中国大恒（集团）有限公司北京图像视觉技术分公司

Python 接口开发说明书

版本：V1.0.8

发布日期：2024-04-03

DAHENG IMAGING 大恒图像

本手册中所提及的其它软硬件产品的商标与名称，都属于相应公司所有。

本手册的版权属于中国大恒（集团）有限公司北京图像视觉技术分公司所有。未得到本公司的正式许可，任何组织或个人均不得以任何手段和形式对本手册内容进行复制或传播。

本手册的内容若有任何修改，恕不另行通知。

网　　站：www.daheng-imaging.com
公 司 总 机：010-82828878
客户服务热线：400-999-7595
销 售 信 箱：sales@daheng-imaging.com
支 持 信 箱：support@daheng-imaging.com

目录

1. 相机工作流程 1

1.1. 整体工作流程 1

1.2. 功能控制流程 2

1.3. 整体代码样例 3

2. 编程指引 5

2.1. 搭建编程环境 5

2.1.1. Linux 5

2.1.2. Windows 5

2.2. 快速上手 6

2.2.1. 引入库 6

2.2.2. 枚举设备 7

2.2.3. 打开关闭设备 8

2.2.4. 采集控制 9

2.2.5. 图像处理 10

2.2.6. 相机控制 13

2.2.7. 导入导出相机配置参数 18

2.2.8. 错误处理 19

3. 附录 20

3.1. 属性参数 20

3.1.1. 设备属性参数 20

3.1.2. 流属性参数 27

3.2. 功能类定义 28

3.2.1. Feature_s(推荐) 28

3.2.2. Feature(不再维护) 34

3.3. 数据类型定义 43

3.3.1. GxDeviceClassList 43

3.3.2. GxAccessStatus 43

3.3.3. GxAccessMode 44

3.3.4. GxPixelFormatEntry 44

3.3.5. GxFrameStatusList 46

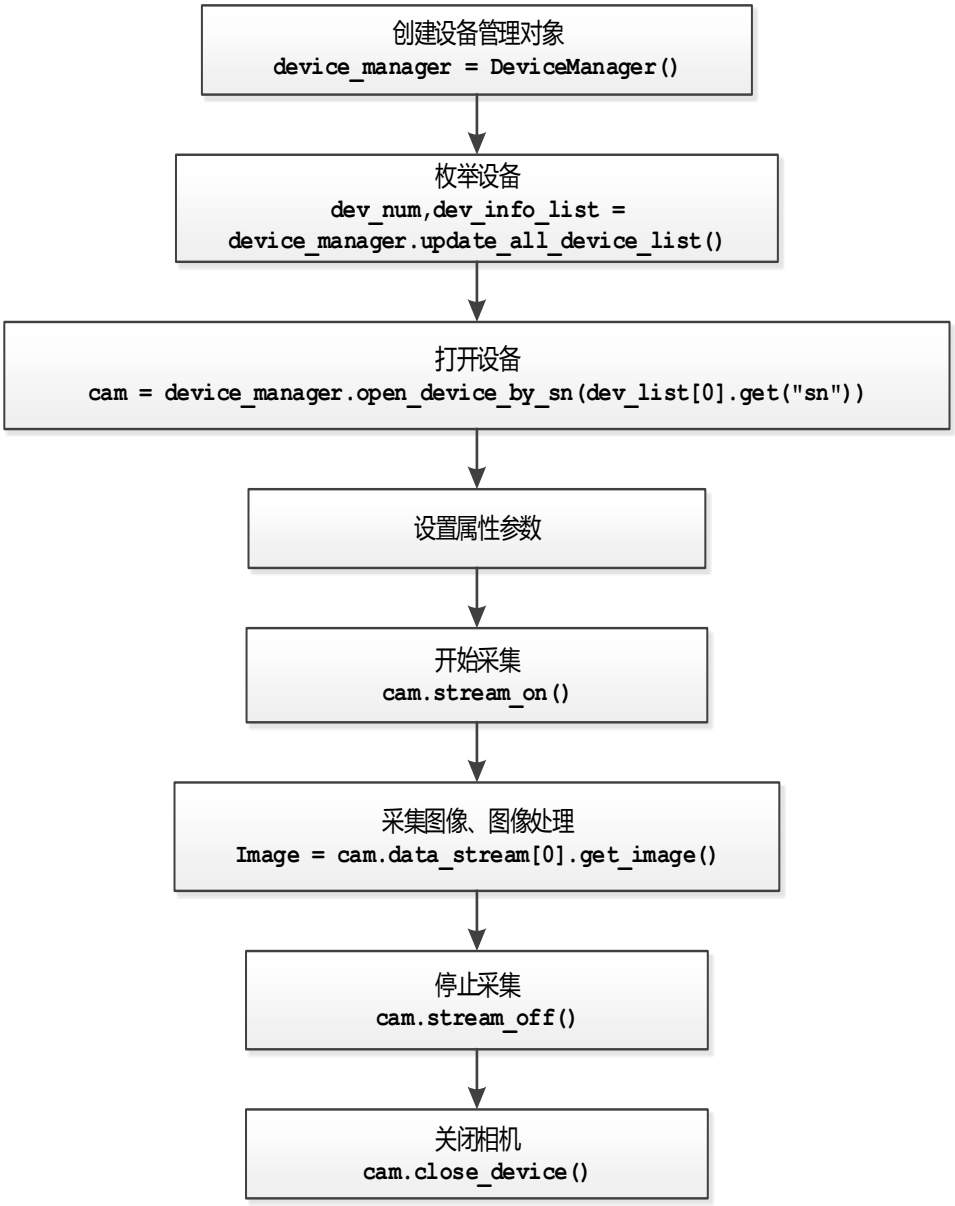
3.3.6. GxDeviceTemperatureSelectorEntry 46

3.3.7. GxPixelSizeEntry 46
3.3.8. GxPixelColorFilterEntry 47
3.3.9. GxAcquisitionModeEntry 47
3.3.10. GxTriggerSourceEntry 47
3.3.11. GxTriggerActivationEntry 47
3.3.12. GxExposureModeEntry 47
3.3.13. GxUserOutputSelectorEntry 48
3.3.14. GxUserOutputModeEntry 48
3.3.15. GxGainSelectorEntry 48
3.3.16. GxBlackLevelSelectEntry 48
3.3.17. GxBalanceRatioSelectorEntry 48
3.3.18. GxAALightEnvironmentEntry 48
3.3.19. GxUserSetEntry 48
3.3.20. GxAWBLampHouseEntry 49
3.3.21. GxUserDataFieldSelectorEntry 49
3.3.22. GxTestPatternEntry 49
3.3.23. GxTriggerSelectorEntry 49
3.3.24. GxLineSelectorEntry 49
3.3.25. GxLineModeEntry 50
3.3.26. GxLineSourceEntry 50
3.3.27. GxEventSelectorEntry 50
3.3.28. GxLutSelectorEntry 51
3.3.29. GxTransferControlModeEntry 51
3.3.30. GxTransferOperationModeEntry 51
3.3.31. GxTestPatternGeneratorSelectorEntry 51
3.3.32. GxChunkSelectorEntry 51
3.3.33. GxBinningHorizontalModeEntry 51
3.3.34. GxBinningVerticalModeEntry 51
3.3.35. GxSensorShutterModeEntry 51
3.3.36. GxAcquisitionStatusSelectorEntry 52
3.3.37. GxExposureTimeModeEntry 52
3.3.38. GxGammaModeEntry 52
3.3.39. GxLightSourcePresetEntry 52
3.3.40. GxColorTransformationModeEntry 52
3.3.41. GxColorTransformationValueSelectorEntry 52
3.3.42. GxAutoEntry 53
3.3.43. GxSwitchEntry 53
3.3.44. GxRegionSendModeEntry 53
3.3.45. GxRegionSelectorEntry 53
3.3.46. GxTimerSelectorEntry 53
3.3.47. GxTimerTriggerSourceEntry 53
3.3.48. GxCounterSelectorEntry 54
3.3.49. GxCounterEventSourceEntry 54
3.3.50. GxCounterResetSourceEntry 54

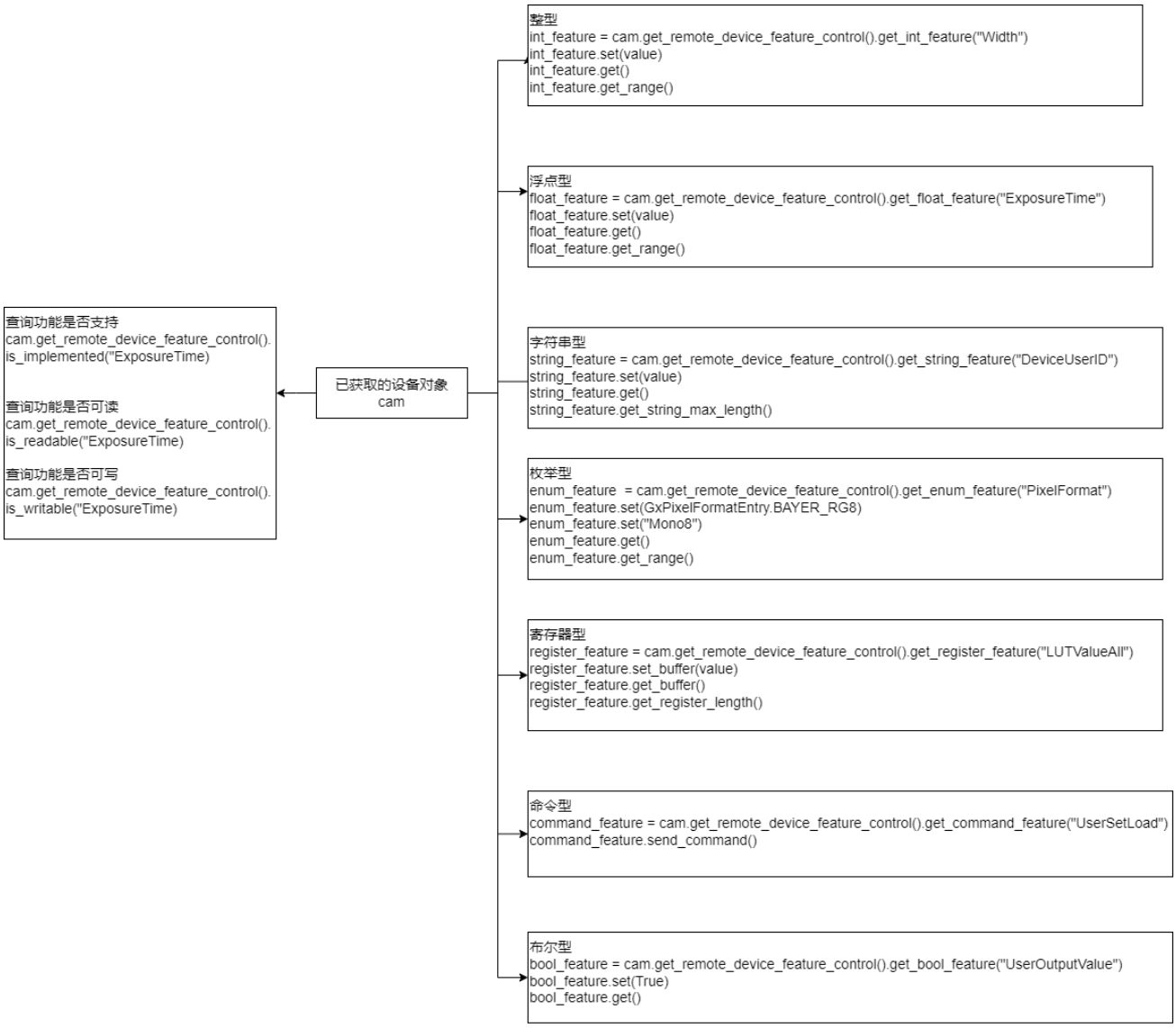
3.3.51. GxCounterResetActivationEntry 54
3.3.52. GxCounterTriggerSourceEntry 54
3.3.53. GxTimerTriggerActivationEntry 55
3.3.54. GxStopAcquisitionModeEntry 55
3.3.55. GxDSStreamBufferHandlingModeEntry 55
3.3.56. GxResetDeviceModeEntry 55
3.3.57. DxBayerConvertType 55
3.3.58. DxValidBit 55
3.3.59. DxImageMirrorMode 56
3.3.60. DxRGBChannelOrder 56
3.3.61. GxTLClassList 56
3.3.62. GxImageInfo 56
3.4. 模块接口定义 56
3.4.1. DeviceManager 56
GxTLClassList 59
3.4.2. Device 63
3.4.3. Interface 70
GxTLClassList 70
3.4.4. DataStream 71
3.4.5. FeatureControl 73
3.4.6. RGBImage(不再维护) 79
3.4.7. RawImage 82
3.4.8. Buffer 89
3.4.9. ImageProcessConfig 90
3.4.10. Utility 98
3.4.11. ImageProcess 100
3.4.12. ImageFormatConvert 102
4. 常见问题解答 107
5. 版本说明 108

1. 相机工作流程

1.1. 整体工作流程



1.2. 功能控制流程



1.3. 整体代码样例

```
    # 用户可自定义调用前缀，样例中使用了 gx
    import gxipy as gx

    # 枚举设备。dev_info_list 是设备信息列表，列表的元素个数为枚举到的设备个数，列表元素是字# 典，其中包含设备索引（index）、ip 信息（ip）等设备信息
device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_all_device_list()
if dev_num == 0:
    sys.exit(1)

# 打开设备
    # 获取设备基本信息列表
strSN = dev_info_list[0].get("sn")
# 通过序列号打开设备
cam = device_manager.open_device_by_sn(strSN)

# 开始采集
cam.stream_on()

    # 获取流通道个数
    # 如果 int_channel_num == 1，设备只有一个流通道，列表 data_stream 元素个数为 1
    # 如果 int_channel_num > 1，设备有多个流通道，列表 data_stream 元素个数大于 1
    # 目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道。
    # int_channel_num = cam.get_stream_channel_num()

# 获取数据
    # num 为采集图片次数
num = 1
for i in range(num):
    # 从第 0 个流通道获取一幅图像
    raw_image = cam.data_stream[0].get_image()
    # 从彩色原始图像获取 RGB 图像
    # 创建格式转换器
    image_convert = device_manager.create_image_format_convert()

    # 设置要转换的目标像素格式
    image_convert.set_dest_format(GxPixelFormatEntry.RGB8)

    # 设置要转换的有效位假设有效位有 12 位，选取高 8 位
    image_convert.set_valid_bits(DxValidBit.BIT4_11)

    # 创建转换后的图像缓存
    rgb_image_buffer_length =
image_convert.image_format_convert_get_buffer_size_for_conversion(raw_image)
    rgb_image_array = (c_ubyte * buffer_out_size)()
    rgb_image = addressof(output_image_array)

    # 转换像素格式成 RGB8
```

```
    image_convert.convert(raw_image, rgb_image, buffer_out_size, False)
    if output_image is None:
       continue

    # 创建 numpy 数组
    numpy_image = numpy.frombuffer(rgb_image_array, dtype=numpy.ubyte,
count=rgb_image_buffer_length). \
              reshape(raw_image.frame_data.height,
raw_image.frame_data.width, 3)
if numpy_image is None:
    continue

    # 显示并保存获得的 RGB 图片
    image = Image.fromarray(numpy_image, 'RGB')
    image.show()
    image.save("image.jpg")

# 停止采集，关闭设备
cam.stream_off()
    cam.close_device()
```

2. 编程指引

2.1. 搭建编程环境

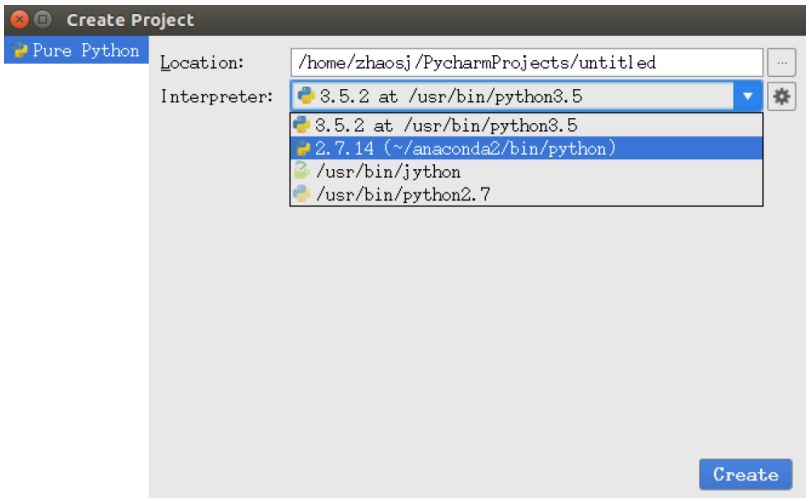
推荐用户优先使用 Python2.7、Python3.5 版本，本接口库已在上述两版本环境中测试通过。

2.1.1. Linux

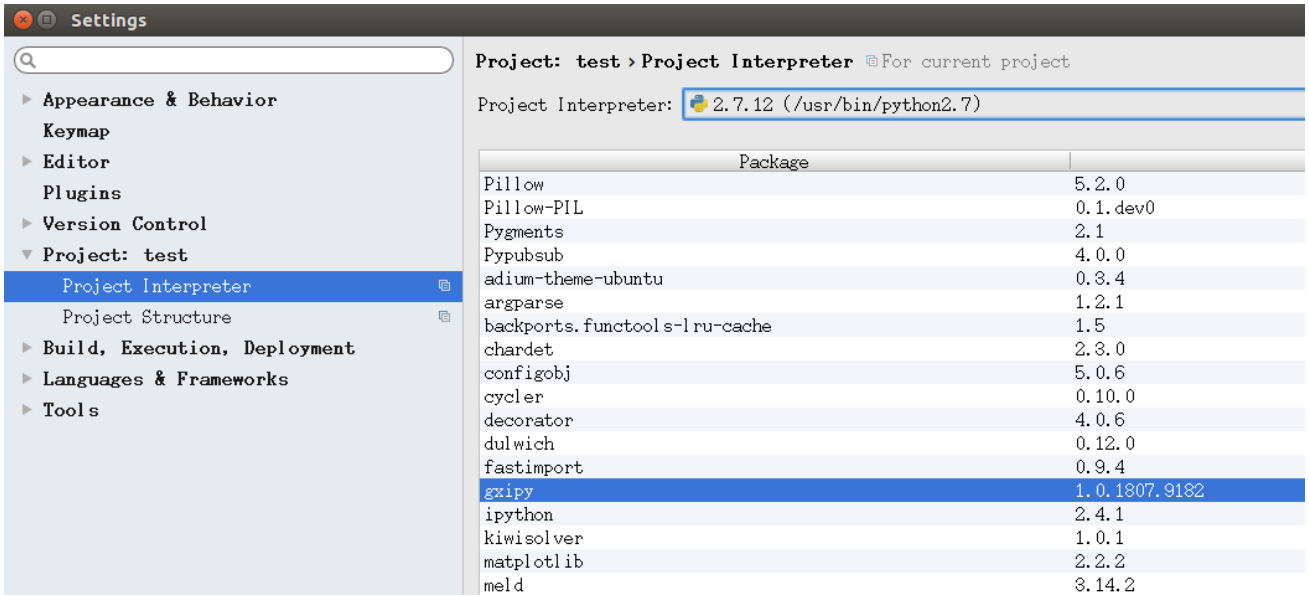
1) 安装 pycharm-community（社区免费版本）

sudo apt-get install pycharm-community

2) 新建 project，选择已安装 gxipy 库的 python 解释器，如果没有可选项，点击右侧的“设置”图标按钮，添加对应版本的 python 解释器。



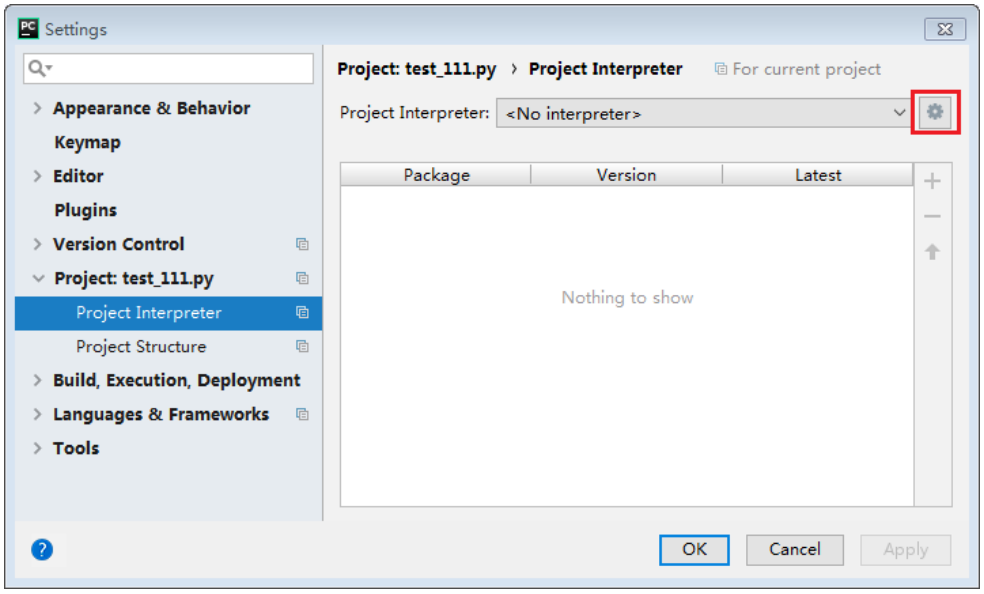
3) File->Settings->Project->Project Interpreter 可以查看 python 解释器已安装的 Package，如图表示 gxipy 库已正常安装，可以导入 gxipy 模块使用。



