SDK 属性说明手册

C++

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

1 各组件下的 API 及读写说明

1.1 Device 组件下的 API

1.1.1 相机工作模式设置（TY_TRIGGER_PARAM_EX）

图漾相机包含多种工作模式设置

TY_TRIGGER_MODE_OFF：自由采集模式

TY_TRIGGER_MODE_SLAVE：软触发/硬触发模式

TY_TRIGGER_MODE_M_SIG：相机接收到软触发信号后，自身触发的同时，在 OutPut 引脚输出信号，以触发从设备

上述工作模式设置方法统一如下：

```
TY_TRIGGER_PARAM_EX trigger;
trigger.mode = TY_TRIGGER_MODE_OFF;//根据需要的工作模式进行配置
ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_PARAM_EX, &trigger,
sizeof(trigger)));
```

TY_TRIGGER_MODE_M_PER：相机按照特定的帧率触发，同时在 OutPut 引脚输出信号，以触发从设备。设置方法如下：

```
TY_TRIGGER_PARAM_EX param;
param.mode = TY_TRIGGER_MODE_M_PER;
param.fps = 5;
ASSERT_OK(TYSetStruct(cams[count].hDev, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_PARAM_EX,
(void*)&param, sizeof(param)));
```

注意：

TY_TRIGGER_MODE_M_SIG 和 TY_TRIGGER_MODE_M_PER 固件版本需大于 3.13.68，否则使用 TY_TRIGGER_PARAM 进行设置

TY_TRIGGER_MODE_TIMER_LIST：列表触发模式，为 LR 定制功能。根据设置的触发开始时间（start_time_us）、每两帧的时间间隔数组（offset_us_list[ ]）和触发次数（offset_us_count），相机定时采集（1 + offset_us_count）帧图像并输出图像数据。**启用此工作模式要求相机先启动 PTP 对时**，且 offset_us_count ≤ 50。设置方法如下：

```
TY_TRIGGER_PARAM_EX trigger;
```

```
trigger.mode = TY_TRIGGER_MODE_TIMER_LIST;
ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_PARAM_EX, &trigger,
sizeof(trigger)));
TY_TRIGGER_TIMER_LIST list_timer;
list_timer.start_time_us = (getSystemTime() + 3000) * 1000;
list_timer.offset_us_count = 4;
list_timer.offset_us_list[0] = 1000000;
list_timer.offset_us_list[1] = 1000000;
list_timer.offset_us_list[2] = 1000000;
list_timer.offset_us_list[3] = 1000000;
ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_TIMER_LIST,
&list_timer, sizeof(list_timer)));
```

1.1.2 TY_INT_FRAME_PER_TRIGGER

该功能可以设置相机在接收到一次软触发/硬触发后的出图数量

注意：

相机默认接收一次信号后只出一帧图像，生命周期：open-close。

设置方法

```
int32_t value = 2;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_FRAME_PER_TRIGGER, value));
```

1.1.3 TY_INT_PACKET_DELAY

该功能用于设置相机数据包传输时包与包之间延迟时间，用于网络环境不理想的情况下

设置方法

```
int32_t value = 10000;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_PACKET_DELAY, value));
```

1.1.4 TY_INT_PACKET_SIZE

该功能用于设置相机数据包的大小，用于网络环境不理想的情况下

设置方法

```
int32_t value = 100;
```

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_PACKET_SIZE, value));
```

1.1.5 TY_BOOL_GVSP_RESEND

网络相机图像重传功能

设置方法

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_GVSP_RESEND, true));
```

1.1.6 TY_BOOL_TRIGGER_OUT_IO

该功能用于反转 trigger_out 的输出电平

设置方法

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TRIGGER_OUT_IO, false));
```

该属性可写，但是不可读

1.1.7 TY_BOOL_KEEP_ALIVE_ONOFF

SDK 与相机维持通信状态保持机制，默认为 true。表示通讯保持

设置方法

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_KEEP_ALIVE_ONOFF, false));
```

1.1.8 TY_INT_KEEP_ALIVE_TIMEOUT

SDK 与相机维持通信状态保持时间，usb 默认 15000ms，网络相机默认 3000ms。单位：ms

设置方法

```
int32_t value = 30000;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_KEEP_ALIVE_TIMEOUT, value));
```

1.1.9 TY_INT_TRIGGER_DELAY_US

软/硬触发延迟时间设置。相机在收到硬触发信号后，延迟一段时间后出图。单位：us

设置方法

```
int32_t value = 1300000;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_TRIGGER_DELAY_US, value));
```

1.1.10 TY_INT_TRIGGER_DURATION_US

输出信号的电平保持时间，单位：us

设置方法

```
int32_t value = 100000;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_TRIGGER_DURATION_US, value));
```

