Teledyne FLIR Boson & Boson+ MIPI

APPLICATION NOTE

Official Publication Date: December 2023

Document Number: 102-2013a-106 Rev 110

Information on this page is subject to change without notice
EAR Controlled: EAR99
This document Is subject to the U.S. Export Administration Regulations (EAR). Diversion contrary to U.S. law Is prohibited

Table of Contents

1 Introduction

For technical support, contact your applications engineering support contact, sales person, or visit the FLIR Support Center (web: http://flir.custhelp.com/).

1.1 Revision History

Version	Date	Comments
100	10/21/2022	Initial Release
110	12/31/2023	Boson+ MIPI release

1.2 Reference Documents

Ref Number	Document number	Comments
1	102-2013-43	Boson Engineering datasheet
2	102-2013-44	Boson Software IDD
3	102-2013-106	Boson MIPI I2C & API App Note

1.3 SCOPE

This application note describes the Boson MIPI video streaming feature, options, and physical interface as well as brief mention of the I2C interface supporting the Boson SDK. Detailed explanation of the required steps to send/receive the Boson SDK commands over the I2C interface is described in a separate application note; 102-2013-106.

2 Boson+ MIPI Description

Boson provides the option of a digital data protocol typical of a MIPI CSI v1.1, and DPHY 1.0 camera in a software configurable single or dual lane configuration.

1. The MIPI video channel consists of three differential pairs (D-PHY); a clock lane and two data lanes supporting one or two virtual channels of up to 14 parallel bits of data. The channel utilizes 1.2V logic levels. See Section 2.3 Electrical Interface for pin assignments.

2. The host supplied fixed MIPI DDR clock rate is 246 MHz, either continuous or discontinuous,

3. Frame synch input requires 1.8V 10usec width pixel minimum (rising edge),

4. Latency < 4.8ms

5. Pixel formats are; RAW8 (monochrome, post-AGC), 16-bit UYVY (color, post-AGC) and RAW14 (pre-AGC)

6. A single virtual channel can operate at 60Hz frame rate transmitting either,

   - Monochrome 8-bit
   - Monochrome 14-bit
   - Post-AGC 8-bit monochrome
   - Post-AGC 16-bit color YUV422 (UYVY)

7. In a dual virtual channel configuration, the maximum frame rate is limited to 30Hz per channel, transmitting either:

   - Monochrome 8-bit & Monochrome 14-bit
   - Post-AGC 16-bit color YUV422(UYVY) & Monochrome 8-bit
   - Post-AGC 16-bit color YUV422(UYVY) & Monochrome 14-bit

2.1 Boson+ Hardware Configuration

All Boson+ core configurations as of June 2023 natively support MIPI as a configurable option using the ***dvoSetOutputInterface(FLR_DVO_OUTPUT_INTERFACE_E .FLR_DVO_MIPI)*** SDK command or via the Boson Windows Application GUI v4.2. CMOS is the power up default DVO (digital video output) for all generic Boson+ part numbers. If MIPI is desired as a startup configuration, the setting can be written to the dynamic header to be enabled on boot on supported cameras.

Earlier versions of Boson or Boson+ without MIPI can be identified by the board inside the frame of the camera being blue or green, while later versions of Boson+ or Boson with MIPI capability can be identified by a black PCB.

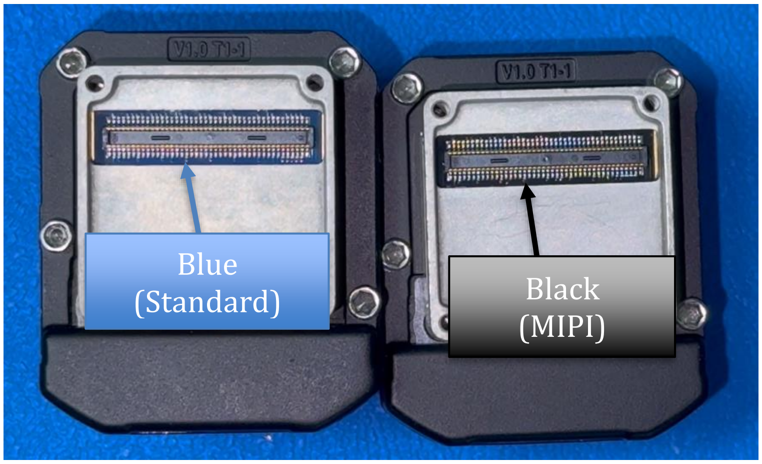
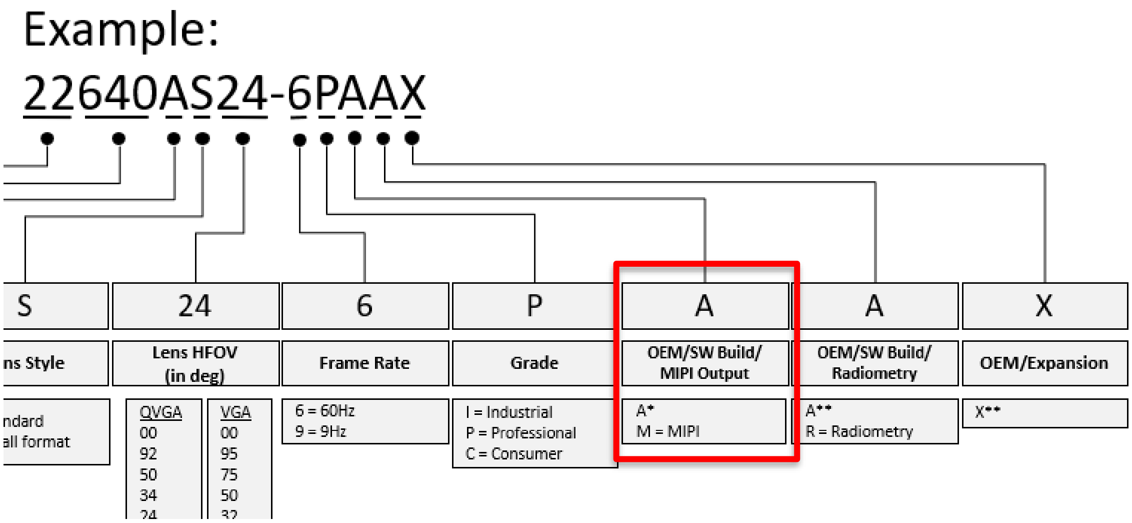