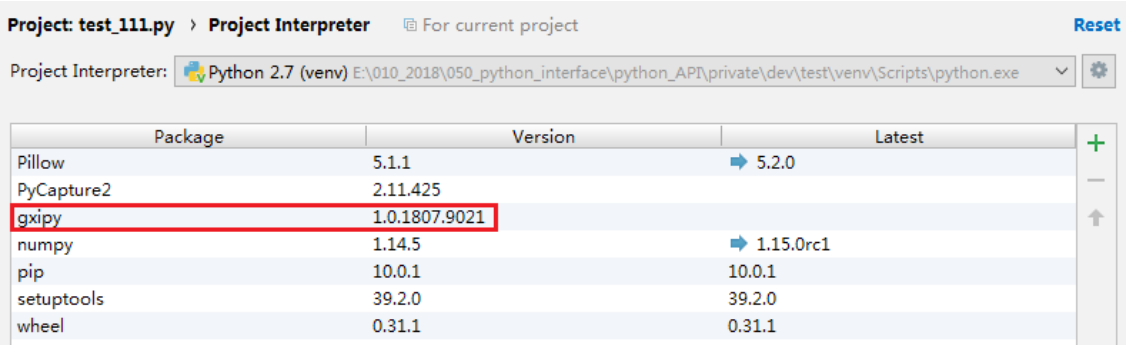
2.1.2. Windows

以 Python2.7、pycharm2018 平台为例，在 windows 7 64 位操作系统环境下演示安装 gxipy 库。初次安装 gxipy 库时：

1) 打开 pycharm2018 的工程配置 File->Settings->Project->Project Interpreter，点击下图红色方框出现的 Add。



2) 选择 New environment，选中 Inherit global site-packages 和 Make available to all projects，点击 OK。这时用户将在新建的 interpreter 中看到 gxipy 库已被成功加载到当前工程环境中。



2.2. 快速上手

2.2.1. 引入库

为使用已安装的库，在程序的开始引入库。在代码中任何使用库中的类、方法、数据类型时，请加自定义的前缀。

代码样例：

```
# 用户可自定义调用前缀，样例中使用了 gx
import gxipy as gx

device_manager = gx.DeviceManager()
```

2.2.2. 枚举设备

用户通过调用 DeviceManager.update_device_list()枚举当前所有可用设备，函数返回值为设备数量 dev_num 和设备信息列表 dev_info_list。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型为字典，字典的键详见下表：

键名称	意义	类型
index	设备索引	整型
vendor_name	厂商名称	字符串
model_name	设备型号	字符串
sn	设备序列号	字符串
display_name	设备显示名称	字符串
device_id	设备标识	字符串
user_id	用户自定义名称	字符串
access_status	权限状态	GxAccessStatus
device_class	设备类	GxDeviceClassList
mac	mac 地址（GEV 相机特有）	字符串
ip	ip 地址（GEV 相机特有）	字符串
subnet_mask	子网掩码（GEV 相机特有）	字符串
gateway	网关（GEV 相机特有）	字符串
nic_mac	nic_mac 地址（GEV 相机特有）	字符串
nic_ip	nic_ip 地址（GEV 相机特有）	字符串
nic_subnet_mask	nic 子网掩码（GEV 相机特有）	字符串
nic_gateway	nic 网关（GEV 相机特有）	字符串
nic_description	nic 描述（GEV 相机特有）	字符串

枚举设备代码片段如下：

```
    # 枚举设备
    device_manager = gx.DeviceManager()
dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)
```

注意：

除上面的枚举接口，DeviceManager 还提供了另两个枚举接口 DeviceManager.update_all_device_list(),DeviceManager.update_device_list_ex()：

1) 对于非千兆网相机来说，这两个枚举接口功能上是一样的；
2) 对千兆网相机来说，库内部使用的枚举机制不一样：

update_all_device_list：使用全网枚举，能够枚举到局域网内的所有千兆网相机

update_device_list：使用子网枚举，只能枚举到局域网内的同一网段的千兆网相机

update_device_list_ex：使用全网枚举可以根据特定类型枚举设备 update_device_list_ex 枚举设备代码片段如下：

```
    # 枚举设备
    device_manager = gx.DeviceManager()
dev_num, dev_info_list =
device_manager.update_device_list_ex(GxTLClassList.TL_TYPE_CXP)
if dev_num == 0:
    sys.exit(1)
```

2.2.3. 打开关闭设备

用户通过调用以下五种不同的方式打开设备：

DeviceManager.open_device_by_sn(self, sn, access_mode=GxAccessMode.CONTROL)
DeviceManager.open_device_by_user_id(self,user_id,access_mode=GxAccessMode.CONTROL)
DeviceManager.open_device_by_index(self,index,access_mode=GxAccessMode.CONTROL)
DeviceManager.open_device_by_ip(ip, access_mode=GxAccessMode.CONTROL)
DeviceManager.open_device_by_mac(mac, access_mode=GxAccessMode.CONTROL)

其中：

sn 为设备序列号

use_id 为用户自定义名称

index 为设备索引（1,2,3...）

mac 为设备 mac 地址（非千兆网相机不可用）

ip 为设备 ip 地址（非千兆网相机不可用）

注意：

最后两个函数只针对千兆网相机使用。

用户可以调用 Device 提供的 Device. close_device()接口来关闭设备，释放所有设备资源。

代码样例：

```
    # 用户可自定义调用前缀，样例中使用了 gx
    import gxipy as gx

    # 枚举设备。dev_info_list 是设备信息列表，列表的元素个数为枚举到的设备个数，列表元素是
    # 字典，其中包含设备索引（index）、ip 信息（ip）等设备信息
device_manager = gx.DeviceManager()
    tl_type = GxTLClassList.TL_TYPE_U3V | GxTLClassList.TL_TYPE_USB |
GxTLClassList.TL_TYPE_GEV | GxTLClassList.TL_TYPE_CXP
dev_num, dev_info_list = device_manager.update_device_list_ex(tl_type)
if dev_num == 0:
    sys.exit(1)

# 打开设备

# 方法一
    # 获取设备基本信息列表
str_sn = dev_info_list[0].get("sn")
# 通过序列号打开设备
cam = device_manager.open_device_by_sn(str_sn)
```

```
    # 方法二
# 通过用户 ID 打开设备
# str_user_id = dev_info_list[0].get("user_id")
# cam = device_manager.open_device_by_user_id(str_user_id)
# 方法三
# 通过索引打开设备
# str_index = dev_info_list[0].get("index")
    # cam = device_manager.open_device_by_index(str_index)

# 下面为只针对千兆网相机使用的打开方式

    # 方法四
# 通过 ip 地址打开设备
# str_ip= dev_info_list[0].get("ip")
# cam = device_manager.open_device_by_ip(str_ip)

# 方法五
# 通过 mac 地址打开设备
# str_mac = dev_info_list[0].get("mac")
# cam = device_manager.open_device_by_mac(str_mac)

    # 关闭设备
    cam.close_device()
```

2.2.4. 采集控制

在设备正常开启并设置好相机采集参数后，用户可调用 Device.stream_on()和 Device.stream_off()执行开停采：

与采集相关的接口和控制都由 DataStream 提供，可通过设置循环次数控制采集图像次数。通过 Device.get_stream_channel_num()接口获得设备的流通道个数。获取的图像使用 RawImage.get_status()判断是否为残帧。代码如下：

- **get_image 方式**

```
    # 开始采集
cam.stream_on()

    # 获取流通道个数
    # 如果 int_channel_num == 1，设备只有一个流通道，列表 data_stream 元素个数为 1
    # 如果 int_channel_num > 1，设备有多个流通道，列表 data_stream 元素个数大于 1
    # 目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道
    # int_channel_num = cam.get_stream_channel_num()
    # 获取数据
    # num 为采集图片次数
num = 1
for i in range(num):
    # 打开第 0 通道数据流
    raw_image = cam.data_stream[0].get_image()
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
        print("incomplete frame")
```

```
    # 停止采集
    cam.stream_off()
```

◆ **回调方式**

```
    # 定义采集回调函数
    def capture_callback(raw_image):
    if raw_image.get_status() == gx.GxFrameStatusList.INCOMPLETE:
       print("incomplete frame")

    # 注册回调
    cam.data_stream[0].register_capture_callback(capture_callback)

    # 开始采集
cam.stream_on()

    # 等待一段时间，这段时间会自动调用采集回调函数
    time.sleep(1)

    # 停止采集
    cam.stream_off()

    # 注销回调
    cam.data_stream[0].unregister_capture_callback()
```

2.2.5. 图像处理

◆ **图像格式转换**

图像格式转换的对象是采集图像 DataStream.get_image()得到的 raw_image。

功能描述：

支持任意像素格式的转换。详见的 ImageFormatConvert.convert 接口。

代码样例：

```
    # 创建设备管理器
    device_manager = gx.DeviceManager()

    # 枚举设备
    dev_num, dev_info_list = device_manager.update_device_list()
if dev_num == 0:
    sys.exit(1)

    #打开第一台设备
    cam = self.device_manager.open_device_by_index(1)

    # 开采
    cam.stream_on()

    # 获取 raw 图，假设其像素格式为 BayerRG12
    raw_image = cam.data_stream[0].get_image()

    # 停采
    cam.stream_off()
```

```
    # 创建格式转换器
    image_convert = device_manager.create_image_format_convert()

    # 设置要转换的目标像素格式
    image_convert.set_dest_format(GxPixelFormatEntry.RGB8)

    # 设置要转换的有效位
    image_convert.set_valid_bits(DxValidBit.BIT4_11)

    # 创建转换后的图像缓存
    buffer_out_size =
image_convert.get_buffer_size_for_conversion(raw_image)
    output_image_array = (c_ubyte * buffer_out_size)()
    output_image = addressof(output_image_array)

    # 转换像素格式成 RGB8
    image convert.convert(raw image, output image, buffer out size, False)
    if output_image is None:
       continue

    # 创建 numpy 数组
    numpy_image = numpy.frombuffer(rgb_image_array, dtype=numpy.ubyte,
count=rgb_image_buffer_length). \
                reshape(raw_image.frame_data.height,
raw_image.frame_data.width, 3)
if numpy_image is None:
    continue
    # 之后，用户可根据获取的 numpy_array 显示、保存图像
```

◆

◆ **图像质量提升**

本接口库还提供了软件端的图像质量提升接口，用户可以有选择的进行颜色校正、对比度、Gamma 等图像质量提升的操作。用户可调用 ImageProcess 的接口 ImageProcess.image_improvement()实现功能。一个简单的使用范例如下：

```
    # 假设对像素格式为 RGB8 的图像做图像质量提升
    # 创建设备管理器
    device_manager = gx.DeviceManager()
    # 打开第一台设备
    cam = device_manager.open_device_by_index(1)

    # 创建图像质量提升处理对象
    image_process = device_manager.create_image_process()

    # 创建图像质量提升参数配置对象
    image_config = cam.create_image_process_config()

    # 创建属性控制器
    remote_device_feature = cam.get_remote_device_feature_control()

    #获取对比度、Gamma 参数
```

```
    if remote_device_feature.is_readable("GammaParam"):
        gamma_param =
remote_device_feature.get_float_feature("GammaParam").get()
    else:
        gamma_param = 0.1

    if remote_device_feature.is_readable("ContrastParam"):
        contrast_param =
remote_device_feature.get_int_feature("ContrastParam").get()
    else:
        contrast_param = 0

    #设置对比度、Gamma 等参数
    image_config.set_contrast_param(gamma_param)
    image_config.set_gamma_param(contrast_param)

    # 创建图像质量提升后存放的 Buffer 缓存
    output_image_array = (c_ubyte * raw_image.frame_data.image_size)()
    output_image = addressof(output_image_array)

    # 采集获取图像 raw_image，调用图像处理接口实现质量提升
    image_process.image_improvement(raw_image, output_image, image_config)
```

1) 颜色校正

设备属性参数名称：ColorCorrectionParam

名词解释：提高相机色彩还原度，使图像更加接近人眼视觉感受。

2) 对比度调节

Contrast 值：整型，范围[-50, 100]，缺省值为 0

设备属性参数名称：ContrastParam

名词解释：图像明亮部分与黑暗部分的亮度比称为对比度，又叫反差。对比度高或者反差大的图像，其中被摄景物的轮廓较清楚，图像也较清晰；反之，对比度低的图像轮廓不清，图像也不太清晰。

3) Gamma 调节

Gamma 值：整型或浮点型，范围[0.1, 10.0]，缺省值为 1

设备属性参数名称： GammaParam

名词解释：Gamma 调节是为了让显示器的输出尽量接近输入。

◆ **图像显示和保存**

调用 PIL(Python Imaging Library)的接口 Image.fromarray()，将 numpy 数组转换成 Image 图像，显示并保存。代码如下：

1) 黑白相机

```
    # 显示并保存获得的黑白图片
image = Image.fromarray(numpy_image, 'L')
image.show()
    image.save("acquisition_mono_image.jpg")
```

2) 彩色相机

```
    # 显示并保存获得的彩色图片
image = Image.fromarray(numpy_image, 'RGB')
image.show()
    image.save("acquisition_RGB_image.jpg")
```

2.2.6. 相机控制

◆ **属性参数访问类型**

属性参数的访问分三种类型：是否实现、是否可读、是否可写。接口设计如下：

FeatureControl.is_implement(feature_name)	此功能是否支持
FeatureControl.is_readable(feature_name)	此功能是否可读
FeatureControl.is_writable(feature_name)	此功能是否可写

建议用户在操作属性参数前，先查询属性的访问类型。

代码样例：

```
    # 查询是否支持
    remote_device_feature = cam.get_remote_device_feature_control()
is_implemented = remote_device_feature.is_implemented("PixelFormat")
if is_implemented == True:
    # 查询是否可写
    is_writable = remote_device_feature.is_writable("PixelFormat")
    if is_writable == True:
    # 设置像素格式
    remote_device_feature.get_enum_feature("PixelFormat").set("Mono8")
    # 查询是否可读
    is_readable = remote_device_feature.is_readable("PixelFormat")
    if is_readable == True:
    # 打印像素格式
    value, str_value =
remote_device_feature.get_enum_feature("PixelFormat").get()
    print(str_value)
```

◆ **属性控制种类**

属性控制分如下几种。

1） Device.get_local_device_feature_control(): 获取本地备属性控制器

2） Device.get_remote_device_feature_control(): 获取远端设备属性控制器

3） DeviceManagetr.get_interface().get_feature_control(): 获取 interface 属性控制器

4） Device.get_stream().get_feature_control(): 获取流属性控制器

通过 GalaxyView 演示程序打开设备，见右侧属性控制树，在设备属性页中分三个区间，上面区间对应功能是 Device.get_remote_device_feature_control()返回的属性集合（图 2-4），中间区间对应功能是 Device.get_local_device_feature_control()返回属性集合（图 2-4），下面区间对应功能是 Device.get_stream().get_feature_controll()返回的属性集合（图 2-4）。

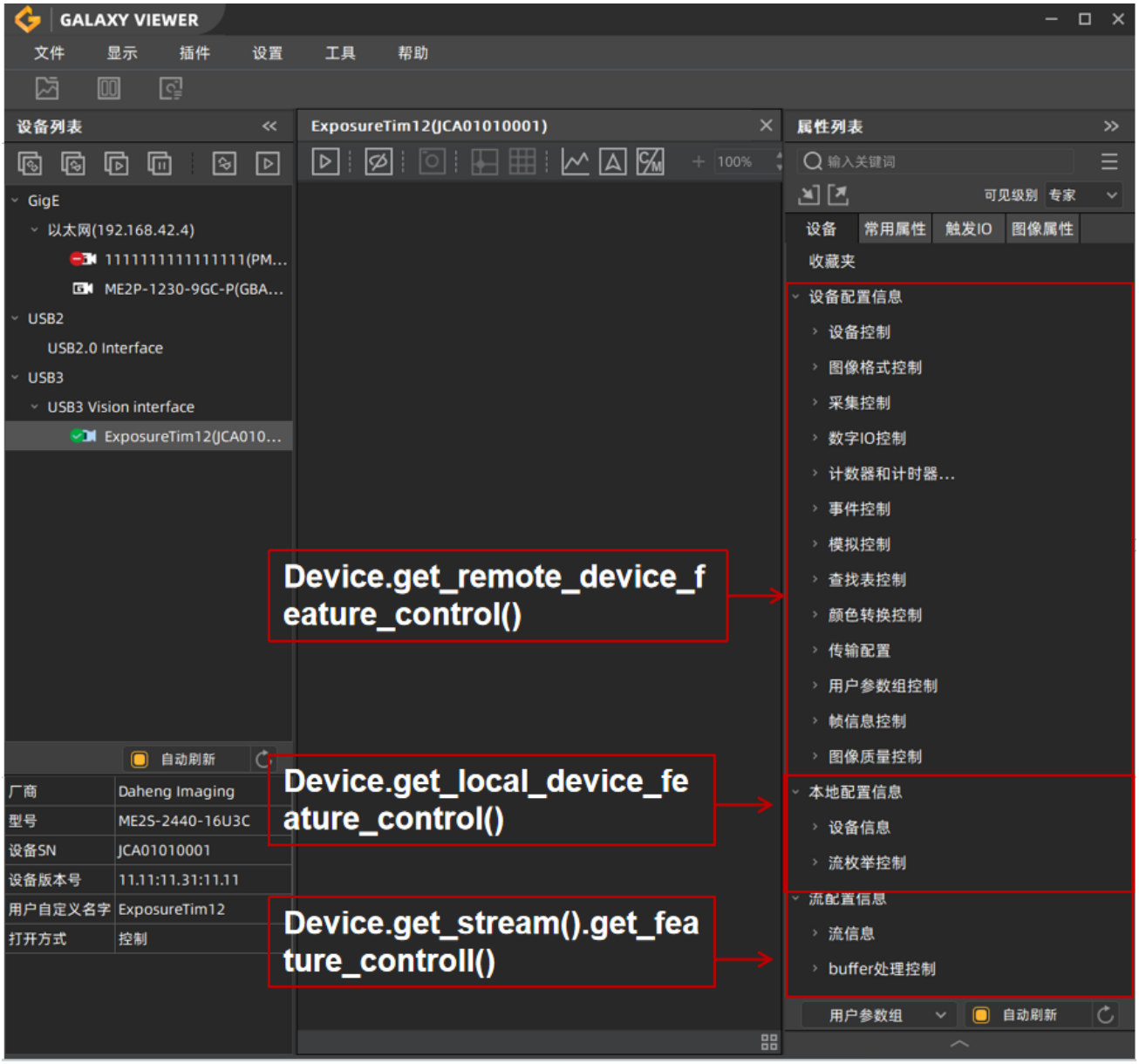


图 2-1 远端、本地和流属性集合

通过 GalaxyView 演示程序点击 Interface 节点，见右侧属性控制树，在属性页对应功能是 DeviceManagetr.get_interface().get_feature_control()返回的属性集合（图 2-5）。

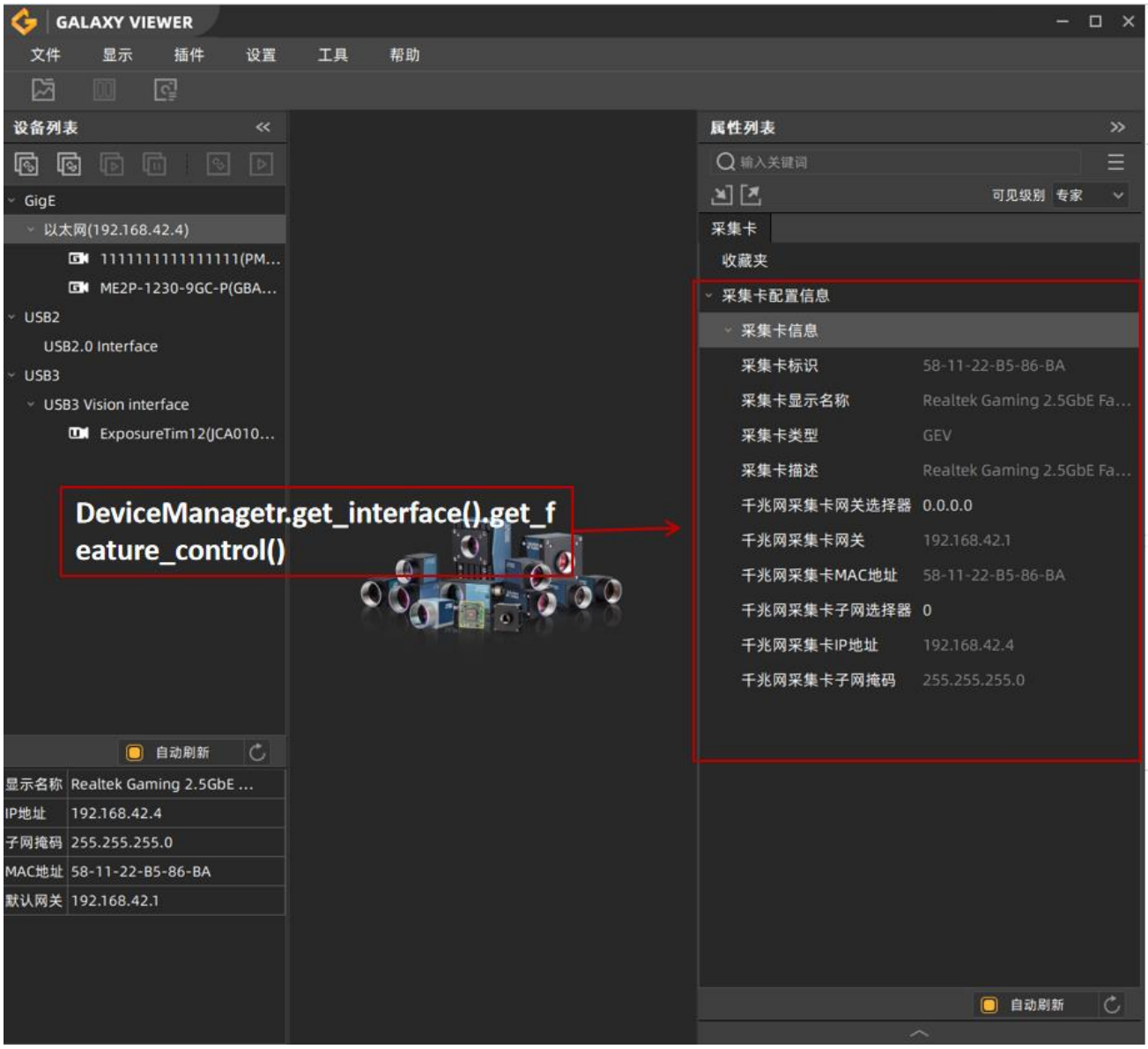


图 2-5 Interface 属性集合

- **整型**

相关接口：

IntFeature_s.set(int_value)　　//设置

IntFeature_s.get()　　//读取

IntFeature_s.get_range()　　//获取最小值、最大值、步长

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取图像宽度属性对象
    width_feature = remote_device_feature.get_int_feature("Width")

