Blackmagicdesign

Developer Information

Blackmagic PTZ Control

Camera Control using ATEM, PTZ Control over SDI, VISCA commands, Controlling Pan, Tilt and Zoom with Blackmagic 3G-SDI Arduino Shield, Controlling your Arduino

April 2018

Contents

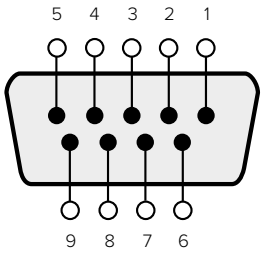
Blackmagic PTZ Control

Camera Control using ATEM

VISCA

All ATEM switchers with a remote port support VISCA camera control via RS-422. VISCA commands are defined by controlling the cameras via ATEM external hardware panels, such as ATEM 1 M/E Advanced Panel and ATEM Broadcast Panels.

Refer to the ATEM Production Studio Switchers and ATEM Television Studio Switchers manuals for more information.



Receive (–)	Receive (+)	Transmit (–)	Transmit (+)	Ground Pins
8	3	2	7	1, 4, 6, 9

RS-422 PTZ pin connections.

PTZ Control over SDI

ATEM

ATEM external hardware panels, such as ATEM 1 M/E Advanced Panel and ATEM Broadcast Panels can control PTZ camera heads via your switcher's SDI program return output. By connecting the program return feed from your switcher to a Blackmagic Micro Studio Camera, then connecting the SDI output from the camera's expansion cable to your PTZ head, you can control the head via the SDI signal.

For more information on PTZ control using a Blackmagic Micro Studio Camera refer to the Blackmagic Studio Cameras manual. This manual can be downloaded from the Blackmagic Design support center at www.blackmagicdesign.com/support

Blackmagic 3G-SDI Arduino Shield

Blackmagic Micro Studio Camera 4K supports PTZ output in the form of VISCA commands, which can be sent to a compatible motorized head. By using a Blackmagic 3G-SDI Arduino Shield, you can send pan, tilt and zoom commands over SDI to your Blackmagic Micro Studio Camera 4K. Your camera will then translate these SDI camera control protocol commands into the VISCA protocol, and send them to a compatible motorized head via the 9-pin connector on the expansion cable labelled 'PTZ control'.

This means that you can use one SDI cable in a live production environment, to send camera control commands to remotely control any setting in the camera, as well as send PTZ commands to a compatible motorized head to control pan and tilt. The pan and tilt commands will be sent by your Blackmagic Micro Studio Camera 4K to the motorized head, whereas lens related commands such as iris, focus and zoom commands will be sent to the active lens that is connected to the camera.

The commands that the Micro Studio Camera 4K can accept over SDI are:

- Lens Zoom
- Lens Focus
- Lens Iris
- Pan Tilt
- Memory Set
- Memory Recall
- Memory Reset

These commands are referenced in the ‘Blackmagic SDI Control Protocol’ in the ‘Blackmagic Camera Control’ developer information document which can be downloaded at www.blackmagicdesign.com/developer/

Most PTZ heads support the setting and recalling of their positions but it is a good idea to check which commands are supported by each PTZ head manufacturer.

The commands that are output through the ‘PTZ control’ connector in the form of VISCA commands are:

