SDK 属性说明手册

Python

文档版本：V1.0

发布时间：2025.04.11

上海图漾信息科技有限公司

SHANGHAI PERCIPIO TECHNOLOGY LIMITED

声明

版权声明

商标声明

责任声明

1. 在现行法律许可的情况下，本手册仅基于产品目前的现状，对产品将来是否适销、品质是否良好、是否侵犯他人产品的权益、是否适用等问题不做任何形式的声明与保证。
2. 在将来任何情况下，对使用本手册所造成的任何损失和伤害（包括但不限于直接损失、间接损失、特别损失、附随损失或惩罚性赔偿），图漾将不承担责任，即使这些损失和损害是可以预见的，或图漾曾被告知将有可能造成这些损失。
3. 图漾保证本产品符合注明的质量标准，并在质保期内承担产品的质保责任。但本产品只能用作指定用途，将产品挪作它用而造成的损失，图漾不承担任何责任。
4. 如本手册所涉数据可能因环境等因素而产生差异，图漾不承担由此产生的后果。

手册声明

本手册仅作为相关产品的指导说明，可能与实际产品存在差异，请以实物为准。因产品版本升级或其他需要，图漾可能会对本手册进行更新。

除本手册外，图漾还提供在线文档，供用户参考：doc.percipio.xyz/cam/latest/index.html。

目录

Python 属性读写说明

TY_BOOL_AUTO_EXPOSURE

```
param = cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_EXPOSURE, param)

status =cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_EXPOSURE)

m_status=status.toBool()
```

TY_BOOL_GVSP_RESEND

```
param =  cl.DevParamFromBool(True)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_GVSP_RESEND, param)

status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_GVSP_RESEND)

m_status=status.toBool()
```

TY_INT_ACCEPTABLE_PERCENT

```
para =cl.DevParamFromInt(80)

cl.DeviceSetParameter(handle,TY_COMPONENT_DEVICE,TY_INT_ACCEPTABLE_PERCENT,para)

value = cl.DeviceGetParameter(handle,TY_COMPONENT_DEVICE,TY_INT_ACCEPTABLE_PERCENT)

m_value = value.toInt()

value_min = value.mMin()

value_max = value.mMax()

value_inc = value.mInc()

print('min {} max {} inc {} cunrrent {}'.format(value_min,value_max,value_inc,m_value))
```

TY_INT_NTP_SERVER_IP

```
ip = cl.IPv4StringToInt('0.0.0.0')

param =  cl.DevParamFromInt(ip)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_NTP_SERVER_IP, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_NTP_SERVER_IP)

read_param_m  = read_param.toInt()

print('ip',read_param_m)
```

TY_INT_PACKET_SIZE

```
param =  cl.DevParamFromInt(1500)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PACKET_SIZE, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PACKET_SIZE)

m_read_param=read_param.toInt()
```

TY_ENUM_IMAGE_MODE

```
para =cl.DevParamFromEnum(TY_PIXEL_FORMAT_DEPTH16 | TY_RESOLUTION_MODE_1280x960)

cl.DeviceSetParameter(handle,TY_COMPONENT_DEPTH_CAM,TY_ENUM_IMAGE_MODE,para)

value = cl.DeviceGetParameter(handle,TY_COMPONENT_DEPTH_CAM,TY_ENUM_IMAGE_MODE)

m_value = value.toEnum()

print('current value {}'.format(m_value))

value_list = value.eList()

for i in range (len(value_list)):

  print('description {}  , value {}'.format(value_list[i].description,value_list[i].value))
```

TY_FLOAT_SCALE_UNIT

```
param =  cl.DevParamFromFloat(0.0125)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_FLOAT_SCALE_UNIT, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_FLOAT_SCALE_UNIT)

m_read_param=read_param.toFloat()
```

TY_ENUM_TRIGGER_POL

```
para =cl.DevParamFromEnum(TY_TRIGGER_POL_FALLINGEDGE)

cl.DeviceSetParameter(handle,TY_COMPONENT_DEVICE,TY_ENUM_TRIGGER_POL,para)

value = cl.DeviceGetParameter(handle,TY_COMPONENT_DEVICE,TY_ENUM_TRIGGER_POL)

m_value = value.toEnum()

print('current value {}'.format(m_value))

value_list = value.eList()

for i in range (len(value_list)):

  print(' {}'.format(value_list[i].value))
```

