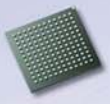
OmniVision



application note

IMPLEMENTATION GUIDE

OV490/OV10640 camera solution

OV490/OV10640

camera solution implementation guide

application note
IMPLEMENTATION GUIDE

version 1.1
january 2015

To learn more about OmniVision Technologies, visit www.ovt.com.
OmniVision Technologies is publicly traded on NASDAQ under the symbol OVTI.

table of contents

list of figures

list of tables

1 application system of OV490/OV10640 camera solution

1.1 overview

The OV10640 is an optical RAW format 1280x1080, low power CMOS, active-pixel digital high dynamic range (HDR) sensor for both human vision and machine vision automotive applications.

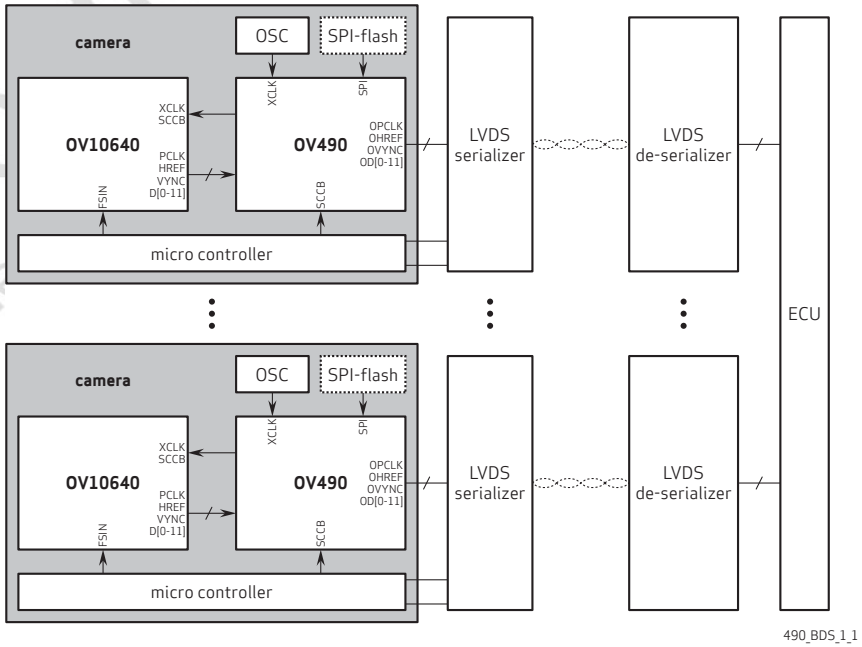
The OV490 is a companion ISP processor which performs sophisticated camera functions, including lens shading correction, defect pixel correction, color interpolation and correction, de-noise and sharpening, high dynamic range (HDR) combination, gamma correction and tone mapping. It enables advanced HDR imaging in a simple system.

The OV490 and OV10640 camera solution provides a full system solution with the fully processed YUV output for automotive applications.

1.2 typical OV490 and OV10640 multi-camera system 1

figure 1-1 shows a block diagram of a typical multi-camera system using the OV10640 and OV490. The OV490 combined with the OV10640 can operate as a single camera where the OV10640 is fully controlled by the OV490. Each camera contains a local microcontroller. A centralized ECU can control each OV490/OV10640 via the micro-controller. The microcontroller sends its commands to the OV490 via the SCCB interface. The OV490 then handle these commands and takes care of any necessary sensor control timing. The firmware of the OV490 is stored in an external SPI flash and is automatically read by the OV490 when powered up.

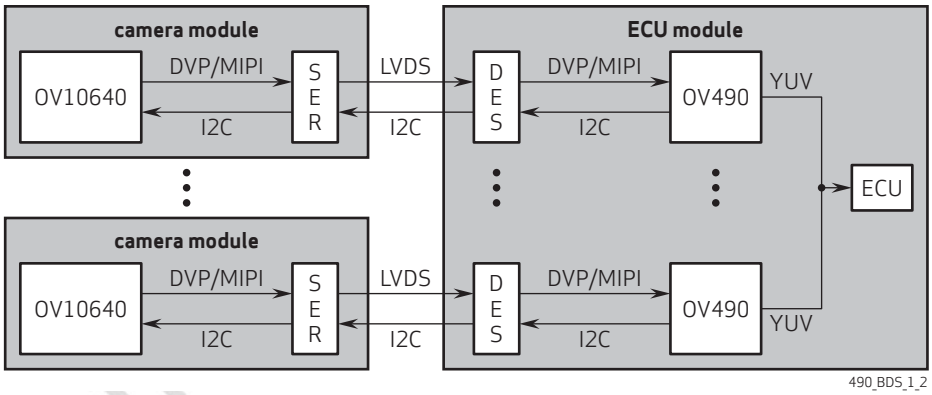
figure 1-1 OV490/OV10640 system 1 block diagram



1.3 typical OV490 and OV10640 multi-camera system 2

Another typical multi-camera system architecture is shown in **figure 1-2**.

figure 1-2 OV490/OV10640 system 2 block diagram



In this system, the OV10640 is embedded in the camera module, while the OV490 is placed near the ECU. The OV10640 outputs raw image data through the LVDS to the OV490. The OV490 also controls the OV10640 behavior by I2C protocol through the LVDS back channel.

2 hardware operation

2.1 reference design of OV490 and OV10640

2.1.1 DVP input

figure 2-1 OV490/OV10640 (DVP input) reference design schematic

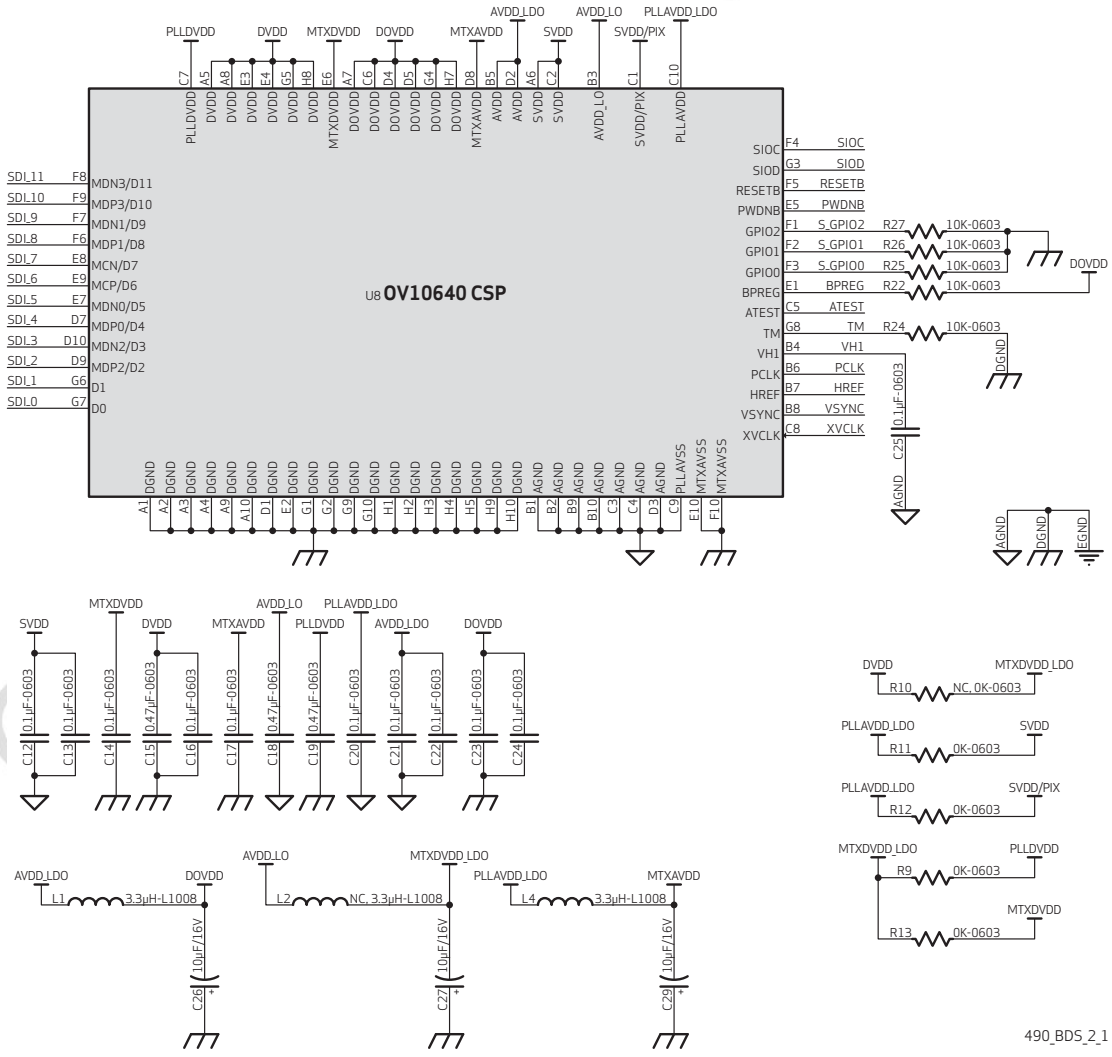
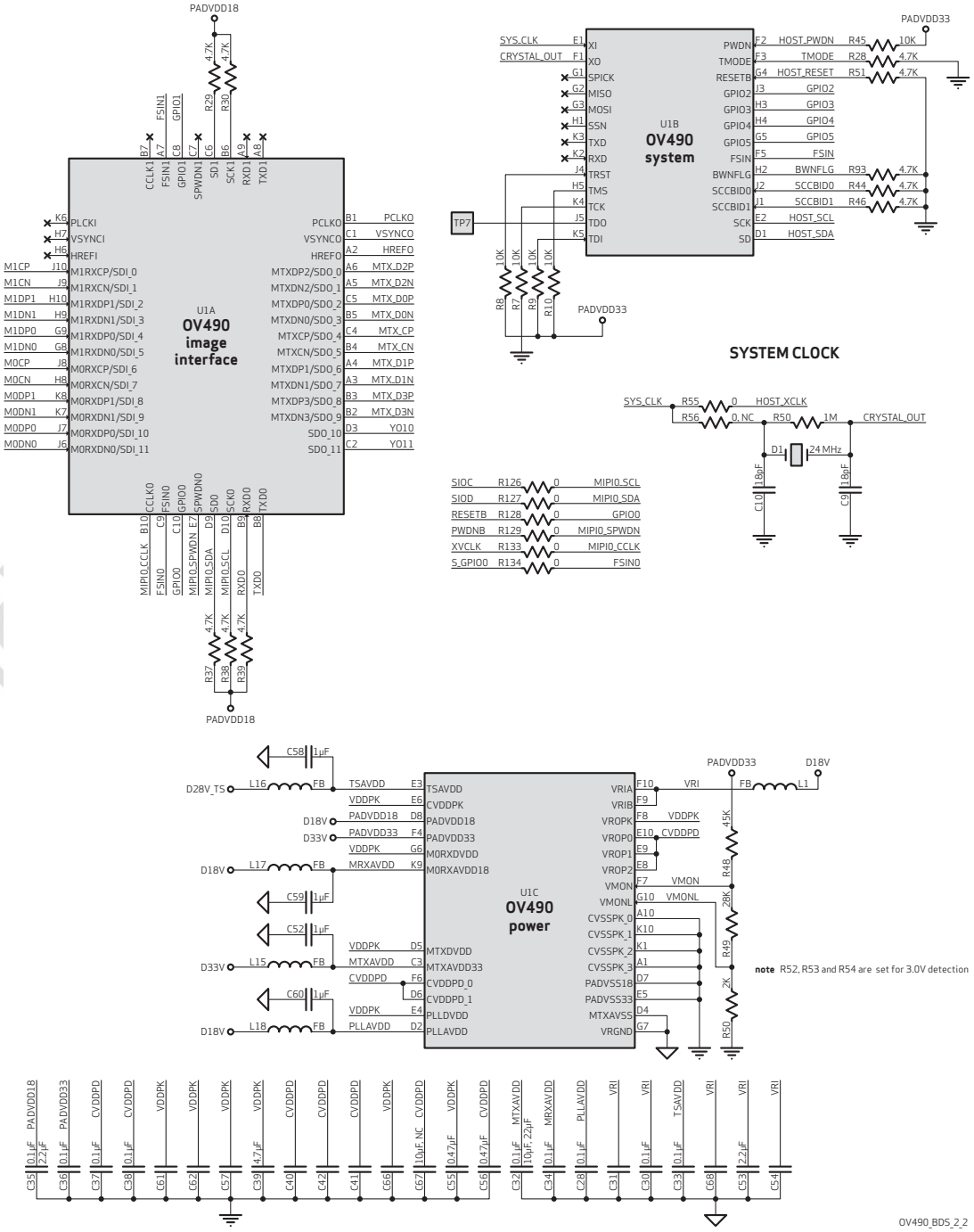


figure 2-2 OV490/OV10640 (DVP input) interface schematic



2.1.2 MIPI input

figure 2-3 OV490/OV10640 (MIPI input) reference design schematic

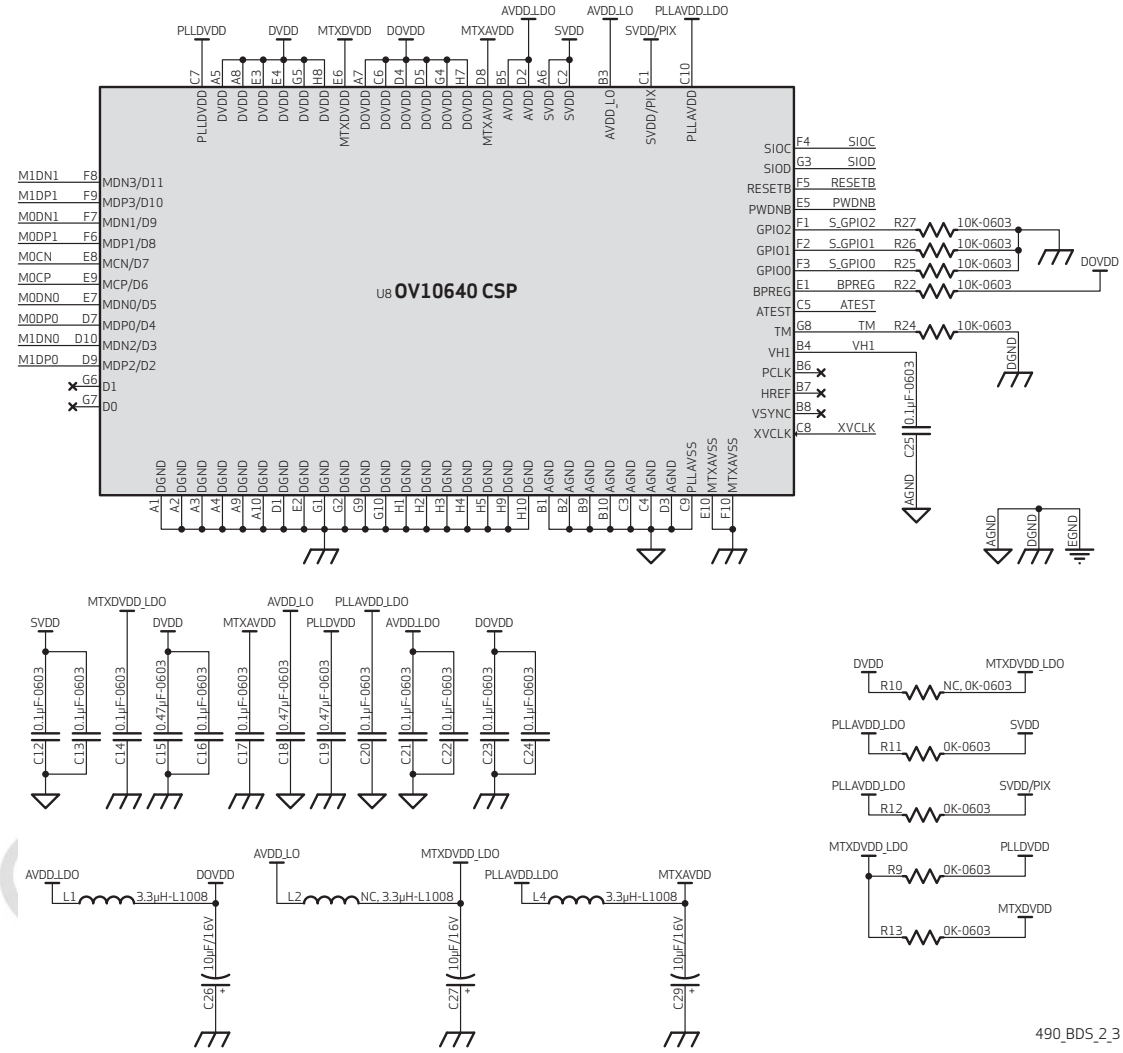
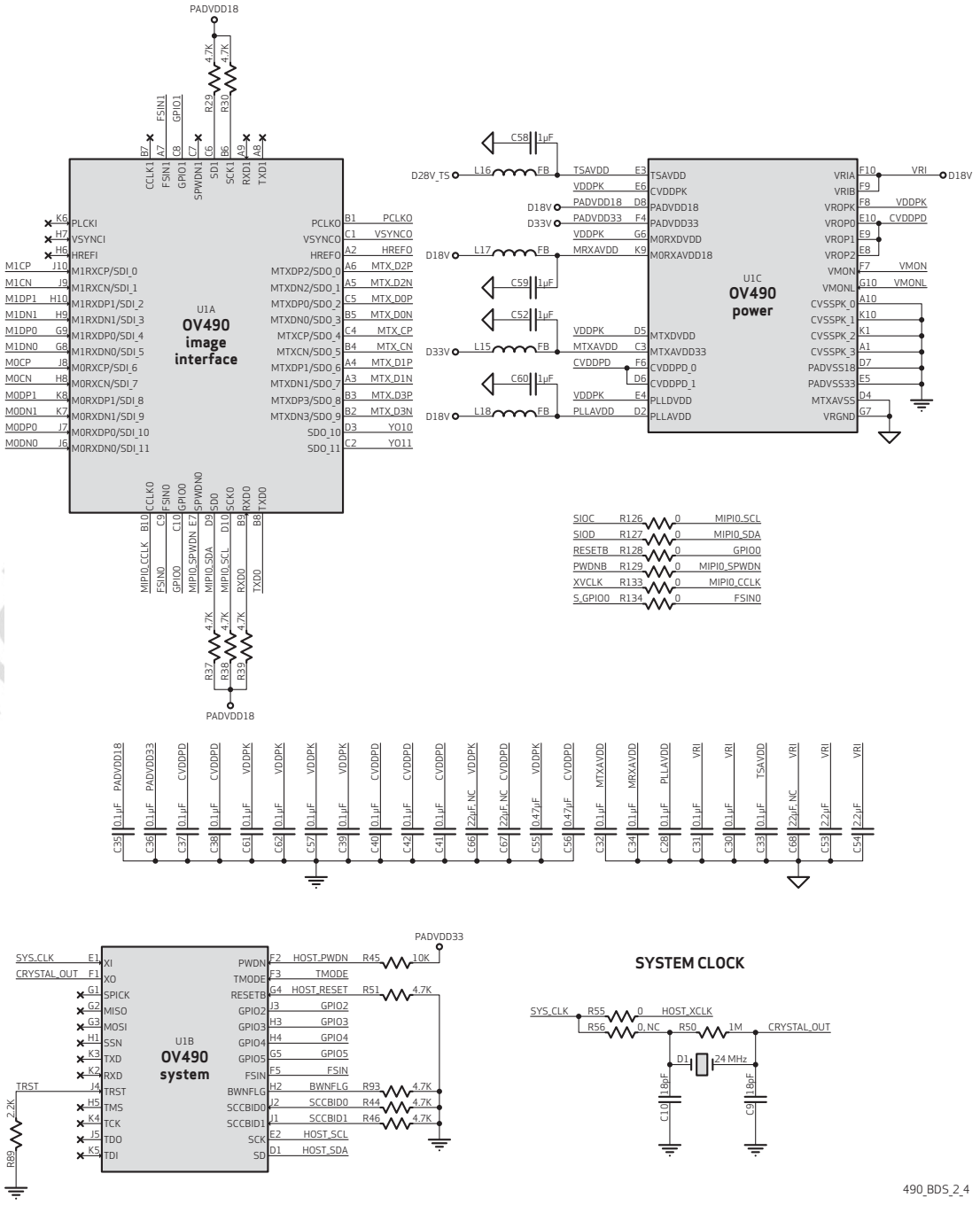


figure 2-4 OV490/OV10640 (MIPI input) interface schematic



2.1.3 some un-used pins

Not all functions are used such as SPI and brown-out. The recommended configuration is shown in **table 2-1**.

table 2-1 unused pins select

function	pins	if used	if unused
UART	RXD	10K pull-up to 3.3V	10K pull-up to 3.3V
UART1	RXD1	10K pull-up to 1.8V	10K pull-up to 1.8V
SPI	SICK	adjust pull-down resistor according to timing	10K pull-down
	MISO	refer to reference design	10K pull-down
	MOSI		10K pull-down
	SSN	10K pull-up to 3.3V	10K pull-up to 3.3V
JTAG	TMS	10K pull-up to 3.3V	10K pull-up to 3.3V
	TCK	1K pull-down	1K pull-down
	TDI	10K pull-up to 3.3V	10K pull-up to 3.3V
	TRST	10K pull-up to 3.3V	10K pull-up/down to 3.3V
brown-out	VMON	refer to reference design	floating
	VMONL		10K pull-down
	BWNFLG		floating
FSIN	FSIN	refer to reference design	2.2K pull-down
TMODE	TMODE	for testing purposes only	floating

2.2 boot-up pin selection

By connecting FSIN1 and FSIN0 to pull-up/pull-down resistors, FSIN1 and FSIN0 are used as boot select at power up. After bootup, these pins are automatically configured to output pins to work as frame sync control.

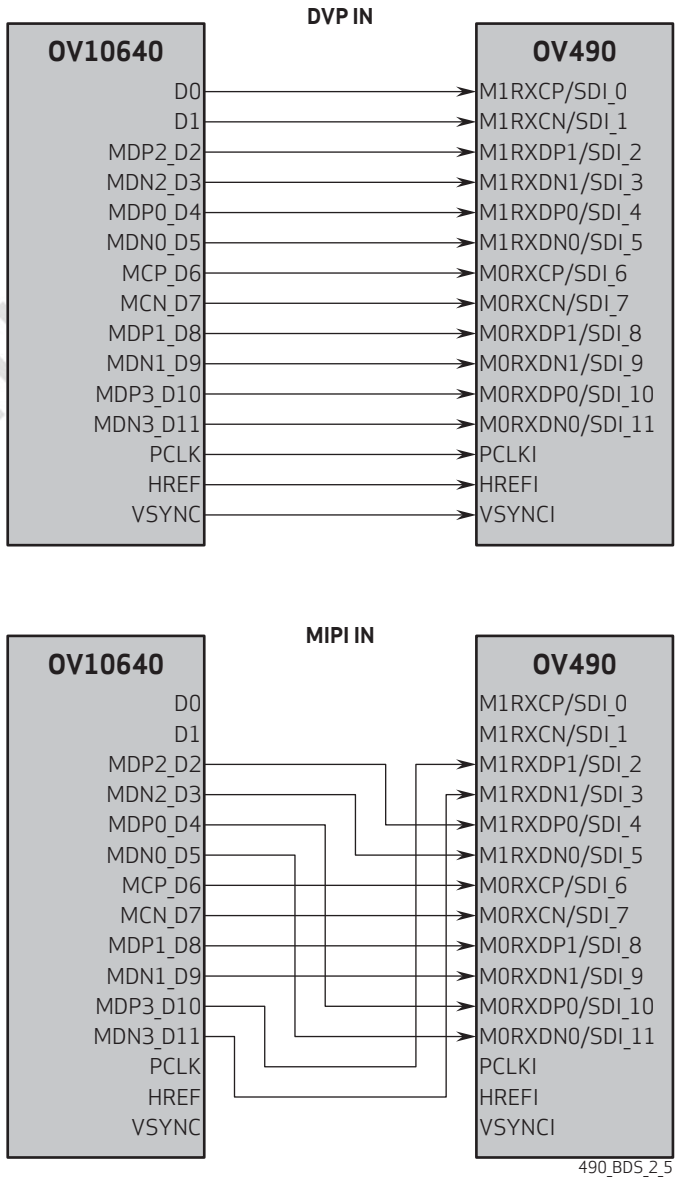
table 2-2 boot-up pin select

FSIN1	FSIN0	boot method
0	0	UART or SCCB slave boot-up (by host download)
0	1	SCCB master boot-up (with external EEPROM)
1	0	SPI master boot-up (with external SPI flash)
1	1	SPI slave boot-up (by host download)

2.3 OV490 and OV10640 interconnection for DVP and MIPI

The interface between the OV490 and the OV10640 can be via DVP or MIPI but they are not inter-operable. Refer to **figure 2-5** showing the hardware connection for DVP and MIPI interconnection between the OV490 and the OV10640.

figure 2-5 OV490 and OV10640 interconnection for DVP and MIPI



2.4 interface between OV490 and host

table 2-3 signal descriptions

function	signal name	I/O	description
image	MTXDP2/SDO_0	output	MIPI TX data lane 2 positive output / DVP image data bit 0
	MTXDN2/SDO_1	output	MIPI TX data lane 2 negative output / DVP image data bit 1
	MTXDP0/SDO_2	output	MIPI TX data lane 0 positive output / DVP image data bit 2
	MTXDN0/SDO_3	output	MIPI TX data lane 0 negative output / DVP image data bit 3
	MTXCP/SDO_4	output	MIPI TX clock lane positive output / DVP image data bit 4
	MTXCN/SDO_5	output	MIPI TX clock lane negative output / DVP image data bit 5
	MTXDP1/SDO_6	output	MIPI TX data lane 1 positive output / DVP image data bit 6
	MTXDN1/SDO_7	output	MIPI TX data lane 1 negative output / DVP image data bit 7
	MTXDP3/SDO_8	output	MIPI TX data lane 3 positive output / DVP image data bit 8
	MTXDN3/SDO_9	output	MIPI TX data lane 3 negative output / DVP image data bit 9
	SDO_10	output	DVP image data bit 10
	SDO_11	output	DVP image data bit 11
	PCLKO	output	DVP sensor video pixel clock
	VSYNCO	output	DVP sensor video vertical sync
	HREFO	output	DVP sensor video horizontal reference
control	XI	input	crystal / oscillator input or system reference clock
	TXD	output	UART serial data transmission
	RXD	input	UART serial data receiving
	PWDN	input	system power down control
	RESETB	input	system reset
	SCK	input	SCCB clock
	SD	I/O	SCCB data
	GPIO[5:2]	I/O	general purpose input/output pins 2~5
	SCCBID0	input	SCCB ID select 0
	SCCBID1	input	SCCB ID select 1
	SPICK	I/O	serial interface master/slave bit clock
	MISO	I/O	serial interface master/salve data
	MOSI	I/O	serial interface master/salve data
	SSN	I/O	serial interface chip select
	BWNFLG	output	brown-out flag
	FSIN	I/O	frame sync control

2.5 power supply

table 2-4 OV490/OV10640 power supply

OV10640		OV490	
pin	power supply	pin	power supply
PLLDVDD	1.5V	TSAVDD	2.8V~3.3V
DVDD	1.5V	PADVDD18	1.8V
MTXDVDD	1.5V	PADVDD33	3.3V
DOVDD	1.8V	M0RXAVDD18	1.8V
MTXAVDD	3.3V	MTXAVDD33	3.3V
AVDD	1.8V	PLLAVDD	1.8V
SVDD	3.3V	VRIA/VRIB	1.8V
AVDD_LO	1.5V		
SVDD/PIX	3.3V		
PLLAVDD	3.3V		

2.6 power sequence

2.6.1 power on

The OV490 requires 3.3V and 1.8V power inputs (PADVDD and AVDD) from an external power source. The embedded voltage regulators take 1.8V~3.3V input and produce 1.2V to supply the digital core. Also, the regulator needs set-up time to output stable 1.2V. For safety's sake, the application system should take the set-up time into account when powering up the OV490.

figure 2-6 power on sequence timing

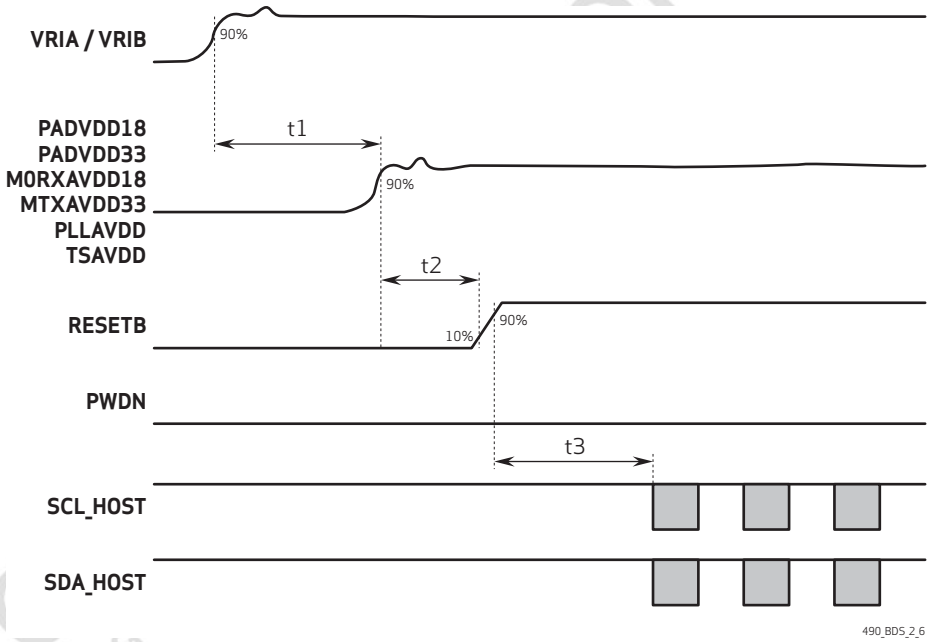


table 2-5 power on timing specifications

symbol	parameter	value
t1	delay from VRIA/VRIB stable to PADVDD18/PADVDD33/M0RXAVDD18/MTXAVDD33/PLLAVDD/TSAVDD stable	≥250 µs
t2	delay from PADVDD18/PADVDD33/M0RXAVDD18/MTXAVDD33/PLLAVDD/TSAVDD stable to RESETB pulled high	≥250 µs
t3	delay from RESETB pulled high to SCCB access	≥1 ms

2.6.2 power off

To avoid an unknown state on the I/O, resetting the OV490 is recommended before powering off.

figure 2-7 power off sequence

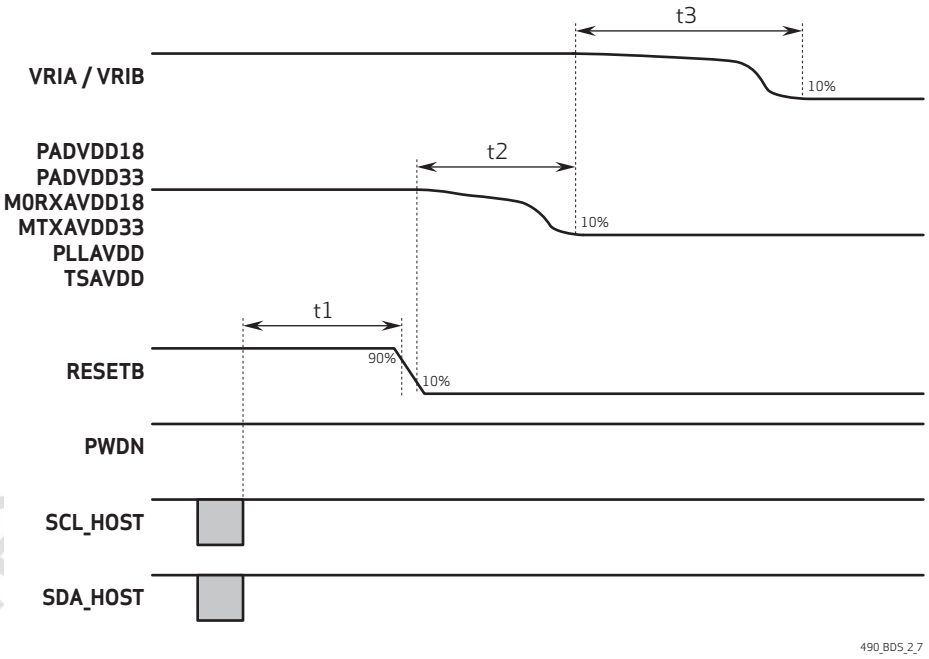


table 2-6 power off timing specifications

symbol	parameter	value
t1	delay from SCCB stream off command to RESETB pulled low	≥1 frame
t2	delay from RESETB pulled low to PADVDD18/PADVDD33/M0RXAVDD18/MTXAVDD33/PLLAVDD/TSAVDD power off	≥100 µs
t3	delay from PADVDD18/PADVDD33/M0RXAVDD18/MTXAVDD33/PLLAVDD/TSAVDD power off to VRIA/VRIB power off	≥0 µs

2.7 hardware layout notes

Please refer to the *High Speed Serial Interface Routing Guideline for Camera Module Design Application Note*.

