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.

English 3
日本語 32
Français 62
Deutsch 92
Español 122
中文 152
한국어 182
Русский 212
Italiano 242
Português 272
Türkçe 302

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

Grant Petty
CEO Blackmagic Design

Contents

Blackmagic 3G-SDI Shield for Arduino

Getting Started 5
Attaching and Soldering Headers 5
Mounting to the Arduino Board 6
Plugging in Power 6
Connecting to SDI Equipment 7
Software Installation 8
Installing Internal Software 8
Installing Arduino Library Files 9
Blackmagic Shield for Arduino Setup 10
I^{2}C Address 10
Video Format 11
Programming Arduino Sketches 11
Testing your Blackmagic Shield and Library Installation 12
LED Indicators 13
Attaching Shield Components 14
Communicating with your Blackmagic Shield for Arduino 14
High Level Overview 14
I^{2}C Interface 15
Serial Interface 15
Example Usage 15
Studio Camera Control Protocol 16
Blackmagic SDI Camera Control Protocol 17
Overview 17
Assumptions 17
Blanking Encoding 17
Message Grouping 17
Abstract Message Packet Format 17
Defined Commands 18
Example Protocol Packets 25
Developer Information 26
Physical Encoding - I^{2}C 26
Physical Encoding - UART 26
Help 30
Warranty 31

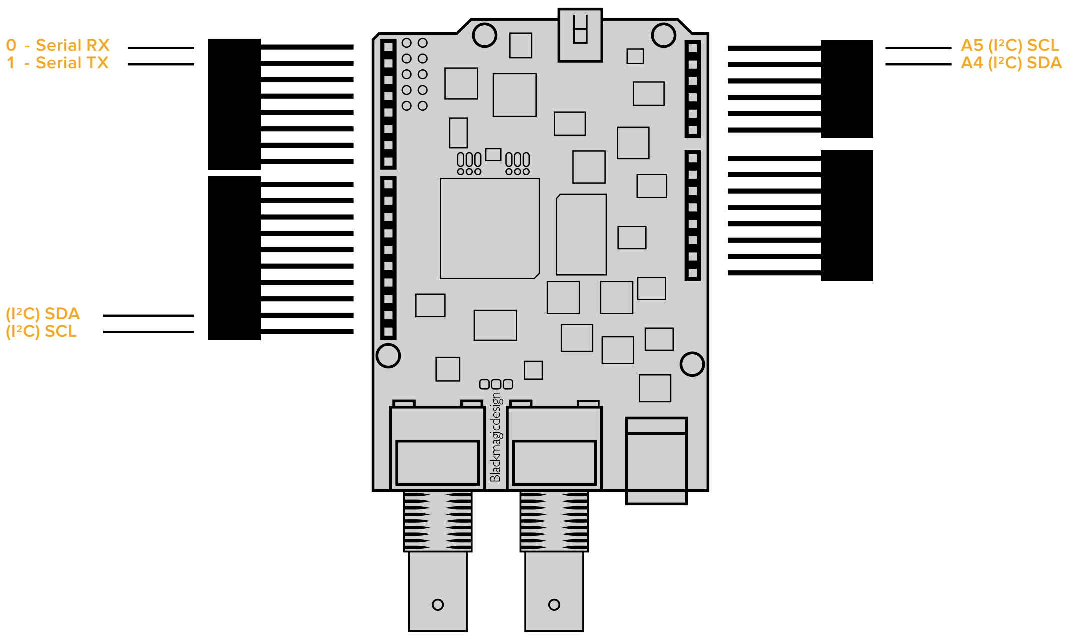
Getting Started

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.



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.

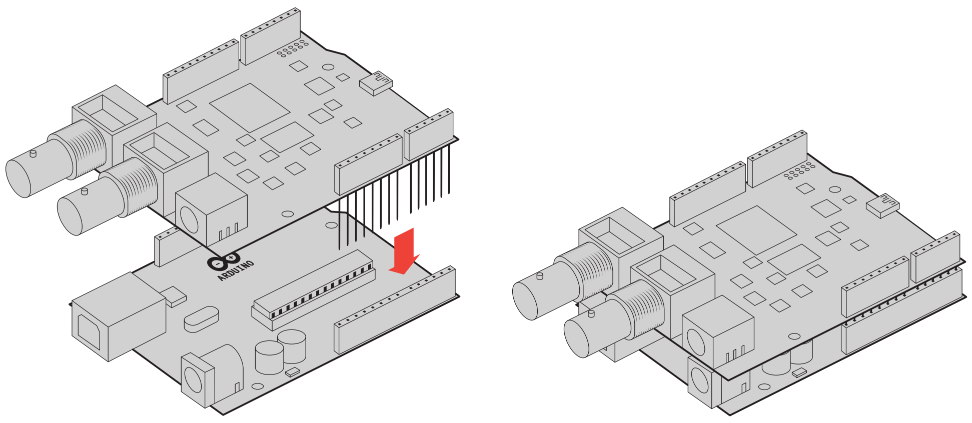
2 Solder the base of each header pin to the underside of your shield. Make sure the solder on each pin creates a firm join with the pin hole, but does not touch the solder on nearby pins.