*Figure 1: **Left** - MIPI not supported. **Right** - MIPI supported*

2.2 Standard Boson Hardware Configuration



Example:

22640AS24-6PAAX

S	24	6	P	A	A	X
ns Style	Lens HFOV (in deg)	Frame Rate	Grade	OEM/SW Build/ MIPI Output	OEM/SW Build/ Radiometry	OEM/Expansion
ndard all format	QVGA 00 92 50 34 24 VGA 00 95 75 50 32	6 = 60Hz 9 = 9Hz	I = Industrial P = Professional C = Consumer	A* M = MIPI	A** R = Radiometry	X**

For standard boson part numbers, MIPI is only available if specified in the PN with the "M" in the third to last character of the part number. This is not necessary for Boson+ as of June 2023, as Boson+ cameras having 4.0 and newer software revisions.

MIPI enabled standard Bosons (not Boson+) have a specific camera software and hardware configuration that does not allow use of the CMOS interface. This is also applied to standard CMOS enabled bosons not being able to use the MIPI or I2C interface. Cameras cannot be upgraded to use a different interface than the configuration shipped from the factory.

2.3 Electrical Interface

To use the camera in the MIPI configuration, understand the following:

- 3.3V power is the only power input required to use the MIPI and i2c or UART interface
    - If use of USB is desired, 3.3V power and 5V VBUS input is required.
- i2c access would be available once the camera is booted, about 2.5 seconds after power is supplied.
- No external MCLK is required to use MIPI.
- Reset (Pin 24) should be left floating to allow for the camera to boot properly.

The pin pairs are shown below that are specific to the MIPI hardware configuration.

Table 1: MIPI Specific Pin Assignments

Connector Pin	Signal Name	Signal Description
42	MIPI_CLKP	MIPI Clock Positive
44	MIPI_CLKN	MIPI Clock Negative
35	MIPI DP0	MIPI Data Lane 0 Positive (VC0+)
37	MIPI DN0	MIPI Data Lane 0 Negative (VC0-)
34	MIPI DP1	MIPI Data Lane 1 Positive (VC1+)
36	MIPI DN1	MIPI Data Lane 1 Negative (VC1-)
63	i2c_SCL	i2c Clock
67	i2c_SDA	i2c Data

External pull-up resistors (2.2Kohm) are recommended on all I2C signals if the channel is utilized

Boson+ has a new board inside the camera that disables the 16-23bits on the CMOS interface in exchange for the pins to support MIPI

Pin assignments and description for the main connector are shown in Table 2 and Table 3. Any channels or signals which will not be used should be left floating (for example the reset pin can affect the boot sequence of the camera if not left floating).

Table 2: Boson Pin Assignments and Pin Description

PIN #	PIN NAME	SIGNAL TYPE	SIGNAL LEVEL	DESCRIPTION
1, 3, 5, 7, 10, 13, 19, 20, 29, 30, 39, 40, 49, 50, 59, 60, 69, 70, 79	DGND	Power	GND	Digital Ground
2, 4, 6, 8	3V3	Power	3.3V	Input Power
11	USB_D_P	Diff Pair	USB spec compliant	USB2 data+
9	USB_D_N	Diff Pair	USB spec compliant	USB2 data-
15	USB_VBUS	Power	USB spec compliant	USB VBUS (sense line only, not used to power the Boson core)
17	USB_ID	I/O	USB spec compliant	USB ID
14	USB_TX_P	Diff Pair	USB spec compliant	USB3 transmit+
12	USB_TX_N	Diff Pair	USB spec compliant	USB3 transmit-
18	USB_RX_P	Diff Pair	USB spec compliant	USB3 receive+
16	USB_RX_N	Diff Pair	USB spec compliant	USB3 receive-
21, 22, 23, 25, 26, 27, 28, 31, 32, 33, 34, 35, 36, 37, 38, 41, 42, 43, 44, 45, 46, 47, 48, 51, 52, 53, 54, 55, 56, 57, 58, 61, 62, 63, 64, 65, 66, 67, 68, 71, 73, 74, 75, 76, 77, 78	GPIO	I/O	1.8V	See Table 3
34, 35, 36, 37, 42, 44, 74, 76	MIPI	Diff Pairs	MIPI spec compliant 1.2V	MIPI transmit pairs See Table Below.

24	RESET	I/O	1.8V (asserted low)	See datasheet
72	EXT_SYNC	I/O	1.8V	See datasheet
80	No Connect		N/A	

Table 3: Assignment of GPIO Pins

Connector pin	Signal Name	Signal Description
72	Reserved (frame sync)	
23	GPIO	Discrete I/O pin 0
26	GPIO	Discrete I/O pin 1
33	uart_apb_sin	UART input
43	uart_apb_sout	UART output
41	cmos_data_13	CMOS bit13
21	cmos_data_14	CMOS bit14
38	cmos_data_15	CMOS bit15
34	cmos_data_16 mipi_data_P11	CMOS bit16 MIPI bit11P
22	cmos_data_17	CMOS bit17
42	cmos_data_18 mipi_clock_CP5	CMOS bit18 MIPI CLK
37	cmos_data_19 mipi_data_N10	CMOS bit19 MIPI bit10N
52	cmos_data_20	CMOS bit20 MIPI bit9P
54	cmos_data_21	CMOS bit21 MIPI bit9N
35	cmos_data_22 mipi_data_P10	CMOS bit22 MIPI bit10P
36	cmos_data_23 mipi_data_N11	CMOS bit23 MIPI bit11N
58	SYS_USE	See section 4.2
44	GPIO mipi_clock_CN5	Discrete I/O (unused) MIPI CLK
51	cmos_data_2	CMOS bit2
56	cmos_data_3	CMOS bit3
27	cmos_data_4	CMOS bit4
28	cmos_data_5	CMOS bit5
32	cmos_data_6	CMOS bit6
31	cmos_data_7	CMOS bit7
25	cmos_data_8	CMOS bit8
46	cmos_data_9	CMOS bit9
45	cmos_data_10	CMOS bit10
48	cmos_data_11	CMOS bit11
47	cmos_data_12	CMOS bit12
55	cmos_pclk	CMOS pixel clk
53	cmos_vsync	CMOS vsync
73	cmos_hsync	CMOS hsync
78	cmos_data_valid	CMOS data valid
77	cmos_data_0	CMOS bit0
62	cmos_data_1	CMOS bit1
63	i2c_scl	I2C Clk
67	i2c_sda	I2C Data
75	sd_clk	SD clk
66	sd_cmd	SD command/response
65	sd_dat_0	SD data0
68	sd_dat_1	SD data1
61	sd_dat_2	SD data2
64	sd_dat_3	SD data3
57	spi_mosi	SPI master-out slave-in
76	spi_miso	MIPI bit8P
74	spi_sclk_out	SPI clk
71	spi_ss_out_in_1	SPI chip select