3 software operation

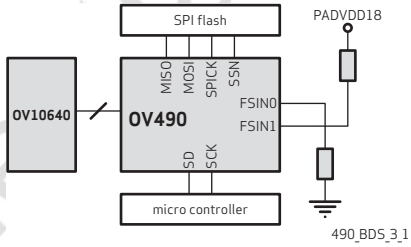
3.1 boot scheme

In the OV490 and OV10640 camera systems, the OV490 plays two roles. One is as master to control the sensor and output processed data and the other as slave to accept commands from an external controller. To provide a high performance flexible application, the OV490 has an embedded 32-bit RISC processor specifically for application firmware development. There are 128KB embedded SRAM and 32KB ROM to store firmware and parameters. Firmware is loaded externally and is necessary in order to run ISP algorithms and perform function control. There are four boot methods to load the firmware and boot OV490 system.

3.1.1 boot from SPI flash

3.1.1.1 hardware connection

figure 3-1 hardware connection



3.1.1.2 flash memory arrangement

The OV490 will boot up with low speed (4MHz) to load the header and quick_set instruction, if any, and speed up to 12MHz or 24MHz for loading the main firmware.

table 3-1 flash memory arrangement

byte index	context
0~39	header
start address of header information	firmware

3.1.1.3 header introduction

The first 40 bytes in SPI flash is the configuration for boot ROM. The format is shown in **table 3-2**:

table 3-2 SPI flash boot format

DWORD	item	description
0	0x4F565449	Magic Number
1	CONFIG	Bit[31]: set_div2zero_en 0: No change allowed 1: Change SPI divider Bit[7:0]: div div= (sysclk/(4-wire serial interface_clk*2))-1
2	SOURCE START ADDRESS	Flash/EEPROM Start Address
3	DESTINATION START ADDRESS	Memory Start Address
4	LENGTH	Firmware Data Length
5	SYS_CONFIG	Bit[7]: New format mode Bit[4]: CRC16 check FW flag 0: Do not check 1: Check Bit[3]: Load firmware select 0: CPU mode 1: Load firmware in DMA mode Bit[2]: DMA 32-bit R/W enable 0: Disable 1: Enable Bit[1]: Dcache enable 0: Disable 1: Enable Bit[0]: Lcache enable 0: Disable 1: Enable
6	DELAY	Interval of Read Firmware
7	RD_SIZE[a]	Read Size per Read Command (0~0x1FF)
8	QUICK_SET	Bit[31]: Quick set flag 0: Do not set 1: Set flag Bit[23:0]: Quick set table address in SPI flash
9	CRC	Bit[31:16]:Quick set table length Bit15:0]: Header CRC (CRC of 38 byte in header)

a. RD_SIZE should be 4 bytes aligned if SYS_CONFIG[2] is set to 1

3.1.1.4 quick set introduction

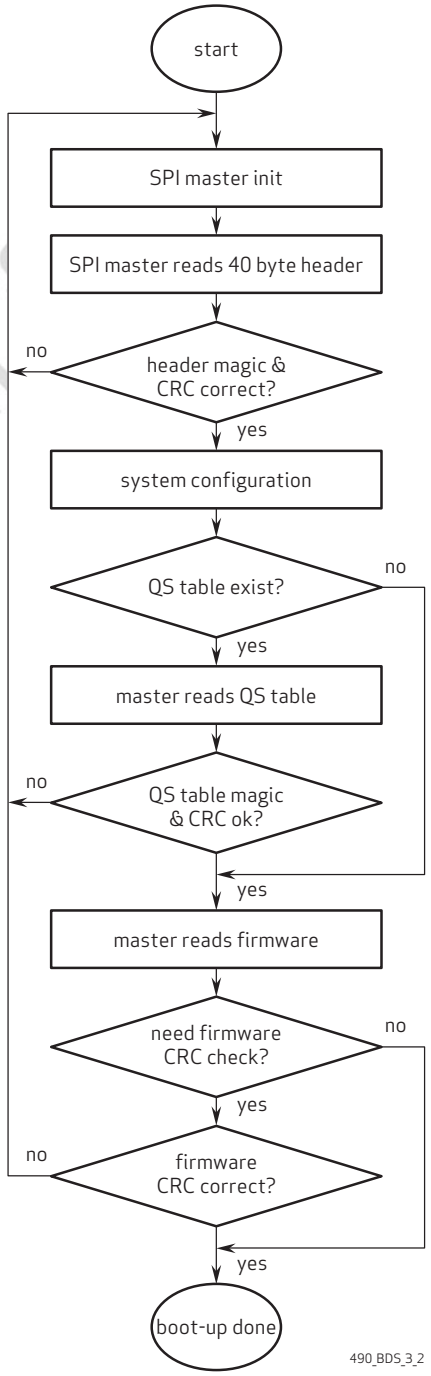
The format of the quick set table is shown in **table 3-3**.

table 3-3 header introduction

DWORD	item	description
0	0x6F767469	Magic Number
1	CRC	Bit[15:0]: Quick set table CRC 16 value
2	REG0_ADDR	Register Address
3	REG0_DATA	Data Set to Register
...	...	...
2+ (n-1) × 2	REG (n-1)_ADDR	Register Address
2+(n-1) × 2 +1	REG (n-1)_DATA	Data Set to Register

3.1.1.5 working flow

figure 3-2 working flow



3.1.1.6 timing

figure 3-3 timing diagram

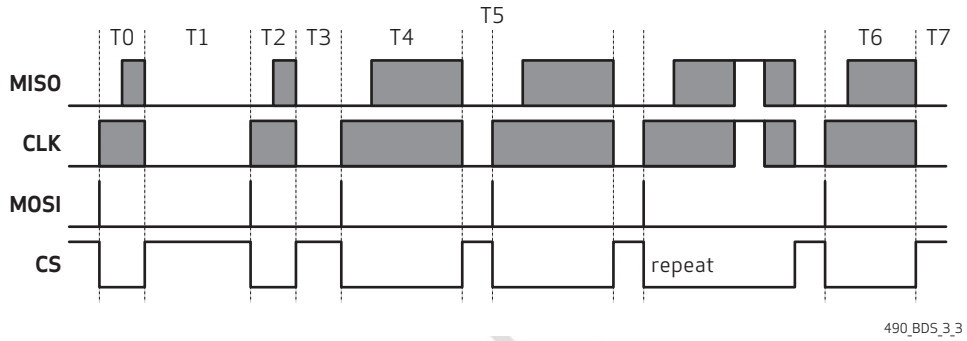


table 3-4 timing specifications

symbol	parameter	requirement	time
T0	SPI master read header	SPI_CLK = 4MHz, SYS_CLK = 24MHz	96.8 µs
T1	check header and system configuration		287.1 µs
T2	read quick set table	quick set table length = 256 bytes	350 µs
T3	check quick set table	set SYS_CLK 160MHz	499.4 µs
T4	read FW, first packet, 508 bytes[a]		170 µs
T5	delay and save data		8.8 µs
(T4+T5)×N	read N times, 508 bytes each packet		35,760 µs
T6	read last pack		114 µs
T7	check firmware CRC		11,000 µs

a. testing firmware length is 100KByte

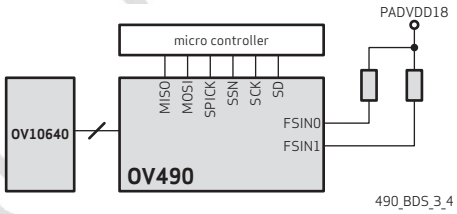
3.1.2 boot from SPI host

3.1.2.1 hardware connection

In this boot-up mode, the following pull-up/pull-down resistors are recommended in case the OV490 SPI function enters an unknown state:

– SPICK should be pulled down by a 10K resistor
– SSN should be pulled up by a 10K resistor

figure 3-4 hardware connection



3.1.2.2 protocol

1. Master resets slave and delays some time.
2. Master sends 32 bytes header to slave, delays some time and then receives the ACK.
3. Slave parses the header and responds to the master with 4 bytes ACK1 "0x55CC55CC".
4. If master receives ACK1 “0x55CC55CC”, go to step 5; otherwise, go to step 1.
5. If the quick set table exists, master sends the quick set table to slave, delays some time and then receives the ACK, go to step 6; otherwise, go to step 8.
6. Slave parses the quick set table and responds to the master with 4 bytes ACK2 "0x11AA11AA".
7. If master receives ACK2 “0x11AA11AA”, go to step 8; otherwise, go to step 1.
8. Master sends firmware to slave, delays some time and then receives the ACK.
9. Slave checks the firmware and responds to the master with 4 bytes ACK3 "0x22DD22DD".
10. If master receives ACK3 “0x22DD22DD”, go to step 11; otherwise, go to step 1.
11. Boot-up done.

3.1.2.3 header introduction

table 3-5 header introduction[ab]

DWORD	item	description
0	0x4F565449	Magic Number
1	DESTINATION START ADDRESS	Stand For Memory Start Address
2	LENGTH	Firmware Data Length
3	SYS_CONFIG	Bit[7]: New format mode Bit[4]: CRC16 check FW flag 0: Do not check 1: Check Bit[3]: Load firmware select 0: CPU mode 1: Load firmware in DMA mode Bit[2]: DMA 32-bit R/W enable 0: Disable 1: Enable Bit[1]: D-cache enable 0: Disable 1: Enable Bit[0]: L-cache enable 0: Disable 1: Enable
4	RD_SIZE	Size per Packet in Quick Set/Firmware Download Stage
5	QUICK SET	Bit[0]: Quick set flag 0: Do not set 1: Set flag
6	DELAY/TIME	Delay for Synchronizing with Master
7	CRC	Bit[31:16]:Quick set table length Bit[15:0]: Header CRC16 value

a. RD_SIZE should be 4 bytes aligned if SYS_CONFIG[2] is set to1.

b. Header crc16 value is calculated based on crc16_ccitt algorithm
For example:

```
DWORD header[8] = {
 0x4F565449,
0x00190000,
0x00005738,  //firmware size
0x0000009F,
0x000001FC,
0x00000001,
0x00000000,
0x0018BA13  //[15:0]: header CRC16, calculated using previous 30 byte data.
};
```

3.1.2.4 quick set introduction

table 3-6 quick set format

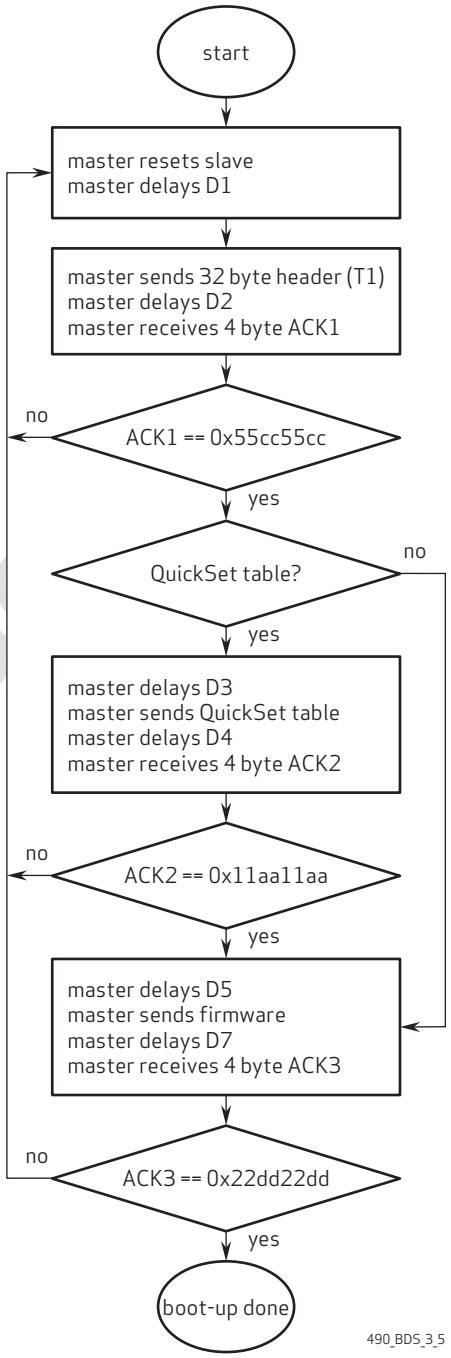
DWORD	item	description
0	0x6F767469	Magic Number
1	CRC	Bit[15:0]: Quick set table CRC 16 value
2	REG0_ADDR	Register Address
3	REG0_DATA	Data Set to Register
...	...	...
2+ (n-1) × 2	REG(n-1)_ADDR	Register Address
2+(n-1) × 2 +1	REG(n-1)_DATA	Data Set to Register

The recommended quick set table is:

```
DWORD QS[6] = {
      0x6F767469,
      0x00008888,
      0x00800020,
      0x00041601,
      0x008000a0,
      0x2974a11d
};
```

3.1.2.5 flow chart

figure 3-5 working flow



3.1.2.6 timing

figure 3-6 timing diagram

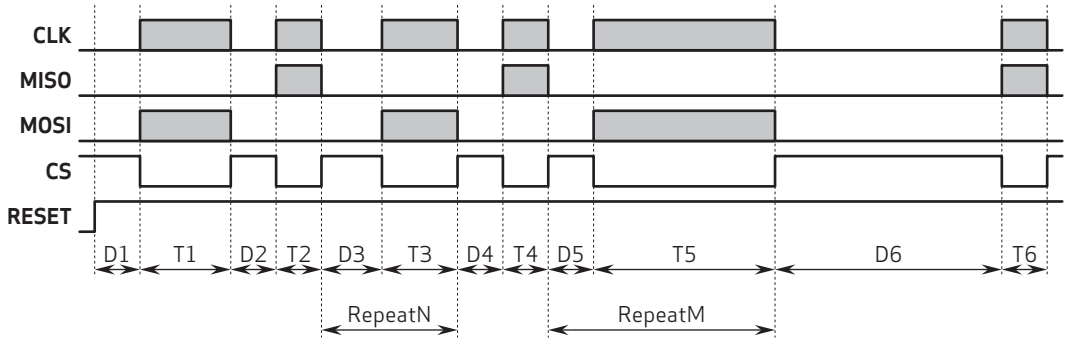


table 3-7 timing specifications (sheet 1 of 2)

symbol	parameter	requirement	time
D1	reset -> send header		500 µs
T1	send header	SPI_CLK ≤ 6MHz	41 µs
D2	header done -> receive ACK1		375 µs
T2	receive ACK1	SPI_CLK ≤ 6MHz	6 µs
D3	delay before sending Quick Set table packet		103 µs
T3	send Quick Set table packet	SPI_CLK ≤ 6MHz	30 µs
(D3+T3)×N		if Quick Set table is larger than header -> RD_SIZE, then should send table in packet, each packet is header>RD_SIZE size, except for the last packet. each packet should have D3 delay before sending.	133 µs × N
D4	apply Quick Set table	depends on Quick Set table length	173 µs
T4	receive ACK2	if PLL is enabled in Quick Set, the system clock is 160MHz now, the SPI_CLK ≤ 24MHz from now on.	3 µs
D5	delay before sending firmware[a] packet		24 µs
T5	send firmware packet	max SPI_CLK is same as T4	162 µs

table 3-7 timing specifications (sheet 2 of 2)

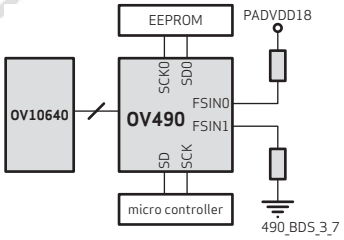
symbol	parameter	requirement	time
(D5+T5)×M		if firmware is larger than header -> RD_SIZE, then should send firmware in packet, each packet is header>RD_SIZE size, except for the last packet. each packet should have D5 delay before sending	186 µs × N
D6	check firmware CRC	depends on firmware length and system clock	11,000 µs
T6	receive ACK3	max SPI_CLK same as T4	3 µs

a. tested firmware length is 100KByte

3.1.3 boot from EEPROM

3.1.3.1 hardware connection

figure 3-7 hardware connection



3.1.3.2 header introduction

The first 32 bytes in EEPROM is the configuration for boot loader.

See **table 3-6** for the format of the 32 bytes.

table 3-8 header format (sheet 1 of 2)

DWORD	item	description
0	0x4F565449	Magic Number
1	BAUD_CFG	Value to be Written to the SCCB Baud Configure Register
2	SOURCE START ADDRESS	Flash/EEPROM Start Address
3	DESTINATION START ADDRESS	Memory Start Address

table 3-8 header format (sheet 2 of 2)

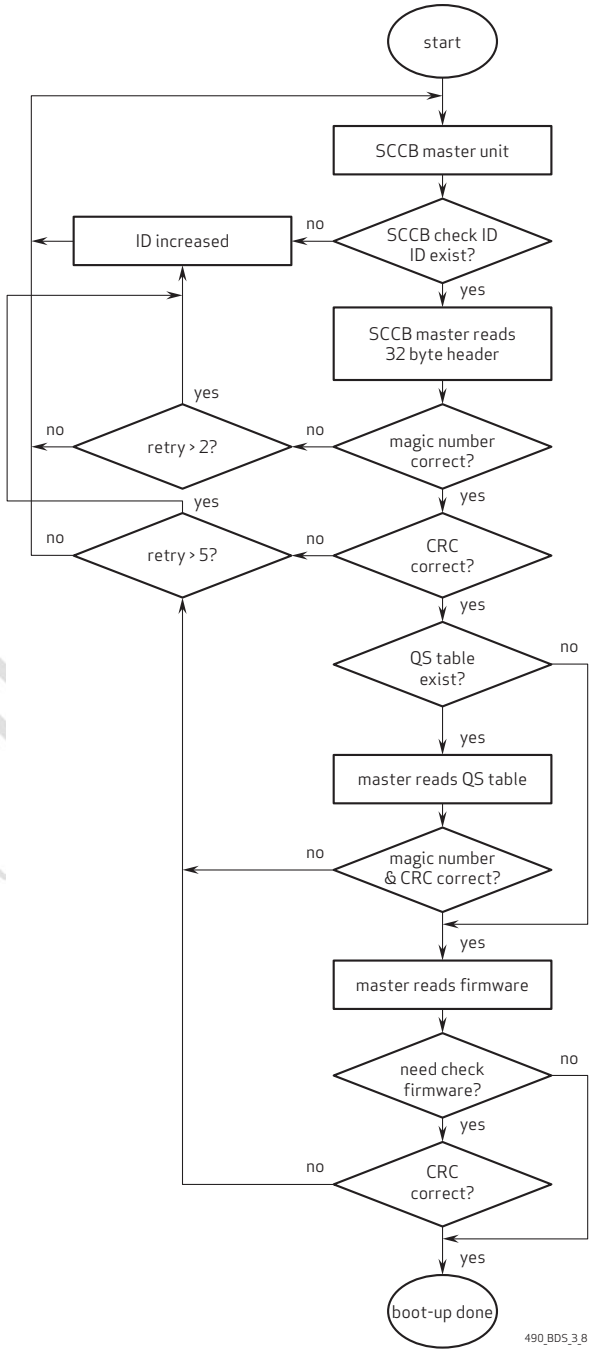
DWORD	item	description
4	LENGTH	Firmware Data Length
5	SYS_CONFIG	Bit[7]: New format mode Bit[4]: CRC16 check FW flag 0: Do not check 1: Check Bit[1]: D-cache enable 0: Disable 1: Enable Bit[0]: L-cache enable 0: Disable 1: Enable
6	QUICK_SET	Bit[31]: Quick set flag 0: Do not set 1: Set flag Bit[23:0] Quick set table address in SPI flash
7	CRC	Bit[31:16]:Quick set table length Bit[15:0]: Header CRC (CRC of 30 bytes in header)

3.1.3.3 quick set information

The format for the Quick Set table is the same as shown in **section 3.1.1.4**.

3.1.3.4 working flow

figure 3-8 working flow



3.1.3.5 timing

figure 3-9 timing diagram

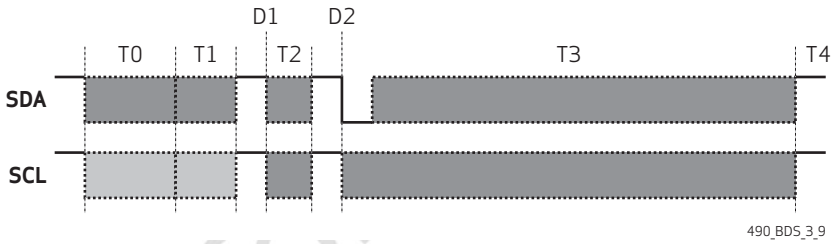


table 3-9 timing specifications

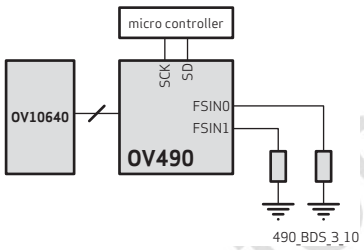
symbol	parameter	requirement	time
T0	SCCB master check ID	SCCB baud = 100KHz, SYS_CLK = 24MHz	1,020 µs
T1	read header		3,310 µs
D1	check header and system configuration		172 µs
T2	read Quick Set table		6,350 µs
D2	check Quick Set table	SYS_CLK=160MHz, SCCB baud = 400KHz	315 µs
T3	read firmware[a]		228,000 µs
T4	check CRC of firmware and uncompress		41,000 µs

a. tested firmware length is 10KByte

3.1.4 boot from SCCB host

3.1.4.1 hardware connection

figure 3-10 hardware connection



3.1.4.2 protocol

Following are boot flag states definitions:

```
#define SCCB_BOOT_RDY                (0xa1)

#define SCCB_BOOT_JUMPBOOT_START     (0xc2)

#define SCCB_BOOT_JUMPBOOT_START_CACHE   (0xc5)

#define SCCB_BOOT_JUMPBOOT_OK        (0xc3)

#define SCCB_BOOT_JUMPBOOT_ERR       (0xc4)

#define FW_RUNNING_STAGE_INIT        (0x01)

#define FW_RUNNING_STAGE_SNR_INIT_DONE   (0x11)

#define FW_RUNNING_STAGE_FINAL_DONE      (0x02)

#define FW_RUNNING_STAGE_SNR_ID_ERR      (0xEE)

#define FW_RUNNING_STAGE_SLAVEIF_NACK    (0xFF)
```

Note:

- SCCB_BOOT_JUMPBOOT_START_CACHE means checking CRC with I-Cache and D-Cache enabled
- SCCB_BOOT_JUMPBOOT_OK means CRC check is OK
- SCCB_BOOT_JUMPBOOT_ERR means CRC check is ERROR
- FW_RUNNING_STAGE_INIT means the firmware has booted up successfully and is running
- FW_RUNNING_STAGE_SNR_INIT_DONE means the sensor is initialized by the setting successfully
- FW_RUNNING_STAGE_FINAL_DONE means the firmware is already finished initialization and ready to accept any host commands from the host
- FW_RUNNING_STAGE_SNR_ID_ERR mean the sensor I2C slave ID is not detected and then the firmware will not initialize the sensor
- FW_RUNNING_STAGE_SLAVEIF_NACK mean the host cannot get any I2C to acknowledge from the OV490. This only happens when OV490 is in reset state or the OV490 slave ID is changed.

Furthermore, the following definitions which are used in the protocol:

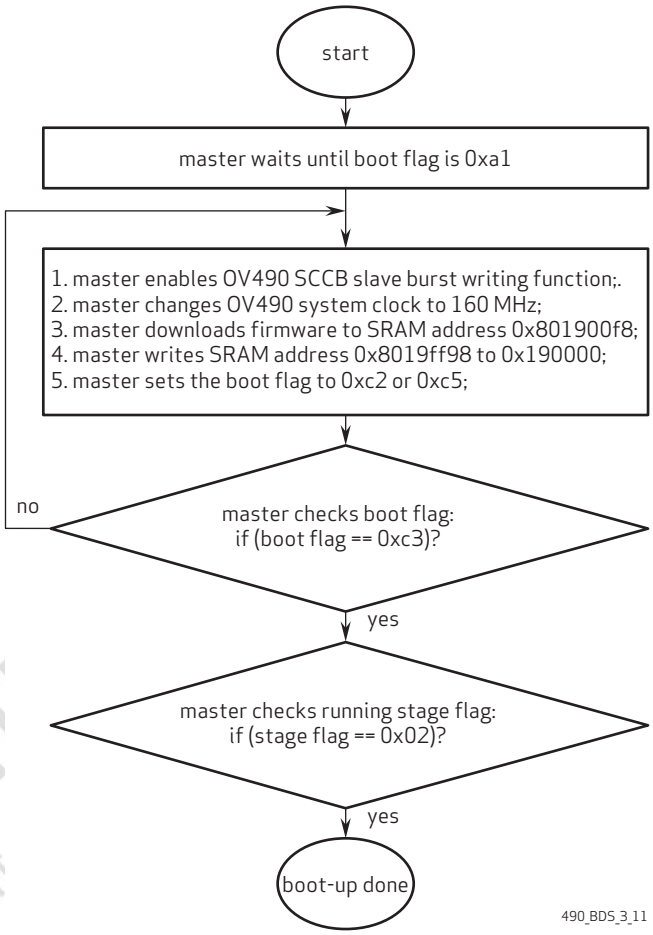
```
#define SRAM_FLAG_ADDR                  (0x8019ffc8)

#define SRAM_FW_ADDR_BOOT               (0x8019ffc4)

#define SRAM_FW_START                   (0x801900f8)

#define ADDR_FW_BOOT                    ((SRAM_FW_START &0x00ffffff)-0xf8)

#define SCCB_BURST_ADDR                 (0x802a6000)

#define FW_RUNNING_STAGE_ADDR           (0x80800120s)
```

1. Master reads boot flag from SRAM address SRAM_FLAG_ADDR and waits until value is changed to SCCB_BOOT_RDY
2. Master writes value 0x01 to SCCB_BURST_ADDR in order to enable OV490 SCCB slave burst writing function
3. Master changes OV490 system clock to 160MHz as follows:
    - writes value 0x01 to register 0x80800020
    - writes value 0x16 to register 0x80800021
    - writes value 0x1D to register 0x808000A0
    - writes value 0xA1 to register 0x808000A1
    - writes value 0x74 to register 0x808000A2
    - writes value 0x29 to register 0x808000A3
4. Master download the firmware to SRAM address *SRAM_FW_START*
5. Master writes value ADDR_FW_BOOT to SRAM address SRAM_FW_ADDR_BOOT;
    - writes value 0x00 to register 0x8019ffc4
    - writes value 0x19 to register 0x8019ffc5
    - writes value 0x00 to register 0x8019ffc6
    - writes value 0x00 to register 0x8019ffc7
6. Master writes value SCCB_BOOT_JUMPBOOT_START or SCCB_BOOT_JUMPBOOT_START_CACHE to SRAM address SRAM_FLAG_ADDR to inform OV490 to start the CRC check
    - writes value 0xc2 or 0xc5 to register 0x8019ffc8
7. Master checks the boot flag:
    - If SCCB_BOOT_JUMPBOOT_ERR, then jump to step 2 to download firmware again;
    - If SCCB_BOOT_JUMPBOOT_OK, then boot ok and go to step 8;
8. Master checks the firmware running stage register and only if the value is FW_RUNNING_STAGE_FINAL_DONE, then it can begin to send host command (i.e. STREAM_ON) to OV490

Note that SRAM_FW_START can be defined by the user, but the sum of SRAM_FW_START and firmware length (FW_SIZE) should be less than 0x8019A000 (i.e., SRAM_FW_START + FW_SIZE < 0x8019A000).