    # 获取图像宽度可设置范围
```

```
    width_range = width_feature.get_range()

    # 设置当前图像宽度为范围内任意值
    Width_feature.set(800)

    # 获取当前图像宽度
    int_Width_value = width_feature.get()
```

- **浮点型**

相关接口：

FloatFeature_s.set(float_value)　　//设置

FloatFeature_s.get()　　//读取

FloatFeature_s.get_range()　　//获取最小值、最大值、步长、单位、单位是否有效

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取曝光时间属性对象
    exposure_time_feature =
remote_device_feature.get_float_feature("ExposureTime")

    # 获取曝光值可设置范围和最大值
    float_range = rexposure_time_feature.get_range()
    float_max = float_range["max"]
    # 设置当前曝光值范围内任意值
    exposure_time_feature.set(10.0)
    # 获取当前曝光值
    float_exposure_value = exposure_time_feature.get()
```

- **枚举型**

相关接口：

EnumFeature_s.set(enum_value)　　//设置

EnumFeature_s.get()　　//读取

EnumFeature_s.get_range()　　//获取字典

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取像素格式属性对象
    pixel_format_feature =
remote_device_feature.get_enum_feature("PixelFormat")

    # 获取枚举值可设置范围。
    enum_range = pixel_format_feature.get_range()

# 设置当前枚举值
```

```
    pixel_format_feature.set("Mono8")

# 打印当前枚举值
enum_PixelForamt_value, enum_PixelFormat_key = pixel_format_feature.get()
```

- **布尔型**

相关接口：

BoolFeature_s.set(bool_value) //设置

BoolFeature_s.get() //读取

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取引脚电平反转属性对象
    line_inverter_feature =
remote_device_feature.get_bool_feature("LineInverter")

    # 设置当前布尔值
    line_inverter_feature.set(True)

    # 读取布尔值
    bool_LineInverter_value = line_inverter_feature.get()
```

- **字符串型**

相关接口：

StringFeature_s.set(string_value) //设置

StringFeature_s.get() //读取

StringFeature_s.get_string_max_length() //获取字符串型属性参数的最大长度值

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取用户自定义名称属性对象
    device_user_id_feature =
remote_device_feature.get_string_feature("DeviceUserID")


    # 读取最长字符串长度
string_max_length = device_user_id_feature.get_string_max_length()
    # 读取当前字符串值
current_string = device_user_id_feature.get()

    # 设置字符串值
device_user_id_feature.set("MyUserID")
```

- **Buffer 型**

相关接口：

RegisterFeature_s.set_buffer(buf) //设置

RegisterFeature_s.get_buffer() //读取

RegisterFeature_s.get_register_length() //获取 Buffer 型属性参数的长度

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取寄存器属性对象
    data_field_value_all_feature =
remote_device_feature.get_register_feature("DataFieldValueAll")

    # 读取缓冲数据长度
    buffer_length = data_field_value_all_feature.get_register_length()

    # 设置缓冲数据
    string_set = "abcd"
    set_buffer = gx.Buffer.from_string(string_set_1.encode('utf-8'))
    data_field_value_all_feature.set_buffer(string_set)

# 读取缓冲数据
buffer_data = data_field_value_all_feature.get_buffer()
    print("buffer_data: %s" % (buffer_data.get_data().decode())
```

- **Command 型**

相关接口：

CommandFeature_s.send_command //发送命令

代码样例：

```
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 获取软触发命令属性对象
    trigger_soft_ware_feature =
remote_device_feature.get_register_feature("TriggerSoftware")

    # 发送命令：软触发
trigger_soft_ware_feature.send_command()
```

不同类型的设备具备的属性功能也略有差别。

在附录【属性参数】中可获取相机所有的属性参数。

2.2.7. 导入导出相机配置参数

在接口库中有供用户调用的导入导出设备配置文件的接口代码样例：

```
    # 导入设备远端属性控制器配置参数文件
    # 获取远端属性控制器
remote_device_feature = cam.get_remote_device_feature_control()
    remote_device_feature.feature_load("import_config_file.txt")
    # 导出设备远端属性控制器配置参数文件
```

```
    remote_device_feature.feature_save("export_config_file.txt")
```

2.2.8. 错误处理

当调用接口函数内部出现异常时，错误处理机制会检测并抛出不同类型异常，异常类型均继承自Exception。

一个典型的错误处理代码样例：

```
    try:
# 调用接口函数时函数内部抛异常
dev_num, dev_info_list = device_manager.updata_device_list()
    except Exception as exception:
    print("打印错误信息：%s" % exception)
    exit(1)
```

用户也可通过判断捕获的具体错误类型，进行分类处理：

```
    if isinstance(exception, OutOfRange):
    print("OutOfRange: %s" % exception)
elif isinstance(exception, OffLine):
    print("OffLine: %s" % exception)
else:
    print("Other Error Type %s" % exception)
```

异常类型：

异常类型	意义
UnexpectedError	未预测
NotFoundTL	没找到 TL
NotFoundDevice	未找到设备
OffLine	掉线
InvalidParameter	参数无效
InvalidHandle	句柄无效
InvalidCall	无效回调
InvalidAccess	无效获取
NeedMoreBuffer	Buffer 不足
FeatureTypeError	功能码错误
OutOfRange	超过范围
NotInitApi	未初始化
Timeout	超时
ParameterTypeError	参数类型错误

3. 附录

3.1. 属性参数

3.1.1. 设备属性参数

属性参数	解释	属性类
DeviceInformation Section		
DeviceVendorName	厂商名称	StringFeature
DeviceModelName	设备型号	StringFeature
DeviceFirmwareVersion	设备固件版本	StringFeature
DeviceVersion	设备版本	StringFeature
DeviceSerialNumber	设备序列号	StringFeature
FactorySettingVersion	出厂参数版本	StringFeature
DeviceUserID	用户自定义名称	StringFeature
DeviceLinkSelector	设备链路选择，详见 C 软件开发说明书	IntFeature
DeviceLinkThroughputLimit	设备链路带宽限制	IntFeature
DeviceLinkThroughputLimitMode	设备带宽限制模式，详见 GxSwitchEntry	EnumFeature
DeviceLinkCurrentThroughput	当前设备采集带宽	IntFeature
DeviceReset	设备复位	CommandFeature
TimestampTickFrequency	时间戳时钟频率	IntFeature
TimestampLatch	时间戳锁存	CommandFeature
TimestampReset	重置时间戳	CommandFeature
TimestampLatchReset	重置时间戳锁存	CommandFeature
TimestampLatchValue	时间戳锁存值	IntFeature
DevicePHYVersion	设备网络芯片版本	StringFeature
DeviceTemperatureSelector	设备温度选择，详见 GxDeviceTemperatureSelectorEntry	EnumFeature
DeviceTemperature	设备温度	FloatFeature
ImageFormat Section		
SensorWidth	传感器宽度	IntFeature
SensorHeight	传感器高度	IntFeature
WidthMax	最大宽度	IntFeature
HeightMax	最大高度	IntFeature

OffsetX	水平偏移	IntFeature
OffsetY	垂直偏移	IntFeature
Width	图像宽度	IntFeature
Height	图像高度	IntFeature
BinningHorizontal	水平像素 Binning	IntFeature
BinningVertical	垂直像素 Binning	IntFeature
DecimationHorizontal	水平像素抽样	IntFeature
DecimationVertical	垂直像素抽样	IntFeature
PixelSize	像素位深，详见 GxPixelSizeEntry	EnumFeature
PixelColorFilter	Bayer 格式，详见 GxPixelColorFilterEntry	EnumFeature
PixelFormat	像素格式，详见 GxPixelFormatEntry	EnumFeature
ReverseX	水平翻转	BoolFeature
ReverseY	垂直翻转	BoolFeature
TestPattern	测试图，详见 GxTestPatternEntry	EnumFeature
TestPatternGeneratorSelector	测试图源选择，详见 C 软件开发说明书和 GxTestPatternGeneratorSelectorEntry	EnumFeature
RegionSendMode	ROI 输出模式，详见 GxRegionSendModeEntry	EnumFeature
RegionMode	区域开关，详见 GxSwitchEntry	EnumFeature
RegionSelector	区域选择，详见 C 软件开发说明书和 GxRegionSelectorEntry	EnumFeature
CenterWidth	窗口宽度	IntFeature
CenterHeight	窗口高度	IntFeature
BinningHorizontalMode	水平像素 Binning 模式，详见 GxBinningHorizontalModeEntry	EnumFeature
BinningVerticalMode	垂直像素 Binning 模式，详见 GxBinningVerticalModeEntry	EnumFeature
SensorShutterMode	Sensor 曝光时间模式，详见 GxSensorShutterModeEntry	EnumFeature
TransportLayer Section		
PayloadSize	数据大小	IntFeature
CenterWidth	窗口宽度	IntFeature
CenterHeight	窗口高度	IntFeature
GevCurrentIPConfigurationLLA	LLA 方式配置 IP	BoolFeature

GevCurrentIPConfigurationDHCP	DHCP 方式配置 IP	BoolFeature
GevCurrentIPConfigurationPersistentIP	永久 IP 方式配置 IP	BoolFeature
EstimatedBandwidth	预估带宽	IntFeature
GevHeartbeatTimeout	心跳超时时间	IntFeature
GevSCPSPacketSize	流通道包长	IntFeature
GevSCPD	流通道包间隔	IntFeature
GevLinkSpeed	连接速度	IntFeature
DigitalIO Section		
UserOutputSelector	用户自定义输出选择，详见 C 软件开发说明书和 GxUserOutputSelectorEntry	EnumFeature
UserOutputValue	用户自定义输出值	BoolFeature
UserOutputMode	用户 IO 输出模式，详见 GxUserOutputModeEntry	EnumFeature
StrobeSwitch	闪光灯开关，详见 GxSwitchEntry	EnumFeature
LineSelector	引脚选择，详见 C 软件开发说明书和 GxLineSelectorEntry	EnumFeature
LineMode	引脚方向，详见 GxLineModeEntry	EnumFeature
LineSource	引脚输出源，详见 GxLineSourceEntry	EnumFeature
LineInverter	引脚电平反转	BoolFeature
LineStatus	引脚状态	BoolFeature
LineStatusAll	所有引脚的状态	IntFeature
AnalogControls Section		
GainAuto	自动增益，详见 GxAutoEntry	EnumFeature
GainSelector	增益通道选择，详见 C 软件开发说明书和 GxGainSelectorEntry	EnumFeature
BlackLevelAuto	自动黑电平，详见 GxAutoEntry	EnumFeature
BlackLevelSelector	黑电平通道选择，详见 C 软件开发说明书和 GxBlackLevelSelectEntry	EnumFeature
BalanceWhiteAuto	自动白平衡，详见 GxAutoEntry	EnumFeature
BalanceRatioSelector	白平衡通道选择，详见 C 软件开发说明书和 GxBalanceRatioSelectorEntry	EnumFeature
BalanceRatio	白平衡系数	FloatFeature
DeadPixelCorrect	坏点校正，详见 GxSwitchEntry	EnumFeature
Gain	增益	FloatFeature

BlackLevel	黑电平	FloatFeature
GammaEnable	Gamma 使能	BoolFeature
GammaMode	Gamma 模式，详见 GxGammaModeEntry	EnumFeature
Gamma	Gamma	FloatFeature
DigitalShift	数字移位	IntFeature
LightSourcePreset	环境光源预设，详见 GxLightSourcePresetEntry	EnumFeature
CustomFeature Section		
ADCLevel	AD 转换级别	IntFeature
HBlanking	水平消隐	IntFeature
VBlanking	垂直消隐	IntFeature
UserPassword	用户加密区密码	StringFeature
VerifyPassword	用户加密区校验密码	StringFeature
UserData	用户加密区内容	BufferFeature
ExpectedGrayValue	期望灰度值	IntFeature
AALightEnvironment	自动曝光、自动增益，光照环境类型，详见 GxAALightEnvironmentEntry	EnumFeature
ImageGrayRaiseSwitch	图像亮度拉伸开关，详见 GxSwitchEntry	EnumFeature
AAROIOffsetX	自动调节感兴趣区域 X 坐标	IntFeature
AAROIOffsetY	自动调节感兴趣区域 Y 坐标	IntFeature
AAROIWidth	自动调节感兴趣区域宽度	IntFeature
AAROIHeight	自动调节感兴趣区域高度	IntFeature
AutoGainMin	自动增益最小值	FloatFeature
AutoGainMax	自动增益最大值	FloatFeature
AutoExposureTimeMin	自动曝光最小值	FloatFeature
AutoExposureTimeMax	自动曝光最大值	FloatFeature
ContrastParam	对比度参数	IntFeature
ColorCorrectionParam	颜色校正系数	IntFeature
AWBROIOffsetX	自动白平衡感兴趣区域 X 坐标	IntFeature
AWBROIOffsetY	自动白平衡感兴趣区域 Y 坐标	IntFeature
AWBROIWidth	自动白平衡感兴趣区域宽度	IntFeature
AWBROIHeight	自动白平衡感兴趣区域高度	IntFeature
GammaParam	伽马参数	FloatFeature

AWBLampHouse	自动白平衡光源，详见 GxAWBLampHouseEntry	EnumFeature
SharpnessMode	锐化模式，详见 GxSwitchEntry	EnumFeature
Sharpness	锐度	FloatFeature
FrameInformation	图像帧信息	BufferFeature
DataFieldSelector	用户选择 Flash 数据区域，详见 GxUserDataFieldSelectorEntry	EnumFeature
DataFieldValue	用户区内容	BufferFeature
FlatFieldCorrection	平场校正开关，详见 GxSwitchEntry	EnumFeature
NoiseReductionMode	降噪开关，详见 GxSwitchEntry	EnumFeature
NoiseReduction	降噪	FloatFeature
FFCLoad	获取平场校正参数	BufferFeature
FFCSave	设置平场校正参数	BufferFeature
StaticDefectCorrection	静态坏点校正开关，详见 GxSwitchEntry	EnumFeature
UserSetControl Section		
UserSetLoad	加载参数组	CommandFeature
UserSetSave	保存参数组	CommandFeature
UserSetSelector	参数组选择，详见 C 软件开发说明书和 GxUserSetEntry	EnumFeature
UserSetDefault	启动参数组，详见 GxUserSetEntry	EnumFeature
Event Section		
EventSelector	事件源选择，详见 GxEventSelectorEntry	EnumFeature
EventNotification	事件使能开关，详见 GxSwitchEntry	EnumFeature
EventExposureEnd	曝光结束事件 ID	IntFeature
EventExposureEndTimestamp	曝光结束事件时间戳	IntFeature
EventExposureEndFrameID	曝光结束事件帧 ID	IntFeature
EventBlockDiscard	数据块丢失事件 ID	IntFeature
EventBlockDiscardTimestamp	数据块丢失事件时间戳	IntFeature
EventOverrun	事件队列溢出事件 ID	IntFeature
EventOverrunTimestamp	事件队列溢出事件时间戳	IntFeature
EventFrameStartOvertrigger	触发信号被屏蔽事件 ID	IntFeature
EventFrameStartOvertriggerTimestamp	触发信号被屏蔽事件时间戳	IntFeature
EventBlockNotEmpty	帧存不为空事件 ID	IntFeature
EventBlockNotEmptyTimestamp	帧存不为空事件时间戳	IntFeature

EventInternalError	内部错误事件 ID	IntFeature
EventInternalErrorTimestamp	内部错误事件时间戳	IntFeature
EventFrameBurstStartOvertrigger	多帧触发屏蔽事件 ID	IntFeature
EventFrameBurstStartOvertriggerFrameID	多帧触发屏蔽事件帧 ID	IntFeature
EventFrameBurstStartOvertriggerTimestamp	多帧触发屏蔽事件时间戳	IntFeature
EventFrameStartWait	帧等待事件 ID	IntFeature
EventFrameStartWaitTimestamp	帧等待事件时间戳	IntFeature
EventFrameBurstStartWait	多帧等待事件 ID	IntFeature
EventFrameBurstStartWaitTimestamp	多帧等待事件时间戳	IntFeature
EventBlockDiscardFrameID	数据块丢失事件帧 ID	IntFeature
EventFrameStartOvertriggerFrameID	触发信号被屏蔽事件帧 ID	IntFeature
EventBlockNotEmptyFrameID	帧存不为空事件帧 ID	IntFeature
EventFrameStartWaitFrameID	帧等待事件帧 ID	IntFeature
EventFrameBurstStartWaitFrameID	多帧等待事件帧 ID	IntFeature
LUT Section		
LUTValueAll	查找表内容	BufferFeature
LUTSelector	查找表选择，详见 C 软件开发说明书和 GxLutSelectorEntry	EnumFeature
LUTEnable	查找表使能	BoolFeature
LUTIndex	查找表索引	IntFeature
LUTValue	查找表值	IntFeature
Color Transformation Control		
ColorTransformationMode	颜色转换模式，详见 GxColorTransformationModeEntry	EnumFeature
ColorTransformationEnable	颜色转换使能	BoolFeature
ColorTransformationValueSelector	颜色转换矩阵元素选择，详见 GxColorTransformationValueSelectorEntry	EnumFeature
ColorTransformationValue	颜色转换矩阵元素	FloatFeature
SaturationMode	饱和度模式开关，详见 GxSwitchEntry	EnumFeature
Saturation	饱和度	IntFeature
ChunkData Section		
ChunkModeActive	帧信息使能	BoolFeature
ChunkEnable	单项帧信息使能	BoolFeature
ChunkSelector	帧信息项选择，详见 C 软件开发说明书	EnumFeature

	和 GxChunkSelectorEntry	
Device Feature		
DeviceCommandTimeout	命令超时	IntFeature
DeviceCommandRetryCount	命令重试次数	IntFeature
AcquisitionTrigger Section		
FrameBufferOverwriteActive	帧存覆盖使能	BoolFeature
AcquisitionStart	开始采集	CommandFeature
AcquisitionStop	停止采集	CommandFeature
TriggerSoftware	软触发	CommandFeature
TransferStart	开始传输	CommandFeature
AcquisitionMode	采集模式，详见 GxAcquisitionModeEntry	EnumFeature
TriggerMode	触发模式，详见 GxSwitchEntry	EnumFeature
TriggerActivation	触发极性，详见 GxTriggerActivationEntry	EnumFeature
ExposureAuto	自动曝光，详见	EnumFeature
TriggerSource	触发源，详见 GxTriggerSourceEntry	EnumFeature
ExposureMode	曝光模式，详见 GxExposureModeEntry	EnumFeature
TriggerSelector	触发类型选择，详见 C 软件开发说明书和 GxTriggerSelectorEntry	EnumFeature
TransferControlMode	传输控制模式，详见 GxTransferControlModeEntry	EnumFeature
TransferOperationMode	传输操作模式，详见 GxTransferOperationModeEntry	EnumFeature
AcquisitionFrameRateMode	采集帧率调节模式，详见 GxSwitchEntry	EnumFeature
FixedPatternNoiseCorrectMode	模板噪声校正，详见 GxSwitchEntry	EnumFeature
ExposureTime	曝光时间	FloatFeature
TriggerFilterRaisingEdge	上升沿触发滤波	FloatFeature
TriggerFilterFallingEdge	下降沿触发滤波	FloatFeature
TriggerDelay	触发延迟	FloatFeature
AcquisitionFrameRate	采集帧率	FloatFeature
CurrentAcquisitionFrameRate	当前采集帧率	FloatFeature
TransferBlockCount	传输帧数	IntFeature
TriggerSwitch	外触发开关，详见 GxSwitchEntry	EnumFeature
AcquisitionSpeedLevel	采集速度级别	IntFeature
AcquisitionFrameCount	多帧采集帧数	IntFeature

AcquisitionBurstFrameCount	高速连拍帧数	IntFeature
AcquisitionStatusSelector	采集状态选择,详见 C 软件开发说明书和 GxAcquisitionStatusSelectorEntry	EnumFeature
AcquisitionStatus	采集状态	BoolFeature
ExposureDelay	曝光延迟	FloatFeature
ExposureOverlapTimeMax	交叠曝光时间最大值	FloatFeature
ExposureTimeMode	曝光时间模式，详见 GxExposureTimeModeEntry	EnumFeature
CounterAndTimerControl Section		
TimerSelector	计时器选择，详见 GxTimerSelectorEntry	EnumFeature
TimerDuration	计时器持续时间	FloatFeature
TimerDelay	计时器延迟	FloatFeature
TimerTriggerSource	计时器触发源，详见 GxTimerTriggerSourceEntry	EnumFeature
CounterSelector	计数器选择，详见 GxCounterSelectorEntry	EnumFeature
CounterEventSource	计数器事件触发源，详见 GxCounterEventSourceEntry	EnumFeature
CounterResetSource	计数器复位源，详见 GxCounterResetSourceEntry	EnumFeature
CounterResetActivation	计数器复位信号极性，详见 GxCounterResetActivationEntry	EnumFeature
CounterReset	计数器复位	CommandFeature
CounterTriggerSource	计数器触发源，详见 GxCounterTriggerSourceEntry	EnumFeature
CounterDuration	计数器持续时间	IntFeature
TimerTriggerActivation	计时器触发极性，详见 GxTimerTriggerActivationEntry	EnumFeature
RemoveParameterLimitControl Section		
RemoveParameterLimit	取消参数范围限制开关，详见 GxSwitchEntry	EnumFeature

