダーを表示できます。Blackmagic URSA Miniカメラを使用する場
合、カメラのビューファインダーでタリー表示を有効にするには、グリッドオーバーレイをオンにしてく
ださい。詳細は、Blackmagicカメラマニュアルを参照してください。

ソフトウェアのインストール

> **メモ** Blackmagic Shield for Arduino Setup Utilityのインストール前に、最新のArduino IDE
> ソフトウェアを<u>www.arduino.cc</u>からダウンロードして、コンピューターにインストールしてくだ
> さい。
>
> Arduinoソフトウェアのインストール後、Blackmagic 3G-SDI Shieldの内部ソフトウェアをインス
> トールできます。

内蔵ソフトウェアのインストール

Blackmagic Shield for Arduino Setupを使ってシールドの内部ソフトウェアをアップデートできます。内部
ソフトウェアはArduinoボードと通信し、Arduinoライブラリファイルを使ってボードをコントロールします。こ
れらのライブラリファイルは、セットアップソフトウェアでインストールできます。必要な作業は、ファイルを含
むフォルダーをコピーして、Arduinoアプリケーションフォルダーにペーストするだけです。このマニュアルの次
セクションで、ライブラリファイルおよびそのインストール方法に関して説明します。

新しい機能および改良機能を使用できるよう、最新のBlackmagic Shield for Arduinoソフトウェアをダウ
ンロードしてシールドをアップデートすることをお勧めします。最新バージョンは、Blackmagic Designサポ
ートセンター (<u>www.blackmagicdesign.com/jp/support</u>) でダウンロードできます。

Mac OS Xで内部ソフトウェアをインストールする：

1   Blackmagic Shield for Arduinoソフトウェアをダウンロードして解凍します。

2   ディスクイメージを開いてBlackmagic Shield for Arduino Installerを起動します。スクリーン上の
     指示に従ってください。

3   最新バージョンのBlackmagic Shield for Arduino Installerをインストールしたら、Blackmagic
     シールドの電源を入れて、USBケーブルでコンピューターと接続します。

4   Setup Utilityを起動し、スクリーンの指示に従ってシールドの内部ソフトウェアをアップデートします。
     内蔵ソフトウェアが最新で何もする必要がない場合、指示は表示されません。

WIndowsで内部ソフトウェアをインストールする：

1   Blackmagic Shield for Arduinoソフトウェアをダウンロードして解凍します。

2   このマニュアルおよびBlackmagic Shield for Arduino Installerを含むBlackmagic Shield for
     Arduinoフォルダーが確認できます。インストーラーをダブルクリックし、画面に表示される指示に従
     ってインストールします。

3   最新バージョンのBlackmagic Shield for Arduino Installerをインストールしたら、Blackmagic
     シールドの電源を入れて、USBケーブルでコンピューターと接続します。

4   Setup Utilityを起動し、スクリーンの指示に従ってシールドの内部ソフトウェアをアップデートします。
     内蔵ソフトウェアが最新で何もする必要がない場合、指示は表示されません。

Arduinoライブラリファイルのインストール

Arduinoをコントロールするために書かれたプログラムはスケッチと呼ばれます。Blackmagic 3G-SDI Shield for Arduinoは、スケッチを簡単に書くことができる、Arduinoライブラリファイルを使用します。シールドのセットアップソフトウェアをインストールしたら、ライブラリファイルは「Library」と名前の付いたフォルダーにインストールされます。必要な作業は、ライブラリファイルを含むフォルダーをコピーして、Arduinoのライブラリフォルダーにペーストするだけです。

> **メモ** ライブラリをインストール中は、Arduino IDEソフトウェアを閉じる必要があります。

Mac OS Xでライブラリファイルをインストール：

1   「Application」フォルダーから「Blackmagic Shield for Arduino」を開きます。

2   「Library」フォルダーを開いて、「BMDSDIControl」というフォルダーを右クリックでコピーします。

3   コンピューターの「Documents」フォルダーへ行き、Arduinoフォルダーを開きます。

4   「Libraries」という名前のサブフォルダーがあるので、そこに「BMDSDIControl」フォルダーをペーストします。

Windowsでライブラリファイルをインストール：

1   Programs/Blackmagic Shield for Arduinoフォルダーを開きます。

2   「Libraries」という名前のサブフォルダーがあるので、「BMDSDIControl」というフォルダーを右クリックでコピーします。

3   コンピューターの「Documents」フォルダーへ行き、Arduinoフォルダーを開きます。

4   「Libraries」という名前のサブフォルダーがあるので、そこに「BMDSDIControl」フォルダーをペーストします。

これで、Blackmagic Designライブラリファイルをコンピューターにインストールできました。Arduinoソフトウェアを起動するとBlackmagic Designのスケッチ例を選択できるようになります。

Arduinoソフトウェアのメニューバーから「File」ドロップダウンメニューへ行き、「Examples」を選択します。次に「BMDSDIControl」を選択すると、使用可能なスケッチ例のリストが表示されます。

ライブラリファイルが適切なフォルダーに保存されていれば、シールドはこれらのファイルを使用してArduinoボードと通信できます。必要な作業はArduino IDEソフトウェアのプログラム作成のみです。詳細は、「Arduinoスケッチのプログラミング」セクションを参照してください。

> **メモ** 将来、アップデートされたライブラリファイルがリリースされた場合、古い「BMDSDIControl」フォルダーを削除し、上記に記載された方法で新しいフォルダーに置き換える必要があります。

Blackmagic Shield for Arduino Setup
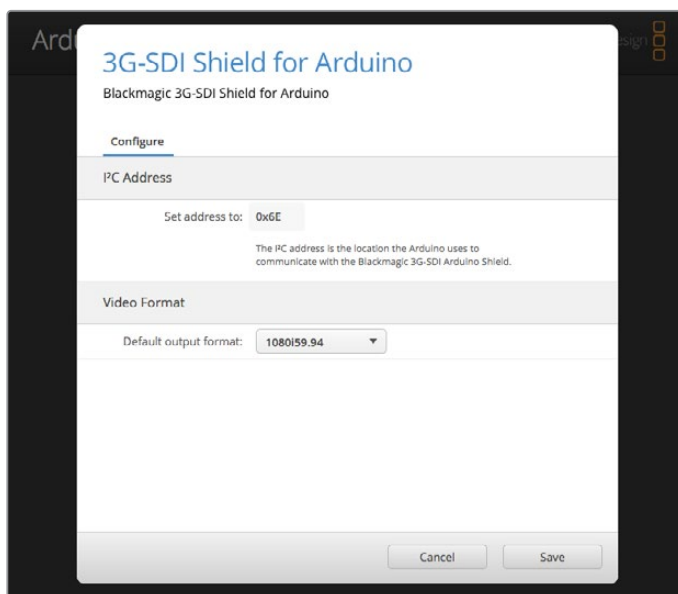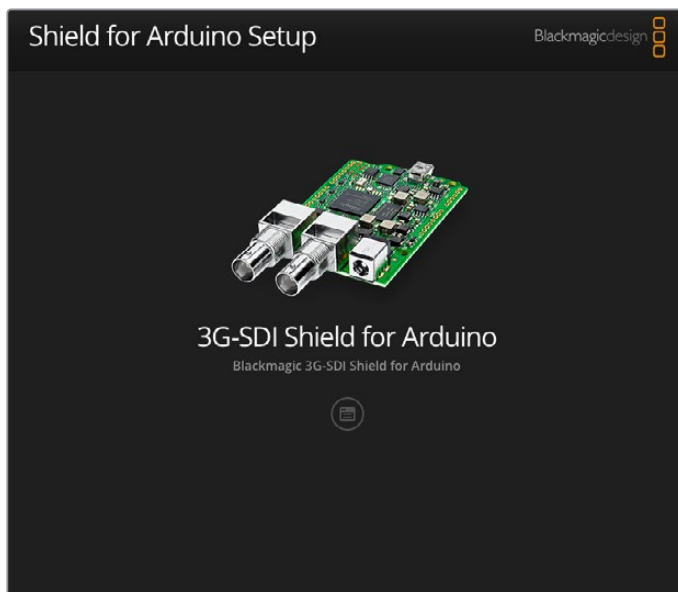
Blackmagic Shield for Arduino Setupソフトウェアを使って、I2Cアドレスや
ビデオ出力フォーマットなど、シールドの設定を変更できます。。

Blackmagic Shield for Arduino Setupをコンピューターにインストールしていれば、シールド設定を変更
できます。これには、シールドを特定してArduinoボードと通信可能にする「I2Cアドレス」や、シールドの出力
フォーマットを設定する「ビデオフォーマット」などの設定が含まれます

I²Cアドレス

ごく稀に、Blackmagicシールドにマウントした別のシールドが、シールドのデフォルトアドレスと同一の I²Cア
ドレスを共有しており、問題が発生するケースがあります。この場合、シールドのデフォルトアドレス設定を変更
できます。

シールドのデフォルトアドレスは0x6Eですが、0x08から0x77までの範囲でアドレスを選択できます。

シールドのアドレスを変更：

1   Blackmagic Shield for Arduino Setupを起動し、シールドの「Settings」アイコンをクリックします。

2   「Set address to:」の編集ボックスで 使用したいアドレスを入力します。

3   「Save」をクリックします。

ビデオフォーマット

入力が接続されていない場合、デフォルトの出力フォーマットは、セットアップユーティリティで選択されます。入力が検出されると、出力は入力フォーマットと同じになります。入力が途切れると、出力はユーティリティで選択したデフォルト出力フォーマットに戻ります。「Default output format」のドロップダウンメニューをクリックして使用したいフォーマットを選択すればビデオフォーマットを変更できます。

以下のビデオ出力フォーマットから選択できます：

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Arduinoスケッチのプログラミング

Arduinoソフトウェアのプログラム、あるいはスケッチは簡単に書き込みできます。スケッチは共通の「C」プログラミング言語を使用して書き込まれます。Studio Camera Control Protocolからのコマンドを使用してスケッチをプログラミングする際、同シールドはこれらのコマンドをSDI出力にエンベッドし、Blackmagic URSA MiniあるいはBlackmagic Studio Cameraをコントロールできるようになります。

すべての対応コマンドは、同マニュアルのStudio Camera Control Protocolセクションに記載されています。プロトコルからコマンドを取り出してスケッチに使用してください。

Blackmagic Shieldのテストとライブラリのインストール

「はじめに」セクションに記載されている通りにすべての接続が完了し、セットアップソフトウェアおよびライブラリファイルをインストールしたら、シールドがArduinoボードと通信可能となっておりすべてが順調に動作しているかどうかを確認します。

一番スピーディな方法は、タリー点滅のスケッチ例を開いて使用してみることです。

以下の手順に従います：

1. Arduino IDEソフトウェアを起動する。
2. 「Tools」メニューへ行き、Arduinoボードとポート番号を選択します。
3. 「File」メニューから「Examples/BMDSDIControl」を選択し、「TallyBlink」という名前のスケッチを選択します。
4. ボードにスケッチをアップロードします。


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                  // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);     // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                  // initialize tally control
  sdiTallyControl.setOverride(true);        // enable tally override

  pinMode(13, OUTPUT);                      // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                   // turn the LED ON

  sdiTallyControl.setCameraTally(           // turn tally ON
    1,                                      // Camera Number
    true,                                   // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it ON for 1 second

  digitalWrite(13, LOW);                    // turn the LED OFF

  sdiTallyControl.setCameraTally(           // turn tally OFF
    1,                                      // Camera Number
    false,                                  // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it OFF for 1 second
}
```

```
No changes necessary for Auto Format.
```

```
Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.
```

タリー点滅のスケッチ例は、最もスピーディかつ簡単にBlackmagic 3G-SDI Shield for Arduinoシールドのテストが可能です。RAWデータは、Studio Camera Protocolドキュメントからのコマンドを使用して、I2C経由でシールドに送信されますが、簡単にスケッチをプログラミングできるよう、カスタムライブラリも提供しています。

これでBlackmagic Studio Cameraのタリーライトが1秒に1度点滅するはずです。タリーライトの点滅が確認できたら、BlackmagicシールドがArduinoと通信できており、すべてが正常に動作しているということです。

タリーが点滅しない場合、Blackmagicカメラのタリー番号が1に設定されているか、そしてカメラのグリッドオーバーレイがオンになっているかを確認してください。

サポートやアドバイスが必要な場合は、Blackmagic Designのサポートセンター（www.blackmagicdesign.com/jp/support）をご利用ください。シールドの設定に関するサポートの詳細は、同マニュアルの「ヘルプ」セクションを参照してください。

LEDインジケーター

Blackmagic 3G-SDI Shield for Arduinoには6つのインジケーターLEDが付いており、電源、UART、I2C、SPI通信シールドなどのアクティビティを確認できます。さらにタリーおよびカメラコントロールのオーバーライドが有効になっていることを示すインジケーターがあります。
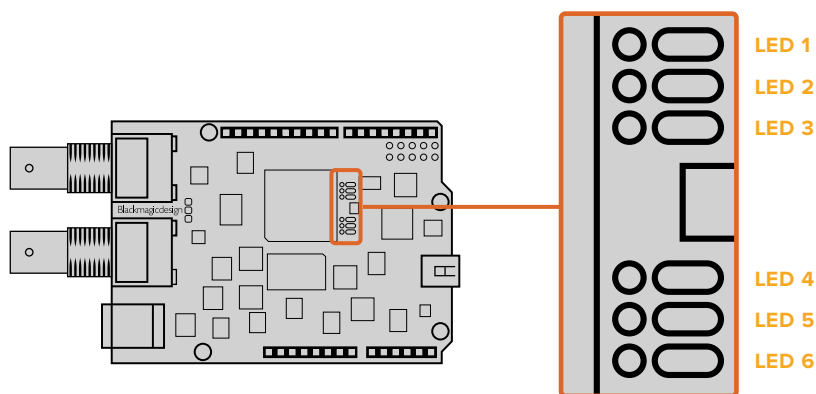
LED 1 - システム・アクティブ
電源がシールドに接続されている時に光ります。

LED 2 - コントロールオーバーライド有効
Arduinoスケッチでカメラコントロールを有効にすると光ります。

LED 3 - タリーオーバーライド有効
Arduinoスケッチでタリーを有効にすると光ります。

LED 5 - I^{2}C パーサ使用中
シールドとArduinoの間でI2Cプロトコルを使用した通信が検出されると光ります。

LED 6 - シリアルパーサ使用中
UART通信が検出されると光ります。

Blackmagicシールドのブート中、電源インジケーターはオフのままで、LED3、4、5は以下のアクティビティが行われていることを意味します。

LED 3 - アプリケーションイメージのロード

LED 4 - EEPROMの初期化

LED 5 - メモリーチェック処理中

ブートが適切に終了したら電源LEDが光り、すべてのLEDが操作中の通常機能に戻ります。

ごく稀にブートに失敗した場合は、失敗したアクティビティ以外のすべてのLEDが高速点滅するので失敗の原因が分かります。

シールドコンポーネントの取り付け

独自のハードウェアコントローラーを構築したい場合、ボタン、ノブ、ジョイスティックなどを使い、より触覚的かつ実践的な新しいシールドを作成できます。カスタムシールドをヘッダースロットに接続して、Blackmagic 3G-SDI Shield for Arduinoにマウントします。作成するコントローラーのタイプに制限はありません。古いCCUの回路を独自のカスタムArduinoソリューションと交換して、業界標準のカメラコントロールユニットを作成することもできます。
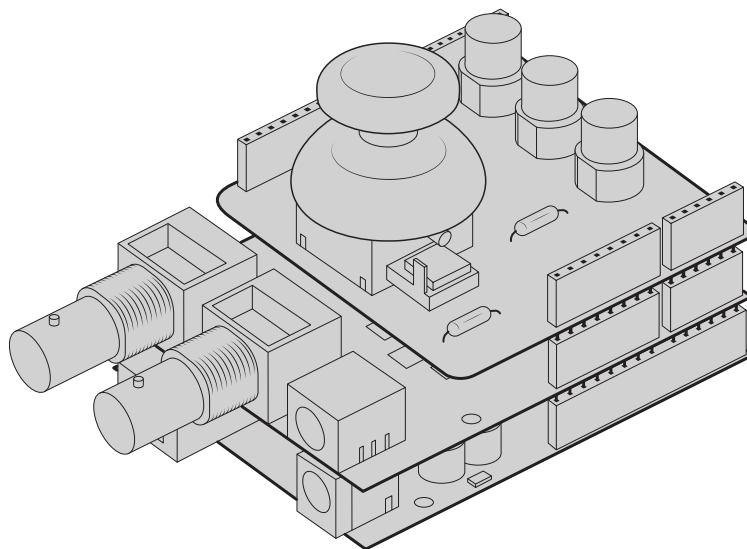
独自のハードウェアコントローラーを作成し、Blackmagic 3G-SDI Shield for Arduino に接続して、よりインタラクティブで精密なコントロールが行えます。。

Communicating with your Blackmagic Shield for Arduino

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I2C or Serial. We recommend I2C because of the low pin count and it frees up the serial monitor. This also allows you to use more I2C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I^{2}C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int              shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial     sdiCameraControl;
BMD_SDITallyControl_Serial      sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int              shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^{11}. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides. Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	–	–	BCD - HHMMSSFF (UCT)
				[1] date	–	–	BCD - YYYYMMDD
	7.1	System language	string	–	–	–	ISO-639-1 two character language code
	7.2	Timezone	int32	–	–	–	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	–	–	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	–	–	BCD - sDDDddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header		command				data									
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.	Standard mode (100 kbit/s)
2.	Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
-------------------- |----------------------------
 A4                  | Serial Data (SDA)
 A5                  | Serial Clock (SCL)
```

I²C Protocol (Writes):

	(START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

	(START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
-------------------- |----------------------------
 IO1                 | Serial Transmit (TX)
 IO0                 | Serial Receive (RX)
```

UART Protocol (Writes):

	[0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

	[0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name        | R/W    | Register Description
-------------------- |----------   |-----   |------------------------------

 0x0000 - 0x0003     | IDENTITY    | R      | Hardware Identifier
 0x0004 - 0x0005     | HWVERSION   | R      | Hardware Version
 0x0006 - 0x0007     | FWVERSION   | R      | Firmware Version
                     |             |        |
 0x1000              | CONTROL     | R/W    | System Control
                     |             |        |
 0x2000              | OCARM       | R/W    | SDI Control Override Arm
 0x2001              | OCLENGTH    | R/W    | SDI Control Override Length
 0x2100 - 0x21FE     | OCDATA      | R/W    | SDI Control Override Data
                     |             |        |
 0x3000              | ICARM       | R/W    | SDI Control Incoming Arm
 0x3001              | ICLENGTH    | R      | SDI Control Incoming Length
 0x3100 - 0x31FE     | ICDATA      | R      | SDI Control Incoming Data
                     |             |        |
 0x4000              | OTARM       | R/W    | SDI Tally Override Arm
 0x4001              | OTLENGTH    | R/W    | SDI Tally Override Length
 0x4100 - 0x41FE     | OTDATA      | R/W    | SDI Tally Override Data
                     |             |        |
 0x5000              | ITARM       | R/W    | SDI Tally Incoming Arm
 0x5001              | ITLENGTH    | R      | SDI Tally Incoming Length
 0x5100 - 0x51FE     | ITDATA      | R      | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

```
[ IDENTITY ]
31        0
```

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7           0
```

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7           0
```

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

```
[ RESERVED ] [ OVERRIDE OUTPUT ] [ RESET TALLY ] [ OVERRIDE TALLY ] [
OVERIDE CONTROL ]
7       4           3            2             1              0
```

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_
was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from
the next video frame until new data is set. Automatically cleared once
the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one
byte per camera). Bit zero indicates a Program tally, while bit one
indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame.
Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a
new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was
reset (one byte per camera). Bit zero indicates a Program tally, while
bit one indicates a Preview tally.

ヘルプ

ヘルプライン

Blackmagic 3G-SDI Shield for Arduinoは、ユーザーの要望に基づいて単独で開発できるよう設計された
デベロッパーツールです。

シールドに関する最新の情報が欲しい方は、Blackmagic Designオンラインサポートページで、最新サポート情報を確認できます。

Blackmagic Design オンラインサポートページ

最新のマニュアル、ソフトウェア、サポートノートは、www.blackmagicdesign.com/jp/supportのBlackmagic
サポートセンターで確認できます。

Arduino開発フォーラム

プログラミングに関する質問がある方は、インターネットのArduino開発フォーラムで助けを得られます。
Arduinoデベロッパーのコミュニティは幅広く、ソフトウェアに関する質問ができる良質なフォーラムが数多く
存在します。また、エンジニアを雇ってソリューションを作成してもらうこともできるでしょう。

Blackmagic Designフォーラム

弊社ウェブサイトのBlackmagic Designフォーラムは、様々な情報やクリエイティブなアイデアを共有できる
有益なリソースです。経験豊富なユーザーやBlackmagic Designスタッフによって、すでに多くの問題の解決
策が公開されていますので、このフォーラムを参考にすることで、現在の問題をすばやく解決できることがあり
ます。ぜひご利用ください。Blackmagicフォーラムには、http://forum.blackmagicdesign.com からアクセ
スできます。

現在インストールされているソフトウェアのバージョンを確認

コンピューターにインストールされているBlackmagic 3G-SDI Shield for Arduino Setupソフトウェアのバー
ジョンを確認するには、「About Blackmagic 3G-SDI Shield for Arduino Setup」ウィンドウを開きます。

- Mac OS Xでは、アプリケーションフォルダーから「Blackmagic 3G-SDI Shield for Arduino Setup」
  を開きます。アプリケーションメニューから「About Blackmagic Shield for Arduino Setup」を
  選択し、バージョンを確認します。
- Windows 7では、スタートメニューから「Blackmagic 3G-SDI Shield for Arduino Setup」を開き
  ます。ヘルプメニューをクリックして「About Blackmagic Shield for Arduino Setup」を選択し、
  バージョンを確認します。
- Windows 8では、スタートページの「Blackmagic 3G-SDI Shield for Arduino Setup」タイルから
  Blackmagic 3G-SDI Shield for Arduino Setupを開きます。ヘルプメニューをクリックして「About
  Blackmagic Shield for Arduino Setup」を選択し、バージョンを確認します。

最新のソフトウェアアップデートを入手する

コンピューターにインストールされたBlackmagic 3G-SDI Shield for Arduino Setupソフトウェアのバー
ジョンを確認した後、Blackmagicサポートセンター（www.blackmagicdesign.com/jp/support）
で最新のソフトウェアアップデートを確認してください。常に最新のソフトウェアを使用することを推奨します
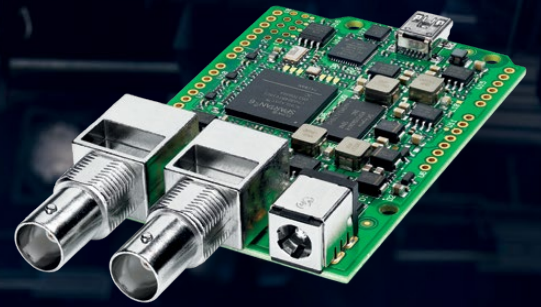が、重要なプロジェクトの実行中は、ソフトウェアのアップデートは行わない方がよいでしょう。

保証

12ヶ月限定保証

Blackmagic Designは、お買い上げの日から12ヶ月間、Blackmagic 3G-SDI Shield for Arduinoの部品および仕上がりについて瑕疵がないことを保証します。この保証期間内に製品に瑕疵が見つかった場合、Blackmagic Designは弊社の裁量において部品代および人件費無料で該当製品の修理、あるいは製品の交換のいずれかで対応いたします。

この保証に基づいたサービスを受ける際、お客様は必ず保証期限終了前にBlackmagic Designに瑕疵を通知し、適応する保証サービスの手続きを行ってください。お客様の責任において不良品を梱包し、Blackmagic Designが指定するサポートセンターへ配送料前払で送付いただきますようお願い致します。理由の如何を問わず、Blackmagic Designへの製品返送のための配送料、保険、関税、税金、その他すべての費用はお客様の自己負担となります。

不適切な使用、または不十分なメンテナンスや取扱いによる不具合、故障、損傷に対しては、この保証は適用されません。Blackmagic Designはこの保証で、以下に関してサービス提供義務を負わないものとします。 a) 製品のインストールや修理、サービスを行うBlackmagic Design販売代理人以外の者によって生じた損傷の修理、b) 不適切な使用や互換性のない機器への接続によって生じた損傷の修理、c) Blackmagic Designの部品や供給品ではない物を使用して生じたすべての損傷や故障の修理、d) 改造や他製品との統合により時間増加や製品の機能低下が生じた場合のサービス。この保証は Blackmagic Designが保証するもので、明示または黙示を問わず他の保証すべてに代わるものです。Blackmagic Designとその販売社は、商品性と特定目的に対する適合性のあらゆる黙示保証を拒否します。Blackmagic Designの不良品の修理あるいは交換の責任が、特別に、間接的、偶発的、または結果的に生じる損害に対して、Blackmagic Designあるいは販売社がそのような損害の可能性についての事前通知を得ているか否かに関わらず、お客様に提供される完全唯一の救済手段となります。Blackmagic Designはお客様による機器のあらゆる不法使用に対して責任を負いません。Blackmagic Designは本製品の使用により生じるあらゆる損害に対して責任を負いません。使用者は自己の責任において本製品を使用するものとします。

Manuel de la Blackmagic 3G-SDI Shield for Arduino

Cher client, chère cliente,

Nous vous remercions d'avoir fait l'acquisition d'une carte Blackmagic 3G-SDI Shield for Arduino.

Nous nous intéressons aux nouvelles technologies et souhaitons développer des façons innovantes d'utiliser nos produits SDI. Grâce à la 3G-SDI Shield for Arduino, vous pouvez désormais intégrer une carte Arduino à votre workflow SDI et ainsi ajouter des options de contrôle à votre équipement Blackmagic Design.

Par exemple, les mélangeurs ATEM peuvent contrôler la Blackmagic URSA Mini et les Blackmagic Studio Cameras via des paquets de données intégrés au signal SDI. Si vous n'utilisez pas de mélangeur ATEM, mais que vous souhaitez pouvoir contrôler vos caméras Blackmagic, vous pouvez créer des solutions de contrôle personnalisées à l'aide de la 3G-SDI Shield for Arduino. Cette carte vous offre une plateforme de création, qui vous permet d'envoyer le signal de retour du programme provenant du mélangeur vers la carte, puis vers l'entrée de programme de votre caméra Blackmagic.

Il est très facile d'écrire un code pour envoyer des commandes à la caméra. De plus, toutes les commandes prises en charge sont décrites dans ce manuel.

Vous pouvez contrôler les caméras à l'aide d'un ordinateur ou ajouter des boutons, molettes et joysticks à votre carte. Vous disposerez ainsi de solutions de contrôle dynamiques qui permettent de modifier des fonctionnalités telles que la mise au point, le zoom, l'ouverture, le niveau de noir, la balance des blancs, le correcteur de couleurs intégré à la caméra et autres. Créer sa propre solution de contrôle est utile à la production, mais aussi très amusant !

Nous sommes très heureux de pouvoir proposer cette technologie et nous avons hâte de découvrir les solutions de contrôle SDI que vous avez conçues avec la 3G-SDI Shield for Arduino !

Ce manuel d'utilisation comprend toutes les informations dont vous avez besoin pour utiliser la Blackmagic 3G-SDI Shield for Arduino. Consultez notre page d'assistance sur www.blackmagicdesign.com/fr pour obtenir la dernière version du manuel et les mises à jour du logiciel interne de la carte. Nous vous recommandons de mettre le logiciel à jour régulièrement afin de travailler avec les fonctions les plus récentes. N'oubliez pas d'enregistrer vos coordonnées lorsque vous téléchargerez le logiciel afin d'être informé des dernières mises à jour. Nous souhaitons continuer à améliorer nos produits, n'hésitez donc pas à nous faire part de vos commentaires !

Grant Petty
PDG de Blackmagic Design

Sommaire

# Manuel de la Blackmagic 3G-SDI Shield for Arduino

Mise en route

Fixer et souder les barrettes

La Blackmagic 3G-SDI Shield for Arduino est fournie avec 4 barrettes, dont deux barrettes mâles à 8 broches, une à 10 broches et une à 6 broches. Les barrettes sont des connecteurs qui permettent d'installer votre carte extension sur l'Arduino. Comme elles se superposent, vous pouvez installer d'autres cartes les unes sur les autres ainsi que des composants, notamment des boutons de contrôle, des molettes et des joysticks. Ces barrettes sont conçues pour être installées sur les cartes Arduino R3, telles que l'Arduino UNO.

Pour fixer les barrettes à votre carte extension :

1   Insérez les broches de chaque barrette dans les trous correspondants situés de part et d'autre de la Blackmagic 3G-SDI Shield. L'illustration ci-dessous vous indique la disposition des barrettes.
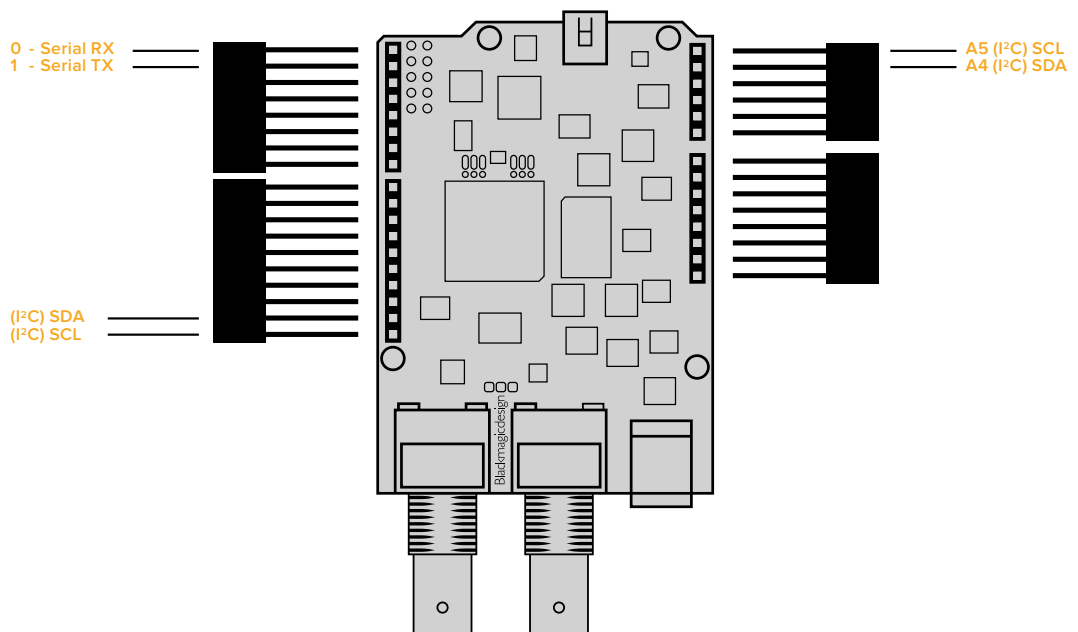
REMARQUE Lors de la connexion à la carte extension, la communication est établie via le protocole I^{2}C ou le protocole série. Nous recommandons de choisir I^{2}C afin de pouvoir utiliser le moniteur série et les autres broches. Sélectionnez le mode de communication lorsque vous définissez l'objet BMDSDIControl dans le sketch. Pour plus d'informations, consultez la section « Communication avec la Blackmagic 3G-SDI Shield for Arduino » de ce manuel.

2   Soudez la base de chaque broche à la face inférieure de votre carte extension. Veillez à ce que chaque broche soit solidement raccordée au trou correspondant, sans entrer en contact avec la soudure des autres broches.

Afin que toutes les broches de la carte extension soient bien alignées avec les trous de la barrette de l'Arduino, nous vous recommandons, dans un premier temps, de ne souder qu'une broche sur chaque barrette. Placez ensuite la carte extension sur l'Arduino pour vérifier l'alignement des broches. Si certaines barrettes ont besoin d'être ajustées, vous pouvez réchauffer la soudure et ainsi améliorer leur alignement. C'est bien plus facile que de souder toutes les broches et d'essayer de les ajuster par la suite.

Installation sur la carte Arduino

Une fois les barrettes soudées sur votre carte extension, vous pouvez l'installer sur la carte Arduino.

Saisissez les deux côtés de la carte extension et alignez les broches avec les barrettes de l'Arduino. Poussez délicatement les broches dans les trous des barrettes. Veillez à ne pas plier les broches lors de l'installation.
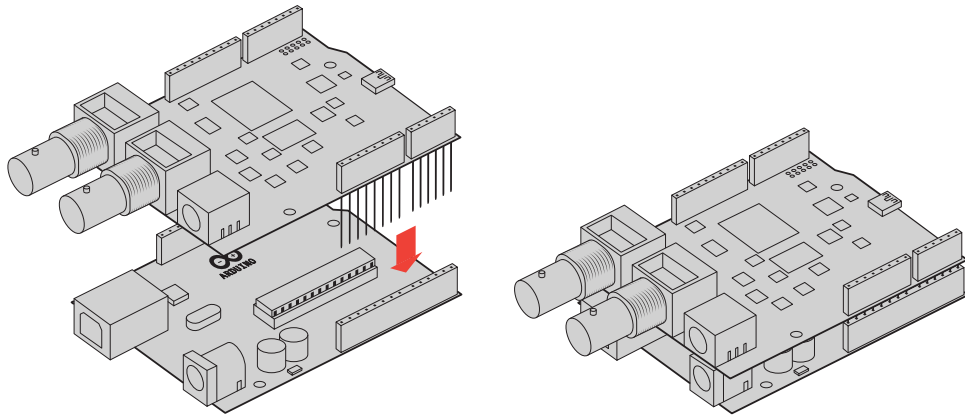
Une fois les broches enfoncées, la carte extension Blackmagic et la carte Arduino doivent être fermement connectées.

Branchement de l'alimentation

Pour alimenter votre Blackmagic 3G-SDI Shield for Arduino, il suffit de brancher un adaptateur 12V à l'entrée d'alimentation 12V de la carte extension Blackmagic.

REMARQUE L'alimentation de la carte Arduino n'est pas suffisante pour les deux cartes. En revanche, si vous alimentez la carte Blackmagic, la carte Arduino sera également alimentée. Veillez donc à n'alimenter que la carte Blackmagic.

Connexion à du matériel SDI

Une fois la carte alimentée, vous pouvez la connecter à du matériel SDI. Par exemple, un mélangeur et une Blackmagic URSA Mini :

1 Branchez la sortie programme du mélangeur à l'entrée SDI de la Blackmagic 3G-SDI Shield.

2 Branchez la sortie SDI de la Blackmagic 3G-SDI Shield à l'entrée programme SDI (PMG) de la Blackmagic URSA Mini.

Vous trouverez ci-dessous un schéma de connexion.

Mélangeur

Entrée SDI

Sortie SDI

Blackmagic 3G-SDI Arduino Shield

Entrée SDI PGM

Blackmagic URSA Mini
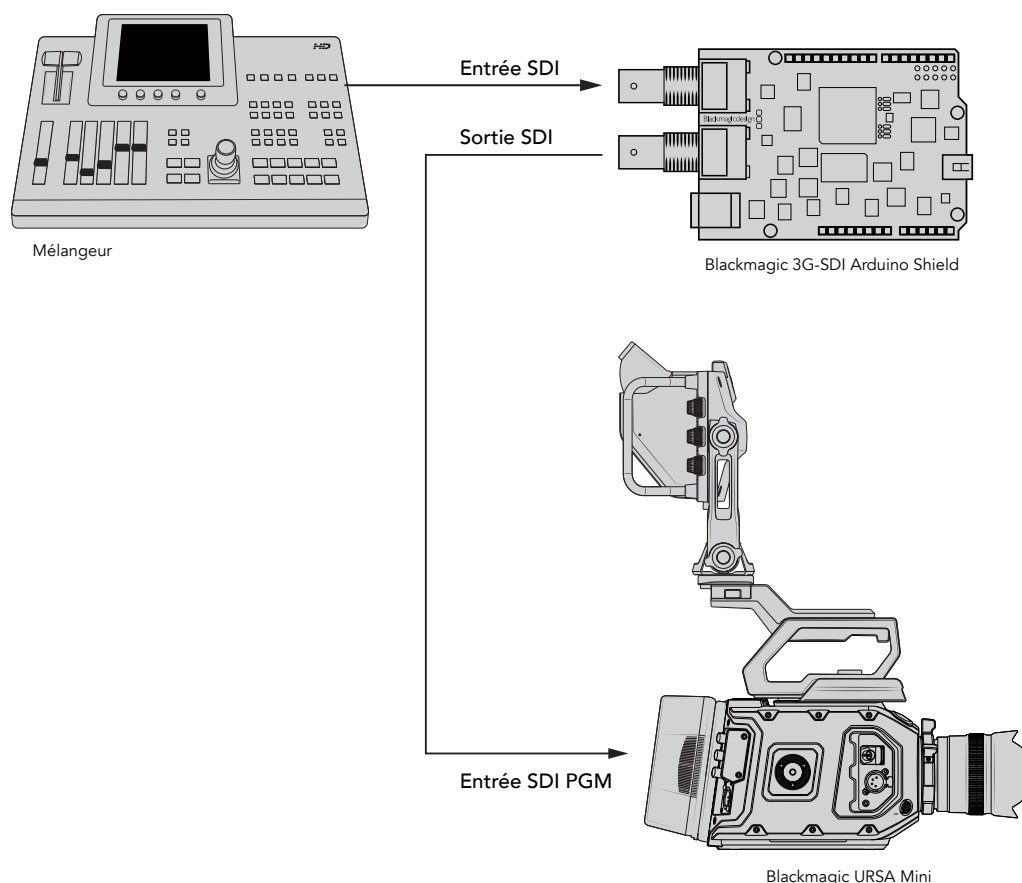
C'est tout ce que vous devez savoir pour démarrer !

Maintenant que votre carte est montée sur l'Arduino, qu'elle est alimentée et connectée à du matériel SDI, vous pouvez installer le logiciel interne et les fichiers bibliothèque.

Consultez le reste du manuel pour obtenir plus d'informations sur l'installation du logiciel interne de la carte extension et sur l'installation des fichiers bibliothèque afin que la carte extension puisse communiquer avec l'Arduino.

> **CONSEIL** Vous pouvez également utiliser la Blackmagic 3G-SDI Shield for Arduino pour contrôler d'autres produits Blackmagic Design, tels que le Blackmagic MultiView 16. Par exemple, lorsque la carte est connectée à l'entrée 16 de l'appareil, une bordure tally s'affiche sur le multivue. Lorsque vous utilisez des caméras Blackmagic URSA Mini, veillez à activer la grille afin d'afficher le tally sur le viseur de la caméra. Pour plus d'informations, consultez le manuel des caméras Blackmagic.

Installation du logiciel

Installation du logiciel interne

Le Blackmagic Shield for Arduino Setup permet de mettre à jour le logiciel interne de la carte extension. Le logiciel interne communique avec l'Arduino et contrôle la carte à l'aide des fichiers bibliothèque Arduino. Ces fichiers bibliothèque sont installés avec le logiciel d'installation. Il suffit donc de copier le dossier contenant les fichiers et de le coller dans le dossier application de l'Arduino. Pour plus d'informations concernant l'installation des fichiers bibliothèque, consultez la section suivante du manuel.

Nous vous recommandons de télécharger la dernière version du logiciel Blackmagic Shield for Arduino et de mettre la carte à jour afin de bénéficier des nouvelles fonctionnalités et améliorations. Vous pouvez télécharger la dernière version de ce manuel sur la page d'assistance de Blackmagic Design www.blackmagicdesign.com/fr/support.

Pour installer le logiciel interne avec Mac OS X :

1   Téléchargez et dézippez le logiciel Blackmagic Shield for Arduino.

2   Ouvrez l'image disque et lancez le programme d'installation Blackmagic Shield for Arduino. Suivez les instructions affichées à l'écran.

3   Après avoir installé la dernière version du programme d'installation Blackmagic Shield for Arduino, alimentez la carte extension Blackmagic et connectez-la à votre ordinateur à l'aide d'un câble USB.

4   Lancez le programme d'installation et suivez les informations affichées à l'écran pour mettre à jour le logiciel interne de la carte extension. Si aucune mise à jour n'apparaît, le logiciel interne est à jour.

Pour installer le logiciel interne avec Windows :

1   Téléchargez et dézippez le logiciel Blackmagic Shield for Arduino.

2   Le dossier Blackmagic Shield for Arduino s'affiche, il contient le manuel et le programme d'installation Blackmagic Shield for Arduino. Double-cliquez sur le programme d'installation et suivez les instructions à l'écran pour terminer l'installation.

3   Après avoir installé la dernière version du programme d'installation Blackmagic Shield for Arduino, alimentez la carte extension Blackmagic et connectez-la à votre ordinateur à l'aide d'un câble USB.

4   Lancez le programme d'installation et suivez les informations affichées à l'écran pour mettre à jour le logiciel interne de la carte extension. Si aucune mise à jour n'apparaît, le logiciel interne est à jour.

Installation des fichiers bibliothèque Arduino

Les programmes conçus pour contrôler l'Arduino sont appelés croquis ou sketches. La Blackmagic 3G-SDI Shield for Arduino utilise des fichiers bibliothèque Arduino pour faciliter l'écriture des sketches. Après avoir installé le logiciel de la carte extension, les fichiers bibliothèque sont installés dans un dossier intitulé **Library**. Il suffit à présent de copier le dossier contenant ces fichiers et de le coller dans le dossier bibliothèque de l'Arduino.

> **REMARQUE** Le logiciel IDE Arduino doit être fermé lorsque vous installez les bibliothèques.

Pour installer les fichiers bibliothèque sur Mac OS X :

1   Ouvrez **Blackmagic Shield for Arduino** dans le dossier **Applications**.

2   Ouvrez le dossier **Library** et faites un clic droit/copiez le dossier intitulé : BMDSDIControl.

3   Dans le dossier **Documents** de votre ordinateur, ouvrez le dossier Arduino.

4   Vous verrez un sous-dossier intitulé **Libraries**. Collez le dossier BMDSDIControl dans le dossier **Libraries**.

Pour installer les fichiers bibliothèque sur Windows :

1   Ouvrez **Programmes/ Blackmagic Shield for Arduino**.

2   Vous verrez un sous-dossier intitulé **Library**. Ouvrez ce dossier et faites un clic droit/copiez le dossier intitulé : BMDSDIControl.

3   Dans le dossier **Documents** de votre ordinateur, ouvrez le dossier Arduino.

4   Vous verrez un sous-dossier intitulé **Libraries**. Collez le dossier BMDSDIControl dans le dossier **Libraries**.

C'est tout ce que vous devez faire pour installer les fichiers bibliothèque Blackmagic Design sur votre ordinateur. Lorsque vous utilisez le logiciel Arduino, vous disposerez désormais d'exemples de sketches Blackmagic Design.

Dans le menu déroulant **Fichier** du logiciel Arduino, sélectionnez **Exemples**. Sélectionnez BMDSDIControl pour afficher la liste d'exemples de sketches que vous pouvez utiliser.

Une fois les fichiers bibliothèque stockés dans le dossier approprié, la carte extension peut les utiliser pour communiquer avec l'Arduino. Il ne reste plus qu'à programmer le logiciel IDE Arduino. Pour plus d'informations à ce sujet, consultez la section « Programmation des sketches Arduino » de ce manuel.

> **REMARQUE** Si une mise à jour de la bibliothèque comportant des exemples est disponible, vous devrez supprimer l'ancien dossier BMDSDIControl et le remplacer par le nouveau dossier en suivant les étapes décrites ci-dessus.

Blackmagic Shield for Arduino Setup
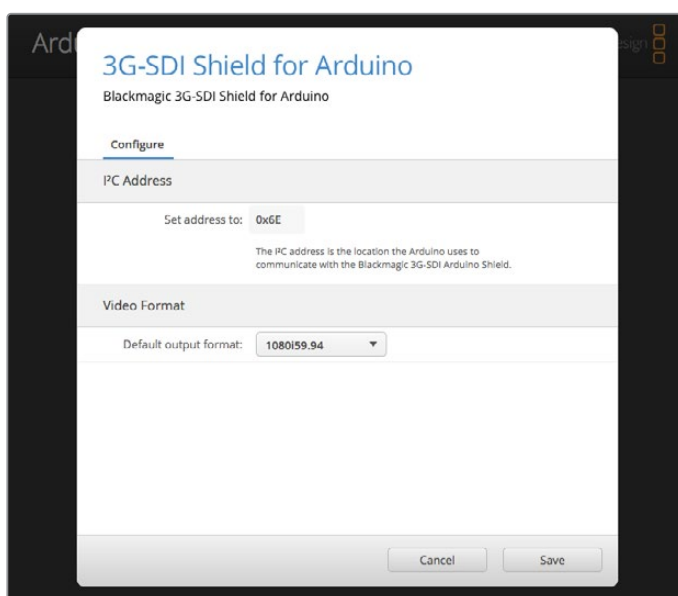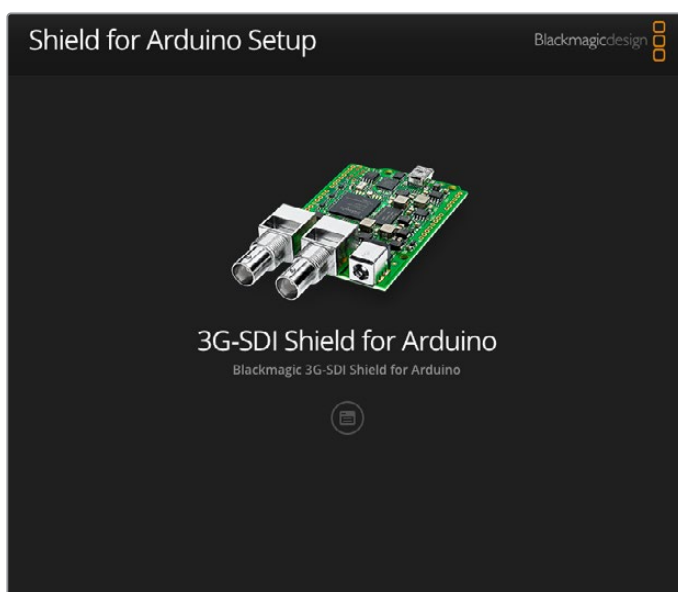
Le logiciel Blackmagic Shield for Arduino Setup vous permet de changer les paramètres de la carte extension, notamment l'adresse I²C et le format de sortie vidéo.

Grâce au logiciel Blackmagic Shield for Arduino Setup installé sur votre ordinateur, vous pouvez changer les paramètres de la carte extension : **I²C address**, permet de reconnaître la carte extension afin que l'Arduino puisse communiquer avec elle ; **Video Format**, permet de régler le format de sortie de la carte extension.

I²C Address

Exceptionnellement, il se peut qu'une autre carte extension montée sur la carte extension Blackmagic partage l'adresse I²C par défaut, ce qui peut créer un conflit. Si cela ce produit, vous pouvez changer l'adresse par défaut de la carte extension.

L'adresse par défaut pour la carte extension est 0x6E, cependant, vous pouvez choisir des adresses comprises entre 0x08 et 0x77.

Changer l'adresse de la carte extension :

1 Lancez le Blackmagic Shield for Arduino Setup et cliquez sur l'icône de paramétrage.

2 Dans le paramètre **Set address to:**, saisissez l'adresse que vous souhaitez utiliser.

3 Cliquez sur **Save**.

Format vidéo

Si aucun signal d'entrée n'est connecté, le format de sortie par défaut est choisi par le logiciel d'installation. Lorsqu'un signal d'entrée est connecté, la sortie est configurée au même format que celui de l'entrée. Si l'entrée est déconnectée, le format de sortie par défaut sera choisi par le logiciel. Vous pouvez changer le format vidéo dans le menu déroulant **Default output format**.

Les formats vidéo suivants sont disponibles :

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Programmation des sketches Arduino

Les programmes Arduino, également appelés croquis ou sketches, sont très faciles à écrire. Ces sketches sont programmés en langage C. Lorsque vous programmez les sketches avec les commandes du Studio Camera Control Protocol, la carte extension intègre ces commandes à la sortie SDI qui permet de contrôler la Blackmagic URSA Mini ou les Blackmagic Studio Camera.

Toutes les commandes prises en charge sont incluses dans la section « Studio Camera Control Protocol » de ce manuel. Vous pouvez donc les utiliser pour vos sketches.
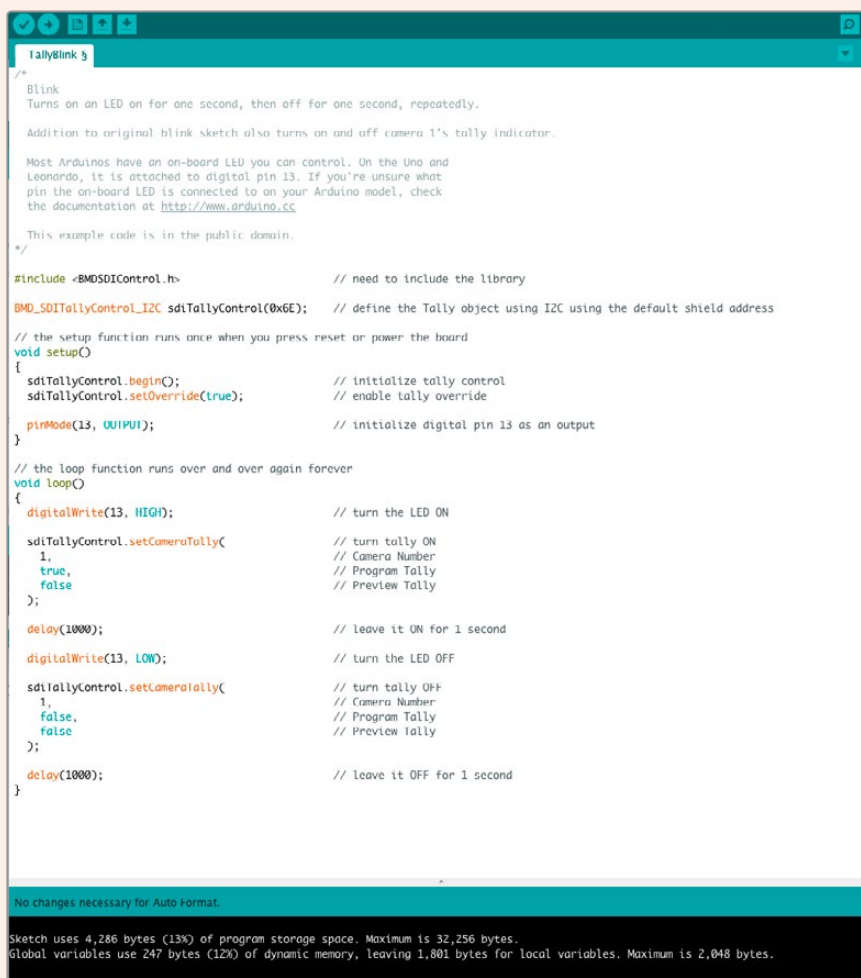
Test de la carte extension Blackmagic et installation de la bibliothèque

Une fois la carte extension connectée comme indiqué dans la section « Mise en route », et le logiciel et les fichiers bibliothèque installés, veuillez vérifier que la carte extension communique correctement avec la carte Arduino.

Pour vérifier cela, ouvrez et exécutez l'exemple de sketch TallyBlink présenté ci-dessous.

Suivez les étapes suivantes :

1 Lancez le logiciel IDE Arduino.

2 Allez dans le menu **Outils** et sélectionnez le type de carte Arduino et le port.

3 Dans le menu **Fichier**, sélectionnez **Exemples/BMDSDIControl** et choisissez le sketch appelé **TallyBlink**.

4 Chargez le sketch sur la carte.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                  // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);  // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                 // initialize tally control
  sdiTallyControl.setOverride(true);       // enable tally override

  pinMode(13, OUTPUT);                      // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                   // turn the LED ON

  sdiTallyControl.setCameraTally(           // turn tally ON
    1,                                      // Camera Number
    true,                                   // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it ON for 1 second

  digitalWrite(13, LOW);                    // turn the LED OFF

  sdiTallyControl.setCameraTally(           // turn tally OFF
    1,                                      // Camera Number
    false,                                  // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it OFF for 1 second
}
```

No changes necessary for Auto Format.

Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.

L'exemple de sketch Tally Blink est simple à réaliser pour tester votre Blackmagic 3G-SDI Shield for Arduino. Les données brutes peuvent être envoyées à la carte via I2C à l'aide des commandes du Studio Camera Protocol. Nous avons également inclus des bibliothèques personnalisées pour rendre la programmation de sketches encore plus facile.

Vous devriez voir le voyant tally de la Blackmagic Studio Camera clignoter une fois par seconde. Si le voyant tally clignote, cela signifie que la carte extension Blackmagic communique avec la carte Arduino et que tout fonctionne correctement.

Si le voyant ne clignote pas, vérifiez que le numéro du tally de la caméra est bien réglé sur 1 et que la grille de la caméra est activée.

Si vous avez besoin d'aide, consultez la page d'assistance technique Blackmagic Design à l'adresse suivante www.blackmagicdesign.com/fr/support. Veuillez lire la section d'assistance de ce manuel pour obtenir davantage d'informations sur le fonctionnement de votre carte extension.

Voyants LED

La Blackmagic 3G-SDI Shield for Arduino comprend six voyants LED qui permettent de confirmer les activités de la carte extension, par exemple, l'alimentation, la liaison UART ainsi que la communication I^{2}C et SPI. Les voyants indiquent également lorsque la prise de contrôle manuel du tally et des commandes de la caméra est activée.
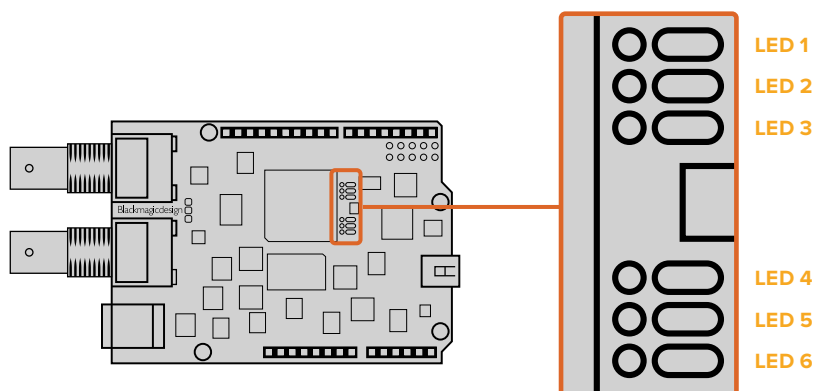
LED 1 - Système actif
S'allume lorsque l'alimentation est connectée à la carte extension.

LEC 2 - Prise de contrôle manuel de la commande activée
S'allume lorsque les commandes de la caméra sont activées sur le sketch Arduino.

LEC 3 - Prise de contrôle manuel du tally activée
S'allume lorsque les commandes du tally sont activées sur le sketch Arduino.

LED 5 - Analyseur I^{2}C occupé
S'allume lorsque la communication est détectée entre la carte extension et l'Arduino avec le protocole I^{2}C.

LED 6 - Analyseur série occupé
S'allume quand la liaison UART est détectée.

Lorsque la carte extension Blackmagic s'allume, le voyant d'alimentation reste éteint et les LED 3, 4 et 5 indiquent les activités suivantes.

LED 3 - Chargement de l'image de l'application

LED 4 - Initialisation de la mémoire EEPROM

LED 5 - Vérification de la mémoire en cours

Une fois la carte extension démarrée, le voyant d'alimentation s'allume et les autres LED reprennent leur fonction standard.

En cas de problème au cours du démarrage, les LED qui rencontrent des problèmes ne clignotent pas. En revanche, celles qui fonctionnent clignotent rapidement. Cela vous permet d'identifier facilement l'origine du problème.

Fixation de composants sur la carte extension

Si vous souhaitez créer votre propre contrôleur matériel, vous pouvez construire une nouvelle carte extension dotée de boutons, de molettes et d'un joystick pour effectuer un contrôle plus précis. Fixez simplement la carte extension personnalisée à la Blackmagic 3G-SDI Shield for Arduino en l'insérant dans les trous de la barrette. Vous pouvez ajouter le nombre de contrôleurs dont vous avez besoin. Si vous le souhaitez, vous pouvez même remplacer le circuit électronique d'une ancienne voie de commande par votre propre Arduino.
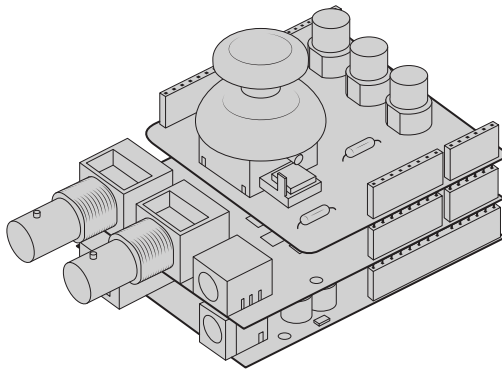
Créez votre propre contrôleur matériel et fixer-le à la Blackmagic 3G-SDI Shield for Arduino pour un contrôle extrêmement précis.

Communiquer avec votre Blackmagic Shield for Arduino

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I²C or Serial. We recommend I²C because of the low pin count and it frees up the serial monitor. This also allows you to use more I²C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I²C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                 shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial      sdiCameraControl;
BMD_SDITallyControl_Serial       sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                 shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

<blockquote>
1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels
</blockquote>

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	–	–	BCD - HHMMSSFF (UCT)
				[1] date	–	–	BCD - YYYYMMDD
	7.1	System language	string	–	–	–	ISO-639-1 two character language code
	7.2	Timezone	int32	–	–	–	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	–	–	BCD - s0DDddddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, ddddddddddd decimal degrees
				[1] longitude	–	–	BCD - sDDDddddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, dddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte 0 destination	1 length	2 command	3 reserved	4 category	5 parameter	6 type	7 operation	8	9	10	11	12	13	14	15
		header				command				data							
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Informations pour les développeurs

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
-------------------- |----------------------------
 A4                  | Serial Data (SDA)
 A5                  | Serial Clock (SCL)
```

I²C Protocol (Writes):

   (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

   (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
-------------------- |----------------------------
 IO1                 | Serial Transmit (TX)
 IO0                 | Serial Receive (RX)
```

UART Protocol (Writes):

   [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

   [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
 Address              | Name         | R/W     | Register Description
-------------------- |-----------   |-----    |------------------------------

 0x0000 - 0x0003      | IDENTITY     | R       | Hardware Identifier
 0x0004 - 0x0005      | HWVERSION    | R       | Hardware Version
 0x0006 - 0x0007      | FWVERSION    | R       | Firmware Version
                      |              |         |
 0x1000               | CONTROL      | R/W     | System Control
                      |              |         |
 0x2000               | OCARM        | R/W     | SDI Control Override Arm
 0x2001               | OCLENGTH     | R/W     | SDI Control Override Length
 0x2100 - 0x21FE      | OCDATA       | R/W     | SDI Control Override Data
                      |              |         |
 0x3000               | ICARM        | R/W     | SDI Control Incoming Arm
 0x3001               | ICLENGTH     | R       | SDI Control Incoming Length
 0x3100 - 0x31FE      | ICDATA       | R       | SDI Control Incoming Data
                      |              |         |
 0x4000               | OTARM        | R/W     | SDI Tally Override Arm
 0x4001               | OTLENGTH     | R/W     | SDI Tally Override Length
 0x4100 - 0x41FE      | OTDATA       | R/W     | SDI Tally Override Data
                      |              |         |
 0x5000               | ITARM        | R/W     | SDI Tally Incoming Arm
 0x5001               | ITLENGTH     | R       | SDI Tally Incoming Length
 0x5100 - 0x51FE      | ITDATA       | R       | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the
                         shield generates its own SDI signal on the SDI output connector.
                         When 0, the input signal is passed through to the output if
                         present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately
                         copied over to the override tally data register. Automatically
                         cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data.
                         When 0, input tally data is passed through to the output
                         unmodified.

Override Control: When 1, control data is overridden with the user supplied data.
                         When 0, input control data is passed through to the output
                         unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in
                         the next video frame. Automatically cleared once the control has
                         been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in
                         the next video frame. Automatically cleared once a control
                         packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when
                         a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM. ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Assistance

Obtenir de l'aide

La Blackmagic 3G-SDI Shield for Arduino est un outil pour développeurs créé pour vous aider à développer en fonction de vos besoins.

Pour avoir accès aux informations les plus récentes concernant les cartes extension, consultez les pages d'assistance en ligne de Blackmagic Design.

Pages d'assistance en ligne de Blackmagic Design

Les dernières versions du manuel, du logiciel et des notes d'assistance peuvent être consultées sur la page d'assistance technique de Blackmagic Design : www.blackmagicdesign.com/fr/support.

Forum pour développeurs Arduino

Si vous souhaitez poser des questions concernant la programmation, consultez les forums pour développeurs Arduino sur Internet. Il existe une large communauté de développeurs Arduino et de nombreux forums sur Internet qui vous aideront à répondre à vos questions.

Forum Blackmagic Design

Le forum Blackmagic Design est une source d'information utile qui offre des idées innovantes pour vos productions. Cette plate-forme d'aide vous permettra également d'obtenir des réponses rapides à vos questions, car un grand nombre de sujets peuvent avoir déjà été abordés par d'autres utilisateurs. Pour vous rendre sur le forum : http://forum.blackmagicdesign.com/fr

Vérification du logiciel actuel

Pour vérifier quelle version du logiciel Blackmagic 3G-SDI Shield for Arduino Setup est installée sur votre ordinateur, ouvrez la fenêtre About Blackmagic 3G-SDI Shield for Arduino Setup.

- Sur Mac OS X, ouvrez le logiciel Blackmagic 3G-SDI Shield for Arduino Setup dans le dossier Applications. Sélectionnez About Blackmagic Shield for Arduino Setup dans le menu d'application pour connaître le numéro de version.
- Sur Windows 7, ouvrez le logiciel Blackmagic 3G-SDI Shield for Arduino Setup dans le menu de Démarrage. Cliquez sur le menu Aide et sélectionnez À propos de Blackmagic 3G-SDI Shield for Arduino Setup pour connaître le numéro de version.
- Sur Windows 8, ouvrez le logiciel Blackmagic 3G-SDI Shield for Arduino Setup à partir de la vignette Blackmagic 3G-SDI Shield for Arduino Setup située sur l'écran d'accueil. Cliquez sur le menu Aide et sélectionnez À propos de Blackmagic Shield for Arduino Setup pour connaître le numéro de version.

Comment obtenir les dernières mises à jour du logiciel

Après avoir vérifié quelle version du logiciel Blackmagic 3G-SDI Shield for Arduino Setup est installée sur votre ordinateur, veuillez vous rendre au centre de support technique Blackmagic Design à l'adresse suivante www.blackmagicdesign.com/fr/support pour vérifier les dernières mises à jour. Même s'il est généralement conseillé d'installer les dernières mises à jour, il est prudent d'éviter d'effectuer ces mises à jour au milieu d'un projet important.

Garantie

Garantie limitée à 12 mois

Par la présente, Blackmagic Design garantit que le produit Blackmagic 3G-SDI Shield for Arduino sera exempt de défauts matériels et de fabrication pendant une durée d'un an à compter de la date d'achat. Si un produit s'avère défectueux pendant la période de garantie, Blackmagic Design peut, à sa seule discrétion, réparer le produit défectueux sans frais pour les pièces et la main-d'œuvre, ou le remplacer.

Pour se prévaloir du service offert en vertu de la présente garantie, il vous incombe d'informer Blackmagic Design de l'existence du défaut avant expiration de la période de garantie, et de prendre les mesures nécessaires pour l'exécution des dispositions de ce service. Le consommateur a la responsabilité de s'occuper de l'emballage et de l'expédition du produit défectueux au centre de service nommément désigné par Blackmagic Design, en frais de port prépayé. Il incombe au consommateur de payer l'intégralité des frais de transport, d'assurance, des droits de douane et taxes et toutes autres charges relatives aux produits qui nous auront été retournés, et ce quelle que soit la raison.

La présente garantie ne saurait en aucun cas s'appliquer à des défauts, pannes ou dommages causés par une utilisation inappropriée ou un entretien inadéquat ou incorrect. Blackmagic Design n'a en aucun cas l'obligation de fournir un service en vertu de la présente garantie : a) pour réparer les dommages résultant de tentatives de réparations, d'installations ou tous services effectués par du personnel non qualifié par Blackmagic Design, b) pour réparer tout dommage résultant d'une utilisation inadéquate ou d'une connexion à du matériel incompatible, c) pour réparer tout dommage ou dysfonctionnement causés par l'utilisation de pièces ou de fournitures n'appartenant pas à la marque de Blackmagic Design, d) pour examiner un produit qui a été modifié ou intégré à d'autres produits quand l'impact d'une telle modification ou intégration augmente les délais ou la difficulté d'examiner ce produit. CETTE GARANTIE REMPLACE TOUTE GARANTIE EXPLICITE OU TACITE. BLACKMAGIC DESIGN ET SES REVENDEURS DÉCLINENT EXPRESSÉMENT TOUTE GARANTIE TACITE DE COMMERCIALISATION OU D'ADÉQUATION À UNE FIN PARTICULIÈRE. LA RESPONSABILITÉ DE BLACKMAGIC DESIGN POUR RÉPARER OU REMPLACER UN PRODUIT S'AVÉRANT DÉFECTUEUX CONSTITUE LA TOTALITÉ ET LE SEUL RECOURS EXCLUSIF PRÉVU ET FOURNI AU CONSOMMATEUR POUR TOUT DOMMAGE INDIRECT, SPÉCIFIQUE, ACCIDENTEL OU CONSÉCUTIF, PEU IMPORTE QUE BLACKMAGIC DESIGN OU SES REVENDEURS AIENT ÉTÉ INFORMÉS OU SE SOIENT RENDU COMPTE AU PRÉALABLE DE L'ÉVENTUALITÉ DE CES DOMMAGES. BLACKMAGIC DESIGN NE PEUT ÊTRE TENU POUR RESPONSABLE DE TOUTE UTILISATION ILLICITE DU MATÉRIEL PAR LE CONSOMMATEUR. BLACKMAGIC DESIGN N'EST PAS RESPONSABLE DES DOMMAGES RÉSULTANT DE L'UTILISATION DE CE PRODUIT. LE CONSOMMATEUR UTILISE CE PRODUIT À SES SEULS RISQUES.
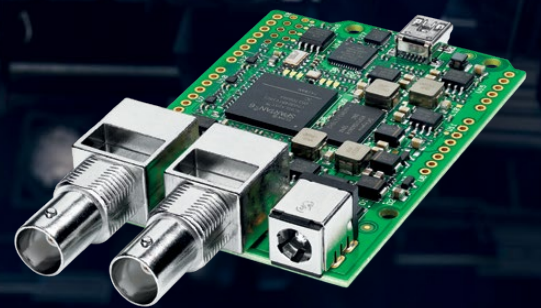
Installations- und Bedienungsanleitung

Blackmagic 3G-SDI Shield for Arduino

Februar 2020

Deutsch

Willkommen!

Danke, dass Sie sich für den Kauf des neuen Blackmagic 3G-SDI Shield for Arduino entschieden haben.

Wir haben ein ungemeines Interesse an neuen Technologien und sind immer wieder begeistert, auf welch kreative Art und Weise unsere SDI-Produkte eingesetzt werden. Mit dem 3G-SDI Shield for Arduino können Sie den Arduino jetzt in Ihren SDI-Workflow integrieren, um Ihr Blackmagic Design Equipment um noch mehr Steuerungsmöglichkeiten zu erweitern.

So lassen sich bspw. die URSA Mini und Blackmagic Studio Cameras von einem ATEM Mischer aus steuern. Das geschieht mithilfe von Datenpaketen, die in das SDI-Signal eingebettet sind. Sollten Sie über keinen ATEM Mischer verfügen, aber dennoch die Möglichkeit haben wollen, Ihre Blackmagic Kameras zu steuern, können Sie mit Ihrem 3G-SDI Shield for Arduino auch Steuerungslösungen nach Ihrem Gusto kreieren. Das Shield dient Ihnen hierbei als SDI-Grundgerüst, damit Sie den Programm-Feed von Ihrem Mischer anhand des Shields rückführen und durch den Programmeingang Ihrer Blackmagic Cameras schleifen können.

Der Code für die Befehle, die an die Kamera gesendet werden, ist im Handumdrehen geschrieben. Alle unterstützten Befehle finden Sie außerdem in diesem Handbuch.

Steuern Sie Ihre Kameras entweder über einen Computer oder erweitern Sie Ihr Shield um Tasten, Regler und Joysticks. Bauen Sie sich so dynamische Hardware-Controller, mit denen sich Funktionen wie Fokus, Zoom, Blendeneinstellungen, Schwarzabhebung, Weißabgleich sowie der leistungsstarke Farbkorrektor der Kamera und vieles mehr anpassen lassen. Sich einen Controller nach den eigenen Bedürfnissen zu basteln ist nicht nur hilfreich für die Produktion, sondern macht auch eine Menge Spaß!

Wir sind von dieser Technologie begeistert und würden uns freuen, wenn Sie Ihre Ideen mit uns teilen und uns erzählen, welche SDI-Steuerelemente Sie für Ihr 3G-SDI Shield for Arduino zusammengestellt haben.

Diese Bedienungsanleitung gibt Ihnen alle Informationen, die Sie für die Inbetriebnahme Ihres Blackmagic 3G-SDI Shield for Arduino brauchen. Bitte sehen Sie auf der Support-Seite unter www.blackmagicdesign.com/de/support nach der aktuellsten Auflage der Bedienungsanleitung sowie Aktualisierungen der Produktsoftware Ihres Shields. Indem Sie Ihre Software auf dem neuesten Stand halten, haben Sie stets Zugriff auf neue, aktuelle Features! Wenn Sie Software herunterladen, empfehlen wir Ihnen, sich zu registrieren, sodass wir Sie über neue Updates informieren können, sobald diese zur Verfügung stehen. Da wir ständig an neuen Features und Verbesserungen arbeiten, freuen wir uns jederzeit, von Ihnen zu hören.

Grant Petty

Grant Petty
CEO, Blackmagic Design

Inhalt

# Blackmagic 3G-SDI Shield for Arduino

Erste Schritte

Aufstecken und Verlöten von Stiftleisten

Ihr Blackmagic 3G-SDI Shield for Arduino wird mit vier stapelbaren Stiftleisten geliefert, darunter zwei 8-polige, eine 10-polige und eine 6-polige. Stiftleisten sind Verbindungsstecker, mit denen Sie Ihr Shield auf das Arduino-Board aufstecken. Da die Stiftleisten stapelbar sind, können Sie weitere Shields mit zusätzlichen Komponenten wie Steuerungstasten, Reglern und Joysticks übereinander anbringen. Die Stiftleisten sind so aufgebaut, dass sie für die Montage auf Arduino-Boards mit R3-Fläche wie dem Arduino UNO geeignet sind.

So verbinden Sie die Stiftleisten mit Ihrem Shield:

1 Führen Sie die Stifte der Leisten in die jeweiligen Buchsen auf beiden Seiten Ihres Blackmagic 3G-SDI Shields ein. Die Anordnung der Stiftleisten finden Sie in der Abbildung unten.
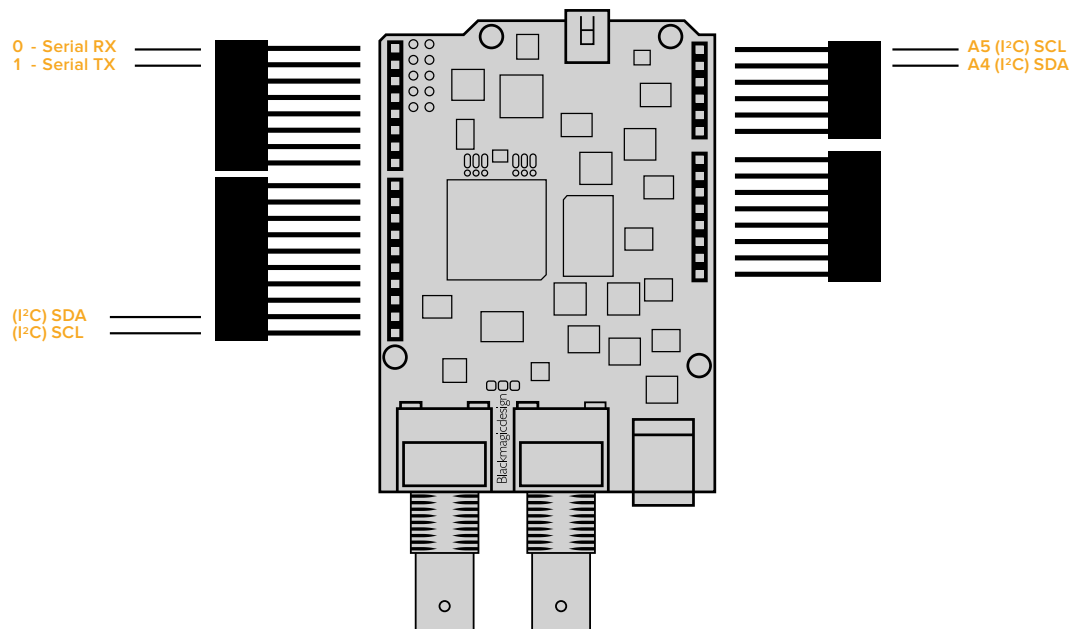
HINWEIS Wenn Sie sich mit dem Shield verbinden, läuft die Kommunikation über I²C oder seriell. Wir empfehlen I²C, da dies den seriellen Monitor unterstützt und alle Pins verfügbar bleiben. Wählen Sie den Kommunikationsmodus, wenn Sie das BMDSDIControl-Objekt im Sketch festlegen. Weitere Informationen finden Sie im Abschnitt „Kommunizieren mit Ihrem Blackmagic 3G-SDI Shield for Arduino".

2 Verlöten Sie den Ansatz eines jeden Stifts mit der Unterseite Ihres Shields. Vergewissern Sie sich, dass das Lot eine feste Verbindung zwischen Stift und Buchse erzeugt, sich die einzelnen Lötstellen nebeneinanderliegender Stifte jedoch nicht berühren.

Anbringen an das Arduino-Board

Sobald die Stiftleisten mit Ihrem Shield verlötet sind, können Sie das 3G-SDI Shield auf Ihr Arduino-Board aufstecken.

Halten Sie das Shield an beiden Seiten fest und richten Sie die Stiftleisten auf die Buchsenleisten Ihres Arduino-Boards aus. Drücken Sie nun die Stifte vorsichtig in die Buchsen. Achten Sie darauf, dass sich die Stifte dabei nicht verbiegen.
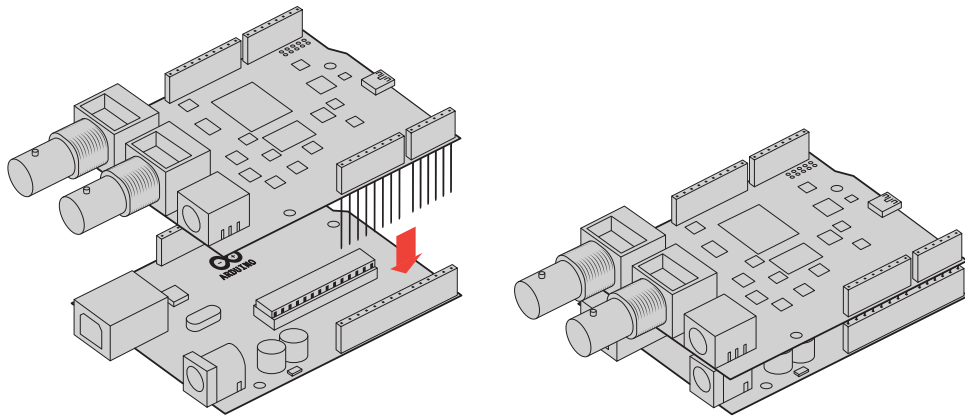
Wenn alle Stifte eingeführt sind, sollte eine feste und stabile Verbindung zwischen dem Blackmagic Shield und dem Arduino-Board bestehen

Anschließen an das Stromnetz

Schließen Sie Ihr Blackmagic 3G-SDI Shield for Arduino an das Stromnetz an, indem Sie einfach einen 12V-Stromadapter in den 12V-Stromausgang Ihres Blackmagic Shields stecken.

Anschließen an SDI-Equipment

Ist eine Verbindung zum Stromnetz hergestellt, können Sie das Blackmagic 3G-SDI Shield for Arduino an Ihr SDI-Equipment anschließen. So schließen Sie bspw. einen Mischer und eine Blackmagic URSA Mini an:

1 Schließen Sie den Programmausgang Ihres Mischers an den SDI-Eingang des Blackmagic 3G-SDI Shields an.

2 Verbinden Sie den SDI-Ausgang Ihres Blackmagic 3G-SDI Shields mit dem mit „PGM" gekennzeichneten SDI-Programmeingang Ihrer URSA Mini.

Ein Anschlussdiagramm finden Sie weiter unten.

SDI IN

SDI OUT

Switcher

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini
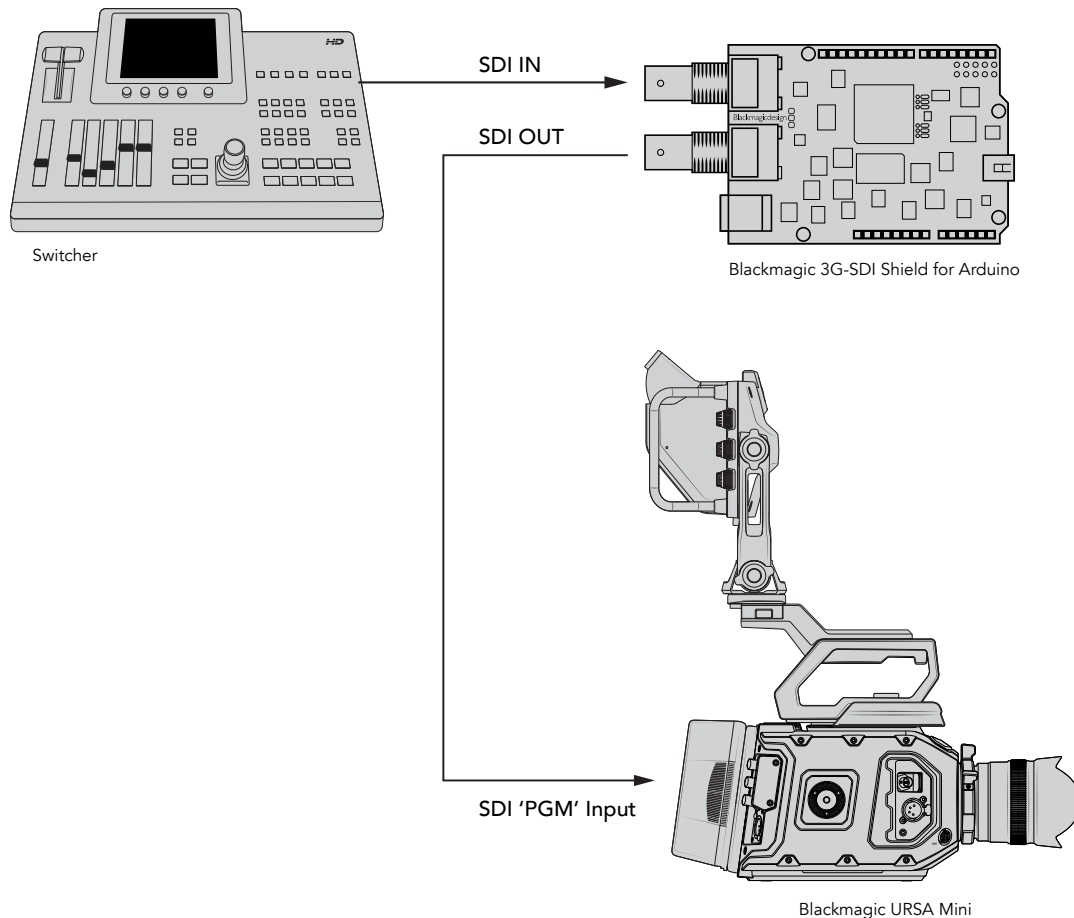
Für den Einstieg war das schon alles.

Nun, da Ihr Shield auf dem Arduino-Board angebracht, es mit Strom versorgt und mit Ihrem SDI-Equipment verbunden ist, können Sie die Produktsoftware sowie die Bibliotheksdateien installieren, die Arduino-Software programmieren und mit dem Shield Ihr Equipment steuern.

Im weiteren Verlauf des Handbuchs finden Sie Informationen darüber, wie Sie die Produktsoftware des Shields installieren und wo Sie die Arduino-Bibliotheksdateien speichern sollten, damit das Shield mit Ihrem Arduino kommunizieren kann.

> **TIPP** Ihr Blackmagic 3G-SDI Shield for Arduino können Sie auch für die Steuerung anderer Blackmagic Design Produkte wie bspw. des Blackmagic MultiView 16 benutzen. Wenn Sie Ihr Shield an Eingang 16 koppeln, können Sie so Tally-Umrandungen in der Mehrfachansicht anzeigen. Wenn Sie Blackmagic URSA Mini Kameras verwenden, stellen Sie sicher, dass das Grid-Overlay eingeschaltet ist, damit das Tally-Display im Suchmonitor aktiviert wird. Für weitere Informationen sehen Sie in der Bedienungsanleitung Ihrer Blackmagic Kamera nach.

Softwareinstallation

HINWEIS Laden Sie die aktuellste Arduino IDE Software von www.arduino.cc herunter und installieren Sie sie auf Ihrem Computer. Installieren Sie erst dann das Blackmagic 3G-SDI Shield for Arduino Setup-Dienstprogramm.

Nachdem Sie die Arduino-Software installiert haben, können Sie die Blackmagic 3G-SDI Shield Produktsoftware installieren.

Installieren der Produktsoftware

Blackmagic Shield for Arduino Setup dient der Aktualisierung der Produktsoftware Ihres Shields. Die Produktsoftware kommuniziert mit dem Arduino-Board und steuert es mithilfe von Arduino-Bibliotheksdateien. Diese Bibliotheksdateien werden zusammen mit der Setup-Software installiert. Sie müssen lediglich den Ordner mit den Dateien kopieren und in den Arduino-Anwendungsordner einfügen. Informationen zu den Bibliotheksdateien und wie diese installiert werden, finden Sie im nächsten Kapitel dieses Handbuchs.

Wir empfehlen Ihnen, die aktuellste Blackmagic 3G-SDI Shield for Arduino Software herunterzuladen und Ihr Shield auf den aktuellsten Stand zu bringen, damit Sie von neuen Funktionen und Verbesserungen profitieren können. Die aktuellste Version steht im Blackmagic Design Support Center unter www.blackmagicdesign.com/de/support zum Download bereit.

So installieren Sie die Produktsoftware unter Mac OS X:

1   Laden Sie die Blackmagic Shield for Arduino Software herunter und entpacken Sie sie.

2   Öffnen Sie das Speicherabbild und starten Sie das Blackmagic Shield for Arduino Installationsprogramm. Folgen Sie den Anweisungen auf Ihrem Bildschirm.

3   Nachdem die aktuellste Version des Blackmagic Shield for Arduino Installationsprogramms eingespielt wurde, versorgen Sie Ihr Blackmagic Shield mit Strom und verbinden Sie es per USB-Kabel mit Ihrem Computer.

4   Starten Sie nun das Setup-Dienstprogramm und folgen Sie der Aufforderung auf Ihrem Bildschirm für die Aktualisierung der Produktsoftware Ihres Shields. Sollte diese Aufforderung nicht erscheinen, so ist Ihre Produktsoftware auf dem neuesten Stand. Keine weiteren Aktionen Ihrerseits sind notwendig.

So installieren Sie die Produktsoftware unter Windows:

1   Laden Sie die Blackmagic Shield for Arduino Software herunter und entpacken Sie sie.

2   Nun sollten Sie einen Blackmagic Shield for Arduino Ordner sehen. Darin finden Sie dieses Handbuch und das Blackmagic Shield for Arduino Installationsprogramm. Doppelklicken Sie auf das Installationsprogramm und folgen Sie den Anweisungen zur Fertigstellung der Installation auf dem Bildschirm.

3   Nachdem die aktuellste Version des Blackmagic Shield for Arduino Installationsprogramms eingespielt wurde, versorgen Sie Ihr Blackmagic Shield mit Strom und verbinden Sie es per USB-Kabel mit Ihrem Computer.

4   Starten Sie nun das Setup-Dienstprogramm und folgen Sie der Aufforderung auf Ihrem Bildschirm für die Aktualisierung der Produktsoftware Ihres Shields. Sollte diese Aufforderung nicht erscheinen, so ist Ihre Produktsoftware auf dem neuesten Stand. Keine weiteren Aktionen Ihrerseits sind notwendig.

Installieren der Arduino Bibliotheksdateien

Die Programme zur Steuerung Ihres Arduinos werden als Sketche bezeichnet. Ihr Blackmagic 3G-SDI Shield for Arduino macht von Bibliotheksdateien Gebrauch, um das Schreiben von Sketchen zu vereinfachen. Nach der Installation der Konfigurationssoftware Ihres Shields werden die Bibliotheksdateien in einem Ordner namens „Library" (Bibliothek) gespeichert. Kopieren Sie den Ordner mit den Bibliotheksdateien und fügen Sie ihn in den „Library"-Ordner Ihres Arduino ein.

> **HINWEIS** Während der Installation von Bibliotheken muss die IDE-Software geschlossen sein.

So installieren Sie Bibliotheksdateien unter Mac OS X:

1   Öffnen Sie „Blackmagic Shield for Arduino" im Ordner „Programme".

2   Öffnen Sie den Ordner „Library" und führen Sie auf dem Ordner „BMDSDIControl" einen Rechtsklick aus. Wählen Sie anschließend „Kopieren".

3   Gehen Sie nun zum Ordner „Dokumente" und öffnen Sie den „Arduino"-Ordner.

4   Dort befindet sich ein Unterordner namens „Libraries". Fügen Sie den Ordner „BMDSDIControl" in den Ordner „Libraries" ein.

So installieren Sie Bibliotheksdateien unter Windows:

1   Öffnen Sie den Ordner „Programme" bzw. „Blackmagic Shield for Arduino".

2   Sie sehen nun einen Unterordner mit dem Namen „Library". Öffnen Sie diesen und kopieren Sie den Ordner mit dem Namen „BMDSDIControl" per Rechtsklick.

3   Gehen Sie nun zum Ordner „Dokumente" und öffnen Sie den „Arduino"-Ordner.

4   Dort befindet sich ein Unterordner namens „Libraries". Fügen Sie den Ordner „BMDSDIControl" in den Ordner „Libraries" ein.

Das ist schon alles, um die Blackmagic Design Bibliotheksdateien auf Ihrem Computer zu installieren. Wenn Sie mit der Arduino-Software arbeiten, stehen Ihnen jetzt auch Blackmagic Design Beispiel-Sketche zur Verfügung.

Öffnen Sie dazu einfach das Aufklappmenü „File" (Datei) in der Menüzeile der Arduino-Software und wählen Sie „Examples" (Beispiele). Klicken Sie auf „BMDSDIControl". Nun erscheint eine Liste mit Beispiel-Sketchen zu Ihrer Auswahl.

Wenn sich die Bibliotheksdateien im korrekten Ordner befinden, kann Ihr Shield nun auf sie zugreifen und mit dem Arduino-Board kommunizieren. Alles, was Sie dafür tun müssen, ist die Arduino-IDE-Software zu programmieren. Weitere Informationen finden Sie im Abschnitt „Programmieren von Arduino-Sketchen".

> **HINWEIS** Wenn in Zukunft ein Update der Bibliotheksdatei mit Beispielen herausgegeben wird, müssen Sie den alten „BMDSDIControl"-Ordner löschen und ihn mit dem neuen Ordner ersetzen. Folgen Sie den Anweisungen oben.

Blackmagic Shield for Arduino Setup
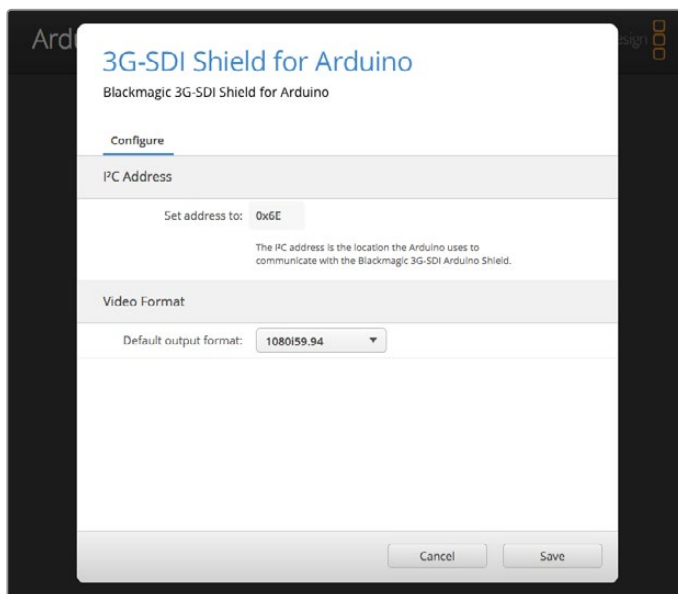
Über die Blackmagic Shield for Arduino Setup Software können Sie Änderungen der Einstellungen an Ihrem Shield vornehmen. Dazu gehören bspw. die I²C-Adresse und das Videoausgabeformat

Wenn Blackmagic Shield for Arduino Setup auf Ihrem Computer installiert ist, können Sie darüber die Einstellungen Ihres Shields ändern. Dazu gehört die „I²C Address" (I²C-Adresse), die Ihr Shield identifiziert, damit das Arduino-Board mit ihm kommunizieren kann, und das „Video Format" (Videoformat), das das Ausgabeformat Ihres Shields festlegt.

„I²C Address"

In seltenen Fällen ist es möglich, dass ein weiteres Shield, das auf Ihrem Blackmagic Shield angebracht ist, dieselbe standardmäßige I²C-Adresse wie Ihr Shield verwendet. Das führt zu einem Konflikt. Sollte dies der Fall sein, können Sie die Standardadresse ändern.

Die Standardadresse für Ihr Shield ist 0x6E. Sie haben jedoch eine Reihe von Adressen zwischen 0x08 und 0x77 zur Wahl.

So ändern Sie die Adresse Ihres Shields:

1   Starten Sie Blackmagic Shield for Arduino Setup und klicken Sie auf das Einstellungssymbol.

2   Tragen Sie in den Kasten „Set Address to:" (Adresse einstellen auf:) die Adresse ein, die Sie verwenden möchten.

3   Klicken Sie auf „Save" (Speichern).

„Video Format"

Wenn keine Eingabe angeschlossen ist, wird das im Setup-Dienstprogramm eingestellte Standardausgabeformat gewählt. Wird eine Eingabe erkannt, erfolgt die Ausgabe im selben Format wie die Eingabe. Wird diese Eingabe entfernt, setzt sich der Ausgang auf das Ausgabeformat zurück, das im Dienstprogramm festgelegt wurde. Sie können das Videoformat ändern, indem Sie im Ausklappmenü unter „Default Output Format" (Standardausgabeformat) das gewünschte Format auswählen.

Die folgenden Videoausgabeformate sind verfügbar:

- 720p/50
- 720p/59,94
- 720p/60
- 1080i/50
- 1080i/59,94
- 1080i/60
- 1080p/23,98
- 1080p/24
- 1080p/25
- 1080p/29,97
- 1080p/30
- 1080p/50
- 1080p/59,94
- 1080p/60

Programmieren von Arduino-Sketchen

Die Programme – oder Sketche – für die Arduino-Software sind kinderleicht zu schreiben! Sketche werden in gängigen C-Programmiersprachen geschrieben. Wenn Sie Sketche mithilfe von Befehlen des Studio Camera Control Protocols programmieren, bettet das Shield diese Befehle in die SDI-Ausgabe ein. Mit dieser lassen sich Ihre Blackmagic URSA Mini oder Blackmagic Studio Cameras daraufhin steuern.

Alle unterstützten Befehle finden Sie im Abschnitt „Studio Camera Control Protocol" dieses Handbuchs. Nehmen Sie diese und benutzen Sie sie für Ihren Sketch.

Testen der Blackmagic Shield und Bibliotheken-Installation

Nachdem alles wie im Abschnitt „Erste Schritte" beschrieben verbunden wurde und Sie die Konfigurationssoftware und die Bibliotheksdateien installiert haben, möchten Sie wahrscheinlich testen, ob Ihr Shield mit dem Arduino-Board kommuniziert und alles einwandfrei funktioniert.

Am schnellsten geht das, wenn Sie den vorhandenen Beispiel-Sketch für das Blinken des Tally-Lichts öffnen und ausführen.

Das geht so:

1 Starten Sie die Arduino-IDE-Software.

2 Gehen Sie zum „Tools"-Menü (Werkzeuge) und wählen Sie das Arduino-Board und die Port-Nummer aus.

3 Klicken Sie im „File"-Menü (Datei) auf „Examples/BMDSDIControl" (Beispiele/BMDSDIControl) und wählen Sie „TallyBlink" aus.

4 Laden Sie den Sketch auf Ihr Board.

Das Beispiel mit dem Blinken des Tally-Lichts ist eine schnelle und einfache Möglichkeit, Ihr Blackmagic 3G-SDI Shield for Arduino zu testen. Rohdaten können mithilfe von Befehlen aus dem Studio Camera Protocol Dokument über I²C an Ihr Shield gesendet werden. Wir haben Ihnen aber zudem eine spezielle Bibliothek zur Verfügung gestellt, damit Sie Sketche sehr viel einfacher programmieren können.

Nun sollte das Tally-Licht an Ihrer Blackmagic Studio Camera einmal pro Sekunde blinken. Ist dies der
Fall, können Sie sicher sein, dass Ihr Blackmagic Shield mit dem Arduino kommuniziert und alles
einwandfrei funktioniert.

Blinkt das Tally-Licht nicht, vergewissern Sie sich zunächst, dass die Tally-Nummer der Blackmagic
Kamera auf „1" eingestellt und das Grid-Overlay eingeschaltet ist.

Sollten Sie weitere Hilfe benötigen, besuchen Sie das Blackmagic Design Support Center unter
www.blackmagicdesign.com/de/support. Im Abschnitt „Hilfe" dieses Handbuchs finden Sie weitere
Informationen dazu, wie Sie Hilfestellungen bei der Konfiguration Ihres Shields bekommen.

Status-LEDs

Ihr Blackmagic 3G-SDI Shield for Arduino verfügt über sechs Status-LEDs, die unterschiedliche
Aktivitätszustände wie Stromzufuhr, UART-, I²C- und SPI-Kommunikation anzeigen. Die LEDs zeigen
außerdem an, wenn die Tally- und Camera-Control-Overrides aktiviert sind.
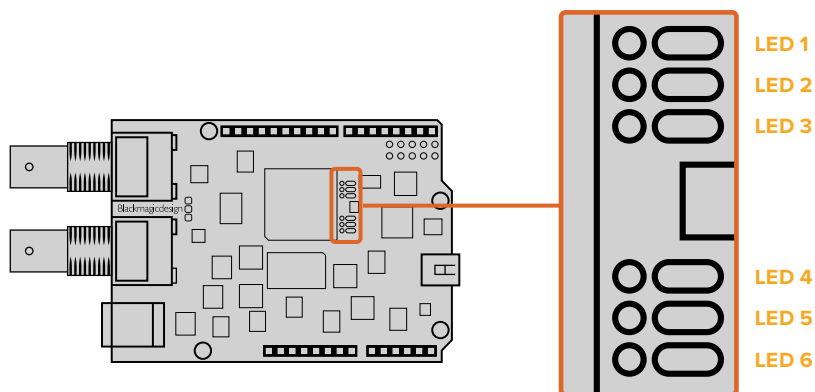
LED 1 – System aktiv
Leuchtet auf, wenn das Shield mit Strom versorgt wird.

LED 2 – Control-Overrides aktiviert
Leuchtet auf, wenn Sie in Ihrem Arduino-Sketch die Kamerasteuerung aktiviert haben.

LED 3 – Tally-Overrides aktiviert
Leuchtet auf, wenn Sie in Ihrem Arduino-Sketch Tally aktiviert haben.

LED 5 – I²C-Parser aktiv
Leuchtet auf, wenn Ihr Shield und der Arduino über das I²C-Protokoll kommunizieren.

LED 6 – Serieller Parser aktiv
Leuchtet auf, wenn UART-Kommunikation erkannt wird.

Wenn Ihr Blackmagic Shield bootet, bleibt die Strom-LED unbeleuchtet. LEDs 3, 4 und 5 zeigen die
folgenden Aktivitäten an:

LED 3 – Anwendungs-Image lädt

LED 4 – EEPROM wird initialisiert

LED 5 – Speichertest wird durchgeführt

Nach erfolgreichem Start leuchtet die Strom-LED auf. Alle anderen LEDs übernehmen daraufhin wieder ihre betriebliche Standardfunktion.

Im seltenen Fall eines fehlgeschlagenen Systemstarts beginnen alle LEDs – bis auf die für die fehlgeschlagene Aktivität – schnell zu blinken. So ist die Ursache leicht identifiziert.

Anbringen von Shield-Komponenten

Wenn Sie Ihren eigenen Hardware-Controller bauen möchten, können Sie für eine greifbarere und praktischere Handhabung ein neues Shield mit Tasten, Reglern und einem Joystick kreieren. Montieren Sie Ihr eigens angefertigtes Shield einfach auf Ihrem Blackmagic 3G-SDI Shield for Arduino, indem Sie es auf die dafür vorgesehenen Leisten Ihres Shields aufstecken. Den Arten von Steuerelementen, die Sie bauen können, sind keine Grenzen gesetzt. Es lässt sich sogar der Schaltkreis einer alten CCU durch Ihre selbst zusammengestellte Arduino-Lösung austauschen. Und schon haben Sie eine Kamerasteuerungseinheit nach Industriestandard.
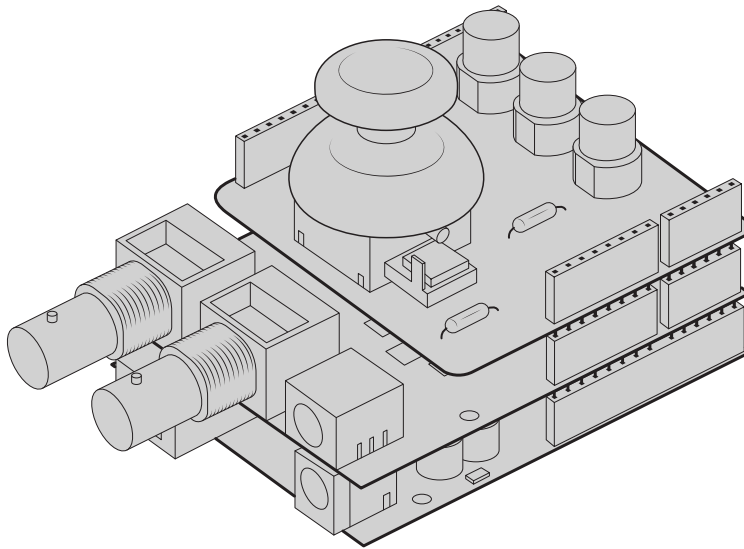
Sie können Ihren eigenen Hardware-Controller kreieren und diesen für eine noch interaktivere und präzisere Steuerung auf Ihrem Blackmagic 3G-SDI Shield for Arduino montieren

Kommunizieren mit Ihrem Blackmagic Shield for Arduino (English)

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I^{2}C or Serial. We recommend I^{2}C because of the low pin count and it frees up the serial monitor. This also allows you to use more I^{2}C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I^{2}C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                 shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial    sdiCameraControl;
BMD_SDITallyControl_Serial     sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                 shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol (English)

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
Video	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	–	–	BCD - HHMMSSFF (UCT)
				[1] date	–	–	BCD - YYYYMMDD
	7.1	System language	string	–	–	–	ISO-639-1 two character language code
	7.2	Timezone	int32	–	–	–	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	–	–	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	–	–	BCD - sDDDddddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header				command				data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information (English)

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin         | Function
-------------------- |----------------------------
 A4                 | Serial Data (SDA)
 A5                 | Serial Clock (SCL)
```

I²C Protocol (Writes):

       (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

       (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin         | Function
-------------------- |----------------------------
 IO1                | Serial Transmit (TX)
 IO0                | Serial Receive (RX)
```

UART Protocol (Writes):

       [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

       [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name         | R/W   | Register Description
-------------------- |-----------   |-----  |------------------------------

0x0000 - 0x0003      | IDENTITY     | R     | Hardware Identifier
0x0004 - 0x0005      | HWVERSION    | R     | Hardware Version
0x0006 - 0x0007      | FWVERSION    | R     | Firmware Version
                     |              |       |
0x1000               | CONTROL      | R/W   | System Control
                     |              |       |
0x2000               | OCARM        | R/W   | SDI Control Override Arm
0x2001               | OCLENGTH     | R/W   | SDI Control Override Length
0x2100 - 0x21FE      | OCDATA       | R/W   | SDI Control Override Data
                     |              |       |
0x3000               | ICARM        | R/W   | SDI Control Incoming Arm
0x3001               | ICLENGTH     | R     | SDI Control Incoming Length
0x3100 - 0x31FE      | ICDATA       | R     | SDI Control Incoming Data
                     |              |       |
0x4000               | OTARM        | R/W   | SDI Tally Override Arm
0x4001               | OTLENGTH     | R/W   | SDI Tally Override Length
0x4100 - 0x41FE      | OTDATA       | R/W   | SDI Tally Override Data
                     |              |       |
0x5000               | ITARM        | R/W   | SDI Tally Incoming Arm
0x5001               | ITLENGTH     | R     | SDI Tally Incoming Length
0x5100 - 0x51FE      | ITDATA       | R     | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

```
[ IDENTITY ]
31        0
```

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7          0
```

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7          0
```

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

```
[ RESERVED ] [ OVERRIDE OUTPUT ] [ RESET TALLY ] [ OVERRIDE TALLY ] [
OVERIDE CONTROL ]
7        4        3        2        1        0
```

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Hilfe

Hilfestellung

Ihr Blackmagic 3G-SDI Shield for Arduino ist ein Werkzeug für Entwickler, mit dem Sie sich autonom Ihre ganz eigene maßgeschneiderte Lösung zusammenstellen können.

Die aktuellsten Informationen zu diesem Shield finden Sie auf den Blackmagic Design Support Seiten im Internet. Suchen Sie dort einfach nach den neuesten Hilfsmaterialien.

Blackmagic Design Online Support Seiten

Die aktuellsten Versionen der Bedienungsanleitung, Produktsoftware und Support-Hinweise finden Sie im Blackmagic Support Center unter www.blackmagicdesign.com/de/support.