1.1.11 TY_ENUM_STREAM_ASYNC

数据流异步功能

TY_STREAM_ASYNC_OFF：数据流同步

TY_STREAM_ASYNC_DEPTH：depth 数据流异步

TY_STREAM_ASYNC_RGB：RGB 数据流异步

TY_STREAM_ASYNC_DEPTH_RGB：depth 和 RGB 数据流异步

TY_STREAM_ASYNC_ALL：所有数据流都异步

设置方法

```
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_STREAM_ASYNC,

TY_STREAM_ASYNC_RGB));
```

1.1.12 TY_INT_CAPTURE_TIME_US

读取深度计算的时间，仅适用于触发模式下，单位 us。

注意：

自由采集模式下，读取的深度计算时间为 0。

使用方法

```
int32_t default_value=0;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_CAPTURE_TIME_US, &default_value));
```

1.1.13 TY_ENUM_TIME_SYNC_TYPE

相机的对时功能

TY_TIME_SYNC_TYPE_NONE：不进行对时。

TY_TIME_SYNC_TYPE_HOST：相机与上位机对时

TY_TIME_SYNC_TYPE_NTP：相机与 NTP 服务器对时

TY_TIME_SYNC_TYPE_PTP：相机与 PTP 服务器对时

TY_TIME_SYNC_TYPE_CAN：相机与 can 网络对时，仅 FM862-GDW 支持，不进行验证

TY_TIME_SYNC_TYPE_PTP_MASTER：设置相机为 PTP 服务器

设置方法

```
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE,
TY_TIME_SYNC_TYPE_HOST));
while (1) {
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
if (sync_ready) {
break;
}
MSLEEP(10);
}
```

对于 NTP 对时，需要额外验证指定服务器 ip 后的对时情况。方法如下：

```
const char* ntp_ip = " 119.29.26.206 ";
int32_t ip_i[4];
uint8_t ip_b[4];
int32_t ip32;
sscanf(ntp_ip, "%d.%d.%d.%d", &ip_i[0], &ip_i[1], &ip_i[2], &ip_i[3]);
ip_b[0] = ip_i[0]; ip_b[1] = ip_i[1]; ip_b[2] = ip_i[2]; ip_b[3] = ip_i[3];
ip32 = TYIPv4ToInt(ip_b);
LOGI("Set persistent IP 0x%x(%d.%d.%d.%d)", ip32, ip_b[0], ip_b[1], ip_b[2], ip_b[3]);
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_NTP_SERVER_IP, ip32));
ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEVICE,TY_INT_NTP_SERVER_IP, &ip32));
TYIntToIPv4(ip32, ip_b);
LOGD("%d %d %d %d", ip_b[0], ip_b[1], ip_b[2], ip_b[3]);
LOGD("Set type of time sync mechanism");
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE,
TY_TIME_SYNC_TYPE_NTP));
LOGD("Wait for time sync ready");
while (1) {
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
if (sync_ready) {
```

```
break;
}
MSLEEP(10);
}
```

1.1.14 TY_BOOL_TIME_SYNC_READY

对时是否成功判断 API

使用方法

```
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
```

1.1.15 TY_BOOL_CMOS_SYNC

左右 ir 异步曝光开关。true：同步曝光；false：异步曝光。

设置方法

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_CMOS_SYNC, false));
```

1.1.16 TY_INT_ACCEPTABLE_PERCENT

网络数据包丢包容忍度，上位机接收到的图像数据包百分比低于此阈值的图像将被丢弃，单位：%。

设置方法

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_ACCEPTABLE_PERCENT, 90));
```

1.1.17 TY_STRUCT_CAM_STATISTICS

获取网络相机的传图情况。

设置方法

```
 TY_CAMERA_STATISTICS st;
 ASSERT_OK( TYGetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_CAM_STATISTICS, &st, sizeof(st)) );
LOGI("Statistics:");
LOGI("  packetReceived: %" PRIu64 " ", st.packetReceived);
LOGI("  packetLost    : %" PRIu64 " ", st.packetLost);
LOGI("  imageOutputed : %" PRIu64 " ", st.imageOutputed);
LOGI("  imageDropped  : %" PRIu64 " ", st.imageDropped);
```

1.1.18 TY_ENUM_TEMPERATURE_ID

与 TY_STRUCT_TEMPERATURE 一起使用，用于读取指定位置处的温度值。当前支持读取的位置有：左 IR 位置处、右 IR 位置处、color 位置处、cpu 位置处、主板位置处。