3.1.2. 流属性参数

属性参数	解释	属性类
StreamAnnouncedBufferCount	声明的 Buffer 个数	IntFeature
StreamDeliveredFrameCount	接收帧个数(包括残帧)	IntFeature
StreamLostFrameCount	buffer 不足导致的丢帧个数	IntFeature

StreamIncompleteFrameCount	接收的残帧个数	IntFeature
StreamDeliveredPacketCount	接收到的包数	IntFeature
StreamResendPacketCount	重传包个数	IntFeature
StreamRescuedPacketCount	重传成功包个数	IntFeature
StreamResendCommandCount	重传命令次数	IntFeature
StreamUnexpectedPacketCount	异常包个数	IntFeature
MaxPacketCountInOneBlock	数据块最大重传包数	IntFeature
MaxPacketCountInOneCommand	一次重传命令最大包含的包数	IntFeature
ResendTimeout	重传超时时间	IntFeature
MaxWaitPacketCount	最大等待包数	IntFeature
ResendMode	重传模式，详见 GxSwitchEntry	EnumFeature
StreamMissingBlockIDCount	BlockID 丢失个数	IntFeature
BlockTimeout	数据块超时时间	IntFeature
MaxNumQueueBuffer	采集队列最大 Buffer 个数	IntFeature
PacketTimeout	包超时时间	IntFeature
StreamTransferSize	传输数据块大小	IntFeature
StreamTransferNumberUrb	传输数据块数量	IntFeature
SocketBufferSize	套接字缓冲区大小	IntFeature
StopAcquisitionMode	停采模式。详见 GxStopAcquisitionModeEntry	EnumFeature
StreamBufferHandlingMode	Buffer 处理模式，详见 GxDSStreamBufferHandlingModeEntry	EnumFeature

3.2. 功能类定义

3.2.1. Feature_s(推荐)

3.2.1.1. Feature_s

Feature_s 类是 IntFeature_s/FloatFeature_s/EnumFeature_s/BoolFeature_s/StringFeature_s/BufferFeature_s/CommandFeature_s 属性类的父类。

已知：通过 DeviceManager 已经打开相机相机获取了 Device 对象，变量名是 cam。

第一步：通过 Device 对象获取对应的 feature_control 对象，比如：obj_feature_control = cam.get_remote_device_feature_control()

第二步：通过 feature_control 对象获取指定属性名称的属性对象:

比如访问一个整型的属性 Width

obj_width = remote_device_feature.get_int_feature("Width")

obj_width.set(2048)
再比如访问一个浮点型的属性 Gain
obj_gain = remote_device_feature.get_float_feature("Gain")
obj_gain.set(1.0)
总之，用户根据属性名称字符串配合对应的属性类型(整形、浮点型、枚举型等等)进行属性的访问

3.2.1.2. IntFeature_s

负责读写、控制相机的整型值功能，继承自 Feature_s 类。

接口列表：

get_range()	获取整型属性参数范围字典
get()	获取当前值
set(int_value)	设置当前值

◆ **接口说明**

➢ **get_range**

声明：
IntFeature_s.get_range()
意义：
获取整型属性参数范围字典。
返回值：
记录整型属性参数范围字典。键包含：min 最小值，max 最大值，step 步长，value 当前值。
异常处理：

3) 如果获取该整型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：
IntFeature_s.get()
意义：
获取当前值。
返回值：
获取功能节点对应的整型值。
异常处理：

4) 如果获取该整型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：
IntFeature_s.set(int_value)
意义：
设置当前值。
形参：
设置的整型数值。
异常处理：

5) 如果设置该整型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.3. FloatFeature_s

负责查看、控制相机的浮点型值功能，继承自 Feature_s 类。

接口列表：

get_range()	获取浮点型属性参数范围字典
get()	获取当前值
set(float_value)	设置当前值

◆ **接口说明**

➢ **get_range**

声明：

FloatFeature_s.get_range()

意义：

获取浮点型属性参数范围字典。

返回值：

记录浮点型属性参数值范围的字典。键包含：min 最小值，max 最大值，inc 步长，unit 单位，inc_is_valid 单位是否有效，cur_value 当前值。

异常处理：

6) 如果获取该浮点型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

FloatFeature_s.get()

意义：

获取当前值

返回值：

获取的浮点型属性参数值。

异常处理：

7) 如果获取浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

FloatFeature_s.set(float_value)

意义：

设置当前值

形参：

[in]float_value 设置的浮点型数值

异常处理：

8) 如果设置该浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.4. EnumFeature_s

负责查看、控制相机的枚举型值功能，继承自 Feature_s 类。

接口列表：

get_range()	获取枚举型属性参数范围字典

get()	获取当前的枚举项值
set(enum_value)	获取当前的枚举项值

◆ **接口说明**

➢ **get_range**

声明：

EnumFeature_s.get_range()

意义：

获取枚举型属性参数范围字典。

返回值：

记录枚举型属性参数范围的字典。

异常处理：

9) 如果获取该枚举型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

EnumFeature_s.get()

意义：

获取当前的枚举项值

返回值：

1) 枚举型属性参数的数值。

2) 枚举型属性参数的字符串。

异常处理：

10) 如果获取枚举型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

EnumFeature_s.set(enum_value)

意义：

设置当前的枚举项值

形参：

[in]enum_value 设置的枚举型数值

注：可以是枚举值，也可以枚举类型对应的字符串值

异常处理：

11) 如果设置该枚举型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.5. BoolFeature_s

负责查看、控制相机的布尔型值功能，继承自 Feature_s 类。

接口列表：

get()	获取当前值
set(bool_value)	设置当前值

◆ **接口说明**

➢ **get**

声明：

BoolFeature_s.get()

意义：

获取当前值

返回值：

获取的布尔值

异常处理：

12) 如果获取布尔型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

BoolFeature_s.set(bool_value)

意义：

设置当前值

形参：

[in]bool_value　设置的布尔型数值

异常处理：

13) 如果设置该布尔型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.6. StringFeature_s

负责查看、控制相机的字符串型值功能，继承自 Feature_s 类。

接口列表：

get_string_max_length()	获取可设置的最大长度，不包含结束符
get()	获取当前值
set(input_string)	设置当前值

◆ **接口说明**

➢ **get_string_max_length**

声明：

StringFeature_s.get_string_max_length()

意义：

获取可设置的最大长度，不包含结束符

返回值：

字符串型属性参数可设置的最大长度

异常处理：

14) 如果获取字符串型属性参数值可设置最大长度不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

StringFeature_s.get()

意义：

获取当前值

返回值：

获取的字符串型属性参数值。

异常处理：

15）如果获取该字符串型属性参数值不成功，则抛出异常，异常类型详见错误处理。

- **set**

声明：

StringFeature_s.set(input_string)

意义：

设置当前值

形参：

[in]input_string　　设置的字符串型数值

异常处理：

16）如果设置该字符串型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.7. RegisterFeature_s

负责查看、控制相机的缓冲功能，继承自 Feature_s 类。

接口列表：

get_register_length()	获取 buffer 的长度。用户读取此长度用来读写 buffer
get_buffer()	获取 buffer 值。输入参数 ptrBuffer 的长度必须等于 get_register_length()接口获取的长度
set_buffer(buf)	设置 buffer 值。输入参数 ptrBuffer 的长度必须等于 get_register_length()接口获取的长度

◆ **接口说明**

- **get_register_length**

声明：

RegisterFeature_s.get_register_length()

意义：

获取 buffer 的长度。用户读取此长度用来读写 buffer。

返回值：

寄存器型属性参数的长度。

异常处理：

17）如果获取 Buffer 型属性参数长度不成功，则抛出异常，异常类型详见错误处理。

- **get_buffer**

声明：

RegisterFeature_s.get_buffer()

意义：

获取 buffer 值。输入参数 ptrBuffer 的长度必须等于 get_register_length()接口获取的长度

返回值：

寄存器型数据。

异常处理：

18) 如果获取该寄存器型属性参数数据不成功，则抛出异常，异常类型详见错误处理。

➢ **set_buffer**

声明：

RegisterFeature_s.set_buffer(buf)

意义：

设置 buffer 值。输入参数 ptrBuffer 的长度必须等于 get_register_length()接口获取的长度

形参：

[in]buffer 设置的缓冲数据[Buffer 类型]

异常处理：

19) 如果设置该寄存器型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.1.8. CommandFeature_s

负责发送命令功能，继承自 Feature_s 类。

接口列表：

send_command()	执行命令

◆ **接口说明**

➢ **send_command**

声明：

CommandFeature_s.send_command()

意义：

执行命令

异常处理：

20) 如果发送命令不成功，则抛出异常，异常类型详见错误处理。

3.2.2. Feature(不再维护)

3.2.2.1. Feature

负责查看各种数据类型功能的基础功能，判断其是否已实现、可读、可写。

Feature 类是

IntFeature/FloatFeatureEnumFeature/BoolFeature/StringFeature/BufferFeature/CommandFeature 属性类的父类。

这种方式的属性控制是通过 Device 对象直接访问。

已知：通过 DeviceManager 已经打开相机相机获取了 Device 对象，变量名是 cam。

比如访问一个整型的属性 Width：

```
cam.Width.set(2048)
```

比如访问一个浮点型属性 Gain：

```
cam.Gain.set(1.0)
```

这种方式的弊端是当 cam 有新增属性的时候，需要升级 python 库以支持新功能，对旧版本 python 库的用户不友好，所以此方式的属性控制不再维护升级新增功能，已有功能对象仍然可以继续使用。

后续推荐用户使用 feature_control 对象通过属性字符串名称获取属性对象访问，这种方式可以不需要升级 python 库就可以兼容新的相机功能。

3.2.2.2. IntFeature

负责查看、控制相机的整型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断整型属性参数是否已实现
is_readable()	判断整型属性参数是否可读
is_writable ()	判断整型属性参数是否可写
get_range()	获取整型属性参数范围字典
get()	读取整型属性参数值
set(int_value)	设置整型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_range**

声明：

IntFeature.get_range()

意义：

获取整型属性参数范围字典

返回值：

记录整型属性参数范围字典。键包含：min 最小值，max 最大值，step 步长

异常处理：

1) 如果该整型属性参数功能未实现，则打印不支持该整型属性参数获取范围的信息，函数返回 None。
2) 如果获取该整型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

IntFeature.get()

意义：

读取整型属性参数值

返回值：

获取的整型值

异常处理：

1) 如果该整型属性参数功能未实现或不可读，则打印该整型属性参数不可读的信息，函数返回 None。
2) 如果获取该整型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：
IntFeature.set(self,int_value)
意义：
设置整型属性参数值
形参：
设置的整型数值
异常处理：

1) 如果输入参数不是整型值，则抛出 ParameterTypeError 异常。
2) 如果该整型属性参数功能未实现或不可写，则打印该整型属性参数不可写的信息，函数返回 None。
3) 如果输入参数不在该整型属性参数的范围内，则打印超过该整型属性参数范围的信息并打印范围，函数返回 None。
4) 如果设置该整型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.3. FloatFeature

负责查看、控制相机的浮点型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断浮点型属性参数是否已实现
is_readable()	判断浮点型属性参数是否可读
is_writable ()	判断浮点型属性参数是否可写
get_range()	获取浮点型属性参数范围字典
get()	读取浮点型属性参数值
set(float_value)	设置浮点型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_range**

声明：
FloatFeature.get_range()
意义：
获取浮点型属性参数范围字典
返回值：

记录浮点型属性参数值范围的字典。键包含：min 最小值，max 最大值，inc 步长，unit 单位，inc_is_valid 单位是否有效

异常处理：

1) 如果该浮点型属性参数功能未实现，则打印不支持该浮点型属性参数获取范围的信息，函数返回 None。
2) 如果获取该浮点型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

3.2.2.3.1. get

声明：

FloatFeature.get()

意义：

读取浮点型属性参数值

返回值：

获取的浮点型属性参数值

异常处理：

1) 如果该浮点型属性参数未实现或不可读，则打印该浮点型属性参数不可读的信息，函数返回 None。
2) 如果获取浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.3.2. set

声明：

FloatFeature.set(float_value)

意义：

设置浮点型属性参数值

形参：

[in]float_value 设置的浮点型数值

异常处理：

1) 如果输入参数不是浮点型值，则抛出 ParameterTypeError 异常。
2) 如果该浮点型属性参数功能未实现或不可写，则打印该浮点型属性参数不可写的信息，函数返回 None。
3) 如果输入参数不在该浮点型属性参数的范围内，则打印超过该浮点型属性参数范围的信息并打印范围，函数返回 None。
4) 如果设置该浮点型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.4. EnumFeature

负责查看、控制相机的枚举型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断枚举型属性参数是否已实现
is_readable()	判断枚举型属性参数是否可读
is_writable ()	判断枚举型属性参数是否可写
get_range()	获取枚举型属性参数范围字典
get()	读取枚举型属性参数的值和字符串
set(enum_value)	设置枚举型属性参数数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_range**

声明：

EnumFeature.get_range()

意义：

获取枚举型属性参数范围字典

返回值：

记录枚举型属性参数范围的字典

异常处理：

1) 如果该枚举型属性参数功能未实现，则打印不支持该枚举型属性参数获取范围的信息，函数返回 None。
2) 如果获取该枚举型属性参数范围不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

EnumFeature.get()

意义：

读取枚举型属性参数的值和字符串

返回值：

1) 枚举型属性参数的数值
2) 枚举型属性参数的字符串

异常处理：

1) 如果该枚举型属性参数功能未实现或不可读，则打印该枚举型属性参数不可读的信息，函数返回 None。
2) 如果获取枚举型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

EnumFeature.set(enum_value)

意义：

设置枚举型属性参数值

形参：

[in]enum_value　设置的枚举型数值

异常处理：

1) 如果输入参数不是整型值，则抛出 ParameterTypeError 异常。

2) 如果该枚举型属性参数功能未实现或不可写，则打印该枚举型属性参数不可写的信息，函数返回None。
3) 如果输入参数不在枚举型属性参数“值”的范围内，则打印超过该枚举型属性参数范围的信息并打印范围，函数返回 None。
4) 如果设置该枚举型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.5. BoolFeature
负责查看、控制相机的布尔型值功能，继承自 Feature 类。
接口列表：

is_implemented()	判断布尔型属性参数是否已实现
is_readable()	判断布尔型属性参数是否可读
is_writable ()	判断布尔型属性参数是否可写
get()	读取布尔型属性参数值
set(bool_value)	设置布尔型属性参数值

◆ **接口说明**

➢ **is_implemented**
详见 Feature::is_implemented()。

➢ **is_readable**
详见 Feature::is_readable()。

➢ **is_writable**
详见 Feature::is_writable()。

➢ **get**
声明：
BoolFeature.get()
意义：
读取布尔型属性参数值
返回值：
获取的布尔值
异常处理：
1) 如果该布尔型属性参数功能未实现或不可读，则打印该布尔型属性参数不可读的信息，函数返回None。
2) 如果获取布尔型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**
声明：
BoolFeature.set(bool_value)
意义：
设置浮点型属性参数值
形参：
[in]bool_value　设置的布尔型数值

异常处理：

1) 如果输入参数不是布尔型值，则抛出 ParameterTypeError 异常。
2) 如果该布尔型属性参数功能未实现或不可写，则打印该布尔型属性参数不可写的信息，函数返回 None。
3) 如果设置该布尔型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.6. StringFeature

负责查看、控制相机的字符串型值功能，继承自 Feature 类。

接口列表：

is_implemented()	判断字符串型属性参数是否已实现
is_readable()	判断字符串型属性参数是否可读
is_writable ()	判断字符串型属性参数是否可写
get_string_max_length()	获取字符串型属性参数值可设置的最长长度
get()	读取字符串型属性参数值
set(input_string)	设置字符串型属性参数值

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_string_max_length**

声明：

StringFeature.get_string_max_length()

意义：

获取字符串型属性参数可设置的最大长度

返回值：

字符串型属性参数可设置的最大长度

异常处理：

1) 如果该字符串型属性参数功能未实现，则打印不支持获取该字符串型属性参数的信息，函数返回 None。
2) 如果获取字符串型属性参数值可设置最大长度不成功，则抛出异常，异常类型详见错误处理。

➢ **get**

声明：

StringFeature.get()

意义：

读取字符串型属性参数值

返回值：

获取的字符串型属性参数值

异常处理：

1) 如果该字符串型属性参数功能未实现或不可读，则打印该字符串型属性参数不可读的信息，函数返回 None。
2) 如果获取该字符串型属性参数值不成功，则抛出异常，异常类型详见错误处理。

➢ **set**

声明：

StringFeature.set(input_string)

意义：

设置字符串型属性参数值

形参：

[in]input_string　　设置的字符串型数值

异常处理：

1) 如果输入参数不是字符串型值，则抛出 ParameterTypeError 异常。
2) 如果该字符串型属性参数功能未实现或不可写，则打印该字符串型属性参数不可写的信息，函数返回 None。
3) 如果输入参数长度大于可设置最大长度，则打印超过该字符串型属性参数长度最大值的信息并打印最大值，函数返回 None。
4) 如果设置该字符串型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.7. BufferFeature

负责查看、控制相机的缓冲功能，继承自 Feature 类。

接口列表：

is_implemented()	判断 Buffer 型属性参数是否已实现
is_readable()	判断 Buffer 型属性参数是否可读
is_writable ()	判断 Buffer 型属性参数是否可写
get_buffer_length()	获取 Buffer 型属性参数的长度
get_buffer()	读取 Buffer 型属性参数数据
set_buffer(buf)	设置 Buffer 型属性参数数据

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **get_buffer_length**

声明：
BufferFeature.get_buffer_length()
意义：
获取 Buffer 型属性参数的长度
返回值：
Buffer 型属性参数的长度
异常处理：

1) 如果该 Buffer 型属性参数功能未实现，则打印不支持该 Buffer 属性参数获取范围的信息，函数返回 None。
2) 如果获取 Buffer 型属性参数长度不成功，则抛出异常，异常类型详见错误处理。

- **get_buffer**

声明：
BufferFeature.get_buffer()
意义：
读取 Buffer 型属性参数数据
返回值：
Buffer 对象
异常处理：

1) 如果该 Buffer 型属性参数功能未实现或不可读，则打印该 Buffer 型属性参数不可读的信息，函数返回 None。
2) 如果获取该 Buffer 型属性参数数据不成功，则抛出异常，异常类型详见错误处理。

- **set_buffer**

声明：
BufferFeature.set_buffer(buf)
意义：
设置 Buffer 型属性参数数据
形参：
[in]buffer　　设置的缓冲数据[Buffer 类型]
异常处理：

1) 如果输入参数不是 Buffer 类型，则抛出 ParameterTypeError 异常。
2) 如果该 Buffer 型属性参数功能未实现或不可写，则打印该 Buffer 型属性参数不可写的信息，函数返回 None。
3) 如果输入 Buffer 型属性参数数据长度大于最大长度，则打印超过该 Buffer 型属性参数最大长度的信息并打印最大值，函数返回 None。
4) 如果设置该 Buffer 型属性参数不成功，则抛出异常，异常类型详见错误处理。

3.2.2.8. CommandFeature
发送命令功能，继承自 Feature 类。
接口列表：

is_implemented()	判断命令型属性参数是否已实现

is_readable()	判断命令型属性参数是否可读
is_writable ()	判断命令型属性参数是否可写
send_command()	发送命令

◆ **接口说明**

➢ **is_implemented**

详见 Feature::is_implemented()。

➢ **is_readable**

详见 Feature::is_readable()。

➢ **is_writable**

详见 Feature::is_writable()。

➢ **send_command**

声明：

CommandFeature.send_command()

意义：

发送命令

异常处理：

1) 如果该 Command 型属性参数功能未实现，则打印不支持该 Command 型属性参数的信息，函数返回 None。
2) 如果发送命令不成功，则抛出异常，异常类型详见错误处理。

3.3. 数据类型定义

3.3.1. GxDeviceClassList

定义	值	解释
UNKNOWN	0	未知设备种类
USB2	1	USB2.0 相机
GEV	2	千兆网相机（GigE Vision）
U3V	3	USB3.0 相机（USB3 Vision）
SMART	4	网络智能相机
CXP	5	CXP 相机

3.3.2. GxAccessStatus

定义	值	解释
UNKNOWN	0	设备当前状态未知
READWRITE	1	设备当前可读可写
READONLY	2	设备当前仅支持读
NOACCESS	3	设备当前既不支持读，又不支持写

3.3.3. GxAccessMode

定义	值	解释
READONLY	2	以只读方式打开设备
CONTROL	3	以控制方式打开设备
EXCLUSIVE	4	以独占方式打开设备

3.3.4. GxPixelFormatEntry

定义	值	解释
UNDEFINED	0x00000000	-
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO10_P	0x010A0046	Monochrome 10-bit packed
MONO12	0x01100005	Monochrome 12-bit unpacked
MONO12_P	0x010C0047	Monochrome 12-bit packed
MONO14	0x01100025	Monochrome 14-bit unpacked
MONO14_P	0x010E0104	Monochrome 14-bit packed
MONO16	0x01100007	Monochrome 16-bit
BAYER_GR8	0x01080008	Bayer Green-Red 8-bit
BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
BAYER_GR10_P	0x010A0056	Bayer Green-Red 10-bit packed
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_RG10_P	0x010A0058	Bayer Red-Green 10-bit packed
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_GB10_P	0x010A0054	Bayer Green-Blue 10-bit packed
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
BAYER_BG10_P	0x010A0052	Bayer Blue-Green 10-bit packed
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_GR12_P	0x010C0057	Bayer Green-Red 12-bit packed
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_RG12_P	0x010C0059	Bayer Red-Green 12-bit packed
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_GB12_P	0x010C0055	Bayer Green-Blue 12-bit packed
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_BG12_P	0x010C0053	Bayer Blue-Green 12-bit packed
BAYER_GR14	0x01100109	Bayer Green-Red 14-bit
BAYER_GR14_P	0x010E0105	Bayer Green-Red 14-bit packed

