MOBILE IMAGE PARAMETRIC SYSTEMS

IMAGE QUALITY AND AUTOMATED TEST

Image Acquisition Control Sequencer Command List

IMAGE QUALITY AND AUTOMATED TEST

Image Acquisition Control Sequencer Command List

Mobile Image Parametric Systems
www.mip-systems.com
support@mip-systems.com

Table of Contents

Purpose/Scope

The purpose of this document is to describe the Sequencer Command set used in the Image Acquisition Control application.

Image Acquisition Control Sequencer Command List

The commands listed below are supported by the IAC Sequencer.

COMMAND	PARAMETER	DESCRIPTION	
BAYER_FILTER_TYPE		Specifies the display filter used within the IAC application for BAYER image buffer type.	
	Filter Type	0 = BAYER RAW to 16-bit Image 1 = BAYER Red Plane to 16-bit Image 2 = BAYER Green-Red Plane to 16-bit Image 3 = BAYER Green-Blue Plane to 16-bit Image 4 = BAYER Blue Plane to 16-bit Image 5 = BAYER Byte 1 Bayer Interpolation to RGB Image 6 = BAYER Byte 2 Bayer Interpolation to RGB Image 7 = BAYER n-bits (Bit Depth) to RGB *Example:* **BAYER_FILTER_TYPE=5 ; use byte 1 to RGB**	
BAYER_PATTERN		Specifies which Bayer encoding is used in the Bayer encoded image data.	
	Pattern	0, 1, 2, or 3 0 = GBGB::RGRG 1 = GRGR::BGBG 2 = BGBG::GRGR 3 = RGRG::GBGB *Example:*	

		BAYER_PATTERN=1 ; use GRGR::BGBG	
BOUT_SET		Sets the state of the four IAC digital output lines.	
	Bit Output State	Specifies the bit number (0,1,2, or 3) to control True or False to specify the output sate *Example:* **BOUT_SET=0,False ; Set Bit 0 to False**	
CAPTURE_CONFIG		Configures the image capture parameters of the IAC hardware.	
	PixelDataSelect	0 = 16-Bit Word (bits 15:0) 1 = Low Byte (bits 7:0) 2 = High Byte (bits 15:8) 3 = 16-Bit Continuous Count	
	FrameStart	0 = VSYNC rising edge 1 = VSYNC falling edge	
	FrameEnd	0 = VSYNC rising edge 1 = VSYNC falling edge	
	XSYNC0	0 = disabled 1 = enabled active high 2 = enabled active low	
	XSYNC0	0 = disabled 1 = enabled active high 2 = enabled active low	
	XSYNC1	0 = disabled 1 = enabled active high 2 = enabled active low	
	XSYNC2	0 = disabled 1 = enabled active high 2 = enabled active low	
	ClockSelect	0 = PIXCLK 1 = REFCLK	
	ClockEdge	0 = rising edge	

		1 = falling edge *Example:* **CAPTURE_CONFIG=1,0,1,1,0,0,0,1** ; IAC capture hardware Low byte only ; Frame Start at Rising Edge ; Frame End at Falling Edge ; Xsync0 enable and active high ; Xsync1 disabled ; Xsync2 disabled ; Clock Select = Pixel Clock ; Clock Edge = Use falling edge of Pixel Clock	
CAPTURE_CONTINUOUS		Signals the Image Acquisition Control application software to continuously acquire, process, and display images from the image sensor. NOTE: This command does not actually cause the IAC hardware to acquire images. It is just a simple message or "application" command used to control the application software.	
	Enabled	0 = Continuous capture OFF 1 = Continuous capture ON *Example:* **CONTINUOUS_CAPTURE=1 ; enable**	
CAPTURE_MODE		Selects between Parallel and High-speed Serial image capture modes	
	CAPTURE_MODE	0 = Parallel image data capture 1= MIPI high-speed serial image data capture 2= SMIA high-speed serial image data capture *Example:* **CAPTURE_MODE=0 ; parallel mode**	