读取方法

```
uint32_t n = 0;
ASSERT_OK(TYGetEnumEntryCount(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID, &n));
LOGD("===        %14s: entry count %d", "", n);
std::vector<TY_ENUM_ENTRY> feature_info(n);
if (n == 0) {
   LOGD("None temperature sensor exist!\n");
}
else {
   ASSERT_OK(TYGetEnumEntryInfo(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID,
&feature_info[0], n, &n));
   for (int i = 0; i < n; i++) {
      int ret = TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID,
feature_info[i].value);
      if (ret < 0) { LOGD("Set temperature id[%d](%s) failed %d(%s)\n", feature_info[i].value,
feature_info[i].description, ret, TYErrorString(ret));
         break;          }
      TY_TEMP_DATA temp;
      memset(&temp, 0, sizeof(temp));
      ret = TYGetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TEMPERATURE, &temp, sizeof(temp));
      if (ret < 0) { LOGD("Get temperature [%d](%s) failed %d(%s)\n", feature_info[i].value,
feature_info[i].description, ret, TYErrorString(ret));
         break;
      }
      LOGD("Get temperature [%d](%s) temp %s\n", feature_info[i].value, feature_info[i].description,
temp.temp);
   } }
```

1.1.19 IP 设置

ip 设置涉及 3 个 API：

IP 设置 API：TY_INT_PERSISTENT_IP、

子网掩码设置 API：TY_INT_PERSISTENT_SUBMASK、

网关设置 API：TY_INT_PERSISTENT_GATEWAY

1.2 Laser 组件下的 API

1.2.1 TY_BOOL_LASER_AUTO_CTRL

激光自动控制开关

设置方法

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_LASER_AUTO_CTRL, false));
```

1.2.2 TY_INT_LASER_POWER

用于设置激光/条纹光投射器光源强度

设置方法

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_LASER_POWER, value));
```

1.2.3 TY_BOOL_IR_FLASHLIGHT

用于开启 IR 的泛光灯源（FM855-E1-G、VSX3000）

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_IR_FLASHLIGHT, value));
```

1.2.4 TY_BOOL_RGB_FLASHLIGHT

用于开启 RGB 的泛光灯源（FM855-E1-G、VSX3000）

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_RGB_FLASHLIGHT, value));
```

1.2.5 TY_INT_IR_FLASHLIGHT_INTENSITY

用于设置 IR 泛光亮度强度（FM855-E1-G、VSX3000）。

设置方法

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_IR_FLASHLIGHT_INTENSITY, value));
```

1.2.6 TY_INT_RGB_FLASHLIGHT_INTENSITY

用于设置 RGB 泛光亮度强度（FM855-E1-G、VSX3000）

设置方法

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_RGB_FLASHLIGHT_INTENSITY, value));
```

1.3 Depth 组件下的 API

1.3.1 TY_FLOAT_SCALE_UNIT

深度数值单位。

设置方法

```
float value = 0.25;

ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_DEPTH_CAM, TY_FLOAT_SCALE_UNIT,value));
```

1.3.2 TY_INT_SGBM_IMAGE_NUM

SGBM 计算深度时需要的 ir 图像数量

设置方法

```
int32_t value = 10;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_IMAGE_NUM, value));
```

1.3.3 TY_INT_SGBM_DISPARITY_NUM

SGBM 计算深度时的视差搜索范围

设置方法

```
int32_t value = 10;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_NUM, value));
```

1.3.4 TY_INT_SGBM_DISPARITY_OFFSET

SGBM 计算深度时开始搜索的视差值

设置方法

```
int32_t value = 10;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_OFFSET, value));
```

1.3.5 TY_INT_SGBM_MATCH_WIN_HEIGHT

SGBM 计算深度时匹配窗口的高

设置方法

```
int32_t value = 10;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_HEIGHT, value));
```

注意：

TY_INT_SGBM_MATCH_WIN_HEIGHT 和 TY_INT_SGBM_IMAGE_NUM 存在约束关系，以相机 config 文件的约束为准。

1.3.6 TY_INT_SGBM_MATCH_WIN_WIDTH

SGBM 计算深度时匹配窗口的高

设置方法

```
int32_t value = 10;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_WIDTH, value));
```

1.3.7 TY_INT_SGBM_SEMI_PARAM_P1

相邻像素 (+/-1) 约束惩罚参数

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1, value));
```

1.3.8 TY_INT_SGBM_SEMI_PARAM_P1_SCALE

相邻像素 (+/-1) 约束惩罚参数 P1_scale

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1_SCALE, value));
```

1.3.9 TY_INT_SGBM_SEMI_PARAM_P2

周围像素约束惩罚参数 P2

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P2, value));
```

1.3.10 TY_INT_SGBM_UNIQUE_FACTOR

唯一性检查参数 2，即最优与次优匹配点的差值

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR, value));
```