3.1.4.3 working flow

figure 3-11 working flow



3.1.4.4 timing

figure 3-12 timing diagram

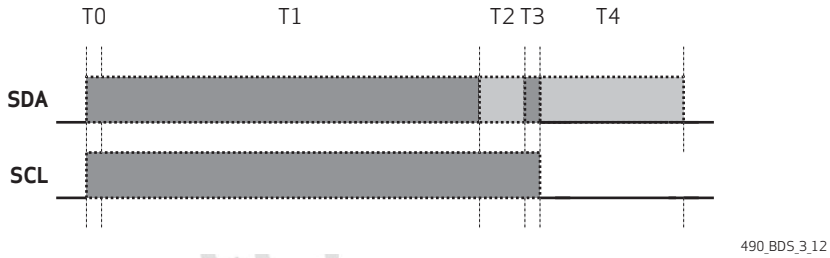


table 3-10 timing specifications

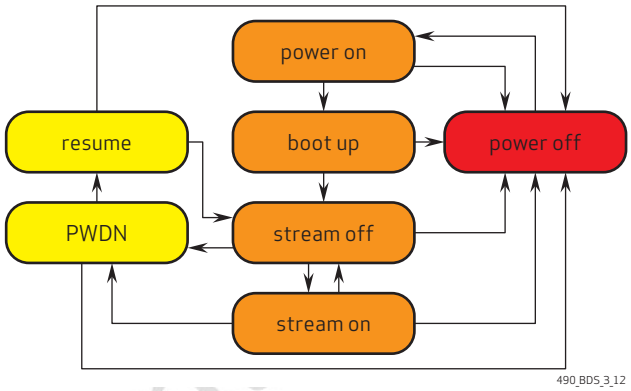
symbol	parameter[a]	requirement	time
T0	master reads boot flag and waits until value changes to 0xA1	SCCB_CLK = 400KHz	100 µs
T1	1. master downloads firmware to SRAM 0x801900F8 2. master writes SRAM address 0x8019FF98 to 0x190000 3. master enables OV490 SCCB slave burst writing function 4. master changes OV490 system clock to 160MHz 5. master sets boot flag to 0xC5	SCCB_CLK = 400KHz	235,000 µs
T2	OV490 checks firmware CRC	SYSTEM_CLK = 160MHz	2000 µs
T3	Master keep checking the boot flag to see whether boot flag = 0xC3	SCCB_CLK = 400KHz	100 µs
T4	OV490 start to boot itself and then initialize firmware. When initialize is done, then running flag is set as 0x02		200000 µs

a. Regarding to T4, the period from the moment (boot flag = 0xC3) to the moment (running flag = 0x02) is within 100~200ms. For the sake of safety, host had better check the running flag after 200ms when it detected boot flag = 0xC3 and the time out limit is suggested to set as 400ms.

3.2 OV490 status switch flow

The OV490 status switch flow is shown in **figure 3-13**.

figure 3-13 status switch flow



- stream on
  As orange the block shows after power on, the OV490 will boot up and enter stream off state. It can be switched to stream on state by user's command.
- stream off
  When the OV490 is in stream on state, it can be directly switched to stream off state. It also can be first switched to PWDN state and then resumed to stream off state as the yellow block shows.
- power off
  As red color shows, all of the OV490 status can be switched to power off state.

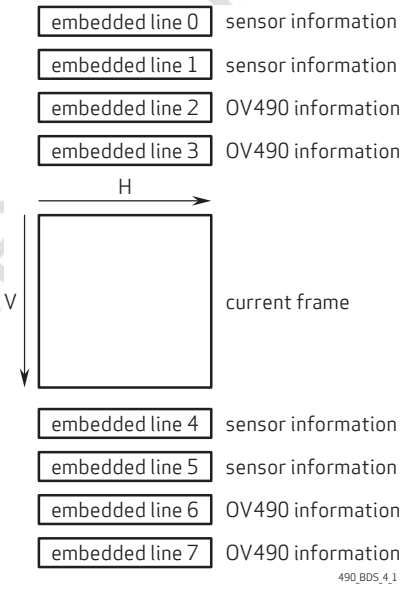
4 embedded lines

Additional information about setup and configuration of the sensor and the OV490 can be embedded in the video stream. The embedded data contains the programmable list values to describe the frame state including frame counter, exposure time and gain, etc. The OV490 supports embedded line mode in both MIPI and DVP interfaces. It can receive the input embedded line from the sensor and can also append its embedded line to the output stream.

4.1 OV490 and OV10640 embedded line structure

The OV490 can append one/two embedded lines at the beginning of frame and one/two lines at the end of frame for one sensor. Refer to **figure 4-1** for the position of each embedded line that can be configurable.

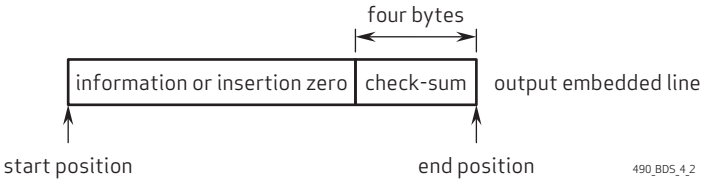
figure 4-1 embedded line diagram



4.1.1 format of output embedded line under normal case

The OV490 will put the valid information into an embedded line from the start position and the rest of the positions, except for the last four byte positions, will be filled by zero. The last four bytes positions are for check-sum (see **figure 4-2**).

figure 4-2 format of output embedded line



4.2 OV10640 embedded line organization after ISP processed by OV490

The OV10640 outputs two embedded lines at the beginning of frame and two lines at the end of frame. The OV10640 can also output one embedded line at the beginning of frame and one line at the end of frame. The OV490 will output the OV10640 embedded line following the rules listed below:

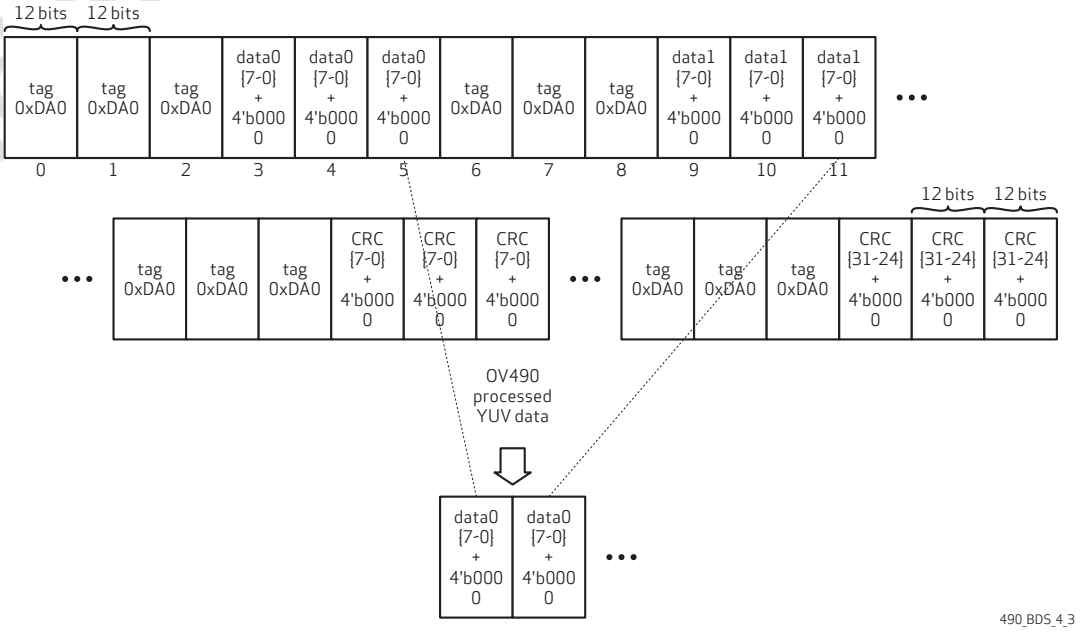
- all embedded lines output from the OV490 do not have tags, the OV490 will remove the tags when it receives the OV10640 sensor embedded line and then send them out.
- length of output embedded line is the same as a normal image line (e.g., output 1280x1080 by YUV422 mode, the size of embedded line is 1280x2 beat, where the beat means 12-bit data when DVP 12-bit interface. The blank space will be filled by 0x00, and this 0x00 will affect CRC result).

4.2.1 OV10640 output 3x12 bits RAW data to OV490

Refer to **figure 4-3** for the OV10640 output 3x12 bits RAW data embedded line format and the response format after processing by the OV490 ISP.

1. If even position in embedded line of the OV10640 is 0x00, it means that there is no more valid data in this embedded line.
2. Tag value of sensor is {0xDA, 4'b0000}

figure 4-3 OV10640 output 3x12 bits RAW data to OV490

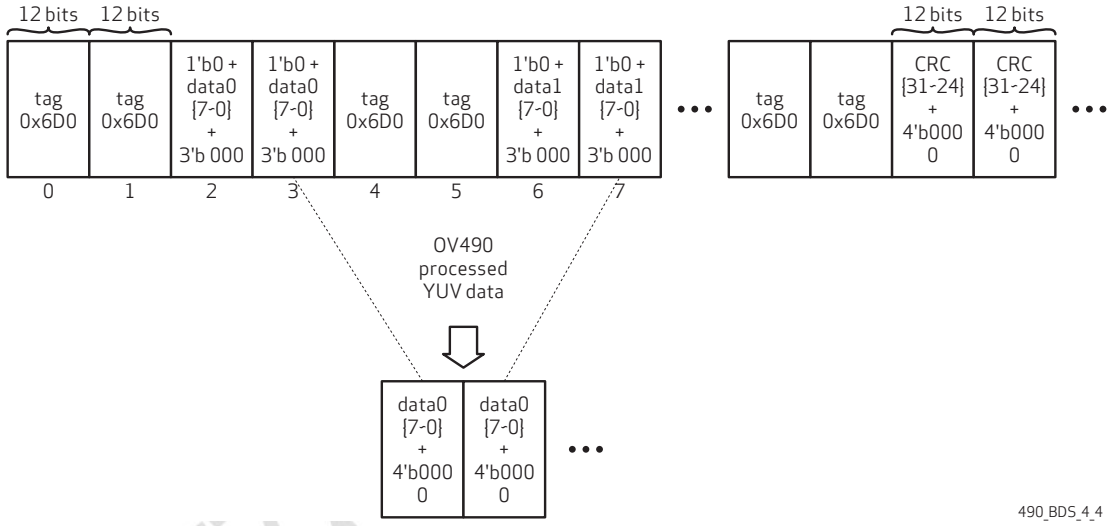


4.2.2 OV10640 output 2x11 bits RAW data to OV490

Refer to **figure 4-4** for the OV10640 output 2x11 bits RAW data embedded line format and the response format after processing by the OV490 ISP.

- Tag value of sensor is {1'b0, 0xDA, 3'b000}

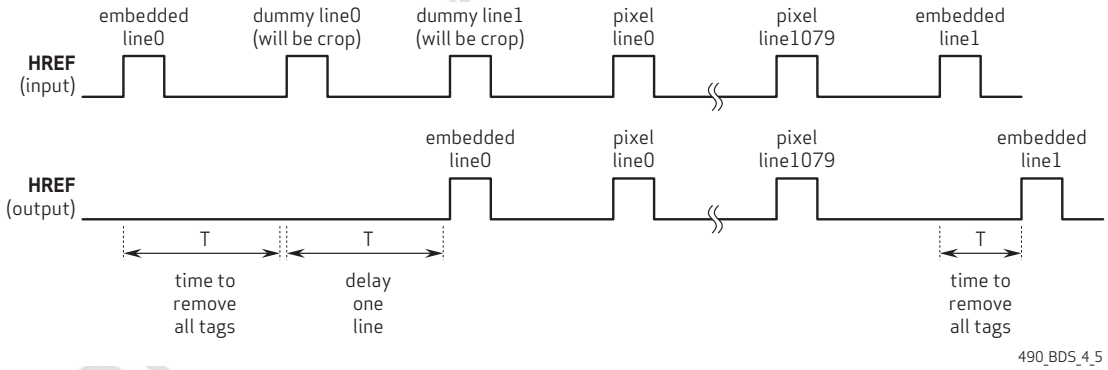
figure 4-4 OV10640 output 2x11 bits RAW data to OV490



4.3 OV10640 embedded line organization with OV490 ISP bypassed

This output format will be the same as the normal case in which the OV490 receives embedded line from the sensor and removes all of the tags. In this mode, the OV490 needs some time to remove all of the tags and the ISP has been disabled. The OV10640 needs to output two dummy lines at the beginning of frame. Refer to **figure 4-5** for DVP output timing when bypassing the ISP.

figure 4-5 output timing when bypassing ISP



4.4 CRC algorithm

The CRC32C (Castagnoli) algorithm is applied to protect the embedded data. The polynomial is 0x1EDC6F41, so formulas for one-bit update are:

```
Nxt[0] = d ^ C[31];
Nxt[1] = d ^ C[0] ^ C[31];
Nxt[2] = d ^ C[1] ^ C[31];
Nxt[3] = C[2];
Nxt[4] = d ^ C[3] ^ C[31];
Nxt[5] = d ^ C[4] ^ C[31];
Nxt[6] = C[5];
Nxt[7] = d ^ C[6] ^ C[31];
Nxt[8] = d ^ C[7] ^ C[31];
Nxt[9] = C[8];
Nxt[10] = d ^ C[9] ^ C[31];
Nxt[11] = d ^ C[10] ^ C[31];
```

```
Nxt[12] = d ^ C[11] ^ C[31];
Nxt[13] = C[12];
Nxt[14] = C[13];
Nxt[15] = C[14];
Nxt[16] = d ^ C[15] ^ C[31];
Nxt[17] = C[16];
Nxt[18] = C[17];
Nxt[19] = C[18];
Nxt[20] = C[19];
Nxt[21] = C[20];
Nxt[22] = d ^ C[21] ^ C[31];
Nxt[23] = d ^ C[22] ^ C[31];
Nxt[24] = C[23];
Nxt[25] = C[24];
Nxt[26] = d ^ C[25] ^ C[31];
Nxt[27] = C[26];
Nxt[28] = C[27];
Nxt[29] = C[28];
Nxt[30] = C[29];
Nxt[31] = C[30];
```

Note that C is the previous CRC (reset to 0) and D is the new data bit. The CRC is sent as the last data of the embedded data, most significant byte first.

4.5 notes

The OV490 has two limitations when it receives embedded line from OV10640:

- valid byte number of embedded line must be times 4.
- embedded information must occupy more than one line, that means the CRC will occur at the second line. CRC cannot occur at the start-point of the second line, it must occur after at least 4 bytes in the second line.

Also, note:

- for MIPI output mode, the OV490 cannot support output embedded data and image data via different virtual channels. Embedded data and image data can be set as different data type (DT) in the same channel and the host can receive them based on the DT.
- for DVP output mode, embedded line data will combine with image data and output together. For example, if there are four embedded lines and image data resolution is 640x480, then the DVP output resolution is 640x484.

5 host command control

5.1 host control concept

Host command control has the following features/advantages:

- Host does not need to directly access the OV490's trivial 32-bit registers. Simply, it may send a task command with one or more parameters to the OV490 and wait for the OV490 to accomplish the task by checking one status register of the OV490 or one GPIO interrupt of the OV490.
- Host does not need to care about task timing. For some tasks, which need to be accomplished in the interval of adjacent frames, the OV490's firmware can arrange to do this in the interrupt of EOF. While if the host directly accesses the OV490's registers, it cannot ensure the correct timing and may cause a bad image output.
- Fast speed when lots of registers need to be accessed. DMA of the OV490 can be arranged to speed up these kinds of tasks.
- For host control via I2C protocol, there can be a maximum of but not limited to 256 types of host control depending on the design of host control protocol.

5.2 general host control flow chart

5.2.1 definitions

```
#define OV490_STATUS_ADDR          (0x80195ffc)
#define HOST_CMD_PARA_ADDR         (0x80195000)
#define HOST_CMD_RESULT_ADDR       (0x80195000)
#define OV490_HOST_INT_ADDR        (0x808000c0)
#define STATUS_FINISH              (0x99)
#define STATUS_OUTGOING            (0x88)
#define STATUS_ERROR               (0x55)

#define CMD_BRIGHTNESS_SET         (0xf1)
#define CMD_SATURATION_SET         (0xf3)
#define CMD_HUE_SET                (0xf5)
#define CMD_FRAMERATE_SET          (0xf7)
#define CMD_GAMMA_SET              (0xf9)
#define CMD_SHARPNESS_SET          (0xfb)
#define CMD_CONTRAST_SET           (0xfd)

#define CMD_GROUPWRITE_SET         (0xe1)
#define CMD_STREAMING_CTRL         (0xe2)
#define CMD_CONTEXT_SWITCH_CONFIG (0xe3)
```

```
#define CMD_CONTEXT_SWITCH_CTRL   (0xe4)
#define CMD_MULT_CMD              (0xe5)
#define CMD_GPIO_SET              (0xe6)
#define CMD_GPIO_GET              (0xe7)
#define CMD_FORMAT_SET            (0xe8)
#define CMD_TEMP_GET              (0xe9)
#define CMD_EXPOSURE_GAIN_SET     (0xea)
#define CMD_AWBGAIN_SET           (0xeb)
#define CMD_DENOISE_SET           (0xec)
#define CMD_TONECURVE_SET         (0xed)
#define CMD_COMB_WEIGHT_SET       (0xee)
#define CMD_AEC_WEIGHT_SET        (0xd2)
#define CMD_AWB_ROI_SET           (0xd3)
#define CMD_TONEMAPPING_ROI_SET   (0xd4)
#define CMD_STAT_ROI_SET          (0xd5)
#define CMD_TESTPATTERN_SET       (0xd6)
#define CMD_MTF_SET               (0xd7)
#define CMD_LENC_SET              (0xd8)
#define CMD_BLC_SET               (0xd9)
#define CMD_GROUPWRITE_LAUNCH     (0xda)
#define CMD_EMBLINE_CTRL          (0xdb)
#define CMD_MIRRFLIP_CTRL         (0xdc)
#define CMD_EXTRA_VTS_SET         (0xde)
#define CMD_SNR_REG_ACCESS        (0xc1)
#define CMD_POSTAWBGAIN_SET       (0xc2)
#define CMD_CROP_SET              (0xc3)
#define CMD_FRAMESYNC             (0xc4)
#define CMD_BANDING_SET           (0xc5)
#define CMD_TOPEMB_SET            (0xc7)
#define CMD_FWREG_ACCESS          (0x35)
#define CMD_FADE_CTRL             (0x37)
```

5.2.2 host command list

The host commands are programmable by firmware. They can be added or removed according to the project requirements. For reference of basic commands, see **table 5-1**.

table 5-1 host command list (sheet 1 of 2)

function	description	reference
color hue level adjust	adjust image color hue value	**section 5.4.1**
color saturation	adjust image color saturation	**section 5.4.2**
brightness level adjust	adjust image brightness	**section 5.4.3**
sharpness level adjust	enhance or weaken image sharpness	**section 5.4.4**
contrast level adjust	adjust image contrast	**section 5.4.5**
manual exposure gain set	disable AEC/AGC and allow manual configuration of sensor exposure and gain values	**section 5.4.6**
manual white balance gain set	disable AWB and allow manual setting of R/G/B gain white balance value	**section 5.4.7**
de-noise level set	set eight steps of de-noise levels for RAWDNS, CIPDNS, RGBDNS, and UVDNS	**section 5.4.8**
HDR combination weight set	set combine weight	**section 5.4.9**
AEC weight set	set AEC weights of 13 windows for L and S channels	**section 5.4.10**
AWB ROI set	enable AWB statistics and allow configuration of the ROI statistics position	**section 5.4.11**
tone mapping ROI set	configure tone mapping ROI	**section 5.4.12**
manual tone mapping curve set	disable auto tone mapping curve and allow manual setting of tone mapping curve	**section 5.4.13**
gamma set	set RGB gamma curve	**section 5.4.14**
GPIO control	configure state of up to six GPIO pins	**section 5.4.15**
frame rate change	change sensor frame rate	**section 5.4.16**
stream ON/OFF	enable stream output or stop stream output	**section 5.4.17**
test pattern select	enable three kinds of test patterns output	**section 5.4.18**
output format select	select YUV and RAW formats for L/S/VS	**section 5.4.19**
BLC set	set sensor BLC target	**section 5.4.20**
temperature read	support reading of measured temperature	**section 5.4.21**
YUV statistics windows set	configure up to five ROI windows	**section 5.4.22**
AB context switch	enable AB frame context switch	**section 5.4.23**

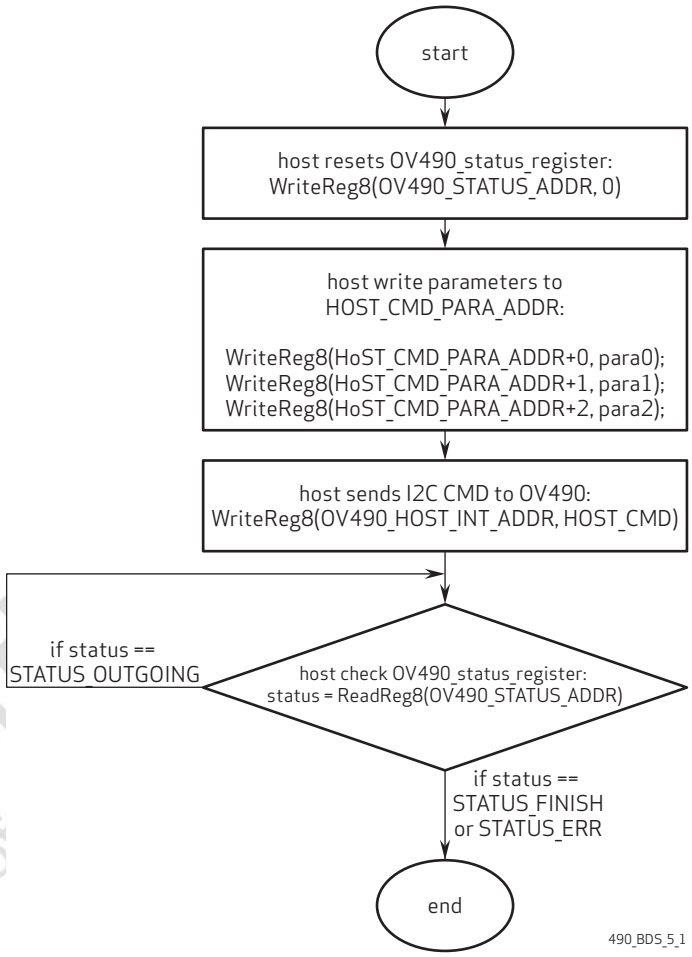
table 5-1 host command list (sheet 2 of 2)

function	description	reference
single sub-pixel RAW12 mode	select long or short exposure channel input and YUV output	section 5.4.24
mirror/flip control	enable image mirror/flip	section 5.4.25
LENC parameter set	set 384 LENC parameters	section 5.4.26
embedded line control	enable embedded line output and allow configuration of embedded line data	section 5.4.27
group write launch	enable four groups of register writes and launch	section 5.4.28
extra VTS control	sensor extra VTS control	section 5.4.29
post AWB gain set	set post AWB gain	section 5.4.30
access sensor register	sensor register read write	section 5.4.31
access firmware register	firmware register read write	section 5.4.32
frame sync (+/- VTS)	sensor VTS adjustment for frame sync	section 5.4.33
cropping	enable output of smaller window from original size	section 5.4.34
fade in/out switch	enable fade in/out effect switch	section 5.4.35
50Hz/60Hz banding control	switch between 50Hz and 60Hz banding filter	section 5.4.36
multi-commands	send multi-command	section 5.5

5.2.3 working flow

figure 5-1 displays the host control flow chart for a set operation.

figure 5-1 host control flow chart



There are three host command statuses in **figure 5-1**:

- STATUS_FINISH: denotes the host command has been implemented finish, host can send next command as needed
- STATUS_ERR: denotes that the command parameters are set to error, the OV490 firmware rejects this command and the command is not implemented again
- STATUS_ONGOING: enables all the host commands
  - for commands which are implemented immediately when the OV490 receives the information, the firmware does not place the status to enable
  - for commands that need special timing to be implemented (in VSYNC timing or in several frames), the OV490 firmware is enabled for this status

5.3 host control API

```
void HostControl_Set(unsigned char host_cmd, unsigned char* para, unsigned short number)
{
      unsigned short i = 0;

      //Host reset OV490_status_register;
      I2C_Write8(OV490_STATUS_ADDR, 0);

      // Host write parameters to OV490 SRAM;
      For (i=0; i<number; i++)
      {
           I2C_Write8(HOST_CMD_PARA_ADDR+i, *(para+i));
      }

      //Host send I2C CMD to OV490 to start task:
      I2C_Write8(OV490_HOST_INT_ADDR, host_cmd);

      // Host check OV490_status_register val to determine whether the task is finished
      or not
      while (I2C_Read8(OV490_STATUS_ADDR) != STATUS_FINISH)
      {
           // if task is not finished, wait n ms;
           wait(n);
      }
}

void HostControl_Get(unsigned char host_cmd, unsigned char* para, unsigned short number)
{
      unsigned short i = 0;

      //Host reset OV490_status_register;
      I2C_Write8(OV490_STATUS_ADDR, 0);

      //Host send I2C CMD to OV490 to start task:
      I2C_Write8(OV490_HOST_INT_ADDR, host_cmd);

      // Host check OV490_status_register val to determine whether the task is finished
      or not
      while (I2C_Read8(OV490_STATUS_ADDR) != STATUS_FINISH)
      {
           // if task is not finished, wait n ms;
           Host_wait(n);
      }

      // Host read results from OV490 SRAM;
      For (i=0; i<number; i++)
      {
           *(para+i) = I2C_Read8(HOST_CMD_RESULT_ADDR+i);
      }
}

void HostControl_GetEx(unsigned char host_cmd, unsigned char* paraIn, unsigned short
numberIn, unsigned char* paraOut, unsigned short numberOut)
      {
           unsigned short i = 0;
           //Host reset OV490_status_register;
           I2C_Write8(OV490_STATUS_ADDR, 0);

                // Host write parameters to OV490 SRAM;
                For (i=0; i< numberIn; i++)
```

```
        {
              I2C_Write8(HOST_CMD_PARA_ADDR+i, *(paraIn+i));
        }

        //Host send I2C CMD to OV490 to start task:
        I2C_Write8(OV490_HOST_INT_ADDR, host_cmd);

        // Host check OV490_status_register val to determine whether the task is
        finished or not while (I2C_Read8(OV490_STATUS_ADDR) != STATUS_FINISH)
        {
        // if task is not finished, wait n ms;
        Host_wait(n);
        }

        // Host read results from OV490 SRAM;
        For (i=0; i< numberOut; i++)
        {
        *( paraOut +i) = I2C_Read8(HOST_CMD_RESULT_ADDR+i);
        }
}
```

Note that the first parameter of `I2C_Write8` and `I2C_Read8` functions are 32-bit values. The host needs to access the 32-bit address by the following format:

- for read operation (e.g., `I2C_Read8(0x80190000)`):

```
I2C_Write(0xfffd, 0x80);
I2C_Write(0xfffe, 0x19); //first change the high 16bit base address to 8019
I2C_Read(0x0000); // then read the low 16bit address (0x0000) directly
```

- for write operation (e.g., `I2C_Write8(0x808000C0, 0xF1)`):

```
I2C_Write(0xfffd, 0x80);
I2C_Write(0xfffe, 0x80); //first change the high 16bit base address to 8080
I2C_Write(0x00c0, 0xf1); // then write 0xf1 to the low 16bit address (0x00c0)directly.
```

Note the high 16-bit base address needs to be set only once if it is the same for more than one register. For example:

```
I2C_Write8(0x80190000, 0xf1);
I2C_Write8(0x80190002, 0xf2);
I2C_Write8(0x80190004, 0xf3);
```

Can be implemented as below:

```
I2C_Write(0xfffd, 0x80);
I2C_Write(0xfffe, 0x19); //Set the high 16bit base address to 8019 only once!
I2C_Write(0x0000, 0xf1);
I2C_Write(0x0002, 0xf2);
I2C_Write(0x0004, 0xf3);
```

5.4 single host command

5.4.1 color hue level adjust

[Parameter]

Parameter number: 1

Parameter 0: level, range: [0, 23]

Related hardware registers: 0x80207B80, 0x80207B97, 0x80207B98

[Usage]

Suppose host wants to set hue level as 5, then:

```
unsigned char level = 5;
HostControl_Set(CMD_HUE_SET, &level, 1);
```

5.4.2 color saturation level adjust

[Parameter]

Parameter number: 1

Parameter 0: level, range: [0, 7]

Related hardware registers: 0x80207B80, 0x80207B91, 0x80207B92

[Usage]

Suppose host wants to set saturation level as 7, then:

```
unsigned char level = 7;
HostControl_Set(CMD_SATURATION_SET, &level, 1);
```

5.4.3 brightness level adjust

[Parameter]

Parameter number: 4

Related hardware and firmware registers:

TYPE_SDE: 0x80207B80, 0x80207B84

TYPE_TONEMAP: pLogTargetY

TYPE_GAMMA: bGlobalGammaShiftEnable, nGlobalGammaShift

table 5-2 brightness level adjust parameters

index	description	range/bits
0	brightness type	0: TYPE_SDE 1: TYPE_TONEMAP 2: TYPE_GAMMA
1	brightness level	TYPE_SDE: [0x00, 0x10] TYPE_TONEMAP: [0x00, 0x06]
2~3	reserved	default: 0

[Usage 1]

Suppose host wants to set SDE brightness level to 3, then:

```
unsigned char param[4] = {0, 3, 0, 0};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
```

[Usage 2]

Suppose the host does not want to use the predefined TONEMAP 7-level adjusting, then it can set the specific value to achieve fine tuning:

```
unsigned char param[4] = {0x01, 0x40, 0x00, 0x00};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
unsigned char param[4] = {0x01, 0x45, 0x00, 0x00};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
unsigned char param[4] = {0x01, 0x6A, 0x00, 0x00};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
```

[Usage 3]

Suppose the host does not want to use the predefined GAMMA 7-level adjusting, then it can be set to the specific value to achieve more fine adjusting:

```
unsigned char param[4] = {0x02, 0x40, 0x00, 0x00};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
unsigned char param[4] = {0x02, 0x45, 0x00, 0x00};
HostControl_Set(CMD_BRIGHTNESS_SET, param, 4);
```

5.4.4 sharpness level adjust

[Parameter]

Parameter number: 1

Parameter 0: level, Range: [0, 10]

Related hardware registers: 0x8020788A, 0x8020788B

[Usage]

Suppose host wants to set sharpness level to 5, then:

```
unsigned char level = 5;
HostControl_Set(CMD_SHARPNESS_SET, &level, 1);
```

5.4.5 contrast level adjust

[Parameter]

Parameter number: 1

Parameter 0: level, range: [0, 10]

Related firmware registers: nLowContrastLevel, nHighContrastLevel

[Usage]

Suppose host wants to set contrast level to 5, then:

```
unsigned char level = 5;
HostControl_Set(CMD_CONTRAST_SET, &level, 1);
```

5.4.6 manual exposure/gain set

[Parameter]

Parameter number: 13

Related firmware registers: bManualAECEnable, pAECManualExp, pAECManualGain

table 5-3 manual exposure/gain set parameters (sheet 1 of 2)

index	description	range/bits
0	manual AWB gain set enable	Bit[0]: Manual AWB gain set enable 0: Auto mode 1: Manual mode
1	manual exposure for L[15:8]	Bit[7:0]
2	manual exposure for L[7:0]	Bit[7:0]
3	manual exposure for S[15:8]	Bit[7:0]
4	manual exposure for S[7:0]	Bit[7:0]
5	manual exposure for VS[15:8]	Bit[7:0]

table 5-3 manual exposure/gain set parameters (sheet 2 of 2)

index	description	range/bits
6	manual exposure for VS[7:0]	Bit[7:0]
7	manual gain for L[15:8]	Bit[7:0]
8	manual gain for L[7:0]	Bit[7:0]
9	manual gain for S[15:8]	Bit[7:0]
10	manual gain for S[7:0]	Bit[7:0]
11	manual gain for VS[15:8]	Bit[7:0]
12	manual gain for VS[7:0]	Bit[7:0]