CLOCKS_PER_PIXEL		Specifies the number of clocks it takes to clock out one image pixel from the image sensor.	
	Clocks/pixel	(1, 2, 3, or 4) This value is dependent on the image sensor design. The IAC image buffers have the following pixel bit depths: RAW = 32 bits/pixel BAYER = 16 bits/pixel RGB = 32 bits/pixel YCbCr = 16 bits/pixel You cannot select clocks/pixel that will result in pixel bit depths that exceed the capabilities of the buffers. If an image sensor only clocks out 8-bits per clock then the clocks/pixel setting could potentially be set to 4 for an RGB or RAW image buffer type. If the image sensor clocks out 8 to 16 bits per clock then the clocks per pixel can only be set to 1 for BAYER or YCbCr image buffer types. *Example:* **CLOCKS_PER_PIXEL=2 ; Map data from 2** ; clocks to each pixel	
COUNTER_ARM		Arms the IAC on-board timer/counter. NOTE: There are no parameters for this command.	
		Example: **COUNTER_ARM ; Arm the event counter**	

COUNTER_CONFIG		Configures the IAC on-board timer/counter to measure the transitions of the specified "event parameter" over the period specified in the "gate select".	
	Gate Select	0 = Frame Window 1 = 1-second Timer	
	Event Parameter	0 = Data[0] lsb image sensor data bit. 1 = Data[1] 2 = Data[2] 3 = Data[3] 4 = Data[4] 5 = Data[5] 6 = Data[6] 7 = Data[7] 8 = Data[8] 9 = Data[9] 10 = Data[10] 11 = Data[11] 12 = Data[12] *Example:* **COUNTER_CONFIG=0,17 ; measure Hsync/Vsync**	13 = Data[13] 14 = Data[14] 15 = Data[15] 16 = VSYNC 17 = XSYNC0 18 = XSYNC1 19 = XSYNC2 20 = XSYNC3 21 = DIN0 22 = DIN1 23 = DIN2 24 = PIXCLK 25 = REFCLK
DELAY		Delay the sequence file execution	
	Delay	Seconds to delay *Example:*	

		DELAY=1.2 ; Delay 1.2 seconds	
DIALOG		Display a simple modal user dialog with an OK button.	
	Dialog Text	Text to display in the user dialog window *Example:* **DIALOG=PRESS OK TO CONTINUE**	
ENABLE_BITIO		Enables/Disables the FET switches which connect or disconnect the sensor Digital I/O lines to/from the IAC hardware.	
	ENABLED?	True or False indicating the state of the sensor digital I/O line FET switches. *Example:* **ENABLE_BITIO=True ; Connect the sensor SYNC lines to the IAC hardware.**	
EXTRACT_FROM_RGB		Specifies which Plane to extract from a RGB image.	
	Extract From RGB	0 = No Extraction (Pass RGB image through) 1 = Red 2 = Green 3 = Blue 4 = Hue 5 = Saturation 6 = Luminance *Example:* **EXTRACT_FROM_RGB=6 ;Luminance**	
FPGA_CLOCK_SELECT		Specifies which FPGA clock mode to use.	
	Mode	0=Oscillator clock doubler *Example:* **FPGA_CLOCK_SELECT=0 ; Select the oscillator clock doubler function of the FPGA based clock.**	

FPGA_RESET		Specifies the state of the FPGA Reset function.	
	STATE	1=FPGA in RESET State 0=FPGA normal operation *Example:* **FPGA_RESET=1 ; Hold the FPGA in RESET state.**	
HARDWARE_DECODE		Specifies the hardware decoding map used by the FPGA during image data acquisition.	
	DecodeMap	The decode map is a 16 character text string where each character in the string represents a bit in the 16-bit data word. The first character in the 16-character string represents the MSB, and if the intended destination of this bit is to remain in the most significant bit location then the DecodeMap string character should be set to “F”. If the intended destination of this bit is to switch it to the lower location or LSB, then the DecodeMap string character should be set to “0”. If the intended destination of this bit is one down from the MSB, the set the DecodeMap character to an “E”. The decode map character is the Hexadecimal equivalent to the bit number of the destination location. *Example:* **HARDWARE_DECODE=FEDCBA9876543210 ; use a straight through mapping (i.e. keep the original bit locations)**	
I2C__BUFFER_SEND		Sends all buffered I2C commands in a single I/O transaction.	