BAYER_RG14	0x0110010A	Bayer Red-Green 14-bit
BAYER_RG14_P	0x010E0106	Bayer Red-Green 14-bit packed
BAYER_GB14	0x0110010B	Bayer Green-Blue 14-bit
BAYER_GB14_P	0x010E0107	Bayer Green-Blue 14-bit packed
BAYER_BG14	0x0110010C	Bayer Blue-Green 14-bit
BAYER_BG14_P	0x010E0108	Bayer Blue-Green 14-bit packed
BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit
BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit unpacked
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit unpacked
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar
RGB8	0x2180014	Red-Green-Blue 8-bit
RGB10	0x2300018	Red-Green-Blue 10-bit
RGB12	0x230001A	Red-Green-Blue 12-bit
RGB14	0x230005E	Red-Green-Blue 14-bit
RGB16	0x2300033	Red-Green-Blue 16-bit
BGR8	0x2180015	Blue-Green-Red 8-bit
BGR10	0x2300019	Blue-Green-Red 10-bit
BGR12	0x230001B	Blue-Green-Red 12-bit
BGR14	0x230004A	Blue-Green-Red 14-bit
BGR16	0x230004B	Blue-Green-Red 16-bit
RGBA8	0x2200016	Red-Green-Blue-Alpha 8-bit
BGRA8	0x2200017	Blue-Green-Red-Alpha 8-bit
ARGB8	0x2200018	Alpha-Red-Green-Blue 8-bit
ABGR8	0x2200019	Alpha-Blue-Green-Red 8-bit
R8	0x010800C9	Red 8-bit
G8	0x010800CD	Green 8-bit
B8	0x010800D1	Blue 8-bit
COORD3D_ABC32F	0x026000C0	3D coordinate A-B-C 32-bit floating point
COORD3D_ABC32F_PLANAR	0x026000C1	3D coordinate A-B-C 32-bit floating point planar
COORD3D_C16	0x011000B8	3D coordinate C 16-bit
COORD3D_C16_I16	0x0110FF02	custom pixel format
COORD3D_C16_S16	0x0110FF03	custom pixel format
COORD3D_C16_I16_S16	0x0110FF04	custom pixel format
YUV444_8	0x2180020	YUV444 8-bit
YUV422_8	0x2100032	YUV422 8-bit

YUV411_8	0x20C001E	YUV411 8-bit
YUV420_8_PLANAR	0x20C0040	YUV420 8-bit planar
YCBCR444_8	0x218005B	YCBCR 444 8-bit
YCBCR422_8	0x210003B	YCBCR 422 8-bit
YCBCR411_8	0x20C005A	YCBCR 411 8-bit
YCBCR601_444_8	0x218003D	YCBCR601 444 8-bit
YCBCR601_422_8	0x210003E	YCBCR601 422 8-bit
YCBCR601_411_8	0x20C003F	YCBCR601 411 8-bit
YCBCR709_444_8	0x2180040	YCBCR709 444 8-bit
YCBCR709_422_8	0x2100041	YCBCR709 422 8-bit
YCBCR709_411_8	0x20C0042	YCBCR709 411 8-bit
MONO10_PACKED	0x010C0004	GigE Vision specific format, Monochrome 10-bit packed
MONO12_PACKED	0x010C0006	GigE Vision specific format, Monochrome 12-bit packed
BAYER_BG10_PACKED	0x010C0029	GigE Vision specific format, Bayer Blue-Green 10-bit packed
BAYER_BG12_PACKED	0x010C002D	GigE Vision specific format, Bayer Blue-Green 12-bit packed
BAYER_GB10_PACKED	0x010C0028	GigE Vision specific format, Bayer Green-Blue 10-bit packed
BAYER_GB12_PACKED	0x010C002C	GigE Vision specific format, Bayer Green-Blue 12-bit packed
BAYER_GR10_PACKED	0x010C0026	GigE Vision specific format, Bayer Green-Red 10-bit packed
BAYER_GR12_PACKED	0x010C002A	GigE Vision specific format, Bayer Green-Red 12-bit packed
BAYER_RG10_PACKED	0x010C0027	GigE Vision specific format, Bayer Red-Green 10-bit packed
BAYER_RG12_PACKED	0x010C002B	GigE Vision specific format, Bayer Red-Green 12-bit packed

3.3.5. GxFrameStatusList

定义	值	解释
SUCCESS	0	正常帧
IMCOMPLETE	-1	残帧

3.3.6. GxDeviceTemperatureSelectorEntry

定义	值	解释
SENSOR	1	传感器温度
MAINBOARD	2	主板温度

3.3.7. GxPixelSizeEntry

定义	值	解释
BPP8	8	像素大小 BPP8
BPP10	10	像素大小 BPP10
BPP12	12	像素大小 BPP12

BPP16	16	像素大小 BPP16
BPP24	24	像素大小 BPP24
BPP30	30	像素大小 BPP30
BPP32	32	像素大小 BPP32
BPP36	36	像素大小 BPP36
BPP48	48	像素大小 BPP48
BPP64	64	像素大小 BPP64

3.3.8. GxPixelColorFilterEntry

定义	值	解释
NONE	0	无
BAYER_RG	1	RG 格式
BAYER_GB	2	GB 格式
BAYER_GR	3	GR 格式
BAYER_BG	4	BG 格式

3.3.9. GxAcquisitionModeEntry

定义	值	解释
SINGLE_FRAME	0	单帧模式
MULITI_FRAME	1	多帧模式
CONTINUOUS	2	连续模式

3.3.10. GxTriggerSourceEntry

定义	值	解释
SOFTWARE	0	软触发
LINE0	1	触发源 0
LINE1	2	触发源 1
LINE2	3	触发源 2
LINE3	4	触发源 3
COUNTER2END	5	COUNTER2END 触发信号

3.3.11. GxTriggerActivationEntry

定义	值	解释
FALLING_EDGE	0	下降沿触发
RISING_EDGE	1	上升沿触发

3.3.12. GxExposureModeEntry

定义	值	解释
TIMED	1	曝光时间寄存器控制曝光时间
TRIGGER_WIDTH	2	触发信号宽度控制曝光时间

3.3.13. GxUserOutputSelectorEntry

定义	值	解释
OUTPUT0	1	输出 0
OUTPUT1	2	输出 1
OUTPUT2	4	输出 2

3.3.14. GxUserOutputModeEntry

定义	值	解释
STROBE	0	闪光灯
USER_DEFINED	1	用户自定义

3.3.15. GxGainSelectorEntry

定义	值	解释
ALL	0	所有增益通道
RED	1	红通道增益
GREEN	2	绿通道增益
BLUE	3	蓝通道增益

3.3.16. GxBlackLevelSelectEntry

定义	值	解释
ALL	0	所有黑电平通道
RED	1	红通道黑电平
GREEN	2	绿通道黑电平
BLUE	3	蓝通道黑电平

3.3.17. GxBalanceRatioSelectorEntry

定义	值	解释
RED	0	红通道
GREEN	1	绿通道
BLUE	2	蓝通道

3.3.18. GxAALightEnvironmentEntry

定义	值	解释
NATURE_LIGHT	0	自然光
AC50HZ	1	50 赫兹日光灯
AC60HZ	2	60 赫兹日光灯

3.3.19. GxUserSetEntry

定义	值	解释
DEFAULT	0	默认参数组
USER_SET0	1	用户参数组 0

3.3.20. GxAWBLampHouseEntry

定义	值	解释
ADAPTIVE	0	自适应光源
D65	1	指定色温 6500k
FLUORESCENCE	2	指定荧光灯
INCANDESCENT	3	指定白炽灯
D75	4	指定色温 7500k
D50	5	指定色温 5000k
U30	6	指定色温 3000k

3.3.21. GxUserDataFieldSelectorEntry

定义	值	解释
FIELD_0	0	Flash 数据区域 0
FIELD_1	1	Flash 数据区域 1
FIELD_2	2	Flash 数据区域 2
FIELD_3	3	Flash 数据区域 3

3.3.22. GxTestPatternEntry

定义	值	解释
OFF	0	关闭
GRAY_FRAME_RAMP_MOVING	1	静止灰度递增
SLANT_LINE_MOVING	2	滚动斜条纹
VERTICAL_LINE_MOVING	3	滚动竖条纹
HORIZONTAL_LINE_MOVING	4	滚动横条纹
GREY_VERTICAL_RAMP	5	垂直灰度递增
SLANT_LINE	6	静止斜条纹

3.3.23. GxTriggerSelectorEntry

定义	值	解释
FRAME_START	1	采集一帧
FRAME_BURST_START	2	帧高速连拍开始

3.3.24. GxLineSelectorEntry

定义	值	解释
LINE0	0	引脚 0
LINE1	1	引脚 1
LINE2	2	引脚 2
LINE3	3	引脚 3
LINE4	4	引脚 4
LINE5	5	引脚 5

LINE6	6	引脚 6
LINE7	7	引脚 7
LINE8	8	引脚 8
LINE9	9	引脚 9
LINE10	10	引脚 10
LINE_STROBE	11	专用闪光灯引脚

3.3.25. GxLineModeEntry

定义	值	解释
INPUT	0	输入
OUTPUT	1	输出

3.3.26. GxLineSourceEntry

定义	值	解释
OFF	0	关闭
STROBE	1	闪光灯
USER_OUTPUT0	2	用户自定义输出 0
USER_OUTPUT1	3	用户自定义输出 1
USER_OUTPUT2	4	用户自定义输出 2
EXPOSURE_ACTIVE	5	曝光有效
FRAME_TRIGGER_WAIT	6	单帧触发等待
ACQUISITION_TRIGGER_WAIT	7	多帧触发等待
TIMER1_ACTIVE	8	定时器 1 有效
USER_OUTPUT3	9	用户自定义输出 3
USER_OUTPUT4	10	用户自定义输出 4
USER_OUTPUT5	11	用户自定义输出 5
USER_OUTPUT6	12	用户自定义输出 6

3.3.27. GxEventSelectorEntry

定义	值	解释
EXPOSURE_END	0x0004	曝光结束
BLOCK_DISCARD	0x9000	图像帧丢弃
EVENT_OVERRUN	0x9001	事件队列溢出
FRAME_START_OVER_TRIGGER	0x9002	触发信号溢出
BLOCK_NOT_EMPTY	0x9003	图像帧存不为空
INTERNAL_ERROR	0x9004	内部错误事件
FRAME_BURST_START_OVERTRIGGER	0x9005	多帧触发屏蔽事件
FRAME_START_WAIT	0x9006	帧等待事件

FRAME_BURST_START_WAIT	0x9007	多帧等待事件

3.3.28. GxLutSelectorEntry

定义	值	解释
LUMINANCE	0	亮度

3.3.29. GxTransferControlModeEntry

定义	值	解释
BASIC	0	基础模式
USER_CONTROLED	1	用户控制模式

3.3.30. GxTransferOperationModeEntry

定义	值	解释
MULTI_BLOCK	0	指定发送帧数

3.3.31. GxTestPatternGeneratorSelectorEntry

定义	值	解释
SENSOR	0	sensor 的测试图
REGION0	1	FPGA 的测试图

3.3.32. GxChunkSelectorEntry

定义	值	解释
FRAME_ID	1	帧号
TIME_STAMP	2	时间戳
COUNTER_VALUE	3	计数器值

3.3.33. GxBinningHorizontalModeEntry

定义	值	解释
SUM	0	BINNING 水平值和
AVERAGE	1	BINNING 水平值平均值

3.3.34. GxBinningVerticalModeEntry

定义	值	解释
SUM	0	BINNING 垂直值和
AVERAGE	1	BINNING 垂直值平均值

3.3.35. GxSensorShutterModeEntry

定义	值	解释
GLOBAL	0	所有的像素同时曝光且曝光时间相等
ROLLING	1	所有的像素曝光时间相等，但曝光起始时间不同

GLOBALRESET	2	所有的像素曝光起始时间相同，但曝光时间不相等

3.3.36. GxAcquisitionStatusSelectorEntry

定义	值	解释
ACQUISITION_TRIGGER_WAIT	0	采集触发等待
FRAME_TRIGGER_WAIT	1	帧触发等待

3.3.37. GxExposureTimeModeEntry

定义	值	解释
ULTRASHORT	0	极小曝光
STANDARD	1	标准

3.3.38. GxGammaModeEntry

定义	值	解释
SRGB	0	默认 Gamma 校正
USER	1	用户自定义 Gamma 校正

3.3.39. GxLightSourcePresetEntry

定义	值	解释
OFF	0	关闭
CUSTOM	1	用户自定义
DAYLIGHT_6500K	2	标准日光灯（6500K）
DAYLIGHT_5000K	3	标准日光灯（5000K）
COOL_WHITE_FLUORESCENCE	4	冷白光源（4150K）
INCA	5	螺旋钨丝灯（2856K）

3.3.40. GxColorTransformationModeEntry

定义	值	解释
RGB_TO_RGB	0	默认颜色校正
USER	1	用户自定义颜色校正

3.3.41. GxColorTransformationValueSelectorEntry

定义	值	解释
GAIN00	0	颜色转换分量增益值 GAIN00
GAIN01	1	颜色转换分量增益值 GAIN01
GAIN02	2	颜色转换分量增益值 GAIN02
GAIN10	3	颜色转换分量增益值 GAIN10
GAIN11	4	颜色转换分量增益值 GAIN11
GAIN12	5	颜色转换分量增益值 GAIN12

GAIN20	6	颜色转换分量增益值 GAIN20
GAIN21	7	颜色转换分量增益值 GAIN21
GAIN22	8	颜色转换分量增益值 GAIN22

3.3.42. GxAutoEntry

定义	值	解释
OFF	0	关闭
CONTINUOUS	1	连续
ONCE	2	单次

3.3.43. GxSwitchEntry

定义	值	解释
OFF	0	关闭
ON	1	开启

3.3.44. GxRegionSendModeEntry

定义	值	解释
SINGLE_ROI	0	单 ROI
MULTI_ROI	1	多 ROI

3.3.45. GxRegionSelectorEntry

定义	值	解释
REGION0	0	区域 0
REGION1	1	区域 1
REGION2	2	区域 2
REGION3	3	区域 3
REGION4	4	区域 4
REGION5	5	区域 5
REGION6	6	区域 6
REGION7	7	区域 7

3.3.46. GxTimerSelectorEntry

定义	值	解释
TIMER1	1	定时器 1

3.3.47. GxTimerTriggerSourceEntry

定义	值	解释
EXPOSURE_START	1	曝光开始信号开始计时
LINE10	10	接收引脚 10 信号开始计时
STROBE	16	接收闪光灯信号开始计时

3.3.48. GxCounterSelectorEntry

定义	值	解释
COUNTER1	1	计数器 1
COUNTER2	2	计数器 2

3.3.49. GxCounterEventSourceEntry

定义	值	解释
FRAME_START	1	统计 "帧开始" 事件的数量
FRAME_TRIGGER	2	统计 "帧触发" 事件的数量
ACQUISITION_TRIGGER	3	统计 "采集触发" 事件的数量
OFF	4	关闭
SOFTWARE	5	统计 "软触发" 事件的数量
LINE0	6	统计 "Line 0 触发" 事件的数量
LINE1	7	统计 "Line 1 触发" 事件的数量
LINE2	8	统计 "Line 2 触发" 事件的数量
LINE3	9	统计 "Line 3 触发" 事件的数量

3.3.50. GxCounterResetSourceEntry

定义	值	解释
OFF	0	无复位源
SOFTWARE	1	软触发
LINE0	2	引脚 0
LINE1	3	引脚 1
LINE2	4	引脚 2
LINE3	5	引脚 3
COUNTER2END	6	COUNTER2END 触发信号

3.3.51. GxCounterResetActivationEntry

定义	值	解释
RISINGEDGE	1	上升沿触发

3.3.52. GxCounterTriggerSourceEntry

定义	值	解释
OFF	0	无触发源
SOFTWARE	1	软触发
LINE0	2	引脚 0
LINE1	3	引脚 1
LINE2	4	引脚 2
LINE3	5	引脚 3

3.3.53. GxTimerTriggerActivationEntry

定义	值	解释
RISINGEDGE	1	上升沿触发

3.3.54. GxStopAcquisitionModeEntry

定义	值	解释
GENERAL	0	普通停采
LIGHT	1	轻量级停采

3.3.55. GxDSStreamBufferHandlingModeEntry

定义	值	解释
OLDEST_FIRST	1	OldestFirst 模式
OLDEST_FIRST_OVERWRITE	2	OldestFirstOverwrite 模式
NEWEST_ONLY	3	NewestOnly 模式

3.3.56. GxResetDeviceModeEntry

定义	值	解释
RECONNECT	1	重连设备
RESET	2	复位设备

3.3.57. DxBayerConvertType

定义	值	解释
NEIGHBOUR	0	邻域平均插值算法
ADAPTIVE	1	边缘自适应插值算法
NEIGHBOUR3	2	更大区域的邻域平均插值算法

3.3.58. DxValidBit

定义	值	解释
BIT0_7	0	0-7 位
BIT1_8	1	1-8 位
BIT2_9	2	2-9 位
BIT3_10	3	3-10 位
BIT4_11	4	4-11 位
BIT5_12	5	5-12 位
BIT6_13	6	6-13 位
BIT7_14	7	7-14 位
BIT8_15	8	8-15 位

3.3.59. DxImageMirrorMode

定义	值	解释
HORIZONTAL_MIRROR	0	水平镜像
VERTICAL_MIRROR	1	垂直镜像

3.3.60. DxRGBChannelOrder

定义	值	解释
ORDER_RGB	0	RGB 通道
ORDER_BGR	1	BGR 通道

3.3.61. GxTLClassList

定义	值	解释
TL_TYPE_UNKNOWN	0	未知 TL 类型
TL_TYPE_USB	1	USB TL 类型
TL_TYPE_GEV	2	GEV TL 类型
TL_TYPE_U3V	4	U3V TL 类型
TL_TYPE_CXP	8	CXP TL 类型

3.3.62. GxImageInfo

定义	解释
image_width	图像宽
image_height	图像高
image_buf	图像 buffer
image_pixel_format	图像像素格式

3.4. 模块接口定义

3.4.1. DeviceManager

负责相机设备的管理，包括枚举设备、打开设备、获取设备数量信息等。

接口列表：

update_device_list (timeout=200)	枚举同一网段中的设备
update_all_device_list (timeout=200)	枚举不同网段中的设备
update_device_list_ex (tl_type, timeout=2000)	按照指定 TL 类型枚举对应设备
get_device_number ()	获取设备数量
get_device_info ()	获取设备信息
get_interface_number()	获取 Interface 数量
get_interface_info()	获取 Interface 信息

get_interface()	获取 Interface 对象
open_device_by_sn (sn, access_mode=GxAccessMode.CONTROL)	通过序列号打开设备
open_device_by_user_id (user_id, access_mode=GxAccessMode.CONTROL)	通过用户 ID 号打开设备
open_device_by_index (index, access_mode=GxAccessMode.CONTROL)	通过设备索引打开设备
open_device_by_ip (ip, access_mode=GxAccessMode.CONTROL)	通过 IP 地址打开设备
open_device_by_mac (mac, access_mode=GxAccessMode.CONTROL)	通过 mac 地址打开设备
gige_reset_device (mac_address, reset_device_mode)	执行设备重连或复位操作
gige_force_ip(mac_address, ip_address, subnet_mask, default_gate_way)	设置设备 ForceIP
gige_ip_configuration(mac_address, ipconfig_flag, ip_address, subnet_mask, default_gateway, user_id)	进行 IP 配置
create_image_format_convert()	创建图像像素格式转换对象
create_image_process()	创建图像处理对象

◆ **接口说明**

➢ **update_device_list**

声明：

DeviceManager.update_device_list (timeout=200)

意义：

对于非千兆网相机，枚举所有设备；对于千兆网相机，枚举同一网段设备。

形参：

[in]timeout　枚举超时[0，0xffffffff]，缺省值为 200（ms）

返回值：

枚举得到设备数量和记录枚举设备信息的列表（list）。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型字典，字典中的键名称详见枚举设备

异常处理：

1) 如果输入参数不是整型值，则抛出 ParameterTypeError 异常。
2) 如果输入参数小于 0 或大于无符号整型的最大值，则打印”DeviceManager.update_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff”，函数返回 None。
3) 如果枚举同一网段中设备不成功，则抛出异常，异常类型详见错误处理。
4) 如果获取所有设备基本信息不成功，则抛出异常，异常类型详见错误处理。

➢ **update_all_device_list**

声明：

DeviceManager.update_all_device_list (timeout=200)

意义：

对于非千兆网相机，枚举所有设备；对于千兆网相机，枚举全网设备。

形参：

[in]timeout　枚举超时[0，0xffffffff]，缺省值为 200（ms）

返回值：

枚举得到设备数量和记录枚举设备信息的列表（list）。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型字典，字典中的键名称详见枚举设备

异常处理：

1) 如果输入参数不是整型值，则抛出 ParameterTypeError 异常。
2) 如果输入参数小于 0 或大于无符号整型的最大值，则打印”DeviceManager.update_all_device_list: Out of bounds, timeout:minimum=0, maximum= 0xffffffff”，函数返回 None。
3) 如果枚举不同网段中设备不成功，则抛出异常，异常类型详见错误处理。
4) 如果获取所有设备基本信息不成功，则抛出异常，异常类型详见错误处理。

- **update_device_list_ex**

声明：

DeviceManager.update_device_list_ex (tl_type, timeout=2000)

意义：

按照指定 TL 类型枚举对应设备。

形参：

[in]tl_type　TL 类型，详见 GxTLClassList

[in]timeout　枚举超时[0，0xffffffff]，缺省值为 2000（ms）

返回值：

枚举得到设备数量和记录枚举设备信息的列表（list）。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型字典，字典中的键名称详见枚举设备

异常处理：

21) 如果获取所有设备基本信息不成功，则抛出异常，异常类型详见错误处理。
22)

- **get_device_number**

声明：

DeviceManager.get_device_number ()

意义：

获取设备数量

返回值：

设备数量

- **get_device_info**

声明：

DeviceManager.get_device_info()