Arduino Entwickler-Forum

Sollten Sie Programmierfragen haben, bieten Arduino Entwickler-Foren im Internet Hilfestellung. Es gibt eine Community von Arduino-Entwicklern sowie viele hilfreiche Foren für softwarebezogene Fragen. Dort finden Sie sogar bereitwillige Ingenieure, die Sie anheuern können, um Ihnen Ihre ganz persönlichen Lösungen umzusetzen.

Blackmagic Design Forum

Das Blackmagic Design Forum auf unserer Website ist eine praktische Ressource, die Sie für mehr Information und kreative Ideen aufsuchen können. Manchmal finden Sie dort schnellere Lösungen, da möglicherweise bereits Antworten auf ähnliche Fragen von anderen erfahrenen Anwendern und Blackmagic Design Mitarbeitern vorliegen, die Ihnen weiterhelfen. Das Forum finden Sie unter http://forum.blackmagicdesign.com.

Überprüfen der aktuell installierten Softwareversion

Um zu überprüfen, welche Version der Blackmagic 3G-SDI Shield for Arduino Setup Software auf Ihrem Computer installiert ist, öffnen Sie das „About Blackmagic 3G-SDI Shield for Arduino Setup" Fenster.

- Öffnen Sie unter Mac OS X Blackmagic 3G-SDI Shield for Arduino Setup über den Ordner Programme. Wählen Sie im Anwendungsmenü „About Blackmagic 3G-SDI Shield for Arduino Setup" aus, um die Versionsnummer nachzusehen.

- Öffnen Sie unter Windows 7 das Blackmagic 3G-SDI Shield for Arduino Setup über das Menü „Start". Klicken Sie auf das „Hilfe"-Menü und wählen Sie „About Blackmagic 3G-SDI Shield for Arduino Setup", um die Versionsnummer nachzusehen.

- Öffnen Sie unter Windows 8 Blackmagic 3G-SDI Shield for Arduino Setup über die Blackmagic 3G-SDI Shield for Arduino Setup Kachel auf Ihrer Startseite. Klicken Sie auf das „Hilfe"-Menü und wählen Sie „About Blackmagic 3G-SDI Shield for Arduino Setup", um die Versionsnummer nachzusehen.

So erhalten Sie die aktuellsten Software-Updates

Prüfen Sie zunächst die Versionsnummer der auf Ihrem Computer installierten Blackmagic 3G-SDI Shield for Arduino Setup Software. Sehen Sie dann im Blackmagic Design Support Center unter www.blackmagicdesign.com/de/support nach den neuesten Aktualisierungen. In der Regel empfiehlt es sich, die neuesten Updates zu laden. Vermeiden Sie jedoch Software-Updates mitten in einem wichtigen Projekt.

Garantie

12 Monate eingeschränkte Garantie

Die Firma Blackmagic Design gewährt eine Garantie auf Material- und Verarbeitungsfehler des Blackmagic 3G-SDI Shield for Arduino von 12 Monaten ab Kaufdatum. Sollte sich ein Produkt innerhalb dieser Garantiezeit als fehlerhaft erweisen, wird die Firma Blackmagic Design nach ihrem Ermessen das defekte Produkt entweder ohne Kostenerhebung für Teile und Arbeitszeit reparieren oder Ihnen das defekte Produkt ersetzen.

Zur Inanspruchnahme der Garantieleistungen müssen Sie als Kunde Blackmagic Design über den Defekt innerhalb der Garantiezeit in Kenntnis setzen und die entsprechenden Vorkehrungen für die Leistungserbringung treffen. Es obliegt dem Kunden, für die Verpackung und den bezahlten Versand des defekten Produkts an ein spezielles von Blackmagic Design benanntes Service Center zu sorgen und hierfür aufzukommen. Sämtliche Versandkosten, Versicherungen, Zölle, Steuern und sonstige Ausgaben im Zusammenhang mit der Rücksendung von Waren an uns, ungeachtet des Grundes, sind vom Kunden zu tragen.

Diese Garantie gilt nicht für Mängel, Fehler oder Schäden, die durch unsachgemäße Handhabung oder unsachgemäße oder unzureichende Wartung und Pflege verursacht wurden. Blackmagic Design ist im Rahmen dieser Garantie nicht verpflichtet, die folgenden Serviceleistungen zu erbringen: a) Behebung von Schäden infolge von Versuchen Dritter, die Installation, Reparatur oder Wartung des Produkts vorzunehmen, b) Behebung von Schäden aufgrund von unsachgemäßer Handhabung oder Anschluss an nicht kompatible Geräte, c) Behebung von Schäden oder Störungen, die durch die Verwendung von nicht Blackmagic-Design-Ersatzteilen oder -Verbrauchsmaterialien entstanden sind, d) Service für ein Produkt, das verändert oder in andere Produkte integriert wurde, sofern eine solche Änderung oder Integration zu einer Erhöhung des Zeitaufwands oder zu Schwierigkeiten bei der Wartung des Produkts führt. ÜBER DIE IN DIESER GARANTIEERKLÄRUNG AUSDRÜCKLICH AUFGEFÜHRTEN ANSPRÜCHE HINAUS ÜBERNIMMT BLACKMAGIC DESIGN KEINE WEITEREN GARANTIEN, WEDER AUSDRÜCKLICH NOCH STILLSCHWEIGEND. DIE FIRMA BLACKMAGIC DESIGN UND IHRE HÄNDLER LEHNEN JEGLICHE STILLSCHWEIGENDEN GARANTIEN IN BEZUG AUF AUSSAGEN ZUR MARKTGÄNGIGKEIT UND GEBRAUCHSTAUGLICHKEIT FÜR EINEN BESTIMMTEN ZWECK AB. DIE VERANTWORTUNG VON BLACKMAGIC DESIGN, FEHLERHAFTE PRODUKTE ZU REPARIEREN ODER ZU ERSETZEN, IST DIE EINZIGE UND AUSSCHLIESSLICHE ABHILFE, DIE GEGENÜBER DEM KUNDEN FÜR ALLE INDIREKTEN, SPEZIELLEN, NEBEN- ODER FOLGESCHÄDEN ZUR VERFÜGUNG GESTELLT WIRD, UNABHÄNGIG DAVON, OB BLACKMAGIC DESIGN ODER DER HÄNDLER VON DER MÖGLICHKEIT SOLCHER SCHÄDEN ZUVOR IN KENNTNIS GESETZT WURDE. BLACKMAGIC DESIGN IST NICHT HAFTBAR FÜR JEGLICHE WIDERRECHTLICHE VERWENDUNG DER GERÄTE DURCH DEN KUNDEN. BLACKMAGIC HAFTET NICHT FÜR SCHÄDEN, DIE SICH AUS DER VERWENDUNG DES PRODUKTS ERGEBEN. NUTZUNG DES PRODUKTS AUF EIGENE GEFAHR.
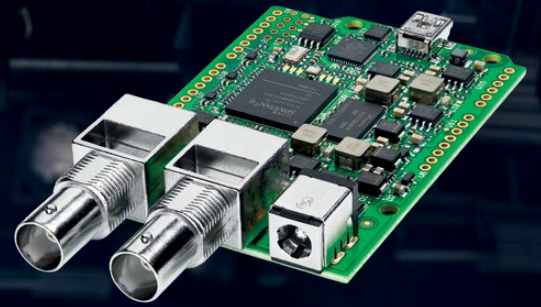
Manual de instalación y funcionamiento

Blackmagic 3G-SDI Shield for Arduino

Febrero 2020

Español

Bienvenido

Gracias por haber adquirido este producto.

Siempre estamos interesados en nuevas tecnologías y nos complace saber que los usuarios emplean nuestros productos SDI de manera creativa. Con la tarjeta 3G-SDI Shield for Arduino, ahora es posible integrar esta plataforma en cualquier dinámica de trabajo basada en dicho formato digital, a fin de obtener una variedad más amplia de opciones de control para los dispositivos de Blackmagic Design.

A modo de ejemplo, los modelos Blackmagic URSA Mini y Blackmagic Studio Camera pueden controlarse desde un mezclador ATEM mediante paquetes de datos integrados en la señal SDI. Incluso si no se utilizan estos dispositivos pero aún se desea contar con la posibilidad de operar las cámaras, la tarjeta 3G-SDI Shield for Arduino facilita la creación de soluciones personalizadas. Esta brinda una plataforma SDI que permite derivar la señal principal proveniente del mezclador y transmitirla directamente a las mismas.

El código empleado para enviar los comandos a la cámara es sencillo, y estos se encuentran detallados en el manual.

Las cámaras pueden manejarse a través de cualquier equipo informático. De manera alternativa, es posible crear controladores dinámicos con botones, perillas y palancas de mando, a efectos de modificar el enfoque o la distancia focal, la apertura del diafragma, el pedestal y el balance de blancos, o bien realizar ajustes cromáticos mediante las herramientas de etalonaje que ofrece la cámara. El diseño de plataformas personalizadas no solo resulta de gran utilidad, sino que además puede ser muy entretenido.

Estamos muy entusiasmados con esta tecnología innovadora y nos gustaría saber más sobre los controladores que hayas creado. ¡No dudes en compartir tus experiencias con nosotros!

Este manual de instrucciones brinda toda la información necesaria sobre el producto. La versión más reciente y las actualizaciones para el sistema operativo interno del dispositivo se encuentran disponibles en nuestra página de soporte técnico: https://www.blackmagicdesign.com/es/support. Al mantener el dispositivo actualizado, siempre podrás contar con las prestaciones más recientes. Por último, no olvides registrarte al descargar las actualizaciones para que podamos mantenerte informado sobre nuevos lanzamientos. Trabajamos constantemente para desarrollar herramientas innovadoras y superarnos, de modo que nos encantaría conocer tu opinión.

Grant Petty
Director ejecutivo de Blackmagic Design

Índice

Blackmagic 3G-SDI Shield for Arduino

Primeros pasos

Colocación de los conectores

La tarjeta Blackmagic 3G-SDI Shield for Arduino viene con dos conectores Berg de ocho pines, uno de diez y uno de seis. Estos se utilizan para montarla sobre la placa Arduino y además permiten añadir otros componentes, incluidos botones, perillas o palancas de mando. Por otra parte, su distribución brinda compatibilidad con tarjetas R3, tales como el modelo Arduino Uno.

Para colocar los conectores:

1 Inserte los pines de cada conector en los agujeros correspondientes situados a ambos lados de la tarjeta. Consulte la siguiente ilustración para obtener más información al respecto.
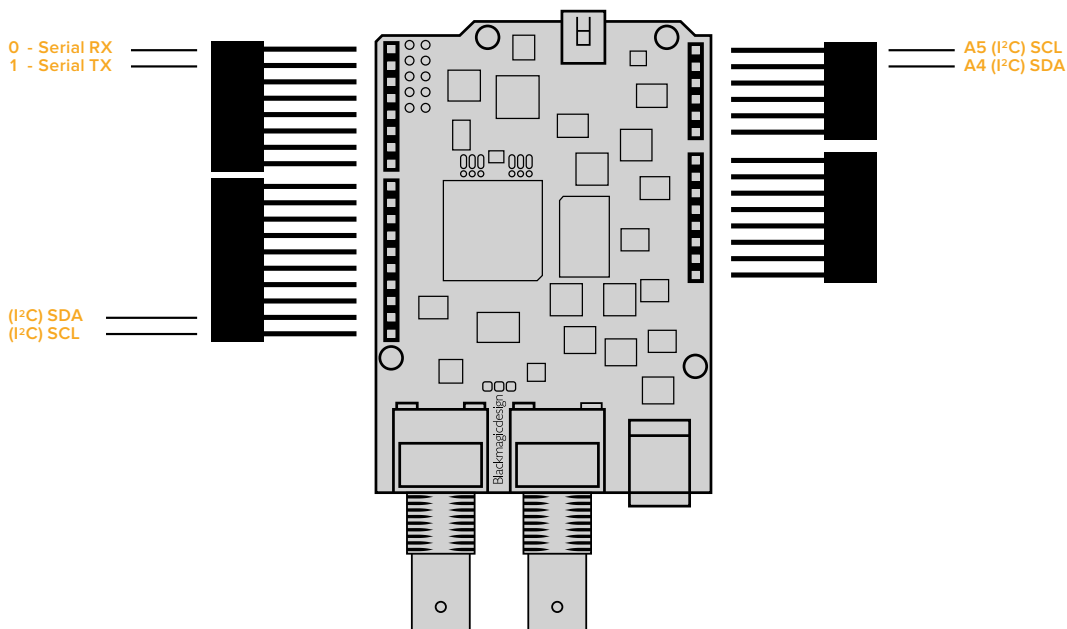
NOTA: La comunicación con la placa se realiza mediante la interfaz serial o I^2C. Recomendamos la segunda opción, dado que permite utilizar el Monitor Serie. Por otra parte, de esta forma los restantes pines estarán disponibles. Seleccione el modo de comunicación al definir el objeto BMDSDIControl en el entorno de programación. Consulte el apartado correspondiente en este manual para obtener más información al respecto.

2 Suelde la base de cada conector a la parte inferior de la tarjeta. Cada patilla metálica debe quedar firme, pero compruebe que la soldadura no esté en contacto con otras adyacentes.

Montaje sobre la placa Arduino

Luego de soldar los conectores, la tarjeta puede colocarse sobre la placa Arduino.

Sostenga cuidadosamente la tarjeta de los costados y compruebe que los conectores en ambos dispositivos estén alineados. A continuación, inserte las patillas metálicas en los agujeros correspondientes y empuje la tarjeta con suavidad. Tenga precaución de no doblar los pines al colocar la tarjeta sobre la placa.
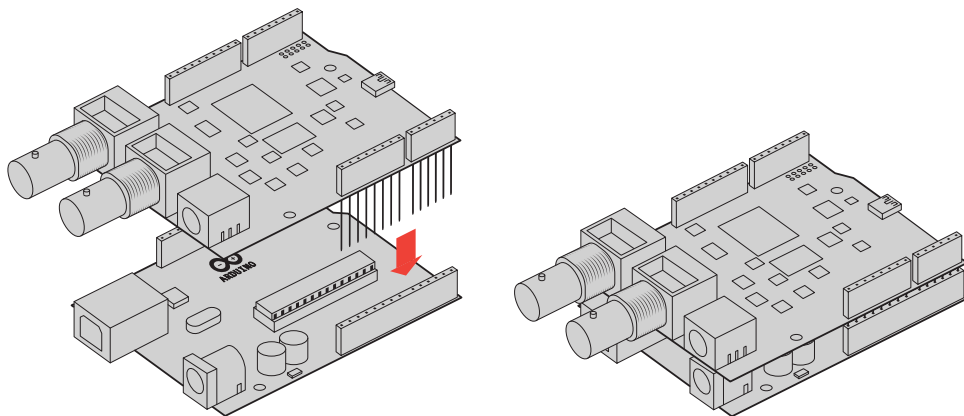
Una vez que los conectores se han acoplado, ambos dispositivos deberían quedar unidos con firmeza.

Conexión del cable de alimentación

Conecte un transformador de 12 voltios para suministrar corriente eléctrica a la tarjeta.

Conexión de dispositivos SDI

Una vez que la tarjeta cuenta con una fuente de suministro eléctrico, es posible conectarla a otros dispositivos tales como mezcladores o cámaras Blackmagic URSA Mini.

1   Conecte la salida principal del mezclador a la entrada SDI en la tarjeta.

2   Conecte la salida SDI de la tarjeta a la entrada PGM de la cámara.

El siguiente diagrama ilustra las conexiones entre los equipos.

Mezclador

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

Entrada PGM

Blackmagic URSA Mini
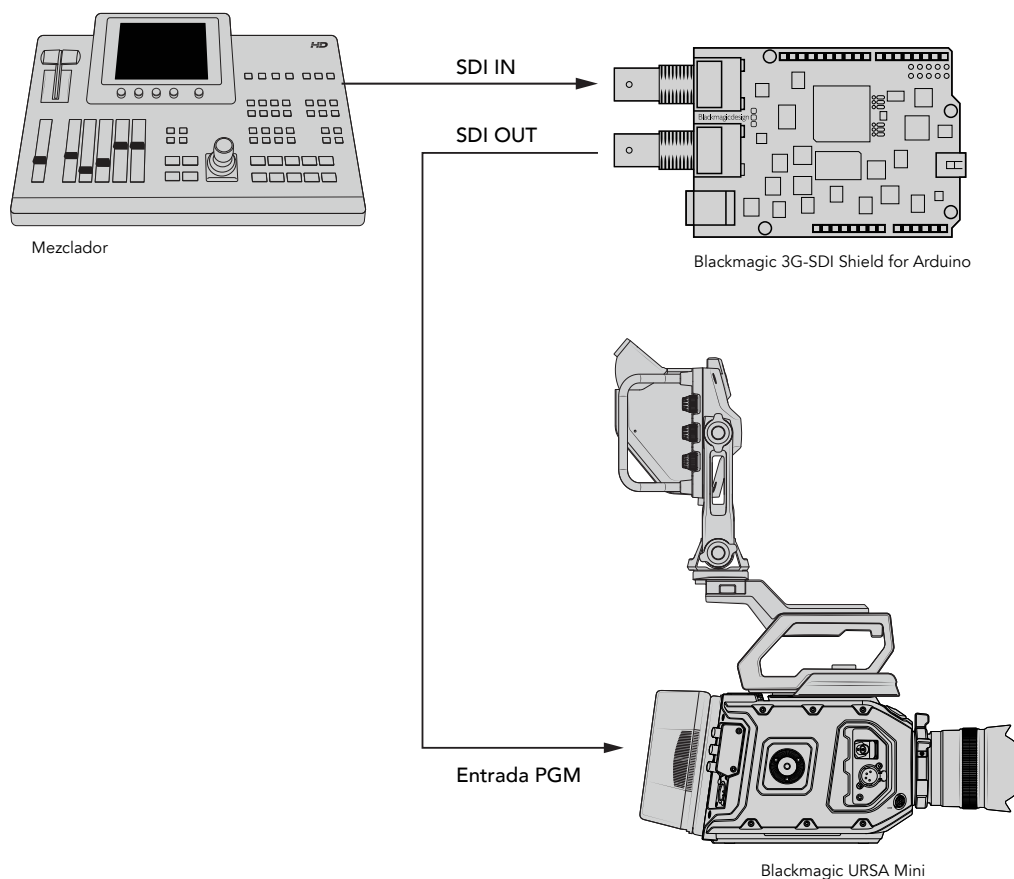
Esto es todo lo que hay que hacer para comenzar a utilizar la tarjeta.

A continuación, basta con instalar el software interno y las librerías, a efectos de programar la placa y emplearla para controlar otros dispositivos.

Continúe leyendo el manual para obtener más información sobre cómo establecer la comunicación entre la tarjeta y la placa Arduino.

SUGERENCIA: La tarjeta también permite controlar otros dispositivos, tales como el monitor Blackmagic Multiview 16. Por ejemplo, cuando se encuentra conectada a la entrada 16 del mismo, es posible mostrar u ocultar un borde indicador alrededor de cada ventana en el modo de visualización simultánea. Al usar el modelo Blackmagic URSA Mini, compruebe que la cuadrícula esté activada para poder ver el indicador en el visor de la cámara. Consulte el manual de la unidad para obtener más información al respecto.

Instalación del software

NOTA: Antes de instalar el programa Blackmagic 3G-SDI Shield for Arduino, descargue la última versión del software Arduino IDE desde el sitio web www.arduino.cc y ejecútela en su equipo informático.

Una vez llevado a cabo este procedimiento, puede instalar el sistema operativo interno de la tarjeta.

Instalación del software interno

El programa Blackmagic Shield for Arduino Setup brinda la posibilidad de actualizar el software interno de la tarjeta. Este permite establecer la comunicación con la placa Arduino y controlarla mediante los archivos de la librería que se instalan junto con el programa. Solo es necesario copiarlos y pegarlos en la carpeta Arduino. Consulte el siguiente apartado del manual para obtener más información al respecto.

Recomendamos descargar la última versión del software y actualizar la tarjeta para poder aprovechar las prestaciones más recientes. Esta se encuentra disponible en la página de soporte técnico de Blackmagic Design: https://www.blackmagicdesign.com/es/support.

Para instalar el software interno mediante Mac OS X:

1   Descargue y descomprima el archivo correspondiente.

2   Abra la imagen de disco resultante y ejecute el instalador. Siga las instrucciones que aparecen en pantalla.

3   Luego de instalar la última versión del programa, conecte la tarjeta a un equipo informático mediante un cable USB.

4   Ejecute la aplicación y siga las instrucciones que aparecen en pantalla para actualizar el software interno. Si no aparece ningún mensaje significa que la actualización finalizó con éxito y no es necesario hacer nada más.

Para instalar el software interno mediante Windows:

1   Descargue y descomprima el archivo correspondiente.

2   Verá una carpeta denominada **Blackmagic Shield for Arduino Utility** que contiene el manual y el instalador. Haga doble clic en el archivo correspondiente a este programa y siga las instrucciones en pantalla para completar la instalación.

3   Luego de instalar la última versión del programa, conecte la tarjeta a un equipo informático mediante un cable USB.

4   Ejecute la aplicación y siga las instrucciones que aparecen en pantalla para actualizar el software interno. Si no aparece ningún mensaje significa que la actualización finalizó con éxito y no es necesario hacer nada más.

Instalación de librerías para Arduino

El código que permite controlar la placa Arduino se denomina «sketch», y los archivos de la librería facilitan su desarrollo. Al finalizar la instalación del programa de configuración de la placa, estos archivos se guardan en una carpeta denominada **Library**. Basta con copiarlos y pegarlos en el directorio de Arduino.

NOTA: El entorno de programación de Arduino debe estar cerrado al instalar las librerías.

Para instalar librerías en Mac OS X:

1 Abra el programa **Blackmagic Shield for Arduino** situado en la carpeta **Aplicaciones**.

2 Abra la carpeta **Library** y haga clic con el botón derecho sobre la carpeta **BMDSDIControl** para copiar su contenido.

3 Abra la carpeta Arduino situada en **Documentos**.

4 Dentro de la misma, verá otra carpeta denominada **libraries**. Pegue el contenido copiado en esta carpeta.

Para instalar librerías en Windows:

1 Abra la carpeta **Blackmagic Shield for Arduino** en la opción **Programas**.

2 Dentro de la misma, verá otra carpeta denominada **Library**. Ábrala y haga clic con el botón derecho sobre la carpeta **BMDSDIControl** para copiar su contenido.

3 Abra la carpeta Arduino situada en **Documentos**.

4 Dentro de la misma, verá otra carpeta denominada libraries. Pegue el contenido copiado en esta carpeta.
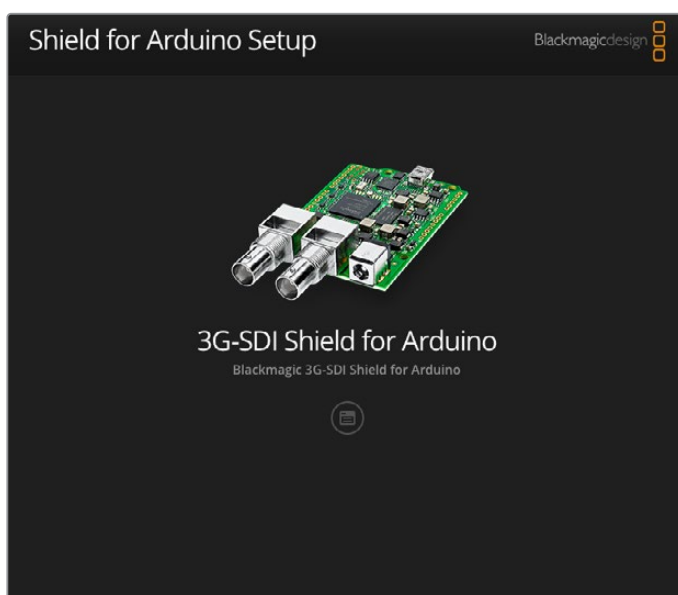
Esto es todo lo necesario para instalar la librería de Blackmagic Design en el equipo informático. Al ejecutar la aplicación Arduino, podrá encontrar un ejemplo de código dentro de las opciones disponibles.

Para ello, basta con seleccionar **Ejemplos** en el menú **Archivo** situado en la barra superior. A continuación, elija la opción **BMDSDIControl** para ver una lista con distintos ejemplos de código que puede utilizar.

Una vez que los archivos de la librería se han instalado en la carpeta correcta, la tarjeta podrá valerse de ellos para establecer la comunicación con la placa Arduino. A tales efectos, solo es necesario realizar la programación correspondiente mediante el software Arduino IDE. Consulte el apartado correspondiente en este manual para obtener más información al respecto.

NOTA: Cuando haya disponible un nuevo archivo de librería con ejemplos adicionales, deberá eliminar la carpeta BMDSDIControl y remplazarla por la nueva versión siguiendo el método descrito anteriormente.

Programa Blackmagic Shield for Arduino
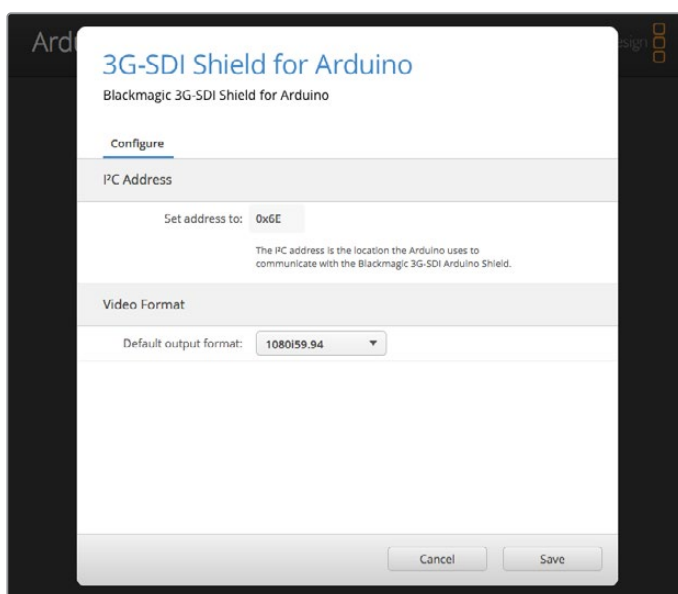
Al instalar el programa Blackmagic Shield for Arduino Setup, podrá modificar ciertos ajustes de la tarjeta, tales como la dirección I²C y el formato de video.

Al instalar el programa Blackmagic Shield for Arduino Setup en el equipo informático, podrá modificar ciertos ajustes de la tarjeta. Estos incluyen la dirección I²C, que identifica la unidad para facilitar la comunicación con la placa Arduino, y el formato de video, que determina el tipo de señal transmitida por el dispositivo.

Dirección I²C

En raras ocasiones, cabe la posibilidad de que otra unidad montada sobre la tarjeta de Blackmagic comparta la misma dirección I²C predeterminada, lo cual generará un conflicto. En este caso, es posible modificar dicho parámetro.

La dirección predeterminada de la tarjeta es 0x6E. Sin embargo, el usuario puede escoger cualquier otra dentro del rango 0x08 - 0x77.

Para cambiar la dirección de la tarjeta:

1   Ejecute el programa Blackmagic Shield for Arduino Setup y haga clic en el ícono de configuración.

2   En la opción **Set address to:**, ingrese la dirección que desea utilizar.

3   Haga clic en el botón **Save**.

Formato de video

Si no se detecta una señal entrante, la opción **Default output format** indica el formato predeterminado. Al conectar una señal, el formato cambiará automáticamente para que coincida con la misma. Cuando el cable se desconecta, esta opción vuelve a mostrar el parámetro por defecto. Para seleccionar un formato determinado, haga clic en el menú desplegable situado a la derecha.

Es posible seleccionar cualquiera de los siguientes formatos:

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Entorno de programación

Arduino facilita en gran medida el desarrollo del código. Este se escribe empleando el lenguaje de programación C. Al utilizar los comandos del protocolo de control Studio Camera, estos se integran en la señal SDI saliente, permitiendo de tal modo controlar unidades Blackmagic URSA Mini o Blackmagic Studio Camera.

Todos los comandos compatibles que pueden emplearse para escribir el código se detallan más adelante en este manual.
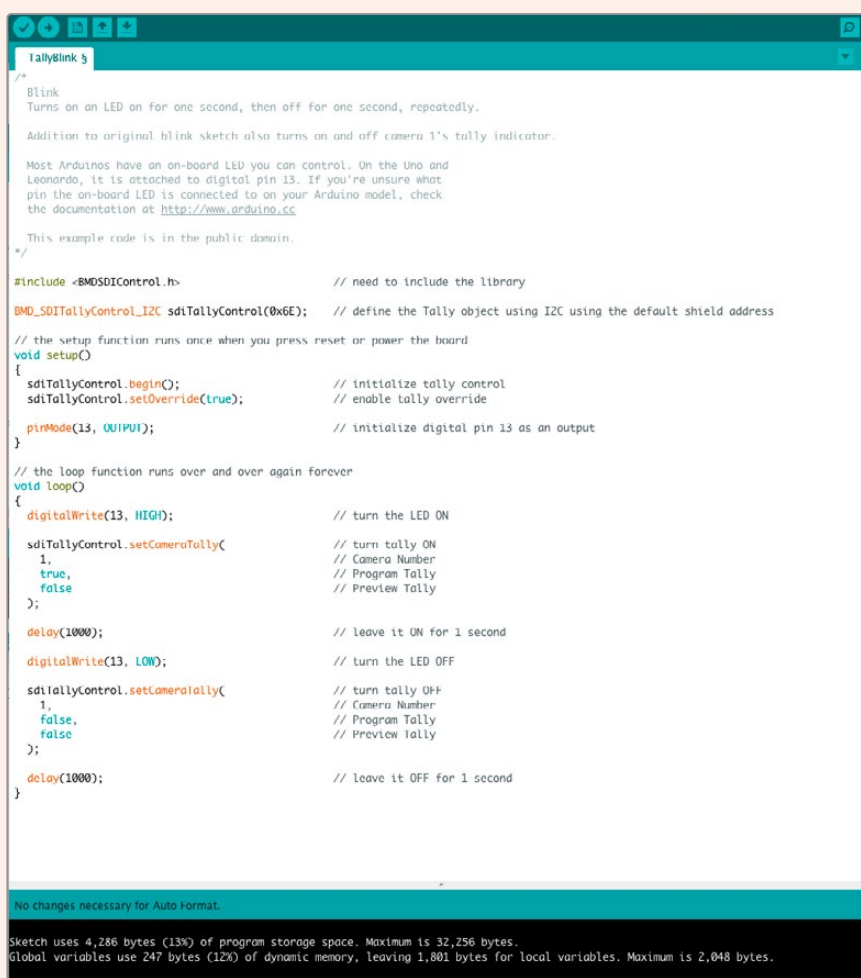
Cómo comprobar el funcionamiento de la tarjeta

Luego de realizar todos los procedimientos descritos en el apartado «Primeros pasos» e instalar tanto el programa de configuración como las librerías, es preciso comprobar si la tarjeta efectivamente se comunica con la placa Arduino.

Una forma rápida de lograr este cometido es ejecutar el código ilustrativo para luces piloto.

Para ello, siga los pasos descritos a continuación:

1 Ejecute la aplicación Arduino IDE.

2 En el menú **Herramientas**, seleccione la placa y el puerto.

3 En el menú **Archivo**, seleccione la opción **Ejemplos**, luego **BMDSDIControl** y a continuación **TallyBlink**.

4 Cargue el programa en la placa.

Este ejemplo proporciona una manera rápida y efectiva de comprobar si la tarjeta funciona correctamente. Los datos pueden transmitirse mediante la interfaz I^2C empleando comandos del protocolo Studio Camera, aunque también proporcionamos librerías particulares a efectos de facilitar la programación.

La luz piloto de la cámara debería parpadear cada un segundo. Esto indica que la comunicación entre la tarjeta y la unidad funciona correctamente.

De no ser así, verifique que se haya asignado el número 1 a la cámara y que la cuadrícula esté activada.

Para obtener más información al respecto, visite la página de soporte técnico en nuestro sitio web. Consulte el apartado *Ayuda* de este manual a fin de ver las distintas formas de obtener asistencia para configurar la tarjeta.

Indicadores luminosos

La tarjeta Blackmagic 3G-SDI Shield for Arduino cuenta con seis indicadores luminosos que permiten confirmar la actividad del dispositivo. Consulte el diagrama que figura a continuación para obtener más información al respecto.
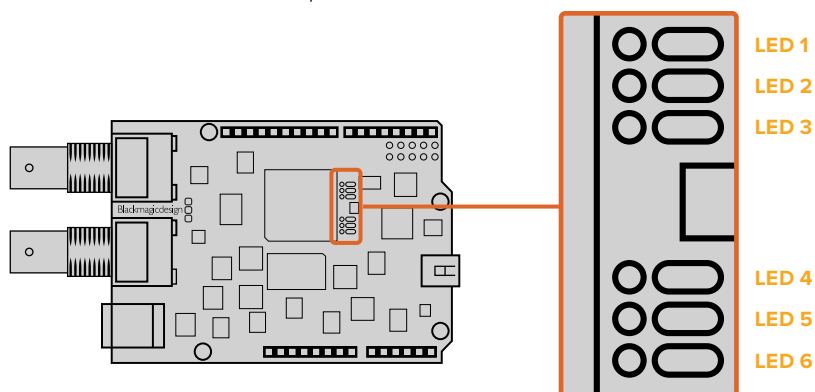
LED 1 - Sistema activo
Este indicador se enciende cuando hay una fuente de alimentación conectada a la tarjeta.

LED 2 - Control anula función activa
Este indicador se enciende cuando el código escrito en el entorno de programación de Arduino permite controlar la cámara.

LED 3 - Luz piloto anula función activa
Este indicador se enciende cuando el código escrito en el entorno de programación de Arduino permite controlar la luz piloto.

LED 5 - Analizador I²C activo
Este indicador se enciende cuando se establece una comunicación entre la tarjeta y la placa Arduino mediante el protocolo I²C.

LED 6 - Analizador serial activo
Este indicador se enciende cuando se establece una comunicación mediante el dispositivo UART.

Al reiniciar la tarjeta, el indicador 1 permanece apagado. Los indicadores 3, 4 y 5 brindan la siguiente información:

LED 3 - Cargando imagen de la aplicación

LED 4 - Iniciando EEPROM

LED 5 - Verificando memoria

A continuación, el indicador 1 se enciende y los restantes indicadores asumen sus funciones predeterminadas.

En caso de que ocurra un error al reiniciar la tarjeta, todos los indicadores parpadearán rápidamente, a excepción de aquel que indica la causa de la falla. Esto brinda la posibilidad de identificarla fácilmente.

Montaje de otros componentes

Es posible crear controladores personalizados con botones, perillas y palancas de mando que faciliten su manejo. A tales efectos, basta con montar estos dispositivos a la tarjeta Blackmagic 3G-SDI Shield for Arduino insertando los conectores en las ranuras correspondientes. No hay limitaciones en cuanto al diseño de los controladores. Incluso se pueden remplazar los circuitos de una unidad de control de cámaras con una solución basada en la plataforma Arduino para adaptarla a las necesidades particulares de un determinado proyecto.
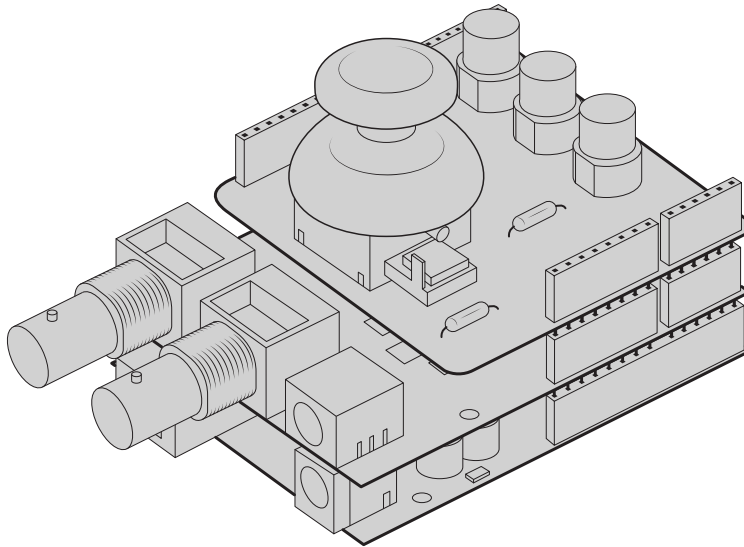
Es posible crear controladores personalizados y montarlos sobre la tarjeta Blackmagic 3G-SDI Shield for Arduino para lograr una mayor interactividad y precisión.

Comunicación con la tarjeta

La comunicación con la tarjeta puede establecerse mediante un bus serial o I^2C. Recomendamos la segunda opción, dado que presenta una menor cantidad de pines. Además, de este modo, el Monitor Serie estará disponible. A su vez, esto permite utilizar otros equipos I^2C con la tarjeta.

Interfaces de alto nivel

La librería contiene dos objetos principales: BMD_SDITallyControl y BMD_SDICameraControl. Estos permiten activar las funcionalidades para la luz piloto y el control de las cámaras en la tarjeta. Ambos pueden crearse mediante el entorno de programación de Arduino para transmitir los comandos correspondientes. Existen distintas variantes de estos objetos para cada bus serial o I^2C.

Interfaz I²C

Para utilizar la interfaz I²C:

```
// NOTE: Must match address set in the setup utility software
const intshieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);
```

Interfaz serial

Para utilizar la interfaz serial:

```
BMD_SDICameraControl_Serial      sdiCameraControl;
BMD_SDITallyControl_Serial       sdiTallyControl;
```

Nótese que la librería determinará la interfaz serial según la velocidad de baudios requerida (38 400). Para enviar mensajes de depuración al Monitor Serie cuando se emplea esta interfaz, modifique este parámetro según corresponda. Al utilizar el Monitor Serie, es posible que aparezcan datos binarios, dado que el entorno de programación no será capaz de distinguir entre los comandos y los mensajes del usuario.

Ejemplo de uso

Una vez creados en el entorno de programación, estos objetos permitirán transmitir comandos a la tarjeta mediante el bus seleccionado para activar determinadas funcionalidades.
A continuación se proporciona un breve ejemplo de código que utiliza la librería mediante el bus I²C.

```
// NOTE: Must match address set in the setup utility software
const intshieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

Las funciones compatibles con estos objetos se detallan más adelante en este manual. Nótese que, en primer lugar, deberá ejecutar la función **begin** para cada objeto antes de transmitir cualquier otro comando.

En el entorno de programación de Arduino se incluyen otros ejemplos ilustrativos. Para acceder a los mismos, seleccione **BMDSDIControl** en la opción **Ejemplos** del menú **Archivo**.

Protocolo de control Studio Camera

Este apartado contiene el protocolo de control detallado en el manual del modelo Studio Camera. Los comandos permiten manejar las cámaras compatibles de Blackmagic Design a través de la tarjeta Blackmagic 3G-SDI Shield for Arduino.

Cada parámetro corresponde a un grupo determinado según se describe a continuación:

Número identificatorio	Grupo
0	Objetivo
1	Video
2	Audio
3	Información
4	Pantalla
5	Luz piloto
6	Referencia
7	Configuración
8	Etalonaje
10	Medios
11	Control PTZ

El número identificatorio se utiliza en el entorno de programación de Arduino para determinar el parámetro que se modifica.

La función sdiCameraControl.writeXXXX se denomina a partir del valor que se desea cambiar, y el sufijo depende del grupo controlado.

A modo de ejemplo, sdiCameraControl.writeFixed16 se emplea para el enfoque, la apertura del diafragma, la distancia focal, la pantalla, la luz piloto y el etalonaje al modificar valores absolutos.

La sintaxis completa para este comando es la siguiente:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

El tipo de operación especifica la acción realizada en el parámetro indicado.

0 = valor asignado. El valor suministrado se asigna al parámetro indicado.

1 = valor de compensación. Cada valor especifica la compensación que se añade al parámetro «Valor».

Por ejemplo:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
        8 = Color Correction group
        0 = Lift Adjust
        0 = assign value
        liftAdjust = setting the value for the RGB and luma levels

Según se describe en el protocolo, liftAdjust es una secuencia de cuatro elementos: RED[0], GREEN[1], BLUE[2] y LUMA[3]. La secuencia completa se envía con el comando.

Los ejemplos de código incluidos en los archivos de la librería contienen elementos descriptivos que explican su funcionamiento.

Protocolo de control SDI de cámaras de Blackmagic

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
Lens	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides. Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDdddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, dddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length				header			command				data							
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Información para desarrolladores

Este apartado brinda toda la información necesaria para crear librerías personalizadas y desarrollar soportes físicos compatibles con la tarjeta Blackmagic 3G-SDI Shield for Arduino.

Codificación física - I²C

La tarjeta funciona a las siguientes velocidades (I²C):

1.   Modo estándar (100 kbit/s)
2.   Máxima velocidad (400 kbit/s)

La dirección I²C predeterminada de la tarjeta es 0x6E.

```
 Pin      | Función
-------------------- |----------------------------
 A4       | Datos seriales (SDA)
 A5       | Reloj serial (SCL)
```

Protocolo I²C (escritura):

> (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

Protocolo I²C (lectura):

> (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

La carga útil máxima (indicada como **VAL** en los ejemplos anteriores) de lectura y escritura (luego de la dirección del registro interno) en una transacción es de 255 bytes.

Codificación física - UART

La velocidad de transmisión en baudios de la tarjeta al emplear este modo es de 115200 (formato 8-N-1).