TY_INT_FRAME_PER_TRIGGER

```
param =  cl.DevParamFromInt(1)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_FRAME_PER_TRIGGER, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_FRAME_PER_TRIGGER)
m_read_param=read_param.toInt()
```

TY_BOOL_KEEP_ALIVE_ONOFF

```
    param = cl.DevParamFromBool(True)
    cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_KEEP_ALIVE_ONOFF, param)
    status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_KEEP_ALIVE_ONOFF)
    m_status=status.toBool()
    print('TY_BOOL_KEEP_ALIVE_ONOFF status {}'.format(m_status))
```

TY_INT_KEEP_ALIVE_TIMEOUT

```
param =  cl.DevParamFromInt(3000)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_KEEP_ALIVE_TIMEOUT, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_KEEP_ALIVE_TIMEOUT)
m_read_param=read_param.toInt()
```

TY_INT_PACKET_DELAY

```
param =  cl.DevParamFromInt(0)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PACKET_DELAY, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PACKET_DELAY)
m_read_param=read_param.toInt()
```

TY_BOOL_CMOS_SYNC

```
param =  cl.DevParamFromBool(False)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_CMOS_SYNC, param)
status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_CMOS_SYNC)
m_status=status.toBool()
```

TY_ENUM_STREAM_ASYNC

```
para =cl.DevParamFromEnum(TY_STREAM_ASYNC_ALL)
cl.DeviceSetParameter(handle,TY_COMPONENT_DEVICE,TY_ENUM_STREAM_ASYNC,para)
value = cl.DeviceGetParameter(handle,TY_COMPONENT_DEVICE,TY_ENUM_STREAM_ASYNC)
m_value = value.toEnum()
print('current value {}'.format(m_value))
value_list = value.eList()
for i in range (len(value_list)):
  print('description {} , value {}'.format(value_list[i].description,value_list[i].value))
```

TY_INT_CAPTURE_TIME_US

```
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_CAPTURE_TIME_US)
m_read_param=read_param.toInt()
```

TY_ENUM_TIME_SYNC_TYPE

```
param =  cl.DevParamFromEnum(TY_TIME_SYNC_TYPE_HOST)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE)
m_read_param=read_param.toEnum()
value_list = read_param.eList()
for i in range (len(value_list)):
  print('description {} , value {}'.format(value_list[i].description,value_list[i].value))
```

TY_BOOL_TIME_SYNC_READY

```
status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY)
m_status=status.toBool()
```

TY_INT_EXPOSURE_TIME

```
param =  cl.DevParamFromInt(4096)
cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_EXPOSURE_TIME, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_EXPOSURE_TIME)
m_read_param=read_param.toInt()
```

TY_INT_GAIN

```
para =cl.DevParamFromInt(80)

cl.DeviceSetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_INT_GAIN,para)

value = cl.DeviceGetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_INT_GAIN)

m_value = value.toInt()

value_min = value.mMin()

value_max = value.mMax()

value_inc = value.mInc()

print('min {} max {} inc {} cunrrent {}'.format(value_min,value_max,value_inc,m_value))
```

TY_BOOL_AUTO_GAIN

```
param = cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_GAIN, param)

status =cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_GAIN)

m_status=status.toBool()
```

TY_INT_TOF_HDR_RATIO

```
param =  cl.DevParamFromInt(100)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_HDR_RATIO, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_HDR_RATIO)

m_read_param=read_param.toInt()
```

TY_INT_TOF_JITTER_THRESHOLD

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_JITTER_THRESHOLD, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_JITTER_THRESHOLD)

m_read_param=read_param.toInt()
```

TY_BOOL_LASER_AUTO_CTRL

```
param=cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_LASER_AUTO_CTRL, param)

status =cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_LASER_AUTO_CTRL)
```

