Miramar Camera GenICam Interface

Revision History

Date	Rev No.	Description	By
Sep 2023	1	Initial draft	Jeremy Hong
6/10/2024	2	Add register definitions	Bing Wen
6/12/2024	2.01	Add more register definitions	Bing Wen
6/14/2024	2.02	Add user data flash address	Bing Wen
7/16/2024	3.0	Modified Register definitions for firmware 3.0	Bing Wen
8/23/2024	3.1	Register value change for firmware >3.1, FPGA >=1.6.7	Bing Wen
9/30/2024	3.1.1	Remove invalid registers	Bing Wen
2/24/2025	3.3	Register value change for firmware >3.2, FPGA >=1.7.6	Bing Wen
3/21/2025	3.3.1	Change length of 0x27000008 From 2 to 4	Bing Wen
10/20/2025	4.0	Register value change for firmware > 4.0, FPGA >=1.7.38	Bing Wen
2025/12/24	4.5	Register value change for firmware > =4.5, FPGA >=1.7.51	Bing Wen
2026/07/20	4.8	Coretemp, firmware >=4.7 FPGA>=1.7.77	Elliot Ahn

Contents

REVISION HISTORY **1**

NEW IN THIS VERSION **2**

INTRODUCTION **2**

GENICAM PROTOCOL **2**

HARDWARE IMPLEMENTATION **2**

REGISTER DEFINITIONS **2**

EXAMPLE TASKS **4**

READ MCU FIRMWARE VERSION AND FPGA VERSION 4

NUC USING EXTERNAL SHUTTER 4

OUTPUT FORMAT SELECTION 4

SAVE DATA TO FLASH 6
SAVE CURRENT SETTINGS AS DEFAULT 6
ENABLE EXTERNAL TRIGGER ON GPIO 1 6

New in this version

New tone mapping method: CORETEMP: Color Optimzed Radiometrically Encoded Thermal EnhanceMent Protocol

Introduction

Miramar cameras use Genicam protocol. This document will discuss implementation of writing and reading registers from the camera, and registers that control the camera behavior.

GenICam Protocol

For details on how to format writes and reads for GenICam protocol, please refer to the GenICam document “GenICam GenCP Generic Control Protocol Version 1.3”.

Hardware implementation

Miramar Cameras with USB_C connector implemented the GenICam interface on serial ports using USB. The USB port has VID=0x1FC9 and PID=00A3. If the correct serial port is chosen, reading 64 bytes from address 0x04 will return “OBSIDIAN SENSORS INC.”.

Cameras with MIPI/GMSL connector has GenICam protocol implemented through I2C bus. It is documented separately.

The existing communication buffer size is **4096** bytes. Do not send or request more than 4096 bytes of data in each transaction.

Register definitions

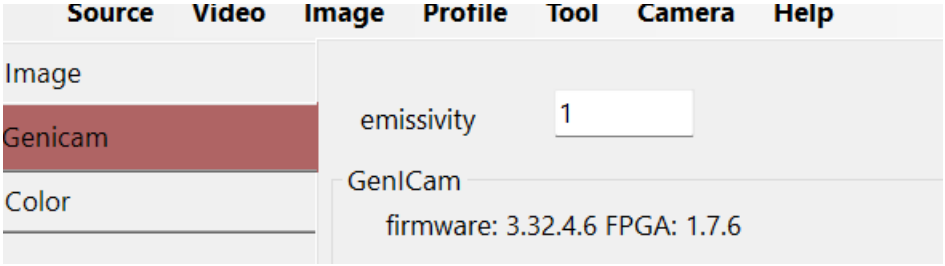
Adress	Length (bytes)	Read/ Write	functions
0x0000_0000	0x024C	R	BootStrap following GenICam CP 1.3
0x020C	64	R	Firmware Version String (ASCII)
0x1000	1	R	Camera board temperature, in Centigrade
0x100E	1	R	shutter temperature = (value[0x100E]+256*value[0x100F])/100, in Centigrade
0x100F	1	R	
0x1010	1	R	FPGA Major version
0x1011	1	R	FPGA Minor version
0x1012	1	R	FPGA Sub Minor version
0x2001	1	R/W	0/1: Video stream disable/enable

0x2005	1	R/W	0: shutter disabled 1: shutter enabled, odd polarity 2: shutter enabled, even polarity
0x200A	1	R/W	Colormap index
0x200D	1	W	Write 1 to save current MCU settings as default setting, which is loaded after every reboot
0x2030	1	W	Write 1 to save current FPGA settings as default setting, which is loaded after every reboot
0x200F	1	R/W	Automatic shutter calibration 0: disable 1:enable
0x2010	1	W	X10 = Maximal number of seconds between shutter calibration
0x2011	1	R/W	Delay between shutter movement and background accumulation, in number of frames
0x202C	1	R/W	LSB of lower limit of radiometric temperature dynamic range(unit: 10mK)
0x202D	1	R/W	MSB of lower limit of radiometric temperature dynamic range
0x202E	1	R/W	LSB of upper limit of radiometric temperature dynamic range(unit: 10mK)
0x202F	1	R/W	MSB of upper limit of radiometric temperature dynamic range
0x2030	1	W	Write 1 to save current FPGA settings as default setting, which is loaded after every reboot
0x2035	1	W	Write 1 to reboot immediately
0x2036	1	W	Perform background accumulation immediately
0x2088	1	W/R	Bit 0: CVBS Bit 1:EXT_triger Enable Bit 2: =0 Ext_trigger on GPIO1, =1 Ext_trigger on spare2
0x24000008	4	R/W	Bit 4:0: number of frames to collect during background accumulation
0x2B000000	4	R/W	CMAP_CONTROL
0x2C000000	4	R/W	TM_CONTROL
0x2C000034	4	R/W	Bit 23-16: Brightness Bit 7-0: Darkness
0x2C00006C	4	R/W	Bit 15-8: Contrast Bit 7-0:Sharpness
0x33000000	4	R/W	CT_CONTROL
0x33000008	4	R/W	CT_CAMP_SCALING
0x609D3000	2,650,111	R/W	0x609D3000 - 0x60c59FFF stores user defined data in flash. Users can use these registers to store data that needs to be preserved after power cycles. - Data is stored in sectors. Each sector is 4096 bytes in size. - The starting address must be dividable by 4096. The address should have format 0Xxxxx x000. User data should be saved when video stream is turned off