[Usage]

Suppose host wants to set manual exposures and gains for three channels as below:

L channel's exposure: *0x0080*

S channel's exposure: *0x0050*

VS channel's exposure: *0x002A*

L channel's gain: *0x100*

S channel's gain: *0x100*

VS channel's gain: *0x100*

Then:

```
unsigned char param[13] = {0x01, 0x00, 0x80, 0x00, 0x50, 0x00, 0x2A, 0x01, 0x00,
                           0x01, 0x00, 0x01, 0x00};
HostControl_Set(CMD_EXPOSURE_GAIN_SET,  param, 13);
```

5.4.7 manual AWB R/G/B gain set

[Parameter]

Parameter number: 25

Related hardware and firmware registers: 0x80800005, pManualAWBGain[3][4]

table 5-4 manual AWB R/G/B gain set parameters (sheet 1 of 2)

index	description	range/bits
0	manual exposure enable	Bit[0]: Manual AWB gain set enable 0: Auto mode 1: Manual mode
1	manual gain for L exposure channel RED component[11:8]	Bit[3:0]

table 5-4 manual AWB R/G/B gain set parameters (sheet 2 of 2)

index	description	range/bits
2	manual gain for L exposure channel RED component[7:0]	Bit[7:0]
3	manual gain for L exposure channel GREEN R component[11:8]	Bit[3:0]
4	manual gain for L exposure channel GREEN R component[7:0]	Bit[7:0]
5	manual gain for L exposure channel GREEN B component[11:8]	Bit[3:0]
6	manual gain for L exposure channel GREEN B component[7:0]	Bit[7:0]
7	manual gain for L exposure channel BLUE component[11:8]	Bit[3:0]
8	manual gain for L exposure channel BLUE component[7:0]	Bit[7:0]
9	manual gain for S exposure channel RED component[11:8]	Bit[3:0]
10	manual gain for S exposure channel RED component[7:0]	Bit[7:0]
11	manual gain for S exposure channel GREEN R component[11:8]	Bit[3:0]
12	manual gain for S exposure channel GREEN R component[7:0]	Bit[7:0]
13	manual gain for S exposure channel GREEN B component[11:8]	Bit[3:0]
14	manual gain for S exposure channel GREEN B component[7:0]	Bit[7:0]
15	manual gain for S exposure channel BLUE component[11:8]	Bit[3:0]
16	manual gain for S exposure channel BLUE component[7:0]	Bit[7:0]
17	manual gain for VS exposure channel RED component[11:8]	Bit[3:0]
18	manual gain for VS exposure channel RED component[7:0]	Bit[7:0]
19	manual gain for VS exposure channel GREEN R component[11:8]	Bit[3:0]
20	manual gain for VS exposure channel GREEN R component[7:0]	Bit[7:0]
21	manual gain for VS exposure channel GREEN B component[11:8]	Bit[3:0]
22	manual gain for VS exposure channel GREEN B component[7:0]	Bit[7:0]
23	manual gain for VS exposure channel BLUE component[11:8]	Bit[3:0]
24	manual gain for VS exposure channel BLUE component[7:0]	Bit[7:0]

[Usage]

Suppose host wants to set manual AWB gains as below:

L exposure channel:	*RED component's gain:*	*0x100*
	GREEN_R component's gain:	*0x100*
	GREEN_B component's gain:	*0x100*
	BLUE component's gain:	*0x100*
S exposure channel:	*RED component's gain:*	*0x110*
	GREEN_R component's gain:	*0x110*
	GREEN_B component's gain:	*0x110*
	BLUE component's gain:	*0x110*
VS exposure channel:	*RED component's gain:*	*0x120*
	GREEN_R component's gain:	*0x120*
	GREEN_B component's gain:	*0x120*
	BLUE component's gain:	*0x120*

Then:

```
unsigned char param[25] = {0x01, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x10,
                           0x01, 0x10, 0x01, 0x10, 0x01, 0x20, 0x01, 0x20, 0x01,
                           0x20};
HostControl_Set(CMD_AWBGAIN_SET, param, 25);
```

5.4.8 de-noise level set

5.4.8.1 RAW DNS

[Parameter]

Parameter number: 25

Related hardware registers: 0x80206300~0x80206307(L), 0x8020630A~0x80206311(S), 0x80206314~0x8020631B (VS)

table 5-5 de-noise level set parameters (sheet 1 of 2)

index	description	range/bits
0	DNS type	0
1	NoiseList0 for L exposure channel RAW de-noise	Bit[7:0]
2	NoiseList1 for L exposure channel RAW de-noise	Bit[7:0]
...	...	...
8	NoiseList7 for L exposure channel RAW de-noise	Bit[7:0]
9	NoiseList0 for S exposure channel RAW de-noise	Bit[7:0]
10	NoiseList1 for S exposure channel RAW de-noise	Bit[7:0]

table 5-5 de-noise level set parameters (sheet 2 of 2)

index	description	range/bits
...	...	...
16	NoiseList7 for S exposure channel RAW de-noise	Bit[7:0]
17	NoiseList0 for VS exposure channel RAW de-noise	Bit[7:0]
18	NoiseList1 for VS exposure channel RAW de-noise	Bit[7:0]
...	...	...
24	NoiseList7 for VS exposure channel RAW de-noise	Bit[7:0]

[Usage]

Suppose host wants to set RAW de-noise level as shown below:

L : *{0x04, 0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40}*

S : *{0x04, 0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40}*

VS : *{0x04, 0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40}*

Then:

```
unsigned char dns[25]= { 0x00, 0x04, 0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40,
                         0x04, 0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40, 0x04,
                         0x08, 0x10, 0x18, 0x20, 0x30, 0x40, 0x40};
HostControl_Set(CMD_DENOISE_SET, dns, 25);
```

5.4.8.2 RGB DNS

[Parameter]

Parameter number: 13

Related hardware registers: 0x80207B02~0x80207B0D

table 5-6 RGB DNS parameters (sheet 1 of 2)

index	description	range/bits
0	DNS type	1
1	Bit[5:3]: Y noise list 1 Bit[2:0]: Y noise list 0	Bit[5:0]
2	Bit[5:3]: Y noise list 3 Bit[2:0]: Y noise list 2	Bit[5:0]
3	Bit[5:3]: Y noise list 5 Bit[2:0]: Y noise list 4	Bit[5:0]

table 5-6 RGB DNS parameters (sheet 2 of 2)

index	description	range/bits
4	Bit[5:3]: Y noise list 7 Bit[2:0]: Y noise list 6	Bit[5:0]
5	Bit[5:0]: UV noise list 0	Bit[5:0]
6	Bit[5:0]: UV noise list 1	Bit[5:0]
7	Bit[5:0]: UV noise list 2	Bit[5:0]
8	Bit[5:0]: UV noise list 3	Bit[5:0]
9	Bit[5:0]: UV noise list 4	Bit[5:0]
10	Bit[5:0]: UV noise list 5	Bit[5:0]
11	Bit[5:0]: UV noise list 6	Bit[5:0]
12	Bit[5:0]: UV noise list 7	Bit[5:0]

[Usage]

Suppose host wants to set RGB de-noise level as shown below:

{0x09, 0x09, 0x09, 0x09, 0x04, 0x08, 0x0A, 0x10,0x1C,0x20,0x22,0x24}

Then:

```
unsigned char dns[13]= { 0x01, 0x09, 0x09, 0x09, 0x09, 0x04, 0x08, 0x0a,
                         0x10,0x1c,0x20,0x22,0x24};
HostControl_Set(CMD_DENOISE_SET, dns,13);
```

5.4.8.3 CIP DNS

[Parameter]

Parameter number: 9

Related hardware registers: 0x80207882~0x80207889

table 5-7 CIP DNS parameters (sheet 1 of 2)

index	description	range/bits
0	DNS type	2
1	NoiseList [0]	Bit[7:0]
2	NoiseList [1]	Bit[7:0]
3	NoiseList [2]	Bit[7:0]
4	NoiseList [3]	Bit[7:0]
5	NoiseList [4]	Bit[7:0]

table 5-7 CIP DNS parameters (sheet 2 of 2)

index	description	range/bits
6	NoiseList [5]	Bit[7:0]
7	NoiseList [6]	Bit[7:0]
8	NoiseList [7]	Bit[7:0]

[Usage]

Suppose host wants to set CIP de-noise level as shown below:

{0x41, 0x42, 0x43, 0x45, 0x48, 0x4A, 0x54, 0x5E}

Then:

```
unsigned char dns[9]= { 0x02, 0x41, 0x42, 0x43, 0x45, 0x48, 0x4a, 0x54, 0x5e};
HostControl_Set(CMD_DENOISE_SET, dns,9);
```

5.4.8.4 UV DNS

[Parameter]

Parameter number: 9

Related hardware registers: 0x80207D00~0x80207D07

table 5-8 UV DNS parameters

index	description	range/bits
0	DNS type	3
1	NoiseList [0]	Bit[7:0]
2	NoiseList [1]	Bit[7:0]
3	NoiseList [2]	Bit[7:0]
4	NoiseList [3]	Bit[7:0]
5	NoiseList [4]	Bit[7:0]
6	NoiseList [5]	Bit[7:0]
7	NoiseList [6]	Bit[7:0]
8	NoiseList [7]	Bit[7:0]

[Usage]

Suppose host wants to set UV de-noise level as below,

{0x9, 0xA, 0xB, 0xD, 0x10, 0x12, 0x1C, 0x1E}

Then:

```
unsigned char dns[9]= { 0x03, 0x9, 0xa, 0xb, 0xd, 0x10, 0x12, 0x1c, 0x1e};
HostControl_Set(CMD_DENOISE_SET, dns,9);
```

5.4.9 HDR combination weight set

[Parameter]

Parameter number: 16

Related hardware registers: 0x80206804~0x80206813

table 5-9 HDR combination weight set parameters

index	description	range/bits
0	combine weight [0][0]	Bit[7:0]
1	combine weight [0][1]	Bit[7:0]
2	combine weight [0][2]	Bit[7:0]
3	combine weight [0][3]	Bit[7:0]
4	combine weight [1][0]	Bit[7:0]
5	combine weight [1][1]	Bit[7:0]
6	combine weight [1][2]	Bit[7:0]
7	combine weight [1][3]	Bit[7:0]
8	combine weight [2][0]	Bit[7:0]
9	combine weight [2][1]	Bit[7:0]
10	combine weight [2][2]	Bit[7:0]
11	combine weight [2][3]	Bit[7:0]
12	combine weight [3][0]	Bit[7:0]
13	combine weight [3][1]	Bit[7:0]
14	combine weight [3][2]	Bit[7:0]
15	combine weight [3][3]	Bit[7:0]

[Usage]

Suppose host wants to set HDR combination weight as below,

[3][0]:	0x80	[3][1]:	0x80	[3][2]:	0x00	[3][3]:	0x00
[2][0]:	0x80	[2][1]:	0x60	[2][2]:	0x00	[2][3]:	0x00
[1][0]:	0x80	[1][1]:	0x80	[1][2]:	0x40	[1][3]:	0x20
[0][0]:	0x80	[0][1]:	0x80	[0][2]:	0x60	[0][3]:	0x40

Then:

```
unsigned char weight[16]= { 0x80, 0x80, 0x60, 0x40, 0x80, 0x80, 0x40, 0x20, 0x80,
                            0x60, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00};
HostControl_Set(CMD_COMB_WEIGHT_SET, weight, 16);
```

5.4.10 AEC weight set

[Parameter]

Parameter number: 13

Related hardware registers: 0x80207022~ 0x8020702E

table 5-10 AEC weight set parameters

index	description	range/bits
0	weights[][0]	Bit[7:4]: Weights[0][0] Bit[3:0]: Weights[1][0]
1	weights[][1]	Bit[7:4]: Weights[0][1] Bit[3:0]: Weights[1][1]
...	...	...
12	weights[][12]	Bit[7:4]: Weights[0][12] Bit[3:0]: Weights[1][12]

[Usage]

Suppose host wants to set AEC weight as below:

L/VS weight: {2,2,2,2,2,2,2,2,2,2,2,2,2};

S weight: {1,1,1,1,1,1,1,1,1,1,1,1,1};

Then:

```
unsigned char weight[13]= {0x21, 0x21, 0x21, 0x21, 0x21, 0x21, 0x21, 0x21, 0x21,
                           0x21, 0x21, 0x21,0x21};
HostControl_Set(CMD_AEC_WEIGHT_SET, weight, 16);
```

5.4.11 AWB ROI set

[Parameter]

Parameter number: 9

Related hardware registers: 0x80206C00~0x80206C08

table 5-11 AWB ROI set parameters

index	description	range/bits
0	advanced AWB statistic ROI enable	0: Disable 1: Enable
1	statistic ROI left coordinate[11:8]	Bit[3:0]
2	statistic ROI left coordinate[7:0]	Bit[7:0]
3	statistic ROI top coordinate[10:8]	Bit[2:0]
4	statistic ROI top coordinate[7:0]	Bit[7:0]
5	statistic ROI right coordinate[11:8]	Bit[3:0]
6	statistic ROI right coordinate[7:0]	Bit[7:0]
7	statistic ROI bottom coordinate[10:8]	Bit[2:0]
8	statistic ROI bottom coordinate[7:0]	Bit[7:0]

[Usage]

Suppose host wants to enable and set AWB ROI window:

left: 0x00, right: 0x0100, top: 0x00; bottom: 0x0200

then:

```
unsigned char roi[9]= {0x1, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x02, 0x00};
HostControl_Set(CMD_AWB_ROI_SET, roi, 9);
```

5.4.12 tone mapping ROI set

[Parameter]

Parameter number: 9

Related hardware registers: 0x80207400, 0x8020740C~0x80207414

table 5-12 tone mapping ROI set parameters

index	description	range/bits
0	enable tone mapping ROI	0: Disable 1: Enable
1	ROI top boundary line[10:8]	Bit[2:0]
2	ROI top boundary line[7:0]	Bit[7:0]
3	ROI bottom boundary line[10:8]	Bit[2:0]
4	ROI bottom boundary line[7:0]	Bit[7:0]
5	ROI left boundary column[11:8]	Bit[3:0]
6	ROI left boundary column[7:0]	Bit[7:0]
7	ROI right boundary column[11:8]	Bit[3:0]
8	ROI right boundary column[7:0]	Bit[7:0]

[Usage]

Suppose host wants to set tone mapping ROI:

{left: 0, right: 0x280, top: 0, bottom: 0x1E0}

Then:

```
unsigned char roi[9] = {0x01,//enable
                        0x00, 0x00, //top
                        0x01, 0xe0, //bottom
                        0x00, 0x00, //left
                        0x02, 0x80 //right
                        };
HostControl_Set(HOSTCMD_TONEMAPPING_ROI_SET, roi, 9);
```

5.4.13 manual tone mapping curve set

[Parameter]

Parameter number: 33

Related firmware registers: pManualToneCurve[16], bAutoToneMappingCurve

table 5-13 manual tone mapping curve set parameters

index	description	range/bits
0	manual set enable	Bit[0]: Manual set enable 0: Auto mode 1: Manual mode
1	manual tone curve[0] high bits	Bit[1:0]
2	manual tone curve[0] low bits	Bit[7:0]
3	manual tone curve[1] high bits	Bit[1:0]
4	manual tone curve[1] low bits	Bit[7:0]
...	...	...
31	manual tone curve[15] high bits	Bit[1:0]
32	manual tone curve[15] low bits	Bit[7:0]

[Usage]

Suppose host wants to set tone mapping curve as below:

{1, 16, 32, 64, 128, 181, 256, 304, 362, 431, 512, 609, 724, 861, 939, 1023}

Then:

```
unsigned char curve[33] = {0x01, 0x00, 0x01, 0x00, 0x10, 0x00, 0x20, 0x00, 0x40,
                           0x00, 0x80, 0x00, 0xB5, 0x01, 0x00, 0x01, 0x30, 0x01,
                           0x6A, 0x01, 0xAF, 0x02, 0x00, 0x02, 0x61, 0x02, 0xD4,
                           0x03, 0x5D, 0x03, 0xAB, 0x03, 0xFF};
HostControl_Set(CMD_TONECURVE_SET, curve, 33);
```

5.4.14 gamma set

[Parameter]

Parameter number: 15

Related hardware registers: 0x80207A01~0x80207A1D

table 5-14 gamma set parameters

index	description	range/bits
0	gamma list00 for Y	Bit[7:0]
1	gamma list01 for Y	Bit[7:0]
2	gamma list02 for Y	Bit[7:0]
3	gamma list03 for Y	Bit[7:0]
4	gamma list04 for Y	Bit[7:0]
...	...	...
14	gamma list0e for Y	Bit[7:0]

[Usage]

Suppose host wants to set gamma curve as below:

{0x12, 0x20, 0x3B, 0x5D, 0x6A, 0x76, 0x81, 0x8B, 0x96, 0x9E, 0xAE, 0xBC, 0xCF, 0xDE, 0xEC}

Then:

```
unsigned char gamma[15] = {0x12, 0x20, 0x3b, 0x5d, 0x6a, 0x76, 0x81, 0x8b, 0x96,
                           0x9e, 0xae, 0xbc, 0xcf, 0xde, 0xec};
HostControl_Set(CMD_GAMMA_SET, gamma, 15);
```

5.4.15 GPIO control

[Parameter]

Parameter number: 2

Related hardware registers: 0x80800050, 0x80800054, 0x80800058

table 5-15 GPIO control parameters

index	description	range/bits
0	GPIO direction	Bit[7:6]: Reserved Bit[5]: Direction select bit for GPIO5 0: Input 1: Output Bit[4]: Direction select bit for GPIO4 0: Input 1: Output Bit[3]: Direction select bit for GPIO3 0: Input 1: Output Bit[2]: Direction select bit for GPIO2 0: Input 1: Output Bit[1]: Direction select bit for GPIO1 0: Input 1: Output Bit[0]: Direction select bit for GPIO0 0: Input 1: Output
1	GPIO value	Bit[7:6]: Reserved Bit[5]: GPIO5 output value 0: Low 1: High Bit[4]: GPIO4 output value 0: Low 1: High Bit[3]: GPIO3 output value 0: Low 1: High Bit[2]: GPIO2output value 0: Low 1: High Bit[1]: GPIO1 output value 0: Low 1: High Bit[0]: GPIO0 output value 0: Low 1: High

[Usage]

Suppose host wants to set GPIO5~GPIO 0 output high, then:

```
unsigned char val[2] = {0x3f, 0x3f};
HostControl_Set(CMD_GPIO_SET, val, 2);
```

5.4.16 frame rate change

[Parameter]

Parameter number: 1

Parameter 0: frameRate, values: {30, 25, 20, 15, 12}

[Usage]

Suppose host wants to set frame rate as 15, then:

```
unsigned char framerate = 15;
HostControl_Set(CMD_FRAMERATE_SET, &framerate , 1);
```

5.4.17 stream ON/OFF

[Parameter]

Parameter number: 1

Parameter 0: stream state, range: [0, 1]

0: stream off, 1: stream on

Related hardware register: 0x8029D000

[Usage]

Suppose host wants to set stream off, then:

```
unsigned char state = 0;
HostControl_Set(CMD_STREAMING_CTRL, &state , 1);
```

5.4.18 test pattern select

[Parameter]

Parameter number: 1

Parameter 0: format

table 5-16 test pattern parameters (sheet 1 of 2)

format	description
0x00	normal video mode
0x01	OV10640 color bar, which is normally used for OV10640 and OV490 DVP connection verification

table 5-16 test pattern parameters (sheet 2 of 2)

format	description
0x02	OV490 front RAW color bar, which passes through OV490 ISP and is mainly used for OV490 output DVP connection verification
0x03	OV490 back YUV color bar, which is measurable and consistent regardless of ISP status

[Usage]

Suppose host wants to set test pattern as 0x02 (OV490 front raw color bar), then:

```
unsigned char val = 0x02;
HostControl_Set(HOSTCMD_TESTPATTERN_SET, &val , 1);
```

5.4.19 output format select

[Parameter]

Parameter number: 1

Parameter 0: format

table 5-17 output format select parameters

format	description
0x00	YUV 12-bit format
0x01	YUV 12-bit format - sensor L/S channel only
0x12	sensor RAW 12-bit format for VS channel
0x13	sensor RAW 12-bit format for S channel
0x14	sensor RAW 12-bit format for L channel
0x15	RAW 12-bit format for VS channel (after OV490 AWB)
0x16	RAW 12-bit format for S channel (after OV490 AWB)
0x17	RAW 12-bit format for L channel (after OV490 AWB)
0x18	linear RAW 12-bit format
0x19	tone mapping row

[Usage]

Suppose host wants to set video format as sensor RAW 12-bit (long exposure channel), then:

```
unsigned char val = 0x14;
HostControl_Set(CMD_FORMAT_SET, &val , 1);
```

5.4.20 BLC set

[Parameter]

Parameter number: 6

Related sensor registers: 0x30C3~ 0x30C8

table 5-18 BLC set parameters

index	description	range/bits
0	BLC target for L exposure[11:8]	Bit[3:0]
1	BLC target for L exposure[7:0]	Bit[7:0]
2	BLC target for S exposure[11:8]	Bit[3:0]
3	BLC target for S exposure[7:0]	Bit[7:0]
4	BLC target for VSexposure[11:8]	Bit[3:0]
5	BLC target for VS exposure[7:0]	Bit[7:0]

[Usage]

Suppose host wants to set BLC as {0x00, 0x80, 0x00, 0x80, 0x00, 0x80}, then:

```
unsigned char blc[6] = {0x00, 0x80, 0x00, 0x80, 0x00, 0x80};
HostControl_Set(CMD_BLC_SET, blc, 6);
```

5.4.21 temperature read

[Input]

Parameter number: 1

table 5-19 temperature read input parameters

index	description	range/bits
0	select temperature read source	0: Read from OV490 1: Read from OV10640

[Output]

- Reading from OV490:
  Parameter number: 2
  Related hardware register: 0x80260015~ 0x80260016

table 5-20 temperature read OV490 output parameters

index	description	range/bits
0	temperature (integer part)[a]	Bit[7:0]
1	temperature (decimal part)	Bit[7:0]

a. if index 0 value is larger than 0xC0, then temperature value is negative

- Reading from OV10640:
  Parameter number: 1

table 5-21 temperature read OV10640 output parameters

index	description	range/bits
0	temperature[a]	Bit[7:0]

a. if index 0 value is larger than 0xC0, then real temperature should be (temperature - 0x100)

[Usage]

If host wants to read out the temperature from OV490, then:

```
unsigned char paraIn = 0;
unsigned char tempOut[2]={0, 0};
HostControl_GetEx(CMD_TEMP_GET, &paraIn, 1, tempOut, 2);
Temp_int = tempOut[0];
Temp_dec=tempOut[1];
```

Then, the resulting temperature should be calculated as follows:

```
if (Temp_int>0xC0)
     T = -1*((Temp_int-0xC0) + Temp_dec/256);
else
     T = Temp_int + Temp_dec/256;
```

If host wants to read out the temperature from the OV10640, then:

```
unsigned char paraIn = 1;
unsigned char Temp_int =0;
HostControl_GetEx(CMD_TEMP_GET, &paraIn, 1, &Temp_int, 1);
```

Then, the resulting temperature should be calculated as follows:

```
if (Temp_int>0xC0)
     T = Temp_int-0x100;
else
     T = Temp_int;
```

5.4.22 YUV statistics windows set

[Parameter]

Parameter number: 9

Related hardware registers: 0x80207C00~0x80207C1F (OV490), 0x4000~0x4007 (OV10640)

table 5-22 YUV statistics windows set parameters

index	description	range/bits
0	window index (total of five windows can be selected)	[0, 4], in which 0~3 windows are for OV490 and the fourth window is for OV10640
1	x_start[15:8]	Bit[7:0]
2	x_start[7:0]	Bit[7:0]
3	y_start[15:8]	Bit[7:0]
4	y_start[7:0]	Bit[7:0]
5	width[15:8]	Bit[7:0]
6	width[7:0]	Bit[7:0]
7	height[15:8]	Bit[7:0]
8	height[7:0]	Bit[7:0]

[Usage]

Suppose host wants to set second window with size and position listed below:

x_start: 0x100, y_start: 0x100, width: 0x140, height: 0xF0

Then:

```
unsigned char win[9] = {
     0x01,
     0x01 0x00,
     0x01, 0x00,
     0x01, 0x40,
     0x00, 0xf0};
HostControl_Set(CMD_STAT_ROI_SET, win, 9);
```

5.4.23 AB context switch

AB context switch has two operation stages:

- configure the context switch parameters
- trigger context switch function

5.4.23.1 context switch config

[Parameter]

Parameter number: 37

table 5-23 AB context switch parameters

index	description	note
0	total bytes	
1	reserved	
2	cmd_number	set 2 for AB switch
3	cmd0_index	set 0 for A
4	cmd0_parameter_number	13 by default
5	cmd0_param[0]	parameter 0
...	...	...
17	cmd0_param[12]	parameter 12
18	cmd1_index	set 1 for B
19	cmd1_parameter_number	13 by default
20	cmd0_param[1]	parameter 0
...	...	...
32	cmd1_param[12]	parameter 12
33	A_delay_frameNumber	
34	B_delay_frameNumber	
35	0x4F[a]	
36	0x56	

a. last two bytes (0x4F and 0x56) are magic numbers, which are used to judge whether the parameters are sent correctly or not. If not, the config will not be executed.

[Usage 1: both A frame and B frame are manual exposure/gain]

```
unsigned char config[37] ={
37, 0,
```

```
2,
0, 13,
{1,0x01,0xd0,0x01,0x20,0x1,0x20,0x00,0x25,0x00,0x45,0x00,0x55}, // manual
                          expo/gain for A
1, 13,
{1,0x00,0x20,0x01,0x50,0x1,0x50,0x00,0x25,0x00,0x45,0x00,0x55}, // manual
                          expo/gain for B
0, 0,                    //reserved
0x4f,0x56                // magic number
};
HostControl_Set(CMD_CONTEXT_SWITCH_CONFIG, config, 37);
```

[Usage 2: A-Auto exposure/gain, B-Manual exposure/gain]

```
unsigned char config[37] ={
37, 0,
2,
0, 13,
{,0x00,0x00,0x00,0x00,0x0,0x00,0x00,0x00,0x00,0x00,0x00,0x00}, // manual
                          expo/gain for A
1, 13,
{1,0x00,0x20,0x01,0x50,0x1,0x50,0x00,0x25,0x00,0x45,0x00,0x55}, // manual
                          expo/gain for B
0, 0,                    //reserved
0x4f,0x56                // magic number
};
HostControl_Set(CMD_CONTEXT_SWITCH_CONFIG, config, 37);
```

5.4.23.2 context switch trigger

[Parameter]

Parameter number: 0

[Usage]

```
HostControl_Set(HOSTCMD_CONTEXT_SWITCH_CTRL, 0, 0);
```

5.4.24 single sub-pixel RAW12 mode

[Parameter]

Parameter number: 1

Parameter 0: mode

table 5-24 single sub-pixel RAW12 mode parameters

format	description
0x00	switch to long exposure mode
0x01	switch to short exposure mode
0x02	reserved
0x03	switch to HDR mode

[Usage]

Suppose host wants to set mode as 0x00(Long Exposure mode), then:

```
unsigned char val = 0x00;
HostControl_Set(CMD_MTF_SET, &val , 1);
```

5.4.25 mirror/flip control

[Parameter]

Parameter number: 2

Related sensor registers: 0x3127, 0x3090, 0x3291

table 5-25 mirror/flip control parameters

index	description
0	enable/disable 0: disable 1: enable
1	control type 0: mirror 1: flip

[Usage 1]

Suppose host wants to enable mirror, then:

```
unsigned char para[2] = {
    0x01, // Enable
    0x00 // Mirror
```

```
};
HostControl_Set(CMD_MIRRFLIP_CTRL, para, 2);
```

[Usage 2]

Suppose host wants to disable flip, then:

```
unsigned char para[2] = {
    0x00, // Disable
    0x01 // Flip
};
HostControl_Set(CMD_MIRRFLIP_CTRL, para, 2);
```

5.4.26 LENC parameter set

[Parameter]

Parameter number: 385

Related hardware registers: 0x80206400~0x8020657F

Related firmware registers: pLensCPArray[][][]

table 5-26 LENC set parameters (sheet 1 of 2)

index	description	range/bits
0x00~0x07	g00~g07 of 8x8 G control point array for L/VS	Bit[7:0]
0x08~0x0F	g10~g17 of 8x8 G control point array for L/VS	Bit[7:0]
0x10~0x17	g20~g27 of 8x8 G control point array for L/VS	Bit[7:0]
...	...	...
0x38~0x3F	g70~g77 of 8x8 G control point array for L/VS	Bit[7:0]
0x40~0x47	b00~b07 of 8x8 B control point array for L/VS	Bit[7:0]
...	...	...
0x78~0x7F	b70~b77 of 8x8 B control point array for L/VS	Bit[7:0]
0x80~0x87	r00~r07 of 8x8 R control point array for L/VS	Bit[7:0]
...	...	...
0xB8~0xBF	r70~r77 of 8x8 R control point array for L/VS	Bit[7:0]
0xC0~0xC7	g00~g07 of 8x8 G control point array for S	Bit[7:0]
...	...	...
0xF8~0xFF	g70~g77 of 8x8 G control point array for S	Bit[7:0]
0x100~0x107	b00~b07 of 8x8 B control point array for S	Bit[7:0]
...	...	...

table 5-26 LENC set parameters (sheet 2 of 2)

index	description	range/bits
0x138~0x13F	b70~b77 of 8x8 B control point array for S	Bit[7:0]
0x140~0x147	r00~r07 of 8x8 R control point array for S	Bit[7:0]
...	...	...
0x178~0x17F	r70~r77 of 8x8 R control point array for S	Bit[7:0]
0x180	mode[a]	Bit[7:0]: Mode 0x0: Auto mode 0x1: Manual mode 0x2: D65 0x3: CWF 0x4: A 0x5: Same for 3 lights 0x6: Restore to default

a. when mode is set as follows:
2, 3, or 4: directly set the parameters to firmware registers pLensCPArray[][][]
5: firmware registers are set to the same group value for 3 kinds of lights (D65, CWF, A)
6: firmware registers pLensCPArray[][][] are restored to default values

[Usage]

Suppose host wants to set manual LENC parameters, then:

```
unsigned char param[385]= {....., 1};
HostControl_Set(CMD_LENC_SET, param , 385);
```

5.4.27 embedded line control

5.4.27.1 general configure

[Parameter]