		This command has no parameters. If the I2C_BUFFER_START command is received then all subsequent I2C_BUS_WR commands are buffered on the host computer instead of being immediately sent to the sensor. Once a I2C_BUFFER_SEND (or I2C_BUFFER_WRITE) command is sent all the buffered I2C settings are sent at once greatly reducing the overall time to setup sensor registers. Use the I2C_BUFFER_STOP command to cease buffering I2C commands and resume sending each individual command separately. *Example:* I2C_BUFFER_SEND	
I2C__BUFFER_START		Clears the I2C command buffer.	
		This command has no parameters. If this I2C_BUFFER_START command is received then all subsequent I2C_BUS_WR commands are buffered on the host computer instead of being immediately sent to the sensor. Once a I2C_BUFFER_SEND (or I2C_BUFFER_WRITE) command is sent all the buffered I2C settings are sent at once greatly reducing the overall time to setup sensor registers. Use the I2C_BUFFER_STOP command to cease buffering I2C commands and resume sending each individual command separately. *Example:* **I2C_BUFFER_START**	
I2C__BUFFER_STOP		Stops the I2C command buffering process.	

		This command has no parameters. If the I2C_BUFFER_START command is received then all subsequent I2C_BUS_WR commands are buffered on the host computer instead of being immediately sent to the sensor. Once a I2C_BUFFER_SEND (or I2C_BUFFER_WRITE) command is sent all the buffered I2C settings are sent at once greatly reducing the overall time to setup sensor registers. Use the I2C_BUFFER_STOP command to cease buffering I2C commands and resume sending each individual command separately. *Example:* **I2C_BUFFER_STOP**	
I2C__BUS_CLOCK_FREQ		Specifies the clock rate for I2C communications.	
	Target Frequency	Specifies the frequency of the clock used for I2C communications *Example:* **I2C_BUS_CLOCK_FREQ=100k ; use 100KHz clock**	
I2C_BUS_CONFIG		Configures the I2C Bus Number, Bus Slew Rate, Bus Clock Rate, and Timeout used for I2C communications. Note: This function consolidates several functions above so that all I2C configuration can be handled in one step.	
	I2C Bus No I2C Bus Slew Rate	Specifies the bit number (1 or 2) to use for further I2C communications 0 = Enabled for High Speed (400KHz) 1= Disabled for Standard Speed (100KHz or 1MHz)	

	I2C Clock Rate I2C Timeout	Specifies the frequency of the clock used for I2C communications Specifies the timeout in msec used for I2C communications transactions. *Example:* **I2C_BUS_CONFIG=1,1,100k,100**	
I2C_BUS_NO		Specifies the use one of two possible I2C busses in the IAC available to users. I2C Bus 1 is typically used to communicate with the sensor. I2C Bus 2 is typically used to communicate with other devices on the sensor adapter board.	
	Bus Number	Specifies the bit number (1 or 2) to use for further I2C communications *Example:* **I2C_BUS_NO=1 ; Use I2C Bus 1 for I2C commands**	
I2C_BUS_WR		Perform an I2C Write Operation.	
	Transmit Data	The I2C Transmit buffer in Hex dot notation. *Example:* **I2C_BUS_WR=F0.00.02 ; Set register 0xF0 to 2.** ; F0 = 8-bit reg addr ; 16-bit reg value = 2	
I2C_BUS_WR_MSK		Perform an I2C Write Operation with Bit Mask.	
	Transmit Data	The I2C Transmit buffer in Hex dot notation. *Example:* **I2C_BUS_WR_MSK=F0.00.02,00.02** ; Set the second bit only of register 0xF0. ; F0 = 8-bit reg add ; 16-bit reg value = 00.02 ;16-bit mask value = 00.02	

I2C_BUS_WR_RD_FROM_FILE		Perform an I2C Write Operation direct from a file. The I2C Transmit buffer in continuous bytes, no delimiters used.	
	Filepath	An absolute file path to byte file. *Example:* **I2C_BUS_WR_RD_FROM_FILE=C:\myFiles\ByteLoad.txt**	
I2C_DEVICE_ADDRESS		Specifies the I2C Device Address used for further I2C communications	
	Device Address	A 2 digit hex code specifying the 7-bit device address used for I2C communications *Example:* **I2C_DECVICE_ADDRESS=BA ; Use hex addr 0xBA**	
I2C_SLEW_RATE		Specifies the slew rate mode for I2C communications.	
	Mode	0 = Enabled for High Speed (400KHz) 1= Disabled for Standard Speed (100KHz or 1MHz) *Example:* **I2C_SLEW_RATE=1 ; set slew rate for 100Khz clock.**	
IMAGE_BIT_DEPTH		Sets the bit depth of image for post processing of RAW or BAYER data. This is used by the Interpolation Method and other color conversion items. NOTE: This is not required for capturing image data.	
	Bit Depth	Bit Depth of image, range is 8 to 32 bits. *Example:* **IMAGE_BIT_DEPTH=10; set to 10 bits**	