```
m_status=status.toBool()
```

TY_INT_LASER_POWER

```
param =  cl.DevParamFromInt(100)

cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_INT_LASER_POWER, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_INT_LASER_POWER)

m_read_param=read_param.toInt()
```

TY_BOOL_AUTO_AWB

```
param =cl.DevParamFromBool(True)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_AWB, param)

status = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_AWB)

m_status=status.toBool()
```

TY_INT_R_GAIN

```
param =  cl.DevParamFromInt(4096)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_R_GAIN, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_R_GAIN)

m_read_param=read_param.toInt()
```

TY_INT_G_GAIN

```
param =  cl.DevParamFromInt(4096)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_G_GAIN, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_G_GAIN)

m_read_param=read_param.toInt()
```

TY_INT_B_GAIN

```
param =  cl.DevParamFromInt(4096)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_B_GAIN, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_B_GAIN)

m_read_param=read_param.toInt()
```

TY_INT_ANALOG_GAIN

```
param =  cl.DevParamFromInt(4096)

cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_ANALOG_GAIN, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_ANALOG_GAIN)

m_read_param=read_param.toInt()
```

TY_BOOL_HDR

```
param =  cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_HDR, param)

status = cl.DeviceGetParameter(handle, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_HDR)

m_status=status.toBool()
```

TY_INT_SGBM_IMAGE_NUM

```
param =  cl.DevParamFromInt(3)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_IMAGE_NUM, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_IMAGE_NUM)

m_read_param=read_param.toInt()
```

TY_INT_SGBM_DISPARITY_NUM

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_NUM, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_NUM)

m_read_param=read_param.toInt()
```

TY_INT_SGBM_DISPARITY_OFFSET

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_OFFSET, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_OFFSET)

m_read_param=read_param.toInt()
```

TY_INT_SGBM_MATCH_WIN_HEIGHT

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_HEIGHT, param)
```

```
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_HEIGHT)
m_read_param=read_param.toInt()
```

TY_INT_SGBM_SEMI_PARAM_P1

```
param =  cl.DevParamFromInt(10000)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1)
m_read_param=read_param.toInt()
```

TY_INT_SGBM_SEMI_PARAM_P2

```
param =  cl.DevParamFromInt(0)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P2, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P2)
m_read_param=read_param.toInt()
```

TY_INT_SGBM_UNIQUE_FACTOR

```
param =  cl.DevParamFromInt(511)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR)
m_read_param=read_param.toInt()
```

TY_INT_SGBM_UNIQUE_ABSDIFF

```
param =  cl.DevParamFromInt(10000)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_ABSDIFF, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_ABSDIFF)
m_read_param=read_param.toInt()
```

TY_BOOL_SGBM_HFILTER_HALF_WIN

```
param =  cl.DevParamFromBool(False)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_HFILTER_HALF_WIN, param)
status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_HFILTER_HALF_WIN)
m_status=status.toBool()
```

TY_INT_SGBM_MATCH_WIN_WIDTH

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM,  TY_INT_SGBM_MATCH_WIN_WIDTH, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_WIDTH)

m_read_param=read_param.toInt()
```

TY_BOOL_SGBM_MEDFILTER

```
param =  cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_MEDFILTER, param)

status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_MEDFILTER)

m_status=status.toBool()
```

TY_BOOL_SGBM_LRC

```
param =  cl.DevParamFromBool(False)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_LRC, param)

status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_LRC)

m_status=status.toBool()
```

TY_INT_SGBM_LRC_DIFF

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM,  TY_INT_SGBM_LRC_DIFF, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_LRC_DIFF)

m_read_param=read_param.toInt()
```

TY_INT_SGBM_MEDFILTER_THRESH

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM,  TY_INT_SGBM_MEDFILTER_THRESH, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MEDFILTER_THRESH)

m_read_param=read_param.toInt()
```

TY_INT_SGBM_SEMI_PARAM_P1_SCALE

```
param =  cl.DevParamFromInt(1)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM,  TY_INT_SGBM_SEMI_PARAM_P1_SCALE, param)
```

```
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM,
TY_INT_SGBM_SEMI_PARAM_P1_SCALE)
m_read_param=read_param.toInt()
```

TY_ENUM_DEPTH_QUALITY

```
para =cl.DevParamFromEnum(TY_DEPTH_QUALITY_BASIC)
cl.DeviceSetParameter(handle,TY_COMPONENT_DEPTH_CAM,TY_ENUM_DEPTH_QUALITY,para)
value = cl.DeviceGetParameter(handle,TY_COMPONENT_DEPTH_CAM,TY_ENUM_DEPTH_QUALITY)
m_value = value.toEnum()
value_list = value.eList()
for i in range (len(value_list)):
   print(' {}'.format(value_list[i].value))
```

TY_INT_TOF_CHANNEL

```
param =  cl.DevParamFromInt(1)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM,  TY_INT_TOF_CHANNEL, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_CHANNEL)
m_read_param=read_param.toInt()
```

TY_INT_TOF_MODULATION_THRESHOLD