1.3.11 TY_INT_SGBM_UNIQUE_ABSDIFF

唯一性检查参数 1，即最优与次优匹配点的百分比

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR, value));
```

1.3.12 TY_BOOL_SGBM_HFILTER_HALF_WIN

搜索滤波开关。用于进一步优化深度图，去除噪声和不连续性，对物体边缘点云更友好

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_HFILTER_HALF_WIN, value));
```

1.3.13 TY_BOOL_SGBM_MEDFILTER

中值滤波开关，用于消除孤立的噪声点，同时尽可能地保留图像的边缘信息

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_MEDFILTER, value));
```

1.3.14 TY_INT_SGBM_MEDFILTER_THRESH

中值滤波阈值。设定值越大，过滤的噪点越多，但也可能会导致深度图的细节信息丢失

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MEDFILTER_THRESH, value));
```

1.3.15 TY_BOOL_SGBM_LRC

左右一致性检查开关

设置方法

```
bool value = false;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_LRC, value));
```

1.3.16 TY_INT_SGBM_LRC_DIFF

中值滤波阈值。设定值越大，过滤的噪点越多，但也可能会导致深度图的细节信息丢失

设置方法

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_LRC_DIFF, value));
```

1.3.17 TY_ENUM_DEPTH_QUALITY

ToF 相机的深度图像质量

设置方法

```
uint32_t value = 0;

ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEPTH_CAM, feature_id, value));
```

1.3.18 TY_INT_TOF_MODULATION_THRESHOLD

ToF 深度相机接收激光调制光强的阈值，小于此阈值的像素点不参与计算深度，即像素点的深度值赋值为 0

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_MODULATION_THRESHOLD, value));
```

1.3.19 TY_INT_TOF_JITTER_THRESHOLD

ToF 深度相机抖动过滤阈值，阈值设置值越大，深度图边缘抖动的深度数据过滤得越少

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_JITTER_THRESHOLD, value));
```

1.3.20 TY_INT_FILTER_THRESHOLD

ToF 深度相机飞点滤波阈值，默认值为 0，即不加滤波。滤波阈值设置越小，过滤的飞点越多

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_FILTER_THRESHOLD, value));
```

1.3.21 TY_INT_TOF_CHANNEL

ToF 深度相机调制频道。不同调制频道的调制频率不同，互不干扰。如需在同一场景运行多台 ToF 深度相机，首先需确保同系列相机的调制频道不同

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_CHANNEL, value));
```

1.3.22 TY_INT_TOF_HDR_RATIO

高动态范围比阈值，需在 TY_ENUM_DEPTH_QUALITY=HIGH 模式下使用

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_HDR_RATIO, value));
```

1.3.23 TY_INT_TOF_ANTI_SUNLIGHT_INDEX

ToF 抗阳光指数

设置方法

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_ANTI_SUNLIGHT_INDEX, value));
```

1.3.24 TY_INT_MAX_SPECKLE_DIFF

斑点滤波器聚类阈值，单位：mm。若相邻像素的深度差值小于该阈值，则认为该相邻像素属于同一个聚类斑点

设置方法

```
int32_t value = 200;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_DIFF, value));
```

1.3.25 TY_INT_MAX_SPECKLE_SIZE

斑点滤波器面积阈值，单位：像素。面积小于该阈值的聚类斑点会被滤除

设置方法

```
int32_t value = 200;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_SIZE, value));
```

1.3.26 TY_BOOL_TOF_ANTI_INTERFERENCE

抗多机干扰开关，适用于场景中 dToF 相机数量大于 ToF_channel 数量时的场景，可以有效避免多机干扰现象

设置方法

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_TOF_ANTI_INTERFERENCE, value));
```

1.3.27 TY_ENUM_CONFIG_MODE

V 系列相机预设的参数，不同模式下相机的精度不同

设置方法

```
uint32_t value = TY_CONFIG_MODE_PRESET1;
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_CONFIG_MODE, value));
```

1.3.28 TY_INT_SGBM_TEXTURE_THRESH

解决被测物品后无背景或背景较远时，深度图像异常的情况。

设置方法

```
int32_t value = 500;
```

```
TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_THRESH, value);
```

1.3.29 TY_INT_SGBM_TEXTURE_OFFSET

解决被测物体背景或者背景较远深度成像不佳的问题，该属性为只读。

读取方法

```
int32_t value;

TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_OFFSET, &value);
```

1.3.30 TY_INT_DEPTH_MIN_MM

设置深度值最小截止距离，单位：mm。

设置方法

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MIN_MM, 1400));
```

读取方法

```
int min = 0;;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MIN_MM, &min));
```

1.3.31 TY_INT_DEPTH_MAX_MM

设置深度值最大截止距离，单位：mm。

设置方法

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MAX_MM, 1500));
```