2.3.1 External Sync

Boson comes configured by default with external sync disabled. When boson is configured in external sync slave mode, Boson expects external sync at 60 Hz rate . When the camera is configured for 60 fps operation (single VC) or 30 fps (dual VC, each VC at 30 fps), this implies a frame time of 16.67 milliseconds. Duty cycles can vary significantly, and active pulse width can be as low as 0.1ms. 1-8ms are acceptable active pulse widths. External sync is triggered on the rising edge.

The Boson MIPI camera generates its own clocks internally with an oscillator when in external sync disabled mode, and does NOT require MCLK from the host. The frames emitted by the camera over MIPI can be synchronized (i.e., the timing of frame arrival from the host MIPI Rx point-of-view) using the external sync signal, which is an input to the camera. Typically, the clocking on the host MIPI Rx is setup to correspond to the expected clock lane rate as presented by camera.

Please refer to the external sync app note for more information on how to use the external sync feature.

2.4 Video output modes

MIPI video is available as either a single video stream (single virtual channel (VC)) with maximum frame rate of 60 fps, or dual video streams (dual virtual channel) each limited to 30 fps.

Table 4: Single MIPI video stream MIPI options:

MIPI TYPE	DVO TYPE (Eng GUI)
RAW8 (postAGC monochrome video)	MONO8
UYVY (postAGC color video)	COLOR
RAW14 (NUC output)	MONO14

Note: All single-VC output modes use MIPI VC0.

Table 5: Dual MIPI video stream options:

MIPI TYPE [VC0 / VC1] *(Note: RAW8 is postAGC)*	DVO TYPE (Eng GUI)
RAW8 / RAW14	MONO8MONO14
UYVY / RAW14	COLORMONO14
UYVY / RAW8	COLORMONO8

Where the first mode is output on VC0, and the second mode is output on VC1.

2.4.1 Telemetry

For MIPI camera models, Telemetry is available as an additional top row of the digital output image (513 x 640), containing the same information and in the same formatting as used for CMOS video. Telemetry is also available as an independent meta-data stream. Telemetry can be either enabled or disabled.

16bit mode will populate the color and luma channels together in the telemetry format when in COLOR mode.

Telemetry encoding determines how the telemetry line is provided on the CMOS pins:

- **16b mode**: In this mode, the factory-default, each telemetry datum is provided as a 16b word
- **8b mode**: In this mode, each telemetry datum is provided as two consecutive bytes (big-endian order) on each pixel. Consequently, twice as many clock periods are required to transmit the data, so each datum takes up twice as many "pixels" compared to 16b mode.
- **8b-swapped mode ("Y"mode)**: This mode is identical to 8b mode except bytes are provided in little-endian order (least-significant byte provided first).

2.5 MIPI Video-Tap Modes

There are multiple locations in the signal pipeline where video can be tapped for output on the MIPI channel. MIPI specific details can be found in the Boson MIPI application note; 102-2013a-106. Boson provides the following MIPI video-tap modes:

- **Pre-AGC (14-bit)**: The output is linearly proportional to incident irradiance; output resolution is the same as FPA resolution (e.g., 320x256 or 640x512). Note that AGC settings, zoom settings, and color-encoding settings have no effect on the output signal at this tap point. Note that the Radiometry Modes described in the engineering datasheet affect the Pre-AGC output tap.

- **Radiometry Enabled Pre-AGC (14-bit) (factory default with Radiometry enabled)**: On Radiometry enabled cameras the T-stable or T-linear tap point is available. See *Figure 2* below, as well as datasheet. The ***T-linear conversion from 14-bit image to temperature in Kelvin is counts*/25** rather than *counts*/100

- **Post-AGC / Pre-Zoom (8-bit) (factory default)**: The output is contrast enhanced via the AGC algorithm; output resolution is the same as FPA resolution. Note that zoom settings and color-encoding settings have no effect on the output signal at this tap point.

Post-Zoom, Post-Colorize (various bit-width options depending upon color-encoding mode, see datasheet): The output is stretched to 640x512 resolution regardless of array format, and the displayed field of view is a function of the user-specified zoom parameters. The output is transformed to color space using the specified color palette (see datasheet) and specified color encoding mode (see datasheet).

NOTE: USB and the digital output modes CMOS video or MIPI video can access different tap points simultaneously. This is especially useful when Radiometric and Post-AGC or Post-Colorized data are required simultaneously.

Boson provides the following color-encoding modes which affect formatting of the output video signal when the post-colorize tap is selected:

- **YCbCr 4:2:2 (16-bit per pixel) (factory-default)**: The signal consists of a luminance channel (8 bits per clock) a blue chrominance channel (8 bits on each even clock cycle) and a red chrominance channel (8 bits on each odd clock cycle).

  For reference, the color encoding is depicted graphically in **Error! Reference source not found.**. Note the MIPI color-encoding mode has no effect on the video signal unless MIPI video-tap mode is "post colorize".