TIP To help make sure all pins on your shield are aligned with the female header pin slots on the Arduino board, it's helpful to solder just one pin on each header first. Now place the shield onto the Arduino board to check the pin alignment. If any headers need adjusting, you can then warm the solder joint on the corresponding header and improve its alignment. This is a much easier method than soldering all the joints first and then trying to make adjustments.

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.

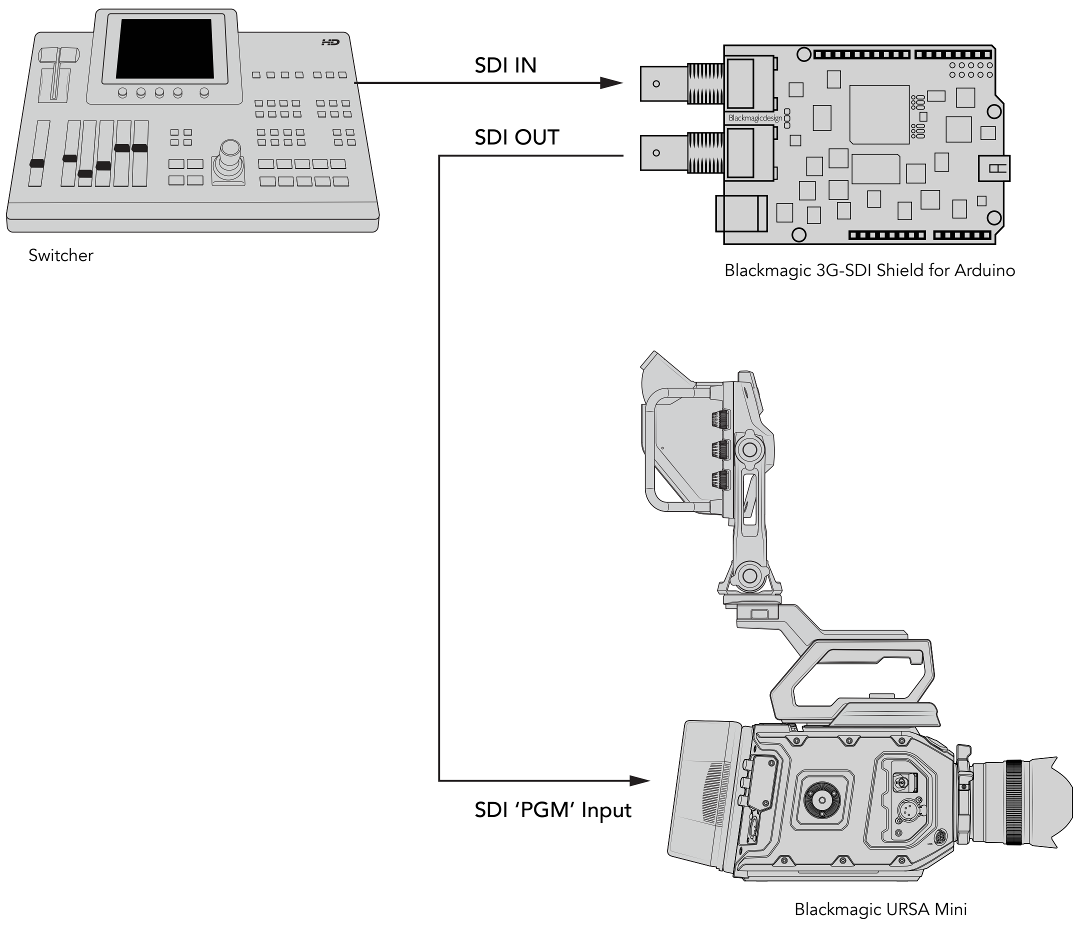
NOTE Plugging power into the Arduino board will not provide sufficient power to the Blackmagic shield, however, powering the Blackmagic shield will provide power to the Arduino as well, so make sure power is connected to your Blackmagic shield only.

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.



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

NOTE Before installing the Blackmagic Shield for Arduino setup utility, download the latest Arduino IDE software from www.arduino.cc and install it on your computer.

After installing the Arduino software, you can now install your Blackmagic 3G-SDI Shield's internal software.

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.

