

```

1 // 添加对应的P/V函数
2 [DllImport(SLCamDll, ExactSpelling = true, CallingConvention = cc)]
3     public static extern int l_sdk_md_get(int id, int chnn, int idx, int
md_id, out IntPtr p_data);
4
5 [DllImport(SLCamDll, ExactSpelling = true, CallingConvention = cc)]
6     public static extern void l_sdk_md_buf_sub(IntPtr p_data);

```

```

1 // 添加对应的c结构体转换
2 [StructLayout(LayoutKind.Sequential, Pack = 1)]
3 public struct l_frame_t
4 {
5     public byte size; // 本结构体长度 <= 256
6     public byte ver; // 版本: L_MD_F_VER
7     public byte fmt; // 媒体格式: L_MF_H264, L_MF_H265
8     public byte v_type; // 视频帧类型: fmt = H264, H265 时有效
9     public uint len; // 媒体包长度: 不包含本结构体
10
11     [StructLayout(LayoutKind.Explicit, Pack = 1)]
12     public struct MediaInfo
13     {
14         [FieldOffset(0)] public VideoInfo video;
15
16         [FieldOffset(0)] public AudioInfo audio;
17     }
18
19     [StructLayout(LayoutKind.Sequential, Pack = 1)]
20     public struct VideoInfo
21     {
22         public uint w; // 视频宽 (无效,即将弃用)
23         public uint h; // 视频高 (无效,即将弃用)
24     }
25
26     [StructLayout(LayoutKind.Sequential, Pack = 1)]
27     public struct AudioInfo
28     {
29         public ushort track; // 音频, 声道数: 1单声道, 2双声道
30         public ushort bit_wide; // 音频, 采样位宽: 8, 16, 32
31         public uint sample; // 音频, 采样率: 8,000 Hz, 44,100 Hz, 96,000 Hz,
192,000 Hz
32     }
33
34     public MediaInfo mediaInfo; // union 的实现
35
36     public long time; // 时间戳 (微秒)
37     public ushort chnn; // 通道
38     public ushort idx; // 流序号
39     public uint resv2; // 预留2: 默认0
40 }
41
42 [StructLayout(LayoutKind.Sequential, Pack = 1)]
43 public struct l_md_buf_t
44 {
45     public uint tc; // 系统滴答数 [生产者读写,消费者只读]

```

```

46     public uint md_id; // 媒体id [生产者读写,消费者只读]
47
48     public byte ver; // frame头版本 [生产者读写,消费者只读]
49     public byte resv1;
50     public ushort resv2;
51
52     [StructLayout(LayoutKind.Explicit, Pack = 1)]
53     public struct FrameUnion
54     {
55         [FieldOffset(0)] public l_frame_t f_v1; // ver = L_MD_F_VER 时, 有效
[生产者读写,消费者只读]
56     }
57
58     public FrameUnion frameUnion;
59
60     [StructLayout(LayoutKind.Sequential, Pack = 1)]
61     public struct BufferInfo
62     {
63         public IntPtr p_buf; // 缓存指针 [生产者读写,消费者只读]
64         public int buf_len; // 缓存长度 [生产者读写,消费者只读]
65
66         public int start; // 数据起始位置 [生产者读写,消费者只读]
67         public int end; // 数据结束位置 [生产者读写,消费者只读]
68     }
69
70     public BufferInfo buffer;
71
72     [StructLayout(LayoutKind.Sequential, Pack = 1)]
73     public struct RefCountInfo
74     {
75         public IntPtr p_count; // 引用计数器 [消费者不可见]
76     }
77
78     public RefCountInfo refCount;
79 }

```

```

1 // 添加对应函数
2 // 取图idx固定为64
3 public bool SaveImage(string filePath = "./img.jpg")
4 {
5     // 查询图片流可用参数
6     // StreamPic(0, 64);
7     // 设置图片流 注: 图片流中chnn固定为0 idx固定为64
8     // SetStreamPic(0, 64, "1920*1080", "high", 333);
9     RequestImageStream();
10    IntPtr pBuf = IntPtr.Zero;
11    int loginId = m_id;
12    int channel = 0;
13    int idx = 64;
14    int mdId = 0;
15
16    // l_sdk_md_get(loginId, channel, idx, mdId, out pBuf);
17    // 调用 l_sdk_md_get 获取数据
18    int result = l_sdk_md_get(loginId, channel, idx, mdId, out pBuf);
19
20    if (result != 0 || pBuf == IntPtr.Zero)
21    {