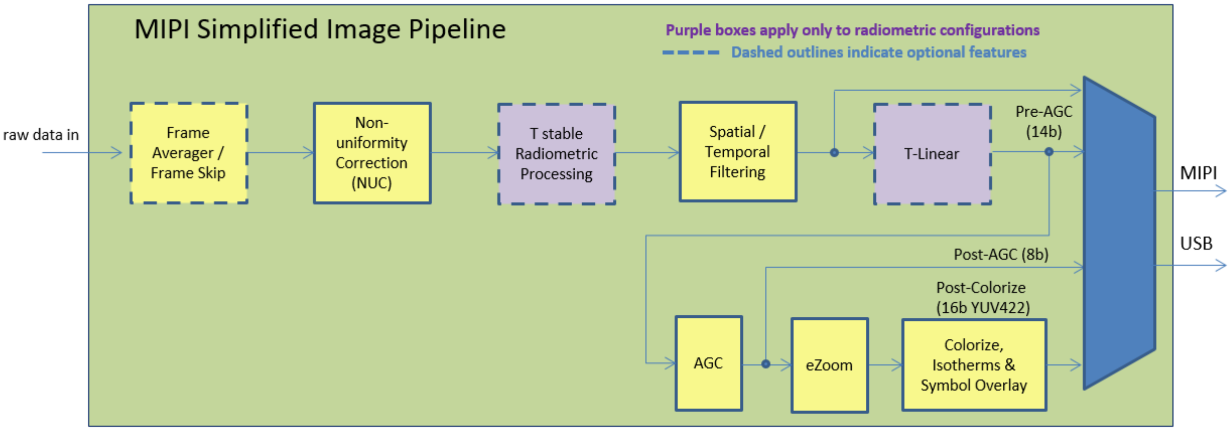

MIPI Simplified Image Pipeline
Purple boxes apply only to radiometric configurations
Dashed outlines indicate optional features
raw data in
Frame Averager / Frame Skip
Non-uniformity Correction (NUC)
T stable Radiometric Processing
Spatial / Temporal Filtering
T-Linear
Pre-AGC (14b)
MIPI
USB
Post-AGC (8b)
Post-Colorize (16b YUV422)
AGC
eZoom
Colorize, Isotherms & Symbol Overlay

Figure 2: Boson Signal Pipeline – MIPI OUTPUT ENABLED

2.6 FLIR MIPI implementation additional information

2.6.1 Start of frame and end of frame encoding

The SOF and EOF are used to delimit the lines that belong to a frame. Like the long packets used to convey the lines of pixels, the SOF and EOF include a data ID field, which has a virtual circuit (VC) ID, and data type (DT) subfields. So if two VCs are in use, the frames belonging to VC0 or VC1 can be distinguished. The frame duration is measured from SOF to SOF, and at 60 fps, is approximately 16.7 milliseconds. There is a gap between the EOF and a subsequent SOF that is counted as part of the vertical blanking time.

2.6.2 MIPI Data types used by the boson:

For UYVY, the data type == 0x1E.
For RAW8, the data type == 0x2A.
For RAW14, the data type == 0x2C.
If telemetry is used, in can be configured to use the same data type as the video pixels, or can be configured to use data type == 0x12.

2.6.3 Factory Default configuration:

To stream MIPI video, a new Boson+ from the factory will need to be configured to enable the output.

The minimum command to start streaming using the MIPI channel are below:

1. ***dvoSetOutputInterface(FLR_DVO_OUTPUT_INTERFACE_E .FLR_DVO_MIPI)***
2. **dvoSetMipiState(FLR_DVO_MIPI_STATE_E .FLR_DVO_MIPI_STATE_ACTIVE)**

A sequence of commands is necessary for the camera to start transmitting MIPI video in formats not in the default configuration. The camera will start in the OFF state (i.e., NOT transmitting video). The output format will default to 14-bit pre-AGC monochrome data, with telemetry disabled.

The enumeration of the states used for MipiStartState are OFF, PAUSED, and ACTIVE. END is just the terminating value of the enumeration and can be ignored. Please refer to section

3.1 The boson MIPI interface state machine: for more information on configuration considerations for the MIPI interface.

2.6.4 MIPI detailed signal description

The MIPI interface toggles back and forth between LP and HS at a high rate. The two signals in each data lane are used for single-ended signaling in LP state (1.2V). In HS state, the two signals in each data lane are used for differential signaling (+/- 100mV centered at 200mV). There are two sets of transmitters and receivers (one set for each type signaling), and they are switched between at a high rate. The LP signals are unterminated, and the HS signals are terminated.

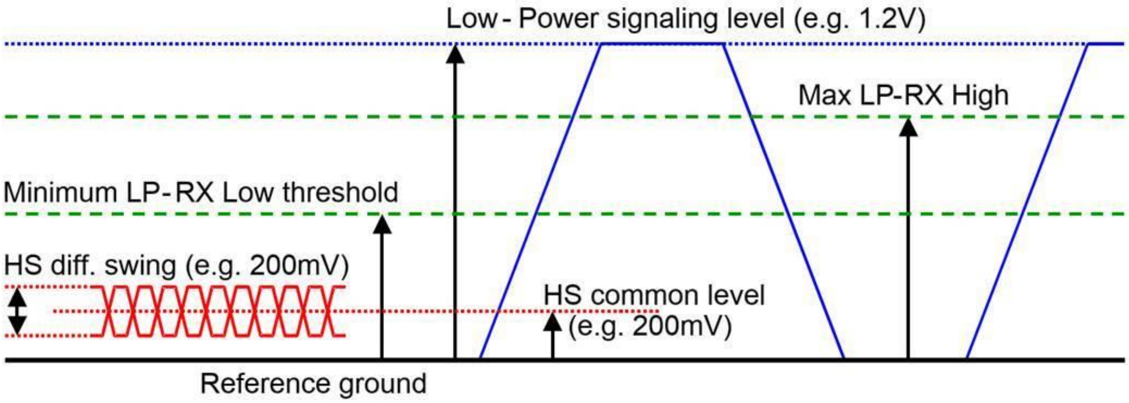

Low - Power signaling level (e.g. 1.2V)
Max LP-RX High
Minimum LP-RX Low threshold
HS diff. swing (e.g. 200mV)
HS common level
(e.g. 200mV)
Reference ground

Figure 3: MIPI LP and HS Logic Levels

In HS state, the signaling is simplex from camera towards host (i.e. the host is a passive receiver). The primary use of the LP state is to signal to the host the entry and exit from HS state. Again, the LP signaling is simplex from camera towards host. Although it is possible to use the LP state for half-duplex signaling, we do not support it, nor do most MIPI products. (The half duplex signaling could be used to convey short control messages from host to camera). Instead, all camera control is effected by host to camera via an out-of-band I2C channel (which is also specified by the MIPI standards).