```
param =  cl.DevParamFromInt(1)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_MODULATION_THRESHOLD, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM,
TY_INT_TOF_MODULATION_THRESHOLD)
m_read_param=read_param.toInt()
```

TY_INT_TOF_ANTI_SUNLIGHT_INDEX

```
param =  cl.DevParamFromInt(1)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_ANTI_SUNLIGHT_INDEX, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_ANTI_SUNLIGHT_INDEX)
m_read_param=read_param.toInt()
```

TY_INT_MAX_SPECKLE_SIZE

```
param =  cl.DevParamFromInt(200)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_SIZE, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_SIZE)

m_read_param=read_param.toInt()
```

TY_INT_MAX_SPECKLE_DIFF

```
param =  cl.DevParamFromInt(200)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_DIFF ,param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_DIFF)

m_read_param=read_param.toInt()
```

TY_INT_PERSISTENT_IP

```
ip = cl.IPv4StringToInt('0.0.0.0')

param =  cl.DevParamFromInt(ip)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_IP, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_IP)

read_param_m  = read_param.toInt()

print('ip',read_param_m)
```

TY_INT_PERSISTENT_SUBMASK

```
netmask = cl.IPv4StringToInt('0.0.0.0')

param =  cl.DevParamFromInt(netmask)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_SUBMASK, param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_SUBMASK)

read_param_m  = read_param.toInt()

print('netmask',read_param_m)
```

TY_INT_PERSISTENT_GATEWAY

```
gateway =cl.IPv4StringToInt('0.0.0.0')

param =  cl.DevParamFromInt(gateway)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_GATEWAY, param)
```

```
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_INT_PERSISTENT_GATEWAY)
read_param_m  = read_param.toInt()
print('gateway',read_param_m)
```

TY_BYTEARRAY_HDR_PARAMETER

```
para_array = cl.DevParamFromByteArray([2, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 8, 0, 0, 0, 31, 0, 0,
0])
cl.DeviceSetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_BYTEARRAY_HDR_PARAMETER,para_array)
hdr_array =cl.DeviceGetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_BYTEARRAY_HDR_PARAMETER)
hdr_array_1 =hdr_array.toByteArray()
for i in range (len(hdr_array_1)):
   print(',',hdr_array_1[i],end='')
```

TY_STRUCT_AEC_ROI

```
roi = PercipioAecROI(0,0,640,480)
param =cl.DevParamFromPercipioAecROI(roi)
cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_STRUCT_AEC_ROI, param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_STRUCT_AEC_ROI)
m_read_param=read_param.toArray()
print('aec roi',m_read_param)
```

TY_INT_AE_TARGET_Y

```
param =  cl.DevParamFromInt(4000)
cl.DeviceSetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_AE_TARGET_Y ,param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_RGB_CAM, TY_INT_AE_TARGET_Y)
m_read_param=read_param.toInt()
```

TY_BOOL_TOF_ANTI_INTERFERENCE

```
param =  cl.DevParamFromBool(False)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_TOF_ANTI_INTERFERENCE, param)
status = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_BOOL_TOF_ANTI_INTERFERENCE)
m_status=status.toBool()
```

TY_ENUM_CONFIG_MODE

```
para =cl.DevParamFromEnum(TY_CONFIG_MODE_PRESET0)
cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_CONFIG_MODE,para)
value = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_CONFIG_MODE)
m_value = value.toEnum()
value_list = value.eList()
for i in range (len(value_list)):
  print(' {}'.format(value_list[i].value))
```

TY_BOOL_IR_FLASHLIGHT

```
param =  cl.DevParamFromBool(False)
cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_IR_FLASHLIGHT, param)
status = cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_IR_FLASHLIGHT)
m_status=status.toBool()
```

TY_INT_IR_FLASHLIGHT_INTENSITY

```
param =  cl.DevParamFromInt(0)
cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_INT_IR_FLASHLIGHT_INTENSITY ,param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_INT_IR_FLASHLIGHT_INTENSITY)
m_read_param=read_param.toInt()
```

TY_BOOL_RGB_FLASHLIGHT

```
param =  cl.DevParamFromBool(False)
cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_RGB_FLASHLIGHT, param)
status = cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_BOOL_RGB_FLASHLIGHT)
m_status=status.toBool()
```

TY_INT_RGB_FLASHLIGHT_INTENSITY