意义：

获取设备信息

返回值：

设备信息列表。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型为字典，字典的键详见枚举设备

➢ **get_interface_number**

声明：

DeviceManager.get_interface_number ()

意义：

获取 Interface 数量。

返回值：

Interface 数量

➢ **get_interface_info**

声明：

DeviceManager.get_interface_info ()

意义：

获取 Interface 信息。

返回值：

Interface 信息列表。设备信息列表的元素个数为枚举到的 Interface 个数，列表中元素的数据类型为字典，键包含：

键名称	意义	类型
type	TL 类型	GxTLClassList
display_name	显示名称	字符串
interface_id	Interface ID	字符串
serial_number	设备序列号	字符串
description	设备显示名称	字符串
init_flag	初始标识（CXP 采集卡特有）	整型
reserved	用户自定义名称	字符串数组

➢ **get_interface**

声明：

DeviceManager.get_interface()

意义：

获取 Interface 对象。

返回值：

Interface 对象，详细见 Interface。

➢ **open_device_by_sn**

声明：

DeviceManager.open_device_by_sn (sn, access_mode=GxAccessMode.CONTROL)

意义：

通过序列号打开设备

形参：

[in]sn 序列号[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值:

设备对象

异常处理:

1) 如果输入参数 1 不是字符串型，则抛出 ParameterTypeError 异常。
2) 如果输入参数 2 不是整型，则抛出 ParameterTypeError 异常。
3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值信息，函数返回 None。
4) 如果重复获取设备类不成功，则抛出 NotFoundDevice 异常。
5) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。
6) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_user_id**

声明:

DeviceManager.open_device_by_user_id (user_id, access_mode=GxAccessMode.CONTROL)

意义:

通过用户 ID 号打开设备

形参:

[in]user_id 用户 ID 号[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值:

设备对象

异常处理:

23) 如果输入参数 1 不是字符串型，则抛出 ParameterTypeError 异常。

1) 如果输入参数 2 不是整型，则抛出 ParameterTypeError 异常。
2) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。
3) 如果重复获取设备类不成功，则抛出 NotFoundDevice 异常。
4) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。
5) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_index**

声明:

DeviceManager.open_device_by_index (index, access_mode=GxAccessMode.CONTROL)

意义:

通过设备索引打开设备

形参:

[in]index 设备索引[1,2,3...0xffffffff]

[in]access_mode 打开设备方式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值:

设备对象

异常处理：

1) 如果输入参数 1 或 2 不是整型值，则抛出 ParameterTypeError 异常。
2) 如果输入参数 1 小于 0 或大于无符号整型的最大值，则打印”DeviceManager.open_device_by_index: index out of bounds, index: minimum=1, maximum=0xffffffff”，函数返回 None。
3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。
4) 如果设备数量小于输入参数 1 索引，则抛出 NotFoundDevice 异常。
5) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。
6) 如果打开获取的设备不是 U3V/USB2/GEV 类中的一种，则抛出 NotFoundDevice 异常。

- **open_device_by_ip**

声明：

DeviceManager.open_device_by_ip (ip, access_mode=GxAccessMode.CONTROL)

意义：

通过设备 ip 地址打开千兆网相机设备

形参：

[in]ip 设备 ip 地址[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出 ParameterTypeError 异常。
2) 如果输入参数 2 不是整型，则抛出 ParameterTypeError 异常。
3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。
4) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。

- **open_device_by_mac**

声明：

DeviceManager.open_device_by_mac (mac, access_mode=GxAccessMode.CONTROL)

意义：

通过设备 mac 地址打开千兆网相机设备

形参：

[in]mac 设备 mac 地址[字符串类型]

[in]access_mode 打开设备模式，缺省值为 GxAccessMode.CONTROL，查看 GxAccessMode

返回值：

设备对象

异常处理：

1) 如果输入参数 1 不是字符串型，则抛出 ParameterTypeError 异常。
2) 如果输入参数 2 不是整型，则抛出 ParameterTypeError 异常。

3) 如果输入参数 2 不在打开设备模式 GxAccessMode 中，则打印接口名称、打开设备方式不在范围内和当前参数所支持的枚举值的信息，函数返回 None。

4) 如果打开设备不成功，则抛出异常，异常类型详见错误处理。

- **gige_reset_device**

声明：

DeviceManager.gige_reset_device (mac_address, reset_device_mode)

意义：

执行设备重连或复位操作。

设备重连通常应用在调试千兆网相机时，设备已经被打开，此时程序异常，而后立即重新打开设备报错（因为调试心跳 5 分钟，设备依然处于打开状态），这时可通过设备重连功能，使设备处于未打开状态，而后再次打开设备即可成功。

设备复位通常应用在相机状态异常，此时设备重连功能也无法生效，也不具备给设备重新上电的条件，可尝试使用设备复位功能，使设备掉电再上电。设备复位后，需要重新执行枚举、打开设备操作。

注意：

1) 设备复位时间需要 1s 左右，因此需要确保 1s 后再调用枚举接口。

2) 如果设备正在正常采集，禁止使用设备复位和重连功能，否则会导致设备掉线。

形参：

[in]mac_address 设备 mac 地址[字符串类型]

[in]reset_device_mode 重置设备模式，参考 GxResetDeviceModeEntry。

返回值：

None

异常处理：

24) 如果执行命令失败，则抛出异常，异常类型详见错误处理。

- **gige_force_ip**

声明：

DeviceManager.gige_force_ip(mac_address, ip_address, subnet_mask, default_gate_way)

意义：

执行 force ip 操作。

形参：

[in]mac_address	mac 地址
[in]ip_address	ip 地址
[in]subnet_mask	子网掩码
[in]default_gate_way	默认网关

异常处理：

25) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **gige_ip_configuration**

声明：

DeviceManager.gige_ip_configuration(mac_address, ipconfig_flag, ip_address, subnet_mask, default_gateway, user_id)

意义：

执行 ip 配置操作。

形参：

[in]mac_address	mac 地址
[in]ip_address	ip 地址
[in]subnet_mask	子网掩码
[in]default_gate_way	默认网关
[in]ipconfig_flag	默认网关
[in]user_id	默认网关

异常处理：

26) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **create_image_format_convert**

声明：

DeviceManager.create_image_format_convert()

意义：

创建图像像素格式转换对象。

返回值：

像素格式转换对象，详见 ImageFormatConvert。

- **create_image_process()**

声明：

DeviceManager.create_image_process()

意义：

创建图像处理对象。

返回值：

图像处理对象，详见 ImageProcess。

27)

3.4.2. Device

负责相机设备的采集控制、设备关闭、配置文件导入导出和获取设备句柄等。

接口列表：

get_stream_channel_num()	获取当前设备支持的流通道个数
stream_on ()	发送开始命令，相机开始传送图像数据
stream_off ()	发送结束命令，相机结束传送图像数据
export_config_file (file_path)	导出当前配置文件
import_config_file (file_path, verify=False)	导入配置文件
close_device ()	关闭设备，销毁设备句柄，将句柄置为空
get_stream_channel_num()	获取流通道数量
get_stream(stream_index)	获取流对象
get_local_device_feature_control()	获取本地属性控制器

get_remote_device_feature_control()	获取远端属性控制器
register_device_feature_callback_by_string(callback_func, feature_name, args)	通过字符串注册功能属性更新回调函数
unregister_device_feature_callback_by_string(feature_name, feature_callback_handle)	通过字符串注销功能属性更新回调函数
create_image_process_config()	创建图像处理配置对象
read_remote_device_port(address, buff, size)	读远端寄存器地址值(不再维护)
write_remote_device_port(address, buf, size)	写远端寄存器地址值(不再维护)
read_remote_device_port_stacked(entries, size)	批量读用户指定寄存器的值(不再维护)
write_remote_device_port_stacked(entries, size)	批量写用户指定寄存器的值(不再维护)
register_device_feature_callback(callback_func, feature_id, args)	通过功能码注册功能属性更新回调函数(不再维护)
unregister_device_feature_callback(feature_id, feature_callback_handle)	通过功能码注销功能属性更新回调函数(不再维护)

◆ **接口说明**

➢ **get_stream_channel_num**

声明:

Device.get_stream_channel_num()

意义:

获取当前设备支持的流通道个数

返回值:

流通道个数

注：目前千兆网相机、USB3.0、USB2.0 相机均不支持多流通道

➢ **stream_on**

声明:

Device.stream_on()

意义:

发送开始命令，相机开始传送图像数据

返回值:

None

异常处理:

1) 如果发送开始命令不成功，则抛出异常，异常类型详见错误处理。

➢ **stream_off**

声明:

Device.stream_off()

意义:

发送停止命令，相机停止传送图像数据

返回值:

None

异常处理:

1) 如果发送停止命令不成功，则抛出异常，异常类型详见错误处理。

- **export_config_file**

声明：
Device.export_config_file(file_path)
意义：
导出当前配置文件
形参：
[in]file_path 文件路径
返回值：
None
异常处理：
1) 如果输入参数不是字符串型，则抛出 ParameterTypeError 异常。
2) 如果导出当前配置文件不成功，则抛出异常，异常类型详见错误处理。

- **import_config_file**

声明：
Device.import_config_file(file_path, verify=False)
意义：
导入配置文件
形参：
[in]file_path 文件路径
[in]verify 是否所有导入值将被验证一致性，缺省值为 False
返回值：
None
异常处理：
1) 如果输入参数 1 不是字符串型，则抛出 ParameterTypeError 异常。
2) 如果输入参数 2 不是布尔型，则抛出 ParameterTypeError 异常。
3) 如果导入配置文件不成功，则抛出异常，异常类型详见错误处理。

- **close_device**

声明：
Device.close_device()
意义：
关闭设备，销毁设备句柄，将句柄置为空
返回值：
None
异常处理：
1) 如果关闭设备不成功，则抛出异常，异常类型详见错误处理。
注意：
当执行关闭设备后，如果还想使用此相机，请重新打开后再操作。

- **register_device_offline_callback**

声明：

Device.register_device_offline_callback(callback_func)

意义：

注册掉线回调函数。

形参：

[in]callback_func　掉线回调函数

返回值：

None

异常处理：

1) 如果输入参数不是函数类型，则抛出 ParameterTypeError 异常。
2) 如果注册掉线回调函数失败，则抛出异常，异常类型详见错误处理。

- **unregister_device_offline_callback**

声明：

Device.unregister_device_offline_callback()

意义：

注销设备掉线回调函数。

返回值：

None

异常处理：

28) 如果注销掉线回调函数失败，则抛出异常，异常类型详见错误处理。
29)

- **get_stream_channel_num**

声明：

Device.get_stream_channel_num()

意义：

获取流通道数量。

返回值：

流通道数量。

- **get_stream**

声明：

Device.get_stream(stream_index)

意义：

获取流对象。

形参：

[in]stream_index　流数组索引值，从 1 开始

返回值：

流对象，详见 DataStream。

异常处理：

30) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_local_device_feature_control**

声明：

Device.get_local_device_feature_control()

意义：

获取本地属性控制器。

返回值：

属性控制器对象，详见 FeatureControl。

异常处理：

31）如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_remote_device_feature_control**

声明：

Device.get_remote_device_feature_control()

意义：

获取远端属性控制器。

返回值：

属性控制器对象，详见 FeatureControl。

- **register_device_feature_callback_by_string**

声明：

Device.register_device_feature_callback_by_string(callback_func, feature_name, args)

意义：

通过字符串注册功能属性更新回调函数。

形参：

[in]callback_func	属性更新回调函数
[in]feature_name	功能字符串节点
[in]args	用户参数可为 None

返回值：

回调函数句柄。

异常处理：

32）如果注册回调函数失败，则抛出异常，异常类型详见错误处理。

- **unregister_device_feature_callback_by_string**

声明：

Device.unregister_device_feature_callback_by_string(feature_name, feature_callback_handle)

意义：

通过功能码注销功能属性更新回调函数。

形参：

[in]feature_name	功能字符串节点
[in]feature_callback_handle	回调函数句柄

返回值：

None

异常处理：

33）如果注销回调函数失败，则抛出异常，异常类型详见错误处理。

- **create_image_process_config()**

声明：

Device.create_image_process_config()

意义：

创建图像处理配置对象。

返回值：

图像处理配置对象，详见 ImageProcessConfig。

异常处理：

34) 如果输入参数不支持功能节点"ColorCorrectionParam"，或支持但不可读则抛出 UnexpectedError 异常。

- **read_remote_device_port(不再维护)**

推荐用 FeatureControl.read_port()

声明：

Device.read_remote_device_port(address, buff, size)

意义：

读用户指定寄存器的值。

形参：

[in]address	寄存器地址
[out]buff	返回寄存器内容的缓存地址
[in|out]size	用户申请的 Buffer 大小,调用完后返回实际大小

异常处理：

35) 如果调用函数失败，则抛出异常，异常类型详见错误处理。

- **write_remote_device_port(不再维护)**

- **推荐用 FeatureControl.write_port()**

声明：

Device.write_remote_device_port(address, buf, size)

意义：

向用户指定的寄存器中写入用户给定的数据。

形参：

[in]address	寄存器地址
[in]buff	要写的寄存器内容的缓存地址
[in]size	用户申请的 Buffer 大小,调用完后返回实际大小

异常处理：

36) 如果调用函数失败，则抛出异常，异常类型详见错误处理。

- **read_remote_device_port_stacked(不再维护)**

- **推荐用 FeatureControl.read_port_stacked()**

声明：

Device.read_remote_device_port_stacked(entries, size)

意义：

批量读用户指定寄存器的值（仅限命令值为 4 字节长度的寄存器）。

形参：

[in]entries 批量读寄存器结构体地址

[in|out]size 读取设备寄存器的个数

异常处理：

37) 如果调用函数失败，则抛出异常，异常类型详见错误处理。

- **write_remote_device_port_stacked(不再维护)**

- **推荐用 FeatureControl.write_port_stacked()**

声明：

Device.write_remote_device_port_stacked(entries, size)

意义：

批量向用户指定的寄存器中写入用户给定的数据（仅限命令值为 4 字节长度的寄存器）

形参：

[in]entries 批量写寄存器结构体地址

[in]size 写设备寄存器的个数

异常处理：

38) 如果调用函数失败，则抛出异常，异常类型详见错误处理。

- **register_device_feature_callback(不再维护)**

- **推荐用 register_device_feature_callback_by_string**

声明：

Device.register_device_feature_callback(callback_func, feature_id, args)

意义：

通过功能码注册功能属性更新回调函数。

形参：

[in]callback_func 属性更新回调函数

[in]feature_id 功能码节点

[in]args 用户参数可为 None

返回值：

回调函数句柄。

异常处理：

39) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **unregister_device_feature_callback(不再维护)**

- **推荐用 unregister_device_feature_callback_by_string**

声明：

Device.unregister_device_feature_callback(feature_id, feature_callback_handle)

意义：

通过功能码注销功能属性更新回调函数。

形参：

[in]feature_id	功能码节点
[in]feature_callback_handle	回调函数句柄

返回值：

None

异常处理：

40) 如果注销回调函数失败，则抛出异常，异常类型详见错误处理。

41)

3.4.3. Interface

封装 Interface 类。

接口列表：

get_interface_info()	获取 Interface 信息
get_feature_control()	获取 Interface 属性控制对象

◆ **接口说明**

➢ **get_interface_info**

声明：

get_interface_info()

意义：

获取 Interface 信息。.

返回值：

返回 Interface 信息字典，键包含：

键名称	意义	类型
type	TL 类型	GxTLClassList
display_name	显示名称	字符串
interface_id	Interface ID	字符串
serial_number	设备序列号	字符串
description	设备显示名称	字符串
init_flag	初始标识（CXP 采集卡特有）	整型
reserved	用户自定义名称	字符串数组

➢ **get_feature_control**

声明：

get_feature_control()

意义：

获取 Interface 属性控制对象。

返回值：

Interface 属性控制对象，详细见 FeatureControl

3.4.4. DataStream

负责相机设备的数据流设置、控制，获取图像等。

接口列表：

set_acquisition_buffer_number(buf_num)	设置采集缓冲的大小
get_image(timeout=1000)	获取图像，成功创建图像类对象
flush_queue()	清除相机采集缓冲队列
register_capture_callback(callback_func)	注册采集回调函数
unregister_capture_callback()	注销采集回调函数
get_featrue_control()	获取属性控制对象
get_payload_size()	获取 Payloadsize

◆ **接口说明**

➢ **set_acquisition_buffer_number**

声明：

DataStream.set_acquisition_buffer_number(buf_num)

意义：

设置采集缓冲的大小

形参：

[in]buf_num　缓冲区地址的长度[1，0xffffffff]

返回值：

None

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。

2) 如果输入参数小于 1 或大于无符号整型的最大值，则打印“DataStream.set_acquisition_buffer_number: buf_num out of bounds, minimum=1, maximum=0xffffffff”，函数返回 None。

3) 如果设置采集缓冲大小不成功，则抛出异常，异常类型详见错误处理。

➢ **get_image**

声明：

DataStream.get_image(timeout=1000)

意义：

获取图像，成功创建图像类对象

形参：

[in]timeout 获取超时[0，0xffffffff]，缺省值为 1000（ms）

返回值：

图像对象：　获取成功

None:　超时

抛出异常：　其他错误

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果输入参数小于 0 或大于 0xffffffff，则打印"DataStream.get_image: timeout out of bounds, minimum=0, maximum=0xffffffff"，函数返回 None。
3) 如果获取数据大小不成功，则抛出异常，异常类型详见错误处理。
4) 如果超时导致未获取图像成功，则函数返回 None。
5) 如果未超时但获取图像失败，则打印"status，DataStream，get_image"，函数返回 None。

- **flush_queue**

声明：
DataStream.flush_queue()
意义：
清除相机采集缓冲队列
异常处理：
1) 如果清除相机采集缓冲队列不成功，则抛出异常，异常类型详见错误处理。

- **register_capture_callback**

声明：
DataStream.register_capture_callback(callback_func)
意义：
注册采集回调函数，回调方式采集使用方式见：采集控制-回调方式
形参：
[in]callback_func 采集回调函数
返回值：
None
异常处理：
1) 如果输入参数不是函数类型，则抛出 ParameterTypeError 异常。
2) 如果注册采集回调函数失败，则抛出异常，异常类型详见错误处理。
注意：
需要先注册采集回调函数，再执行 stream_on 开采。

- **unregister_capture_callback**

声明：
DataStream.unregister_capture_callback()
意义：
注销采集回调函数
返回值：
None
异常处理：
42) 如果注销采集回调函数失败，则抛出异常，异常类型详见错误处理。
43)

- **get_featrue_control**

声明：
DataStream.get_featrue_control()

意义：

获取流属性控制对象。

返回值：

流属性控制对象，详见 FeatureControl。

➢ **get_payload_size**

声明：

DataStream.get_payload_size

意义：

获取 payloadsize。

返回值：

返回 payloadsize。

异常处理：

如果调用函数失败，则抛出异常，异常类型详见错误处理。

3.4.5. FeatureControl

负责查询各节点访问属性是否支持、可读、可写以及进行设置数据、获取数据。

接口列表：

is_implemented(feature_name)	查询当前功能是否支持
is_readable(feature_name)	查询当前功能是否可读
is_writable(feature_name)	查询当前功能是否可写
get_int_feature(feature_name)	获取一个整型控制对象
get_enum_feature(feature_name)	获取一个浮点型控制对象
get_float_feature(feature_name)	获取一个枚举型控制对象
get_bool_feature(feature_name)	获取一个布尔型控制对象
get_string_feature(feature_name)	获取一个字符串型控制对象
get_command_feature(feature_name)	获取一个命令型控制对象
get_register_feature(feature_name)	获取一个寄存器类型控制对象
feature_save(file_path)	保存用户参数组
feature_load(file_path, verify=False)	加载用户参数组
read_port(address, size)	读指定寄存器的值
write_port(address, buff, size)	写指定寄存器的值
read_port_stacked(entries, size)	批量读取指定的寄存器值
write_port_stacked(entries, size)	批量写入指定的寄存器值

◆ **接口说明**

➢ **is_implemented**

声明：

is_implemented(feature_name)

意义：

查询当前功能是否支持

形参：

[in]feature_name 属性字符串

返回值：

支持返回 true,否则返回 false

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_readable**

声明：

is_readable(feature_name)

意义：

查询当前功能是否可读。

形参：

[in]feature_name 属性字符串

返回值：

可读返回 true,否则返回 false

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_writable**

声明：

is_writable(feature_name)

意义：

查询当前功能是否可写。

形参：

[in]feature_name 属性字符串

返回值：

可写返回 true,否则返回 false

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_int_feature**

声明：

get_int_feature(feature_name)

意义：

获取一个整型控制对象。

形参：

[in]feature_name 属性字符串

返回值：

返回整型控制对象，详见 IntFeature_s。

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_enum_feature**

声明：
get_enum_feature(feature_name)
意义：
获取一个枚举型控制对象。
形参：
[in]feature_name 属性字符串
返回值：
返回枚举型控制对象，详见 EnumFeature_s
异常处理：
如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_bool_feature**

声明：
get_bool_feature(feature_name)
意义：
获取一个布尔型控制对象。
形参：
[in]feature_name 属性字符串
返回值：
返回布尔型控制对象，详见 BoolFeature_s
异常处理：
如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_string_feature**

声明：
get_string_feature(feature_name)
意义：
获取一个字符串型控制对象。
形参：
[in]feature_name 属性字符串
返回值：
返回字符串型控制对象，详见 StringFeature_s
异常处理：
如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_command_feature**

声明：
get_command_feature(feature_name)
意义：
获取一个命令型控制对象，详见 CommandFeature_s
形参：
[in]feature_name 属性字符串

返回值：

返回命令型控制对象

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_register_feature**

声明：

get_register_feature(feature_name)

意义：

获取一个寄存器类型控制对象，详见 RegisterFeature_s

形参：

[in]feature_name　　属性字符串

返回值：

返回寄存器类型控制对象

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **feature_save**

声明：

feature_save(file_path)