Example tasks

Read MCU firmware version and FPGA version

The MCU firmware version and FPGA version will be displayed by ThermalCam App once a camera is connected successfully.



By code, for MCU firmware version, read 64 bytes from 0x020c. For FPGA version read 4 bytes from 0x2000_0014.

Example code in c#:

```
var value = readRegister(0x020c, 64, CloseAfter: false);
_FirmwareVersionString = System.Text.Encoding.Default.GetString(value);

value = readRegister(0x20000014, 4, CloseAfter: false); ;
_FPGAVersionString = value[2].ToString() + "." + value[1].ToString() + "." +
value[0].ToString();
```

NUC using external shutter

Disable shutter, write 0x00 to 0x2005

Put blackbody in front of the lens, then write 0x01 to 0x2036

Output format selection

	CMAP_CONTROL	TM_CONTROL	CT_CONTROL	CT_CAMP_SCALING	
YUYV	0x0000_0001	0x0000_3811	0x0000_0006	0x0004_0000	Tone mapped YUYV format
Raw16	0x0000_0000	0x0000_0004	0x0000_0000	-	Raw 16 Bit format
TY16	0x0000_0000	0x0000_0000	0x0000_0001	-	Temperature 16 Bit output
TMT2	0x0000_0000	0x0000_3811	0x0000_000F	-	Simultaneous tone mapped data and temperature output
CORETEMP	0x0000_0001	0x0000_3811	0x0000_0026	0x0004_0000	Colormap output

In TY16 format, each pixel output is the temperature x 100 in Kelvins represented in 2 bytes. That is, to get the temperature recorded by the pixel in Kelvins, divide the 2-byte output by 100.

In TMT2 format, each pixel occupies two bytes: the first byte is a tone mapped pixel value byte, the second byte is its temperature. By splitting the odd and even bytes, the resulting two images can be directly used for on-screen display and for temperature calculation.

Numpy way of splitting:

```
even_bytes = flat_frame[0::2]
odd_bytes = flat_frame[1::2]
```

Opencv way of splitting:

```
two_channel_img = flat_frame.reshape((480, 640, 2))
even_bytes, odd_bytes = cv2.split(two_channel_img)
```

Registers 0x202C, 0x202D, 0x202E, 0x202F decide how the odd bytes translate into temperature:
A linear mapping used to translate (0,…,255) to (minTemp, maxTemp). minTemp and maxTemp are saved as 16-bit integers, using 10mK as unit.

The following code demonstrates how to map 10°C to 0, 150°C to 255. Please notice that those registers should be written one at a time.

```
minTemp=10   #temperature represent by 0
maxTemp=150  #temperature represent by 255
inTemp = int( 100*(minTemp + 273.15))
maxTemp = int( 100*(maxTemp + 273.15))

minlsb = minTemp  & 0xFF
minmsb = (minTemp >> 8) & 0xFF

maxlsb = maxTemp & 0xFF
maxmsb = (maxTemp>>8) & 0xFF

#write the value to these registers, one byte a time
gc.writeRegister(0x202C, bytearray([minlsb]))
gc.writeRegister(0x202D, bytearray([minmsb]))
gc.writeRegister(0x202E, bytearray([maxlsb]))
gc.writeRegister(0x202F, bytearray([maxmsb]))
```

In CORETEMP mode, the colormap is stretched between two temperatures encoded in the register 0x33000008 (CT_CAMP_SCALING). The following code shows how to keep the colormap in between temperatures T1 and T2 (in Celsius)

```
# Convert temperatures to kelvin times 100
Tlow = (T1 + 273.15) * 100

Thigh = (T2 + 273.15) * 100
a = 65535 / (Thigh - Tlow)
b = Tlow

# Encode these temperatures to correct format into 4-byte register
data = (int(a * 4) << 16) | int(b)
data = data.to_bytes(4, 'little')
gc.writeRegister(0x33000008, bytearray(data))
```

To choose a non-gray colormap, write 0x01 or 0x02 (up to 0x05) into the register 0x200A. 0x00 is the gray colormap.

Save Data to flash

Disable video stream, write 0x00 to 0x2001
Write data to address between 0x609D3000 and 0x60c59FFF in unit of sectors.
Reboot by writing 0x01 to 0x2035

Save current settings as default

Write 0x01 to 0x200D
Write 0x01 to 0x2030

Enable external trigger on GPIO 1

Write 0x02 to 0x2088