In the application described below, the customer wishes to use a host SoC that has two MIPI inputs, one for a visible camera, and one for the Boson MIPI. In the case of the Boson MIPI, the host may receive either one or two video streams (when two streams are present, usually one is human viewable (VC0), and the second contains raw radiometric data

(VC1)). The use of MIPI inputs for both cameras simplifies the compositing of the two camera outputs on the host.

the HS state is used to send messages from the camera to the host, such as FRAME START, FRAME END, and LINE. The LINE message contains either a telemetry line or video line. So a video frame would be transmitted from camera to host as a sequence of FRAME START message, 512 (or 513 with telemetry enabled) LINE messages, and a terminating FRAME END message. The LINE message contains all the pixels for a single video line (i.e. the number of pixels in the LINE is equal to the horizontal width of 640). The data lane transitions from LP to HS before each HS message is sent, and from HS to LP after each message is sent. So, the toggling between LP and HS, and HS to LP, occurs thousands of times per second.

2.6.4.1 *What is dvoSetMipiClockLaneMode() ?*

These additional commands are not needed in most circumstances. For example, dvoSetMipiClockLaneMode() can be used to change the clock lane behavior from a continuous clock to a discontinuous clock (i.e., clock is only transmitted during high speed (HS) transmission used for video transmission). By default the Boson MIPI uses a continuous clock on the clock lane, which means the clock is always transmitted, even when video is not being transmitted.

The choice of clock lane behavior is dependent on the host MIPI Rx requirements.

2.7 Timing: Vertical and horizontal blanking or line timing

The horizontal line timing including blanking is 31.7us and the vertical blanking of frames is about 300us.

The line times are slightly less than 32 microseconds, and the lines are sent in a rolling shutter fashion, i.e. they are evenly distributed across frame time). There is a significant vertical blanking period between frame end message and subsequent frame start message of > 300 microseconds. The amount of horizontal blanking for each line varies depending on the pixel type in use for a given VC, but the total line time is approximately the same.

MIPI cameras operate in one-shot mode, not continuous mode so there can be a small amount of jitter on these signals and when they come through the video interface.

The final stage in a non-MIPI Boson signal pipeline is a multi-frame buffer, while for MIPI Boson cameras adhering to the MIPI CSI standard there is no multi-frame buffer, and no continuous mode option as described for CMOS output (datasheet). Instead MIPI always operates in the equivalent of a low latency (< 4.8msec) one-shot mode with the video pipeline serving as the software trigger. Consequently, the number of clocks per frame period is not a constant but can instead vary slightly. In this mode of operation, every frame is unique. This is preferred behavior for interfacing to a frame grabber able to tolerate slight frame-rate jitter or when in external-sync (slave) mode. That is, the output frame rate is equal to the effective frame rate.

NOTE: The MIPI Output mode has no effect on the USB video channel. The USB video channel always operates in one-shot mode.

3 Boson MIPI & Boson+ I2C Description

The Boson SDK has traditionally only been supported over the camera’s native UART and USB Serial Comm interfaces. Code and examples have been provided in the published SDKs to allow users to easily integrate these interfaces for both Python and embedded C host environments.

Compatible versions of Boson and Boson+ have added support for handling Boson SDK APIs using FSLP over I2C for Python host environments. Refer to the Boson MIPI I2C API application note 102-2013-106 application note for details for extending feature to a different embedded environment.

It takes about 2.5 seconds for the camera to boot once power is applied. Commands should not be sent until the camera is booted.

The choice of whether to embed Boson MIPI commands into a camera driver is a design decision. Many camera drivers do embed command sequences so that various sequences of commands are issued via I2C when certain driver functions are called. For example, if a camera is configured to boot up without video transmission enabled, then a “start streaming” function in the driver would send the commands to the camera that are necessary to start the video streaming. Some visible light cameras don’t have onboard flash storage, so the entire register set of the camera is written when the camera is initialized, or when the camera resolution is changed.

Alternatively, the camera may be controlled “out-of-band” by sending commands directly through the appropriate I2C interface. In this case, the application would issue I2C commands directly as needed, usually at points where the video configuration is being manipulated (such as through a V4L2 interface). The Boson MIPI SDK includes a Python library, along with language bindings, that can be used in this way.

A hybrid approach where some commands are embedded into the camera driver, and some commands are issued “out-of-band” is also possible.

It is typically not necessary to implement all the API commands provided by the Boson MIPI camera. The Boson MIPI API includes many commands that are only used during development. The actual number of API commands needed to control the camera in a production situation is usually quite small.

3.1 The boson MIPI interface state machine:

The Boson MIPI camera implements the Master side of the MIPI link, and is responsible for clock transmission via the clock lane, and all data transmission on its two data lanes. The device to which the camera is attached by the MIPI interface implements the Slave side of the MIPI link, and is responsible for following the link state, and the reception of any MIPI messages transmitted by the Master. The behavior of the Boson MIPI transmitter is controlled through a state machine that facilitates the management of the MIPI link between the Boson MIPI transmitter and the slaved MIPI receiver. The Boson Must be configured into DVO mode MIPI using the SDK command: ***dvoSetOutputInterface(FLR_DVO_OUTPUT_INTERFACE_E .FLR_DVO_MIPI)*** or via the Boson+ v4.2 Windows Application GUI. If desired as a startup configuration, the setting can be written to the dynamic header to be enabled on boot.

The state machine has three states: 1) OFF; 2) PAUSED; and 3) ACTIVE. The starting state for the MIPI state machine following boot is stored in flash, and can be changed by the user. For example, if the starting state stored in flash is ACTIVE, then following boot the MIPI state machine will enter the ACTIVE state, so that the camera comes up with active MIPI transmission. At run-time, the MIPI interface can be switched between states via API commands sent from the Slave side via I2C.

In the OFF state, the MIPI transmitter is disconnected, so that the MIPI clock and data lane signals are floating. In the ACTIVE state, the MIPI transmitter is connected, so that the MIPI clock lane and data lane signals are active, and transmission of MIPI frames occurs. During active transmission, the MIPI signals transition between LP states and HS transmission per the D-PHY standard. In the PAUSED state, transmission of MIPI frames is halted, and the data lane signals remain in LP11 state. Transitions between any pair of states are supported.