```
 Pin      | Función
-------------------- |----------------------------
 IO1      | Transmisión serial (TX)
 IO0      | Recepción serial (RX)
```

Protocolo UART (escritura):

> [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

Protocolo UART (lectura):

> [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

La carga útil máxima (indicada como **VAL** en los ejemplos anteriores) de lectura y escritura (según el parámetro **LENGTH**) en una transacción es de 255 bytes.

Mapa de direcciones del registro

El mapa de direcciones de la tarjeta se detalla a continuación:

Dirección	Nombre	L/E	Descripción
0x0000 - 0x0003	IDENTITY	L	Identificador del hardware
0x0004 - 0x0005	HWVERSION	L	Versión del hardware
0x0006 - 0x0007	FWVERSION	L	Versión del firmware
0x1000	CONTROL	L/E	Control del sistema
0x2000	OCARM	L/E	Anular control SDI (ARM)
0x2001	OCLENGTH	L/E	Anular control SDI (longitud)
0x2100 - 0x21FE	OCDATA	L/E	Anular control SDI (datos)
0x3000	ICARM	L/E	Control SDI entrante (ARM)
0x3001	ICLENGTH	L	Control SDI entrante (longitud)
0x3100 - 0x31FE	ICDATA	L	Control SDI entrante (datos)
0x4000	OTARM	L/E	Anular SDI luz piloto (ARM)
0x4001	OTLENGTH	L/E	Anular SDI luz piloto (longitud)
0x4100 - 0x41FE	OTDATA	L/E	Anular SDI luz piloto (datos)
0x5000	ITARM	L/E	SDI entrante luz piloto (ARM)
0x5001	ITLENGTH	L	SDI entrante luz piloto (longitud)
0x5100 - 0x51FE	ITDATA	L	SDI entrante luz piloto (datos)

Los campos numéricos con múltiples bytes se almacenan en orden de relevancia, según el sistema «little-endian». Las direcciones no utilizadas se reservan y equivalen a 0.

Registro: IDENTITY (Identificador de la placa)

[IDENTITY]
31   0

SDIC de la cadena ASCII **Identity:**   (i.e. 0x43494453) en hexadecimales.

Registro: HWVERSION (Versión del hardware)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Revisión, componente principal.

Version Minor: Revisión, componente secundario.

Registro: FWVERSION (Versión del firmware)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Revisión, componente principal.

Version Minor: Revisión, componente secundario.

Registro: CONTROL (Control del sistema)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved:	Siempre 0.
Override Output:	Cuando este valor es 1, la señal SDI entrante (si existe) se descarta y la tarjeta genera su propia señal en la salida SDI. Cuando este valor es 0, la señal se transmite a través de la tarjeta o se genera una señal SDI si el dispositivo no la detecta.
Reset Tally:	Cuando este valor es 1, los últimos datos recibidos relativos a la luz piloto se copian al registro correspondiente. Borrado automático mediante el hardware.
Override Tally:	Cuando este valor es 1, la información suministrada por el usuario anula los datos recibidos. Cuando este valor es 0, los datos se transmiten sin modificación alguna.
Override Control:	Cuando este valor es 1, la información de control suministrada por el usuario anula los datos recibidos. Cuando este valor es 0, los datos se transmiten sin modificación alguna.

Registro: OCARM (Salida control - ARM)

[RESERVED] [ARM]
7   1   0

Reserved:	Siempre 0.
Arm:	Cuando este valor es 1, los datos ARM se transmiten en el siguiente fotograma. Borrado automático una vez enviada la información.

Registro: OCLENGTH (Salida control - longitud)

[LENGTH]
7   0

Length:	Longitud en bytes de los datos enviados.

Registro: OCDATA (Salida control - carga útil)

[CONTROL DATA]
255*8-1   0

Control Data:	Datos de control que se integrarán en un fotograma.

Registro: ICARM (Entrada control - ARM)

[RESERVED] [ARM]
7   1   0

Reserved:	Siempre 0.
Arm:	Cuando este valor es 1, los datos ARM se reciben en el siguiente fotograma. Borrado automático una vez leído el paquete de control.

Registro: ICLENGTH (Entrada control - longitud)

[LENGTH]
7   0

Length:	Longitud en bytes de los datos en _ICDATA_. Se establece automáticamente al almacenar un nuevo paquete en la memoria caché.

Registro: ICDATA (Entrada control - carga útil)

[CONTROL DATA]
255*8-1   0

Control Data: Últimos datos extraídos de un fotograma luego de restablecer
ICARM.ARM.

Registro: OTARM (Salida luz piloto - ARM)

[RESERVED] [ARM]
7   1   0

Reserved: Siempre 0.

Arm: Cuando este valor es 1, los datos salientes relativos a la luz
piloto se transmiten a partir del siguiente fotograma de manera
continua hasta que se establezca un nuevo valor. Borrado
automático una vez enviada la información, al menos en un
fotograma.

Registro: OTLENGTH (Salida luz piloto - longitud)

[LENGTH]
7   0

Length: Longitud en bytes de los datos enviados en OTDATA.

Registro: OTDATA (Salida luz piloto - datos)

[TALLY DATA]
255*8-1   0

Tally Data: Datos que deben integrarse en un próximo fotograma (1 byte por
cámara). El 0 indica la señal principal, mientras que 1 hace
referencia al anticipo.

Registro: ITARM (Entrada luz piloto - ARM)

[RESERVED] [ARM]
7   1   0

Reserved: Siempre 0.

Arm: Cuando este valor es 1, los datos se reciben en el siguiente
fotograma. Borrado automático una vez leída la información.

Registro: ITLENGTH (Entrada luz piloto - longitud)

[LENGTH]
7   0

Length: Longitud en bytes de los datos en _ITDATA_. Se establece
automáticamente al almacenar un nuevo paquete en la
memoria caché.

Registro: ITDATA (Entrada luz piloto - datos)

[TALLY DATA]
255*8-1   0

Tally Data: Últimos datos extraídos de un fotograma luego de restablecer
ITARM.ARM (1 byte por cámara). El 0 indica la señal principal,
mientras que 1 hace referencia al anticipo.

Ayuda

Ayuda

La tarjeta Blackmagic 3G-SDI Shield for Arduino es un dispositivo que permite desarrollar soluciones personalizadas según los requerimientos de cada usuario.

Visite nuestra página de soporte técnico para obtener información actualizada al respecto.

Página de soporte técnico

Las versiones más recientes del manual, el software de los dispositivos y el material de apoyo se encuentran disponibles en el centro de soporte técnico de Blackmagic Design: www.blackmagicdesign.com/es/support

Foro de programadores

En el foro para desarrolladores de Arduino es posible encontrar respuestas a las preguntas sobre el entorno de programación. Existe una gran cantidad de sitios donde el usuario podrá aclarar dudas relativas a esta plataforma o incluso encontrar personas dispuestas a implementar soluciones hechas a medida.

Foro de Blackmagic Design

Este foro permite compartir ideas creativas y constituye un recurso útil para obtener más información sobre nuestros productos. Además, brinda la posibilidad de encontrar rápidamente respuestas proporcionadas por el personal de Blackmagic Design u otros expertos en la materia. Para acceder al mismo, visite el sitio https://forum.blackmagicdesign.com.

Cómo comprobar la versión instalada del software

Para comprobar la versión del programa de configuración instalada en su equipo informático, seleccione la opción **About Blackmagic Shield for Arduino Setup**.

- En Mac OS X, abra el programa desde la carpeta de aplicaciones. Seleccione la opción **About Blackmagic Shield for Arduino Setup** en el menú superior para ver el número de versión.

- En Windows 7, abra el programa desde el menú **Inicio**. Seleccione **Help** y haga clic en la opción **About Blackmagic Shield for Arduino Setup** para ver el número de versión.

- En Windows 8, abra el programa desde la página de inicio. Seleccione **Help** y haga clic en la opción **About Blackmagic Shield for Arduino Setup** para ver el número de versión.

Cómo obtener las últimas actualizaciones

Luego de verificar la versión instalada en el equipo informático, visite nuestro centro de soporte técnico para comprobar si existen actualizaciones disponibles. Aunque generalmente es recomendable instalar el software más reciente, evite realizar cualquier modificación si se encuentra en medio de un proyecto importante.

Garantía
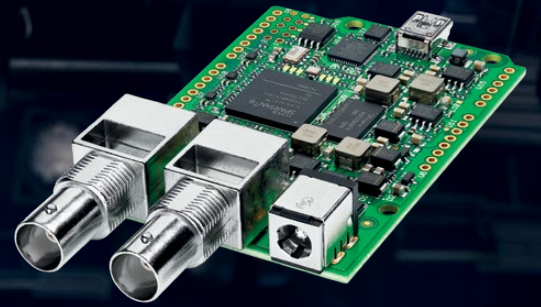
Garantía limitada por 12 meses

Blackmagic Design garantiza que el producto adquirido no presentará defectos en los materiales o en su fabricación por un período de 12 meses a partir de su fecha de compra. Si un producto resulta defectuoso durante el período de validez de la garantía, Blackmagic Design podrá optar por sustituirlo o repararlo sin cargo alguno por concepto de piezas y/o mano de obra.

Para acceder al servicio proporcionado bajo los términos de esta garantía, el Cliente deberá dar aviso del defecto a Blackmagic Design antes de su vencimiento y encargarse de los arreglos necesarios para la prestación del mismo. El Cliente será responsable por el empaque y el envío del producto defectuoso al centro de servicio técnico designado por Blackmagic Design y deberá abonar las tarifas postales por adelantado. El Cliente será responsable por todos los gastos de envío, seguros, aranceles, impuestos y cualquier otro importe que surja con relación a la devolución de productos por cualquier motivo.

Esta garantía carecerá de validez ante defectos o daños causados por un uso indebido del producto, o por falta de cuidado y mantenimiento. Blackmagic Design no tendrá obligación de prestar el servicio estipulado en esta garantía para (a) reparar daños provocados por intentos de personal ajeno a la empresa de instalar, reparar o realizar un mantenimiento del producto; (b) reparar daños resultantes del uso de equipos incompatibles o conexiones a los mismos; (c) reparar cualquier daño o mal funcionamiento provocado por el uso de piezas o repuestos no suministrados por la empresa; o (d) brindar servicio técnico a un producto que haya sido modificado o integrado con otros productos, cuando dicha modificación o integración tenga como resultado un aumento de la dificultad o el tiempo necesario para reparar el producto. ESTA GARANTÍA OFRECIDA POR BLACKMAGIC DESIGN SUSTITUYE CUALQUIER OTRA GARANTÍA, EXPRESA O IMPLÍCITA. POR MEDIO DE LA PRESENTE, BLACKMAGIC DESIGN Y SUS DISTRIBUIDORES RECHAZAN CUALQUIER GARANTÍA IMPLÍCITA DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO PARTICULAR. LA RESPONSABILIDAD DE BLACKMAGIC DESIGN EN CUANTO A LA REPARACIÓN O SUSTITUCIÓN DE PRODUCTOS DEFECTUOSOS CONSTITUYE UNA COMPENSACIÓN COMPLETA Y EXCLUSIVA PROPORCIONADA AL CLIENTE POR CUALQUIER DAÑO INDIRECTO, ESPECIAL, FORTUITO O EMERGENTE, AL MARGEN DE QUE LA EMPRESA O SUS DISTRIBUIDORES HAYAN SIDO ADVERTIDOS CON ANTERIORIDAD SOBRE LA POSIBILIDAD DE TALES DAÑOS. BLACKMAGIC DESIGN NO SE HACE RESPONSABLE POR EL USO ILEGAL DE EQUIPOS POR PARTE DEL CLIENTE. BLACKMAGIC NO SE HACE RESPONSABLE POR DAÑOS CAUSADOS AL EMPLEAR ESTE PRODUCTO. EL USUARIO UTILIZA EL PRODUCTO BAJO SU PROPIA RESPONSABILIDAD.

安装操作手册

Blackmagic 3G-SDI Shield for Arduino

2020年2月

中文

欢迎辞

感谢您购买新品Blackmagic 3G-SDI Shield for Arduino。

我们一直都密切关注前沿科技，并且热衷于为我们的SDI产品拓宽应用方面的创新思维。这款3G-SDI Shield for Arduino能够将Arduino整合到您的SDI工作流程当中，为您的Blackmagic Design设备带来更多控制方案。

举例说明，您可以将数据包嵌入到SDI信号上，从而实现通过ATEM切换台来操控Blackmagic URSA Mini和Blackmagic Studio Camera控制方案。如果您不使用ATEM切换台，但依然想要控制Blackmagic摄影机，还可以使用这款3G-SDI Shield for Arduino来打造自己的控制方案。这款盾板可为您提供搭建SDI的平台，将来自切换台的节目返送信号通过盾板发送到Blackmagic摄影机的节目输入上。

编写代码并将命令发送到摄影机非常简单，而且本手册包含了所有支持的命令。

您可以通过计算机控制摄影机，或者在盾板上添加按钮、旋钮以及操纵杆等工具，制作动态硬件控制器用来调整镜头对焦和变焦、光圈设置、消隐脉冲电平、白平衡控制以及摄影机内置的强大调色工具等。自行定制您的控制器不仅能有助于制作，同时还是一个充满趣味的过程！

这是一项令人激动的技术，希望您能使用3G-SDI Shield for Arduino构建出各类SDI控制器，并与我们分享心得！

本操作手册包含使用Blackmagic 3G-SDI Shield for Arduino所需的全部信息。请登陆公司网站 www.blackmagicdesign.com/cn并进入支持页面获得这款盾板的最新版操作手册及其内部软件更新。请注意定期更新您的软件以便获得最新功能！下载软件时，请注册您的相关信息，以便我们发布新软件时能及时通知您。我们会不断地增加新功能，提升产品性能，同时也希望聆听您的宝贵意见！

Grant Petty

Grant Petty
Blackmagic Design首席执行官

目录

Blackmagic 3G-SDI Shield for Arduino

入门

安装并焊接排针连接器

Blackmagic 3G-SDI Shield for Arduino内附4个堆叠式排针连接器，包括两个8针连接器、一个10针连接器以及一个6针连接器。排针连接器相当于电桥连接器，您可通过它将盾板安装到Arduino开发板上。由于连接器可相互堆叠，因此您还可以再添加其他盾板来安装更多元件，如控制按钮、旋钮以及操纵杆等。这些排针连接器的布局可实现R3式连接方案，比如Arduino UNO。

如何为盾板安装排针连接器：

1 将每个排针连接器上的引脚插入Blackmagic 3G-SDI Shield盾板每一侧相应的针孔内。请参考下图的示范完成排针连接器布局。

0 - Serial RX
1 - Serial TX

A5 (I²C) SCL
A4 (I²C) SDA

(I²C) SDA
(I²C) SCL

Blackmagicdesign
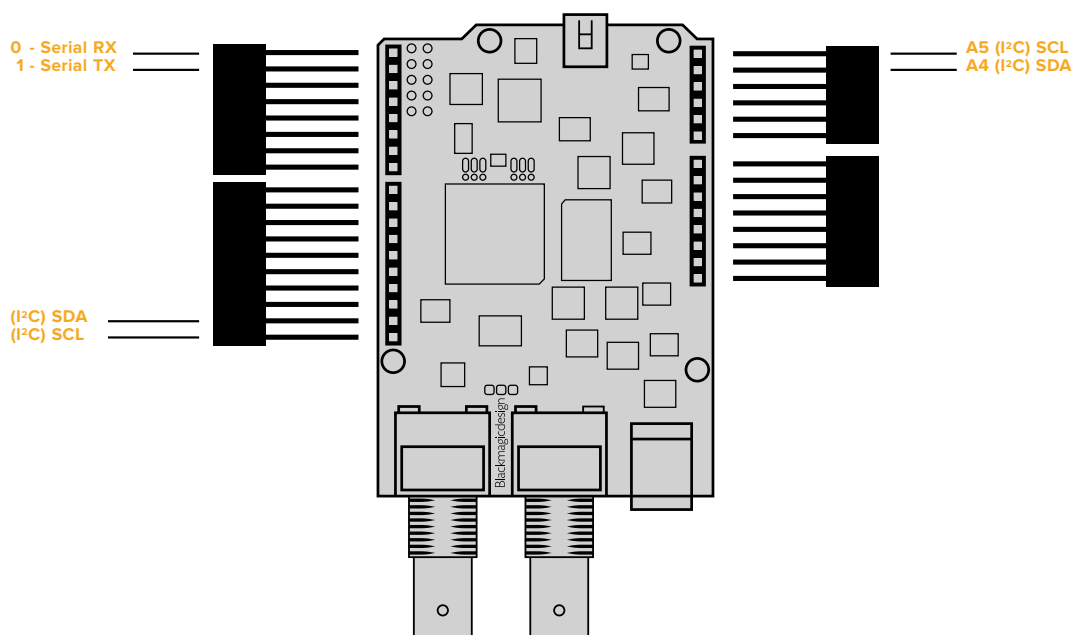
> **备注** 连接盾板时，可使用I²C通信或串行通信。我们推荐您使用I²C通信，因为这一方案可使用串口监视器，从而能更好地利用其它引脚。您可以在Sketch中定义BMDSDIControl时选择通信模式。请阅读"Blackmagic 3G-SDI Shield for Arduino盾板的通信"章节了解详细信息。

2 将每个连接器底部的引脚焊接到盾板反面。请确保每个引脚与针孔焊接牢固，并且不接触到邻近的引脚。

为有助于确保盾板上的所有引脚都能与Arduino开发板连接器上的排针母座对齐，请先焊接每个排针连接器上的一个引脚。然后，将盾板放置于Arduino开发板上，检查引脚对齐情况。如果需要调整任何排针连接器，您可以预热相应拍针连接器上的焊接点来改善对齐情况。这比完成所有排针焊接之后再进行调整要容易得多。

安装Arduino开发板

将排针连接器焊接到盾板上之后，您就可以将3G-SDI盾板安装到Arduino开发板上了。

小心握住盾板两侧，将它的连接器引脚与Arduino开发板的连接器对齐，并轻轻推按到连接器插槽内。请注意，切勿在安装过程中将任何引脚压弯。
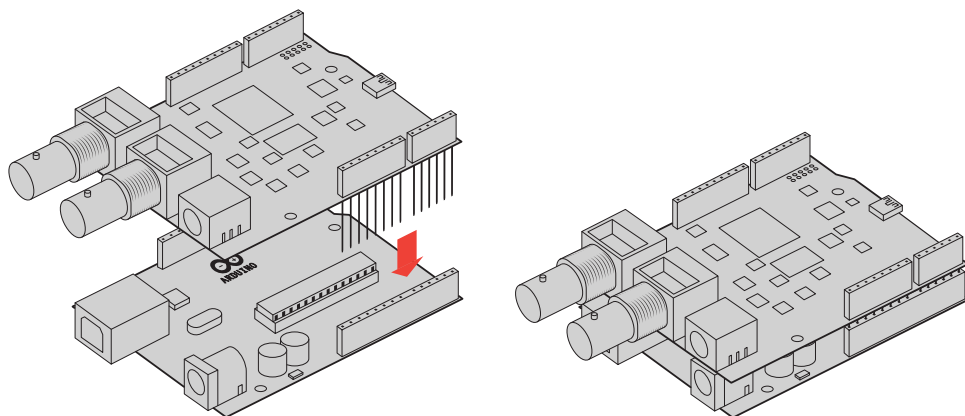
完全插入之后，Blackmagic盾板和Arduino开发板之间应已紧密牢固连接。

连接电源

要开启Blackmagic 3G-SDI Shield for Arduino，只需将12V电源适配器连接到Blackmagic盾板的12V电源输入即可。

备注 将Arduino开发板连接到电源无法为Blackmagic盾板提供足够电源，但是将Blackmagic盾板连接到电源则可以同时为Arduino供电。因此，请确保您只将电源连接到Blackmagic盾板。

连接SDI设备

连接电源之后，将您的Blackmagic 3G-SDI Shield盾板插入SDI设备。例如，将其连接到切换台及Blackmagic URSA Mini：

1   将切换台的节目输出接口连接到Blackmagic 3G-SDI Shield盾板的SDI输入接口。

2   将Blackmagic 3G-SDI Shield盾板的SDI输出连接到Blackmagic URSA Mini上标有PGM的SDI节目输入接口。

连接示意图如下：

切换台

SDI输入

SDI输出

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' 输入

Blackmagic URSA Mini
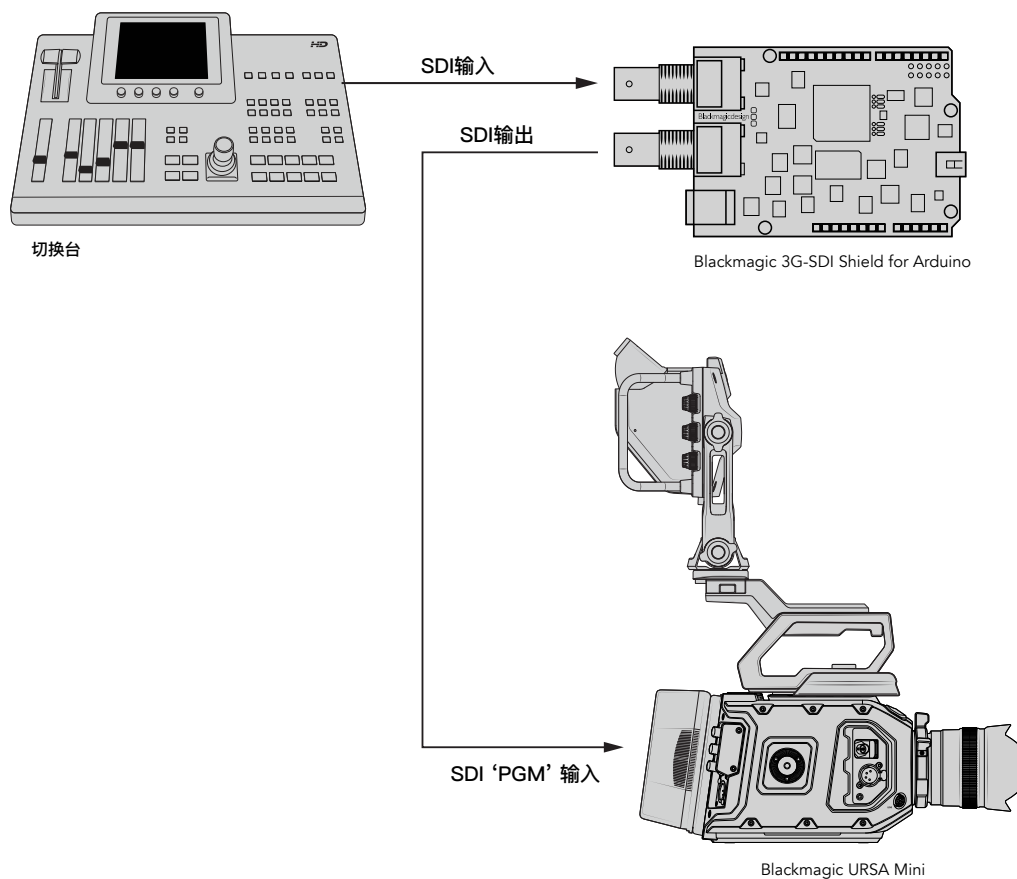
一切准备就绪！

将盾板安装到Arduino开发板并连接了电源和SDI设备时候，您就可以开始安装内部软件和库文件，对Arduino软件进行编程并开始使用盾板来控制您的设备了。

请继续阅读本操作手册，了解如何为盾板安装内部软件，以及Arduino库文件的安装位置，以便盾板和Arduino建立通信。

提示　您还可以使用Blackmagic 3G-SDI Shield for Arduino来控制其他Blackmagic Design产品，例如Blackmagic MultiView 16。举例说明，当您的盾板连接到输入16上时，您可以在多画面分割上显示Tally边框。使用Blackmagic URSA Mini摄影机时，请确保网格叠加处于开启状态，从而启用摄影机寻像器上的Tally显示。请参考您的Blackmagic Camera手册获取更多信息。

软件安装

> **备注** 安装Blackmagic Shield for Arduino实用程序前，请先从网站www.arduino.cc 下载最新版Arduino IDE软件并安装到您的计算机上。
>
> 安装Arduino软件后，您可以安装Blackmagic 3G-SDI Shield的内部软件。

安装内部软件

Blackmagic Shield for ArduinoSetup可用来更新盾板的内部软件。这一内部软件可与Arduino开发板进行通信，并使用Arduino的库文件来控制开发板。这些库文件会和设置软件一同被安装，您只需将包含这些文件的文件夹复制并粘贴到Arduino应用程序的文件夹内即可。关于库文件以及如何安装这些库文件的信息，请参考本手册下一章节。

建议您下载最新版Blackmagic Shield for Arduino软件并更新盾板，以便获得最新功能和改进。请到Blackmagic Design的支持中心下载最新版本，网址：www.blackmagicdesign.com/cn/support。

在Mac OS X系统上安装内部软件步骤如下：

1　下载并解压缩Blackmagic Shield for Arduino软件。

2　打开生成的硬盘图标并运行Blackmagic Shield for Arduino安装程序。根据屏幕提示完成安装。

3　安装完最新版Blackmagic Shield for Arduino之后，为您的Blackmagic盾板连接电源，然后通过USB线缆连接到计算机。

4　运行该实用程序，根据屏幕提示更新盾板的内部软件。如果系统未弹出任何提示信息，即表示当前内部软件已是最新版本，无需升级。

在WIndows系统上安装内部软件步骤如下：

1　下载并解压缩Blackmagic Shield for Arduino软件。

2　您会看到一个名为"Blackmagic Shield for Arduino"的文件夹，该文件夹中含有本操作手册以及Blackmagic Shield for Arduino安装程序。双击该安装程序，并根据屏幕提示完成安装。

3　安装完最新版Blackmagic Shield for Arduino安装程序之后，为您的Blackmagic盾板连接电源，然后通过USB线缆连接到计算机。

4　运行该实用程序，根据屏幕提示更新盾板的内部软件。如果系统未弹出任何提示信息，即表示当前内部软件已是最新版本，无需升级。

安装Arduino库文件

为控制Arduino所编写的程序叫做Sketch, 您的Blackmagic 3G-SDI Shield for Arduino可使用Arduino库文件使Sketch的编写更为便捷。为盾板安装了设置软件之后, 库文件也会被安装到一个名为"Library"的文件夹中。您只需将包含这些库文件的文件夹复制并粘贴到Arduino库文件夹内即可。

> **备注** 安装库时, 请关闭Arduino IDE软件。

在Mac OS X系统下安装库文件:
1. 打开"应用程序"文件夹中的"Blackmagic Shield for Arduino"。
2. 打开"Library"文件夹, 右键点击并复制名为 "BMDSDIControl"的文件夹。
3. 打开计算机"文件"文件夹中的Arduino文件夹。
4. 您将会看到一个名为"Libraries"的子文件夹。将"BMDSDIControl"文件夹粘贴至这个"Libraries"文件夹中。

在Windows系统下安装库文件:
1. 依次打开"程序"/"Blackmagic Shield for Arduino"文件夹。
2. 您将会看到一个名为"Library"的子文件夹。打开该文件夹, 右键点击并复制名为"BMDSDIControl"的文件夹。
3. 打开计算机"文件"文件夹中的Arduino文件夹。
4. 您将会看到一个名为"Libraries"的子文件夹。将"BMDSDIControl"文件夹粘贴至这个"Libraries"文件夹中。

Blackmagic Design库文件就会被安装到您的计算机上了。运行Arduino软件时, 您还可以选择Blackmagic Design示例Sketch。

只需进入Arduino软件菜单栏的"文件"下拉菜单选择"Examples"即可。选择BMDSDIControl后, 您会看到可使用的Sketch示例列表。

库文件被安装到正确的文件夹之后, 您的盾板就可以使用这些文件来与Arduino开发板建立通信。只需对Arduino IDE软件进行编程即可。详情请参阅"Arduino Sketch编程"部分的内容。

> **备注** 如果将来发布更新版示例库文件, 您需要删除旧的BMDSDIControl文件夹, 使用上面描述的方法将其替换成新的文件夹。

Blackmagic Shield for Arduino Setup
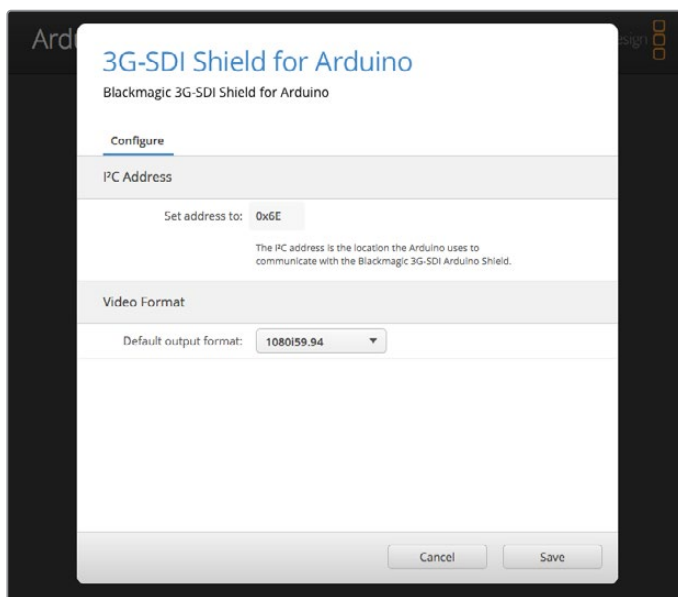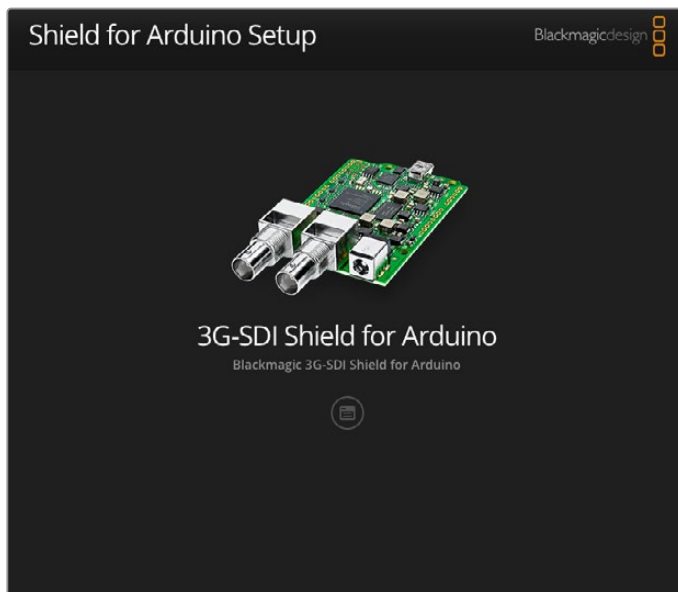
Blackmagic Shield for Arduino Setup软件可用来更改盾板的各项设置，比如I²C地址以及视频输出格式等

在计算机上安装了Blackmagic Shield for Arduino Setup之后，您就可以更改盾板的各项设置了。这些设置包括I²C地址，它可用于识别盾板，以便Arduino开发板可与其进行通信；以及视频格式，它可设置盾板的输出格式。

I²C地址

极少情况下，另一块安装到Blackmagic盾板上的盾板可能会使用与您盾板默认地址相同的I²C地址，这时就会造成冲突。发生此类情况时，您可以更改盾板的默认地址设置。

盾板的默认地址为0x6E，但是您可以将其更改为0x08到0x77之间。

如何更改盾板的地址：

1 运行Blackmagic Shield for Arduino Setup，点击您盾板的设置图标。

2 在"Set Address to:"（将地址设置为）编辑框中键入您想要使用的地址。

3 点击"Save"（保存）。

视频格式

没有连接输入时，实用程序会选择默认的输出格式。检测到输入时，则输出将使用与该输入相同的格式。移除输入时，相应的输出将还原到实用程序默认选择的输出格式。您可以点击"Default Output Format"（默认输出格式）下拉菜单并选择相应的视频格式进行更改。

您可以从以下视频输出格式中进行选择：

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Arduino Sketch编程

Arduino软件的程序称为Sketch，它非常容易编写！Sketch使用常见的C编程语言。使用Studio Camera Control Protocol控制协议命令编写Sketch时，盾板可将这些命令嵌入到SDI输出上，以便您可以控制您的Blackmagic URSA Mini或Blackmagic Studio Camera。

本手册的"Studio Camera Control Protocol"章节会介绍所有支持的命令，您可以将这一协议中的命令应用到您的Sketch编写中。

测试您的Blackmagic盾板和库安装情况

根据"入门"章节中的介绍顺利完成所有连接，并且安装完设置软件和库文件之后，您需要检查盾板是否和Arduino开发板建立通信，以确保一切运行正常。

一个快速的检查方法就是打开并运行我们提供的Tally闪烁（Blink）示例Sketch。

具体步骤如下：

1 运行Arduino IDE软件。

2 到"工具"菜单中选择Arduino开发板以及端口号码。

3 到"文件"菜单下，选择"示例"/"BMDSDIControl"，再选择名为"TallyBlink"的Sketch程序。

4 将这一Sketch程序上传到开发板。

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                  // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);   // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                   // initialize tally control
  sdiTallyControl.setOverride(true);         // enable tally override

  pinMode(13, OUTPUT);                       // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                    // turn the LED ON

  sdiTallyControl.setCameraTally(
    1,                                       // Camera Number
    true,                                    // Program Tally
    false                                    // Preview Tally
  );

  delay(1000);                               // leave it ON for 1 second

  digitalWrite(13, LOW);                     // turn the LED OFF

  sdiTallyControl.setCameraTally(
    1,                                       // Camera Number
    false,                                   // Program Tally
    false                                    // Preview Tally
  );

  delay(1000);                               // leave it OFF for 1 second
}
```

No changes necessary for Auto Format.

Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.

Tally Blink示例Sketch是测试Arduino盾板的快捷方法。原始数据可使用来自Studio Camera Protocol协议文件中的命令通过I²C发送到您的盾板。同时，我们也提供了自定义库帮助您更简单地进行Sketch编程。

此时，您应可看到Blackmagic Studio Camera上的Tally提示灯开始闪烁，且闪烁间隔为一秒。如果您看到Tally灯闪烁，就可以肯定您的Blackmagic盾板已与Arduino建立通信，一切运行正常。

如果Tally灯没有闪烁，请检查Blackmagic摄影机的Tally号码是否设为1，并且摄影机的网格叠加是否处于开启状态。

如果您需要帮助，请登陆网址www.blackmagicdesign.com/cn/support访问Blackmagic Design支持中心。请参考本手册的帮助章节获取更多关于盾板设置以及调试方面的帮助信息。

LED提示灯

您的Blackmagic 3G-SDI Shield for Arduino设有六个LED提示灯，可用来确认盾板的各项活动情况，比如电源、UART、I²C和SPI通信，还能用于提示您何时启用了Tally与摄影机控制优先功能。
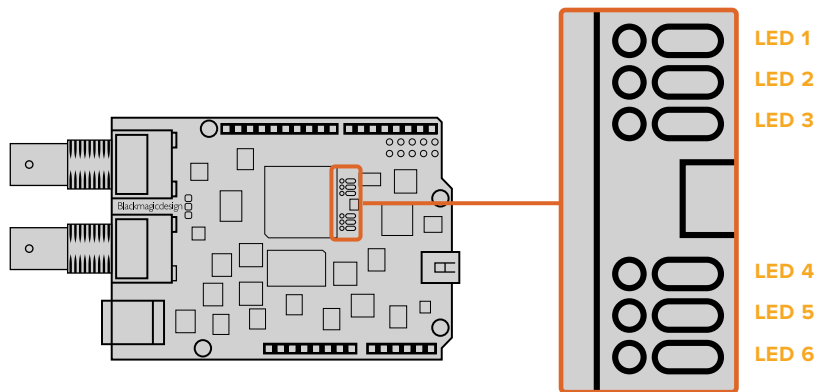
LED 1 - 系统开启
可在盾板连接电源时亮起。

LED 2 - 启用控制优先
可在您启用了Arduino Sketch中的摄影机控制时亮起。

LED 3 - 启用Tally优先
可在您启用了Arduino Sketch中的Tally功能时亮起。

LED 5 - I²C语法分析程序忙
可在检测到盾板与Arduino使用I²C协议进行通信时亮起。

LED 6 - 串行语法分析器忙
可在检测到UART通信时亮起。

当您的Blackmagic盾板启动时，电源提示灯将保持熄灭状态，LED 3、4、5将用来提示以下活动：

LED 3 - 正在加载应用程序图像

LED 4 - 正在初始化EEPROM

LED 5 - 正在检查内存

成功启动后，电源LED提示灯将亮起，所有LED提示灯会回到其操作时各自对应的标准功能。

极少情况下会出现启动失败的情况，此时故障LED提示灯会反复闪烁，所有其他LED灯保持熄灭状态，方便您进行故障排查。

安装盾板元件

如果您想要构建自己的硬件控制器，就需要重新制作一个含有按钮、旋钮和操纵杆的盾板，获得更为精细的手控。只要将自定义盾板插入Blackmagic 3G-SDI Shield for Arduino的连接器插槽中就可以完成安装。您可以构建任何类型的控制器，甚至将原先的CCU电路替换成自制的Arduino方案，获得一个符合工业标准的摄影机控制单元。
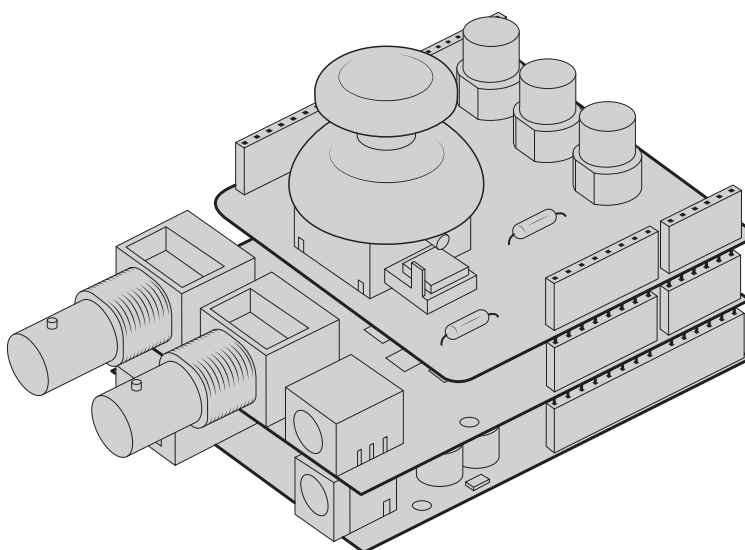
您可以创建自己的硬件控制器，并将其插入Blackmagic 3G-SDI Shield for Arduino上获得更为互动且精细的控制。

Blackmagic Shield for Arduino盾板的通信

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I²C or Serial. We recommend I²C because of the low pin count and it frees up the serial monitor. This also allows you to use more I²C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I²C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial     sdiCameraControl;
BMD_SDITallyControl_Serial      sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol控制协议

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
        8 = Color Correction group
        0 = Lift Adjust
        0 = assign value
        liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header				command				data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin           | Function
--------------------- |----------------------------
  A4                  | Serial Data (SDA)
  A5                  | Serial Clock (SCL)
```

I²C Protocol (Writes):

>   (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

>   (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin           | Function
--------------------- |----------------------------
  IO1                 | Serial Transmit (TX)
  IO0                 | Serial Receive (RX)
```

UART Protocol (Writes):

>   [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

>   [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name        | R/W   | Register Description
-------------------- |----------   |-----  |-----------------------------

0x0000 - 0x0003      | IDENTITY    | R     | Hardware Identifier
0x0004 - 0x0005      | HWVERSION   | R     | Hardware Version
0x0006 - 0x0007      | FWVERSION   | R     | Firmware Version
                     |             |       |
0x1000               | CONTROL     | R/W   | System Control
                     |             |       |
0x2000               | OCARM       | R/W   | SDI Control Override Arm
0x2001               | OCLENGTH    | R/W   | SDI Control Override Length
0x2100 - 0x21FE      | OCDATA      | R/W   | SDI Control Override Data
                     |             |       |
0x3000               | ICARM       | R/W   | SDI Control Incoming Arm
0x3001               | ICLENGTH    | R     | SDI Control Incoming Length
0x3100 - 0x31FE      | ICDATA      | R     | SDI Control Incoming Data
                     |             |       |
0x4000               | OTARM       | R/W   | SDI Tally Override Arm
0x4001               | OTLENGTH    | R/W   | SDI Tally Override Length
0x4100 - 0x41FE      | OTDATA      | R/W   | SDI Tally Override Data
                     |             |       |
0x5000               | ITARM       | R/W   | SDI Tally Incoming Arm
0x5001               | ITLENGTH    | R     | SDI Tally Incoming Length
0x5100 - 0x51FE      | ITDATA      | R     | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the
                          shield generates its own SDI signal on the SDI output connector.
                          When 0, the input signal is passed through to the output if
                          present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately
                          copied over to the override tally data register. Automatically
                          cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data.
                          When 0, input tally data is passed through to the output
                          unmodified.

Override Control: When 1, control data is overridden with the user supplied data.
                          When 0, input control data is passed through to the output
                          unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in
                          the next video frame. Automatically cleared once the control has
                          been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in
                          the next video frame. Automatically cleared once a control
                          packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when
                          a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM. ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

帮助

获得帮助

Blackmagic 3G-SDI Shield for Arduino是一款专为有定制需求的用户进行独立开发所设计的开发人员工具。

请登陆Blackmagic Design在线支持页面获取盾板的最新支持信息。

Blackmagic Design在线支持页面

请登陆Blackmagic Design支持中心www.blackmagicdesign.com/cn/support获得最新版操作手册、软件以及技术答疑文章。

Arduino开发论坛

如果您有编程方面的问题，请登陆Arduino开发论坛获取帮助。Arduino拥有庞大的开发人员社区和众多优质论坛，定能解答您在软件方面遇到的问题，甚至还可以招聘工程师助您实现解决方案!

Blackmagic Design论坛

您可以登陆我们的网站访问Blackmagic Design论坛，获得更多信息和有用的创意资源。访问论坛也是获取帮助的一个捷径，因为论坛中不乏经验丰富的用户和Blackmagic Design的员工，他们都能为您答疑解惑。请登陆网址http://forum.blackmagicdesign.com进入论坛。

查看当前安装的软件版本

要检查您的计算机上安装的Blackmagic 3G-SDI Shield for Arduino Setup软件版本，请打开"About Blackmagic 3G-SDI Shield for Arduino Setup"窗口查看。

- 在Mac OS X系统下，请到"应用程序"文件夹下打开Blackmagic 3G-SDI Shield for Arduino Setup。点击程序菜单中的"About Blackmagic Shield for Arduino Setup"后即可查看版本号。

- 在Windows 7系统下，请到开始菜单打开Blackmagic 3G-SDI Shield for Arduino Setup软件。点击"帮助"菜单并选择"About Blackmagic 3G-SDI Shield for Arduino Setup"后即可查看版本号。

- 在Windows 8系统下，请从开始页面的Blackmagic 3G-SDI Shield for Arduino Setup板块打开Blackmagic 3G-SDI Shield for Arduino Setup。点击"帮助"菜单并选择"About Blackmagic Shield for Arduino Setup"后即可查看版本号。

如何获得软件更新

检查完您计算机上安装的Blackmagic 3G-SDI Shield for Arduino Setup软件版本号之后，请登录网址www.blackmagicdesign.com/cn/support访问Blackmagic Design支持中心查看最新版本。请及时将软件升级到最新版本，但切勿在重要项目制作过程中升级软件。
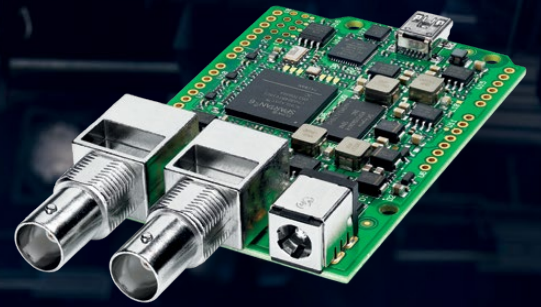
保修

12个月有限保修

Blackmagic Design保证Blackmagic 3G-SDI Shield for Arduino 产品自购买之日起12个月内不会有材料和工艺上的缺陷。若本产品在保修期内出现质量问题，Blackmagic Design可选择为产品提供免费修理或更换零部件，或者更换缺陷产品。

为确保消费者有权享受本保修条款中的服务，如遇产品质量问题请务必在保修期内联系Blackmagic Design并妥善安排保修事宜。消费者应将缺陷产品包装并运送到Blackmagic Design的指定服务中心进行维修，运费由消费者承担并预先支付。若消费者因任何原因退货，所有运费、保险费、关税等各项税务以及其他费用均由消费者承担。

本保修条款不适用于任何因使用、维护不当或保养不周造成的缺陷、故障或损坏。根据本保修服务，Blackmagic Design的保修服务范围不包括以下内容：1. 对由非Blackmagic Design专门人员进行的安装、维修或保养所造成的损坏进行维修，2. 对因使用不当或连接到不兼容设备所造成的损坏进行维修，3. 对因使用了非Blackmagic Design生产的零部件所导致的损坏或故障进行维修，及 4. 对经过改装或和其他产品进行组装的产品进行保养维修（因为产品经改装或组装后会增加保养维修所需时间或保养难度）。本保修条款由BLACKMAGIC DESIGN提供，它可取代所有其他明示或隐含的保修。BLACKMAGIC DESIGN及其供应商对任何有关适销性及就特定用途的适用性等隐含保证不作任何担保。BLACKMAGIC DESIGN负责为消费者提供缺陷产品的维修或更换服务是完整和排他性补救措施，不论BLACKMAGIC DESIGN或其供应商是否事先获悉发生间接、特殊、偶然或必然损坏等损坏的可能性。若消费者对本设备进行非法使用，BLACKMAGIC DESIGN概不负责。对因使用本产品造成的损失，BLACKMAGIC DESIGN概不负责。本产品的操作风险由用户自行承担。

설치 및 사용 설명서

Blackmagic 3G-SDI Shield for Arduino

2020년2월

한국어

환영합니다.

새로운 Blackmagic 3G-SDI Shield for Arduino를 구입해 주셔서 감사합니다.

저희는 늘 새로운 기술에 관심을 갖고 있으며 모든 창의적인 방식으로 자사 SDI 제품을 사용할 수 있게 되어 상당히 기쁩니다. 이제 3G-SDI Shield for Arduino를 사용해 Arduino를 SDI 워크플로에 통합하여 Blackmagic Design 장비를 제어할 수 있는 더욱 다양한 옵션을 얻을 수 있습니다.

예를 들어, SDI 신호에 임베드된 데이터 패킷을 통해 ATEM 스위처에서 Blackmagic URSA Mini 와 Blackmagic Studio Camera를 제어할 수 있습니다. ATEM 스위처를 사용하진 않지만 여전히 Blackmagic 카메라를 제어하고자 할 경우 3G-SDI Shield for Arduino를 사용해 자신만의 커스텀 컨트롤 솔루션을 구축할 수 있습니다. 3G-SDI Shield for Arduino를 SDI 플랫폼으로 사용할 수 있어 쉴드를 통해 스위처의 프로그램 리턴 피드를 Blackmagic 카메라 프로그램 입력으로 루프 출력할 수 있습니다.

카메라에 명령어를 전송하는 코드는 쉽게 작성할 수 있으며 모든 지원 명령어는 본 사용 설명서에서 확인할 수 있습니다.

카메라는 컴퓨터에서 제어할 수 있습니다. 또 다른 방법으로 쉴드에 버튼과 노브, 조이스틱을 추가하여 강력한 하드웨어 컨트롤러를 구축하면 렌즈 초점 및 줌, 조리개 설정, 페디스털 및 화이트 밸런스 제어, 카메라에 내장된 강력한 컬러 커렉터 등의 기능을 조절할 수 있습니다. 자신만의 커스텀 컨트롤러를 구축하면 프로덕션에 도움이 될 뿐만 아니라 아주 재미있게 사용할 수 있습니다.

저희는 이러한 놀라운 기술을 제공할 수 있게 된 것을 아주 기쁘게 생각하며 3G-SDI Shield for Arduino를 위해 사용자들의 SDI 컨트롤러 제작 후기 또한 항상 기다리고 있습니다.

본 설명서에는 Blackmagic 3G-SDI Shield for Arduino 사용에 필요한 모든 정보가 담겨있습니다. 자사 웹사이트 www.blackmagicdesign.com/kr 고객 지원 페이지에서 최신 버전의 사용 설명서와 쉴드의 내부 소프트웨어 업데이트를 확인하실 수 있습니다. 소프트웨어 업데이트를 통해 모든 새로운 기능을 이용하실 수 있습니다. 소프트웨어를 다운로드할 때 사용자 정보를 등록하시면 새로운 소프트웨어가 출시될 때마다 업데이트 소식을 받아보실 수 있습니다. 저희는 새로운 기능 및 제품 향상을 위해 끊임없이 노력하고 있으며, 항상 고객 여러분의 의견을 기다립니다!

Grant Petty

Blackmagic Design의
CEO 그랜트 패티

목차

Blackmagic 3G-SDI Shield for Arduino 설명서

시작하기

헤더 장착 및 납땜하기

Blackmagic 3G-SDI Shield for Arduino는 다른 쉴드를 쌓아 올릴 수 있는 8핀 헤더 두 개, 10핀 헤더 한 개, 6핀 헤더 한 개 등 총 4개의 헤더와 함께 제공됩니다. 헤더는 쉴드를 Arduino 보드에 장착하기 위한 연결 커넥터이며 그 위에 쉴드를 쌓아 올릴 수 있도록 설계되어 제어 버튼과 노브, 조이스틱 등의 추가 구성 요소가 담긴 쉴드를 장착할 수 있습니다. 헤더는 Arduino UNO와 같은 R3 크기의 Arduino 보드에 장착할 수 있도록 설계되었습니다.

헤더를 쉴드에 장착하는 방법은 다음과 같습니다.

1 Blackmagic 3G-SDI Shield 각 면의 핀 홀에 맞는 헤더의 핀을 삽입하세요. 헤더의 레이아웃 배열은 아래 그림을 참고하세요.
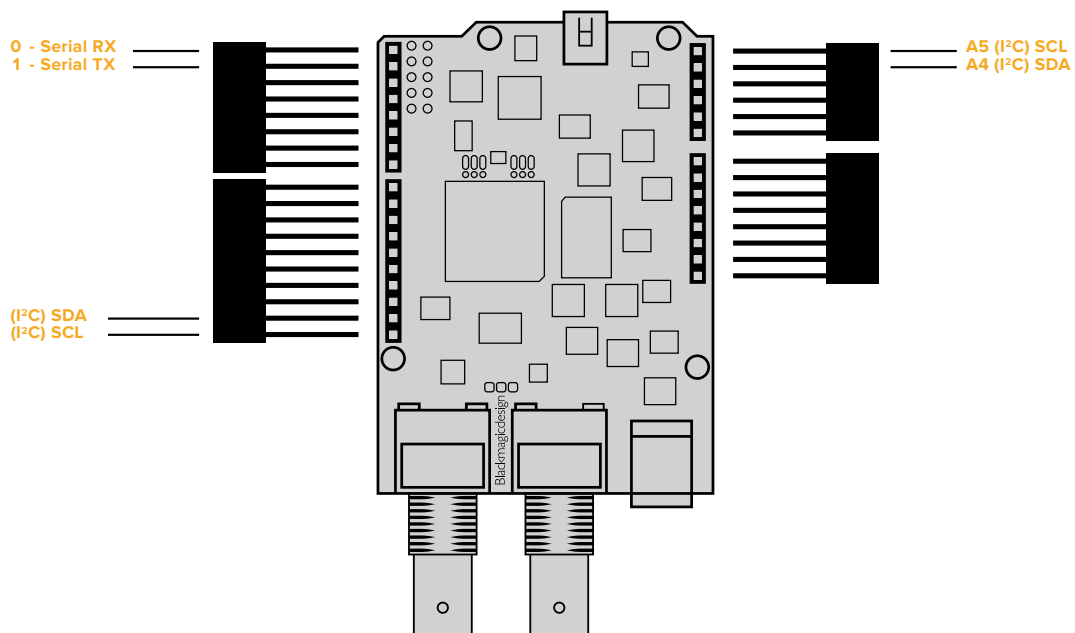
> **참고** 쉴드에 연결할 경우 I²C 또는 Serial을 통해 통신합니다. 시리얼 모니터를 사용할 수 있고 다른 모든 핀들을 활용할 수 있는 I2C를 사용할 것을 권장합니다. 스케치에서 BMDSDIControl 항목을 설정할 경우 통신 모드를 선택하세요. 더 자세한 정보는 [Blackmagic 3G-SDI Shield for Arduino와 통신하기] 부분을 참고하세요.

2 각 헤더 핀의 바닥 면을 쉴드 아래쪽에서 납땜합니다. 핀 홀과 각 핀이 확실히 연결되도록 납땜하되 주변의 다른 핀에 닿지 않도록 주의하세요.

Arduino 보드에 장착하기

헤더를 쉴드에 납땜했으므로 이제 3G-SDI 쉴드를 Arduino 보드에 장착할 수 있습니다.

쉴드의 양옆을 조심스럽게 잡고 헤더 핀을 Arduino 보드의 헤더와 잘 맞춘 뒤, 핀을 헤더 슬롯에 부드럽게 밀어 넣습니다. 쉴드를 장착하는 동안 핀이 구부러지지 않도록 주의하세요.
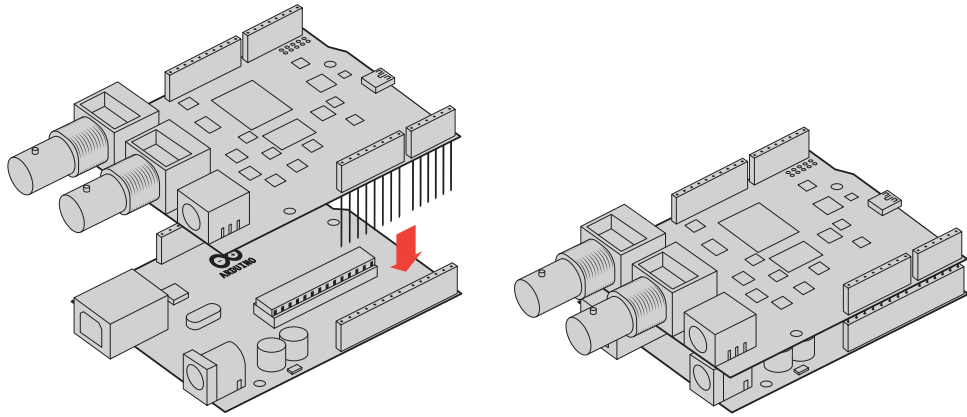
모든 핀을 꼽으면 Blackmagic 쉴드와 Arduino 보드가 안정적으로 연결됩니다.

전원 연결하기

12V 전원 어댑터를 Blackmagic 3G-SDI Shield for Arduino의 12V 전원 입력에 연결하기만 하면 전원이 공급됩니다.

SDI 장비에 연결하기

전원이 연결된 Blackmagic 3G-SDI Shield for Arduino를 SDI 장비에 연결할 수 있습니다. 예를 들어, 다음과 같은 방식으로 스위처와 Blackmagic URSA Mini를 연결할 수 있습니다.

1   스위처의 프로그램 출력을 Blackmagic Arduino 쉴드의 SDI 입력에 연결합니다.

2   Blackmagic 3G-SDI Shield의 SDI 출력을 PGM이라고 표시된 Blackmagic URSA Mini의 프로그램 SDI 입력에 연결하세요.

접속도는 아래와 같습니다.

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini
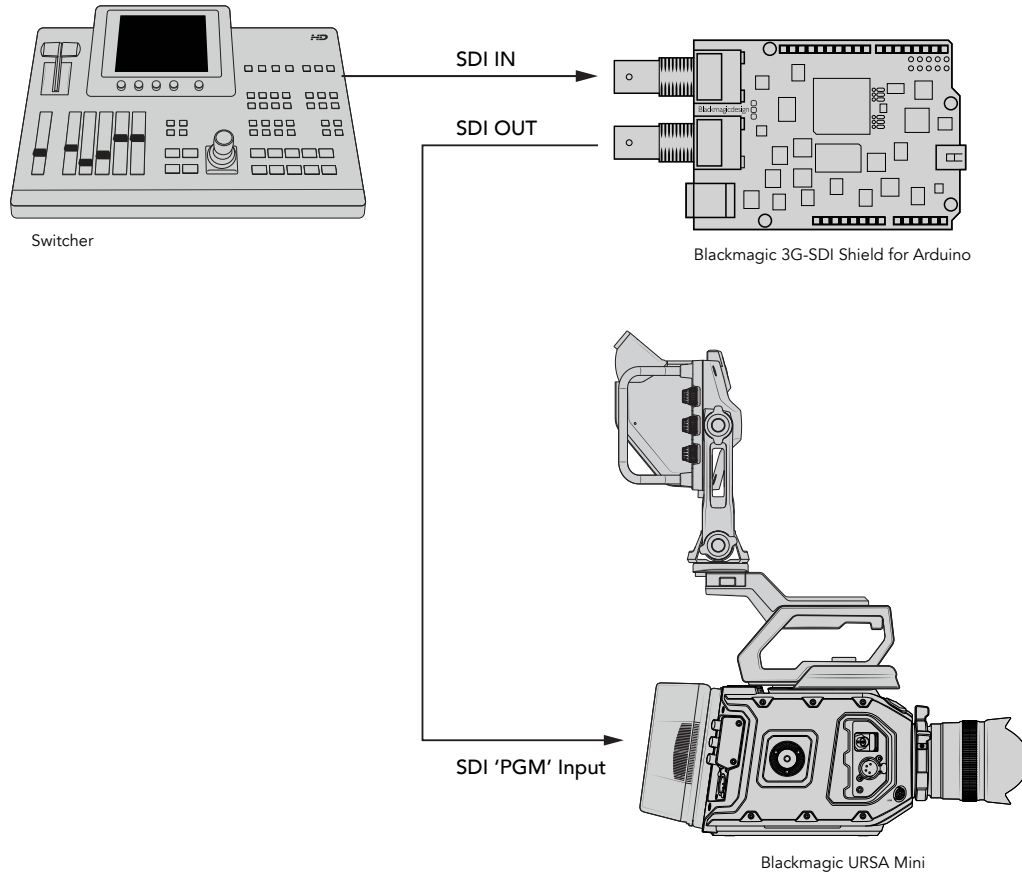
사용 준비가 완료되었습니다.

이제 쉴드를 Arduino 보드에 장착하여 전원을 공급하고 SDI 장비에 연결했으므로 내부 소프트웨어 및 라이브러리 파일을 설치하고 Arduino 소프트웨어를 프로그래밍하여 쉴드로 장비를 제어할 수 있습니다.

본 사용 설명서의 나머지 부분을 모두 읽고 쉴드의 내부 소프트웨어 설치 방법 및 Arduino와의 통신을 위한 Arduino 라이브러리 파일의 저장 위치에 대한 정보를 확인하시기 바랍니다.

> **정보** Blackmagic 3G-SDI Shield for Arduino를 사용하여 Blackmagic MultiView 16과 같은 다른 Blackmagic Design 제품을 제어할 수도 있습니다. 예를 들어, 쉴드를 입력 16에 연결하면 멀티뷰에 탈리 테두리가 나타나도록 할 수 있습니다. Blackmagic URSA Mini 카메라 사용 시 카메라 뷰파인더에 탈리 디스플레이 기능이 활성화될 수 있도록 그리드 오버레이가 켜져 있는지 확인하세요. 더욱 자세한 정보는 Blackmagic 카메라 설명서를 참조하세요.

소프트웨어 설치하기

참고 Blackmagic Shield for Arduino 설정 유틸리티를 설치하기 전에 www.arduino.cc 에서 최신 버전의 Arduino IDE 소프트웨어를 다운로드하여 컴퓨터에 설치해주십시오.

Arduino 소프트웨어를 설치하면 Blackmagic 3G-SDI Shield의 내부 소프트웨어도 설치할 수 있습니다.

내부 소프트웨어 설치하기

Blackmagic Shield for Arduino Setup은 쉴드의 내부 소프트웨어를 업데이트할 때 사용합니다. 내부 소프트웨어는 Arduino 보드와 통신하며 Arduino 라이브러리 파일을 사용해 보드를 제어합니다. 라이브러리 파일은 설치 소프트웨어와 함께 설치되며 이 파일이 담긴 폴더를 복사해 Arduino 애플리케이션 폴더에 붙여넣기만 하면 됩니다. 라이브러리 파일 및 설치 방법에 대한 정보는 본 사용 설명서의 다음 설명 부분에서 확인할 수 있습니다.

최신 Blackmagic 3G-SDI Shield for Arduino 소프트웨어를 다운로드해 쉴드를 업데이트하면 새로운 기능 및 개선된 기능을 사용할 수 있습니다. Blackmagic 고객 지원 센터 (www.blackmagicdesign.com/kr/support)에서 최신 버전을 다운로드할 수 있습니다.

Mac OS X에서 내부 소프트웨어 설치하기

1   Blackmagic Shield for Arduino 소프트웨어를 다운로드해 압축을 풉니다.

2   작업이 완료된 디스크 이미지를 열고 Blackmagic Shield for Arduino 설치 프로그램을 실행합니다. 화면에 나타나는 지시 사항을 따르세요.