读取方法

```
int max = 0;;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_ MAX _MM, &max));
```

1.4 RGB 组件下的 API

1.4.1 TY_INT_ANALOG_GAIN

用于设置 RGB 模拟增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_ANALOG_GAIN, value));
```

1.4.2 TY_INT_R_GAIN

用于设置 RGB 红色通道增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_R_GAIN, value));
```

1.4.3 TY_INT_G_GAIN

用于设置 RGB 绿色通道增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_G_GAIN, value));
```

1.4.4 TY_INT_B_GAIN

用于设置 RGB 蓝色通道增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_B_GAIN, value));
```

1.4.5 TY_INT_EXPOSURE_TIME

用于设置 RGB 曝光时间

设置方法

```
int32_t value = 100;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_EXPOSURE_TIME, value));
```

1.4.6 TY_FLOAT_EXPOSURE_TIME_US

用于设置彩色传感器的真实曝光时间，单位：us。

设置方法

```
float value = 23000.0;

ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_RGB_CAM, TY_FLOAT_EXPOSURE_TIME_US, value));
```

1.4.7 TY_INT_AE_TARGET_Y

AEC 调节的目标明亮度

设置方法

```
int32_t value = 200;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_AE_TARGET_Y, value));
```

1.4.8 TY_STRUCT_AEC_ROI

AEC 调节感兴趣区域，设置后，根据感兴趣区域的亮度，自动调整曝光时间

设置方法

```
TY_AEC_ROI_PARAM roi;

roi.x = 0;

roi.y = 0;

roi.w = 100;

roi.h = 100;

ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_RGB_CAM, TY_STRUCT_AEC_ROI, &roi, sizeof(roi)));
```

1.4.9 TY_BOOL_AUTO_EXPOSURE

RGB 自动曝光开关

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_EXPOSURE, value));
```

1.4.10 TY_BOOL_AUTO_AWB

RGB 自动白平衡功能

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_AWB, value));
```

1.4.11 AUTO_ISP

图漾针对 RGB（135）镜头研发的软件 isp 功能，涉及 TYISPCreate()、ColorIspInitSetting（）、TYISPUpdateDevice（）3 个 API

注意：

启用 auto isp 后，TY_INT_ANALOG_GAIN 会被设置为 1。

1.5 IR 组件下的 API

1.5.1 TY_INT_EXPOSURE_TIME

用于设置左右 IR 曝光时间

设置方法

```
int32_t value = 100;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_EXPOSURE_TIME, value));
```

1.5.2 TY_FLOAT_EXPOSURE_TIME_US

用于设置左右 IR 曝光时间的绝对真值，单位：us。

设置方法

```
float value = 23000.0;

ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_FLOAT_EXPOSURE_TIME_US, value));
```

1.5.3 TY_INT_ANALOG_GAIN

用于设置左右 IR 模拟增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_ANALOG_GAIN, value));
```

1.5.4 TY_INT_GAIN

用于设置左右 IR 增益

设置方法

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_GAIN, value));
```

1.5.5 TY_BOOL_UNDISTORTION

左右 IR 畸变校正开关，开启则表示进行校正，默认为不开启状态

设置方法

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_UNDISTORTION, value));
```

注意：

左右 ir 为绑定状态，要开一起开，要关一起关。以“后设置”的为准。

1.5.6 TY_BOOL_HDR

左右 IR HDR 开关。

设置方法

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_HDR, value));
```

1.5.7 TY_BYTEARRAY_HDR_PARAMETER

HDR 曝光比参数

设置方法

1.读取 HDR 数组的大小

```
uint32_t hdr_size;
ASSERT_OK(TYGetByteArraySize(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
&hdr_size));
printf("hdr_size %d\n", hdr_size);
```

2.读取默认的 HDR 参数

```
uint32_t hdr_param[8];
hdr_param[0] = -1;
hdr_param[1] = -1;
ASSERT_OK(TYGetByteArray(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
(uint8_t*)&hdr_param[0], hdr_size));
printf("default %d  %d\n", hdr_param[0], hdr_param[1]);
```

3.设置 HDR 参数

```
hdr_param[0] = 0;
hdr_param[1] = 0;
ASSERT_OK(TYSetByteArray(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
(uint8_t*)&hdr_param[0], hdr_size));
```

1.5.8 TY_STRUCT_CAM_RECTIFIED_ROTATION

用来获取新的 config 中 IR 的旋转，目前主要是 V 的基于 IR 的标定；

读取方法

```
TY_CAMERA_ROTATION rotation;
```

```
  ASSERT_OK(TYGetStruct(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_STRUCT_CAM_RECTIFIED_ROTATION,
&rotation, sizeof(rotation)));
  LOGD("===%23s%f %f %f", "", rotation.data[0], rotation.data[1], rotation.data[2]);
  LOGD("===%23s%f %f %f", "", rotation.data[3], rotation.data[4], rotation.data[5]);
  LOGD("===%23s%f %f %f", "", rotation.data[6], rotation.data[7], rotation.data[8]);
```