IMAGE_COLUMNS		Sets the number of columns in the image. This value is used when converting the raw sensor data stream into an image.	
	Columns	Number of columns to use when reformatting the sensor data stream into an image *Example:* **IMAGE_COLUMNS=1280; set columns to 1280**	
IMAGE_ROWS		Sets the number of rows in the image. This value is used when converting the raw sensor data stream into an image.	
	Rows	Number of rows to use when reformatting the sensor data stream into an image *Example:* **IMAGE_ROWS=1024; set number of rows to 1024**	
IMAGE_TYPE		Specifies the IAC Image Buffer Type used for acquisition and display	
	Image Type	0= Raw 1= BAYER 2= RGB 3= YCbCr *Example:* **IMAGE_TYPE=BAYER ; use BAYER image buffer**	
INTERPOLATION_METHOD		Specifies the color conversion method for Bayer image data.	
	Interpolation Method	0= Bi-Linear 1= Gradient Based 2= Red Clear 3= Gradient for Red Clear Only	

		Example: **IINTERPOLATION_METHOD=1 ; use Gradient Based**	
LSB_ENABLE		Enables/Disables the FET switches which connect or disconnect the LSB sensor data connections from the IAC.	
	ENABLED?	True or False indicating the state of the sensor data LSB FET switches. *Example:* **LSB_ENABLE=True ; Connect the sensor LSB data lines to the IAC hardware**	
MSB_DIRECTION		Specifies the direction of the image sensor MSB data line buffers. By default the sensor data line buffers are configured as inputs, as the image sensor is usually the one driving those lines.	
	Direction	1=Output 0=Input *Example:* **MSB_DIRECTION=0 ; Configure the IAC MSB buffers as inputs from the image sensor.**	
MSB_ENABLE		Enables/Disables the FET switches which connect or disconnect the MSB sensor data connections from the IAC.	
	ENABLED?	True or False indicating the state of the sensor data MSB FET switches. *Example:* **MSB_ENABLE=True ; Connect the sensor MSB data lines to the IAC**	
OSCCLK_SET		Sets the output frequency of the IAC on-board oscillator clock.	
	On/Off	Turns the oscillator clock on or off	

	Frequency	Frequency setpoint for the IAC reference clock output. Note: a single scaling character G,M,K,k,,m,u,n, or p is allowed directly following the number. *Example:* **OSCCLK_SET=True, 54M**	
PIXEL_MASK		Specifies a 16-bit pixel mask applied to the image sensor data just before the data is captured into on-board memory.	
	Mask	A 4 character hexadecimal string representing the 16-bit mask value applied to the image sensor data. *Example:* **PIXEL_MASK=FFFF ; enable all 16-bits.**	
PS_ENABLE		Enables/Disables the specified power supply.	
	PS ID ENABLED?	ID of power supply (A, B, C, D, or E) True or False indicating the state of the Power Supply Enable. *Example:* **PS_ENABLE=E,True ; Enable PSE**	
PS_V_SET		Set the voltage output setpoint of an individual power supply	
	PS ID Vset	ID of power supply (A, B, C, D, or E) A floating point number between 0.0 and 4.0 that is used as the output setpoint for the specified power supply. *Example:* **PS_V_SET=E,2.8 ; Set PSE to 2.8 Volts**	
RAW_FILTER_TYPE		Specifies the display filter used within the IAC application for RAW image buffer type	
	Filter Type	0 = RAW SENSOR to 32-bit RGB Image 1 = RAW SENSOR Byte 1 to 8-bit Image	

		2 = RAW SENSOR Byte 2 to 8-bit Image 3 = RAW SENSOR Byte 3 to 8-bit Image 4 = RAW SENSOR Byte 4 to 8-bit Image 5 = RAW SENSOR Lower 16 bits to 16-bit Image 6 = RAW SENSOR Upper 16 bits to 16-bit Image *Example:* **RAW_FILTER_TYPE=5 ; map low word to 16-bit**	
REFCLK_SOURCE		Specifies which clock, the IAC on-board oscillator, or the IAC FPGA-based clock, should be routed to the image sensor as the REFCLK.	
	Source	0=Oscillator clock 1=FPGA Clock 2=None/External *Example:* **REFCLK_SOURCE=0 ; Use the IAC on-board oscillator as the REFCLK routed to the image sensor.**	
RGB_FILTER_TYPE		Specifies the display filter used within the IAC application for RGB image buffer type.	
	Filter Type	0 = RGB RAW to RGB 1 = RGB Red 2 = RGB Green 3 = RGB Blue 4 = RGB Hue 5 = RGB Saturation 6 = RGB Luminance *Example:* **RGB_FILTER_TYPE=0 ; use full RGB image**	