3   최신 버전의 Blackmagic Shield for Arduino 설치 프로그램을 설치한 뒤, Blackmagic 쉴드의 전원을 켜고 USB 케이블을 사용해 컴퓨터에 연결하세요.

4   이제 설치 유틸리티를 실행해 화면에 나타나는 지시 사항에 따라 쉴드의 내부 소프트웨어를 업데이트하세요. 내부 소프트웨어가 최신 버전일 경우 어떠한 메시지도 나타나지 않으며 더 이상의 추가 작업이 필요 없습니다.

Windows에서 내부 소프트웨어 설치하기

1   Blackmagic Shield for Arduino 소프트웨어를 다운로드해 압축을 풉니다.

2   본 사용 설명서와 Blackmagic Shield for Arduino 설치 프로그램이 담긴 Blackmagic Shield for Arduino 폴더가 나타납니다. 설치 프로그램을 더블 클릭한 뒤, 화면에 나타나는 지시 사항에 따라 소프트웨어 설치를 진행합니다.

3   최신 버전의 Blackmagic Shield for Arduino 설치 프로그램을 설치한 뒤, Blackmagic 쉴드의 전원을 켜고 USB 케이블을 사용해 컴퓨터에 연결하세요.

4   이제 설치 유틸리티를 실행해 화면에 나타나는 지시 사항에 따라 쉴드의 내부 소프트웨어를 업데이트하세요. 내부 소프트웨어가 최신 버전일 경우, 어떠한 메시지도 나타나지 않으며 더 이상의 추가 작업이 필요 없습니다.

Arduino 라이브러리 파일 설치하기

Arduino를 제어하기 위해 작성된 프로그램을 스케치라고 부르며 Blackmagic 3G-SDI Shield for Arduino에서는 스케치를 쉽게 작성할 수 있는 Arduino 라이브러리 파일을 사용합니다. 쉴드의 설치 소프트웨어를 설치하고 나면 라이브러리 폴더에 라이브러리 파일이 설치됩니다. 이제 라이브러리 파일이 담긴 폴더를 복사해 Arduino 라이브러리 폴더에 붙여넣기만 하면 됩니다.

> **참고** 라이브러리 설치 시에는 Arduino IDE 소프트웨어를 종료하세요.

Mac OS X에서 라이브러리 파일 설치하기
1   응용 프로그램 폴더에서 Blackmagic Shield for Arduino를 엽니다.
2   라이브러리 폴더를 열고 마우스를 우클릭해 BMDSDIControl 폴더를 복사하세요.
3   이제 컴퓨터의 도큐먼트 폴더로 이동해 Arduino 폴더를 엽니다.
4   라이브러리라는 이름의 하위 폴더를 볼 수 있습니다. BMDSDIControl 폴더를 라이브러리 폴더에 붙여넣기 하세요.

Windows에서 라이브러리 파일 설치하기
1   프로그램 폴더를 열고 Blackmagic Shield for Arduino 폴더로 이동하세요.
2   라이브러리라는 하위 폴더를 볼 수 있습니다. 폴더를 열고 마우스를 우클릭해 BMDSDIControl 폴더를 복사하세요.
3   이제 컴퓨터의 문서 폴더로 이동해 Arduino 폴더를 엽니다.
4   라이브러리라는 이름의 하위 폴더를 볼 수 있습니다. BMDSDIControl 폴더를 라이브러리 폴더에 붙여넣기 하세요.

Blackmagic Design 라이브러리 파일을 컴퓨터에 설치하는 데 필요한 모든 준비가 완료되었습니다. 이제 Arduino 소프트웨어를 실행하면 선택 가능한 Blackmagic Design의 스케치 예시가 나타납니다.

Arduino 소프트웨어 메뉴바에서 파일 드롭다운 메뉴로 이동한 뒤, 예제를 선택하세요. 이제 BMDSDIControl을 선택하면 사용 가능한 스케치 예시 목록이 나타납니다.

라이브러리 파일이 올바른 폴더에 저장된 경우 쉴드에서 이 파일을 사용해 Arduino 보드와 통신하게 됩니다. 이제 Arduino IDE 소프트웨어를 프로그래밍하기만 하면 됩니다. 더 자세한 정보는 [Arduino 스케치 프로그래밍] 부분을 참고하세요.

> **참고** 추후 예제가 담긴 라이브러리 파일이 업데이트될 경우, 기존의 BMDSDIControl 폴더를 삭제하고 위에서 설명한 방법대로 새로운 폴더를 설치하세요.

Blackmagic Shield for Arduino Setup
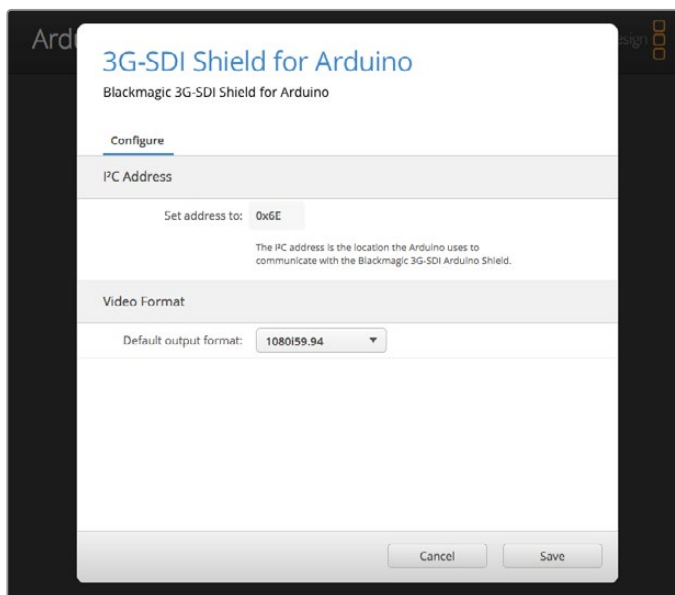
Blackmagic Shield for Arduino Setup 소프트웨어를 통해 I²C
주소와 비디오 출력 포맷 등의 쉴드 설정을 변경할 수 있습니다.

이제 컴퓨터에 설치된 Blackmagic Shield for Arduino Setup을 통해 쉴드의 설정을 변경할 수
있습니다. 예를 들어, 쉴드를 인식해 Arduino 보드와 통신할 수 있도록 돕는 'I²C Address'와
쉴드를 위한 출력 포맷을 설정하는 'Video Format' 등의 쉴드 설정 항목을 변경할 수 있습니다.

I²C 주소

매우 드문 경우이긴 하지만, 쉴드의 기본 설정 주소와 동일한 I²C 주소를 공유하는 또 다른
쉴드가 Blackmagic 쉴드에 장착된 경우에는 충돌이 발생할 가능성이 있습니다. 이런 경우에는
쉴드의 기본 설정 주소를 변경할 수 있습니다.

기본 설정 주소는 0x6E이지만, 0x08과 0x77 사이의 다양한 주소 중 원하는 주소를
선택할 수 있습니다.

쉴드 주소 변경 방법은 다음과 같습니다.

　1　Blackmagic Shield for Arduino Setup을 실행한 뒤, 쉴드의 설정 아이콘을 클릭하세요.

　2　Set address to 설정란에 사용하고자 하는 주소를 입력하세요.

　3　Save를 클릭합니다.

비디오 포맷

아무런 입력이 연결되지 않을 경우에 대비해 설치 유틸리티에 기본 출력 포맷이 선택되어
있습니다. 입력이 감지되면 입력과 같은 포맷으로 출력됩니다. 해당 입력의 연결이 해제되면
유틸리티에서 설정해둔 기본 출력 포맷으로 되돌아갑니다. Default output format 드롭다운
메뉴를 클릭해 원하는 비디오 포맷으로 변경할 수 있습니다.

다음과 같은 비디오 출력 포맷을 선택할 수 있습니다.

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Arduino 스케치 프로그래밍

Arduino 소프트웨어에 작성된 프로그램 또는 스케치는 작성 방법이 매우 간단합니다! Sketches는
일반 C 프로그래밍 언어를 사용해 작성합니다. Studio Camera Control Protocol의 명령어를
사용해 스케치를 프로그래밍하는 경우 쉴드에서 명령어를 SDI 출력에 임베드하는 방식을 통해
Blackmagic URSA Mini 또는 Blackmagic Studio Camera를 제어하게 됩니다.

모든 지원 명령어는 본 사용 설명서의 [Studio Camera Control Protocol] 부분에 나와 있으므로
원하는 프로토콜 명령어를 스케치에 사용할 수 있습니다.

Blackmagic Shield 및 라이브러리 설치 테스트하기

[시작하기] 부분에 소개된 대로 모든 연결을 마치고 설치 소프트웨어 및 라이브러리 파일을 설치하고 나면 쉴드가 Arduino 보드와 통신하며 모든 기능이 제대로 작동하는지 확인해야 합니다.

탈리 블링크 스케치 예제를 적용해 이런 사항을 신속하게 확인할 수 있습니다.

다음의 단계를 따르세요.

1 Arduino IDE 소프트웨어를 실행하세요.

2 '툴' 메뉴에서 Arduino 보드 및 포트 번호를 선택합니다.

3 '파일' 메뉴에서 예제와 BMDSDIControl을 선택한 뒤, TallyBlink라고 적힌 스케치를 선택합니다.

4 해당 스케치를 보드에 업로드하세요.

```
/*
 Blink
 Turns on an LED on for one second, then off for one second, repeatedly.

 Addition to original blink sketch also turns on and off camera 1's tally indicator.

 Most Arduinos have an on-board LED you can control. On the Uno and
 Leonardo, it is attached to digital pin 13. If you're unsure what
 pin the on-board LED is connected to on your Arduino model, check
 the documentation at http://www.arduino.cc

 This example code is in the public domain.
*/

#include <BMDSDIControl.h>                    // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);    // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                    // initialize tally control
  sdiTallyControl.setOverride(true);          // enable tally override

  pinMode(13, OUTPUT);                        // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                     // turn the LED ON

  sdiTallyControl.setCameraTally(             // turn tally ON
    1,                                        // Camera Number
    true,                                     // Program Tally
    false                                     // Preview Tally
  );

  delay(1000);                                // leave it ON for 1 second

  digitalWrite(13, LOW);                      // turn the LED OFF

  sdiTallyControl.setCameraTally(             // turn tally OFF
    1,                                        // Camera Number
    false,                                    // Program Tally
    false                                     // Preview Tally
  );

  delay(1000);                                // leave it OFF for 1 second
}
```

No changes necessary for Auto Format.

Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.

Tally Blink 예제 스케치를 통해 Arduino 쉴드를 쉽고 빠르게 테스트할 수 있습니다. Studio Camera Protocol 문서의 명령어를 사용하면 I²C를 통해 Raw 데이터를 쉴드에 전송할 수 있지만, 스케치를 더욱 쉽게 프로그래밍할 수 있는 커스텀 라이브러리 또한 함께 제공됩니다.

Blackmagic Studio Camera의 탈리 라이트가 1초에 한 번씩 깜빡이는 것을 확인할 수 있습니다. 탈리 라이트가 깜빡이면 Blackmagic 쉴드가 Arduino와 통신하며 모든 기능이 정상적으로 작동하는 것을 의미합니다.

탈리 라이트가 깜빡이지 않을 경우 Blackmagic 카메라의 탈리 번호가 1로 설정되어있는지 그리고 카메라의 그리드 오버레이가 켜져있는지 확인하세요.

지원이 필요한 경우에는 Blackmagic Design 지원 센터(www.blackmagicdesign.com/kr/support)를 방문하세요. 쉴드 설정과 관련해 도움을 얻을 수 있는 다른 방법에 대한 자세한 정보는 본 사용 설명서의 [지원] 부분을 참고하세요.

LED 표시 장치

Blackmagic 3G-SDI Shield for Arduino에는 전원과 UART, I²C, SPI 통신 등의 작동 상태를 알려주는 표시 장치와 탈리 및 카메라 오버라이드 제어 기능의 활성화 여부를 보여주는 표시 장치 등 총 6개의 LED 표시 장치가 탑재되어 있습니다.
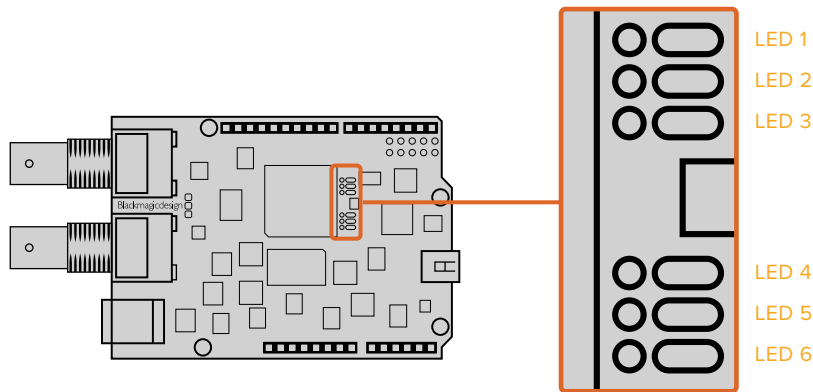
LED 1 - 시스템 활성화
쉴드에 전원이 연결되면 불이 들어옵니다.

LED 2 - 오버라이드 제어 활성화
Arduino 스케치에서 카메라 제어를 활성화한 경우에 불이 들어옵니다.

LED 3 - 탈리 오버라이드 활성화
Arduino 스케치에서 탈리를 활성화한 경우에 불이 들어옵니다.

LED 5 - I²C 파서 사용 중
쉴드와 Arduino가 I²C 프로토콜을 사용해 통신하는 것이 감지된 경우에 불이 들어옵니다.

LED 6 - 시리얼 파서 사용 중
UART 통신이 감지된 경우에 불이 들어옵니다.

Blackmagic 쉴드 부팅 중에는 전원 표시 장치가 꺼져 있으며 LED 3, 4, 5는 다음과 같은 상태를 나타냅니다.

LED 3 - 애플리케이션 이미지 로딩

LED 4 - EEPROM 초기화

LED 5 - 메모리 검사 진행 중

부팅이 성공적으로 완성되면 전원 LED가 켜지며 작동과 함께 모든 LED가 표준 기능으로 되돌아 갑니다.

드문 경우이긴 하지만 부팅에 실패할 경우, 오류가 발생한 LED를 제외한 모든 LED가 빠르게 깜빡거려 오류의 원인을 쉽게 파악할 수 있습니다.

쉴드 부품 장착하기

자신만의 하드웨어 컨트롤러를 만들고자 할 경우, 새로운 쉴드에 버튼과 노브, 조이스틱을 장착하여 수동으로 직접 제어할 수 있습니다. 커스텀 쉴드를 Blackmagic 3G-SDI Shield for Arduino의 헤더 슬롯에 간단히 장착하세요. 원하는 모든 종류의 컨트롤러를 구축할 수 있습니다. 오래된 CCU의 회로를 자신만의 커스텀 Arduino 솔루션으로 교체하여 산업 표준 카메라 제어 장치로 사용할 수 있습니다.
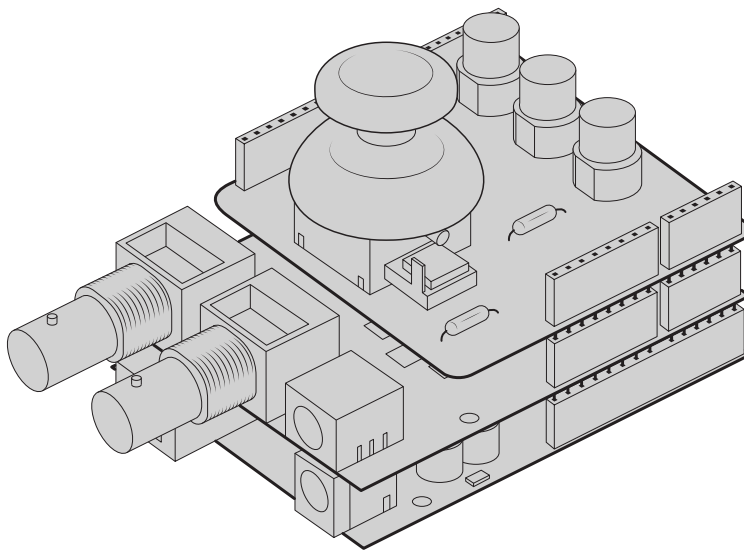
자신만의 하드웨어 컨트롤러를 구축하여 Blackmagic 3G-SDI Shield for Arduino에 장착하면 더 나은 상호 연동성을 가진 정확한 제어를 수행할 수 있습니다.

Communicating with your Blackmagic Shield for Arduino

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I^{2}C or Serial. We recommend I^{2}C because of the low pin count and it frees up the serial monitor. This also allows you to use more I^{2}C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I^{2}C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                shieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial    sdiCameraControl;
BMD_SDITallyControl_Serial     sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                shieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

        1 = camera number 1
        8 = Color Correction group
        0 = Lift Adjust
        0 = assign value
        liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDddddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, ddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDddddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, dddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header		command						data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
-------------------- |----------------------------
  A4                 | Serial Data (SDA)
  A5                 | Serial Clock (SCL)
```

I²C Protocol (Writes):

        (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

        (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
-------------------- |----------------------------
  IO1                | Serial Transmit (TX)
  IO0                | Serial Receive (RX)
```

UART Protocol (Writes):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name        | R/W    | Register Description
-------------------- |----------   |-----   |-----------------------------

0x0000 - 0x0003      | IDENTITY    | R      | Hardware Identifier
0x0004 - 0x0005      | HWVERSION   | R      | Hardware Version
0x0006 - 0x0007      | FWVERSION   | R      | Firmware Version
                     |             |        |
0x1000               | CONTROL     | R/W    | System Control
                     |             |        |
0x2000               | OCARM       | R/W    | SDI Control Override Arm
0x2001               | OCLENGTH    | R/W    | SDI Control Override Length
0x2100 - 0x21FE      | OCDATA      | R/W    | SDI Control Override Data
                     |             |        |
0x3000               | ICARM       | R/W    | SDI Control Incoming Arm
0x3001               | ICLENGTH    | R      | SDI Control Incoming Length
0x3100 - 0x31FE      | ICDATA      | R      | SDI Control Incoming Data
                     |             |        |
0x4000               | OTARM       | R/W    | SDI Tally Override Arm
0x4001               | OTLENGTH    | R/W    | SDI Tally Override Length
0x4100 - 0x41FE      | OTDATA      | R/W    | SDI Tally Override Data
                     |             |        |
0x5000               | ITARM       | R/W    | SDI Tally Incoming Arm
0x5001               | ITLENGTH    | R      | SDI Tally Incoming Length
0x5100 - 0x51FE      | ITDATA      | R      | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the
                        shield generates its own SDI signal on the SDI output
                        connector. When 0, the input signal is passed through to the
                        output if present, or the shield generates its own SDI
                        signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied
                        over to the override tally data register. Automatically cleared
                        by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data.
                        When 0, input tally data is passed through to the output
                        unmodified.

Override Control: When 1, control data is overridden with the user supplied
                        data. When 0, input control data is passed through to the
                        output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the
                next video frame. Automatically cleared once the control has
                been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the
                next video frame. Automatically cleared once a control packet has
                been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a
                new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_
was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from
the next video frame until new data is set. Automatically cleared once
the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one
byte per camera). Bit zero indicates a Program tally, while bit one
indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame.
Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a
new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was
reset (one byte per camera). Bit zero indicates a Program tally, while
bit one indicates a Preview tally.

지원

지원 받기

Blackmagic 3G-SDI Shield for Arduino는 사용자가 원하는 사항을 독자적으로 개발할 수 있는 개발자 도구입니다.

쉴드에 관한 최신 지원 정보는 Blackmagic Design 온라인 고객 지원 페이지를 방문해 확인하실 수 있습니다.

Blackmagic Design 온라인 고객 지원 페이지

최신 사용 설명서와 소프트웨어, 지원 노트는 Blackmagic 고객 지원 센터 (www.blackmagicdesign.com/kr/support)에서 확인하실 수 있습니다.

Arduino 개발 포럼

프로그래밍에 관한 질문은 인터넷상의 Arduino 개발 포럼에 문의하시기 바랍니다. 전체 Arduino 개발자 커뮤니티와 수준 높은 여러 포럼을 통해 소프트웨어에 대한 궁금증을 문의할 수 있으며 원하는 솔루션을 설치해 줄 엔지니어를 고용할 수도 있습니다.

Blackmagic Design 포럼

저희 웹사이트에 있는 Blackmagic Design 포럼은 유용한 정보를 제공하는 곳으로 방문을 통해 자세한 정보와 창의적인 아이디어를 얻을 수 있습니다. 또한, 숙련된 사용자들이나 Blackmagic Design 직원들이 기존에 올려놓은 해결책을 통해 원하는 해답을 얻을 수도 있으므로 여러 가지 도움을 빠르게 받아 한 단계 성장할 수 있는 방법이기도 합니다. 포럼은 http://forum. blackmagicdesign.com를 통해 방문할 수 있습니다.

현재 설치된 소프트웨어 버전 확인하기

컴퓨터에 설치된 Blackmagic 3G-SDI Shield for Arduino Setup 소프트웨어의 버전을 확인하려면 About Blackmagic Teranex Setup 창을 열어주세요.

- Mac OS X에서는 응용 프로그램 폴더에 있는 Blackmagic 3G-SDI Shield for Arduino Setup을 엽니다. 응용 프로그램 메뉴에서 Blackmagic Shield for Arduino Setup을 선택하고 버전을 확인하세요.
- Windows 7에서는 시작 메뉴에서 Blackmagic 3G-SDI Shield for Arduino Setup을 실행합니다. 도움말 메뉴를 클릭한 뒤 About Blackmagic Shield for Arduino Setup을 선택하면 버전을 확인할 수 있습니다.
- Windows 8에서는 시작 페이지에 있는 Blackmagic 3G-SDI Shield for Arduino Setup 타일에서 Blackmagic 3G-SDI Shield for Arduino Setup을 실행합니다. 도움말 메뉴를 클릭한 뒤 About Blackmagic Shield for Arduino Setup을 선택하면 버전을 확인할 수 있습니다.

최신 버전 소프트웨어 업데이트하기

컴퓨터에 설치된 Blackmagic 3G-SDI Shield for Arduino Setup 소프트웨어 버전을 확인한 뒤, Blackmagic Design 고객 지원 센터(www.blackmagicdesign.com/kr/support)에 방문하여 최신 업데이트를 확인하세요. 최신 버전으로 업데이트하는 것을 권장하지만, 중요한 프로젝트를 실행하는 도중에는 소프트웨어 업데이트를 하지 않는 것이 좋습니다.

보증

12개월 한정 보증

Blackmagic Design은 Blackmagic 3G-SDI Shield for Arduino 제품의 부품 및 제조에 어떠한 결함도 없음을 제품 구매일로부터 12개월 동안 보증합니다. 보증 기간 내에 결함이 발견될 경우, Blackmagic Design은 당사의 결정에 따라 무상 수리 또는 새로운 제품으로 교환해드립니다.

구매 고객은 반드시 보증 기간이 만료되기 전에 결함 사실을 Blackmagic Design에 통지해야 적절한 보증 서비스를 제공받을 수 있습니다. 구매 고객은 지정된 Blackmagic Design 서비스 센터로 결함 제품을 포장 및 운송할 책임이 있으며, 운송 비용은 선불로 지급되어야 합니다. 구매 고객은 또한 이유를 불문하고 제품 반송에 대한 운송료와 보험, 관세, 세금, 기타 비용을 부담해야 합니다.

이 보증은 부적절한 사용 및 관리, 취급으로 인한 파손, 고장, 결함에는 적용되지 않습니다. Blackmagic Design은 다음과 같은 경우에 보증 서비스를 제공할 의무가 없습니다. a) Blackmagic Design 판매 대리인이 아닌 개인에 의해 발생한 제품 손상. b) 부적절한 사용 및 호환하지 않는 장비와의 연결로 인한 제품 손상. c) Blackmagic Design사의 부품 및 공급품이 아닌 것을 사용하여 발생한 손상 및 고장. d) 제품을 개조하거나 다른 제품과 통합하여 제품 작동 시간 증가 및 기능 저하가 발생한 경우. BLACKMAGIC DESIGN에서 제공하는 제품 보증은 다른 모든 명시적 또는 묵시적 보증을 대신합니다. BLACKMAGIC DESIGN사와 관련 판매 회사는 상품성 및 특정 목적의 적합성과 관련된 모든 묵시적 보증을 부인합니다. 구매 고객에게 제공되는 BLACKMAGIC DESIGN의 결함 제품 수리 및 교환 관련 책임은 BLACKMAGIC DESIGN 또는 판매 회사에서 관련 위험의 가능성에 대한 사전 통보의 여부와 관계없이 모든 간접적, 특별, 우발적, 결과적 손해에 대한 유일한 배상 수단입니다. BLACKMAGIC DESIGN은 고객이 사용한 불법 장비에 대해서는 어떤 법적 책임도 지지 않습니다. BLACKMAGIC은 본 제품의 사용으로 인해 발생하는 손해에 대해서는 어떤 법적 책임도 지지 않습니다. 제품 사용으로 인해 발생할 수 있는 위험에 대한 책임은 본인에게 있습니다.

Руководство по установке и эксплуатации

Blackmagic 3G-SDI Shield for Arduino

Февраль 2020 г.

Русский

Добро пожаловать!

Благодарим вас за покупку модуля Blackmagic 3G-SDI Shield for Arduino.

Мы всегда стремимся внедрять последние технологии, чтобы предложить дополнительные варианты использования нашей техники на основе SDI. Новый модуль 3G-SDI Shield for Arduino поможет интегрировать платы Arduino в рабочий процесс для увеличения возможностей управления оборудованием Blackmagic Design через SDI-сигнал.

Например, встроенные в SDI-сигнал пакеты с данными позволяют управлять камерами Blackmagic URSA Mini и Blackmagic Studio с видеомикшера ATEM. Если вы не используете ATEM, то для этих же целей можно создать собственное решение на базе модуля 3G-SDI Shield for Arduino. Благодаря SDI-интерфейсу модуль обеспечивает передачу обратного сигнала с видеомикшера на программный вход камер Blackmagic.

В этом случае можно создавать алгоритмы для управления камерой, а список поддерживаемых команд приведен в данном руководстве.

Модуль позволяет дистанционно управлять камерами с компьютера, добавить к плате кнопки, ручки прокрутки или джойстики, а также установить аппаратные средства для настройки таких функций, как фокусировка и зум, изменение экспозиции, встроенный инструмент цветокоррекции, баланс белого и черного. Разработка собственных решений будет интересной как профессионалам, так и любителям.

Будем рады узнать о тех решениях, которые созданы вами на основе модуля 3G-SDI Shield for Arduino.

Это руководство содержит всю информацию, необходимую для работы с Blackmagic 3G-SDI Shield for Arduino. Последнюю версию руководства и программного обеспечения для модуля можно найти в разделе поддержки на веб-сайте www.blackmagicdesign.com/ru. Использование актуальной версии ПО гарантирует доступ ко всем имеющимся функциям. Чтобы узнавать о выходе обновлений, зарегистрируйтесь при загрузке программного обеспечения. Мы продолжаем работать над совершенствованием наших продуктов, поэтому ваши отзывы помогут нам сделать их еще лучше!

Грант Петти
Генеральный директор Blackmagic Design

Содержание

# Blackmagic 3G-SDI Shield for Arduino

Подготовка к работе

Монтаж и распайка разъемов

В комплект поставки Blackmagic 3G-SDI Shield for Arduino входят четыре межплатные стойки — 6-контактная, 10-контактная и две 8-контактных. Они представляют собой соединительные разъемы, с помощью которых модуль крепится на плату Arduino. Благодаря их наращиваемой конструкции можно также добавить плату расширения, дополнительно оснащенную джойстиком, кнопками и ручками. Схема разъемов поддерживает установку модуля на такие платы, как Arduino UNO R3.

Порядок монтажа разъемов

1 Вставьте контакты каждого из разъемов в соответствующие отверстия на обеих сторонах модуля Blackmagic 3G-SDI Shield for Arduino. Схема подключения изображена на рисунке ниже.
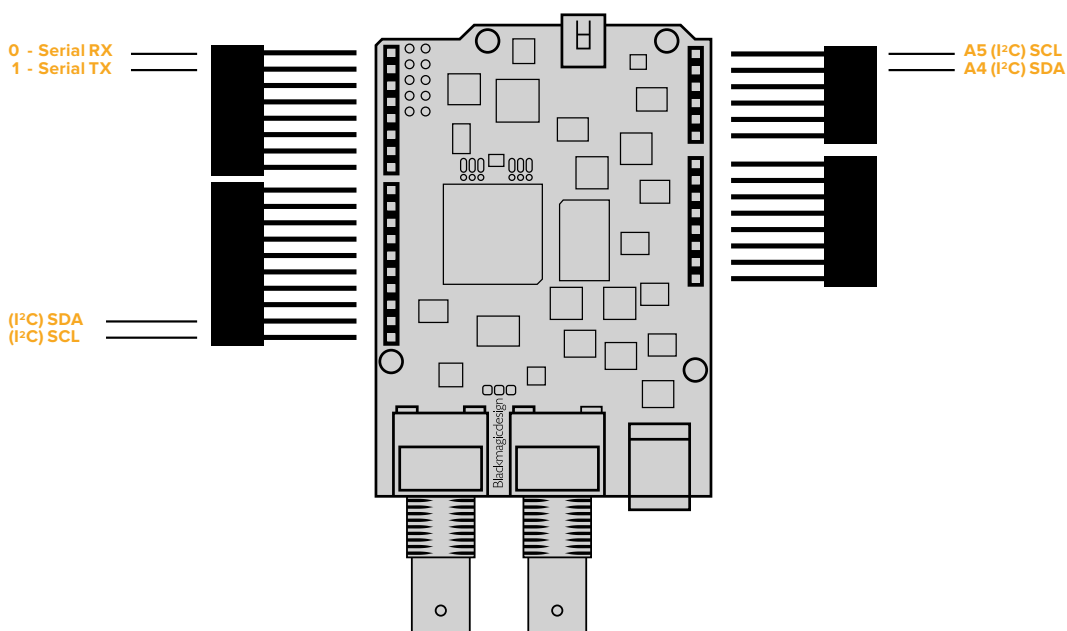
ПРИМЕЧАНИЕ. Связь с подключенным модулем устанавливается по I^2C- или последовательному протоколу. Рекомендуется выбирать протокол I^2C, чтобы использовать монитор с последовательным интерфейсом и обеспечить доступ ко всем разъемам. При назначении в скетче объекта BMDSDIControl выберите режим связи. Подробнее см. раздел «Обмен данными с модулем Blackmagic 3G-SDI Shield for Arduino».

2 Припаяйте каждый из контактов разъема к соответствующему отверстию на нижней панели модуля, не затрагивая при этом соседние.

Установка модуля на плату Arduino

После распайки разъемов модуль Blackmagic 3G-SDI Shield for Arduino можно устанавливать на плату Arduino.

Удерживая модуль за края, совместите его контакты с разъемами платы Arduino и осторожно установите их в слоты. Чтобы не погнуть контакты, будьте внимательны при установке модуля.
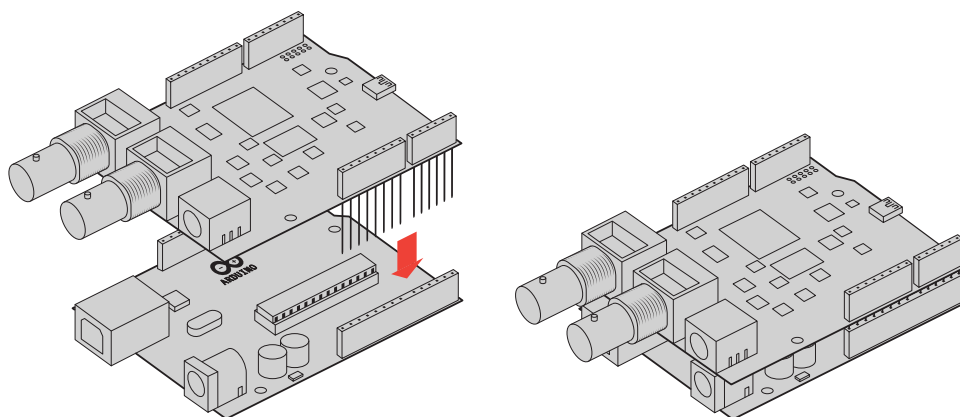
Модуль Blackmagic 3G-SDI Shield for Arduino прикреплен к плате Arduino, когда все его контакты надежно соединены

Подключение питания

Для подачи питания соедините адаптер переменного тока 12 В со входом питания 12 В на Blackmagic 3G-SDI Shield for Arduino.
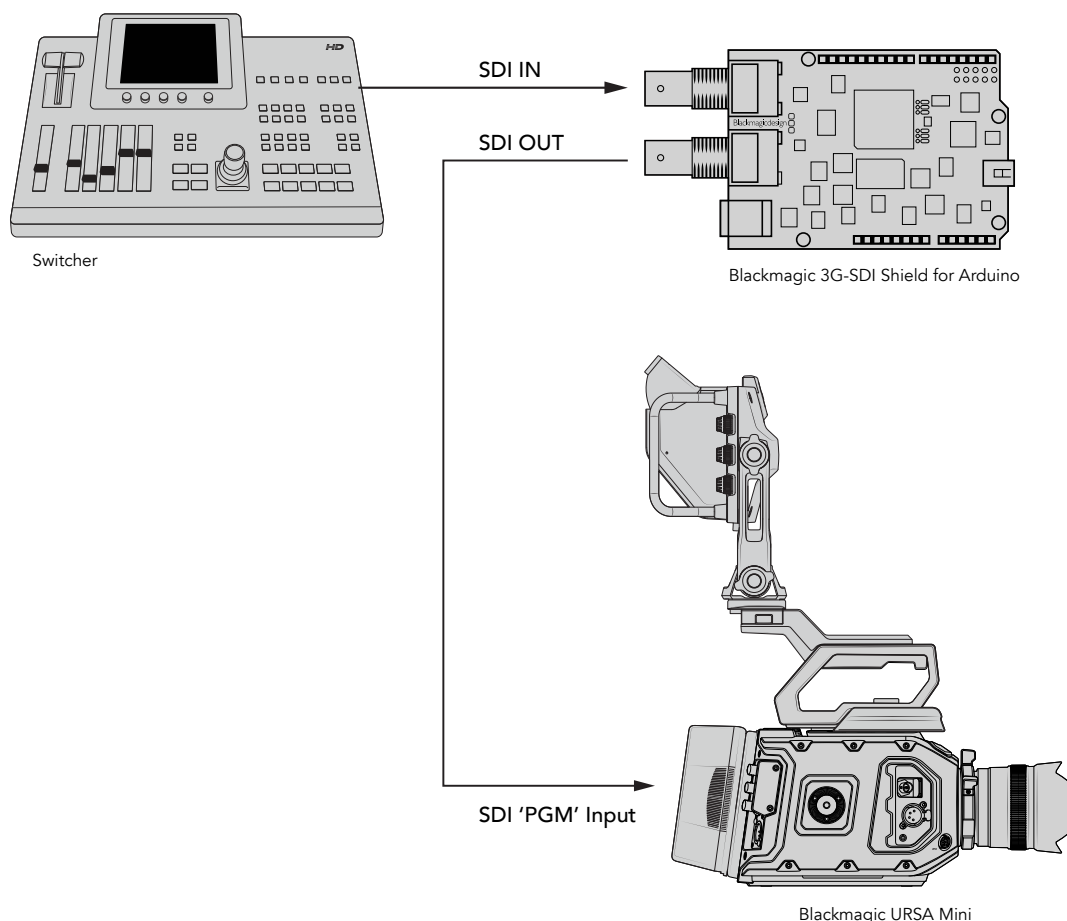
Подключение к **SDI-оборудованию**

Теперь модуль Blackmagic 3G-SDI Shield можно подключить к SDI-оборудованию. Ниже описан порядок подключения к видеомикшеру и камере Blackmagic URSA Mini.

1   Соедините программный выход видеомикшера с SDI-входом модуля.

2   Соедините SDI-выход модуля Blackmagic 3G-SDI Shield с SDI-входом (PGM) камеры Blackmagic URSA Mini.

Схематически это выглядит так

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini

Все готово для начала работы!

После монтажа модуля на плату расширения Arduino, подключения питания и соединения с SDI-оборудованием, можно установить внутреннее ПО, загрузить файлы библиотеки и выполнить программные установки Arduino.

В следующих разделах руководства содержится информация о порядке установки внутреннего ПО модуля и библиотечных файлов платы Arduino.

> **СОВЕТ.** Модуль Blackmagic 3G-SDI Shield for Arduino можно также использовать для управления другим оборудованием производства Blackmagic Design, в том числе решением многооконного мониторинга Blackmagic MultiView 16. При подключении модуля ко входу 16 на мониторе будет отображаться красная рамка состояния. Для показа рамки на видоискателе камеры Blackmagic URSA Mini убедитесь, что включена функция Grid overlay. Подробнее см. руководство по работе с камерами Blackmagic.

Установка программного обеспечения

ПРИМЕЧАНИЕ. Сначала необходимо загрузить и установить последнюю версию Arduino IDE, доступную на сайте www.arduino.cc.

После этого можно установить внутреннее ПО модуля Blackmagic 3G-SDI Shield.

Установка внутреннего ПО

Для обновления прошивки модуля используется утилита Blackmagic Shield for Arduino Setup. Внутреннее ПО предназначено для управления платой расширения с помощью библиотеки Arduino. Библиотечные файлы загружаются при установке утилиты Blackmagic Shield for Arduino Setup. Для этого папку с файлами необходимо скопировать в папку приложения Arduino. Подробнее о библиотечных файлах и их загрузке см. в следующем разделе.

Для оптимальной производительности модуля рекомендуется использовать последнюю версию ПО для Blackmagic 3G-SDI Shield for Arduino, которая доступна в центре поддержки Blackmagic Design на странице www.blackmagicdesign.com/ru/support.

Установка внутреннего ПО на платформе Mac OS X

1 Загрузите и распакуйте ПО Blackmagic Shield for Arduino.

2 Выберите соответствующий диск для просмотра его содержимого и запустите установщик Blackmagic Shield for Arduino. Следуйте инструкциям на экране.

3 Когда у вас есть последняя версия установщика Blackmagic Shield for Arduino, подключите модуль к источнику питания, а затем к компьютеру с помощью USB-кабеля.

4 Для обновления прошивки модуля запустите утилиту Blackmagic Shield for Arduino Setup и следуйте инструкциям на экране. Если инструкции не появятся, используемая версия является актуальной.

Установка внутреннего ПО на Windows

1 Загрузите и распакуйте ПО Blackmagic Shield for Arduino.

2 Найдите папку Blackmagic Shield for Arduino, в которой содержатся данное руководство и установщик Blackmagic Shield for Arduino. Щелкните кнопкой мыши дважды по значку установщика и следуйте инструкциям на экране.

3 Когда у вас есть последняя версия установщика Blackmagic Shield for Arduino, подключите модуль к источнику питания, а затем к компьютеру с помощью USB-кабеля.

4 Для обновления прошивки модуля запустите утилиту Blackmagic Shield for Arduino Setup и следуйте инструкциям на экране. Если они не появятся, используемая версия является актуальной.

Загрузка библиотеки для Arduino

Программы, написанные для управления платой Arduino, называются скетчами. Модуль Blackmagic 3G-SDI Shield for Arduino использует библиотечные файлы Arduino, которые упрощают создание скетчей. После установки утилиты Blackmagic Shield for Arduino Setup библиотечные файлы будут сохранены в папке Library, откуда их необходимо скопировать в папку Arduino > libraries.

> **ПРИМЕЧАНИЕ.** Перед загрузкой библиотечных файлов необходимо закрыть программу Arduino IDE.

Порядок установки библиотечных файлов на Mac OS X

1 Откройте Blackmagic Shield for Arduino в папке «Приложения» (Applications).

2 Выберите папку Library и щелкните правой кнопкой мыши, чтобы скопировать папку BMDSDIControl.

3 Перейдите в папку «Документы» (Documents) на вашем компьютере и откройте папку Arduino.

4 В папке Arduino найдите вложенную папку libraries и вставьте в нее BMDSDIControl.

Порядок установки библиотечных файлов на Windows

1 Откройте Blackmagic Shield for Arduino в папке «Программы» (Programs).

2 Выберите папку Library и нажмите правой кнопкой мыши, чтобы скопировать папку BMDSDIControl.

3 Перейдите в папку «Документы» (Documents) на вашем компьютере и откройте папку Arduino.

4 В папке Arduino найдите вложенную папку libraries и вставьте в нее BMDSDIControl.

Это все, что требуется для загрузки библиотечных файлов на ваш компьютер. При работе с ПО Arduino можно также использовать примеры скетчей, созданных Blackmagic Design.

Для этого из раскрывающегося меню приложения Arduino выберите examples. Для вывода списка готовых скетчей нажмите BMDSDIControl.

После сохранения библиотечных файлов в соответствующих папках, их можно использовать для работы с модулем Arduino. Теперь остается написать программу в среде Arduino. Подробнее об этом см. раздел «Написание программ для Arduino».

> **ПРИМЕЧАНИЕ.** При обновлении библиотечных файлов необходимо удалить старую папку BMDSDIControl и загрузить новую, следуя инструкциям выше.

Утилита **Blackmagic Shield for Arduino Setup**
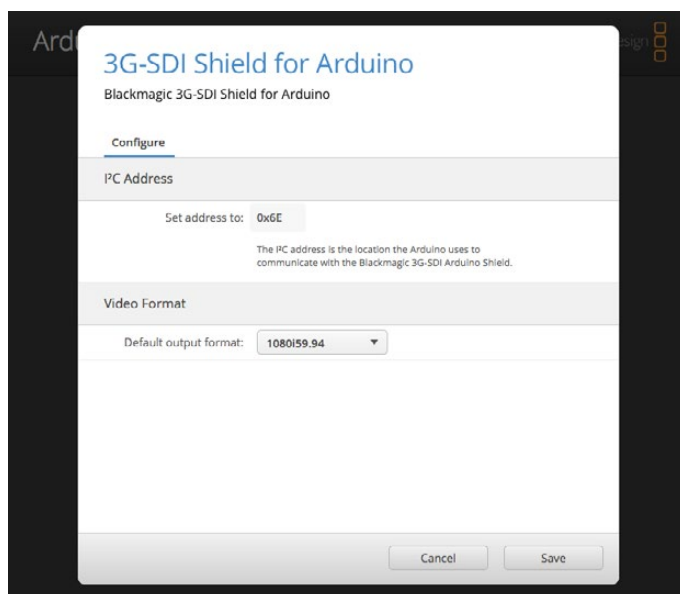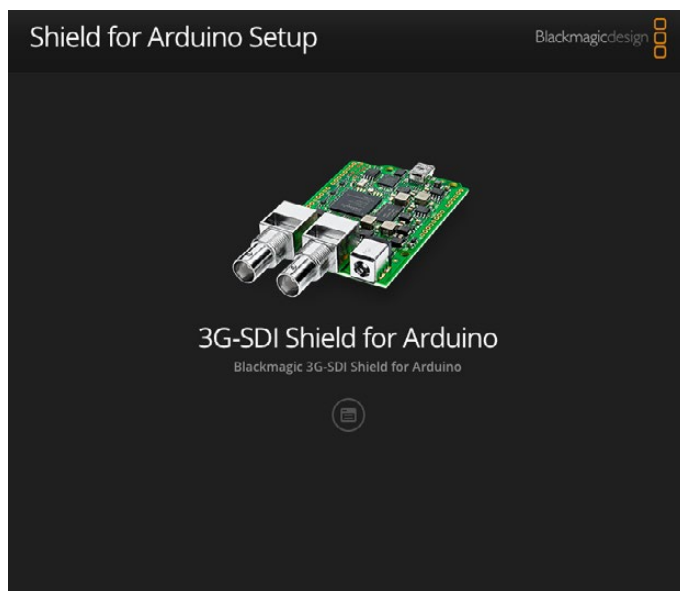
Утилита Blackmagic Shield for Arduino Setup позволяет менять настройки на модуле, в том числе I²C-адрес и формат видео на выходе

Утилита Blackmagic Shield for Arduino Setup позволяет менять настройки, в том числе "I²C address", предназначенные для идентификации модуля и его соединения с платой Arduino.

I²C-адрес

Иногда возможна установка дополнительного модуля на Blackmagic Arduino Shield. Однако, если у обоих модулей одинаковый I²C-адрес, может возникнуть сбой. В этом случае следует изменить заданный по умолчанию адрес модуля.

Заданный по умолчанию адрес модуля — 0x6E, но можно выбрать любой в диапазоне от 0x08 до 0x77.

Порядок изменения адреса модуля

   1   Запустите Blackmagic Shield for Arduino Setup и нажмите на значок настроек модуля.

   2   В строке "Set address to" введите нужный адрес.

   3   Нажмите Save.

Формат видеосигнала

Если источник сигнала не подключен, видеоформат на выходе задается по умолчанию. При обнаружении входного сигнала на выходе задается такой же формат, как на входе. При отключении входного сигнала формат возвращается к настройке по умолчанию. Для того, чтобы изменить видеоформат по умолчанию, из раскрывающегося меню "Default output format" выберите необходимую настройку.

Список доступных видеоформатов

- 720p/50
- 720p/59,94
- 720p/60
- 1080i/50
- 1080i/59,94
- 1080i/60
- 1080p/23,98
- 1080p/24
- 1080p/25
- 1080p/29,97
- 1080p/30
- 1080p/50
- 1080p/59,94
- 1080p/60

Создание скетчей Arduino

Для написания программ (скетчей) используется язык Си. Обычно это не вызывает больших сложностей. Если при написании скетчей используются команды протокола Studio Camera Control Protocol, они включаются в выходной SDI-сигнал, что позволяет управлять камерами Blackmagic URSA Mini или Blackmagic Studio Camera.

Список поддерживаемых команд см. в разделе "Studio Camera Control Protocol".
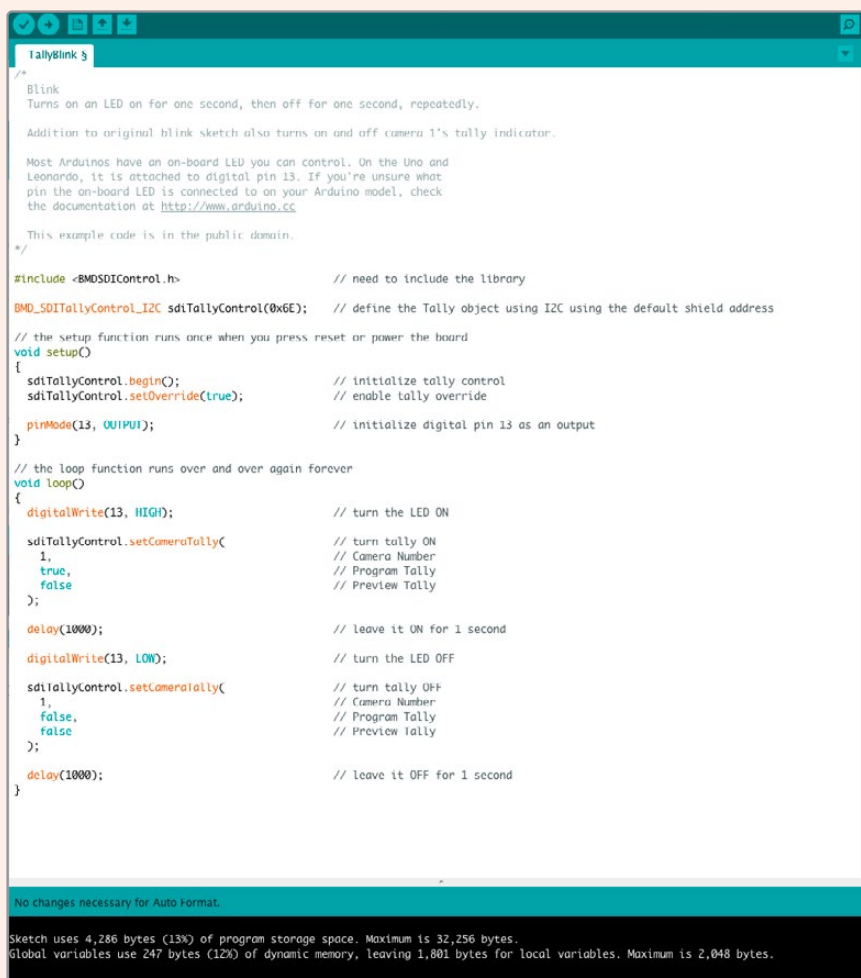
Проверка Blackmagic Arduino Shield и загрузка библиотеки

После того, как будут выполнены все инструкции раздела «Подготовка к работе», установлено необходимое ПО и загружены библиотечные файлы, следует проверить наличие связи между модулем и платой Arduino.

Самый быстрый способ — это открыть и запустить имеющийся скетч TallyBlink (мигание индикатора).

Для этого выполните следующие действия

1   Запустите ПО Arduino.

2   В меню tools выберите плату Arduino и Port number (номер порта).

3   В меню File перейдите в Examples/BMDSDIControl и выберите скетч TallyBlink.

4   Загрузите его на вашу плату.