Depending on the user-selected configuration, the MIPI link may carry a single virtual channel (VC), or two VCs. Each VC corresponds to a video stream. When a dual VC configuration is in use, any state transition to or from the ACTIVE state will start or stop MIPI frame transmission on both VCs at the same time. When a dual VC configuration is in use, and per-frame telemetry is enabled, frame synchronization between the VCs may be obtained from the frame counter present in the telemetry.

The API command dvoSetMipiState() is used to initiate a MIPI state machine transition at run-time. The current MIPI state machine state may be determined by the API command dvoGetMipiState(). The API command dvoSetMipiStartState() is used to set MIPI state machine start state, and the new MIPI state machine start state is persisted by the API command bosonWriteDynamicHeaderToFlash(). The current MIPI state machine start state may be determined by the API command dvoGetMipiStartState().
There is an important special case where the MIPI state machine will transition as a side effect of another API command. When the API command dvoSetType() is used to select a new VC configuration for MIPI, the MIPI state machine will automatically be forced to the OFF state. The user must follow the dvoSetType() command with a dvoSetMipiState() command to transition the MIPI state to either PAUSED or ACTIVE. This two-step command sequence is necessary to effect a clean shutdown and startup of the MIPI link. Otherwise, MIPI link errors may occur at the MIPI receiver. The suggested MIPI link VC reconfiguration sequence is:

1. halt the MIPI receiver;
2. issue the dvoSetType() command to the camera to effect the desired configuration;
3. reconfigure the MIPI receiver for the expected video stream(s);
4. start the MIPI receiver; and
5. issue the dvoSetMipiState() command to transition to ACTIVE state.

With some MIPI receivers, it may be necessary to issue two dvoSetMipiState() commands, to transition through the PAUSED state, followed by a delay, then on to the ACTIVE state. This is because some MIPI receivers must receive a LP11 state for a minimum period of time before receipt of frames may occur. Note that changes to the VC configuration of the MIPI link can also be persisted with the API command: bosonWriteDynamicHeaderToFlash().

3.2 Example Startup command sequence:

To achieve the MODE10 -- YUV 4:2:2 (UYVY), 640x513, no lines telemetry configuration using VC0:

1. *Power the camera up and wait >2.5 seconds.*
2. ***dvoSetOutputInterface(FLR_DVO_OUTPUT_INTERFACE_E .FLR_DVO_MIPI)***
3. ***dvoSetMipiState(**FLR_DVO_MIPI_STATE_E **.FLR_DVO_MIPI_STATE_OFF)***
4. ***telemetrySetState(FLR_ENABLE_E.FLR_ENABLE)*** – disabled
5. ***telemetrySetPacking(FLR_TELEMETRY_PACKING_E. FLR_TELEMETRY_PACKING_DEFAULT)*** – Color for Ycbcr, Y or 8-bit are other options
    a. ***telemetrySetPackingVC1()*** - would be used in the case of changing VC1 changes
6. ***telemetrySetMipiEmbeddedDataTag(FLR_ENABLE_E.FLR_DISABLE)*** – enabled for 640x512 no lines telemetry (uses the 0x12 embedded data tag)
    a. ***telemetrySetMipiEmbeddedDataTagVC1(FLR_ENABLE_E.FLR_DISABLE)*** would be used for VC1 changes
7. ***dvoSetOutputFormat(FLR_DVO_OUTPUT_FORMAT_E.FLR_DVO_YCBCR)***
    a. ***dvoSetOutputFormatVC1()*** used for VC1
8. ***dvoSetType(FLR_DVO_TYPE_E.FLR_DVO_TYPE_COLOR)***
9. ***dvoApplyCustomSettings()***
10. ***dvoSetMipiState(**FLR_DVO_MIPI_STATE_E **.FLR_DVO_MIPI_STATE_ACTIVE)***
11. *The camera should now be streaming video that in the specified format. Open the video port on the host device with a matching video format.*

SDK function IDs are described in the software IDD (SDK_Documentation), but also provided below as an example.

Command Name	Command ID	Description
dvoSetMipiStartState()	*0x00060022*	*to set the MIPI state machine start state*
dvoSetMipiState()	*0x00060024*	*to set the MIPI state machine state to OFF*
dvoSetType()	*0x0006000F*	*to set new VC0 output configuration*
dvoSetMipiState()	*0x00060024*	*to set the MIPI state machine state to ON*

3.3 Similarities to Visible camera API

Thermal cameras inherently operate differently than visible cameras for the most part because they are a less mature technology, but also because they operate in fundamentally different ways. For example, exposure time and white balance, gain, EV compensation are all terms that do not directly apply to microbolometers and their output. The Boson has a unique interface, and controls for controlling image characteristics are defined in very specific ways.

The 14-bit output from the camera would be considered more of a RAW output from the camera, before the AGC (14-> 8 bit transformation). The MONO8 output is a greyscale post-AGC output, while the colorized outputs available are post-color formatting, post-zoom, and post-on screen symbols.

AGC settings like gain and white balance for example would be similar to AGC parameters in the boson that would affect the MONO8 and COLOR output modes. Please refer to the AGC adjustments guide for more information on specifics regarding this topic. In general, it is best to leave these parameters alone during driver development, and to experiment using the USB capable Boson App for windows computers available on the web. Here, AGC settings can be tested and modified in real time with a simple GUI interface. If you would like to configure AGC settings with I2C commands, that can be done as well, but FLIR suggests familiarization first using the GUI.

3.4 On board non-volatile memory

JFFS2 memory is available to the user over the UART or I2C interface. MEMOPS and JFFS2 sections of the Software IDD depict commands to do memory operations. Files can be uploaded or read from the Boson with this feature.

4 Timeouts and Special Commands

4.1 Timeouts

Some API commands can cause a significant amount of delay before the camera can generate a response packet. Commands like:

- BOSON_WRITELENSNVFFCTOFLASH
- BOSON_WRITELENSSFFCTOFLASH
- BOSON_WRITELENSGAINTOFLASH
- BOSON_WRITEUSERBADPIXELSTOALLTABLES

- MEM_ERASEFLASH
- BOSON_REBOOT

Due to the nature of the I2C implementation on the Boson Camera, these commands will likely generate an error condition on the I2C bus, requiring the host to reset the bus before continuing operation. When issuing commands that require substantial execution time on the camera, it is recommended that the host perform an I2C bus reset every second until the camera becomes available.