ROI_DESCRIPTION		Specifies the Region of Interest (ROI) used during image capture	
	ROI	A string containing the comma separated coordinates of the left, top, right, and bottom of the region of interest (ROI). *Example:* **ROI_DESCRIPTION=100,100,200,200) ; use a square ROI 100 pixels by 100 pixels with the top left corner of the ROI square at location (100,100).**	
ROI_ENABLE		Enables or Disables Region of Interest (ROI) during image capture.	
	Enable	0= Disabled 1=Enabled *Example:* **ROI_ENABLE=1 ; Enable Region of Interest Capture Mode.**	
SCRIPT		Execute the specified script	
	File Name	Name of the script file to execute. File must be in the same directory as the calling script file. *Example:* **SCRIPT=POWER OFF.txt**	
SENSOR_COMM_ENABLE		Enables/Disables the FET switches which connect or disconnect the sensor communications connections to/from the IAC hardware.	
	ENABLED?	True or False indicating the state of the sensor communications FET switches. *Example:*	

		SENSOR_COMM_ENABLE=True ; Connect the sensor communication lines to the IAC hardware.	
SENSOR_COMM_TYPE		Specifies the type of sensor communications to be used. Either I2C or SPI communication protocol can be used for sensor communications.	
	Type	0=I2C 1=SPI *Example:* **SENSOR_COMM_TYPE=0 ; Use the I2C protocol for sensor communications.**	
SENSOR_CONNECT		Enables/Disables the FET switches which connect or disconnect the sensor data, sync, and I/O connections from the IAC hardware.	
	ENABLED?	NOTE: This command is the equivalent of performing the following individual commands: - MSB_ENABLE - LSB_ENABLE - SYNC_ENABLE - ENABLE_BITIO True or False indicating the state of the image sensor connection. *Example:* **SENSOR_CONNECT=True ; Connect the sensor to the IAC hardware**	
SEQUENCE_IMAGES		Specifies the number of images to acquire during any given capture. These images are acquired at full video rate and are guaranteed to be sequential.	
	Number	Number of sequential images to capture *Example:*	

		SEQUENCE_IMAGES=5 ; Capture 5 consecutive images from the sensor at full video rate.	
SNAP		Signals the Image Acquisition Control application software to snap an image. NOTE: This command does not actually cause the IAC hardware to acquire an image. It is just a simple message or "application" command used to control the application software.	
		Example: **SNAP ; signal the image Acquisition Control Application Software to snap an image.**	
SPI_IO		Signals the Image Acquisition Control application software to continuously acquire, process, and display images from the image sensor. NOTE: This command does not actually cause the IAC hardware to acquire images. It is just a simple message or "application" command used to control the application software.	
	Enabled	0 = Continuous capture OFF 1 = Continuous capture ON *Example:* **CONTINUOUS_CAPTURE=1 ; enable continuous**	
SVR_CFG_SET		The SVR_CFG_SET command is the primary configuration command for the Serial Video Receiver and is used to configure both MIPI and SMIA modes.	
	CSI_NCCP	Primary mode control 0=CCP2 (SMIA)	

		1=CSI2 (MIPI)	
	STB/CLK	Selects the Strobe/Clock mode for the CCP2 (SMIA) mode 0=Clock mode 1=Strobe mode	
	Cont/Clk	Continuous clock mode for CSI1 (MIPI) and CCP2 (SMIA) 0=STB/CLK may be stopped between packets/data transmissions. 1=STB/CLK is constantly toggling. NOTE: In CCP mode, clock-miss is not checked when Continuous Clock Mode is off. In CSI mode, is I always checked, but used as timing indicator in non-continuous clock mode, and as error indication in continuous clock mode.	
	Dual_Lane	Dual lane mode (CSI2/MIPI only). 0=Single data lane (Lane 1). Lane 2 will be disabled with it's termination disconnected and with leakage power consumption only. 1=Dual data lane (Lanes 1+2)	
	ECC_Enable	0=The ECC field of CSI2(MIPI) packet headers is ignored. 1=ECC is checked and headers are corrected	
	CCP video format	0=RAW8 1=RAW10 2=RAW12 *Example:* **SVR_CFG_SET=1,1,1,0,1,1**	