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                    // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);    // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                  // initialize tally control
  sdiTallyControl.setOverride(true);        // enable tally override

  pinMode(13, OUTPUT);                      // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                   // turn the LED ON

  sdiTallyControl.setCameraTally(           // turn tally ON
    1,                                      // Camera Number
    true,                                   // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it ON for 1 second

  digitalWrite(13, LOW);                    // turn the LED OFF

  sdiTallyControl.setCameraTally(           // turn tally OFF
    1,                                      // Camera Number
    false,                                  // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it OFF for 1 second
}
```

```
No changes necessary for Auto Format.
```

```
Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.
```

Самый быстрый способ — это открыть и запустить имеющийся скетч TallyBlink. Исходные данные можно передавать на модуль Blackmagic 3G-SDI Arduino Shield по протоколу I^{2}C, используя команды из Studio Camera Protocol. Чтобы упростить процесс написания скетчей, мы создали пользовательские библиотеки.

Tally-индикатор камеры Blackmagic Studio Camera должен мигать с интервалом в одну секунду, что означает наличие связи между Blackmagic 3G-SDI Arduino Shield и платой Arduino.

Если индикатор не мигает, проверьте, что Tally number (номер камеры) задан на 1, а функция Grid overlay камеры включена.

Помощь можно получить в разделе поддержки на странице www.blackmagicdesign.com/ru/support. Подробнее о настройках модуля см. в разделе «Помощь».

Светодиодные индикаторы

Blackmagic 3G-SDI Shield for Arduino имеет шесть светодиодных индикаторов, отображающих такие состояния модуля, как питание, работа в режиме UART, подключение шин I^{2}C и SPI, а также включение управления камерой и Tally-индикации.
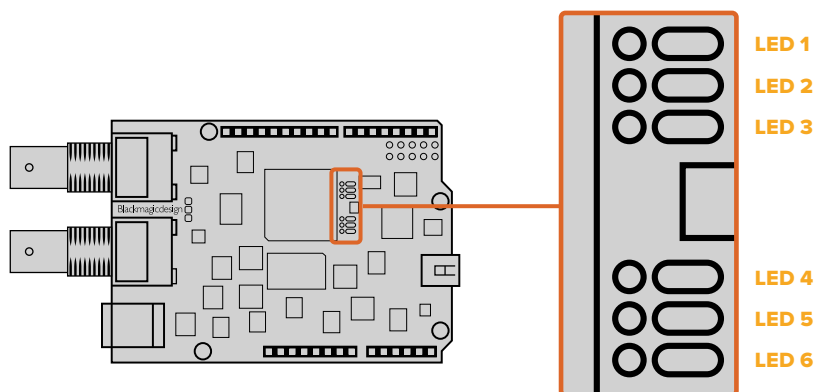
LED 1 - Система включена
Питание подается на модуль.

LED 2 - Включен режим управления
Режим управления камерой включен в скетч Arduino.

LED 3 - Включен Tally-индикатор
Tally-индикация включена в скетч Arduino.

LED 5 - Анализатор I^{2}C занят
Между модулем и платой Arduino обнаружена связь по протоколу I^{2}C.

LED 6 - Последовательный анализатор занят
Обнаружен сигнал в режиме UART.

Если при запуске модуля индикатор LED 1 отключен, а LED 3, LED 4 и LED 5 горят, это указывает на следующее

LED 3 - Идет загрузка блока памяти ПО

LED 4 - Инициализация микросхемы ЭСППЗУ

LED 5 - Идет проверка памяти

После успешного подключения модуля LED 1 загорится и все остальные индикаторы будут работать в обычном режиме.

В некоторых случаях, если при запуске модуля произошел сбой, будут мигать все индикаторы. Тот индикатор, где произошла ошибка, не будет мигать, что поможет выявить проблему.

Установка компонентов модуля

При разработке собственного контроллера для управления камерами легко создать дополнительный модуль любого типа и оснастить его кнопками, ручками и джойстиком. Соедините новый модуль с разъемами Blackmagic 3G-SDI Shield for Arduino. Потенциал создания таких контроллеров неограничен. При желании можно заменить прежнюю схему блока CCU на собственное решение на основе Arduino.
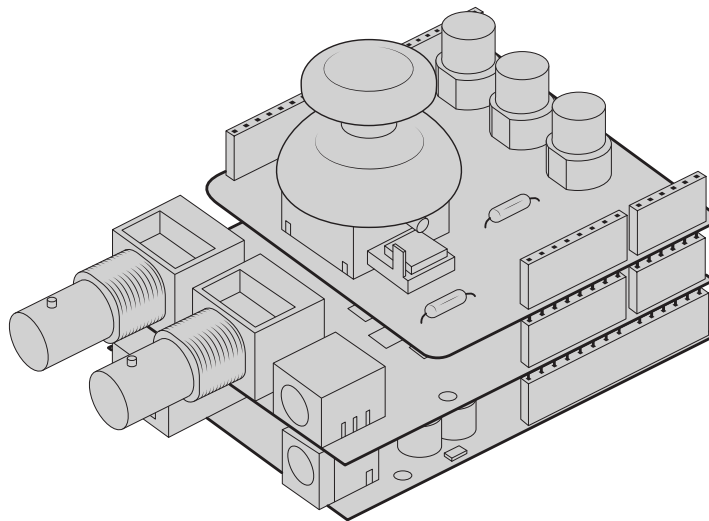
Разрабатывайте собственные микроконтроллеры
и подключайте их к Blackmagic 3G-SDI Shield for Arduino
для повышения точности управления

Обмен данными с модулем Blackmagic Shield for Arduino

Связь с модулем Blackmagic 3G-SDI Shield for Arduino возможна по I²C- или последовательному протоколу. Рекомендуется выбирать I²C-протокол из-за низкого числа контактов и доступности использования монитора с последовательным интерфейсом. Также это обеспечивает работу модуля с большим количеством I²C-устройств.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I²C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C    sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C     sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial        sdiCameraControl;
BMD_SDITallyControl_Serial         sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C    sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C     sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
Lens	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Confi-guration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header				command				data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Developer Information

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino..

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
--------------------  |----------------------------
 A4                   | Serial Data (SDA)
 A5                   | Serial Clock (SCL)
```

I²C Protocol (Writes):

          (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

          (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
--------------------  |----------------------------
 IO1                  | Serial Transmit (TX)
 IO0                  | Serial Receive (RX)
```

UART Protocol (Writes):

          [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

          [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
 Address            | Name         | R/W   | Register Description
-------------------- |-----------   |-----  |-----------------------------

 0x0000 - 0x0003     | IDENTITY     | R     | Hardware Identifier
 0x0004 - 0x0005     | HWVERSION    | R     | Hardware Version
 0x0006 - 0x0007     | FWVERSION    | R     | Firmware Version
                     |              |       |
 0x1000              | CONTROL      | R/W   | System Control
                     |              |       |
 0x2000              | OCARM        | R/W   | SDI Control Override Arm
 0x2001              | OCLENGTH     | R/W   | SDI Control Override Length
 0x2100 - 0x21FE     | OCDATA       | R/W   | SDI Control Override Data
                     |              |       |
 0x3000              | ICARM        | R/W   | SDI Control Incoming Arm
 0x3001              | ICLENGTH     | R     | SDI Control Incoming Length
 0x3100 - 0x31FE     | ICDATA       | R     | SDI Control Incoming Data
                     |              |       |
 0x4000              | OTARM        | R/W   | SDI Tally Override Arm
 0x4001              | OTLENGTH     | R/W   | SDI Tally Override Length
 0x4100 - 0x41FE     | OTDATA       | R/W   | SDI Tally Override Data
                     |              |       |
 0x5000              | ITARM        | R/W   | SDI Tally Incoming Arm
 0x5001              | ITLENGTH     | R     | SDI Tally Incoming Length
 0x5100 - 0x51FE     | ITDATA       | R     | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved:	Always zero.
Override Output:	When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.
Reset Tally:	When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.
Override Tally:	When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.
Override Control:	When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved:	Always zero.
Arm:	When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length:	Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data:	Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved:	Always zero.
Arm:	When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length:	Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Помощь

Как получить помощь

Модуль Blackmagic 3G-SDI Shield for Arduino позволяет создавать собственные решения.

Мы рекомендуем обратиться к страницам поддержки на сайте Blackmagic Design и проверить наличие последних справочных материалов по Blackmagic 3G-SDI Arduino Shield.

Страницы поддержки на сайте Blackmagic Design

Последние версии руководства по эксплуатации, программного обеспечения и дополнительную информацию можно найти в центре поддержки Blackmagic Design на странице www.blackmagicdesign.com/ru/support.

Форум разработчиков Arduino

Если у вас возникли вопросы по программированию, посетите интернет-форумы разработчиков Arduino. Вы не только найдете ответы на многие вопросы, но получите техническую поддержку по реализации программных решений.

Форум сообщества Blackmagic Design

Посетите форум сообщества Blackmagic Design на нашем веб-сайте, чтобы получить дополнительную информацию и узнать об интересных творческих идеях. Там также можно найти ответы опытных пользователей и сотрудников Blackmagic Design на часто задаваемые вопросы. Адрес форума http://forum.blackmagicdesign.com.

Проверка используемой версии программного обеспечения

Чтобы узнать версию Blackmagic 3G-SDI Shield for Arduino Setup, установленную на вашем компьютере, откройте окно About Blackmagic 3G-SDI Shield for Arduino Setup.

- На компьютере с операционной системой Mac OS X откройте Blackmagic 3G-SDI Shield for Arduino Setup в папке «Приложения». В меню выберите About Blackmagic 3G-SDI Shield for Arduino Setup, чтобы узнать номер версии.
- При работе в операционной системе Windows 7 откройте меню «Пуск» и выберите Blackmagic 3G-SDI Shield for Arduino Setup. В меню «Помощь» выберите About Blackmagic 3G-SDI Shield for Arduino Setup, чтобы узнать номер версии.
- При работе в операционной системе Windows 8 на экране «Пуск» выберите Blackmagic 3G-SDI Shield for Arduino Setup. В меню «Помощь» выберите About Blackmagic 3G-SDI Shield for Arduino Setup, чтобы узнать номер версии.

Загрузка последних версий программного обеспечения

Узнав установленную версию ПО Blackmagic 3G-SDI Shield for Arduino Setup, перейдите в центр поддержки Blackmagic Design на странице www.blackmagicdesign.com/ru/support, чтобы проверить наличие обновлений. Рекомендуется всегда использовать последнюю версию программного обеспечения, однако обновление лучше всего выполнять после завершения текущего проекта.

Гарантия

Ограниченная гарантия сроком 12 месяцев

Компания Blackmagic Design гарантирует отсутствие в данном модуле Blackmagic 3G-SDI Shield for Arduino дефектов материала и производственного брака в течение 12 месяцев с даты продажи. Если во время гарантийного срока будут выявлены дефекты, Blackmagic Design по своему усмотрению выполнит ремонт неисправного изделия без оплаты стоимости запчастей и трудозатрат или заменит такое изделие новым.

Чтобы воспользоваться настоящей гарантией, потребитель обязан уведомить компанию Blackmagic Design о дефекте до окончания гарантийного срока и обеспечить условия для предоставления необходимых услуг. Потребитель несет ответственность за упаковку и доставку неисправного изделия в соответствующий сервисный центр Blackmagic Design с оплатой почтовых расходов. Потребитель обязан оплатить все расходы по доставке и страхованию, пошлины, налоги и иные сборы в связи с возвратом изделия вне зависимости от причины возврата.

Настоящая гарантия не распространяется на дефекты, отказы и повреждения, возникшие из-за ненадлежащего использования, неправильного ухода или обслуживания. Компания Blackmagic Design не обязана предоставлять услуги по настоящей гарантии: а) для устранения повреждений, возникших в результате действий по установке, ремонту или обслуживанию изделия лицами, которые не являются персоналом Blackmagic Design; б) для устранения повреждений, возникших в результате ненадлежащего использования или подключения к несовместимому оборудованию; в) для устранения повреждений или дефектов, вызванных использованием запчастей или материалов других производителей; г) если изделие было модифицировано или интегрировано с другим оборудованием, когда такая модификация или интеграция увеличивает время или повышает сложность обслуживания изделия. НАСТОЯЩАЯ ГАРАНТИЯ ПРЕДОСТАВЛЯЕТСЯ КОМПАНИЕЙ BLACKMAGIC DESIGN ВМЕСТО ЛЮБЫХ ДРУГИХ ПРЯМО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ. КОМПАНИЯ BLACKMAGIC DESIGN И ЕЕ ДИЛЕРЫ ОТКАЗЫВАЮТСЯ ОТ ЛЮБЫХ ПОДРАЗУМЕВАЕМЫХ ГАРАНТИЙ КОММЕРЧЕСКОЙ ЦЕННОСТИ ИЛИ ПРИГОДНОСТИ ДЛЯ КАКОЙ-ЛИБО ОПРЕДЕЛЕННОЙ ЦЕЛИ. ОТВЕТСТВЕННОСТЬ BLACKMAGIC DESIGN ПО РЕМОНТУ ИЛИ ЗАМЕНЕ НЕИСПРАВНЫХ ИЗДЕЛИЙ ЯВЛЯЕТСЯ ПОЛНЫМ И ИСКЛЮЧИТЕЛЬНЫМ СРЕДСТВОМ ВОЗМЕЩЕНИЯ, ПРЕДОСТАВЛЯЕМЫМ ПОТРЕБИТЕЛЮ В СВЯЗИ С КОСВЕННЫМИ, ФАКТИЧЕСКИМИ, СОПУТСТВУЮЩИМИ ИЛИ ПОСЛЕДУЮЩИМИ УБЫТКАМИ, ВНЕ ЗАВИСИМОСТИ ОТ ТОГО, БЫЛА ИЛИ НЕТ КОМПАНИЯ BLACKMAGIC DESIGN (ЛИБО ЕЕ ДИЛЕР) ПРЕДВАРИТЕЛЬНО ИЗВЕЩЕНА О ВОЗМОЖНОСТИ ТАКИХ УБЫТКОВ. BLACKMAGIC DESIGN НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА ПРОТИВОПРАВНОЕ ИСПОЛЬЗОВАНИЕ ОБОРУДОВАНИЯ СО СТОРОНЫ ПОТРЕБИТЕЛЯ. BLACKMAGIC DESIGN НЕ НЕСЕТ ОТВЕТСТВЕННОСТИ ЗА УБЫТКИ, ВОЗНИКАЮЩИЕ ВСЛЕДСТВИЕ ИСПОЛЬЗОВАНИЯ ЭТОГО ИЗДЕЛИЯ. ПОТРЕБИТЕЛЬ ПРИНИМАЕТ НА СЕБЯ РИСКИ, СВЯЗАННЫЕ С ЕГО ЭКСПЛУАТАЦИЕЙ.

Manuale di istruzioni

Blackmagic 3G-SDI Shield for Arduino

Febbraio 2020

Italiano

Gentile utente

Grazie per aver acquistato il nuovo Blackmagic 3G-SDI Shield for Arduino.

Il team Blackmagic è sempre alla ricerca di nuove tecnologie per consentire il controllo dei nostri dispositivi SDI senza limiti di creatività. Grazie a questo shield, Arduino si integra perfettamente nel flusso di lavoro SDI come soluzione di controllo aggiuntiva per i dispositivi Blackmagic Design.

Per esempio, puoi gestire Blackmagic URSA Mini e Blackmagic Studio Camera dallo switcher ATEM tramite il pacchetto dati integrato al segnale SDI. Se lo switcher ATEM non è in funzione, ma desideri comunque controllare le telecamere Blackmagic, puoi creare una soluzione di controllo su misura con il 3G-SDI Shield for Arduino. Lo shield funge da piattaforma SDI attraverso cui passa il flusso video del programma in arrivo dallo switcher e verso le telecamere Blackmagic.

In questo manuale troverai tutti i comandi utili per scrivere il codice con facilità e controllare la telecamera.

Puoi gestire la telecamera tramite software da un computer, oppure aggiungere pulsanti, manopole e joystick fisici allo shield per regolare diverse funzioni, tra cui messa a fuoco, zoom, apertura, bilanciamento nero/bianco, e i potenti strumenti di correzione colore della camera. Creare dispositivi personalizzati è utilissimo per le tue produzioni, ma anche divertente!

Questa è una tecnologia innovativa e versatile, che apre le porte a innumerevoli soluzioni di controllo SDI su misura.

Il manuale di istruzioni contiene tutte le informazioni per installare e operare Blackmagic 3G-SDI Shield for Arduino. La versione più recente di questo manuale e gli aggiornamenti del software interno dello shield sono disponibili sulla pagina Supporto del nostro sito www.blackmagicdesign.com/it. È importante aggiornare regolarmente il software per disporre sempre delle ultime funzioni. Una volta scaricato il software, ti invitiamo a registrare i tuoi dati personali per ricevere le notifiche sugli aggiornamenti futuri. Blackmagic è in costante stato di innovazione e i tuoi suggerimenti ci aiutano a migliorare prestazioni e funzionalità!

Grant Petty
AD di Blackmagic Design

Indice

# Blackmagic 3G-SDI Shield for Arduino

Operazioni preliminari

Inserire e saldare i connettori

Blackmagic 3G-SDI Shield for Arduino include 4 connettori sovrapponibili: due connettori a 8 pin, un connettore a 10 pin, e un connettore a 6 pin. Questi connettori fungono da collegamento tra lo shield e la scheda Arduino, ed essendo sovrapponibili, permettono di aggiungere altri shield dotati di ulteriori componenti, per esempio pulsanti, manopole e joystick. Il connettore consente di montare lo shield su schede Arduino UNO R3.

Per applicare i connettori allo shield:

1 Inserisci i pin di ogni connettore negli appositi fori per pin in entrambi i lati dello shield Blackmagic. L'illustrazione qui sotto mostra la disposizione dei connettori.
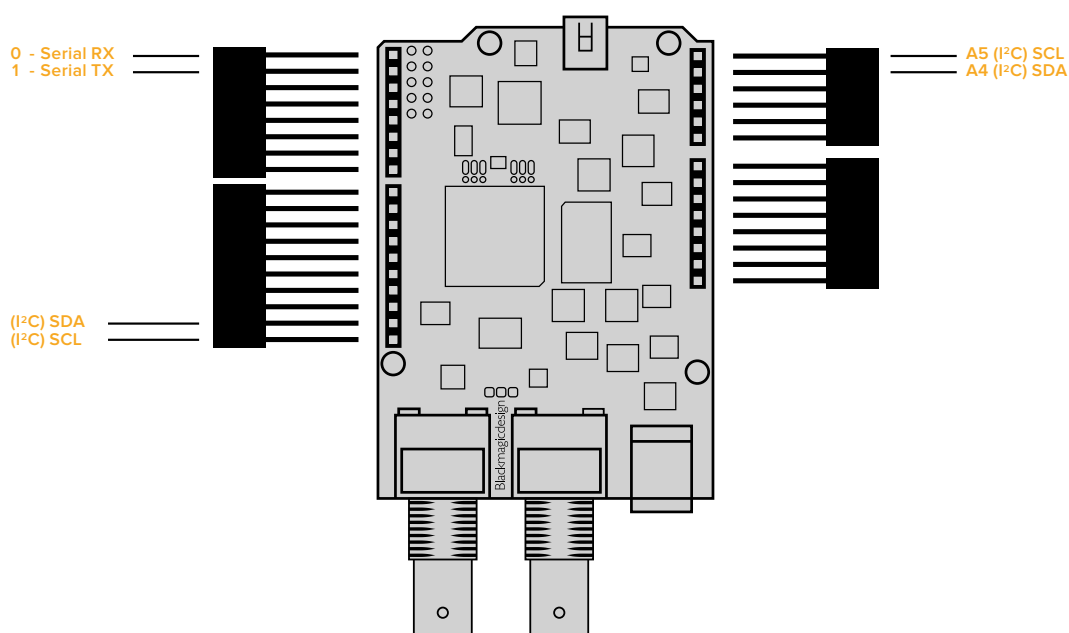
NOTA La comunicazione con lo shield avviene tramite I²C o è seriale. Consigliamo I²C perché usa il monitor seriale e lascia liberi tutti gli altri pin. Seleziona la modalità di comunicazione in fase di scrittura del comando BMDSDIControl nello sketch. Per tutti i dettagli, consulta il capitolo "Comunicare con lo shield".

2 Salda la base di ogni connettore sulla parte inferiore dello shield. Assicurati di saldare fermamente ogni singolo pin, evitando di toccare la saldatura degli altri pin.

Inserire lo shield nella scheda Arduino

Ora che i connettori sono saldati allo shield 3G-SDI, puoi montarlo sulla scheda Arduino.

Allinea i connettori dello shield ai fori per pin della scheda Arduino, e inseriscili delicatamente, evitando di piegarli.
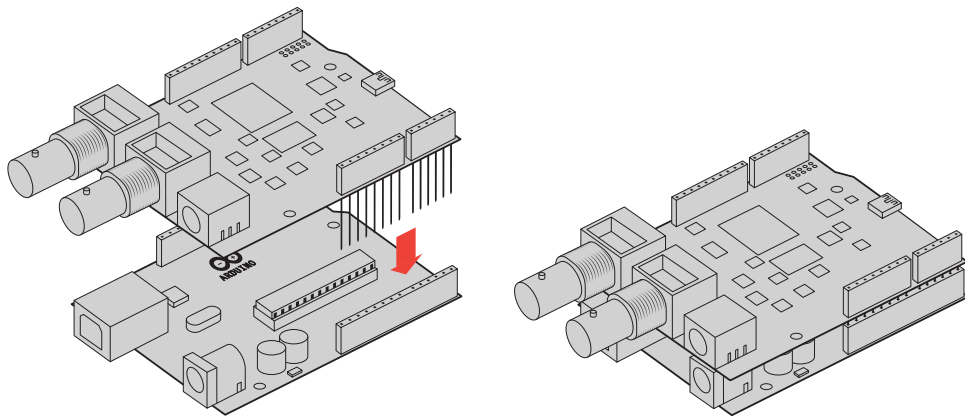
Una volta inseriti i pin, lo shield Blackmagic e la scheda Arduino si incastrano perfettamente.

Collegare l'alimentazione

Per alimentare Blackmagic 3G-SDI Shield for Arduino, collega un adattatore all'ingresso per alimentazione 12V dello shield.
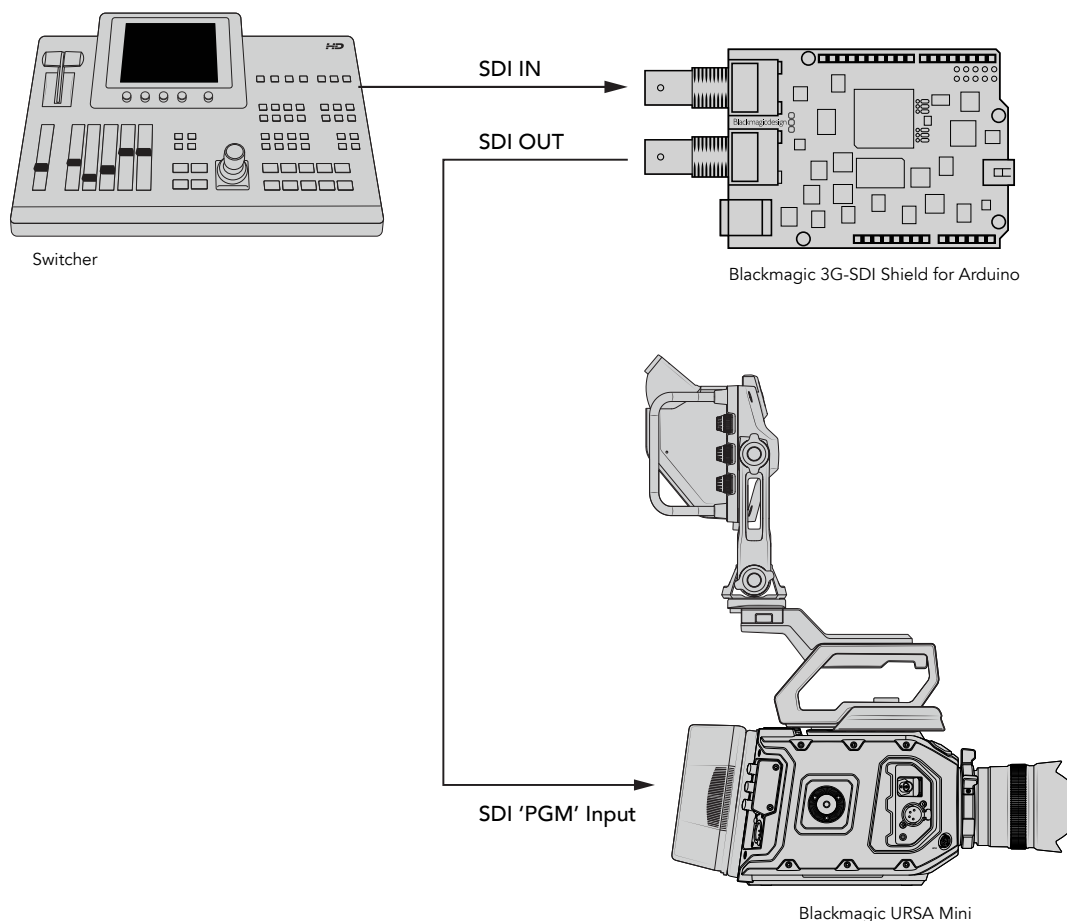
Collegare i dispositivi SDI

Una volta collegata l'alimentazione, puoi connettere Blackmagic 3G-SDI Shield for Arduino ai dispositivi SDI. Ad esempio, per collegare lo shield a uno switcher e a una Blackmagic URSA Mini:

1 Collega l'uscita di programma dello switcher all'ingresso SDI dello shield Blackmagic.

2 Collega l'uscita SDI dello shield Blackmagic all'ingresso PGM di Blackmagic URSA Mini.

Segui lo schema di connessione qui sotto.

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini

Con lo shield montato sulla scheda Arduino, alimentato e connesso ai dispositivi SDI, è il momento di installare il software interno e i file della libreria, programmare il software Arduino e cominciare a utilizzare lo shield!

Il prossimo capitolo spiega come installare il software interno dello shield e dove installare i file della libreria di Arduino, cosicché lo shield comunichi con la scheda Arduino.

> **SUGGERIMENTO** Con Blackmagic 3G-SDI Shield for Arduino puoi controllare anche altri dispositivi Blackmagic Design, tra cui Blackmagic MultiView 16. Per esempio se lo shield è connesso all'ingresso 16, il bordo tally si visualizza sui riquadri multiview. Se usi una Blackmagic URSA Mini, seleziona l'overlay delle griglie sulla cinepresa per abilitare il tally sul viewfinder. Consulta il manuale di URSA Mini per maggiori informazioni.

Installare il software

NOTA Prima di installare l'utilità Blackmagic 3G-SDI Shield for Arduino, scarica l'ultimo software di programmazione (IDE) di Arduino da www.arduino.cc e installalo sul tuo computer.

Dopo aver installato il software di Arduino, puoi installare il software interno dello shield Blackmagic.

Installare il software interno

Installa Blackmagic Shield for Arduino Setup per aggiornare il software interno dello shield. Il software interno comunica con la scheda Arduino, e la controlla utilizzando i file della libreria Arduino. I file sono installati dal software, quindi basta fare un copia e incolla della "Library" nella cartella Arduino in Applicazioni. Per maggiori informazioni consulta la sezione "Impostazioni" di questo manuale.

Consigliamo di scaricare l'ultima versione del software Blackmagic Shield for Arduino e aggiornare lo shield per sfruttare appieno tutte le funzioni e i potenziamenti futuri. La versione più recente è disponibile per il download sulla pagina Supporto di Blackmagic Design www.blackmagicdesign.com/it/support

Per installare il software interno con Mac OS X:

1   Scarica e decomprimi il software Blackmagic Shield for Arduino Setup.

2   Apri l'immagine disco e lancia l'applicazione di installazione. Segui le istruzioni sullo schermo.

3   Una volta installata l'ultima versione del software dello shield Blackmagic, alimenta lo shield e connettilo al computer tramite un cavo USB.

4   Ora lancia l'applicazione di installazione e segui le istruzioni per aggiornare il software interno dello shield. Se la finestra non appare, il software interno è già aggiornato.

Per installare il software interno con Windows:

1   Scarica e decomprimi il software Blackmagic Shield for Arduino Setup.

2   Apparirà la cartella Blackmagic Shield for Arduino, che contiene questo manuale e l'installer dello shield Blackmagic. Fai doppio clic sull'icona dell'applicazione e segui le istruzioni per completare l'installazione.

3   Una volta installata l'ultima versione del software dello shield Blackmagic, alimenta lo shield e connettilo al computer tramite un cavo USB.

4   Ora lancia l'applicazione di installazione e segui le istruzioni per aggiornare il software interno dello shield. Se la finestra non appare, il software interno è già aggiornato.

Installare i file della libreria Arduino

I programmi sviluppati per il controllo della scheda Arduino sono chiamati "sketch". Blackmagic 3G-SDI Shield for Arduino si serve dei file della libreria Arduino, che facilitano la scrittura degli sketch. Dopo l'installazione del software dello shield, i file della libreria appaiono nella cartella denominata "Library", che basta copiare e incollare nella cartella Arduino.

NOTA Chiudi l'IDE Arduino prima di installare le librerie.

Per installare i file della libreria su Mac OS X:

1 Apri la cartella Applicazioni e seleziona Blackmagic Shield for Arduino.

2 Apri la cartella "Library" e fai clic destro/copia della cartella BMDSDIControl.

3 Vai a Documenti e apri la cartella Arduino.

4 Troverai una sotto-cartella chiamata "Libraries". Incolla la cartella BMDSDIControl copiata nella cartella "Libraries".

Per installare i file della libreria su Windows

1 Vai a Programmi/Blackmagic Shield for Arduino

2 Apri la sotto-cartella "Library", poi fai clic destro/copia della cartella BMDSDIControl.

3 Vai a Documenti e apri la cartella Arduino.

4 Troverai una sotto-cartella chiamata "Libraries". Incolla la cartella BMDSDIControl copiata nella cartella "Libraries".

Ora i file della libreria Blackmagic Design sono installati sul tuo computer, inclusi gli esempi di sketch Blackmagic Design utilizzabili quando il software Arduino è in esecuzione.

Clicca "File" sulla barra menù del software Arduino e seleziona "Examples". Clicca su BMDSDIControl per accedere agli esempi di sketch.

Ora che i file della libreria sono salvati nella cartella corretta, lo shield può utilizzarli per comunicare con la scheda Arduino. A questo punto basta programmare l'IDE Arduino. Per tutti i dettagli, consulta il capitolo "Programming Arduino Sketches".

NOTA Se in futuro viene resa disponibile una nuova libreria di esempi, cestina la cartella BMDSDIControl e sostituiscila con la nuova cartella, seguendo le stesse istruzioni qui sopra.

Blackmagic Shield for Arduino Setup
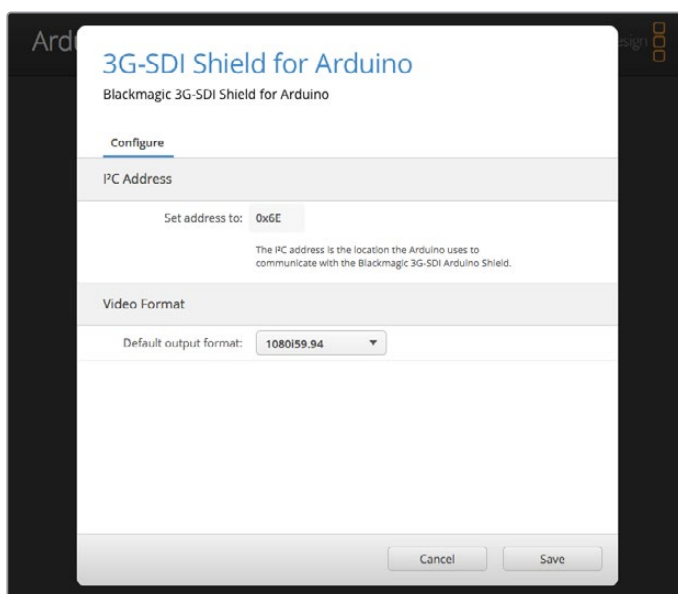
Blackmagic Shield for Arduino Setup consente di cambiare le impostazioni
dello shield, così come l'indirizzo I²C e il formato video di uscita.

A installazione effettuata, puoi cambiare sia l'indirizzo "I²C address" per identificare lo shield e comunicare con la scheda Arduino, sia il formato "video format" di uscita dello shield.

Indirizzo I²C

Nei rari casi in cui è presente un secondo shield che utilizza lo stesso indirizzo I²C dello shield Blackmagic, sarà necessario cambiare le impostazioni di default.

L'indirizzo di default del tuo shield è 0x6E, ma puoi scegliere un valore compreso tra 0x08 e 0x77.

Per cambiare l'indirizzo dello shield:

1 Lancia Blackmagic Shield for Arduino Setup e clicca sull'icona delle impostazioni.

2 Assegna l'indirizzo desiderato in "Set address to:"

3 Clicca su "Save".

Formato video

Se non è collegata nessuna fonte, il formato di uscita è selezionato di default. Quando è rilevata una fonte, il formato di uscita corrisponde a quello di entrata. Scollegando la fonte di entrata, il formato di uscita ritorna a quello di default. Per cambiare il formato video di default, clicca su "Default output format" dalla barra menù e seleziona il formato desiderato.

Scegli tra i seguenti formati di uscita:

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Programmare gli sketch di Arduino

I programmi Arduino, detti "sketch", sono facilissimi da scrivere e usano il comune linguaggio di programmazione C. Se programmi gli sketch utilizzando i comandi del protocollo dello Studio Camera Control Protocol, lo shield li integra all'uscita SDI, permettendo di controllare Blackmagic URSA Mini o Blackmagic Studio Camera.

Tutti i comandi supportati sono inclusi nella sezione "Studio Camera Control Protocol" del manuale.

Testare lo shield Blackmagic e installazione della libreria

Una volta completate le operazioni preliminari e installato il software e i file della libreria, accertati che lo shield e la scheda Arduino comunichino correttamente.

Il metodo più veloce è di aprire e testare l'esempio di sketch per fare lampeggiare la spia tally.

Segui le istruzioni qui sotto:

1 Lancia il software di programmazione (IDE) di Arduino.

2 Dal menù "Tools" seleziona la scheda Arduino e il numero della porta seriale.

3 Dal menù "File" seleziona "Examples/BMDSDIControl" e clicca sullo sketch "TallyBlink".

4 Carica lo sketch sulla scheda.

```
TallyBlink §
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                    // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);    // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                    // initialize tally control
  sdiTallyControl.setOverride(true);          // enable tally override

  pinMode(13, OUTPUT);                        // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                     // turn the LED ON

  sdiTallyControl.setCameraTally(             // turn tally ON
    1,                                        // Camera Number
    true,                                     // Program Tally
    false                                     // Preview Tally
  );

  delay(1000);                                // leave it ON for 1 second

  digitalWrite(13, LOW);                      // turn the LED OFF

  sdiTallyControl.setCameraTally(             // turn tally OFF
    1,                                        // Camera Number
    false,                                    // Program Tally
    false                                     // Preview Tally
  );

  delay(1000);                                // leave it OFF for 1 second
}

No changes necessary for Auto Format.

Sketch uses 4,786 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.
```

Con l'esempio di sketch per illuminare la spia tally, verificare il funzionamento dello Blackmagic 3G-SDI Shield for Arduino è facile e veloce. Invia i dati direttamente allo shield tramite I²C, utilizzando i comandi del protocollo Studio Camera Protocol. Per facilitare la programmazione degli sketch puoi consultare le librerie incluse.

Se la spia tally lampeggia su Blackmagic Studio Camera a intervalli di un secondo, la comunicazione tra lo shield e Arduino è stata instaurata con successo.

Se la spia tally non lampeggia, controlla che il numero di tally della camera si impostato su 1 e che l'overlay delle griglie sia abilitato.

Per assistenza, visita il sito Blackmagic Design alla pagina Supporto www.blackmagicdesign.com/it/support. Consulta la sezione "Assistenza" di questo manuale per scoprire come ottenere supporto tecnico.

Spie LED

Blackmagic 3G-SDI Shield for Arduino include sei LED che confermano varie operazioni dello shield, tra cui alimentazione, UART, I²C e comunicazione SPI, e spie che mostrano quando il controllo tally e il controllo telecamera sono abilitati.
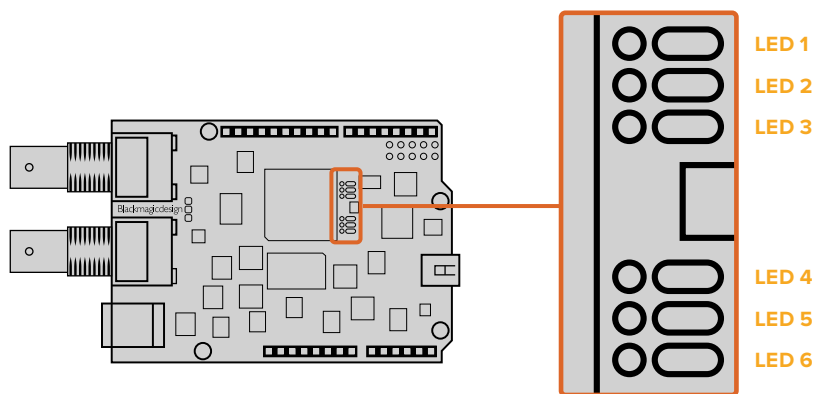
LED 1 - Sistema attivo
Si illumina quando l'alimentazione è collegata allo shield.

LED 2 - Controllo telecamera abilitato
Si illumina quando lo sketch di Arduino prevede il controllo della telecamera.

LED 3 - Controllo tally abilitato
Si illumina quando lo sketch di Arduino prevede il controllo del tally.

LED 5 - I²C Parsing in corso
Si illumina quando viene rilevata la comunicazione tra shield e Arduino con il protocollo I²C.

LED 6 - Parsing UART in corso
Si illumina quando viene rilevata la comunicazione seriale UART.

Quando lo shield Blackmagic è in fase di avvio, l'indicatore dell'alimentazione rimane spento, e i LED 3, 4 e 5 si accendono a seconda dell'attività:

LED 3 - Caricamento in corso dell'immagine dell'applicazione

**LED 4 - Avvio di EEPROM**

LED 5 - Controllo memoria in corso

Ad avvio riuscito, il LED di alimentazione si accende, e tutti i LED riprendono le loro funzioni standard durante l'elaborazione.

Nel raro caso che l'avvio fallisca, tutti i LED lampeggiano velocemente tranne quelli che mostrano la causa dell'anomalia.

Applicare componenti allo shield

Per ottenere un dispositivo di controllo hardware su misura e interamente manuale, puoi creare un nuovo shield con pulsanti, manopole e joystick. Per montarlo su Blackmagic 3G-SDI Shield for Arduino basta inserire i connettori dello shield personalizzato nei fori per pin del connettore dello shield. Non c'è limite al tipo di microcontrollori che puoi creare. Puoi addirittura sostituire completamente i circuiti di una vecchia CCU con la tua soluzione Arduino su misura e di standard professionale.
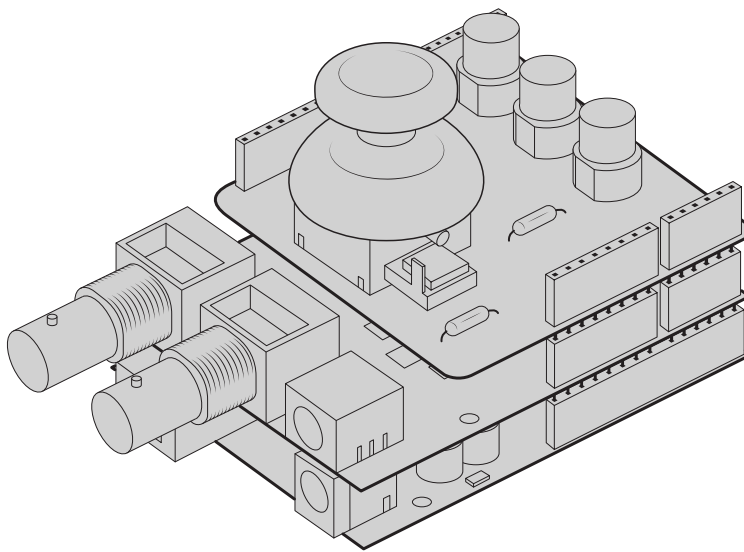
Crea il tuo controller su misura e connettilo a Blackmagic 3G-SDI Shield for Arduino per un controllo mirato e interattivo.

Comunicare con Blackmagic Shield for Arduino (English)

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I²C or Serial. We recommend I²C because of the low pin count and it frees up the serial monitor. This also allows you to use more I²C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I²C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I²C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I²C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial      sdiCameraControl;
BMD_SDITallyControl_Serial       sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I²C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I²C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol (English)

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
                8 = Color Correction group
                0 = Lift Adjust
                0 = assign value
                liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
Video	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Confi-guration	7.0	Real Time Clock	int32	[0] time	–	–	BCD - HHMMSSFF (UCT)
				[1] date	–	–	BCD - YYYYMMDD
	7.1	System language	string	–	–	–	ISO-639-1 two character language code
	7.2	Timezone	int32	–	–	–	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	–	–	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	–	–	BCD - sDDDddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte 0 destination	1 length	2 command	3 reserved	4 category	5 parameter	6 type	7 operation	8	9	10	11	12	13	14	15
		header		command						data							
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Informazioni per gli sviluppatori (English)

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin        | Function
-------------------- |----------------------------
 A4                 | Serial Data (SDA)
 A5                 | Serial Clock (SCL)
```

I²C Protocol (Writes):