1.6 Storage 组件下的 API

使用方法

1. 导出原始的标定参数文件：

```
.\DumpCalibInfo.exe -id 207000151696 -cs 0 -ds 0 -out_json FM855-E1-G.json
```

-cs 0 ：表示设置 RGB 分辨率为列表中的第一个。

-ds 0：表示设置 depth 分辨率为列表中的第一个。

-out_json：指定输出文件的名字及路径。

2. 导出当前的标定参数文件：

```
.\DumpCalibInfo.exe -id 207000151696 -cs 1 -ds 2  -mode 0 -out_json FM855-E1-G.json
```

-mode 0：表示读取当前设置后的标定参数，读取到的内参会随分辨率变化，其他模式均为原始的标定参数。

1.6.1 TY_BYTEARRAY_CUSTOM_BLOCK

用于获取 custom_block.bin 存储空间大小

使用方法

```
uint32_t block_size;
ASSERT_OK( TYGetByteArraySize(hDevice, TY_COMPONENT_STORAGE, TY_BYTEARRAY_CUSTOM_BLOCK,
&block_size) );
LOGD("custom block bin size %d", block_size);
```

1.6.2 TY_BYTEARRAY_ISP_BLOCK

用于获取 isp_block.bin 存储空间大小

使用方法

```
uint32_t block_size;
ASSERT_OK(TYGetByteArraySize(hDevice, TY_COMPONENT_STORAGE, TY_BYTEARRAY_ISP_BLOCK,
&block_size));
LOGD("isp block bin size %d", block_size);
```

2 其他常用 API

2.1.1 TYHasFeature()

用于判断相机是否具备该功能

使用方法

```
bool has_feature = false;
ASSERT_OK(TYHasFeature(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1,
&has_feature));
```

注意：

没有对应的 Component，TYHasFeature 上报-1008 错误(TY_STATUS_INVALID_COMPONENT)

2.1.2 TYGetFeatureInfo()

用于获取功能的属性，如：isValid、accessMode、writableAtRun、componentID、featureID 等

使用方法

```
TY_FEATURE_INFO feature_info;
int32_t err = (TYGetFeatureInfo(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1,
&feature_info));
LOGD("API return %d ,  isValid %d , accessMode %d  writeableatrun %d ,\n", err,feature_info.isValid,
feature_info.accessMode, feature_info.writableAtRun);
```

2.1.3 TYGetDeviceFeatureNumber()

用于获取组件下 feature 的数量

使用方法

```
uint32_t feature_number = 0;
ASSERT_OK(TYGetDeviceFeatureNumber(hDevice, TY_COMPONENT_DEPTH_CAM, &feature_number));
LOGD("feature_number %d", feature_number);
```

2.1.4 TYGetDeviceFeatureInfo()

和 TYGetDeviceFeatureNumber 搭配使用，可以获取组件下全部的 feature 信息。

使用方法

```
uint32_t feature_number = 0;
ASSERT_OK(TYGetDeviceFeatureNumber(hDevice, TY_COMPONENT_STORAGE, &feature_number));
LOGD("feature_number %d", feature_number);

std::vector <TY_FEATURE_INFO> feature_info(feature_number);
uint32_t entry_count = 0;
ASSERT_OK(TYGetDeviceFeatureInfo(hDevice, TY_COMPONENT_STORAGE, &feature_info[0], feature_number,
&entry_count));
for (uint i = 0; i < feature_number; i++) {
LOGD("feature name [%s], writebleAtRun[%d], TY_FEATURE_ID [%x]", feature_info[i].name,
feature_info[i].writableAtRun, feature_info[i].featureID);
}
```

2.1.5 write_parameters_to_storage()

SDK 3.6.51 及以上版本新增功能，用于将 PV 导出的 json 文件写入相机 custom_block.bin 文件

使用方法

1. 使用 PV（2.5.0 版本及以上）生成一份相机当前参数的 json 文件（只保存用 PV 打开相机后调整过的参数）
2. 运行 SimpleView_SaveLoadConfig -id 207000139089 -s xxxx.json

注意：

该步骤会将 json 文件写入相机

验证方法

相机断电重启后，使用 PV 直接加载参数，观察参数是否发生变化

注意：

SDK3.6.65 及其之后，写入参数时会使用哈夫曼压缩方法，故通过直接查看文件为乱码，需要通过加载的方式验证。

2.1.6 load_parameters_from_storage()