- CAM_Memory
- Pan-tiltDrive

VISCA commands

Pan-tiltDrive	Up	8x 01 06 01 VV WW 03 01 FF	**VV:** Pan speed 01 to 18 **WW:** Tilt speed 01 to 17 **YYYY:** Pan position F725 to 08DB (center 0000) **ZZZZ:** Tilt position FE70 to 04B0 (image flip: OFF) (center 0000) Tilt position FB50 to 0190 (image flip: ON) (center 0000)
	Down	8x 01 06 01 VV WW 03 02 FF	
	Left	8x 01 06 01 VV WW 01 03 FF	
	Right	8x 01 06 01 VV WW 02 03 FF	
	UpLeft	8x 01 06 01 VV WW 01 01 FF	
	UpRight	8x 01 06 01 VV WW 02 01 FF	
	DownLeft	8x 01 06 01 VV WW 01 02 FF	
	DownRight	8x 01 06 01 VV WW 02 02 FF	
	Stop	8x 01 06 01 VV WW 03 03 FF	
	AbsolutePosition	8x 01 06 02 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	
	RelativePosition	8x 01 06 03 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	
	Home	0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	
	Reset	8x 01 06 05 FF	
CAM_Memory	Reset	8x 01 04 3F 00 0p FF	**p:** Memory number (=0 to 5) Corresponds to 1 to 6 on the remote commander.
	Set	8x 01 04 3F 01 0p FF	
	Recall	8x 01 04 3F 02 0p FF	

Compatible motorized heads include the following:

- KXWell KT-PH180BMD
- PTZOptics PT-Broadcaster
- RUSHWORKS PTX Model 1

Controlling Pan, Tilt and Zoom with Blackmagic 3G-SDI Arduino Shield

Using the Blackmagic 3G-SDI Arduino Shield with an Arduino board, a joystick and a switch, you can control a PTZ head via Blackmagic Micro Studio Camera 4K.

Connecting your Blackmagic Micro Studio Camera 4K to the Blackmagic Design 3G-SDI Shield

1. Connect the Blackmagic Design 3G-SDI Shield to an Arduino board.
2. Connect the custom shield to the Arduino board.
3. Attach the SDI output connector from the shield to the SDI input on your Blackmagic Micro Studio Camera 4K and set the camera as camera number 1.
4. Connect the joystick and button to the shield

The joystick is mapped as follows:

- X axis adjusts the PTZ head's pan.
- Y axis adjusts the PTZ head's tilt.
- Pressing the joystick button tells the PTZ head to store the current X, Y position in memory.
- Pressing the switch recalls the stored position.

NOTE The ATEM SDK supports the Blackmagic SDI Camera Control Protocol, and is an alternative to using a Blackmagic 3G-SDI Arduino Shield for control.

Refer to the ATEM Switchers SDK manual for more information. The ATEM Switchers SDK manual can be downloaded at www.blackmagicdesign.com/support.

Controlling your Arduino

The following sketch demonstrates a simple example of using a joystick and button with an Arduino board and the Blackmagic 3G-SDI Arduino Shield, to control a PTZ head via a Blackmagic Micro Studio Camera 4K.