意义：

保存当前设备配置参数到文本文件。

形参：

[in]file_path　　导出文件路径名

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理

- **feature_load**

声明：

feature_load(file_path, verify=False)

意义：

配置文件中的参数加载到设备。

形参：

[in]file_path　　加载文件路径名

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **read_port**

声明：

read_port(address, size)

意义：

读取相机指定寄存器的值。

注意：

读取到千兆网相机属性值为大端数据。

形参：

[in]address　　属性寄存器地址

[in]size　　要读取属性的字节大小

返回值：

返回指定寄存器的值

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **write_port**

声明：

write_port(address, buff, size)

意义：

设置相机指定寄存器的值。

注意：

1) 千兆网相机设置的属性值需转换为大端后设置。
2) 调用当前接口后，使用 get_enum_feature、get_int_feature 等接口获取到的节点值仍为修改前值，可使用 read_port 或 read_port_stacked 接口获取最新的属性值。

形参：

[in]address　　属性寄存器地址

[out]buff　　要写入的属性缓存

[in]size　　要写入的属性大小

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **read_port_stacked**

声明：

read_port_stacked(entries, size)

意义：

批量读取相机指定的多个寄存器的值，当前仅支持整型（4 字节）。

注意：

获取千兆网相机属性值为大端数据。

形参：

[in]entries　　执向 GxRegisterStackEntry 类型的指针数组

[in]size　　数组大小

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

代码样例：

```
    # 获取属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 结构体个数
```

```
    i = 2

    # 为 buffer 申请内存
    outbuffer = 64
    buffer = c_void_p()
    buffer.value = id(outbuffer)

    outbuffer1 = 1024
    buffer1 = c_void_p()
    buffer1.value = id(outbuffer1)

    # 定义结构体列表
    struct_list = [c_ulonglong(0x00900018), buffer, c_uint(4)]
    struct_list1 = [c_ulonglong(0x00900008), buffer1, c_uint(4)]

    # 结构体初始化
    struct_Reg = GxRegisterStackEntry(*struct_list)
    struct_Reg1 = GxRegisterStackEntry(*struct_list1)

    # 创建多链表
    list = [struct_Reg, struct_Reg1]

    # 定义结构体数组，类型为 GxRegisterStackEntry，长度为 2
    struct_array = GxRegisterStackEntry * 2

    # 结构体数组初始化
    array_instance = struct_array(*list)

    # 转换为指针批量读取寄存器
    struct = pointer(array_instance)
    status = remote_device_feature.read_port_stacked(struct, i)
    assert status == 0

    # 获取内存中的数据
    length = c_int
    width_value = cast(array_instance[0].buffer,
POINTER(length)).contents.value
    width_value1 = cast(array_instance[1].buffer,
POINTER(length)).contents.value
```

- **write_port_stacked**

声明：

write_port_stacked(entries, size)

意义：

批量设置相机指定的多个寄存器的值，当前仅支持整型（4 字节）。

注意：

1) 千兆网相机设置的属性值需转换为大端后设置。
2) 调用当前接口后，使用 GetEnumFeature、GetIntFeature、GetBoolFeature 等接口获取到的节点值仍为修改前值，可使用 ReadPort 或 ReadPortStacked 接口获取最新的属性值。

形参：

[in]entries 执向 GxRegisterStackEntry 类型的指针数组

[in]size 数组大小

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

代码样例：

```
    # 获取属性控制器
remote_device_feature = cam.get_remote_device_feature_control()

    # 结构体个数
    i = 2

    size = 2
    value = c_int(64)
    point = byref(value)
    bufer = cast(point, c_void_p)
    buffer_c = c_void_p()
    buffer_c = bufer

    # 定义结构体列表
    struct_list = [c_ulonglong(0x00900008), buffer_c, c_uint(4)]
    struct_list1 = [c_ulonglong(0x0090000C), buffer_c, c_uint(4)]

    struct_Reg = GxRegisterStackEntry(*struct_list)
    struct_Reg1 = GxRegisterStackEntry(*struct_list1)
    list = [struct_Reg, struct_Reg1]

    struct_array = GxRegisterStackEntry * 2
    array_instance = struct_array(*list)

    struct = pointer(array_instance)

    # 批量写寄存器
    status = remote_device_feature.write_port_stacked(struct, size)
    assert status == 0
    assert status == 0
```

3.4.6. RGBImage(不再维护)

推荐用 ImageProcess

负责 RGB 图像操作。

接口列表：

image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None, channel_order=DxRGBChannelOrder.ORDER_RGB)	图像质量提高
brightness(factor)	图像进行亮度调节
contrast(factor)	图像进行对比度调节
saturation(factor)	图像进行饱和度调节
sharpen(factor)	图像进行锐化处理
get_white_balance_ratio()	获取图像白平衡系数

get_numpy_array()	将 RGB 数据转换为 numpy 对象
get_image_size()	获取 RGB 数据大小

◆ **接口说明**

➢ **image_improvement**

声明：

RGBImage.image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None, channel_order=DxRGBChannelOrder.ORDER_RGB)

意义：

图像质量提高

形参：

[in]contrast_lut 对比度 LUT

[in]gamma_lut gamma LUT

[in]color_collect 颜色校正

[in] channel_order 图像通道顺序，缺省值为 DxRGBChannelOrder.ORDER_RGB，参考 DxRGBChannelOrder

异常处理：

1) 如果参数 1、2 不是 Buffer 类型或 None，则抛异常 ParameterTypeError。
2) 如果参数 3 不是整型或 None，则抛异常 ParameterTypeError。
3) 如果参数 4 不是整型，则抛异常 ParameterTypeError。
4) 如果提高图像质量不成功，则抛出异常 UnexpectedError。
5) 如果参数 1、2、3 都是默认缺省值，则不进行图像质量提升处理，函数退出。

➢ **brightness**

声明：

RGBImage. brightness(factor)

意义：

对 RGB24 图像进行亮度调节

形参：

[in]factor 亮度调节因子，值范围-150~150。

其中： 0：亮度没有变化；

大于 0：增加亮度；

小于 0：减小亮度；

返回值：

None

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果亮度调节失败，则抛出 UnexpetedError 异常。

➢ **contrast**

声明：

RGBImage.contrast(factor)

意义：

对 RGB24 图像进行对比度调节

形参：

[in]factor 对比度调节因子，值范围-50~150。

其中： 0：对比度没有变化；

大于 0：增加对比度；

小于 0：减小对比度；

返回值：

None

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。

2) 如果对比度调节失败，则抛出 UnexpetedError 异常。

- **saturation**

声明：

RGBImage.saturation(factor)

意义：

对 RGB 图像进行饱和度调节

形参：

[in]factor 饱和度调节参数，范围：0 ~ 128，

其中：64：饱和度没有变化；

大于 64：增加饱和度；

小于 64：减小饱和度；

128：饱和度为当前两倍；

0：黑白图像

返回值：

None

异常处理：

1) 如果图像饱和度调节失败，则抛出异常 UnexpectedError。

- **sharpen**

声明：

RGBImage.sharpen()

意义：

对 RGB 图像进行锐化处理

形参：

[in]factor 锐化调节参数，范围：0.1 ~ 5.0

返回值：

None

异常处理：

1) 如果图像锐化处理失败，则抛出异常 UnexpectedError。

- **get_white_balance_ratio**

声明：

RGBImage.get_white_balance_ratio()

意义：

获取白平衡系数

返回值：

返回 RGB 分量系数元组

- **get_numpy_array**

声明：

RGBImage.get_numpy_array()

意义：

获取 numpy 对象

返回值：

numpy 对象

- **get_image_size**

声明：

RGBImage.get_image_size()

意义：

获取 RGB 数据大小

返回值：

RGB 图的大小

3.4.7. RawImage

负责 Raw 图像操作。

接口列表：

defective_pixel_correct()	图像坏点校正
raw8_rotate_90_cw()	将 8 位图像顺时针旋转 90 度
raw8_rotate_90_ccw()	将 8 位图像逆时针旋转 90 度
mirror(mirror_mode)	对 8 位图像进行镜像处理
get_ffc_coefficients(dark_img=None, target_value=None)	计算图像平场校正系数
flat_field_correction(ffc_coefficients)	对图像进行平场校正处理
get_numpy_array()	将 raw 数据转换为 numpy 对象
get_data()	获取 raw 数据
save_raw(file_path)	保存 raw 图数据
get_status()	获取 raw 图状态
get_width()	获取 raw 图宽度
get_height()	获取 raw 图高度
get_pixel_format()	获取图像像素格式
get_image_size()	获取 raw 图数据大小
get_frame_id()	获取帧 ID
get_timestamp()	获取时间戳

convert(mode, flip=False, valid_bits=DxValidBit.BIT4_11, convert_type=DxBayerConvertType.NEIGHBOUR, channel_order=DxRGBChannelOrder.ORDER_RGB)	图像格式转换(不再维护)
brightness(factor)	对 8 位灰度图像进行亮度调节(不再维护)
contrast(factor)	对 8 位灰度图像进行对比度调节(不再维护)

◆ **接口说明**

➢ **defective_pixel_correct**

声明：

RawImage.defective_pixel_correct()

意义：

对 raw 数据进行坏点校正

返回值：

None

异常处理：

1) 如果坏点校正不成功，则抛出异常 UnexpetedError。

➢ **raw8_rotate_90_cw**

声明：

RawImage.raw8_rotate_90_cw()

意义：

对 8 位图像顺时针旋转 90 度

返回值：

旋转后的 RawImage 图像对象

异常处理：

1) 如果旋转失败，则抛出异常 UnexpetedError。

➢ **raw8_rotate_90_ccw**

声明：

RawImage.raw8_rotate_90_ccw()

意义：

对 8 位图像逆时针旋转 90 度

返回值：

旋转后的 RawImage 图像对象

异常处理：

1) 如果旋转失败，则抛出异常 UnexpetedError。

➢ **mirror**

声明：

RawImage.mirror(mirror_mode)

意义：

为 8 位图像产生一个与原图像在水平方向或者垂直方向相对称的镜像图像

形参：

[in]mirror_mode 图像镜像翻转方式，参考 DxImageMirrorMode。

返回值：

镜像后的 RawImage 图像对象

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果不是 8 位图像格式，则抛出 InvalidParameter 异常。
3) 如果图像镜像失败，则抛出 UnexpetedError 异常。

- **get_ffc_coefficients**

声明：

RawImage. get_ffc_coefficients(dark=None, target_value=None)

意义：

计算图像平场校正系数，仅支持 8~12 位 Raw 图

形参：

[in]dark 暗场图像

[in]target_value 期望灰度值

返回值：

平场校正系数

异常处理：

1) 如果 dark 类型不是 RawImage，则抛出 ParameterTypeError 异常。
2) 如果 target_value 不是 INT 类型，则抛出 ParameterTypeError 异常。
3) 如果图像格式不是 8/10/12 位，则抛出 InvalidParameter 异常。
4) 如果获取平场校正系数调用失败，则抛出 UnexpetedError 异常。

- **flat_field_correction**

声明：

RawImage. flat_field_correction(ffc_coefficients)

意义：

为 8~12 位的 Raw 图进行平场校正操作

形参：

[in] ffc_coefficients 平场校正系数。

返回值：

None

异常处理：

1) 如果 ffc_coefficients 不是 Buffer 类型，则抛出 ParameterTypeError 异常。
2) 如果图像格式不是 8/10/12 位图像格式，则抛出 InvalidParameter 异常。
3) 如果图像平场校正失败，则抛出 UnexpetedError 异常。

- **get_numpy_array**

声明：

RawImage.get_numpy_array()

意义：

将 raw 数据转换为 numpy 对象

返回值：

numpy 对象： 成功

None： 失败

异常处理：

1) 如果帧信息状态不成功，则打印错误信息”RawImage.get_numpy_array:This is a incomplete image”，函数返回 None。
2) 如果像素格式不为 8 位或 16 位，则返回 None。

- **get_data**

声明：

RawImage.get_data()

意义：

获取 raw 数据

返回值：

raw 数据[字符串型]

- **save_raw**

声明：

RawImage.save_raw(file_path)

意义：

保存 raw 图数据

形参：

[in]file_path 文件路径。

例如：file_path = ’raw_image.raw’，则将 raw 图保存到当前工程路径下；

file_path = ’E://python_gxiapi/raw_image.raw’，则将 raw 图保存到绝对路径’E://python_gxiapi/’下。

返回值：

None

异常处理：

1) 如果参数不是字符串格式，则抛出异常 ParameterTypeError。
2) 如果保存 raw 图数据未成功，则抛异常 UnexpectedError。

- **get_status**

声明：

RawImage.get_status()

意义：

获取 raw 图状态

返回值：

raw 图状态，数据类型参考 GxFrameStatusList

- **get_width**

声明：

RawImage.get_width()

意义：

获取 raw 图宽度

返回值：

raw 图宽度

- **get_height**

声明：

RawImage.get_height()

意义：

获取 raw 图高度

返回值：

raw 图高度

- **get_pixel_format**

声明：

RawImage.get_pixel_format()

意义：

获取图像像素格式

返回值：

像素格式

- **get_image_size**

声明：

RawImage.get_image_size()

意义：

获取 raw 图数据大小

返回值：

Raw 图的大小

- **get_frame_id**

声明：

RawImage.get_frame_id()

意义：

获取帧 ID

返回值：

帧 ID

- **get_timestamp**

声明：

RawImage.get_timestamp()

意义：

获取时间戳

返回值：

时间戳

- **convert(不再维护)**

- **推荐用 ImageFormatConvert.convert()或 ImageFormatConvert.convert_ex()**

声明：

RawImage.convert(mode, flip=False, valid_bits=DxValidBit.BIT4_11,
convert_type=DxBayerConvertType.NEIGHBOUR,
channel_order=DxRGBChannelOrder.ORDER_RGB)

意义：

图像格式转换。

1) 当 mode = 'RAW8'模式时，将 16 位 raw 图转换为 8 位 raw 图，截取的有效位默认为当前像素格式的高 8 位。用户也可通过参数 valid_bits 手动设置有效位。仅支持 10/12 位的 Raw 图。
2) 当 mode = 'RGB'模式时，将 raw 图转换为 RGB 图。如果输入为 10/12 位 raw 图，先转换为 8 位 raw 图，再转换为 RGB 图。

形参：

[in]mode 'RAW8'：将 16 位 raw 图转换为 8 位 raw 图
'RGB'：将 raw 图转换为 RGB24 图

[in]flip 输出的 RGB 图像是否上下翻转，缺省值为 False，该功能仅支持 mode = 'RGB'模式

[in]valid_bits 有效位数，缺省值为当前像素格式的高 8 位，参考 DxValidBit

[in]convert_type 转换类型，缺省值为 DxBayerConvertType.NEIGHBOUR，参考 DxBayerConvertType，

仅对 mode = 'RGB'模式有效

[in]channel_order 图像通道顺序，缺省值为 DxRGBChannelOrder.ORDER_RGB，参考 DxRGBChannelOrder

返回值：

RGB 图像对象

异常处理：

1) 如果帧信息状态不成功，则打印错误信息" RawImage.convert:This is a incomplete image" ，函数返回 None。
2) 如果参数 1 不是字符串型，则抛异常 ParameterTypeError。
3) 如果参数 2 不是布尔型，则抛异常 ParameterTypeError。
4) 如果参数 3、4 不是整型，则抛异常 ParameterTypeError。
5) 如果参数 4 不在 DxBayerConvertType 中，则打印：提示参数越界、当前参数所支持的枚举值，函数返回 None。
6) 如果参数 4 不在 DxValidBit 中，则打印：提示参数越界、当前参数所支持的枚举值，函数返回 None。
7) 如果像素不是 8/10/12 位，则打印错误信息" RawImage.convert:This pixel format is not support" ，函数返回 None。
8) 如果参数 1 为 'RAW8' 且参数 2 为 True，则打印错误信息"RawImage.convert:mode = 'RAW8' don't support flip = True" ，函数返回 None。

9) mode = ‘RAW8’，位深不是 10/12 位，则打印错误信息”RawImage.convert: mode="RAW8" only support 10bit and 12bit”，函数返回 None。
10) 如果参数 1 不为‘RAW8’或’RGB’，则打印接口名称和输入的 mode 不在范围内的信息，函数返回 None。
44)

- **brightness(不再维护)**

- **推荐用 ImageProcessConfig.set_contrast_param() 设置后用 ImageProcess.image_improvement()处理**

-

声明:
RawImage. brightness(factor)
意义:
对 8 位灰度图像进行亮度调节
形参:
[in]factor 亮度调节因子，值范围-150~150。
其中： 0：亮度没有变化;
大于 0：增加亮度;
小于 0：减小亮度;
返回值:
None
异常处理:
1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果图像类型不是 Mono8，则打印“RawImage.brightness only support mono8 image”，抛出 InvalidParameter 异常。
3) 如果亮度调节失败，则抛出 UnexpetedError 异常。

- **contrast(不再维护)**

- **推荐用 ImageProcessConfig.set_lightness_param() 设置后 ImageProcess.image_improvement()处理**

声明:
RawImage.contrast(factor)
意义:
对 8 位灰度图像进行对比度调节
形参:
[in]factor 对比度调节因子，值范围-50~150。
其中： 0：对比度没有变化;
大于 0：增加对比度;
小于 0：减小对比度;
返回值:

None

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果图像类型不是 MONO8，则打印“RawImage.brightness only support mono8 image”，抛出 InvalidParameter 异常。
3) 如果对比度调节失败，则抛出 UnexpetedError 异常。

45)

3.4.8. Buffer

负责 Buffer 类的操作。Buffer 类将在图像质量提升的部分使用，Utility.get_gamma_lut(gamma)和 Utility.get_contrast_lut(contrast)接口返回的 Buffer 类型对象将作为参数传给 RGBImage.image_improvement(color_correction_param=0, contrast_lut=None, gamma_lut=None)接口。

接口列表：

from_file(file_name)	从文件获取 Buffer 对象
from_string(string_data)	从字符串获取 Buffer 对象
get_data()	返回 Buffer 对象的字符串数据
get_ctype_array()	返回 Buffer 对象的数据数组
get_numpy_array()	返回 Buffer 对象的 numpy 数组
get_length()	返回 Buffer 对象的数据数组长度

◆ **接口说明**

➢ **from_file（静态函数）**

声明：

Buffer.from_file(file_name)

意义：

从文件获取 Buffer 对象

形参：

[in]file_name 文件路径

返回值：

Buffer 对象

➢ **from_string（静态函数）**

声明：

Buffer.from_string(string_data)

意义：

从字符串获取 Buffer 对象

形参：

[in]string_data 字符串

返回值：

Buffer 对象

- **get_data**

声明：
Buffer.get_data()
意义：
返回 Buffer 对象的字符串数据
返回值：
string_data 字符串数据
注：python2.7：返回字符串类型；python3.5：返回 bytes 类型

- **get_ctype_array**

声明：
Buffer.get_ctype_array()
意义：
返回 Buffer 对象的数据数组
返回值：
Buffer 对象的数据数组[ctype 类型]

- **get_numpy_array**

声明：
Buffer.get_numpy_array()
意义：
返回 Buffer 对象的 numpy 数组
返回值：
Buffer 对象的数据数组[numpy 类型]

- **get_length**

声明：
Buffer.get_length()
意义：
返回 Buffer 对象的数据数组长度
返回值：
数据数组长度

3.4.9. ImageProcessConfig

负责图像处理属性配置。
接口列表：

set_valid_bits(valid_bits)	选择获取指定 8 位有效数据位，此接口设置指定哪 8 位
get_valid_bits()	查询当前指定的哪 8 位有效位
enable_defective_pixel_correct(enable)	使能坏点校正
is_defective_pixel_correct()	查询当前是否使能坏点校正
enable_sharpen(enable)	使能锐化
is_sharpen()	查询当前是否使能锐化

set_sharpen_param(param)	设置锐化强度因子
get_sharpen_param()	查询当前使用的锐化强度因子
set_contrast_param(param):	设置对比度调节参数
get_contrast_param()	查询当前使用的对比度调节参数
set_gamma_param(param)	设置 Gamma 系数
get_gamma_param()	获取 Gamma 系数
set_lightness_param(param)	设置亮度调节参数
get_lightness_param()	获取亮度调节参数
enable_denoise(enable)	使能降噪
is_denoise()	查询当前是否使能降噪
set_saturation_param(param)	设置饱和度系数
get_saturation_param()	获取饱和度系数
set_convert_type(cv_type)	设置图像格式转换算法
get_convert_type()	获取图像格式转换算法
enable_convert_flip(flip)	使能图像格式转换翻转
is_convert_flip()	查询当前是否使能图像格式转换翻转
enable_accelerate(accelerate)	使能加速图像处理
is_accelerate()	查询当前是否工作在加速使能状态
enable_color_correction(enable)	使能颜色校正
is_color_correction()	查询是否使能颜色校正
enable_user_set_ccparam(enable)	使能颜色校正用户设置模式
is_user_set_ccparam()	查询颜色校正是否用户设置模式
set_user_ccparam(color_transform_factor)	设置颜色转换因子结构体
get_user_ccparam()	获取颜色转换因子结构体
reset()	恢复默认调节参数

◆ **接口说明**

➢ **set_valid_bits**

声明：

set_valid_bits(valid_bits)

意义：

选择获取指定 8 位有效数据位，此接口针对非 8 位原始数据设立。

形参：

[in]param　　有效位数，缺省值为当前像素格式的高 8 位，参考 DxValidBit

异常处理：

46) 如果设置有效位不成功，则抛出异常，异常类型详见错误处理。

47)

➢ **get_valid_bits()**

声明：

get_valid_bits()
意义：
获取指定 8 位有效数据位，此接口针对非 8 位原始数据设立。
返回值：
有效数据位，详见 DxValidBit

- **enable_defective_pixel_correct**

声明：
enable_defective_pixel_correct(enable)
意义：
使能坏点校正。bEnable 为 true 则使能坏点校正；为 false 则禁用坏点校正。
形参：
[in]enable　　使能坏点校正。enable 为 true 则使能坏点校正；为 false 则禁用坏点校正。
异常处理：

48) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_defective_pixel_correct**

声明：
is_defective_pixel_correct()
意义：
获取坏点校正状态。
返回值：

49) 坏点校正状态

- **enable_sharpen**

声明：
enable_sharpen(enable)
意义：
使能锐化。bEnable 为 true 则使能锐化；为 false 则禁用锐化。
形参：
[in]enable　　使能锐化。bEnable 为 true 则使能锐化；为 false 则禁用锐化。
异常处理：

50) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_sharpen**

声明：
is_sharpen()
意义：
查询当前是否使能锐化。
返回值：

51) 使能锐化状态。

- **set_sharpen_param**

声明：
set_sharpen_param(param)
意义：
设置锐化强度因子
形参：
[in]param 锐化强度因子。
异常处理：
1) 如果输入参数不是浮点型，则抛出 ParameterTypeError 异常。
2) 如果输入参数范围不在 0.1~5.0，则抛出 UnexpectedError 异常。

- **get_sharpen_param**

声明：
get_sharpen_param()
意义：
查询当前使用的锐化强度因子。
返回值：
52) 返回锐化强度因子。

- **set_contrast_param**

声明：
set_contrast_param(param):
意义：
设置对比度调节参数。
形参：
[in]param 对比度调节参数。
异常处理：
1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果输入参数范围不在-50~100，则抛出 UnexpectedError 异常。

- **get_contrast_param**

声明：
get_contrast_param()
意义：
查询当前使用的对比度调节参数。
返回值：
53) 返回对比度调节参数。

- **set_gamma_param**

声明：
set_gamma_param(param)
意义：
设置 Gamma 系数。
形参：

[in]param　　Gamma 系数。

异常处理：

1) 如果输入参数不是整型、浮点型，则抛出 ParameterTypeError 异常。

2) 如果输入参数范围不在-0.1~10.0，则抛出 UnexpectedError 异常。

- **get_gamma_param**

声明：

get_gamma_param()

意义：

获取 Gamma 系数。

返回值：

54) 返回 Gamma 系数。

- **set_lightness_param**

声明：

set_lightness_param(param)

意义：

设置亮度调节参数。

形参：

[in]param　　亮度调节参数

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。

2) 如果输入参数范围不在-150~150，则抛出 UnexpectedError 异常

- **get_lightness_param**

声明：

get_lightness_param()

意义：

获取亮度调节参数。

返回值：

55) 返回亮度调节参数。

- **enable_denoise**

声明：

enable_denoise(enable)

意义：

使能降噪。

形参：

[in]param　　使能降噪开关（黑白相机不支持）。enable 为 true 则使能降噪功能；为 false 则禁用降噪功能。

异常处理：

56) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_denoise()**

声明：
is_denoise()
意义：
获取降噪开关（黑白相机不支持）。
返回值：
57) 返回降噪开关标志。

- **set_saturation_param**

声明：
set_saturation_param(param)
意义：
设置饱和度调节系数（黑白相机不支持）。
形参：
[in]param 饱和度调节系数。
异常处理：
1) 如果输入参数不是整型、浮点型，则抛出 ParameterTypeError 异常。
2) 如果输入参数范围不在 0~128，则抛出 UnexpectedError 异常。

- **get_saturation_param**

声明：
get_saturation_param()
意义：
获取饱和度调节系数（黑白相机不支持）。
返回值：
58) 返回饱和度调节系数。

- **set_convert_type**

声明：
set_convert_type(cv_type)
意义：
设置图像插值处理算法（黑白相机不支持）。
形参：
[in]cv_type 转换类型，缺省值为 DxBayerConvertType.NEIGHBOUR，参考 DxBayerConvertType
异常处理：
59) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_convert_type**

声明：
get_convert_type()
意义：
获取图像插值处理算法（黑白相机不支持）。
返回值：
获取图像插值处理算法（黑白相机不支持），详见 DxBayerConvertType

- **enable_convert_flip**

声明：

enable_convert_flip(flip)

意义：

使能插值翻转（黑白相机不支持）。

形参：

[in]flip　　flip 为 true 则使能翻转；为 false 则禁用翻转功能。

异常处理：

60) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_convert_flip**

声明：

is_convert_flip()

意义：

获取插值翻转标识（黑白相机不支持），true 表示翻转，false 表示不翻转。

返回值：

61) 返回插值翻转标识。

- **enable_accelerate**

声明：

enable_accelerate(accelerate)

意义：

图像处理加速使能。设置 ture 为使能加速，为 false 禁用加速。

形参：

[in]accelerate　　accelerate 为 true 则使能加速；为 false 则禁用加速功能。

异常处理：

62) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_accelerate**

声明：

is_accelerate()

意义：

查询当前是否工作在加速使能状态，true 表示加速，false 表示不加速。

返回值：

63) 加速使能状态。

64)

- **enable_color_correction**

声明：

enable_color_correction(enable)

意义：

使能颜色校正（黑白相机不支持），true 表示使能颜色校正，false 表示禁用颜色校正。

形参：

[in]enable　　enable 为 true 则使能颜色校正；为 false 则禁用颜色校正功能。

异常处理：

65) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_color_correction**

声明：

is_color_correction()

意义：

查询是否使能颜色校正（黑白相机不支持）。

返回值：

66) 使能颜色校正状态。

- **enable_user_set_ccparam**

声明：

enable_user_set_ccparam(enable)

意义：

使能颜色校正用户设置模式（黑白相机不支持）， true 表示用户模式， false 表示非用户模式。

形参：

[in]enable enable 为 true 则使能颜色校正用户设置模式；为 false 则禁用颜色校正用户设置模式功能。

异常处理：

67) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **is_user_set_ccparam**

声明：

is_user_set_ccparam()

意义：

查询颜色校正是否用户设置模式（黑白相机不支持）， true 表示用户模式， false 表示非用户模式。

返回值：

68) 校正是否用户设置模式。

- **set_user_ccparam**

声明：

set_user_ccparam(color_transform_factor)

意义：

设置颜色转换因子结构体（颜色校正结构体参数建议范围-4~4，如果参数设置小于-4，校正效果与-4一致，如果参数设置大于 4，校正效果与 4 一致）。

[in]color_transform_factor 设置颜色转换因子结构体。

异常处理：

69) 如果输入参数不是 ColorTransformFactor 型，则抛出 ParameterTypeError 异常。

- **get_user_ccparam**

声明：

get_user_ccparam()

意义：

获取颜色转换因子结构体。

返回值：

70) 转换因子结构体。

➢ **reset**

声明：

reset()

意义：

恢复最佳默认参数配置

71)

3.4.10. Utility

负责参数 gamma 和 contrast 的操作。

接口列表：

get_gamma_lut(gamma=1)	通过 gamma 值获取 gamma 查找表的 Buffer 类型对象
get_contrast_lut(contrast=0)	通过对比度值获取对比度查找表的 Buffer 类型对象
get_lut(contrast=0, gamma=1, lightness=0)	计算图像处理 8 位查找表
calc_cc_param(color_correction_param, saturation=64)	计算图像处理色彩调节数组
calc_user_set_cc_param(color_transform_factor, saturation=64)	根据用户设置计算图像处理色彩调节数组

◆ **接口说明**

➢ **get_gamma_lut（静态函数）**

声明：

Utility.get_gamma_lut(gamma=1)（静态函数）

意义：

通过 gamma 值获取 gamma 查找表的 Buffer 类型对象

形参：

[in]gamma　整型或浮点型，范围[0.1, 10.0]，缺省值为 1

返回值：

gamma 查找表的 Buffer 类型对象

异常处理：

1) 如果输入参数不是整型或浮点型，则抛出 ParameterTypeError 异常。
2) 如果输入参数不在 0.1~10.0 范围内，则打印错误信息”Utility.get_gamma_lut:gamma out of bounds, range:[0.1, 10.0]”，函数返回 None。
3) 如果获取 gamma 查找表失败，则打印接口名称、获取 gamma lut 失败和错误码的信息，函数返回 None。

➢ **get_contrast_lut（静态函数）**

声明：

Utility.get_contrast_lut(contrast=0)（静态函数）

意义：

通过对比度值获取对比度查找表的 Buffer 类型对象

形参：

[in]contrast　整型，范围[-50, 100]，缺省值为 0

返回值：

对比度查找表的 Buffer 类型对象

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果输入参数不在-50~100 范围内，则打印错误信息”Utility.get_contrast_lut:contrast out of bounds, range:[-50, 100]”，函数返回 None。
3) 如果获取对比度查找表失败，则打印接口名称、获取 constrast lut 失败和错误码的信息，函数返回 None。

- **get_lut（静态函数）**

声明：

Utility.get_lut(contrast=0, gamma=1, lightness=0)（静态函数）

意义：

计算图像处理 8 位查找表

形参：

[in]contrast　对比度调节参数，整型，范围[-50, 100]，缺省值为 0

[in]gamma　调节参数，浮点型，范围[0.1, 10]，缺省值为 1

[in]lightness　亮度调节参数，整型，范围[-150, 150]，缺省值为 0

返回值：

查找表的 Buffer 类型对象

异常处理：

1) 如果 contrast/gamma/lightness 不是整型/浮点型/整型，则抛出 ParameterTypeError 异常。
2) 如果获取查找表失败，则打印接口名称、获取 lut 失败和错误码的信息，函数返回 None。

- **calc_cc_param（静态函数）**

声明：

Utility.calc_cc_param(color_correction_param, saturation=64)（静态函数）

意义：

计算图像处理色彩调节数组

形参：

[in]color_correction_param　颜色校正值，整型，可以从相机获取或者设为 0

[in]saturation　饱和度调节参数，整型，范围[0, 128]，缺省值为 64

返回值：

色彩调节参数的 Buffer 类型对象

异常处理：

1) 如果输入参数不是整型，则抛出 ParameterTypeError 异常。
2) 如果计算色彩调节参数失败，则打印接口名称、获取调节参数失败和错误码的信息，函数返回 None。

➢ **calc_user_set_cc_param（静态函数）**

声明：

Utility.calc_user_set_cc_param(color_transform_factor, saturation=64)（静态函数）

意义：

根据用户设置计算图像处理色彩调节数组

形参：

[in]color_transform_factor 颜色校正/颜色转换数组，列表或者元组类型

[in]saturation 饱和度调节参数，整型，范围[0, 128]，缺省值为 64

返回值：

色彩调节参数的 Buffer 类型对象

异常处理：

1) 如果输入参数不是列表（元组）、整型，则抛出 ParameterTypeError 异常。
2) 如果计算色彩调节参数失败，则打印接口名称、获取调节参数失败和错误码的信息，函数返回 None

3.4.11. ImageProcess

对当前图像做图像效果增强，返回效果增强之后的图像数据

注意：

1) 如果 ImageProcessConfig.is_accelerate()返回 true，说明此时将以加速方式处理图像，加速方式处理图像需要图像的高度必须是 4 的整倍数，否则接口报错；
2) 黑白相机 void*返回的是 8bit 图像数据，图像大小为图像宽 x 图像高；
3) 彩色相机 void*返回的是 RGB 图像数据，图像大小为图像宽 x 图像高 x3。

接口列表：

image_improvement(image, output_address, image_process_config)	图像质量提升处理
static_defect_correction(input_address, output_address, defect_correction, defect_pos_buffer_address, defect_pos_buffer_size)	图像静态坏点矫正
calcula_lut(contrast_param, gamma, light_ness, lut_address, lut_length_address)	计算相机查找表
read_lut_file(lut_file_path, lut_address, lut_length_address)	读取查找表文件

◆ **接口说明**

➢ **image_improvement**

声明：

image_improvement(image, output_address, image_process_config)

意义：

对输入的图像做图像质量提升，注：Bayer 格式的图片图像质量提升处理之后的格式为 RGB。

形参：

[in]image 源图像（可变类型），

1）类型为 RawImage，可通过 DataStream.get_image()获得

2）类型为 GxImageInfo

[out]output_address 返回图像质量提升后的图像 Buffer

[in]image_process_config 图像质量提升辅助配置对象 可控制图像锐化、亮度等信息。

异常处理：

72）如果调用不成功，则抛出异常，异常类型详见错误处理。

- **static_defect_correction**

声明：

static_defect_correction(input_address, output_address, defect_correction, defect_pos_buffer_address, defect_pos_buffer_size)

意义：

对输入的图像做静态坏点矫正。

形参：

[in]input_address 要矫正的原始图片缓存地址

[out]output_address 存放输出图像缓存地址

[in]defect_correction 图像静态坏点矫正参数

[in]defect_pos_buffer_address 检测静态坏点文件 buffer

[in]defect_pos_buffer_size 检测静态坏点文件 buffer 位置

异常处理：

73）如果调用不成功，则抛出异常，异常类型详见错误处理。

- **calcula_lut**

声明：

calcula_lut(contrast_param, gamma, light_ness, lut_address, lut_length_address)

意义：

计算查找表。

形参：

[in]contrast_param contrast 参数

[in]gamma gamma 参数

[in]light_ness 亮度

[out]lut_address 查找表地址

[out]lut_length_address 查找表长度地址

异常处理：

74）如果调用不成功，则抛出异常，异常类型详见错误处理。

- **read_lut_file**

声明：

read_lut_file(lut_file_path, lut_address, lut_length_address)

意义：

从文件读取查找表到程序内存。

形参：

[in]lut_file_path	查找表文件路径
[out]lut_address	要存放的查找表地址
[out]lut_length_address	查找表长度

异常处理：

75) 如果调用不成功，则抛出异常，异常类型详见错误处理。

3.4.12. ImageFormatConvert

负责图像格式转换，可通过此对象转换图像格式。

接口列表：

set_dest_format(dest_pixel_format)	设置目标像素格式
get_dest_format()	获取目标像素格式
set_interpolation_type(cvt_type)	设置转换算法
get_interpolation_type()	获取转换算法
set_alpha_value(alpha_value)	设置 Alpha 通道
get_alpha_value()	获取 Alpha 通道
set_valid_bits(valid_bits)	设置要转换像素格的有效位
get_valid_bits()	获取要转换像素格的有效位
get_buffer_size_for_conversion_ex(width, height, pixel_format)	获取转换像素格式对应的 Buffer 的大小
get_buffer_size_for_conversion(raw_image)	获取转换像素格式对应的 Buffer 的大小
convert_ex(input_address, input_width, input_height, src_fixel_format, output_address, output_length, flip)	转换像素格式
convert(raw_image, output_address, output_length, flip)	转换像素格式
set_3d_calib_param(buffer_address, buffer_size)	该函数用于加载 3D 标定参数，将高度图转换为点云图
get_3d_calib_param()	获取 3D 标定参数
set_y_step(y_step)	设置高度图转点云的 Y 方向步长
get_y_step()	获取高度转点云的 Y 方向步长

◆ **接口说明**

➢ **set_dest_format**

声明：

set_dest_format(dest_pixel_format)

意义：

设置期望转换格式，详见 GxPixelFormatEntry

形参：

[in]dest_pixel_format　　期望的转换格式，详见 GxPixelFormatEntry 支持转换类型

异常处理：

76) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_dest_format**

声明：

get_dest_format()

意义：

获取期望转换格式。

返回值：

返回期望转换格，详见 GxPixelFormatEntry

异常处理：

77) 如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **set_interpolation_type**

声明：

set_interpolation_type(cvt_type)

意义：

设置图像格式转换算法。

形参：

[in]cvt_type　　转换算法，详见 DxBayerConvertType

默认值为：DxBayerConvertType.NEIGHBOUR

异常处理：

78) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_interpolation_type**

声明：

get_interpolation_type()

意义：

获取图像格式转换算法。

返回值：

返回图像格式转换算法，详见 DxBayerConvertType

- **set_alpha_value**

声明：

set_alpha_value(alpha_value)

意义：

设置带有 Alpha 通道图像的 Alpha 值。

形参：

[in]alpha_value　　Alpha 通道值，取值范围 0~255，默认值为：255。

异常处理：

79) 如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_alpha_value**

声明：

get_alpha_value()

意义：

获取图像格式转换算法。

返回值：

获取 Alpha 通道值。

- **set_valid_bits**

声明：

set_valid_bits(valid_bits)

意义：

设置转换有效位。

形参：

[in]valid_bits　　有效位数，缺省值为当前像素格式的高 8 位，参考 DxValidBit

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **get_valid_bits**

声明：

get_valid_bits()

意义：

获取转换有效位。

返回值：

有效位，详见 DxValidBit

- **get_buffer_size_for_conversion_ex**

声明：

get_buffer_size_for_conversion_ex(width, height, pixel_format)

意义：

根据输入图像数据获取期望格式的图像 Buffer 长度。

形参：

[in]width　　源图像宽。

[in]height　　源图像高。

[in]pixel_format　　源图像格式。

返回值：

期望格式的图像 Buffer 长度。

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_buffer_size_for_conversion**

声明：

get_buffer_size_for_conversion(raw_image)

意义：

根据输入图像指针获取期望格式的图像 Buffer 长度。

形参：

[in]raw_image 源图像，可通过 DataStream.get_image()获得

返回值：

期望格式的图像 Buffer 长度。

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **convert_ex**

声明：

convert_ex(input_address, input_width, input_height, src_fixel_format, output_address, output_length, flip)

意义：

将输入源图像转换为期望的图像格式，可转换类型参考支持转换类型。

形参：

[in]input_address 源图像 Buffer 指针

[in]input_width 源图像宽度

[in]input_height 源图像高度

[in]src_fixel_format 源图像格式

[out]output_address 目的图像 Buffer 指针，该内存由用户申请，长度为 output_length

[in]output_length 目的图像 Buffer 长度，可通过 get_buffer_size_for_conversion_ex()获得

[in]flip 图像是否翻转（true-翻转，false-不翻转）

异常处理：

如果函数调用不成功，则抛出异常，异常类型详见错误处理。

- **convert**

声明：

convert(raw_image, output_address, output_length, flip)

意义：

将输入源图像转换为期望的图像格式，可转换类型参考支持转换类型。

形参：

[in]raw_image 源图像，可通过 DataStream.get_image()获得

[in]output_address 目的图像 Buffer 指针，该内存由用户申请，长度为 output_length

[in]output_length 目的图像 Buffer 长度，可通过 get_buffer_size_for_conversion()获得

[in]flip 图像是否翻转（true-翻转，false-不翻转）

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **set_3d_calib_param**

声明：set_3d_calib_param(self, buffer_address, buffer_size)

意义：该函数用于加载 3D 标定参数，将高度图转换为点云图

形参：

[in]pCalibParamBuffer　　3D 标定参数内存

[in]参数 nBufferSize　　3D 标定参数内存长度

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_3d_calib_param**

声明：get_3d_calib_param()

意义：获取 3D 标定参数

形参：无

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **set_y_step**

声明：set_y_step(y_step)

意义：设置高度图转点云的 Y 方向步长

形参：

[in]参数 dY　　Y 方向步长

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

- **get_y_step**

声明：get_y_step()

意义：获取高度转点云的 Y 方向步长

形参：无

异常处理：

如果调用不成功，则抛出异常，异常类型详见错误处理。

4. 常见问题解答

序号	常见问题	解决办法
1	程序运行中出现如下错误”NotInitApi: DeviceManager.update_device_list:{-13}{Not init API}”	1) 请检查并删除程序中调用 DeviceManager 类对象的__del__()函数的语句。因为 Python 的垃圾回收机制会自动调用__del__()函数销毁对象，所以不需要、不允许用户显示调用__del__()函数，如：”device_manager.__del__()”。

5. 版本说明

序号	修订版本号	所做改动	发布日期
1	V1.0.0	初始发布	2018-08-10
2	V1.0.1	添加水星二代相机新增功能说明	2018-10-31
3	V1.0.2	修改部分标题，更正了部分不准确的描述	2019-04-12
4	V1.0.3	补充了部分描述	2019-05-07
5	V1.0.4	添加了掉线回调和采集回调的注册和注销接口，同步新的功能码和对应的数据类型	2021-03-04
6	V1.0.5	修改部分描述的问题，去掉不准确的描述	2021-03-08
7	V1.0.6	添加设备重连接口以及调节亮度、镜像等图像处理库接口	2021-05-27
8	V1.0.7	1. 修改 2.2.2 节，更新新增枚举接口。 2. 修改 2.2.5 节，更新图像格式转换接口，图像质量提升接口。 3. 修改 2.2.6 节，更新属性读写查询为新接口。 4. 修改 2.2.7 节，更新导入导出相机配置为新接口。 5. 新增 3.2.1 节，基础数据类型读写说明。 6. 修改 3.3.1 节，更新增加 cxp 类型 7. 修改 3.3.4 节，更新增加像素格式 MONO10P, MONO12P, R8,B8,G8,RGB8,BGR8 8. 修改 3.3.58 节，新增有效位。 9. 新增 3.3.61 节，TL 类型说明 10.新增 3.3.62，图像处理辅助结构说明 11.修改 3.4.1，增加 get_interface_number、get_interface_info、get_interface、gige_force_ip、gige_ip_configuration、create_image_format_convert、create_image_process 接口。 12.修改 3.4.2 节，增加 get_stream_channel_num、get_stream、get_local_device_feature_control、get_remote_device_feature_control、register_device_feature_callback、register_device_feature_callback_by_string、unregister_device_feature_callback、unregister_device_feature_callback_by_string、read_remote_device_port、write_remote_device_port、read_remote_device_port_stacked、write_remote_device_port_stacked、create_image_process_config 接口。	2023-03-12

		13.修改 3.4.3 节，增加 get_featrue_control、get_payload_size、set_payload_size 接口。 14.修改 3.4.7 节，增加 is_gray 接口。 15.新增 3.4.8，3.4.9 节，图像处理相关说明。 16.新增 3.4.10 节，图像格式转换相关说明。 17.新增 3.4.11，Interface 相关说明。 18.新增 3.4.12，查询读写各节点属性相关说明。	
9	V1.0.8	1. 扩展定义 GxPixelFormatEntry 像素格式 2. 新增接口 set_3d_calib_param、get_3d_calib_param、set_y_step、get_y_step	2024-04-03