        (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

        (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin        | Function
-------------------- |----------------------------
 IO1                | Serial Transmit (TX)
 IO0                | Serial Receive (RX)
```

UART Protocol (Writes):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address              | Name        | R/W     | Register Description
-------------------- |----------   |-----    |------------------------------

0x0000 - 0x0003      | IDENTITY    | R       | Hardware Identifier
0x0004 - 0x0005      | HWVERSION   | R       | Hardware Version
0x0006 - 0x0007      | FWVERSION   | R       | Firmware Version
                     |             |         |
0x1000               | CONTROL     | R/W     | System Control
                     |             |         |
0x2000               | OCARM       | R/W     | SDI Control Override Arm
0x2001               | OCLENGTH    | R/W     | SDI Control Override Length
0x2100 - 0x21FE      | OCDATA      | R/W     | SDI Control Override Data
                     |             |         |
0x3000               | ICARM       | R/W     | SDI Control Incoming Arm
0x3001               | ICLENGTH    | R       | SDI Control Incoming Length
0x3100 - 0x31FE      | ICDATA      | R       | SDI Control Incoming Data
                     |             |         |
0x4000               | OTARM       | R/W     | SDI Tally Override Arm
0x4001               | OTLENGTH    | R/W     | SDI Tally Override Length
0x4100 - 0x41FE      | OTDATA      | R/W     | SDI Tally Override Data
                     |             |         |
0x5000               | ITARM       | R/W     | SDI Tally Incoming Arm
0x5001               | ITLENGTH    | R       | SDI Tally Incoming Length
0x5100 - 0x51FE      | ITDATA      | R       | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [
OVERIDE CONTROL]
7   4   3   2   1   0

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Assistenza

Assistenza clienti

Blackmagic 3G-SDI Shield for Arduino è uno strumento per programmatori ideato per sviluppare soluzioni di controllo su misura.

Per ottenere maggiori informazioni, visita la pagina Supporto di Blackmagic Design, dove troverai sempre il materiale di supporto più recente.

Pagina di supporto online di Blackmagic Design

Per il materiale più recente, inclusi software e note di supporto, visita il sito Blackmagic Design alla pagina www.blackmagicdesign.com/it/support

Forum per sviluppatori Arduino

Trova la risposta alle tue domande di programmazione nei forum di sviluppo Arduino online. La comunità di sviluppatori Arduino è numerosa, e sui forum online puoi persino trovare un ingegnere per implementare la soluzione su misura per te.

Il Forum Blackmagic Design

La pagina Forum del nostro sito Blackmagic Design è un'ottima risorsa per ottenere informazioni utili e condividere idee creative. Qui troverai le risposte alle domande più frequenti, oltre ai consigli degli utenti esistenti e dello staff Blackmagic Design. Visita il nostro Forum su https://forum.blackmagicdesign.com

Verificare la versione del software

Per verificare quale versione del software Blackmagic 3G-SDI Shield for Arduino è installata sul tuo computer, apri la tab "About" del software.

- Su Mac OS X, apri la cartella Applicazioni e seleziona Blackmagic 3G-SDI Shield for Arduino. Dal menù clicca su About Blackmagic Shield for Arduino per scoprirne la versione.

- Su Windows 7, seleziona Blackmagic 3G-SDI Shield for Arduino Setup dal menù Start. Dal menù Help clicca su About Blackmagic 3G-SDI Shield for Arduino per scoprirne la versione.

- Su Windows 8, seleziona Blackmagic 3G-SDI Shield for Arduino Setup dall'omonima icona nella pagina Start. Dal menù Help clicca su About Blackmagic Shield for Arduino Setup per scoprirne la versione.

Dove trovare gli aggiornamenti più recenti del software

Dopo aver verificato quale versione del software Blackmagic 3G-SDI Shield for Arduino è installata sul tuo computer, visita il Centro assistenza di Blackmagic Design su www.blackmagicdesign.com/it/support per scaricare gli aggiornamenti più recenti. Consigliamo di non aggiornare il software se stai già lavorando a un progetto importante.

Garanzia

Garanzia limitata di un anno

Blackmagic Design garantisce che Blackmagic 3G-SDI Shield for Arduino è fornito privo di difetti nei materiali e nella manifattura per un periodo di un anno a partire dalla data d'acquisto. Durante il periodo di garanzia Blackmagic Design riparerà o, a sua scelta, sostituirà tutti i componenti che risultino difettosi esonerando il Cliente da costi aggiuntivi, purché questi vengano restituiti dal Cliente.

Per ottenere l'assistenza coperta dalla presente garanzia, il Cliente deve notificare Blackmagic Design del difetto entro il periodo di garanzia. Il Cliente è responsabile del costo di imballaggio e di spedizione del prodotto al centro di assistenza indicato da Blackmagic Design, con spese di spedizione prepagate. Il costo include spedizione, assicurazione, tasse, dogana, e altre spese pertinenti alla resa del prodotto a Blackmagic Design.

Questa garanzia perde di validità per danni causati da utilizzo improprio, o da manutenzione e cura inadeguate del prodotto. Blackmagic Design non ha obbligo di assistenza e riparazione sotto garanzia per danni al prodotto risultanti da: a) precedenti tentativi di installazione, riparazione o manutenzione da personale non autorizzato, ovvero al di fuori del personale Blackmagic Design, b) precedenti usi impropri o tentativi di connessione ad attrezzatura incompatibile al prodotto, c) precedente uso di parti o ricambi non originali Blackmagic Design, o d) precedenti modifiche o integrazione del prodotto ad altri prodotti, con il risultato di rendere la riparazione più difficoltosa o di allungare le tempistiche di eventuali ispezioni atte alla riparazione. La presente garanzia di Blackmagic Design sostituisce qualsiasi altra garanzia, esplicita o implicita. Blackmagic Design e i suoi fornitori escludono qualsiasi altra garanzia implicita di commerciabilità o di idoneità ad un uso specifico. L'intera responsabilità di Blackmagic Design e l'unico esclusivo ricorso dell'utente per qualsiasi danno arrecato di natura indiretta, specifica, accidentale o consequenziale, anche qualora Blackmagic Design fosse stata avvertita della possibilità di tali danni, è la riparazione o la sostituzione dei prodotti difettosi. Blackmagic Design non si assume alcuna responsabilità per qualsiasi uso illegale del dispositivo da parte del Cliente. Blackmagic Design non si assume alcuna responsabilità per danni derivanti dall'uso di questo prodotto. Il Cliente utilizza questo prodotto a proprio rischio.
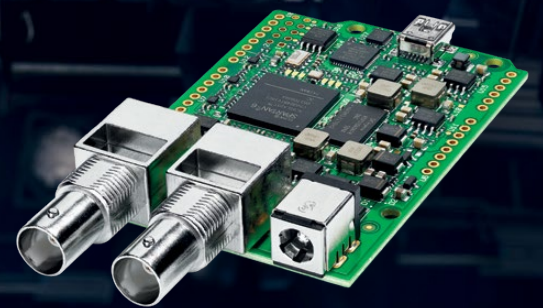
Manual de Instalação e Operação

Blackmagic 3G-SDI Shield for Arduino

Fevereiro 2020

Português

Prezado Cliente,

Obrigado por adquirir o seu novo Blackmagic 3G-SDI Shield for Arduino.

Estamos sempre interessados em novas tecnologias e bastante entusiasmados por todas as maneiras criativas em que os nossos produtos SDI podem ser utilizados. Com o 3G-SDI Shield for Arduino, você agora pode integrar o Arduino ao seu fluxo de trabalho SDI para obter mais opções de controle com seus equipamentos Blackmagic Design.

Por exemplo, os switchers ATEM podem controlar unidades Blackmagic URSA Mini e Blackmagic Studio Camera através de pacotes de dados integrados ao sinal SDI. Caso não esteja operando um switcher ATEM, mas ainda gostaria de ter a capacidade de controlar suas câmeras Blackmagic, você pode criar soluções personalizadas com o seu 3G-SDI Shield for Arduino. O shield oferece a plataforma SDI para desenvolver, assim você pode fazer o loop da alimentação de retorno de programa do seu switcher, pelo shield, com a alimentação de programa das câmeras Blackmagic.

Programar o código para enviar os comandos à câmera é fácil e todos os comandos suportados estão incluídos neste manual.

É possível controlar as câmeras usando um computador ou você pode adicionar botões, potenciômetros e joysticks ao seu shield e montar controladores hardware dinâmicos para ajustar configurações como, o foco e zoom da lente, ajustes de abertura, controle de balanço do pedestal e brancos, o poderoso corretor de cores integrado à câmera e muito mais. Montar o seu próprio controlador personalizado é útil para a produção, mas também é muito divertido.

Estamos muito satisfeitos com esta tecnologia e gostaríamos de ouvir sobre os controladores SDI que você criar para o seu 3G-SDI Shield for Arduino!

Este manual de instruções contém todas as informações necessárias para começar a usar o seu Blackmagic 3G-SDI Shield for Arduino. Consulte a página de suporte no nosso site em www.blackmagicdesign.com/br para obter a versão mais recente deste manual e as atualizações do software interno do seu shield. Para garantir que você receba todos os recursos mais recentes, mantenha o seu programa atualizado. Ao baixar o software, registre suas informações para que possamos informá-lo quando novas versões do software forem lançadas. Estamos sempre trabalhando com novos recursos e aprimoramentos, portanto adoraríamos ouvir a sua opinião.

Grant Petty

Grant Petty
Diretor Executivo da Blackmagic Design

Índice

Blackmagic 3G-SDI Shield for Arduino

Instruções preliminares

Encaixe e soldagem dos headers

O seu módulo Blackmagic 3G-SDI Shield for Arduino é fornecido com quatro headers empilháveis, incluindo dois headers de 8 pinos, um de 10 pinos e um de 6 pinos. Os headers são conectores ponte utilizados para encaixar seu shield à placa Arduino e, como são empilháveis, você pode encaixar outros shields acima com componentes adicionais, como botões, potenciômetros e joysticks de controle. A disposição do header é compatível para montagem em placas Arduino com representação real (footprint) R3, como a Arduino UNO.

Como encaixar os headers ao seu shield:

1 Insira os pinos de cada header pelos orifícios de pino correspondentes em cada lateral do Blackmagic 3G-SDI Shield. Consulte a ilustração abaixo para um esquema de disposição dos headers.
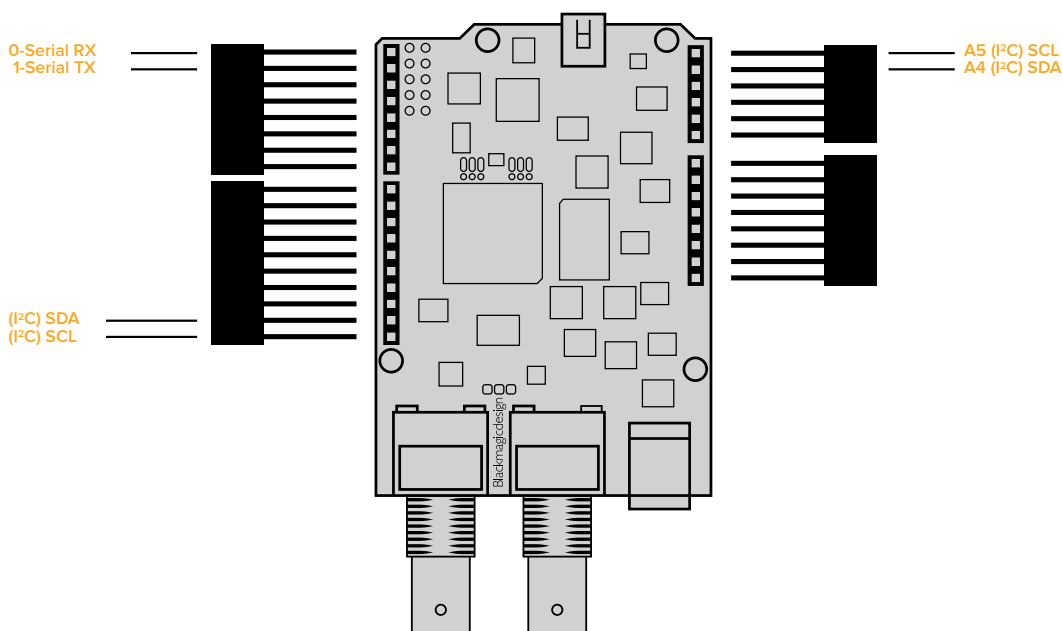
OBSERVAÇÃO Ao se conectar com o shield, a comunicação é através de I²C ou Serial. Recomendamos I²C, já que isso possibilita usar o Serial Monitor e disponibiliza todos os outros pinos. Selecione o modo de comunicação ao definir o objeto BMDSDIControl no sketch. Consulte a seção 'Comunicação com o Blackmagic 3G-SDI Shield for Arduino' para mais informações.

2 Solde a base de cada pino header ao lado de baixo do seu shield. Certifique-se de que a soldagem de cada pino crie uma junção firme com o orifício do pino, mas não encoste na solda dos pinos vizinhos.

Encaixe à placa Arduino

Agora que seus headers estão soldados ao shield, você pode encaixar o shield 3G-SDI à sua placa Arduino.

Segure cada lateral do shield cuidadosamente, alinhe os pinos header com os headers da sua placa Arduino e insira-os pinos suavemente nos compartimentos do header. Cuidado para não dobrar um dos pinos ao encaixar no shield.
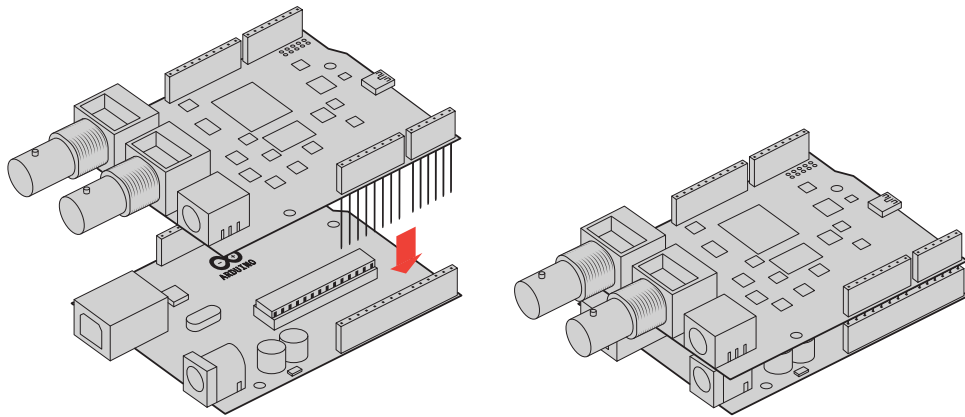
Com todos os pinos plugados, a conexão entre o Blackmagic Shield e a placa Arduino deve ser firme e estável.

Conexão de alimentação

Para alimentar seu Blackmagic 3G-SDI Shield for Arduino, basta conectar um adaptador de alimentação de 12 V à entrada de alimentação de 12 V do shield Blackmagic.
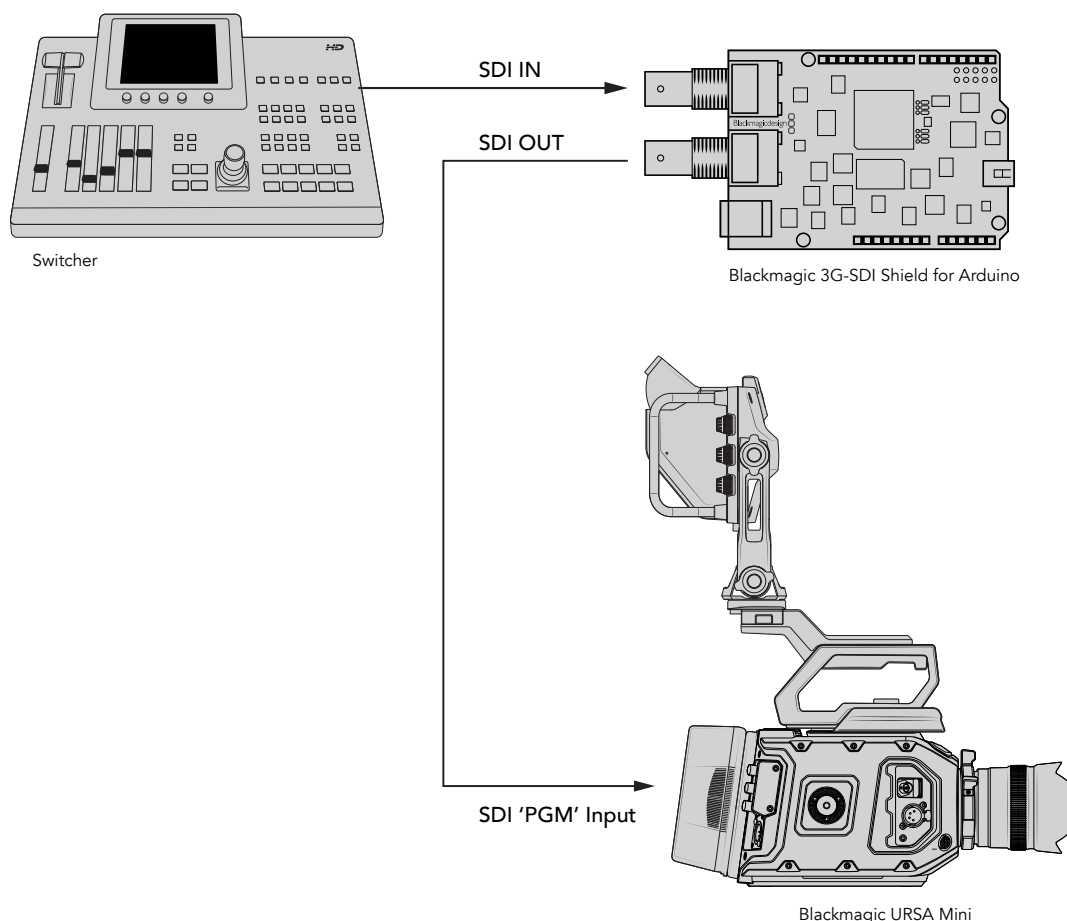
Conexão de equipamentos SDI

Com alimentação fornecida, você pode agora plugar seu 3G-SDI Shield for Arduino aos seus equipamentos SDI. Por exemplo, como se conectar a um switcher e uma Blackmagic URSA Mini:

1   Conecte a saída de programa do switcher à entrada SDI do Blackmagic 3G-SDI Shield.

2   Plugue a saída SDI do Blackmagic 3G-SDI Shield à entrada de programa marcada como PGM na sua Blackmagic URSA Mini.

Um esquema de conexão é fornecido abaixo.

SDI IN

SDI OUT

Switcher

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini

Isso é tudo que você precisa saber para começar!

Agora que o seu shield está encaixado à placa Arduino, alimentado e conectado aos seus equipamentos SDI, você pode instalar o software interno e arquivos da biblioteca, programar o software Arduino e começar a utilizar o shield para controlar seus equipamentos.

Continue a ler o manual para informações sobre como instalar o software interno do shield e onde instalar os arquivos da biblioteca Arduino para a comunicação com seu Arduino.

> **DICA** Você também pode utilizar seu Blackmagic 3G-SDI Shield for Arduino para controlar outros produtos Blackmagic MultiView 16. Por exemplo, quando seu shield estiver conectado à entrada 16, você pode exibir uma borda de sinalização na multivisualização. Ao utilizar câmeras Blackmagic URSA Mini, certifique-se de que a sobreposição de grades esteja habilitada na exibição de sinalização do visor da câmera. Consulte o manual da câmera Blackmagic para mais informações.

Instalação do software

> **OBSERVAÇÃO** Antes de instalar o utilitário Blackmagic Shield for Arduino, baixe o software Arduino IDE mais recente em www.arduino.cc e instale-o no seu computador.
>
> Após instalar o software Arduino, você deve instalar o software interno do Blackmagic 3G-SDI Shield.

Instalação do software interno

O Blackmagic Shield for Arduino Setup é usado para atualizar o software interno do shield. O software interno se comunica com a placa Arduino e a controla usando os arquivos da biblioteca Arduino. Esses arquivos da biblioteca são instalados com o software de instalação e tudo que você precisa fazer é copiar a pasta contendo os arquivos e colar na pasta do aplicativo Arduino. Você pode encontrar informações sobre os arquivos da biblioteca e como instalá-los na próxima seção deste manual.

Recomendamos baixar a versão mais recente do software Blackmagic Shield for Arduino e atualizar seu shield para poder utilizar novos recursos e aprimoramentos. A versão mais recente pode ser baixada na Central de Suporte Técnico da Blackmagic Design em www.blackmagicdesign.com/br/support.

Como instalar o software interno usando Mac OS X:

1   Baixe e descompacte o software Blackmagic Shield for Arduino.

2   Abra a imagem de disco resultante e inicie o instalador Blackmagic Shield for Arduino. Siga as instruções na tela.

3   Após instalar a versão mais recente do instalador Blackmagic Shield for Arduino, ligue o Blackmagic Shield e conecte-o ao seu computador com um cabo USB.

4   Agora, inicie o utilitário de instalação e siga as instruções na tela para atualizar o software interno do seu shield. Se nenhum comando aparecer, o software interno está atualizado e nenhuma outra ação é necessária.

Como instalar o software interno usando Windows:

1   Baixe e descompacte o software Blackmagic Shield for Arduino.

2   Você deverá ver uma pasta Blackmagic Shield for Arduino contendo este manual e o instalador Blackmagic Shield for Arduino. Dê um clique duplo no instalador e siga as orientações na tela para concluir a instalação.

3   Após instalar a versão mais recente do instalador Blackmagic Shield for Arduino, ligue o Blackmagic Shield e conecte-o ao seu computador com um cabo USB.

4   Agora, inicie o utilitário de instalação e siga as instruções na tela para atualizar o software interno do seu shield. Se nenhum comando aparecer, o software interno está atualizado e nenhuma outra ação é necessária.

Instalação dos arquivos de biblioteca Arduino

O código de programação para controlar seu Arduino é conhecido como "sketch" e seu Blackmagic 3G-SDI Shield for Arduino utiliza os arquivos de biblioteca para facilitar a programação dos sketches. Após instalar o software de configuração do shield, os arquivos da biblioteca são instalados em uma pasta chamada "Library". Basta copiar a pasta contendo os arquivos de biblioteca e colar na pasta de bibliotecas Arduino.

> **OBSERVAÇÃO** O software Arduino IDE precisa estar fechado ao instalar as bibliotecas.

Como instalar os arquivos de biblioteca no Mac OS X:

 1 Abra o "Blackmagic Shield for Arduino" na sua pasta de aplicativos.

 2 Abra a pasta "Library" e clique com o botão direito para copiar a pasta denominada: BMDSDIControl.

 3 Vá até a pasta "Documentos" no seu computador e abra a pasta Arduino.

 4 Você deverá ver uma subpasta chamada "Libraries". Cole a pasta BMDSDIControl na pasta "Libraries".

Como instalar os arquivos de biblioteca no Windows:

 1 Abra a pasta Programas/Blackmagic Shield for Arduino.

 2 Você deverá ver uma subpasta chamada "Library". Abra essa pasta e clique com o botão direito para copiar a pasta denominada: BMDSDIControl.

 3 Vá até a pasta "Documentos" no seu computador e abra a pasta Arduino.

 4 Você deverá ver uma subpasta chamada "Libraries". Cole a pasta BMDSDIControl na pasta "Libraries".

Isso é tudo que você precisa fazer para instalar os arquivos da biblioteca Blackmagic Design no seu computador. Ao executar o software Arduino, você também contará com exemplos de sketches Blackmagic Design para selecionar.

Basta ir até o menu suspenso "File" na barra de menu do software Arduino e selecionar "Examples". Agora, selecione BMDSDIControl e você deverá ver uma lista de exemplos de sketches que poderá usar.

Com os arquivos de biblioteca armazenados na pasta correta, seu shield agora poderá utilizá-los para se comunicar com a placa Arduino. Tudo que você precisa fazer é programar o software Arduino IDE. Consulte a seção 'Programação de sketches Arduino' para mais informações.

> **OBSERVAÇÃO** Se um arquivo de biblioteca atualizado for lançado futuramente, será necessário remover a pasta BMDSDIControl anterior e substituí-la pela nova pasta usando o mesmo método listado acima.

Blackmagic Shield for Arduino Setup
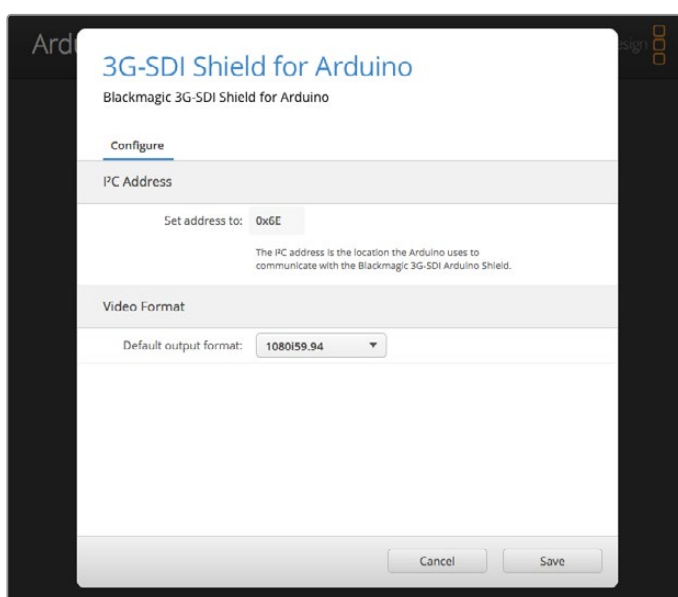
O software Blackmagic Shield for Arduino Setup permite alterar as configurações no seu shield como, o endereço I²C e o formato da saída de vídeo.

Com o Blackmagic Shield for Arduino Setup instalado no seu computador, você pode alterar as configurações do shield, como o "I²C Address", que identifica seu shield para que a placa Arduino possa se comunicar com ele, além de "Video Format", que configura o formato de saída do shield.

I²C Address

Em casos muito raros, existe a possibilidade de que outro shield instalado e encaixado ao seu shield Blackmagic possa compartilhar o mesmo endereço I²C padrão, o que criará um conflito. Se isso ocorrer, é possível alterar a configuração do endereço padrão do seu shield.

O endereço padrão do shield é 0x6E, mas, você pode selecionar entre uma variedade de endereços entre 0x08 e 0x77.

Como alterar o endereço do shield:

1 Inicie o Blackmagic Shield for Arduino Setup e clique no ícone de configurações do seu shield.

2 Na caixa de diálogo "Set address to:", digite o endereço que deseja usar.

3 Clique em "Salvar".

Video Format

O formato de saída padrão é selecionado no utilitário de configuração quando não há uma entrada conectada. Quando uma entrada é detectada, a saída seguirá o mesmo formato de entrada. Se a entrada é removida, a saída reverterá para o formato de saída padrão selecionado no utilitário. Você pode alterar o formato de vídeo ao clicar no menu suspenso "Default Output Format" e selecionar o formato desejado.

É possível selecionar os formatos de saída de vídeo abaixo:

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Programação de sketches Arduino

Os projetos, ou sketches, programados no software Arduino são muito fáceis de criar! Os sketches são programados com a conhecida linguagem de programação "C". Ao programar seus sketches usando comandos do Studio Camera Control Protocol, o shield integra os comandos à saída SDI, o que permite controlar suas câmeras Blackmagic URSA Mini ou Blackmagic Studio.

Todos os comandos suportados estão incluídos na seção Studio Camera Control Protocol deste manual, assim você pode obter os comandos do protocolo e utilizá-los no seu sketch.
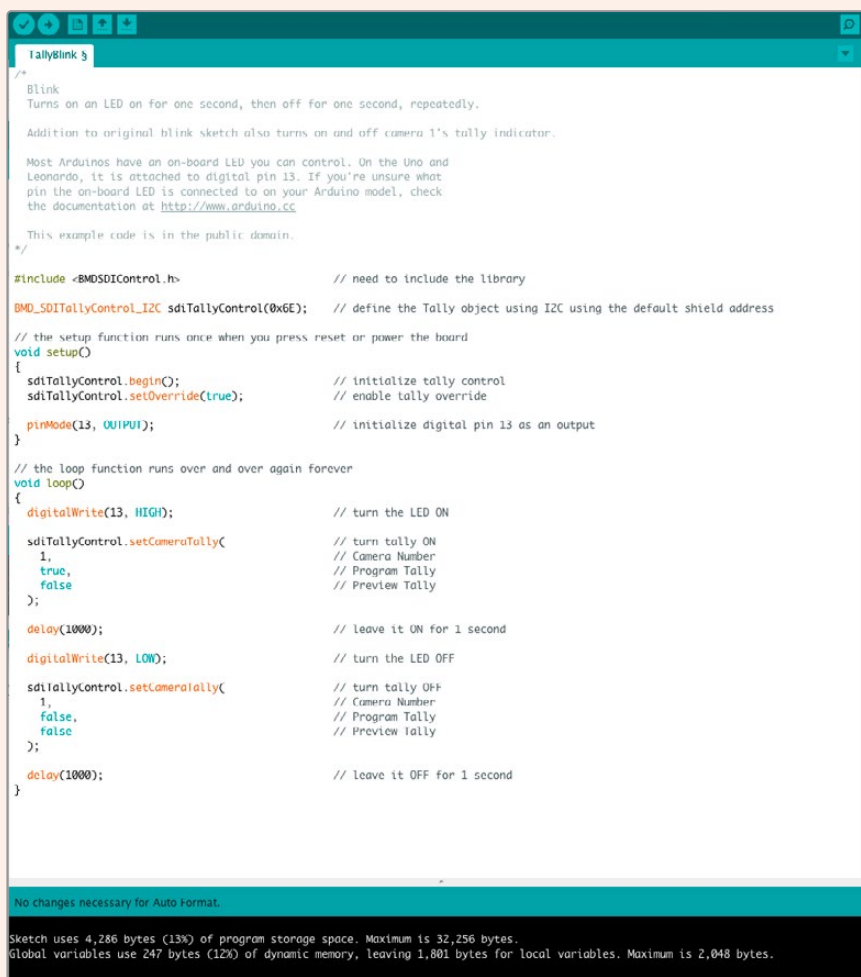
Testes e instalação da biblioteca Blackmagic Shield

Depois que tudo estiver conectado, conforme descrito na seção 'Instruções preliminares' e após ter instalado o software e os arquivos da biblioteca, você deve verificar se o shield está se comunicando com a placa Arduino e que tudo está funcionando como esperado.

Uma maneira rápida é abrir e executar o exemplo de sketch para piscar a sinalização.

Para fazer isso:

1   Inicie o software Arduino IDE.

2   Vá até o menu "Tools" e selecione a placa Arduino e o número da porta.

3   No menu "File", selecione "Examples/BMDSDIControl" e selecione o sketch denominado "TallyBlink".

4   Faça o upload do sketch na sua placa.


```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Addition to original blink sketch also turns on and off camera 1's tally indicator.

  Most Arduinos have an on-board LED you can control. On the Uno and
  Leonardo, it is attached to digital pin 13. If you're unsure what
  pin the on-board LED is connected to on your Arduino model, check
  the documentation at http://www.arduino.cc

  This example code is in the public domain.
*/

#include <BMDSDIControl.h>                  // need to include the library

BMD_SDITallyControl_I2C sdiTallyControl(0x6E);    // define the Tally object using I2C using the default shield address

// the setup function runs once when you press reset or power the board
void setup()
{
  sdiTallyControl.begin();                  // initialize tally control
  sdiTallyControl.setOverride(true);        // enable tally override

  pinMode(13, OUTPUT);                      // initialize digital pin 13 as an output
}

// the loop function runs over and over again forever
void loop()
{
  digitalWrite(13, HIGH);                   // turn the LED ON

  sdiTallyControl.setCameraTally(           // turn tally ON
    1,                                      // Camera Number
    true,                                   // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it ON for 1 second

  digitalWrite(13, LOW);                    // turn the LED OFF

  sdiTallyControl.setCameraTally(           // turn tally OFF
    1,                                      // Camera Number
    false,                                  // Program Tally
    false                                   // Preview Tally
  );

  delay(1000);                              // leave it OFF for 1 second
}
```

```
No changes necessary for Auto Format.
```

```
Sketch uses 4,286 bytes (13%) of program storage space. Maximum is 32,256 bytes.
Global variables use 247 bytes (12%) of dynamic memory, leaving 1,801 bytes for local variables. Maximum is 2,048 bytes.
```

O exemplo de sketch para piscar a sinalização é uma maneira rápida de testar seu Blackmagic 3G-SDI Shield for Arduino. Dados brutos podem ser enviados ao shield via I²C usando comandos do documento Studio Camera Protocol, mas também fornecemos bibliotecas personalizadas para facilitar a programação dos sketches.

Você deverá ver a luz de sinalização piscar na sua Blackmagic Studio Camera uma vez a cada segundo. Caso veja a luz de sinalização piscando, você pode confirmar que o shield Blackmagic está se comunicando com a placa Arduino e tudo está funcionando perfeitamente.

Se a luz de sinalização não estiver piscando, verifique se o número de sinalização da câmera Blackmagic está definido com 1 e se a sobreposição de grades da câmera está habilitada.

Caso mais assistência seja necessária, visite a central de suporte técnico da Blackmagic Design em www.blackmagicdesign.com/br/support. Consulte a seção de ajuda neste manual para mais informações sobre as diferentes maneiras de obter assistência ao configurar seu shield.

Indicadores LED

Seu Blackmagic 3G-SDI Shield for Arduino possui seis indicadores LED que confirmam a atividade do shield, como alimentação, UART e a comunicação de I²C e SPI, além de indicadores que exibem quando substituições de sinalização e controle de câmera estão ativadas.
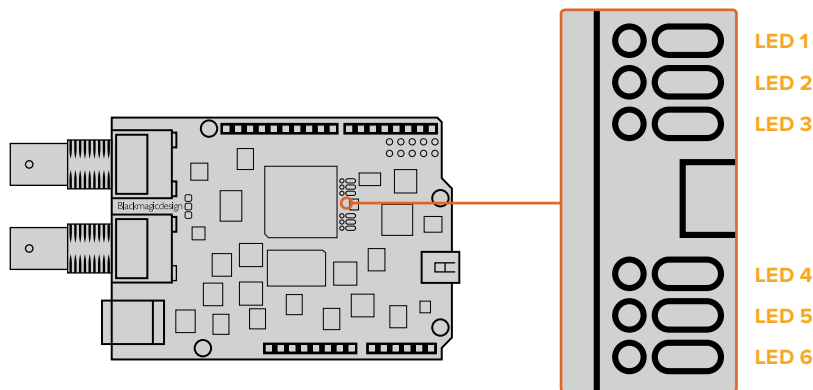
LED 1 - Sistema Ativo
Acende quando a alimentação está conectada ao shield.

LED 2 - Substituições de Controle Ativadas
Acende se você tiver habilitado o controle de câmera no seu sketch Arduino.

LED 3 - Substituições de Sinalização Ativadas
Acende se você tiver habilitado a sinalização no seu sketch Arduino.

LED 5 - Analisador I²C Ocupado
Acende quando comunicação é detectada entre seu shield e o Arduino usando o protocolo I²C.

LED 6 - Analisador Serial Ocupado
Acende quando comunicação UART é detectada.

Quando seu Blackmagic shield está inicializando, o indicador de alimentação permanecerá apagado e os LEDs 3, 4 e 5 indicarão a atividade abaixo.

LED 3 - Imagem do aplicativo carregando

LED 4 - EEPROM inicializando

LED 5 - Verificação de memória em andamento

Após uma inicialização bem-sucedida, o LED de alimentação acende e todos os LEDs retomam as suas funções de operação padrão.

Na rara eventualidade de uma falha de inicialização, todos os LEDs, com exceção da atividade que falhou, irão piscar rapidamente para que você possa identificar a causa da falha.

Encaixe de componentes do shield

Caso queira criar seu próprio controlador hardware, é possível montar um novo shield com botões, potenciômetros e um joystick para controle mais tátil e prático. Basta encaixar o shield personalizado ao seu Blackmagic 3G-SDI Shield for Arduino ao plugá-lo nos compartimentos header do seu shield. Não há limite para os tipos de controladores que você pode criar. É possível inclusive substituir o circuito de uma CCU antiga com sua solução Arduino personalizada para uma unidade de controle de câmera padrão da indústria.
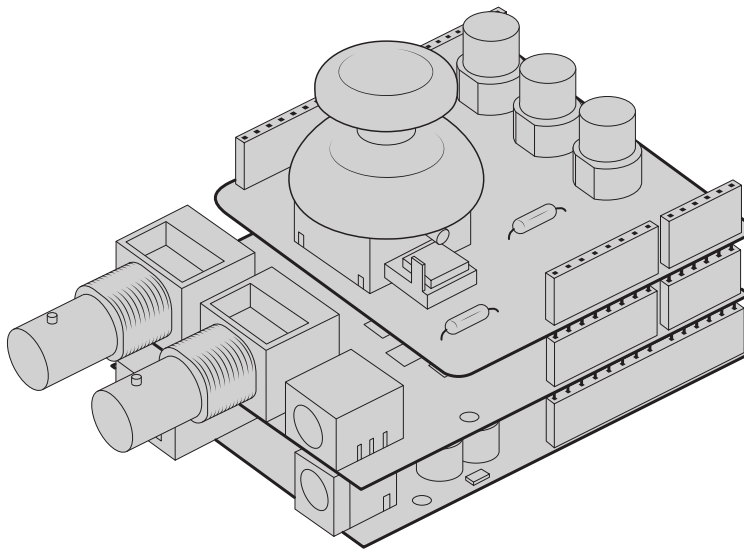
Você pode criar o seu próprio controlador em hardware e conectá-lo ao seu Blackmagic 3G-SDI Shield for Arduino para controle mais interativo e aperfeiçoado.

Comunicação com o Blackmagic Shield for Arduino

Você pode se comunicar com o Blackmagic 3G-SDI Shield for Arduino via I2C ou Serial. Recomendamos I2C devido a quantidade reduzida de pinos, além de liberar o Serial Monitor. Isso também permite usar mais dispositivos I2C com o shield.

Visão geral avançada

A biblioteca oferece dois objetos de núcleo, BMD_SDITallyControl e BMD_SDICameraControl, que podem ser usados para a interagir com as funcionalidades de sinalização e controle de câmera do shield. Um ou ambos os objetos podem ser criados no seu sketch para emitir comandos de controle de câmera ou ler e programar dados de sinalização respectivamente. Esses objetos existem em variações diferentes, uma para cada um dos barramentos de comunicação I2C ou Serial suportados pelo shield.

Interface I²C

Para usar a I²C interface com o shield:

```
// NOTE: Must match address set in the setup utility software
const intshieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);
```

Interface Serial

Para usar a interface Serial com o shield:

```
BMD_SDICameraControl_Serial sdiCameraControl;
BMD_SDITallyControl_Serial  sdiTallyControl;
```

Observe que a biblioteca irá configurar a interface serial Arduino na taxa baud 38400 necessária. Caso queira imprimir mensagens de depuração para o Serial Monitor ao utilizar essa interface, altere a taxa baud do Serial Monitor para corresponder. Se o Serial Monitor é utilizado, alguns dados binários estarão visíveis, já que o IDE não será capaz de diferenciar as mensagens de usuário dos comandos do shield.

Exemplo de uso

Depois de criados em um sketch, esses objetos permitirão que você emita comando ao shield através do barramento selecionado ao chamar funções no(s) objeto(s) criado(s). Um sketch menor que usa a biblioteca via o barramento I²C é demonstrado abaixo.

```
// NOTE: Must match address set in the setup utility software
const intshieldAddress = 0x6E;
BMD_SDICameraControl_I2C sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C  sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

A lista das funções que podem ser chamadas nos objetos criados está incluída mais adiante neste documento. Observe que antes de usar, você deve chamar a função "begin" em cada objeto antes de emitir outros comandos.

Alguns exemplos de sketch demonstrados nessa biblioteca estão incluídos no menu File > Examples > BMDSDIControl do menu do Arduino IDE.

Studio Camera Control Protocol (em inglês)

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

1 = camera number 1
8 = Color Correction group
0 = Lift Adjust
0 = assign value
liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol (em inglês)

Versão 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5 UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Lens	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
Video	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0]=file-M-rate
					–	–	[1]=sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum. Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDdddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, dddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header		command				data									
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Informações para Desenvolvedores (em inglês)

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin  | Function
--------------------|----------------------------
 A4          | Serial Data (SDA)
 A5          | Serial Clock (SCL)
```

I²C Protocol (Writes):

        (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

        (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin  | Function
--------------------|----------------------------
 IO1         | Serial Transmit (TX)
 IO0          | Serial Receive (RX)
```

UART Protocol (Writes):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address        | Name    | R/W | Register Description
--------------------|----------|-----|-----------------------------
0x0000 - 0x0003 | IDENTITY | R   | Hardware Identifier
0x0004 - 0x0005 | HWVERSION| R   | Hardware Version
0x0006 - 0x0007 | FWVERSION| R   | Firmware Version
        |        |   |
0x1000        | CONTROL  | R/W | System Control
        |        |   |
0x2000        | OCARM    | R/W | SDI Control Override Arm
0x2001        | OCLENGTH | R/W | SDI Control Override Length
0x2100 - 0x21FE | OCDATA   | R/W | SDI Control Override Data
        |        |   |
0x3000        | ICARM    | R/W | SDI Control Incoming Arm
0x3001        | ICLENGTH | R   | SDI Control Incoming Length
0x3100 - 0x31FE | ICDATA   | R   | SDI Control Incoming Data
        |        |   |
0x4000        | OTARM    | R/W | SDI Tally Override Arm
0x4001        | OTLENGTH | R/W | SDI Tally Override Length
0x4100 - 0x41FE | OTDATA   | R/W | SDI Tally Override Data
        |        |   |
0x5000        | ITARM    | R/W | SDI Tally Incoming Arm
0x5001        | ITLENGTH | R   | SDI Tally Incoming Length
0x5100 - 0x51FE | ITDATA   | R   | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

[IDENTITY]
31   0

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

[VERSION MAJOR] [VERSION MINOR]
15   8 7   0

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

[RESERVED] [OVERRIDE OUTPUT] [RESET TALLY] [OVERRIDE TALLY] [OVERIDE CONTROL]
7   4   3   2   1   0

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]

255*8-1   0

Control Data: Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]

7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]

7   0

Length: Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]

255*8-1   0

Tally Data: Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]

7   1   0

Reserved: Always zero.

Arm: When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]

7   0

Length: Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]

255*8-1   0

Tally Data: Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Ajuda

Obtendo ajuda

Seu Blackmagic 3G-SDI Shield for Arduino é uma ferramenta para desenvolvedores criada para ajudar a desenvolver independemente com base nas suas necessidades personalizadas.

Para informações mais atualizadas sobre seu shield, visite as páginas de suporte técnico da Blackmagic Design e consulte os materiais de suporte mais recentes.

Central de suporte técnico online da Blackmagic Design

O manual, o programa e as notas de suporte mais recentes podem ser encontrados na Central de Suporte Técnico da Blackmagic Design em www.blackmagicdesign.com/br/support.

Fórum de desenvolvimento Arduino

Caso tenha dúvidas sobre programação, você pode obter ajuda nos fóruns online para desenvolvedores Arduino na internet. Há uma grande comunidade de desenvolvedores Arduino e muitos fóruns de excelente qualidade onde é possível tirar dúvidas sobre o software ou até encontrar um engenheiro disposto a implementar uma solução para você!

Fórum Blackmagic Design

O fórum da Blackmagic Design no nosso site é um recurso útil que você pode acessar para obter mais informações e ideias criativas. Também pode ser uma maneira mais rápida de obter ajuda, pois já podem existir respostas de outros usuários experientes e da equipe da Blackmagic Design, o que o ajudará a seguir em frente. Você pode visitar o fórum em https://forum.blackmagicdesign.com.

Verificação da versão de software atualmente instalada

Para verificar qual versão do software Blackmagic 3G-SDI Shield for Arduino está instalada no seu computador, abra a janela "About" no Blackmagic 3G-SDI Shield for Arduino Setup.

- No Mac OS X, abra o Blackmagic 3G-SDI Shield for Arduino Setup na pasta de Aplicativos. Selecione "Sobre Blackmagic Shield for Arduino Setup" no menu de aplicativos para revelar o número da versão.
- No Windows 7, abra o Blackmagic 3G-SDI Shield for Arduino Setup a partir do menu "Iniciar". Clique no menu "Ajuda" e selecione Blackmagic 3G-SDI Shield for Arduino Setup para revelar o número da versão.
- No Windows 8, abra o Blackmagic 3G-SDI Shield for Arduino Setup a partir do ladrilho Blackmagic 3G-SDI Shield for Arduino Setup na sua página "Iniciar". Clique no menu "Ajuda" e selecione Blackmagic 3G-SDI Shield for Arduino Setup para revelar o número da versão.

Como obter as atualizações de software mais recentes

Após verificar a versão do software Blackmagic 3G-SDI Shield for Arduino Setup instalado no seu computador, consulte a Central de Suporte Técnico da Blackmagic Design em www.blackmagicdesign.com/br/support para conferir as últimas atualizações. Embora seja uma boa ideia instalar as últimas atualizações, é recomendável evitar atualizar qualquer programa caso esteja no meio de um projeto importante.
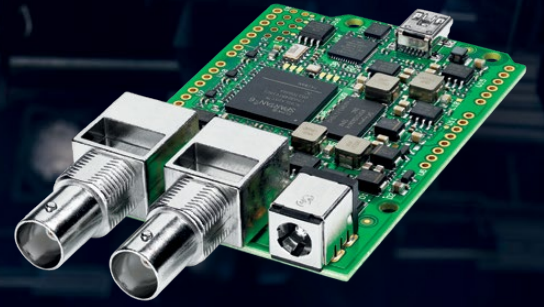
Garantia

12 Meses de Garantia Limitada

A Blackmagic Design garante que este produto estará isento de quaisquer defeitos de materiais e fabricação por um período de 12 meses a partir da data de compra. Se um produto revelar-se defeituoso durante este período de garantia, a Blackmagic Design, a seu critério, consertará o produto defeituoso sem cobrança pelos componentes e mão-de-obra, ou fornecerá a substituição em troca pelo produto defeituoso.

Para obter o serviço sob esta garantia você, o Consumidor, deve notificar a Blackmagic Design do defeito antes da expiração do período de garantia e tomar as providências necessárias para o desempenho do serviço. O Consumidor é responsável pelo empacotamento e envio do produto defeituoso para um centro de assistência designado pela Blackmagic Design com os custos de envio pré-pagos. O Consumidor é responsável pelo pagamento de todos os custos de envio, seguro, taxas, impostos e quaisquer outros custos para os produtos retornados para nós por qualquer razão.

Esta garantia não se aplica a defeitos, falhas ou danos causados por uso inadequado ou manutenção e cuidado inadequado ou impróprio. A Blackmagic Design não é obrigada a fornecer serviços sob esta garantia: a) para consertar danos causados por tentativas de instalar, consertar ou fornecer assistência técnica ao produto por pessoas que não sejam representantes da Blackmagic Design, b) para consertar danos causados por uso ou conexão imprópria a equipamentos não compatíveis, c) para consertar danos ou falhas causadas pelo uso de componentes ou materiais que não são da Blackmagic Design, d) para fornecer assistência técnica de um produto que foi modificado ou integrado a outros produtos quando o efeito de tal modificação ou integração aumenta o tempo ou a dificuldade da assistência técnica do serviço. ESTA GARANTIA É FORNECIDA PELA BLACKMAGIC DESIGN NO LUGAR DE QUAISQUER OUTRAS GARANTIAS, EXPLÍCITAS OU IMPLÍCITAS. A BLACKMAGIC DESIGN E SEUS FORNECEDORES NEGAM QUAISQUER GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO OU ADEQUAÇÃO A UMA FINALIDADE ESPECÍFICA. A RESPONSABILIDADE DA BLACKMAGIC DESIGN DE CONSERTAR OU SUBSTITUIR PRODUTOS DEFEITUOSOS É A ÚNICA E EXCLUSIVA MEDIDA FORNECIDA AO CONSUMIDOR PARA QUAISQUER DANOS INDIRETOS, ESPECIAIS OU ACIDENTAIS INDEPENDENTEMENTE DA BLACKMAGIC DESIGN OU DO FORNECEDOR TIVER INFORMAÇÃO PRÉVIA SOBRE A POSSIBILIDADE DE TAIS DANOS. A BLACKMAGIC DESIGN NÃO É RESPONSÁVEL POR QUAISQUER USOS ILEGAIS DO EQUIPAMENTO PELO CONSUMIDOR. A BLACKMAGIC NÃO É RESPONSÁVEL POR QUAISQUER DANOS CAUSADOS PELO USO DESTE PRODUTO. O USUÁRIO DEVE OPERAR ESTE PRODUTO POR CONTA E RISCO PRÓPRIOS.

Kurulum ve Kullanım Kılavuzu

Blackmagic 3G-SDI Shield for Arduino

Şubat 2020

Türkçe

Hoş Geldiniz

Yeni Blackmagic 3G-SDI Shield for Arduino'nuzu satın aldığınız için teşekkür ederiz.

Yeni teknolojilerle daima ilgileniyoruz ve SDI ürünlerimizin her türlü yaratıcı şekilde kullanımı, bizi heyecanlandırıyor. 3G-SDI Shield for Arduino'nuz ile, Blackmagic Design ekipmanlarınızla daha fazla kontrol seçeneğine sahip olmak için, SDI iş akışınıza artık Arduino'yu entegre edebilirsiniz.

Örneğin: ATEM switcher'ler, SDI sinyallerine gömülü veri paketleri üzerinden, Blackmagic URSA Mini ve Blackmagic Studio kameralarını kontrol edebilirler. Bir ATEM switcher olmadan da Blackmagic kameralarınızı kontrol edebilmeyi istiyorsanız, 3G-SDI Shield for Arduino ile size özel kontrol çözümleri oluşturabilirsiniz. Bu shield, üzerine inşa etmeniz için gerekli SDI platformunu size sağlar; böylelikle switcher'inizden gelen program dönüş sinyalini, shield'den geçirerek Blackmagic kameralarınızın program girişine bağlayabilirsiniz.

Kameraya komutları göndermek için kodun yazılması kolaydır ve desteklenen komutların hepsi, bu kılavuza dahil edilmiştir.

Bir bilgisayar ile kameraları kontrol edebilir veya shield'e butonlar, düğmeler ve kumanda kolları ekleyebilir ve mercek netliği ve optik yaklaştırma, diyafram ayarları, siyah ve beyaz ayar kontrolü, kamerada yerleşik güçlü renk düzeltici ve bunlar gibi nice özelliği ayarlamak için dinamik donanım kontrolörleri oluşturabilirsiniz. Kendinize özel bir kontrolör oluşturmak, yapım için faydalı olmanın yanında, aynı zamanda çok eğlencelidir!

Bu teknoloji, bizi çok heyecanlandırdı ve 3G-SDI Shield for Arduino için oluşturduğunuz herhangi bir SDI kontrolör hakkındaki deneyimlerinizi duymayı çok isteriz!

Bu kurulum ve kullanım kılavuzu, Blackmagic 3G-SDI Shield for Arduino'nuzu kullanmaya başlamanız için ihtiyacınız olan tüm bilgileri içerir. Bu kullanım kılavuzunun en güncel versiyonunu ve shield'inizin dahili yazılımı için güncellemeleri indirmek üzere, lütfen internet sitemizdeki destek sayfamıza www.blackmagicdesign.com/tr adresinden ulaşarak kontrol edin. Yazılımınızın güncel olması, en son özelliklere sahip olmanızı garantiler. Yeni yazılımlar piyasaya sürüldüğünde size duyurabilmemiz için, bilgisayarınıza yazılımı indirirken, bilgilerinizi sitemize kaydetmenizi rica ediyoruz. Sürekli yeni özellikler ve geliştirmeler için çaba içinde olduğumuzdan, yorumlarınızı almaktan mutluluk duyarız.

Grant Petty

Grant Petty
Blackmagic Design CEO

İçindekiler

# Blackmagic 3G-SDI Shield for Arduino

Başlarken

Başlıkların Bağlanması ve Lehimlenmesi

Blackmagic 3G-SDI Shield for Arduino'nuz; 2 adet 8 pimli başlık, bir adet 10 pimli ve bir adet 6 pimli başlık olmak üzere 4 adet istiflenebilir başlık ile gelir. Başlıklar; shield'inizi Arduino karta monte etmek üzere kullanılan köprüleme konektörlerdir ve istiflenebilir oldukları için; kontrol butonları, düğmeler ve kumanda kolu gibi ilave parçaları olan, başka shield'ler de bağlayabilirsiniz. Başlık düzeni, Arduino UNO gibi, bir R3 taban alanı olan Arduino kartların montajını destekler.

Shield'inize başlıklar bağlamak için:

1 Her başlığın pimlerini, Blackmagic 3G-SDI Shield'inizin her iki tarafındaki ilgili pim deliklerine yerleştirin. Başlık düzen sıralaması için aşağıdaki şemaya bakın.
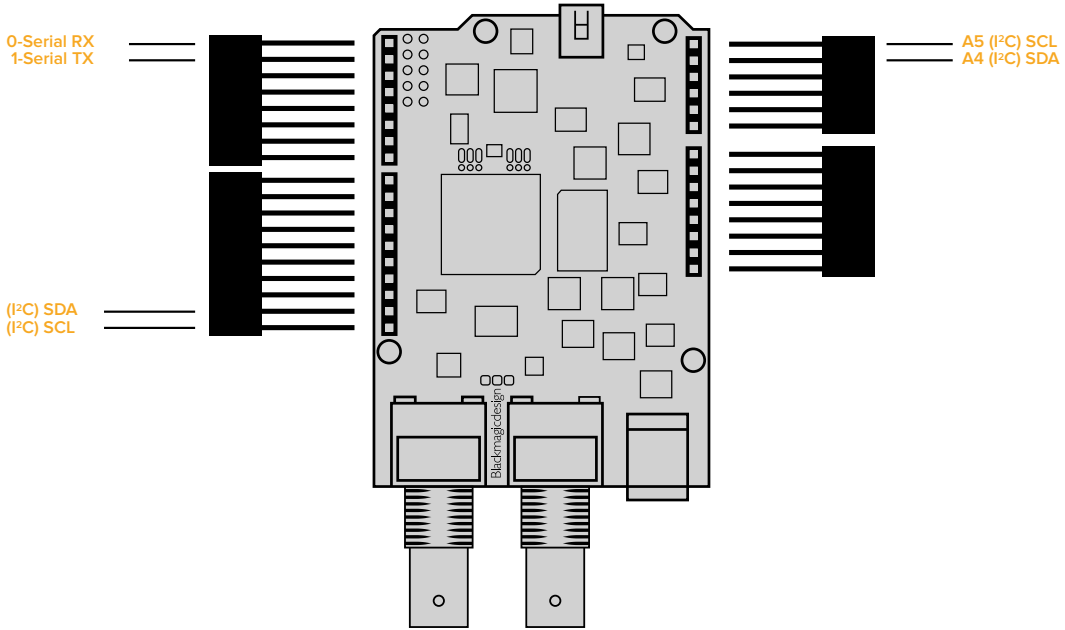
> **NOT** Shield'e bağlanırken iletişim I²C veya Serial (seri) üzerindendir. Seri mönitörün kullanılmasını sağladığı ve diğer tüm pimleri hazır bulundurduğu için I²C'yi tavsiye ederiz. Şemada BMDSDIControl nesnesini tanımlarken, iletişim modunu seçin. Daha fazla bilgi için, 'Blackmagic 3G-SDI Arduino Shield ile İletişim' bölümüne bakınız.

2 Her bir başlık piminin alt kısmını, shield'inizin alt tarafına lehimleyin. Her pimdeki lehimin pim deliği ile sağlam bir birleşme noktası oluşturduğundan emin olun, fakat lehimi yakındaki pimlere değdirmeyin.

Arduino Karta Monte Etme

Başlıklar shield'e artık lehimli olduğuna göre, 3G-SDI shield'i Arduino kartınıza monte edebilirsiniz.

Shield'i her iki kenarından dikkatlice tutarak, başlık pimlerini Arduino kartın başlıkları ile hizaya getirin ve pimleri başlık yuvalarına yavaşça itin. Monte işlemi süresince, pimlerin bükülmemesi için dikkatli olun.
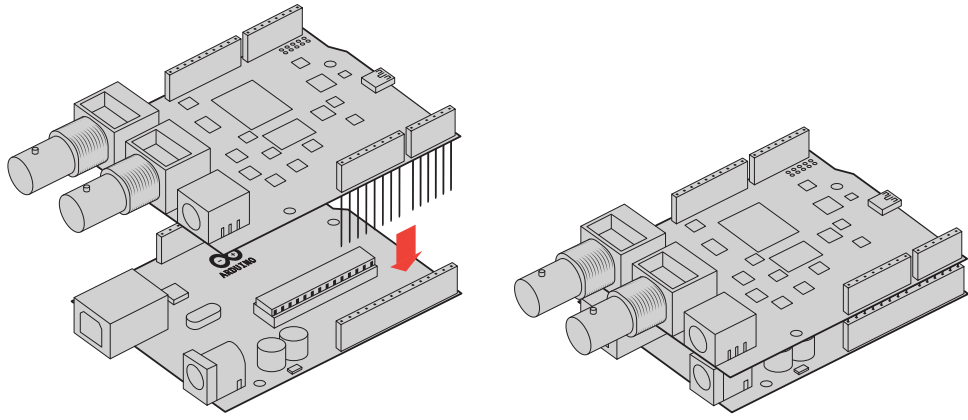
Tüm pimler takılı olarak, Blackmagic shield ve Arduino kart arasındaki bağlantının sıkı ve sağlam olması gerekir.

Güç Kaynağının Takılması

Blackmagic 3G-SDI Shield for Arduino'nuzu çalıştırmak için, Blackmagic shield'de bulunan 12V güç girişine, 12V'luk bir güç adaptörü takmanız yeterlidir.
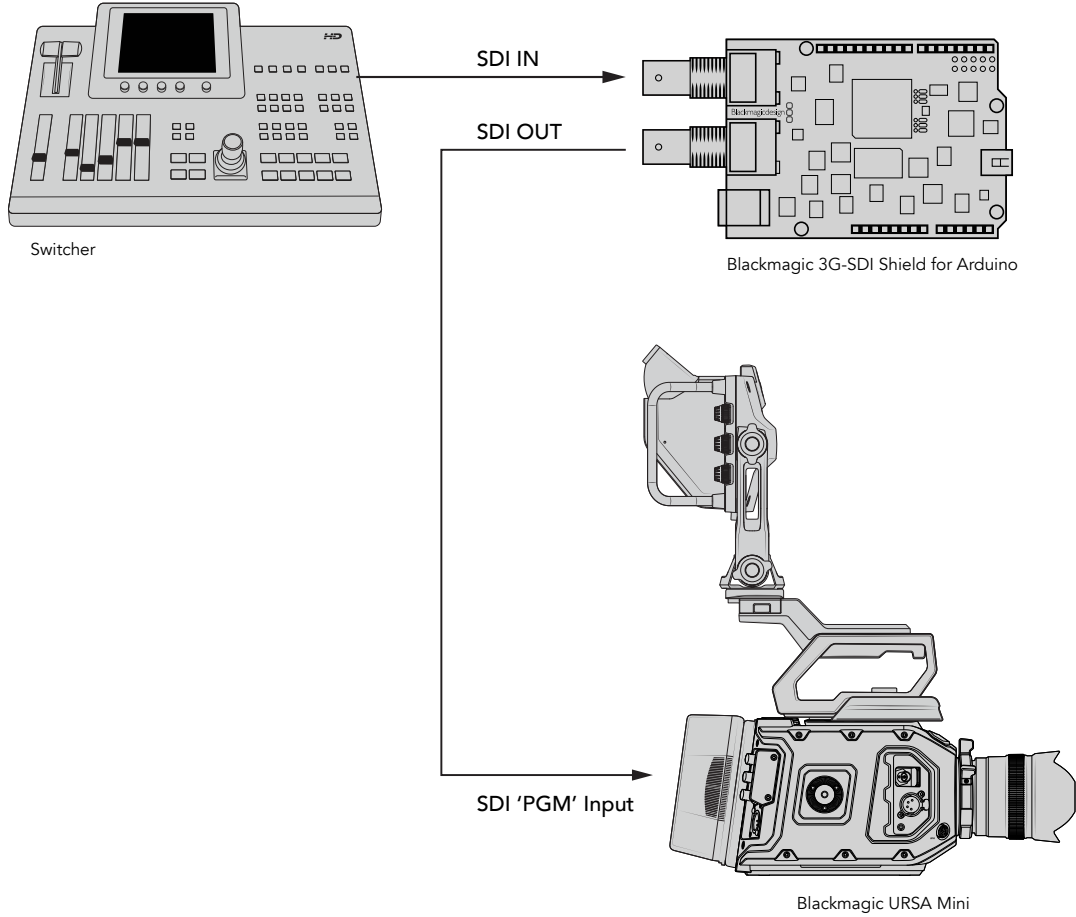
SDI Cihazlara Bağlanma

Güç sağlandığına göre, artık Blackmagic 3G-SDI Shield'inizi SDI cihazlarınıza bağlayabilirsiniz. Örneğin; bir switcher'e ve bir Blackmagic URSA Mini'ye takmak için:

1 Switcher'inizden program çıkışını, Blackmagic 3G-SDI Shield'in SDI girişine takın.

2 Blackmagic 3G-SDI Shield'inizin SDI çıkışını, Blackmagic URSA Mini'nizde PGM olarak işaretli 'program' SDI girişine takın.

Aşağıda bir bağlantı şeması sunulmuştur.

Switcher

SDI IN

SDI OUT

Blackmagic 3G-SDI Shield for Arduino

SDI 'PGM' Input

Blackmagic URSA Mini

Kullanmaya başlamanız için yapmanız gerekenler bu kadar!

Shield'iniz artık Arduino karta monte edildiğine, güç sağlandığına ve SDI cihazınıza bağlandığına göre, dahili yazılımı ve kütüphane dosyalarını yükleyebilir, Arduino yazılımını programlayabilir ve ekipmanınızı kontrol etmek için shiled'i kullanmaya başlayabilirsiniz.

Shield için dahili yazılımı nasıl yükleyeceğinize ve shield'in Arduino ile iletişimde olabilmesi için Arduino kütüphane dosyalarını nereye yükleyeceğinize dair bilgi için, bu kullanım kılavuzunu okumaya devam edin.

BİLGİ Ayrıca, Blackmagic MultiView 16 gibi başka Blackmagic Design ürünlerini kontrol edebilmesi için de Blackmagic 3G-SDI Shield for Arduino'nuzu kullanabilirsiniz. Örneğin; shield'iniz giriş 16'ya bağlı olduğunda, çoklu görüntü ekranında bir tally kenarlığı görüntüleyebilirsiniz. Blackmagic URSA Mini kameralarını kullanırken, kamera bakacında tally göstergesini aktif hale getirmek için, ızgara katmanının açık olduğundan emin olun. Daha fazla bilgi için, Blackmagic kameranızın kullanım kılavuzuna bakın.

Yazılımın Yüklenmesi

NOT Blackmagic Shield for Arduino kurulum yardımcı yazılımını yüklemeden önce, www.arduino.cc sitesinden en son Arduino IDE (entegre geliştirme ortamı) yazılımını indirin ve bilgisayarınıza yükleyin.