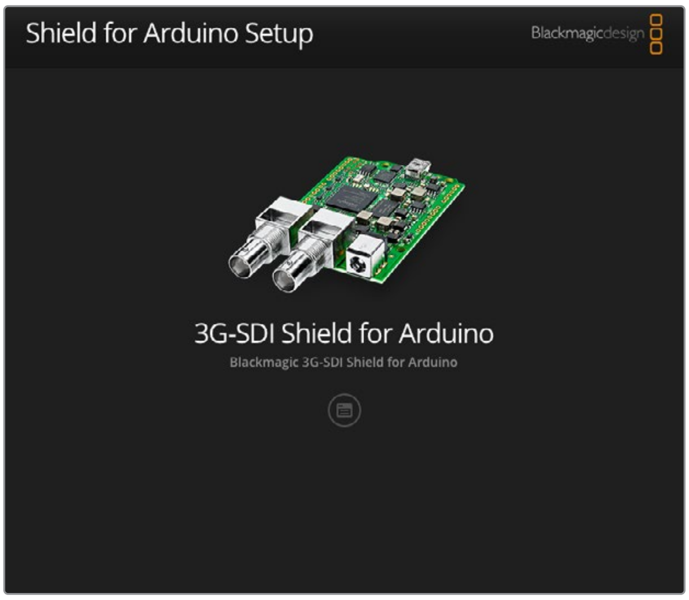
That's all you need to do to install the Blackmagic Design library files on your computer. When running the Arduino software, you will now also have Blackmagic Design example sketches to choose from.

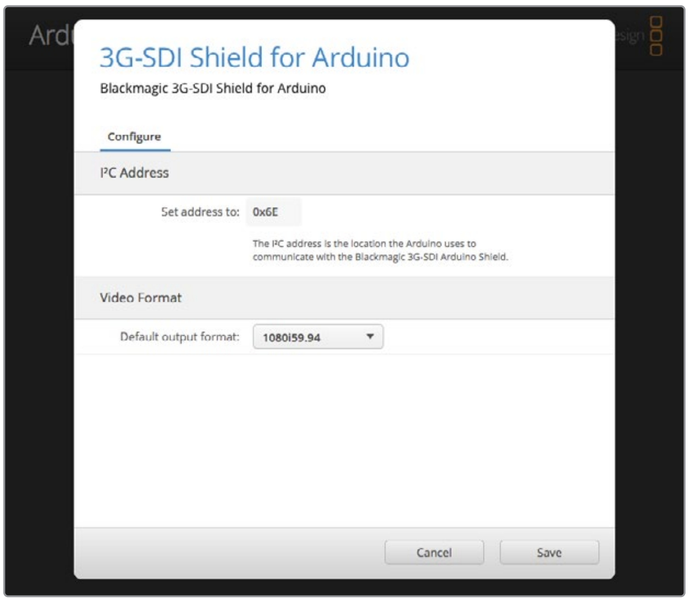
Simply go to the 'file' drop down menu in the Arduino software menu bar, and select 'examples'. Now select BMDSDIControl and you will see a list of example sketches you can use.

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


Shield for Arduino Setup
Blackmagicdesign
3G-SDI Shield for Arduino
Blackmagic 3G-SDI Shield for Arduino


3G-SDI Shield for Arduino
Blackmagic 3G-SDI Shield for Arduino
Configure
Cancel
Save

I²C Address

Set address to: 0x6E

The I²C address is the location the Arduino uses to communicate with the Blackmagic 3G-SDI Arduino Shield.

Video Format

Default output format: 1080i59.94

The Blackmagic Shield for Arduino Setup software lets you change settings on your shield such as the I^{2}C 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 'I^{2}C 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.

I^{2}C Address

In very rare cases, there is a potential for another shield mounted to your Blackmagic shield to share the same I^{2}C 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.