Parameter number: 16

table 5-27 embedded line control parameters

index	description	range/bits
0	embedded line enable	Bit[7:4]: Reserved Bit[3:2]: OV490 embedded line 00: Disable OV490 embedded line 01: Enable top embedded line 10: Enable bottom embedded line 11: Enable top and bottom embedded line Bit[1:0]: Sensor embedded line 00: Disable sensor embedded line 01: Enable top embedded line 10: Enable bottom embedded line 11: Enable top and bottom embedded line
1	reserved	Bit[7:0]
2	start address for sensor top embedded data range[15:8]	Bit[7:0]
3	start address for sensor top embedded data range[7:0]	Bit[7:0]
4	end address for sensor top embedded data range[15:8]	Bit[7:0]
5	end address for sensor top embedded data range[7:0]	Bit[7:0]
6	start address for sensor bottom embedded data range[15:8]	Bit[7:0]
7	start address for sensor bottom embedded data range[7:0]	Bit[7:0]
8	end address for sensor bottom embedded data range[15:8]	Bit[7:0]
9	end address for sensor bottom embedded data range[7:0]	Bit[7:0]
10	start address for OV490 top embedded data range[23:16]	Bit[7:0]
11	start address for OV490 top embedded data range[15:8]	Bit[7:0]
12	start address for OV490 top embedded data range[7:0]	Bit[7:0]
13	end address for OV490 top embedded data range[23:16]	Bit[7:0]
14	end address for OV490 top embedded data range[15:8]	Bit[7:0]
15	end address for OV490 top embedded data range[7:0]	Bit[7:0]

[Usage]

Suppose host can set both sensor's embedded line and OV490's embedded line as follows:

```
unsigned char para[16] = {
0x0f ,                    //Enable both top and bottom embedded line of sensor and
                          OV490
0x00,                     //reserved
0x30, 0x00, 0x34, 0xfb,   //Sensor top embedded line register dump range
                          [0x3000~0x34fb], total 1276 registers
0x40, 0x00, 0x44, 0xfb,   //Sensor bottom embedded line register dump range
                          [0x4000~0x44fb], total 1276 registers
0x20, 0x60, 0x00, 0x20, 0x64, 0x83  //OV490 top embedded line register dump range
                                    [0x80206000~0x80206483], total 1156 registers
};
HostControl_Set(CMD_EMBL_SET, para , 16);
```

[Notes]

Suppose the image format is YUV422 and image width is 1280, then:

1. register number num _snr in sensor embedded line should satisfy the following conditions:
   - 1280/2 ≤ num _snr ≤ 1280-4
   - num_snr % 4 = 0 (that is: num_snr can be divisible by 4)
2. register number num_490 in OV490 top embedded line should satisfy the following conditions:
   - 4 ≤ num _490 ≤ 1296
   - num_490 % 4 = 0 (that is: num_490 can be divisible by 4)

Note that the first 104 data in the OV490 top embedded line are pre-defined in the OV490 firmware already and cannot be configured by the user. The data structure in this embedded line is designed as follows:

table 5-28 OV490 top embedded line data structure (sheet 1 of 2)

index	description	range/bits
0	frame number[15:8][a]	Bit[7:0]
1	frame number[7:0]	Bit[7:0]
2	temperature[15:8]	Bit[7:0]
3	temperature[7:0]	Bit[7:0] 0x80260015: temperature integer part
4	ABC frame[15:8]	Bit[7:0]
5	ABC frame[7:0]	Bit[7:0]
6	firmware version[15:8][b]	Bit[7:0]
7	firmware version[7:0]	Bit[7:0]

table 5-28 OV490 top embedded line data structure (sheet 2 of 2)

index	description	range/bits
8~31	exposure gain	Bit[7:0] R80207d80 ~ R80207d97
32-103	AWB gain	R80206100 ~ R80206147: R80206100~R80206107: Gain for L exposure channel Red, GreenR, GreenB, Blue component R80206108~R8020610f: Gain for S exposure channel Red, GreenR, GreenB, Blue component R80206110~R80206117: Gain for VS exposure channel Red, GreenR, GreenB, Blue component R80206119~R80206127: Gain offset for L exposure channel Red, GreenR, GreenB, Blue component R80206129~R80206137: Gain offset for S exposure channel Red, GreenR, GreenB, Blue component R80206139~R80206147: Gain offset for VS exposure channel Red, GreenR, GreenB, Blue component For detailed register information, refer to *OV490 Preliminary Specification.*
104~1295	user-defined data	Register dump range defined by user Note that the data length, DL, should satisfy the following condition: DL + 104 ≤ 1296

a. frame number can be read out from registers 0x80800100, 0x80800101

b. firmware version can be read out from registers 0x80800102, 0x80800103

3. OV490 bottom embedded line is also pre-defined in OV490 firmware already and cannot be configured by the user. The data structure in this embedded line is designed as follows:

table 5-29 OV490 bottom embedded line data structure (sheet 1 of 2)

index	description	range/bits
0~7	AEC/AGC statistic	R80207060 ~ R80207067 Note that only four registers (R80207062~R80207d65) are for AEC/AGC statistics R80207062: Mean Y [0], bit[1:0] R80207063: Mean Y [0], bit[7:0] R80207064: Mean Y [1], bit[1:0] R80207065: Mean Y [1], bit[7:0]

table 5-29 OV490 bottom embedded line data structure (sheet 2 of 2)

index	description	range/bits
8~223	YUV ROI statistics	R80207c28 ~ R80207cff R80207c28 ~ R80207c29: ROI 0 Y average R80207c2a ~ R80207c2b: ROI 0 U average R80207c2c ~ R80207c2d: ROI 0 V average R80207c2e ~ R80207c5d: ROI 0 Y histogram bin R80207c5e ~ R80207c5f: ROI 1 Y average R80207c60 ~ R80207c61: ROI 1 U average R80207c62 ~ R80207c63: ROI 1 V average R80207c64 ~ R80207c93: ROI 1 Y histogram bin R80207c94 ~ R80207c95: ROI 2 Y average R80207c96 ~ R80207c97: ROI 2 U average R80207c98 ~ R80207c99: ROI 2 V average R80207c9a ~ R80207cc9: ROI 2 Y histogram bin R80207cca ~ R80207ccb: ROI 3 Y average R80207ccc ~ R80207ccd: ROI 3 U average R80207cce ~ R80207ccf: ROI 3 V average R80207cd0 ~ R80207cff: ROI 3 Y histogram bin
224	frame number[15:8]	Bit[7:0] same as frame number of OV490 top embedded line
225	frame number[7:0]	Bit[7:0]
226	reserved	Bit[7:0]

[Example]

Now, take a 1280x800 image data (including four sensor embedded lines and two OV490 embedded lines) for example:

- Sensor top first embedded line data:

```
0000000: 3103 0107 4803 0107 0001 40a6 f060 0080  1...H.....@..`..
0000010: a27f 0001 0000 001f 0012 0100 0001 0000  ................
0000020: 0000 0a00 0000 0204 4040 4040 0103 0201  ........@@@@....
0000030: 0030 fb34 0040 fb44 0000 de01 dede dede  .0.4.@.D........
0000040: 0071 0000 00ff b300 d27b 8080 dfc1 10c6  .q.......{......
0000050: 02fa 7100 0800 8c03 e4cc 0006 1c30 1816  ..q..........0..
0000060: f0f0 0000 3c3c 0506 0c87 0100 33b9 0023  ....<<......3..#
0000070: 00c2 1e1e 0400 8500 0305 ad03 0005 1e03  ................
0000080: ac07 4003 0000 0f00 4000 0000 9201 00ff  ..@.....@.......
0000090: 0c00 0000 00a5 3a03 3b01 2660 0100 b508  ......:.;.&`....
```

```
0000250: 0400 1006 1020 0806 0602 1009 0410 080c  ..... ..........
0000260: 0400 1006 1020 0806 0804 1810 3020 4040  ..... ......0 @@
0000270: 0400 043f 1008 2018 4030 0040 3f04 0804  ...?.. .@0.@?...
0000280: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000290: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

```
00009b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00009c0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00009d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00009e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00009f0: 0000 0000 0000 0000 0000 0000 f30f ac78  ...............x
```

Above is part of the embedded data in the sensor top embedded line. Since sensor top embedded line register dump range is [0x3000~0x34FB], the first line data is arranged as below:

register	R3001, R3000	R3003, R3002	R3005, R3004	R3007, R3006	R3009, R3008	R300B, R300A	R300D, R300C	R300F, R300E
emb data (0000000)	31 03	01 07	48 03	01 07	00 01	40 A6	F0 60	00 80
register	R3011, R3010	R3013, R3012	R3015, R3014	R3017, R3016	R3019, R3018	R301B, R301A	R301D, R301C	R301F, R301E
emb data (0000010)	A2 7F	00 01	00 00	00 1F	00 12	01 00	00 01	00 00
register	R3021, R3020	R3023, R3022	R3025, R3024	R3027, R3026	R3029, R3028	R302B, R302A	R302D, R302C	R302F, R302E
emb data (000020)	00 00	0A 00	00 00	02 04	40 40	40 40	01 03	02 01
...	...	...	...	...	...	...	...	...
register							crc_high	crc_low
emb data (00009F0)	00 00	00 00	00 00	00 00	00 00	00 00	f30f	ac78

Note that only 1280/2=640 registers' values are output through this embedded line due to the fact that the OV490 will remove the tags when it receives the OV10640 embedded line and then send them out.

Refer to **section 4** for detailed information.

- Sensor top second embedded line data:

```
0000a00: 1810 3020 4040 0400 003f 0000 0200 02f0  ..0 @@...?......
0000a10: 0008 0005 2d03 0000 0000 0000 0000 0000  ....-...........
0000a20: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000a30: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000a40: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000a50: 0000 0000 0000 2020 2020 2020 2020 2020  ......
0000a60: 2020 2020 2020 2020 2020 2020 2020 2020
0000a70: 2020 2020 2020 2020 2020 2020 2020 2020
```

```
00013d0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00013e0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00013f0: 0000 0000 0000 0000 0000 0000 c56e 30ac  ..............n0.
```

Since the first 640 registers' values are output in the first line, the second top line will start with the 641st register (i.e., R3280) as shown in the following table:

register	R3281, R3280	R3283, R3282	R3285, R3284	R3287, R3286	R3289, R3288	R328B, R328A	R328D, R328C	R328F, R328E
emb data (0000A00)	18 10	30 20	40 40	04 00	00 3F	00 00	02 00	02 F0
register	R3291, R3290	R3293, R3292	R3295, R3294	R3297, R3296	R3299, R3298	R329B, R329A	R329D, R329C	R329F, R329E
emb data (0000A10)	00 08	00 05	2D 03	00 00	00 00	00 00	00 00	00 00
...	...	...	...	...	...	...	...	...
register							crc_high	crc_low
emb data (00013F0)	00 00	00 00	00 00	00 00	00 00	00 00	C56E	30AC

- OV490 top embedded line data:

```
0001400: 0ab0 2649 000a 778d 8000 5000 002a 0001  ..&I..w...P..*..
0001410: 0001 0001 3602 9402 9402 4000 4000 4000  ....6.....@.@.@.
0001420: 8e03 0001 0001 2d02 4603 0001 0001 b001  ......-.F.......
0001430: 8e03 0001 0001 2d02 0000 80a3 0000 0000  ......-.........
0001440: 0000 0000 0000 404b 0000 8091 0000 0000  ......@K........
0001450: 0000 0000 0000 002c 0000 80a3 0000 0000  .......,........
0001460: 0000 0000 0000 404b 0061 0000 0000 0000  ......@K.a......
0001470: 0000 0000 0101 0000 fff7 80ff 6600 0001  ............f...
0001480: 0000 0000 0005 3804 0018 0000 3800 0045  ......8.....8..E
0001490: 0001 0000 0000 0000 fff0 80ff 6600 0001  ............f...
00014a0: 0000 0000 0005 3804 0030 0000 3800 0045  ......8..0..8..E
```

```
0001dc0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0001dd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0001de0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0001df0: 0000 0000 0000 0000 0000 0000 86b6 4c6c  ..............Ll
```

register	frame number	temperature	ABC flag	firmware version	R80207d81, R80207d80	R80207d83, R80207d82	R80207d85, R80207d84	R80207d87, R80207d86
emb data (0001400)	0AB0	2649	000A	778D	80 00	50 00	00 2A	00 01
...	...	...	...	...	...	...	...	...
register	R80206141, R80206140	R80206143, R80206142	R80206145, R80206144	R80206147, R80206146	R80206001, R80206000	R80206003, R80206002	R80206005, R80206004	R80206007, R80206006
emb data (0001460)	00 00	00 00	00 00	40 4B	00 61	00 00	00 00	00 00

register	frame number	temperature	ABC flag	firmware version	R80207d81, R80207d80	R80207d83, R80207d82	R80207d85, R80207d84	R80207d87, R80207d86
...	...	...	...	...	...	...	...	...
register							crc_high	crc_low
emb data (0001DF0)	00 00	00 00	00 00	00 00	00 00	00 00	86B6	4C6C

- Sensor bottom first embedded line data:
- Sensor bottom second embedded line data:
- Sensor bottom embedded line data arrangement is same as sensor top embedded line data.
- OV490 bottom embedded line data:

```
01f3600: 1410 f300 8c00 0000 8501 f801 0602 0000  ................
01f3610: 0000 0000 0000 0000 2d00 0500 00c3 6965  ........-.....ie
01f3620: 8400 0000 a210 0000 0005 0000 0000 0000  ................
01f3630: 0000 0000 0000 0000 0000 0000 0000 8501  ................
01f3640: f801 0602 0000 0000 0000 0000 0000 2d00  ..............-.
01f3650: 0500 00c3 6965 8400 0000 a210 0000 0005  ....ie..........
01f3660: 0000 0000 0000 0000 0000 0000 0000 0000  ................
01f3670: 0000 0000 8501 f801 0602 0000 0000 0000  ................
01f3680: 0000 0000 2d00 0500 00c3 6965 8400 0000  ....-.....ie....
```

```
01f3fd0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
01f3fe0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
01f3ff0: 0000 0000 0000 0000 0000 0000 312b ac37  ............1+.7
```

As mentioned before, the bottom embedded line data is pre-defined in the OV490 firmware already and cannot be configured by the user. Data arrangement is shown as follows:

table 5-30 OV490 bottom embedded line data arrangement

index	description	range/bits
0~7	AEC/AGC statistic	R80207060 ~ R80207067 10 14 00 F3 00 8C 00 00
8~223	YUV ROI statistics	R80207c28 ~ R80207cff 01 85 01 F8 02 06 00 00
224	reserved	0x00

5.4.27.2 special configure

This section is dedicated for the OV490 top embedded line user defined data range. In general configure, only one group of registers (start register address -> end register address) can be set up. This special configure supports up to four groups of register range configure.

[Parameter]

Parameter number: 14

table 5-31 special configure parameters

index	description	range/bits
0	group number	Bit[7:3]: Reserved Bit[2:0]: Group number 1~4
1	reserved	0
2	start address for OV490 top embedded data range 1[23:16]	Bit[7:0]
3	start address for OV490 top embedded data range 1[15:8]	Bit[7:0]
4	start address for OV490 top embedded data range 1[7:0]	Bit[7:0]
5	start address for OV490 top embedded data range 2[23:16]	Bit[7:0]
6	start address for OV490 top embedded data range 2[15:8]	Bit[7:0]
7	start address for OV490 top embedded data range 2[7:0]	Bit[7:0]
8	start address for OV490 top embedded data range 3[23:16]	Bit[7:0]
9	start address for OV490 top embedded data range 3[15:8]	Bit[7:0]
10	start address for OV490 top embedded data range 3[7:0]	Bit[7:0]
11	start address for OV490 top embedded data range 4[23:16]	Bit[7:0]
12	start address for OV490 top embedded data range 4[15:8]	Bit[7:0]
13	start address for OV490 top embedded data range 4[7:0]	Bit[7:0]

Note:

- supports 1,2,3,4 groups configure
- 1152 bytes range are divided for groups configure, that is to say:
  - for 1 group: register range is fixed as 1152 bytes
  - for 2 groups: each group register range is fixed as 1152/2 = 576 bytes
  - for 3 groups: each group register range is fixed as 1152/3 = 384 bytes
  - for 4 groups: each group register range is fixed as 1152/4 = 288 bytes
- so only start address for each group is needed in API parameter list

[Usage]

- Configure 1 group:
  If host wants to output only one group which start address is 0x80100000, then:

```
unsigned char para[] = {
     0x01, 0x00,
     0x10,0x00,0x00
};
HostControl_Set(CMD_TOPEMB_SET, &para, sizeof(para)/sizeof(unsigned char));
```

- Configure 2 groups:
  If host wants to output two groups which start addresses are 0x80100000, 0x80206000 separately, then:

```
unsigned char para[] = {
     0x02, 0x00,
     0x10,0x00,0x00,
     0x20,0x60,0x00
};
HostControl_Set(CMD_TOPEMB_SET, &para, sizeof(para)/sizeof(unsigned char));
```

- Configure 3 groups:
  If host wants to output three groups which start addresses are 0x80100000, 0x80206000, 0x80205000 separately, then:

```
unsigned char para[] = {
     0x03, 0x00,
     0x10,0x00,0x00,
     0x20,0x60,0x00,
     0x20,0x50,0x00
};
HostControl_Set(CMD_TOPEMB_SET, &para, sizeof(para)/sizeof(unsigned char));
```

- Configure 4 groups:
  If host wants to output four groups which start addresses are 0x80100000, 0x80204000, 0x80205000, 0x80206000 separately, then:

```
unsigned char para[] = {
     0x04, 0x00,
     0x10,0x00,0x00,
     0x20,0x40,0x00,
     0x20,0x50,0x00,
     0x20,0x60,0x00
};
HostControl_Set(CMD_TOPEMB_SET, &para, sizeof(para)/sizeof(unsigned char));
```

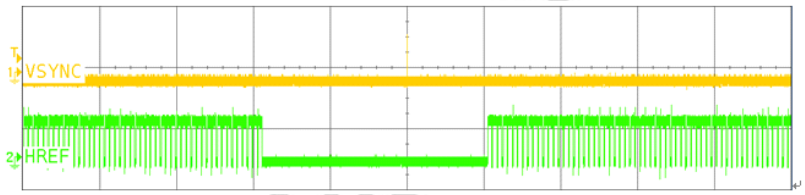
5.4.27.3 waveform of embedded line

Suppose the following embedded lines are configured in the OV490 firmware:

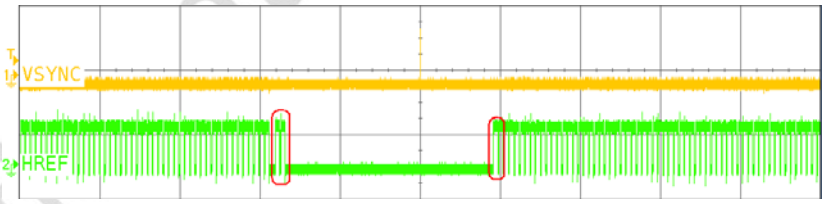
- 2x sensor embedded lines (top)
- 1x OV490 embedded lines (top)
- 2x sensor embedded lines (bottom)
- 1x OV490 embedded lines (bottom)

Then the following figures display the embedded line waveform in different configurations.

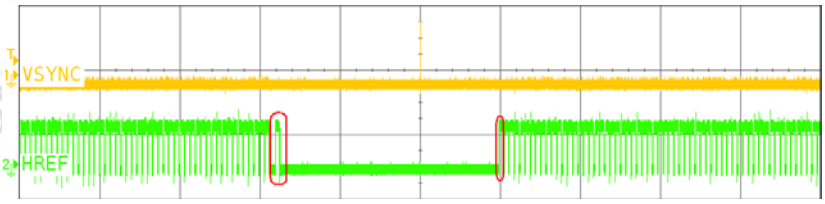
a. both sensor's and OV490's embedded line disabled



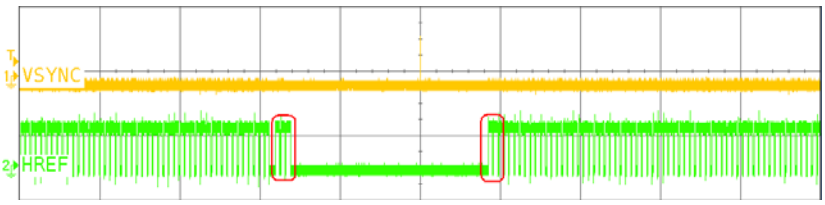
b. only the sensor embedded line is enabled (4x, 2x top, 2x bottom)



c. only the OV490 embedded line is enabled (2x, 1x top, 1x bottom)

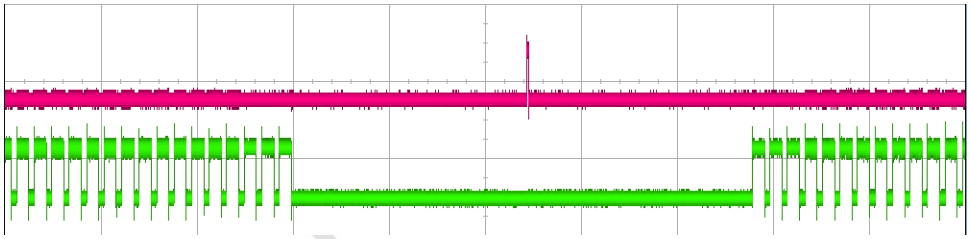


d. both sensors and the OV490 embedded line enabled (6x, 3x top, 3x bottom)

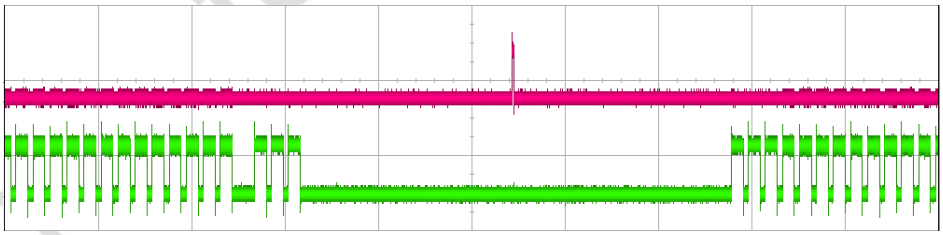


5.4.27.4 bottom embedded line delay from image data can be adjusted by register 0x8029E035

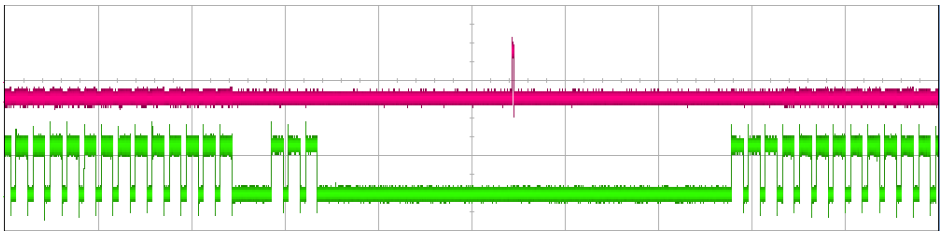
a. 0 line delay (0x8029E035 = 0)



b. 1 line delay (0x8029E035 = 1)



c. 2 lines delay (0x8029E035 = 2)



5.4.28 group write/launch

The OV490 supports up to four group writes and launch. Each group can have up to 125 registers. The operations are divided into two stages: one is group write, the other is group launch. The group write operation let the host write up to four groups of registers to store in OV490 SRAM, while the group launch operation let the host trigger the launch.

5.4.28.1 group write

[Parameter]

*Parameter number: 6+(n-1)*4*

table 5-32 group write parameters

index	description	range/bits
0	group ID and mode	Bit[7:3]: Reserved Bit[2:0]: Group number 1~4
1	group register number	1~125
2	first register base low address	Bit[7:0]: Valid for OV490 register 0: For sensor register
3	first register 8-bit high address	Bit[7:0]
4	first register 8-bit low address	Bit[7:0]
5	first register 8-bit value	Bit[7:0]
...	...	...
2+(n-1)*4	nth register base low address	Bit[7:0]: Valid for OV490 register 0: For sensor register
3+(n-1)*4	nth register 8-bit high address	Bit[7:0]
4+(n-1)*4	nth register 8-bit low address	Bit[7:0]
5+(n-1)*4	nth register 8-bit value	Bit[7:0]

[Usage]

- Group write OV490 registers to group 1:

```
unsigned long OV490_setting[] = {
     0x80194f00, 0x51,
     0x80195f01, 0x55,
     0x80196f02, 0x53,
     0x80197f03, 0x57};
```

Then:

```
unsigned char param[]= {
     0x01,    // group id:1, mode: OV490 register
```

```
        0x04,    // 4 sets of registers
        0x19, 0x4f, 0x00, 0x51,
        0x19, 0x5f, 0x01, 0x55,
        0x19, 0x6f, 0x02, 0x53,
        0x19, 0x7f, 0x03, 0x57};
    HostControl_Set(CMD_GROUPWRITE_SET, param , sizeof(param)/sizeof(unsigned char));
```

- Group write sensor registers to group 2:

```
    unsigned long sensor_setting[] = {
        0x3000, 0x03,
        0x3001, 0x62,
        0x3002, 0x07,
        0x3003, 0x01,
        0x3004, 0x03};
```

Then:

```
    unsigned char param[]= {
        0x82,    // group id:2, mode: sensor register
        0x05,    // 5 sets of registers
        0x00, 0x30, 0x00, 0x03,
        0x00, 0x30, 0x01, 0x62,
        0x00, 0x30, 0x02, 0x07,
        0x00, 0x30, 0x03, 0x01,
        0x00, 0x30, 0x04, 0x03};
    HostControl_Set(HOSTCMD_GROUPWRITE_SET, param , sizeof(param)/sizeof(unsigned
                                char));
```

5.4.28.2 group launch

[Parameter]

Parameter number: 1

table 5-33 group launch parameters

index	description	range/bits
0	launch group index	0~2

[Usage]

If host has already written one or more groups registers to SRAM and the host wants to launch group 1, then:

```
    unsigned char id = 1;
    HostControl_Set(CMD_GROUPWRITE_LAUNCH, &id, 1);
```

5.4.29 extra VTS control

[Parameter]

Parameter number: 1

Related sensor register: 0x30A4

table 5-34 extra VTS control parameters

index	description	range/bits
0	sensor extra VTS	Bit[7:0]

[Usage]

If host wants to set sensor extra VTS as 0x40, then:

```
unsigned char vts = 0x40;
HostControl_Set(CMD_EXTRA_VTS_SET, &vts, 1);
```

5.4.30 post AWB gain set

[Parameter]

Parameter number: 7

Related firmware registers: nBGainShift, nGGainShift, nRGainShift, bManualAWBGainShiftEnable

table 5-35 post AWB gain set parameters

index	description	range/bits
0	post AWB gain set enable	Bit[0]: Post AWB gain set enable 0: Disable 1: Enable
1	B gain shift[15:8]	Bit[7:0]
2	B gain shift[7:0]	Bit[7:0]
3	G gain shift[15:8]	Bit[7:0]
4	G gain shift[7:0]	Bit[7:0]
5	R gain shift[15:8]	Bit[7:0]
6	R gain shift[7:0]	Bit[7:0]

[Usage]

If host wants to set post awb gain, then:

```
unsigned char gain[7] = { 1,
     0x00, 0x80,  //B gain shift
     0x00, 0x80,  //G gain shift
     0x00, 0x90  //R gain shift
};
HostControl_Set(CMD_POSTAWBGAIN_SET, gain, 7);
```

5.4.31 access sensor/I2C_dev register

[Parameter]

Parameter number: 5

table 5-36 access sensor/I2C_dev parameters

index	description	range/bits
0	I2C ID + access mode	Bit[7:1]: High 7 bits of slave address 0x0: Sensor Others: Other I2C device Bit[0]: 0: Write 1: Read
1	register high 8-bit address	Bit[7:0]
2	register low 8-bit address	Bit[7:0]
3	register value	Bit[7:0]: Reserved for read operation Valid for write operation
4	register address width	Bit[7:1]: Reserved Bit[0]: 0: 8-bit address 1: 16-bit address Note that this parameter is valid only for other I2C devices.

[Usage]

- Read operation:
  If host wants to read sensor register 0x300A via OV490, then:

```
unsigned char paraIn[5] = {
     0x01,  // read
     0x30,  //high 8 bit address
```

```
    0x0a,  //low 8 bit address
    0x00,  // reserved
    0x00  // reserved
};
unsigned char value=0;
HostControl_GetEx(CMD_SNR_REG_ACCESS, paraIn, sizeof(paraIn), &value,1);
```

If host wants to read EEPROM 16-bit address register 0x0001 via OV490, then:

```
unsigned char paraIn[5] = {
    0xA5,  // [7:1], EEPROM slave address: 0xA4, [0], 1: read
    0x00,  //high 8 bit address
    0x01,  //low 8 bit address
    0x00,  //reserved
    0x01  //16bit address access
};
unsigned char value=0;
HostControl_GetEx(CMD_SNR_REG_ACCESS, paraIn, sizeof(paraIn), &value,1);
```

- Write operation:

  If host wants to write sensor register 0x308C to 0x30 via OV490, then:

```
unsigned char para[5] = {
    0x00,  // write
    0x30,  //high 8 bit address
    0x8c,  //low 8 bit address
    0x30,  // value to write
    0x00                  //reserved
};
HostControl_Set(CMD_SNR_REG_ACCESS, para, 5);
```

  If host wants to write EEPROM 16-bit address register 0x0001 to 0x30 via OV490, then:

```
unsigned char para[5] = {
    0xA4,  // [7:1], slave address: 0xA4, [0], 0: write
    0x00,  //high 8 bit address
    0x01,  //low8 bit address
    0x30,  // value to write
    0x01  //16bit address
};
HostControl_Set(CMD_SNR_REG_ACCESS, para, 5);
```

5.4.32 access firmware register

[Parameter]

Parameter number: n+5

table 5-37 access firmware parameters

index	description	range/bits
0	access mode	Bit[0]: Access mode 0: Read 1: Write
1	register offset high 8-bit address	Bit[7:0]
2	register offset low 8-bit address	Bit[7:0]
3	high 8 bits of number of registers to be accessed	Bit[7:0]
4	low 8 bits of number of registers to be accessed	Bit[7:0] suppose n
5[a]	register values 1	Bit[7:0]
6	register values 2	Bit[7:0]
...	...	...
5+n-1	register values n	Bit[7:0]

a. parameters (index 5~5+n-1) are only needed for write operation

[Usage]

- Read operation:
  Suppose host wants to read post-AWB gain registers (nBGainShift, nGGainShift, nRGainShift), hence:

```
unsigned char para[] = {
     0x00,        // read
     0x00, 0xbc,  //offset
     0x00, 0x06,  //register number
     };
unsigned char value[6] = {0, 0, 0, 0, 0, 0};
HostControl_GetEx(CMD_FWREG_ACCESS, paraIn, sizeof(paraIn), value,
sizeof(value));
nBGainShift = (value[0]<<8)+value[1];
nGGainShift = (value[2]<<8)+value[3];
nRGainShift = (value[4]<<8)+value[5];
```

- Write operation:
  Suppose host wants to write post-AWB gain registers (nBGainShift, nGGainShift, nRGainShift), which firmware addresses are continuous and the first register(nBGainShift)'s offset is 0xBC, hence:

```
unsigned char para[] = {
     0x01,         // write
     0x00, 0xbc,  //offset
     0x00, 0x07,  //register number
     0x00, 0x90, 0x00, 0x80, 0x00, 0x80, 0x01 //value to write
     };
HostControl_Set(CMD_FWREG_ACCESS, &para, sizeof(para)/sizeof(unsigned char));
```

5.4.33 frame sync (+/- VTS)

If user wants to slow a camera, some lines will be added. If user wants to speed up a camera, it will reduce a few lines.

[Parameter]

Parameter number: 1 or 2

table 5-38 frame sync (+/- VTS) parameters

index	description	range/bits
0	access mode[a]	Bit[7]: Adjustment flag 0: Positive adjust[b] 1: Negative adjust[c] Bit[6]: Adjustment mode 0: One-shot adjust[d] 1: Continuous adjust[e] Bit[7]: Line number flag 0: Adjusted line number ≤ 31[f] 1: Adjusted line number > 31 Bit[4:0]: Adjustment line number (or high 4 bits when bit[5] = 1)
1	line number	Bit[7:0]: Adjusted line number low 8 bits when bit[5] = 1

a. based on sensor timing design, if sensor receives this adjust command in frame 1, adjustment will take effect in frame 3

b. positive means lines are added

c. negative means lines are reduced

d. one-shot adjustment means only adjust in one frame and it will be changed back in the next frame

e. continuous adjustment means adjustment will be done every frame

f. if adjusted line number is equal or smaller than 31 lines, only one parameter is needed; otherwise, two parameters are needed

[Usage 1]

If host wants to perform a one-shot positive adjustment, then:

a. only one parameter needed:

```
unsigned char para = 0x12; //+18 lines, only one parameter needed
HostControl_Set(CMD_FRAMESYNC, &para, 1);
```

b. two parameters needed:

```
unsigned char para[2] = {0x20, 0x20}; //+32 lines, two parameters needed
HostControl_Set(CMD_FRAMESYNC, para, 2);
```

[Usage 2]

If host wants to perform a continuous negative adjustment, then:

a. only one parameter needed:

```
unsigned char para = 0xD2;//-18 lines, only one parameter needed
HostControl_Set(CMD_FRAMESYNC, &para, 1);
```

b. two parameters needed:

```
unsigned char para = {0xE0, 0x20};//-32 lines, two parameters needed
HostControl_Set(CMD_FRAMESYNC, para, 2);
```

5.4.34 cropping

[Parameter]

Parameter number: 8

table 5-39 cropping parameters

index	description	range/bits
0	cropped width[15:8]	Bit[7:0]
1	cropped width[7:0]	Bit[7:0]
2	cropped height[15:8]	Bit[7:0]
3	cropped height[7:0]	Bit[7:0]
4	crop start x position[15:8]	Bit[7:0]
5	crop start x position[7:0]	Bit[7:0]
6	crop start y position[15:8]	Bit[7:0]
7	crop start y position[7:0]	Bit[7:0]

[Usage]

If host wants to output an 800x480 image cropped start from (128, 128) position, then:

```
unsigned char para[] = {
     0x03, 0x20,  //800
     0x01, 0xE0,  //480
     0x00, 0x80,  //x start
     0x00, 0x80  //y start
     };
HostControl_Set(CMD_CROP_SET, para, 8);
```

5.4.35 fade in/out switch

[Parameter]

Parameter number: 0

[Usage]

If host wants to switch fade in and fade out, then:

```
HostControl_Set(CMD_FADE_CTRL, NULL, 0);
```

5.4.36 50Hz/60Hz banding control

[Parameter]

Parameter number: 1

table 5-40 50Hz/60Hz banding control parameters

index	description	range/bits
0	banding filter selection	Bit[7:1]: Reserved Bit[0]: Banding filter select 0: 50Hz 1: 60Hz

[Usage]

If host wants to enable 60Hz banding filter, then:

```
unsigned char para = 1;
HostControl_Set(CMD_BANDING_SET, &para, 1);
```

5.5 host multi-commands

5.5.1 concept

Host multi-commands means the host sends one or more commands (N) with CMD parameters to the OV490 and the OV490 receives and executes those commands at one time.

table 5-41 lists the multi-commands format.

table 5-41 multi-commands format

index	parameters
0	total bytes (n+3)
1	reserved (0)
2	cmd_number(N)
3	cmd0_ID
4	cmd0_parameter_number
5	cmd0_param[0]
6	cmd0_param[1]
m	...
m+1	cmd1_ID
m+2	cmd1_parameter_number
m+3	cmd1_param[0]
m+3	cmd1_param[1]
n	...
n+1	0x4F (magic number)[a]
n+2	0x56 (magic number)[a]

a. last two bytes (0x4F, 0x56) are magic numbers, which are used by the OV490 to judge whether the multi-cmd parameters are sent correctly; otherwise, the multi-cmd will not be executed.

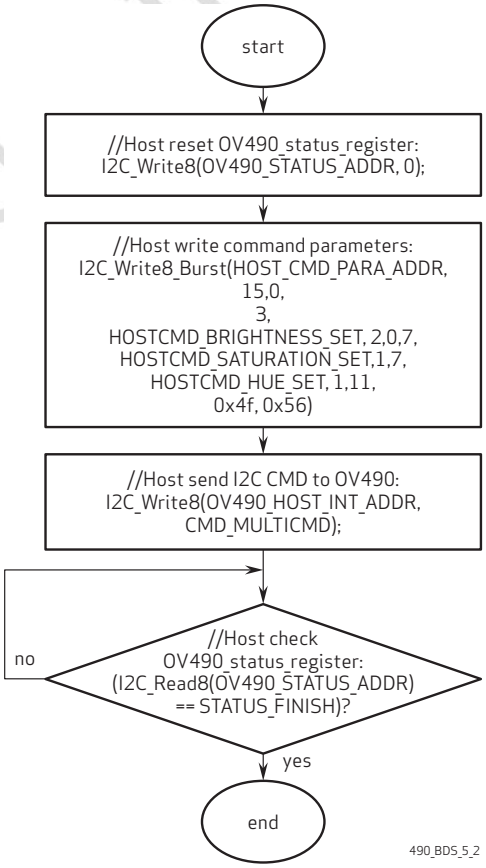
5.5.2 examples

Suppose there are three different commands (brightness, saturation and hue) which need to be sent at one time, then:

```
unsigned char multicmd[15] ={
     15, 0,
     3,
     HOSTCMD_BRIGHTNESS_SET, 2, 0, 7, // For brightness adjust command
     HOSTCMD_SATURATION_SET, 1, 7,    // For saturation adjust command
     HOSTCMD_HUE_SET, 1, 11,          // For hue adjust command
     0x4f,0x56                        // magic number
};
HostControl_Set(HOSTCMD_MULT_CMD, multicmd, 15);
```

5.5.3 working flow

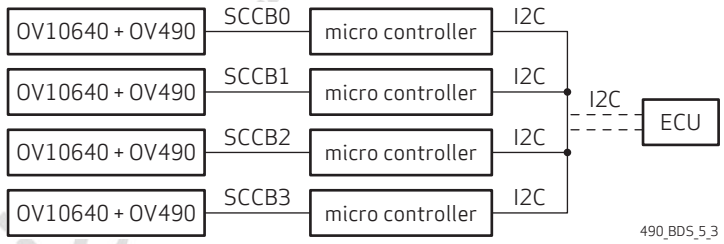
figure 5-2 working flow



5.6 multi-camera system

The OV10640 features frame sync input to synchronize the video streaming timing between multiple sensors in a multi OV10640 and OV490 camera system. Typically, there is a local micro-controller in each camera in which the ECU can control the camera via micro controller. **figure 5-3** shows the block diagram of a typical multi-camera system using the OV10640 and OV490.

figure 5-3 multi-camera system block diagram

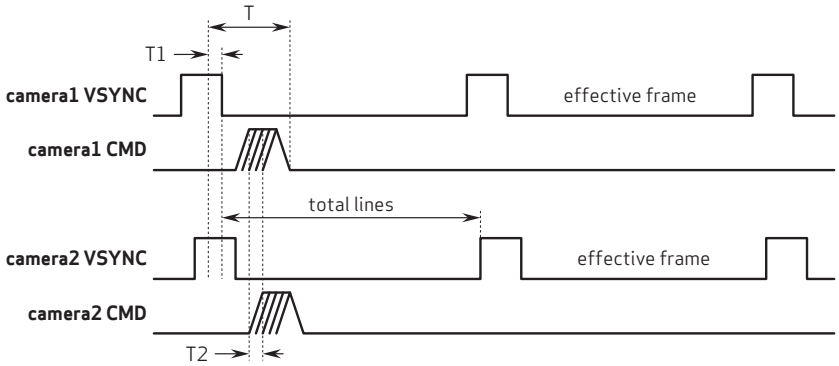


5.6.1 apply command to multi-camera at one time

In a multi-camera system, the ECU needs to write a command to all cameras and requires all cameras effective at the same frame. All commands, including single commands and multi-commands, introduced above are supported in this feature.

The OV490 is a ISP bridge processor which has embedded firmware running inside. To avoid firmware latency and make sure all cameras are effective with same command, specific timing is required. Refer to the proposed timing chart below. The timing requirement is that OV490 should receive the command before half-frame time. The OV490 will launch the command sets and apply them to the ISP and sensor within the left half-frame time.

figure 5-4 multi-camera system timing diagram



note 1 T1 is the latency between cameras. It should be controlled by FSIN pin to sync among cameras.

note 2 T2 is the command latency between micro-controllers.

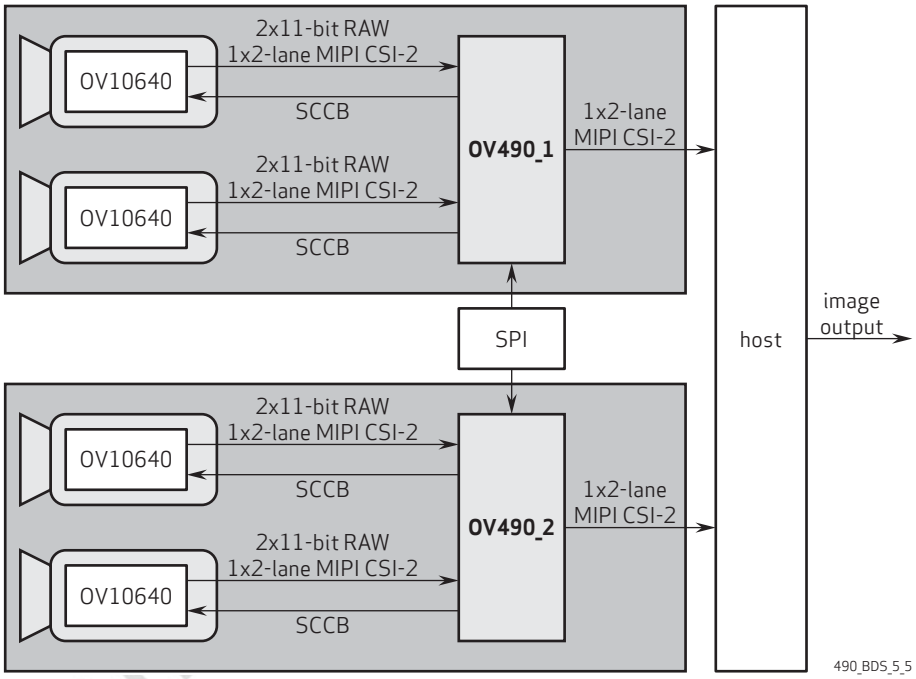
note 3 T1+T2 < T < total lines/2

490_BDS_5_4

5.6.2 host synchronize mechanism

5.6.2.1 system diagram

figure 5-5 host synchronize system block diagram

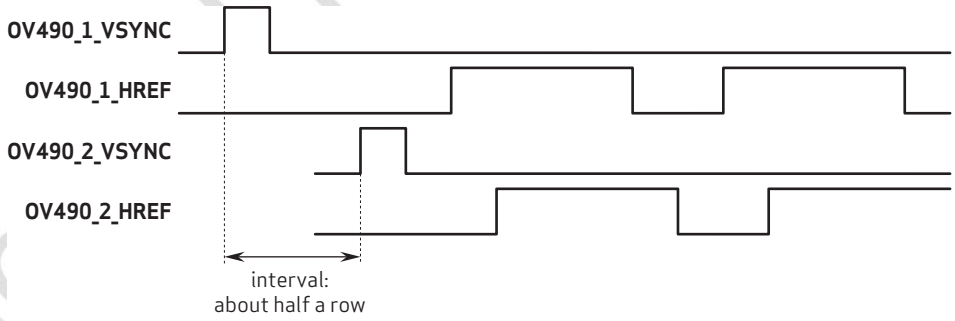


5.6.2.2 synchronize mechanism

For a multi-OV490 system, the host needs to perform the following sync steps:

1. Host resets OV490_1 and OV490_2 and waits for OV490s to output stable image stream.
2. Host needs to measure the interval between SOF1 (OV490_1) and SOF2 (OV490_2).
3. Convert the interval to VTS_SKEW (based on row) and EXTRA_SKEW (based on cycle).
4. Coarse tune: Set VTS_SKEW to OV490_1 through host command (CMD_SYNC_VTS).
5. Fine tune: If EXTRA_SKEW ≥ HTS/2, then set (EXTRA_SKEW-HTS/2) to OV490_1 through host command (CMD_SYNC_EXTRA); else, if (EXTRASKEW < HTS/2), then set (HTS/2-EXTRA_SKEW) to OV490_2 through host command (CMD_SYNC_EXTRA).
6. After three frames, OV490_1 should be about a half row earlier than OV490_2 as below:

figure 5-6 host synchronize system timing diagram



Note:

1. Due to different reference clocks, the interval between SOF1 and SOF2 may be changed every frame, so host need to perform the fine tune operation (above 5th step) every frame;
2. OV490_1 and OV490_2 will sync the related sensors by using dual channel sync mechanism (which is implemented in OV490 firmware). In the meantime, they also receive the host command to adjust their own sensors further. Then, the four sensors can be synchronized up.

5.6.2.3 related host commands

coarse tune

[Command]

#define CMD_SYNC_VTS (0x13)

[Parameter]

Parameter number: 2

table 5-42 VTS coarse tuning parameters

index	description	range/bits
0	VTS_skew[15:8]	Bit[7:0]
1	VTS_skew[7:0]	Bit[7:0]

fine tune

[Command]

#define CMD_SYNC_EXTRASKEW (0x12)

[Parameter]

Parameter number: 2

table 5-43 VTS fine tuning parameters

index	description	range/bits
0	extra_skew[15:8]	Bit[7:0]
1	extra_skew[7:0]	Bit[7:0]

5.6.2.4 example

Known:

Sensor VTS , HTS;

T_frame: time cost in 1 frame, which is measured by host;

Interval: time difference between SOF1 and SOF2 ,which also is measured by host;

Then:

```
 //how long time in 1 line;
line_cnt = T_frame /VTS;
// calculate vts_skew for coarse tune
vts_skew = Interval*128/(T_frame*128/VTS);
// calculate extra_skew for fine tune
mod = ((Interval *128)%(( T_frame *128)/VTS))/128;
if (mod >= line_cnt/2){
     extra_skew = mod*HTS/line_cnt - HTS/2;
}else{    // (mode < line_cnt/2)
     extra_skew = HTS/2- mod*HTS/line_cnt ;
}
// coarse tune:
unsigned char para_vts[2];
para_vts [0] = (vts_skew>>8)&0xff;
para_vts [1] = vts_skew&0xff;
HostControl_Set(CMD_SYNC_VTS, para_vts, 2);
// fine tune:
unsigned char para_extra[2];
para_extra [0] = (extra_skew>>8)&0xff;
para_extra [1] = extra_skew&0xff;
HostControl_Set(CMD_SYNC_ EXTRASKEW, para_extra, 2);
```

appendix A firmware register table

The following table list the firmware register address offsets:

- Currently the base address is 0x80180120. Besides using the host command in 5.4.32 section, users can directly access the hardware address (base + offset). However, the base address may be changed to a different value, OmniVision recommends to use the host command to access the firmware register.
- Firmware registers default values may be changed based on different firmware revisions.

A.1 ISP general control

table A-1 ISP general control registers

address	register name	default value	R/W	description
0x00	nTARGETNUM			
0x02	nVTS_H			
0x03	nVTS_L			
0x04	nEXPOSUREMODE			

A.2 auto white balance (AWB)

table A-2 AWB registers (sheet 1 of 5)

address	register name	default value	R/W	description
0x0C	MAXAWBGAIN_0_H	0x07	RW	Max AWB Gain (B Gain)
0x0D	MAXAWBGAIN_0_L	0xFF	RW	Max AWB Gain (B Gain)
0x0E	MAXAWBGAIN_1_H	0x07	RW	Max AWB Gain (G Gain)
0x0F	MAXAWBGAIN_1_L	0xFF	RW	Max AWB Gain (G Gain)
0x10	MAXAWBGAIN_2_H	0x07	RW	Max AWB Gain (R Gain)
0x11	MAXAWBGAIN_2_L	0xFF	RW	Max AWB Gain (R Gain)
0x12	CURVEAWBOPTIONS	0xA1	RW	Bit[7]: Enable advanced AWB map Bit[6:0]: Reserved
0x20	CURVEAWBMINNUM_H	0x04	RW	AWB Min Num_h
0x21	CURVEAWBMINNUM_L	0x00	RW	AWB Min Num_l

table A-2 AWB registers (sheet 2 of 5)

address	register name	default value	R/W	description
0x22	CURVEAWBXSCALE_H	0x00	RW	XScale_h
0x23	CURVEAWBXSCALE_L	0x2E	RW	XScale_l
0x24	CURVEAWBYSCALE_H	0x00	RW	YScale_h
0x25	CURVEAWBYSCALE_L	0x19	RW	YScale_l
0x37	CURVEAWBDEBUG	0x00	RW	Debug option
0x4C	ADJUSTSMALLPDGAINENBLE	0x01	RW	AdjustSmallPDGainEnble
0x4D	ADAPTIVEPDGAINENABLE	0x00	RW	AdaptivePDGainEnable
0x4E	BRGAINTH_0_H	0x00	RW	BRGainTh_0_h
0x4F	BRGAINTH_0_L	0x81	RW	BRGainTh_0_l
0x50	BRGAINTH_1_H	0x00	RW	BRGainTh_1_h
0x51	BRGAINTH_1_L	0xB0	RW	BRGainTh_1_l
0x52	BRGAINTH_2_H	0x01	RW	BRGainTh_2_h
0x53	BRGAINTH_2_L	0x1E	RW	BRGainTh_2_l
0x54	PDGAINB_0_H	0x00	RW	PDGainB_0_h
0x55	PDGAINB_0_L	0x80	RW	PDGainB_0_l
0x56	PDGAINB_1_H	0x00	RW	PDGainB_1_h
0x57	PDGAINB_1_L	0x80	RW	PDGainB_1_l
0x58	PDGAINB_2_H	0x00	RW	PDGainB_2_h
0x59	PDGAINB_2_L	0x80	RW	PDGainB_2_l
0x5A	PDGAING_0_H	0x00	RW	PDGainG_0_h
0x5B	PDGAING_0_L	0x94	RW	PDGainG_0_l
0x5C	PDGAING_1_H	0x00	RW	PDGainG_1_h
0x5D	PDGAING_1_L	0x94	RW	PDGainG_1_l
0x5E	PDGAING_2_H	0x00	RW	PDGainG_2_h
0x5F	PDGAING_2_L	0x94	RW	PDGainG_2_l
0x60	PDGAINR_0_H	0x00	RW	PDGainR_0_h
0x61	PDGAINR_0_L	0xA0	RW	PDGainR_0_l
0x62	PDGAINR_1_H	0x00	RW	PDGainR_1_h
0x63	PDGAINR_1_L	0xA0	RW	PDGainR_1_l
0x64	PDGAINR_2_H	0x00	RW	PDGainR_2_h

table A-2 AWB registers (sheet 3 of 5)

address	register name	default value	R/W	description
0x65	PDGAINR_2_L	0xA0	RW	PDGainR_2_l
0x66	AWBSTEP1	0x04	RW	AWBStep1
0x67	AWBSTEP2	0x80	RW	AWBStep2
0x7C	MANUALAWBGAIN_0_0_H	0x01	RW	ManualAWBGain_0_0_h
0x7D	MANUALAWBGAIN_0_0_L	0x28	RW	ManualAWBGain_0_0_l
0x7E	MANUALAWBGAIN_0_1_H	0x01	RW	ManualAWBGain_0_1_h
0x7F	MANUALAWBGAIN_0_1_L	0x00	RW	ManualAWBGain_0_1_l
0x80	MANUALAWBGAIN_0_2_H	0x01	RW	ManualAWBGain_0_2_h
0x81	MANUALAWBGAIN_0_2_L	0x00	RW	ManualAWBGain_0_2_l
0x82	MANUALAWBGAIN_0_3_H	0x02	RW	ManualAWBGain_0_3_h
0x83	MANUALAWBGAIN_0_3_L	0x00	RW	ManualAWBGain_0_3_l
0x84	MANUALAWBGAIN_1_0_H	0x01	RW	ManualAWBGain_1_0_h
0x85	MANUALAWBGAIN_1_0_L	0x28	RW	ManualAWBGain_1_0_l
0x86	MANUALAWBGAIN_1_1_H	0x01	RW	ManualAWBGain_1_1_h
0x87	MANUALAWBGAIN_1_1_L	0x00	RW	ManualAWBGain_1_1_l
0x88	MANUALAWBGAIN_1_2_H	0x01	RW	ManualAWBGain_1_2_h
0x89	MANUALAWBGAIN_1_2_L	0x00	RW	ManualAWBGain_1_2_l
0x8A	MANUALAWBGAIN_1_3_H	0x02	RW	ManualAWBGain_1_3_h
0x8B	MANUALAWBGAIN_1_3_L	0x00	RW	ManualAWBGain_1_3_l
0x8C	MANUALAWBGAIN_2_0_H	0x01	RW	ManualAWBGain_2_0_h
0x8D	MANUALAWBGAIN_2_0_L	0x28	RW	ManualAWBGain_2_0_l
0x8E	MANUALAWBGAIN_2_1_H	0x01	RW	ManualAWBGain_2_1_h
0x8F	MANUALAWBGAIN_2_1_L	0x00	RW	ManualAWBGain_2_1_l
0x90	MANUALAWBGAIN_2_2_H	0x01	RW	ManualAWBGain_2_2_h
0x91	MANUALAWBGAIN_2_2_L	0x00	RW	ManualAWBGain_2_2_l
0x92	MANUALAWBGAIN_2_3_H	0x02	RW	ManualAWBGain_2_3_h
0x93	MANUALAWBGAIN_2_3_L	0x00	RW	ManualAWBGain_2_3_l
0x94	MANUALAWBOFFSET_0_BYTE0	0x00	RW	ManualAWBOffset_0_Byte0
0x95	MANUALAWBOFFSET_0_BYTE1	0x00	RW	ManualAWBOffset_0_Byte1
0x96	MANUALAWBOFFSET_0_BYTE2	0x00	RW	ManualAWBOffset_0_Byte2

table A-2 AWB registers (sheet 4 of 5)

address	register name	default value	R/W	description
0x97	MANUALAWBOFFSET_0_BYTE3	0x00	RW	ManualAWBOffset_0_Byte3
0x98	MANUALAWBOFFSET_1_BYTE0	0x00	RW	ManualAWBOffset_1_Byte0
0x99	MANUALAWBOFFSET_1_BYTE1	0x00	RW	ManualAWBOffset_1_Byte1
0x9A	MANUALAWBOFFSET_1_BYTE2	0x00	RW	ManualAWBOffset_1_Byte2
0x9B	MANUALAWBOFFSET_1_BYTE3	0x00	RW	ManualAWBOffset_1_Byte3
0x9C	MANUALAWBOFFSET_2_BYTE0	0x00	RW	ManualAWBOffset_2_Byte0
0x9D	MANUALAWBOFFSET_2_BYTE1	0x00	RW	ManualAWBOffset_2_Byte1
0x9E	MANUALAWBOFFSET_2_BYTE2	0x00	RW	ManualAWBOffset_2_Byte2
0x9F	MANUALAWBOFFSET_2_BYTE3	0x00	RW	ManualAWBOffset_2_Byte3
0xA0	MANUALAWBOFFSET_3_BYTE0	0x00	RW	ManualAWBOffset_3_Byte0
0xA1	MANUALAWBOFFSET_3_BYTE1	0x00	RW	ManualAWBOffset_3_Byte1
0xA2	MANUALAWBOFFSET_3_BYTE2	0x00	RW	ManualAWBOffset_3_Byte2
0xA3	MANUALAWBOFFSET_3_BYTE3	0x00	RW	ManualAWBOffset_3_Byte3
0xA4	MANUALAWBOFFSETS_0_BYTE0	0x00	RW	ManualAWBOffsetS_0_Byte0
0xA5	MANUALAWBOFFSETS_0_BYTE1	0x00	RW	ManualAWBOffsetS_0_Byte1
0xA6	MANUALAWBOFFSETS_0_BYTE2	0x00	RW	ManualAWBOffsetS_0_Byte2
0xA7	MANUALAWBOFFSETS_0_BYTE3	0x00	RW	ManualAWBOffsetS_0_Byte3
0xA8	MANUALAWBOFFSETS_1_BYTE0	0x00	RW	ManualAWBOffsetS_1_Byte0
0xA9	MANUALAWBOFFSETS_1_BYTE1	0x00	RW	ManualAWBOffsetS_1_Byte1
0xAA	MANUALAWBOFFSETS_1_BYTE2	0x00	RW	ManualAWBOffsetS_1_Byte2
0xAB	MANUALAWBOFFSETS_1_BYTE3	0x00	RW	ManualAWBOffsetS_1_Byte3
0xAC	MANUALAWBOFFSETS_2_BYTE0	0x00	RW	ManualAWBOffsetS_2_Byte0
0xAD	MANUALAWBOFFSETS_2_BYTE1	0x00	RW	ManualAWBOffsetS_2_Byte1
0xAE	MANUALAWBOFFSETS_2_BYTE2	0x00	RW	ManualAWBOffsetS_2_Byte2
0xAF	MANUALAWBOFFSETS_2_BYTE3	0x00	RW	ManualAWBOffsetS_2_Byte3
0xB0	MANUALAWBOFFSETS_3_BYTE0	0x00	RW	ManualAWBOffsetS_3_Byte0
0xB1	MANUALAWBOFFSETS_3_BYTE1	0x00	RW	ManualAWBOffsetS_3_Byte1
0xB2	MANUALAWBOFFSETS_3_BYTE2	0x00	RW	ManualAWBOffsetS_3_Byte2
0xB3	MANUALAWBOFFSETS_3_BYTE3	0x00	RW	ManualAWBOffsetS_3_Byte3
0xB4	AWBREGIONLLBRTH0_H	0x08	RW	AWBRegionLLBrTh0_h

table A-2 AWB registers (sheet 5 of 5)

address	register name	default value	R/W	description
0xB5	AWBREGIONLLBRTH0_L	0x20	RW	AWBRegionLLBrTh0_l
0xB6	AWBREGIONLLBRTH1_H	0x05	RW	AWBRegionLLBrTh1_h
0xB7	AWBREGIONLLBRTH1_L	0x00	RW	AWBRegionLLBrTh1_l
0xB8	AWBREGIONLLBRTH2_H	0x04	RW	AWBRegionLLBrTh2_h
0xB9	AWBREGIONLLBRTH2_L	0x10	RW	AWBRegionLLBrTh2_l
0xBA	AWBREGIONLLBRTH3_H	0x02	RW	AWBRegionLLBrTh3_h
0xBB	AWBREGIONLLBRTH3_L	0x08	RW	AWBRegionLLBrTh3_l
0xBC	BGAINSHIFT_H	0x01	RW	BGainShift_h
0xBD	BGAINSHIFT_L	0x01	RW	BGainShift_l
0xBE	GGAINSHIFT_H	0x01	RW	GGainShift_h
0xBF	GGAINSHIFT_L	0x00	RW	GGainShift_l
0xC0	RGAINSHIFT_H	0x01	RW	RGainShift_h
0xC1	RGAINSHIFT_L	0x00	RW	RGainShift_l
0xC2	MANUALAWBGAINSHIFTENABLE	0x01	RW	ManualAWBGainShiftEnable
0xC3	AWBMAP_0	0x08	RW	AWB Map_0
0xC4~ 0x141	AWBMAP REGISTERS	0x08	RW	AWB Map Registers
0x142	AWBMAP_127	0x08	RW	AWB Map_127
0x143	MASK1_0	0xFF	RW	Mask1_0
0x144~ 0x161	MASK1 REGISTERS	0xFF	RW	Mask 1 Registers
0x162	MASK1_31	0xFF	RW	Mask1_31
0x163	MASK2_0	0xFF	RW	Mask2_0
0x164~ 0x181	MASK2 REGISTERS	0xFF	RW	Mask 2 Registers
0x182	MASK2_31	0xFF	RW	Mask2_31
0x183	REGIONAWBMODE	0x00	RW	RegionAWBMode
0x184	REGIONSTATISTICSENABLE	0x01	RW	RegionStatisticsEnable
0x195	nAWBSTABLERANGE_1		RW	Stable Range Threshold 1
0x196	nAWBSTABLERANGE_2		RW	Stable Range Threshold 2

A.3 high dynamic range (HDR) control

table A-3 HDR control registers (sheet 1 of 11)

address	register name	default value	R/W	description
0x1A4	COMBINEERRORCONTROL POINT_BYTE0	0x00	RW	Bit[7:0]: Threshold to enable combination error statistics calculation[31:24]
0x1A5	COMBINEERRORCONTROL POINT_BYTE1	0x02	RW	Bit[7:0]: Threshold to enable combination error statistics calculation[23:16]
0x1A6	COMBINEERRORCONTROL POINT_BYTE2	0x00	RW	Bit[7:0]: Threshold to enable combination error statistics calculation[15:8]
0x1A7	COMBINEERRORCONTROL POINT_BYTE3	0x00	RW	Bit[7:0]: Threshold to enable combination error statistics calculation[7:0]
0x1A8	STEPE	0x02	RW	Combine Error Change Step
0x1A9	TONEMAPPINGENABLE	0x01	RW	Tone Mapping Enable 0: Disable 1: Enable
0x1AA	GAINENHANCEMENT CONTROL	0x01	RW	Auto Tone Curve Enhancement Level Reduction Enable 0: Disable 1: Enable
0x1AB	MANUALGAMMAENABLE	0x00	RW	Manual Tone Compression Gamma Enable 0: Disable 1: Enable
0x1AC	AUTOTONEMAPPING CURVE	0x01	RW	Auto Tone Mapping Curve Enable 0: Disable 1: Enable
0x1AD	CURVESTEP	0x02	RW	Tone Curve Change Step
0x1B4	MINDYNAMICRANGE_H	0x00	RW	Bit[7:0]: Low bound of dynamic range for tone curve reduction[15:8]
0x1B5	MINDYNAMICRANGE_L	0x10	RW	Bit[7:0]: Low bound of dynamic range for tone curve reduction[7:0]
0x1B6	MAXDYNAMICRANGE_H	0x00	RW	Bit[7:0]: High bound of dynamic range for tone curve reduction[15:8]
0x1B7	MAXDYNAMICRANGE_L	0x40	RW	Bit[7:0]: High bound of dynamic range for tone curve reduction[7:0]
0x1B8	GAMMALIST_0_H	0x00	RW	Bit[7:0]: Base curve for tone curve P0[15:8]
0x1B9	GAMMALIST_0_L	0x2C	RW	Bit[7:0]: Base curve for tone curve P0[7:0]
0x1BA	GAMMALIST_1_H	0x00	RW	Bit[7:0]: Base curve for tone curve P1[15:8]

table A-3 HDR control registers (sheet 2 of 11)

address	register name	default value	R/W	description
0x1BB	GAMMALIST_1_L	0x9B	RW	Bit[7:0]: Base curve for tone curve P1[7:0]
0x1BC	GAMMALIST_2_H	0x00	RW	Bit[7:0]: Base curve for tone curve P2[15:8]
0x1BD	GAMMALIST_2_L	0xD4	RW	Bit[7:0]: Base curve for tone curve P2[7:0]
0x1BE	GAMMALIST_3_H	0x01	RW	Bit[7:0]: Base curve for tone curve P3[15:8]
0x1BF	GAMMALIST_3_L	0x22	RW	Bit[7:0]: Base curve for tone curve P3[7:0]
0x1C0	GAMMALIST_4_H	0x01	RW	Bit[7:0]: Base curve for tone curve P4[15:8]
0x1C1	GAMMALIST_4_L	0x8E	RW	Bit[7:0]: Base curve for tone curve P4[7:0]
0x1C2	GAMMALIST_5_H	0x01	RW	Bit[7:0]: Base curve for tone curve P5[15:8]
0x1C3	GAMMALIST_5_L	0xD2	RW	Bit[7:0]: Base curve for tone curve P5[7:0]
0x1C4	GAMMALIST_6_H	0x02	RW	Bit[7:0]: Base curve for tone curve P6[15:8]
0x1C5	GAMMALIST_6_L	0x21	RW	Bit[7:0]: Base curve for tone curve P6[7:0]
0x1C6	GAMMALIST_7_H	0x02	RW	Bit[7:0]: Base curve for tone curve P7[15:8]
0x1C7	GAMMALIST_7_L	0x4E	RW	Bit[7:0]: Base curve for tone curve P7[7:0]
0x1C8	GAMMALIST_8_H	0x02	RW	Bit[7:0]: Base curve for tone curve P8[15:8]
0x1C9	GAMMALIST_8_L	0x7E	RW	Bit[7:0]: Base curve for tone curve P8[7:0]
0x1CA	GAMMALIST_9_H	0x02	RW	Bit[7:0]: Base curve for tone curve P9[15:8]
0x1CB	GAMMALIST_9_L	0xB3	RW	Bit[7:0]: Base curve for tone curve P9[7:0]
0x1CC	GAMMALIST_10_H	0x02	RW	Bit[7:0]: Base curve for tone curve P10[15:8]
0x1CD	GAMMALIST_10_L	0xEB	RW	Bit[7:0]: Base curve for tone curve P10[7:0]
0x1CE	GAMMALIST_11_H	0x03	RW	Bit[7:0]: Base curve for tone curve P11[15:8]
0x1CF	GAMMALIST_11_L	0x29	RW	Bit[7:0]: Base curve for tone curve P11[7:0]
0x1D0	GAMMALIST_12_H	0x03	RW	Bit[7:0]: Base curve for tone curve P12[15:8]
0x1D1	GAMMALIST_12_L	0x6B	RW	Bit[7:0]: Base curve for tone curve P12[7:0]
0x1D2	GAMMALIST_13_H	0x03	RW	Bit[7:0]: Base curve for tone curve P13[15:8]
0x1D3	GAMMALIST_13_L	0xB2	RW	Bit[7:0]: Base curve for tone curve P13[7:0]
0x1D4	GAMMALIST_14_H	0x03	RW	Bit[7:0]: Base curve for tone curve P14[15:8]
0x1D5	GAMMALIST_14_L	0xD8	RW	Bit[7:0]: Base curve for tone curve P14[7:0]

table A-3 HDR control registers (sheet 3 of 11)

address	register name	default value	R/W	description
0x1D6	GAMMALIST_15_H	0x03	RW	Bit[7:0]: Base curve for tone curve P15[15:8]
0x1D7	GAMMALIST_15_L	0xFF	RW	Bit[7:0]: Base curve for tone curve P15[7:0]
0x1D8	LIGHTCURVE_0_H	0x01	RW	Bit[7:0]: High bound of tone curve P0[15:8]
0x1D9	LIGHTCURVE_0_L	0x28	RW	Bit[7:0]: High bound of tone curve P0[7:0]
0x1DA	LIGHTCURVE_1_H	0x01	RW	Bit[7:0]: High bound of tone curve P1[15:8]
0x1DB	LIGHTCURVE_1_L	0x45	RW	Bit[7:0]: High bound of tone curve P1[7:0]
0x1DC	LIGHTCURVE_2_H	0x01	RW	Bit[7:0]: High bound of tone curve P2[15:8]
0x1DD	LIGHTCURVE_2_L	0x88	RW	Bit[7:0]: High bound of tone curve P2[7:0]
0x1DE	LIGHTCURVE_3_H	0x01	RW	Bit[7:0]: High bound of tone curve P3[15:8]
0x1DF	LIGHTCURVE_3_L	0xEE	RW	Bit[7:0]: High bound of tone curve P3[7:0]
0x1E0	LIGHTCURVE_4_H	0x02	RW	Bit[7:0]: High bound of tone curve P4[15:8]
0x1E1	LIGHTCURVE_4_L	0x17	RW	Bit[7:0]: High bound of tone curve P4[7:0]
0x1E2	LIGHTCURVE_5_H	0x02	RW	Bit[7:0]: High bound of tone curve P5[15:8]
0x1E3	LIGHTCURVE_5_L	0x40	RW	Bit[7:0]: High bound of tone curve P5[7:0]
0x1E4	LIGHTCURVE_6_H	0x02	RW	Bit[7:0]: High bound of tone curve P6[15:8]
0x1E5	LIGHTCURVE_6_L	0x6B	RW	Bit[7:0]: High bound of tone curve P6[7:0]
0x1E6	LIGHTCURVE_7_H	0x02	RW	Bit[7:0]: High bound of tone curve P7[15:8]
0x1E7	LIGHTCURVE_7_L	0x98	RW	Bit[7:0]: High bound of tone curve P7[7:0]
0x1E8	LIGHTCURVE_8_H	0x02	RW	Bit[7:0]: High bound of tone curve P8[15:8]
0x1E9	LIGHTCURVE_8_L	0xC2	RW	Bit[7:0]: High bound of tone curve P8[7:0]
0x1EA	LIGHTCURVE_9_H	0x02	RW	Bit[7:0]: High bound of tone curve P9[15:8]
0x1EB	LIGHTCURVE_9_L	0xE7	RW	Bit[7:0]: High bound of tone curve P9[7:0]
0x1EC	LIGHTCURVE_10_H	0x03	RW	Bit[7:0]: High bound of tone curve P10[15:8]
0x1ED	LIGHTCURVE_10_L	0x20	RW	Bit[7:0]: High bound of tone curve P10[7:0]
0x1EE	LIGHTCURVE_11_H	0x03	RW	Bit[7:0]: High bound of tone curve P11[15:8]
0x1EF	LIGHTCURVE_11_L	0x4E	RW	Bit[7:0]: High bound of tone curve P11[7:0]
0x1F0	LIGHTCURVE_12_H	0x03	RW	Bit[7:0]: High bound of tone curve P12[15:8]
0x1F1	LIGHTCURVE_12_L	0x89	RW	Bit[7:0]: High bound of tone curve P12[7:0]

table A-3 HDR control registers (sheet 4 of 11)

address	register name	default value	R/W	description
0x1F2	LIGHTCURVE_13_H	0x03	RW	Bit[7:0]: High bound of tone curve P13[15:8]
0x1F3	LIGHTCURVE_13_L	0xBF	RW	Bit[7:0]: High bound of tone curve P13[7:0]
0x1F4	LIGHTCURVE_14_H	0x03	RW	Bit[7:0]: High bound of tone curve P14[15:8]
0x1F5	LIGHTCURVE_14_L	0xE0	RW	Bit[7:0]: High bound of tone curve P14[7:0]
0x1F6	LIGHTCURVE_15_H	0x03	RW	Bit[7:0]: High bound of tone curve P15[15:8]
0x1F7	LIGHTCURVE_15_L	0xFF	RW	Bit[7:0]: High bound of tone curve P15[7:0]
0x1F8	DARKCURVE_0_H	0x00	RW	Bit[7:0]: Low bound of tone curve P0[15:8]
0x1F9	DARKCURVE_0_L	0x11	RW	Bit[7:0]: Low bound of tone curve P0[7:0]
0x1FA	DARKCURVE_1_H	0x00	RW	Bit[7:0]: Low bound of tone curve P1[15:8]
0x1FB	DARKCURVE_1_L	0x28	RW	Bit[7:0]: Low bound of tone curve P1[7:0]
0x1FC	DARKCURVE_2_H	0x00	RW	Bit[7:0]: Low bound of tone curve P2[15:8]
0x1FD	DARKCURVE_2_L	0x5B	RW	Bit[7:0]: Low bound of tone curve P2[7:0]
0x1FE	DARKCURVE_3_H	0x00	RW	Bit[7:0]: Low bound of tone curve P3[15:8]
0x1FF	DARKCURVE_3_L	0xC8	RW	Bit[7:0]: Low bound of tone curve P3[7:0]
0x200	DARKCURVE_4_H	0x01	RW	Bit[7:0]: Low bound of tone curve P4[15:8]
0x201	DARKCURVE_4_L	0x09	RW	Bit[7:0]: Low bound of tone curve P4[7:0]
0x202	DARKCURVE_5_H	0x01	RW	Bit[7:0]: Low bound of tone curve P5[15:8]
0x203	DARKCURVE_5_L	0x57	RW	Bit[7:0]: Low bound of tone curve P5[7:0]
0x204	DARKCURVE_6_H	0x01	RW	Bit[7:0]: Low bound of tone curve P6[15:8]
0x205	DARKCURVE_6_L	0x9B	RW	Bit[7:0]: Low bound of tone curve P6[7:0]
0x206	DARKCURVE_7_H	0x01	RW	Bit[7:0]: Low bound of tone curve P7[15:8]
0x207	DARKCURVE_7_L	0xD4	RW	Bit[7:0]: Low bound of tone curve P7[7:0]
0x208	DARKCURVE_8_H	0x02	RW	Bit[7:0]: Low bound of tone curve P8[15:8]
0x209	DARKCURVE_8_L	0x01	RW	Bit[7:0]: Low bound of tone curve P8[7:0]
0x20A	DARKCURVE_9_H	0x02	RW	Bit[7:0]: Low bound of tone curve P9[15:8]
0x20B	DARKCURVE_9_L	0x25	RW	Bit[7:0]: Low bound of tone curve P9[7:0]
0x20C	DARKCURVE_10_H	0x02	RW	Bit[7:0]: Low bound of tone curve P10[15:8]
0x20D	DARKCURVE_10_L	0x64	RW	Bit[7:0]: Low bound of tone curve P10[7:0]

table A-3 HDR control registers (sheet 5 of 11)

address	register name	default value	R/W	description
0x20E	DARKCURVE_11_H	0x02	RW	Bit[7:0]: Low bound of tone curve P11[15:8]
0x20F	DARKCURVE_11_L	0x9B	RW	Bit[7:0]: Low bound of tone curve P11[7:0]
0x210	DARKCURVE_12_H	0x02	RW	Bit[7:0]: Low bound of tone curve P12[15:8]
0x211	DARKCURVE_12_L	0xF9	RW	Bit[7:0]: Low bound of tone curve P12[7:0]
0x212	DARKCURVE_13_H	0x03	RW	Bit[7:0]: Low bound of tone curve P13[15:8]
0x213	DARKCURVE_13_L	0x4E	RW	Bit[7:0]: Low bound of tone curve P13[7:0]
0x214	DARKCURVE_14_H	0x03	RW	Bit[7:0]: Low bound of tone curve P14[15:8]
0x215	DARKCURVE_14_L	0x99	RW	Bit[7:0]: Low bound of tone curve P14[7:0]
0x216	DARKCURVE_15_H	0x03	RW	Bit[7:0]: Low bound of tone curve P15[15:8]
0x217	DARKCURVE_15_L	0xFF	RW	Bit[7:0]: Low bound of tone curve P15[7:0]
0x218	MANUALTONECURVE_0_H	0x00	RW	Bit[7:0]: Manual tone curve P0[15:8]
0x219	MANUALTONECURVE_0_L	0x01	RW	Bit[7:0]: Manual tone curve P0[7:0]
0x21A	MANUALTONECURVE_1_H	0x00	RW	Bit[7:0]: Manual tone curve P1[15:8]
0x21B	MANUALTONECURVE_1_L	0x10	RW	Bit[7:0]: Manual tone curve P1[7:0]
0x21C	MANUALTONECURVE_2_H	0x00	RW	Bit[7:0]: Manual tone curve P2[15:8]
0x21D	MANUALTONECURVE_2_L	0x20	RW	Bit[7:0]: Manual tone curve P2[7:0]
0x21E	MANUALTONECURVE_3_H	0x00	RW	Bit[7:0]: Manual tone curve P3[15:8]
0x21F	MANUALTONECURVE_3_L	0x40	RW	Bit[7:0]: Manual tone curve P3[7:0]
0x220	MANUALTONECURVE_4_H	0x00	RW	Bit[7:0]: Manual tone curve P4[15:8]
0x221	MANUALTONECURVE_4_L	0x80	RW	Bit[7:0]: Manual tone curve P4[7:0]
0x222	MANUALTONECURVE_5_H	0x00	RW	Bit[7:0]: Manual tone curve P5[15:8]
0x223	MANUALTONECURVE_5_L	0xB5	RW	Bit[7:0]: Manual tone curve P5[7:0]
0x224	MANUALTONECURVE_6_H	0x01	RW	Bit[7:0]: Manual tone curve P6[15:8]
0x225	MANUALTONECURVE_6_L	0x00	RW	Bit[7:0]: Manual tone curve P6[7:0]
0x226	MANUALTONECURVE_7_H	0x01	RW	Bit[7:0]: Manual tone curve P7[15:8]
0x227	MANUALTONECURVE_7_L	0x30	RW	Bit[7:0]: Manual tone curve P7[7:0]
0x228	MANUALTONECURVE_8_H	0x01	RW	Bit[7:0]: Manual tone curve P8[15:8]
0x229	MANUALTONECURVE_8_L	0x6A	RW	Bit[7:0]: Manual tone curve P8[7:0]
0x22A	MANUALTONECURVE_9_H	0x01	RW	Bit[7:0]: Manual tone curve P9[15:8]
0x22B	MANUALTONECURVE_9_L	0xAF	RW	Bit[7:0]: Manual tone curve P9[7:0]

table A-3 HDR control registers (sheet 6 of 11)

address	register name	default value	R/W	description
0x22C	MANUALTONECURVE_10_H	0x02	RW	Bit[7:0]: Manual tone curve P10[15:8]
0x22D	MANUALTONECURVE_10_L	0x00	RW	Bit[7:0]: Manual tone curve P10[7:0]
0x22E	MANUALTONECURVE_11_H	0x02	RW	Bit[7:0]: Manual tone curve P11[15:8]
0x22F	MANUALTONECURVE_11_L	0x61	RW	Bit[7:0]: Manual tone curve P11[7:0]
0x230	MANUALTONECURVE_12_H	0x02	RW	Bit[7:0]: Manual tone curve P12[15:8]
0x231	MANUALTONECURVE_12_L	0xD4	RW	Bit[7:0]: Manual tone curve P12[7:0]
0x232	MANUALTONECURVE_13_H	0x03	RW	Bit[7:0]: Manual tone curve P13[15:8]
0x233	MANUALTONECURVE_13_L	0x5D	RW	Bit[7:0]: Manual tone curve P13[7:0]
0x234	MANUALTONECURVE_14_H	0x03	RW	Bit[7:0]: Manual tone curve P14[15:8]
0x235	MANUALTONECURVE_14_L	0xAB	RW	Bit[7:0]: Manual tone curve P14[7:0]
0x236	MANUALTONECURVE_15_H	0x03	RW	Bit[7:0]: Manual tone curve P15[15:8]
0x237	MANUALTONECURVE_15_L	0xFF	RW	Bit[7:0]: Manual tone curve P15[7:0]
0x238	MINGAMMALIST_0_H	0x01	RW	Bit[7:0]: Min tone compression gamma list 1[15:8]
0x239	MINGAMMALIST_0_L	0x00	RW	Bit[7:0]: Min tone compression gamma list 1[7:0]
0x23A	MINGAMMALIST_1_H	0x01	RW	Bit[7:0]: Min tone compression gamma list 2[15:8]
0x23B	MINGAMMALIST_1_L	0x00	RW	Bit[7:0]: Min tone compression gamma list 2[7:0]
0x23C	MINGAMMALIST_2_H	0x01	RW	Bit[7:0]: Min tone compression gamma list 3[15:8]
0x23D	MINGAMMALIST_2_L	0x00	RW	Bit[7:0]: Min tone compression gamma list 3[7:0]
0x23E	MAXGAMMALIST_0_H	0x01	RW	Bit[7:0]: Max tone compression gamma list 1[15:8]
0x23F	MAXGAMMALIST_0_L	0x32	RW	Bit[7:0]: Max tone compression gamma list 1[7:0]
0x240	MAXGAMMALIST_1_H	0x01	RW	Bit[7:0]: Max tone compression gamma list 2[15:8]
0x241	MAXGAMMALIST_1_L	0xE8	RW	Bit[7:0]: Max tone compression gamma list 2[7:0]
0x242	MAXGAMMALIST_2_H	0x02	RW	Bit[7:0]: Max tone compression gamma list 3[15:8]

table A-3 HDR control registers (sheet 7 of 11)

address	register name	default value	R/W	description
0x243	MAXGAMMALIST_2_L	0x74	RW	Bit[7:0]: Max tone compression gamma list 3[7:0]
0x244	DYNAMICRANGELIST_0_H	0x00	RW	Bit[7:0]: Dynamic range list 1[15:8]
0x245	DYNAMICRANGELIST_0_L	0x10	RW	Bit[7:0]: Dynamic range list 1[7:0]
0x246	DYNAMICRANGELIST_1_H	0x00	RW	Bit[7:0]: Dynamic range list 2[15:8]
0x247	DYNAMICRANGELIST_1_L	0x40	RW	Bit[7:0]: Dynamic range list 2[7:0]
0x248	DYNAMICRANGELIST_2_H	0x00	RW	Bit[7:0]: Dynamic range list 3[15:8]
0x249	DYNAMICRANGELIST_2_L	0x80	RW	Bit[7:0]: Dynamic range list 3[7:0]
0x24A	MANUALGAMMA_H	0x00	RW	Bit[7:0]: Manual tone compression gamma[15:8]
0x24B	MANUALGAMMA_L	0x32	RW	Bit[7:0]: Manual tone compression gamma[7:0]
0x24C	LOGTARGETY_0_H	0x97	RW	Bit[7:0]: Image mean value target for frame A in ABC mode[15:8]
0x24D	LOGTARGETY_0_L	0x00	RW	Bit[7:0]: Image mean value target for frame A in ABC mode[7:0]
0x24E	LOGTARGETY_1_H	0x97	RW	Bit[7:0]: Image mean value target for frame B in ABC mode[15:8]
0x24F	LOGTARGETY_1_L	0x00	RW	Bit[7:0]: Image mean value target for frame B in ABC mode[7:0]
0x250	LOGTARGETY_2_H	0x97	RW	Bit[7:0]: Image mean value target for frame C in ABC mode[15:8]
0x251	LOGTARGETY_2_L	0x00	RW	Bit[7:0]: Image mean value target for frame C in ABC mode[7:0]
0x254	GLOBALGAMMASTEP	0x04	RW	Tone Compression Gamma Step
0x255	REDUCENOISEINDARK ENABLE	0x00	RW	Reduce Noise for Dark Scenes Enable 0: Disable 1: Enable
0x256	REDUCENOISEMAXGAIN_H	0x02	RW	Bit[7:0]: High gain boundary for reduce noise function[15:8]
0x257	REDUCENOISEMAXGAIN_L	0x00	RW	Bit[7:0]: High gain boundary for reduce noise function[7:0]
0x258	REDUCENOISEMINGAIN_H	0x00	RW	Bit[7:0]: Low gain boundary for reduce noise function[15:8]
0x259	REDUCENOISEMINGAIN_L	0xA0	RW	Bit[7:0]: Low gain boundary for reduce noise function[7:0]

table A-3 HDR control registers (sheet 8 of 11)

address	register name	default value	R/W	description
0x25A	REDUCENOISESTEP	0x02	RW	Gain Step Value for Reduce Noise
0x25B	GLOBALMEANW	0x00	RW	Current Global Mean Value Weight to the Final Target
0x25C	GAMMASTEP	0x04	RW	Tone Compression Gamma Change Step
0x25D	FRAMESTABLE	0x05	RW	Frame Stable Number for Saturated Pixel Preserved Mode (2^framestable)
0x25E	SATURATEDPIXEL PRESERVED	0x00	RW	Saturated Pixel Preserved Mode 0: Disable 1: Enable
0x25F	GLOBALGAMMASHIFT ENABLE	0x00	RW	Tone Compression Gamma Shift Enable 0: Disable 1: Enable
0x260	GLOBALGAMMASHIFT	0x00	RW	Tone Compression Gamma Shift Value
0x261	LOCALENHANCEMENT LEVEL	0x21	RW	Local Enhancement Level
0x262	LOWCONTRASTLEVEL	0x01	RW	Tone Curve Enhancement Low Level
0x263	HIGHCONTRASTLEVEL	0x03	RW	Tone Curve Enhancement High Level
0x264	SPREMATRIXENABLE	0x00	RW	SPD Pre Matrix Enable 0: Disable 1: Enable
0x265	SPREAUTOCTENABLE	0x01	RW	SPD Pre Matrix Illuminant Adaptation Enable 0: Disable 1: Enable
0x266	EXPCHANGELOGTH_H	0x00	RW	Bit[7:0]: Exposure change threshold[15:8]
0x267	EXPCHANGELOGTH_L	0x10	RW	Bit[7:0]: Exposure change threshold[7:0]
0x268	YTH0TARGET_0_H	0x78	RW	Bit[7:0]: Dark contrast enhancement Y Th0[15:8]
0x269	YTH0TARGET_0_L	0x00	RW	Bit[7:0]: Dark contrast enhancement Y Th0[7:0]
0x26A	YTH1TARGET_1_H	0x80	RW	Bit[7:0]: Dark contrast enhancement Y Th1[15:8]
0x26B	YTH1TARGET_1_L	0x00	RW	Bit[7:0]: Dark contrast enhancement Y Th1[7:0]
0x26C	YTH0TARGET_2_H	0x98	RW	Bit[7:0]: Dark contrast enhancement Y Th2[15:8]
0x26D	YTH0TARGET_2_L	0x00	RW	Bit[7:0]: Dark contrast enhancement Y Th2[7:0]

table A-3 HDR control registers (sheet 9 of 11)

address	register name	default value	R/W	description
0x26E	YTH1TARGET_3_H	0x9C	RW	Bit[7:0]: Dark contrast enhancement Y Th3[15:8]
0x26F	YTH1TARGET_3_L	0xEC	RW	Bit[7:0]: Dark contrast enhancement Y Th3[7:0]
0x270	SPRECTLIST_0_H	0x00	RW	Bit[7:0]: B/R for D65[15:8]
0x271	SPRECTLIST_0_L	0x98	RW	Bit[7:0]: B/R for D65[7:0]
0x272	SPRECTLIST_1_H	0x00	RW	Bit[7:0]: B/R for CWF[15:8]
0x273	SPRECTLIST_1_L	0xAC	RW	Bit[7:0]: B/R for CWF[7:0]
0x274	SPRECTLIST_2_H	0x01	RW	Bit[7:0]: B/R for A[15:8]
0x275	SPRECTLIST_2_L	0x00	RW	Bit[7:0]: B/R for A[7:0]
0x276	INVMLMS_D_0_0_H	0x01	RW	Bit[7:0]: SPD to LPD D65 matrix 00[15:8]
0x277	INVMLMS_D_0_0_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 00[7:0]
0x278	INVMLMS_D_0_1_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 01[15:8]
0x279	INVMLMS_D_0_1_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 01[7:0]
0x27A	INVMLMS_D_0_2_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 02[15:8]
0x27B	INVMLMS_D_0_2_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 02[7:0]
0x27C	INVMLMS_D_1_0_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 10[15:8]
0x27D	INVMLMS_D_1_0_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 0[7:0]
0x27E	INVMLMS_D_1_1_H	0x01	RW	Bit[7:0]: SPD to LPD D65 matrix 11[15:8]
0x27F	INVMLMS_D_1_1_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 11[7:0]
0x280	INVMLMS_D_1_2_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 12[15:8]
0x281	INVMLMS_D_1_2_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 12[7:0]
0x282	INVMLMS_D_2_0_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 20[15:8]
0x283	INVMLMS_D_2_0_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 0[7:0]
0x284	INVMLMS_D_2_1_H	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 21[15:8]
0x285	INVMLMS_D_2_1_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 21[7:0]
0x286	INVMLMS_D_2_2_H	0x01	RW	Bit[7:0]: SPD to LPD D65 matrix 22[15:8]
0x287	INVMLMS_D_2_2_L	0x00	RW	Bit[7:0]: SPD to LPD D65 matrix 22[7:0]
0x288	INVMLMS_C_0_0_H	0x01	RW	Bit[7:0]: SPD to LPD CWF matrix 00[15:8]
0x289	INVMLMS_C_0_0_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 00[7:0]
0x28A	INVMLMS_C_0_1_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 01[15:8]

table A-3 HDR control registers (sheet 10 of 11)

address	register name	default value	R/W	description
0x28B	INVMLMS_C_0_1_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 01[7:0]
0x28C	INVMLMS_C_0_2_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 02[15:8]
0x28D	INVMLMS_C_0_2_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 02[7:0]
0x28E	INVMLMS_C_1_0_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 10[15:8]
0x28F	INVMLMS_C_1_0_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 0[7:0]
0x290	INVMLMS_C_1_1_H	0x01	RW	Bit[7:0]: SPD to LPD CWF matrix 11[15:8]
0x291	INVMLMS_C_1_1_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 11[7:0]
0x292	INVMLMS_C_1_2_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 12[15:8]
0x293	INVMLMS_C_1_2_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 12[7:0]
0x294	INVMLMS_C_2_0_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 20[15:8]
0x295	INVMLMS_C_2_0_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 20[7:0]
0x296	INVMLMS_C_2_1_H	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 21[15:8]
0x297	INVMLMS_C_2_1_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 21[7:0]
0x298	INVMLMS_C_2_2_H	0x01	RW	Bit[7:0]: SPD to LPD CWF matrix 22[15:8]
0x299	INVMLMS_C_2_2_L	0x00	RW	Bit[7:0]: SPD to LPD CWF matrix 22[7:0]
0x29A	INVMLMS_A_0_0_H	0x01	RW	Bit[7:0]: SPD to LPD A matrix 00[15:8]
0x29B	INVMLMS_A_0_0_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 00[7:0]
0x29C	INVMLMS_A_0_1_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 01[15:8]
0x29D	INVMLMS_A_0_1_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 01[7:0]
0x29E	INVMLMS_A_0_2_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 02[15:8]
0x29F	INVMLMS_A_0_2_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 02[7:0]
0x2A0	INVMLMS_A_1_0_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 10[15:8]
0x2A1	INVMLMS_A_1_0_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 10[7:0]
0x2A2	INVMLMS_A_1_1_H	0x01	RW	Bit[7:0]: SPD to LPD A matrix 11[15:8]
0x2A3	INVMLMS_A_1_1_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 11[7:0]
0x2A4	INVMLMS_A_1_2_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 12[15:8]
0x2A5	INVMLMS_A_1_2_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 12[7:0]
0x2A6	INVMLMS_A_2_0_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 20[15:8]
0x2A7	INVMLMS_A_2_0_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 20[7:0]
0x2A8	INVMLMS_A_2_1_H	0x00	RW	Bit[7:0]: SPD to LPD A matrix 21[15:8]

table A-3 HDR control registers (sheet 11 of 11)

address	register name	default value	R/W	description
0x2A9	INVMLMS_A_2_1_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 21[7:0]
0x2AA	INVMLMS_A_2_2_H	0x01	RW	Bit[7:0]: SPD to LPD A matrix 22[15:8]
0x2AB	INVMLMS_A_2_2_L	0x00	RW	Bit[7:0]: SPD to LPD A matrix 22[7:0]
0x2AC	LOGTARGETLD_H	0x98	RW	Bit[7:0]: Low dynamic range scene target[15:8]
0x2AD	LOGTARGETLD_L	0x00	RW	Bit[7:0]: Low dynamic range scene target[7:0]
0x2B2	MANUALHDRGAINENABLE	0x00	RW	Manual HDR Gain Enable
0x2B4	MANUALHDRGAIN_H	0x01	RW	Manual HDR Gain High Byte
0x2B5	MANUALHDRGAIN_L	0x00	RW	Manual HDR Gain Low Byte
0x2B6	MAXSUPPRESSGAIN_H	0x03	RW	Max Suppress Gain for Shadow and Highlights High Byte
0x2B7	MAXSUPPRESSGAIN_L	0x00	RW	Max Suppress Gain for Shadow and Highlights Low Byte
0x2B8	MANUALSUPPRESSGAIN	0x00	RW	Manual Suppress Gain Enable
0x2B9	HDRGAINBUFOPTION			
0x2BA	HTLIMITCOMBERROR ENABLE			
0x2BB	HTLIMITCOMBERROR STARTT			
0x2BC	HTLIMITCOMBERROR ENDT			
0x2BD	HTLIMITCOMBFLAG			
0x2BE	HDRGAINAVGBUF[0]			
0x2BF~ 0x2CE	HDRGAINAVGBUF[1]~ HDRGAINAVGBUF[16]			
0x2CF	HDRGAINAVGBUF[17]			
0x2D0	MANUALSUPPRESSHIFT			

A.4 auto exposure control (AEC)

table A-4 AEC registers (sheet 1 of 7)

address	register name	default value	R/W	description
0x2E1	FIXEDRATIOENABLE	0x00	RW	Fixed Ratio Between L and S Enable 0: Disable 1: Enable
0x2E2	GPMODEEN	0x00	RW	Geometric Proportion Mode 0: Disable 1: Enable
0x2E3	NIGHTMODE	0x00	RW	Night Mode 0: Disable 1: Enable
0x2E4	ONLYINSERTFRAMEEN	0x00	RW	Only insert frame when in night mode 0: Disable 1: Enable
0x2E5	BANDINGFILTERENABLE	0x00	RW	Banding filter 0: Disable 1: Enable
0x2E6	BANDINGFILTERSHORT ENABLE	0x00	RW	Short banding filter 0: Disable 1: Enable
0x2E7	BANDINGFILTERVSENABLE	0x00	RW	VS banding filter 0: Disable 1: Enable
0x2E8	LESSTHAN1BANDENABLE	0x01	RW	Less Than One Band Exposure Mode 0: Disable 1: Enable
0x2E9	LESSTHAN1BANDSHORT ENABLE	0x01	RW	Less Than One Band Exposure for Short 0: Disable 1: Enable
0x2EA	LESSTHAN1BANDVSENABLE	0x01	RW	Less Than One Band Exposure for VS 0: Disable 1: Enable
0x2EB	BANDINGFILTERFLAG	0x01	RW	Light Source Type 00: Frequency is zero or very high 01: 60HZ 10: 50HZ
0x2EC	TARGETY_0_0	0x40	RW	Target of Raw Data for Long 0
0x2ED	TARGETY_0_1	0x40	RW	Target of Raw Data for Long 1
0x2EE	TARGETY_0_2	0x40	RW	Target of Raw Data for Long 2

table A-4 AEC registers (sheet 2 of 7)

address	register name	default value	R/W	description
0x2EF	TARGETY_1_0	0x80	RW	Target of Raw Data for Short 0
0x2F0	TARGETY_1_1	0x80	RW	Target of Raw Data for Short 1
0x2F1	TARGETY_1_2	0x80	RW	Target of Raw Data for Short 2
0x2F2	SLOWRANGE_0	0x20	RW	Slow Range for Long
0x2F3	SLOWRANGE_1	0x20	RW	Slow Range for Short
0x2F4	STABLERANGE_0	0x08	RW	Range Becomes Stable from Unstable
0x2F5	STABLERANGE_1	0x18	RW	Range Becomes Unstable from Stable
0x2F6	FASTSTEP_0	0x0C	RW	Fast AEC Adjustment Step for Long
0x2F7	FASTSTEP_1	0x0C	RW	Fast AEC Adjustment Step for Short
0x2F8	SLOWSTEP_0	0x02	RW	Slow AEC Adjustment Step for Long
0x2F9	SLOWSTEP_1	0x08	RW	Slow AEC Adjustment Step for Short
0x2FA	MAXFASTRATIO	0x40	RW	Maximum Fast Adjustment Ratio
0x2FB	MAXSLOWRATIO	0x04	RW	Maximum Slow Adjustment Ratio
0x2FC	LSFIXEDRATIO	0x20	RW	Fixed Ratio Between L and S
0x2FD	NIGHTMODESTEP	0x0C	RW	AEC Adjustment Step in Night Mode
0x304	MAXCAMERAGAIN_0_H	0x02	RW	Bit[1:0]: Max gain for long[9:8]
0x305	MAXCAMERAGAIN_0_L	0x00	RW	Bit[7:0]: Max gain for long[7:0]
0x306	MAXCAMERAGAIN_1_H	0x02	RW	Bit[1:0]: Max gain for short[9:8]
0x307	MAXCAMERAGAIN_1_L	0x00	RW	Bit[7:0]: Max gain for short[7:0]
0x308	MAXCAMERAGAIN_2_H	0x02	RW	Bit[1:0]: Max gain for VS[9:8]
0x309	MAXCAMERAGAIN_2_L	0x00	RW	Bit[7:0]: Max gain for VS[7:0]
0x30A	MINCAMERAGAIN_0_H	0x00	RW	Bit[1:0]: Min gain for long[9:8]
0x30B	MINCAMERAGAIN_0_L	0x20	RW	Bit[7:0]: Min gain for long[7:0]
0x30C	MINCAMERAGAIN_1_H	0x00	RW	Bit[1:0]: Min gain for short[9:8]
0x30D	MINCAMERAGAIN_1_L	0x20	RW	Bit[7:0]: Min gain for short[7:0]
0x30E	MINCAMERAGAIN_2_H	0x00	RW	Bit[1:0]: Min gain for VS[9:8]
0x30F	MINCAMERAGAIN_2_L	0x20	RW	Bit[7:0]: Min gain for VS[7:0]
0x310	MAXEXPOSURE_0_H	0x02	RW	Bit[7:0]: Max exposure for long[15:8]
0x311	MAXEXPOSURE_0_L	0x08	RW	Bit[7:0]: Max exposure for long[7:0]
0x312	MAXEXPOSURE_1_H	0x02	RW	Bit[7:0]: Max exposure for short[15:8]

table A-4 AEC registers (sheet 3 of 7)

address	register name	default value	R/W	description
0x313	MAXEXPOSURE_1_L	0x08	RW	Bit[7:0]: Max exposure for short[7:0]
0x314	MAXEXPOSURE_2_H	0x00	RW	Bit[7:0]: Max exposure for VS[15:8]
0x315	MAXEXPOSURE_2_L	0x78	RW	Bit[7:0]: Max exposure for VS[7:0]
0x316	MINEXPOSURE_0_H	0x00	RW	Bit[7:0]: Min exposure for long[15:8]
0x317	MINEXPOSURE_0_L	0x01	RW	Bit[7:0]: Min exposure for long[7:0]
0x318	MINEXPOSURE_1_H	0x00	RW	Bit[7:0]: Min exposure for short[15:8]
0x319	MINEXPOSURE_1_L	0x01	RW	Bit[7:0]: Min exposure for short[7:0]
0x31A	MINEXPOSURE_2_H	0x00	RW	Bit[7:0]: Min exposure for VS[15:8]
0x31B	MINEXPOSURE_2_L	0x01	RW	Bit[7:0]: Min exposure for VS[7:0]
0x31C	LSSENSITIVITYRATIO_H	0x08	RW	Bit[7:0]: L/S sensitivity ratio[15:8]
0x31D	LSSENSITIVITYRATIO_L	0x00	RW	Bit[7:0]: L/S sensitivity ratio[7:0]
0x31E	SVSFIXEDRATIO_H	0x00	RW	Bit[7:0]: S/VS exposure ratio[15:8]
0x31F	SVSFIXEDRATIO_L	0x80	RW	Bit[7:0]: S/VS exposure ratio[7:0]
0x320	BANDVALUE60HZ_H	0x20	RW	Bit[7:0]: Band filter value for 60Hz[15:8]
0x321	BANDVALUE60HZ_L	0xD0	RW	Bit[7:0]: Band filter value for 60Hz[7:0]
0x322	BANDVALUE50HZ_H	0x27	RW	Bit[7:0]: Band filter value for 50Hz[15:8]
0x323	BANDVALUE50HZ_L	0x60	RW	Bit[7:0]: Band filter value for 50Hz[7:0]
0x324	MAXRATIO_H	0x00	RW	Bit[7:0]: Maximum L/S ratio[15:8]
0x325	MAXRATIO_L	0x20	RW	Bit[7:0]: Maximum L/S ratio[7:0]
0x326	MINRATIO	0x02	RW	Minimum L/S Ratio
0x327	MAXINSERTFRAME	0x03	RW	Maximum Insert Frames in Night Mode
0x328	nVTSWRITEMODE	0x01	RW	VTS Write Mode 0: Write with exposure 1: Write with gain
0x329	AECMANUALMODE	0x07	RW	Bit[2]: Manual AEC A mode selected 0: Disable 1: Enable Bit[1]: Manual AEC B mode selected 0: Disable 1: Enable Bit[0]: Manual AEC C mode selected 0: Disable 1: Enable

table A-4 AEC registers (sheet 4 of 7)

address	register name	default value	R/W	description
0x32A	AECMANUALDONE	0x02	RW	Manual AEC Start Mode 00: Inactive 01: Active 10: Always active
0x32B	ALLOWFRACTALEXP	0x01	RW	Allow Fractal Resolution for VS Exposure
0x32C	AECMANUALEXP_0_0_H	0x00	RW	Bit[7:0]: Manual exposure for 0 for long[15:8]
0x32D	AECMANUALEXP_0_0_L	0x40	RW	Bit[7:0]: Manual exposure for 0 for long[7:0]
0x32E	AECMANUALEXP_0_1_H	0x00	RW	Bit[7:0]: Manual exposure for 1 for long[15:8]
0x32F	AECMANUALEXP_0_1_L	0x40	RW	Bit[7:0]: Manual exposure for 1 for long[7:0]
0x330	AECMANUALEXP_0_2_H	0x00	RW	Bit[7:0]: Manual exposure for 2 for long[15:8]
0x331	AECMANUALEXP_0_2_L	0x40	RW	Bit[7:0]: Manual exposure for 2 for long[7:0]
0x332	AECMANUALEXP_1_0_H	0x00	RW	Bit[7:0]: Manual exposure for 0 for short[15:8]
0x333	AECMANUALEXP_1_0_L	0x10	RW	Bit[7:0]: Manual exposure for 0 for short[7:0]
0x334	ECMANUALEXP_1_1_H	0x00	RW	Bit[7:0]: Manual exposure for 1 for short[15:8]
0x335	AECMANUALEXP_1_1_L	0x10	RW	Bit[7:0]: Manual exposure for 1 for short[7:0]
0x336	AECMANUALEXP_1_2_H	0x00	RW	Bit[7:0]: Manual exposure for 2 for short[15:8]
0x337	AECMANUALEXP_1_2_L	0x10	RW	Bit[7:0]: Manual exposure for 2 for short[7:0]
0x338	AECMANUALEXP_2_0_H	0x00	RW	Bit[7:0]: Manual exposure for 0 for VS[15:8]
0x339	AECMANUALEXP_2_0_L	0x20	RW	Bit[7:0]: Manual exposure for 0 for VS[7:0]
0x33A	AECMANUALEXP_2_1_H	0x00	RW	Bit[7:0]: Manual exposure for 1 for VS[15:8]
0x33B	AECMANUALEXP_2_1_L	0x20	RW	Bit[7:0]: Manual exposure for 1 for VS[7:0]

table A-4 AEC registers (sheet 5 of 7)

address	register name	default value	R/W	description
0x33C	AECMANUALEXP_2_2_H	0x00	RW	Bit[7:0]: Manual exposure for 2 for VS[15:8]
0x33D	AECMANUALEXP_2_2_L	0x20	RW	Bit[7:0]: Manual exposure for 2 for VS[7:0]
0x33E	AECMANUALGAIN_0_0_H	0x00	RW	Bit[7:0]: Manual gain for 0 for long[15:8]
0x33F	AECMANUALGAIN_0_0_L	0x20	RW	Bit[7:0]: Manual gain for 0 for long[7:0]
0x340	AECMANUALGAIN_0_1_H	0x00	RW	Bit[7:0]: Manual gain for 1 for long[15:8]
0x341	AECMANUALGAIN_0_1_L	0x20	RW	Bit[7:0]: Manual gain for 1 for long[7:0]
0x342	AECMANUALGAIN_0_2_H	0x00	RW	Bit[7:0]: Manual gain for 2 for long[15:8]
0x343	AECMANUALGAIN_0_2_L	0x20	RW	Bit[7:0]: Manual gain for 2 for long[7:0]
0x344	AECMANUALGAIN_1_0_H	0x00	RW	Bit[7:0]: Manual gain for 0 for short[15:8]
0x345	AECMANUALGAIN_1_0_L	0x20	RW	Bit[7:0]: Manual gain for 0 for short[7:0]
0x346	AECMANUALGAIN_1_1_H	0x00	RW	Bit[7:0]: Manual gain for 1 for short[15:8]
0x347	AECMANUALGAIN_1_1_L	0x20	RW	Bit[7:0]: Manual gain for 1 for short[7:0]
0x348	AECMANUALGAIN_1_2_H	0x00	RW	Bit[7:0]: Manual gain for 2 for short[15:8]
0x349	AECMANUALGAIN_1_2_L	0x20	RW	Bit[7:0]: Manual gain for 2 for short[7:0]
0x34A	AECMANUALGAIN_2_0_H	0x00	RW	Bit[7:0]: Manual gain for 0 for VS[15:8]
0x34B	AECMANUALGAIN_2_0_L	0x20	RW	Bit[7:0]: Manual gain for 0 for VS[7:0]
0x34C	AECMANUALGAIN_2_1_H	0x00	RW	Bit[7:0]: Manual gain for 1 for VS[15:8]
0x34D	AECMANUALGAIN_2_1_L	0x20	RW	Bit[7:0]: Manual gain for 1 for VS[7:0]
0x34E	AECMANUALGAIN_2_2_H	0x00	RW	Bit[7:0]: Manual gain for 2 for VS[15:8]
0x34F	AECMANUALGAIN_2_2_L	0x20	RW	Bit[7:0]: Manual gain for 2 for VS[7:0]
0x350	MAXTOTALEXP_0_BYTE0	0x00	RW	Bit[7:0]: Maximum total exposure for long[31:24]
0x351	MAXTOTALEXP_0_BYTE1	0x0A	RW	Bit[7:0]: Maximum total exposure for long[23:16]

table A-4 AEC registers (sheet 6 of 7)

address	register name	default value	R/W	description
0x352	MAXTOTALEXP_0_BYTE2	0x28	RW	Bit[7:0]: Maximum total exposure for long[15:8]
0x353	MAXTOTALEXP_0_BYTE3	0x00	RW	Bit[7:0]: Maximum total exposure for long[7:0]
0x354	MAXTOTALEXP_1_BYTE0	0x00	RW	Bit[7:0]: Maximum total exposure for short[31:24]
0x355	MAXTOTALEXP_1_BYTE1	0x04	RW	Bit[7:0]: Maximum total exposure for short[23:16]
0x356	MAXTOTALEXP_1_BYTE2	0x10	RW	Bit[7:0]: Maximum total exposure for short[15:8]
0x357	MAXTOTALEXP_1_BYTE3	0x00	RW	Bit[7:0]: Maximum total exposure for short[7:0]
0x358	MAXTOTALEXP_2_BYTE0	0x00	RW	Bit[7:0]: Maximum total exposure for VS[31:24]
0x359	MAXTOTALEXP_2_BYTE1	0x02	RW	Bit[7:0]: Maximum total exposure for VS[23:16]
0x35A	MAXTOTALEXP_2_BYTE2	0x60	RW	Bit[7:0]: Maximum total exposure for VS[15:8]
0x35B	MAXTOTALEXP_2_BYTE3	0x00	RW	Bit[7:0]: Maximum total exposure for VS[7:0]
0x35C	MINTOTALEXP_0_H	0x00	RW	Bit[7:0]: Minimum total exposure for long[15:8]
0x35D	MINTOTALEXP_0_L	0x20	RW	Bit[7:0]: Minimum total exposure for long[7:0]
0x35E	MINTOTALEXP_1_H	0x00	RW	Bit[7:0]: Minimum total exposure for short[15:8]
0x35F	MINTOTALEXP_1_L	0x20	RW	Bit[7:0]: Minimum total exposure for short[7:0]
0x360	MINTOTALEXP_2_H	0x00	RW	Bit[7:0]: Min total exposure for VS[15:8]
0x361	MINTOTALEXP_2_L	0x20	RW	Bit[7:0]: Min total exposure for VS[7:0]
0x366	nTRAFFICMODEMIN EXPOSURELINES_H		RW	Traffic Mode Minimum Exposure for Short Exposure High Byte
0x367	nTRAFFICMODEMIN EXPOSURELINES_L		RW	Traffic Mode Minimum Exposure for Short Exposure Low Byte
0x369	BLCTRIGTEMPTHRE	0x02	RW	BLC Temperature Difference Trigger Threshold

table A-4 AEC registers (sheet 7 of 7)

address	register name	default value	R/W	description
0x36D	nHTLIMITSTARTT		RW	High Temperature BLC Temperature Start Threshold
0x36E	nHTLIMITENDT		RW	High Temperature BLC Temperature End Threshold
0x370	nHTLIMITSTARTG_H		RW	High Temperature BLC Analog Gain Start Threshold High Byte
0x371	nHTLIMITSTARTG_L		RW	High temperature BLC: Analog Gain Start Threshold Low Byte
0x372	nHTLIMITENDG_H		RW	High temperature BLC Analog Gain End Threshold High Byte
0x373	nHTLIMITENDG_L		RW	High temperature BLC Analog Gain End Threshold Low Byte
0x374	bADDBLCOFFSETENABLE		RW	High Temperature BLC Target Offset Enable
0x375	nHTLIMITBLCTHREL		RW	High Temperature BLC Target Offset: BLC Offset for Long Exposure
0x376	nHTLIMITBLCTHRES		RW	High Temperature BLC Target Offset: BLC Offset for Short Exposure
0x377	nHTLIMITBLCTHREVS		RW	High Temperature BLC Target Offset: BLC Offset for VS Exposure
0x378	bHTLIMITGAMMAENABLE		RW	High Temperature BLC Limit Gamma Function Enable
0x379	nHTLIMITMINGAMMATHRE		RW	High Temperature BLC Limit Minimum Gamma Threshold
0x37A	nHTLIMITMAXGAMMATHRE		RW	High Temperature BLC Limit Maximum Gamma Threshold
0x37C	bREALGAINSHIFTEN			

A.5 LSC

table A-5 LSC registers (sheet 1 of 4)

address	register name	default value	R/W	description
0x3A4	bLENSAUTOGAINENABLE		RW	
0x3A5	bLENSAUTOCTENABLE		RW	

table A-5 LSC registers (sheet 2 of 4)

address	register name	default value	R/W	description
0x3A6	pLENSGAIN[0][0]_H		RW	
0x3A7	pLENSGAIN[0][0]_L		RW	
0x3A8	pLENSGAIN[0][1]_H		RW	
0x3A9	pLeNSGAIN[0][1]_L		RW	
0x3AA	pLENSGAIN[1][0]_H		RW	
0x3AB	pLENSGAIN[1][0]_L		RW	
0x3AC	pLENSGAIN[1][1]_H		RW	
0x3AD	pLENSGAIN[1][1]_L		RW	
0x3AE	pLENSGAIN[2][0]_H		RW	
0x3AF	pLENSGAIN[2][0]_L		RW	
0x3B0	pLENSGAIN[2][1]_H		RW	
0x3B1	pLENSGAIN[2][1]_L		RW	
0x3B2	pLENSCTTHRE [0][0]_H		RW	
0x3B3	pLENSCTTHRE [0][0]_L		RW	
0x3B4	pLENSCTTHRE [0][1]_H		RW	
0x3B5	pLENSCTTHRE [0][1]_L		RW	
0x3B6	pLENSCTTTHRE [0][2]_H		RW	
0x3B7	pLENSCTTTHRE [0][2]_I		RW	
0x3B8	pLENSCTTTHRE [0][3]_H		RW	
0x3B9	pLENSCTTTHRE [0][3]_I		RW	
0x3BA	pLENSCTTTHRE [1][0]_H		RW	
0x3BB	pLENSCTTTHRE [1][0]_L		RW	
0x3BC	pLENSCTTTHRE [1][1]_H		RW	
0x3BD	pLENSCTTTHRE [1][1]_L		RW	
0x3BE	pLENSCTTTHRE [1][2]_H		RW	
0x3BF	pLENSCTTTHRE [1][2]_L		RW	
0x3C0	pLENSCTTTHRE [1][3]_H		RW	
0x3C1	pLENSCTTTHRE [1][3]_L		RW	
0x3C2	pLENSCTTTHRE [2][0]_H		RW	
0x3C3	pLENSCTTTHRE [2][0]_L		RW	

table A-5 LSC registers (sheet 3 of 4)

address	register name	default value	R/W	description
0x3C4	pLENSCTTTHRE [2][1]_H		RW	
0x3C5	pLENSCTTTHRE [2][1]_L		RW	
0x3C6	pLENSCTTTHRE [2][2]_H		RW	
0x3C7	pLENSCTTTHRE [2][2]_L		RW	
0x3C8	pLENSCTTTHRE [2][3]_H		RW	
0x3C9	pLENSCTTTHRE [2][3]_L		RW	
0x3CA	pLENSMAXQ_0_H		RW	
0x3CB	pLENSMAXQ_0_L		RW	
0x3CC	pLENSMAXQ_1_H		RW	
0x3CD	pLENSMAXQ_1_L		RW	
0x3CE	pLENSMAXQ_2_H		RW	
0x3CF	pLENSMAXQ_2_L		RW	
0x3D0	pLENSMINQ_0_H		RW	
0x3D1	pLENSMINQ_0_L		RW	
0x3D2	pLENSMINQ_1_H		RW	
0x3D3	pLENSMINQ_1_L		RW	
0x3D4	pLENSMINQ_2_H		RW	
0x3D5	pLENSMINQ_2_L		RW	
0x3D6	pLENSMANUALCT_0_h		RW	
0x3D7	pLENSMANUALCT_0_l		RW	
0x3D8	pLENSMANUALCT_1_H		RW	
0x3D9	pLENSMANUALCT_1_l		RW	
0x3DA	pLENSMANUALCT_2_H		RW	
0x3DB	pLENSMANUALCT_2_l		RW	
0x3DC	pLENSCPARRAY[0][0][]		RW	
0x3DD~ 0x49B	~		RW	
0x49C	pLENSCPARRAY[0][1][]		RW	
0x49D~ 0x55B	~		RW	
0x55C	pLENSCPARRAY[0][2][]		RW	

table A-5 LSC registers (sheet 4 of 4)

address	register name	default value	R/W	description
0x55D~ 0x61B	~		RW	
0x61C	pLENSCPARRAY[1][0][]		RW	
0x61D~ 0x6DB	~		RW	
0x6DC	pLENSCPARRAY[1][1][]		RW	
0x6DD~ 0x79B	~		RW	
0x79C	pLENSCPARRAY[1][2][]		RW	
0x79D~ 0x85B	~		RW	
0x85C	bMANUALLENCEN		RW	

A.6 high temperature function

table A-6 high temperature function registers (sheet 1 of 2)

address	register name	default value	R/W	description
0x2D1	BLCBASES_H	0x00	RW	Bit[7:0]: HTBLC offset for S[15:8]
0x2D2	BLCBASES_L	0x88	RW	Bit[7:0]: HTBLC offset for S[7:0]
0x2D8	BLCBASEVS_H	0x00	RW	Bit[7:0]: HTBLC offset for VS[15:8]
0x2D9	BLCBASEVS_L	0x88	RW	Bit[7:0]: HTBLC offset for VS[7:0]
0x38C	BLCBASEL_H	0x00	RW	Bit[7:0]: HTBLC offset for L[15:8]
0x38D	BLCBASEL_L	0x80	RW	Bit[7:0]: HTBLC offset for L[7:0]
0x86F	HTLIMITHDR_ENDTEMP	0x64	RW	Bit[7:0]: High temperature end
0x870	HTLIMITHDR_STARTG HIGH	0x00	RW	Bit[7:0]: High temperature HDR gain start value[15:8]
0x871	HTLIMITHDR_STARTG LOW	0x20	RW	Bit[7:0]: High temperature HDR gain start value[7:0]
0x872	HTLIMITHDR_ENDG HIGH	0x01	RW	Bit[7:0]: High temperature HDR gain end value[15:8]

table A-6 high temperature function registers (sheet 2 of 2)

address	register name	default value	R/W	description
0x873	HTLIMITHDR_ENDG LOW	0x00	RW	Bit[7:0]: High temperature HDR gain end value[7:0]
0x876	HTHDRGAINTARGET	0x03	RW	Bit[7:0]: High temperature HDR gain target value[15:8]
0x877	HTHDRGAINTARGET	0x00	RW	Bit[7:0]: High temperature HDR gain target value[7:0]
0x89C	HTLIMITSATURATIONEN	0x01	RW	Bit[7:1]: Reserved Bit[0]: HTLimitSaturationEn
0x89D	SATURATION_ORIGHIGH	0x38	RW	Bit[7:0]: Normal CMX maximum factor
0x89E	SATURATION_ORIGLOW	0x18	RW	Bit[7:0]: Normal CMX minimum factor
0x89F	SATURATION_TARGETHIGH	0x18	RW	Bit[7:0]: High temperature CMX maximum factor
0x8A0	SATURATION_TARGETLOW	0x10	RW	Bit[7:0]: High temperature CMX minimum factor
0x8A2	HTLIMITLENCEN	0x01	RW	Bit[7:1]: Reserved Bit[0]: bHTLimitLencEn
0x8A3	LENC_ORIGHIGH	0x38	RW	Bit[7:0]: Normal lens maximum factor
0x8A4	LENC_ORIGLOW	0x28	RW	Bit[7:0]: Normal lens minimum factor
0x8A5	LENC_TARGETHIGH	0x00	RW	Bit[7:0]: High temperature maximum factor
0x8A6	LENC_TARGETLOW	0x00	RW	Bit[7:0]: High temperature minimum factor

A.7 CRC16 calculation