Arduino yazılımını yükledikten sonra, Blackmagic 3G-SDI Shield'in dahili yazılımını yükleyebilirsiniz.

Dahili Yazılımın Yüklenmesi

Blackmagic Shield for Arduino Setup uygulaması, shield'inizin dahili yazılımını güncellemek için kullanılır. Dahili yazılım, Arduino kartıyla iletişimde olur ve Arduino kütüphane dosyalarını kullanarak kartı kontrol eder. Kütüphane dosyaları kurulum yazılımıyla yüklenir ve tek yapmanız gereken; dosyaları içeren klasörü kopyalamak ve Arduino uygulama dosyasına yapıştırmaktır. Kütüphane dosyaları ve onları nasıl yükleyeceğiniz konusunda daha fazla bilgiyi, kılavuzun bir sonraki bölümünde bulabilirsiniz.

Yeni özelliklerden ve geliştirmelerden yararlanmanız için, en son Blackmagic Shield for Arduino yazılımını indirmenizi ve shield'inizi güncellemenizi öneririz. En son sürüm, www.blackmagicdesign.com/tr/support adresindeki Blackmagic Design destek merkezinden indirilebilir.

Mac OS X ile dahili yazılımı yüklemek için:

1   Blackmagic Shield for Arduino yazılımını indirin ve zip dosyasını açın.

2   Karşınıza çıkan disk görüntüsünü açın ve Blackmagic Shield for Arduino yükleyicisini başlatın. Ekrandaki talimatları takip edin.

3   En son Blackmagic Shield for Arduino yükleyicisini yükledikten sonra, Blackmagic shield'inizi çalıştırın ve bir USB kablosu ile bilgisayarınıza bağlayın.

4   Şimdi kurulum yardımcı yazılımını başlatın ve shield'inizin dahili yazılımını güncellemek için ekrandaki komutları takip edin. Herhangi bir komut belirmezse bu, dahili yazılımın güncel olduğuna işarettir ve yapmanız gereken başka bir şey yoktur.

Windows ile dahili yazılımı yüklemek için:

1   Blackmagic Shield for Arduino yazılımını indirin ve zip dosyasını açın.

2   Bu kılavuzu ve Blackmagic Shield for Arduino yükleyicisini içeren bir Blackmagic Shield for Arduino dosyası göreceksiniz. Yükleyiciyi çift tıklayın ve kurulumu tamamlamak için ekrandaki komutları takip edin.

3   Blackmagic Shield for Arduino yükleyicisinin en son sürümünü yükledikten sonra, Blackmagic shield'inizi çalıştırın ve bir USB kablosu ile bilgisayarınıza bağlayın.

4   Şimdi kurulum yardımcı yazılımını başlatın ve shield'inizin dahili yazılımını güncellemek için ekrandaki komutları takip edin. Herhangi bir komut belirmezse bu, dahili yazılımın güncel olduğuna işarettir ve yapmanız gereken başka bir şey yoktur.

Arduino Kütüphane Dosyalarının Yüklenmesi

Arduino'nuzu kontrol etmek için yazılan programlara 'sketch' ismi verilir ve Blackmagic 3G-SDI Shield for Arduino'nuz, 'sketch' yazmayı daha kolaylaştıran Arduino kütüphane dosyalarını kullanır. Shield'inizin kurulum yazılımını yükledikten sonra, kütüphane dosyaları 'Library' isminde bir klasöre yüklenir. Şimdi tek yapmanız gereken; kütüphane dosyalarını içeren klasörü kopyalamak ve Arduino 'libraries' klasörüne yapıştırmaktır.

> **NOT** Kütüphane dosyaları yüklenirken Arduino IDE yazılımının kapalı olması gerekir.

Mac OS X bilgisayara kütüphane dosyalarını yüklemek için:

1 Uygulamalar dosyanızdan 'Blackmagic Shield for Arduino'yu açın.

2 'Library' isimli klasörü açın ve şu dosyayı çift tıklayın: BMDSDIControl.

3 Şimdi, bilgisayarınızın 'belgeler' klasörüne gidin ve Arduino klasörünü açın.

4 'libraries' isminde bir alt klasör göreceksiniz. BMDSDIControl klasörünü, 'libraries' klasörü içine yapıştırın.

Windows bilgisayara kütüphane dosyalarını yüklemek için:

1 Programlar/ Blackmagic Shield for Arduino klasörünü açın.

2 'Library' isminde bir alt klasör göreceksiniz. Bu klasörü açın ve şu dosyayı çift tıklayın: BMDSDIControl.

3 Şimdi, bilgisayarınızın 'belgeler' klasörüne gidin ve Arduino klasörünü açın.

4 'Libraries' isminde bir alt klasör göreceksiniz. BMDSDIControl klasörünü, 'libraries' klasörü içine yapıştırın.

Blackmagic Design kütüphane dosyalarını bilgisayarınıza yüklemek için yapmanız gerekenler bu kadar. Arduino yazılımını çalıştırırken, artık içinden seçebileceğiniz Blackmagic Design örnek sketch'ler de olacak.

Sadece Arduino yazılımında, aşağı açılır 'dosya' (file) menüsüne gidin ve örneklerin olduğu 'examples' seçeneğini tıklayın. Şimdi, BMDSDIControl seçeneğini seçin ve kullanabileceğiniz örnek sketch'lerin bir listesini göreceksiniz.

Kütüphane dosyaları doğru klasörde kayıtlı olarak, Arduino kartıyla iletişim için, shield'iniz artık bunları kullanabilir. Yapmanız gereken tek şey Arduino IDE yazılımını programlamaktır. Daha fazla bilgi için, 'Arduino Sketch'lerinin Programlanması' bölümüne bakın.

> **NOT** Önümüzdeki günlerde örnekler içeren bir güncellenmiş kütüphane dosyası yayınlanırsa eski BMDSDIControl klasörünü silmeniz ve yukarıda açıklanan yöntemi kullanarak, yerine yeni klasörü eklemeniz gerekecek.

Blackmagic 3G-SDI Shield
for Arduino Kurulumu
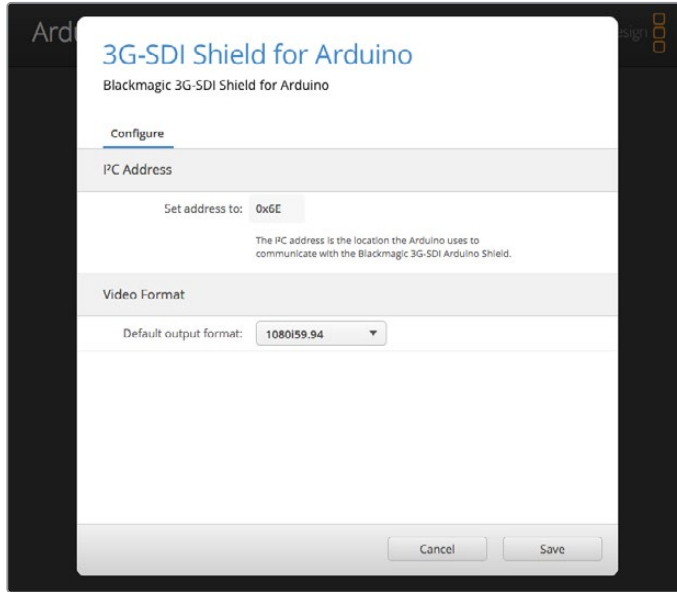
Blackmagic Shield for Arduino Setup yazılımı; I²C adresi ve video çıkış
formatı gibi ayarları, shield'inizde değiştirmenize imkan tanır.

Bilgisayarınıza Blackmagic Shield for Arduino Setup yüklendiğine göre, artık shield'iniz için ayarları
değiştirebilirsiniz. Değiştirebileceğiniz ayarların arasında; Arduino kartın shield ile iletişimini sağlayan
'I²C adresi' ve shield'iniz için çıkış formatını ayarlayan 'video formatı' bulunur.

I²C Adresi

Bazı nadir durumlarda, Blackmagic shield'inize monte edilmiş başka bir shield'in, varsayılan
adres olarak aynı I²C adresini paylaşma olasılığı vardır ve bu da bir anlaşmazlığa neden olabilir.
Bu meydana gelirse shield'inizin varsayılan adres ayarını değiştirebilirsiniz.

Shield'iniz için varsayılan adres 0x6E'dir, fakat 0x08 ve 0x77 arasında değişen çeşitli adreslerden birisini seçebilirsiniz.

Shield'iniz için adresi değiştirmek üzere:

1   Blackmagic Shield for Arduino Setup yazılımını başlatın ve shield'inizin 'ayarlar' ikonunu tıklayın.

2   'Set address to:' isimli adres değiştirme kutusuna kullanmayı istediğiniz adresi girin.

3   'Save' (kaydet) ibaresini tıklayın.

Video Formatı

Bir girişin bağlı olmadığı durumlar için kurulum yardımcı yazılımında varsayılan çıkış formatı seçilmiştir. Bir giriş tespit edildiğinde, çıkış, girişle aynı formatı takip eder. Bu giriş çıkarıldığında, çıkış, yardımcı yazılımda seçilen varsayılan çıkış formatına geri döner. Aşağı açılır menüden 'default output format' (varsayılan çıkış formatı) ibaresini tıklayıp, istediğiniz formatı seçerek video formatını değiştirebilirsiniz.

Aşağıdaki video çıkış formatları arasından seçiminizi yapabilirsiniz.

- 720p50
- 720p59.94
- 720p60
- 1080i50
- 1080i59.94
- 1080i60
- 1080p23.98
- 1080p24
- 1080p25
- 1080p29.97
- 1080p30
- 1080p50
- 1080p59.94
- 1080p60

Arduino Sketch'lerinin Programlanması

Arduino yazılımına yazılan programların veya 'sketch'lerin yazılması çok kolaydır! Sketch'ler yaygın olan 'C' programlama dili kullanılarak yazılır. Studio Camera Control Protocol'ünden komutlar kullanarak sketch'lerinizi programlarken, shield bu komutları, Blackmagic URSA Mini veya Blackmagic Studio kameralarınızı kontrol etmenizi sağlayan SDI çıkışına gömer.

Desteklenen tüm komutlar, bu kullanım kılavuzunun Studio Camera Control Protocol bölümünde dahil edilmiştir, yani protokolden komutları alabilir ve onları sketch'inizde kullanabilirsiniz.

Blackmagic Shield ve Kütüphane Yüklemesinin Kontrolü

Her şey, "Başlarken' bölümünde anlatıldığı gibi bağlandıktan ve kurulum yazılımı ile kütüphane dosyalarını yükledikten sonra, shield'inizin Arduino kartı ile iletişimde olduğunu ve her şeyin olması gerektiği gibi çalıştığını kontrol etmeniz gerekir.

Bunun yapmanın hızlı bir yolu, size sağlanan tally yakıp söndürme örnek sketch'ini açmak ve çalıştırmaktır.

Bunu yapmak için:

1 Arduino IDE yazılımını başlatın.

2 'Tools' ismindeki araçlar menüsüne gidin ve Arduino kartı ve Port numarasını seçin.

3 'File' isimli dosya menüsünden 'Examples/BMDSDIControl' seçeneğini tıklayıp 'TallyBlink' isimli sketch'i seçin.

4 Kartınıza bu sketch'i yükleyin.

Blackmagic 3G-SDI Shield for Arduino'nuzu test etmenin hızlı ve kolay bir yolu, Tally Blink örnek sketch'idir. Studio Camera Protocol'den komutlar kullanarak I²C üzerinden shield'inize ham veri (Raw data) gönderilebilir, ancak sketch'lerin programlanmasını çok daha kolaylaştırmak için özel kütüphaneler de sağladık.

Blackmagic Studio Camera'nızdaki tally lambasının saniyede bir kez yanıp söndüğünü şimdi görmeniz gerekir. Tally lambasının yanıp söndüğünü görüyorsanız Blackmagic shield'in Arduino ile iletişimde olduğundan ve her şeyin düzgün bir şekilde çalıştığından emin olabilirsiniz.

Tally lambası yanıp sönmüyorsa Blackmagic kameranın tally lambasının 1 olarak ayarlı olup olmadığını ve kameranın ızgara kılavuz katmanının açık olup olmadığını kontrol edin.

Daha fazla yardım için, www.blackmagicdesign.com/tr/support adresindeki Blackmagic Design destek merkezini ziyaret edin. Shield'inizin kurulumu için alabileceğiniz farklı yardım yolları hakkında bilgi için, bu kılavuzun yardım bölümüne bakın.

LED Göstergeler

Blackmagic 3G-SDI Shield for Arduino'nuzda; güç, UART, I²C ve SPI iletişimi gibi faaliyetleri konfirme eden 6 adet LED göstergenin yanı sıra, tally ve kamera kontrolü geçersiz kılma fonksiyonunun aktif olduğunu gösteren göstergeler bulunur.
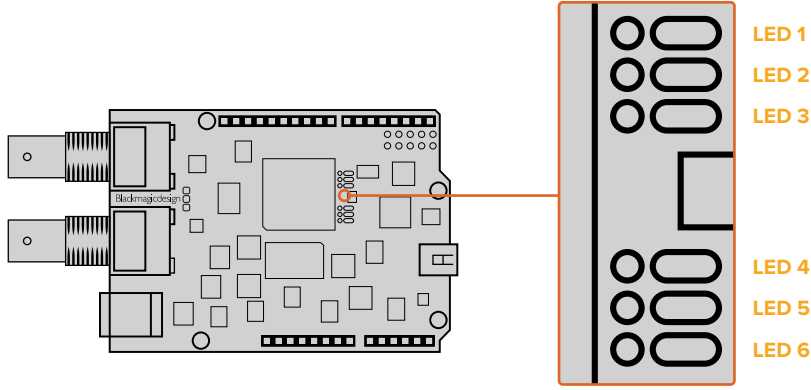
LED 1 - Sistem Aktif
Shield'e güç bağlandığında yanar.

LED 2 - Kontrolü Geçersiz Kılma Etkin
Arduino sketch'inizde kamera kontrolü etkinleştirdiğiniz zaman yanar.

LED 3 - Tally'i Geçersiz Kılma Etkin
Arduino sketch'inizde tally özelliğini etkinleştirdiğiniz zaman yanar.

LED 5 - I²C Ayrıştırıcı Meşgul
Shield'iniz ve Arduino arasında I²C prokolünü kullanan iletişim tespit edildiğinde yanar.

LED 6 - Seri Ayrıştırıcı Meşgul
UART iletişimi tespit edildiğinde yanar.

Blackmagic shield'iniz başlatılırken, güç göstergesi kapalı kalacak ve 3, 4 ve 5 numaralı LED'ler aşağıdaki faaliyetleri bildirecektir.

LED 3 - Uygulama görüntü yüklüyor

LED 4 - EEPROM başlatılıyor

LED 5 - Bellek kontrolü yapılıyor

Başarılı bir başlatmadan sonra, LED güç lambası yanacak ve tüm LED'ler çalışma esnasında standart fonksiyonlarını sürdürecektir.

Nadir olarak meydana gelen bir başlatma hatası durumunda, hatalı faaliyetin LED lambası haricindeki tüm LED lambaları hızla yanıp sönerler, böylelikle hata nedenini tespit edebilirsiniz.

Shield Bileşenlerinin Takılması

Kendinize ait bir donanım kontrolörü oluşturmayı istiyorsanız daha elle tutulur bir kontrol için; butonları, düğmeleri ve bir kumanda kolu olan yeni bir shield oluşturabilirsiniz. Shield'inizin başlık yuvalarına takarak, özel shield'inizi Blackmagic 3G-SDI Shield for Arduino'nuza monte edebilirsiniz. Oluşturabileceğiniz kontrolör tipleri sınırsızdır. Endüstri standardında bir kamera kontrol ünitesi için, eski bir kamera kontrol ünitesindeki (CCU) devre sistemini, kendinize özel Arduino çözümü ile değiştirmeniz bile mümkündür.
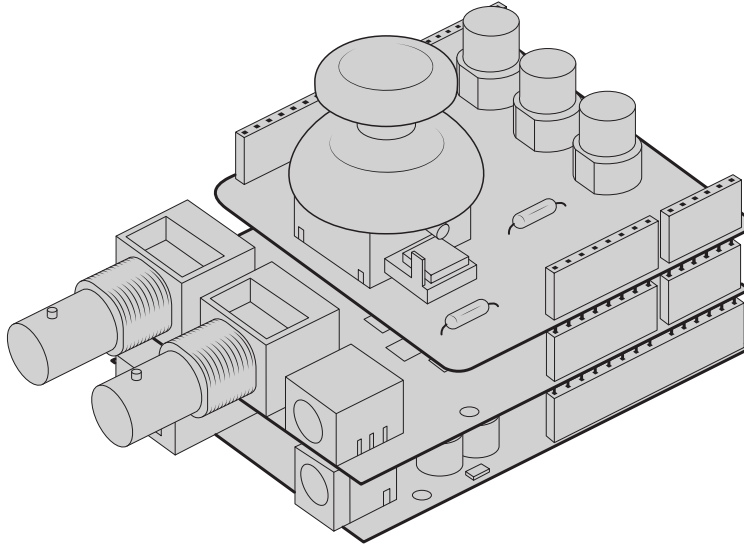
Daha etkileşimli ve geliştirilmiş kontrol için; kendinize ait bir donanım kontrolörü oluşturabilir ve onu Blackmagic 3G-SDI Shield for Arduino cihazınıza takabilirsiniz.

Blackmagic Shield for Arduino ile İletişim (İngilizce)

You can communicate with your Blackmagic 3G-SDI Shield for Arduino via I^{2}C or Serial. We recommend I^{2}C because of the low pin count and it frees up the serial monitor. This also allows you to use more I^{2}C devices with the shield.

High Level Overview

The library provides two core objects, BMD_SDITallyControl and BMD_SDICameraControl, which can be used to interface with the shield's tally and camera control functionalities. Either or both of these objects can be created in your sketch to issue camera control commands, or read and write tally data respectively. These objects exist in several variants, one for each of the physical I^{2}C or Serial communication busses the shield supports.

I²C Interface

To use the I²C interface to the shield:

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);
```

Serial Interface

To use the Serial interface to the shield:

```
BMD_SDICameraControl_Serial     sdiCameraControl;
BMD_SDITallyControl_Serial      sdiTallyControl;
```

Note that the library will configure the Arduino serial interface at the required 38400 baud rate. If you wish to print debug messages to the Serial Monitor when using this interface, change the Serial Monitor baud rate to match. If the Serial Monitor is used, some binary data will be visible as the IDE will be unable to distinguish between user messages and shield commands.

Example Usage

Once created in a sketch, these objects will allow you to issue commands to the shield over selected bus by calling functions on the created object or objects. A minimal sketch that uses the library via the I²C bus is shown below.

```
// NOTE: Must match address set in the setup utility software
const int                  shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);
BMD_SDITallyControl_I2C   sdiTallyControl(shieldAddress);

void setup() {
  // Must be called before the objects can be used
  sdiCameraControl.begin();
  sdiTallyControl.begin();

  // Turn on camera control overrides in the shield
  sdiCameraControl.setOverride(true);

  // Turn on tally overrides in the shield
  sdiTallyControl.setOverride(true);
}

void loop() {
    // Unused
}
```

The list of functions that may be called on the created objects are listed further on in this document. Note that before use, you must call the 'begin' function on each object before issuing any other commands.

Some example sketches demonstrating this library are included in the Arduino IDE's File->Examples->BMDSDIControl menu.

Studio Camera Control Protocol (İngilizce)

This section contains the Studio Camera Control Protocol from the Blackmagic Studio Camera manual. You can use the commands in this protocol to control supported Blackmagic Design cameras via your Blackmagic 3G-SDI Shield for Arduino.

The Blackmagic Studio Camera Protocol shows that each camera parameter is arranged in groups, such as:

Group ID	Group
0	Lens
1	Video
2	Audio
3	Output
4	Display
5	Tally
6	Reference
7	Configuration
8	Color Correction
10	Media
11	PTZ Control

The group ID is then used in the Arduino sketch to determine what parameter to change.

The function: sdiCameraControl.writeXXXX, is named based on what parameter you wish to change, and the suffix used depends on what group is being controlled.

For example sdiCameraControl.writeFixed16 is used for focus, aperture, zoom, audio, display, tally and color correction when changing absolute values.

The complete syntax for this command is as follows:

```
sdiCameraControl.writeFixed16 (
Camera number,
Group,
Parameter being controlled,
Operation,
Value
);
```

The operation type specifies what action to perform on the specified parameter

0 = assign value. The supplied Value is assigned to the specified parameter.

1 = offset value. Each value specifies signed offsets of the same type to be added to the current parameter Value.

For example:

```
sdiCameraControl.writeCommandFixed16(
1,
8,
0,
0,
liftAdjust
);
```

        1 = camera number 1
        8 = Color Correction group
        0 = Lift Adjust
        0 = assign value
        liftAdjust = setting the value for the RGB and luma levels

As described in the protocol section, liftAdjust is a 4 element array for RED[0], GREEN[1], BLUE[2] and LUMA[3]. The complete array is sent with this command.

The sketch examples included with the library files contain descriptive comments to explain their operation.

Blackmagic SDI Camera Control Protocol

Version 1.3

If you are a software developer you can use the SDI Camera Control Protocol to construct devices that integrate with our products. Here at Blackmagic Design our approach is to open up our protocols and we eagerly look forward to seeing what you come up with!

Overview

The Blackmagic SDI Camera Control Protocol is used by ATEM switchers, Blackmagic 3G-SDI Shield for Arduino and the Blackmagic Camera Control app to provide Camera Control functionality with supported Blackmagic Design cameras. Please refer to the 'Understanding Studio Camera Control' chapter section of this manual, or the ATEM Switchers Manual and SDK manual for more information. These can be downloaded at www.blackmagicdesign.com/support.

This document describes an extensible protocol for sending a uni directional stream of small control messages embedded in the non-active picture region of a digital video stream. The video stream containing the protocol stream may be broadcast to a number of devices. Device addressing is used to allow the sender to specify which device each message is directed to.

Assumptions

Alignment and padding constraints are explicitly described in the protocol document. Bit fields are packed from LSB first. Message groups, individual messages and command headers are defined as, and can be assumed to be, 32 bit aligned.

Blanking Encoding

A message group is encoded into a SMPTE 291M packet with DID/SDID x51/x53 in the active region of VANC line 16.

Message Grouping

Up to 32 messages may be concatenated and transmitted in one blanking packet up to a maximum of 255 bytes payload. Under most circumstances, this should allow all messages to be sent with a maximum of one frame latency.

If the transmitting device queues more bytes of message packets than can be sent in a single frame, it should use heuristics to determine which packets to prioritize and send immediately. Lower priority messages can be delayed to later frames, or dropped entirely as appropriate.

Abstract Message Packet Format

Every message packet consists of a three byte header followed by an optional variable length data block. The maximum packet size is 64 bytes.

Destination device (uint8)	Device addresses are represented as an 8 bit unsigned integer. Individual devices are numbered 0 through 254 with the value 255 reserved to indicate a broadcast message to all devices.
Command length (uint8)	The command length is an 8 bit unsigned integer which specifies the length of the included command data. The length does NOT include the length of the header or any trailing padding bytes.
Command id (uint8)	The command id is an 8 bit unsigned integer which indicates the message type being sent. Receiving devices should ignore any commands that they do not understand. Commands 0 through 127 are reserved for commands that apply to multiple types of devices. Commands 128 through 255 are device specific.
Reserved (uint8)	This byte is reserved for alignment and expansion purposes. It should be set to zero.
Command data (uint8[])	The command data may contain between 0 and 60 bytes of data. The format of the data section is defined by the command itself.
Padding (uint8[])	Messages must be padded up to a 32 bit boundary with 0x0 bytes. Any padding bytes are NOT included in the command length.

Receiving devices should use the destination device address and or the command identifier to determine which messages to process. The receiver should use the command length to skip irrelevant or unknown commands and should be careful to skip the implicit padding as well.

Defined Commands

Command 0 : change configuration

Category (uint8)	The category number specifies one of up to 256 configuration categories available on the device.
Parameter (uint8)	The parameter number specifies one of 256 potential configuration parameters available on the device. Parameters 0 through 127 are device specific parameters. Parameters 128 though 255 are reserved for parameters that apply to multiple types of devices.
Data type (uint8)	The data type specifies the type of the remaining data. The packet length is used to determine the number of elements in the message. Each message must contain an integral number of data elements.

Currently defined values are:

0: void / boolean	A void value is represented as a boolean array of length zero. The data field is a 8 bit value with 0 meaning false and all other values meaning true.
1: signed byte	Data elements are signed bytes
2: signed 16 bit integer	Data elements are signed 16 bit values
3: signed 32 bit integer	Data elements are signed 32 bit values
4: signed 64 bit integer	Data elements are signed 64 bit values
5: UTF-8 string	Data elements represent a UTF-8 string with no terminating character.

Data types 6 through 127 are reserved.

128: signed 5.11 fixed point	Data elements are signed 16 bit integers representing a real number with 5 bits for the integer component and 11 bits for the fractional component. The fixed point representation is equal to the real value multiplied by 2^11. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

Data types 129 through 255 are available for device specific purposes.

Operation type (uint8)	The operation type specifies what action to perform on the specified parameter. Currently defined values are:
0: assign value	The supplied values are assigned to the specified parameter. Each element will be clamped according to its valid range. A void parameter may only be 'assigned' an empty list of boolean type. This operation will trigger the action associated with that parameter. A boolean value may be assigned the value zero for false, and any other value for true.
1: offset / toggle value	Each value specifies signed offsets of the same type to be added to the current parameter values. The resulting parameter value will be clamped according to their valid range. It is not valid to apply an offset to a void value. Applying any offset other than zero to a boolean value will invert that value.

Operation types 2 through 127 are reserved.

Operation types 128 through 255 are available for device specific purposes.

Data (void)	The data field is 0 or more bytes as determined by the data type and number of elements.

The category, parameter, data type and operation type partition a 24 bit operation space.

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
	0.0	Focus	fixed16	–	0	1	0.0 = near, 1.0 = far
	0.1	Instantaneous autofocus	void	–	–	–	trigger instantaneous autofocus
	0.2	Aperture (f-stop)	fixed16	–	-1	16	Aperture Value (where fnumber = sqrt(2^AV))
	0.3	Aperture (normalised)	fixed16	–	0	1	0.0 = smallest, 1.0 = largest
	0.4	Aperture (ordinal)	int16	–	0	n	Steps through available aperture values from minimum (0) to maximum (n)
Lens	0.5	Instantaneous auto aperture	void	–	–	–	trigger instantaneous auto aperture
	0.6	Optical image stabilisation	boolean	–	–	–	true = enabled, false = disabled
	0.7	Set absolute zoom (mm)	int16	–	0	max	Move to specified focal length in mm, from minimum (0) to maximum (max)
	0.8	Set absolute zoom (normalised)	fixed16	–	0	1	Move to specified focal length: 0.0 = wide, 1.0 = tele
	0.9	Set continuous zoom (speed)	fixed16	–	-1	+1.0	Start/stop zooming at specified rate: -1.0 = zoom wider fast, 0.0 = stop, +1 = zoom tele fast

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Video	1.0	Video mode	int8	[0] = frame rate	–	–	24, 25, 30, 50, 60
				[1] = M-rate	–	–	0 = regular, 1 = M-rate
				[2] = dimensions	–	–	0 = NTSC, 1 = PAL, 2 = 720, 3 = 1080, 4 = 2k, 5 = 2k DCI, 6 = UHD
				[3] = interlaced	–	–	0 = progressive, 1 = interlaced
				[4] = Color space	–	–	0 = YUV
	1.1	Gain (up to Camera 4.9)	int8		1	16	1 = 100 ISO, 2 = 200 ISO, 4 = 400 ISO, 8 = 800 ISO, 16 = 1600 ISO
	1.2	Manual White Balance	int16	[0] = color temp	2500	10000	Color temperature in K
			int16	[1] = tint	-50	50	tint
	1.3	Set auto WB	void	–	–	–	Calculate and set auto white balance
	1.4	Restore auto WB	void	–	–	–	Use latest auto white balance setting
	1.5	Exposure (us)	int32		1	42000	time in us
	1.6	Exposure (ordinal)	int16	–	0	n	Steps through available exposure values from minimum (0) to maximum (n)
	1.7	Dynamic Range Mode	int8 enum	–	0	1	0 = film, 1 = video,
	1.8	Video sharpening level	int8 enum	–	0	3	0 = off, 1 = low, 2 = medium, 3 = high
	1.9	Recording format	int16	[0] = file frame rate	–	–	fps as integer (eg 24, 25, 30, 50, 60, 120)
				[1] = sensor frame rate	–	–	fps as integer, valid when sensor-off-speed set (eg 24, 25, 30, 33, 48, 50, 60, 120), no change will be performed if this value is set to 0
				[2] = frame width	–	–	in pixels
				[3] = frame height	–	–	in pixels
				[4] = flags	–	–	[0] = file-M-rate
					–	–	[1] = sensor-M-rate, valid when sensor-off-speed-set
					–	–	[2] = sensor-off-speed
					–	–	[3] = interlaced
					–	–	[4] = windowed mode
	1.10	Set auto exposure mode	int8	–	0	4	0 = Manual Trigger, 1 = Iris, 2 = Shutter, 3 = Iris + Shutter, 4 = Shutter + Iris
	1.11	Shutter angle	int32	–	100	36000	Shutter angle in degrees, multiplied by 100
	1.12	Shutter speed	int32	–	24	2000	Shutter speed value as a fraction of 1, so 50 for 1/50th of a second
	1.13	Gain	int8	–	-128	127	Gain in decibel (dB)
	1.14	ISO	int32	–	0	2147483647	ISO value

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Audio	2.0	Mic level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.1	Headphone level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.2	Headphone program mix	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.3	Speaker level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	2.4	Input type	int8	–	0	2	0 = internal mic, 1 = line level input, 2 = low mic level input, 3 = high mic level input
	2.5	Input levels	fixed16	[0] ch0	0	1	0.0 = minimum, 1.0 = maximum
				[1] ch1	0	1	0.0 = minimum, 1.0 = maximum
	2.6	Phantom power	boolean	–	–	–	true = powered, false = not powered
Output	3.0	Overlay enables	uint16 bit field	–	–	–	bit flags: [0] = display status, [1] = display frame guides. Some cameras don't allow separate control of frame guides and status overlays.
	3.1	Frame guides style (Camera 3.x)	int8	[0] = frame guides style	0	8	0 = HDTV, 1 = 4:3, 2 = 2.4:1, 3 = 2.39:1, 4 = 2.35:1, 5 = 1.85:1, 6 = thirds
	3.2	Frame guides opacity (Camera 3.x)	fixed16	[1] = frame guide opacity	0.1	1	0.0 = transparent, 1.0 = opaque
	3.3	Overlays (replaces .1 and .2 above from Cameras 4.0)	int8	[0] = frame guides style	–	–	0 = off, 1 = 2.4:1, 2 = 2.39:1, 3 = 2.35:1, 4 = 1.85:1, 5 = 16:9, 6 = 14:9, 7 = 4:3, 8 = 2:1
				[1] = frame guide opacity	0	100	0 = transparent, 100 = opaque
				[2] = safe area percentage	0	100	percentage of full frame used by safe area guide (0 means off)
				[3] = grid style	–	–	bit flags: [0] = display thirds, [1] = display cross hairs, [2] = display center dot

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Display	4.0	Brightness	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.1	Overlay enables	int16 bit field	–	–	–	0x4 = zebra
				–	–	–	0x8 = peaking
				–	–	–	
	4.2	Zebra level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.3	Peaking level	fixed16	–	0	1	0.0 = minimum, 1.0 = maximum
	4.4	Color bars display time (seconds)	int8	–	0	30	0 = disable bars, 1-30 = enable bars with timeout (s)
	4.5	Focus Assist	int8	[0] = focus assist method	–	–	0 = Peak, 1 = Colored lines
				[1] = focus line color	–	–	0 = Red, 1 = Green, 2 = Blue, 3 = White, 4 = Black
Tally	5.0	Tally brightness	fixed16	–	0	1	Sets the tally front and tally rear brightness to the same level. 0.0 = minimum, 1.0 = maximum
	5.1	Front tally brightness	fixed16	–	0	1	Sets the tally front brightness. 0.0 = minimum, 1.0 = maximum
	5.2	Rear tally brightness	fixed16	–	0	1	Sets the tally rear brightness. 0.0 = minimum, 1.0 = maximum Tally rear brightness cannot be turned off
Reference	6.0	Source	int8 enum	–	0	2	0 = internal, 1 = program, 2 = external
	6.1	Offset	int32	–	–	–	+/- offset in pixels
Configuration	7.0	Real Time Clock	int32	[0] time	_	_	BCD - HHMMSSFF (UCT)
				[1] date	_	_	BCD - YYYYMMDD
	7.1	System language	string	_	_	_	ISO-639-1 two character language code
	7.2	Timezone	int32	_	_	_	Minutes offset from UTC
	7.3	Location	int64	[0] latitude	_	_	BCD - s0DDdddddddddddd where s is the sign: 0 = north (+), 1 = south (-); DD degrees, dddddddddddd decimal degrees
				[1] longitude	_	_	BCD - sDDDdddddddddddd where s is the sign: 0 = west (-), 1 = east (+); DDD degrees, ddddddddddddd decimal degrees

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Color Correction	8.0	Lift Adjust	fixed16	[0] red	-2	2	default 0.0
				[1] green	-2	2	default 0.0
				[2] blue	-2	2	default 0.0
				[3] luma	-2	2	default 0.0
	8.1	Gamma Adjust	fixed16	[0] red	-4	4	default 0.0
				[1] green	-4	4	default 0.0
				[2] blue	-4	4	default 0.0
				[3] luma	-4	4	default 0.0
	8.2	Gain Adjust	fixed16	[0] red	0	16	default 1.0
				[1] green	0	16	default 1.0
				[2] blue	0	16	default 1.0
				[3] luma	0	16	default 1.0
	8.3	Offset Adjust	fixed16	[0] red	-8	8	default 0.0
				[1] green	-8	8	default 0.0
				[2] blue	-8	8	default 0.0
				[3] luma	-8	8	default 0.0
	8.4	Contrast Adjust	fixed16	[0] pivot	0	1	default 0.5
				[1] adj	0	2	default 1.0
	8.5	Luma mix	fixed16	–	0	1	default 1.0
	8.6	Color Adjust	fixed16	[0] hue	-1	1	default 0.0
				[1] sat	0	2	default 1.0
	8.7	Correction Reset Default	void	–	–	–	reset to defaults

Group	ID	Parameter	Type	Index	Minimum	Maximum	Interpretation
Media	10.0	Codec	int8 enum	[0] = basic codec	–	–	0 = RAW, 1 = DNxHD, 2 = ProRes, 3 = Blackmagic RAW
				[1] = codec variant	–	–	RAW: 0 = Uncompressed, 1 = lossy 3:1, 2 = lossy 4:1
					–	–	ProRes: 0 = HQ, 1 = 422, 2 = LT, 3 = Proxy, 4 = 444, 5 = 444XQ
					–	–	Blackmagic RAW: 0 = Q0, 1 = Q5, 2 = 3:1, 3 = 5:1, 4 = 8:1, 5 = 12:1
	10.1	Transport mode	int8	[0] = mode	–	–	0 = Preview, 1 = Play, 2 = Record
				[1] = speed	–	–	-ve = multiple speeds backwards, 0 = pause, +ve = multiple speeds forwards
				[2] = flags	–	–	1<<0 = loop, 1<<1 = play all, 1<<5 = disk1 active, 1<<6 = disk2 active, 1<<7 = time-lapse recording
				[3] = slot 1 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
				[4] = slot 2 storage medium	–	–	0 = CFast card, 1 = SD, 2 = SSD Recorder
PTZ Control	11.0	Pan/Tilt Velocity	fixed 16	[0] = pan velocity	-1.0	1.0	-1.0 = full speed left, 1.0 = full speed right
				[1] = tilt velocity	-1.0	1.0	-1.0 = full speed down, 1.0 = full speed up
	11.1	Memory Preset	int8 enum	[0] = preset command	–	–	0 = reset, 1 = store location, 2 = recall location
			int8	[1] = preset slot	0	5	–

Example Protocol Packets

Operation	Packet Length	Byte															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		header				command				data							
		destination	length	command	reserved	category	parameter	type	operation								
trigger instantaneous auto focus on camera 4	8	4	4	0	0	0	1	0	0								
turn on OIS on all cameras	12	255	5	0	0	0	6	0	0	1	0	0	0				
set exposure to 10 ms on camera 4 (10 ms = 10000 us = 0x00002710)	12	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00				
add 15% to zebra level (15 % = 0.15 f = 0x0133 fp)	12	4	6	0	0	4	2	128	1	0x33	0x01	0	0				
select 1080p 23.98 mode on all cameras	16	255	9	0	0	1	0	1	0	24	1	3	0	0	0	0	0
subtract 0.3 from gamma adjust for green & blue (-0.3 ~= 0xfd9a fp)	16	4	12	0	0	8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0
all operations combined	76	4	4	0	0	0	1	0	0	255	5	0	0	0	6	0	0
		1	0	0	0	4	8	0	0	1	5	3	0	0x10	0x27	0x00	0x00
		4	6	0	0	4	2	128	1	0x33	0x01	0	0	255	9	0	0
		1	0	1	0	24	1	3	0	0	0	0	0	4	12	0	0
		8	1	128	1	0	0	0x9a	0xfd	0x9a	0xfd	0	0				

Yazılım Geliştiricileri için Bilgiler (İngilizce)

This section of the manual provides all the details you will need if you want to write custom libraries and develop your own hardware for your Blackmagic 3G-SDI Shield for Arduino.

Physical Encoding - I²C

The shield operates at the following I²C speeds:

1.   Standard mode (100 kbit/s)
2.   Full speed (400 kbit/s)

The default 7-bit shield I²C slave address is 0x6E.

```
 Shield Pin          | Function
-------------------- |----------------------------
 A4                  | Serial Data (SDA)
 A5                  | Serial Clock (SCL)
```

I²C Protocol (Writes):

        (START W) [REG ADDR L] [REG ADDR H] [VAL] [VAL] [VAL] ... (STOP)

I²C Protocol (Reads):

        (START W) [REG ADDR L] [REG ADDR H] ... (STOP) (START R) [VAL] [VAL] [VAL] ... (STOP)

The maximum payload (shown as **VAL** in the examples above) read/write length (following the internal register address) in a single transaction is 255 bytes.

Physical Encoding - UART

The shield operates with a UART baud rate of 115200, 8-N-1 format.

```
 Shield Pin          | Function
-------------------- |----------------------------
 IO1                 | Serial Transmit (TX)
 IO0                 | Serial Receive (RX)
```

UART Protocol (Writes):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['W'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

UART Protocol (Reads):

        [0xDC] [0x42] [REG ADDR L] [REG ADDR H] ['R'] [LENGTH] [0x00] [VAL] [VAL] [VAL] ...

The maximum payload (shown as **VAL** in the examples above) read/write length (specified in the **LENGTH** field) in a single transaction is 255 bytes.

Register Address Map

The shield has the following user address register map:

```
Address               | Name         | R/W    | Register Description
--------------------- |----------    |-----   |-----------------------------

0x0000 - 0x0003       | IDENTITY     | R      | Hardware Identifier
0x0004 - 0x0005       | HWVERSION    | R      | Hardware Version
0x0006 - 0x0007       | FWVERSION    | R      | Firmware Version
                      |              |        |
0x1000                | CONTROL      | R/W    | System Control
                      |              |        |
0x2000                | OCARM        | R/W    | SDI Control Override Arm
0x2001                | OCLENGTH     | R/W    | SDI Control Override Length
0x2100 - 0x21FE       | OCDATA       | R/W    | SDI Control Override Data
                      |              |        |
0x3000                | ICARM        | R/W    | SDI Control Incoming Arm
0x3001                | ICLENGTH     | R      | SDI Control Incoming Length
0x3100 - 0x31FE       | ICDATA       | R      | SDI Control Incoming Data
                      |              |        |
0x4000                | OTARM        | R/W    | SDI Tally Override Arm
0x4001                | OTLENGTH     | R/W    | SDI Tally Override Length
0x4100 - 0x41FE       | OTDATA       | R/W    | SDI Tally Override Data
                      |              |        |
0x5000                | ITARM        | R/W    | SDI Tally Incoming Arm
0x5001                | ITLENGTH     | R      | SDI Tally Incoming Length
0x5100 - 0x51FE       | ITDATA       | R      | SDI Tally Incoming Data
```

All multi-byte numerical fields are stored little-endian. Unused addresses are reserved and read back as zero.

Register: IDENTITY (Board Identifier)

```
[ IDENTITY ]
31        0
```

Identity: ASCII string 'SDIC' (i.e. `0x43494453`) in hexadecimal.

Register: HWVERSION (Hardware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7            0
```

Version Major: Hardware revision, major component.

Version Minor: Hardware revision, minor component.

Register: FWVERSION (Firmware Version)

```
[ VERSION MAJOR ] [ VERSION MINOR ]
15          8 7            0
```

Version Major: Firmware revision, major component.

Version Minor: Firmware revision, minor component.

Register: CONTROL (System Control)

```
[ RESERVED ] [ OVERRIDE OUTPUT ] [ RESET TALLY ] [ OVERRIDE TALLY ] [
OVERIDE CONTROL ]
7       4            3              2              1              0
```

Reserved: Always zero.

Override Output: When 1, the input SDI signal (if present) is discarded and the shield generates its own SDI signal on the SDI output connector. When 0, the input signal is passed through to the output if present, or the shield generates its own SDI signal if not.

Reset Tally: When 1, the last received incoming tally data is immediately copied over to the override tally data register. Automatically cleared by hardware.

Override Tally: When 1, tally data is overridden with the user supplied data. When 0, input tally data is passed through to the output unmodified.

Override Control: When 1, control data is overridden with the user supplied data. When 0, input control data is passed through to the output unmodified.

Register: OCARM (Output Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, the outgoing control is data armed and will be sent in the next video frame. Automatically cleared once the control has been sent.

Register: OCLENGTH (Output Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data to send in OCDATA.

Register: OCDATA (Output Control Payload Data)

[CONTROL DATA]
255*8-1   0

Control Data: Control data that should be embedded into a future video frame.

Register: ICARM (Incoming Control Arm)

[RESERVED] [ARM]
7   1   0

Reserved: Always zero.

Arm: When 1, incoming control data is armed and will be received in the next video frame. Automatically cleared once a control packet has been read.

Register: ICLENGTH (Incoming Control Length)

[LENGTH]
7   0

Length: Length in bytes of the data in _ICDATA_. Automatically set when a new packet has been cached.

Register: ICDATA (Incoming Control Payload Data)

[CONTROL DATA]
255*8-1　　0

Control Data:　Last control data extracted from a video frame since _ICARM.ARM_ was reset.

Register: OTARM (Output Tally Arm)

[RESERVED] [ARM]
7　　1　　0

Reserved:　　Always zero.

Arm:　　When 1, the outgoing tally data is armed and will be continuously from the next video frame until new data is set. Automatically cleared once the tally has been sent in at least one frame.

Register: OTLENGTH (Output Tally Length)

[LENGTH]
7　　0

Length:　　Length in bytes of the data to send in OTDATA.

Register: OTDATA (Output Tally Data)

[TALLY DATA]
255*8-1　　0

Tally Data:　Tally data that should be embedded into a future video frame (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Register: ITARM (Input Tally Arm)

[RESERVED] [ARM]
7　　1　　0

Reserved:　　Always zero.

Arm:　　When 1, tally data armed and will be received in the next video frame. Automatically cleared once the tally has been read.

Register: ITLENGTH (Input Tally Length)

[LENGTH]
7　　0

Length:　　Length in bytes of the data in _ITDATA_. Automatically set when a new packet has been cached.

Register: ITDATA (Input Tally Data)

[TALLY DATA]
255*8-1　　0

Tally Data:　Last tally data extracted from a video frame since _ITARM.ARM_ was reset (one byte per camera). Bit zero indicates a Program tally, while bit one indicates a Preview tally.

Yardım

Yardım İçin

Blackmagic 3G-SDI Shield for Arduino'nuz özel gereksinimlerinize dayanarak bağımsız geliştirmeler yapmanız için tasarlanmış, geliştirici aracıdır.

Shield'iniz hakkında en güncel bilgileri edinmek için, Blackmagic Design online destek sayfalarına girip en yeni destek malzemelerini inceleyin.

Blackmagic Design Online Destek Sayfaları

En son kılavuz, yazılım ve destek notlarına, www.blackmagicdesign.com/tr/support adresindeki, Blackmagic Design destek merkezinden ulaşılabilir.

Arduino Geliştirme Forumu

Programlama hakkında sorularınız varsa internetteki Arduino geliştirme forumlarından yardım alabilirsiniz. Yazılım hakkında sorular sorabileceğiniz veya sizin için çözümünüzü gerçekleştirebilecek, kiralayabileceğiniz istekli bir mühendis bulabileceğiniz birçok iyi nitelikte forumlar ve tam bir Arduino geliştirici topluluğu vardır.

Blackmagic Design Forum

Web sitemizdeki Blackmagic Design forum, daha fazla bilgi ve yaratıcı fikirler için ziyaret edebileceğiniz faydalı bir kaynaktır. Burası yardım alabilmeniz için daha hızlı bir yol da olabilir; çünkü başka deneyimli kullanıcılar ya da Blackmagic Design çalışanları tarafından sorularınıza yanıtlar bulabilir ve çalışmalarınıza bu sayede devam edebilirsiniz. Foruma, https://forum.blackmagicdesign.com adresinden ulaşabilirsiniz.

Mevcut Yazılım Sürümünün Kontrol Edilmesi

Bilgisayarınızda Blackmagic 3G-SDI Shield for Arduino Setup yazılımının hangi sürümünün yüklü olduğuna göz atmak için, About Blackmagic 3G-SDI Shield for Arduino Setup penceresini açın.

- Mac OS X sistemlerde, Uygulamalar dosyasından Blackmagic 3G-SDI Shield for Arduino Setup yazılımını açın. Sürüm numarasını görüntülemek için, uygulamalar menüsünden About Blackmagic 3G-SDI Shield for Arduino Setup' sekmesini seçin.
- Windows 7 sistemlerde, 'Blackmagic 3G-SDI Shield for Arduino Setup' dosyasını Başlat (Start) menüsünden açın. Sürüm numarasını görüntülemek için, Help (Yardım) menüsüne tıklayın ve About Blackmagic 3G-SDI Shield for Arduino Setup sekmesini seçin.
- Windows 8 sistemlerde, 'Blackmagic 3G-SDI Shield for Arduino Setup' dosyasını, Başlat (Start) sayfasındaki Blackmagic 3G-SDI Shield for Ardunio butonundan açın. Sürüm numarasını görüntülemek için, Help (Yardım) menüsünü tıklayın ve About Blackmagic 3G-SDI Shield for Arduino Setup sekmesini seçin.

En Son Yazılım Güncellemelerine Erişim

Bilgisayarınızda yüklü bulunan Blackmagic 3G-SDI Shield for Arduino Setup yazılımının sürümünü denetledikten sonra, lütfen Blackmagic Design destek merkezine www.blackmagicdesign.com/tr/support adresinden girerek en son güncellemeleri gözden geçirin. En son güncellemeleri çalıştırmak faydalı olsa da önemli bir projenin ortasındayken, yazılımı güncellemekten kaçınmakta yarar vardır.

Garanti

12 Ay Sınırlı Garanti

Blackmagic Design şirketi, Blackmagic 3G-SDI Shield for Arduino ürününün satın alındığı tarihten itibaren 12 ay boyunca malzeme ve işçilik bakımından kusursuz olacağını garantilemektedir. Üründe, bu garanti süresi içinde bir arıza ve kusur söz konusu olursa Blackmagic Design, kendi seçimi doğrultusunda ya arızalı ürünü parça ve işçilik bedeli talep etmeksizin tamir edecektir ya da arızalı ürünü yenisiyle değiştirecektir.

Bu garanti kapsamındaki hizmetten yararlanmak için, kusur ve hataya ilişkin garanti süresi sona ermeden, Müşteri Blackmagic Design'i bilgilendirmeli ve söz konusu hizmetin sağlanması için uygun düzenlemeleri yapmalıdır. Blackmagic Design tarafından özel belirlenmiş ve yetkilendirilmiş bir hizmet merkezine arızalı ürünün ambalajlanarak nakliyesi, Müşteri'nin sorumluluğudur ve nakliye ücretleri, peşin ödenmiş olmalıdır. Bize herhangi bir sebepten dolayı iade edilen ürünlerin tüm nakliyat değişiklikleri, sigorta, yasal bedel, vergi ve diğer tüm masrafları, müşterinin sorumluluğundadır.

Bu garanti; yanlış kullanım ya da yanlış veya kusurlu bakımdan kaynaklanan herhangi bir arızayı, bozukluğu ya da hasarı kapsamaz. Blackmagic Design burada açıklanan durumlarda, bu garanti kapsamında hizmet sağlamak zorunda değildir: a) Blackmagic Design temsilcileri haricindeki başka personelin ürünü kurma, tamir etme ya da bakımını yapma girişimlerinden kaynaklanan hasarın tamir edilmesi, b) uygun olmayan kullanım veya uyumlu olmayan ekipmanlara bağlanılmasından kaynaklanan hasarın tamir edilmesi, c) Blackmagic Design parçaları ya da malzemesi olmayan ürünlerin kullanımından kaynaklanan hasarın ya da arızanın tamir edilmesi ya da d) Modifiye veya başka ürünlerle entegre edilmiş bir ürünün; söz konusu modifikasyon ya da entegrasyonun gereken tamiratın süresini uzattığı ya da ürün bakımını zorlaştırdığı durumlarda, tamir edilmesi. BU GARANTİ, BLACKMAGIC DESIGN TARAFINDAN VERİLMİŞTİR VE AÇIK YA DA ZIMNİ, HERHANGİ BİR GARANTİNİN YERİNİ TUTAR. BLACKMAGIC DESIGN VE SATICILARI, ZIMNİ TİCARİ UYGUNLUK GARANTİSİNİ YA DA ÖZEL BİR AMACA UYGUNLUK GARANTİSİNİ KABUL ETMEZ. KUSURLU BİR ÜRÜNÜN TAMİRİ VEYA DEĞİŞTİRİLMESİ, BLACKMAGIC DESIGN'IN MÜŞTERİLERİNE SUNDUĞU TAM VE MÜNHASIR ÇÖZÜMDÜR. BLACKMAGIC DESIGN YA DA SATICILARININ, OLABİLECEK HASARLAR HAKKINDA ÖNCEDEN BİLGİSİ OLMASINI GÖZETMEKSİZİN; ÜRÜNDE DOLAYLI, ÖZEL, TESADÜFİ YA DA NETİCE OLARAK ORTAYA ÇIKAN HERHANGİ BİR HASAR İÇİN, BLACKMAGIC DESIGN SORUMLU DEĞİLDİR. BLACKMAGIC DESIGN, MÜŞTERİLER TARAFINDAN CİHAZIN YASAL OLMAYAN HERHANGİ BİR KULLANIMINDAN, SORUMLU DEĞİLDİR. BLACKMAGIC DESIGN, BU ÜRÜNÜN KULLANIMINDAN KAYNAKLANAN HERHANGİ BİR HASARDAN, SORUMLU DEĞİLDİR. BU ÜRÜNÜN ÇALIŞTIRILMASINDAN DOĞAN RİSK, KULLANICININ KENDİSİNE AİTTİR.