SDK 3.6.51 及以上版本新增功能，用于从 custom_block.bin 文件加载参数，并将其保存至本地文件

使用方法

运行 SimpleView_SaveLoadConfig -id 207000139089 -o xxxxxxx.json

该步骤会从 custom_block.bin 加载参数并将参数输出至程序目录下的 xxxxxxx.json 文件中

2.1.7 clear_storage()

清除相机 custom_block.bin 存储区域的内容

使用方法

```
ASSERT_OK(clear_storage(hDevice));
```

2.1.8 selectDevice()

该 API 可以使用指定的接口，根据 id 或者 ip 打开图漾相机

使用方法

1. 指定使用网络接口，打开图漾相机

```
std::vector<TY_DEVICE_BASE_INFO> selected;
  ASSERT_OK( selectDevice(TY_INTERFACE_ETHERNET, ID, IP, 1, selected) );
```

2. 使用指定 ID 或 IP 打开相机

```
// ID = "207000149647";
IP = "192.168.6.150";
std::vector<TY_DEVICE_BASE_INFO> selected;
ASSERT_OK( selectDevice(TY_INTERFACE_ETHERNET, ID, IP, 1, selected) );
```

2.1.9 TYOpenInterface()

该 API 可以打开指定的网络接口

如网卡 MAC 地址为：88-a4-c2-b1-35-e3（ipconfig /all 查看），IP 地址为：192.168.6.45 (对应 16 进制的小端存储为 2d06a8c0)，则指定网卡时需要写成“eth-88-a4-c2-b1-35-e32d06a8c0”，**注意大小写，需要做区分**。

使用方法

```
char* iface_id = "eth-88-a4-c2-b1-35-e32d06a8c0";
ASSERT_OK(TYOpenInterface(iface_id, &hIface));
```

2.1.10 TYOpenDevice()

该 API 用于打开图漾相机，若相机能够正常打开，则 API 返回结果为 0。其他返回结果均为异常。

2.1.11 parse_firmware_errcode()

用于打印-1024 错误对应的错误码

使用方法

```
TY_FW_ERRORCODE err_code;
int32_t TYOpenDevice_err = ( TYOpenDevice(hIface, selectedDev.id, &hDevice ,&err_code) );
printf("TYOpenDevice_err %d\n", TYOpenDevice_err);
parse_firmware_errcode(err_code);
```

2.1.12 TYGetDeviceXMLSize()

SDK3.6.52 版本上首次加入此 API，用以获取相机 xml 文件的大小

使用方法

```
uint32_t size;
TYGetDeviceXMLSize(hDevice, &size);
LOGD("XML size %d", size);
```

2.1.13 TYGetDeviceXML()

SDK3.6.52 版本首次加入，用以获取相机的 xml 文件

使用方法

```
uint32_t size;
ASSERT_OK(TYGetDeviceXMLSize(hDevice, &size));
LOGD("XML size %d", size);
std::vector<char> xmlBuffer(size);
ASSERT_OK(TYGetDeviceXML(hDevice, xmlBuffer.data(),size,&size));
```

1. API 内容全部打印的方法：

```
std::cout << std::string(xmlBuffer.data(), size) << std::endl;
```

2. 打印指定行的内容：

```
int32_t lineCount = 0;
size_t start = 0;
size_t end = 0;
while (lineCount < 50 && end != std::string::npos) {
end = std::string(xmlBuffer.data(), size).find('\n', start);
```

```
if (end != std::string::npos) {
lineCount++;
if (lineCount == 49) {
std::string line(xmlBuffer.data() + start, end - start);
printf("The firmware version is: %s\n", line.c_str());
break;
}
start = end + 1;
}
}
```

2.1.14 TYImageMode2

在 TY_RESOLUTION_MODE_LIST 中，没有对 1024x768 这个分辨率进行定义，故 TY_IMAGE_MODE_DEPTH16_1024x768 为未定义的标识符。在这种情况下， 则需要通过 TYImageMode2()接口，构造一个参数。

使用方法

```
TY_IMAGE_MODE ImageMode = TYImageMode2(TY_PIXEL_FORMAT_DEPTH16, 1024, 736);
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEPTH_CAM, TY_ENUM_IMAGE_MODE, ImageMode));
```

注意：

用法不限于深度图分辨率，可以扩展到 RGB 和 IR 分辨率上。

3 LOG API 测试

3.1.1 TYSetLogLevel

功能描述：设置日志输出级别

```
typedef enum TY_LOG_LEVEL_LIST{
 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,
}TY_LOG_LEVEL_LIST;
typedef int32_t TY_LOG_LEVEL;
```