```

```

22         Console.WriteLine("Failed to retrieve media data.");
23         return false;
24     }
25
26     try
27     {
28         // 将 IntPtr 转为 l_md_buf_t 结构体
29         l_md_buf_t buf = Marshal.PtrToStructure<l_md_buf_t>(pBuf);
30
31         // 计算数据大小并保存为图片
32         int dataSize = buf.buffer.end - buf.buffer.start;
33
34         if (dataSize > 0)
35         {
36             // 复制缓冲区数据到字节数组
37             byte[] imageData = new byte[dataSize];
38             Marshal.Copy(buf.buffer.p_buf + buf.buffer.start, imageData,
0, dataSize);
39
40             // 保存为 JPG 文件
41             File.WriteAllBytes(filePath, imageData);
42             Console.WriteLine($"Image saved to {filePath}");
43
44             return true;
45         }
46     }
47     catch (Exception ex)
48     {
49         Console.WriteLine($"Error saving image: {ex.Message}");
50     }
51     finally
52     {
53         // 释放缓冲区
54         if (pBuf != IntPtr.Zero)
55         {
56             l_sdk_md_buf_sub(pBuf);
57         }
58
59         // 释放图片流
60         CloseImageStream();
61     }
62
63     return false;
64 }
65
66 public bool SetStreamPic(int chnn, int idx, string wh = "1920*1080",
string quality = "high",
67     int intervalMs = 333)
68 {
69     JObject request = new JObject
70     {
71         ["cmd"] = "set_stream_pic",
72         ["set_stream_pic"] = new JObject
73         {
74             ["chnn"] = chnn,
75             ["idx"] = idx,
76             ["fmt"] = "jpeg",
77             ["wh"] = wh,

```

```

78         ["quality"] = quality,
79         ["interval_ms"] = intervalMs
80     }
81 };
82
83     return SetRequest(request);
84 }
85
86 //图片流 chnn固定为0 idx固定为64
87 public JObject StreamPic(int chnn, int idx)
88 {
89     JObject request = new JObject
90     {
91         ["cmd"] = "stream_pic",
92         ["stream_pic"] = new JObject
93         {
94             ["chnn"] = chnn,
95             ["idx"] = idx
96         }
97     };
98
99     return GetRequest(request);
100 }
101
102 public bool CloseImageStream()
103 {
104     var request = new JObject
105     {
106         ["cmd"] = "close_stream",
107         ["close_stream"] = new JObject
108         {
109             ["chnn"] = m_chnn,
110             ["idx"] = 64
111         }
112     };
113     return SetRequest(request);
114 }
115
116 private bool RequestImageStream()
117 {
118     JObject request = new JObject
119     {
120         ["cmd"] = "open_stream",
121         ["open_stream"] = new JObject
122         {
123             ["chnn"] = m_chnn,
124             ["idx"] = 64
125         }
126     };
127     return SetRequest(request);
128 }

```

```

// _camera.getStreamInfoJson();
// 以C314为例 A系列可从get_stream_ability中的wh中获得受支持的分辨率集合 F801W 可从ge
resolu_cBox.Items.Add("1920*1080");
// resolu_cBox.Items.Add("1280*720");
// resolu_cBox.Items.Add("2592*1944");
// resolu_cBox.Items.Add("3840*2160");
// resolu_cBox.Items.Add("4000*3000");
_camera.SetStreamPic(0, 64, "1920*1080", "high", 333);
// _camera.RequestImageStream(); You, 3分钟前 • Uncommitted changes

```

```

private void saveImg_btn_Click(object sender, EventArgs e)
{
    if (i == 0)
    {
        _camera.RequestImageStream();
        i++;
    }
    // 延时200毫秒
    System.Threading.Thread.Sleep(millisecondsTimeout: 200);
    _camera?.SaveImage(filePath: "C:\\Users\\14228\\Desktop\\backImg\\a123.jpg");
    // _captureImage = true;
}

```

取图时先设置图片流的格式（可通过StreamPic获取）

第一次打开图片流（RequestImageStream）后延时一段时间再获取图片，立即获取会获取失败，且第一次设置RequestImageStream()会有一段较长延时，所以建议在程序运行期间只设置一次，使用完毕后调用CloseImageStream关闭图片流