PTZ_Example §

```
#include <BMDSDIControl.h>

/**
 *  Blackmagic Design 3G-SDI Shield Example Sketch
 *
 *  This sketch demonstrates using a custom shield which contains a joystick and buttons, to control a connected camera.
 *
 *  The joystick is mapped as follows:
 *    - X axis adjusts the horizontal movement of the pan tilt head
 *    - Y axis adjusts the vertical movement of the  pan tilt head
 *
 *  The three buttons are mapped as follows:
 *    - Button 1, stores the current pan tilt position in memory location 1
 *    - Button 2, recalls the saved pan tilt position from memory location 1
 *    - Button 3, resets the pan tilt position from memory location 1
 *
 *  Setup Steps:
 *    1) Connect the Blackmagic Design 3G-SDI Shield to an Arduino board.
 *    2) Connect the custom shield to the Arduino board.
 *    3) Attach a camera's SDI input connector to the output SDI connector of
 *       the shield. Configure the camera as camera number 1.
 *    4) Connect the 9-pin PTZ Control connector to the VISCA input of the PTZ head being controlled
 *    4) Build and run the example sketch.
 */

// Hardware pin mappings
const int             joystickXPin         = A2;
const int             joystickYPin         = A1;
const int             button1Pin           = 5;
const int             button2Pin           = 6;
const int             button3Pin           = 7;

// Blackmagic Design SDI control shield globals
const int                shieldAddress = 0x6E;
BMD_SDICameraControl_I2C  sdiCameraControl(shieldAddress);

// Button debouncing globals
unsigned long         lastStableButtonTime[32];
int                   rawButtonLevels[32];
int                   stableButtonLevels[32];

float panTiltValues[] = {1.0, 1.0};

void setup() {
  // Configure digital inputs
  pinMode(button1Pin       , INPUT_PULLUP);
  pinMode(button2Pin       , INPUT_PULLUP);
  pinMode(button3Pin       , INPUT_PULLUP);

  // Set up the BMD SDI control library
  sdiCameraControl.begin();

  // The shield supports up to 400KHz, use faster
  // I2C speed to reduce latency
  Wire.setClock(400000);

  // Enable both tally and control overrides
  sdiCameraControl.setOverride(true);
}

void loop() {

  if (getButtonStableEdge(button1Pin) == true) {
        int8_t memoryValues[] = {
            1, // Store memory
            0, // First slot
          };

  sdiCameraControl.writeCommandInt8(
      1,
      11,
      1,
      0,
      memoryValues
    );

  }

    if (getButtonStableEdge(button2Pin) == true) {
        int8_t memoryValues[] = {
            2, // Recall memory
            0, // Second slot
          };
     sdiCameraControl.writeCommandInt8(
      1,
      11,
      1,
      0,
      memoryValues
    );

  }
```


PTZ_Example §
if (getButtonStableEdge(button3Pin) == true) {
int8_t memoryValues[] = {
0, // Reset memory
0, // First slot
};
sdiCameraControl.writeCommandInt8(
1,
11,
1,
0,
memoryValues
);
}
float panTiltValues[] = {0.0, 0.0};
int currentJoystickY = getJoystickAxisPercent(joystickYPin);
if (currentJoystickY > 15 || currentJoystickY < -15) {
panTiltValues[0] = (float)currentJoystickY / 100.0;
}
int currentJoystickX = getJoystickAxisPercent(joystickXPin);
if (currentJoystickX > 15 || currentJoystickX < -15) {
panTiltValues[1] = (float)currentJoystickX / 100.0;
}
sdiCameraControl.writeCommandFixed16(
1, // Destination: Camera 1
11, // Category: External Device
0, // Param: Pan Tilt Speed
0, // Operation: Set Absolute,
panTiltValues // Values
);
}
int getJoystickAxisPercent(int analogPin) {
// Reads the joystick axis on the given analog pin as a [-100 - 100] scaled value
int rawAnalogValue = analogRead(analogPin);
int scaledAnalogValue = map(rawAnalogValue, 0, 1023, -100, 100);
// Consider values close to zero as zero, so that when the joystick is
// centered it reports zero even if it is slightly mis-aligned
if (abs(scaledAnalogValue) < 10) {
scaledAnalogValue = 0;
}
return scaledAnalogValue;
}
bool getButtonStableEdge(int digitalPin) {
// Detects debounced edges (i.e. presses and releases) of a button
bool previousLevel = stableButtonLevels[digitalPin];
bool newLevel = getButtonStableLevel(digitalPin);
return previousLevel != newLevel;
}
int getButtonStableLevel(int digitalPin) {
// Reads a digital pin and filters it, returning the stable button position
int pinLevel = digitalRead(digitalPin);
unsigned long currentTime = millis();
// If the button is rapidly changing (bouncing) during a press, keep
// resetting the last stable time count
if (pinLevel != rawButtonLevels[digitalPin]) {
lastStableButtonTime[digitalPin] = currentTime;
rawButtonLevels[digitalPin] = pinLevel;
}
// Once the button has been stable for
if ((currentTime - lastStableButtonTime[digitalPin]) > 20) {
stableButtonLevels[digitalPin] = pinLevel;
}
return stableButtonLevels[digitalPin];
}