5 MIPI Development Accessory Board PN 250-0853-00

A test board is available for development purposes to fan out several of the interfaces provided on the primary I/O connector. It is not intended to be integrated into final end-use products. Design and fabrication of an additional interface board is recommended for non-development use cases.

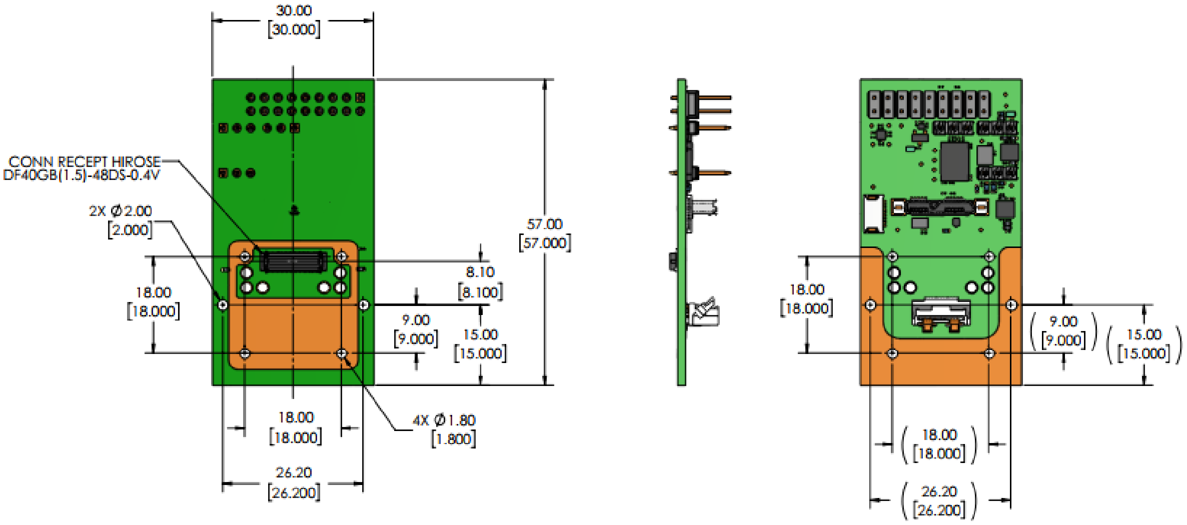


Figure 4: Test Board Dimensions (Rev 1)

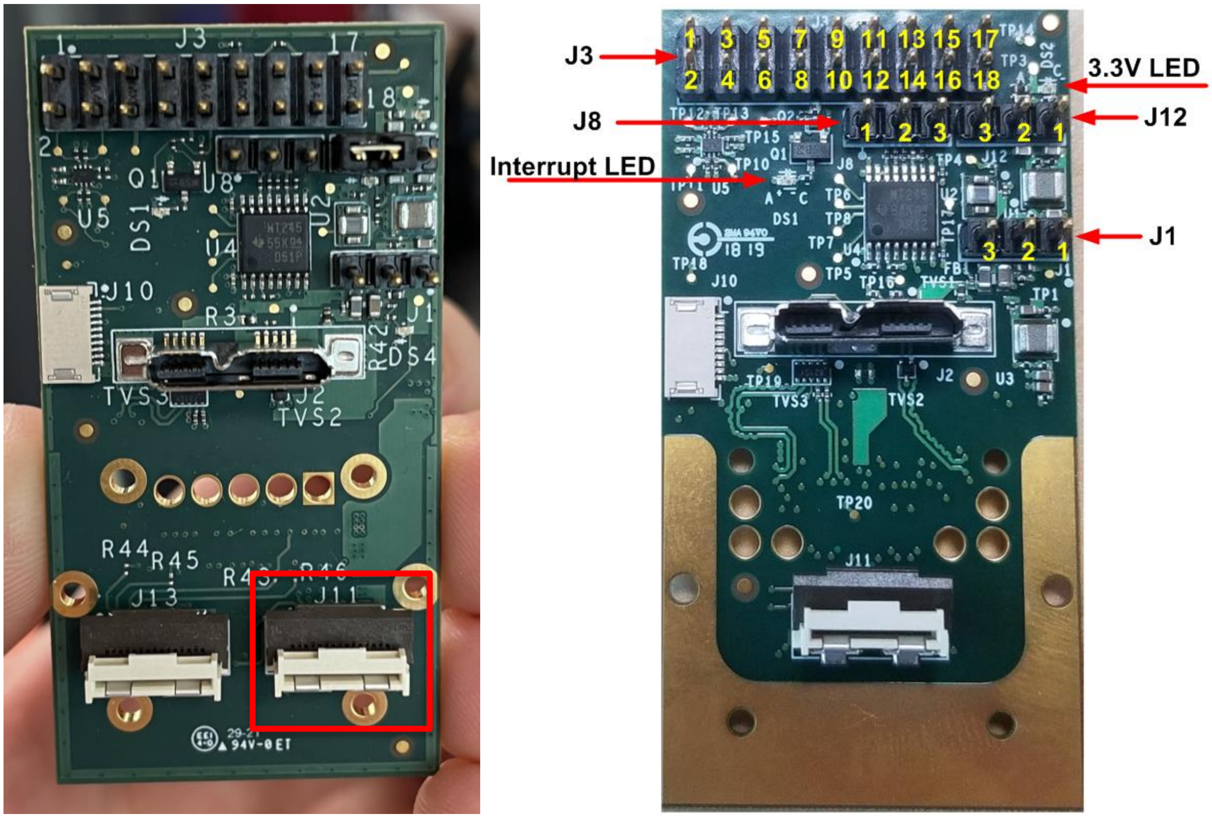


Figure 5: Test Board Pin Locations (Left is the new version)

5.1 Connectors

J2: USB 3.0 Micro B Connector.