```
param =  cl.DevParamFromInt(0)
cl.DeviceSetParameter(handle, TY_COMPONENT_LASER, TY_INT_RGB_FLASHLIGHT_INTENSITY ,param)
read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_LASER, TY_INT_RGB_FLASHLIGHT_INTENSITY)
m_read_param=read_param.toInt()
```

TY_INT_SGBM_TEXTURE_THRESH

```
param =  cl.DevParamFromInt(0)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_THRESH,param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_THRESH)

m_read_param=read_param.toInt()

value_min = read_param.mMin()

value_max = read_param.mMax()

value_inc = read_param.mInc()

print('min {} max {} inc {} current {}'.format(value_min,value_max,value_inc,m_read_param))
```

TY_INT_SGBM_TEXTURE_OFFSET

```
param =  cl.DevParamFromInt(500)

cl.DeviceSetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_OFFSET,param)

read_param = cl.DeviceGetParameter(handle, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_OFFSET)

m_read_param=read_param.toInt()

value_min = read_param.mMin()

value_max = read_param.mMax()

value_inc = read_param.mInc()

print('min {} max {} inc {} current {}'.format(value_min,value_max,value_inc,m_read_param))

# 运行上报[error] The feature is not writable. 改属性仅可读
```

TY_FLOAT_EXPOSURE_TIME_US

```
para =cl.DevParamFromFloat(10000)

cl.DeviceSetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_FLOAT_EXPOSURE_TIME_US,para)

read_param = cl.DeviceGetParameter(handle,TY_COMPONENT_IR_CAM_LEFT,TY_FLOAT_EXPOSURE_TIME_US)

m_read_param = read_param.toFloat()

value_min = read_param.fMin()

value_max = read_param.fMax()

value_inc = read_param.fInc()

print('min {} max {} inc {} current {}'.format(value_min,value_max,value_inc,m_read_param))
```

TY_ENUM_TEMPERATURE_ID

```
# 读取有哪些温度传感器

param5 = cl.DevParamFromEnum(TY_TEMPERATURE_LEFT)

error5 = cl.DeviceSetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID, param5)

read_param5 = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID)

m_read_param5 = read_param5.toEnum()

print(f"current value {m_read_param5}")

l_read_param5 = read_param5.eList()

for i in range(len(l_read_param5)):

print(f"{l_read_param5[i].value}")

# 读取温度值

temp_mode = cl.DeviceGetParameter(handle, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID)

if temp_mode.isEmpty():

print('Temperature is not support!')

else :

list = temp_mode.eList()

for idx in range(len(list)):

mode = list[idx]

temp = cl.DeviceControlReadTemperature(handle, cl.Value(mode))

print('{}: {} TEMP: {}'.format(idx, cl.Description(mode), temp))
```

TYSetLogLevel

```
TYSetLogLevel(1)

# 控制台输出的日志信息，设置 Log 等级详情：TY_LOG_LEVEL_VERBOSE = 1,TY_LOG_LEVEL_DEBUG = 2,

# TY_LOG_LEVEL_INFO = 3,TY_LOG_LEVEL_WARNING = 4,

# TY_LOG_LEVEL_ERROR = 5,TY_LOG_LEVEL_NEVER = 9,
```

TYSetLogPrefix

```
TYSetLogPrefix(“Percipio PYTHON SDK”)

# 观察控制台上 SDK 输出的日志前缀是否为 Percipio PYTHON SDK
```

图漾科技（Percipio.XYZ）是全球领先的3D机器视觉供应商，为工业和行业应用提供高性价比的3D工业相机和配套软件方案。公司总部位于上海、在南京、深圳和广州设有研发及销售服务中心。

基于创新并拥有核心专利的3D视觉技术， 图漾不断推出富有竞争力的产品线，满足工业自动化、工业测量、物流科技、商业应用和其他多种场景，产品出货量已经全球领先。

图漾秉持独立视觉产品供应商的商业模式，为各行业的设备和系统集成商客户提供优质产品和服务。 图漾的创新产品方案与合作伙伴的行业专家知识、系统集成能力及市场资源优势相整合，共同帮助最终用户降本增效、创造使用价值，实现3D机器视觉无处不在的愿景。

存在即被感知

联系信息

商务咨询：info@percipio.xyz
技术支持：support@percipio.xyz
公司网站：www.percipio.xyz
在线文档：doc.percipio.xyz/cam/latest/

微信公众号