SVR_ENABLE		Enables the Serial Video Receiver engine for the MIPI and SMIA high-speed serial sensor interfaces. NOTE: Enabling or disabling the SVR engine is independent of the capture mode (Parallel vs. SVR). Use the CAPTURE_MODE script command to select either parallel data capture or the SVR output	
	Enable	(TRUE or FALSE) TRUE will enable the Serial Video Receiver engine FALSE will disable the Serial Video Receiver engine *Example:* **SVR_ENABLE=TRUE ; enable the Serial Video**	
SVR_LANE_SKEW		The SVR_LANE_SKEW command is used to skew the clock and data lane signals relative to each other to compensate for timing skews during signal transmission from the source.	
	CLK_DELAY D1_DELAY D2_DELAY	Sets the CLK lane skew relative to the data lanes. Default=0 Range=-15 to +15 Sets the Data Lane 1 skew relative to the CLK and D2 data lanes. Default=0 Range=-15 to +15 Sets the Data Lane 2 skew relative to the CLK and D1 data lanes. Default=0 Range=-15 to +15 *Example:* **SVR_LANE_SKEW=0,0,0**	

SVR_TIMER_CFG_SET		The SVR_TIMER_CFG_SET command is used to control SVR internal timers and counters that affect the receiver functionality. These setting typically need to be adjusted for each new sensor type and sometimes for the same sensor in different operating modes.	
	Term_count_data	Timeout counter to connect termination, data lanes Default=0 Range= 0..16	
	Term_count_clk	Timeout counter to connect termination, clock lane Default=0 Range= 0..16	
	Th_settle_count_data	Timeout until PHY looks for sync, data lane Default=0 Range= 0..255	
	Th_settle_count_clk	Timeout until hs clock enables Default=0 Range= 0..255	
	Clock_miss_count	Timer until no-toggle in PHY byte-clock is considered clock miss. Default=0 Range= 0..16 Notes: 1. Incoming serial bits are counted by s divide-by-8 byte_clock counter. A special 5-bit clock_miss_counter is cleared whenever there is a positive or a negative edge on byte_clock. If the count reached clock_miss_count*2, a clock miss is detected and signaled. 2. In CCP Strobe mode, clock ^data comprises the input to the byte_clock counter: otherwise the operation is unchanged. *Example:*	

		SVR_TIMER_CFG_SET=0,0,32,32,15	
SWAP_BYTES		Perform a swap byte operation on the image sensor data stream.	
	Swap Bytes	(TRUE or FALSE) TRUE will swap bytes in the image sensor data stream output. FALSE is equivalent to a "No Operation" *Example:* **SWAP_BYTES=FALSE**	
SYNC_ENABLE		Enables/Disables the FET switches which connect or disconnect the sensor SYNC lines to/from the IAC hardware.	
	ENABLED?	True or False indicating the state of the sensor data SYNC line FET switches. *Example:* **SYNC_ENABLE=True ; Connect the sensor SYNC lines to the IAC hardware.**	
TALK		Speak a string of text	
	Text to speak	The text that will be converted to speech and played on the computers sound output *Example:* **TALK=Setting Sensor to BAYER mode.**	
TRANSPOSE		Transpose the 2D Image data before processing and display	
	Transpose	(TRUE or FALSE) TRUE will transpose the image data before processing and display. FALSE is equivalent to a "No Operation" *Example:* **TRANSPOSE=FALSE**	
YCBCR_FILTER_TYPE		Specifies the display filter used within the IAC application for YCbCr image buffer type.	

	Filter Type	0 = YCbCr RAW 1 = YCbCr Byte 1 2 = YCbCr Byte 2 3 = YCbCr Y Plane 4 = YCbCr Cb Plane 5 = YCbCr Cr Plane 6 = RGB Interpolation to RGB *Example:* **YCBCR_FILTER_TYPE=3 ; luminance plane**	
YCBCR_PATTERN		Specifies which YCbCr encoding is used in the encoded image data.	
	Pattern	0, 1, 2, or 3 0 = Y-Cb-Y-Cr 1 = Y-Cr-Y-Cb 2 = Cb-Y-Cr-Y 3 = Cr-Y-Cb-Y *Example:* **YCBCR_PATTERN=1 ; use Y-Cb-Y-Cr**	