- Power
- Communications with FLIR FLSP, Boson SDK, or Boson GUI
- Video with I420, NV12 (8-bit Colorized post AGC) or IR16 (16-bit)
- Compatible with Boson GUI on Windows 10

J11: MIPI/I2C Interface (red box figure 4)

- I2C at 1.8V logic levels
- MIPI Clock and Data Lanes 0 and 1
- 3.3V power can be supplied to the camera through this connector if the J1 and J12 jumpers are properly configured

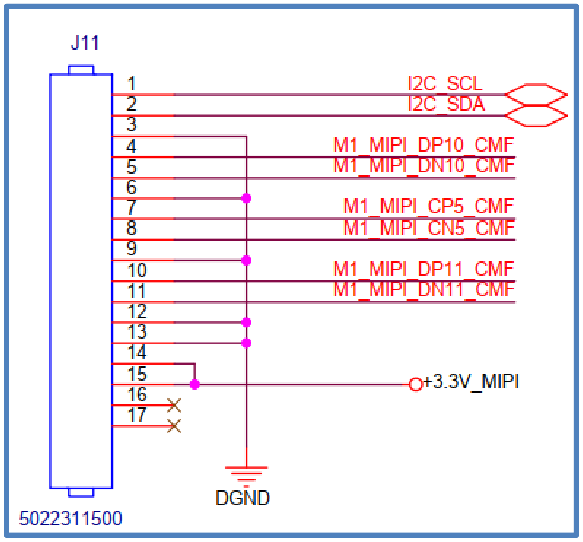


Figure 6: Test Board J11 Pin Definition

An example of a flex cable that can be used for this board is:

Digi-Key Part Number	WM25235-ND
Manufacturer	Molex
Manufacturer Product Number	0150211015
Description	LVDS 0.5MM 254MM 15CKTS AU 50224
Detailed Description	15 Position FFC, FPC Cable 0.020" (0.50mm) 10.0" (254.0mm)

Figure 7: 15 Position FFC, FPC Cable 0.020" (0.50mm) 10.0" (254.0mm) PN 0150211015

Additional interfaces exposed on J13 shown below:

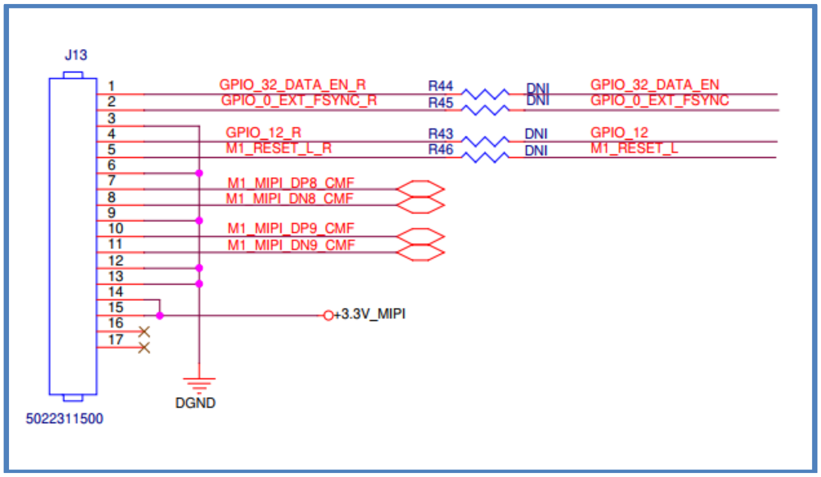


Figure 8: J13 connector interface

5.2 Jumper Settings and Description

J1: Selecting Power Source 1

- Connect Jumper to pins 1 and 2 for a USB Powered Camera.
- For an Externally Powered Camera connect a Lab Supply to pins 2 and 3.
  - Pin 2 = 5V
  - Pin 3 = GND
- This jumper does not have to be connected when this board is configured as a MIPI Powered Camera.
- When USB power is used, the camera must be powered on via USB prior to connecting the TX2.

J12: Selecting Power Source 2

- Connect Jumper to pins 1 and 2 to power the camera via USB or an External Power Supply.
- Connect Jumper to pins 2 and 3 to power the camera via 3.3V on J11.
- LED: A green LED will light with a valid Supply Input and Jumper (J1, J12) Configuration.

J8: UART Interface, 3.3V Logic Levels

- This interface is for 3.3V Logic Levels using an FTDI Cable.
  - Pin 1 = GND
  - Pin 2 = RX, Input
  - Pin 3 = TX, Output

J3: Discreet I/O

- **SYSTEM_FRAME_REF** Input, 1.8V Logic
  - Connect a Jumper to pins 1 and 3 to set the signal to a logic high.

  - Connect a Jumper to pins 3 and 5 to set the signal to a logic low.
  - When using a signal generator connect as shown.
    - Pin 3 = SYSTEM_FRAME_REF
    - Pin 5 = GND
  - Note: The Camera will image fine without a jumper placed in this location.

- **SHUTDOWN_L** Input, 1.8V Logic
  - Connect a Jumper to pins 7 and 9 to set the signal to a logic low.
  - Connect a Jumper to pins 9 and 11 to set the signal to a logic low thru a 100K resistor.
    - This 100K resistor is a System Strapping resistor required in the ICD. This is how it will be connected at System Level.
  - Note: The Camera will image fine without a jumper placed in this location.

- **SPARE_IO0**
  - The ICD Requires a Spare IO. This is not defined in Software yet.

- **I2C Interface**, 3.3V Logic Levels
  - This interface is for 3.3V Logic Levels using an Aardvark.
    - Pin 2 = SCL
    - Pin 4 = SDA
    - Pin 6 = GND

- **ROW_INTEGRATION,** Output, 1.8V Logic Levels
  - Pin 8 = ROW_INTEGRATION

- **INTERRUPT_L**, Output, 1.8V Logic Levels
  - Pin 10 = INTERRUPT_L
  - This signal includes a 100k Pull-Up System Strapping Resistor required by the ICD.
  - LED: A green LED will light when INTERRUPT_L is driven Low.

- **I2C_ADDRESS_LSB**, Input, 1.8V Logic
  - Connect a Jumper to pins 14 and 16 to set the signal to a logic high.
  - Connect a Jumper to pins 16 and 18 to set the signal to a logic low