```
static const unsigned short crc16tab[256]= {

    0x0000,0x1021,0x2042,0x3063,0x4084,0x50a5,0x60c6,0x70e7,
    0x8108,0x9129,0xa14a,0xb16b,0xc18c,0xd1ad,0xe1ce,0xf1ef,
    0x1231,0x0210,0x3273,0x2252,0x52b5,0x4294,0x72f7,0x62d6,
    0x9339,0x8318,0xb37b,0xa35a,0xd3bd,0xc39c,0xf3ff,0xe3de,
    0x2462,0x3443,0x0420,0x1401,0x64e6,0x74c7,0x44a4,0x5485,
    0xa56a,0xb54b,0x8528,0x9509,0xe5ee,0xf5cf,0xc5ac,0xd58d,
    0x3653,0x2672,0x1611,0x0630,0x76d7,0x66f6,0x5695,0x46b4,
    0xb75b,0xa77a,0x9719,0x8738,0xf7df,0xe7fe,0xd79d,0xc7bc,
    0x48c4,0x58e5,0x6886,0x78a7,0x0840,0x1861,0x2802,0x3823,
    0xc9cc,0xd9ed,0xe98e,0xf9af,0x8948,0x9969,0xa90a,0xb92b,
```

```
    0x5af5,0x4ad4,0x7ab7,0x6a96,0x1a71,0x0a50,0x3a33,0x2a12,
    0xdbfd,0xcbdc,0xfbbf,0xeb9e,0x9b79,0x8b58,0xbb3b,0xab1a,
    0x6ca6,0x7c87,0x4ce4,0x5cc5,0x2c22,0x3c03,0x0c60,0x1c41,
    0xedae,0xfd8f,0xcdec,0xddcd,0xad2a,0xbd0b,0x8d68,0x9d49,
    0x7e97,0x6eb6,0x5ed5,0x4ef4,0x3e13,0x2e32,0x1e51,0x0e70,
    0xff9f,0xefbe,0xdfdd,0xcffc,0xbf1b,0xaf3a,0x9f59,0x8f78,
    0x9188,0x81a9,0xb1ca,0xa1eb,0xd10c,0xc12d,0xf14e,0xe16f,
    0x1080,0x00a1,0x30c2,0x20e3,0x5004,0x4025,0x7046,0x6067,
    0x83b9,0x9398,0xa3fb,0xb3da,0xc33d,0xd31c,0xe37f,0xf35e,
    0x02b1,0x1290,0x22f3,0x32d2,0x4235,0x5214,0x6277,0x7256,
    0xb5ea,0xa5cb,0x95a8,0x8589,0xf56e,0xe54f,0xd52c,0xc50d,
    0x34e2,0x24c3,0x14a0,0x0481,0x7466,0x6447,0x5424,0x4405,
    0xa7db,0xb7fa,0x8799,0x97b8,0xe75f,0xf77e,0xc71d,0xd73c,
    0x26d3,0x36f2,0x0691,0x16b0,0x6657,0x7676,0x4615,0x5634,
    0xd94c,0xc96d,0xf90e,0xe92f,0x99c8,0x89e9,0xb98a,0xa9ab,
    0x5844,0x4865,0x7806,0x6827,0x18c0,0x08e1,0x3882,0x28a3,
    0xcb7d,0xdb5c,0xeb3f,0xfb1e,0x8bf9,0x9bd8,0xabbb,0xbb9a,
    0x4a75,0x5a54,0x6a37,0x7a16,0x0af1,0x1ad0,0x2ab3,0x3a92,
    0xfd2e,0xed0f,0xdd6c,0xcd4d,0xbdaa,0xad8b,0x9de8,0x8dc9,
    0x7c26,0x6c07,0x5c64,0x4c45,0x3ca2,0x2c83,0x1ce0,0x0cc1,
    0xef1f,0xff3e,0xcf5d,0xdf7c,0xaf9b,0xbfba,0x8fd9,0x9ff8,
    0x6e17,0x7e36,0x4e55,0x5e74,0x2e93,0x3eb2,0x0ed1,0x1ef0
};

unsigned short crc16_ccitt(const BYTE *buf, int len)
{
    register int counter;
    register unsigned short crc = 0;
    for( counter = 0; counter < len; counter++)
        crc = (crc<<8) ^ crc16tab[((crc>>8) ^ buf[counter])&0x00FF];
    return crc;
}
```

revision history

version 1.0 05.15.2014

- initial release

version 1.1 01.16.2015

- made major updates throughout entire chapter 2
- in chapter 2, added section 2.1.3 some un-used pins
- made major updates throughout entire chapter 3
- in section 3.1.2, changed title from "boot-up from SPI host" to "boot from SPI host"
- in table 3-5, updated header introduction
- made major updates throughout entire chapter 5
- in chapter 5, added to section 5-3, host control API "HostControl_GetEx" command
- in chapter 5, updated section 5.4.21, temperature read
- in chapter 5, updated section 5.4.23, changing the configure and trigger mode
- in chapter 5, updated section 5.4.31, access sensor/I2C_Dev register
- in chapter 5, updated section 5.4.32, access firmware register
- in appendix A, added section A.6, CRC16 calculation

defining the future of digital imaging™

OmniVision Technologies, Inc.

UNITED STATES

4275 Burton Drive
Santa Clara, CA 95054

tel: +1 408 567 3000

fax: +1 408 567 3001

email: sales@ovt.com

UNITED KINGDOM

Hook, Hampshire +44 1256 744 610

GERMANY

Munich +49 89 63 81 99 88

INDIA

Bangalore +91 80 4112 8966

CHINA

Beijing + 86 10 6580 1690

Shanghai + 86 21 6175 9888

+86 21 5774 9288

Shenzhen + 86 755 8384 9733

JAPAN

Yokohama +81 45 478 7977

Osaka +81 6 4964 2606

KOREA

Seoul + 82 2 3478 2812

SINGAPORE +65 6933 1933

TAIWAN

Taipei +886 2 2657 9800

Hsin-chu +886 3 5656688

website: **www.ovt.com**