本文中定义：VERBOSE 为最高级别，ERROR 为最低级别的日志。

设置方法：

```
ASSERT_OK(TYSetLogLevel(TY_LOG_LEVEL_DEBUG)); // 设置为 DEBUG 级别
```

运行结果：

VERBOSE ⊃ DEBUG ⊃ INFO ⊃ WARNING ⊃ ERROR。

设置 TY_LOG_LEVEL_NEVER 不输出 SDK 日志。

3.1.2 TYSetLogPrefix

功能描述：设置日志前缀，依赖 TYSetLogLevel 属性。

设置方法：

```
ASSERT_OK(TYSetLogPrefix("PERCIPIO SDK"));
```

3.1.3 TYAppendLogToFile

功能描述：将指定等级及其以下的日志输出到指定文件。

设置方法：

```
ASSERT_OK(TYAppendLogToFile("test_log.txt", TY_LOG_LEVEL_DEBUG));
```

3.1.4 TYAppendLogToServer

功能描述：将指定等级及其以下的日志发送至 TCP 服务端。

设置方法：

```
ASSERT_OK(TYAppendLogToServer("tcp", "192.168.2.70", 8000, TY_LOG_LEVEL_DEBUG));
```

3.1.5 TYRemoveLogFile

功能描述：关闭日志文件输出并释放资源。

设置方法：

```
ASSERT_OK(TYRemoveLogFile("test_log.txt"));
```

3.1.6 TYRemoveLogServer

功能描述：断开与日志服务器的连接。

设置方法：

```
ASSERT_OK(TYRemoveLogServer("tcp", "192.168.2.70", 8000));
```

4 获取相机标定参数 API

4.1.1 TY_STRUCT_CAM_INTRINSIC

获取相机内参 API，可用于获取 depth、color、ir_left、right_ir 的内参。

使用方法

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_RGB_CAM;
featureID = TY_STRUCT_CAM_INTRINSIC;
TY_CAMERA_INTRINSIC intri;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &intri, sizeof(TY_CAMERA_INTRINSIC)));
LOGD("===%23s%f %f %f", "", intri.data[0], intri.data[1], intri.data[2]);
LOGD("===%23s%f %f %f", "", intri.data[3], intri.data[4], intri.data[5]);
LOGD("===%23s%f %f %f", "", intri.data[6], intri.data[7], intri.data[8]);
```

4.1.2 TY_STRUCT_EXTRINSIC_TO_DEPTH

获取 ir_right/rgb 到左 ir 的外参。

使用方法

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_IR_CAM_RIGHT;
featureID = TY_STRUCT_EXTRINSIC_TO_DEPTH;
TY_CAMERA_EXTRINSIC extri;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &extri, sizeof(TY_CAMERA_EXTRINSIC)));
LOGD("===%23s%f %f %f %f", "", extri.data[0], extri.data[1], extri.data[2], extri.data[3]);
LOGD("===%23s%f %f %f %f", "", extri.data[4], extri.data[5], extri.data[6], extri.data[7]);
LOGD("===%23s%f %f %f %f", "", extri.data[8], extri.data[9], extri.data[10], extri.data[11]);
LOGD("===%23s%f %f %f %f", "", extri.data[12], extri.data[13], extri.data[14], extri.data[15]);
```

4.1.3 TY_STRUCT_CAM_DISTORTION

畸变参数，可用于 rgb、ir_left、ir_right 的畸变校正

使用方法

```
TY_COMPONENT_ID componentID;

TY_FEATURE_ID featureID;

componentID = TY_COMPONENT_RGB_CAM;

featureID = TY_STRUCT_CAM_DISTORTION;

TY_CAMERA_DISTORTION dist;

ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &dist, sizeof(TY_CAMERA_DISTORTION)));

LOGD("===%23s%f %f %f %f", "", dist.data[0], dist.data[1], dist.data[2], dist.data[3]);

LOGD("===%23s%f %f %f %f", "", dist.data[4], dist.data[5], dist.data[6], dist.data[7]);

LOGD("===%23s%f %f %f %f", "", dist.data[8], dist.data[9], dist.data[10], dist.data[11]);
```

4.1.4 TY_STRUCT_CAM_RECTIFIED_INTRI

获取校正后的 ir_left/ir_right 内参。

使用方法

```
TY_COMPONENT_ID componentID;

TY_FEATURE_ID featureID;

componentID = TY_COMPONENT_IR_CAM_LEFT;

featureID = TY_STRUCT_CAM_RECTIFIED_INTRI;

TY_CAMERA_INTRINSIC intri;

ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &intri, sizeof(TY_CAMERA_INTRINSIC)));

LOGD("===%23s%f %f %f", "", intri.data[0], intri.data[1], intri.data[2]);

LOGD("===%23s%f %f %f", "", intri.data[3], intri.data[4], intri.data[5]);
```

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