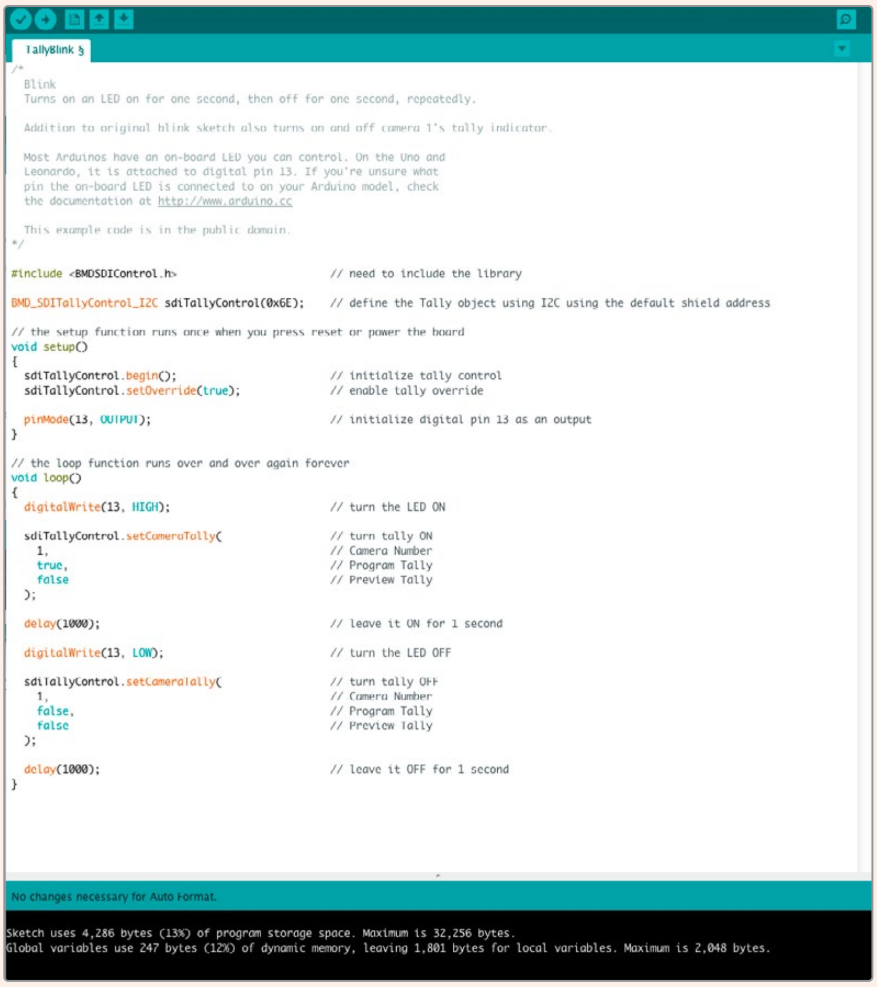
Testing your Blackmagic Shield and Library Installation

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.



```
/*
  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.
```

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.

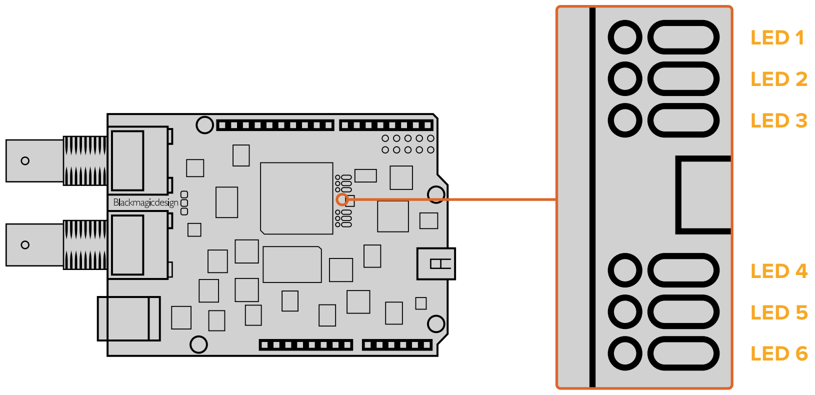
You should now see the tally light on your Blackmagic Studio Camera blink once every second. If you see the tally light blinking you can be sure your Blackmagic shield is communicating with the Arduino and everything is working properly.

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

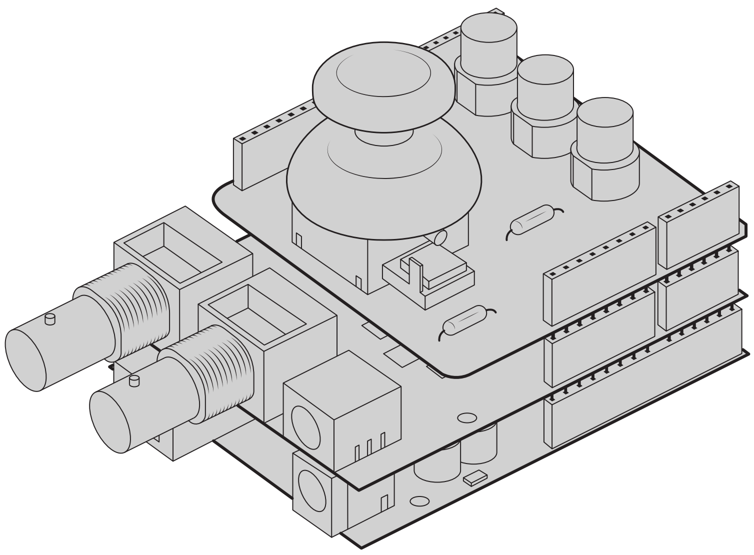
LED 5 - Memory check in progress

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 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
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 - 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	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 - I^{2}C

The shield operates at the following I^{2}C speeds:

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

The default 7-bit shield I^{2}C slave address is 0x6E.

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

I^{2}C Protocol (Writes):

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

I^{2}C 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.